Université du Québec en Outaouais
Départment d'Informatique et d'Ingénierie

Thèse de doctorat

Sécurité des modèles d'apprentissage automatique

Présenté dans le cadre des exigences du programme de doctorat en sciences et technologies de l'information

Myria Bouhaddi

sous la direction de Prof. Kamel Adi

Décembre 2024

Jury d'évaluation :

Président du Jury	:	Prof. Luigi Logrippo, Ph.D. Département d'informatique et d'ingénierie, Université du Québec en Outaouais.
Examinateur interne	:	Prof. Omer Landry Nguena Timo, Ph.D. Département d'informatique et d'ingénierie, Université du Québec en Outaouais.
Examinateur externe	:	Prof. Nora Boulahia-Cuppens, Ph.D Département de génie informatique et génie logiciel, Polytechnique Montréal.
Directeur de recherche	:	Prof. Kamel Adi, Ph.D. Département d'informatique et d'ingénierie, Université du Québec en Outaouais.

Liste de publications

Les travaux de recherche effectués au cours de cette thèse ont donné lieu à plusieurs publications, certaines déjà parues et d'autres à paraître, dans des revues internationales reconnues ainsi que dans plusieurs actes de congrès.

JOURNAUX

1. M. Bouhaddi et K. Adi, Adversarial Machine Learning Strategies for Mitigating Membership Inference Attacks in MLaaS, Computers & Security Journal. (Soumis)
2. M. Bouhaddi et K. Adi, Advanced Defense Mechanisms for Deep Reinforcement Learning. (à soumettre)

CONFÉRENCES AVEC COMITÉ DE LECTURE

1. M. Bouhaddi et K. Adi, Enhancing Privacy in Machine Learning : A Robust Approach to Preventing Attribute Inference Attacks, 21st International conference on Security and cryptography, Juillet 2024, Dijon, France.
2. M. Bouhaddi et K. Adi, When Rewards Deceive : Counteracting Reward Poisoning in Deep Reinforcement Learning, IEEE International Conference on Cyber Security and Resilience, September 2024, London, UK. (soumis)
3. M. Bouhaddi et K. Adi, Multi-environement training against reward poisoning attacks on deep reinforcement learning, SECRYPT 2023 20th International Conference on Security and Cryptography July 2023, Roma, Italy,
4. M. Bouhaddi et K. Adi, Mitigating Membership Inference Attacks in Machine Learning as a Service, IEEE International Conference on Cyber Security and Resilience, August 2023, Venice, Italy.

COMMUNICATIONS DANS DES SÉMINAIRES ET FORUMS

1. M. Bouhaddi, Identité numérique et cybersécurité, Journée du LRSI, UQO, Mars 2023.
2. M. Bouhaddi, Pour une IA digne de confiance, Masterclass au Forum International de la Cybersécurité (FIC 2022), Montréal, Novembre 2022.

Remerciements

Je tiens tout d'abord à exprimer ma profonde gratitude à mon directeur de thèse, le Professeur Kamel Adi, pour son accompagnement exceptionnel, ses conseils avisés et son soutien infaillible tout au long de mon parcours académique. Le Professeur Adi a été un pilier dans le façonnement de ma thèse, m'offrant non seulement son expertise mais aussi l'espace nécessaire pour explorer, réfléchir, choisir et aborder des problématiques réelles de cybersécurité. Cette liberté de penser, que je considère comme essentielle dans le voyage que représente un parcourt doctoral, m'a permis de me développer tant sur le plan professionnel que personnel. Je lui suis infiniment reconnaissante pour cette opportunité précieuse de croissance et de découverte.

Je souhaite également exprimer ma gratitude sincère aux membres de mon jury, qui ont généreusement consacré du temps, malgré leurs emplois du temps chargés, pour évaluer mon travail. Un merci particulier au Professeur Luigi Logrippo de présider ce jury, ainsi qu'aux Professeurs Nora Boulahia-Cuppens et Omer Landry Nguena, qui ont aimablement accepté la responsabilité d'évaluer ma thèse.

Mes remerciements vont aussi à toute l'équipe du Laboratoire de Recherche en Sécurité Informatique (LRSI), à l'Université du Québec en Outaouais. Travailler au sein de ce groupe a été une expérience enrichissante et stimulante.

Je remercie également tous ceux qui ont contribué, directement ou indirectement, à l'aboutissement de cette thèse. Votre aide a été précieuse à chaque étape.

Résumé

L'intelligence artificielle (IA) et l'apprentissage machine (ML) sont devenus des composantes essentielles dans divers secteurs d'activité tels la santé, la finance ou la cybersécurité et permettant d'apporter des améliorations substantielles dans les processus décisionnels. Cependant, la sécurité de ces technologies représente une préoccupation majeure, car elles sont de plus en plus ciblées par des attaques sophistiquées exploitant leurs vulnérabilités inhérentes. Cette thèse aborde les défis de cybersécurité associés à l'apprentissage machine, en particulier l'apprentissage machine comme service (MLaaS), où les modèles sont accessibles via des interfaces en ligne, exposant les données et les modèles à des risques de sécurité significatifs.

L'objectif principal de cette recherche est de renforcer la sécurité des systèmes d'apprentissage machine face à des menaces spécifiques, telles que les attaques par inférence d'appartenances, les attaques par inférence d'attributs et les attaques par empoisonnement des récompenses. Le but est de développer des méthodes rigoureuses et robustes qui détectent et déjouent ces attaques, tout en préservant la performance et l'utilité des modèles ML.

La thèse propose plusieurs contributions : d'abord, nous avons mis au point une nouvelle méthode pour protéger la confidentialité des données d'entraînement contre les attaques par inférence d'appartenance, en ajoutant un bruit soigneusement calculé aux vecteurs de prédiction. Cette approche diminue efficacement la précision des attaques tout en conservant l'intégrité des sorties des modèles. Ensuite, nous avons proposé des techniques pour prévenir l'inférence d'attributs sensibles à partir des scores de prédiction, comme le masquage des scores et la création d'exemples adverses, renforçant ainsi la résilience des modèles contre ces attaques. Enfin, nous avons abordé les attaques par empoisonnement des récompenses en proposant une stratégie d'entraînement multi-environnements pour l'apprentissage par renforcement, permettant ainsi à l'agent d'apprentissage de distinguer et de neutraliser les récompenses manipulées. De plus, l'intégration d'outils de la théorie des jeux nous a permis d'affiner nos modèles pour qu'ils s'adaptent plus efficacement aux contextes adverses et garantissant, ainsi, que nos défenses sont à la fois dynamiques et robustes.

Nos résultats de recherche contribuent significativement à la sécurité de l'IA, en intégrant de nouvelles solutions qui renforcent la robustesse des systèmes d'apprentissage machine face aux menaces de cybersécurité évolutives. L'efficacité de ces solutions a été validée à travers des tests rigoureux dans des environnements simulés, prouvant non seulement leurs avantages pratiques mais aussi posant les bases pour des applications réelles.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction générale

L'intelligence artificielle (IA) révolutionne de nombreux aspects de notre quotidien et s'impose dans divers domaines allant de la santé à l'industrie en passant par la finance et l'éducation. Elle façonne un nouveau cadre où les décisions peuvent être prises de manière automatisée et intelligente. Le terme "intelligence artificielle" réfère à la simulation de l'intelligence humaine par des machines conçues pour apprendre, comprendre et agir face à différentes situations [175]. Ces capacités permettent aux machines non seulement d'exécuter des tâches spécifiques mais aussi d'améliorer leurs performances au fil du temps grâce à l'expérience acquise.

1.1 Apprentissage automatique

L'apprentissage automatique est la principale discipline informatique qui a le plus nourri les progrès de l'IA ces dernières années. Cette discipline offre un nouveau paradigme de programmation dirigé par les données. Elle permet d'extraire des connaissances à partir de données, permettant ainsi de pouvoir les comprendre et d'identifier des relations entre ces données afin de les généraliser. L'apprentissage automatique a marqué des avancées significatives sur des problèmes qui sont jugés difficiles. En effet, nous avons assisté à des progrès spectaculaires dans le domaine des réseaux neuronaux profonds, en particulier dans leurs applications à la perception (comme la vision artificielle ou la reconnaissance de la parole) et qui ont joué, par exemple, un rôle crucial dans le développement du transport intelligent. Dans les systèmes critiques, plus particulièrement dans le domaine de l'aérospatial, l'apprentissage automatique forge sa place comme un outil de choix. Par exemple, très récemment, le projet DEEL-Québec a permis d'initier un travail R&D sur le développement de structures de base intégrant l'IA dans les systèmes critiques avionique et visant à réduire les risques d'accident dans les avions [139]. Dans le domaine de la santé, les performances de l'IA ont aussi fait leurs preuves dans l'aide au diagnostic, la chirurgie assistée par ordinateur, la médecine prédictive, l'anticipation d'épidémies et le développement de nouveaux médicaments. Par ailleurs, de nombreuses entreprises utilisent l'apprentissage automatique en interne pour améliorer leur marketing, générer de la publicité ciblée ou recommander des produits et services aux utilisateurs. De plus, la capacité de l'IA à analyser rapidement de grandes quantités de données et à identifier des tendances ou des anomalies (threat intelligence) est particulièrement précieuse dans le domaine de la cybersécurité. Cette analyse ne prend que quelques secondes et permet aux analystes de la cybersécurité de répondre jusqu'à 60 fois plus rapidement aux menaces qu'avec une analyse manuelle [4].

Un autre indicateur de l'effervescence autour de l'intelligence artificielle est l'accélération des investissements, qu'il s'agisse des grands acteurs de l'industrie ou des fonds de capital-risque, comme illustré par la Figure 1.1. Google, Apple et Microsoft investissent chacun plusieurs milliards USD depuis plusieurs années [184] pour le développement de solutions novatrices à base d'IA.

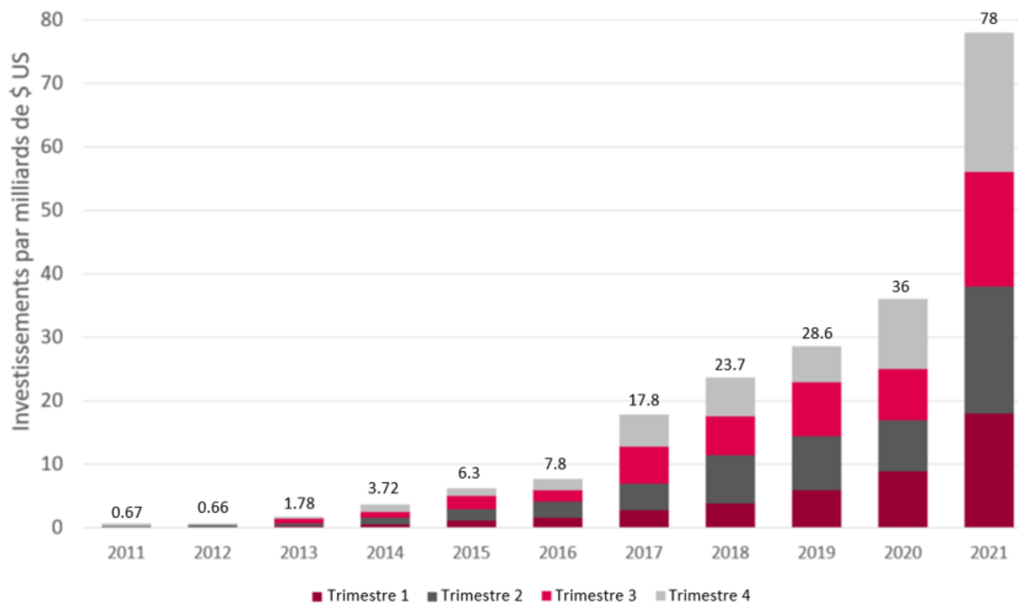

FIGURE 1.1 – Financement des "startups" d'intelligence artificielle (IA) dans le monde de 2011 à 2021 (en milliards de dollars américains), par trimestre.

Bien que l'IA ait réalisé des avancées significatives, les entreprises tendent souvent à négliger leurs impacts en termes de cybersécurité, particulièrement pour les technologies émergentes. En effet, l'adoption croissante de l'IA, et plus spécifiquement de l'apprentissage automatique, soulève de sérieuses préoccupations sécuritaires. Les comportements imprévisibles et l'opacité des modèles d'IA, souvent qualifiés d'effet « boîte noire », compliquent la compréhension des décisions prises par ces systèmes, augmentant le risque d'erreurs et de manipulations malintentionnées.

L'opacité des mécanismes d'apprentissage/prédiction rend les systèmes d'IA particulièrement vulnérables aux attaques. Les utilisateurs ont accès aux données en entrée et en sortie, mais ils manquent souvent de visibilité sur le processus décisionnel interne. Les cyberattaquants peuvent alors exploiter cette lacune, orchestrant des attaques dites « adversariales » pour manipuler les entrées et détourner les décisions du réseau de neurones. Ces manipulations sont conçues pour paraître bénignes aux yeux des opérateurs humains, mais poussent l'IA à agir contre ses directives initiales. Un rapport récent d'Adversa [2], un institut de recherche sur l'IA, confirme cette vulnérabilité. En analysant plus de 2000 études publiées au cours de la dernière décennie, les chercheurs ont constaté que la majorité des algorithmes d'apprentissage automatique présentent des failles en termes de confidentialité et de sécurité. Ce constat alarmant suggère une tendance à la hausse des risques associés à l'IA, soulignant l'importance cruciale de renforcer les mesures de sécurité autour de ces technologies.

1.2 Sécurité de l'apprentissage automatique

L'analyse approfondie des mécanismes de l'apprentissage automatique révèle des risques significatifs pour les individus et les organisations et pouvant engendrer des conséquences lourdes et parfois inacceptables. Les chercheurs ont identifié un large éventail de menaces ciblant les systèmes basés sur l'apprentissage automatique et ont élaboré une classification selon deux critères principaux : le timing de l'attaque (pendant la phase d'apprentissage ou de déploiement) et l'objectif visé par l'attaque. Selon cette typologie, il est possible de distinguer trois principales catégories d'attaques contre les systèmes d'apprentissage automatique : les attaques par manipulation, par infection, et par exfiltration. Chacune de ces catégories d'attaques comprend des stratégies spécifiques qui exploitent les vulnérabilités particulières.

- **Attaques par manipulation**
  Ces attaques visent à altérer le comportement prévu d'un modèle, poussant ainsi le système d'apprentissage automatique à exécuter des actions non anticipées. Ainsi, les attaquants peuvent utiliser des entrées malicieuses pour induire des attaques d'évasion, reprogrammer les systèmes en temps réel, ou même réaliser des attaques de déni de service. Ces actions peuvent avoir un effet durable et significatif, représentant ainsi une menace sérieuse pour la sécurité des systèmes d'apprentissage automatique.
- **Attaques par infection**
  Les attaques par infection compromettent la qualité des décisions prises par le système et permettent aux attaquants de manipuler subtilement les systèmes d'apprentissage. Ces attaques peuvent corrompre les données d'entraînement, activer des déclencheurs cachés dans la structure des modèles, ou propager des modèles d'apprentissage malveillants à travers des techniques d'empoisonnement, de portes dérobées, ou de chevaux de Troie.
- **Attaques par exfiltration**
  Cette forme d'attaque vise à dérober les données d'apprentissage ou les secrets des modèles d'IA. Les attaques par exfiltration menacent directement la confidentialité et la protection des données personnelles, en ciblant les données utilisées pour l'entraînement des modèles, les composants internes des algorithmes, ou les modèles eux-mêmes. Les techniques couramment utilisées incluent l'inférence d'appartenance/d'attributs sensibles, l'inversion de modèles ou l'extraction de modèles, qui compromettent l'intégrité et la confidentialité des systèmes d'apprentissage automatique.

1.3 Confiance accordée aux modèles d'apprentissage automatique

La confiance envers les modèles d'apprentissage automatique est fondamentale pour encourager leur adoption généralisée par les organisations et la société. Ces modèles doivent non seulement remplir efficacement leur fonction mais le faire de manière fiable, éthique et sécuritaire. Ainsi, il est impératif d'aborder la question de la confiance sous divers angles et perspectives.

Selon une étude de Microsoft Research [3], pour qu'un modèle d'apprentissage automatique soit considéré comme digne de confiance, il doit satisfaire à trois critères fondamentaux : la fiabilité, la résilience, et la responsabilité. Ces critères, illustrés dans la Figure 1.2, sont essentiels pour garantir la confiance des utilisateurs dans ces technologies.

FIGURE 1.2 – Qualités essentielles d'un modèle d'apprentissage automatique digne de confiance.

Cette thèse met particulièrement l'accent sur la résilience, notamment sous l'angle de la cybersécurité. Renforcer la cybersécurité est crucial pour bâtir une confiance durable envers les technologies d'IA. Cela implique une protection efficace contre les attaques malveillantes, qu'elles ciblent directement les algorithmes ou qu'elles exploitent des vulnérabilités plus générales. Par l'intégration de mesures de sécurité avancées, telles que le masquage efficace des vecteurs de prédiction ou l'adoption de nouvelles architectures d'entraînement, nous pouvons non seulement protéger les informations sensibles mais également renforcer significativement la confiance des utilisateurs dans ces technologies. L'accent mis sur la cybersécurité permet de protéger les systèmes d'IA contre des menaces externes et de raffiner continuellement les modèles pour mieux résister aux environnements adverses. Cela constitue une composante essentielle de la résilience en IA, soulignant qu'une sécurité accrue contribue directement à une plus grande confiance en ces technologies, essentielle pour leur adoption et leur utilisation dans des contextes critiques.

1.4 Contributions scientifiques

Dans cette thèse, nous abordons les défis associés à la cybersécurité des modèles d'apprentissage automatique, en mettant l'accent sur le développement de stratégies de défense innovantes contre diverses formes d'attaques. À travers quatre contributions, chacune détaillée dans les chapitres subséquents, cette recherche vise à assurer la cybersécurité des modèles d'apprentissage automatique et ainsi renforcer la confiance dans ces technologies.

Notre première contribution est l'élaboration d'une taxonomie détaillée des diverses attaques ciblant les modèles d'apprentissage automatique. Cette taxonomie se distingue par sa capacité à classer les attaques non seulement selon les objectifs de l'attaquant, tels que le vol d'informations, la corruption du modèle ou le sabotage, mais également en fonction du moment où elles se produisent dans le cycle de vie du modèle. Ceci inclut les phases d'entraînement, de test et de déploiement. Une telle structuration nous permet d'identifier et de comparer les vulnérabilités spécifiques à chaque étape, offrant une perspective globale qui contribue à renforcer la sécurité des systèmes basés sur l'intelligence artificielle face aux attaques ciblées.

La seconde contribution propose une approche novatrice pour protéger la confidentialité des données d'entraînement dans les modèles d'apprentissage automatique, offerts en tant que service (MLaaS), contre les attaques par inférence d'appartenance dans un contexte de boîte noire. Notre approche consiste à ajouter un bruit soigneusement conçu au vecteur de prédiction associé à la donnée d'entrée du modèle, ce qui le rend inutilisable pour les attaquants tout en préservant la classe de prédiction indiquée. L'avantage de cette méthode est sa capacité à être appliquée à un classificateur existant sans nécessiter de réentraînement du modèle, évitant ainsi les coûts et les impraticabilités associés. Cette approche est guidée par trois objectifs principaux. Premièrement, elle vise à introduire une incertitude au niveau de l'attaquant concernant l'appartenance ou non d'une donnée à l'ensemble des données d'apprentissage. Deuxièmement, elle cherche à préserver l'utilité et la performance du modèle de classification. Troisièmement, elle limite la perte de confidentialité en cas d'inférence réussie, tout en restant dans un budget spécifié. Pour atteindre ces objectifs, nous les avons formulés comme un problème d'optimisation complexe en raison de l'espace multidimensionnel du bruit. Nous prenons également en considération le comportement dynamique et stratégique de l'adversaire en adoptant un modèle de jeu non-coopératif inspiré des travaux de Bouhaddi *et al.* [22]. Cette démarche intègre récompenses et coûts dans le processus décisionnel pour mieux comprendre la nature de l'interaction entre le défenseur du modèle et l'attaquant de celui-ci. Notre approche représente une perspective novatrice sur le problème des attaques par inférence, étant la première à considérer l'aspect de comportement stratégique. Ce travail a fait l'objet d'une publication dans [23].

Notre troisième contribution explore les défenses contre les attaques par inférence d'attribut sensible, adoptant des stratégies comme le masquage des scores et la création d'exemples adverses pour augmenter la résilience des modèles MLaaS. Cette partie se concentre sur les vulnérabilités des modèles de classification à l'inférence d'attributs, où des données privées sont utilisées pour prédire des attributs sensibles, tirant parti de la précision des scores de prédiction. Les attaquants, avec un accès en boîte noire au modèle MLaaS, exploitent ces scores pour inférer des attributs sensibles. Cette attaque met en lumière le comportement de sortie du modèle nuancé face à diverses entrées. Les méthodes traditionnelles de préservation de la vie privée, comme la confidentialité différentielle, s'avèrent souvent insuffisantes pour contrecarrer les risques liés aux attaques par inférence d'attribut. Elles peuvent altérer l'équilibre entre l'utilité des données et la confidentialité, diminuant la précision du modèle. Face à cette limite, les techniques centrées sur le masquage des scores émergent comme une alternative viable et efficace pour préserver la confidentialité sans sacrifier l'utilité des sorties du modèle. Nous avons développé une solution pratique qui utilise des exemples adverses, traditionnellement considérés comme une technique offensive, comme un outil défensif pour induire en erreur l'attaquant.Par ailleurs, nous avons introduit un algorithme, nommé NOISY, qui emploie la technique JSMA pour générer des exemples adverses qui déstabilisent les efforts d'inférence d'attributs de l'adversaire, tout en préservant la précision du modèle original. Nos expérimentations ont validé l'efficacité de notre méthode, démontrant sa robustesse sur divers jeux de données et renforçant sa faisabilité dans des scénarios du monde réel. Ce travail a fait l'objet d'une publication dans [25].

Notre quatrième contribution explore les défis de sécurité des systèmes d'apprentissage par renforcement profond en ligne (DRL), avec un accent particulier mis sur les attaques par empoisonnement des récompenses. Nous considérons un scénario d'attaque en boîte blanche où l'attaquant possède une connaissance exhaustive de l'environnement d'apprentissage, incluant les algorithmes et les valeurs de récompenses utilisées. Notre approche met en lumière une limitation intrinsèque de la formation du modèle en ligne : sans un point de référence précis, il est difficile pour l'agent d'apprentissage de distinguer entre les récompenses légitimes et celles qui sont malicieusement altérées. Pour surmonter ce défi, nous avons mis en place une architecture d'entraînement multi-environnements. Cette architecture enrichit le processus de formation en permettant à l'agent d'effectuer des comparaisons crédibles et de purifier les signaux de rétroaction dans différents contextes. Cependant, elle introduit également une contrainte critique : bien que l'architecture multi-environnements améliore la robustesse contre les attaques par empoisonnement, elle peut ralentir la vitesse d'apprentissage de l'agent. Dans notre modèle, nous augmentons le réalisme en présumant que l'attaquant peut adapter sa stratégie basée sur l'observation des comportements et des actions de l'agent d'apprentissage à la fin de chaque période. Cette interaction dynamique entre l'agent d'apprentissage et l'attaquant est analysée de manière critique et modélisée comme un jeu bayésien. Dans notre modèle, l'agent doit équilibrer prudemment le risque d'empoisonnement des récompenses avec le besoin d'apprentissage efficace. Simultanément, l'attaquant, agissant de manière rationnelle, ajuste son approche d'empoisonnement pour exploiter les vulnérabilités de l'agent. Cette structure de jeu bayésien offre une perspective approfondie sur la façon dont chaque partie ajuste ses stratégies en réponse aux actions de l'autre, menant à une compréhension plus profonde des états d'équilibre possibles pour l'agent d'apprentissage et l'attaquant. Ce travail a fait l'objet d'une publication dans [24]. Notre travail de recherche a aussi permis la rédaction de plusieurs autres articles qui sont soumis pour publication [26].

1.5 Organisation du rapport

Le document de thèse se compose de la présente introduction générale, cinq chapitres, une conclusion générale et une liste de références bibliographiques pertinentes à notre travail de recherche. Le Chapitre 2 traite des généralités sur la sécurité de l'apprentissage automatique. Ce chapitre commence par une exploration des modèles d'apprentissage automatique, en se concentrant sur leur structure et leurs principes de fonctionnement. Cette analyse vise à mieux comprendre les points de vulnérabilité susceptibles d'être exploités par les attaquants. Par la suite, nous identifions les différentes vulnérabilités auxquelles les modèles d'apprentissage automatique peuvent être exposés et proposons une taxonomie des attaques ciblant ces systèmes. Dans le Chapitre 3, nous examinons l'état de l'art sur la cybersécurité des modèles d'apprentissage automatique. Nous mettons en lumière les techniques utilisées par les attaquants pour perturber ou manipuler les modèles d'apprentissage automatique et les stratégies de défense connues. Les perspectives futures et les défis à surmonter pour sécuriser les systèmes d'apprentissage automatique sont aussi discutés. Le Chapitre 4 introduit une nouvelle approche pour protéger les données d'entraînement dans les modèles de "Machine Learning as a Service" (MLaaS) contre les attaques par inférence d'appartenance. La méthode développée permet de trouver la quantité de bruit optimale à ajouter au vecteur de prédiction pour chaque requête de données, rendant ainsi ces prédictions inutilisables par les attaquants tout en conservant l'exactitude de la classe prédite. Dans le Chapitre 5, nous explorons les stratégies défensives contre les attaques par inférence d'attribut sensible. La méthode proposée utilise des exemples adverses pour brouiller la sortie du modèle de manière à déjouer les tentatives d'inférence tout en maintenant l'intégrité des prédictions légitimes. Le chapitre 6 introduit un modèle innovant visant à réduire l'impact des attaques par empoisonnement dans les systèmes d'apprentissage par renforcement profond (DRL) en ligne. Il présente une architecture multi-environnements pour la formation du modèle qui permet à l'agent d'améliorer sa capacité à détecter les récompenses malicieusement altérées. Cette approche renforce la résilience des systèmes DRL face aux attaques d'empoisonnement des récompenses, en augmentant la robustesse et la fiabilité de ces systèmes dans des environnements adverses. Enfin, le Chapitre 7 sert de conclusion générale qui récapitule les réalisations de la thèse, discute de l'impact des recherches menées et propose des directions pour les travaux futurs.

Chapitre 2

Fondamentaux et sécurité de l'apprentissage automatique

2.1 Introduction

L'apprentissage automatique (ML) est au cœur de nombreuses applications modernes, extrayant des informations précieuses à partir de données provenant de multiples sources. Il a induit une transformation majeure dans la société, en offrant de nouvelles fonctionnalités et en améliorant la qualité de vie des utilisateurs à travers la personnalisation, l'utilisation optimisée des ressources, et l'automatisation de nombreux processus [192]. Cependant, avec l'augmentation de son utilisation, les systèmes d'apprentissage automatique sont devenus des cibles de choix pour les attaquants, qui cherchent à tirer parti des vulnérabilités inhérentes aux algorithmes d'apprentissage [126].

Ce chapitre examine les différentes formes d'apprentissage automatique—apprentissage supervisé, non supervisé, et par renforcement et discute comment chacun de ces paradigmes peut être affecté par des attaques malveillantes. Nous décrivons les vulnérabilités spécifiques à chaque type d'apprentissage et les mécanismes que les attaquants utilisent pour compromettre ces systèmes, que ce soit par l'injection de données malveillantes ou par l'exploitation des faiblesses et angles morts des algorithmes.

Cette exploration détaillée nous permet de poser les bases nécessaires pour comprendre les défis liés à la sécurité dans l'apprentissage automatique et pour introduire les concepts de modèles de menace et de stratégies d'attaque, qui seront explorés plus en détail dans les sections suivantes du chapitre.

2.2 Apprentissage automatique

L'apprentissage automatique est un domaine interdisciplinaire à la croisée de l'informatique, des statistiques et de la science des données, jouant un rôle vital dans la recherche sur l'intelligence artificielle. Les algorithmes d'apprentissage automatique sont utilisés dans la conception et la mise en œuvre d'un système d'acquisition automatique de connaissances en imitant le comportement humain.

La formation d'un modèle d'apprentissage automatique consiste à apprendre à partir d'événements passés à l'aide de données capturées. Disposer de données de haute qualité est donc une condition préalable à toute activité d'apprentissage automatique. Les données sont dispersées dans de nombreux emplacements (tels que des fichiers journaux, des flux d'événements, des bases de données, etc.), souvent dans un environnement très dispersé et géographiquement distribué. L'ingestion de données est le processus de collecte de données à partir de leurs sources d'origine dans un référentiel de données central doté de solides capacités de traitement analytique pour permettre un traitement et une analyse approfondis des données. L'ingestion de données est généralement effectuée à l'aide d'un pipeline de données. Une fois ces données collectées et analysées, ce qui permet une bonne compréhension des données et des problèmes à résoudre, nous sommes prêts à créer nos modèles - souvent en utilisant un processus itératif.

L'approche générale pour créer un modèle d'apprentissage automatique est composée de trois étapes [72] : la phase d'entraînement, la phase de validation (test) et la phase d'inférence, voir la Figure 2.1.

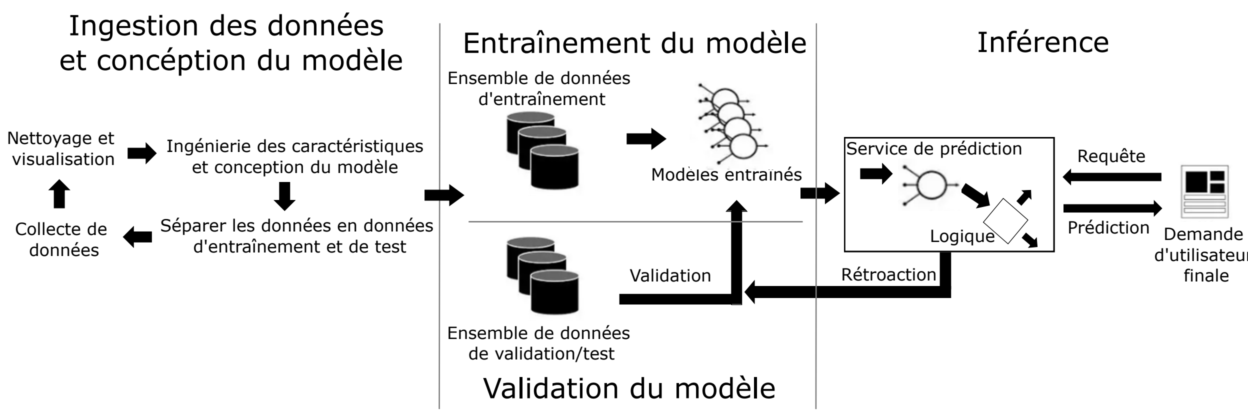

FIGURE 2.1 – Cycle de vie d'un modèle d'apprentissage automatique.

Phase d'entraînement : pendant l'étape d'entraînement, on utilise un jeu de données d'entraînement pour "enseigner" à un modèle d'apprentissage automatique à effectuer une tâche spécifique, comme prédire la probabilité qu'un client achète un produit ou reconnaître des objets dans une image. La plupart des modèles d'apprentissage automatique peuvent être perçus comme des fonctions paramétriques $f(x;\theta)$ prenant comme entrée x et un vecteur de paramètres $\theta \in \Theta$[1]. L'entrée x est souvent représentée comme un vecteur de valeurs appelées *caractéristiques* (feartues). L'espace des fonctions $\mathcal{F} = \{\forall\theta, x \rightarrow f(x;\theta)\}$ est l'ensemble d'hypothèses candidates pour modéliser la distribution à partir de laquelle l'ensemble de données a été échantillonné. On utilise généralement un algorithme d'apprentissage automatique et des techniques de traitement de données pour entraîner le modèle sur les données d'entraînement. Cet algorithme d'apprentissage analyse les données d'apprentissage pour trouver la ou les valeurs du paramètre(s) θ. En d'autres termes, pendant l'entraînement, un algorithme d'apprentissage automatique vise à apprendre les "caractéristiques" de certaines données par rapport à une "tâche" donnée - par exemple, étant donné suffisamment d'images d'ani-

1. Certains modèles (exemple : voisin le plus proche (nearest-neighbor) [6]) ne sont pas paramétriques.

maux de compagnie, l'algorithme doit apprendre à distinguer ce qui différencie les images de, disons, chats de chiens.

Phase de validation (test) : à cette étape, les performances du modèle sont validées sur un jeu de données de test, qui doit être disjoint du jeu de données d'apprentissage afin de mesurer la généralisation du modèle. En d'autres termes, la validation effectue une vérification que l'algorithme a bien appris ce qu'il est censé savoir.

Phase d'inférence : une fois que le modèle a été entraîné et testé avec succès, il peut être déployé en production, c'est-à-dire utilisé pour effectuer la tâche pour laquelle il a été entraîné dans un environnement réel. Le modèle fera des prédictions sur des entrées qu'il n'a jamais vue (ni pendant l'entraînement ni pendant la validation). La valeur du vecteur des paramètres θ est fixe et le modèle calcule $f(x;\theta)$ pour les nouvelles entrées x. La prédiction du modèle peut prendre différentes formes mais, par exemple pour la classification, le plus courant est un vecteur attribuant une probabilité pour chaque classe du problème, qui caractérise la probabilité que l'entrée appartienne à cette classe. Par exemple, un modèle de reconnaissance de la parole peut être déployé dans un système de commande vocale ou bien un modèle de prédiction de la demande peut être utilisé pour aider une entreprise à planifier ses productions.

Sur la base des caractéristiques des données, les modèles d'apprentissage automatique peuvent être classés en trois catégories [149] : modèles d'apprentissage supervisés, non supervisés et les modèles d'apprentissage par renforcement.

2.2.1 Apprentissage supervisé

Durant la phase d'entraînement des méthodes d'apprentissage supervisé, les données d'entrée sont étiquetées avec leur sortie correspondante. Les algorithmes utilisent les caractéristiques des échantillons de données ainsi que leurs étiquettes correspondantes pendant la procédure de formation. À partir des données d'apprentissage, des algorithmes d'apprentissage supervisé déduisent la relation entre les échantillons de données et leurs étiquettes respectives. L'apprentissage supervisé est principalement utilisé dans les applications de classification et pour l'analyse de séquences de données. Les machines à vecteurs de support (SVM) [68], les réseaux de neurones artificiels (NN) [67], la régression linéaire [205] et les arbres de décision [145] sont des algorithmes d'apprentissage supervisé populaires largement utilisés pour les problèmes de classification. De nombreuses tâches pratiques sont basées sur les méthodes d'apprentissage supervisé, par exemple, le traitement du langage naturel [57], la reconnaissance d'objets dans les images [51] et la traduction assistée par ordinateur [159].

Plus formellement, l'apprentissage supervisé peut être défini comme suit :

Définition 2.2.1 ([133]). *Soit $D_{train} = \{x^{(n)}, y^{(n)}\}_{n=1}^{N}$ est l'ensemble de données d'entraînement, où N est le nombre de données d'entraînement, x est le vecteur des caractéristiques (features) et y est l'étiquette. Un modèle d'apprentissage automatique est une fonction $f(x;\theta)$ qui prend en entrée x et fait ressortir $y = f(x;\theta)$, où θ sont les paramètres appris à partir de D_{train}. On utilise un algorithme d'apprentissage $\mathcal{A}$ pour entraîner le modèle. Lorsque les valeurs de y sont discrètes, la tâche d'apprentissage de $f(x;\theta)$ est appelée classification. Lorsque les valeurs de y sont continues, la tâche d'apprentissage s'agira alors de régression. Ce processus d'apprentissage est décrit par la Figure 2.2*

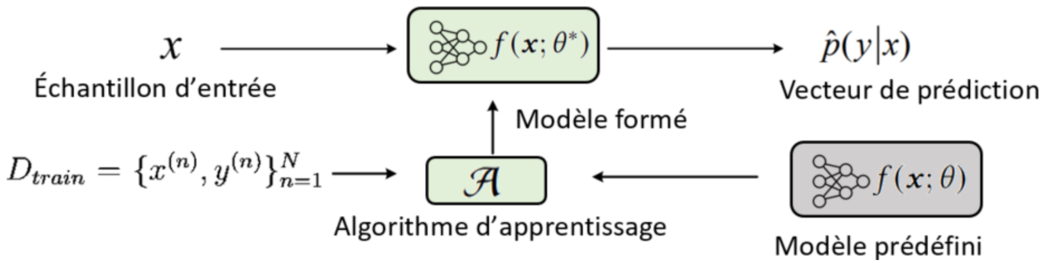

FIGURE 2.2 – Un processus typique d'apprentissage en profondeur pour les modèles de classification.

Un modèle d'apprentissage automatique supervisé est dit bien formé s'il a une petite perte d'espérance pour les données sur lesquelles il fonctionne. Cependant, comme nous ne connaissons pas la véritable distribution des données, nous ne pouvons pas calculer le risque attendu du modèle. Une approche réaliste pour former des modèles d'apprentissage automatique supervisé est la minimisation empirique des risques [189]. L'idée centrale est de mesurer les performances du modèle sur un jeu de données d'apprentissage connu. Pour un jeu de données donné D_{train}, la méthode de la minimisation des risques empiriques essaie de trouver les paramètres θ^* qui minimisent la fonction objectif suivante :

$$\min \mathcal{R}_{D_{train}}(\theta) = \frac{1}{N}\sum_{n=1}^{N}\mathcal{L}(y^{(n)}, f(x^{(n)};\theta)), \tag{2.1}$$

où $\mathcal{L}(.,.)$ est la fonction de perte.

2.2.1.1 Optimisation des paramètres par la méthode de gradient

Une famille de méthodes très utilisée en apprentissage automatique pour résoudre un programme d'optimisation est la descente de gradient. Cette méthode garantit la convergence vers un optimum lorsque la fonction à minimiser est convexe.

Elle nécessite d'autre part la sous-différentiabilité du risque, ce qui impose des contraintes sur la fonction de coût que nous utilisons. Cependant, il y a certains modèles d'apprentissage comme l'apprentissage profond [2] où la fonction objectif que l'on cherche à minimiser est souvent non convexe. La convergence de la descente du gradient vers le minimum global n'est donc pas garantie et la convergence même vers un minimum local peut être extrêmement lente. Une solution à ce problème consiste en l'utilisation de l'algorithme de *descente de gradient stochastique* (stochastic gradient descent : SGD). L'idée de l'approche est de chercher à minimiser une fonction qui peut être écrite sous la forme de la somme de fonctions différentiables en calculant les paramètres optimaux θ^* à chaque fois. Ce processus est alors réalisé de manière itérative sur des lots de données tirés aléatoirement. Chaque fonction objective minimisée de cette manière est une approximation de la fonction objective globale. Partant d'une initialisation aléatoire des paramètres, d'itération en itération, l'algorithme modifie les paramètres de sorte à minimiser le risque. L'algorithme de la descente du gradient stochastique

2. L'apprentissage profond ou apprentissage en profondeur (deep learning) est un ensemble de méthodes d'apprentissage automatique tentant de modéliser avec un haut niveau d'abstraction des données grâce à des architectures articulées de différentes transformations non linéaires.

est comme suit :

$$\theta_{t+1} = \theta_t - \alpha \frac{\partial \mathcal{R}_{\mathcal{D}(\theta)}}{\partial \theta}, \tag{2.2}$$

$$\frac{\partial \mathcal{R}_{\mathcal{D}}(\theta)}{\partial \theta} = \frac{1}{K} \sum_{n=1}^{K} \frac{\partial \mathcal{L}(y^{(n)}, f(x^{(n)}; \theta))}{\partial \theta} \tag{2.3}$$

où K est la taille du mini lot, θ_t sont les paramètres à la i$^{\text{ème}}$ itération et α est le taux d'apprentissage. L'apprentissage est terminé lorsque le modèle converge vers un minimum local où le gradient est proche de zéro.

2.2.2 Apprentissage non supervisé

La principale caractéristique des méthodes d'apprentissage non supervisé est que les entrées d'entraînement ne sont pas étiquetées. Des exemples d'applications d'apprentissage non supervisé sont le pré-entraînement du modèle [50], le clustering [85], la réduction de la dimensionnalité [103], la détection de valeurs aberrantes et l'exploration de règles d'association [44].

Récemment, les modèles génératifs, qui visent à apprendre à générer des échantillons à partir de la distribution de données sous-jacente, attirent de plus en plus l'attention en tant que méthode d'apprentissage non supervisée typique. Il existe deux modèles génératifs typiques, les réseaux antagonistes génératifs (Generative Adversarial Networks : GANs) [60] et les auto-encodeurs variationnels (Variational Auto-Encoders : VAEs) [97]. Les GANs et les VAEs ont été largement utilisés pour générer de l'image, du texte et du son de haute qualité et ont été appliqués à de nombreux domaines, notamment la reconnaissance d'image, la synthèse de parole et la génération de contenu pour les médias.

2.2.2.1 Entraînement des modèles d'apprentissage non supervisé

Un GAN se compose de deux modules de réseau de neurones concurrents, un générateur $\mathcal{G}$ et un discriminateur $\mathcal{D}$, qui sont entraînés pour se faire concurrence. Le générateur prend la variable latente z et génère des échantillons $\mathcal{G}_{\theta_{\mathcal{G}}}(z)$ qui se rapprochent de la distribution des données de D_{train}. Le discriminateur reçoit des échantillons de D_{train} et les échantillons générés, et il est formé pour apprendre la différence entre les deux (voir la Figure 2.3).

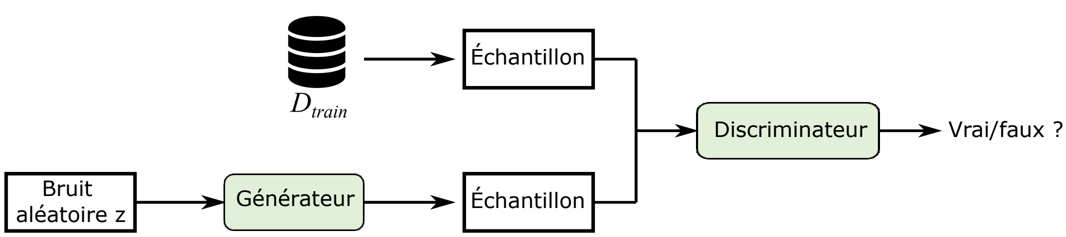

Figure 2.3 – Architecture d'un réseau antagoniste génératif (GAN).

Le discriminateur est essentiellement un classificateur binaire qui détermine si x a été prise de D_{train} ou de $\mathcal{G}$. Après apprentissage, $\mathcal{G}$ peut recevoir différents z et générer des échantillons synthétiques. Comme le générateur et le discriminateur sont tous deux des réseaux de neurones, l'algorithme de la descente de gradient stochastique (SGD) est généralement utilisé pour la formation des GAN. SGD essaie de trouver les paramètres $\theta^*_{\mathcal{G}}$ des GANs suivant la fonction objectif suivante :

$$\min_{\theta_{\mathcal{G}}} \max_{\theta_{\mathcal{D}}} \quad \mathbb{E}_{x \sim P_{data}}[log(\mathcal{D}_{\theta_{\mathcal{D}}}(x))] + \mathbb{E}_{z \sim P_z}[log(1 - \mathcal{D}_{\theta_{\mathcal{D}}}(\mathcal{G}_{\theta_{\mathcal{G}}}(z)))], \tag{2.4}$$

où $\theta_{\mathcal{G}}$ et $\theta_{\mathcal{D}}$ sont les paramètres du générateur et du discriminateur dans un GAN respectivement. P_{data} est la distribution de D_{train} et P_z est la distribution de la variable latente z.

Les VAEs visent à trouver une fonction de distribution a posteriori approchée $q(z|x)$ qui paramètre la distribution latente P_z en fonction des données d'entrée [97]. Un VAE est composé d'un encodeur et d'un décodeur. L'encodeur mappe les données dans un espace latent, tandis que le décodeur mappe la représentation latente codée vers l'espace de données. Au début, la distribution a priori de la variable latente P_z est définie comme une distribution normale unitaire. En conséquence, l'encodeur et le décodeur sont entraînés conjointement de sorte que la sortie du décodeur minimise une erreur de reconstruction entre la postérieure paramétrique et la vraie postérieure mesurée par la divergence de Kullback-Leibler (KL) [3] [101]. Formellement, pour entraîner les VAE, l'algorithme de la descente de gradient stochastique cherche à trouver les paramètres θ^* de suite à la fonction objectif :

$$\min_{\theta_{en}, \theta_{de}} \quad -\mathbb{E}_{q_{\theta_{en}}(z|x)}[p_{\theta_{de}}(x|z)] + KL(q_{\theta_{en}}(z|x)||P_z), \tag{2.5}$$

où $q_{\theta_{en}}(z|x)$ et $p_{\theta_{de}}(x|z)$ sont l'encodeur et le décodeur respectivement et θ_{en} et θ_{de} sont leurs paramètres. $KL(.||.)$ est la divergence de KL et P_z est la distribution de z.

2.2.3 Apprentissage par renforcement

Contrairement à l'apprentissage supervisé et non-supervisé, en apprentissage par renforcement on ne fournit pas de données d'entraînement à l'algorithme. Il acquiert ces données directement au contact de son environnement et il les mémorise. À chaque pas de temps, l'algorithme fait certaines actions qui vont modifier son état.

Ce type d'apprentissage est souvent utilisé dans les domaines où il est difficile de définir explicitement les règles de décision à suivre, comme dans les jeux de stratégie ou les systèmes de contrôle de robotique. Il a également été appliqué avec succès dans d'autres domaines, tels que la recommandation de produits, la gestion de la trésorerie et la gestion de la chaîne d'approvisionnement.

Les méthodes d'apprentissage par renforcement découvrent le comportement optimal de l'agent [4] par un processus d'essais et d'erreurs basé sur les actions, les observations et les récompenses de l'agent [76, 179]. Un agent apprend à effectuer des actions dans un environnement en tentant d'optimiser une récompense ou une récompense future. L'agent prend des actions dans l'environnement, reçoit une rétroaction sous la forme d'une récompense ou d'une

3. En statistique mathématique, la divergence Kullback–Leibler (également appelée entropie relative et I-divergence [39]), notée $D_{\mathrm{KL}}(P||Q)$, est un type de distance statistique : une mesure de la différence entre une distribution de probabilité P et une seconde distribution de probabilité de référence Q.

4. entité capable de percevoir son environnement grâce à des capteurs et d'agir sur celui-ci à travers des effecteurs afin de réaliser des buts.

pénalité, et utilise cette rétroaction pour ajuster sa stratégie d'action en vue de maximiser la récompense à long terme.

2.2.3.1 Environnement, politique et critère d'optimalité

Le cadre mathématique des Processus Décisionnels de Markov (ou MDP) permet de décrire formellement la problématique de l'apprentissage par renforcement. Un Processus Décisionnel de Markov (MDP) est formellement décrit par un tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, R, \gamma, \sigma)$, où :

— $\mathcal{S}$ est un ensemble d'états discret et fini ;
— $\mathcal{A}$ est un ensemble d'actions discret et fini ;
— $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ est une fonction de transition. $\Delta(\mathcal{S})$ est l'ensemble des distributions de probabilité sur $\mathcal{S}$. Nous utiliserons la notation $\mathcal{T}(s'|a, s)$ pour désigner la probabilité que le processus transite vers l'état s' quand on effectue l'action a dans l'état s comme suit :

$$\mathcal{T}(s'|a, s) = P(s_{t+1} = s'|s_t = s, a_t = a).$$

d est la distribution initiale sur l'ensemble des états $\mathcal{S}$.
— $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ est une fonction de récompense [5]. Effectuer l'action a dans l'état s donne une récompense de $R(s, a)$.
— $\gamma \in [0, 1)$ est le facteur d'actualisation.
— σ est la distribution initiale sur les états.

Le lien avec le cadre de l'apprentissage par renforcement se fait très naturellement. Les états sont les états de l'environnement tels qu'ils sont perçus par l'agent et les actions sont les actions effectuées par l'agent. Les fonctions de transition et de récompense modélisent la dynamique de l'environnement et du signal de récompense qui est transmis à l'agent. Ces fonctions ne sont pas connues de l'agent qui doit néanmoins apprendre à agir au mieux, c'est-à-dire à contrôler un MDP de manière à optimiser un certain critère de la fonction de récompense. Il existe plusieurs critères, celui classiquement utilisé est le *critère actualisé* à horizon infini : $G_t = \sum_{t=0}^{\infty} \gamma^t R_t$ où R_t est la récompense reçue à l'instant t et γ un réel de $[0; 1[$. Le facteur γ est un artifice mathématique qui assure la convergence du critère mais qui peut être interprété de plusieurs façons :

— à chaque instant, le processus peut s'arrêter avec une probabilité de $1 - \gamma$;
— γ permet de pondérer l'importance des récompenses à court terme par rapport aux récompenses à long terme. Plus γ est proche de 0, moins les décisions de l'agent seront influencées par le long terme alors que si γ est proche de 1, les récompenses à long terme sont aussi importantes que celles à court terme.

À chaque pas de temps t, l'agent perçoit son état $s_t \in \mathcal{S}$. C'est une variable aléatoire. Il perçoit a priori l'ensemble des actions possibles dans l'état s_t, même si l'on peut supposer, pour simplifier, que l'ensemble des actions est le même dans tous les états. Il choisit une action $a_t \in \mathcal{A}$ et reçoit de l'environnement un nouvel état s_{t+1} et une récompense R_{t+1}. Ainsi, l'agent évolue dans l'environnement et la séquence des états-actions-récompenses s'appelle une trajectoire, et est définie comme suit :

$$s_0, a_0, R_1, s_1, a_1, R_2, s_2, a_2, R_3, \cdots$$

5. On parlera aussi de renforcement ou de gain.

À partir de ses interactions, un algorithme d'apprentissage par renforcement calcule une politique $\pi : \mathcal{S} \rightarrow \mathcal{A}$, c'est-à-dire une fonction qui, à chaque état, préconise une action à exécuter, dont on espère qu'elle maximise les récompenses. La politique peut aussi être probabiliste. Dans ce cas, la politique s'écrit $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$, c'est-à-dire que $\pi(a, s) = P(a_t = a|s_t = s)$ est la probabilité que l'agent choisisse d'exécuter a dans l'état s.

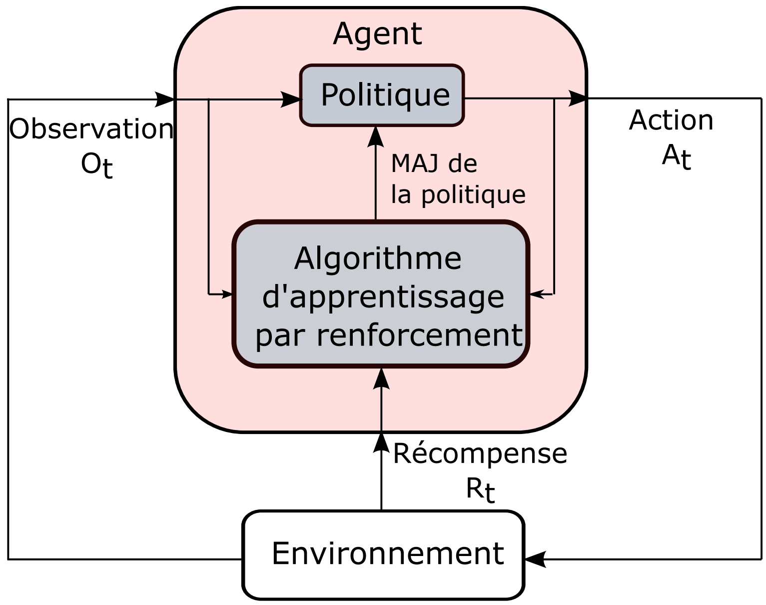

Figure 2.4 – Apprentissage par renforcement.

Dans ce cadre bien défini, un agent cherche à exploiter ces informations ainsi que son rendement actualisé qu'il obtient à chaque action qu'il choisit. Presque tous les algorithmes d'apprentissage par renforcement exécutés par les agents impliquent l'estimation de fonctions de valeur - des fonctions d'états ou des fonctions de paires état-action.

Une fonction de valeur estime à quel point il est bon pour l'agent d'être dans un état donné (ou à quel point il est bon d'effectuer une action donnée dans un état donné) en termes de rendement G. Il est à noter que le retour G d'un agent peut dépendre des actions qu'il va entreprendre. En conséquence, les fonctions de valeur vont dépenser des politiques π.

La première fonction de valeur que nous allons définir est la fonction V. la fonction V , également appelée fonction état-valeur , ou même fonction valeur , ou simplement V , mesure la qualité de chaque état. En d'autres termes, à quel point il est bon ou mauvais d'être dans un état particulier selon le rendement G en suivant une politique π. Autrement dit, nous pouvons définir la fonction V comme une récompense totale attendue (actualisée ou non, selon la valeur de γ) pouvant être obtenue de l'état.

Définition 2.2.2. *On définit la valeur d'un état s pour une politique fixée π par la fonction $V^\pi : \mathcal{S} \rightarrow \mathbb{R}$, définie de la manière suivante :*

$$V^\pi(s) = \mathbb{E}[G_t|s_t = s, \pi] = \mathbb{E}[\sum_{k=0}^{T} \gamma^k R_{t+k+1}|s_t = s, \pi] \tag{2.6}$$

L'équation 6.1 décrit la valeur attendue du rendement total G, à l'instant T en partant de l'état s à l'instant $t = 0$ puis en suivant la politique π. L'espérance $\mathbb{E}[.]$ est utilisée dans cette définition car la fonction de transition de l'environnement peut agir de manière stochastique.

La définition de la fonction V (état-valeur) peut être étendue aux paires état-action, également connue sous le nom de fonction Q. Elle définit la valeur de l'action a dans l'état s sous une politique π, notée Q^π, comme le rendement attendu G à partir de s, en choisissant l'action a, puis en suivant la politique π.

Définition 2.2.3. *On définit la valeur d'une paire état-action (ou sa qualité) (s, a) pour une politique fixée π par une fonction $Q^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$:*

$$Q^\pi(s,a) = \mathbb{E}[G_t | s_t = s, a_t = a, \pi] = \mathbb{E}[\sum_{k=0}^{T} \gamma^k R_{t+k+1} | s_t = s, a_t = a, \pi] \tag{2.7}$$

Dans cette équation, on utilise de nouveau l'espérance $\mathbb{E}[.]$ car la fonction de transition d'environnement peut agir de manière stochastique.

Il y a naturellement une forte relation entre l'équation 6.1 et l'équation 6.2. Notons que la somme des probabilités de toutes les actions sortantes a de l'état s est égale à 1 :

$$\sum_a \pi(a, s) = 1,$$

Par conséquent, nous pouvons affirmer que la fonction état-valeur (fonction V) est équivalente à la somme des fonctions action-valeur (fonction Q) de toutes les actions sortantes de l'état s, multipliée par la probabilité de sélectionner chaque action a (définition de la politique).

$$V^\pi(s) = \sum_{a \in \mathcal{A}} \pi(a, s) Q^\pi(s, a) \tag{2.8}$$

Maintenant que nous avons défini les équations de base pour résoudre un problème d'apprentissage par renforcement, nous devons maintenant nous concentrer sur la recherche d'une solution optimale (politique optimale dans le cas de l'apprentissage par renforcement). Nous allons passer en revue le domaine d'optimalité de l'apprentissage par renforcement et nous aurons également besoin de définir l'équation d'optimalité de Bellman.

Définition 2.2.4. *La politique π peut être considérée comme meilleure ou identique à la politique π' si le rendement attendu de la politique π est supérieur ou égal au rendement attendu de la politique π' pour tous les états. Une telle politique π est appelée* politique optimale*.*

$$\pi \geq \pi' \text{ si et seulement si } V^\pi(s) \geq V^{\pi'}(s), \forall s \in \mathcal{S}$$

La politique optimale doit être associée à une fonction optimale d'état-valeur ou à une fonction d'action-valeur.

Définition 2.2.5. *Fonctions de valeur optimales*
La politique optimale π^* a une fonction état-valeur optimale associée $V^{\pi^*}(s)$ et une fonction action-valeur optimale associée $Q^{\pi^*}(s, a)$. Pour tout $s \in \mathcal{S}$ et toute action $a \in \mathcal{A}$, les fonctions de valeurs optimales sont définies telles que :

$$V^{\pi^*}(s) = \max_\pi V^\pi(s).$$
$$Q^{\pi^*}(s,a) = \max_\pi Q^\pi(s,A).$$

La fonction de valeur optimale associe la plus grande valeur attribuable à un état ou un couple état-action sur l'espace des politiques [143].

• Équation de Bellman

L'équation de Bellman apparaît partout dans la littérature sur l'apprentissage par renforcement, étant l'un des éléments centraux de nombreux algorithmes d'apprentissage par renforcement. En résumé, nous pouvons dire que l'équation de Bellman décompose les fonctions de valeur en deux parties : la récompense immédiate plus les valeurs futures actualisées.

Cette équation simplifie le calcul de la fonction de valeur, de sorte qu'au lieu de faire la somme sur plusieurs pas de temps, nous pouvons trouver la solution optimale d'un problème complexe en le décomposant en sous-problèmes récursifs plus simples et en trouvant leurs solutions optimales.

Par exemple, pour la fonction état-valeur satisfait à des équations récursives. On peut d'abord écrire

$$\begin{aligned} V^\pi(s) = &= \mathbb{E}[\sum_{k=0}^{T} \gamma^k R_{t+k+1} | s_t = s, \pi] && (2.9) \\ &= \mathbb{E}[r_{t+1} + \gamma \sum_{k=0}^{T} \gamma^k R_{t+k+2} | s_t = s, \pi] && (2.10) \end{aligned}$$

Ensuite en développant sur toutes les actions a possibles à partir de s, et sur tous les états suivants s', on a :

$$\begin{aligned} V^\pi(s) = &= \sum_a \pi(s,a) \sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma \mathbb{E}[\sum_{k=0}^{T} \gamma^k R_{t+k+2} | s_{t+1} = s', \pi] && (2.11) \\ &= \sum_a \pi(s,a) \sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma V^\pi(s')] && (2.12) \end{aligned}$$

L'équation (2.12) s'appelle l'équation de Bellman de V^π.

Nous pouvons faire pareil pour la fonction valeur Q^π :

$$\begin{aligned} Q^\pi(s,a) &= \mathbb{E}[\sum_{k=0}^{T} \gamma^k R_{t+k+1} | s_t = s, a_t = a, \pi] && (2.13) \\ &= \mathbb{E}[r_{t+1} + \gamma \sum_{k=0}^{T} \gamma^k R_{t+k+2} | s_t = s, a_t = a, \pi] && (2.14) \\ &= \sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma \sum_{a'} \pi(s',a') Q^\pi(s',a')] && (2.15) \end{aligned}$$

L'équation (2.15) s'appelle l'équation de Bellman de Q^π.

2.2.3.2 Dilemme exploitation Vs exploration

Le dilemme d'exploitation et d'exploration est un concept important en apprentissage par renforcement, qui se réfère à la tension entre la nécessité de maximiser les récompenses à court terme en utilisant les actions qui ont déjà été apprises et la nécessité de découvrir de nouvelles actions qui pourraient mener à des récompenses plus élevées à long terme. Il est souvent difficile de trouver un équilibre entre ces deux aspects. Si un agent choisit d'explorer trop souvent, il peut passer à côté de nombreuses opportunités de récompense à court terme. D'un autre côté, si un agent choisit d'exploiter trop souvent, il peut manquer des opportunités de découvrir de nouvelles actions qui pourraient mener à des récompenses plus élevées à long terme. Pour résoudre ce dilemme, de nombreux algorithmes d'apprentissage par renforcement utilisent des techniques d'exploration telles que l'exploration aléatoire, la découverte de modèles ou l'apprentissage par renforcement par imitation pour encourager l'agent à essayer de nouvelles actions tout en continuant à maximiser les récompenses.

Voici quelques algorithmes couramment utilisés pour résoudre le dilemme d'exploration et d'exploitation en apprentissage par renforcement :

- Équation de Thompson [185] : l'équation de Thompson est une technique de choix de stratégie qui utilise une distribution de probabilité sur les actions possibles pour décider si l'agent doit explorer ou exploiter. L'idée est que plus l'incertitude sur la qualité d'une action est grande, plus l'agent doit être enclin à explorer cette action.
- Algorithme ϵ-greedy [179] : l'algorithme ϵ-greedy est une technique d'exploration simple qui consiste à choisir une action au hasard avec une certaine probabilité ϵ, tandis que pour les autres actions, l'agent choisit l'action qui a donné les meilleures récompenses jusqu'à présent avec une probabilité de 1-ϵ.
- Algorithme de Softmax [179] : l'algorithme de Softmax est similaire à l'algorithme ϵ-greedy, mais il utilise une fonction de Softmax pour déterminer la probabilité de chaque action plutôt que de simplement utiliser une probabilité fixe ϵ. Cela permet de prendre en compte la différence de qualité entre les actions possibles.
- Algorithme UCB (Upper Confidence Bound) [179] : l'algorithme UCB utilise une approche Bayésienne pour prendre en compte l'incertitude sur la qualité des actions possibles et décider si l'agent doit explorer ou exploiter.
- Algorithme de découverte de modèles [179] : l'algorithme de découverte de modèles consiste à construire un modèle de l'environnement à partir des expériences de l'agent et l'utiliser pour décider quelles actions l'agent devrait explorer ou exploiter.

2.2.3.3 Mise à jour de la politique d'apprentissage automatique

Très souvent, l'agent doit décider de la meilleure action à entreprendre en temps réel dépendamment de l'information dont il dispose au moment de sa prise de décision par rapport à l'état de son environnement ou aux conséquences de ses actions. Il peut agir face à un environnement qui est complètement connu ou bien à un environnement inconnu. Un environnement connu est un environnement où les fonctions de transition et la fonction de récompense sont connues. Les principaux algorithmes d'apprentissage par renforcement pour trouver la meilleure action à choisir sont décrits ci-dessous.

1. L'algorithme d'itération de valeur (Value Iteration) [180] : cet algorithme utilise une équation de Bellman pour mettre à jour la valeur de chaque état en fonction de la valeur des actions possibles dans un environnement connu, afin de trouver la politique optimale qui maximise la récompense :

   $$V^{\pi}(s) = \max_{a}(R(s, a) + \gamma V_{\pi}(s'))$$

   Cet algorithme nécessite une modélisation complète de l'environnement et peut être difficile à utiliser lorsque l'environnement est complexe ou lorsque l'on a un grand nombre d'états et d'actions.
2. L'algorithme Q-learning (Q-Learning) [194] : cet algorithme est utilisé dans des situations où l'on ne connaît pas l'environnement et où l'on cherche à apprendre la politique optimale en interagissant avec l'environnement. L'agent apprend la valeur des actions dans chaque état en utilisant une fonction Q qui est mise à jour en fonction de l'expérience de l'agent dans l'environnement :

   $$Q(s, a) = Q(s, a) + \alpha(R(s, a) + \gamma \max_{a'} Q(s', a)) - Q(s, a)$$

   où s' est l'état suivant après avoir exécuté l'action a dans l'état s, et α est le taux d'apprentissage (un paramètre compris entre 0 et 1 qui détermine la vitesse de convergence vers l'optimum). Cet algorithme peut être lent à converger vers la solution optimale et peut être sujet à des oscillations si l'on ne choisit pas les paramètres adéquats.
3. L'algorithme d'apprentissage par policy gradient (Policy Gradient) [181] : cet algorithme utilise un réseau de neurones pour prédire l'action à entreprendre en fonction de l'état courant, et ajuste les poids du réseau en fonction de la performance de l'agent dans l'environnement. La formule utilisée pour mettre à jour les poids du réseau est la suivante :

   $$\Delta w = \alpha * \nabla w * log(\pi(a, s)) * R$$

   où Δw est la modification apportée aux poids du réseau, α est le taux d'apprentissage, $\nabla w * log(\pi(a, s))$ est le gradient de la fonction de coût par rapport aux poids du réseau, $\pi(a, s)$ est la politique prédite par le réseau de neurones pour l'action a dans l'état s, et R est la récompense obtenue en exécutant l'action a dans l'état s. Cet algorithme est souvent utilisé dans des situations où l'espace d'actions est continu et où il est difficile de définir une fonction Q. Cependant, il peut être instable et difficile à mettre en œuvre correctement.
4. L'algorithme d'apprentissage par évolution (Evolutionary Learning) [174] : cet algorithme utilise une méthode d'optimisation par essais et erreurs pour trouver la meilleure politique. Il consiste à générer un grand nombre de politiques aléatoires, à évaluer leur performance et à sélectionner les meilleures politiques pour la génération suivante. La formule utilisée pour mettre à jour la population de politiques est la suivante :

   $$\pi' = s\acute{e}lection(\pi) + croisement(\pi) + mutation(\pi)$$

   où π' est la nouvelle population de politiques, $s\acute{e}lection(\pi)$ est la sélection des meilleures politiques de la population π, $croisement(\pi)$ est le croisement de deux politiques sélectionnées pour générer de nouvelles politiques, et $mutation(\pi)$ est la mutation aléatoire de certaines politiques pour générer de nouvelles politiques.

5. L'algorithme d'apprentissage par renforcement par monte-carlo (Monte-Carlo reinforcement learning) [179] : cet algorithme utilise des échantillons de trajectoires complètes pour évaluer la valeur de chaque état et de chaque action. La formule utilisée pour mettre à jour la valeur de chaque état et action est la suivante :

$$V(s) = V(s) + \alpha * (G - V(s))$$

où $V(s)$ est la valeur de l'état s, G est la récompense finale obtenue dans la trajectoire, et α est le taux d'apprentissage.

6. L'algorithme d'apprentissage par renforcement par TD (Temporal Difference reinforcement learning) [178] : cet algorithme utilise des échantillons de transitions d'état pour estimer la valeur de chaque état et de chaque action. La formule utilisée pour mettre à jour la valeur de chaque état et action est la suivante :

$$V(s) = V(s) + \alpha * (R(s, a) + \gamma * V(s') - V(s))$$

où $V(s)$ est la valeur de l'état s, $R(s, a)$ est la récompense obtenue en exécutant l'action a dans l'état s, γ est le taux d'actualisation, s' est l'état suivant après avoir exécuté l'action a dans l'état s, et α est le taux d'apprentissage.

7. L'algorithme d'apprentissage par renforcement par échantillonnage d'importance (Importance Sampling reinforcement learning) [163] : cet algorithme utilise des échantillons de transitions d'état pour estimer la valeur de chaque état et de chaque action, en utilisant une technique d'échantillonnage d'importance pour attribuer une plus grande importance aux transitions qui ont un impact significatif sur la valeur estimée. La formule utilisée pour mettre à jour la valeur de chaque état et action est la suivante :

$$V(s) = V(s) + \alpha w(R(s, a) + \gamma * V(s') - V(s))$$

où $V(s)$ est la valeur de l'état s, $R(s, a)$ est la récompense obtenue en exécutant l'action a dans l'état s, γ est le taux d'actualisation, s' est l'état suivant après avoir exécuté l'action a dans l'état s.

2.3 Vulnérabilités de l'apprentissage automatique

Malgré les avancées technologiques et les avantages significatifs offerts par les modèles basés sur l'apprentissage automatique, ces systèmes présentent des vulnérabilités qui peuvent être exploitées par des cybercriminels pour mener à bien leurs activités illicites. Les algorithmes d'apprentissage automatique, en particulier, sont susceptibles d'être la cible d'attaques exploitant ces faiblesses pour compromettre les données ou le fonctionnement des modèles [117].

L'usage accru de l'apprentissage automatique dans des domaines critiques tels que la prise de décision automatisée, les véhicules autonomes, la fabrication intelligente, et la santé en ligne, expose les individus et les organisations à des risques nouveaux et parfois imprévisibles. Ces applications peuvent ouvrir la voie à de nouvelles techniques d'attaque et poser des défis considérables en matière de protection des données. L'apprentissage automatique peut ainsi devenir le maillon faible de la chaîne de sécurité, où ses vulnérabilités sont exploitables pour compromettre des infrastructures entières.

Ces vulnérabilités peuvent émaner des différents composants d'un modèle d'apprentissage automatique. Par exemple, lors de la phase d'entraînement, les données sensibles ou privées concernant les activités des utilisateurs, telles que leurs achats, préférences, données de santé et transactions—peuvent être exposées. L'un des objectifs d'un adversaire serait de faire révéler des informations que les concepteurs des modèles veulent garder confidentielles. En effet, des études récentes ont montré que les modèles d'apprentissage automatique sont enclins à mémoriser des informations sensibles des données d'entraînement, les rendant vulnérables à plusieurs attaques liées à la confidentialité et à la protection de la vie privée.

En effet, les échantillons de données utilisés pour l'entraînement de l'IA, le modèle sur lequel elle s'appuie et les éléments internes aux algorithmes utilisés peuvent être exfiltrés par des attaques telles que l'inférence d'appartenance [166], l'inversion de modèle, ou l'extraction de modèle.

En outre, les modèles d'apprentissage automatique, particulièrement lorsqu'ils fonctionnent comme un service sur le cloud, peuvent être interrogés via des API de prédiction. Cette configuration risque de compromettre non seulement la confidentialité des données d'entraînement mais également les paramètres du modèle, qui possèdent une grande valeur commerciale.

En effet, les auteurs dans [207] étudient l'effet du sur-apprentissage et son influence sur la récupération de données ou d'attributs d'entraînement sensibles par un adversaire. Ils montrent que le sur-apprentissage joue un rôle important pour permettre à l'attaquant de mener des attaques par inférence d'appartenance.

De surcroît, un autre aspect à considérer est qu'une grande quantité de données provient d'utilisateurs dont la fiabilité n'est pas garantie ou de tiers sans subir de véritables validations de données. Les attaquants peuvent, par conséquent, injecter des données malveillantes pour empoisonner le processus d'apprentissage ou manipuler les données au moment du test, en exploitant les angles morts et les faiblesses de l'algorithme d'apprentissage pour échapper à la détection.

Des études [182, 209] suggèrent également que la présence de données d'entraînement incomplètes est l'une des raisons pour lesquelles les attaques de type exemples contradictoires gagnent du terrain. D'autre part, les travaux de Goodfellow [61] indiquent que la linéarité dans les modèles de réseaux de neurones profonds dans un espace de grande dimension est une raison plausible pour leur vulnérabilité aux attaques d'exemples contradictoires.

Face à ces vulnérabilités, il est clair que les modèles d'apprentissage automatique peuvent être vulnérables aux attaques informatiques. Il est donc important de définir un modèle de menace d'attaques qui permette de comprendre les différents types d'attaques auxquels un modèle peut être exposé, ainsi que les conséquences potentielles de ces attaques. Cela peut aider à évaluer les risques pour un modèle particulier et à mettre en place des mesures de sécurité appropriées pour le protéger. Il est également important de continuer à surveiller l'évolution des techniques d'attaque et à mettre à jour le modèle de menace en conséquence. De cette façon, les modèles d'apprentissage automatique peuvent être protégés contre les attaques informatiques et continuer à fournir des résultats précis et fiables.

2.4 Modèle de menace

Pour sécuriser efficacement les modèles d'apprentissage automatique, cela nécessite de développer un modèle de menace adapté, tout comme dans la sécurité traditionnelle. Inspirés par les travaux de référence dans le domaine [77, 11, 12, 18, 125], nous caractérisons les attaques basées sur plusieurs critères clés : l'objectif de l'attaquant, sa capacité à manipuler les données et à influencer le système, sa connaissance des algorithmes utilisés, les données impliquées, et sa stratégie globale.

Ces dimensions permettent de délimiter les différents scénarios d'attaque contre les systèmes d'apprentissage, y compris ceux qui utilisent l'apprentissage profond. De plus, elles orientent la conception des stratégies d'attaque comme des problèmes d'optimisation où l'attaquant cherche le meilleur moyen de compromettre le système en maximisant l'efficacité de l'attaque tout en minimisant les risques de détection.

Dans cette section, nous introduisons une taxonomie des modèles de menace applicables aux systèmes d'apprentissage automatique. Nous commençons par définir la surface d'attaque de ces systèmes pour identifier les points où un adversaire pourrait tenter de subvertir le système. Cette analyse de la surface d'attaque est cruciale pour comprendre où et comment les défenses doivent être renforcées pour protéger efficacement les systèmes d'apprentissage automatique contre les intrusions malveillantes.

2.4.1 Surface d'attaque de l'apprentissage automatique

La notion de surface d'attaque dans le contexte de l'apprentissage automatique, introduite par Papernot *et al.* [138], détaille les vulnérabilités potentielles durant le cycle de vie d'un modèle. Cette analyse prend en compte la présence d'adversaires opérant à divers niveaux de l'infrastructure d'apprentissage, de la collecte de données à l'inférence finale.

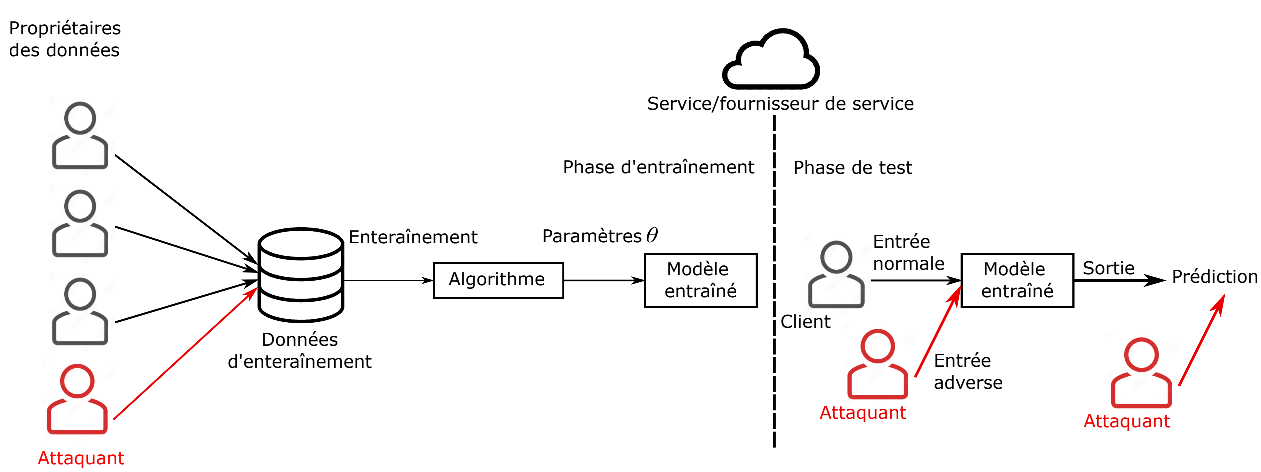

FIGURE 2.5 – Surface d'attaque de l'apprentissage automatique.

Un système typique d'apprentissage automatique implique plusieurs acteurs : les propriétaires des données, qui fournissent les données nécessaires à l'entraînement des modèles ; le fournisseur de service, qui développe et maintient l'algorithme et le modèle ; et les clients, qui utilisent le service, souvent via des API de prédiction. À ces entités s'ajoutent les attaquants, qui peuvent être des adversaires externes cherchant à exploiter le système, ou des acteurs internes curieux cherchant à accéder à des informations confidentielles d'autres utilisateurs.

La figure illustre les différentes phases du cycle de vie d'un modèle d'apprentissage où des attaques peuvent être lancées. Les risques incluent :

1. **Collecte de données :** les attaquants peuvent manipuler ou intercepter les données à cette étape initiale, affectant la qualité et l'intégrité des informations entrant dans le système d'apprentissage.
2. **Entraînement du modèle :** pendant cette phase, les attaquants peuvent introduire des exemples contradictoires pour altérer l'apprentissage, influençant ainsi le comportement du modèle de manière indésirable.
3. **Prédiction :** à ce stade, il est possible d'exfiltrer des informations sur les modèles ou les données utilisées pour l'entraînement par des attaques telles que l'inférence d'appartenance, l'inversion de modèle ou l'extraction de modèle.

Ces vulnérabilités soulignent l'importance de protéger chaque étape du processus, du début à la fin, pour maintenir la sécurité et l'intégrité des systèmes d'apprentissage automatique.

2.5 Taxonomie des modèles de menaces

Pour aborder les défis liés à la sécurité des systèmes d'apprentissage automatique face à des adversaires, il est essentiel de disposer d'une classification structurée des modèles de menaces. Cette section se base sur les travaux pionniers de Barreno *et al.* [11, 12] et Huang *et al.* [77], enrichis par les extensions de Biggio *et al.* [18] et Munoz-Gonzalez *et al.* [125]. Ces études ont posé les fondations pour une taxonomie des menaces dans l'apprentissage automatique, structurée autour de la capacité de l'attaquant, de ses connaissances et de ses objectifs stratégiques.

Pour mieux illustrer cette classification, nous introduisons la Figure 2.6 qui décrit le modèle d'attaque en fonction de ces trois axes.

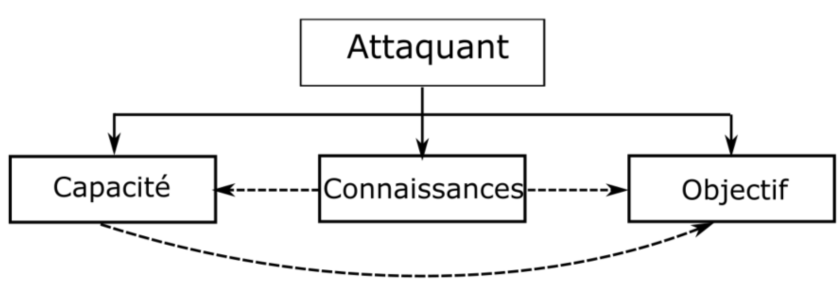

FIGURE 2.6 – Modèle de l'attaquant.

L'efficacité d'une attaque sur un modèle d'apprentissage automatique repose sur ces trois caractéristiques de l'attaquant : sa capacité, sa connaissance du modèle, et son objectif. La capacité détermine à quel stade de l'apprentissage ou de l'exploitation du modèle l'attaque est réalisée. La connaissance influence directement la capacité de l'attaquant à cibler et à formuler efficacement son attaque, en fonction de ce qu'il sait des algorithmes, des données, ou de la configuration du modèle. L'objectif définit l'intention finale de l'attaque, qu'elle vise une perturbation générale ou une exploitation spécifique, et est intrinsèquement lié à la connaissance et à la capacité de l'attaquant. Ensemble, ces éléments façonnent la stratégie d'attaque et déterminent quel type de violation de sécurité peut être tentée.

2.5.1 Capacité de l'attaquant

La capacité des attaquants à compromettre un système d'apprentissage automatique définit comment et dans quelle mesure l'attaquant peut contrôler le processus d'apprentissage [18]. Elle peut être définie en fonction de l'influence que les attaquants ont sur les données utilisées par l'algorithme d'apprentissage et sur la présence de contraintes de manipulation de données. La Figure 2.7 détaille les différentes caractéristiques qui décrivent la capacité de l'attaquant.

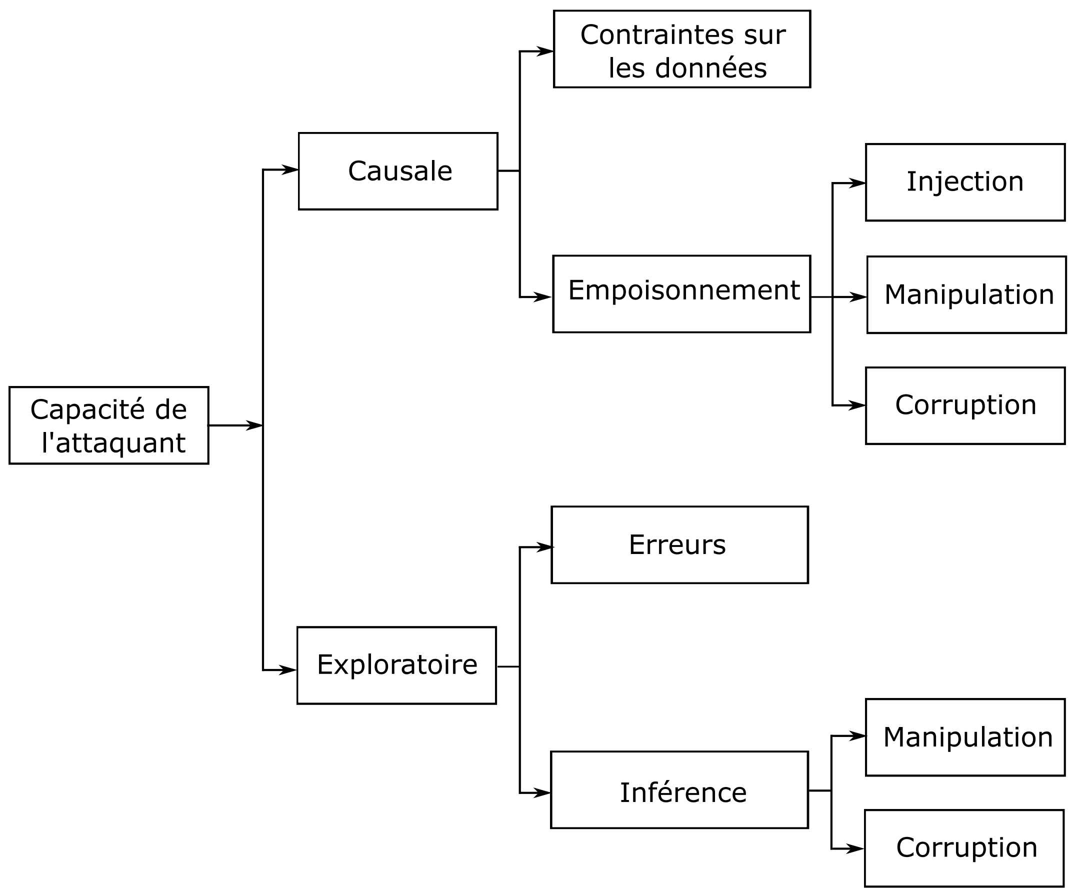

FIGURE 2.7 – Modèle décrivant la capacité de l'attaquant.

2.5.1.1 Influence de l'attaque

On retrouve le terme "influence" pour définir la capacité de l'attaquant [12, 77]. Lors de la mise en place d'un modèle d'apprentissage automatique, deux phases sont vulnérables aux attaques malveillantes : la phase d'entraînement et la phase d'inférence. Une attaque pendant la phase d'apprentissage est dite "causale" (Causative Attack), la caractéristique principale de ce type d'attaque est qu'elle modifie la structure de l'algorithme pour que celui-ci produise des erreurs de classifications lors de la mise en production. Les attaques "exploratoires" (Exploratory Attack) surviennent lors de la phase d'inférence, lorsque le modèle est mis en production. Ce type d'attaque ne modifie pas le modèle mais cherche plutôt à le leurrer ou à le rendre inutilisable.

- **Attaques causales**

Les attaques causales interviennent pendant la phase d'entraînement du modèle, on les appelle aussi les attaques par "empoisonnement". Les attaques par empoisonnement se produisent lorsque les données collectées pour entraîner les algorithmes d'apprentissage ne sont pas fiables, c'est-à-dire que les données sont collectées et étiquetées à partir de capteurs, de personnes ou d'appareils qui peuvent être compromis ou manipulés de manière malveillante. Dans ces applications, étant donné l'énorme quantité de données collectées, la curation manuelle des données pour éliminer les exemples malveillants et les valeurs aberrantes n'est souvent pas réalisable.

Il existe trois catégories d'attaques par empoisonnement [183] :

— Injection de données : cette attaque revient à ajouter des échantillons malveillants dans l'ensemble des données d'entraînement pour dégrader les performances de l'algorithme. Elle permet à l'attaquant d'introduire des backdoor pour pouvoir les utiliser lors de la mise en production et d'éviter la détection.

— Manipulation des données : cette attaque est une extension de l'injection, puisqu'elle permet également à l'attaquant de pouvoir modifier/supprimer les échantillons d'entraînement. Cela peut être une modification des données en entrée (x^i) ou bien des labels (y^i).

— Corruption du modèle (Logic Corruption) : est l'attaque la plus difficile à mettre en place mais la plus "puissante". L'attaquant cible le modèle d'apprentissage lui-même, il modifie ainsi le processus d'apprentissage. Ce type d'attaque peut survenir à l'entraînement initial ou alors pendant les phases de ré-entrainement si le système le permet.

Un autre aspect concernant la capacité de l'attaquant est la présence potentielle de contraintes sur la manipulation des données d'entrée, qui est cependant fortement dépendante du scénario pratique donné [126]. Par exemple, si l'attaquant cherche à échapper à un système de classification de malware, il doit manipuler le code d'exploitation intégré dans l'échantillon de malware sans compromettre sa fonctionnalité intrusive. En cas d'empoisonnement, les étiquettes attribuées aux échantillons d'apprentissage ne sont généralement pas sous le contrôle de l'attaquant. Il devrait donc considérer des contraintes supplémentaires tout en manipulant les échantillons d'empoisonnement pour les faire étiqueter comme souhaité ; par exemple, une quantité maximale de perturbation sur les données d'entrée. Cela peut également être important pour fabriquer des échantillons d'empoisonnement qui sont plus difficiles à détecter avec des techniques de pré-filtrage des données ou de détection des valeurs aberrantes, bien que leur impact puisse également être réduit [140]. Typiquement, ces contraintes peuvent néanmoins être prises en compte dans la définition de la stratégie d'attaque optimale. En particulier, elle peuvent être caractérisées en supposant qu'un ensemble initial d'échantillons d'attaque D_p est donné, et qu'il est modifié selon un espace de modifications possibles $\phi(D_c)$ (eg, contraindre la norme de la perturbation d'entrée sur chaque échantillon d'empoisonnement).

- **Attaques exploratoires**

Dans les attaques exploratoires, l'attaquant n'a pas accès au modèle (il peut tout de même connaître des informations sur ce dernier). Même si l'ensemble de données utilisé pour entraîner l'algorithme d'apprentissage est fiable, l'attaquant peut sonder le système pour découvrir les faiblesses et les angles morts afin de produire des erreurs intentionnelles dans le système d'apprentissage (attaques par évasion). Cependant, les attaques exploratoires peuvent également inclure des scénarios dans lesquels l'objectif de l'attaquant est d'obtenir des informations sur le modèle d'apprentissage automatique utilisé ou les données d'entraînement, ce qui, dans les deux cas, signifie une violation de la vie privée (attaques par inférence).

2.5.2 Connaissances de l'attaquant

Un attaquant peut disposer de plusieurs niveaux de connaissances à propos du système d'apprentissage automatique cible. Les informations dont dispose l'attaquant sont essentielles dans l'élaboration de sa stratégie. Ces informations incluent ce qui suit :

- L'ensemble de données d'entraînement de l'algorithme d'apprentissage, $\mathcal{D}_{train}$.
- L'ensemble de caractéristiques utilisées par l'algorithme d'apprentissage, X.
- L'algorithme d'apprentissage, $\mathcal{M}$.
- La fonction objectif de l'algorithme d'apprentissage visant à optimiser la fonction $\mathcal{L}$.
- Les paramètres de l'algorithme d'apprentissage, θ.

Ainsi, la connaissance de l'attaquant peut être décrite en termes d'espace $\Theta = (\mathcal{D}_{train}, X, \mathcal{M}, \mathcal{L}, \theta)$ qui code les éléments susmentionnés du système d'apprentissage automatique. Cette représentation englobe de nombreux scénarios d'attaque différents, selon les hypothèses faites sur chaque composante de Θ.

L'attaquant peut connaître tout ou une partie de ces cinq paramètres. Cependant, les auteurs dans [12] supposent que l'algorithme $\mathcal{M}$ est connu de tous. En effet, cela est expliqué dans [77], en citant le principe de Kerckhoffs, qui indique que la sécurité d'un système ne doit pas reposer sur des attentes irréalistes de confidentialité. Dans la littérature, on retrouve deux visions pour étudier la connaissance de l'attaquant, décrites par la Figure 2.8 :

— Connaissances limitées (limited knowledge) Vs connaissances parfaites (perfect knowledge)

— Boite noire (black box) Vs boite grise (grey box) Vs boite blanche (white box)

2.5.2.1 Attaques sur la base d'une connaissance parfaite/limitée

Dans le cas d'attaques sur la base d'une connaissance parfaite, l'attaquant est supposé tout savoir sur le système cible. Bien que cela puisse être irréaliste dans la plupart des cas, les attaques sur la base d'une connaissance parfaite permettent d'effectuer des évaluations

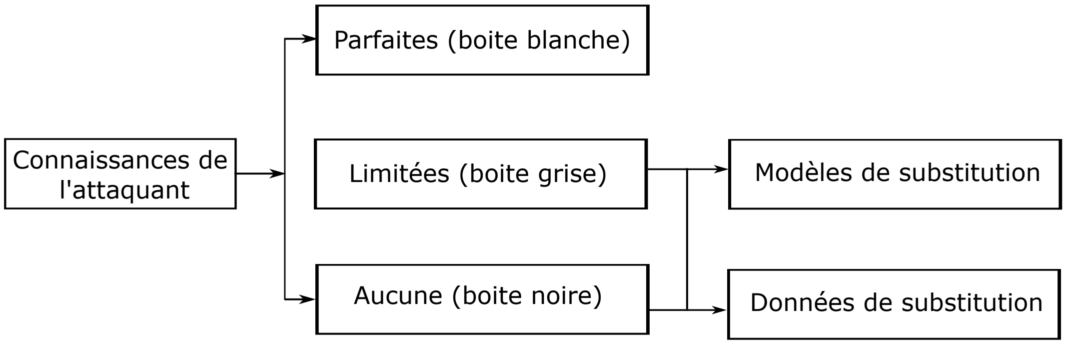

FIGURE 2.8 – Modèle décrivant la connaissance de l'attaquant.

dans le pire des cas de la sécurité des systèmes d'apprentissage automatique. Cela permet d'estimer les limites supérieures de la dégradation des performances d'un système attaqué. Cela peut également être utile pour la sélection de modèles, en comparant les performances de différents algorithmes d'apprentissage à différents paramètres tout en tenant compte de la possibilité d'être attaqué.

Les attaques sur la base d'une connaissance limitée incluent un large éventail de possibilités. La littérature considère généralement deux cas de figure : les attaques avec des données de substitution et les attaques avec des apprenants de substitution.

- Les attaques à connaissance limitée sur la base des données de substitution sont des attaques dans lesquelles l'attaquant connaît la représentation des caractéristiques X, l'algorithme d'apprentissage $\mathcal{M}$ et la fonction objectif optimisée par l'algorithme d'apprentissage $\mathcal{L}$. La confidentialité des données étant aujourd'hui un enjeu majeur, $\mathcal{D}_{train}$ est très souvent inconnu. Par conséquent, ces attaques supposent que l'attaquant n'a pas accès aux données d'apprentissage $\mathcal{D}_{train}$, mais qu'il a accès à un ensemble de données de substitution, $\hat{\mathcal{D}}_{train}$, avec des caractéristiques et une distribution de données similaires (statistiquement proche) à celles de $\mathcal{D}_{train}$. Ensuite, l'attaquant estime les paramètres de l'algorithme d'apprentissage $\hat{\theta}$ en optimisant la fonction objectif $\mathcal{L}$ sur l'ensemble de données de substitution $\mathcal{D}_{train}$.
- Dans les attaques à connaissance limitée sur la base des modèles de substitution, l'attaquant est supposé connaître les données d'apprentissage $\mathcal{D}_{train}$ et l'ensemble de caractéristiques X (par exemple, si l'algorithme d'apprentissage est formé sur des données disponibles publiquement), mais pas l'algorithme d'apprentissage et la fonction objectif optimisée $\mathcal{L}$. Dans ce cas, le vecteur de paramètres estimé $\hat{\theta}$ peut également appartenir à un espace vectoriel différent de celui du système cible, car le modèle de l'attaquant peut être différent. Ces attaques incluent également des cas dans lesquels, que l'attaquant connaisse ou non l'algorithme d'apprentissage du système ciblé, il n'est pas possible pour l'attaquant de dériver une stratégie d'attaque optimale. Cela se produit si le problème d'optimisation correspondant est insoluble ou difficile à résoudre, auquel cas un modèle de substitution peut aider à surmonter cette difficulté en effectuant une attaque traitable mais (éventuellement) moins efficace.

2.5.2.2 Attaques boite noire, boite grise et boite blanche

Dans cette seconde vision sur la connaissance de l'attaquant, on accepte d'étudier le cas où l'adversaire n'a aucune information sur le modèle (boite noire), ce qui est rarement le cas dans une situation réelle. Des connaissances boite grise signifie que l'attaquant a des connaissances limitées, et boite blanche, signifie que l'attaquant dispose d'une connaissance parfaite. Implémenter une défense contre les attaques boite blanche renforce considérablement la résistance du modèle puisque celles-ci sont d'ordinaire, optimisées.

2.5.3 Objectif de l'attaquant

L'objectif de l'attaque peut être décrit en termes de violation de sécurité souhaitée et de spécificité d'attaque. Dans certaines tâches, comme dans les scénarios de classification multi-classes, l'objectif de l'attaquant peut également être décrit en termes de spécificité d'erreur, c'est-à-dire le type d'erreurs que l'attaquant vise à produire dans le système. En conséquence, l'objectif de l'attaquant est étendu en introduisant le concept de spécificité d'erreur. Ces trois caractéristiques sont détaillées ci-dessous et décrits par la Figure 2.9.

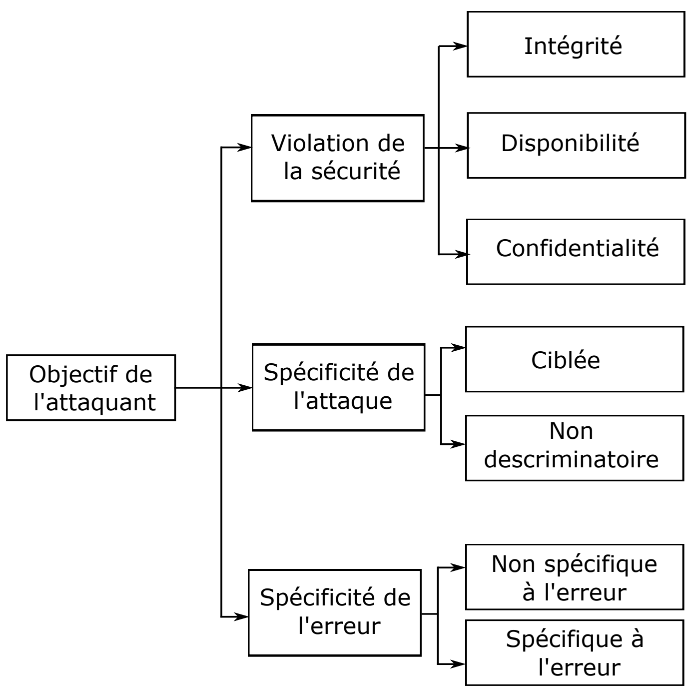

Figure 2.9 – Modèle décrivant l'objectif de l'attaquant.

2.5.3.1 Violation de la sécurité

Trois violations de sécurité différentes peuvent être distinguées par rapport aux systèmes d'apprentissage automatique :

- Violation de l'intégrité : l'attaque échappe à la détection sans compromettre le fonctionnement normal du système. Par exemple, dans une application de filtre anti-spam, cela peut être réalisé en générant des e-mails de spam mal classés comme e-mails sollicités (produisant de faux négatifs).

- Violation de la disponibilité : l'attaquant vise à compromettre la fonctionnalité du système, par exemple en augmentant le taux d'erreur global dans un algorithme de classification.
- Violation de la confidentialité : l'attaquant obtient des informations privées sur le système d'apprentissage automatique, les données utilisées pour l'entraînement ou les utilisateurs du système.

2.5.3.2 Spécificité de l'attaque

La spécificité de l'attaque définit le degré de spécificité de l'intention de l'attaquant, représentant un spectre continu de possibilités allant des scénarios ciblés aux scénarios non discriminatoires.

- Attaque ciblée : l'attaquant vise à produire des erreurs ou à dégrader les performances du système pour un ensemble réduit d'échantillons.
- Attaque non discriminatoire (aveugle) : l'attaquant vise à dégrader les performances du système ou à produire des erreurs de manière non discriminante (pour un large éventail d'échantillons).

2.5.3.3 Spécificité de l'erreur

La spécificité des erreurs permet de lever l'ambiguïté dans les cas où la nature des erreurs peut être différente, comme dans les tâches de classification multi-classes [125]. Par conséquent, du point de vue de la spécificité de l'erreur, une attaque peut être l'une des suivantes :

- Attaque non spécifique à l'erreur (générique) : l'attaquant vise à produire n'importe quel type d'erreur dans le système. Par exemple, dans une application de vision par ordinateur, une attaque non spécifique à l'erreur est produite lorsqu'un attaquant modifie une image de sorte que le système d'apprentissage automatique classe mal l'objet représenté dans cette image, quelle que soit la catégorie (incorrecte) prédite.
- Attaque spécifique à l'erreur : l'attaquant vise à produire un type d'erreur spécifique. Dans une application de vision par ordinateur, par exemple, l'attaquant gère l'image malveillante pour qu'elle soit mal classée par le système d'apprentissage automatique dans une classe spécifique (incorrecte). Par exemple, si l'attaquant vise à classer mal l'image d'un chien en tant que chat.

2.5.4 Stratégie d'attaque

Une fois que nous avons tous ces éléments qui caractérisent une attaque dans un modèle d'apprentissage, à savoir la capacité, la connaissance et l'objectif, nous sommes en mesure de décrire la stratégie de l'attaquant ou d'attaque. La stratégie d'attaque peut être formulée comme un problème d'optimisation qui prend en compte les différents aspects du modèle de menace. Ainsi, compte tenu des connaissances de l'attaquant Θ et d'un ensemble d'échantillons malveillants $\mathcal{D}_p \in \phi(\mathcal{D}_p)$ que l'attaquant souhaite produire, l'objectif de l'attaquant peut être caractérisé sous forme d'une fonction objectif $A(\mathcal{D}_p, \Theta) \in \mathbb{R}$, qui mesure l'efficacité de l'attaque sous les contraintes des points d'attaque $\mathcal{D}_p$. En se basant sur ces informations, la stratégie d'attaque optimale peut être écrite comme le problème d'optimisation suivant :

$$\mathcal{D}_p^* \in \underset{\mathcal{D}_p \in \phi(\mathcal{D}_p)}{\operatorname{argmax}} A(\mathcal{D}_p, \Theta) \tag{2.16}$$

Cette formulation de haut niveau englobe à la fois les attaques d'évasion et d'empoisonnement, dans les problèmes binaires et multiclasses.

2.6 Modèle de menace considéré

En tenant compte de la surface d'attaque détaillée dans la Figure 2.5, il est possible de proposer une classification des attaques connues contre les systèmes d'apprentissage automatique. Cette classification se base sur deux dimensions principales : le moment de l'attaque (soit lors de la phase d'apprentissage, soit lors de la phase de production) et l'objectif de l'attaque. Trois grandes familles d'attaques émergent de cette classification :

- Attaques par manipulation
  Ces attaques permettent aux adversaires de contourner le comportement attendu ou même de faire en sorte que les systèmes d'IA effectuent des tâches inattendues. Avec des entrées malicieuses, les attaquants peuvent mener des attaques d'évasion (evasion attacks), reprogrammer les systèmes d'IA en temps réel (reprogramming attacks) voire procéder à des attaques beaucoup plus rudimentaires, sortes de transposition des attaques par déni de service appliquées aux systèmes d'IA.
- Attaques par infection
  Ces attaques corrompent la qualité des décisions et permettent aux attaquants d'exercer un contrôle des systèmes d'IA de façon dissimulée. Les attaquants contaminent les données utilisées pour l'entraînement, exploitent des déclencheurs cachés dans les comportements de l'IA ou distribuent des modèles d'IA malveillants via des attaques par empoisonnement (poisoning attacks), des portes dérobées (backdooring attacks) ou des chevaux de Troie (trojaning attacks).
- Attaques par exfiltration
  Ciblant la confidentialité et la protection de la vie privée, ces attaques cherchent à voler des données depuis les systèmes d'IA. Les éléments susceptibles d'être compromis incluent les échantillons de données utilisés pour l'entraînement, les détails du modèle lui-même, et les éléments internes des algorithmes. Les méthodes d'attaque courantes comprennent l'inférence d'appartenance, l'inversion de modèle et l'extraction de modèle.

La Figure 2.10 illustre la répartition des récentes attaques qui ont ciblé l'IA, classées en trois principales catégories.

Les attaques de manipulation sont les plus courantes, constituant 82.6 % des attaques contre les systèmes d'intelligence artificielle. Les attaques par infection suivent avec 9.5 %, tandis que les attaques d'exfiltration, bien qu'elles soient les moins fréquentes à 7.8 %, ne doivent pas être sous-estimées. Ces dernières, malgré leur faible fréquence, représentent des risques significatifs de violation de la confidentialité et de la protection des données. Ce

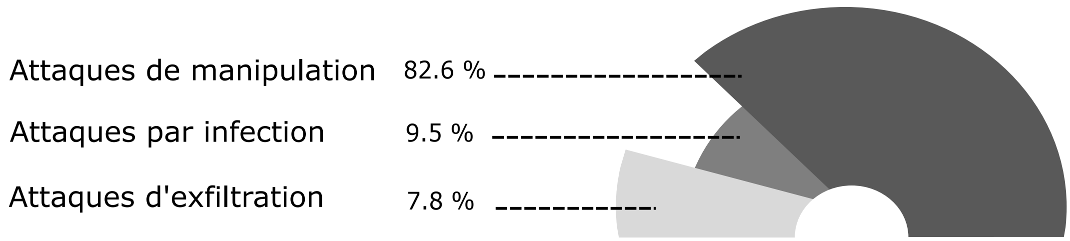

FIGURE 2.10 – Différentes catégories d'attaques contre l'apprentissage automatique [2].

constat met en évidence la nécessité d'une vigilance soutenue et de la mise en place de mesures de sécurité robustes pour toutes les catégories d'attaques.

2.7 Conclusion

L'apprentissage automatique est un domaine en rapide évolution. C'est une technologie qui permet aux systèmes informatiques d'apprendre et de s'améliorer de manière autonome pour des tâches spécifiques sans programmation explicite. Il repose sur des algorithmes complexes pour traiter et analyser les données. Toutefois, malgré ses avantages, l'apprentissage automatique n'est pas exempt de vulnérabilités pouvant être exploitées pour mener des attaques de cybersécurité.

Traditionnellement, les modèles d'apprentissage automatique opèrent sous l'hypothèse d'un environnement bienveillant. Cette supposition se révèle souvent irréaliste, offrant ainsi une opportunité d'attaques pour les adversaires en manipulant malicieusement les données d'entraînement ou de test. Ces attaques peuvent être particulièrement destructrices étant donné l'intégration profonde de l'apprentissage automatique dans diverses applications essentielles, amplifiant ainsi la nécessité d'une vigilance accrue en matière de cybersécurité de cette technologie.

Ce chapitre a formellement exploré les trois principales catégories de l'apprentissage automatique : supervisé, non supervisé, et par renforcement. Nous avons examiné les vulnérabilités inhérentes à chacune de ces catégories et décrit différents scénarios d'attaque basés sur un modèle de menace bien défini. Nous avons également présenté une taxonomie des attaques ciblant l'apprentissage automatique, comprenant les attaques de modification de données, de falsification de résultats et de diffusion de données sensibles. Cette taxonomie prend en compte le type de données ciblées (entraînement ou test), le type de modèle visé (classification ou régression) et la nature de l'attaque (perturbation ou falsification).

Le prochain chapitre présente une revue de littérature critique sur les avancées dans la protection des modèles d'apprentissage automatique. Il explore en détail les dernières recherches et les stratégies de défense, fournissant une compréhension essentielle des efforts actuels et des besoins futurs en matière de sécurisation de apprentissage machine.

Chapitre 3

État de l'art sur la sécurité de l'apprentissage automatique

3.1 Introduction

Ce chapitre offre un panorama exhaustif des attaques les plus actuelles et des défenses correspondantes affectant les modèles d'apprentissage automatique, mettant en lumière la complexité et la diversité des stratégies d'attaques. Nous analysons des tactiques spécifiques utilisées pour compromettre ou manipuler ces modèles, depuis les attaques par inférence d'appartenance jusqu'aux manipulations plus sophistiquées telles que les attaques par empoisonnement de données. La discussion s'étend également sur les défis que pose la sécurisation de ces technologies avancées. En scrutant les diverses méthodes employées pour assurer la protection de ces systèmes, ce chapitre souligne l'importance de développer des mécanismes de défense innovants et robustes. En réalisant cet état de l'art, nous identifions des pistes prometteuses pour parer aux attaques émergentes sur les systèmes d'apprentissage automatique.

3.2 Attaques d'exfiltration

Les modèles d'apprentissage automatique et plus particulièrement les modèles d'apprentissage profond se nourrissent de grands ensembles de données. Étant donné que ces ensembles de données peuvent contenir des informations privées telles que la parole de l'utilisateur, des images et des dossiers médicaux, il est essentiel que les modèles d'apprentissage automatique ne divulguent pas d'informations sensibles sur la confidentialité concernant leurs données d'entraînement. Cependant, des études récentes [28, 170, 210] ont montré que ces modèles sont susceptibles de mémoriser les informations des ensembles de données d'entraînement, ce qui les rend vulnérables à plusieurs attaques de confidentialité telles que les attaques d'extraction de modèle [186], les attaques par inférence d'attribut (également appelées attaques d'inversion de modèle) [54], les attaques par inférence de propriété [56] et les attaques par inférence d'appartenance [166]. Les attaques par extraction de modèle visent à dupliquer la fonctionnalité d'un modèle d'apprentissage automatique, c'est-à-dire qu'un attaquant tente de construire un autre modèle dont les performances prédictives sont similaires au modèle cible.

Contrairement aux attaques d'extraction de modèle ciblant le modèle d'apprentissage automatique, les autres types d'attaques par : inférence d'attribut, inférence de propriété ou inférence d'appartenance se concentrent sur la déduction d'informations privées des données d'apprentissage. Plus précisément, les attaques par inférence d'attribut visent à déduire les attributs sensibles d'un enregistrement de données cible compte tenu de la sortie d'un modèle et des informations sur les attributs non sensibles. Les attaques par inférence de propriété visent à déduire la propriété globale de l'ensemble de données d'apprentissage. Par exemple, étant donné un modèle de classificateur de logiciels malveillants dont les données de formation consistent en des traces d'exécution de logiciels malveillants et bénins, les attaques par inférence de propriété déduisent la propriété de l'environnement de test, qui peut être considérée comme une propriété de l'ensemble de données de formation [75]. Les attaques par inférence d'appartenance, connues sous le nom de Membership Inference Attacks (MIA), visent à déduire les membres de l'ensemble de données d'apprentissage du modèle.

Les attaques par inférence d'appartenance (MIA) sur les modèles d'apprentissage automatique visent à déduire si un enregistrement de données a été utilisé pour former le modèle d'apprentissage cible ou non. Les MIA peuvent entraîner de graves risques pour la vie privée des individus. Par exemple, en identifiant le fait qu'un dossier clinique a été utilisé pour former un modèle associé à une certaine maladie, les MIA peuvent déduire que le propriétaire du dossier clinique a la maladie avec un risque élevé. Un rapport récent [183] publié par le National Institute of Standards and Technology (NIST) mentionne spécifiquement qu'une attaque MIA qui détermine si un individu a été inclus dans l'ensemble de données d'entraînement utilisé pour former le modèle cible est une violation de la confidentialité. De plus, ces risques de confidentialité causés par les attaques MIA peuvent amener les entreprises commerciales qui souhaitent déployer l'apprentissage automatique en tant que service à enfreindre les réglementations en matière de confidentialité.

Le concept de MIA est d'abord proposé par Homer *et al.* [73] où ils démontrent qu'un attaquant peut exploiter les statistiques publiées sur un ensemble de données génomiques pour déduire la présence d'un génome particulier dans cet ensemble de données. De plus, des articles récents [144] ont montré la faisabilité des MIA sur les données de localisation. Outre les MIA sur ces bases de données, Shokri *et al.* [166] ont proposé les premiers MIA sur plusieurs modèles de classification dans le cadre de l'apprentissage automatique. Ils démontrent qu'un attaquant peut identifier si un enregistrement de données a été utilisé pour former un classificateur basé sur un réseau neuronal ou non, uniquement sur la base du vecteur de prédiction de l'enregistrement de données (qui est également connu sous le nom d'accès par boîte noire au modèle d'apprentissage cible). Depuis lors, il y a eu un nombre croissant d'études qui étudient les MIA sur divers modèles d'apprentissage, y compris les modèles de régression, les modèles de classification, les modèles de génération et les modèles d'intégration. Pendant ce temps, un grand nombre de travaux proposent différentes défenses d'inférence d'appartenance à partir de différentes perspectives pour se défendre contre les MIA tout en préservant l'utilité des modèles d'apprentissage cibles. Nous allons voir, dans ce qui suit, plus en détails les principaux travaux.

3.2.1 Définition formelle des attaques par inférence d'appartenance (MIA)

Pour mieux illustrer la définition des MIA, nous considérons le processus d'apprentissage décrit par la Figure 2.2, défini dans le chapitre 2. C'est un processus d'apprentissage typique d'un classificateur de réseau neuronal profond. Un algorithme d'apprentissage $\mathcal{A}$ est utilisé pour former le classificateur $f(x, \theta)$ en utilisant l'ensemble de données $D_{train} = \{x^{(n)}, y^{(n)}\}_{n=1}^{N}$. Une fois le processus de formation terminé, le modèle appris $f(x, \theta^*)$ peut être utilisé pour faire des prédictions pour de nouvelles données. La définition des MIA sur les modèles d'apprentissage automatique est la suivante : étant donné une entrée exacte x et un accès au modèle formé $f(x, \theta^*)$, un attaquant en déduit si $x \in D_{train}$ ou pas.

3.2.2 Connaissances de l'attaquant

Comme nous l'avons vue dans le chapitre 2, pour réaliser des attaques sur des modèles d'apprentissage automatique, un attaquant peut avoir accès à deux types d'information : la connaissance des données d'entraînement et la connaissance du modèle cible. La connaissance des données d'entraînement se réfère à la distribution des données utilisées pour entraîner le modèle. Dans la plupart des situations d'attaque, il est supposé que l'attaquant a accès à la distribution des données d'apprentissage. Cela signifie qu'il peut obtenir un jeu de données "fantôme" comprenant des enregistrements de données provenant de la même distribution que les données utilisées pour entraîner le modèle. Cette hypothèse est raisonnable car le jeu de données fantôme peut être obtenu en utilisant soit une synthèse basée sur les statistiques lorsque la distribution des données est connue, soit une synthèse basée sur un modèle lorsque la distribution des données est inconnue. Dans la plupart des cas, il est supposé que le jeu de données fantôme et le jeu de données d'apprentissage sont disjoints pour réaliser une attaque non triviale [166]. La connaissance du modèle cible se réfère à la façon dont le modèle a été formé (c'est-à-dire l'algorithme d'apprentissage utilisé $mathcalA$), sa structure et ses paramètres appris. En utilisant ces connaissances de l'adversaire, nous pouvons caractériser les niveaux de dangerosité des attaques existantes.

L'attaquant peut disposer de deux types de connaissances : les connaissances en boîte blanche et les connaissances en boîte noire. Dans le premier cas de figure, l'attaquant a accès à toutes les informations qui composent le modèle d'apprentissage, de la distribution des données d'entraînement, à l'algorithme d'apprentissage en passant par les paramètres du modèle et enfin le vecteur de prédiction. Quant au cas de la boîte noire, les informations dont il dispose sont limitées. L'adversaire peut connaître une partie de la distribution des données d'entraînement et la réponse à ses requêtes lorsqu'il interroge le modèle entraîné. Par exemple, l'attaquant interroge le classificateur cible (si le modèle cible est un modèle de classification) et n'obtient que la sortie de prédiction de l'enregistrement d'entrée. Cependant, si les MIA en boîte noire peuvent fonctionner, elles seraient plus dangereuses que les MIA en boîte blanche, car l'attaquant peut violer la confidentialité des membres avec une connaissance limitée.

3.2.3 Approches pour des attaques par inférence d'appartenance (MIA)

Les modèles de machine learning (ML) tels que les réseaux de neurones profonds (DNN) ont souvent plus de paramètres qu'il n'en faut, ce qui signifie qu'ils ont la capacité de mémoriser les détails de leur jeu de données d'apprentissage. [28, 128, 170, 210].

De plus, les jeux de données d'entraînement ont une taille finie et les modèles d'apprentissage sont entraînés sur des époques successives (souvent des dizaines ou des centaines) en utilisant les mêmes données à plusieurs reprises. Par conséquent, les modèles d'apprentissage ont un comportement différent sur les enregistrements de données d'entraînement (c'est-à-dire les "membres") par rapport aux enregistrements de données de test (c'est-à-dire les "non-membres"), ainsi que sur les paramètres du modèle qui stockent des informations statistiquement corrélées sur des données spécifiques. Par exemple, un modèle de classification pourrait classer un enregistrement de données d'apprentissage dans la bonne classe avec une forte confiance, tout en classant un enregistrement de données de test dans la bonne classe avec une confiance relativement faible. Ces différences de comportement des modèles d'apprentissage permettent à un attaquant de créer des modèles d'attaque pour distinguer les membres des non-membres du jeu de données d'apprentissage. Selon la façon dont est construit le modèle d'attaque, il existe deux principaux types d'approches d'attaque, à savoir les approches basées sur un classificateur binaire et les approches basées sur des métriques.

3.2.3.1 Attaques par inférence d'appartenance basée sur un classificateur binaire

Essentiellement, une attaque de type boîte noire basée sur un classificateur binaire consiste à entraîner un classificateur binaire capable de distinguer le comportement du modèle cible sur ses données d'apprentissage ("membres") de son comportement sur les données de test ("non-membres"). Le défi est de savoir comment entraîner ce classificateur binaire. Une technique efficace appelée "shadow training" proposée par Shokri *et al.* [166] est l'une des premières et peut-être la plus largement utilisée pour entraîner une attaque de type boîte noire basée sur un classificateur binaire. L'idée principale est que l'attaquant peut créer plusieurs modèles "fantômes" qui imitent le comportement du modèle cible, car il est supposé connaître la structure et l'algorithme d'apprentissage utilisés par le modèle cible. Pour ces modèles fantômes, l'attaquant dispose de ses propres jeux de données d'apprentissage et de test, et peut donc créer un jeu de données contenant les caractéristiques et la vérité terrain de l'appartenance des enregistrements de données d'apprentissage et de test. En utilisant ce jeu de données, l'attaquant peut entraîner le modèle d'attaque basé sur un classificateur binaire.

La Figure 3.1 décrit comment utiliser un processus appelé "entraînement shadow" pour entraîner un modèle d'attaque à partir de classificateurs binaires. L'ensemble de données d'entraînement privé D_{train} est utilisé pour entraîner un classificateur cible en utilisant l'algorithme d'apprentissage $\mathcal{A}$. Ensuite, l'attaquant crée des ensembles de données d'entraînement shadow $D'_1, \cdots, D'k$ qui viennent de la même distribution que celle de $Dtrain$. Plus il y a de modèles shadow, plus le modèle d'attaque peut être précis, car on aura plus de matériel d'entraînement pour le modèle d'attaque [166]. Ces ensembles de données shadow sont utilisés pour entraîner k modèles shadow qui sont censés imiter le comportement du modèle cible. Chaque modèle shadow est testé sur un ensemble de données de test shadow

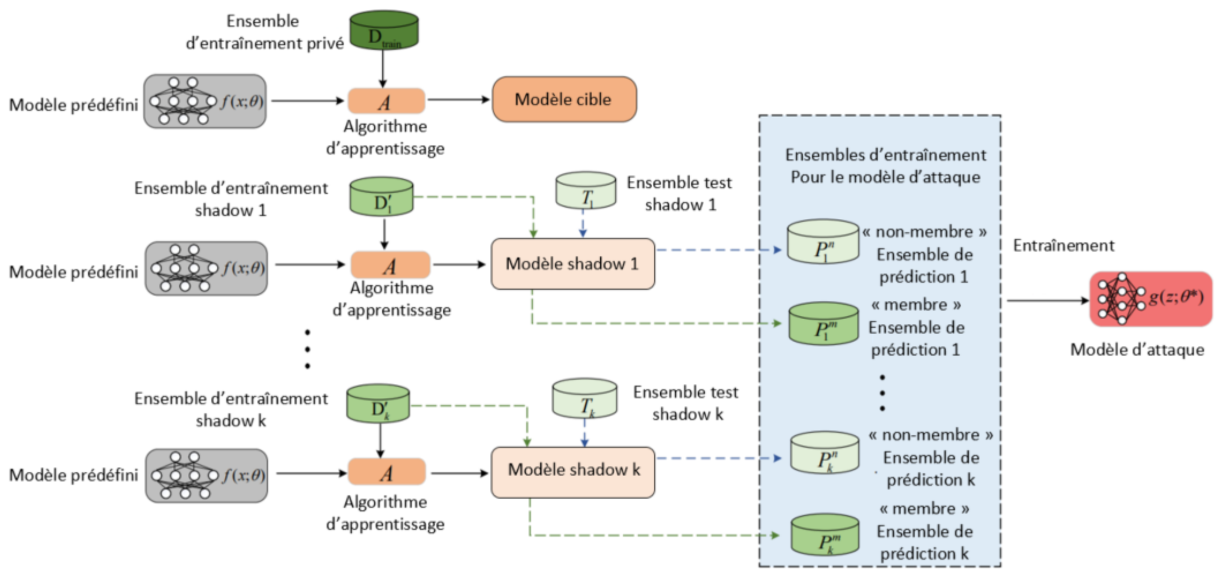

FIGURE 3.1 – Aperçu de la technique d'entraînement "shadow".

correspondant $T_1, \cdots, T_k$. L'attaquant utilise les résultats de ces tests pour créer k ensembles de données "membres" et "non-membres" qui sont utilisés pour entraîner un modèle d'attaque. Ce modèle d'attaque est utilisé pour résoudre un problème de classification binaire qui consiste à déterminer si un élément appartient à l'ensemble de données d'apprentissage ou non. Étant donné que la classification binaire est une tâche d'apprentissage automatique standard, l'attaquant peut utiliser n'importe quel modèle d'apprentissage automatique de pointe qui peut être utilisé pour créer ce modèle d'attaque.

La technique d'entraînement shadow peut être utilisée pour créer des modèles d'attaque en "boîte blanche" ou en "boîte noire". Dans les deux cas, le processus de création du modèle d'attaque est le même, comme illustré par la figure 3.1. Toutefois, selon le type de connaissances que possède l'attaquant (boîte blanche ou boîte noire), la quantité d'informations collectées sur les membres et les non-membres peut varier. Si l'attaquant a des connaissances en boîte noire, il ne peut obtenir que le vecteur de prédiction d'un enregistrement donné lorsqu'il interroge le modèle cible. Ainsi, lors de l'interrogation des modèles shadow avec leurs propres ensembles de données d'apprentissage et de test, l'attaquant ne collecte que les vecteurs de prédiction de chaque enregistrement de données. Cependant, avec des connaissances en boîte blanche, l'attaquant a un accès complet au modèle cible, ce qui signifie qu'il peut voir les calculs intermédiaires dans les couches cachées et le vecteur de prédiction d'un enregistrement donné. Ainsi, lors de l'interrogation des modèles shadow en boîte blanche, l'attaquant peut collecter des vecteurs de prédiction ainsi que les calculs intermédiaires de chaque enregistrement de données. En comparaison avec les attaques MIA en boîte noire, l'attaquant en boîte blanche a beaucoup plus d'informations pour construire son modèle d'attaque.

3.2.3.2 Attaques par inférence d'appartenance basée sur des métriques

Les attaques d'inférence sur l'appartenance (MIA) basées sur des métriques sont plus simples et moins coûteuses en termes de calcul que les MIA basées sur un classificateur binaire.

Ces attaques MIA basées sur des métriques prennent des décisions sur l'appartenance d'un enregistrement de données en calculant d'abord des métriques sur son vecteur de prédiction, puis en comparant ces métriques à un seuil prédéterminé. Il existe quatre types principaux d'attaques MIA basées sur des métriques : basées sur l'exactitude de la prédiction, sur la perte de prédiction, sur la confiance de la prédiction et sur l'entropie de la prédiction. Chaque type d'attaque est désigné par M(·), qui code les membres en 1 et les non-membres en 0. Des détails sur chaque type d'attaque sont donnés dans les références des articles qui les ont proposées ou utilisées.

a. MIA basée sur l'exactitude des prédictions [207]

Un attaquant considère qu'un enregistrement d'entrée x appartient à l'ensemble de données d'entraînement (membre) s'il est correctement prédit par le modèle cible, sinon il considère qu'il n'en fait pas partie (non-membre). L'idée est que le modèle cible est formé pour bien prédire sur ses données d'entraînement, mais peut ne pas généraliser bien ces prédictions sur des données de test. L'attaque $\mathcal{M}_{corr}(.,.)$ est définie comme suit :

$$\mathcal{M}_{corr}(\hat{p}(y|x), y) = \mathbf{1}(\text{argmax} \;\; \hat{p}(y|x) = y)$$

où $\mathbf{1}(.)$ est la fonction indicatrice définie comme suit :

$$\mathbf{1} = \begin{cases} 1 & \text{si l'évènement A se produit} \\ 0 & \text{sinon.} \end{cases}$$

b. MIA basée sur la perte de prédiction [207]

Un attaquant considère qu'un enregistrement d'entrée appartient à l'ensemble de données d'entraînement (membre) si sa perte de prédiction est inférieure à la perte de prédiction moyenne de tous les membres d'entraînement, sinon il considère qu'il n'en fait pas partie (non-membre). L'idée est que le modèle cible est entraîné sur ses membres d'entraînement en minimisant leur perte de prédiction. Ainsi, la perte de prédiction d'un enregistrement d'entraînement devrait être inférieure à celle d'un enregistrement de test. L'attaque $\mathcal{M}_{loss}(.,.)$ est définie comme suit :

$$\mathcal{M}_{loss}(\hat{p}(y|x), y) = \mathbf{1}(\mathcal{L}(\hat{p}(y|x), y)) \leq \tau$$

où $\mathcal{L}(.,.)$ est la fonction de perte d'entropie croisée et τ est un seuil prédéfini.

c. MIA basée sur la confiance des prévisions [155]

Lorsqu'un attaquant tente de déterminer si un enregistrement est un membre ou non, il utilise un modèle de prédiction qui lui donne un score de confiance. Si ce score est supérieur à un certain seuil prédéfini, l'attaquant considère que l'enregistrement est un membre. Si le score est inférieur à ce seuil, l'attaquant considère que l'enregistrement n'est pas un membre. Le modèle de prédiction est entraîné de manière à minimiser les erreurs de prédiction sur les données d'entraînement, ce qui signifie que le score de confiance maximal pour un enregistrement membre devrait être proche de 1. L'attaque $\mathcal{M}_{conf}(.,.)$ est définie comme suit :

$$\mathcal{M}_{conf}(\hat{p}(y|x), y) = \mathbf{1}(max(\hat{p}(y|x), y)) \leq \tau$$

d. MIA basée sur l'entropie de prédiction [155]

Lorsqu'un attaquant tente de déterminer si un enregistrement est un membre ou non, il utilise un modèle de prédiction qui lui donne une mesure de l'incertitude de la prédiction appelée entropie. Si cette entropie est inférieure à un certain seuil prédéfini, l'attaquant considère que l'enregistrement est un membre, sinon il considérera l'enregistrement comme non-membre. On s'attend généralement à ce que les distributions d'entropie de prédiction entre les données d'entraînement et de test soient très différentes, avec le modèle cible ayant une entropie de prédiction plus élevée sur les données de test que sur les données d'entraînement. L'entropie d'un vecteur de prédiction $\hat{p}(y|x)$ est définie comme suit :

$$H(\hat{p}(y|x)) = -\sum_i p_i log(p_i),$$

où p_i est le score de confiance dans $\hat{p}(y|x)$, L'attaque $\mathcal{M}_{entr}(.,.)$ est définie comme suit :

$$\mathcal{M}_{entr}(\hat{p}(y|x)) = \mathbf{1}(H(\hat{p}(y|x), y)) \leq \tau$$

3.2.4 Attaques par inférence d'appartenance sur différents modèles d'apprentissage automatique

Depuis les premiers travaux sur les attaques MIAs pour les modèles de classification [166], de nombreuses études ont été menées sur les MIA pour différents types de modèles d'apprentissage automatique, comme les modèles de classification, les modèles génératifs et les modèles de régression. Dans cette partie du rapport, nous présentons quelques exemples de la littérature sur les MIA pour ces types de modèles. Ces articles proposent de nouvelles techniques de MIA pour étudier les risques pour la confidentialité des membres.

3.2.4.1. MIAs dans les modèles de classification

Les attaques par inférence d'appartenance (MIA) sont souvent utilisées pour cibler des modèles de classification, dans lesquels un attaquant essaie de déterminer si une donnée a été utilisée pour former un classificateur cible. Shokri *et al.* [166] ont été les premiers à proposer une MIA pour les modèles de classification en utilisant une technique de formation de modèles shadow dans un environnement de boîte noire. Salem *et al.* [155] ont ensuite amélioré cette technique en allégeant deux hypothèses principales, à savoir l'utilisation de plusieurs modèles shadow et la connaissance de la distribution des données d'entraînement. Ils ont montré qu'un seul modèle shadow pouvait être aussi efficace qu'une utilisation de plusieurs modèles et ont proposé une attaque de transfert de données dans laquelle les données utilisées pour entraîner le modèle shadow n'ont pas besoin d'avoir la même distribution que les données d'entraînement privées du modèle cible. De plus, le modèle shadow n'a pas besoin d'avoir la même structure que le modèle cible. Salem *et al.* ont également proposé deux nouvelles attaques utilisant les scores de confiance et l'entropie de prédiction les plus élevés en plus d'étendre les MIA pour les classificateurs binaires existants.

Nasr *et al.* [130] ont proposé les premières attaques par inférence d'appartenance (MIAs) en boîte blanche, dans lesquelles un attaquant connaît les paramètres internes du modèle cible. Ces MIAs en boîte blanche sont une extension des MIAs en boîte noire pour les classificateurs binaires. Elles visent à améliorer les performances d'attaque en utilisant les calculs intermédiaires d'une entrée au travers du modèle cible. Elles utilisent les gradients de la perte de prédiction d'une entrée par rapport aux paramètres du modèle cible comme caractéristiques supplémentaires pour déduire l'appartenance de l'entrée. Leino et Fredrikson [105] ont critiqué les travaux de boîte blanche de Nasr *et al.* comme étant des hypothèses trop fortes, s'écartant de la plupart des cadres des MIA. Ils ont proposé une MIA en boîte blanche qui n'a pas besoin de membres de formation du modèle cible.

3.2.4.2. MIAs dans les modèles génératifs

Un réseau antagoniste génératif (GAN) est un modèle d'apprentissage automatique qui comprend deux réseaux de neurones qui s'opposent l'un l'autre, appelés le générateur G et le discriminateur D. Le but de ce modèle est de produire du contenu qui ressemble à une entrée donnée. Les MIAs sur les modèles génératifs visent à déterminer si un enregistrement de données a été utilisé pour entraîner le générateur ou non, ce qui est plus difficile à faire que pour les modèles de classification.

Hayes *et al.* [66] ont présenté la première attaque par inférence d'appartenance contre les modèles génératifs, qui peut être utilisée dans le cadre d'une "boîte noire" ou d'une "boîte blanche". Dans le cadre d'une boîte blanche, l'attaquant soumet tous les enregistrements de données au discriminateur du GAN cible, qui produit des scores de confiance indiquant la probabilité qu'un enregistrement de données ait été utilisé pour entraîner le générateur. L'attaquant trie ces scores de confiance par ordre décroissant et sélectionne les enregistrements de la première moitié comme étant ceux qui ont été utilisés pour l'entraînement. Dans le cadre d'une boîte noire, l'attaquant utilise des enregistrements générés par le générateur cible pour entraîner un GAN local, qu'il utilise ensuite pour mener une attaque MIA en suivant la même approche que dans le cadre d'une boîte blanche.

Hilprecht *et al.* [70] ont proposé deux attaques MIA contre des modèles génératifs. L'une de ces attaques, l'attaque d'intégration de Monte Carlo [150], s'applique aux GANs dans le cadre d'une "boîte noire" et utilise des enregistrements générés proches d'un enregistrement cible pour estimer la probabilité que cet enregistrement ait été utilisé pour l'entraînement du générateur. L'autre attaque, l'attaque de reconstruction, s'applique aux VAEs dans le cadre d'une "boîte blanche" et utilise la fonction de perte du VAE pour calculer l'erreur de reconstruction d'un enregistrement cible et déterminer s'il a été utilisé pour l'entraînement. Liu *et al.* [111] ont également proposé une attaque, appelée inférence de co-appartenance, qui vise à déterminer si un ensemble d'enregistrements a été utilisé pour l'entraînement. Cette attaque commence par tenter d'identifier si un seul enregistrement cible a été utilisé pour l'entraînement, puis étend cette identification à un ensemble d'enregistrements. Contrairement aux attaques de Hilprecht *et al.*, cette attaque nécessite de recycler de nouveaux réseaux de neurones pour chaque enregistrement d'entrée, tandis que les attaques de Hilprecht *et al.* ne nécessitent que des enregistrements synthétiques fixes générés par le générateur cible.

3.2.4.3 MIAs dans les modèles de régression

Gupta *et al.* [65] ont présenté les premières attaques MIA contre les modèles de régression profonde. Ils se sont concentrés sur les problèmes de prédiction d'âge où les modèles de régression sont utilisés pour prédire l'âge d'une personne à partir de son IRM cérébrale. Pour démontrer la vulnérabilité de ces modèles aux MIA, Gupta *et al.* ont supposé que l'attaquant avait accès en "boîte blanche" au modèle cible et avait accès à certains enregistrements de l'ensemble de données d'entraînement privé. L'attaque utilisée est un classificateur binaire qui exploite les caractéristiques des gradients des paramètres, des activations, des prédictions et des étiquettes d'enregistrements cibles pour déterminer s'ils ont été utilisés pour l'entraînement ou non.

3.2.5 Techniques de défense contre les MIAs

Dans cette section, nous allons présenter les défenses contre les attaques MIAs sur les modèles d'apprentissage automatique. Les méthodes de défense existantes contre les MIA peuvent être regroupées en quatre grandes catégories : le masquage du score de confiance, la régularisation, la distillation des connaissances et la confidentialité différentielle.

3.2.5.1 Masquage du score de confiance

Le masquage du score de confiance est une technique utilisée pour protéger les modèles de classification contre les attaques MIA de type "boîte noire". Il s'agit de masquer les scores de confiance réels renvoyés par le modèle cible, ce qui réduit l'efficacité des attaques. Il existe trois méthodes pour mettre en œuvre cette défense : la première consiste à ne fournir que les top-k des scores de confiance lorsque l'attaquant interroge un enregistrement d'entrée ; la deuxième consiste à ne fournir que l'étiquette de prédiction ; et la troisième consiste à ajouter du bruit au vecteur de prédiction. Ces méthodes ne nécessitent pas de modification du modèle cible et n'affectent pas sa précision.

Shokri *et al.* [166] ont évalué l'efficacité de la méthode consistant à limiter le vecteur de prédiction aux trois premières classes sur un classificateur basé sur un réseau de neurones entièrement connecté, en utilisant deux ensembles de données. Ils ont constaté que cette technique ne réduisait pas la précision de leur attaque basée sur l'entraînement shadow. Cela n'est pas surprenant, car une étude ultérieure [155] a montré qu'une attaque basée sur un classificateur binaire en boîte noire qui utilise des scores de confiance partiels peut atteindre des performances similaires à celles obtenues avec l'utilisation de vecteurs de prédiction complets. La seconde méthode, qui consiste à ne révéler que l'étiquette de prédiction, a été étudiée par Li et Zhang [108] et Choquette *et al.* [35]. Ces études ont montré que même lorsque l'attaquant ne dispose que d'étiquettes de prédiction, il peut toujours obtenir de bonnes performances d'attaque.

Des chercheurs (Jia *et al.*, 2019 [92]) ont découvert que lorsqu'un réseau de neurones profond (DNN) est utilisé comme modèle d'attaque en tant que classificateur binaire, il est vulnérable aux exemples contradictoires. Ils ont donc utilisé une technique d'apprentissage automatique contradictoire (Kurakin, 2016 [103]) pour développer une méthode de défense appelée MemGuard. MemGuard ajoute un vecteur de bruit spécifiquement conçu au vecteur de prédiction du modèle cible afin de masquer les scores de confiance réels et de rendre l'attaque moins efficace. Cette technique peut être utilisée sans avoir à modifier le modèle cible ni affecter sa précision.

3.2.5.2 Régularisation

Le surapprentissage des modèles cibles est l'un des principaux facteurs qui peut favoriser le succès des attaques de type MIA. Selon plusieurs études [30, 105, 155, 166, 207], le surapprentissage des modèles d'apprentissage automatique se produit lorsqu'un modèle performe mieux sur ses données d'entraînement que sur ses données de test, ce qui signifie qu'il ne parvient pas à généraliser ses résultats sur de nouvelles données. Le surapprentissage des modèles d'apprentissage est souvent causé par une complexité élevée du modèle et/ou une taille limitée de l'ensemble de données d'apprentissage [20]. Les modèles d'apprentissage profond, tels que les réseaux de neurones profonds (DNN), ont tendance à être surparamétrés et à avoir une complexité élevée, ce qui peut les aider à apprendre efficacement à partir de grandes quantités de données, mais peut également les rendre capables de mémoriser inutilement le bruit ou les détails spécifiques à un ensemble de données d'apprentissage donné [28, 128].

La régularisation vise à réduire le surapprentissage des modèles et peut être utilisée pour se protéger contre les MIA. Il existe plusieurs méthodes de régularisation couramment utilisées, telles que la norme L2, l'abandon, l'argumentation des données, l'empilement de modèles, l'arrêt précoce, le lissage d'étiquettes, qui ont toutes été conçues pour améliorer la généralisabilité d'un modèle d'apprentissage automatique. Ces techniques ont été efficaces pour atténuer les erreurs de généralisation, car elles aident le modèle formé à mieux généraliser sur de nouvelles données et à réduire la différence entre ses comportements sur les données d'apprentissage et de test. De plus, il existe deux méthodes de régularisation spécialement conçues pour atténuer les MIA, connues sous le nom de régularisation contradictoire et Mixup + MMD [1]. Ces deux techniques ajoutent de nouvelles contraintes de régularisation à l'objectif d'un classifieur pendant la phase d'apprentissage, obligeant le classifieur à produire des distributions de sortie similaires pour les éléments appris et non appris.

La régularisation vise à réduire le degré de surapprentissage des modèles cibles et par conséquent elle peut être utilisée pour se défendre contre les MIA. Nous avons les méthodes de régularisation classiques suivantes : la norme L2, l'abandon, l'argumentation des données, l'empilement de modèles, l'arrêt précoce, le lissage d'étiquettes qui sont proposées pour améliorer la généralisabilité d'un modèle d'apprentissage automatique formé. Ces méthodes se sont avérées efficaces pour atténuer les MIA. En effet, elles aident le modèle formé à mieux généraliser pour tester les données et à réduire la différence des comportements du modèle sur ses données d'apprentissage et ses données de test. D'autre part, nous avons la régularisation contradictoire [130] et Mixup + MMD [107] qui sont des techniques de régularisation spécialement conçues pour atténuer les MIA. Les deux méthodes proposées ajoutent de nouveaux

1. La régularisation Mixup consiste à mélanger deux exemples de données de manière à créer de nouvelles données synthétiques. Cela permet de lisser les frontières de décision du modèle et de réduire la sur-spécialisation sur les données d'entraînement. Le MMD (Maximum Mean Discrepancy) est une mesure de la divergence de deux distributions de données. La régularisation MMD consiste à ajouter un terme de régularisation à la fonction de coût du modèle qui pénalise les modèles qui ont des distributions de prédiction trop différentes de celle des données d'entraînement. La régularisation Mixup + MMD combine ces deux techniques de régularisation en utilisant Mixup pour générer de nouvelles données synthétiques et MMD pour s'assurer que la distribution de prédiction du modèle reste proche de celle des données d'entraînement. Cela peut aider à améliorer la généralisation du modèle et à réduire l'overfitting.

termes de régularisation à la fonction objectif d'un modèle d'apprentissage cible pendant la phase d'apprentissage et l'obligent à générer des distributions de sortie similaires pour les données "membres" et les données "non-membres" de l'apprentissage.

Il est important de noter que les techniques de régularisation ne sont pas limitées aux modèles de classification et peuvent également être utilisées pour atténuer les erreurs de généralisation sur les modèles de génération (GAN). Par exemple, il a été démontré [66] que l'abandon peut être utilisé comme une technique de défense efficace contre les erreurs de généralisation dans les GAN (réseaux de génération adverses). Contrairement aux méthodes de masquage des scores de confiance, la régularisation protège contre les erreurs de généralisation, quel que soit le contexte de l'attaquant (boîte noire ou boîte blanche). En effet, les techniques de régularisation modifient non seulement la distribution de sortie des modèles cibles, mais aussi leurs paramètres internes, tandis que les méthodes de masquage des scores de confiance ne modifient que les vecteurs de prédiction des modèles.

Bien que les méthodes de régularisation soient efficaces et largement applicables, l'un de leurs inconvénients est qu'elles peuvent ne pas être en mesure de fournir des compromis satisfaisants entre la confidentialité et l'utilité des membres. Par exemple, Shokri *et al.* [166] montrent que la régularisation de la norme L2 peut atténuer la précision des MIA à un niveau de conjecture aléatoire lors de la définition du facteur de régularisation à des valeurs relativement élevées. Cependant, cela entraîne une réduction significative de la précision de prédiction du modèle cible.

3.2.5.3 Distillation des connaissances

La distillation des connaissances est une technique de transfert de connaissances qui consiste à entraîner un modèle de petite taille (appelé "étudiant") pour qu'il reproduise les prédictions d'un modèle de grande taille (appelé "enseignant") [71]. Le modèle enseignant est généralement pré-entraîné sur une grande base de données et possède de bonnes performances, tandis que le modèle étudiant est entraîné à partir de zéro ou à partir d'un modèle pré-entraîné de petite taille. Cela permet d'améliorer les performances de modèles de petite taille en leur permettant de bénéficier des connaissances acquises par un modèle plus grand et plus puissant [38].

La méthode de défense Distillation for Membership Privacy (DMP) [161] propose d'utiliser la distillation des connaissances pour protéger la vie privée des membres d'un ensemble de données d'entraînement. DMP nécessite deux ensembles de données : un ensemble de données d'entraînement privé et un ensemble de données de référence non étiqueté. Pour utiliser DMP, on commence par entraîner un modèle enseignant non protégé sur l'ensemble de données d'entraînement privé et on l'utilise pour étiqueter les enregistrements de l'ensemble de données de référence. Ensuite, on sélectionne les enregistrements de l'ensemble de données de référence étiqueté qui ont une faible entropie de prédiction et on utilise ces enregistrements pour entraîner le modèle cible. L'idée de cette sélection est que ces enregistrements sont faciles à classer et ne seront pas significativement affectés par les membres de l'ensemble de données d'entraînement privé.

En fin de compte, DMP entraîne un modèle privé en utilisant uniquement les enregistrements de l'ensemble de données de référence étiqueté qui ont été sélectionnés. L'idée derrière DMP est de limiter l'accès direct du modèle privé à l'ensemble de données d'entraînement privé afin de réduire au minimum la fuite d'informations sur les membres de cet ensemble de données.

3.2.5.4 Confidentialité différentielle

La confidentialité différentielle (DP) [46] est un mécanisme de confidentialité probabiliste qui permet de protéger les données individuelles tout en permettant de collecter des informations statistiques utiles à partir d'un grand ensemble de données. Elle consiste à ajouter un bruit aléatoire aux données avant de les partager. Elle a été utilisée dans de nombreux travaux [32, 66, 79, 80, 87, 89, 92, 107, 107, 146, 162, 208] pour réduire les fuites de données sensibles dans les modèles d'apprentissage automatique. En effet, au cours de la dernière décennie, la confidentialité différentielle a attiré une attention considérable en raison de ses garanties de confidentialité prouvées et de sa capacité à quantifier la perte de confidentialité.

La confidentialité différentielle a été initialement proposée pour les requêtes statistiques interactives à une base de données. Nous dirons qu'une opération sur un ensemble de données D_1 est différentiellement privée si nous pouvons inférer environ la même quantité d'information sur un individu I, qu'il soit présent ou non dans D_1. Autrement dit, le résultat d'une analyse différentiellement privée sera à peu près le même qu'on inclut ou pas l'individu I. Une analyse différentiellement privée est souvent appelée mécanisme, et nous le notons M. Plus formellement, nous pouvons dire :

Définition 3.2.1. ϵ-confidentialité différentielle
Un mécanisme aléatoire M satisfait ϵ-confidentialité différentielle si pour tous les ensembles voisins de données D_1 et D_2, qui diffèrent d'un seul enregistrement, et tous les $S \subset Range(M)$, on suppose que :

$$Pr(M(D_1) \in S) \leq exp(\epsilon) \times Pr(M(D_2) \in S). \tag{3.1}$$

En d'autres termes, la présence ou l'absence d'un seul enregistrement ne doit pas être perceptible dans les réponses du mécanisme, jusqu'à un facteur exponentiel de ϵ qui dénote le budget de confidentialité. Nous contrôlons la force de la garantie de la confidentialité en ajustant le budget de confidentialité. Plus ϵ est petit, plus la protection est élevée. Le bruit à ajouter à la réponse pour imposer un certain ϵ-confidentialité différentielle dépend de la sensibilité globale de la requête à la présence ou à l'absence d'un seul enregistrement. Un bruit léger peut suffire pour les requêtes statistiques telles que la moyenne, tandis qu'un bruit plus important est nécessaire pour les requêtes d'identité renvoyant le contenu d'un enregistrement spécifique. Ce bruit supplémentaire a le potentiel de rendre les résultats moins utiles. Le défi consiste alors à déterminer où ajouter le bruit et quelle quantité à ajouter.
Nous pouvons également relaxer la définition en ajoutant un paramètre additif δ qui nous permet d'ignorer les événements très rares [46].

Définition 3.2.2. (ϵ, δ)-confidentialité différentielle
Un mécanisme aléatoire M satisfait (ϵ, δ)-confidentialité différentielle si pour tous les ensembles voisins de données D_1 et D_2. qui diffèrent d'un enregistrement, et tous les $S \subset Range(M)$, on suppose que :

$$Pr(M(D_1) \in S) \leq exp(\epsilon) \times Pr(M(D_2) \in S) + \delta. \tag{3.2}$$

Deux théorèmes sont largement utilisés pour résoudre des problèmes complexes liés à la protection de la vie privée des données. Ils permettent de concevoir des méthodes différentiellement privées complexes en utilisant des blocs de construction privés plus simples [47].

Théorème 3.2.1. *Composition parallèle [118]*
Soit un ensemble de mécanismes privés $M = \{M_1, \cdots, M_m\}$, où chaque mécanisme M_i fournit une garantie de confidentialité différentielle de niveau ϵ_i sur un sous-ensemble disjoint de l'ensemble de données, alors l'ensemble de mécanismes M fournira une garantie de confidentialité privée maximale de niveau $max\{\epsilon_1, \cdots, \epsilon_m\}$.

Théorème 3.2.2. *Composition séquentielle [118]*
Soit un ensemble de mécanismes privés $M = \{M_1, \cdots, M_m\}$, exécutés séquentiellement sur un ensemble de données, chaque M_i fournit une garantie de confidentialité différentielle de niveau ϵ_i, alors M fournira une confidentialité différentielle de niveau $\sum_i \epsilon_i$.

La sensibilité est un concept clé dans la confidentialité différentielle. Elle mesure l'impact sur la confidentialité des données lorsqu'une seule entrée est modifiée. Plus précisément, elle définit la quantité d'information supplémentaire qui peut être révélée sur une entrée privée lorsqu'elle est comparée à une entrée similaire. Le niveau de sensibilité détermine également la quantité minimale de perturbation nécessaire pour rendre les réponses à une requête privées. Il est donc important de comprendre et de gérer adéquatement la sensibilité pour garantir la protection de la vie privée des données dans le contexte de la confidentialité différentielle.

Définition 3.2.3. *L_1-sensibilité globale [45]*
Soit deux jeux de données voisins D_1 et D_2, la L_1-sensibilité globale d'une requête $f : D_1 \to \mathbb{R}^n$ est définie comme :

$$\Delta f = max_{D_1,D_2}||f(D_1) - f(D_2)||_1 \tag{3.3}$$

La L_1-sensibilité globale d'une requête f mesure la variation maximale entre les résultats obtenus lorsque f est effectuée sur D_1 et D_2 . Elle reflète le niveau de protection nécessaire de la vie privée d'un individu. La L_2-sensibilité, quant à elle, est basée sur la norme L_2 des résultats de la requête.

La confidentialité différentielle présente plusieurs avantages importants par rapport aux techniques de confidentialité précédentes, comme le fait qu'elle suppose que toutes les informations sont des informations d'identification, éliminant ainsi la tâche difficile (et parfois impossible) de comptabiliser tous les éléments d'identification des données. De plus, elle résiste aux attaques de confidentialité basées sur des informations auxiliaires, de sorte qu'elle peut efficacement empêcher les attaques de liaison qui sont possibles sur des données anonymisées. Elle présente également une propriété de composition qui nous permet de déterminer la perte de confidentialité de l'exécution de deux analyses différentiellement privées sur les mêmes données en additionnant simplement les pertes de confidentialité individuelles pour les deux analyses. La compositionnalité signifie que nous pouvons apporter des garanties significatives en matière de confidentialité, même lorsque nous publions plusieurs résultats d'analyse à partir des mêmes données. Des techniques telles que l'anonymisation ne sont pas compositionnelles, et plusieurs diffusions dans le cadre de ces techniques peuvent entraîner une perte catastrophique de confidentialité.

Shokri *et al.* [166] ont d'abord discuté du fait que les modèles différentiellement privés devraient pouvoir atténuer les MIA sur les modèles d'apprentissage. Yeom *et al.* [208] relient théoriquement la confidentialité différentielle aux MIA et prouvent que l'avantage d'adhésion d'un attaquant est limité par une fonction du budget de confidentialité ϵ. Rahman *et al.* [146] évaluent d'abord empiriquement les MIA sur des classificateurs basés sur des DNN différentiellement privés. Ils constatent que les modèles différentiellement privés offrent une protection de la vie privée contre les attaquants puissants en n'offrant qu'une utilité de modèle médiocre.

Nous allons présenter deux méthodes pour entraîner des réseaux de neurones profonds de manière confidentielle. La première, appelée DP-SGD [5], consiste à ajouter du bruit aux gradients calculés par la méthode de descente de gradient stochastique (SGD) pour garantir la confidentialité. La seconde, appelée Model-agnostic private learning [132], consiste à entraîner de nombreux modèles (non confidentiels) sur des sous-ensembles de données sensibles, puis utilise une technique de confidentialité différentielle pour combiner les résultats.

- **Descente de gradient stochastique différentiellement privée (DP-SGD)**

La descente du gradient différentiellement privée est une méthode qui utilise la descente de gradient stochastique (SGD), une technique courante en apprentissage automatique, pour entraîner un modèle de manière différentiellement confidentielle. Ce processus est réalisé en ajoutant un bruit gaussien à chaque itération d'apprentissage, et en effectuant un clipping du gradient avant d'ajouter ce bruit. Le but de ce processus est de respecter les propriétés mathématiques de la confidentialité différentielle tout en évitant le sur-apprentissage.

La descente de gradient stochastique est une méthode d'optimisation utilisée pour entraîner un modèle de manière itérative. Elle s'exécute en plusieurs étapes : tout d'abord, un petit nombre d'exemples de données (appelé "mini-lot") est sélectionné à chaque itération à partir de l'ensemble de données d'apprentissage. Ensuite, l'optimiseur calcule l'erreur moyenne du modèle sur ces exemples, puis utilise cette erreur pour calculer un vecteur de gradient. Enfin, les paramètres du modèle sont mis à jour en soustrayant ce gradient multiplié par un petit nombre appelé taux d'apprentissage, qui détermine la vitesse à laquelle les paramètres sont mis à jour (voir la Section 2.2.1.1).

La confidentialité différentielle peut être appliquée aux modèles d'apprentissage automatique en rendant la descente stochastique du gradient différentiellement confidentielle. Cela consiste à ajouter un bruit Gaussien b_t à chaque itération d'apprentissage pour le gradient ∇t.

$$\theta_t \leftarrow \theta_t - \alpha(\nabla t + b_t).$$

Il est important de noter qu'avant d'ajouter le bruit, il faut effectuer un clipping du gradient pour éviter le surapprentissage et respecter les propriétés mathématiques de la confidentialité différentielle. Ce processus est décrit par la Figure 3.2.

- **Modèle d'apprentissage privé agnostique**

DP-SGD ajoute du bruit au gradient pendant la formation du modèle, ce qui peut nuire à la précision. L'apprentissage privé indépendant du modèle (agnostique) adopte une ap-

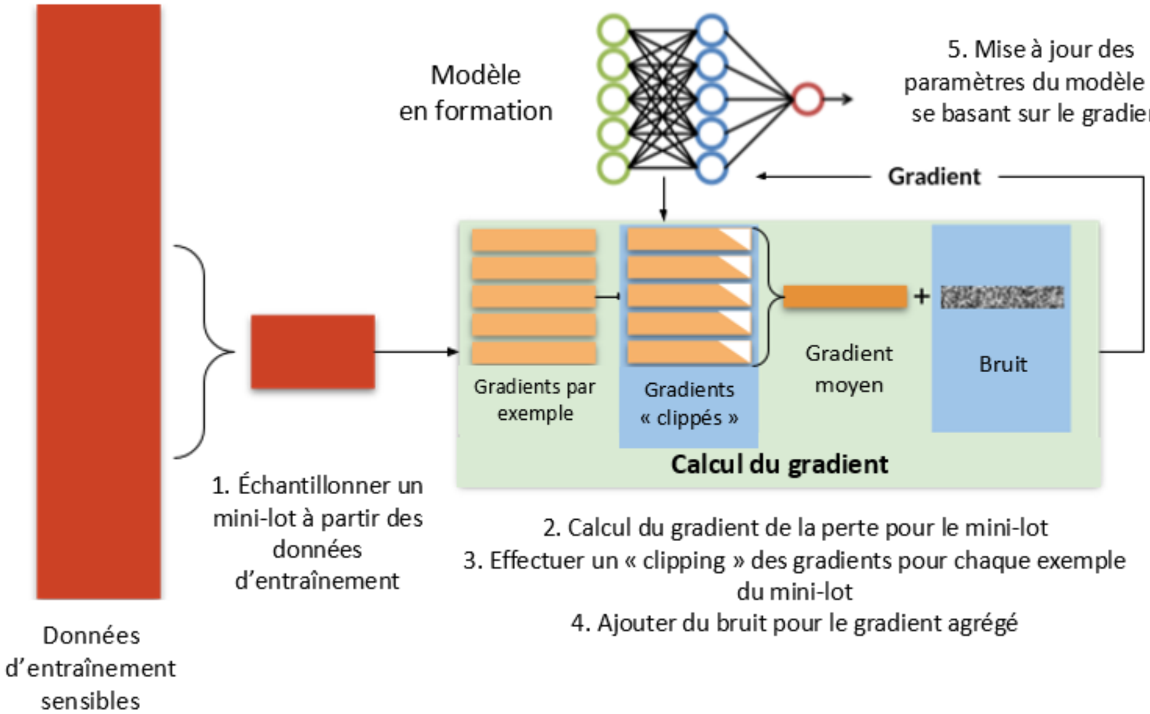

FIGURE 3.2 – Description de l'algorithme de la descente du gradient différentiellement privé.

proche différente et, dans certains cas, atteint une meilleure précision pour le même niveau de confidentialité par rapport à DP-SGD [134].

L'apprentissage privé indépendant du modèle utilise une méthode générique appelée "Sample and Aggregate" [132] pour ajouter de la confidentialité différentielle à un algorithme d'apprentissage automatique sans se soucier de son fonctionnement interne. En utilisant cette méthode pour un problème de classification multi-classes, les données d'apprentissage sont divisées en sous-ensembles disjoints et des modèles indépendants sont entraînés sur chacun d'eux. Pour prédire sur un exemple de test, on calcule un histogramme privé basé sur les prédictions des modèles indépendants, puis on sélectionne la prédiction la plus fréquente après avoir ajouté un peu de bruit (Laplacien/gaussien) pour maintenir la confidentialité.

L'approche de l'apprentissage privé indépendant du modèle a été utilisée dans deux contextes différents de l'apprentissage différentiellement privé, le PATE [132] et l'apprentissage privé agnostique de modèle (MAPL) [14] . Alors que MAPL se concentre sur la recherche théorique pour trouver un compromis entre confidentialité et précision pour une classe de tâches d'apprentissage, PATE se concentre sur l'application pratique. Ces deux approches partagent une idée commune : si les prédictions des modèles indépendants $\theta_1(x), \cdots, \theta_k(x)$ sont suffisamment cohérentes, alors le coût de confidentialité en termes de confidentialité différentielle est faible, ce qui permet de faire un grand nombre de requêtes de prédiction sans violer les contraintes de confidentialité.

Plus particulièrement, l'approche d'agrégation privée d'ensembles d'enseignants (PATE) est une technique de confidentialité différentielle pour l'apprentissage automatique qui permet de générer des prédictions privées en agrégeant les résultats d'un ensemble de modèles formés indépendamment sur des sous-ensembles de données disjoints. Au lieu de révéler les prédictions individuelles des modèles, l'approche d'histogramme privé est utilisée pour agréger les résultats. Le bruit aléatoire est ajouté aux votes pour empêcher que les résultats de l'agrégation ne reflètent ceux d'un seul modèle et protéger la confidentialité. Si les modèles sont suffisamment cohérents, le bruit n'altère pas la sortie de l'agrégation. Cette démarche est décrite par la Figure 3.3.

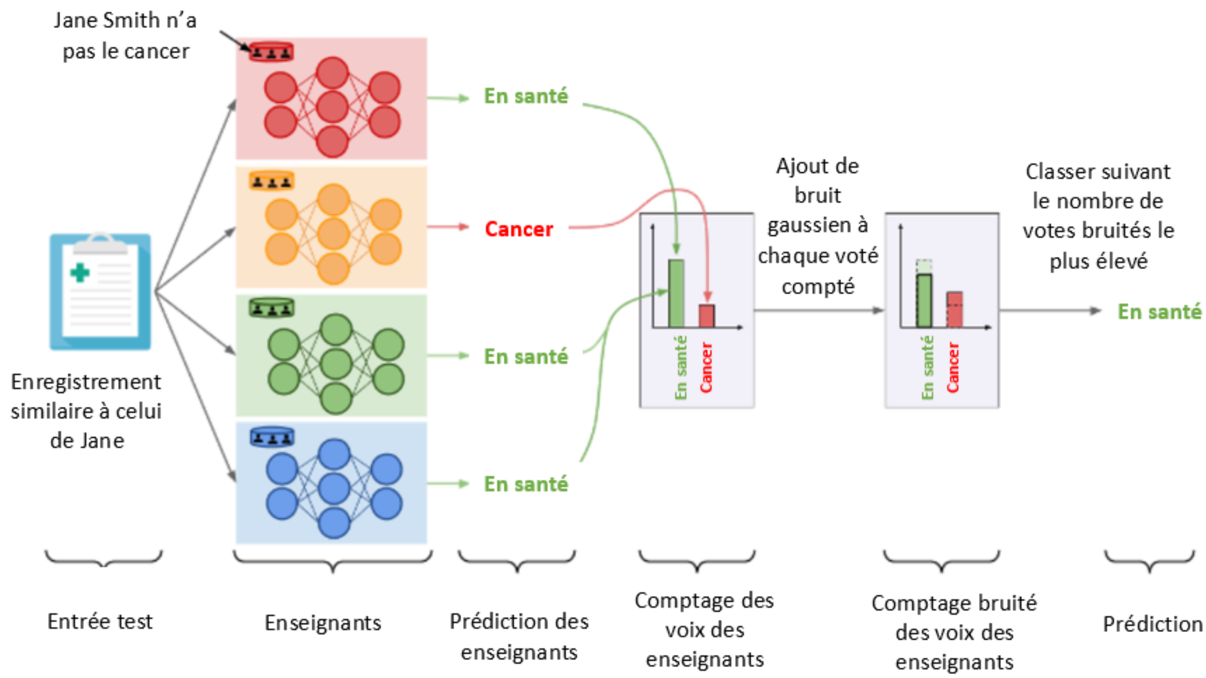

FIGURE 3.3 – Le cadre PATE. Cet algorithme forme de nombreux modèles non privés (les "enseignants") sur des sous-ensembles de données disjoints, puis demande aux modèles de "voter" sur la prédiction correcte en utilisant un mécanisme d'agrégation différentiellement privé.

En somme, la confidentialité différentielle est utilisée pour protéger les données d'entraînement en garantissant théoriquement leur confidentialité. Elle peut être utilisée pour réduire les risques liés à la fuite d'informations dans les modèles de classification et génératifs, indépendamment du contexte d'attaque. Bien que cette technique soit largement applicable et efficace, il existe des obstacles concrets au déploiement de l'apprentissage automatique différentiellement privé. L'obstacle principal est le compromis entre la **précision** du modèle et la préservation de la vie privée. Il est difficile de trouver un équilibre satisfaisant entre l'utilité et la confidentialité pour les tâches d'apprentissage complexes, souvent en raison de l'échantillonnage de données à partir de distributions à queues lourdes, c'est-à-dire qu'il y a généralement moins d'exemples de patients atteints d'une condition médicale donnée, limitant la capacité à apprendre de ces patients pour lesquels il y a très peu d'exemples. Il y a également souvent un compromis entre la précision du modèle et la force de la garantie de confidentialité différentielle utilisée pour former le modèle, mais il peut y avoir des cas où la confidentialité et l'exactitude sont complémentaires.

Un autre défi est la **surcharge de calcul**, qui peut rendre plus difficile l'utilisation des optimisations de l'apprentissage automatique pour tirer parti des accélérateurs matériels comme les GPU. Par exemple, le calcul des gradients plutôt que des gradients moyens dans DP-SGD, ou l'utilisation de plusieurs modèles (enseignants) plutôt qu'un seul dans PATE, peut augmenter les coûts de calcul lors de la formation des modèles.

Le choix approprié d'un **budget de confidentialité** est un autre défi dans l'utilisation de l'apprentissage automatique en situation de confidentialité. Plus le budget est petit, plus les garanties de confidentialité sont élevées, mais il peut être difficile de déterminer ce qui est suffisamment petit pour un budget donné. Il peut s'avérer particulièrement problématique pour les applications de confidentialité différentielle qui nécessitent souvent des garanties théoriques limitées pour former des modèles suffisamment précis pour justifier un déploiement. Il est recommandé aux praticiens d'évaluer les garanties de confidentialité de leurs algorithmes en utilisant des attaques spécifiques pour mieux comprendre les garanties théoriques obtenues. Ces attaques ne doivent pas être considérées comme des substituts à l'analyse théorique, mais plutôt comme un moyen d'interpréter les garanties théoriques obtenues.

3.3 Attaques par infection

Les attaques par infection sabotent la qualité des décisions et permettent aux attaquants d'exercer un contrôle des systèmes d'apprentissage automatique de façon dissimulée. Les attaquants contaminent les données utilisées pour l'entraînement, exploitent des déclencheurs cachés dans les comportements des modèles d'apprentissage ou distribuent des modèles d'apprentissage malveillants via des attaques par empoisonnement (poisoning attacks), des portes dérobées (backdooring attacks) ou des chevaux de Troie (trojaning attacks) [188].

Le but de ces attaques est de rendre le modèle vulnérable aux erreurs de prédiction, en fonction des besoins spécifiques de l'attaquant. Ces attaques peuvent être utilisées pour des buts malveillants tels que la dissémination d'informations fausses ou la prise de décisions erronées dans des systèmes critiques. Il est important de noter que les attaques d'empoisonnement de données peuvent être difficiles à détecter et à prévenir car elles peuvent être dissimulées dans des données qui semblent légitimes. De plus, les attaques par empoisonnement ne concernent pas seulement la sécurité des données, mais également tous les autres segments (par exemple, les algorithmes de formation et la procédure de modélisation) dans la phase de formation.

En effet, avec le déploiement généralisé des services basés sur les données, la demande de volumes de données continue de croître. À l'heure actuelle, de nombreuses applications manquent d'une supervision humaine fiable dans le processus de collecte de données, ce qui fait que les données collectées contiennent des données de mauvaise qualité ou même des données malveillantes (empoisonnées) [193].

3.3.1 Objectifs de l'attaquant

L'empoisonnement est d'abord proposé par Barreno *et al.* [12]. Depuis lors, les attaques par empoisonnement ont reçu une attention généralisée et sont devenues la principale menace pour la sécurité dans la phase d'entraînement de l'apprentissage automatique. Du point de vue des objectifs de l'attaquant, les attaques d'empoisonnement peuvent être classées en deux classes : les attaques d'empoisonnement non ciblées et les attaques d'empoisonnement ciblées. L'objectif des attaques d'empoisonnement non ciblées est d'entraver la convergence du modèle cible et de conduire éventuellement à un déni de service. Par exemple, les adversaires peuvent inverser les étiquettes de certains échantillons dans l'ensemble de données d'apprentissage pour perturber les connaissances contenues dans les paires des échantillons de l'ensemble d'entraînement [19]. Dans les attaques d'empoisonnement ciblées, l'objectif de l'attaquant est de forcer le modèle cible à produire des prédictions anormales sur des entrées particulières [160]. Par exemple, un adversaire peut forcer le modèle cible à classer des paquets malveillants comme un trafic normal dans la détection d'intrusion [214]. De plus, il existe un type avancé d'attaque par empoisonnement ciblé, qui est l'attaque par porte dérobée. Les attaques par porte dérobée franchissent une étape supplémentaire en fonction des attaques d'empoisonnement ciblées. Le but des attaques par porte dérobée est de rendre l'attaque ciblée inaperçue. Une porte dérobée peut être implantée dans le modèle cible à travers certains échantillons d'empoisonnement avec un motif fixe. Le modèle cible de porte dérobée ne fonctionnera anormalement que sur les entrées contenant le même motif [33, 64].

3.3.1.1 Empoisonnement non-ciblé

L'attaque par empoisonnement non ciblé est le type d'attaque le plus intuitif. Comme son nom l'indique, ces attaques n'ont pas de classe cible spécifique. Le but de l'adversaire est de dégrader les performances globales du modèle cible, telles que la précision de classification et la courbe caractéristique opérationnelle du récepteur (ROC).

Le défi consiste donc à utiliser les plus petites données d'empoisonnement possibles pour affecter au maximum la fonction d'apprentissage du modèle cible. Par exemple, dans [120], les auteurs ont proposé un cadre algorithmique général pour l'optimisation des instances d'attaques par empoisonnement sur l'apprentissage automatique conventionnel. Dans leurs recherches, une modification optimale de l'ensemble de données d'entraînement est optimisée par une perte d'optimisation convexe. De cette façon, ils pourraient empoisonner un modèle cible avec une petite perturbation des données.

Les caractéristiques des attaques par empoisonnement non-ciblées font que ce type de méthode d'attaque est facilement repérable. La victime peut facilement remarquer l'anomalie après la baisse des performances du modèle. Pour mieux tirer parti du modèle cible, mais aussi pour mieux dissimuler leurs traces, les adversaires ont développé un autre type d'attaque, les attaques d'empoisonnement ciblées.

3.3.1.2 Empoisonnement ciblé

Dans les attaques par empoisonnement ciblé, les adversaires ne se contentent plus de rendre le modèle victime incapable de fournir des services, mais se concentrent davantage sur la classe des prédictions erronées ou la classe d'entrée à l'origine des prédictions erronées. L'attaque par empoisonnement ciblé pourrait être formulée comme un problème multitâche.

Les attaquants font en sorte que le modèle cible fonctionne de manière anormale sur des échantillons spécifiques tout en maintenant sa fonctionnalité normale sur d'autres échantillons. Par exemple, pour une tâche de reconnaissance de chiffres, l'attaquant cherche à amener le modèle à mal classer le chiffre "7" tout en continuant à fonctionner correctement pour les autres chiffres.

Les attaques d'empoisonnement ciblées sont plus complexes à mettre en œuvre car elles nécessitent la création minutieuse d'échantillons d'empoisonnement qui peuvent amener le modèle à adopter des comportements spécifiques indésirables. Ces attaques ne sont considérées comme réussies que lorsque l'erreur se produit uniquement sur les échantillons ciblés par les attaquants, tout en fonctionnant correctement sur les autres échantillons. À la différence des attaques d'empoisonnement non ciblées qui peuvent être réalisées en introduisant du bruit aléatoire dans un ensemble de données d'entraînement, les attaques d'empoisonnement ciblées exigent une construction soignée d'échantillons d'empoisonnement, ce qui augmente les exigences techniques pour leur réalisation.

3.3.1.3 Empoisonnement par porte dérobée

Avec l'augmentation de la sophistication des cyberattaques, les adversaires cherchent des avantages à long terme de leurs actions. Cela inclut la dissimulation des traces d'attaque. Pour masquer ces traces, un moyen courant est de rendre les anomalies de performance du modèle cible moins évidentes en réduisant l'étendue des performances anormales. Les attaques qui visent cet objectif complexe sont connues sous le nom d'attaques d'empoisonnement par porte dérobée. Pour ce faire, les adversaires insèrent des modèles spécifiques, appelés déclencheurs (trigger), dans les données d'entraînement pour créer une porte dérobée. Seuls les échantillons contenant ces déclencheurs peuvent activer la porte dérobée cachée. Par exemple, Chen *et al.* [33] ont tenté de créer des portes dérobées dans des modèles d'apprentissage en utilisant des échantillons déclencheurs qui feront erronément classer ces échantillons comme une étiquette cible spécifiée par l'adversaire. Par exemple, ils ont réussi à implanter un type spécifique de lunettes comme déclencheur pour identifier à tort une personne spécifiée lorsqu'elle portait ces lunettes. Pendant ce temps, le modèle cible fonctionne normalement lorsque les entrées ne contiennent aucun déclencheur. Il est important de mentionner que les modèles d'apprentissage en profondeur sont particulièrement visés par les attaques d'empoisonnement ciblées. Cela est dû aux avancées importantes réalisées dans les méthodes d'apprentissage en profondeur ces dernières années, qui attirent davantage l'attention des adversaires.

3.3.2 Techniques d'attaques

Les modèles d'apprentissage automatique sont entraînés sur des données issues de sources externes, telles que des ensembles de données ouverts ou des capteurs. Ceci rend les attaques par empoisonnement possibles. Ces données peuvent être manipulées ou falsifiées par des attaquants pour introduire des biais dans le modèle d'apprentissage automatique ou pour causer des erreurs de classification.

Les attaques d'empoisonnement sont courantes dans les modèles d'apprentissage automatique car il est difficile de vérifier chaque échantillon individuellement. Par exemple, la base de données ImageNet [42] contient plus de 14 millions d'images avec une étiquette pour chaque image, mais cette base de données peut être affectée par des échantillons malveillants. La construction de ces échantillons d'empoisonnement est un enjeu crucial de la recherche dans la sécurité de l'apprentissage automatique. Il est donc important de mettre en place des mécanismes pour détecter ces échantillons malveillants pour éviter des résultats inattendus ou erronés. Dans cette partie du rapport, nous discuterons des techniques d'empoisonnement et nous les classerons en deux catégories en fonction de la cible manipulée : la manipulation d'étiquettes et la manipulation de données.

3.3.2.1 Manipulation d'étiquettes

La manipulation d'étiquettes est une méthode couramment utilisée pour empoisonner les modèles d'apprentissage automatique. La qualité des modèles d'apprentissage dépend essentiellement des paires d'échantillon-étiquette. Si ces étiquettes sont altérées, comme le montre la Figure 3.4 les performances du modèle en souffrent [19]. Un adversaire peut inverser des étiquettes aléatoirement pour causer une attaque d'empoisonnement. Barreno *et al.* [12] ont étudié l'utilisation de ces techniques pour tromper les systèmes de classification basés sur l'apprentissage automatique, en discutant des stratégies possibles pour placer des échantillons d'empoisonnement dans le processus de formation. Biggio *et al.* [19] ont ensuite examiné les effets de ce bruit causé par l'inversion d'étiquettes et les moyens de se protéger contre ces attaques dans les systèmes SVM.

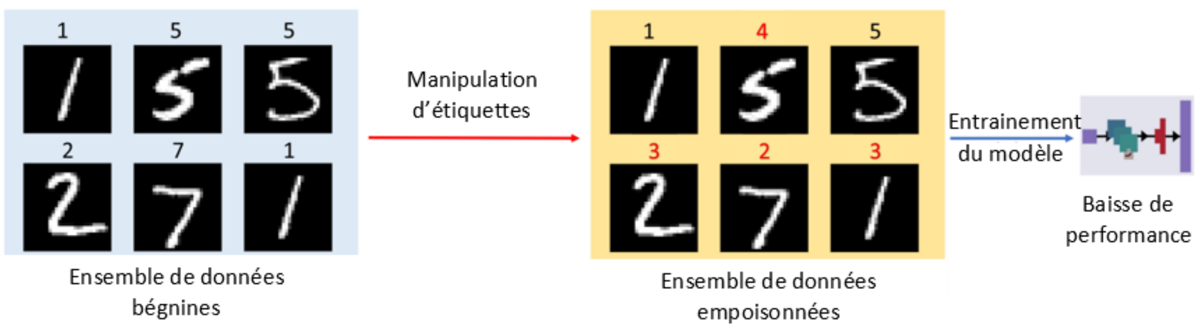

FIGURE 3.4 – Exemple de manipulation d'étiquettes.

Les algorithmes d'apprentissage en profondeur sont tout aussi vulnérables aux attaques d'empoisonnement en raison de la nature même de leur fonctionnement. Zhang *et al.* [210] ont montré que les modèles peuvent être formés avec des étiquettes aléatoires et les performances ne seront pas meilleures que le hasard. Cela montre que l'algorithme d'apprentissage en profondeur peut être forcé à apprendre des connaissances fausses et nous nous retrouvons face à un cas de surajustement. Les adversaires peuvent améliorer l'efficacité de l'attaque en sélectionnant les étiquettes ou les points de données les plus perturbateurs pour amplifier l'impact de l'empoisonnement. Cependant, cela nécessite des ressources informatiques et du temps importants. Il existe des méthodes pour trouver les paires échantillon-étiquette les plus optimales pour l'attaque. Par exemple, Biggio *et al.* [19] utilisent des probabilités non uniformes pour retourner les étiquettes des échantillons et évaluent comment cela affecte les performances du modèle. Dans [199], les auteurs introduisent la notion de budget de l'attaquant, ce dernier vise à trouver une combinaison de retournements d'étiquettes, sous un certain budget, afin que le modèle cible entraîné sur un tel ensemble de données ait une erreur de classification maximale. Dans la même optique, Zhao *et al.* [215] ont étudié une méthode de manipulation d'étiquettes en boîte noire. Ils ont formulé le problème d'attaque comme un problème d'optimisation à deux niveaux en supposant que l'objectif de l'adversaire est de maximiser le cosinus de l'angle entre le vecteur de poids appris de l'apprenant et le vecteur de poids du modèle objectif. Ensuite, ils développent un algorithme Projected Gradient Ascent (PGA) pour résoudre ce problème.

La manipulation d'étiquettes est une méthode simple pour empoisonner les modèles d'apprentissage automatique, qui consiste à modifier les étiquettes de certains points de données. Cependant, cette méthode a des inconvénients, tels que la limitation de la réalisation d'objectifs sophistiqués et la naïveté de l'empoisonnement, qui peut facilement être repéré par la victime. Les objectifs antagonistes sophistiqués nécessitent souvent de la furtivité et des connaissances avancées sur l'empoisonnement, ce qui est difficile à obtenir en modifiant uniquement les étiquettes. En conséquence, les recherches sur les attaques basées sur la manipulation d'étiquettes sont de plus en plus limitées.

3.3.2.2 Manipulation de données

Il est possible de réaliser des attaques plus sophistiquées en modifiant les données des échantillons plutôt que leurs étiquettes. Selon la Figure 3.5, un adversaire peut empoisonner le modèle cible en insérant un certain motif (boîte blanche dans le coin supérieur gauche de l'échantillon) dans les échantillons d'apprentissage. Par exemple, dans la littérature [131], l'adversaire essaie de perturber la méthode d'apprentissage SpamBayes [121] en incluant des mots plus susceptibles de se produire dans un e-mail légitime dans les e-mails d'attaque.

Il est décrit dans la littérature [64] que les adversaires peuvent intégrer un motif de pixels fixes dans les échantillons d'empoisonnement pour créer une porte dérobée dans le modèle d'apprentissage en profondeur cible. Toutefois, cette technique de manipulation manuelle des données est de moins en moins utilisée car la prise de conscience de la sécurité augmente. Les adversaires ont tendance à utiliser deux types de techniques pour empoisonner les données : la manipulation de données basée sur l'optimisation et la manipulation de données basée sur l'apprentissage.

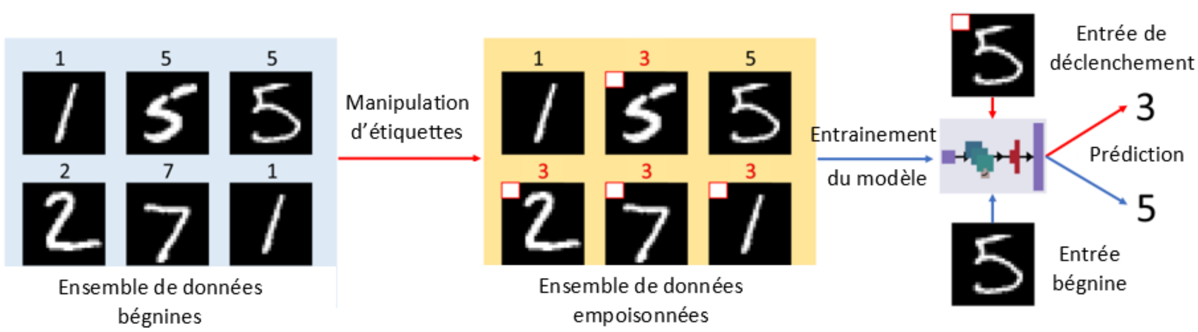

FIGURE 3.5 – Exemple de manipulation de données.

• Manipulation de données basée sur l'optimisation

La manipulation de données basée sur l'optimisation, comme son nom l'indique, est une technique pour créer des échantillons d'empoisonnement en optimisant une fonction objectif. Le but est de produire des échantillons qui ont un impact négatif sur l'apprentissage du modèle cible. Pour cela, un adversaire commence par concevoir une fonction objectif, puis utilise des méthodes d'optimisation pour atteindre cet objectif. Les deux éléments clés de cette technique sont la conception de la fonction objectif et la méthode d'optimisation utilisée.

La création d'échantillons d'empoisonnement est souvent réalisée en résolvant un problème d'optimisation à deux niveaux qui comporte deux fonctions objectifs. La formulation générale de ce type de problème d'optimisation peut être résumée de cette manière :

$$D_p^* \in \underset{D_p}{\operatorname{argmax}} \quad F(D_p, w^*) = L_1(D_{val}, w^*), \tag{3.4}$$

$$s.t. \qquad w^* \in \underset{w}{\operatorname{argmin}} \quad L(D_{train} \cup D_p, w). \tag{3.5}$$

Comme nous l'avons défini dans le chapitre 2, D_{train} fait référence à l'ensemble de données d'apprentissage d'origine qui a servi à l'entraînement du modèle et D_{val} désigne l'ensemble de données de validation. Nous utiliserons D_p pour décrire l'ensemble des échantillons empoisonnés. L'objectif de l'optimisation externe (3.4) est de fabriquer les échantillons empoisonnés D_p les plus efficaces, qui maximisent la fonction de perte empirique $L_1(D_{val}, w^*)$ sur le jeu de données de validation D_{val} et le modèle empoisonné w^*. L'optimisation interne ((3.5)) vise à mettre à jour le paramètre w^* du modèle victime sur le jeu de données empoisonné $D_{train} \cup D_p$. Il faut souligner que w^* dépend implicitement des échantillons empoisonnés D_p, donc F est défini en fonction de deux variables D_val et w^*.

$$\begin{aligned} & \underset{x \in X, y \in Y}{\operatorname{argmin}} \quad F(x, y) \\ s.t. \qquad & \underset{z \in Y}{\operatorname{argmin}} \quad G(x, y) \end{aligned}$$

Dans cette formulation, F est la fonction objectif du niveau supérieur, G est la fonction objectif du niveau inférieur, x est le vecteur de décision du niveau supérieur et y est le vecteur de décision du niveau inférieur. La fonction objectif du niveau inférieur peut être utilisée pour définir une contrainte sur la fonction du niveau supérieur. Il est important de noter que l'opération d'argmin peut être remplacée par argmax, car maximiser une fonction objectif équivaut à minimiser son opposé.

Dans [19], la fonction objectif pour entraîner un modèle SVM est conçue pour maximiser la perte de charnière (hinge loss) [2] causée par des données de validation (échantillon non-empoisonné) sur un SVM entraîné avec des données empoisonnées. Cette fonction objectif est optimisée en utilisant la technique de montée de gradient. Cette approche permet à un adversaire de créer des données empoisonnées qui diminuent considérablement la précision de classification du SVM. Nous retrouvons également des attaques d'empoisonnement basées sur l'optimisation à deux niveaux qui ont été utilisées pour cibler les modèles de régression linéaire [84] et de régression logistique [41].

Les attaques d'empoisonnement basées sur l'optimisation ont l'avantage de permettre un contrôle précis des échantillons d'empoisonnement générés et de pouvoir atteindre des

2. La perte de charnière (hinge loss en anglais) est une fonction de coût utilisée dans les algorithmes d'apprentissage supervisé pour les machines à vecteurs de support (SVM). Elle mesure l'erreur commise par un modèle en fonction de la différence entre la prédiction du modèle (f(x)) et la valeur réelle (y) pour un exemple donné (x), où la formule mathématique de la perte de charnière est donnée par : $max(0, 1 - y * f(x))$.

objectifs contradictoires avec une conception raisonnable. Cependant, il existe également des inconvénients, tels que la difficulté à générer plusieurs échantillons d'empoisonnement, la tendance à être bloqué dans des optima locaux et la difficulté à cacher les traces de manipulation des échantillons.

- **Manipulation de données basée sur l'entraînement**

En manipulant des données d'entraînement, un adversaire utilise un modèle auxiliaire pour créer des échantillons d'empoisonnement. Il existe deux types de modèles auxiliaires : l'isomorphisme et la génération. Il est difficile pour les adversaires d'accéder au modèle cible sans restriction dans une situation réelle (n'ayant accès ni à la sortie du modèle, ni à son architecture ni à ses paramètres), donc pour générer des échantillons de contamination plus efficaces, un adversaire peut entraîner un modèle auxiliaire isomorphe pour imiter des modèles cibles potentiels puis construire des échantillons de contamination en fonction de ses performances. Par exemple, Zhu *et al.* [218] propose un cadre formel où l'adversaire entraîne des modèles de substitution sur un ensemble de données auxiliaires et optimise une fonction objectif qui force les données empoisonnées à se regrouper dans un espace de caractéristiques qui enferme la cible dans sa coque convexe (polytope). Cela veut dire que le modèle formé à partir de ces données empoisonnées classera de manière erronée la donnée cible dans une classe spécifique.

De nombreux travaux sur l'empoisonnement utilisent les modèles génératifs pour améliorer l'efficacité de la construction des échantillons d'empoisonnement. Le défi de cette méthode est de savoir comment générer des échantillons de contamination qui sont proches des expériences humaines dans les domaines d'application. Yang *et al.* [206] ont proposé une méthode d'utilisation d'un GAN pour accélérer la création d'échantillons d'empoisonnement. Ils ont utilisé un auto-encodeur comme générateur et considèrent le modèle cible comme discriminateur. Le générateur a été formé pour générer des échantillons d'empoisonnement à travers une fonction de récompense basée sur la perte. Comme le montre la Figure 3.6(a), le modèle est composé de deux composants incluant le générateur g et le classificateur victime f. À l'itération t de l'attaque, les étapes sont comme suit : (1) le générateur produit les poisons x_p^t ; (2) l'adversaire insère les poisons dans les données d'entraînement et met à jour le classificateur de $w^{(t-1)}$ à $w^{(t)}$; (3) l'adversaire évalue le modèle empoisonné w^t sur les données de validation D_{val} et récupère suffisamment d'information pour orienter la direction de la mise à jour du générateur ; (4) l'adversaire met à jour le générateur et passe à la prochaine itération. Cette procédure itérative peut être formulée comme suit :

$$\begin{aligned} g \quad &= \quad \underset{g}{\text{argmax}} \sum_{(x,y)\in D_{val}} \mathcal{L}(f_{w^*}(x), y), \\ s.t. \qquad & w^* = \underset{w}{\text{argmin}} \sum_{(x,y)\in D_p} \mathcal{L}(f_\theta(g(x)), y). \end{aligned}$$

Un autre modèle, illustré par la Figure 3.6(b), décrit le modèle pGAN [127]. Ce dernier contient trois composants : le générateur G, le discriminateur D et le classificateur cible C (c'est-à-dire le modèle visé par l'attaque). Le générateur est utilisé pour créer des échantillons qui maximisent l'erreur du modèle cible, mais minimisent la capacité du discriminateur à les

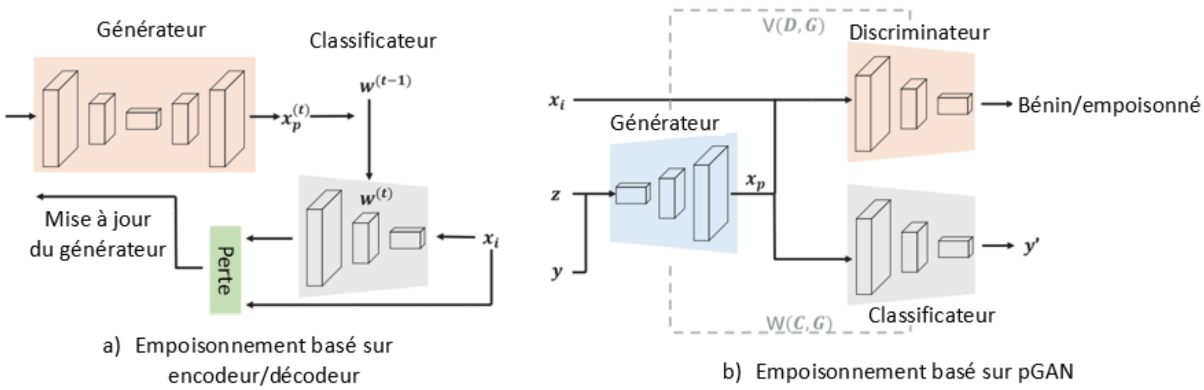

FIGURE 3.6 – Méthodes d'empoisonnement basées sur les GANs.

identifier comme tels. Le discriminateur a pour objectif de différencier les échantillons bénins des empoisonnés. Quant au modèle cible, ce dernier tente de minimiser certaines fonctions de perte basées sur un ensemble de données d'apprentissage contenant une fraction d'échantillons d'empoisonnement. Le générateur joue un jeu simultané contre le discriminateur et le classificateur, permettant un équilibre entre la force d'attaque et la dissimulation de celle-ci. L'utilisation d'un modèle génératif permet de créer de nombreux points d'empoisonnement pour attaquer des algorithmes d'apprentissage complexes plus efficacement que les modèles basés sur l'optimisation [19, 153]. Ces modèles permettent également d'ajuster le compromis entre efficacité et dissimulation de l'attaque via le jeu complexe entre le classificateur et le discriminateur. Ces méthodes peuvent donc être utilisées pour attaquer des classificateurs d'apprentissage automatique de différents niveaux de risque.

3.3.3 Techniques de défense contre les attaques d'empoisonnement

L'augmentation constante des attaques d'empoisonnement de données est particulièrement alarmante. Il devient de plus en plus ardu pour les méthodes de défense existantes de lutter contre ces attaques, soit en raison de leur puissance croissante, soit en raison de leur impact négatif sur les performances. Pour évaluer les différentes techniques de défense contre les attaques d'empoisonnement, il est courant de les diviser en défense passive et défense active. La défense passive vise à nettoyer les données d'entraînement pour éliminer la cause initiale de l'empoisonnement, tandis que la défense active consiste à prendre des mesures préventives pour protéger contre les menaces potentielles. Il existe différentes méthodes de défense passive, telles que les filtres anti-poison et les méthodes de désinfection de modèle, et différentes méthodes de défense active, telles que les stratégies de prétraitement des données et les méthodes de formation robustes. Il est important de noter que les défenses passives ne peuvent pas protéger contre tous les types d'empoisonnement ; il est donc préférable pour un défenseur expérimenté de prendre des précautions pour éviter les dommages plutôt que de simplement réagir après coup.

Pour évaluer les performances des techniques de défense, on considère les critères d'évaluation suivants :

- L'efficacité de chaque stratégie de défense est évaluée à travers le critère de "force de la défense". Il est généralement admis que les défenses certifiées sont considérées comme plus fiables que les défenses basées sur l'expérience.

- Le critère "Complexité de la défense (CPLX)" permet d'évaluer la difficulté d'implémentation des stratégies de défense considérées, allant de défenses faciles à déployer jusqu'à celles nécessitant des modifications complexes de l'architecture globale.
- Le critère "Robustesse de la défense" permet d'évaluer la résistance des méthodes de défense face à des situations d'empoisonnement excessif. Les défenses qui s'avèrent fragiles ou qui aggravent la situation sont considérées comme "non robustes" contrairement à celles qui sont considérées comme "robustes".
- Le critère "Effet secondaire sur la précision (Acc.)" permet d'évaluer l'impact des méthodes de défense sur la précision du modèle. Si l'application de la méthode de défense augmente la précision du modèle sans qu'il y ait d'attaque, elle sera mieux évaluée qu'une technique qui diminue la précision du modèle.

3.3.3.1 Défense passive

Les défenses passives se concentrent sur la détection des attaques d'empoisonnement et sur la réparation des dommages causés. Ces méthodes assurent que les poisons sont filtrés avant l'entraînement, garantissant ainsi l'intégrité du modèle. Cependant, il existe des situations où les adversaires contaminent directement les poids du modèle, comme l'empoisonnement du modèle, ce qui rend inefficace la désinfection des données. Dans ces cas, les défenseurs doivent se concentrer sur la détection et la désinfection des modèles, ce qui est beaucoup plus complexe et difficile à réaliser.

- **Défense basée sur un filtre anti-poison**. La plupart des ensembles de données d'entraînement réalistes sont inévitablement imparfaits. Ils peuvent contenir des données bruitées qui masquent la relation entre les caractéristiques d'une instance et sa classe (étiquettes) causée par des événements anormaux ou des attaques malveillantes. Lorsque les données de formation sont polluées par des attaques d'empoisonnement, il est courant d'utiliser des approches de filtrage pour améliorer la qualité des données de formation, similaires à la détection de valeurs aberrantes ou d'anomalies [141, 176]. Les données empoisonnées peuvent généralement être identifiées et être ré-étiquetées ou simplement supprimées avant le début du processus d'entraînement.
- **Défense basée sur la désinfection des modèles**. Contrairement aux défenses basées sur le filtre anti-poison qui travaillent sur l'ensemble de données d'entraînement, il existe des situations, comme l'empoisonnement du modèle, où les défenseurs n'ont pas accès aux données d'entraînement originales. L'empoisonnement du modèle peut également altérer la fonction originale des neurones du réseau. Dans ces cas, les défenseurs doivent se concentrer sur la détection et la désinfection du modèle, une tâche beaucoup plus complexe et difficile à réaliser. Par exemple, dans [158], Schuster *et al.* ont proposé une méthode de défense qui agit directement sur les paramètres des modèles. Cette technique utilise des compétences d'élagage fin pour éliminer les neurones inactifs lorsque le modèle est en cours d'utilisation, affaiblissant ainsi l'impact des attaques d'empoisonnement. Ensuite, ils améliorent le modèle en utilisant une petite quantité

de données propres, ce qui permet de compenser la perte de performance causée par l'empoisonnement et l'élagage.

3.3.3.2 Défense active

Il est important de prendre des mesures préventives pour protéger contre les menaces potentielles en utilisant des défenses actives contre des attaques d'empoisonnement inconnues. Nous examinerons les défenses actives sous deux angles différents : celui des données et celui des modèles. Selon [193], les méthodes de prétraitement des données et les méthodes d'entraînement robustes peuvent améliorer la robustesse et la résilience de la phase d'entraînement face aux attaques d'empoisonnement.

- **Prétraitement des données**. Le prétraitement des données est un processus important pour préparer les données d'entraînement pour les modèles d'apprentissage. Il peut inclure des étapes de nettoyage, d'amélioration et de transformation des données pour renforcer la robustesse du modèle. Certaines études récentes [21] ont montré que l'augmentation des données peut être plus efficace pour augmenter la robustesse des modèles que de les réduire. Les méthodes telles que Mixup et CutMix qui consistent à combiner des données d'entraînement de manière aléatoire, ont été montrées pour réduire les risques d'attaques d'empoisonnement sans compromettre les performances des modèles. Ces méthodes aident à prévenir la mémorisation des étiquettes empoisonnées et améliorent la généralisation. Ces approches de prétraitement des données sont efficaces pour se protéger contre l'empoisonnement des données sans sacrifier significativement les performances de validation.
- **Formation de modèle robuste**. La formation robuste se concentre sur la résistance des systèmes face aux anomalies et aux entrées erronées. Elle vise à renforcer la phase de formation pour rendre les attaques d'empoisonnement plus difficiles à réaliser. Roh *et al.* [151] ont proposé une méthode appelée FR-GAN pour améliorer la robustesse des modèles d'apprentissage en utilisant des réseaux antagonistes génératifs (GAN). Cette méthode implique la collecte de données de validation propres pour entraîner un discriminateur supplémentaire. Le but du discriminateur est de vérifier que les prédictions du classificateur cible sont cohérentes avec celles du discriminateur, ce qui renforce la robustesse du classificateur. Une autre technique pour améliorer la robustesse des modèles est d'utiliser des méthodes d'ensemble. Jia *et al.* [90] ont introduit la méthode Bootstrap Aggregating (bagging) qui est une méthode d'apprentissage d'ensemble bien connue qui consiste à former plusieurs modèles de base sur des sous-ensembles aléatoires d'un ensemble de données d'apprentissage. Ensuite, utiliser un vote majoritaire pour prédire les étiquettes des exemples de test. Cette technique de défense est décrite par la Figure 3.7. Elle permet une robustesse certifiée contre les attaques d'empoisonnement des données en raison de l'aléatoire de l'échantillonnage des sous-ensembles et de l'algorithme d'apprentissage de base.

3.3.4 Empoisonnement dans l'apprentissage par renforcement

Les attaques d'empoisonnement sont également une préoccupation pour l'apprentissage par renforcement (AR) [16]. Nous allons présenter une classification des attaques d'empoi-

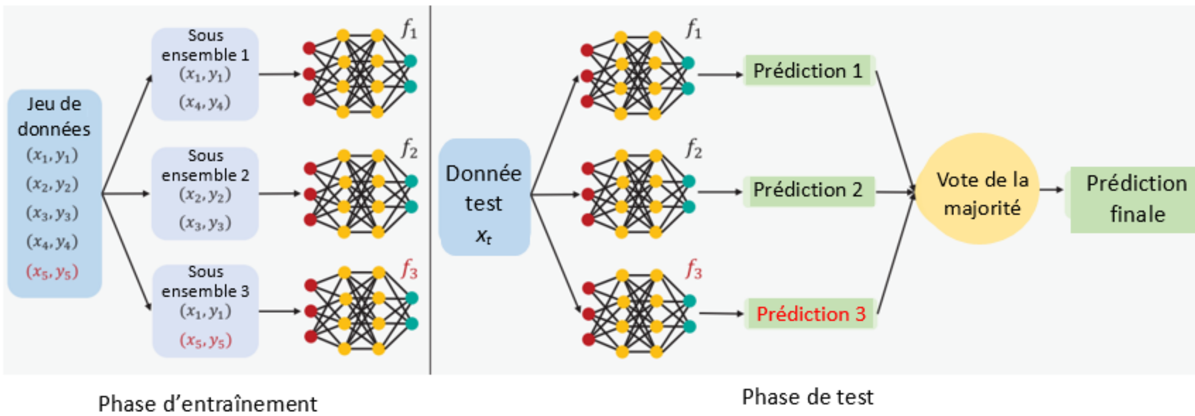

FIGURE 3.7 – Technique de défense avec un modèle de Bootstrap aggregating (Bagging).

sonnement pour l'AR sous différentes configurations, à savoir l'instant de l'attaque (phase d'entraînement Vs phase de test), les connaissances de l'attaquant (boîte blanche Vs boîte noire) et le mode d'apprentissage (en ligne Vs hors ligne) et résumer les défis et les moyens de les contrer. Pour rappel, l'apprentissage par renforcement est une méthode qui consiste à maximiser un signal de récompense numérique en interagissant avec l'environnement et en mappant les situations aux actions. Contrairement à l'apprentissage automatique classique, il n'y a pas d'instructions précises sur les actions à entreprendre, mais plutôt une découverte par essais et erreurs. Les données d'entraînement sont les trajectoires de l'apprenant dans l'environnement, généralement décrites sous forme de tuples (état, action, récompense).

3.3.4.1 Attaques durant la phase de test Vs phase d'entraînement du modèle AR

Les travaux de la littérature ont beaucoup étudié les attaques d'empoisonnement de l'apprentissage par renforcement lors de la phase de test, où la politique de l'agent π est déjà formée et fixe, et où l'attaquant manipule l'état perçu s_t à $s_t^\dagger$ pour causer une action indésirable [99, 78, 109]. Cependant, ces attaques lors de la phase de test peuvent avoir des conséquences importantes sur les performances d'une politique déployée et mise à jour par l'agent. Pour les agents en cours d'apprentissage, la surface d'attaque inclut également la phase d'entraînement, c'est pourquoi il est important d'étudier également les attaques durant cette phase.

Contrairement aux attaques durant la phase de test, qui perturbent les performances de la politique bien formée, les attaques durant l'entraînement visent à appliquer une politique cible arbitraire sur l'agent RL. Les travaux existants empoisonnent la plupart du temps la politique de l'agent en manipulant les récompenses au moment de la formation. Dans les travaux de Ma *et al.* [113], l'attaquant empoisonne les récompenses dans l'ensemble de données d'entraînement et Zhang *et al.* [213] étudient les attaques adaptatives d'empoisonnement par récompense sur un agent qui utilise l'algorithme Q-learning pour mettre à jour sa politique. Les travaux proposés par Rakhsha *et al.* [147] décrivent une attaque qui empoisonne les dynamiques de transition environnementale. La victime cible est un agent qui utilise un algorithme de minimisation de regrets et dont l'objectif est de maximiser les récompenses moyennes non actualisées à horizon infini. Ainsi, cette attaque d'empoisonnement de l'environnement convient aux tâches cycliques qui n'ont pas d'état de terminaison.

Les attaques lors de la phase d'entraînement visent à faire adopter une politique cible arbitraire à l'agent d'apprentissage par renforcement. Les méthodes courantes d'empoisonnement consistent à manipuler les récompenses lors de l'entraînement. Des recherches comme celles de Ma *et al.* [113] et Zhang *et al.* [213] se concentrent sur l'empoisonnement des récompenses dans les données d'entraînement. Rakhsha *et al.* [147] décrivent une attaque qui empoisonne les dynamiques de transition de l'environnement pour les tâches cycliques qui n'ont pas d'état de terminaison. Ces attaques ont pour but d'altérer les performances de la politique de l'agent en la faisant adopter une politique cible arbitraire.

3.3.4.2 Attaques boîte blanche Vs boîte noire

Les recherches sur l'empoisonnement des modèles d'apprentissage par renforcement peuvent également être classées en fonction de la connaissance que l'attaquant a sur le modèle cible. Lors du déploiement de la politique de l'agent victime, l'attaquant perturbe les états perçus de la victime afin de modifier sa décision sur la sélection des actions. Pour les attaques lors de la phase de test, les travaux existants ont étudié à la fois les attaques en boîte blanche [99, 95, 187], où l'attaquant a une connaissance complète de l'algorithme d'apprentissage et du modèle de politique de l'agent, et les attaques en boîte noire [78, 82], où l'attaquant n'a pas accès aux poids du modèle. Les attaques en boîte noire se basent sur le concept de transférabilité des exemples contradictoires [135], qui permet de perturber d'autres modèles avec des entrées malveillantes conçues pour un modèle spécifique. Cependant, il n'est pas possible de réaliser des attaques en boîte noire lors de la phase d'entraînement.

3.3.4.3 Attaques en ligne Vs hors ligne

Il existe principalement deux types d'attaques d'empoisonnement lors de la phase d'entraînement de l'apprentissage par renforcement : les attaques hors ligne et les attaques en ligne. Les attaques hors ligne impliquent que l'attaquant a une connaissance complète de l'ensemble de données d'entraînement et manipule les récompenses avant de fournir les données empoisonnées à l'agent RL pour la construction de sa politique [147, 113]. Les attaques en ligne signifient que l'attaquant manipule séquentiellement le signal de retour lors de l'interaction avec l'agent RL [147, 213]. Ces attaques nécessitent l'accès aux transitions de l'agent et sont adaptatives aux progrès d'apprentissage de l'agent, mais sont limitées aux paramètres en boîte blanche.

En raison de l'importance de l'apprentissage par renforcement dans des domaines tels que la robotique, les jeux vidéo et les systèmes de recommandation, les attaques d'empoisonnement intentionnelles et malveillantes pourraient causer des dégâts considérables. La recherche sur les attaques d'empoisonnement contre l'apprentissage par renforcement est encore en développement. Il y a plusieurs pistes prometteuses pour les recherches futures, telles que l'extension des modèles d'attaque pour inclure des attaques simultanées sur les récompenses et les transitions, ou l'élargissement des objectifs pour cibler une victime spécifique. Il y a également peu de recherches sur la défense contre les menaces pendant la phase d'entraînement, ce qui mérite une attention accrue.

3.4 Attaques par manipulation

Les attaques par manipulation visent à altérer les données d'entrée d'un modèle déjà entraîné pour tromper ses prédictions. Les attaquants injectent des données d'entrée modifiées pour obtenir une sortie différente de celle attendue. Le but est de détourner le comportement de l'application à leur avantage, comme accepter une transaction frauduleuse ou classer une image malveillante comme étant sûre. Cela est différent des attaques par infection qui visent à injecter des données malveillantes dans le jeu de données d'entraînement pour rendre le modèle vulnérable aux attaques futures.

Les attaques par manipulation, également appelées "exemples contradictoires", sont fréquemment étudiées dans les recherches sur les attaques d'apprentissage automatique. L'attaquant crée une entrée qui génère une sortie différente de celle prévue par les concepteurs du système. Ces attaques peuvent prendre différentes formes, telles que celles qui visent à tromper les classifications de panneaux d'arrêt, les erreurs d'identification de spams ou les traitements linguistiques défectueux. Nous allons aborder dans cette section les types d'attaques les plus couramment utilisés, tels que les attaques par évasion, par reprogrammation et par déni de service.

3.4.1 Attaques par évasion (evasion attacks)

Une attaque par évasion se produit lorsque le réseau est alimenté par un « exemple contradictoire » - une entrée soigneusement perturbée qui est quasiment identique à sa copie non altérée pour un humain - mais qui déstabilise complètement le système. De façon imagée, on peut dire qu'il s'agit de créer l'équivalent d'une illusion d'optique pour le système en introduisant un « bruit » judicieusement calculé, et cela quel que soit le type d'entrée pris par le système d'apprentissage automatique (image, texte, son, etc.).

Historiquement, les exemples contradictoires (ou adversaires pour "adversarial examples") ont été introduits par Ian Goodfellow en 2014 [61]. Un exemple contradictoire est une entrée d'un modèle d'apprentissage automatique qui est intentionnellement conçue par un attaquant pour tromper le modèle en produisant une sortie incorrecte. Par exemple, nous pourrions commencer avec une image d'un panda et ajouter une petite perturbation qui a été calculée pour que l'image soit reconnue comme un gibbon avec une haute confiance [61] :

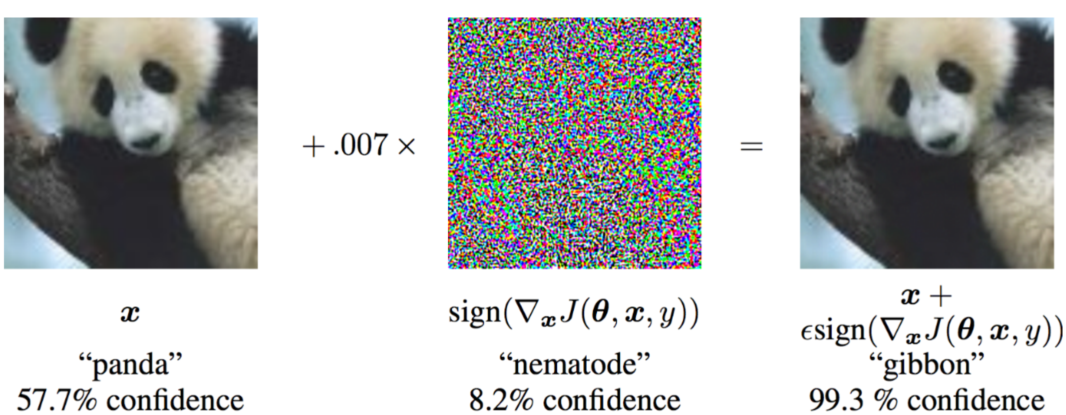

FIGURE 3.8 – Différentes catégories d'attaques contre l'apprentissage automatique [2].

Examinons deux techniques de défense, l'entraînement contradictoire et la distillation défensive, comme exemples de la façon dont un défenseur peut tenter de rendre un modèle d'apprentissage automatique plus robuste et d'atténuer les exemples contradictoires.

Entraînement contradictoire cherche à améliorer la généralisation d'un modèle lorsqu'il est présenté avec des exemples contradictoires au moment du test en générant de manière proactive des exemples contradictoires dans le cadre de la procédure de formation. Cette idée a été introduite pour la première fois par Szegedy *et al.* [182] mais n'était pas encore pratique en raison du coût de calcul élevé de la génération d'exemples contradictoires. Goodfellow *et al.* ont montré comment générer des exemples contradictoires à peu de frais avec la méthode du signe de gradient rapide et l'ont rendue efficace sur le plan informatique pour générer de grands lots d'exemples contradictoires pendant le processus de formation [61]. Le modèle est ensuite entraîné pour attribuer la même étiquette à l'exemple contradictoire qu'à l'exemple original - par exemple, nous pourrions prendre une photo d'un chat et le perturber de manière contradictoire pour tromper le modèle en lui faisant croire qu'il s'agit d'un vautour, puis dire au modèle qu'il doit apprendre que cette image est toujours un chat.

La distillation défensive lisse la surface de décision du modèle dans les directions adverses exploitées par l'adversaire. La distillation est une procédure de formation dans laquelle un modèle est formé pour prédire les probabilités générées par un autre modèle qui a été formé plus tôt. La distillation a été introduite pour la première fois par Hinton *et al.* dans [71], où l'objectif était qu'un petit modèle imite un grand modèle coûteux en calculs. La distillation défensive a un objectif différent de simplement rendre les réponses du modèle final plus fluides, donc cela fonctionne même si les deux modèles ont la même taille. Il peut sembler contre-intuitif d'entraîner un modèle pour prédire la sortie d'un autre modèle qui a la même architecture. La raison pour laquelle cela fonctionne est que le premier modèle est formé avec des étiquettes "dures" (100% de probabilité qu'une image soit un chien plutôt qu'un chat), puis fournit des étiquettes "douces" (95% de probabilité qu'une image soit un chien plutôt qu'un chat) utilisées pour former le deuxième modèle.

Les exemples contradictoires se fondent sur les difficultés des modèles utilisant l'apprentissage automatique à « généraliser », c'est-à-dire à modéliser correctement la tâche demandée sur la base d'un ensemble d'apprentissage limité.

Au cours des dernières années, de nombreuses attaques par évasion utilisant des exemples contradictoires ont été proposées. Quelques illustrations sont données ci-dessous. Si celles-ci sont principalement issues du domaine de la vision par ordinateur (computer vision), car particulièrement parlantes et compréhensibles pour le lecteur, il faut bien noter que les attaques par évasion sont applicables à tout type de système reposant sur l'apprentissage automatique.

3.4.2 Attaques par reprogrammation (adversarial reprogramming attacks)

Les attaques par évasion sont les plus courantes dans les recherches sur les attaques d'apprentissage automatique. Cependant, un nouveau type d'attaque a été introduit récemment dans l'article de Elsayed *et al.* [49]). Il s'agit des attaques par reprogrammation. Ces attaques ont pour but de changer le comportement d'un système de classification d'image en le faisant accomplir une tâche supplémentaire. Les auteurs ont réussi à réaliser cela en utilisant des techniques d'apprentissage par renforcement qui permettent d'injecter des données malveillantes dans les couches de neurones d'un réseau de neurones convolutif utilisé pour la classification d'images, ce qui a entraîné un changement de comportement de ce dernier.

La méthodologie de l'article se décompose en trois étapes :

1. Établir une correspondance entre certains labels d'images (poisson rouge, autruche, requin, etc.) et un certain nombre de carrés blancs.
2. Ajouter un bruit adversaire intégrant le nombre de carrés blancs correspondant à chacune des images correspondantes aux labels sélectionnés.
3. Ré-entrainer le système d'IA à l'aide des images ainsi modifiées.

Une fois le système d'apprentissage ré-entrainé, il est capable de réaliser la nouvelle tâche pour laquelle il a été reprogrammé, qui est de compter les carrés blancs. Cette opération peut être réalisée de façon invisible, car les sorties de l'algorithme restent liées à la tâche initiale, seul l'attaquant connaissant la correspondance entre les deux tâches. Cette étude montre que les systèmes d'apprentissage peuvent être vulnérables aux attaques par reprogrammation et souligne l'importance de la sécurité des systèmes d'IA pour protéger les données et les utilisateurs contre ce type d'attaque.

3.4.3 Attaques par déni de service

Les attaques par déni de service sont un autre type d'attaque de systèmes d'apprentissage automatique par manipulation qui visent principalement la disponibilité du système d'apprentissage. Ces attaques sont des adaptations des attaques classiques aux caractéristiques de l'apprentissage. Par exemple, des images ayant subi des rotations ou des translations peuvent causer des problèmes d'analyse pour des systèmes de classification. De nombreux travaux, comme ceux de (Ford *et al.* [53]), explorent la robustesse des systèmes de classification (en particulier d'images) face à l'ajout de bruit, dont les exemples contradictoires ne sont qu'une sous-partie. Certaines attaques visent également à attenter à la disponibilité du service en soumettant des requêtes particulièrement gourmandes en capacités de calcul (Hong *et al.* [74]).

3.4.4 Techniques de défenses contre les "exemples adverses"

Pour contrer les attaques dites "exemples contradictoires", la défense actuelle se concentre principalement sur deux approches distinctes. La première repose sur une série de méthodes visant à détecter ces exemples, c'est-à-dire à identifier les enregistrements comportant un bruit supplémentaire. La seconde approche consiste à mettre en place un ensemble de méthodes permettant de former un réseau de neurones profonds plus robuste et résistant aux exemples contradictoires.

3.4.4.1 Techniques de détection des exemples adverses

Le but de la détection d'exemples contradictoires est de trouver ces exemples à temps pour protéger les modèles contre les attaques. La complexité de la génération d'exemples contradictoires empêche l'obtention d'une méthode unifiée de détection [58]. De manière générale, deux méthodes sont utilisées : la détection basée sur les caractéristiques et la détection basée sur les invariants.

La détection basée sur les caractéristiques utilise des informations sur les caractéristiques des exemples originaux et contradictoires pour les comparer. Hendrycks *et al.* [69] ont utilisé l'analyse en composantes principales (ACP) pour apprendre les caractéristiques sur des dimensions avec moins d'informations, et ont ajouté des opérations de blanchiment pour réduire la redondance des données. La compression de caractéristiques est une autre méthode utilisée pour réduire le degré de liberté de l'attaquant. Xu *et al.* [201] ont exploré deux méthodes de compression des caractéristiques, l'une consistant à réduire la profondeur de couleur de chaque pixel, l'autre utilisant des filtres pour vérifier l'image originale et l'incohérence de la prédiction des exemples contradictoires. La méthode de détection basée sur les invariants repose sur des différences entre des exemples originaux et contradictoires, comme des modifications dans les réseaux de neurones. Feinman *et al.* [52] ont proposé deux méthodes pour détecter ces différences : la densité du noyau et l'estimation de l'incertitude bayésienne.

La plupart des méthodes de détection d'exemples contradictoires sont entraînées sur la base de modèles de boîte blanche, c'est-à-dire que les informations spécifiques du modèle d'attaque doivent être connues. Cependant, Zheng *et al.* [217] ont proposé une méthode d'apprentissage non supervisé pour modéliser les propriétés du classificateur du réseau de neurones profond sans connaître la méthode d'attaque, ce qui a montré de bonnes performances dans les attaques de boîte noire et de boîte grise.

En résumé, la détection d'exemples contradictoires est une méthode de défense efficace contre les attaques, car elle permet de détecter la présence d'exemples contradictoires et de protéger les modèles. Cependant, il est important de continuer à développer de nouvelles méthodes de détection pour contrer les attaques inconnues.

3.4.4.2. Formation robuste du modèle face aux exemples adverses

Deux principales techniques de défense ont été identifiées dans la littérature, qui ont été utilisées pour développer des modèles d'apprentissage automatique résistants aux attaques adverses.

- Entraînement adverse (contradictoire) : est une technique visant à améliorer la robustesse d'un modèle face à des exemples contradictoires, en ajoutant des exemples de ce type à l'ensemble de données d'apprentissage. Cela permet d'éviter que le taux de précision de la classification ne soit réduit ou n'atteigne zéro en présence d'exemples contradictoires. Pour obtenir des exemples contradictoires, une méthode d'attaque peut être utilisée pour attaquer le modèle. Ensuite, les exemples obtenus peuvent être ajoutés à l'ensemble de données d'apprentissage propre pour former un ensemble de données mixte, sur lequel le modèle sera entraîné.

  La formation contradictoire vise à minimiser le risque de perturbation du modèle dans le pire des cas, ce qui en fait un problème d'optimisation. Le concept de formation contradictoire a été proposé par Szegedy *et al.* [182] et Goodfellow *et al.* [61], tandis que Madry *et al.* [114] ont expliqué cette technique d'un point de vue d'optimisation robuste en utilisant PGD Attack (Projected Gradient Descent Attack). L'entraînement contradictoire peut être réalisé en une seule étape ou en plusieurs étapes, selon le nombre d'itérations de la méthode d'attaque. Cependant, le surajustement robuste peut toujours se produire, même avec un réseau de formation contradictoire bien conçu.

L'entraînement contradictoire en une seule étape peut conduire à un surajustement catastrophique, mais la formation contradictoire guidée peut aider à améliorer ses performances. Pour éviter ce problème, la formation contradictoire en plusieurs étapes peut être utilisée, mais cela peut augmenter considérablement les coûts de formation. Pour résoudre ce problème, des exemples contradictoires transférables peuvent être utilisés pour réduire le temps de formation tout en maintenant une précision similaire. Les méthodes d'entraînement contradictoire sont utiles pour améliorer la robustesse des modèles, mais elles sont plus coûteuses en temps que d'autres techniques de défense.

- Conception robuste de réseaux de neurones : la conception de réseaux de neurones robustes peut être effectuée de deux manières : en modifiant ou en ajoutant des blocs structurels internes de réseaux de neurones lors de l'utilisation d'ensembles de données normaux, ou en utilisant des réseaux supplémentaires pour se défendre contre les perturbations. Pour modifier les réseaux, il existe deux méthodes principales : la régularisation de modèle et le réseau contractif profond. La régularisation de modèle consiste à interférer avec les attaques adverses en rendant invisibles les informations de gradient du modèle et à réduire le taux de réussite des attaques. La régularisation peut être utilisée avec la formation contradictoire, où des exemples contradictoires sont ajoutés aux ensembles de données pour modifier les paramètres du modèle. Pour améliorer la robustesse, les auto-encodeurs contractifs profonds peuvent être utilisés pour apprendre des caractéristiques invariantes de chaque couche. En utilisant des réseaux supplémentaires, la défense contre les perturbations générales peut être effectuée en ajoutant une couche détecteur de perturbation au réseau d'origine pour détecter et corriger les perturbations. Une stratégie de défense basée sur le GAN peut également être utilisée pour modifier les réseaux de neurones.

3.5 Conclusion

Dans ce chapitre, nous avons présenté un état de l'art complet sur les différentes attaques connues ciblant les modèles de l'apprentissage automatique. Nous avons, plus particulièrement, examiné les attaques par infection, par manipulation et par modification, et leur impact sur la qualité des résultats des modèles d'apprentissage automatique. Ces attaques peuvent être pernicieuses et nuire à la confidentialité des données qui ont servi à la formation des modèles et à la fiabilité des résultats obtenus. Elles peuvent être utilisées pour influencer les résultats des modèles d'apprentissage automatique, de manière ciblée ou non, ce qui peut entraîner des conséquences graves pour les utilisateurs finaux.

Nous avons, par la suite, pour chacune de ces catégories d'attaques, présenté une variété de techniques de défense et mis en avant leurs points forts ainsi que leurs points faibles. Finalement, nous avons mis en lumière la nécessité de développer une technique de défense efficace pour garantir la sécurité et la qualité des résultats des modèles d'apprentissage automatique.

Le prochain chapitre présente notre proposition de recherche qui porte sur les techniques de défense pour les modèles d'apprentissage automatique en tant que service pour atténuer le risque des attaques par inférence d'appartenance. Nous avons développé une technique innovante pour assurer la robustesse et la fiabilité des modèles d'apprentissage automatique qui s'exécutent dans des environnements hostiles.

Chapitre 4

Défense contre l'inférence d'appartenance pour l'apprentissage machine en tant que service service

4.1 Introduction

L'essor de l'apprentissage machine en tant que service (Machine Learning as a Service : MLaaS) a démocratisé l'accès aux technologies d'apprentissage automatique, permettant aux entreprises de toutes tailles d'exploiter ces capacités sans la nécessité d'une expertise interne spécialisée. Des acteurs majeurs tels que Amazon [7], Microsoft [37], IBM [81] et Google [62] ont joué un rôle prépondérant dans cette transformation, rendant ces technologies accessibles à un large éventail d'utilisateurs. Toutefois, l'accessibilité accrue soulève des défis significatifs en termes de cybersécurité. Cela est particulièrement préoccupant dans des domaines d'application sensibles tels que la santé, où la divulgation de données d'entraînement sensibles peut entraîner non seulement des coûts financiers, mais aussi une érosion de la confiance des parties prenantes.

Les attaques par inférence d'appartenance mettent en évidence une vulnérabilité majeure dans ces environnements MLaaS : elles permettent aux attaquants d'exploiter les frontières décisionnelles des modèles pour extraire des informations délicates, telles que l'identification des individus dans un ensemble de données d'entraînement. Dans les environnements MLaaS, où les modèles peuvent facilement être accessibles, ces attaques sont particulièrement menaçantes [166]. Bien que certaines techniques de défense, telles que la confidentialité différentielle [5, 83, 165] et la régularisation, soient partiellement efficaces, elles présentent souvent des compromis significatifs qui peuvent réduire l'utilité des modèles. Ce chapitre aborde de manière systématique ces défis en proposant un modèle de défense qui non seulement protège contre ces attaques mais aussi préserve l'utilité des modèles d'apprentissage automatique.

Nous introduisons un nouveau mécanisme pour protéger la confidentialité des données d'entraînement dans un modèle d'apprentissage machine en tant que service contre les attaques par inférence d'appartenance dans un contexte de boîte noire. Notre approche consiste à ajouter un bruit soigneusement élaboré au vecteur de prédiction de chaque échantillon de requête afin de le rendre inutilisable par les attaquants. Ce faisant, nous équilibrons la confidentialité et l'utilité, en nous assurant que le vecteur de prédiction indique toujours la bonne classe de prédiction. Cette méthode présente l'avantage de pouvoir être appliquée à un classificateur existant sans nécessiter de réentraînement, ce qui est souvent coûteux.

Les objectifs de notre approche sont doubles : (i) introduire une incertitude pour l'attaquant quant à la présence ou à l'absence d'un individu dans les données d'apprentissage et (ii) maintenir l'utilité du vecteur de prédiction. Nous nous engageons à préserver la fonctionnalité et l'efficacité du modèle, sans accepter la moindre dégradation de performance.

Notre approche formule la réalisation de ces objectifs comme un problème d'optimisation complexe, en raison de l'espace de bruit de haute dimension. De plus, nous considérons un aspect important : le comportement dynamique et stratégique de l'adversaire. Pour aborder ce scénario adversarial, nous adoptons un modèle de jeu non-coopératif inspiré des travaux de Bouhaddi *et al.* [22]. Le modèle proposé intègre les récompenses et les coûts dans le processus décisionnel pour mieux comprendre la nature de cette interaction entre le défenseur du modèle et l'attaquant ainsi que les problèmes sous-jacents. Notre approche offre une perspective nouvelle sur le problème des attaques par inférence, étant la première en son genre.

4.2 Les attaques par inférence d'appartenance et travaux connexes

Cette section explore les attaques par inférence d'appartenance, en passant en revue les travaux qui se sont concentrés sur la détection et la neutralisation de ces menaces.

4.2.1 Attaques par inférence d'appartenance

Les attaques par inférence d'appartenance révèlent si les données d'un individu ont été utilisées dans l'entraînement d'un modèle d'apprentissage automatique. Elles sont classifiées par le NIST comme une violation de la vie privée et peuvent donc exposer les fournisseurs de services à des risques réglementaires. Ces attaques ciblent une variété de modèles et de domaines d'applications, incluant les informations de localisation et les données génomiques.

Formellement, pour un classificateur cible $f : \mathbb{R}^n \to \Delta_k$ où Δ_k représente l'espace des distributions de probabilité sur k classes, une MIA est définie par :

$$\mathcal{A} : (x, f(x)) \to [0, 1] \tag{4.1}$$

où $\mathcal{A}$ est le modèle d'inférence, qui est un classificateur probabiliste qui produit une estimation ρ indiquant la probabilité que x ait été utilisé dans l'entraînement de f.

Dans la majorité des cas, les attaques MIA s'opèrent dans un contexte de boîte noire, où l'adversaire n'a accès qu'aux sorties du modèle cible pour des raisons pratiques. Shokri *et al.* [166] ont développé une attaque MIA de boîte noire qui utilise les vecteurs de prédictions du modèle cible, combinée à la création de modèles fantômes conçus pour simuler le comportement du modèle cible. Ces modèles fantômes servent ensuite à générer des données qui alimentent un classificateur binaire basé sur des réseaux de neurones, destiné à déterminer si un échantillon faisait partie du jeu de données d'entraînement du modèle initial. Cette méthode a été raffinée par Salem *et al.* [155], qui ont simplifié l'approche en utilisant un seul modèle fantôme, tandis que Nasr *et al.* [130] ont augmenté l'efficacité de cette technique en intégrant des caractéristiques additionnelles, comme les étiquettes de classe, pour renforcer la précision de l'attaque.

4.2.2 Défenses contre les attaques par inférence d'appartenance

Pour contrer les attaques par inférence d'appartenance, diverses études récentes ont adopté des techniques de régularisation et des méthodes d'apprentissage en ensemble afin de réduire le surajustement. Le surajustement survient lorsqu'un modèle d'apprentissage automatique devient excessivement dépendant de ses données d'entraînement. Dans un tel cas, le modèle manifeste un niveau de confiance très important dans ses prédictions pour les données qu'il a déjà vues (les données membres), par rapport à de nouvelles données (les données non-membres). Cette distinction dans la confiance des prédictions pour les membres et les non-membres crée une ouverture que les attaquants peuvent exploiter pour identifier si une donnée spécifique a été utilisée dans l'entraînement du modèle. En atténuant le surajustement, on peut diminuer la différence dans les niveaux de confiance entre membres et non-membres, rendant ainsi les attaques par inférence d'appartenance moins efficaces.

Régulariseur L_2 [165]. Introduit par Shokri *et al.* [165] le régulariseur L_2 ajoute un terme de pénalité pondéré égal à la norme L_2 des paramètres du modèle à la fonction de perte originale. Cette méthode a démontré son efficacité pour réduire le risque des attaques par inférence d'appartenance pendant l'entraînement du modèle.

Dropout [156]. Utilisé pour prévenir le surajustement dans les réseaux de neurones [173], le dropout élimine aléatoirement certains neurones pendant l'entraînement, selon une probabilité prédéfinie. Cette technique aide non seulement à réduire le surajustement mais sert aussi de défense contre les attaques par inférence d'appartenance [156].

Jeu Min-max [130]. Proposé par Nasr *et al.* [130], cette approche consiste à intégrer un régularisateur adversarial dans la fonction de perte pour protéger la confidentialité tout en minimisant la perte de prédiction. Le modèle s'entraîne à optimiser la précision tout en maximisant la confidentialité des données, représentant une solution à double objectif.

Empilage de modèles [156]. L'empilage de modèles est une technique d'ensemble qui implique la combinaison de plusieurs classificateurs simples pour créer un modèle plus complexe capable de faire des prédictions plus précises [142]. Cette approche est couramment utilisée pour réduire le surajustement et peut également être exploitée pour améliorer la confidentialité et atténuer les attaques par inférence d'appartenance [156]. En combinant les sorties de différents modèles, l'empilage de modèles peut aider à réduire l'impact du surajustement sur les données d'entraînement et ainsi rendre plus difficile pour un attaquant de distinguer entre membres et non-membres.

Bien que la réduction du surajustement soit souvent considérée comme une stratégie principale pour contrer les attaques par inférence d'appartenance, des recherches récentes, notamment celles de Shokri [166] *et al.*, ont révélé que ce n'est pas le seul vecteur par lequel ces attaques peuvent être efficaces. En effet, différents modèles d'apprentissage automatique, même s'ils sont comparables en termes de surajustement, peuvent divulguer des informations à des degrés divers en raison de leurs architectures et configurations uniques. Ces différences structurelles entre les modèles peuvent affecter la façon dont les informations sont conservées ou exposées lors des requêtes d'inférence, offrant ainsi différentes opportunités aux attaquants.

Les attaques par inférence d'appartenance exploitent précisément ces variations, en utilisant les vecteurs de prédiction que les modèles génèrent pour les membres et non-membres du jeu de données d'entraînement. Ces vecteurs de prédiction, ou aussi appelés les scores de confiance, lorsqu'ils sont différenciés de manière significative entre les membres et les non-membres, permettent à un adversaire d'inférer l'appartenance avec une certaine précision. Pour aborder cette vulnérabilité sous-jacente qui dépasse le simple surajustement, les chercheurs ont proposé des stratégies diversifiées qui prennent en compte non seulement la réduction de la dépendance excessive aux données d'entraînement mais aussi la modification des caractéristiques intrinsèques des modèles pour rendre les inférences moins prédictibles et plus sécurisées.

MemGuard [92]. Jia *et al.* [92] ont développé une méthode innovante pour défendre contre les attaques par inférence d'appartenance. Cette technique, nommée MemGuard, transforme le vecteur de score de confiance en un exemple adverse en ajoutant un bruit soigneusement calculé. Cette approche tire parti de la transférabilité des exemples adverses, rendant le vecteur de score modifié inutilisable par le classificateur de l'attaquant, et donc protégeant l'appartenance des données [138, 136].

Confidentialité différentielle. La confidentialité différentielle, introduite par Dwork *et al.* [46], est une méthode qui a fait ses preuves pour maintenir la confidentialité dans les systèmes de traitement des données. Elle peut être appliquée de deux manières : en ajoutant du bruit directement à la fonction objectif du modèle [29, 83, 96] ou au gradient pendant l'optimisation [5, 13, 172, 190]. Bien que cette technique offre des garanties solides de préservation de la confidentialité, elle peut considérablement affecter la précision du modèle, ce qui représente une faiblesse significative dans le compromis entre la confidentialité et l'utilité [106, 129].

4.3 Formulation du problème

La dynamique entre le fournisseur du modèle, l'attaquant, et le défenseur forme le cœur de notre étude sur la sécurité dans les environnements d'apprentissage machine. Chacun joue un rôle important dans l'écosystème des attaques par inférence d'appartenance et les défenses correspondantes.

4.3.1 Modèle du fournisseur de services d'apprentissage automatique

Le fournisseur de services d'apprentissage automatique offre l'accès à un modèle cible via une API black-box. Cette API traite les requêtes de prédiction des utilisateurs, recevant une entrée x et renvoyant la classe prédite accompagnée du vecteur de probabilités de prédiction v. Nous définissons le modèle cible par la fonction $f : x \to v$, où le vecteur v représente la distribution de probabilité postérieure des étiquettes pour l'échantillon d'entrée. Concrètement, v_j est la probabilité que l'échantillon appartienne à la classe j, et la classe prédite est celle avec la probabilité la plus élevée, soit $\underset{j}{\operatorname{argmax}} v_j$.

Nous nommons ce modèle le *classificateur cible*, qui est typiquement un réseau de neurones dans notre étude. Les spécifications des formats d'entrée et de sortie de l'API sont déterminées par le fournisseur dans le cadre d'un accord de service. Cependant, la structure interne du modèle f et les données d'entraînement D restent confidentielles. Les utilisateurs ont uniquement accès aux prédictions v, ce qui assure une interaction strictement limitée au mode black-box avec le modèle f.

4.3.2 Modèle de menace de l'attaquant

L'objectif d'un attaquant est de découvrir l'ensemble de données d'entraînement privé du fournisseur du modèle. L'attaquant n'a qu'un accès en mode black-box au classificateur cible, ce qui signifie qu'il peut soumettre des échantillons de données de requête et recevoir leurs vecteurs de score de confiance correspondants prédits par le classificateur cible. Pour atteindre cet objectif, l'attaquant utilise des attaques par inférence d'appartenance en mode black-box [129, 156, 166].

Dans ce type d'attaque, l'attaquant entraîne un classificateur qui prend en entrée le vecteur de score de confiance d'un échantillon de données et prédit si cet échantillon fait partie ou non de l'ensemble d'entraînement du classificateur cible. Ce classificateur de l'attaquant, désigné par $\mathcal{A}$, reçoit en entrée un échantillon de données x et son vecteur de prédiction correspondant $f(x) = v$, et génère une probabilité ρ. Une probabilité proche de 0 indique que la donnée x n'appartient pas à l'ensemble d'entraînement du classificateur cible, tandis qu'une probabilité proche de 1 signifie qu'elle en fait partie. Pour des raisons de clarté, nous désignons le classificateur de l'attaquant comme le *classificateur d'attaque*.

Il est à noter que nous supposons que l'attaquant est conscient de notre mécanisme de défense, mais que le défenseur n'a pas connaissance du classificateur d'attaque, car l'attaquant a plusieurs options pour choisir son classificateur. Cette hypothèse nous permet de considérer des attaques puissantes. De plus, nous faisons l'hypothèse réaliste que le nombre de requêtes provenant de l'attaquant est limité. En pratique, les fournisseurs de MLaaS peuvent mettre en place des mesures de protection en limitant le nombre de requêtes. Cette stratégie courante vise à empêcher les attaquants de réaliser un grand nombre de requêtes et potentiellement d'inférer des informations sur l'appartenance. Par exemple, la plateforme Cloud AI de Google limite le nombre de requêtes de prédiction par jour pour chaque projet, et AWS SageMaker impose une restriction similaire sur le nombre de requêtes par seconde. En limitant le nombre de requêtes, la capacité de l'attaquant à recueillir des informations sur les données d'entraînement est restreinte, rendant plus difficile la réalisation d'une attaque par inférence d'appartenance réussie. Par conséquent, le nombre de requêtes par intervalle de temps est limité à Θ.

4.3.3 Modèle du défenseur

Dans le cadre de la défense contre les attaques par inférence d'appartenance en mode black-box, le défenseur, qui peut être le fournisseur du modèle ou une tierce partie de confiance, cherche à protéger les informations sensibles présentes dans les données d'entraînement du modèle. Pour atteindre cet objectif, le défenseur intègre un vecteur de bruit dans le vecteur de prédiction généré par le classificateur cible pour chaque requête utilisateur. Mathématiquement, cette intervention est exprimée par la formule suivante :

$$v' = v + \delta$$

où v représente le vecteur de prédiction initial, δ désigne le vecteur de bruit ajouté par le défenseur, et v' est le vecteur de prédiction bruité qui est retourné à l'utilisateur. Cette stratégie permet de s'assurer que les attaquants n'ont accès qu'à des vecteurs de score de confiance bruités.

Les objectifs du défenseur sont par conséquent doubles :

1. **Prévention de l'inférence d'appartenance :** le défenseur vise à empêcher l'attaquant de déduire des informations sur l'appartenance à l'ensemble d'entraînement en ajoutant suffisamment de bruit au vecteur de prédiction.
2. **Conservation de la précision des prédictions :** il est essentiel que l'ajout de bruit ne compromette pas significativement la performance du classificateur. La précision du classificateur bruité doit rester comparable à celle du classificateur original pour ne pas impacter l'utilité du service.

4.4 Architecture de notre défense

Pour renforcer la défense contre les attaques par inférence d'appartenance, un des défis majeurs est l'absence d'information du défenseur concernant le classificateur utilisé par l'attaquant. Afin de pallier cette lacune, le défenseur prend l'initiative de former son propre classificateur d'inférence. Cette démarche stratégique permet d'ajuster le vecteur de bruit de façon à ramener l'attaquant dans une zone d'incertitude quant à sa prédiction de l'appartenance des données. En anticipant que l'attaquant pourrait disposer d'un avantage technologique, il est logique de supposer qu'il développe son classificateur dans un cadre hostile, intégrant des exemples adverses pour améliorer sa résilience aux stratégies de défense conventionnelles. En réponse, le défenseur adopte une approche similaire, en formant son *classificateur de défense* dans des conditions adverses similaires, enrichissant ainsi sa capacité à neutraliser et à naviguer à travers les tentatives d'attaque sophistiquées.

Formellement, nous avons :

$$\mathcal{D} : (x, v) \to [0, 1] \tag{4.2}$$

Le classificateur de défense prédira si un point de données x appartient à l'ensemble d'entraînement du classificateur cible en fonction de la probabilité $\mathcal{D}(x, v)$. Si cette probabilité est proche de 1, le classificateur de défense conclura que x est un membre de l'ensemble d'entraînement. D'un autre côté, si la probabilité est proche de 0, il conclura que x n'en est pas membre.

Pour atteindre son premier objectif de tromper le classificateur de l'attaquant, le défenseur ajoutera du bruit au vecteur de prédiction. Cette technique vise à amener le classificateur d'inférence dans une région d'incertitude où il ne peut pas faire de prédictions exactes. Cette situation est mathématiquement décrite par la formule suivante :

$$0.5 - \tau \leq \mathcal{D}(x, v + \delta) \leq 0.5 + \tau,$$

où τ est un paramètre positif réel petit, $\tau \in \mathbb{R}^+$.

Le deuxième objectif stratégique est de minimiser la perte d'utilité du modèle lorsque nous bruitons le vecteur de prédiction. Comme il s'agit d'un modèle de classification, la perte d'utilité doit être inexistante. En effet, l'ajout de bruit doit être géré de manière à ne pas altérer la classe prédite par le modèle. Cette exigence est formulée par l'équation suivante :

$$\underset{j}{\operatorname{argmax}} \upsilon_j = \underset{j}{\operatorname{argmax}} (\upsilon_j + \delta_j).$$

Il est essentiel d'assurer que l'ajout de bruit préserve la structure probabiliste du vecteur de prédiction. Ainsi, nous structurons le problème d'optimisation de défense de la manière suivante :

$$\min_{\delta} \; d(\upsilon, \upsilon + \delta) \tag{4.3}$$
$$0.5 - \tau \leq \mathcal{D}(x, \upsilon + \delta) \leq 0.5 + \tau \tag{4.4}$$
$$\underset{j}{\operatorname{argmax}} \; \upsilon_j = \underset{j}{\operatorname{argmax}} (\upsilon_j + \delta_j) \tag{4.5}$$
$$\upsilon_j + \delta_j \leq 1, \forall j \tag{4.6}$$
$$\sum_j \delta_j = 0 \tag{4.7}$$

La fonction objectif décrite dans l'équation 4.3 vise à minimiser le bruit ajouté au vecteur de prédiction, où la notation $d(\upsilon, \upsilon + \delta)$ fait référence à la norme euclidienne. La formulation du problème de protection des données d'entraînement comme un problème d'optimisation avec contraintes aborde partiellement les objectifs du défenseur, qui est de garantir que son vecteur de prédiction bruité ne contient aucune information sur les données d'entraînement. Pour obtenir une fonction objectif sans contrainte à optimiser, nous utilisons la méthode des multiplicateurs de Lagrange [40] pour transformer le problème (4.3)-(4.7). Cette méthode implique la formulation d'une fonction de Lagrange qui prend en compte la fonction objectif ainsi que les contraintes du problème. Ensuite, nous optimisons la fonction de Lagrange en trouvant les valeurs des variables qui la rendent minimale.

4.4.1 Formulation de Lagrangien

Pour transformer ce problème d'optimisation avec contraintes en une forme plus facile à résoudre à l'aide des techniques d'optimisation standard, nous introduisons la méthode des multiplicateurs de Lagrange. Chaque contrainte aura un multiplicateur associé :

1. . λ_1 et λ_2 pour la contrainte de l'inférence d'appartenance à double bornage,
2. λ_3 pour la contrainte de cohérence de prédiction de classe,
3. pour chaque borne de probabilité $\upsilon_j + \delta_j \leq 1$,
4. pour la somme des perturbations égale à zéro.

Le Lagrangien $\mathcal{L}$ est alors donné par :

$$\begin{aligned} \mathcal{L}(\delta, \lambda) = {} & d(\upsilon, \upsilon + \delta) + \lambda_1(0.5 - \tau - \mathcal{D}(x, \upsilon + \delta)) + \lambda_2(\mathcal{D}(x, \upsilon + \delta) - 0.5 - \tau) \\ & + \lambda_3(\underset{j}{\operatorname{argmax}} \; \upsilon_j - \underset{j}{\operatorname{argmax}} \; (\upsilon_j + \delta_j)) + \sum_j \lambda_{4j}(\upsilon_j + \delta_j - 1) + \lambda_5 \left(\sum_j \delta_j \right). \end{aligned} \tag{4.8}$$

Pour chaque composante δ_j de δ, nous devons calculer la dérivée partielle de $\mathcal{L}$ par rapport à δ_j et l'égaliser à zéro. Cela implique de considérer la dérivée de la fonction de distance d, ainsi que les contributions des différentes contraintes qui impliquent δ.

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \delta_j} =& \frac{\partial}{\partial \delta_j} d(\upsilon, \upsilon + \delta) + \lambda_1 \frac{\partial}{\partial \delta_j}(0.5 - \tau - \mathcal{D}(x, \upsilon + \delta)) + \lambda_2 \frac{\partial}{\partial \delta_j}(\mathcal{D}(x, \upsilon + \delta) - 0.5 - \tau) \\ &+ \lambda_3 \frac{\partial}{\partial \delta_j}(\underset{j}{\operatorname{argmax}}\ \upsilon_j - \underset{j}{\operatorname{argmax}}\ (\upsilon_j + \delta_j)) + \lambda_{4j} + \lambda_5 \end{aligned} \tag{4.9}$$

Cette formulation donne un système d'équations que nous pouvons résoudre pour trouver les valeurs de δ et λ qui minimisent la fonction objectif tout en satisfaisant les contraintes imposées.
En optimisant ce Lagrangien via des méthodes de descente de gradient, nous veillons à ce que les ajustements de δ respectent à la fois la nécessité de confondre le classificateur de l'attaquant et de préserver l'utilité des prédictions.

La solution proposée vise à prévenir efficacement les attaques par inférence d'appartenance en brisant le lien direct entre les données d'entraînement et leurs vecteurs de prédiction associés. Cependant, résoudre de tels problèmes d'optimisation peut être extrêmement gourmand en ressources, impactant potentiellement la performance du modèle en termes de capacité de calcul et de rapidité de traitement.

Pour renforcer cette solution, nous proposons d'incorporer la dynamique stratégique de l'attaquant directement dans le processus décisionnel du défenseur. Bien que peu explorée dans les recherches existantes, cette approche pourrait significativement améliorer l'efficacité du mécanisme de défense en assurant un équilibre optimal entre confidentialité, utilité et applicabilité pratique. En modélisant les interactions entre le défenseur et l'attaquant comme un jeu non-coopératif, où les intérêts sont divergents, nous pouvons mieux comprendre et anticiper les mouvements de l'attaquant. Le processus décisionnel du défenseur est illustré dans la Figure 4.1.

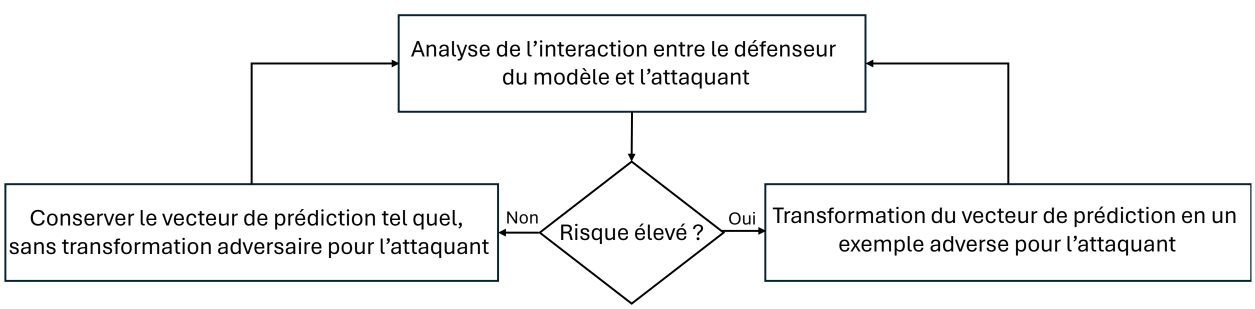

FIGURE 4.1 – Vue d'ensemble du processus décisionnel utilisé par le défenseur de MLaaS.

Dans ce modèle, les gains d'un joueur ne correspondent pas nécessairement aux pertes de l'autre. Par exemple, lorsque l'attaquant cherche à maximiser l'inférence d'informations confidentielles en rapport avec les données d'entraînement, le défenseur vise à minimiser cette fuite tout en maintenant la fonctionnalité du système. Par conséquent, nous utilisons un modèle de jeu à somme non nulle pour simuler cette interaction.

4.4.2 Modèle de jeu non-coopératif et décision dynamique

Lorsqu'un attaquant soumet une requête de classification pour la donnée x_i, il peut choisir entre utiliser son modèle d'attaque pour déterminer si x_i appartient à l'ensemble d'entraînement (stratégie "Inférer"), ou s'abstenir de toute action pour éviter des déductions incorrectes (stratégie "Ne pas Inférer"). En réponse, le défenseur a la possibilité d'appliquer une transformation adverse δ_i^* du vecteur de prédiction pour induire une incertitude chez l'attaquant (stratégie "Bruiter"), ou de retourner le vecteur de prédiction initial υ_i sans modification (stratégie "Ne pas Bruiter"). Chaque décision présente des compromis entre coûts et gains, nécessitant une analyse soignée pour optimiser la réponse du défenseur.

Lorsque l'attaquant décide d'exécuter une attaque par inférence d'appartenance sur un point de données x_i, cela implique un coût, noté c_A. Ce coût pourrait inclure les ressources nécessaires pour entraîner un classificateur d'inférence, des coûts monétaires ou d'autres dépenses. Chaque requête faite par l'attaquant augmente la probabilité qu'il soit détecté et bloqué. Nous utiliserons la notation $\eta_t \leq \Gamma$ pour indiquer le nombre de requêtes effectuées au temps t.

Pour la défense, ajouter du bruit au vecteur de prédiction engendre un coût de sécurité c_S, qui comprend les ressources informatiques et temporelles nécessaires pour trouver le vecteur de bruit optimal qui résout le problème 4.3. Cependant, cette approche procure une récompense en termes de confidentialité des données ϵ_i, avec un gain g_S. Nous fixons $\sum_i \epsilon_i \leq \epsilon$, où le paramètre ϵ peut être vu comme un budget de confidentialité, où une valeur plus élevée signifie que moins de confidentialité est préservée, tandis qu'un budget plus petit conduit à une meilleure protection des données. Si la somme de toutes les pertes de confidentialité dépasse le budget, la seule option pour la défense est d'ajouter du bruit au vecteur de prédiction. Ainsi, ϵ est crucial pour maintenir un équilibre entre la confidentialité et la précision des prédictions. Ainsi, le jeu non-coopératif entre la défense et l'attaquant est formellement défini comme suit.

$$\mathcal{G} = (\{D, A\}, \{S_D, S_A\}, \{U_D, U_A\})$$

où :

- $\{D, A\}$ représente l'ensemble des joueurs, où D est le défenseur et A l'attaquant.
- $\{S_D, S_A\}$ sont les ensembles de stratégies disponibles pour le défenseur et l'attaquant. Pour le défenseur D, les stratégies peuvent être $S_D = \{\text{"Bruiter, "Ne pas Bruiter"}\}$, et pour l'attaquant A, $S_A = \{\text{"Inférer", "Ne pas Inférer"}\}$.
- $\{U_D, U_A\}$ désignent les fonctions d'utilité pour le défenseur et l'attaquant. Ces fonctions quantifient les gains ou pertes associés à chaque combinaison de stratégies des deux joueurs. L'utilité est généralement affectée par la stratégie adoptée par l'adversaire, reflétant la nature non-coopérative du jeu.

Dans ce modèle, chaque joueur choisit une stratégie pour maximiser sa propre utilité, sans coopération avec l'autre joueur. Le défenseur cherche à minimiser l'impact des attaques potentielles tout en préservant la fonctionnalité et la précision des prédictions du système, tandis que l'attaquant tente de maximiser l'efficacité de son attaque pour extraire ou corrompre des informations.

Ce cadre de jeu fournit une structure formelle pour analyser et développer des stratégies de défense contre les attaques dans les systèmes d'apprentissage automatique, tout en tenant compte des actions potentielles et des objectifs de l'attaquant. Les résultats de ces interactions sont souvent présentés sous forme de matrice de paiement, illustrant les résultats pour différentes combinaisons de stratégies. Les gains des deux joueurs sont définis dans le Tableau 4.1.

TABLE 4.1 – Matrice de gains du jeu entre le défenseur et l'attaquant

Défenseur	**Attaquant**	
	Inférer	**Ne pas Inférer**
Bruiter	$(-c_S + g_S, -c_A)$	$(-c_S, 0)$
Ne pas Bruiter	$(-\epsilon_i, \epsilon_i - c_A)$	$(0, 0)$

Nous pouvons raisonnablement supposer que g_S est supérieur ou égal à c_S, et que ϵ_i est supérieur ou égal à c_A. Cette supposition est basée sur l'idée qu'un attaquant ne sera motivé à attaquer que si l'information confidentielle ϵ_i d'un point de données particulier x_i est jugée plus précieuse que le coût de l'attaque elle-même. De même, un défenseur ne sera incité à protéger la confidentialité des données que si le coût pour le faire est inférieur aux récompenses de sécurité potentielles.

4.4.2.1 Analyse de l'équilibre de Nash

L'équilibre de Nash dans ce contexte se définit comme un état où aucun joueur ne peut améliorer son gain en changeant unilatéralement sa stratégie, en supposant que les stratégies des autres joueurs restent inchangées. Pour notre modèle, nous identifions les configurations suivantes comme des équilibres de Nash potentiels, en fonction des paramètres spécifiques du jeu :

1. (Bruiter, Inférer) cette paire forme un équilibre de Nash en stratégies pures si et seulement si la différence entre le coût de la mise en place du bruit $(c_S - g_S)$ est inférieure ou égale à la perte de confidentialité due à l'attaque ϵ_i, et le coût de l'attaque (c_A) est négligeable, soit approximativement zéro.
2. (Ne pas Bruiter, Inférer) et équilibre survient si et seulement si $c_S - g_S \geq \epsilon_i$, impliquant que les coûts de défense excèdent les pertes de confidentialité encourues, rendant la défense inutile.

Ces deux équilibres de Nash en stratégies pures illustrent une diversité de scénarios possibles qui peuvent compliquer la prédiction des actions des joueurs. Cette ambiguïté nous pousse à envisager des équilibres en stratégies mixtes, qui peuvent offrir une représentation plus nuancée et pratique du comportement des joueurs.

Dans l'analyse des stratégies mixtes, nous supposons que l'attaquant décide d'exécuter l'attaque avec une probabilité α et que le défenseur opte pour l'ajout de bruit avec une probabilité β.

Supposons que l'attaquant emploie sa stratégie pure d'exécuter l'attaque d'inférence d'appartenance avec une probabilité α, et que le défenseur choisit sa stratégie pure d'ajouter du bruit au vecteur de prédiction avec une probabilité β. En utilisant le principe d'indifférence, nous pouvons dériver l'équilibre de Nash en stratégies mixtes pour les deux joueurs :

$$\alpha^* = \frac{c_S}{g_S + \epsilon_i}$$

$$\beta^* = \frac{c_A + \epsilon_i}{\epsilon_i}$$

Dans le cadre du jeu où le défenseur d'un modèle d'apprentissage machine cherche à sécuriser les données d'entraînement contre les tentatives d'extraction par un attaquant, chaque joueur a la capacité d'observer les actions de l'autre à chaque instant t et de baser ses décisions futures sur cette observation. Par exemple, l'attaquant peut évaluer si les mesures de défense appliquées par le défenseur à un moment donné justifient ou non la poursuite des tentatives d'inférence lors des tours suivants. Cela dépendra de l'efficacité perçue du bruit ajouté par le défenseur pour protéger le vecteur de prédiction. Inversement, le défenseur peut ajuster sa stratégie de perturbation du vecteur de prédiction en fonction de l'intensité et de la fréquence des attaques d'inférence observées, adaptant ainsi ses tactiques pour maximiser la protection tout en optimisant les coûts associés.

Cette dynamique complexe souligne l'importance des choix stratégiques en temps réel que doivent faire les défenseurs et les attaquants. Elle montre également comment ces décisions sont fortement influencées par les coûts, les bénéfices potentiels et les actions des adversaires. En intégrant ces interactions dans le modèle de jeu, les mesures de défense peuvent être ajustées de façon plus dynamique et ciblée. Cela permet d'améliorer la robustesse du modèle face aux attaques d'inférence d'appartenance tout en minimisant les répercussions sur la performance et l'utilité du modèle. En conséquence, ce jeu raffiné aide à établir un équilibre optimal entre sécurité et efficacité opérationnelle, essentiel pour maintenir la confiance et la fiabilité des systèmes d'apprentissage automatique dans des environnements adverses.

4.5 Algorithme de défense dynamique contre les attaques d'inférence d'appartenance

Dans cette section, nous détaillons un algorithme qui orchestre le processus de décision dynamique entre le défenseur et l'attaquant dans le contexte d'un jeu non-coopératif. Cet algorithme démontre comment chaque acteur adapte ses stratégies en temps réel, en réponse aux actions de son adversaire et aux coûts correspondants. L'objectif principal est de maintenir un équilibre optimal entre la sécurité des données et la performance opérationnelle du système, minimisant ainsi les risques de fuite de confidentialité tout en neutralisant efficacement les attaques par inférence d'appartenance. À travers la formulation du problème et l'application de l'algorithme, nous illustrons l'utilisation des stratégies mixtes pour atteindre un équilibre de Nash, assurant ainsi que chaque partie réalise ses objectifs de la manière la plus efficace possible.

L'algorithme 1 fournit un cadre systématique pour la mise en œuvre de stratégies de défense dans un environnement où les attaques d'inférence d'appartenance sont une menace réelle. En utilisant un modèle de jeu non-coopératif, nous pouvons déterminer les conditions sous lesquelles les défenseurs devraient modifier leur vecteur de prédiction pour induire l'incertitude chez l'attaquant, tout en maintenant l'intégrité et la précision du modèle de prédiction.

La dynamique des décisions, représentée par les ajustements des probabilités α^* et β^*, montre comment les défenseurs et les attaquants réagissent aux coûts et aux bénéfices de leurs actions. En pratique, cela implique que le défenseur doit constamment évaluer le risque

Algorithme 1 Algorithme de défense dynamique contre les attaques d'inférence d'appartenance

Prérequis: Modèle de prédiction f, vecteur de prédiction v, coûts c_A, c_S, gains g_S, ϵ_i, budget de confidentialité ϵ, seuils de décision Γ, fréquence des requêtes η_t.

Assurer: Stratégie optimale de défense et d'attaque.

```
Initialiser les probabilités de stratégie α, β à zéro.
Définir les seuils de décision initiaux pour l'attaquant α* = c_S/(g_S+ε_i) et pour le défenseur β* = (c_A+ε_i)/ε_i.
Tant que le système est en opération faire
    Observer les actions de l'adversaire et les vecteurs de prédiction au temps t.
    Si la probabilité d'une attaque calculée est supérieure à α* alors
        Calculer le bruit optimal δ en utilisant la méthode de Lagrange pour résoudre le problème d'optimisation sous contraintes.
        Appliquer le bruit δ au vecteur de prédiction v pour minimiser la capacité de l'attaquant à réaliser une inférence précise.
    sinon
        Envoyer le vecteur de prédiction v sans modification.
    fin Si
    Mettre à jour les stratégies basées sur les observations récentes :
    Ajuster α* en fonction des coûts cumulés et du risque perçu d'attaques futures.
    Ajuster β* pour équilibrer la confidentialité et la performance du modèle.
    Évaluer si la somme des pertes de confidentialité Σ_i ε_i reste inférieure au budget ε.
    Réajuster les paramètres de défense selon les résultats à l'instant t et les prévisions pour t + 1.
    Réappliquer la matrice de gains pour recalculer les récompenses et les coûts (voir Table 4.1).
    Vérifier et ajuster les conditions pour maintenir l'équilibre de Nash, modifiant les stratégies si nécessaire.
fin Tant que
return α* et β*
```

posé par chaque requête d'inférence et ajuster ses mesures de sécurité en conséquence, une stratégie qui est quantifiée par les coûts c_A et c_S, ainsi que par les gains potentiels en termes de confidentialité g_S et ϵ_i.

L'approche par équilibre de Nash, en particulier, aide à stabiliser les stratégies dans des scénarios où les intentions et capacités de l'adversaire sont connues à un degré raisonnable, permettant ainsi une défense plus robuste et prévisible contre les attaques. Cependant, il est important de noter que le calcul et l'application des stratégies mixtes nécessitent une compréhension approfondie des dynamiques du jeu et une analyse continue du comportement de l'adversaire pour ajuster les stratégies de manière efficace.

Notre approche algorithmique ne se contente pas de fournir une méthode pour contrer les attaques, mais elle offre également une perspective sur la manière dont les interactions entre défenseur et attaquant peuvent être structurées et gérées pour minimiser les risques tout en maximisant la sécurité et l'utilité du système.

4.6 Évaluation expérimentale

Cette section est consacrée à l'évaluation par des simulations de la robustesse de notre modèle conçu pour la préservation de la vie privée face aux attaques par inférence d'appartenance, tout en préservant la précision de classification et le maintien des performances.

4.6.1 Ensembles de données

Nous avons choisi deux ensembles de données fréquemment utilisés pour tester la résistance aux attaques par inférence d'appartenance, comme le suggère Shokri *et al.* [167] : Texas100 et Purchase100. Texas100 inclut 67,330 enregistrements issus de données hospitalières, utilisés avec 6,170 attributs binaires pour prédire les procédures médicales. Purchase100, quant à lui, se compose de données client d'une entreprise de vente au détail, avec 100 attributs binaires pour prédire les comportements d'achat. Ces jeux de données couvrent des applications variées, nous permettant d'évaluer la flexibilité et la robustesse de notre modèle dans des contextes différents.

4.6.2 Modèles de classification

Pour les expérimentations, nous utilisons un réseau de neurones entièrement connecté avec quatre couches cachées équipées respectivement de 1024, 512, 256 et 128 neurones, activés par la fonction ReLU. La couche de sortie emploie une fonction d'activation softmax pour générer une distribution de probabilité sur les classes. Le réseau est entraîné via la descente de gradient stochastique (SGD), utilisant une perte d'entropie croisée avec un taux d'apprentissage initial de 0.01 sur 100 époques, réduit d'un facteur 10 après 75 époques pour prévenir le surajustement et favoriser une meilleure convergence.

Les expérimentations menées avec chaque ensemble de données comprennent diverses configurations du nombre d'échantillons dans les ensembles d'entraînement et de référence, détaillées dans le Tableau 4.2 ci-dessous.

Table 4.2 – Configuration expérimentale montrant les tailles de l'ensemble d'entraînement (D), de l'ensemble de référence (D'), ainsi que des membres connus (D^A) et des non-membres connus (D'^A) de chaque ensemble.

Jeu de données	$\|D\|$	$\|D'\|$	$\|D^A\|$	$\|D'^A\|$
Texas100	10,000	5,000	5,000	10,000
Purchase100	20,000	20,000	5,000	20,000

Pour l'entraînement des classificateurs cibles et d'attaque, chaque ensemble de données a été divisé en quatre sous-ensembles distincts, deux dédiés à chaque type de classificateur. Les dimensions de ces sous-ensembles sont précisées dans le Tableau 4.2. Nous avons utilisé les ensembles D et D^A pour entraîner respectivement les classificateurs cibles et d'attaque, tandis que les autres sous-ensembles étaient réservés pour l'évaluation.

4.6.3 Modèle d'attaque par inférence

Dans cette sous-section, nous développons un modèle d'attaque par inférence d'appartenance basé sur une architecture de réseau neuronal, détaillée précédemment en sous-section 4.6.2. Ce modèle d'attaque est conçu pour estimer la probabilité qu'un enregistrement spécifique (x, v) appartienne à l'ensemble de données d'entraînement utilisé par le modèle de classification f. Le cœur de notre modèle repose sur trois sous-réseaux entièrement connectés qui traitent indépendamment le vecteur de prédiction $f(x)$, l'étiquette codée en one-hot de x, et une combinaison des deux. Cette structure permet d'exploiter les informations jointes entre les prédictions de classe et l'appartenance probable aux données d'entraînement, offrant ainsi une vue plus intégrée sans nécessiter un réseau distinct pour chaque classe, une amélioration notable par rapport aux approches antérieures.

Nous optons pour la fonction d'activation ReLU pour sa capacité à introduire des non-linéarités tout en maintenant une convergence rapide lors de l'entraînement. Tous les poids sont initialisés selon une distribution normale centrée à zéro avec un écart type de 0,01, tandis que les biais sont initialisés à zéro. L'entraînement du modèle s'effectue à l'aide de l'optimiseur Adam, réglé sur un taux d'apprentissage de 0,001. Afin de prévenir toute forme de biais dans le processus d'apprentissage, chaque lot utilisé pour l'entraînement du modèle d'attaque comprend une répartition équilibrée entre les exemples de membres et de non-membres.

4.6.4 Résultats empiriques

Lors de nos simulations, trois configurations différentes ont été testées : un modèle sans aucune mesure défensive, un modèle appliquant systématiquement un bruit sur le vecteur de prédiction, et notre modèle dynamique qui ajuste l'ajout de bruit en fonction de l'activité de l'attaquant et du budget alloué à la confidentialité.

1. Modèle sans défense : il sert de référence, fonctionnant sans mécanismes de protection de la vie privée, pour évaluer l'impact de l'absence de défense sur la sécurité des données.

2. Modèle avec bruit constant : ce modèle applique une perturbation constante au vecteur de prédiction, illustrant une approche de défense plus traditionnelle qui peut potentiellement dégrader la performance du modèle en termes de précision utilitaire.
3. Notre modèle adaptatif : il représente notre approche proposée qui optimise l'application du bruit basée sur une analyse dynamique des menaces et du budget de confidentialité disponible, visant à maintenir un équilibre optimal entre protection des données et efficacité du modèle.

Pour quantifier l'efficacité de chaque modèle, nous avons utilisé deux indicateurs principaux : la précision utilitaire du modèle, mesurant l'efficacité des prédictions dans des conditions normales, et la précision de l'inférence, évaluant la vulnérabilité du modèle aux attaques MIA. Les résultats, illustrés dans les Figures 4.2 et 4.3, montrent une performance remarquable de notre modèle adaptatif, qui parvient à réduire significativement le taux de réussite des attaques MIA tout en préservant une excellente précision utilitaire.

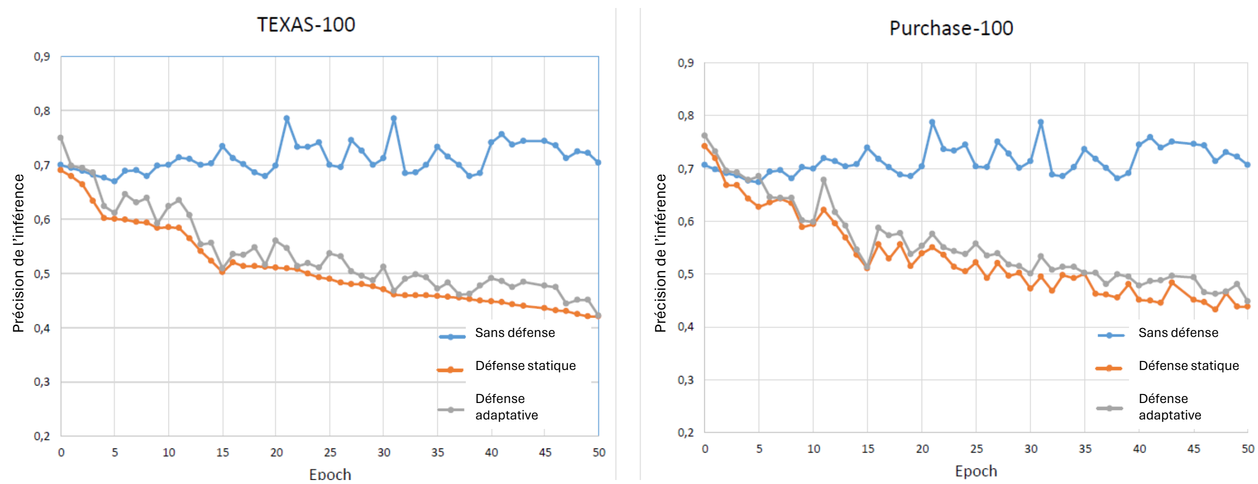

FIGURE 4.2 – Comparaison des mécanismes de défense et de leur impact sur la précision de l'inférence.

4.6.5 Limitations et pistes d'amélioration

Malgré ses avantages, notre méthode présente certaines limitations qui doivent être abordées pour améliorer la sécurité des modèles MLaaS. Premièrement, la complexité inhérente à la modélisation par la théorie des jeux peut poser des défis significatifs en termes de calcul et de mise en œuvre, en particulier dans des environnements à faibles ressources. De plus, bien que notre approche réduise l'impact sur la performance du modèle, elle n'élimine pas complètement le risque de dégradation lorsque le niveau de bruit nécessaire est élevé.

Pour surmonter ces défis, plusieurs pistes d'amélioration sont envisageables. L'optimisation des algorithmes de calcul des équilibres de Nash pourrait réduire leur complexité et leur coût, rendant notre méthode plus accessible et moins gourmande en ressources. En outre, une exploration plus détaillée des réponses du modèle face à différents types et intensités d'attaques pourrait permettre de peaufiner les stratégies de défense. Adapter dynamiquement les

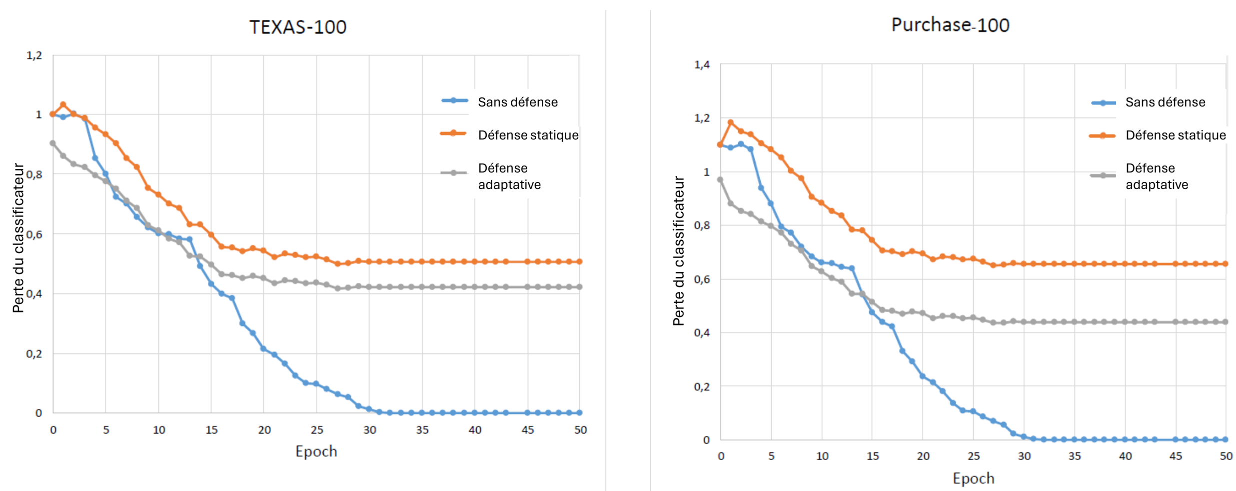

FIGURE 4.3 – Trajectoire de la perte de classification avec différents niveaux de mécanisme de défense sur divers jeux de données pendant l'entraînement.

paramètres de défense en fonction du contexte et de l'agressivité des attaques pourrait renforcer la résilience du système tout en maintenant une haute efficacité opérationnelle. Cette approche adaptative et sensible au contexte est essentielle pour les déploiements en conditions réelles, où les menaces évoluent constamment, garantissant ainsi une balance optimale entre sécurité et performance opérationnelle.

4.7 Conclusion

Dans ce chapitre, nous avons développé une nouvelle approche pour protéger les environnements d'apprentissage automatique en tant que service (MLaaS) contre les attaques par inférence d'appartenance. Notre méthode utilise l'ajout stratégique de bruit aux vecteurs de prédiction, soigneusement optimisé via un problème d'optimisation sous contraintes. Cette approche est spécialement conçue pour équilibrer de manière rigoureuse les exigences de confidentialité avec celles de l'utilité. Cette technique de défense est adaptative : elle évalue en continu le risque d'attaques par inférence et ajuste ses stratégies défensives pour préserver les performances des modèles MLaaS. L'adaptabilité de notre approche découle de la modélisation des interactions entre l'attaquant et la défense sous forme de jeu non-coopératif, utilisant les résultats pour dynamiser la solution et ajuster la stratégie de défense en fonction du risque d'attaque.

Les simulations ont validé l'efficacité de notre méthode, qui améliore significativement la confidentialité sans altérer substantiellement la précision des prédictions. Comparée aux méthodes de défense traditionnelles, notre approche démontre une capacité supérieure à s'adapter aux attaques.

Cependant, notre solution n'est pas sans limitations, particulièrement en termes de complexité computationnelle. Les travaux futurs devront explorer des moyens de réduire la complexité des calculs liés à la théorie des jeux et de combiner notre approche avec d'autres techniques de protection de la vie privée pour mieux équilibrer l'efficacité opérationnelle.

Chapitre 5

Mitigation des risques d'attaques par inférence d'attributs

5.1 Introduction

Ce chapitre approfondit notre analyse de la sécurité dans les environnements de "Machine Learning as a Service" (MLaaS), en mettant l'accent sur un risque souvent sous-estimé : les attaques par inférence d'attribut (AIA). Ces attaques exploitent des connaissances partielles sur les données d'entraînement pour inférer des attributs sensibles à partir des prédictions des modèles, une méthode proche de l'imputation de données mais exploitée de manière malveillante.

Les AIA représentent des risques significatifs, en particulier dans des secteurs sensibles comme la santé, où elles peuvent révéler des informations critiques sans accès direct aux données d'entraînement. Les techniques d'attaque, comme celles identifiées par Fredrikson *et al.* [55], y compris les attaques par inversion de modèles (MIAI) et les attaques de reconstruction d'instance typique (TIR), illustrent que les attaquants peuvent non seulement découvrir des attributs sensibles mais aussi reconstruire des instances représentatives de classes spécifiques, augmentant ainsi le risque de compromission de la confidentialité des données.

Dans les environnements MLaaS, où les modèles sont accessibles via des API qui ne révèlent ni leur fonctionnement interne ni leurs algorithmes d'entraînement, les adversaires peuvent utiliser des scores de prédiction et des informations partielles pour mener des attaques sophistiquées. Ces attaques exploitent les corrélations entre informations privées et publiques pour inférer des attributs sensibles, posant un défi majeur en matière de protection des données. Prenons l'exemple du secteur de la santé, riche en informations sensibles. Les modèles ML qui assistent les diagnostics ou prédisent les issues médicales peuvent involontairement révéler des informations personnelles de santé. Ainsi, par une attaque d'inversion de modèle, un adversaire peut inférer les antécédents médicaux des patients à partir des prédictions, sans accès aux noms des patients. Ceci peut mener à des utilisations abusives telles que des pratiques d'assurance discriminatoires.

Face à ces risques, les méthodes traditionnelles de préservation de la confidentialité telles que la confidentialité différentielle se révèlent souvent insuffisantes. Bien que fournissant une garantie contre l'identification des individus dans un ensemble de données, elles peinent à équilibrer l'utilité des données et la confidentialité, réduisant ainsi la précision des modèles et compliquant leur mise en œuvre pratique [48, 87, 88].

Dans ce contexte, nous explorons des stratégies alternatives, notamment des techniques de masquage des scores et la construction d'exemples adverses, pour développer des défenses stratégiques contre les attaques par inférence [164, 43, 91]. Nous proposons en particulier une solution pratique qui perturbe le vecteur de scores de prédiction pour contrer efficacement les attaques par inférence d'attribut, tout en maintenant la performance du modèle.

5.2 Travaux connexes

Dans cette section, nous explorons le paysage entourant les attaques par inférence d'attributs sensibles. Nous définissons ces attaques, leurs différentes stratégies, ainsi que les méthodes utilisées pour les contrer.

5.2.1 Attaques par inférence d'attributs

Des études récentes [100, 59, 196, 55] ont révélé que les utilisateurs sont exposés aux attaques par inférence d'attributs, qui visent à exploiter les modèles d'apprentissage machine pour exposer des informations sensibles en utilisant des données publiquement disponibles. Ainsi, les attaquants peuvent déduire des attributs sensibles des individus, y compris, mais sans s'y limiter, leur genre, leur localisation ou leurs opinions politiques. Le problème principal provient de la tendance des modèles d'apprentissage machine à divulguer involontairement des informations sensibles lors du processus de prédiction. Les entités malveillantes sont ainsi capables d'extraire des données privées ou sensibles à partir de sources facilement disponibles, telles que les prédictions des modèles. Cette capacité à inférer des détails sur les données d'entraînement ou les entrées, qui resteraient autrement cachés sans l'intervention du modèle, pose des défis significatifs en matière de confidentialité.

Les attaques par inférence d'attributs sensibles peuvent être catégorisées en deux types principaux : les attaques basées sur l'imputation et celles basées sur la représentation. Chacune de ces deux classes d'attaques utilise des stratégies, des hypothèses et des techniques distinctes pour cibler et exploiter les vulnérabilités inhérentes aux modèles d'apprentissage machine.

5.2.1.1 Attaques basées sur l'imputation

Ces attaques exploitent des attributs non sensibles en utilisant les prédictions du modèle et des données contextuelles, comme la probabilité marginale sur un attribut sensible et la matrice de confusion. Le but est d'utiliser l'inférence statistique pour extrapoler ou imputer des informations cachées ou absentes. Jayaraman et Evans [88] ont contesté l'efficacité supposée des attaques d'imputation standard en boîte noire, montrant qu'elles équivalent souvent à une simple imputation de données. Ils ont souligné la différence entre de véritables risques de confidentialité et de simples déductions statistiques. Fredrikson *et al.* [55] ont développé une méthode qui évalue la probabilité que les attributs confidentiels soient corrects, basée sur les réactions d'un classificateur à des entrées spécifiquement conçues. Yeom *et al.* [208] ont proposé diverses techniques d'inférence, en assumant différentes distributions pour l'attribut confidentiel. Ensuite, *et al.* [119] ont observé que la précision d'un modèle est optimale lorsque les attributs sensibles des données d'entraînement correspondent exactement aux prédictions. Ils ont introduit des attaques telles que "Confidence only" et "Label-only" pour exploiter les prédictions du modèle et révéler des attributs sensibles.

5.2.1.2 Attaques basées sur la représentation

Les attaques basées sur la représentation se distinguent par leur capacité à exploiter efficacement les variations dans les sorties des couches intermédiaires ou des prédictions finales des modèles d'apprentissage machine. Ces attaques sont extrêmement sensibles aux changements dans les valeurs attributaires, exploitant par exemple les distinctions notables dans les prédictions associées aux classifications de genre.

Des études telles que celles menées par Song *et al.* [171] et Mahajan *et al.* [115] s'appuient sur l'hypothèse que les données d'entraînement n'intègrent pas directement des attributs sensibles. Ces recherches développent des approches adverses pour décomposer et analyser les réponses du modèle principal, révélant ainsi des attributs cachés, souvent avec l'utilisation d'un seuil de classification arbitraire, typiquement fixé à 0,5.

D'un autre côté, Malekzadeh *et al.* [116] ont proposé une stratégie utilisant une fonction de perte spécialement conçue pour intégrer délibérément l'attribut sensible dans les sorties du modèle. Cette méthode facilite l'extraction de ces attributs pendant l'inférence et suggère une possibilité de manipulation intentionnelle par les concepteurs du modèle, similaire à la création d'une "porte dérobée" qui pourrait être exploitée ultérieurement pour extraire des informations sensibles.

5.2.2 Stratégies d'atténuation pour les attaques par inférence d'attributs sensibles

Les stratégies d'atténuation contre les attaques par inférence d'attributs sensibles dans les modèles d'apprentissage machine (ML) se sont raffinées pour répondre à l'évolution constante du paysage des menaces.

Les cadres théoriques basés sur la théorie des jeux fournissent une base solide pour concevoir des mesures de confidentialité, bien qu'ils soient souvent associés à une lourde charge computationnelle. Cette observation a été soulignée par Shokri *et al.* [168]. Pour les implémentations pratiques, le mappage probabiliste quantifié (QPM) proposé par Salamatian *et al.* [154] simplifie les modèles de défense, optimisant ainsi l'efficacité opérationnelle.

En cryptographie, les techniques comme le chiffrement homomorphe (HE) et le chiffrement totalement homomorphe (FHE) permettent des opérations sur des données chiffrées, assurant ainsi que les serveurs de cloud computing peuvent traiter les informations sans compromettre la confidentialité [148, 31]. Ces technologies offrent des perspectives intéressantes pour le traitement sécurisé des données, mais elles présentent des défis non négligeables, notamment une surcharge computationnelle significative qui peut augmenter le temps de traitement et la consommation d'énergie. Cette lourdeur peut restreindre leur application à des scénarios en temps réel ou nécessitant une réponse rapide, limitant ainsi leur intégration dans des environnements de MLaaS.

La confidentialité différentielle (DP) représente une autre stratégie bien établie, introduisant du bruit statistique dans les données pour masquer les contributions individuelles et protéger la vie privée [88, 5]. Bien que mathématiquement robuste, la DP peut diminuer l'utilité des données, compromettant la qualité des analyses en particulier lorsque la précision est essentielle [93]. Ce dilemme met en lumière une problématique centrale dans l'utilisation des données préservant la confidentialité : trouver un juste équilibre entre la sécurité rigoureuse des données personnelles et la nécessité de préserver leur utilité analytique.

Ces méthodes, prises ensemble, composent un arsenal défensif diversifié, capable de marier la protection de la vie privée avec les impératifs pratiques des applications de l'apprentissage automatique. Toutefois, chacune de ces approches présente des limites qui pourraient compromettre leur efficacité. Dans ce qui suit, nous travaillons à développer une solution plus robuste et adaptable, qui peut non seulement répondre de manière dynamique aux menaces émergentes, mais aussi intégrer de manière plus harmonieuse les contraintes opérationnelles des systèmes de MLaaS.

5.3 Formulation du problème

Dans notre formulation du problème, nous définissons clairement les rôles de trois entités critiques : le modèle d'apprentissage automatique, l'attaquant et le mécanisme de défense.

- Le *modèle d'apprentissage automatique* fonctionne comme le système central de notre étude. Il est rigoureusement entraîné sur des données utilisateur avec pour seul objectif de fournir des prédictions précises et efficaces. Ce modèle, de par son exposition et son accès à d'importantes quantités de données utilisateur, devient involontairement une cible privilégiée pour les entités malveillantes.
- L'*attaquant*, une entité malveillante avec une mission unique qui est d'exploiter le modèle d'apprentissage automatique. Son objectif est d'utiliser les prédictions du modèle pour découvrir des attributs privés et potentiellement sensibles des utilisateurs.
- Le *mécanisme de défense* apparaît comme le défenseur du système. Il est méticuleusement conçu pour contrer efficacement l'attaquant en modifiant stratégiquement la prédiction des scores. Toutefois, il est essentiel de s'assurer que, tandis que l'attaquant est induit en erreur, l'utilité de base et l'efficacité du modèle restent intactes et sans dommage.

5.3.1 Modèle du fournisseur de services d'apprentissage automatique

Un modèle d'apprentissage automatique est couramment vu comme une fonction déterministe :

$$f : X \to Y \tag{5.1}$$

L'entrée de cette fonction est un vecteur dimensionnel $x = [x_1, x_2, \dots, x_d] \in X = \mathbb{R}^d$, représentant d attributs d'entrée non sensibles. Pour les tâches de régression, l'espace de sortie Y est défini comme l'ensemble des nombres réels, $Y = \mathbb{R}$. Dans ce travail, nous nous intéressons aux modèles de classification, où l'espace de sortie est distinct.

Dans le contexte de la classification, la fonction $f : \mathbb{R}^d \to Y$ associe d'abord le vecteur d'entrée x à un ensemble de scores de confiance $\upsilon = [\upsilon_1, \upsilon_2, \dots, \upsilon_m]$, où chaque υ_j représente la confiance du modèle à attribuer l'étiquette de classe j^{me} à x. L'étiquette de classe prédite,

y, est ensuite déterminée en sélectionnant l'étiquette associée au score de confiance le plus élevé dans v, formellement représentée comme $y = \arg\max_j v_j$, où $j \in \{1, \ldots, m\}$. Ici, Y est l'ensemble des étiquettes possibles $\{y_1, y_2, \ldots, y_m\}$.

Les paramètres du modèle, notés θ, sont affinés de manière itérative en fonction du gradient de la fonction de perte, qui quantifie l'écart entre les prédictions du modèle $f(x;\theta)$ et les étiquettes réelles. L'entraînement utilise l'ensemble de données $\mathcal{D} \subseteq X \times Y$, visant à optimiser θ afin que $f(x;\theta)$ puisse mapper avec précision les entrées x à leurs étiquettes correspondantes. En conséquence, pour toute entrée x, la prédiction du modèle est donnée comme $f(x;\theta)$, où θ sont les paramètres affinés par l'entraînement pour minimiser idéalement la fonction de perte.

Nous considérons p comme un attribut sensible appartenant à l'ensemble P. Un individu, lié à un enregistrement de données dans notre ensemble d'entraînement, vise à garder cet attribut p confidentiel. Nous supposons que cet attribut p peut prendre k valeurs distinctes et nos attributs d'entrée $x \in X$ sont considérés comme non sensibles. P représente l'ensemble complet de toutes les valeurs possibles que l'attribut sensible peut prendre. En conséquence, un enregistrement de données est décrite comme $z = (x, p, y)$, où x représente les attributs non sensibles, p l'attribut sensible, et y correspond à l'étiquette de classification. L'association est définie comme $(x, p) \in X \times P$.

Bien que les données soient généralement considérées comme "publiques" aux fins de la formation des modèles d'apprentissage automatique, il existe des scénarios dans lesquels les "attributs sensibles" peuvent être déduits de celles-ci. D'une part, ces attributs sensibles pourraient être utilisés par le modèle d'apprentissage automatique pour améliorer la précision des prédictions, soulevant la possibilité que le modèle conserve une certaine mémoire de ces informations, qu'un adversaire pourrait exploiter en cherchant des traces dans les prédictions du modèle. D'autre part, le modèle d'apprentissage automatique n'a peut-être pas directement rencontré ou utilisé cet attribut sensible ; cependant, il pourrait encore y avoir un lien entre cet attribut et les données publiques d'un enregistrement, ce qui pourrait permettre à un adversaire de faire des inférences. Les propriétaires de données partagent leurs informations pour des applications d'apprentissage automatique avec l'attente que de tels détails sensibles restent dissimulés, s'appuyant sur l'engagement à la confidentialité de ces attributs. Une fois le modèle formé sur l'ensemble de données $\mathcal{D} = \{(x_i, y_i), i = 1..n\}$, il est déployé comme une interface de programmation d'applications (API).

5.3.2 Attaquant

Dans notre scénario, nous introduisons la présence d'une entité adverse $\mathfrak{A}$ qui interagit avec le MLaaS en envoyant une série de requêtes et, en retour, reçoit les vecteurs de scores de confiance associés. Par ailleurs, cet adversaire a préalablement formé un modèle de classification f_{adv} en utilisant des techniques d'apprentissage supervisé. La formation de ce classificateur d'inférence a été facilitée par des données qu'il a acquise, souvent auprès d'utilisateurs qui ont divulgué involontairement leurs attributs sensibles. Nous supposons que l'adversaire a pu avoir accès à une quantité significative d'informations. Plus précisément, il possède tout ou une partie des capacités/connaissances suivantes :

— Capacité d'interagir avec le modèle cible, décrite comme une boîte noire. Plus précisément, l'adversaire peut soumettre des entrées $x = [x_1, x_2, \ldots, x_d]$ pour obtenir

l'étiquette de classe associée y'.
— Connaissance des scores de confiance du modèle cible sur m étiquettes de classe distinctes, v.
— Connaissance d'informations complètes ou sélectives sur les attributs non sensibles, tandis que l'aspect sensible reste dissimulé.
— Disponibilité d'un ensemble de données supplémentaire, $\mathcal{D}_{aux}$, provenant d'une distribution de données similaire à celle de $\mathcal{D}$, sur laquelle f_{target} a été formé. Notamment, $\mathcal{D}$ et $\mathcal{D}_{aux}$ ne partagent aucune entrée commune, c'est-à-dire $\mathcal{D} \cap \mathcal{D}_{aux} = \emptyset$.
— Connaissance de l'ensemble complet (l) des résultats potentiels pour l'attribut sensible p.

Bien que l'adversaire n'ait qu'un accès en boîte noire au modèle réel, il peut utiliser les vecteurs de scores de confiance reçus pour guider son classificateur pré-entraîné, f_{adv}. Ce classificateur adverse apprend à partir d'un ensemble de données auxiliaires, $\mathcal{D}_{aux}$, qui est censé partager la même distribution que $\mathcal{D}$. L'ensemble de données $\mathcal{D}_{aux}$ comprend un ensemble de triplets $\{x_i, p_i, y_i\}_i$, représentant des données publiques, l'attribut sensible et la prédiction du modèle.

L'objectif principal est de découvrir les attributs sensibles cachés des utilisateurs. Par conséquent, l'adversaire construit le modèle d'inférence f_{adv} en utilisant $\mathcal{D}_{aux}$, établissant la relation $f_{adv} : (x, f(x)) \rightarrow p$.

Une visualisation détaillée de ces interactions est fournie par la Figure 5.1, illustrant l'étendue des efforts de l'adversaire pour déduire des informations sensibles par le biais de requêtes systématiques.

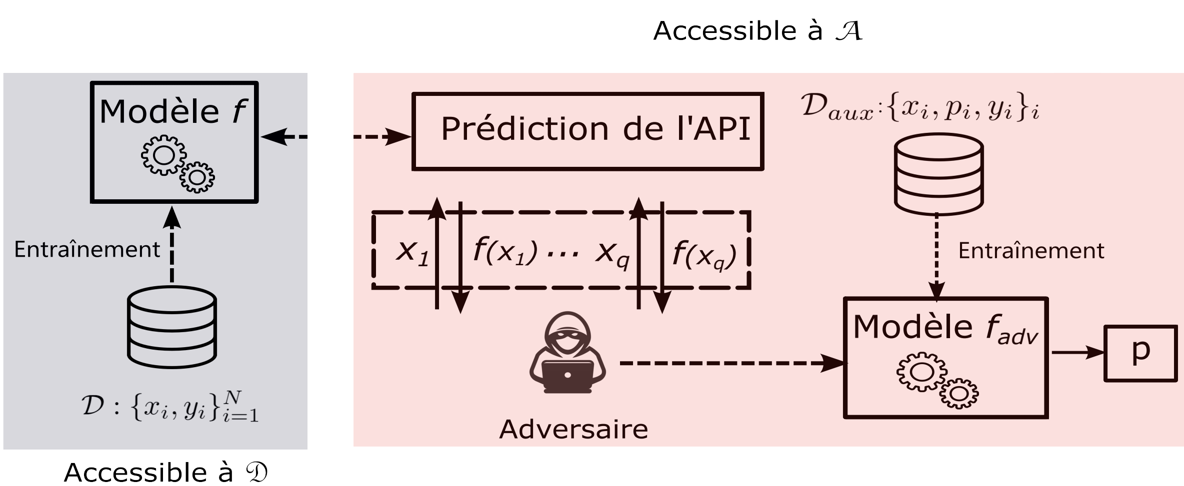

FIGURE 5.1 – Attaque par inférence d'attribut sensible via MLaaS.

L'efficacité du modèle d'inférence de l'attaquant découle de sa capacité à identifier et exploiter des motifs de prédiction spécifiques et des biais inhérents à f_{target}, qui sont indicatifs des valeurs des attributs sensibles.

5.4 Modèle de défense en deux étapes contre les AIA

Le défi principal dans l'identification des vecteurs de bruit pour perturber les vecteurs de scores dans les modèles d'apprentissage automatique réside dans l'explosion combinatoire des paramètres. Trouver un vecteur de bruit δ à ajouter à un autre vecteur v pour répondre à des exigences spécifiques augmente exponentiellement le nombre de possibilités, rendant la mise en œuvre d'une solution d'optimisation en temps réel extrêmement difficile. Par conséquent, nous envisageons de catégoriser les vecteurs de bruit. Supposons qu'il y ait k valeurs possibles pour l'attribut sensible ; nous aurons alors k catégories de vecteurs de bruit, chacune conçue pour induire en erreur le classificateur de l'adversaire en prédisant une valeur différente de l'attribut sensible.

Ces vecteurs de bruit seront sélectionnés aléatoirement selon un mécanisme $\mathcal{N}$ basé sur certaines probabilités, sous la contrainte que le vecteur de score bruité v' prédise toujours la même classe, arg max$v$ = arg maxv'. Le mécanisme aura k différentes probabilités pour choisir un vecteur de bruit dans chaque catégorie. Nous construisons notre mécanisme de telle sorte que notre stratégie de défense déconcerte l'adversaire mais maintienne également la précision prédictive du vecteur de score.

5.4.1 Étape 1 : détermination du vecteur de bruit δ_i

L'objectif de cette première étape est d'identifier les manipulations minimales sur le vecteur de scores de confiance afin de tromper le modèle d'inférence de l'attaquant, et qu'il prédise une valeur spécifique de l'attribut sensible p. Cette formulation reflète étroitement un problème d'apprentissage automatique adversarial (MLA), où l'objectif principal est de modifier subtilement les entrées pour induire des erreurs de classification.

Les contraintes de notre modèle assurent que le vecteur de score modifié reste une distribution de probabilité valide. Cela signifie maintenir la somme totale des éléments du vecteur égale à 1 et chaque élément compris entre 0 et 1. Le problème d'optimisation pour induire le classificateur de l'adversaire à prédire la valeur i pour l'attribut sensible p est donc formulé comme suit :

$$
\begin{aligned}
& \delta^i = \arg\min_{\delta} \|\delta\|_2 \\
\text{s.t} \quad & f_{\text{adv}}(x, v + \delta^i) = i, \\
& \arg\max_j v = \arg\max_j v + \delta^i, \\
& \sum_{j=1}^{m} (v_j + \delta^i_j) = 1, \\
& 0 \leq (v_j + \delta^i_j) \leq 1, \quad \forall j = 1, \ldots, m.
\end{aligned}
\tag{5.2}
$$

Chaque valeur i de l'attribut sensible p correspondra à la résolution d'un problème d'optimisation distinct, totalisant k problèmes à résoudre pour les k valeurs possibles de l'attribut.

Les algorithmes conventionnels d'apprentissage machine adversarial se heurtent fréquemment à des difficultés lorsqu'ils sont confrontés à des problèmes d'optimisation comportant des contraintes complexes. La difficulté principale réside dans la nécessité de maintenir le vecteur de score perturbé comme une distribution de probabilité valide, ce qui peut réduire l'efficacité des méthodes standards. Cette limitation met en évidence le besoin de solutions innovantes et spécialement adaptées pour gérer ce genre de complexités.

Pour répondre à ce défi, nous adoptons une méthode développée par Papernot *et al.* [137], appelée Jacobian-based Saliency Map Attack (JSMA). JSMA est une technique d'attaque adversariale qui cible et perturbe un sous-ensemble restreint de caractéristiques d'entrée pour influencer significativement la classification résultante. Originellement, JSMA se concentre sur la manipulation de caractéristiques basée sur leur saillance, déterminée par l'impact de chaque caractéristique sur la classification visée. Toutefois, dans sa forme standard, JSMA ne répond pas directement à notre exigence de maintenir le vecteur de score en tant que distribution de probabilité.

Pour pallier cette limitation, nous avons développé une variante de JSMA spécifiquement adaptée à notre contexte d'optimisation avec contraintes. Cette version modifiée intègre explicitement la contrainte de maintenir une distribution de probabilité valide pour le vecteur de score, après normalisation. En ajustant la méthodologie JSMA pour respecter cette contrainte, nous pouvons explorer plus efficacement l'espace des solutions possibles tout en adhérant aux exigences de notre problème. Cette adaptation permet une manipulation plus précise et conforme des scores, qui garantit que les perturbations générées restent dans les limites de probabilités valides et sont donc plus susceptibles d'induire en erreur les systèmes adverses sans compromettre la structure du modèle.

Nous introduisons donc un nouvel algorithme, NOISY (Noise Optimizer Inference Sensitive Yielder), conçu pour répondre à notre problème d'optimisation. La tâche principale de NOISY est de trouver et d'appliquer méticuleusement un vecteur de perturbation optimal δ^i au vecteur de score de confiance existant v. Ce vecteur est conçu non seulement pour induire une erreur de prédiction spécifique dans le classificateur de l'adversaire, visant la valeur i^{me} de l'attribut privé, mais aussi pour garantir que le vecteur résultant, une fois ajusté, respecte les contraintes d'être une distribution de probabilité valide. L'algorithme fonctionne à travers une série d'ajustements itératifs, équilibrant soigneusement l'objectif de provoquer la mauvaise classification souhaitée avec l'impératif de maintenir l'intégrité de la prédiction originale. Cela est réalisé en respectant une contrainte rigoureuse : le vecteur de score perturbé doit continuer à prédire la même classe qu'il le faisait avant la perturbation. Par cette approche, NOISY vise à réaliser une manipulation délicate des scores de confiance, garantissant que, bien que l'exactitude du classificateur de l'adversaire soit délibérément compromise, l'utilité et la validité du vecteur de score demeurent préservées.

Les principales étapes de cet algorithme sont les suivantes :

1. **Calcul de la dérivée de la fonction de perte :** cette étape vise à déterminer où une perturbation peut le plus conduire à une erreur de classification.
2. **Optimisation avec contrainte :** utiliser une méthode d'optimisation avec contraintes pour déterminer δ^i, le vecteur de bruit qui minimise la perturbation tout en respectant les contraintes prédéfinies. Cela assure que les modifications apportées sont optimales dans le contexte des limites imposées.
3. **Ajustements itératifs et vérification :** répéter les ajustements de δ pour maintenir l'intégrité du vecteur de score. Vérifier après chaque ajustement si le vecteur de score bruité $v + \delta$, une fois normalisé, prédit correctement l'attribut cible i. Cette étape

garantit que les modifications restent conformes à l'objectif de classification tout en préservant les propriétés de la distribution de probabilité.

Algorithme 2 NOISY : Noise Optimizer Inference Sensitive Yielder

Prérequis: Vecteur de scores de confiance υ, classificateur f_{def}, valeur de l'attribut cible i, pas α, nombre maximal d'itérations maxiter

Assurer: Vecteur de perturbation δ^i

```
Initialiser le nombre d'itérations t = 0
Initialiser δ^i = 0, υ' = υ
Tant que f_def(softmax(υ' + δ^i)) ≠ i) et t < maxiter faire
    Identifier les entrées à modifier en fonction de leur saillance et des contraintes
    e_inc = argmax_{j { ∂f_def/∂x_j (υ') | δ^i_j = 0 }}
    e_dec = argmax_{j { −∂f_def/∂x_j (υ') | δ^i_j > 0 }}
    Modifier les entrées en fonction des contraintes
    δ_{e_inc} = clip(δ^i_{e_inc} + α, 0, 1)
    δ_{e_dec} = clip(δ^i_{e_dec} − α, 0, 1)
    Ajuster δ^i pour maintenir la classe prédite originale
    Si argmax_j(υ_j) ≠ argmax_j(υ_j + δ^i_j) alors
        Réduire l'amplitude de δ^i_{e_inc} et δ^i_{e_dec} pour satisfaire la contrainte
    fin Si
    Ajuster δ^i pour que υ' + δ^i reste une distribution de probabilité valide
    total = Σ_{j=1}^{m} (υ'_j + δ^i_j)
    δ^i = δ^i/total
    Mettre à jour υ' = (υ + δ^i)/total
    t = t + 1
    Mettre à jour υ' = υ + δ^i
fin Tant que
return δ^i
```

Notre algorithme commence avec un vecteur de perturbation δ^i initialisé à zéro et utilise une carte de saillance pour identifier les éléments à modifier dans l'intervalle $[0, 1]$, en utilisant un pas α. Il ajuste le vecteur δ^i pour garantir que le vecteur de scores de confiance bruité, $\upsilon' + \delta^i$, reste une distribution de probabilité valide. Le processus est itératif, se concentrant sur la modification du vecteur de scores de confiance pour induire l'adversaire à prédire une classe cible spécifique, tout en veillant à ce que la classe de prédiction originale du vecteur soit préservée.

5.4.2 Étape 2 : détermination de $\mathcal{N}^*$

Suite à l'étape 1, nous disposons de k vecteurs de perturbation, désignés $\delta^1, \ldots, \delta^k$. L'objectif de l'étape 2 est de développer une distribution de probabilité à partir de ces vecteurs, visant à maximiser l'incertitude dans les choix de l'adversaire en créant une distribution uniforme ("plate") à travers ces vecteurs.

Le deuxième problème d'optimisation vise à élaborer un mécanisme de sélection du vecteur de bruit qui permet d'optimiser l'entropie de la distribution tout en maintenant une prédiction de classe constante. La formulation de ce problème est la suivante :

$$\begin{aligned} \underset{\mu}{\text{maximize}} \quad & -\sum_{i=1}^{k} \mu_i \log \mu_i \\ \text{s.t} \quad & \underset{j}{\text{argmax}}\ \upsilon' = \underset{j}{\text{argmax}}\ \upsilon, \\ & \sum_{i=1}^{k} \mu_i = 1, \\ & \mu_i \geq 0,\ \forall i \in \{1, \ldots, k\}. \end{aligned} \tag{5.3}$$

À travers cette modélisation, nous cherchons à définir une distribution de probabilité optimale à travers plusieurs objectifs et contraintes. Le but principal est de maximiser l'entropie de la distribution, représentée par $-\sum_{i=1}^{k} \mu_i \log \mu_i$. Maximiser l'entropie promeut une distribution uniforme, augmentant l'incertitude et réduisant la prévisibilité, ce qui est souvent souhaitable pour masquer les décisions stratégiques.
La contrainte $\underset{j}{\text{argmax}}\ \upsilon' = \underset{j}{\text{argmax}}\ \upsilon$ s'assure que malgré les modifications apportées à la distribution μ, la prédiction de classe la plus probable reste inchangée.
Pour s'assurer que μ forme une distribution de probabilité légitime, la somme de toutes les probabilités μ_i est contrainte à être égale à 1, comme spécifié par $\sum_{i=1}^{k} \mu_i = 1$. De plus, chaque probabilité individuelle μ_i doit être non-négative, ce que garantit la contrainte $\mu_i \geq 0,\ \forall i \in \{1, \ldots, k\}$.

Pour résoudre ce problème d'optimisation, nous appliquons les conditions de Karush-Kuhn-Tucker (KKT), une méthode fondamentale pour résoudre les problèmes d'optimisation avec contraintes. Initialement, la *faisabilité primale* garantit que notre solution respecte toutes les contraintes établies, maintenant l'intégrité de la formulation de notre problème. Ensuite, la *stationnarité* est atteinte lorsque nous identifions les multiplicateurs de Lagrange appropriés — λ pour les contraintes d'égalité et ν_i pour les contraintes d'inégalité — tels que le gradient de la Lagrangienne par rapport à μ disparaisse au point optimal. Cela garantit que notre solution est non seulement faisable mais aussi optimale, alignée sur notre fonction objectif et les contraintes. La *faisabilité duale* exige que les multiplicateurs de Lagrange associés à nos contraintes d'inégalité soient non-négatifs, une condition qui assure que notre solution se situe dans l'espace de solutions permis. Enfin, la *slackness complémentaire* insiste sur le fait que pour chaque contrainte d'inégalité, le produit de son multiplicateur de Lagrange et la contrainte elle-même soit nul au point optimal, mélangeant les frontières entre la faisabilité et l'optimalité. Ensemble, ces conditions nous guident méticuleusement vers une solution qui est non seulement dans les limites mais aussi optimale, garantissant un respect rigoureux à la fois de la structure de notre problème et de ses contraintes inhérentes.

$$L(\mu, \lambda, \nu) = -\sum_{i=1}^{k} \mu_i \log \mu_i + \lambda \left(\sum_{i=1}^{k} \mu_i - 1 \right) + \sum_{i=1}^{k} \nu_i (-\mu_i) \tag{5.4}$$

Notre stratégie d'optimisation utilise la Lagrangienne $L(\mu, \lambda, \nu)$ pour maximiser l'entropie dans la distribution du vecteur de perturbation μ, sous des contraintes spécifiques. L'objectif, $-\sum_{i=1}^{k} \mu_i \log \mu_i$, cherche une distribution uniforme à travers k vecteurs, renforçant l'imprévisibilité. Le terme $\lambda(\sum_{i=1}^{k} \mu_i - 1)$ assure la normalisation de la distribution, tandis que $\sum_{i=1}^{k} \nu_i(-\mu_i)$ impose la non-négativité sur chaque μ_i, avec λ et ν_i comme multiplicateurs de Lagrange pour les contraintes d'égalité et d'inégalité, respectivement.

5.4.2.1 Interprétation pratique de $\mathcal{N}^*$

Le mécanisme $\mathcal{N}$ joue un rôle pivot dans notre cadre d'optimisation, servant de cœur stratégique pour sélectionner le vecteur de bruit optimal sous des contraintes strictement définies. Ce mécanisme est conçu pour naviguer à travers le paysage complexe des perturbations adverses, visant à identifier une stratégie de perturbation qui non seulement respecte les contraintes opérationnelles, telles que le maintien de la prédiction par le classificateur, mais introduit également un niveau d'indétermination et de diversité dans les exemples adverses générés.

En termes pratiques, $\mathcal{N}$ détermine la distribution de probabilité à travers divers vecteurs de bruit $(\delta^1, \ldots, \delta^k)$ de manière à conserver la cohérence de la sortie du classificateur, tout en rendant les actions de l'adversaire moins discernables. En procédant ainsi, $\mathcal{N}$ augmente efficacement la difficulté pour les mécanismes de défense de prévoir et de contrer ces perturbations adverses, sécurisant un avantage stratégique.

Le mécanisme conçu pour maximiser l'entropie dans la distribution des vecteurs de perturbation a pour but d'induire de l'incertitude dans le choix de l'adversaire. Cette incertitude est cruciale pour garantir que tout vecteur de bruit sélectionné maintienne la prédiction de la classe par le classificateur.

Ainsi, $\mathcal{N}$ joue un rôle clé dans notre modèle de défense, en assurant que la diversité et l'indétermination sont intégrées dans les perturbations adverses tout en préservant l'intégrité et la validité des prédictions du classificateur. Cette approche cherche à compliquer la tâche de l'adversaire en rendant ses prédictions moins fiables, tout en maintenant l'utilité et la validité du vecteur de score original, soulignant ainsi un équilibre entre sécurité et fonctionnalité.

Ce cadre d'optimisation sophistiqué et stratégiquement orienté vise à outiller les systèmes de MLaaS avec les moyens nécessaires pour contrer efficacement les attaques par inférence d'attributs tout en minimisant l'impact sur la performance du modèle.

5.5 Expérimentation

Dans cette section, nous discutons de notre cadre expérimental utilisé pour valider l'efficacité de notre mécanisme de sécurité proposé contre les attaques par inférence d'attributs. L'objectif principal de notre enquête est de démontrer que notre modèle peut tromper de manière significative le classificateur de l'attaquant, protégeant ainsi les attributs sensibles déductibles du vecteur de score. En même temps, nous visons à préserver l'utilité inhérente du vecteur de score pour des fins légitimes. Cette double réussite est facilitée par des perturbations stratégiques introduites dans les scores de confiance.

5.5.1 Ensemble de données et configuration

Texas-100X [86] : le jeu de données que nous utilisons, appelé Texas-100X, sert de version étendue du jeu de données hospitalières Texas-100, introduit précédemment par Shokri *et al.* [167]. Chaque entrée de ce jeu de données fournit des détails démographiques complets des patients — allant de l'âge, du genre et de l'ethnicité aux données médicales nuancées comme la durée des séjours à l'hôpital, le mode d'admission, les raisons diagnostiques, les résultats des patients, les frais encourus et les interventions chirurgicales principales. L'objectif pour ce jeu de données est d'anticiper une des 100 interventions chirurgicales possibles, basées sur les dossiers de santé individuels.

Census19 [1] : le jeu de données Census19 est une extension moderne du célèbre jeu de données Adult [8], dérivé des données du recensement de 1994. Tandis que l'original comprenait environ 48 000 enregistrements avec 14 attributs, la version Census19 provient de la base de données du Bureau du recensement des États-Unis de 2019, offrant 1 676 013 entrées avec 12 attributs pivots. Ces enregistrements, organisés en fonction des zones d'utilisation microdonnées publiques (PUMAs), capturent les aspects démographiques clés des résidents américains : âge, genre, race, statut marital, éducation, occupation, heures de travail, pays d'origine et certains indicateurs de handicap. Le principal défi de classification avec Census19 est de déterminer si le revenu annuel d'un individu dépasse 90 000 $, un chiffre ajusté à l'inflation par rapport au seuil de 50 000 $ du jeu de données Adult.

Dans nos évaluations impliquant à la fois Texas-100X et Census19, nous sélectionnons au hasard 50 000 entrées pour établir le jeu de données d'entraînement et l'utilisons pour entraîner un réseau de neurones à deux couches. De plus, nous isolons 25 000 enregistrements distincts des données restantes pour constituer le jeu de données de test, en veillant à ce qu'il n'y ait pas de chevauchement entre les jeux de données d'entraînement et de test.

Le réseau de neurones utilisé pour notre modèle de défense est structuré comme suit :

- **Couche d'entrée** : configurée pour correspondre à la dimensionalité de l'espace des caractéristiques dans les jeux de données Texas-100X et Census19. Cela garantit que le réseau peut traiter chaque attribut d'entrée sans perte d'information.
- **Couches cachées** : comprend plusieurs couches pour améliorer la capacité du modèle à capturer les relations non linéaires dans les données. Chaque couche est équipée d'une fonction d'activation ReLU pour introduire de la non-linéarité, facilitant la formation de frontières de décision complexes essentielles pour une défense efficace.
- **Couche de sortie** : la dernière couche est conçue pour produire le vecteur de score perturbé. La dimensionalité de cette couche correspond au nombre de classes dans le jeu de données, avec une fonction d'activation softmax appliquée pour convertir la sortie du réseau en une distribution de probabilité sur les classes potentielles.

5.5.2 Intégration du mécanisme de défense

Le cœur de notre stratégie de défense implique le mécanisme de détermination $\mathcal{N}^*$, qui introduit dynamiquement des perturbations dans le vecteur de scores de confiance. Cette architecture de réseau neuronal est essentielle pour évaluer l'impact de telles perturbations, permettant des ajustements en temps réel pour garantir que le vecteur perturbé trompe le classificateur de l'attaquant tout en préservant l'intégrité et l'utilité du vecteur de score original.

Pour implémenter le mécanisme $\mathcal{N}^*$, le réseau est entraîné sur des données perturbées de manière adversariale ainsi que sur des données propres, optimisant deux objectifs principaux : minimiser le taux de réussite des attaques par inférence d'attributs et maintenir une haute précision sur les tâches de classification légitimes. Ce régime d'entraînement à double objectif est essentiel pour renforcer le modèle de défense contre des stratégies adverses sophistiquées.

5.5.3 Formation et évaluation

Le modèle subit un entraînement rigoureux en utilisant un jeu de données sélectionné qui amalgames des échantillons à la fois de Texas-100X et de Census19, assurant une exposition complète à des représentations diverses de données. Le processus de formation utilise une fonction de perte d'entropie croisée, efficace pour les tâches de classification et facilitant l'optimisation des poids du réseau dans le contexte de nos objectifs de défense.

5.5.4 Métriques d'évaluation

Dans notre évaluation, nous visons à effectuer une comparaison complète de notre modèle en deux étapes (Défense adverse à deux étapes) contre deux techniques éprouvées de préservation de la vie privée : la confidentialité différentielle partielle (LDP) [9] et la k-Anonymat [216]. Ces méthodes sont reconnues pour leur capacité à atténuer les risques associés aux attaques par inférence d'attributs sensibles dans les environnements MLaaS, chacune utilisant des mécanismes distincts qui impactent l'utilité des données et l'efficacité du traitement de manière unique.

Notre modèle en deux étapes tire parti d'exemples adverses sophistiqués et d'un mécanisme de sélection stratégique pour préserver la vie privée contre les attaques par inférence d'attributs (AIA). Il sera évalué parallèlement à la confidentialité différentielle partielle (LDP) plutôt qu'à la confidentialité différentielle standard en raison de l'adéquation de la LDP pour les environnements où l'agrégation centrale des données n'est pas faisable ou où la confiance dans un conservateur central est limitée. La LDP applique un bruit statistique contrôlé directement à la source des données, masquant les contributions individuelles avant que l'agrégation des données ne se produise. Cette méthode permet une plus grande assurance de la vie privée directement sur l'appareil de l'utilisateur sans nécessiter de confiance dans le traitement de leurs données par le serveur central. Un paramètre ϵ, sélectionné aléatoirement dans l'intervalle $[0, 10]$ [9], sera ajusté pour équilibrer la protection de la vie privée et l'utilité des prédictions.

Inversement, le K-Anonymat protège la vie privée en garantissant que chaque enregistrement dans un jeu de données est indistinguable d'au moins $k-1$ autres enregistrements avec des attributs similaires. Nous choisirons une valeur k qui maximise la difficulté d'associer des données à des individus spécifiques tout en maintenant une granularité des données suffisante pour une analyse significative.

L'évaluation se concentre sur trois métriques critiques : le taux d'attaques d'inférence réussies, l'impact sur l'utilité du score de confiance, et la vitesse de traitement de chaque méthode. En mesurant le taux d'inférence, nous visons à comprendre dans quelle mesure chaque méthode dissimule efficacement les attributs sensibles des attaquants potentiels. La métrique de perte d'utilité nous aidera à évaluer dans quelle mesure la méthode de protection affecte l'utilité des données pour des tâches analytiques légitimes. Enfin, la vitesse de traitement sera évaluée pour déterminer l'efficacité et la praticité de la mise en œuvre de chaque méthode dans des scénarios réels. Cette analyse comparative mettra non seulement en lumière les forces et les faiblesses de notre modèle, mais contribuera également des insights précieux sur les compromis impliqués dans la mise en œuvre de techniques de préservation de la vie privée dans des applications axées sur les données.

5.5.5 Résultats et analyse

La Figure 5.2 illustre l'analyse comparative des taux d'inférence d'attributs sensibles à travers notre modèle développé, la "Défense Adversaire en Deux Étapes", la Confidentialité Différentielle Locale (LDP) et les méthodes de K-Anonymat. Notre méthode "Défense Adversaire en Deux Étapes" montre systématiquement les taux d'inférence les plus bas, indiquant son efficacité à minimiser la probabilité que les attributs sensibles soient correctement inférés par les adversaires. Cette performance souligne les avantages de notre approche pour améliorer la confidentialité des données par rapport à la LDP et au K-Anonymat.

Dans la Figure 5.3, nous examinons l'impact de chaque méthode de préservation de la vie privée sur les taux d'erreur de classification. Notre méthode "Défense Adversaire en Deux Étapes" fournit non seulement une protection substantielle contre les attaques par inférence d'attributs, mais le fait également avec un impact minimal sur la précision de classification. Ce résultat souligne la capacité de notre méthode à maintenir l'utilité des données pour des fins analytiques légitimes tout en fournissant des mécanismes de défense robustes. Elle équilibre efficacement la sécurité avec l'utilité des données, surpassant à la fois la confidentialité différentielle locale et le K-Anonymat à cet égard.

La Figure 5.4 se concentre sur l'efficacité de la génération de vecteurs de bruit, un aspect crucial dans l'application pratique des techniques de préservation de la vie privée. Notre méthode "Défense Adversaire en Deux Étapes" est démontrée pour générer des vecteurs de bruit plus rapidement que la LDP et la K-Anonymat. Cela suggère que notre méthode améliore non seulement la protection de la vie privée mais le fait également de manière plus efficace. Cette efficacité rend notre méthode proposée particulièrement adaptée aux environnements où un traitement rapide des données est essentiel, offrant ainsi des avantages significatifs par rapport aux méthodologies comparées.

Dans chaque figure, notre méthode "Défense Adversaire en Deux Étapes" dépasse systématiquement ses homologues, démontrant des avantages complets en protégeant les informations sensibles, en préservant l'utilité des données et en assurant une efficacité opérationnelle. Ces résultats confirment l'efficacité de notre approche pour équilibrer une protection robuste de la vie privée avec les exigences pratiques des applications réelles.

Ces résultats mettent en évidence la capacité supérieure de notre méthode "Défense Adversaire en Deux Étapes" à protéger les informations sensibles, à maintenir l'utilité des données et à assurer une efficacité opérationnelle. En conséquence, notre approche se présente comme une solution robuste et pratique pour les applications MLaaS, où la protection de la confidentialité et la performance sont primordiales.

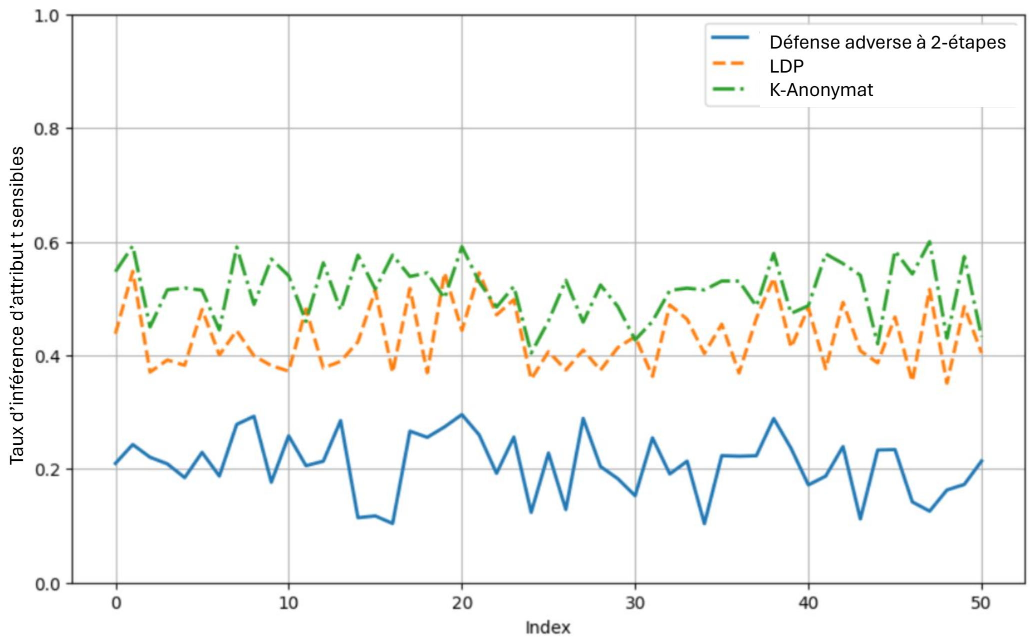

FIGURE 5.2 – Comparaison du taux d'inférence d'attributs sensibles entre la Défense Adversaire en Deux Étapes, la LDP et la K-Anonymat.

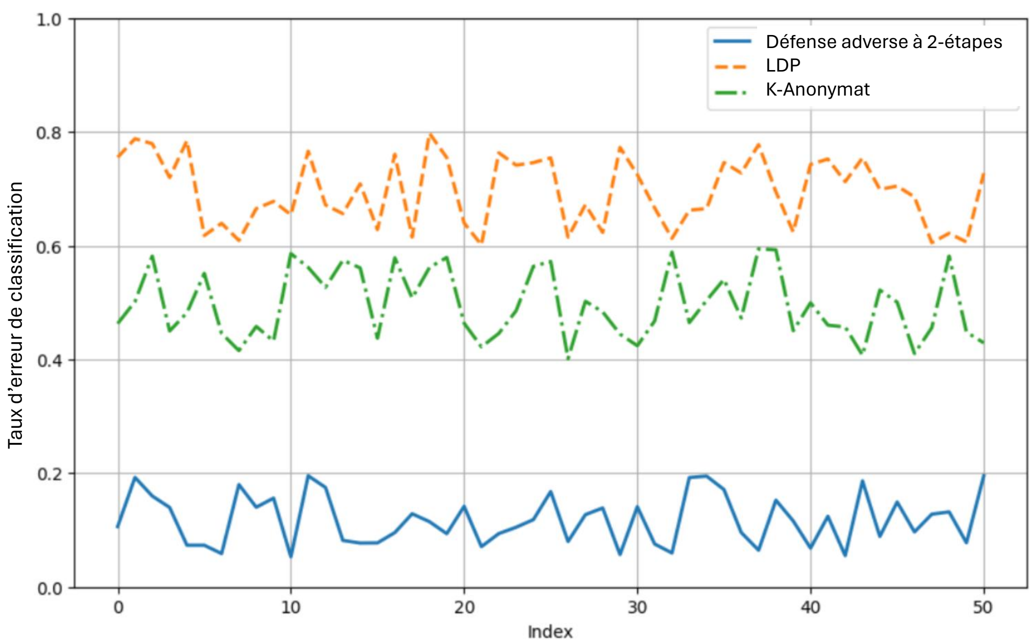

FIGURE 5.3 – Évaluation de l'impact sur le taux d'erreur de classification : Défense Adversaire en Deux Étapes vs. Méthodes de confidentialité.

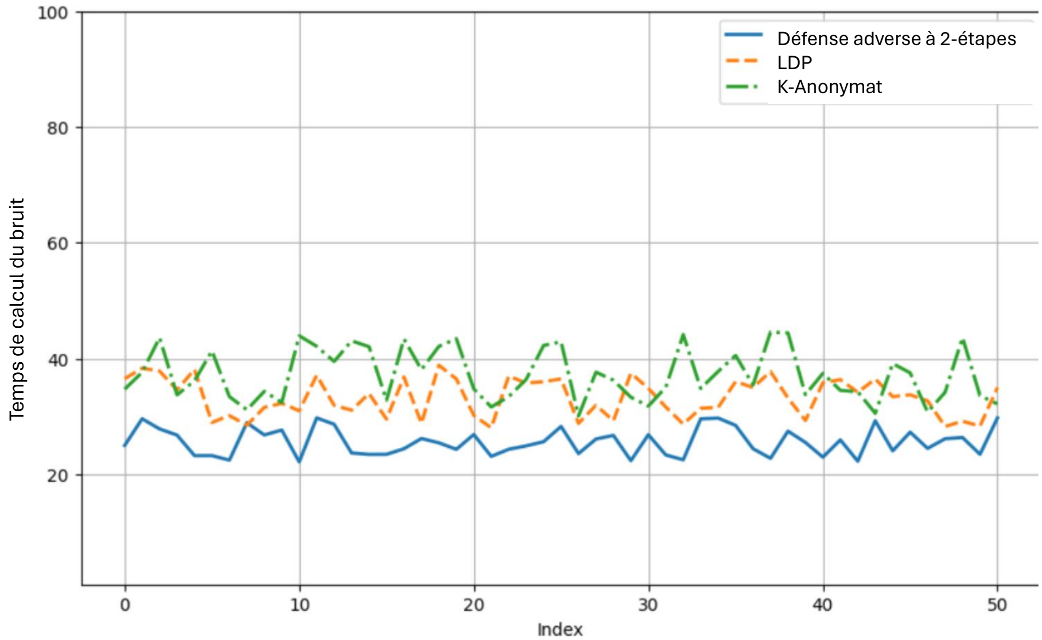

FIGURE 5.4 – Performance de génération de bruit : analyse comparative entre la Défense Adversaire en Deux Étapes, la LDP et la K-Anonymat.

5.6 Conclusion

Dans ce chapitre, nous avons approfondi notre analyse de la sécurité dans les environnements d'apprentissage automatique en tant que service (MLaaS), avec un focus particulier sur les risques souvent sous-estimés liés aux attaques par inférence d'attribut (AIA). Pour contrer ces attaques, nous avons développé une méthode de défense en deux étapes visant à renforcer la confidentialité des données. Cette stratégie commence par l'intégration d'exemples adverses sophistiqués, suivie d'une sélection stratégique de vecteurs de bruit. Nous avons démontré l'efficacité de notre approche, via des simulations, qui réduisent significativement la probabilité de divulgation d'attributs sensibles tout en préservant l'intégrité analytique des données pour des utilisations légitimes.

Pour les développements futurs, il serait intéressant d'approfondir l'étude sur l'amélioration de la résilience des exemples adverses dans les modèles MLaaS face à des attaques sophistiquées d'inférence d'attribut. Une piste prometteuse consiste à développer des techniques de génération d'exemples adverses qui intègrent une analyse plus fine de la dynamique des prédictions adverses, en particulier en évaluant comment les ajustements apportés aux exemples influencent spécifiquement les erreurs de classification des attaquants. Cela pourrait inclure l'usage de techniques avancées de perturbation adaptative, où les exemples adverses sont générés en temps réel en fonction de l'activité de l'attaquant observée, permettant de contrecarrer de manière proactive les tentatives de décodage des attributs privés.

Chapitre 6

Stratégies de défense contre l'empoisonnement des récompenses dans l'apprentissage par renforcement profond

6.1 Introduction

L'apprentissage par renforcement profond (DRL) s'impose aujourd'hui comme une technologie de pointe dans le domaine de l'intelligence artificielle, révolutionnant une multitude de secteurs grâce à sa capacité unique à résoudre des problèmes complexes de prise de décision. Utilisant des algorithmes avancés tels que les Deep Q-Networks (DQN), l'optimisation de politique de région de confiance (TRPO) ou l'algorithme acteur-critique asynchrone (A3C), le DRL exploite des espaces d'état vastes pour établir des politiques précises, répondant ainsi aux défis variés du monde réel [124, 157, 63].

Cependant, l'intégration accrue du DRL dans des applications critiques, notamment la robotique autonome, les soins de santé, et les systèmes financiers, met en évidence la nécessité de sécuriser ces modèles contre des attaques de plus en plus sophistiquées, comme les attaques d'empoisonnement des récompenses. Ces attaques, orchestrées avec subtilité, peuvent manipuler les signaux de récompense ou les dynamiques environnementales, compromettant ainsi la trajectoire optimale des politiques de l'agent d'apprentissage et entraînant des erreurs critiques dans la prise de décision [204, 198, 213, 113].

En réponse à ces menaces, ce chapitre se consacre aux défis de la sécurité dans les systèmes DRL opérationnels, avec un focus particulier sur la prévention des attaques d'empoisonnement des récompenses. Nous examinons un scénario d'attaque où un adversaire, doté d'une connaissance approfondie de l'environnement d'apprentissage, cherche à manipuler discrètement les récompenses. Pour contrecarrer efficacement ces stratégies adverses, nous proposons une architecture d'entraînement multi-environnements qui permet à l'agent de différencier et d'épurer les signaux de récompense. Cette approche utilise la méthode de réduction de la variance pour minimiser l'impact des récompenses empoisonnées sur l'apprentissage de l'agent, améliorant ainsi la robustesse du système face à ces manipulations tout en préservant l'efficacité de l'apprentissage. Bien que cette architecture soit coûteuse en ressources, car elle nécessite un entraînement dans plusieurs environnements, elle est essentielle pour réduire le risque d'empoisonnement. Pour optimiser la performance de cette méthode, nous intégrons une modélisation sous forme de jeu Bayésien à deux joueurs. Cette modélisation permet d'adapter les seuils de décision de manière dynamique en tenant compte des comportements stratégiques de l'adversaire, rendant l'entraînement multi-environnements plus efficace. Ce cadre novateur illustre non seulement les dynamiques stratégiques entre l'agent d'apprentissage et l'attaquant, mais esquisse également des avenues pour une défense optimisée dans des contextes d'apprentissage en ligne complexes.

6.2 Travaux connexes

La littérature sur les attaques contre l'apprentissage par renforcement peut être classée en plusieurs catégories. Une catégorie inclut les attaques temporelles, également connues sous le nom d'attaques par évasion, qui ont été largement étudiées dans le DRL [27, 78, 99, 109]. Ces attaques visent à trouver des exemples adverses qui provoquent des comportements indésirables dans une politique déjà entraînée. Une autre catégorie inclut les attaques de corruption de modèle qui se produisent pendant la phase d'entraînement du DRL. Ces attaques corrompent le signal de récompense pour faire apprendre à l'agent des politiques sous-optimales. Le but de l'attaquant est de dégrader considérablement la performance des méthodes d'interprétation du RL en manipulant les paramètres du modèle appris tout en maintenant la performance originale pour assurer une discrétion maximale.

Des études antérieures ont exploré l'empoisonnement des récompenses dans les contextes d'apprentissage par renforcement (RL) à la fois batch (hors ligne) et en ligne. Dans le RL batch, les récompenses sont pré-collectées par une politique de comportement [1], ce qui facilite leur modification par l'attaquant. En revanche, le RL en ligne exige que l'attaquant modifie les récompenses à la volée, ce qui représente une tâche plus complexe. Les bandits à plusieurs bras, un modèle classique de prise de décision séquentielle, ont été exploités pour étudier l'empoisonnement des récompenses dans l'apprentissage par renforcement (RL) en ligne. Dans ce modèle, un agent doit choisir entre plusieurs options (ou "bras"), chacune offrant une récompense aléatoire basée sur une distribution de probabilité spécifique. L'objectif est de maximiser les récompenses cumulatives au fil du temps. Cependant, lorsque les récompenses sont perturbées par un attaquant, cela peut conduire à des choix sous-optimaux et à un grand "regret", terme désignant la différence entre la récompense obtenue et celle qu'aurait pu obtenir l'agent en faisant des choix optimaux en permanence [203, 94, 110, 202, 113, 147, 213]. Ce cadre de bandit est particulièrement pertinent pour l'étude de l'empoisonnement des récompenses, car il illustre clairement les impacts des manipulations adverses sur la stratégie d'apprentissage de l'agent.

Des études récentes ont exploré l'empoisonnement des récompenses dans le contexte de boîte blanche, où l'attaquant a une connaissance complète du processus de décision de Markov (MDP) ou de l'algorithme d'apprentissage. Ces études se sont concentrées sur l'attaque de la

1. Une politique de comportement dans le RL batch est une stratégie utilisée par l'agent pour choisir des actions à partir desquelles les données d'apprentissage sont ensuite générées. Elle détermine la manière dont l'agent explore son environnement et recueille les informations nécessaires pour l'apprentissage. Ces données pré-collectées, incluant les récompenses, sont susceptibles d'être modifiées par un attaquant, rendant ainsi l'agent vulnérable à apprendre des politiques suboptimales.

fonction de récompense elle-même en utilisant des récompenses adverses qui sont des fonctions de l'état et de l'action mais indépendantes du processus d'apprentissage. Une étude notable de Zhang *et al.* [213] a développé une attaque adaptative qui dépend de la Q-table de la victime, rendant l'attaque exponentiellement plus rapide que les attaques non adaptatives.

Une autre catégorie d'attaques est les attaques de perturbation d'observation, qui modifient l'observation de l'environnement de l'agent pendant le temps d'entraînement [15, 82]. Ces attaques supposent un accès aux capteurs de l'agent et ne changent pas l'état réel ou la récompense de l'environnement.

Enfin, [177] représente l'un des rares travaux abordant les attaques d'empoisonnement des récompenses dans le DRL. Cette étude présente cependant plusieurs contraintes significatives : elle nécessite une connaissance préalable de l'algorithme d'apprentissage utilisé, elle se limite aux algorithmes d'apprentissage sur politique, et elle ne permet pas de manipuler les observations passées dans un lot d'entraînement lors des mises à jour séquentielles par l'agent. Ces limitations contrastent avec notre approche, qui traite l'empoisonnement des récompenses dans un contexte de DRL en ligne. Nous proposons une solution qui ne repose pas sur la connaissance de l'algorithme d'apprentissage spécifique et qui est capable de gérer dynamiquement les ajustements des récompenses au fur et à mesure de leur réception, augmentant ainsi la robustesse et l'adaptabilité de notre système face aux attaques en temps réel.

Pour une présentation cohérente et logique des diverses techniques de sécurité dans le domaine de l'Apprentissage par Renforcement Profond (Deep Reinforcement Learning : DRL), le texte suivant harmonise les idées et crée une progression naturelle dans la discussion des stratégies de défense contre les attaques d'empoisonnement.

Pour assurer la sécurité des politiques de DRL pendant l'entraînement, diverses stratégies de défense ont été explorées, allant de la randomisation des données à l'optimisation inverse, visant à contrer efficacement les attaques d'empoisonnement des données tant dans les domaines de l'apprentissage par renforcement que supervisé.

Tout d'abord, il est crucial d'approfondir le concept de robustesse, qui est la capacité de l'agent à fonctionner de manière fiable en présence de perturbations. Ce concept est exploré à travers deux principales catégories de défenses. D'une part, certaines études offrent des garanties théoriques pour l'apprentissage de politiques sous perturbations dans des plages spécifiques [10, 112, 34, 195, 211, 211], tandis que d'autres examinent empiriquement la robustesse des modèles de formation aux perturbations [15, 17, 191]. Cependant, concevoir des algorithmes DRL qui intègrent la robustesse est coûteux et peut compromettre la performance de la politique.

Par ailleurs, la randomisation des données d'entraînement ou de test est une technique répandue pour augmenter la sécurité des systèmes. Cette approche consiste à faire des prédictions basées sur la probabilité la plus élevée après introduction de bruit aléatoire dans les points de données [102, 197, 104, 36, 152]. De plus, des méthodes statistiques robustes sont employées pour détecter les valeurs aberrantes en comparant les variances des échantillons avec des seuils prédéfinis [211].

Un autre mécanisme de défense innovant est l'utilisation de l'optimisation inverse pour calculer des politiques robustes, particulièrement pertinente dans les scénarios d'apprentissage par renforcement hors ligne [10]. Ces méthodes nécessitent généralement l'entrée d'un jeu de données et d'un paramètre de défense spécifique, comme la variance du bruit pour la randomisation, un seuil pour la détection des valeurs aberrantes, ou un paramètre d'attaque pour l'optimisation inverse.

Cependant, déterminer avec précision les paramètres optimaux représente un défi majeur, particulièrement lorsque l'attaquant adapte dynamiquement sa stratégie pour exploiter les faiblesses du système de défense. Face à ce défi, l'adoption de solutions conventionnelles basées uniquement sur la robustesse peut entraîner des compromis significatifs en termes de performance des politiques.

Dans le contexte actuel, où les attaques d'empoisonnement des récompenses deviennent de plus en plus sophistiquées, en particulier dans le domaine du DRL en ligne, il devient impératif de développer des solutions défensives qui vont au-delà des approches traditionnelles. Ce chapitre présente notre contribution, une technique novatrice conçue pour répondre à ces défis en intégrant des concepts avancés de robustesse tout en préservant, voire en améliorant, la performance des politiques de DRL.

Notre approche propose une architecture multi-environnements qui non seulement teste et valide la fiabilité des récompenses reçues par l'agent, mais utilise également des stratégies de prise de décision fondées sur des jeux Bayésiens pour adapter dynamiquement les mécanismes de défense en fonction des comportements stratégiques de l'adversaire. Ce faisant, nous ne nous contentons pas de réagir aux attaques, mais anticipons et neutralisons activement les tentatives d'empoisonnement, ce qui renforce considérablement la sécurité sans compromettre l'efficacité de l'apprentissage.

Nous croyons que cette méthode, en se distinguant par sa capacité à intégrer une analyse stratégique dans le processus de défense, apporte une contribution significative à la sécurisation de l'apprentissage DRL en ligne. Elle promet non seulement de protéger efficacement contre les attaques d'empoisonnement, mais aussi d'améliorer la fiabilité des politiques apprises, ouvrant ainsi la voie à des applications de DRL plus sûres et plus performantes.

6.3 Principes fondamentaux du DRL

Dans cette section, nous proposons un aperçu complet du processus de DRL, en mettant en lumière ses composants clés et ses principes sous-jacents.

Apprentissage par renforcement profond. Dans l'apprentissage par renforcement (RL), un agent apprend un comportement optimal en interagissant séquentiellement avec un environnement, connu sous le nom de Processus Décisionnel de Markov (MDP), pour atteindre ses objectifs par essais et erreurs. Le MDP est défini comme un tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, R, \gamma, \eta)$, où $\mathcal{S}$ et $\mathcal{A}$ sont respectivement les espaces d'états et d'actions, $\mathcal{P}$ est la dynamique de transition qui détermine la distribution de probabilité de l'état suivant étant donné l'état actuel et l'action, R est la fonction de récompense qui associe des paires état-action à des récompenses scalaires, γ est un facteur de remise qui pondère les récompenses immédiates et futures, et η est la distribution initiale sur les états.

L'agent interagit séquentiellement avec l'environnement, commençant par un état initial $s_0 \in \mathcal{S}$, suivant la distribution initiale η, et sélectionnant une action $a_t \in \mathcal{A}$ à chaque pas de temps t basé sur une politique π qui associe des états à des actions. La politique π, générique ou stochastique, est une fonction $\pi : \mathcal{S} \to P(A)$, où $P(A)$ est l'ensemble de toutes les probabilités dans l'espace des actions A. La notation $\pi(a|s)$ est utilisée pour indiquer la probabilité de choisir l'action a dans l'état s. Les politiques déterministes sont un cas spécial de politiques stochastiques, où π est un mappage direct des états vers les actions, i.e. $\pi : \mathcal{S} \rightarrow \mathcal{A}$.

L'agent passe à un nouvel état $s_{t+1} \in \mathcal{S}$ selon $\mathcal{P}$ et reçoit une récompense $r_{(s_t,a_t)} = R(s_t, a_t)$ qui reflète la qualité de sa décision. Le processus génère une séquence de triplets état-action-récompense $\mathbb{T} = (s_t, a_t, r_{(s_t,a_t)})_{t=0}^{T}$, appelée trajectoire, où T représente le pas de temps à la fin de l'environnement.

Dans l'apprentissage par renforcement, la récompense cumulative ou le retour est la récompense totale qu'un agent reçoit sur une période donnée. Elle est calculée comme la somme des récompenses obtenues à chaque pas de temps, actualisée par un facteur $\gamma \in [0, 1]$, qui indique combien l'agent valorise les récompenses immédiates par rapport aux récompenses futures. Ceci peut être représenté mathématiquement par $CR = \sum_{t=0}^{T} \gamma^t R(s_t, a_t)$. Le facteur de remise γ permet à l'agent de prendre des décisions en équilibrant l'importance des récompenses immédiates et futures.

Nous définissons la valeur d'un état s pour une politique π par la fonction $\mathcal{V}_\pi : \mathcal{S} \rightarrow \mathbb{R}$, définie comme suit :

$$\mathcal{V}_\pi(s) = \mathbb{E}[CR|s = s_t]. \tag{6.1}$$

La définition de la fonction $\mathcal{V}$ (valeur-état) peut être étendue aux paires état-action, également connue sous le nom de fonction Q. Elle définit la valeur de l'action a dans l'état s sous une politique π, notée Q_π, comme le retour attendu CR à partir de s, en choisissant l'action a, puis en suivant la politique π.

$$\mathcal{Q}_\pi(s, a) = \mathbb{E}[CR|s = s_t, a = a_t]. \tag{6.2}$$

L'objectif de l'agent est de trouver une politique π^* qui maximise le retour attendu de tous les états, soit :

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\ \mathcal{Q}_\pi(s, a). \tag{6.3}$$

Le score d'une politique π est noté ρ^π et est défini comme la mesure de la qualité globale d'une politique donnée, qui représente la somme des récompenses attendues que l'agent peut espérer recevoir en suivant cette politique sur une longue période. Le score d'une politique est calculé en prenant en compte toutes les actions possibles que l'agent peut prendre à partir de chaque état, en utilisant les valeurs Q. Le score de la politique π est calculé comme suit :

$$\rho^\pi = \mathbb{E}[(1 - \gamma) \sum_{t=1}^{\infty} \gamma^{t-1} R(s_t, a_t)|\pi, \eta] \tag{6.4}$$

Dans cette formule, l'état initial s_1 est échantillonné à partir de la distribution initiale η, et les états suivants s_t sont obtenus en exécutant la politique π dans le MDP. Ainsi, le score d'une politique est le retour total attendu, réduit par un facteur de $1 - \gamma$.

Le DRL est une intégration de l'apprentissage profond avec l'apprentissage par renforcement, conçu pour surmonter les défis des environnements RL traditionnels. Il excelle dans l'apprentissage de politiques de contrôle directement à partir de données d'entrée brutes de haute dimension et dans la gestion de grands espaces d'états et d'actions. Cette intégration permet à l'agent non seulement d'apprendre mais aussi de généraliser efficacement à travers une gamme diversifiée d'états, améliorant son adaptabilité et ses performances dans des environnements complexes.

Dans le DRL, la politique π est représentée par un réseau neuronal profond avec des paramètres Θ (c'est-à-dire, les poids du réseau de politique). Pour déterminer les paramètres optimaux de la politique, divers algorithmes de DRL ont été proposés, y compris le Deep Q-Network (DQN) [124], l'optimisation de la politique de région de confiance (TRPO) [157], et l'acteur-critique asynchrone avantageux (A3C) [63]. Ces algorithmes diffèrent dans leurs méthodes d'optimisation, mais visent tous à améliorer le réseau de politique en maximisant le retour attendu sur un ensemble d'épisodes de formation.

Q-learning profond. Une manière de trouver une politique optimale pour un agent afin de maximiser sa récompense totale consiste à utiliser le Q-learning profond, qui prend un état s et approxime la valeur Q pour chaque action basée sur cet état (c'est-à-dire, $\mathcal{Q}_\pi(s,a)$). Avec cette approximation, bien que l'agent ne puisse pas extraire explicitement la politique, il pourrait toujours maximiser sa récompense en prenant l'action avec la valeur Q la plus élevée. Cette méthode a démontré un grand succès dans de nombreuses applications, telles que jouer au Go [169] et maîtriser une grande variété de jeux Atari [123].

Algorithme de gradient de politique (Policy Gradient Algorithm,). La méthode gradient de politique consiste à paramétrer directement la politique comme une fonction de l'état et de l'action. Cette fonction, notée $\pi_\theta(s,a) = p(a|s,\theta)$, prend en entrée l'état au temps t et produit l'action correspondante. Dans des études récentes [122], les chercheurs ont modélisé la politique comme un réseau neuronal profond, tel qu'un perceptron multicouche [200] ou un réseau neuronal récurrent [212], et l'ont nommé le réseau de politique (Policy network).

Pour entraîner un agent avec un réseau de politique, l'algorithme de gradient de politique définit une fonction objectif qui représente les récompenses totales actualisées attendues. Cette fonction, notée $J(\theta) = \mathbb{E}_{s_0,a_0,\ldots\sim\pi_\theta}[\sum_{t=0}^{\infty} \gamma^t R(s_t,a_t)]$, est maximisée pour obtenir les paramètres optimaux θ et la politique. L'algorithme calcule le gradient de cette fonction objectif par rapport aux paramètres et applique de manière itérative l'ascension de gradient stochastique pour atteindre un maximum local de $J(\theta)$.

Le théorème du Gradient de Politique (Policy Gradient Theorem) [98] stipule que, pour toute politique différentiable $\pi_\theta(s,a)$, le gradient de politique peut être exprimé comme le produit attendu du gradient du logarithme du réseau de politique et de la fonction de valeur d'action du Processus Décisionnel de Markov correspondant. Cette équation, notée $\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \; log \; \pi_\theta(s,a)\mathcal{Q}_\pi(s,a;\theta)]$, nécessite la connaissance de la fonction de valeur-action $\mathcal{Q}_\pi(s,a;\theta)$ pour être résolue. Dans l'algorithme de gradient de politique, cette fonction est approximée à l'aide d'un réseau neuronal profond noté $\mathcal{Q}(s,a;\omega)$, qui est appris en parallèle avec le réseau de politique.

Ces informations détaillées sur le DRL et ses composantes permettent de mieux comprendre la complexité de l'apprentissage par renforcement profond et l'intégration des techniques d'apprentissage profond pour améliorer la performance des agents dans des environnements complexes. Cette intégration marque un tournant dans la façon dont les algorithmes d'apprentissage par renforcement sont développés et appliqués, offrant de nouvelles perspectives et méthodologies dans le domaine de l'intelligence artificielle.

6.4 Attaques d'empoisonnement de récompenses sur le DRL en ligne

Les attaques d'empoisonnement de récompenses représentent une menace significative dans le domaine de l'apprentissage par renforcement en ligne, où des adversaires manipulent les fonctions de récompense pour détourner les agents de leur trajectoire optimale. Ces attaques visent à forcer les agents à adopter des actions spécifiques, à atteindre ou à éviter certains états, ou à maximiser le regret de l'agent, altérant ainsi leur performance et fiabilité.

Nous nous concentrons ici sur l'analyse de scénarios où un attaquant ajuste subtilement la fonction de récompense pour induire un agent à suivre une politique déterministe non optimale, $\pi^\dagger \in \Pi^{det}$, dénommée politique cible. L'objectif est de maintenir cette politique de manière prolongée, indépendamment de sa pertinence pour la tâche assignée à l'agent.

Cette forme d'attaque, bien documentée dans les études antérieures, est explorée ici en nous appuyant sur des modèles d'attaque développés par Ma *et al.* [113] et Rakhsha *et al.* [147]. Nous introduisons des variations sur ces modèles afin d'en améliorer l'applicabilité et l'efficacité dans des contextes d'apprentissage en ligne réels.

6.4.1 Modèle de menace

Dans notre travail, nous abordons le problème de l'empoisonnement des récompenses dans un contexte de boîte blanche, où l'attaquant a une connaissance complète de l'environnement du processus de décision de Markov (MDP) de l'agent et de l'algorithme Deep Q-Learning, excepté pour les probabilités de transition d'états futurs qui demeurent incertaines. Cette incertitude sur l'aléatoire futur, qui se rapporte à la dynamique imprévisible des transitions entre les états dans le MDP, reste hors de portée de l'attaquant. Cette configuration permet à l'attaquant de se placer entre l'environnement et l'agent, lui permettant de réaliser facilement des attaques de boîte blanche, comme décrit dans la Figure 6.2.

Formellement, à chaque étape t, l'état et l'action pris par l'agent sont s_t et a_t respectivement. L'environnement génère la récompense instantanée $r_t = R(s_t, a_t)$ et l'état suivant s_{t+1}. L'attaquant peut observer le réseau de politique de l'agent $\pi_\theta(s, a)$, l'état actuel s_t, l'action prise par l'agent a_t, le nouvel état résultant s_{t+1} de la transition de l'environnement et la récompense r_t reçue par l'agent. L'attaquant injecte une perturbation δ_t dans la récompense. L'agent reçoit alors la récompense perturbée $r_t + \delta_t$.

Formellement, à chaque point de décision indiqué par l'étape de temps t, l'agent observe un état s_t et sélectionne une action a_t selon sa politique actuelle $\pi_\theta(s, a)$. En réponse, l'environnement fournit une récompense immédiate $r_t = R(s_t, a_t)$ et passe à un nouvel état s_{t+1}. Un adversaire, ayant la capacité d'observer la politique de l'agent, l'action prise a_t, l'état résultant s_{t+1}, et la récompense distribuée r_t, injecte stratégiquement une perturbation δ_t dans le signal de récompense. En conséquence, l'agent reçoit et traite une récompense empoisonnée $r'_t = r_t + \delta_t$, qui peut s'écarter du signal de renforcement prévu par l'environnement.

Sur cette base, notre modèle impose des contraintes réalistes mais moins importantes sur les capacités de l'attaquant. Pour orchestrer des attaques pratiquement réalisables, l'attaquant affine le bruit injecté δ_t, dans le but d'éviter de déclencher les mécanismes défensifs de l'agent d'apprentissage. Dans notre modèle, nous décrivons un attaquant qui, bien qu'ayant

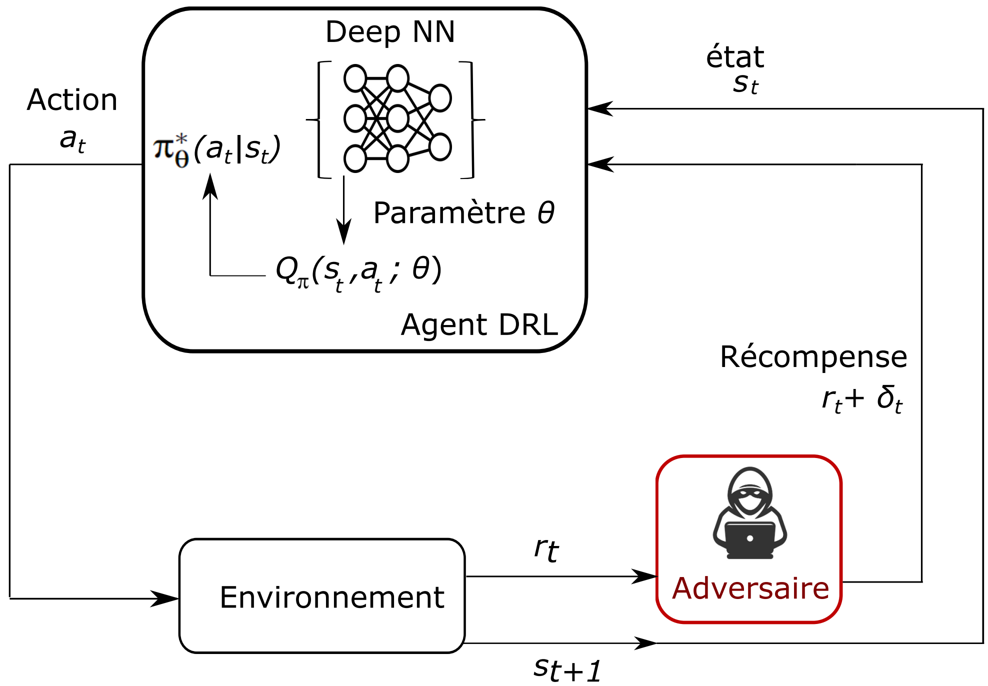

FIGURE 6.1 – Apprentissage par renforcement profond avec des observations de récompenses empoisonnées. L'agent observe une récompense empoisonnée $r_t + \delta_t$ plutôt que la véritable récompense de l'environnement r_t.

un accès complet aux actions de l'agent, doit gérer soigneusement la quantité de bruit qu'il injecte pour éviter d'être détecté.

Le profil de l'attaquant est alors défini par le triplet $\xi = (\pi^\dagger, \Delta_t, \upsilon)$, où la politique cible déterministe $\pi^\dagger$ délimite les résultats souhaités par l'adversaire pour les actions de l'agent. Le terme Δ_t représente le niveau maximal de bruit que l'attaquant peut ajouter à la récompense à chaque étape de temps t sans déclencher les mécanismes défensifs de base de l'agent, $\delta_t \leq \Delta_t, \forall t$. Cette contrainte assure que les perturbations restent suffisamment discrètes pour éviter la détection par l'agent, permettant ainsi à l'attaquant d'agir subtilement tout en encourageant l'agent à adopter la politique cible $\pi^\dagger$. Enfin, υ représente le "seuil de déviation de la politique". En définissant ce seuil, nous supposons que l'attaquant est contraint par la nécessité de rester indétecté tout en influençant la politique de l'agent. Cette hypothèse est réaliste car, dans un environnement de sécurité, les attaquants cherchent souvent à maximiser leur impact tout en évitant de déclencher des alarmes. Si un attaquant modifie trop fréquemment ou de manière trop évidente les récompenses, cela pourrait facilement attirer l'attention et activer des mécanismes de défense. Ajuster υ permet au modèle d'attaque de s'adapter à différentes conditions opérationnelles, des attaques prolongées et discrètes aux actions rapides et précises.

La politique cible est définie comme une fonction de l'espace d'état à l'espace d'action, $\pi^\dagger : S \rightarrow 2^{|\mathcal{A}|}$, qui spécifie l'ensemble des actions désirées par l'attaquant dans l'état s. L'attaquant peut se concentrer sur certains états plus que d'autres, car ces états déclenchent des actions particulières désirées par l'attaquant. Ainsi, $\mathcal{S}^\dagger$, l'ensemble des états cibles, peut être défini comme $\mathcal{S}^\dagger = \{s \in \mathcal{S} : \pi^\dagger(s) \neq \pi^*(s)\}$, avec $\pi^*(s)$ la politique optimale de l'agent, qui représente les actions qu'il aurait choisies en l'absence de perturbations de récompense. Donc, étant donné un ensemble d'états cibles $\mathcal{S}^\dagger \subseteq \mathcal{S}$, la politique cible est notée comme :

$$\pi_\theta^\dagger(s) = \begin{cases} a^\dagger & \text{si } s \in \mathcal{S}^\dagger \\ \pi_\theta(s) & \text{autrement.} \end{cases}$$

où $a^\dagger$ est l'action cible désirée par l'attaquant, $\pi_\theta(s)$ est la politique réelle de la victime et $\pi_\theta^\dagger(s)$ la politique cible partielle, qui est plus adaptée pour les grands espaces d'états, discrets ou continus, par rapport à la politique cible complète qui définit les actions souhaitées dans tous les états.

L'objectif de l'attaquant est de déterminer stratégiquement la séquence optimale de perturbations qui dirigera l'agent vers la politique cible sans déclencher les mécanismes de défense fondamentaux de l'agent d'apprentissage.

Par conséquent, le problème de l'attaquant est modélisé comme le problème d'optimisation avec contraintes suivant :

$$\delta_t^* = \min_{\delta_t} \quad d(r_t, r_t') \tag{6.5}$$

$$\text{s.t} \qquad \rho^{\pi^\dagger} \geq \rho^\pi, \quad \forall \pi \in \Pi^{det} \backslash \{\pi^\dagger\} \tag{6.6}$$

$$\delta_t \leq \Delta_t, \forall t \tag{6.7}$$

$$\sum_t \mathbf{1}[Q_t \neq Q^\dagger] \leq \upsilon \tag{6.8}$$

où d calcule la distance euclidienne entre la vraie récompense et la récompense empoisonnée. Ainsi, le problème de l'attaquant se résume à résoudre le problème d'optimisation (6.5-6.8) pour trouver la perturbation minimale qui permet l'adoption de sa politique cible par l'agent tout en évitant la détection.

6.5 Architecture d'entraînement multi-environnements pour un DRL résilient aux attaques d'empoisonnement des récompenses

Dans cette section, nous présentons une nouvelle architecture d'apprentissage conçue pour améliorer la généralisation et la résilience face aux attaques d'empoisonnement des récompenses dans les environnements de l'apprentissage par renforcement profond (DRL). Cette architecture tire parti d'une stratégie de formation multi-environnements où l'agent d'apprentissage est exposé à n environnements distincts, chacun avec des probabilités de transition uniques. Cette diversification enrichit non seulement l'expérience de formation de l'agent d'apprentissage mais atténue également l'impact des récompenses potentiellement empoisonnées. L'hypothèse sous-jacente est que, pour des raisons de discrétion et afin d'éviter la détection, l'attaquant ne compromettra pas les récompenses dans tous les environnements simultanément. Ainsi, en intégrant plusieurs environnements dans le processus de formation, le modèle peut bénéficier de données moins biaisées, réduisant ainsi le risque d'erreur dû à un empoisonnement ciblé. Cette stratégie permet de comparer et de purifier les signaux de rétroaction entre les différents environnements, offrant une couche supplémentaire de défense contre les tentatives d'empoisonnement.

L'essence de notre stratégie de défense repose sur une configuration d'environnement généralisée $\mathcal{G} = (\mathcal{E}, \mu)$, où $\mathcal{E}$ représente l'ensemble de tous les environnements et μ la distribution de probabilité à travers ces environnements. Chaque environnement $e_i \in \mathcal{E}$ est structuré comme un processus de décision markovien (MDP). Cette approche multi-environnement est principalement conçue pour minimiser l'impact des récompenses empoisonnées, en exploitant l'hypothèse que l'empoisonnement ne peut affecter tous les environnements simultanément sans être détecté. Ainsi, l'agent peut bénéficier de signaux plus fiables et moins biaisés en comparant et purifiant les feedbacks à travers différents contextes. En second lieu, cette stratégie favorise également le développement d'une politique robuste, permettant à l'agent d'apprendre et de s'adapter à une variété de scénarios, ce qui renforce sa capacité à fonctionner efficacement dans des environnements diversifiés.

La fonction Q dans ce cadre est conçue pour tenir compte des variations entre les environnements, calculée comme une moyenne pondérée :

$$Q_{\text{exp}}(s, a) = \sum_i \mu_i \cdot Q_i(s, a; \theta), \tag{6.9}$$

où μ_i indique la probabilité de sélectionner l'environnement i. La politique dérivée $\pi_{\text{avg}}(s)$, déterminant les actions optimales, est :

$$\pi_{\text{avg}}(s) = \underset{a}{\text{argmax}}\ Q_{\text{exp}}(s, a). \tag{6.10}$$

Pour garantir l'intégrité des signaux de récompense, nous mettons en œuvre une méthode de détection basée sur la variance après l'entraînement. La récompense moyenne globale pour une paire état-action (s, a) à travers les environnements est calculée comme suit :

$$\bar{r}_{\text{global}} = \frac{1}{n} \sum_{i=1}^{n} r_i. \tag{6.11}$$

Nous calculons ensuite la variance :

$$\sigma^2 = \frac{1}{n-1} \sum_{i=1}^{n} (r_i - \bar{r}_{\text{global}})^2. \tag{6.12}$$

Si la variance dépasse un seuil prédéfini κ,

$$\frac{1}{n-1} \sum_{i=1}^{n} (r_i - \bar{r}_{\text{global}})^2 > \kappa, \tag{6.13}$$

cela suggère un potentiel empoisonnement des récompenses, nécessitant un ajustement du protocole de formation de l'agent d'apprentissage. Actuellement, le seuil κ est fixe, qui pourrait être établi à partir des résultats de simulations préliminaires. Cependant, pour une adaptation plus précise à l'état réel d'empoisonnement des environnements, il serait avantageux que ce seuil soit dynamique, se modifiant en fonction des variations observées dans les données d'apprentissage. Un seuil dynamique permettrait de refléter plus fidèlement le niveau d'empoisonnement et d'optimiser les réponses du système défensif.

Bien que la stratégie de formation multi-environnements accroisse la capacité des systèmes DRL à résister aux attaques d'empoisonnement des récompenses, elle exige une consommation importante de ressources computationnelles et une gestion méticuleuse. Pour optimiser l'équilibre entre les bénéfices sécuritaires et les coûts associés, nous proposons une approche hybride ajustant dynamiquement le régime d'entraînement de l'agent. Cela lui permet d'apprendre dans plusieurs environnements lorsque le risque d'empoisonnement est élevé et de s'entraîner sur un environnement unique tiré aléatoirement lorsque ce risque est faible.

Nous intégrons un modèle de jeu Bayésien pour modéliser cette dynamique, permettant d'ajuster de manière adaptative les stratégies d'entraînement en tenant compte du comportement dynamique de l'attaquant. Ce modèle traite les décisions concernant l'état de l'environnement, empoisonné ou non empoisonné, et la méthode de formation, un ou plusieurs environnements, comme des choix stratégiques interdépendants. Cette approche de modélisation permet une prise de décision flexible qui s'ajuste en fonction du niveau actuel de menace et de la disponibilité des ressources, facilitant une allocation plus efficace et sécurisée des efforts d'apprentissage.

6.6 Modèle de jeu Bayésien pour l'optimisation stratégique de la sécurité dans le DRL

Dans cette section, nous étudions les interactions entre un agent d'apprentissage par renforcement en ligne et des environnements qui peuvent être empoisonnés, traitant ces interactions comme un jeu bayésien. Ce cadre est particulièrement bien adapté pour aborder les complexités inhérentes à ce type de scénario, où l'incertitude joue un rôle prépondérant. Au cœur de notre analyse réside l'incertitude quant au statut d'empoisonnement de l'environnement, une variable critique qui influence les stratégies adoptées tant par l'agent d'apprentissage que par l'attaquant potentiel.

Durant l'entraînement, l'agent développe une croyance sur l'état de l'environnement dans lequel il s'entraîne, basée sur ses observations. Cette "croyance" peut être vue comme une estimation probabiliste de la nature de l'environnement. Plus précisément, si cet environnement est empoisonné ou non. Cette perception est dynamiquement mise à jour en réponse aux observations recueillies au fil du temps, ce qui constitue un élément essentiel de la prise de décision sous incertitude.

Dans notre cadre de modélisation, nous considérons une interaction entre l'agent d'apprentissage par renforcement et l'environnement, traité comme un joueur avec des informations privées sur son type (empoisonné ou normal). L'agent maintient une croyance préalable concernant le statut d'empoisonnement de l'environnement. Par la suite, nous construisons des stratégies optimales pour les deux adversaires, correspondant à l'équilibre de Nash bayésien.

6.6.1 Paramètres du jeu

Nous établissons les éléments clés et les scénarios possibles qui peuvent survenir pendant le jeu pour chaque joueur. Notre cadre implique un agent d'apprentissage qui interagit avec un environnement potentiellement empoisonné. Dans ce contexte, nous désignons le joueur de l'environnement comme le joueur émetteur, noté i, qui possède des informations privées sur son type. Ce type peut être "normal", représenté par $\lambda_i = 0$, ou "empoisonné", noté $\lambda_i = 1$. Important, le statut d'empoisonnement du joueur i reste inconnu de l'agent j, tandis que le joueur j est intrinsèquement de type normal, représenté par $\lambda_j = 0$, et ce fait est de connaissance commune pour les deux joueurs.

Au début du jeu, le joueur agent a des informations préalables concernant le statut d'empoisonnement du joueur émetteur. Ces informations sont encapsulées dans la probabilité préalable ω_0, qui est dans l'intervalle $[0, 1]$.

Chaque joueur sélectionne une action alignée avec son objectif prédéfini. Dans le cas où le joueur émetteur est de type empoisonné, il fait face à une décision entre deux options : 'Empoisonner l'Environnement' (P) ou 'Ne Pas Empoisonner' (NP). D'autre part, si le joueur émetteur est de type normal, il est limité à l'action 'Ne Pas Empoisonner' (NP). Pour l'apprenant agent, il existe deux actions disponibles : 'Entraînement multi-environnements' (MET) et 'Entraînement en Environnement Unique' (SET).

Le jeu se déroule comme suit, avec plusieurs paramètres à considérer. Lorsque l'attaquant décide de ne pas empoisonner la récompense, il ne gagne ni ne perd rien lors de ce tour. Cependant, s'il choisit d'empoisonner k sur n environnements, cela lui coûte k multiplié par le coût de cette attaque, noté c_a. Ce coût c_a est un composant critique de notre modèle car il quantifie le risque de l'attaquant d'être identifié et détecté par les mécanismes de défense du système. D'un autre côté, l'agent d'apprentissage fait face à différents scénarios. S'il opte pour 'Entraînement en Environnement Unique' et que l'attaquant n'a lancé aucune attaque d'empoisonnement des récompenses, l'agent subit le coût d'entraînement c_t pour cet environnement unique, qui est nettement inférieur au coût d'entraînement sur tous les n environnements (nc_t). Dans le scénario où l'attaquant attaque l'environnement et altère la récompense originale, si l'agent s'entraîne dans un seul environnement, cela coûte à l'attaquant c_a pour infecter cet environnement. Cependant, l'attaquant gagne $-g_a - c_a$ car l'attaque est réussie et l'agent n'a aucune alternative. Pour l'agent, cela entraîne une perte de $-c_t$ due à l'entraînement dans un seul environnement. Dans le cas où l'agent choisit 'Entraînement multi-environnements', cela coûte nc_t, et l'attaquant sélectionne k sur n environnements à empoisonner (où $1 \leq k \leq n$), résultant en un coût de $k \cdot c_a$. En termes de gains, l'attaquant reçoit $\frac{k}{n} \cdot g_a$ des environnements empoisonnés avec succès, tandis que l'agent conserve des environnements non affectés $\frac{n-k}{n} \cdot g_a$. Nous fixons g_a comme le gain de l'attaquant d'une attaque réussie. Nous supposons naturellement que tous ces paramètres sont des valeurs positives. De plus, nous posons que $g_a > c_a$ pour garantir que l'attaquant ait une incitation financière à mener l'attaque. De même, $\frac{n-k}{n} \cdot g_a$ doit être supérieur à $n \cdot c_t$ pour que l'agent soit motivé à adopter l'approche de défense plus robuste de l'entraînement multi-environnements (MET), indiquant que ce qui est protégé justifie l'effort de défense accru.

6.6.2 Mise à jour des croyances

La mise à jour des croyances basée sur les observations : la croyance de l'agent d'apprentissage concernant le statut d'empoisonnement d'un environnement, notée par $\omega_{t+1}(\lambda_i = 1 \mid o_i(t))$, est mise à jour en fonction des actions observées $o_i(t)$ de l'environnement et des croyances précédentes. Cette mise à jour bayésienne est cruciale pour ajuster dynamiquement la stratégie de l'agent en réponse au risque perçu d'empoisonnement :

$$\omega_{t+1}(\lambda_i = 1 \mid o_i(t)) = \frac{\omega_t(\lambda_i = 1)P(o_i(t) \mid \lambda_i = 1)}{P(o_i(t))} \tag{6.14}$$

où $P(o_i(t)) = \omega_t(\lambda_i = 1)P(o_i(t) \mid \lambda_i = 1) + (1 - \omega_t(\lambda_i = 1))P(o_i(t) \mid \lambda_i = 0)$ représente la probabilité totale d'observer l'action $o_i(t)$ sous les deux types possibles d'environnement. Cette formule calcule la probabilité postérieure que l'environnement soit empoisonné étant donné l'observation actuelle $o_i(t)$.

Les différentes valeurs d'utilité, conditionnées aux choix des deux joueurs, sont délimitées dans le tableau suivant.

TABLE 6.1 – Forme stratégique du modèle de jeu Bayésien pour la sécurité dans DRL

Profil des types $\lambda = (1, 0)$		
Stratégies	SET	MET
Poisonner (P)	$(g_a - c_a, -c_t - g_a)$	$\left(\frac{k}{n}g_a - kc_a, \frac{n-k}{n}g_a - nc_t\right)$
Ne Pas Poisonner (NP)	$(0, -c_t)$	$(0, -nc_t)$
Profil des types $\lambda = (0, 0)$		
Ne Pas Poisonner (NP)	$(0, -c_t)$	$(0, -nc_t)$

Nous établissons les éléments clés et les scénarios possibles qui peuvent survenir pendant le jeu pour chaque joueur. Notre cadre implique un agent d'apprentissage qui interagit avec des environnements potentiellement empoisonnés. Dans ce contexte, nous désignons le joueur de l'environnement comme le joueur émetteur, noté par i, qui possède des informations privées sur son type, qui peut être soit "normal" ($\lambda_i = 0$) soit "empoisonné" ($\lambda_i = 1$). Le statut d'empoisonnement du joueur i reste inconnu de l'agent j, qui est intrinsèquement de type normal ($\lambda_j = 0$), et ce fait est de connaissance commune pour les deux joueurs.

Au début du jeu, le joueur agent a des informations préalables sur le statut d'empoisonnement du joueur environnement, encapsulées dans la probabilité préalable ω_0 qui se situe dans l'intervalle $[0, 1]$.

Chaque joueur choisit une action alignée avec son objectif prédéfini. Le joueur environnement, de type empoisonné, décide entre 'Empoisonner l'Environnement' (P) ou 'Ne Pas Empoisonner' (NP), tandis que si le joueur environnement est de type normal, l'action se limite à 'Ne Pas Empoisonner' (NP). L'agent d'Apprentissage dispose de deux actions possibles : 'Entraînement multi-environnements' (MET) et 'Entraînement en Environnement Unique' (SET).

Dans la configuration stratégique de notre modèle de jeu Bayésien pour la sécurité dans le DRL, nous dérivons l'équilibre de Nash Bayésien en stratégies mixtes. Cet équilibre est caractérisé par les probabilités α et β, où α représente la probabilité que l'attaquant choisisse d'empoisonner l'environnement (stratégie P), lorsque son type est empoisonné ($\delta = 1$) et $1-\alpha$ la probabilité de ne pas introduire de perturbation dans le système de récompense (stratégie NP). En revanche, β dénote la probabilité que le défenseur opte pour s'entraîner dans un seul environnement (stratégie SET), et $1 - \beta$ représente la probabilité d'adopter une formation dans plusieurs environnements (stratégie MET).

6.6.3 Équilibre de Nash Bayésien

Notre analyse détermine les stratégies d'équilibre, désignées par α^* et β^*, qui optimisent les récompenses attendues en considérant les coûts et les bénéfices associés à chaque action. Ces stratégies, qui représentent les probabilités avec lesquelles chaque joueur s'engage dans ses actions respectives, garantissent la maximisation de leur utilité attendue compte tenu du comportement stratégique de l'autre joueur. L'adoption de ces probabilités empêche tout joueur de modifier unilatéralement sa stratégie sans se désavantager, renforçant ainsi la stabilité stratégique qui est caractéristique d'un équilibre de Nash bayésien.

α^* et β^* sont calculés comme suit :

$$\alpha^* = \frac{c_t(\omega n - 1)}{\omega g_a \left(\frac{n-k}{n}\right)} \tag{6.15}$$

$$\beta^* = \frac{k c_a - \frac{k}{n} g_a}{c_a(k-1) + g_a \left(1 - \frac{k}{n}\right)} \tag{6.16}$$

α^* illustre la probabilité que l'attaquant choisisse d'empoisonner l'environnement, tandis que β^* et $1 - \beta^*$ indiquent respectivement les probabilités que l'agent DRL soit formé dans un environnement unique ou dans un cadre multi-environnemental. Un point intéressant de notre modèle est que α^* est inversement proportionnel à ω, la croyance de l'agent concernant le risque d'empoisonnement. Cela signifie que plus l'agent perçoit un risque d'empoisonnement élevé (ω), moins l'attaquant est enclin à empoisonner l'environnement pour éviter d'être détecté. À l'inverse, si la croyance de l'agent concernant l'empoisonnement est faible, l'attaquant peut se sentir plus motivé à manipuler l'environnement, exploitant ainsi la négligence ou le manque de suspicion de l'agent. Cette interaction souligne l'importance cruciale de la perception et de la contre-manipulation dans la stratégie de défense.

Notre idée est de lier le seuil de détection, décrit dans l'Équation 6.12, qui délimite les conditions sous lesquelles nous déterminons qu'une récompense a été empoisonnée, directement avec la probabilité optimale que l'environnement soit empoisonné. Pour initier ce processus, nous devons transformer la valeur de variance en une mesure de probabilité, en utilisant la technique de distribution normale gaussienne. Cette approche nous permettra d'ajuster dynamiquement la sensibilité de notre mécanisme de détection, en fonction de la probabilité d'une attaque, telle qu'indiquée par l'équilibre bayésien. Cette adaptation dynamique du seuil garantit que notre stratégie de défense reste robuste et réactive face à l'évolution du paysage des menaces, améliorant ainsi considérablement la sécurité du processus d'apprentissage de l'agent.

$$P(Z \leq \frac{r_i - \bar{r}}{\sigma}) > \alpha^* \tag{6.17}$$

où σ est l'écart-type et Z la variable normale standardisée.

Comme le seuil dynamique, noté α^*, est fondamentalement influencé par la croyance de l'agent d'apprentissage approfondi concernant l'étendue de l'empoisonnement environnemental. Une croyance accrue abaisse le seuil, incitant à une transition rapide vers une formation multi-environnements pour atténuer les risques associés à une contamination potentielle des récompenses. À l'inverse, un niveau de suspicion plus faible élève le seuil, indiquant une approche plus conservatrice où l'agent prolonge la formation dans un environnement unique. Ce seuil adaptatif reflète la réponse stratégique de l'agent à la probabilité perçue d'attaque, équilibrant le compromis entre l'urgence d'une formation robuste et l'efficacité des coûts de la formation continue en environnement unique.

6.7 Protocole de défense contre les attaques d'empoisonnement des récompenses dans le DRL

Notre mécanisme de défense contre les attaques d'empoisonnement des récompenses implique l'interaction de trois entités : l'environnement $e_i \in \mathcal{E}$, l'agent et l'attaquant. Pour mettre en œuvre cette approche, nous proposons l'Algorithme 3, qui décrit les étapes clés de notre approche utilisant une formation multi-environnements pour se défendre contre de telles attaques.

Remarque : le paramètre ν représente le taux d'apprentissage utilisé dans la mise à jour de la politique de l'agent de renforcement profond. Ce taux d'apprentissage contrôle l'ampleur de l'ajustement apporté à la politique de l'agent à chaque étape de l'entraînement, en fonction des gradients accumulés des politiques calculées pour chaque environnement. Un ν plus élevé peut conduire à des ajustements plus rapides mais potentiellement moins stables de la politique, tandis qu'un ν plus faible peut stabiliser le processus d'apprentissage mais au prix d'une convergence plus lente. Le choix de ν est crucial pour équilibrer l'efficacité de l'apprentissage et la stabilité du comportement de l'agent dans des environnements potentiellement perturbés par des actions adverses.

6.7.1 Évaluation expérimentale

Dans cette section, nous procédons à une évaluation de notre stratégie de défense dans un environnement simulé, conçu pour tester son efficacité et sa robustesse. Nous avons mené une série d'expériences pour examiner en détail comment notre modèle de formation multi-environnemental fait face aux attaques d'empoisonnement des récompenses. Les simulations visent à reproduire les défis adverses rencontrés par les systèmes DRL dans des situations réelles, offrant ainsi une analyse détaillée de la résilience et de la capacité d'adaptation de notre stratégie.

Nous utilisons un environnement de grille modifié, tel que conceptualisé par [113], pour tester notre stratégie de défense. Cet environnement est un microcosme de 18 états distincts, offrant chacun quatre actions potentielles : haut, bas, droite et gauche. Le terrain est une mosaïque de cellules blanches, grises et bleues, entrecoupées de cellules noires impénétrables qui forment les murs de la grille. La mission de l'agent est claire : naviguer dans cet environnement, en s'efforçant d'accumuler des récompenses tout en faisant face à la nature

Algorithme 3 Mécanisme de défense contre l'empoisonnement des récompenses dans le DRL

Configurations :

- Ensembles de n environnements $e_1, e_2, \ldots, e_n$.
- Paramètres MDP pour chaque environnement e_i : $(\mathcal{S}, \mathcal{A}, \mathcal{P}_i, R, \eta)$.
- Configurations de l'agent : $(Q_0, \epsilon, \gamma, \alpha)$.
- Configurations de l'adversaire : $(\pi^{\dagger}, \Delta^t, \upsilon)$.
- Paramètres du jeu : c_t, c_a, k, g_a et croyance initiale $\omega = 0.5$.
- Coefficient d'apprentissage ν.

Initialisation :

- $t = 0$.
- Déterminer α^8 et β^8 à partir du jeu Bayésien.
- Initialiser $Gr = 0$ pour l'accumulateur de gradient.

```
Formation multi-environnements initiale :
Pour t = 1 to T faire
    Pour chaque environnement e_i dans e_1, e_2, ..., e_n faire
        Collecter les trajectoires 𝒯_i.
        Calculer ∇π_i à partir de 𝒯_i.
        Gr = Gr + ∇π_i.
    fin Pour
    π = π − ν · Gr/n.
    Estimer la variance des récompenses pour chaque environnement.
    Mise à jour de ω basée sur la variance observée.
    Recalculer α* avec la nouvelle croyance ω.
    Si la variance dépasse α* alors
        Continuer l'entraînement multi-environnements.
    sinon
        Sélectionner aléatoirement un environnement e_i et s'entraîner.
    fin Si
fin Pour
Mise à jour continue de la politique :
Pour tant que condition d'arrêt non atteinte faire
    Estimer la variance des récompenses.
    Ajuster la politique π en fonction de la variance et des résultats du jeu Bayésien.
fin Pour
```

stochastique de l'environnement où il y a 90 % de chances de se déplacer comme prévu et 10 % de chances de transition aléatoire.

La structure des récompenses est conçue pour refléter les complexités de la prise de décision dans le monde réel : les états blancs attribuent une récompense minimale de -1, tandis que les états gris infligent une pénalité plus conséquente de -10, et les précieux états bleus offrent une récompense de 2. Une récompense constante de 0 pour occuper un état bleu assure la motivation continue de l'agent à atteindre ces cibles. De manière significative, les flèches audacieuses dans notre représentation visuelle délimitent la politique idéale de l'attaquant, établissant un objectif adversaire clair dans la simulation. Partant de l'état initial s_0, l'agent est chargé de maximiser sa récompense cumulative, avec un facteur de réduction γ fixé à 0.9, mettant en avant la préférence pour des gains immédiats par rapport aux bénéfices futurs. Cette configuration ne simule pas seulement un défi réaliste pour l'agent, mais sert également de terrain d'essai rigoureux pour tester la robustesse et l'adaptabilité de notre stratégie de défense proposée face aux risques d'empoisonnement des récompenses.

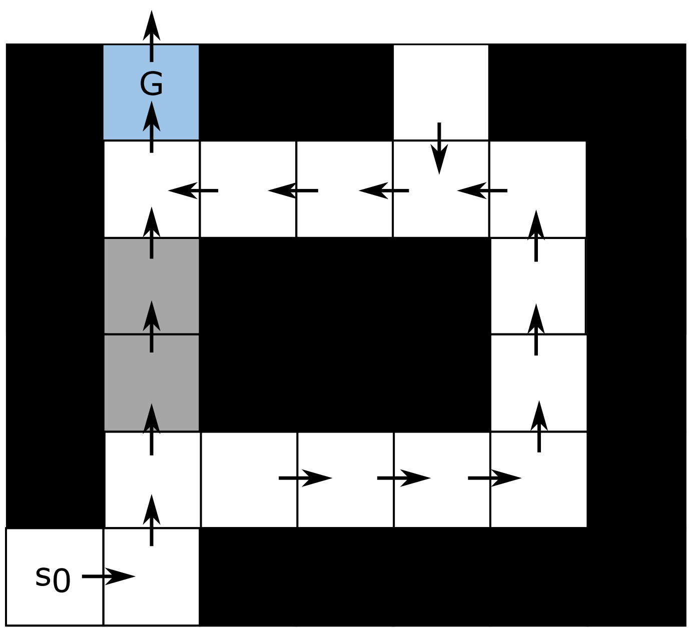

FIGURE 6.2 – Environnement de grille.

Dans notre étude expérimentale, nous avons établi trois scénarios pour évaluer notre approche de résilience des systèmes DRL face aux attaques d'empoisonnement des récompenses. Le premier scénario, « Seuil Bas » (alpha = 0,1), simule une approche de défense conservatrice où l'agent suspecte fréquemment un empoisonnement. Cela entraîne des basculements fréquents vers la formation multi-environnementale, ce qui peut ralentir l'apprentissage en raison d'une prudence accrue. Le deuxième scénario, « Seuil Élevé » (alpha = 0,9), représente une situation où l'agent suspecte rarement un empoisonnement, privilégiant l'entraînement principalement dans un seul environnement pour accélérer l'apprentissage, mais au risque de diminuer la détection efficace. Le troisième scénario, « Seuil Dynamique », ajuste dynamiquement le seuil de suspicion en fonction des actions de l'attaquant, cherchant à équilibrer optimalement la détection de l'empoisonnement et l'efficacité de la formation. L'efficacité de ces approches sera évaluée selon deux critères principaux : le taux de détection des récompenses empoisonnées et la vitesse d'apprentissage au cours de différents cycles de formation.

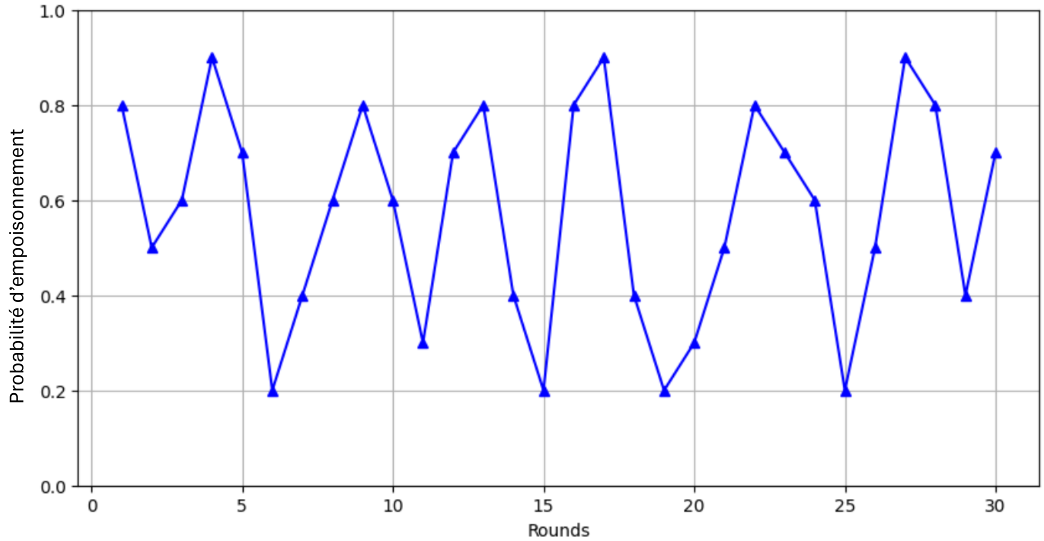

FIGURE 6.3 – Comportement de l'attaquant à travers les tours.

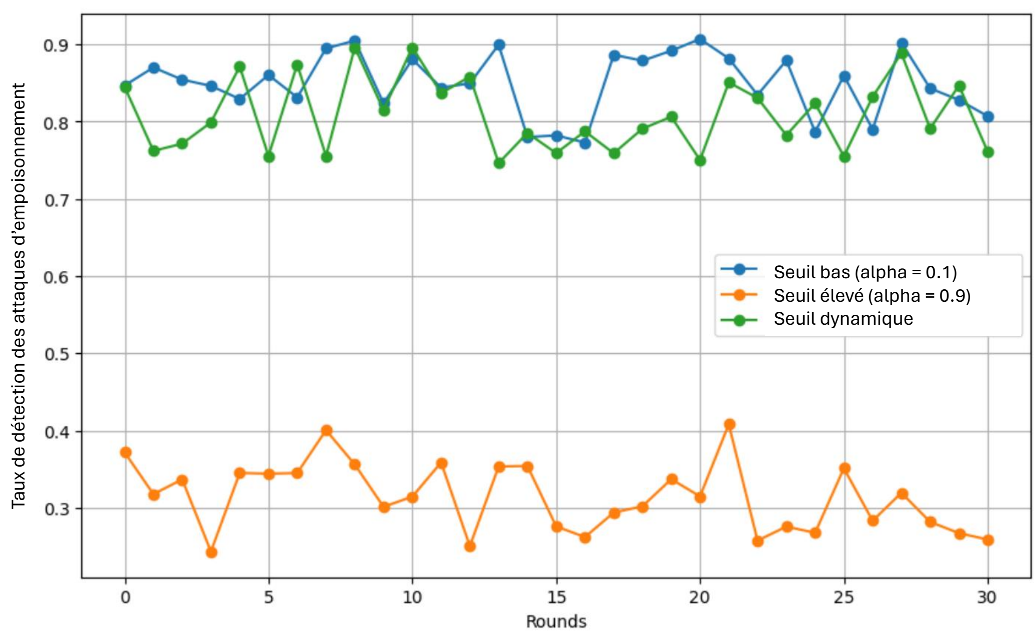

FIGURE 6.4 – Comparaison de la détection des récompenses empoisonnées entre les scénarios.

Nos résultats expérimentaux, issus de l'analyse des trois scénarios—seuil bas, seuil élevé et seuil dynamique—offrent une évaluation détaillée qui met en évidence l'efficacité de notre stratégie de défense dynamique dans l'apprentissage par renforcement profond face à divers niveaux de menace. Dans le scénario de seuil bas (alpha = 0,1), caractérisé par une sensibilité

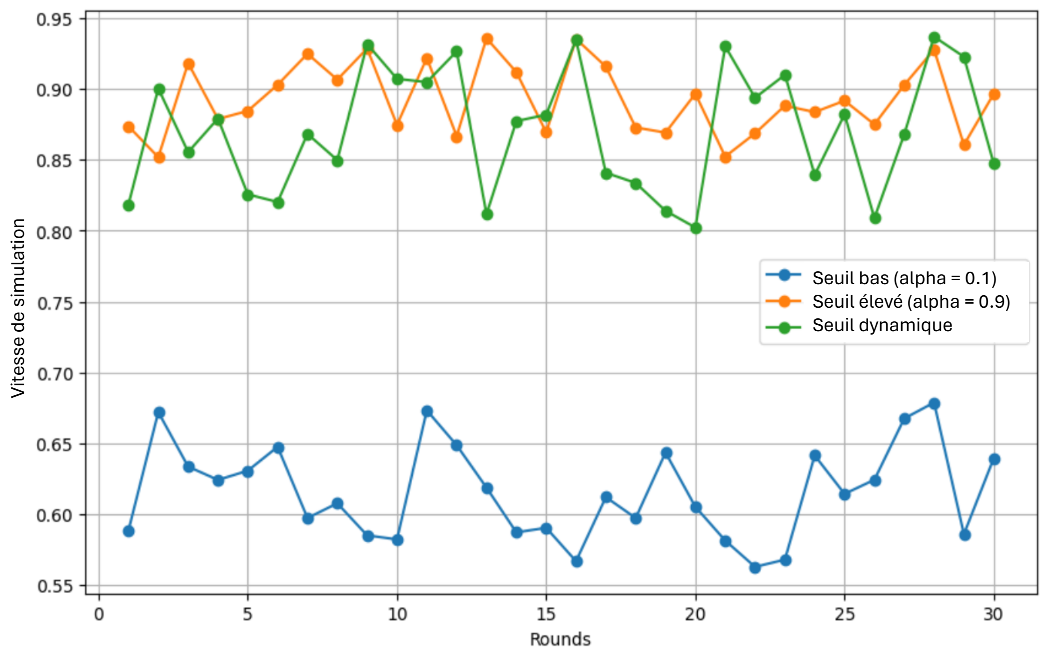

FIGURE 6.5 – Vitesse de simulation à travers les tours.

accrue à la détection, nous constatons des taux élevés de détection de récompenses empoisonnées, ce qui confirme son efficacité en matière de sécurité. Toutefois, cette sensibilité se traduit par des vitesses de simulation plus lentes en raison de l'activation fréquente de la formation multi-environnementale. À l'opposé, le scénario de seuil élevé (alpha = 0,9), où le risque d'attaque est perçu comme plus faible, affiche les vitesses de simulation les plus rapides, mais au détriment de taux de détection plus bas, révélant des vulnérabilités potentielles face à des attaques adverses non détectées.

Le scénario de seuil dynamique, qui ajuste la sensibilité de détection en fonction du comportement observé de l'attaquant, parvient à un équilibre optimal. Il atteint des taux de détection élevés comparables à ceux du seuil bas, tout en maintenant des vitesses de simulation proches de celles observées dans le scénario de seuil élevé. Ce résultat démontre que notre méthode s'adapte efficacement aux niveaux de menace fluctuants et optimise le compromis entre sécurité et performance, en capturant avec succès le comportement stratégique de l'adversaire ainsi que l'état des environnements au travers des croyances mises à jour et des équilibres de Nash Bayésien calculés. Cette harmonisation illustre les avantages pratiques de l'intégration des principes des jeux Bayésiens dans les stratégies défensives pour renforcer à la fois la sécurité et l'efficacité opérationnelle dans des environnements potentiellement hostiles.

Ces résultats empiriques constituent une base solide pour l'adoption et le raffinement continu de stratégies de défense avancées dans les systèmes d'apprentissage par renforcement profond. Ils soulignent clairement le potentiel de notre approche à accroître la résilience face aux manipulations malicieuses tout en préservant l'efficacité nécessaire dans des scénarios réalistes.

6.8 Conclusion

Dans ce travail, nous avons abordé le défi critique de défense des systèmes d'apprentissage par renforcement profond (DRL) contre la menace des attaques par empoisonnement des récompenses. Nous avons développé un cadre d'apprentissage multi-environnements qui non seulement renforce la robustesse des politiques d'apprentissage, mais s'adapte également de manière stratégique aux comportements adverses grâce à des concepts tirés de la théorie des jeux bayésiens. Notre méthode, en incorporant de manière aléatoire la sélection des environnements d'entraînement, dilue systématiquement l'influence des récompenses empoisonnées. Par ailleurs, l'utilisation des techniques de réduction de variance permet en outre à l'agent de détecter et de mitiger efficacement les effets des interventions malveillantes.

Cette combinaison d'entraînement multi-environnements avec des principes de la théorie des jeux crée un ensemble d'outils pionnier pour la sécurité dans le domaine du DRL. Cet ensemble ne se contente pas de mitiger les risques potentiels, mais il garantit aussi l'intégrité et la fiabilité des algorithmes d'apprentissage—des exigences cruciales pour le déploiement du DRL dans des secteurs où la sécurité est primordiale. Nos simulations ont démontré l'efficacité de notre approche, qui permet de choisir le mode d'entraînement selon un seuil dynamique reflétant l'état de l'empoisonnement des récompenses. Cette capacité d'ajuster assure une application efficace et réaliste de notre modèle dans divers scénarios pratiques, augmentant ainsi la résilience des systèmes DRL face à de telles attaques.

À l'avenir, nous visons à confirmer davantage l'efficacité de notre mécanisme de défense en menant des simulations ciblées à travers une variété d'environnements. Nous planifions de tester notre approche dans des contextes à la fois discrets—en utilisant des environnements tels que CartPole, LunarLander, MountainCar et Acrobot—et continus, incluant HalfCheetah, Hopper, Walker2d et Swimmer. Ces expériences sont conçues pour évaluer notre approche par rapport aux défenses existantes, démontrant sa capacité à préserver la fonctionnalité de l'agent même sous la pression de menaces adverses. Nos recherches futures devraient enrichir notre compréhension et inciter à de nouvelles améliorations, consolidant la résistance des systèmes DRL contre les menaces de cybersécurité évolutives.

Chapitre 7

Conclusion générale

La recherche sur la cybersécurité des modèles d'apprentissage automatique revêt une importance capitale. Ces modèles ont véritablement révolutionné de nombreux domaines en offrant des performances et une précision exceptionnelles. Cependant, leur vulnérabilité du point de vue de la cybersécurité peut avoir des conséquences désastreuses, notamment en matière de protection de la vie privée. En effet, la conception et la mise en place de modèles d'apprentissage automatique robustes et sécurisés sont confrontées à de nombreux enjeux et défis complexes. Il est donc crucial de les surmonter pour garantir la confiance des utilisateurs des systèmes qui reposent sur ces modèles, et pour favoriser une adoption et une utilisation plus généralisée de ces technologies novatrices.

Dans le chapitre 2 de cette thèse, nous avons exploré les différentes catégories d'apprentissage automatique, allant de l'apprentissage supervisé au non supervisé, en passant par l'apprentissage par renforcement. Nous avons fourni une description formelle des processus d'apprentissage pour chacune de ces catégories, ainsi que des algorithmes d'optimisation correspondants. Afin de mieux saisir les défis de cybersécurité rencontrés par ces modèles, nous avons détaillé leurs vulnérabilités du point de vue de la sécurité. Nous avons également établi une taxonomie des attaques ciblant les modèles d'apprentissage automatique, classifiant ces attaques en fonction des objectifs, des capacités et des connaissances de l'attaquant. Les attaques sont catégorisées selon le moment de leur occurrence durant le cycle d'apprentissage et selon la propriété de sécurité qu'elles compromettent. Cette analyse approfondie nous a permis de mieux appréhender les vulnérabilités, ainsi que les risques encourus par les modèles, leurs propriétaires et les utilisateurs.

Dans le chapitre 3, nous avons effectué une étude critique approfondie de l'état de l'art sur la sécurité des modèles d'apprentissage automatique. Nous avons classé les attaques qui visent ces modèles en trois catégories : les attaques par inférence, les attaques par empoisonnement et les attaques par manipulation. Dans la première catégorie, nous avons défini formellement les attaques par inférence d'appartenance et présenté quelques techniques d'élaboration de ces attaques. Nous avons ensuite présenté et critiqué les techniques de défense existantes contre ce type d'attaques. La seconde catégorie d'attaques que nous avons présentée concerne les attaques par empoisonnement, dont l'attaque la plus courante est celle par empoisonnement. Nous avons examiné les différentes manières d'empoisonner les éléments du modèle, que ce soit les données, les étiquettes ou même les récompenses dans le cas de l'apprentissage par renforcement. Nous avons également présenté une étude des techniques de défense qui sont classées en défense passive et active. Dans le cas des techniques de défense passive, il s'agit d'utiliser des filtres ou d'assainir les données, alors que pour les attaques actives, il y a un réel effort de la part du modèle pour devenir plus robuste même en présence de données empoisonnées dans les différentes phases du processus. Enfin, la dernière catégorie d'attaques que nous avons présentée était les attaques par manipulation qui se produisent lorsque le modèle est déployé (exemples contradictoires). Nous avons également présenté une série de techniques pour atténuer les risques de ces attaques sur la prédiction des modèles. Cette étude critique de l'état de l'art nous a permis d'identifier les défis clés de cybersécurité auxquels font face les modèles d'apprentissage automatique.

Dans le chapitre 4 de cette thèse, nous avons développé et implémenté des stratégies défensives innovantes pour protéger les modèles d'apprentissage automatique contre les attaques par inférence d'appartenance, en particulier dans les environnements de "Machine Learning as a Service" (MLaaS). Cette problématique est d'autant plus critique que les environnements MLaaS, facilement accessibles via des interfaces de programmation d'applications (API), exposent potentiellement des informations sensibles concernant les données sur lesquelles ces modèles ont été entraînés.
Notre contribution, détaillée dans ce chapitre, repose sur l'introduction d'un mécanisme de défense qui consiste à ajouter stratégiquement du bruit au vecteur de prédiction retourné lors de chaque requête. Cette approche perturbe la précision avec laquelle un attaquant peut déterminer si une donnée spécifique a fait partie de l'ensemble d'entraînement du modèle. Le bruit est conçu pour placer la probabilité d'inférence de l'attaquant dans une zone d'incertitude, tout en minimisant les perturbations pour l'utilisateur légitime. Ce double objectif a été formulé comme un problème d'optimisation complexe avec contraintes, cherchant le meilleur compromis entre la confidentialité et l'utilité.
Nous avons validé par des expériences que l'ajout de ce bruit optimal réduit effectivement la capacité de l'attaquant à faire des inférences correctes sur la présence de données dans le jeu d'entraînement, sans compromettre la précision des prédictions du modèle. Ces résultats démontrent que notre méthode améliore la résilience aux attaques par inférence d'appartenance et préserve l'intégrité ainsi que la fiabilité des prédictions du modèle, ce qui est crucial dans des domaines sensibles tels que le diagnostic médical ou la détection de fraudes.

Le chapitre 5 de cette thèse approfondit notre analyse de la sécurité dans les environnements de MLaaS, en mettant l'accent sur un risque souvent sous-estimé : les attaques par inférence d'attribut (AIA). Ces attaques exploitent des connaissances partielles sur les données d'entraînement pour inférer des attributs sensibles à partir des prédictions des modèles. C'est une technique proche de l'imputation de données mais exploitée de manière malveillante. Nous avons développé un modèle de défense qui exploite l'aspect trompeur des exemples adverses, pour accroître la robustesse des modèles dans les environnements de MLaaS. En effet, nous avons transformé les exemples adverses, traditionnellement perçus comme offensifs, en outils de défense pour tromper les attaquants. Nos simulations ont confirmé l'efficacité de cette méthode, prouvant sa robustesse à travers différents ensembles de données et affirmant sa viabilité dans des scénarios réels.
Notre approche équilibre entre sécurité et utilité, illustrant un modèle aux allures pratiques où la sécurité des données et l'utilité des modèles d'apprentissage automatique ne sont pas des objectifs contradictoires mais plutôt complémentaires. Nos travaux ouvrent des pistes prometteuses pour la conception future de solutions de cybersécurité dans les environnements

MLaaS, un domaine qui continue de se développer à un rythme soutenu.

Le chapitre 6 aborde la sécurité des systèmes d'apprentissage par renforcement profond (DRL) en ligne, avec un accent particulier sur les attaques par empoisonnement des récompenses. Nous avons examiné un scénario d'attaque en boîte blanche, où l'attaquant possède une compréhension détaillée de l'environnement d'apprentissage. Cette étude révèle une contrainte majeure de l'entraînement en ligne : l'absence de références fiables rend difficile pour l'agent d'apprentissage la distinction entre les récompenses légitimes et celles manipulées. Pour surmonter ce défi, nous avons conçu une architecture d'entraînement multi-environnements qui enrichit le processus d'apprentissage en permettant à l'agent d'apprentissage de réaliser des comparaisons fiables pour identifier les récompenses altérées.
Afin de rendre notre modèle plus pragmatique, nous avons intégré la capacité pour l'agent d'estimer le risque d'empoisonnement des récompenses, déclenchant ainsi l'entraînement multi-environnements basé sur ce niveau de risque évalué. De plus, nous avons modélisé les interactions stratégiques entre les acteurs du système, l'attaquant et l'agent d'apprentissage, sous forme de jeu Bayésien avec information incomplète. Cette modélisation permet à la défense d'ajuster sa stratégie en réaction aux actions de l'adversaire, offrant une compréhension profonde des ajustements stratégiques possibles et des équilibres potentiels entre les deux parties.
Nos simulations ont démontré l'efficacité de notre approche qui permet de choisir le mode d'entraînement selon un seuil dynamique reflétant l'état de l'empoisonnement des récompenses. Cette capacité ajustable assure une application efficace et réaliste de notre modèle dans divers scénarios pratiques, augmentant la résilience des systèmes DRL face à de telles attaques, ce qui est crucial pour des applications critiques telles que la navigation autonome ou la gestion des systèmes financiers.
Les expérimentations réalisées ont démontré que notre architecture défensive améliore non seulement la résilience aux attaques par empoisonnement, mais maintient également, voire améliore, la performance générale du système en termes de précision et de fiabilité des prédictions. Ces résultats soulignent l'efficacité de notre approche dans un contexte de défense contre les attaques sophistiquées, offrant ainsi une stratégie viable pour sécuriser les modèles d'apprentissage automatique.

Ces contributions marquent un progrès significatif dans notre compréhension et la mitigation des risques de sécurité liés aux modèles d'apprentissage automatique. Néanmoins, le domaine reste riche en opportunités de recherche, notamment pour renforcer les mécanismes de défense face aux évolutions constantes des menaces. Il est donc important de développer des méthodes de défense qui non seulement réagissent en temps réel aux attaques mais qui sont également capables de s'adapter aux stratégies d'attaque de plus en plus sophistiquées, souvent pilotées par l'intelligence artificielle elle-même.

Une piste intéressante à explorer est l'intégration des jeux à champs moyens (Mean Field Games : MFG) dans notre modèle d'entraînement multi-environnements. Ceci pourrait substantiellement améliorer la manière dont nous abordons la sécurité dans l'apprentissage par renforcement en ligne. En effet, contrairement à la théorie des jeux traditionnelle, les MFG offrent des outils qui permettent de calculer des équilibres plus stables avec une population de joueurs (les environnements dans notre cas). Avec l'adoption du paradigme MFG, nous pourrions simplifier et optimiser le traitement des interactions entre l'agent d'apprentissage et une multitude d'environnements potentiellement empoisonnés. Cela permettrait non seulement de réduire la complexité de calcul des interactions individuelles, mais aussi de formuler une politique de réponse moyenne qui s'adapte aux comportements moyens de l'ensemble des environnements. De plus, les MFG facilitent la gestion de systèmes de grande échelle en réduisant la dimensionnalité des problèmes à résoudre, ce qui est particulièrement bénéfique pour modéliser efficacement des scénarios avec de nombreux environnements.

Par ailleurs, il serait intéressant de se pencher sur la sécurité dans le contexte de l'apprentissage fédéré. Ce paradigme, caractérisé par la décentralisation du processus d'entraînement et la diversité des données dispersées à travers plusieurs nœuds, présente des défis uniques en termes de sécurité. Identifier et comprendre les types d'attaques spécifiques à cet environnement, telles que l'empoisonnement de modèle ou l'inférence de données, est crucial. Développer des stratégies de défense théoriques et pratiques qui sont spécifiquement conçues pour l'apprentissage fédéré pourrait donc aider à protéger les données sensibles dans des secteurs critiques comme la santé et la finance.

Bibliographie

[1] Census19 data set. https ://www.census.gov/programs-surveys/acs/ Accessed : 2023-08-24.

[2] The road to secure and trusted ai, the decade of ai security challenges, version 1.1. 2021.

[3] Ia responsable et fiable. https ://learn.microsoft.com/fr-fr/azure/cloud-adoption-framework/innovate/best-practices/trusted-ai, Mis en ligne le 01 décembre 2022, consulté le 02 décembre 2022.

[4] Ia et machine learning dans le domaine de la cybersécurité. https ://www.kaspersky.fr/resource-center/definitions/ai-cybersecurity, Mis en ligne le 19 juillet 2022, consulté le 23 novembre 2022.

[5] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318, 2016.

[6] Naomi S Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3) :175–185, 1992.

[7] Amazon Web Services. *Amazon Machine Learning : Developer Guide*. Amazon Web Services, 2018. Accessed : [Juin 2024].

[8] Arthur Asuncion and David Newman. Uci machine learning repository, 2007.

[9] Brendan Avent, Aleksandra Korolova, David Zeber, Torgeir Hovden, and Benjamin Livshits. {BLENDER} : Enabling local search with a hybrid differential privacy model. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 747–764, 2017.

[10] Kiarash Banihashem, Adish Singla, and Goran Radanovic. Defense against reward poisoning attacks in reinforcement learning. *arXiv preprint arXiv :2102.05776*, 2021.

[11] Marco Barreno, Blaine Nelson, Anthony D Joseph, and J Doug Tygar. The security of machine learning. *Machine Learning*, 81(2) :121–148, 2010.

[12] Marco Barreno, Blaine Nelson, Russell Sears, Anthony D Joseph, and J Doug Tygar. Can machine learning be secure ? In *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, pages 16–25, 2006.

[13] Raef Bassily, Adam Smith, and Abhradeep Thakurta. Private empirical risk minimization : Efficient algorithms and tight error bounds. In *2014 IEEE 55th annual symposium on foundations of computer science*, pages 464–473. IEEE, 2014.

[14] Raef Bassily, Om Thakkar, and Abhradeep Guha Thakurta. Model-agnostic private learning. *Advances in Neural Information Processing Systems*, 31, 2018.

[15] Vahid Behzadan and Arslan Munir. Whatever does not kill deep reinforcement learning, makes it stronger. *arXiv preprint arXiv :1712.09344*, 2017.

[16] Vahid Behzadan and Arslan Munir. The faults in our pi stars : Security issues and open challenges in deep reinforcement learning. *arXiv preprint arXiv :1810.10369*, 2018.

[17] Vahid Behzadan and Arslan Munir. Mitigation of policy manipulation attacks on deep q-networks with parameter-space noise. In *Computer Safety, Reliability, and Security : SAFECOMP 2018, Västerås, Sweden, September 18, 2018, Proceedings 37*, pages 406–417. Springer, 2018.

[18] Battista Biggio, Giorgio Fumera, and Fabio Roli. Security evaluation of pattern classifiers under attack. *IEEE transactions on knowledge and data engineering*, 26(4) :984–996, 2013.

[19] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. *arXiv preprint arXiv :1206.6389*, 2012.

[20] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[21] Eitan Borgnia, Valeriia Cherepanova, Liam Fowl, Amin Ghiasi, Jonas Geiping, Micah Goldblum, Tom Goldstein, and Arjun Gupta. Strong data augmentation sanitizes poisoning and backdoor attacks without an accuracy tradeoff. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3855–3859. IEEE, 2021.

[22] M. Bouhaddi, M. S. Radjef, and K. Adi. An efficient intrusion detection in resource-constrained mobile ad-hoc networks. *Computers & Security*, 2018.

[23] Myria Bouhaddi and Kamel Adi. Mitigating membership inference attacks in machine learning as a service. In *2023 IEEE International Conference on Cyber Security and Resilience (CSR)*, pages 262–268. IEEE, 2023.

[24] Myria Bouhaddi and Kamel Adi. Multi-environment training against reward poisoning attacks on deep reinforcement learning. In *20th international Conference on Security and Cryptography (SECRYPT)*, 2023.

[25] Myria Bouhaddi and Kamel Adi. Enhancing privacy in machine learning : A robust approach for preventing attribute inference attacks. In *21st international Conference on Security and Cryptography (SECRYPT)*, 2024.

[26] Myria Bouhaddi and Kamel Adi. When rewards deceive : Counteracting reward poisoning on online deep reinforcement learning. In *2024 IEEE International Conference on Cyber Security and Resilience (CSR)*. IEEE, 2024.

[27] Kanting Cai, Xiangbin Zhu, and Zhaolong Hu. Reward poisoning attacks in deep reinforcement learning based on exploration strategies. *Neurocomputing*, 553 :126578, 2023.

[28] N. Carlini, C. Liu, Erlingsson Ú, J. Kos, and D. Song. The secret sharer : Evaluating and testing unintended memorization in neural networks. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 267–284, 2019.

[29] Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12(3), 2011.

[30] Dingfan Chen, Ning Yu, Yang Zhang, and Mario Fritz. Gan-leaks : A taxonomy of membership inference attacks against generative models. In *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*, pages 343–362, 2020.

[31] Jianguo Chen, Kenli Li, and S Yu Philip. Privacy-preserving deep learning model for decentralized vanets using fully homomorphic encryption and blockchain. *IEEE Transactions on Intelligent Transportation Systems*, 23(8) :11633–11642, 2021.

[32] Qingrong Chen, Chong Xiang, Minhui Xue, Bo Li, Nikita Borisov, Dali Kaarfar, and Haojin Zhu. Differentially private data generative models. *arXiv preprint arXiv :1812.02274*, 2018.

[33] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv :1712.05526*, 2017.

[34] Yifang Chen, Simon Du, and Kevin Jamieson. Improved corruption robust algorithms for episodic reinforcement learning. In *International Conference on Machine Learning*, pages 1561–1570. PMLR, 2021.

[35] Christopher A Choquette-Choo, Florian Tramer, Nicholas Carlini, and Nicolas Papernot. Label-only membership inference attacks. In *International conference on machine learning*, pages 1964–1974. PMLR, 2021.

[36] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *international conference on machine learning*, pages 1310–1320. PMLR, 2019.

[37] Marshall Copeland, Julian Soh, Anthony Puca, Mike Manning, and David Gollob. *Microsoft Azure : planning, deploying, and managing your data center in the cloud.* Springer, 2015.

[38] Elliot J Crowley, Gavin Gray, and Amos J Storkey. Moonshine : Distilling with cheap convolutions. *Advances in Neural Information Processing Systems*, 31, 2018.

[39] Imre Csiszár. I-divergence geometry of probability distributions and minimization problems. *The annals of probability*, pages 146–158, 1975.

[40] Guy Degla and Jean Koudi. *Les Multiplicateurs de Lagrange en Dimension Finie : Optimisations sous Contraintes.* Éditions universitaires européennes, 2013.

[41] Ambra Demontis, Marco Melis, Maura Pintor, Jagielski Matthew, Battista Biggio, Oprea Alina, Nita-Rotaru Cristina, Fabio Roli, et al. Why do adversarial attacks transfer ? explaining transferability of evasion and poisoning attacks. In *28th USENIX security symposium*, pages 321–338. USENIX Association, 2019.

[42] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet : A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[43] Flávio du Pin Calmon and Nadia Fawaz. Privacy against statistical inference. In *2012 50th annual Allerton conference on communication, control, and computing (Allerton)*, pages 1401–1408. IEEE, 2012.

[44] Sumeet Dua and Xian Du. *Data mining and machine learning in cybersecurity*. CRC press, 2016.

[45] Cynthia Dwork. A firm foundation for private data analysis. *Communications of the ACM*, 54(1) :86–95, 2011.

[46] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves : Privacy via distributed noise generation. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 486–503. Springer, 2006.

[47] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3-4) :21–407, 2014.

[48] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4) :211–407, 2014.

[49] Gamaleldin F Elsayed, Ian Goodfellow, and Jascha Sohl-Dickstein. Adversarial reprogramming of neural networks. *arXiv preprint arXiv :1806.11146*, 2018.

[50] Dumitru Erhan, Aaron Courville, Yoshua Bengio, and Pascal Vincent. Why does unsupervised pre-training help deep learning ? In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 201–208. JMLR Workshop and Conference Proceedings, 2010.

[51] Fabián Fallas-Moya and Francisco Torres-Rojas. Object recognition using hierarchical temporal memory. In *International Symposium on Intelligent Computing Systems*, pages 1–14. Springer, 2018.

[52] Reuben Feinman, Ryan R Curtin, Saurabh Shintre, and Andrew B Gardner. Detecting adversarial samples from artifacts. *arXiv preprint arXiv :1703.00410*, 2017.

[53] Nic Ford, Justin Gilmer, Nicolas Carlini, and Dogus Cubuk. Adversarial examples are a natural consequence of test error in noise. *arXiv preprint arXiv :1901.10513*, 2019.

[54] M. Fredrikson, S. Jha, and T. Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1322–1333, 2015.

[55] Matthew Fredrikson, Eric Lantz, Somesh Jha, Simon Lin, David Page, and Thomas Ristenpart. Privacy in pharmacogenetics : An {End-to-End} case study of personalized warfarin dosing. In *23rd USENIX security symposium (USENIX Security 14)*, pages 17–32, 2014.

[56] K. Ganju, Q. Wang, W. Yang, C. A. Gunter, and N. Borisov. Property inference attacks on fully connected neural networks using permutation invariant representations. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pages 619–633, 2018.

[57] Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. Allennlp : A deep semantic natural language processing platform. *arXiv preprint arXiv :1803.07640*, 2018.

[58] Justin Gilmer, Ryan P Adams, Ian Goodfellow, David Andersen, and George E Dahl. Motivating the rules of the game for adversarial example research. *arXiv preprint arXiv :1807.06732*, 2018.

[59] Neil Zhenqiang Gong and Bin Liu. You are who you know and how you behave : Attribute inference attacks via users' social friends and behaviors. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 979–995, 2016.

[60] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11) :139–144, 2020.

[61] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv :1412.6572*, 2014.

[62] Google. Cloud machine learning engine documentation, 2018. Accessed : [juin 2024].

[63] Samuel Greydanus, Anurag Koul, Jonathan Dodge, and Alan Fern. Visualizing and understanding atari agents. In *International conference on machine learning*, pages 1792–1801. PMLR, 2018.

[64] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets : Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv :1708.06733*, 2017.

[65] Umang Gupta, Dimitris Stripelis, Pradeep K Lam, Paul Thompson, José Luis Ambite, and Greg Ver Steeg. Membership inference attacks on deep regression models for neuroimaging. In *Medical Imaging with Deep Learning*, pages 228–251. PMLR, 2021.

[66] Jamie Hayes, Luca Melis, George Danezis, and Emiliano De Cristofaro. Logan : Membership inference attacks against generative models. *arXiv preprint arXiv :1705.07663*, 2017.

[67] S Haykin. Neural networks : A comprehensive foundation, prentice hall ptr. *Upper Saddle River, NJ, USA*, 1994.

[68] Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4) :18–28, 1998.

[69] Dan Hendrycks and Kevin Gimpel. Early methods for detecting adversarial images. *arXiv preprint arXiv :1608.00530*, 2016.

[70] Benjamin Hilprecht, Martin Härterich, and Daniel Bernau. Monte carlo and reconstruction membership inference attacks against generative models. *Proc. Priv. Enhancing Technol.*, 2019(4) :232–249, 2019.

[71] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv :1503.02531*, 2(7), 2015.

[72] Ricky Ho. Machine learning model-development lifecycle. https ://octo.vmware.com/machine-learning-model-development-lifecycle/, Mis en ligne le 06 mai 2021, consulté le 05 janvier 2022.

[73] Nils Homer, Szabolcs Szelinger, Margot Redman, David Duggan, Waibhav Tembe, Jill Muehling, John V Pearson, Dietrich A Stephan, Stanley F Nelson, and David W Craig. Resolving individuals contributing trace amounts of dna to highly complex mixtures using high-density snp genotyping microarrays. *PLoS genetics*, 4(8) :e1000167, 2008.

[74] Sanghyun Hong, Yiğitcan Kaya, Ionuţ-Vlad Modoranu, and Tudor Dumitraş. A panda ? no, it's a sloth : Slowdown attacks on adaptive multi-exit neural network inference. *arXiv preprint arXiv :2010.02432*, 2020.

[75] Hongsheng Hu, Zoran Salcic, Lichao Sun, Gillian Dobbie, Philip S Yu, and Xuyun Zhang. Membership inference attacks on machine learning : A survey. *ACM Computing Surveys (CSUR)*, 54(11s) :1–37, 2022.

[76] Junling Hu and Michael P Wellman. Nash q-learning for general-sum stochastic games. *Journal of machine learning research*, 4(Nov) :1039–1069, 2003.

[77] Ling Huang, Anthony D Joseph, Blaine Nelson, Benjamin IP Rubinstein, and J Doug Tygar. Adversarial machine learning. In *Proceedings of the 4th ACM workshop on Security and artificial intelligence*, pages 43–58, 2011.

[78] Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. Adversarial attacks on neural network policies. *arXiv preprint arXiv :1702.02284*, 2017.

[79] Bo Hui, Yuchen Yang, Haolin Yuan, Philippe Burlina, Neil Zhenqiang Gong, and Yinzhi Cao. Practical blind membership inference attack via differential comparisons. *arXiv preprint arXiv :2101.01341*, 2021.

[80] Thomas Humphries, Matthew Rafuse, Lindsey Tulloch, Simon Oya, Ian Goldberg, Urs Hengartner, and Florian Kerschbaum. Differentially private learning does not bound membership inference. *arXiv preprint arXiv :2010.12112*, 2020.

[81] IBM. Ibm data science and machine learning, 2018. Accessed : [Juin 2024].

[82] Matthew Inkawhich, Yiran Chen, and Hai Li. Snooping attacks on deep reinforcement learning. *arXiv preprint arXiv :1905.11832*, 2019.

[83] Roger Iyengar, Joseph P Near, Dawn Song, Om Thakkar, Abhradeep Thakurta, and Lun Wang. Towards practical differentially private convex optimization. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 299–316. IEEE, 2019.

[84] Matthew Jagielski, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. Manipulating machine learning : Poisoning attacks and countermeasures for regression learning. In *2018 IEEE symposium on security and privacy (SP)*, pages 19–35. IEEE, 2018.

[85] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering : a review. *ACM computing surveys (CSUR)*, 31(3) :264–323, 1999.

[86] Bargav Jayaraman. Texas-100x data set. https ://github.com/bargavj/texas100x, 2022. Accessed : 2023-08-24.

[87] Bargav Jayaraman and David Evans. Evaluating differentially private machine learning in practice. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 1895–1912, 2019.

[88] Bargav Jayaraman and David Evans. Are attribute inference attacks just imputation ? In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 1569–1582, 2022.

[89] Bargav Jayaraman, Lingxiao Wang, Katherine Knipmeyer, Quanquan Gu, and David Evans. Revisiting membership inference under realistic assumptions. *arXiv preprint arXiv :2005.10881*, 2020.

[90] Jinyuan Jia, Xiaoyu Cao, and Neil Zhenqiang Gong. Intrinsic certified robustness of bagging against data poisoning attacks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 7961–7969, 2021.

[91] Jinyuan Jia and Neil Zhenqiang Gong. {AttriGuard} : A practical defense against attribute inference attacks via adversarial machine learning. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 513–529, 2018.

[92] Jinyuan Jia, Ahmed Salem, Michael Backes, Yang Zhang, and Neil Zhenqiang Gong. Memguard : Defending against black-box membership inference attacks via adversarial examples. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, pages 259–274, 2019.

[93] Jinyuan Jia, Ahmed Salem, Michael Backes, Yang Zhang, and Neil Zhenqiang Gong. Memguard : Defending against black-box membership inference attacks via adversarial examples. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, pages 259–274, 2019.

[94] Kwang-Sung Jun, Lihong Li, Yuzhe Ma, and Jerry Zhu. Adversarial attacks on stochastic bandits. *Advances in Neural Information Processing Systems*, 31, 2018.

[95] Sarah Keren, Luis Pineda, Avigdor Gal, Erez Karpas, and Shlomo Zilberstein. Equi-reward utility maximizing design in stochastic environments. *HSDIP 2017*, page 19, 2017.

[96] Daniel Kifer, Adam Smith, and Abhradeep Thakurta. Private convex empirical risk minimization and high-dimensional regression. In *Conference on Learning Theory*, pages 25–1. JMLR Workshop and Conference Proceedings, 2012.

[97] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv :1312.6114*, 2013.

[98] Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.

[99] Jernej Kos and Dawn Song. Delving into adversarial attacks on deep policies. *arXiv preprint arXiv :1705.06452*, 2017.

[100] Michal Kosinski, David Stillwell, and Thore Graepel. Private traits and attributes are predictable from digital records of human behavior. *Proceedings of the national academy of sciences*, 110(15) :5802–5805, 2013.

[101] Solomon Kullback. *Information theory and statistics*. Courier Corporation, 1997.

[102] Aounon Kumar, Alexander Levine, and Soheil Feizi. Policy smoothing for provably robust reinforcement learning. *arXiv preprint arXiv :2106.11420*, 2021.

[103] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv :1611.01236*, 2016.

[104] Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified robustness to adversarial examples with differential privacy. In *2019 IEEE symposium on security and privacy (SP)*, pages 656–672. IEEE, 2019.

[105] Klas Leino and Matt Fredrikson. Stolen memories : Leveraging model memorization for calibrated {White-Box} membership inference. In *29th USENIX security symposium (USENIX Security 20)*, pages 1605–1622, 2020.

[106] Jiacheng Li, Ninghui Li, and Bruno Ribeiro. Membership inference attacks and defenses in supervised learning via generalization gap. *arXiv preprint arXiv :2002.12062*, 3(7), 2020.

[107] Jiacheng Li, Ninghui Li, and Bruno Ribeiro. Membership inference attacks and defenses in classification models. In *Proceedings of the Eleventh ACM Conference on Data and Application Security and Privacy*, pages 5–16, 2021.

[108] Zheng Li and Yang Zhang. Membership leakage in label-only exposures. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 880–895, 2021.

[109] Yen-Chen Lin, Zhang-Wei Hong, Yuan-Hong Liao, Meng-Li Shih, Ming-Yu Liu, and Min Sun. Tactics of adversarial attack on deep reinforcement learning agents. *arXiv preprint arXiv :1703.06748*, 2017.

[110] Fang Liu and Ness Shroff. Data poisoning attacks on stochastic bandits. In *International Conference on Machine Learning*, pages 4042–4050. PMLR, 2019.

[111] Kin Sum Liu, Chaowei Xiao, Bo Li, and Jie Gao. Performing co-membership attacks against deep generative models. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 459–467. IEEE, 2019.

[112] Thodoris Lykouris, Max Simchowitz, Alex Slivkins, and Wen Sun. Corruption-robust exploration in episodic reinforcement learning. In *Conference on Learning Theory*, pages 3242–3245. PMLR, 2021.

[113] Yuzhe Ma, Xuezhou Zhang, Wen Sun, and Jerry Zhu. Policy poisoning in batch reinforcement learning and control. *Advances in Neural Information Processing Systems*, 32, 2019.

[114] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv :1706.06083*, 2017.

[115] Amit Sharma Divyat Mahajan, Shruti Tople, and Amit Sharma. Does learning stable features provide privacy benefits for machine learning models. In *NeurIPS PPML Workshop*, 2020.

[116] Mohammad Malekzadeh, Anastasia Borovykh, and Deniz Gündüz. Honest-but-curious nets : Sensitive attributes of private inputs can be secretly coded into the classifiers' outputs. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 825–844, 2021.

[117] Patrick McDaniel, Nicolas Papernot, and Z Berkay Celik. Machine learning in adversarial settings. *IEEE Security & Privacy*, 14(3) :68–72, 2016.

[118] Frank D McSherry. Privacy integrated queries : an extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pages 19–30, 2009.

[119] Shagufta Mehnaz, Sayanton V Dibbo, Roberta De Viti, Ehsanul Kabir, Björn B Brandenburg, Stefan Mangard, Ninghui Li, Elisa Bertino, Michael Backes, Emiliano De Cristofaro, et al. Are your sensitive attributes private ? novel model inversion attribute inference attacks on classification models. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 4579–4596, 2022.

[120] Shike Mei and Xiaojin Zhu. Using machine teaching to identify optimal training-set attacks on machine learners. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[121] Tony A Meyer and Brendon Whateley. Spambayes : Effective open-source, bayesian based, email classification system. In *CEAS*. Citeseer, 2004.

[122] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

[123] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv :1312.5602*, 2013.

[124] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540) :529–533, 2015.

[125] Luis Muñoz-González, Battista Biggio, Ambra Demontis, Andrea Paudice, Vasin Wongrassamee, Emil C Lupu, and Fabio Roli. Towards poisoning of deep learning algorithms with back-gradient optimization. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 27–38, 2017.

[126] Luis Muñoz-González and Emil C Lupu. The security of machine learning systems. In *AI in Cybersecurity*, pages 47–79. Springer, 2019.

[127] Luis Muñoz-González, Bjarne Pfitzner, Matteo Russo, Javier Carnerero-Cano, and Emil C Lupu. Poisoning attacks with generative adversarial nets. *arXiv preprint arXiv :1906.07773*, 2019.

[128] Sasi Kumar Murakonda and Reza Shokri. Ml privacy meter : Aiding regulatory compliance by quantifying the privacy risks of machine learning. *arXiv preprint arXiv :2007.09339*, 2020.

[129] Milad Nasr, Reza Shokri, and Amir Houmansadr. Machine learning with membership privacy using adversarial regularization. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pages 634–646, 2018.

[130] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning : Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE symposium on security and privacy (SP)*, pages 739–753. IEEE, 2019.

[131] Blaine Nelson, Marco Barreno, Fuching Jack Chi, Anthony D Joseph, Benjamin IP Rubinstein, Udam Saini, Charles Sutton, J Doug Tygar, and Kai Xia. Exploiting machine learning to subvert your spam filter. *LEET*, 8(1) :9, 2008.

[132] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 75–84, 2007.

[133] P Russel Norvig and S Artificial Intelligence. A modern approach. *Prentice Hall Upper Saddle River, NJ, USA : Rani, M., Nayak, R., & Vyas, OP (2015). An ontology-based adaptive personalized e-learning system, assisted by software agents on cloud storage. Knowledge-Based Systems*, 90 :33–48, 2002.

[134] Nicolas Papernot, Martín Abadi, Ulfar Erlingsson, Ian Goodfellow, and Kunal Talwar. Semi-supervised knowledge transfer for deep learning from private training data. *arXiv preprint arXiv :1610.05755*, 2016.

[135] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning : from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv :1605.07277*, 2016.

[136] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519, 2017.

[137] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519, 2017.

[138] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael Wellman. Towards the science of security and privacy in machine learning. *arXiv preprint arXiv :1611.03814*, 2016.

[139] Catherine Parke. Réduire les risques d'accident d'avion grâce à l'intelligence artificielle. https ://www.cscience.ca/2022/11/18/reduire-les-risques-daccidents-aeriens-grace-a-lintelligence-artificielle/, Mis en ligne le 18 novembre 2022, consulté le 23 novembre 2022.

[140] Andrea Paudice, Luis Muñoz-González, Andras Gyorgy, and Emil C Lupu. Detection of adversarial training examples in poisoning attacks through anomaly detection. *arXiv preprint arXiv :1802.03041*, 2018.

[141] Neehar Peri, Neal Gupta, W Ronny Huang, Liam Fowl, Chen Zhu, Soheil Feizi, Tom Goldstein, and John P Dickerson. Deep k-nn defense against clean-label data poisoning attacks. In *European Conference on Computer Vision*, pages 55–70. Springer, 2020.

[142] Robi Polikar. Ensemble based systems in decision making. *IEEE Circuits and systems magazine*, 6(3) :21–45, 2006.

[143] Martin L Puterman. *Markov decision processes : discrete stochastic dynamic programming.* John Wiley & Sons, 2014.

[144] Apostolos Pyrgelis, Carmela Troncoso, and Emiliano De Cristofaro. Measuring membership privacy on aggregate location time-series. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 4(2) :1–28, 2020.

[145] JR Qunilan. Induction of decision trees. *Machine learning*, 1 :81–106, 1986.

[146] Md Atiqur Rahman, Tanzila Rahman, Robert Laganière, Noman Mohammed, and Yang Wang. Membership inference attack against differentially private deep learning model. *Trans. Data Priv.*, 11(1) :61–79, 2018.

[147] Amin Rakhsha, Goran Radanovic, Rati Devidze, Xiaojin Zhu, and Adish Singla. Policy teaching via environment poisoning : Training-time adversarial attacks against reinforcement learning. In *International Conference on Machine Learning*, pages 7974–7984. PMLR, 2020.

[148] Ronald L Rivest, Len Adleman, Michael L Dertouzos, et al. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11) :169–180, 1978.

[149] Christian Robert. Machine learning, a probabilistic perspective, 2014.

[150] Christian P Robert, George Casella, and George Casella. *Monte Carlo statistical methods*, volume 2. Springer, 1999.

[151] Yuji Roh, Kangwook Lee, Steven Whang, and Changho Suh. Fr-train : A mutual information-based approach to fair and robust training. In *International Conference on Machine Learning*, pages 8147–8157. PMLR, 2020.

[152] Elan Rosenfeld, Ezra Winston, Pradeep Ravikumar, and Zico Kolter. Certified robustness to label-flipping attacks via randomized smoothing. In *International Conference on Machine Learning*, pages 8230–8241. PMLR, 2020.

[153] Aniruddha Saha, Akshayvarun Subramanya, and Hamed Pirsiavash. Hidden trigger backdoor attacks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 11957–11965, 2020.

[154] Salman Salamatian, Amy Zhang, Flavio du Pin Calmon, Sandilya Bhamidipati, Nadia Fawaz, Branislav Kveton, Pedro Oliveira, and Nina Taft. Managing your private and public data : Bringing down inference attacks against your privacy. *IEEE Journal of Selected Topics in Signal Processing*, 9(7) :1240–1255, 2015.

[155] Ahmed Salem, Yang Zhang, Mathias Humbert, Pascal Berrang, Mario Fritz, and Michael Backes. Ml-leaks : Model and data independent membership inference attacks and defenses on machine learning models. *arXiv preprint arXiv :1806.01246*, 2018.

[156] Ahmed Salem, Yang Zhang, Mathias Humbert, Pascal Berrang, Mario Fritz, and Michael Backes. Ml-leaks : Model and data independent membership inference attacks and defenses on machine learning models. *arXiv preprint arXiv :1806.01246*, 2018.

[157] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[158] Roei Schuster, Congzheng Song, Eran Tromer, and Vitaly Shmatikov. You autocomplete me : Poisoning vulnerabilities in neural code completion. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 1559–1575, 2021.

[159] Rustam Shadiev, Ting-Ting Wu, Ai Sun, and Yueh-Min Huang. Applications of speech-to-text recognition and computer-aided translation for facilitating cross-cultural learning through a learning activity : issues and their solutions. *Educational Technology Research and Development*, 66(1) :191–214, 2018.

[160] Ali Shafahi, W Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison frogs ! targeted clean-label poisoning attacks on neural networks. *Advances in neural information processing systems*, 31, 2018.

[161] Virat Shejwalkar and Amir Houmansadr. Membership privacy for machine learning models through knowledge transfer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 9549–9557, 2021.

[162] Virat Shejwalkar and Amir Houmansadr. Membership privacy for machine learning models through knowledge transfer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 9549–9557, 2021.

[163] Christian Robert Shelton. Importance sampling for reinforcement learning with multiple objectives. 2001.

[164] Reza Shokri. Privacy games : Optimal user-centric data obfuscation. *arXiv preprint arXiv :1402.3426*, 2014.

[165] Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1310–1321, 2015.

[166] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE, 2017.

[167] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE, 2017.

[168] Reza Shokri, George Theodorakopoulos, and Carmela Troncoso. Privacy games along location traces : A game-theoretic framework for optimizing location privacy. *ACM Transactions on Privacy and Security (TOPS)*, 19(4) :1–31, 2016.

[169] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419) :1140–1144, 2018.

[170] C. Song, R. Ristenpart, and V. Shmatikov. Machine learning models that remember too much. In *Proceedings of the 2017 ACM SIGSAC Conference on computer and communications security*, pages 587–601, 2017.

[171] Congzheng Song and Vitaly Shmatikov. Overlearning reveals sensitive attributes. *arXiv preprint arXiv :1905.11742*, 2019.

[172] Shuang Song, Kamalika Chaudhuri, and Anand D Sarwate. Stochastic gradient descent with differentially private updates. In *2013 IEEE global conference on signal and information processing*, pages 245–248. IEEE, 2013.

[173] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout : a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1) :1929–1958, 2014.

[174] Kenneth O Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2) :99–127, 2002.

[175] Luke Stark and Zenon W. Pylyshyn. Intelligence artificielle (ia) au canada. https ://www.thecanadianencyclopedia.ca/fr/article/intelligence-artificielle, Mis en ligne le 14 décembre 2020, consulté le 23 novembre 2022.

[176] Jacob Steinhardt, Pang Wei W Koh, and Percy S Liang. Certified defenses for data poisoning attacks. *Advances in neural information processing systems*, 30, 2017.

[177] Yanchao Sun, Da Huo, and Furong Huang. Vulnerability-aware poisoning mechanism for online rl with unknown dynamics. *arXiv preprint arXiv :2009.00774*, 2020.

[178] Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1) :9–44, 1988.

[179] Richard S Sutton and Andrew G Barto. *Reinforcement learning : An introduction*. MIT press, 2018.

[180] Richard S Sutton, Andrew G Barto, and Ronald J Williams. Reinforcement learning is direct adaptive optimal control. *IEEE control systems magazine*, 12(2) :19–22, 1992.

[181] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

[182] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv :1312.6199*, 2013.

[183] E. Tabassi, K. J. Burns, M. Hadjimichael, A. D. Molina-Markham, and J. T. Sexton. A taxonomy and terminology of adversarial machine learning. *NIST IR*, pages 1–29, 2019.

[184] Boumedyen TAIBI and Khadidja LAMRI. Startups d'intelligence artificielle : une tendance mondiale. 2021.

[185] William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3-4) :285–294, 1933.

[186] F. Tramèr, F. Zhang, A. Juels, Reiter M. K, and T. Ristenpart. Stealing machine learning models via prediction {APIs}. In *25th USENIX security symposium (USENIX Security 16)*, pages 601–618, 2016.

[187] Edgar Tretschk, Seong Joon Oh, and Mario Fritz. Sequential attacks on agents for long-term adversarial goals. *arXiv preprint arXiv :1805.12487*, 2018.

[188] Félicien Vallet. Petite taxonomie des attaques des systèmes d'ia. https ://linc.cnil.fr/fr/petite-taxonomie-des-attaques-des-systemes-dia, Mis en ligne le 04 avril 2022, consulté le 05 janvier 2022.

[189] Vladimir Vapnik. Principles of risk minimization for learning theory. *Advances in neural information processing systems*, 4, 1991.

[190] Di Wang, Minwei Ye, and Jinhui Xu. Differentially private empirical risk minimization revisited : Faster and more general. *Advances in Neural Information Processing Systems*, 30, 2017.

[191] Jingkang Wang, Yang Liu, and Bo Li. Reinforcement learning with perturbed rewards. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 6202–6209, 2020.

[192] Xianmin Wang, Jing Li, Xiaohui Kuang, Yu-an Tan, and Jin Li. The security of machine learning in an adversarial setting : A survey. *Journal of Parallel and Distributed Computing*, 130 :12–23, 2019.

[193] Zhibo Wang, Jingjing Ma, Xue Wang, Jiahui Hu, Zhan Qin, and Kui Ren. Threats to training : A survey of poisoning attacks and defenses on machine learning systems. *ACM Computing Surveys*, 55(7) :1–36, 2022.

[194] Christopher John Cornish Hellaby Watkins. Learning from delayed rewards. 1989.

[195] Chen-Yu Wei, Christoph Dann, and Julian Zimmert. A model selection approach for corruption robust reinforcement learning. In *International Conference on Algorithmic Learning Theory*, pages 1043–1096. PMLR, 2022.

[196] Udi Weinsberg, Smriti Bhagat, Stratis Ioannidis, and Nina Taft. Blurme : Inferring and obfuscating user gender based on ratings. In *Proceedings of the sixth ACM conference on Recommender systems*, pages 195–202, 2012.

[197] Fan Wu, Linyi Li, Chejian Xu, Huan Zhang, Bhavya Kailkhura, Krishnaram Kenthapadi, Ding Zhao, and Bo Li. Copa : Certifying robust policies for offline reinforcement learning against poisoning attacks. *arXiv preprint arXiv :2203.08398*, 2022.

[198] Young Wu, Jeremy McMahan, Xiaojin Zhu, and Qiaomin Xie. Reward poisoning attacks on offline multi-agent reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 10426–10434, 2023.

[199] Han Xiao, Huang Xiao, and Claudia Eckert. Adversarial label flips attack on support vector machines. In *ECAI 2012*, pages 870–875. IOS Press, 2012.

[200] Tianbing Xu, Qiang Liu, Liang Zhao, and Jian Peng. Learning to explore via meta-policy gradient. In *International Conference on Machine Learning*, pages 5463–5472. PMLR, 2018.

[201] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing : Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv :1704.01155*, 2017.

[202] Yinglun Xu, Bhuvesh Kumar, and Jacob D Abernethy. Observation-free attacks on stochastic bandits. *Advances in Neural Information Processing Systems*, 34 :22550–22561, 2021.

[203] Yinglun Xu and Gagandeep Singh. Black-box targeted reward poisoning attack against online deep reinforcement learning. *arXiv preprint arXiv :2305.10681*, 2023.

[204] Yinglun Xu, Qi Zeng, and Gagandeep Singh. Efficient reward poisoning attacks on online deep reinforcement learning. *arXiv preprint arXiv :2205.14842*, 2022.

[205] Xin Yan and Xiao Gang Su. Linear regression analysis. *Theory and Computing*, 2003.

[206] Chaofei Yang, Qing Wu, Hai Li, and Yiran Chen. Generative poisoning attack method against neural networks. *arXiv preprint arXiv :1703.01340*, 2017.

[207] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy risk in machine learning : Analyzing the connection to overfitting. In *2018 IEEE 31st computer security foundations symposium (CSF)*, pages 268–282. IEEE, 2018.

[208] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy risk in machine learning : Analyzing the connection to overfitting. In *2018 IEEE 31st computer security foundations symposium (CSF)*, pages 268–282. IEEE, 2018.

[209] Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. Adversarial examples : Attacks and defenses for deep learning. *IEEE transactions on neural networks and learning systems*, 30(9) :2805–2824, 2019.

[210] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3) :107–115, 2021.

[211] Huan Zhang, Hongge Chen, Duane Boning, and Cho-Jui Hsieh. Robust reinforcement learning on state observations with learned optimal adversary. *arXiv preprint arXiv :2101.08452*, 2021.

[212] Marvin Zhang, Zoe McCarthy, Chelsea Finn, Sergey Levine, and Pieter Abbeel. Learning deep neural network policies with continuous memory states. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 520–527. IEEE, 2016.

[213] Xuezhou Zhang, Yuzhe Ma, Adish Singla, and Xiaojin Zhu. Adaptive reward-poisoning attacks against reinforcement learning. In *International Conference on Machine Learning*, pages 11225–11234. PMLR, 2020.

[214] Zhao Zhang, Yong Zhang, Da Guo, Lei Yao, and Zhao Li. Secfednids : Robust defense for poisoning attack against federated learning-based network intrusion detection system. *Future Generation Computer Systems*, 134 :154–169, 2022.

[215] Mengchen Zhao, Bo An, Wei Gao, and Teng Zhang. Efficient label contamination attacks against black-box learning models. In *IJCAI*, pages 3945–3951, 2017.

[216] Ping Zhao, Hongbo Jiang, Chen Wang, Haojun Huang, Gaoyang Liu, and Yang Yang. On the performance of k-anonymity against inference attacks with background information. *IEEE Internet of Things Journal*, 6(1) :808–819, 2018.

[217] Zhihao Zheng and Pengyu Hong. Robust detection of adversarial attacks by modeling the intrinsic properties of deep neural networks. *Advances in Neural Information Processing Systems*, 31, 2018.

[218] Chen Zhu, W Ronny Huang, Hengduo Li, Gavin Taylor, Christoph Studer, and Tom Goldstein. Transferable clean-label poisoning attacks on deep neural nets. In *International Conference on Machine Learning*, pages 7614–7623. PMLR, 2019.