L'intelligence artificielle : les enjeux de la robustesse

Maxime Darrin

L'intelligence artificielle: les enjeux de la robustesse

Maxime Darrin

Dirigé par Marco Panza

Université Paris 1 Panthéon-Sorbonne – UFR 10 Philosophie
Parcours Logique Philosophie des Sciences (LoPhiSc)

17 septembre 2021

Résumé

Les algorithmes appris issus d'algorithmes d'apprentissage sont de plus en plus utilisés en pratiques et on tend à leur accorder une confiance de plus en plus importante pour prendre des décisions dans des situations potentiellement critiques. Les algorithmes de conduites automatiques, de reconnaissance faciale ou encore de médecine issus de l'apprentissage automatique portaient de nombreux espoirs mais semblent en pratique bien moins fiables et efficace qu'attendu. Et tandis qu'une régulation standardisée tarde à être développée, il n'existe pas de méthode fiable pour évaluer leur robustesse, c'est-à-dire le maintien de leur performance face aux aléas de la vie. La régulation est alors de fait laissée aux acteurs et industriels du secteur qui se contente d'établir des guides de bonnes pratiques de conception sensés limiter les risques. Nous proposons dans ce mémoire un état des lieux de la robustesse et des problèmes pratiques que posent déjà les algorithmes d'appris ainsi que de la régulation de ceux-ci. Nous mettons en lumière différentes sources de leur manque de fiabilité et nous tentons de proposer des définitions formelles de notions de robustesse. Finalement, nous proposons un contournement du problème en définissant une classe de problèmes pour lesquels le manque de robustesse est pallié par la possibilité de vérifier la correction d'une décision prise. Nous partons des classes de complexité $\mathcal{NP}$ et IP, nous montrons qu'elles offrent un cadre permettant de rendre fiable l'emploi d'algorithmes appris et nous en proposons une relaxation humaine socio-technique comme fondement d'un emploi sûr des algorithmes appris.

Mots clefs — Apprentissage automatique, Robustesse, Fiabilité, Régulation, Apprentissage Statistique, Intelligence artificielle, Reconnaissance faciale, Voitures autonomes.

Table des matières

Table des figures

Chapitre 1

Introduction

Les méthodes d'apprentissage – le plus souvent rassemblées sous l'appellation *intelligence artificielle* – fleurissent de toutes part [46, 1]. Cette prolifération est due aux résultats impressionnants obtenus durant les années 2010, notamment par les méthodes dites d'apprentissage profond fondées sur les réseaux de neurones [50, 35, 47]. En effet, si pendant longtemps la mécanisation, c'est-à-dire le remplacement de la main d'œuvre humaine par des machines, s'est cantonnée aux tâches répétitives à faible *intelligence ajoutée*, l'intelligence artificielle permet aujourd'hui l'automatisation de tâches intellectuellement complexes. Cette réussite a été induite par une complexité accrue des algorithmes mis en œuvre, qui ont perdu dans le même temps en intelligibilité et en accessibilité. Comme on le verra, on ne parle pas ici seulement d'une perte d'intelligibilité due à une complexité grandissante mais à une perte intrinsèque aux méthodes utilisées.

Lorsque l'on parle d'algorithmes d'apprentissage, c'est-à-dire généralement de méthodes capables d'induire des règles à partir de données, on omet le plus souvent la distinction entre l'algorithme d'apprentissage – en général l'algorithme d'optimisation – et le modèle appris – résultat de l'optimisation. Si l'algorithme d'apprentissage est totalement maîtrisé et connu, le modèle appris est lui le plus souvent incompris et insaisissable. On ne sait pas vers quoi a convergé l'algorithme d'apprentissage et on ne comprend pas ce que le modèle appris fait en réalité.

En pratique, ce sont ces modèles appris, c'est-à-dire entraînés, qui sont distribués et utilisés, sans que l'on ait d'informations sur la façon dont ils ont été construits, sans informations sur le jeu de données ou la méthode d'apprentissage utilisés. Il arrive même

que les hyperparamètres [1] d'apprentissage aient été perdus ou oubliés, tant la seule chose qui compte est le modèle final. Il est nécessaire d'établir des outils pour évaluer et comprendre ces modèles appris en tant qu'eux-mêmes, de façon indépendante de leur entraînement. Il s'agit de décortiquer ces boîtes noires [**?**].

D'un point de vue politique et d'encadrement de leur emploi dans la société, il est assez clair que l'étude de ces modèles va devenir indispensable. Car si l'emploi de l'intelligence artificielle pour des tâches triviales, comme la recommandation de musique ou de publicité ne pose guère de problèmes, il n'en sera pas de même pour des systèmes critiques. Or, la construction d'un cadre juridique et la gouvernance des outils techniques, nécessitent des méthodes d'évaluation uniformisées et standardisées pour estimer leurs risques ou leur niveau de robustesse. En effet, les seuils de risque seront certainement différents entre un modèle utilisé pour suggérer des films et un modèle utilisé pour suggérer des traitements médicaux : le législateur doit pouvoir s'emparer de la question du risque de ces modèles.

D'abord, nous nous attacherons à décrire trois cas d'emploi de méthodes d'apprentissage pour des tâches critiques pour lesquelles les résultats en pratique sont inquiétants : la reconnaissance faciale (Section 2.2.1), la conduite autonome (Section 2.2.2) et la médecine assistée par intelligence artificielle (Section 2.2.3). Nous verrons en particulier que si les promesses de l'intelligence artificielle pour ces domaines étaient grandes, elles semblent en pratique plus dangereuses qu'utiles dans leur état actuel. Fort de ce constat et des controverses qui en sont issues, nous étudierons l'état actuel de la gouvernance (Section 2.3.2) avant de plaider pour la construction d'une notion de *robustesse* des algorithmes d'apprentissage. Une notion qui soit à la fois satisfaisante politiquement et utilisable en pratique (Section 2.4) afin de pouvoir stabiliser et standardiser l'évaluation *a posteriori* des modèles appris, issus de l'apprentissage.

Nous traiterons alors le double problème de philosophie politique ou philosophie du droit et scientifique théorique : qu'est-ce que la robustesse d'un modèle appris ? voire, qu'est-ce qu'une bonne notion de robustesse ? La notion de robustesse, ici encore virtuelle, floue et indéfinie doit émaner de de la technique scientifique et être politiquement et socialement acceptable et pratique, c'est-à-dire utilisable par le législateur pour construire un cadre juridique raisonnable. D'un point de vue technique, la définition d'une notion de robustesse est loin d'être triviale. Une situation représentative de ce problème est celle des attaques adversaires (*adversarial attacks* en anglais) [**?**, **?**] qui permettent de berner

1. Paramètres fixés à la main qui ne sont pas entraînés lors de l'apprentissage par descente de gradient.

même un modèle dont la précision évaluée atteint les 99%. Dans ce cas, on évalue la précision du modèle sur des tests mais cette évaluation ne donne finalement pas d'informations sur la robustesse du modèle, ici comprise comme sa capacité à donner un même résultat sur deux entrées pourtant quasiment identiques. Ainsi, la précision n'est pas gage de robustesse. Et comme nous le verrons, ce n'est pas le seul problème ou enjeu de la construction d'une *bonne* notion de robustesse mathématique pour les modèles appris. Quant au point de vue socio-politique, il faut que les notions de robustesses proposées par les mathématiques soient praticables. D'abord, il faut qu'elles soient calculables, évaluables en temps raisonnable en pratique et pas seulement un objet théorique pure ; ensuite il faut qu'elles traduisent des notions communes de robustesse. Que dire d'utiliser la constante de Lipschitz d'un modèle pour en évaluer la robustesse ? Alors, comme on le verra plus tard, c'est une idée raisonnable et intéressante, mais nous conviendrons aisément qu'elle est difficilement utilisable ou accessible pour le législateur, il est même difficile de bien comprendre ce qu'elle traduit réellement.

Dans ce travail nous nous attacherons à souligner l'intérêt grandissant qu'aurait une bonne notion de robustesse pour permettre une standardisation de l'évaluation des algorithmes appris (Chapitre 2). Une fois convaincu de l'importance et de l'intérêt de la question nous construirons et détaillerons le cadre théorique des algorithmes d'apprentissage pour adonner une idée précise de ce qu'ils sont réellement (Chapitre 3). Nous pourrons ensuite ébaucher des éléments de réponses concernant les sources techniques et philosophiques du manque de robustesse de ces méthodes (Chapitre 4). Puis nous proposerons des méthodes et définitions formelles d'évaluation de la robustesse d'algorithmes appris (Chapitre 5) ainsi qu'un cadre théorique issu de la théorie de la complexité algorithmique permettant de contrecarrer les risques intrinsèques aux algorithmes appris (Chapitre 6).

Chapitre 2

Une question de robustesse

La question de la robustesse se pose déjà en pratique, au point de ralentir significativement le déploiement de méthodes d'intelligence artificielle. En effet, dans de nombreux domaines, le gouffre entre l'usine et l'usage s'est révélé bien plus grand en pratique qu'en théorie, soulignant les problèmes de robustesse de ces technologies. Nous allons commencer notre exploration par un tour d'horizon des cas où la robustesse fait défaut, avant d'étudier la gouvernance de ces domaines. En effet, le manque de robustesse conduit à s'interroger sur les risques, et donc à s'intéresser aux méthodes de régulation de ces technologies dans la sphère publique. Nous verrons en particulier, au travers des exemples de la reconnaissance faciale (Section 2.2.1), de la conduite autonome (Section 2.2.2) ou encore de la médecine assistée par intelligence artificielle (Section 2.2.3), que les échecs cuisants de l'apprentissage automatique dans ces domaines critiques ont conduit les autorités et l'opinion publique à s'intéresser à la robustesse.

2.1 Robustesse et apprentissage automatique

Il n'existe pas de définition formelle consensuelle de la robustesse pour évaluer les algorithmes appris, et comme nous le verrons par la suite une telle définition serait certainement nécessaire pour permettre l'encadrement et l'emploi pour des tâches critiques (Section 2.2). Nous allons donc commencer par tenter de donner une définition informelle de la robustesse ainsi que des caractéristiques que l'on attendrait d'une bonne notion de robustesse. Commençons par donner une définition de *l'intelligence artificielle* dont on

souhaite évaluer la robustesse.

Définition de *l'intelligence artificielle* Bien que le terme *intelligence artificielle* soit le plus répandu, il serait plus juste de parler d'algorithmes d'apprentissage et d'algorithmes appris, et c'est à ces derniers que nous nous intéressons. Dans tous les exemples décrits précédemment, on parle de modèles inférés – algorithmes appris – à partir de données en utilisant des algorithmes d'optimisation – algorithmes d'apprentissage. Nous nous intéressons ici aux algorithmes appris, c'est-à-dire le produit fini issu de l'apprentissage, et nous soutenons que c'est pour ces algorithmes qu'il s'agit de réguler.

En fait, nous allons voir que la gouvernance de l'intelligence artificielle porte plus sur les méthodes d'entraînement des modèles plutôt que sur les résultats obtenus – les modèles appris (Section 2.3.2) – car nous manquons d'outils pour les évaluer ces algorithmes. En effet, il est plus aisé d'établir des bonnes pratiques de conceptions pour mitiger – au moins en apparence – les risques.

Une technologie immature Les méthodes d'apprentissage automatique sont un cas où la pratique est très en avance sur la théorie [7]. Si on peut facilement voir que ces méthodes semblent fonctionner, voire produire des résultats impressionnants, il est important d'être conscient qu'en l'état actuel, il n'existe presque pas de supports théoriques permettant d'expliquer leur réussite ou de comprendre leur fonctionnement.

En particulier, il est très difficile de donner des garanties de fonctionnement sur les algorithmes appris. On peut seulement les tester sur des jeux de données et voir comment ils s'en sortent. Néanmoins, le plus souvent ces jeux de données de test souffrent des mêmes biais que les jeux utilisés pour l'entraînement. Ainsi, même si les algorithmes appris semblent bien se comporter sur les jeux de test, ceux-ci ne permettent pas de découvrir les écueils les plus importants.

De façon générale, il n'existe pas de méthode théorique bien définie permettant de certifier le fonctionnement d'un algorithme appris ou de détecter aisément ses biais, bien que de plus en plus de travaux se concentrent sur ces questions [13]. Il n'existe pas non plus de notion de *robustesse* théorique bien établie qui pourrait être utilisée pour la régulation de ces techniques. Cette absence de fondements théoriques permettant d'évaluer le produit fini de l'intelligence artificielle conduit à une gouvernance des bonnes pratiques pour mitiger les risques plutôt qu'à une évaluation des modèles appris, ce qui constitue

la principale limite pour leur emploi dans des cas critiques.

La robustesse Ainsi, de façon informelle, la robustesse d'un algorithme appris correspond à la stabilité de la performance de ce modèle face aux aléas des données qui lui sont présentées. Mesurer la robustesse d'un modèle consiste ainsi à évaluer sa proposension à se comporter correctement de façon constante sur la distribution des données qui lui sera présentée en production. On peut aussi réduire la robustesse à évaluer la proposension du modèle à se comporter de la même façon sur les tests d'usines et en pratique. On veut savoir à quel point les garanties et résultats obtenus par les épreuves d'usines se transposent à la vie.

De façon plus générale, on cherche des notions de robustesse qui seraient calculables en pratique et qui traduiraient différentes idées de robustesse. On peut par exemple imaginer une notion qui étudierait la régularité des modèles appris garantissant qu'ils n'ont pas de pentes trop élevées, c'est-à-dire qu'une modification mineure de l'entrée ne modifierait pas significativement la décision du modèle. On peut aussi penser à des mesures et hypothèses fortes sur les jeux de données de test : on pourrait dans certains domaines imposer que le jeu de test soit constitué au plus proche de l'emploi réel. Par exemple pour des algorithmes de reconnaissance d'images médicales, pour des radiographies par exemple, on pourrait imaginer imposer que le jeu de test soit constitué d'images issues spécifiquement de la machine dont seront issues les images dans le futur et que seules ces données de test fournissent une évaluation acceptable pour certifier l'algorithme.

Nous ne développerons pas de réflexion plus poussée ou précise de notions de robustesse envisageables ici, d'abord parce que c'est un problème ouvert et non résolu, et ensuite parce que les quelques options ou approches envisageables nécessiteraient une formalisation mathématique beaucoup plus poussée. Nous verrons cependant dans une dernière partie des cadres d'emplois dans lesquels on dispose de façons efficaces de vérifier des solutions obtenues par des algorithmes appris, mitigeant alors grandement les risques associés (Section 6.2).

2.2 Des résultats pratiques inquiétants

2.2.1 La reconnaissance faciale

Parmi les domaines critiques d'application de l'apprentissage automatique, la reconnaissance faciale est à la fois l'application dont les conséquences néfastes à court terme sont le moins visibles, mais aussi probablement l'une des plus dangereuses. Tout d'abord, nous verrons rapidement comment fonctionnent ces algorithmes et comment ils sont entraînés, pour ensuite pointer les controverses et régulations qui les concernent.

Fonctionnement

Les algorithmes de reconnaissance faciale modernes consistent en des modèles neuronaux entraînés à extraire les caractéristiques importantes des visages pour les différencier. On construit une fonction qui, à une image de visage, associe une représentation abstraite de celui-ci, censée contenir uniquement les caractéristiques utiles pour différencier un visage d'un autre. Pour entraîner une telle fonction, on constitue un jeu de données contenant plusieurs photos de visage par individu. On cherche alors à faire en sorte que pour des images de visage d'une même personne, la fonction produise des représentations abstraites proches. Intuitivement, on espère que cette procédure encourage la sélection des caractéristiques les plus discriminantes entre les visages des individus. Une fois que l'on dispose d'une telle fonction, pour décider si deux images de visages représentent la même personne, on étudie la distance entre leur représentation dans l'espace abstrait : si les représentations sont proches, on décide que c'est la même personne. On peut dès à présent entrevoir un premier problème : que se passe-t-il pour des visages atypiques vis-à-vis du jeu de données ? En effet, les traits caractéristiques des visages sont relativement différents selon les populations humaines. Il est tout à fait possible qu'un trait particulier soit utile pour différencier deux personnes caucasiennes mais que ce trait n'ait aucun intérêt pour différencier, disons, deux afro-américains ou deux asiatiques. Or, on sait que ces derniers sont sous-représentés dans les jeux de données utilisés pour entraîner ces algorithmes. C'est cela qui conduit aux biais et au manque de robustesse observés en pratique.

img/facial_aa.png

FIGURE 2.1 – Exemple d'habit (bonnet) modifié pour porter un *adversarial patch* pour se faire passer pour quelqu'un d'autre aux yeux d'un algorithme de reconnaissance faciale [**?**].

Controverses

L'exemple le plus iconique d'une telle défaillance est l'algorithme commercial *Rekognition* fourni, par *Amazon*, aux Ètats-Unis[44] aux forces de l'ordre. *Amazon* prétend que sa technologie est capable de suivre plus de cent visages dans une unique image et de suivre des personnes dans différents flux vidéos (de caméras de surveillance ou portées par des agents). *Amazon* promeut ces méthodes auprès des autorités qui ont commencé à les utiliser, sans encadrement législatif ou débat public.

L'Union américaine pour les libertés civiles (ACLU) a testé l'outil *Rekognition* en constituant une base de données de personnes recherchées à partir des données publiques des agences fédérales. Ensuite, l'association a utilisé cet outil pour comparer des photos de membres du congrès et du sénat des États-Unis avec celles présentes dans la base de données de personnes recherchées. Il est apparu que 28 parlementaires avaient été (faussement) identifiés comme des personnes recherchées, les erreurs concernant principalement des personnes de couleur.

De façon plus générale, il s'avère que les algorithmes de reconnaissance faciale peuvent être induits en erreur par des coupes de cheveux particulières, par des maquillages [6],ou encore par un post-it bien placé sur son visage [48]. Il est même possible de concevoir des déguisements (Figure 2.1) pour être reconnu comme une autre personne par l'algorithme de reconnaissance faciale [34, 3]. On peut tout à fait imaginer un masque (ceux contre le covid), avec des motifs bien choisis, qui nous ferait passer pour Emmanuel Macron aux yeux de la reconnaissance faciale. Plus simplement, il est aussi possible d'utiliser une telle méthode pour que ces algorithmes ne détectent même pas un visage sur l'image et donc passer sous les radars. Tous ces exemples reposent sur des méthodes d'*adversarial attack* [27, 53].

Il est alors facile d'imaginer les dérives que peuvent engendrer une technologie si peu robuste, laquelle a déjà fait faussement accuser un homme [25], et qui n'est pourtant que très peu encadrée[23].

En France, la question de la reconnaissance faciale se pose de façon tout aussi importante, quoique moins médiatisée. En effet, le fichier de traitement des antécédents judiciaires (TAJ) a été créé par un décret du 4 mai 2012 [22] et est destiné notamment à rassembler toute "photographie comportant des caractéristiques techniques permettant de recourir à un dispositif de reconnaissance faciale (photographie du visage de face)". La Quadrature du Net pointe que ce fichier est utilisé sans contrôle ou presque par les forces de l'ordre pour échanger des informations, que celui-ci n'est que rarement mis à jour, conduisant notamment à maintenir dans le fichier des personnes pourtant mises hors de causes dans les affaires qui les concernaient [32]. L'association note aussi, qu'en pratique, le fichier est rempli de diverses photographies issues de contrôle d'identité (photo de la pièce d'identité), de photos récupérées sur internet ou prises lors des gardes à vue, et donc finalement intégrées aux systèmes de reconnaissance faciale. Selon un rapport parlementaire concernant les fichiers mis à disposition de la police le TAJ contient les "photos de face" d'environ sept millions de français. La Quadrature du net a d'ailleurs rassemblé un certain nombre d'articles qui permettent de penser que la reconnaissance faciale à grande échelle est déjà déployée [1, 16, 20].

Régulations

Il n'existe pour l'instant que peu de tentatives de régulation de l'utilisation de la reconnaissance faciale, et les appels à construire un cadre juridique se font de plus en plus nombreux. Les quelques tentatives de régulation en place ne passent pour l'instant pas tant par les États que par les acteurs et industriels eux-mêmes. Aux États-Unis, les récentes controverses et appels contre l'utilisation de la reconnaissance faciale, pour des raisons de principe ou invoquant une technologie immature, ont conduit les grandes entreprises à y renoncer, au moins temporairement. En effet, *IBM*, *Amazon* et *Microsoft* ont renoncé à fournir à la police des outils de reconnaissance faciale et de surveillance [24]. Les associations de défense des droits ne sont pas les seules à s'élever contre ces technologies : de nombreux chercheurs et scientifiques [2, 41] ont cosigné un appel à ne plus collaborer avec la police et à arrêter de lui fournir des techniques d'intelligence artificielle. Ils pointent notamment les biais inhérents à ces méthodes et leur manque de robustesse.

2.2.2 La conduite autonome

Source de fantasmes et de nombreuses promesses, la conduite autonome a fait l'objet d'un grand nombre de recherches et commence à être exploitée commercialement, notamment par *Tesla*, mais pas seulement. En effet, la grande majorité des accidents de la route sont dus d'une manière ou d'une autre à une erreur humaine, généralement d'inattention. Ainsi une conduite automatique, à l'attention infaillible, devrait réduire significativement les accidents [1] [51].

Les premiers accidents impliquant des véhicules autonomes n'ont cependant pas tardé, relançant les débats techniques sur la robustesse et la maturité de la technologie, mais aussi les débats politiques concernant leur gouvernance et leur encadrement. On décompte pour l'instant au moins trois accidents mortels, dont au moins un d'eux résulte directement d'une erreur d'appréciation de l'algorithme de vision de la voiture [12]. On peut par ailleurs donner l'exemple assez courant des voitures *Tesla* qui confondent la lune orangée et un veut un tricolore orange [2] (Figure 2.3).

On différencie plusieurs niveaux d'automatisation de la conduite de 0 ("aucune assistance à la conduite") à 5 ("véhicule totalement autonome, ne nécessite aucune intervention humaine"). Pour l'instant, le niveau le plus élevé atteint et autorisé sur des routes publiques est le niveau 3 ("véhicule totalement autonome dans certaines conditions, mais le contrôle doit être repris en cas de problèmes") [42]. Il est important de préciser que ce sont des standards exprimés en langage naturel et qu'ils ne définissent pas des seuils ou des manières de décider si le véhicule est effectivement autonome : ce sont des recommandations d'usage.

Controverses

L'accident impliquant une voiture *Tesla*[12] et une semi-remorque est probablement le meilleur exemple du manque de robustesse des algorithmes de vision. D'ailleurs, *Le Parisien* titre "Crash mortel : Mais comment les capteurs de Tesla ont-ils pu rater un semi-remorque ?", une interrogation légitime qui résume à elle seule les problèmes que pose l'emploi d'algorithmes appris à des tâches critiques. En l'occurrence, il semblerait que l'algorithme ait confondu le camion avec un panneau publicitaire, le conduisant à

1. A tel point que certains s'inquiètent d'une pénurie de dons d'organe[28].
2. `https://twitter.com/jordanteslatech/status/1418413307862585344`

ne pas s'en inquiéter. Cette erreur n'est cependant pas surprenante, considérant que les algorithmes de reconnaissance d'objets peuvent être facilement induits en erreur par des changements *a priori* anodins dans le contexte. Par exemple, une position ou un angle inhabituel de l'objet dans l'image peut provoquer des erreurs dans les algorithmes [4].

Dans la Figure 2.2, on a sur la première colonne une image classique d'un objet et la classification (correcte, en vert) produite par l'algorithme de reconnaissance d'objet. Sur les autres colonnes on observer le même objet mais dans une position différente, peu courante, et on a la classification produite par l'algorithme (fausse, en rouge). Ainsi, on observe en particulier qu'un bus scolaire jaune, sur la première ligne, s'il est couché en travers de la route, est mal reconnu.

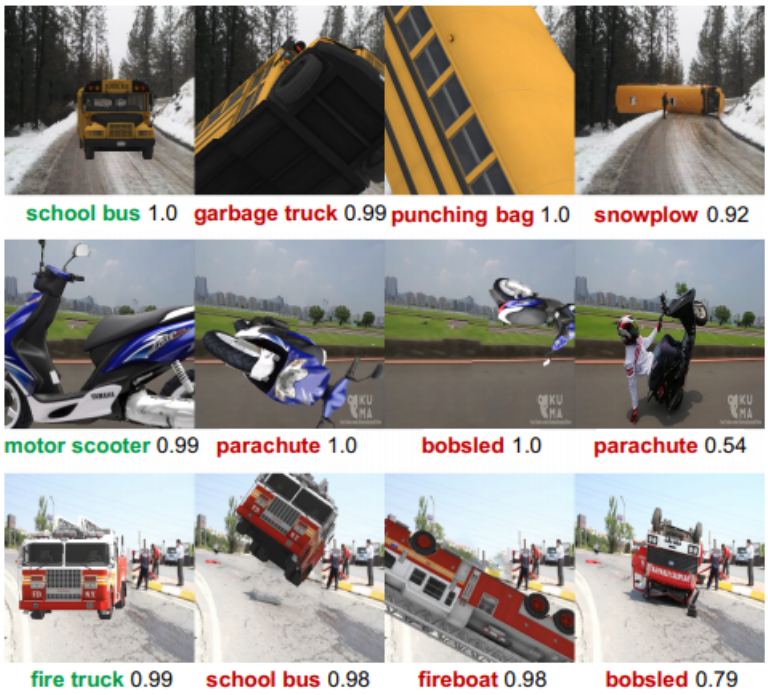

FIGURE 2.2 – Google Inception-v3 classifier [4, 49] classifie correctement les objets lorsqu'ils sont présentés dans une position classique, mais échoue lorsqu'ils sont présentés dans des positions inhabituelles : loin de leur distribution habituelle.

De la même façon que les algorithmes de reconnaissance faciale, les systèmes de vision des voitures autonomes souffrent de biais importants et sont encore plus difficiles à entraîner. En effet, là où pour faire de la reconnaissance faciale, on imagine aisément pouvoir récupérer des photos de visages, comment entraîner efficacement un algorithme à détecter tous les objets qu'il pourra croiser sur les routes, dans toutes les situations ?

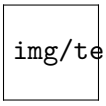

FIGURE 2.3 – Une voiture tesla confond la lune orangée avec un feu tricolore.

Il est assez clair que l'espace à explorer est largement plus important, rendant, de fait, l'entraînement encore plus ardu.

Régulation

Si la conduite autonome soulève de nombreux problèmes d'éthique [31] ou de responsabilités en cas d'accidents [10], ce n'est pas ici notre sujet : nous nous intéressons plutôt à la mesure de fonctionnement de ces systèmes. Les législations concernant la responsabilité évite le plus souvent cette question en considérant qu'un accident est toujours la responsabilité du conducteur [9]. Ce point de vue est une façon de contourner le niveau de fiabilité de la machine : c'est au conducteur de décider de confier la tâche de conduite à la machine s'il a confiance et il doit toujours surveiller de ce fait la voiture : il reste le responsable.

Concernant la mesure de "bonne" conduite, on utilise le plus souvent le nombre de morts par distance parcourue par les véhicules considérés. Aux États-Unis, pour les humains cette valeur est autour de $1,18$ morts par $160 \times 10^6 km$[30]. Par comparaison, les véhicules autonomes aux États-Unis ont parcouru moins de $1 \times 10^7 km$, donc trop peu pour estimer leur robustesse par rapport à la conduite humaine. Néanmoins, il semblerait que les technologies actuelles soient encore loin du niveau humain. Par exemple, les véhiculent autonomes de la flotte d'*Uber* nécessitent en moyenne une intervention tous les $20km$ pour éviter un accident. Cela dit, c'est une première direction intéressante pour concevoir la robustesse des véhicule.

Par rapport à la reconnaissance faciale, la construction du cadre légal d'emploi des voitures autonomes est un peu plus avancée, notamment pour prendre en compte les questions de responsabilité en cas d'accident, mais le cadre actuel ne traite pas de la façon évaluer les voitures autonomes. D'ailleurs, de façon similaire à la reconnaissance faciale, ce sont les acteurs industriels qui tendent à ralentir le déploiement de ces technologies lors d'accidents ou de problèmes [38], pour éviter des régulations défavorables. Nous verrons en effet, plus tard, Section 2.3.2, que les acteurs ont intérêt à éviter tout écart

(de conduite) pour s'assurer une régulation favorable ou préserver leur image.

2.2.3 Intelligence artificielle et médecine

La médecine est un autre domaine gourmand en intelligence artificielle et aux enjeux critiques. De nombreux projets d'application des méthodes d'apprentissage à la médecine ont émergé et sont menés par la plupart des grandes compagnies informatiques comme *Microsoft*, *IBM* ou encore *Google* au travers de sa filiale *DeepMind*, et ce dans de nombreux domaines de la médecine. D'abord, en employant des méthodes de reconnaissance d'image pour traiter des radiographies [26], mais aussi pour effectuer du diagnostic médical [43], étudier les interactions entre médicaments [15] ou prédire les chances de réussites d'une greffe [5, 40].

Évidemment, toutes ces applications soulèvent des questions éthiques considérables et nécessitent des méthodes d'intelligence artificielle dont la robustesse puisse être garantie. Comme pour les voitures autonomes ou la reconnaissance faciale, c'est ce second point qui nous intéresse dans ce mémoire. L'évaluation de la robustesse et la détection des biais sont absolument cruciales pour l'emploi de l'apprentissage automatique en médecine. De la même façon que les jeux de données déséquilibrés conduisent les algorithmes de reconnaissance faciale à confondre des personnes de couleur plus que des caucasiens (Section 2.2.1), ils peuvent conduire à diagnostiquer correctement les uns et se tromper totalement pour les autres. Ce problème est d'autant plus important en médecine que les origines, dites raciales, peuvent avoir des conséquences sur certaines maladies et sur la façon de les diagnostiquer.

Controverses

Le cas d'*IBM Watson* est particulièrement intéressant. En effet, dans la foulée de la démonstration de son efficacité pour jouer au jeu *Jeopardy* en 2011, *IBM* a annoncé qu'elle comptait adapter cette technologie pour le diagnostic médical. L'objectif annoncé était aussi clair qu'ambitieux : proposer un médecin artificiel qui serait capable de produire un diagnostic à partir de la description clinique du patient. Presque une décennie plus tard, cet objectif est toujours très loin d'être atteint et les résultats issus des travaux d'*IBM* sont pour le moins décevants. Tout d'abord, les ambitions d'*IBM* ont été revues à la baisse : au lieu de proposer un outil de diagnostic généraliste, l'entreprise

s'est concentrée sur différents outils pour différents types de maladie ou sous-domaines particuliers[46].

Cependant, même sur des tâches plus spécifiques et restreintes, les résultats obtenus sont en pratiques inutilisables. Par exemple, l'outil d'oncologie de *Watson*, qui est l'un des plus utilisés, donne au mieux 70% de diagnostics similaires à ce qu'auraient donné des médecins humains. Alors, on pourrait penser que ce n'est pas si mal, et que les 30% restant pourraient être des innovations bénéfiques par rapport aux décisions humaines ou alors des décisions pour des cas difficiles où les experts humains divergent eux-mêmes. Il n'en est rien [46]. L'algorithme rate régulièrement des diagnostics très simples, proposent des traitements absurdes ou confond un cas très grave avec une simple suspicion sans gravité.

Certains médecins pointent par ailleurs que dans l'état actuel même si *Watson* fonctionnait parfaitement, il ne serait pas utilisable car sans explications ou arguments étayant les décisions de la machine, on ne pourrait considérer ses suggestions comme "*evidence based*" [46].

Alors que s'est-il passé ? Pourquoi *IBM* se sentait-elle assez confiante pour annoncer un robot médecin aussi rapidement avant de déchanter aussi vite ?

Il semblerait qu'*IBM* se soit heurté à la réalité (et à la difficulté) de la vraie vie. En effet, en environnements contrôlés, sur des données de test, tout semblait parfaitement fonctionner. Il s'agit d'être capable d'évaluer *a priori* la robustesse d'un algorithme d'apprentissage. L'échec d'*IBM* ici montre à quel point cet exercice est difficile, mais d'autant plus essentiel que la tâche à accomplir est critique.

2.3 La gouvernance de l'IA : un processus à deux vitesses

Comme on a pu le voir dans les exemples introductifs (Sections 2.2.1, 2.2.2, 2.2.3), les acteurs tendent à l'autorégulation sous la pression de l'opinion publique et de la peur de voir apparaître des régulations plus strictes de la part des États. Néanmoins, à cause du manque d'outils théoriques pour évaluer les produits finis, cette autorégulation prend la forme de « bonnes pratiques »ou de guide de conception plutôt que le développement de standards d'évaluation ou de certification des modèles appris. L'implication des États est encore relativement faible, et même si des propositions de régulation existent [**?**], elles

sont pour le moins insuffisantes et ne proposent pas réellement de meilleures méthodes que l'instauration de « bonnes pratiques » à une plus large échelle.

2.3.1 Un domaine naissant

Tout d'abord, il est important de noter que si les résultats communiqués de l'intelligence artificielles semblent impressionnants, il n'en reste pas moins que ces méthodes sont encore très peu utilisées en pratique. Ainsi leur régulation n'est devenue réellement nécessaire que très récemment, notamment pour les emplois critiques. L'exemple d'*IBM Watson* en est la caricature : si l'annonce a fait grand bruit, dix ans plus tard, c'est encore un outil peu utilisé car peu fiable et finalement difficile à sécuriser (Section 2.2.3). En effet, la plupart des utilisations qui en sont faites relèvent de la recherche fondamentale [8, 52], du jouet démonstratif ou d'applications non critiques, voire triviales, telles que la recommandation de musiques ou de films. De plus, si l'on parle régulièrement d'intelligence artificielle dans l'industrie ou les startups, elle est pour autant assez peu utilisée : en Europe 40% des *startups* assurant *faire de l'IA* n'utilisent en réalité aucune méthode d'intelligence artificielle [17] mais plutôt des algorithmes plus classiques.

Dès lors, les questions de robustesse et de sécurité ont longtemps été laissées de côté. Ce ne sont pas des questions cruciales lorsque l'on joue à des jeux vidéos tels que *Starcraft*, ou à des jeux de plateaux tels que le Go ou les échecs. Quant aux algorithmes de recommandation, s'ils se trompent régulièrement, on conviendra que ce n'est pas un problème mortel ou réellement inquiétant. Cette faible utilisation de l'IA pour des cas non triviaux en pratique explique que le développement de standards d'évaluation des modèles appris n'ait pas été nécessaire.

Cependant, il existe un certain nombre de cas où *l'intelligence artificielle* au sens *d'apprentissage automatique* commence à être utilisée en pratique, ou est en passe de l'être, soulevant dans le même temps controverses et interrogations (Sections 2.2.1, 2.2.2, 2.2.3). La gouvernance par les bonnes pratiques ou guide de conception ne peut offrir des garanties suffisantes pour de telles applications, car si elle mitige les risques, elle ne permet pas de s'assurer de la robustesse *a priori* des modèles appris.

2.3.2 Une gouvernance par les acteurs

Autorégulation Le cas de la reconnaissance faciale est certainement le plus emblématique de l'autorégulation des industriels. La pression de l'opinion publique, les appels de différentes associations de consommateurs et d'usagers, ainsi que des auditions au congrès des États-Unis ont conduit les grandes firmes du numérique à renoncer à fournir leurs outils de reconnaissance faciale aux autorités. Néanmoins, il apparait que les entreprises qui ont fait ce choix sont des entreprises qui ont une image de marque à protéger et dont la reconnaissance faciale n'est pas une activité principale. En revanche, des entreprises telles que *ClearView*, dont c'est le cœur de métier et qui n'ont pas d'autres activités qui nécessitent de protéger leur image, continuent à distribuer leurs outils.

On peut dire que c'est en partie pour cela que les grands groupes appellent à une régulation publique de ces technologies. En effet, d'une certaine façon, les frasques de *Clearview* affectent l'image du secteur tout entier et plus généralement l'image du domaine que représente l'intelligence artificielle. *Facebook* tente par exemple d'empêcher cette entreprise d'utiliser les photos récupérées par l'entreprise sur le réseau social pour entraîner ses modèles de reconnaissance faciale [39]. C'est un cas assez atypique de l'appel à la force publique pour réguler un marché. Habituellement, ce sont les plus petites entreprises qui comptent sur ces régulations pour être protégées des grands groupes, ici c'est le contraire : une plus petite entreprise dont les agissements – non régulés – mettent en péril l'image d'un secteur.

Vers une gouvernance des « bonnes pratiques » De façon similaire à la gouvernance d'internet, l'intelligence artificielle s'oriente vers une « gouvernance des bonnes pratiques »avec la mise en place de comités d'éthique, de chartes de bonne utilisation et de conception des algorithmes d'intelligence artificielle. Les *GAFAM* ont lancé différentes initiatives en faveur de l'éthique de l'intelligence artificielle. En 2018, les principaux acteurs du secteur ont chacun publié d'une manière ou d'une autres des "principes éthiques de l'AI", que ce soit *Microsoft*, *IBM* ou encore *Google* [21, 29]. D'autres, comme *Facebook* se sont engagés dans le développement d'outils permettant de détecter les biais et d'assurer la *fairness* (l'équité) dans ses algorithmes.

qui semble relever de l'*ethical washing* L'absence de standardisation induite par cette absence d'outils et d'intervention étatique rend très difficile la généralisation et

l'application des bonnes pratiques à l'ensemble du secteur. Car si, comme on l'a vu précédemment, les grandes compagnies ont des intérêts d'image publique à protéger, ce n'est pas le cas pour les petites entreprises spécialisées. Il est d'ailleurs notable que même les grandes compagnies n'ont pas adopté entre elles les mêmes principes éthiques.

Il est difficile de dire si, en pratique, ces initiatives de régulation de l'intelligence artificielle ont de réels effets. Un exemple récent des relations difficiles entre la recherche d'éthique affichée et les *GAFAM* est celui de Timnit Gebru. Renvoyée en 2020 de chez *Google* pour des raisons encore peu claires, Timnit Gebru travaillait notamment sur les problèmes éthiques que posaient les très gros modèles d'apprentissage linguistiques. Il semblerait que ce soient ses travaux remettant en cause l'emploi de ces modèles qui aient conduit à son éviction [33, 11]. En effet, *Google* compte et investit énormément sur ces technologies pour son moteur de recherche.

Dans le même ordre d'idée, Karen Hao relate dans le *New-York Times* l'histoire de l'équipe d'éthique de l'intelligence artificielle de *Facebook* [37]. Elle pointe que cette équipe dont les conclusions ont longtemps été en contradiction avec les objectifs de croissance de l'entreprise était dans le même temps peu dotée et ses propositions ignorées. Ce n'est que récemment qu'elle a reçu de l'attention : lorsque le congrès des États-Unis a menacé de réguler les biais racistes dans les recommandations publicitaires. Elle a alors reçu comme unique mission de se concentrer sur les biais des algorithmes d'apprentissage.

Ces deux exemples sont représentatifs de l'état de la gouvernance du secteur de l'intelligence artificielle : si les initiatives de recherche d'éthique pour les algorithmes d'apprentissage fleurissent, leurs conclusions ou proposition semblent rarement implémentées.

2.3.3 Une gouvernance institutionnelle en construction mais sans révolution

Il n'est pas possible de parler d'encadrement des algorithmes d'intelligence artificielle sans discuter la proposition d'encadrement de l'Union Européenne [**?**]. Bien qu'elle représente un premier pas intéressant vers un emploi sûr de ces technologies, elle semble souffrir des mêmes manques d'outils théoriques pour appréhender la robustesse.

Cette proposition se concentre sur deux axes principaux : définir les niveaux de risques de différents cas d'emplois et définir quels types d'algorithmes relèvent de l'intelligence

artificielle. Pour chaque niveau de risques identifié, la proposition d'encadrement décrit les mesures à mettre en œuvre pour garantir une sécurité adéquate.

Il y a quatre niveaux de risques identifiés : risques inacceptables, risques élevés, risques moyens et risques faibles. La seule catégorie qui représente un réel intérêt ici est "risques élevés". En effet, les emplois présentant des risques inacceptables sont purement interdits tandis que seules quelques recommandations de précautions sont faites pour les risques moyens et faibles. En ce qui concerne les risques élevés, même si c'est la catégorie pour laquelle les recommandations les plus précises sont faites, elles restent vagues et générales. En effet, elles consistent principalement à s'assurer qu'une trace des exécutions est conservée pour pouvoir vérifier *a posteriori* ce qu'il s'est passé en cas de problèmes, et à garantir que l'opérateur s'engage à corriger les problèmes et biais identifiés, ainsi qu'à produire une évaluation des risques. Il parait intéressant de pointer ici que ces recommandations devraient s'appliquer à n'importe quel programme informatique de grande envergure produisant des décisions critiques. Il semble notamment que ces tentatives d'encadrement donnent aux biais appris un statut différents de celui des bugs logiciels issus d'erreurs humaines.

Il est assez clair que le texte se concentre sur les risques de biais et de boucles de rétroactions introduits par des boîtes noires incomprises et non questionnées, telles que décrites dans Weapon of maths destructions de Cathy O'Neil[**?**]. L'objectif n'est pas de produire des standards de sécurité pour les algorithmes d'apprentissage mais bien de proposer des « bonnes pratiques »pour opérer ces algorithmes.

2.4 Vers une étude de la robustesse

Une solution pour stabiliser et standardiser la gouvernance des algorithmes d'apprentissage serait de disposer d'outils formels offrant des garanties de fonctionnement des algorithmes employés. De la même façon que l'on certifie les autopilotes d'avion en utilisant la preuve de programme[45], on pourrait envisager certaines formes de certification des modèles d'apprentissage.

2.4.1 La robustesse comme propriété des modèles

Une notion à définir politiquement La définition de la notion de robustesse à utiliser pour tel ou tel algorithme est une affaire hautement politique et qui devrait être issue d'un travail commun des chercheurs en science sociales, en droit et en apprentissage automatique. En effet, il s'agit ici de convenir ensemble d'une bonne notion de "ce modèle fonctionne raisonnablement".

D'un point de vue méthodologique pour la certification des algorithmes d'apprentissage, il s'agirait d'abord de concevoir une notion de robustesse qui soit à la fois satisfaisante en termes de garanties pour le public, qui ait un sens politiquement, et qui soit formalisable et mesurable en pratique à partir des modèles d'intérêts. Dans cette optique, il est nécessaire de rassembler à la fois des spécialistes de l'apprentissage automatique pour concevoir formellement ces métriques, mais aussi des spécialistes des sciences sociales et des juristes pour décider du sens que doivent porter ces formalisations.

Ces travaux ont déjà commencé dans les domaines des algorithmes équitables (*fairness*) [36, 19] pour lesquels différentes formulations (utilisables en pratiques) de l'équité ont été proposées et peuvent aujourd'hui être utilisées pour évaluer les biais des modèles d'apprentissage. Il s'agirait de mener le même travail sur les questions de robustesse, car il n'existe pas de notion consensuelle de la robustesse.

Objectifs Dans la suite de ce mémoire, nous nous attacherons à donner une définition formelle des algorithmes d'apprentissages que constituent les réseaux de neurones et nous tenterons de proposer des formulations de notions de robustesses. En particulier, nous nous attacherons à construire des notions qui traduisent des attentes naturelles politiquement raisonnables et utilisables en pratiques.

Chapitre 3

Algorithmes d'apprentissage

Nous avons jusqu'ici principalement abordé le problème de *l'intelligence artificielle* de façon générale sans entrer dans des détails techniques et de définitions des objets véritablement considérés. En particulier, nous n'avons pas décortiqué ce qu'étaient réellement ces *algorithmes d'intelligence artificielle*, d'apprentissage ou de *machine learning*, et nous avons amalgamé méthodes d'apprentissages – utilisées pour extraire la connaissance – et l'algorithme résultant de cet apprentissage. Il est crucial de définir précisément les différentes étapes de la mise en œuvre de ces méthodes d'apprentissage pour pouvoir en étudier les affres et les propriétés.

Pour ce faire, nous établirons rapidement la forme typique des problèmes que les méthodes d'apprentissage cherchent à résoudre (Section 3.1) avant d'étudier l'étape dite d'apprentissage (Section 3.2) à proprement dite. C'est cette étape qui se charge d'extraire l'information utile d'un jeu de données ou d'un environnement pour apprendre. C'est à ce moment qu'intervient l'optimisation d'une fonction et donc la production de l'algorithme appris. Finalement, nous reviendrons sur ce qui nous intéresse plus précisément dans ce mémoire : l'étude de ce produit fini qu'est l'algorithme appris, issu de la phase d'apprentissage (Section 3.3).

3.1 Problèmes d'apprentissage

Nous avons vu que *l'intelligence artificielle* avait envahi de nombreux domaines critiques et moins critiques de nos vies. Une des principales raisons de cet engouement est certainement le grand nombre de problèmes pratiques qui peuvent être formalisés en un problème soluble par des méthodes d'apprentissage.

3.1.1 Forme du problème

L'apprentissage automatique s'attache à construire de façon automatique une fonction, au sens mathématique, c'est-à-dire un objet qui associe à des éléments d'un espace de départ, des éléments d'un espace d'arrivée. On suppose alors que la fonction que l'on cherche à approximer existe, et, par induction à partir des données, on tente de construire une approximation de cette fonction. On tente de construire – ou d'approximer – cette loi par induction automatique.

Cas de la reconnaissance des chats Partons du postulat qu'il existe une fonction qui, à une image, associe un score de probabilité de présence d'un chat au sein de cette image. Il y a de bonnes chances qu'une telle fonction soit relativement régulière : un petit changement dans l'image ne devrait pas terriblement changer notre perception d'un chat ou non dedans. La question est alors : sommes-nous capables de construire une approximation de cette fonction de façon mécanique ?

Cas des échecs Un des éléments clefs pour la conception d'algorithmes jouant aux échecs est d'être capable d'évaluer les chances de victoire des joueurs dans un état donné du plateau et d'évaluer la valeur de chacun des coups jouables dans une position. L'idée est d'utiliser une telle fonction pour simuler les coups qui semblent les plus prometteurs, et non de tous les essayer, ce qui serait beaucoup trop coûteux en temps de calcul. Au lieu d'envisager tous les coups, on envisage seulement les 5 meilleurs, on creuse dans cette direction. *DeepBlue*, le célèbre algorithme d'IBM qui a vaincu Garry Kasparov en 1997[**?**], utilisait une fonction similaire conçue par des champions d'échecs et des spécialistes, à la main. L'idée est de simuler l'intuition humaine, l'intuition du joueur d'échecs qui sait quels coups sont envisageables et qu'il explore, et quels coups ne doivent même pas

être regardés ou pensés. La question moderne était donc : est-il possible d'approximer – d'apprendre – cette fonction d'intuition de façon mécanique ?

Lorsque l'on parle d'apprentissage automatique, on parle en fait de problèmes qui ont été mis sous cette forme d'approximation d'une fonction *a priori* insaisissable et que l'on espère être capable d'approximer de façon automatique : l'apprentissage.

3.1.2 Formes des solutions

Pour approximer ces fonctions, il s'agit d'abord de choisir un ensemble de fonctions, une classe de fonctions, dans laquelle on va choisir la meilleure approximation. En apprentissage automatique, on parle de classe d'hypothèse. On choisit en fait ici la forme des fonctions que l'on va considérer. Si *a priori* on peut considérer n'importe quel ensemble de fonctions, on choisit quasi systématiquement une classe de fonctions paramétriques, différentiables selon ses paramètres afin de pouvoir utiliser une descente de gradient pour les optimiser (Voir Section 3.2).

On parle souvent de modèles pour désigner cette classe paramétrique de la même façon qu'en physique : on fait l'hypothèse que la fonction que l'on tente de modéliser a une certaine forme et l'on cherche les paramètres qui permettent de coller au mieux aux observations. La classe de fonctions la plus classique est la droite : on suppose que la relation que l'on cherche est linéaire (choix de la forme), et on cherche la pente et l'ordonnée à l'origine qui permettent de coller au mieux aux observations. C'est exactement la régression linéaire, premier algorithme d'intelligence artificielle et on ne fait rien de plus ou de vraiment différents lorsque l'on utilise des réseaux de neurones.

Les réseaux de neurones sont en fait des classes de fonctions particulières composées de produits de matrices et de fonctions non linéaires (fonctions d'activation). Ceux-ci on la particularité d'être des approximateurs universels[**?**], c'est-à-dire que la classe des réseaux de neurones est dense dans l'espace des fonctions continues. En d'autres mots, on peut approximer aussi finement que l'on veut n'importe quelle fonction continue par une fonction ayant la forme d'un réseau de neurones.

Il existe d'ailleurs différents résultats théoriques plus ou moins puissants, qui vont notamment jusqu'à démontrer que l'on peut approximer n'importe quelle fonction (au

sens mathématique [1]) avec une précision arbitraire avec un réseaux de neurones à trois couches (quitte à faire grandir la taille de ces couches au besoin).

C'est en partie ce statut d'approximateurs universels qui a permis d'espérer pouvoir utiliser l'apprentissage automatique fondé sur des réseaux de neurones pour pouvoir *apprendre* n'importe quelle loi ou relation de façon totalement mécanique, quitte à avoir assez de données. En effet, en considérant que l'on a toutes les données nécessaires et une classe de fonctions dense dans les fonctions, on peut approcher et modéliser toutes les lois de la nature par *simple* optimisation. C'est la transparence absolue[**?**].

C'est aussi à cette étape que l'on adapte la classe de fonctions utilisée au problème : on préférera des modèles convolutionnels pour traiter des images[**?**] ou des modèles attentionnels [**?**] pour le traitement du langage naturel. Il n'est pas nécessaire de développer plus précisément les structures exactes de ces classes de fonction. Elles souffrent toutes des mêmes problèmes et la structure nous importe peu pour étudier le produit fini.

3.1.3 Choix d'une solution

Maintenant que l'on dispose d'un ensemble de fonctions candidates, il s'agit encore de choisir celle qui approxime le mieux (en un certain sens à définir) la fonction – *a priori* insaisissable – qui nous intéresse.

C'est à ce moment qu'entrent en jeu les données, la vraie vie. On utilise les points connus, les exemples résolus dont on dispose pour choisir la fonction de notre classe de fonctions qui se comporte le mieux sur ces données, c'est-à-dire qui donnent de bonnes réponses sur ces exemples.

On cherche alors des paramètres $\mathcal{W}$ qui minimisent une erreur construite à partir des données connues : $(\text{entrée}, \text{résultat\_attendu})$; notons T l'ensemble de ces couples d'exemples résolus.

$$\text{Err}_T(f_{\mathcal{W}}) = \frac{1}{|T|} \sum_{x,y \in T} l(f(x), y)$$

Il s'agit maintenant d'avoir une méthode pour chercher parmi toutes les fonctions can-

1. Qui a un élément d'un ensemble de départ associe un élément dans un ensemble d'arrivé.

didates celle qui minimise cette erreur.

3.2 Algorithmes d'apprentissage

Nous avons défini dans la section précédente la formulation du problème d'optimisation que l'on veut résoudre (Section 3.1), nous disposons d'un espace de fonctions candidates à explorer et d'un score permettant d'évaluer l'efficacité d'une de ces fonctions. L'apprentissage à proprement parler consiste à explorer cet espace pour trouver une fonction qui fonctionne le mieux possible.

En général, il n'existe pas une unique solution et on ne dispose pas de méthode générale garantissant de trouver une fonction qui atteint l'erreur minimale, et même de savoir quelle est l'erreur minimale atteignable. Dès lors, on se contente de trouver une fonction qui semble bien fonctionner. En général, on vérifie qu'elle se trouve au moins dans un minimum local, c'est-à-dire que les fonctions obtenues par de petites variations des paramètres ne donnent pas une meilleure fonction.

Les algorithmes d'apprentissage, sont en fait des algorithmes d'optimisation. L'objectif est de trouver la meilleure interpolation des points du jeu de données. C'est ici qu'il y un des principaux problèmes de sémantique en intelligence artificielle. On qualifie tout ce dont on parle "d'algorithme d'apprentissage" alors que l'algorithme d'apprentissage lui même est une partie spécifique de l'ensemble. L'algorithme d'apprentissage à proprement parler correspond à ce que nous allons voir ici : l'algorithme utilisé pour explorer l'espace des fonctions admissibles pour en trouver une bonne. A l'issue de cette exploration nous obtiendrons alors un algorithme "appris" (Section 3.3), issu de cette exploration – ou apprentissage – et qui constituera le produit fini, l'algorithme de décision finalement utilisable.

3.2.1 Méthodes d'optimisation

En règle générale, les algorithmes d'apprentissage se basent tous sur des méthodes de descente de gradient plus ou moins évoluées. De façon synthétique, l'algorithme de descente de gradient est un algorithme qui consiste à suivre la direction de plus forte pente jusqu'à atteindre un point où la pente est nulle et où l'accélération est positive. c'est-à-dire un

minimum local.

Formellement, supposons que l'on a une classe de fonctions paramétrées par des poids W, f_W et que l'on a une fonction d'erreur associée à ces poids $\mathrm{Err}_T(f_{\mathcal{W}})$ qui dépend du jeu d'entraînement T. On met à jour les poids de notre fonction de la façon suivante :

Algorithm 1 Gradient descent algorithm

for $s = 1$ to *Steps* **do**
  Computes gradient $g_i = \nabla_W \mathrm{Err}_T(f_{\mathcal{W}})$
  $W = W - \eta * g_i$
end for

Le paramètre η correspond à la taille des pas de descente de gradient que l'on effectue. Dans la formulation proposée ici, on calcule le gradient de l'erreur à partir de l'intégralité du jeu de données à chaque pas : c'est une formulation théorique. En pratique, il est impossible d'évaluer le gradient de la fonction à optimiser sur l'ensemble des données à cause de leur taille, ainsi on se contente généralement d'échantillonner à chaque pas un sous-ensemble du jeu de données et d'évaluer le gradient uniquement sur cette petite partie.

Algorithm 2 Stochastic Gradient descent algorithm

for $s = 1$ to *Steps* **do**
  Sample $\beta \subset T$
  Computes gradient $g_i = \nabla_W \mathrm{Err}_\beta(f_{\mathcal{W}})$
  $W = W - \eta * g_i$
end for

Si l'algorithme présenté ici est la version historique la plus simple, il existe aujourd'hui de nombreuses variantes qui prennent en compte non seulement l'inertie de la descente en cours ou qui adapte la taille des pas à effectuer pour chaque élément de W de façon indépendante mais le principe reste identique. Puisque ces algorithmes reposent sur un échantillonnage aléatoire du jeu de données pour évaluer le gradient à chaque pas, l'optimisation n'est pas déterministe et deux exécutions pourraient mener à deux solutions différentes. Il suffit d'imaginer que l'on commence l'optimisation sur une crête et qu'un sous ensemble du jeu de données nous donnent une pente plutôt d'un côté tandis qu'un autre nous aurait conduit à descendre de l'autre côté.

3.2.2 Stabilité de la convergence

Nous rencontrons ici une première classe de notions de robustesse qui s'intéresserait à l'algorithme d'apprentissage lui-même sur laquelle nous passerons rapidement car elle ne constitue pas le sujet de ce mémoire. Une question légitime que l'on pourrait se poser est celle de la stabilité de l'algorithme d'apprentissage : celui-ci étant bien souvent stochastique, que dire de la variance et de l'espérance des solutions qu'il produit ? Ou de l'efficacité des solutions produites ?

En effectuant plusieurs fois l'entraînement, sur les mêmes données, avec la même classe de fonctions à explorer, trouve-t-on des fonctions apprises similaires ? Ces fonctions de décisions proposent-elles toutes les mêmes décisions dans les mêmes situations ou divergent-elles ? (Eventuellement avec des efficacités globales similaires). Ou que se passe-t-il si les données varient un peu ?

Toutes ces questions traiteraient de l'algorithme d'optimisation, de sa convergence et de leurs éventuelles propriétés. En revanche nous nous intéresserons plutôt ici aux informations que l'on peut obtenir à partir du produit fini uniquement. La fonction de décision finale obtenue et maintenant étudiée comme une boîte noire dont la conception a été oubliée.

3.3 Algorithmes issus de l'apprentissage

3.3.1 Nature d'un algorithme appris

Le résultat de l'optimisation – ou apprentissage – est une fonction dont les paramètres – ou poids – ont été trouvés par l'algorithme d'optimisation vu précédemment. Cette fonction apprise est le résultat du choix de la classe de fonctions choisie, de la fonction d'erreur que l'on a choisi d'optimiser et des données dont on disposait. On peut en fait voir l'ensemble des étapes vues précédemment comme de la métaprogrammation consistant à définir les contraintes et propriétés que devrait vérifier le programme que l'on veut construire, puis la méthode d'optimisation construit effectivement ce programme. En somme, l'étape d'optimisation correspond à la recherche d'un programme (ici une fonction) qui vérifie au mieux les spécifications requises. La fonction apprise est alors le programme résultant de la métaprogrammation.

Dans le cas de l'apprentissage automatique, l'algorithme appris prend la forme d'une fonction paramétrées par des poids réels, en général des matrices, mais on pourrait tout à fait imaginer des formes différentes.

Le point crucial pour notre développement est de bien voir que ces fonctions apprises sont des objets issus de l'exploration guidée par la métaprogrammation que l'on a faite, et qu'elles peuvent – et doivent – faire l'objet d'études comme objets indépendants à part entière. Notamment pour vérifier que la fonction produite vérifie bien les spécifications et que ces spécifications ont bien permis de produire une fonction efficace sur les données d'entraînement comme sur les données de test.

3.3.2 Propriétés des algorithmes appris

On peut étudier les propriétés des fonctions ainsi obtenues, notamment pour avoir une idée des procédures ou règles de décisions effectivement induites par la métaprogrammation. On peut évidemment étudier la précision de la fonction obtenue, étudier ses variations ou encore ses réponses face à des situations particulières. Par exemple, une évaluation courante est d'étudier la distribution des réponses faites par une fonction apprise par rapport à la distribution des vraies réponses pour vérifier qu'il n'y a pas de biais évident.

Nous allons considérer la robustesse comme une propriété de ces fonctions apprises. Notre objectif est de réfléchir à des notions raisonnables de robustesse et de proposer des formulations mathématiques de celles-ci pour les fonctions apprises. Pour cela nous verrons plus précisément dans le chapitre suivant les sources et causes des problèmes de robustesse des algorithmes appris avant de proposer des énoncés formels de la robustesse.

Chapitre 4

Robustesse et intelligence artificielle

Avant de nous intéresser aux façons de mesurer la robustesse, nous allons essayer de tracer les contours des problèmes de robustesse que présentent les modèles d'apprentissage, et nous tenterons de donner des pistes expliquant leurs sources. Dans un premier temps, nous verrons que l'échantillonnage des données d'entraînement ignorent nécessairement une grande quantité d'événements rares pourtant existants (Section 4.1). En particulier, nous verrons que cela conduit à un surapprentissage – pas nécessairement négatif – similaire à celui que peuvent rencontrer les humains dans leur expérience de la vie (Section 4.1.2). Nous aborderons ensuite le problème de recouvrement de l'espace pas le jeu d'entraînement (Section 4.1.3), avant de voir que les *adversarial attacks* tirent profit des biais que ces éléments induisent (Section 4.2). Dans une dernière partie consacrée aux biais, nous nous intéresserons aux différents problèmes de biais. D'abord nous couvrirons rapidement les *biais socialement signifiants* traités par la fairness [36] (Section 4.3.1) avant de considérer, de façon plus générale, tous les biais relevant de vraies ou fausses corrélations et qui sont le plus souvent inintelligibles (Section 4.3.2).

4.1 Un problème de confrontation à la réalité

4.1.1 La distribution des données

Les algorithmes d'apprentissage apprennent en optimisant une fonction, de façon à ce qu'elle se comporte bien sur un jeu d'entraînement connu. On espère que la distribution des données (au sens statistique) est identique ou proche des données réelles que rencontrera cette fonction lors de son utilisation en pratique. Cette question de distribution des données d'entraînement est l'un des éléments-clés pour comprendre l'apparition de certains biais des algorithmes appris. Deux problèmes principaux se posent : tout d'abord pour que l'algorithme "apprenne correctement", il faut que le jeu d'entraînement ait une distribution identique ou très proche du monde réel dans lequel il va être utilisé ; ensuite, il faut s'assurer qu'il rencontre les cas rares pour qu'il puisse les traiter.

Concernant les cas rares, puisque le jeu de données d'entraînement est nécessairement fini et échantillonné – *a priori* – à partir du monde réel, il est possible qu'il ne contienne pas de cas particuliers rares, car, par définition, puisqu'ils sont rares, ils ont peu de chances d'êtres présents dans l'échantillon. Or, l'étude de la robustesse consiste aussi – voire surtout ? – en l'étude du comportement en pire cas, et donc potentiellement sur le comportement de l'algorithme face à ces cas particuliers rares qu'il peut n'avoir pas "vu" durant l'entraînement.

Un point important à comprendre est que sur les zones de l'espace dans lesquelles le modèle n'a pas été entraîné, c'est-à-dire des zones dans lesquelles il n'y avait pas ou peu de points de données, la fonction apprise est parfaitement aléatoire. On ne sait rien sur son comportement qui est indéfini, c'est-à-dire non spécifié : il peut arriver n'importe quoi.

Un exemple technique intéressant de ce problème est le problème des *distributions shift* : des cas où les données d'entraînement ne suivent pas la distribution des données de test ou de l'environnement de production. On se rend compte que les corrélations apprises par ces algorithmes ont plus souvent à voir avec le contexte dans lequel se trouve un objet que l'on cherche à reconnaître plutôt qu'avec les caractéristiques intrinsèques de l'objet en question [**?**]. Une vache sera très bien reconnue dans un contexte de champs verdoyants et probablement très mal au milieu d'une rue. On pourrait parler "d'objet situé" ou de "concept situé" appris par l'algorithme : il ne peut reconnaître un élément que dans son

contexte, et s'oriente vers autre chose si le contexte n'a aucun sens. Reconnaîtrions-nous une vache dans l'espace ?

4.1.2 Une analogie historique : le tremblement de terre de Lisbonne

Tout d'abord, commençons par rappeler ce qu'est et représente le tremblement de terre de Lisbonne. On parle en fait d'une succession de trois catastrophes qui se sont enchaînées à Lisbonne à partir du 1er novembre 1755. Premièrement, un tremblement de terre d'une ampleur inhabituelle – aujourd'hui estimée à $8,5$ ou 9 sur l'échelle de Richter – puis un tsunami, et finalement de nombreux incendies frappèrent la ville. Il fallut cinq jours pour maîtriser tous ces incendies. Environ un quart de la population de Lisbonne fut tuée, 10000 rien qu'à cause des premières secousses, d'une violence presque inégalée dans l'histoire humaine, et 85% des bâtiments de la ville furent rasés[**?**].

Cet épisode historique tient une importance particulière en philosophie car il est considéré comme le point de départ de la querelle entre Rousseau et Voltaire sur le statut du hasard et du rapport des hommes à celui-ci [**?**]. Ce tremblement de terre permet aussi de repenser le rapport au réel des Hommes, et c'est ce qui nous intéresse ici. Nous sommes habitués à vivre sans tremblement de terre, notre échantillon de la vie n'en contient pas – pour peu que nous vivions dans une zone géologiquement tranquille – et n'envisageons pas leur possibilité. Nous sommes d'une certaine façon trop spécialisés. Les algorithmes d'apprentissage souffrent des mêmes affres. Le tremblement de terre est un élément rare, probablement pas présent dans l'échantillon d'apprentissage, et ce n'est en moyenne d'ailleurs pas un problème. En effet, il n'est pas nécessaire de se préparer ou de savoir quoi faire en cas de tremblement de terre : ils sont assez rares pour que l'on survive assez pour que cela suffise *en moyenne*.

En raison de leur rareté et de l'absence de nécessité de s'y préparer, lorsqu'ils se produisent, ces évènements induisent la confusion, des réactions hasardeuses et aléatoires à cause du manque d'expérience des populations. C'est exactement le problème auquel font face les algorithmes d'apprentissage. Si leur base d'apprentissage leur permet de bien se comporter la plupart du temps, ils sont vulnérables face à des phénomènes ou situations rares.

Cette remarque peut apparaître comme une trivialité. Bien évidemment, un algorithme d'apprentissage risque de mal se comporter face à l'inconnu, face à des phénomènes

rares : par définition, ils apprennent du passé, de ce qu'on leur a présenté – comme l'humain d'ailleurs. Néanmoins, le problème est bien plus important qu'il n'y paraît. En effet, ici on a donné un exemple de phénomène rare mais perceptible [1] :ce n'est cependant pas le seul type de phénomènes rares auxquels sont sensibles les fonctions apprises par ces algorithmes. Prenons par exemple le cas de la classification d'images, de photos. Imaginons que par hasard, il n'y a aucune photo dont les pixels se trouvant en haut de l'image sont sombres ou noirs. On peut penser par exemple à des photos uniquement prises en extérieurs dont le haut représente donc principalement le ciel, bleu. Dans ce cas, la fonction apprise n'est pas préparée à l'évènement rare "le haut de l'image est sombre" et puisqu'elle n'a jamais pu être entraînée dans cette zone de l'espace des images, son comportement est indéfini, potentiellement explosif. Le problème – et le lien direct avec la robustesse –, c'est que ces événements rares-ci sont insaisissables *a priori* et donc extrêmement difficile à détecter. Il est tout à fait possible que mon algorithme d'apprentissage ait bien appris en moyenne sans que je me rende compte qu'il manque des photos avec un flash lumineux au centre qui brouillerait totalement la décision ou si une certaine forme particulière le perturbe.

4.1.3 Un exemple simple

On peut donner un exemple trivial de problème de recouvrement de l'espace par le jeu de données en utilisant le jeu de données *MNIST*. Ce jeu de données *Modified National Institute of Standards and Technology* correspond à un ensemble de *digits* écrits à la main, en blanc sur fond noir et servant de base pour la recherche en apprentissage automatique. C'est exemple est d'autant plus intéressant car ce jeu de données est le premier jeu utilisé pour la formation à l'apprentissage automatique, c'est généralement le premier que l'on utilise pour tester une nouvelle méthode d'apprentissage. Il a l'avantage d'être petit, de ne pas consommer trop de mémoire et d'avoir été utilisés dans la majorité des recherches précédentes, ce qui permet d'établir aisément des comparaisons. Mais il possède aussi de façon flagrante le défaut que nous essayons de pointer ici.

1. Sans mauvais jeu de mot.

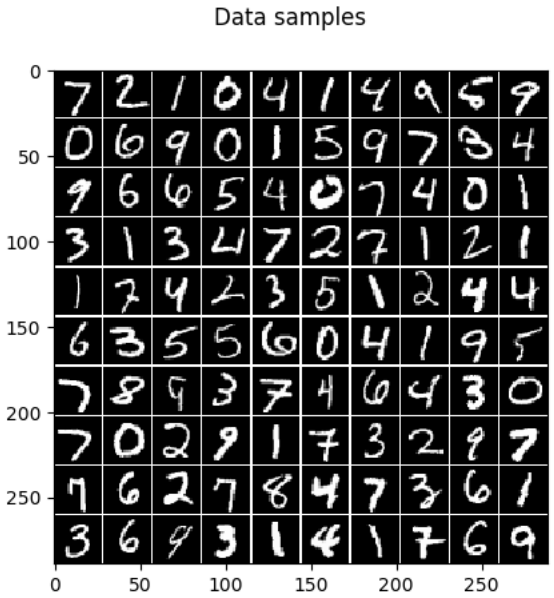

FIGURE 4.1 – Exemples d'images présentes dans le jeu de données *MNIST*

Dans la figure 4.1, on représente un échantillon d'images issues de ce jeu de données. Toutes ces images représentent un digit blanc, bien centré sur un fond uniformément noir. Le problème est là. Aucune de ces images n'a un fond qui varie. En fait, si l'on entraînait une fonction à classer ces images – pour reconnaître les chiffres écrits – cela fonctionnerait très bien, d'ailleurs on arrive à atteindre le taux d'erreur d'un humain... pour peu que les images présentées soient de la même forme : blanches sur fond noir, centrées etc... En revanche, si l'on commence à fournir des entrées dont les côtés – qui sont habituellement toujours noirs – sont allumés, les résultats deviennent imprévisibles – et absurdes.

Il y a une explication simple mathématiquement que je vais essayer de présenter le plus clairement possible. Pour entraîner une fonction, on calcule le gradient d'une fonction d'erreur pour avoir une idée de la direction dans laquelle mettre à jour les paramètres de notre fonction. Disons pour simplifier outrageusement que l'on a des images représentées par deux pixels noir et blancs. Dans ce cas, une image est un couple $(x, y) \in [0, 1]$, 0 signifiant que le pixel est éteint, noir et 1 qu'il est allumé au maximum, blanc. Imaginons la fonction que l'on cherche à apprendre à la forme $f_{a,b}(x, y) = a * x + b * y$ et que l'on cherche donc à optimiser d'une façon ou d'une autre les paramètres a et b. Supposons

maintenant que par hasard, dans les faits, pour toutes les images dont on dispose, le premier pixel du couple est systématiquement éteint, c'est-à-dire à 0. Pour entraîner notre fonction, on va prendre son gradient aux points que l'on connait, c'est-à-dire sur les images de notre jeu de données. Disons que l'on a l'image $(0, 0.8)$, sur celle-ci on a donc $f_{a,b}(0, 0.8) = a * 0 + b * 0.8$ et le gradient par rapport à a et b en ce point vaut $\nabla_{a,b} f(0, 0.8) = (0, 0.8)$. c'est-à-dire qu'en faisant augmenter b de 1, la fonction f augmente de 0.8 – et donc si on diminue b de 1, f diminue de 0.8, tandis que si on fait varier a, rien ne se passe car a est multiplié par 0, donc quelle que soit sa valeur, elle n'influe pas sur le résultat. Imaginons que notre but soit de faire en sorte que $f_{a,b}(0, 0.8)$ soit le plus petit possible, on calcule le gradient, on se rend compte qu'il suffit de faire des pas dans la direction $(0, -1)$, c'est-à-dire il suffit de diminuer b sans toucher à a. En fait, dans cette optimisation, a n'a aucune importance et n'a pas besoin d'être – et donc n'est jamais – mis à jour.

Or, lorsque que l'on entraîne ce genre de fonctions, on commence avec des paramètres tirés au hasard, et si certains sont toujours multipliés par des valeurs nulles dans les données, ils ne sont jamais mis à jour. Dans notre exemple précédent, imaginons que l'on obtienne finalement comme solution optimale pour notre problème $(-5, 0.2)$ pour (a, b), eh bien en général $f_{-5,0.2}(x, y) \in [0, 1]$ sur les données "habituelles", où le premier pixel est toujours nul. Imaginons un instant que le premier pixel ne soit pas nul dans un cas donné. La fonction prend alors une valeur absurde à cause de la valeur de a qui n'a jamais été entraînée.

Dans le cas d'un réseau de neurones dense, il y a au moins un poids du réseau de neurones par pixel multiplié uniquement par la valeur de ce pixel, donc dans le cas des images de *mnist* présentées ci-dessus, en fait une majorité des poids de la première couche du réseau de neurones ne sont pas entraînés ; car multiplié par une valeur nulle issue du jeu de données. Tous ces poids ont donc au final des valeurs parfaitement aléatoires, même si on a entraîné la fonction globalement, et même si la fonction apprise semble bien se comporter sur les données de test semblables : évidemment les données de test semblables affectent des valeurs nulles à ces pixels, et donc ces poids n'entrent même pas en jeu dans la décision. En revanche, si un jour ils entrent en jeu car un pixel habituellement nul ne l'est pas, leur effet est totalement aléatoire.

4.2 Des *Adversarial attacks*

Les *adversarial attacks* correspondent à des attaques que l'on peut construire contre des fonctions apprises pour modifier significativement leur résultat, sans pour autant modifier de façon visible ou importante l'entrée qu'on leur donne. Elles s'appuient d'une part sur le problème de couverture de l'espace d'entraînement vu précédemment (Section 4.1) et de la non-régularité des classes de fonctions employées en pratique.

4.2.1 Fonctionnement des *adversarial attacks*

Les *adversarial attacks* consistent en la recherche d'une entrée proche mais dont les modifications suffisent à faire changer la décision. L'idée consiste à chercher, à partir d'un point donné que l'on souhaite modifier légèrement, la direction qui modifie le plus la sortie. On cherche la plus petite altération de l'entrée qui provoque la plus grosse modification possible de la sortie. Par exemple, cela revient à se rendre compte que si l'on augmente un tout petit peu la quantité de bleu sur telle région de l'image, la classification inférée passe d'un chien à un chat. Formellement, il s'agit de résoudre un problème d'optimisation similaire à celui de l'entraînement de la fonction elle-même, mais cette fois-ci en faisant varier l'entrée et non les poids.

Disons que l'on dispose d'une fonction apprise f_W qui détecte la présence d'un chat dans une image : $f_W(x)$ est proche de 0 s'il n'y a pas de chat et proche de 1 s'il y en a un. On souhaite altérer une image x sur laquelle il n'y a pas de chat pour que l'algorithme appris croit qu'il y en a un. On va chercher pour cela δ une petite modification de x telle que δ minimise :

$$||\delta|| + ||f_W(x+\delta) - 1||$$

Le premier terme s'assure que l'on essaie de minimiser la taille de la modification que l'on effectue, tandis que le second terme cherche à garantir que $f_W(x+\delta)$ est proche de 1, c'est-à-dire de "il y a un chat dans cette image".

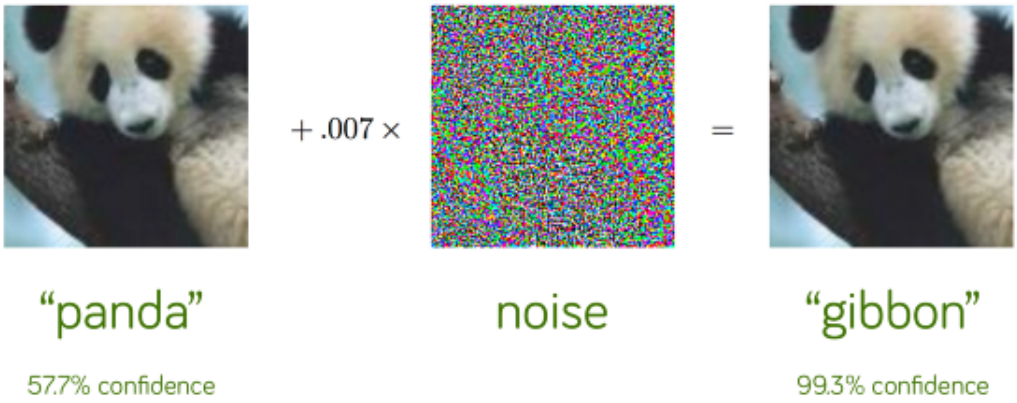

FIGURE 4.2 – Exemple d'*adversarial attack* qui trouve "un bruit blanc bien choisi" qui suffit à induire en erreur le modèle sans pour altérer l'image pour un œil humain.

4.2.2 De l'interpolation en grandes dimensions

Comme nous l'avons vu précédemment, l'apprentissage automatique relève de résoudre un problème d'interpolation – ou d'approximer une interpolation – en grande dimension. Un problème bien connu est la non-régularité des solutions de ce genre de problèmes. Les solutions d'interpolations en grandes dimensions, si elles passent bien par les points spécifiés, ont tendance à osciller énormément entre ceux-ci ou à exploser très rapidement lorsqu'on s'en éloigne. Le problème c'est que ces oscillations permettent de changer significativement la sortie de la fonction apprise en modifiant que très légèrement son entrée – à cause des pentes très importantes qui existent.

C'est cela qui permet aux *adversarial attacks* d'être efficaces – c'est-à-dire être capable de modifier significativement la sortie – tout en n'utilisant que des modifications de faible magnitude sur les images, imperceptibles pour un humain.

Il faut reconnaître qu'il est très difficile en pratique d'employer de telles attaques. On ne peut que très difficilement ajouter un "un bruit blanc bien choisi" sur une caméra de voiture autonome par exemple. Néanmoins, leur existence remet en doute fortement la sûreté des algorithmes appris. En effet, ce que nous disent ces méthodes, c'est qu'il est souvent possible de faire un tout petit pas de côté sur une entrée pour obtenir une

réponse tout à fait différente. Quelle confiance peut-on alors avoir dans des systèmes sensibles à d'aussi infimes fluctuations ? Leurs réponses sont-elles seulement sensées ? En médecine, cela pourrait signifier qu'un dossier médical dans lequel toutes les valeurs numériques varieraient infimement pourrait avoir un diagnostic complètement différent.

4.2.3 Aggravée par les zones non explorées

Les zones de l'espace non couvertes par le jeu d'entraînement décrites Section 4.1.3 augmentent les potentiels effets des *adversarial attacks*. En effet, le nombre potentiellement important de poids encore initialisés à leur valeur de départ, aléatoire, donne une grande marge de manoeuvre à ces attaques pour effectuer des modifications imperceptibles avec des effets importants. Pour reprendre l'exemple de *MNIST*, une *adversarial attack* peut facilement détecter qu'il lui suffit d'éclairer au minimum certains pixels du fond habituellement éteints pour activer des poids très importants et changer totalement le résultat.

4.3 Aux biais

4.3.1 Tangibles

Il y a quelques années, *Amazon*[**?**] a essayé d'entraîner un algorithme d'apprentissage pour pré-étudier les CV qui lui étaient envoyés. Ils ont entraîné une fonction à prédire l'adéquation entre un CV et un poste dans l'entreprise sous la forme d'une fonction de classification. Pour constituer leur jeu de données, c'est-à-dire la vérité (ou *ground truth* en anglais), ils se sont basés ce qu'ils faisaient déjà. Ils ont pris les CV de toutes les personnes déjà en poste et ont décrété que ces CV étaient parfaits pour le poste qu'ils occupaient. Il s'agit alors d'apprendre une fonction qui reproduit les méthodes de recrutement humaines.

Il s'est avéré par la suite que la fonction qu'ils avaient entraînée était furieusement sexiste, affectant presque automatiquement les femmes à des postes de secrétariat et presque jamais à des postes d'ingénierie. Ce biais est – une fois pointé – parfaitement évident : usuellement, les femmes sont sous-représentées en ingénierie et sur-représentées dans les tâches de support – et ce, en raison du recrutement humain existant. L'algorithme

d'apprentissage n'a fait que reproduire – apprendre – le biais existant intrinsèquement dans les données, puisqu'on lui a dit que le "bien" c'était ce que faisaient déjà les humains recruteurs – faute de mieux.

Ce type de biais est généralement étudié lorsque l'on parle de *fairness*, d'équité dans les algorithmes d'apprentissage. En effet, la *fairness* s'intéresse aux biais macroscopiques et discriminant socialement ou qui posent problème politiquement. Mais il nous intéresse ici car, en réalité, la robustesse s'intéresse à une classe plus générale des biais qui inclue ces biais problématiques socialement mais pas seulement.

En fait, la robustesse s'intéresse à étudier si ce que l'on a appris correspond à des informations intrinsèques à la tâche à accomplir, ou si l'on a en réalité appris que de fausses corrélations. Un système qui apprend des corrélations fausses ne peut être robuste : par définition il suffirait de présenter un exemple qui vérifie parfaitement la corrélation, mais qui n'a aucun des traits qui nous intéressent. Dans le cas d'*Amazon*, on parle d'un homme pris en ingénierie car statistiquement les hommes sont ingénieurs mais qui ne serait pas ingénieur du tout, ou d'une femme refusée à un poste d'ingénieur alors qu'elle serait parfaitement qualifiée car l'algorithme a appris que les femmes ne sont pas ingénieurs. Dans ces deux cas, l'algorithme appris échoue à extraire le niveau réel en ingénierie de la personne concernée.

4.3.2 Intangibles

Pour illustrer les dangers des biais et les difficultés que représente leur détection, nous pouvons utiliser une expérience de pensée très simple. Disons que la *CIA* cherche à développer un algorithme de reconnaissance de terroristes sur des photos. L'objectif est d'être capable d'utiliser cet algorithme pour détecter si, d'après son apparence physique, une personne est susceptible d'être un terroriste. Nous ignorerons, pour les besoins de notre expérience, les questions éthiques et morales qu'un tel objectif représente : c'est après tout un projet de la *CIA* qui ne s'embarrasserait certainement pas de telles problématiques.

Imaginons le mode opératoire que pourraient employer les informaticiens de la *CIA* pour construire un tel algorithme. Le principal problème est de construire un jeu de données d'entraînement, et cette tâche n'est pas particulièrement facile dans ce cas : les photos de terroristes ne pas si courantes. Mais cela n'est pas si grave, la *CIA* est certainement

la mieux placée pour disposer de photos de terroristes ou pour en obtenir. Il a été décidé que l'agence se procurerait ces photos *via* ses agents et l'armée qui seraient chargés dans le cadre de leurs missions de photographier des terroristes formellement identifiés comme tels pour fournir la base de données. Afin de garantir qu'il n'y a pas d'images faussement étiquetées comme terroristes, l'agence ne fait confiance qu'à des photos prises par ses propres agents et par l'armée. C'était la partie la plus difficile, il s'agit maintenant de réunir des photos de non-terroristes, rien de plus simple : il suffit d'en prendre un peu partout sur internet et on peut se dire qu'on a aucune chance de prendre une photo d'un véritable terroriste sur les réseaux sociaux classiques.

Nous voilà donc avec un jeu de données constitué de véritables photos de terroristes, prises, formatées et traitées par des agents officiels sérieux et des photos classiques de personnes non terroristes. On entraîne une fonction à classifier ces photos et on le teste sur de nouvelles photos (des photos de non terroristes venant d'internet et des photos de terroristes avérés prises par la *CIA*), et là, miracle : l'algorithme d'apprentissage a fonctionné, la fonction classifie bien les nouvelles images de test.

Alors, s'il est évident que cette approche est absurde – décider des aspirations terroristes d'une personne uniquement à partir de son apparence ne peut être une méthode raisonnable – pourquoi fonctionnerait-elle dans ce cas ou sur ces tests ? D'abord, on peut imaginer des biais évidents : les terroristes traqués par la *CIA* sont certainement musulmans traditionalistes, portent la barbe ou des tenues traditionnelles. Mais maintenant, imaginons que je vous dise que la *CIA* a pris une photo d'un terroriste américain d'extrême droite (qui par définition ne porte aucun trait de musulman traditionaliste), la donne à l'algorithme et là, miracle encore une fois, il le classifie correctement comme un terroriste. Et même, si on prend une photo provenant des réseaux sociaux d'une personne avec les traits caractéristiques d'un musulman traditionaliste, il n'est pas classifié comme terroriste ! Formidable, il semblerait que notre outil fonctionne ! Il n'est pas raciste, il est capable de détecter les terroristes et de ne pas cibler à partir de la tenue ou de l'apparence extérieure.

Alors, il existe bien une explication possible et je vous l'annonce tout de suite, ce n'est pas que l'algorithme fonctionne réellement. J'ai bien insisté sur le fait que toutes les photos de vrais terroristes étaient prises par des agents de l'armée ou de la *CIA* ; on peut imaginer qu'ils ont un matériel normalisé : mêmes modèles d'appareils photos ou mêmes optiques etc... Ces outils ont donc potentiellement les mêmes imperfections, conduisent aux mêmes artefacts imperceptibles sur les photos. Imperfections et artefacts qui ne

sont pas présents dans l'immense majorité des photographies prises sur internet issues de différents appareils photos dans différentes configurations avec différentes optiques. On pourrait découvrir dans cette histoire qu'en fait la principale corrélation – fausse corrélation en réalité – est la présence de certaines imperfections précises sur les photos de la *CIA* (et donc spécifiquement sur les photos de terroristes avérés) et leur absence sur les photos de non-terroristes (prises sur internet).

En fait, ce modèle décide arbitrairement qu'une photo prise par la *CIA* (*i.e.* avec son matériel) est nécessairement une photo de terroriste. C'est un biais particulièrement dangereux car il s'inscrit dans un cadre où il est difficilement repérable. D'abord à cause des biais de confirmation : si la *CIA* prend une personne en photo et se demande si c'est un terroriste, il y a de bonnes chances qu'elle pense déjà que c'est un terroriste et ne sera donc certainement pas prompte à remettre la réponse de l'algorithme en question. De plus, c'est aussi certainement une bonne approximation : si la personne est suivie par la *CIA*, elle a effectivement beaucoup plus de chances d'être un terroriste. Ces deux éléments font que la détection d'un tel biais serait assez improbable ou en tous cas très longue.

Et c'est aussi un problème de robustesse. Par exemple, qu'un vrai terroriste pris en photo avec un appareil photo grand public ne serait certainement pas classifié comme terroriste (alors que les images sembleraient parfaitement identiques). En fait, on peut dire que la robustesse s'attelle à détecter les biais de façon générale, et pas seulement les biais qui portent une signification sociale. En effet, une façon de s'assurer de la robustesse d'un algorithme appris, c'est d'étudier les éléments qu'il prend en compte dans sa décision ou des corrélations qu'il établit. Si on se rend compte qu'il utilise des corrélations absurdes (des artefacts sur des images, des pixels sombres en fond etc...), on se rend compte de potentielles failles où ces (fausses) corrélations causeraient des erreurs de jugement.

Chapitre 5

Robustesse formelle

Nous allons maintenant nous concentrer sur les approches théoriques formelles de la robustesse des algorithmes appris issus d'algorithmes d'apprentissage ou d'optimisation. Notre objectif dans ce chapitre va être de développer des formulations théoriques mathématiques de la robustesse. Nous en détaillerons d'abord la formalisation (Section 5.1), puis les implications théoriques et philosophiques pour comprendre les différents sens et effets qu'elles peuvent produire. Tout d'abord, nous nous intéresserons à une idée naïve de la robustesse (Section 5.2) qui consiste à décider qu'un algorithme appris est robuste s'il produit des résultats similaires pour des entrées similaires. Nous verrons que que si elle semble satisfaisante à première vue, elle nous ramène en réalité au problème d'apprentissage initial. De là, nous nous concentrerons sur deux autres approches de la robustesse. Premièrement, nous nous intéresserons à la robustesse des modèles comme une propriété intrinsèque de ceux-ci, avant de voir en second temps la robustesse comme la quantité d'information que l'évaluation sur une certaine distribution nous apporte sur le comportement du modèle face aux données de la vie.

5.1 Cadre formel

On suppose que l'on dispose ici d'une fonction entraînée qui cherche à classifier des éléments d'un ensemble $X \subset \mathbb{R}^N$ – avec N potentiellement très grand – en K classes. On a donc une fonction de classification $f : X \longrightarrow [0,1]^K$ apprise – ou entraînée – telle

que pour $x \in X$ et $1 \leqslant k \leqslant K$, $f(x)_k$ est la probabilité que x appartienne à la classe k [1]. On suppose aussi que f est différentiable. Cette dernière hypothèse n'est pas absolument nécessaire mais elle est valable pour l'immense majorité des algorithmes d'apprentissages profonds et nous simplifiera la vie.

On va penser cette fonction apprise comme une boîte noire qui nous est fournie pré-faite – *i.e.* pré-entraînée – et on cherche à l'étudier pour s'assurer qu'elle vérifie des critères de robustesse qui peuvent nous intéresser. Cette formulation très faible est cependant assez représentative de situations qui arrivent fréquemment en pratique. On peut simplement souhaiter évaluer l'efficacité de la fonction apprise, mais pas seulement. Il arrive régulièrement que l'on dispose de fonctions entraînées pour lesquelles on a perdu les données ou paramètres d'entraînement, ou pour lesquelles ceux-ci ne sont pas divulgués, par exemple lorsqu'une entreprise déploie un modèle appris sans pour autant divulguer les données ou la méthode d'apprentissage.

De plus, si l'on considère l'apprentissage automatique comme une forme de métaprogrammation dont le résultat est un programme généré automatiquement (= la fonction apprise), ce qui nous intéresse plus particulièrement, ce sont les propriétés de ce résultat obtenu plus que la façon dont il a été obtenu.

5.2 Une première approche naïve

Une première idée de la robustesse peut être de dire que des éléments proches doivent avoir des images proches par la fonction de classification. On suppose alors implicitement que le monde, ou la tâche de classification que l'on essaie de traiter est elle-même continue et régulière [2]. On peut tout d'abord pointer que si c'est une hypothèse qui peut sembler assez raisonnable puisque la majeurde partie des phénomènes sont relativement continus, il est loin d'être évident que ce soit le cas tout le temps. D'autant que l'on emploie des algorithmes d'apprentissage automatique dans les situations où on a *a priori* pas une bonne idée de ce que serait une bonne façon de classifier ou de quelles *features* considérer : dès lors, il est difficile de prétendre savoir si le phénomène est effectivement continu ou non. Mais même en admettant cette hypothèse, nous allons voir que cette formulation souffre de problèmes plus fondamentaux.

1. On note ici $f(x)_k$ la k-ième composante du vecteur de probabilité $f(x) \in [0,1]^K$.
2. `https://fr.wikipedia.org/wiki/Classe_de_r%C3%A9gularit%C3%A9`

5.2.1 Formulation

Pour pouvoir envisager une telle formulation de la robustesse, il est nécessaire de munir l'espace d'entrée X de notre fonction de classification d'une notion de distance. En effet, on cherche à exprimer l'idée que des entrées puissent être "proches" en un sens raisonnable pour nous permettre de dire que l'on veut que la fonction donne la même réponse sur ces entrées. Un gros travail à mener est alors de définir cette notion de distance en rapport avec la tâche spécifique qui nous intéresse.

Supposons que l'on dispose d'une telle notion de distance $d_X : X \times X \longrightarrow \mathbb{R}_+$, il nous faut aussi une notion de distance sur les vecteurs de probabilité que donne la fonction, cette distance est plus facile à concevoir car elle ne dépend pas *a priori* de la tâche que l'on cherche à accomplir. Il existe différentes options de distance sur ces vecteurs de probabilité disponibles : de la simple distance euclidienne à des distances sur les distributions de probabilité. On suppose que l'on a choisi l'une d'elles $d_p[0,1]^K \times [0,1]^K \longrightarrow \mathbb{R}_+$.

On va alors s'intéresser à la constante de Lipschitz de notre fonction apprise. Une fonction est dite L-Lipschitzienne s'il existe une constante L telle que

$$\forall x, y \in X, d_p(f(x), f(y)) \leqslant L d_X(x, y)$$

.

Cela dit exactement que les images de x et y par f sont au plus L fois plus éloignés que le sont x et y dans l'espace de départ. Cette formulation pour les fonctions de façon générale est équivalente, pour les fonctions différentiables, à dire que leur dérivée est bornée par L.

$$\forall x \in X, ||\nabla f(x)|| \leqslant L$$

Cela dit exactement que les variations de la fonction sont bornées : il n'existe pas de pente plus pentue que L, et donc cela nous garantit en particulier que pour s'éloigner d'un point, il faut parcourir un distance minimale définie par cette pente maximale L. On envisage alors dans cette situation d'évaluer la valeur de cette pente L et de l'utiliser comme métrique de robustesse. Plus L est petit, plus il faut que deux points soient

éloignés pour avoir des images très différentes.

Dans le cas d'un *classifier*, on définit donc une norme sur l'espace des éléments à classifier, en général une norme L_2, L_1 ou L_∞, et une norme sur les distributions de probabilité sur les classes en sortie. Les choix de normes peuvent varier largement suivant la tâche à accomplir et les notions de distance qui nous intéressent. On peut alors calculer la constante de Lipschitz pour ces normes, et si les normes choisies traduisent un sens et des propriétés du problème, alors on peut considérer que cette constante nous renseigne sur le niveau de robustesse de la fonction. En particulier, elle nous donne des bornes intéressantes sur la distance minimale à parcourir à partir d'un point donné pour le faire changer de classe. Si cette distance est suffisamment importante pour impliquer des changements macroscopiques significatifs pour la tâche à accomplir, alors la fonction de classification est robuste.

5.2.2 Du choix de la norme

Cette formulation transfère tout le problème de la robustesse au choix de la définition "d'élements proches" dans l'espace de départ. Dire que des éléments proches dans l'espace de départ doivent avoir des images proches par la fonction de classification définit en soi un problème de classification – non supervisé en l'occurrence, et donc *a priori* plus difficile encore que le problème de classification supervisé qui nous intéresse. En fait, choisir une notion de proximité sur l'espace de départ pour garantir que des éléments proches ont la même classification revient exactement à résoudre le problème de classification lui même. Dès lors, il est évident que cette formulation ne pourra permettre de résoudre complètement notre problème de mesure de la robustesse.

Néanmoins, cette formulation permet de traiter des cas locaux, notamment en imagerie computationnelle. En effet, on peut admettre que des modifications d'une image imperceptibles à l'oeil humain ne devraient pas engendrer de modifications significatives de la classification. On note alors que ce point de vue sur la robustesse permet notamment de traiter la question des *adversarial attacks* dont la principale menace est d'altérer le résultat de la fonction par une petite modification de l'entrée, sans même qu'elle soit détectable à l'œil. En effet, pour rappel, les *adversarial attacks* consistent à chercher la meilleure façon d'altérer de façon minimale une entrée pour en changer largement l'image par la fonction de classification. Si on dispose d'une constante de Lipschitz assez faible, alors il n'est pas possible de produire une petite modification de l'entrée provocant une

importante modification de la sortie.

5.3 La robustesse : une généralisation de la *fairness*

Maintenant que nous avons traité la notion la plus courante, et probablement la plus simple, de la robustesse, nous allons étudier la robustesse d'un modèle comme sa capacité à généraliser correctement, c'est-à-dire à apprendre de vraies corrélations. Pour cela, nous allons commencer par voir que la robustesse est en ce sens une généralisation des questions de *fairness*. Les algorithmes d'apprentissage sont généralement mal nommés et désignés par les termes "intelligence artificielle" : il serait plus juste de parler de détection automatique de motifs ou de machines à trouver des corrélations. En effet, les algorithmes d'apprentissage sont uniquement des machines à induction automatique qui tentent de généraliser – c'est-à-dire produire des lois générales – à partir d'exemples. La question fondamentale est donc de savoir quelles corrélations, quelles lois ont été inférées par l'apprentissage à partir des données. Si on peut identifier certaines de ces lois, ou en tout cas certains aspects, on peut alors en tant *qu'expert humain* se demander si elles semblent raisonnables ou sensées. Si ce n'est pas le cas, on est alors capable de rejeter le modèle appris. Et c'est exactement ce qui est fait en *fairness*. Il est important de noter ici que cette expertise humaine est loin d'être toujours possible, et c'est la différence entre ce que nous avons appelés des biais intelligibles et des biais non-intelligibles (Section 4.3). Si on a des biais signifiants socialement, on peut les reconnaître lorsqu'on les voit et on est capable de savoir si on souhaite les conserver ou non : par exemple emploi du genre pour décider de l'adéquation d'une personne pour un emploi donné. En revanche, lorsque ces biais sont de la seconde catégorie, même si on les voit, on est incapable de décider si ce sont de bonnes ou de mauvaises corrélations. Dans l'exemple des photos de la *CIA* (Section 4.3.2), il est à peu près impossible qu'un humain soit capable de se rendre compte que "l'intérêt porté par l'algorithme à tel groupe de pixels dans certaines situations" résulte en fait de l'apprentissage des défauts de l'optique employée.

La *fairness* consiste en l'étude des modèles appris pour s'assurer qu'ils n'utilisent pas de *features* discriminatoires (socialement) pour prendre des décisions. En effet, dans la majorité des cas on est capable de décider que telle ou telle caractéristique (la couleur de peau, la race, le genre, l'orientation sexuelle etc...) n'a aucune raison d'être utilisée ou de servir à discriminer. On peut savoir en effet que même si des corrélations impliquant ces

caractéristiques existent dans les données, celles-ci ne sont que de fausses corrélations [3]. Or, c'est une information que l'algorithme d'apprentissage n'a *a priori* pas et va donc les détecter et s'en servir. La *fairness* consiste alors à détecter l'emploi de ces fausses corrélations et de les interdire.

Il s'agit en fait uniquement de détecter des fausses corrélations particulières, de fausses corrélations qui ont un sens social macroscopique si l'on peut dire. La robustesse va s'attacher de façon plus large à tenter de détecter les fausses corrélations en général, pas seulement les fausses corrélations qui ont un sens social : le genre de fausse corrélation qui serait induit par les défauts optiques des appareils photos utilisés dans l'exemple des terroristes de la *CIA*(Section 4.3.2), ou encore dans le cas des polices de caractères utilisées pour prédire le COVID 19 [**?**]. Il faut noter que, dans ces exemples, les fausses corrélations, bien que difficiles à détecter, restent sensées, appréhensibles une fois expliquées : macroscropiques. Il existe en réalité une infinité de fausses corrélations qui n'ont aucun sens macroscopique, potentiellement des combinaisons de pixels courants mais sans rapport avec les classes, ou des combinaisons de lumières qui, même détectées, ne seraient pas appréhensibles.

De ce point de vue, on va alors s'intéresser aux propriétés intrinsèques de la fonction apprise par l'algorithme d'apprentissage et la robustesse, ou la non robustesse, va découler des résultats des travaux d'interprétabilité et d'explicabilité à l'échelle du modèle. Nous allons en particulier voir une première méthode qui exploite cette notion de robustesse. Étant donné qu'il est extrêmement difficile de reconnaître si un biais correspond à une vraie ou à une fausse corrélation, cette méthode s'en tient à s'assurer que les classes apprises par l'algorithme utilisent bien des corrélations différentes, variées et importantes pour les discriminer au mieux. Si deux classes semblent "trop proches", c'est qu'il existe des façons simples pour passer de l'une à l'autre, c'est-à-dire des biais facilement actionnables pour modifier une classification.

5.3.1 Un exemple de méthode d'évaluation de la robustesse

Une idée raisonnable pour savoir si un algorithme – ou en l'occurrence n'importe quelle entité apprenante – a bien appris différents concepts est de s'intéresser à la netteté de ces concepts dans ses représentations internes – ou son esprit si on parle de quelque chose

3. Ou alors on décide politiquement qu'on ne souhaite pas se servir de ces caractéristiques pour des raisons morales.

de plus évolué qu'un réseau de neurones. En particulier, on va s'intéresser à des notions de marge et de séparabilité des concepts dans la représentation interne de l'algorithme appris. Pour cela, on va donner une définition formelle de la notion de concept appris. Cette définition de concept va en fait être un outil pour étudier les corrélations apprises. En étudiant ces concepts, nous allons pouvoir observer quelles sont les caractéristiques importantes pour effectuer des classifications selon la fonction apprise, nous renseignant sur les potentiels biais de celle-ci. Dans l'exemple des CV d'*Amazon* développé précédemment (Section 4.3.1), on aurait donc ici un générateur de CV auquel on pourrait demander, disons des exemples de CV d'ingénieurs. On se rendrait certainement assez vite compte que ce générateur ne produit que des CV d'hommes. Mais cela repose encore encore sur une expertise humaine qui serait capable de décider si un biais est une vraie corrélation ou non. En revanche, nous allons pouvoir définir une notion de distance sur les concepts permettant de mesurer une quantité "corrélations similaires" entre deux classes, c'est-à-dire à quel point le modèle prend en compte les mêmes éléments, dans les mêmes directions pour discriminer entre des classes. Si deux classes sont proches c'est que le modèle a du mal à les discriminer, tandis que si elles sont éloignées, le modèle a réussi à trouver des discriminateurs efficaces. On va dire qu'un modèle est robuste s'il "utilise des *features* bien séparées pour discriminer chaque classe".

Avant d'entrer dans le cœur du sujet, commençons par une analogie pour comprendre ce dont on va parler et ce que l'on cherche à faire ici. Imaginons que nous avons un enfant qui est censé avoir appris à reconnaître des numéraux formant des entiers, il affirme qu'il sait discriminer les symboles 1, 2 etc... En somme, qu'il a construit une représentation interne abstraite de ces éléments qui lui permette de les différencier, de les reconnaître, voire de les reproduire. On peut alors pour vérifier cela simplement fournir à cet enfant des exemples, et lui demander de les reconnaître – c'est la méthode la plus classique – mais on peut aussi lui demander de nous dessiner plein d'exemples pour chacun des numéraux en question. On va alors pouvoir voir à quel point le concept associé au symbole 5 est proche du concept du symbole 8 en regardant si les dessins produits pour ces deux classes sont proches, si certains se ressemblent beaucoup etc... En fait, on va bâtir notre définition de concept à partir de cette idée. Le concept appris correspond à la distribution de probabilité conditionnée par la classe sur ces dessins, dans le cas des numéraux. Le concept associé au signe 5, c'est la distribution des dessins de symboles produits par l'enfant lorsqu'on lui demande de dessiner un 5.

On peut alors observer les exemples qui sont produits par ces générateurs adversariaux

et déjà avoir une idée des biais ou des fausses corrélations qui ont pu être appris par le modèle d'origine en regardant quels motifs, formes ou images sont produits par le générateurs pour produire des images de la classe voulue. On peut observer à l'œil les dessins produits. Que dire si, lorsque l'on demande de dessiner un 5, on obtient une image ou un dessin contenant uniquement du bruit sans pouvoir distinguer quelque forme que ce soit ? Que dire si au contraire, à chaque fois que l'on demande un dessin de 5, on obtient un symbole de 5 bien net ? Ou alors si parfois le symbole semble se confondre avec un symbole de 8 ? Dans le premier cas, on peut certainement s'inquiéter que notre fonction de classification risque d'identifier des symboles là où il n'y a rien et potentiellement produire des résultats absurdes. c'est-à-dire qu'il existe des biais ou de fausses corrélations qui sont pourtant employés par notre fonction de classification [4]. Dans le second cas, il semble que tout va pour le mieux, tandis que pour le dernier, on identifie de potentielles collisions ou confusions entre les symboles de 5 et de 8. On peut en conclure que notre fonction ne sait pas efficacement les différentier.

Formulation formelle

On va s'intéresser pour toute classe y à la distribution $G(y)$ telle que si $X \sim G(y)$ [5], on a $C(X) = y$ pour toute réalisation de X. C'est exactement la distribution génératrice conditionnée à la classe y pour le *classifier* C. On dit alors que $G(y)$ est le concept de y appris par C. Cet objet est une distribution de probabilité sur les entrées du *classifier* et on peut alors s'intéresser aux relations entre les distributions génératrices de chacune des classes pour un *classifier*. En particulier, on peut étudier la séparation entre ces distributions de probabilité selon différentes notions de distances ou divergences sur les distributions.

Être robuste peut alors être compris comme "utilise des *features* bien séparées pour discrimner les classes" : si les concepts sont éloignés alors le modèle est capable de les séparer. Il se trouve que cette formulation permet aussi de capturer la robustesse face aux *adversarial attacks*. En effet, les *adversarial attacks* tirent profit des corrélations apprises par l'algorithme pour l'induire en erreur : il suffit de trouver une corrélation forte – qui se traduit alors par une pente forte de la fonction dans cette direction – pour modifier

4. Ces corrélations sont évidemment fausses puisque l'on peut observer que l'image ne contient que du bruit : le modèle a donc nécessairement appris des corrélations qui permettent de conclure dans cette situation à un certain entier alors qu'il n'y en a pas.

5. X est une variable aléatoire suivant la distribution $G(y)$.

la classification. Si on peut s'assurer que les corrélations utilisées pour discriminer deux classes sont bien séparées alors on réduit les risques liés aux *adversarial attacks*.

Implémentation pratique

Dans le cadre d'un travail de recherche en mathématiques, j'ai implémenté la méthode décrite précédemment pour apprendre les distributions génératrices associées à un modèle de classification et étudié l'influence d'un *adversarial training* sur les marges entre les concepts appris par le *classifier*. Pour ce faire, on apprend pour chaque classe une distribution paramétrique $\pi_{\theta y}$, par exemple une distribution normale et un *decoder* $D_{\theta y}$ à la façon de différents *variational autoencoders*, de sorte à fabriquer un générateur par classe du *classifier*. On note $G_y = D_y(x), x \sim \pi_{\theta y}$ la variable aléatoire associée à la classe y. On optimise alors ces générateurs de paramètres θ_y de telle sorte que $\mathbb{P}[C(G_y) = y] \approx 1$. c'est-à-dire que la probabilité que le générateur d'exemples aléatoire pour la classe y produise une entrée effectivement classifiée comme y soit proche de 1.

Une fois cet entraînement effectué, on peut alors calculer les distances sur les distributions π_y, et notamment évaluer la distance moyenne entre les concepts ou la distance minimale. Afin de vérifier que c'est une notion raisonnable de robustesse, on va montrer que plus on entraine de façon *adversarial* un *classifier*, plus ces concepts s'éloignent pour différentes divergences. Expérimentalement, on effectue un entraînement *adversarial* avec une *projected gradient descent* pour produire des attaques non ciblées (on veut juste provoquer une mauvaise classification, on ne cherche pas à choisir quelle sera la nouvelle classe) avec une taille de boule fixée. une fois cet entraînement réalisé, on apprend les distributions génératrices pour ce *classifier*. On va effectuer cette opération pour différentes tailles de boules croissantes – donc avec des entraînements qui devraient produire des modèles de plus en plus robustes – et on répète cette expérience une centaine de fois pour chaque taille de boule afin d'obtenir des statistiques raisonnables. On calcule à chaque fois l'entropie moyenne des distributions ainsi que la distance moyenne entre chaque concept et la distance minimale pour la distance de Hellinger [**?**].

La distance de Hellinger est une distance sur les distributions de probabilité qui capture la quantité de superposition entre deux distributions. Et c'est bien ce qui nous importe ici : on souhaite voir à quel point la fonction apprise doute d'une classification. Si les distributions génératrices se superposent, alors les éléments de l'espace présents dans ces superpositions sont des éléments pour lesquels la fonction de classification est indécise.

Je pourrais avoir obtenu cette image en demandant un symbole 5 ou un symbole 8. Réciproquement, si les distributions génératrices ne se superposent presque pas, cela signifie que les éléments des différentes classes sont bien séparés dans l'espace, et donc que les marges entre celles-ci sont assez importantes pour éviter des confusions.

(a) Avergage hellinger distance between concepts (b) Min helinger distance between concepts

FIGURE 5.1 – Corrélation entre la distance de Hellinger entre les classes et la taille de la boule utilisée lors de l'*adversarial training*.

(a) Entropie moyenne des distributions (b) Entropie maximale

FIGURE 5.2 – Corrélation entre l'entropie des distributions génératrices et la taille des boules de la *PGD*

On obtient alors que l'*adversarial training* permet effectivement d'éloigner les concepts mais aussi diminue l'entropie des distributions génératrices, c'est-à-dire que la masse de probabilité est plus concentrée. L'entropie d'une distribution nous informe sur la répartition de la masse de probabilité dans l'espace, une entropie faible traduit le fait que tous les éléments probables sont rassemblés au même endroit, tandis qu'une entropie maximale serait une distribution équiprobable sur tout l'espace. Ainsi, si la distribution associée à une classe a une très faible entropie, cela signifie que les éléments de cette classe sont bien localisés dans l'espace et donc que le modèle a été capable de les rassembler et de les séparer du reste. On peut interpréter ce phénomène comme une amélioration

de la netteté de ces concepts pour le modèle. On pourrait dire que le modèle a une idée "nette et précise" de ce qu'est un symbole représentant un 1.

5.4 Une question de distributions plus que de modèles

Lorsque l'on parle de la robustesse d'une fonction, on peut parler de sa robustesse intrinsèque, c'est-à-dire de la qualité de la généralisation que l'algorithme d'apprentissage a réussi à atteindre, ou on peut plus simplement s'intéresser au lien qui existe entre les résultats obtenus sur un jeu de test et les résultats que l'on espère obtenir en pratique. On veut ici savoir à quel point ce qu'on a pu observer dans le test d'usine nous renseigne sur le comportement face à la vie [**?**].

Dans ce cas-là, on ne s'inquiète pas de l'existence de fausses corrélations, on s'intéresse seulement de savoir si les corrélations que l'on a apprises sur notre jeu d'entrainement et testées sur notre jeu de test auront des effets similaires face à la vraie vie. On peut tout de suite noter que c'est une conception plus faible de la robustesse que la précédente. En effet, si seules de vraies corrélations sont apprises, alors un bon comportement sur le test se transcrira naturellement en pratique puisque la généralisation est correcte. Les vraies corrélations seront présentes quelle que soit la distribution des données auxquelles on confronte un modèle appris.

Il peut exister des corrélations fausses, au sens qu'elles ne font pas sens, ou dont on sait qu'elles n'ont rien à voir avec ce que l'on essaie de prédire en général, mais qui fonctionnent pour les distributions qui nous sont présentées. On peut imaginer une fonction entraînée à détecter des cancers dans des radiographies qui aurait été entraînée uniquement sur des radios issues d'une certaine machine ou d'une certaine marque de machines, de telle sorte qu'elle a tendance à classifier comme cancéreuse ou anormale toute radio issue d'une autre machine. Une telle corrélation apprise par l'algorithme n'a évidemment aucun sens mais elle ne pose pas nécessairement un problème – tant que l'on en connait l'existence. En effet, si en pratique on se cantonne à utiliser ce modèle avec des radios de la machine usuelle, tout ira bien, et peu importe que la généralisation soit fausse en général, elle est suffisante pour la distribution courante des données. C'est un bon exemple de l'apprentissage automatique qui ne fait que prédire sans expliquer ou comprendre.

Ce point de vue pose d'importants problèmes en termes de confiance que l'on peut accorder aux décisions prises par des algorithmes appris. En effet, si l'on s'en tient à cette définition de la robustesse – qui contrairement à la précédente a le mérite d'être plus facilement utilisable en pratique – on admet utiliser un outil qui fait potentiellement des choses qui n'ont pas de sens mais qui fonctionne statistiquement suffisamment bien.

5.5 Un problème ouvert

L'évaluation des modèles appris est un domaine de recherche ouvert et il n'existe pas de solution standardisable permettant de les certificer efficacement. Les deux solutions que nous avons décrit ici ne sont pas viables en pratique car, soit elles sont difficilement calculables, soit elles ne capturent pas suffisamment d'information concernant la robustesse effective de la fonction apprise.

Nous allons donc maintenant construire un cadre d'emploi des algorithmes d'apprentissage qui ne nécessiterait pas que l'on soit capable d'évaluer efficacement la robustesse de l'algorithme appris car on pourrait se reposer sur une vérification tierce du résultat.

Chapitre 6

De l'emploi sûr de l'AI

Nous avons vu jusqu'ici de multiples exemples d'applications critiques pour lesquelles les méthodes d'apprentissage ne sont pas prêtes, telles que la reconnaissance faciale (Section 2.2.1) ou la conduite autonome (Section 2.2.2) et des raisons expliquant les faiblesses et risques intrinsèques aux algorithmes d'apprentissage (Section 4.1). Nous allons maintenant voir que malgré ces défauts et risques, les algorithmes d'apprentissage ont de nombreuses applications robustes pour lesquelles on peut les employer tout en minimisant les risques. En fait, nous allons voir qu'il existe des classes de problèmes pour lesquelles on peut garantir avec grande probabilité la correction d'une solution produite par un algorithme appris.

En particulier, nous allons voir que si les algorithmes d'apprentissage sont peu fiables en pire cas et inadaptés pour nombre d'applications critiques, ils fournissent d'excellentes heuristiques et des outils pour la prise de décision. On parle ici d'heuristiques au sens algorithmique du terme, c'est-à-dire une méthode ou une procédure de décision pour laquelle on n'a aucune garantie : une procédure arbitraire. Partout où, pour résoudre un problème d'exploration de solutions on emploie des heuristiques "expertes" *i.e* construites manuellement par des experts du domaine du problème en question [1], on peut envisager de la remplacer par un algorithme appris sans risques supplémentaires. Si de plus, le problème que l'on cherche à résoudre appartient à certaines classes de problèmes, on peut vérifier *a posteri* avec forte probabilité que la solution est correcte ou non. Nous allons construire un cadre d'emploi robuste des algorithmes d'apprentissage à partir des

1. La fonction d'évaluation d'une position d'échecs de *DeepBlue* par exemple.

classes de complexité $\mathcal{NP}$ et IP issue de l'informatique fondamentale [**?**, **?**], avant de proposer une relaxation humaine de ces cadres théoriques.

6.1 Exploration et vérification

Les algorithmes d'apprentissage se sont révélés excellents pour guider des explorations d'arbres en tant qu'heuristiques se substituant aux règles humaines arbitraires, que ce soit avec l'algorithme *AlphaZero*[**?**], pour des algorithmes d'optimisation combinatoires [**?**] ou du $SAT-$solving [**?**].

Les problèmes de la classe $\mathcal{NP}$ [**?**] sont d'excellents exemples de cas où l'emploi d'algorithmes d'apprentissage est non seulement extrêmement utile, mais pour lesquels leur emploi peut être rendu sûr. Ces problèmes se caractérisent par le fait que si l'on dispose d'une solution candidate et d'un certificat à un problème de décision, on est capable de vérifier rapidement cette solution. Dès lors, on peut employer un algorithme appris comme heuristique pour guider une recherche de solutions candidates et vérifier si les solutions proposées conviennent. Nous verrons une définition formelle de cela (Section6.2).

Le problème principal de la classe $\mathcal{NP}$ est le problème SAT [**?**] :

Définition 1 (Problème SAT)**.** Ce problème s'intéresse à la satisfiabilité d'une formule logique propositionnelle (*ie* une combinaison de variables du premier ordre, de négations, de conjonctions et de disjonctions). C'est-à-dire répondre à la question : existe-il une assignation des variables qui rend la formule vraie ?

Soient $x = (x_1, \cdots, x_n) \in \mathcal{X}$ des variables et ϕ une formule de la logique propositionnelle.

Existe-il $\sigma : \mathcal{X} \longrightarrow \{\top, \bot\}$ telle que $\phi(\sigma(x_1) \cdots \sigma(x_n))$ s'évalue à $\top$?

Produire une solution pour répondre à cette question est particulièrement difficile. En revanche, il est trivial de vérifier si une affectation proposée convient ou non.

Pour résoudre les problèmes de cette classe, il n'existe pas, *a priori*[2], de méthode polynomiale en temps, et il s'agit la plupart du temps d'explorer toutes les solutions possibles pour en trouver une qui convient en temps exponentiel. On peut, pour ce genre de situation, employer des algorithmes d'apprentissage pour aider à guider l'exploration de ces

2. Sauf si $P = \mathcal{NP}$.

solutions sans autres risques que de chercher en premier au mauvais endroit et perdre du temps. En effet, lorsque l'on pense avoir trouvé une solution, par définition de $\mathcal{NP}$ il est relativement facile de vérifier si c'est effectivement une solution.

De façon générale, on peut envisager de remplacer n'importe quelle heuristique par un modèle appris pour guider les explorations ou les choix, et ces méthodes sont sans risques tant qu'il existe une façon simple de vérifier la solution obtenue. Il est par exemple envisageable d'utiliser des algorithmes de génération de texte pour écrire des preuves en langage formel [**?**] (tels que COQ [**?**] ou lean). En effet, les assistants à la preuve se chargent de parcourir la preuve qui leur est fournie et de vérifier si chaque étape est correcte et si à la fin de la preuve tous les *goals* ont été refermés. Dès lors, peu importe la façon dont la preuve a été obtenue si elle passe la vérification par l'assistant. L'intérêt des algorithmes appris statistiquement est qu'ils permettent une exploration plus intelligente et don plus rapide que simplement attendre qu'un chimpanzé écrive quelque chose de correct.

Un algorithme d'apprentissage dans ces situations va en fait s'adapter à la distribution des données qui lui sont présentées et fournir une heuristique qui est bonne dans ces conditions. En effet, même si ces problèmes sont $\mathcal{NP}-$durs dans le cas général, il est peut-être possible de faire mieux sur des classes d'instances particulières de ces problèmes, et les algorithmes d'apprentissages peuvent tirer avantage de cela. Dans cette idée, puisque SAT est $\mathcal{NP}-$complet – c'est-à-dire que l'on peut y ramener n'importe quel problème de $\mathcal{NP}$ – on peut envisager d'apprendre des heuristiques de SAT-solver différentes pour chaque classe de problème que l'on souhaite résoudre *via* une transformation en un problème SAT. En effet, pour un problème A donné, l'ensemble des traductions de ses instance en des instances de problème SAT suivra une distribution particulière associée à A. Dès lors, on peut envisager d'entraîner une heuristique spécialisée de SAT-solver pour A. De cette façon, on conserve une unique architecture de résolution, mais le guidage de l'exploration des solutions est conditionné au type de problème. Et dans tous les cas, on peut vérifier si la solution proposée est correcte.

Cette situation est assez commune pour ne pas réduire l'ensemble des problèmes ML-robuste à quelque chose d'anecdotique. En fait, on peut même montrer que l'ensemble de la classe $PSPACE$ – les problèmes solubles en espace polynomial par une machine de Turing – est une classe de problèmes robustes (Sections 6.2) comme nous allons le voir dans la section suivante.

6.2 Classes de problèmes ML-robustes

Nous allons ici nous éloigner un peu des questions d'intelligence artificielle pour définir des concepts issus de la théorie de la complexité en informatique fondamentale, qui décrivent, en fait, un cadre remarquablement adapté aux questions de robustesse des algorithmes d'apprentissage automatique. On a vu jusqu'ici que les algorithmes appris étaient particulièrement efficaces pour fournir de bonnes solutions *en moyenne* aux problèmes présentés, mais qu'on ne pouvait pas leur faire confiance aveuglément. Il se trouve qu'il existe des classes de complexité décrivant exactement ces situations, dont on peut s'inspirer pour dresser les contours des problèmes que l'on peut envisager de résoudre sans risques en employant des algorithmes appris. Nous allons en particulier voir que les classe $\mathcal{NP}$ et IP (*Interactive Proof system*), que nous définirons dans la suite de cette section, modélisent une situation où l'on dispose d'un algorithme non-fiable et d'un système de vérification des solutions candidates proposées. Dans la section suivante (Section 6.3), nous décrirons une adaptation aux systèmes sociaux techniques de la classe IP.

6.2.1 Une première approche avec la classe $\mathcal{NP}$

Les classes de problèmes P et $\mathcal{NP}$ sont probablement les classes de complexité les plus connues, notamment à cause du problème du millénaire associé [**?**] [3] mais nous allons ici nous intéresser à la caractérisation par certificat de la classe $\mathcal{NP}$ [**?**].

De façon informelle, les problèmes de la classe $\mathcal{NP}$ sont des problèmes pour lesquels il est difficile de trouver une solution, plus précisément, il n'existe *a priori* pas d'algorithme polynomial pour trouver une solution. En revanche, il existe un algorithme – qui s'exécute en temps polynomial – permettant de vérifier si une solution est correcte, pour peu que l'on dispose d'un certificat (de taille polynomiale) de cette solution candidate.

Définitions

Définition 2 ($\mathcal{P}$: Déterministe polynomial)**.** La classe de problèmes déterministe polynomiale, notée $\mathcal{P}$, correspond à l'ensemble des problèmes solubles par un algorithme

3. P versus $\mathcal{NP}$

(*i.e.* une machine de Turing déterministe) en temps polynomial.

Définition 3 ($\mathcal{NP}$: Non Déterministe polynomial)**.** Il existe deux définitions de cette classe, nous nous contenterons de la caractérisation par certificat. La classe de problèmes non déterministe polynomiale, notée $\mathcal{NP}$, correspond à l'ensemble des problèmes dont vérifier si une solution candidate – un certificat – est effectivement une solution ou non s'effectue en temps polynomial par un algorithme [4].

Remarque 1. Cette caractérisation est équivalente à la définition par les machines de Turing non déterministes de la classe $\mathcal{NP}$: la classe $\mathcal{NP}$ est l'ensemble des problèmes solubles en temps polynomial sur une machine de Turing non-déterministe [**?**].

Exemple 1. Soit $G = (V, E)$ un graphe, peut-on le colorier avec moins de k couleurs ? Trouver tel coloriage est difficile, mais si on en a un, il est facile de vérifier qu'il comporte moins de k couleurs et que 2 sommets adjacents n'ont jamais la même couleur.

Exemple 2. Soient W, V, $v_1 \cdots v_n$, $w_1 \cdots w_n$ des entiers représentant un sac à dos pouvant contenir au maximum W kilogrammes, et des objets de valeurs v_i et de poids w_i. Puis-je remplir mon sac à dos de sorte que la somme des valeurs des objets dedans soit plus grande que V et que la somme de leur poids plus petit que W ? Il est difficile de remplir le sac, en revanche, si on dispose d'un sac rempli, on peut facilement vérifier s'il respecte les deux conditions.

Lien avec le *machine learning*

En somme, pour trouver une solution, il faut explorer une nombre exponentiel d'options, mais si on a une solution candidate, on peut vérifier si elle est correcte facilement. On voit déjà pointer l'intérêt du *machine learning* dans ce cadre : on peut se permettre de guider l'exploration ou la génération de solutions candidates avec un algorithme appris – potentiellement peu sûr – puisque de toutes manières, on vérifiera la correction de la solution.

6.2.2 Les systèmes de preuves interactives : la classe IP

Les systèmes de preuves interactives [**?**] correspondent à des machines abstraites qui décrivent un modèle de calculs dans lequel un prouveur et un vérifieur vont échanger.

4. C'est un résultat en fait obtenu à partir du théorème PCP issu des systèmes de preuves interactifs que l'on décrit dans la section 6.2.2.

On suppose que le prouveur est un oracle à la puissance de calculs infinie, mais auquel on ne peut faire confiance, et que le vérifieur, lui, tourne en temps borné, mais que l'on peut faire confiance à ses résultats. Dans ce modèle, le prouveur va produire des solutions candidates et leurs certificats, tandis que le vérifieur doit vérifier que ces solutions sont correctes. C'est une généralisation de $\mathcal{NP}$: dans cette dernière il n'y avait qu'un échange entre le prouveur et le vérifieur, le certificat et la réponse. Ici, le prouveur et le vérifieur peuvent échanger plus longuement, le vérifieur pouvant notamment poser des questions supplémentaires au prouveur. Ces systèmes de preuves doivent vérifier deux propriétés : une première dite de *completeness* ou de complétude en français, et une seconde dite de *soundness* ou de solidité [5].

Proposition 1 (Completeness)**.** *Si une assertion est vraie, alors le vérifieur honnête (vérifiant les spécifications du protocole) peut être convaincu de la véracité de cette assertion par un prouveur quelconque (auquel on peut faire confiance ou non).*

Proposition 2 (Soundness)**.** *Si une assertion est fausse, aucun proveur, suivant ou non les spécifications du protocole, ne peut convaincre un vérifieur honnête de la véracité de celle-ci, sauf avec une probabilité faible.*

Ce cadre traduit en fait une situation où l'on dispose d'un outil boîte noire – le prouveur – produisant des réponses ou des solutions, potentiellement mauvaises ou absurdes, et où l'on est par ailleurs capable de vérifier rapidement l'intérêt ou la correction de ces solutions. On peut imaginer des modèles où le prouveur et le vérifieur échangent un unique message avant que le vérifieur décide si l'assertion est correcte : cela définit la classe de complexité MA [**?**] qui correspond peu ou prou à la classe $\mathcal{NP}$ définie par certificats. On peut aussi envisager un cadre dans lequel le prouveur et le vérifieur peuvent échanger un nombre polynomial, en la taille de l'assertion, messages. On obtient alors la classe IP.

La classe IP

On se donne un langage et on veut vérifier si une chaîne s appartient à ce langage. On suppose que l'on dispose de P un prouveur oracle et de V un vérifieur probabiliste opérant en temps polynomial, et qui a accès à un nombre polynomial en la taille de s de bits aléatoires. Ces deux machines vont échanger un nombre de messages bornés par

5. Nous nous en tiendrons à la terminologie anglophone.

$p(n)$ où p est un polynôme et après l'échange de l'ensemble de ces messages, le vérifieur doit décider si s appartient au langage avec une probabilité d'erreur d'au plus 1/3.

Définition 4 (*Interactive Prover*)**.** Un langage L appartient à la classe IP, c'est-à-dire L est décidable par un protocole du type décrit ci-dessus, s'il existe un vérifieur V et un prouveur P tels que pour tout prouveur Q et mot w :

$$w \in L \implies \mathbb{P}[(V,P) \text{ accepts } w] \geqslant \frac{2}{3}$$
$$w \notin L \implies \mathbb{P}[(V,Q) \text{ accepts } w] \leqslant \frac{1}{3}$$

Intérêt On sait depuis 1992 grâce à Adi Shamir que la classe IP est en fait égale à la classe $PSPACE$ [**?**] (Figure 6.1), c'est-à-dire la classe des problèmes solubles par des machines de Turing qui ont une complexité en espace polynomial. Cette classe est l'une des plus importantes et contient en réalité l'immense majorité des problèmes, correctement posés, que l'on peut vouloir traiter. C'est-à-dire que la classe des problèmes pour lesquels on peut espérer employer des algorithmes d'apprentissage – de façon sûre – est loin d'être anecdotique ou réduite au point d'être inutilisable en pratique.

Réduction de problèmes

Afin de pouvoir envisager d'employer des algorithmes d'apprentissage de façon sûre pour résoudre tout problème de $PSPACE$, il faut pouvoir ramener tout problème de $PSPACE$ à une formulation dans le cadre IP. On utilise pour cela des réductions au sens de Karp.

Définition 5 (Réduction de Karp)**.** Soient A, B deux ensembles de problèmes. Une réduction de A à B est une fonction $f : \begin{cases} A \longrightarrow B \\ \mathcal{I}_A \mapsto f(\mathcal{I}_A) = \mathcal{I}_B \end{cases}$ Où $\mathcal{I}_A$ et $\mathcal{I}_B$ sont des instances de A et B.

Cette fonction vérifiant pour tout $\mathcal{I}_A$, "$f(\mathcal{I}_A)$ a une solution équivaut à $\mathcal{I}_A$ a une solution".

Si de plus f est calculable en temps polynomial par un algorithme (*ie* par une machine de Turing déterministe), on dit que f est une réduction polynomiale.

Définition 6 (Relation d'ordre de complexité)**.** Si une telle réduction polynomiale existe, on dit que B est plus difficile que A. Et on note $A \leqslant_P B$, signifiant : il existe une réduction

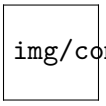

FIGURE 6.1 – Diagramme des classes de complexité.

polynomiale de A en B.

Définition 7 (Problèmes équivalents)**.** Si $A \leqslant_P B$ et $B \leqslant_P A$, alors ces deux problèmes sont équivalents. C'est effectivement une relation d'équivalence. On peut alors quotienter l'ensemble des problèmes de décision algorithmique par cette relation pour obtenir différentes classes d'équivalence de complexité.

Cela revient à dire que l'on peut transformer (rapidement) tout problème de type A en un problème de type B de sorte que le problème transformé ait une solution si et seulement si le problème d'origine a une solution. Ainsi il "suffit" de savoir résoudre les problèmes de B pour résoudre les problèmes de A.

Définition 8 (Complétude d'un problème)**.** Si on a une classe de complexité C, on dit qu'un problème P est C-complet, si tout problème de la classe C peut être ramené par réduction de Karp au problème P. C'est le problème de difficulté maximale dans la classe en question. En particulier pour la classe $\mathcal{NP}$, SAT est un problème complet, $\mathcal{NP}$-complet en vertu du théorème de Cook [**?**].

En l'occurrence, la preuve de $IP = PSPACE$ donnée par Adi Shamir donne une construction explicite d'un protocole IP pour le problème $TQBF$, problème SAT avec des quantificateurs alternés [**?**], qui est connu pour être $PSPACE$-complet. Ainsi, on peu théoriquement ramener n'importe quel problème de $PSPACE$ à un protocole IP avec un prouveur qui utiliserait alors un algorithme d'apprentissage non sûr et un vérifieur sûr [6].

6. Il serait certainement possible d'écrire un compilateur produisant un environnement d'apprentissage par renforcement en utilisant cette traduction pour entraîner des algorithmes d'apprentissage pour n'importe quel problème de $PSPACE$, je garde cette idée pour un futur travail ne pouvant l'intégrer ici par manque de temps.

6.3 *Motivation by design* : la classe IP avec un vérifieur humain

Nous avons dans la section précédente introduit les classes de problèmes abstraites pouvant être transcrites dans le cadre de système de preuves interactives, c'est-à-dire dans un cadre propice à l'utilisation sûre d'algorithmes appris. On peut maintenant s'inspirer de ce cadre pour construire une version relaxée, humaine. Le cadre formel décrit une situation où l'on a un oracle auquel on ne peut faire confiance et un vérifieur qui lui est de confiance et tourne rapidement. Usuellement, on considère que l'humain est l'élément faillible – l'oracle – tandis que l'assistant à la preuve est le vérifieur honnête. Dans le cadre du *machine learning*, nous allons voir qu'il serait intéressant d'employer l'humain comme vérifieur de confiance et d'imposer la génération de certificat aux algorithmes d'apprentissage. On compte ici sur les avancées en *explainable AI* pour produire, en plus d'une décision, des éléments de justification de cette décision. Ce cadre ne permettra bien-sûr pas les décisions totalement automatiques ou rapides – telles que celles nécessaires en conduite automatique – mais il permet de définir un cadre de construction de systèmes d'aide à la décision, et plus précisément la construction d'un cadre socio-technique de prises de décisions.

6.3.1 De l'analytique à la dialectique

Les algorithmes d'apprentissage ont longtemps été – et sont encore souvent – vus comme des outils purement analytiques qui permettraient de répondre à toutes les questions ou de modéliser tous les comportements pour peu que le modèle soit assez expressif (gros) et que l'on dispose d'assez de données. Le problème de cette approche purement analytique est qu'elle est vouée à l'échec, ou en tous cas bien trop sensible aux biais et failles : trop peu robuste [**?**]. Dans un cadre similaire aux preuves interactives, nous pourrions alors envisager en lieu et place d'une décision unique directe donnée par l'algorithme, un échange entre l'algorithme appris (l'oracle) et l'humain pour déterminer si la décision est correcte.

Il s'agirait alors d'imposer aux algorithme appris de ne pas seulement fournir une décision brute, mais de produire un certificat, voire de répondre aux questions du vérifieur humain. On peut imaginer tout d'abord que l'algorithme d'apprentissage produise des éléments similaires de son jeu de données comme proposé dans l'article *This look like*

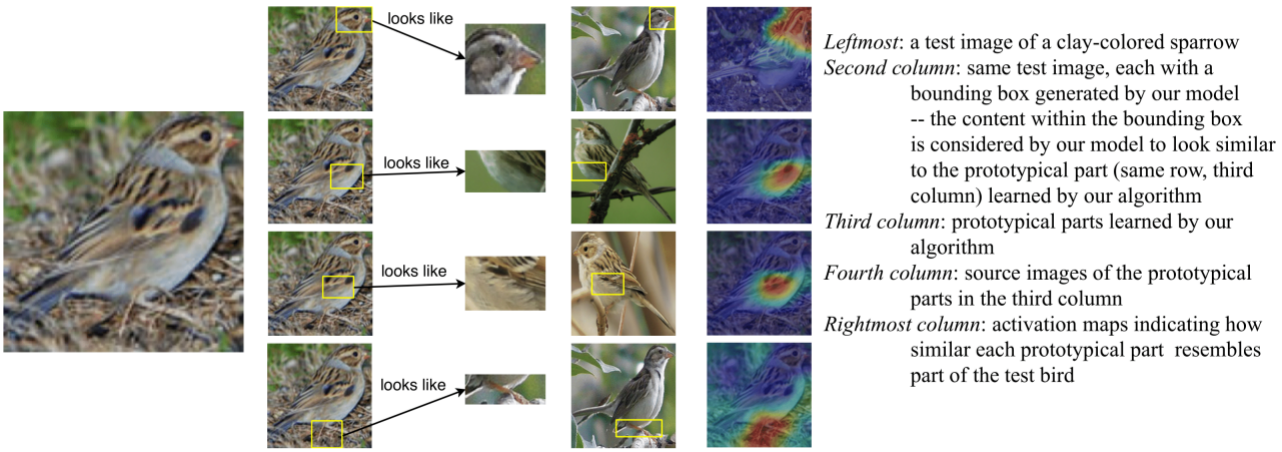

FIGURE 6.2 – Exemple de l'article *This look like that* [14], où pour produire une classification d'un oiseau présenté à gauche, l'algorithme produit des éléments de ressemblance entre cet oiseau et des oiseaux de la même espèce de son jeu d'entraînement.

that [14]. On se place alors dans un cadre où le prouveur – l'algorithme isssu de l'apprentissage – produit des motivations pour soutenir sa décision, pour convaincre le vérifieur que la décision est correcte et fondée sur des corrélations sensées. Dans l'exemple Figure 6.2, l'algorithme a été entraîné à produire une réponse mais aussi à fournir dans le même temps des arguments de ressemblance avec des éléments connus pour soutenir la proposition de décision. Si dans les ressemblances proposées on observait des absurdités, on serait alors en mesure de rejeter la décision. De plus, on se place dans une situation d'erreur humaine acceptable : aucune procédure de décision n'est parfaite et des erreurs sont admissibles. Elles sont admissibles si les décisions étaient correctement justifiées. Si une erreur est commise mais que le décisionnaire, en présentant les éléments dont il disposait au moment de sa décision, est capable de convaincre que cette décision semblait justifiée, alors c'est un accident qui rentre dans l'aléa et le risque qui existe dans toute décision. Dès lors, le certificat produit par l'algorithme tient lieu d'éléments de preuve supplémentaires pour justifier la décision.

On emploierait ici des techniques issues de l'explicabilité ou de l'interprétabilité des algorithmes appris pour construire ces certificats et on décide que l'explicabilité est une caractéristique clef des algorithmes appris. On impose l'explicabilité de toutes les décisions prises, c'est-à-dire la production d'un certificat dialectique en plus de la décision stricte. Ce cadre fonctionne parfaitement dans le cas où l'on a une bonne compréhension

du sujet que l'on traite, mais pas seulement. Dans le cas où le phénomène n'est pas totalement compris, l'opérateur humain pourrait malgré tout être capable de détecter les cas pathologiques où l'algorithme d'apprentissage est perdu et fait des propositions absurdes. Et ce sont biens ces cas qui nous intéressent : les algorithmes appris fonctionnent plutôt bien en moyenne et de temps en temps échouent lamentablement sur ces cas pathologiques.

On peut aussi imaginer des questions plus analytiques de la part du vérifieur, par exemple il pourrait demander si dans un voisinage raisonnable autours du point présenté, il existe des cas où la décision est totalement différente. On remet alors à l'ordre du jour les questions de régularité de la fonction apprise, mais au niveau d'une instance et non au niveau du modèle. Si l'algorithme garantit que, même avec des petites variations, il produit toujours la même décision, on a déjà une information complémentaire utile pour décider de conserver ou non sa décision.

6.3.2 De la motivation juridique au certificat algorithmique

En droit, toutes[7] les décisions de justice doivent être motivées. Le juge doit motiver sa décision ou il encourt la cassation de celle-ci [18]. En effet, même si le juge est le seul décisionnaire et que les délibérations ou ses réflexions personnelles sont protégées par le secret du délibéré, il doit malgré tout étayer en droit les motifs. L'objectif premier de cette disposition est justement de permettre aux parties affectées par la décision de la comprendre et de potentiellement mieux l'accepter. Mais ce n'est pas le seul bénéfice de la motivation des décisions. La motivation des décisions permet aussi la contestation de la décision : il est particulièrement difficile d'attaquer une décision non motivée car on n'a aucune prise dessus, tandis que si une motivation est fournie on peut se servir de ces éléments pour attaquer la décision. Il s'agit alors de construire un mécanisme socio-technique similaire pour l'emploi des algorithmes appris.

Si l'opérateur possède assez de connaissances sur le phénomène sous-jacent, il sera capable d'évaluer les arguments proposés par l'algorithme. Par exemple, en médecine, il parait raisonnable de penser qu'un médecin sera capable d'interpréter et d'évaluer la correction des éléments apportés par l'algorithme. Néanmoins, une connaissance précise du phénomène n'est pas absolument nécessaire pour pouvoir filtrer les réponses absurdes. Un point particulier à noter est que les algorithmes d'apprentissage fonctionnent géné-

7. En fait il existe de rares exceptions, mais ce n'est pas la question.

ralement bien : en moyenne, les résultats proposés sont bons et les arguments fournis relativement cohérents. Le problème vient de cas pathologiques induit par des *adversarial attacks* ou situations rares absurdes. Ces cas, même sans connaissances précises du phénomène sous-jacent, peuvent être discriminés. Si, dans une tâche de classification d'animaux, l'opérateur se rend compte que la zone de l'image sur laquelle l'algorithme s'appuie est vide, il sera tout à fait capable de remarquer cette incohérence. D'ailleurs peut-être qu'en réalité l'algorithme proposait une solution correcte, et une justification fausse, mais dans ce cas on est malgré tout content de filtrer cette décision.

6.3.3 De la remise en question des décisions

L'intérêt principal de cette construction socio-technique (du certificat algorithmique à l'humain vérifieur), est de permettre à l'opérateur humain en charge de contrôler l'algorithme de pouvoir s'y opposer. Tout d'abord, on est forcé de constater que, vu les questions de responsabilité et risques juridiques en jeu en cas d'erreur d'appréciation, il serait *a priori* difficile pour un opérateur humain d'aller à l'encontre d'un algorithme de décision ne produisant qu'une décision brute. En effet, comment expliquer, en cas d'erreur, que l'on a pas suivi la décision mécanique, souvent perçue comme particulièrement fiable. En fait, dans ce cadre, l'opérateur humain n'a jamais intérêt à aller à l'encontre de la décision automatique. Si la décision est correcte et que l'humain ne s'y oppose pas, il n'y a aucun problème. Si la décision de l'algorithme est fausse et que l'humain ne s'y oppose pas, il est difficile de lui reproche quoi que ce soit : il s'est fié à l'outil *a priori* fiable qu'on lui a fourni. En revanche, s'il s'oppose à l'algorithme dans un cas où sa décision était correcte, il sera mis en défaut pour être intervenu sur un outil encore une fois généralement jugé favorablement.

En intégrant la notion de certificat ou de motivation et une interaction entre l'opérateur et l'algorithme, l'opérateur est responsable de vérifier que les motifs proposés sont cohérents et raisonnables. S'ils ne le sont pas, l'opérateur peut s'opposer à la décision proposée et s'en défendre si jamais elle était correcte. Il sera capable de démontrer que les motifs proposés étaient absurdes et que même si la décision s'est avérée correcte par la suite, cela n'aurait été qu'un coup de chance.

Chapitre 7

Conclusions

Nous avons vu que l'emploi de méthodes dites d'intelligence artificielle pour des tâches critiques pose déjà de nombreux problèmes de robustesse et de fiabilité (Sections 2.2.1, 2.2.2 et 2.2.3). En particulier, les algorithmes appris se révèlent particulièrement sensibles aux *adversarial attacks*, mais aussi simplement aux variations dans les distributions de données qui leur sont présentés. Les *distribution shifts* montrent à quel point il est difficile pour ces méthodes de produire des généralisation raisonnables, fiables, et donc utilisables en pratique pour des emplois critiques. Il est plus que nécessaire, lorsque l'on décide d'employer ces technologies, de garder à l'esprit qu'on ne peut accorder une grande confiance dans les algorithmes appris. Cet exercice est d'autant plus difficile qu'il n'existe pas de méthode efficace pour évaluer l'efficacité et la fiabilité (Section 4.1) d'un algorithme appris particulier. Dès lors, les seules options de régulation et d'encadrement envisageables se cantonnent à établir des bonnes pratiques de conception.

La gouvernance et la régulation de ces méthodes sont donc pour l'instant principalement gérées par les industriels eux-mêmes (Section 2.3.2) avec des limites évidentes. Sans organe de régulation et de standardisation des emplois de l'intelligence artificielle, les industriels sont libres de bâtir leurs propres limites. On peut notamment penser à la tentative d'*Apple* de déployer un algorithme dit de *perceptual hash* [**?**] pour permettre la détection d'image pédopornographiques [**?**]. L'entreprise semble avoir décidé d'ignorer le manque de fiabilité notable de ces algorithmes appris, et ce malgré de nombreuses alertes issus de chercheurs en sécurité et en machine learning, ainsi que des associations de consommateurs et de défense de la vie privée [**?**]. Dans ce cas précis, le manque de

fiabilité des algorithmes appris peut conduire au signalement de faux positifs aux forces de l'ordre.

Nous avons de plus vu que la principale tentative de régulation étatique de l'intelligence artificielle émane de l'Union Européenne au travers de sa proposition de régulation [**?**] de l'intelligence artificielle, mais que celle-ci ne propose aucune réelle révolution. En effet, elle se contente de produire un guide de bonnes pratiques de conception et de sécurités à mettre en œuvre pour différents niveaux de criticité. Il s'avère alors que ces sécurités relèvent simplement du bon sens et d'une conservation des archives des décisions algorithmiques pour permettre une revue *a posteriori* des décisions prises.

Une fois les décors industriel et régulatoire posés, nous avons développé une réflexion informelle théorique sur les sources des problèmes de robustesse des algorithmes d'apprentissage (Chapitre 4). Nous avons notamment vu que, si la *fairness* se concentrait sur l'étude des biais socialement significatifs, c'est-à-dire des corrélations avec des caractères sociaux qui nous importent particulièrement (genre, race etc...), la robustesse s'intéressait aussi aux corrélations qui ne sont *a priori* pas sensées, c'est-à-dire à toutes les corrélations, intelligibles ou non, qui guident les décisions de l'algorithme appris. En analysant ces corrélations, on espère pouvoir évaluer la qualité de la généralisation produite par l'apprentissage. On peut notamment identifier si l'algorithme a appris à classifier des objets à partir d'éléments intrinsèques à ces objets, ou plutôt à partir du contexte dans lesquels ils sont présentés. Un algorithme se reposant trop sur le contexte échouerait certainement à reconnaître une vache flottant dans l'espace. Ensuite, nous avons dérivé des notions formelles de robustesse qui s'attachent à capturer les affres identifiés précédemment et à mesurer leur impact sur les décisions d'un algorithme appris.

Finalement, même si nous avons vu que l'établissement de notions raisonnables de robustesse et que l'évaluation des modèles appris étaient des problèmes de recherche fondamentale ouverts, nous proposons un cadre théorique et pratique d'emploi sûr des algorithmes. A partir de la théorie de la complexité, nous avons identifié des classes de problèmes qui peuvent être mis dans une forme permettant la vérification efficace des solutions produites par un algorithme d'apprentissage automatique (Chapitre 5). Dans ce cadre, on considère que les algorithmes appris agissent comme des oracles puissants auxquels on ne peut faire confiance, mais que l'on peut confronter à des vérifieurs efficaces. A partir de cette construction théorique, nous proposons une relaxation humaine du vérifieur afin de construire un système socio-technique de décision se fondant sur un opérateur humain expert, chargé d'interpréter la décision et la motivation d'un algorithme appris.

Annexe A

Bibliographie

[1] Première interpellation grâce au nouveau système de reconnaissance faciale. https://www.lefigaro.fr/actualite-france/2014/04/09/01016-20140409ARTFIG00372-une-premiere-interpellation-grace-a-un-nouveau-systeme-de-reconnaissance-faciale.php, 2014.

[2] Boycott collaboration with police. https://www.math-boycotts-police.net/, 2020.

[3] Computer vision dazzle camouflage. https://cvdazzle.com/, 2020.

[4] Michael A. Alcorn, Qi Li, Zhitao Gong, Chengfei Wang, Long Mai, Wei-Shinn Ku, and Anh Nguyen. Strike (with) a pose : Neural networks are easily fooled by strange poses of familiar objects, 2018.

[5] Mitra Amini and Swapnil Morande. Application of artificial intelligence (a.i.) in organ transplant : A utility in donor exchange program. 02 2020.

[6] Arun Ross Antitza Dantcheva and Cunjian Chen. Makeup challenges automated face recognition systems. https://spie.org/news/4795-makeup-challenges-automated-face-recognition-systems?SSO=1.

[7] Kellyn F Arnold, Vinny Davies, Marc de Kamps, Peter W G Tennant, John Mbotwa, and Mark S Gilthorpe. Reflection on modern methods : generalized linear models for prognosis and intervention—theory, practice and implications for machine learning. *International Journal of Epidemiology*, 49(6) :2074–2082, 05 2020.

[8] Kai Arulkumaran, Antoine Cully, and Julian Togelius. Alphastar. *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, Jul 2019.

[9] Victoria A. Banks and Neville A. Stanton. Analysis of driver roles : modelling the changing role of the driver in automated driving systems using EAST. *Theoretical Issues in Ergonomics Science*, 20(3) :284–300, January 2019.

[10] Martina F. Baumann, Claudia Brändle, Christopher Coenen, and Silke Zimmer-Merkle. Taking responsibility : A responsible research and innovation (rri) perspective on insurance issues of semi-autonomous driving. *Transportation Research Part A : Policy and Practice*, 124 :557–572, 2019.

[11] Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots : Can language models be too big ? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, page 610–623, New York, NY, USA, 2021. Association for Computing Machinery.

[12] Philippe Berry. Crash mortel : Mais comment les capteurs de tesla ont-ils pu rater un semi-remorque ? `https://www.20minutes.fr/high-tech/1879071-20160702-crash-mortel-comment-capteurs-tesla-pu-rater-semi-remorque`, 2016.

[13] Chih-Ling Chang, Jui-Lung Hung, Chin-Wei Tien, Chia-Wei Tien, and Sy-Yen Kuo. Evaluating robustness of ai models against adversarial attacks. In *Proceedings of the 1st ACM Workshop on Security and Privacy on Artificial Intelligence*, SPAI '20, page 47–54, New York, NY, USA, 2020. Association for Computing Machinery.

[14] Chaofan Chen, Oscar Li, Chaofan Tao, Alina Jade Barnett, Jonathan Su, and Cynthia Rudin. This looks like that : Deep learning for interpretable image recognition, 2019.

[15] Feixiong Cheng and Zhongming Zhao. Machine learning-based prediction of drug–drug interactions by integrating drug phenotypic, therapeutic, chemical, and genomic properties. *Journal of the American Medical Informatics Association*, 21(e2) :e278–e286, 03 2014.

[16] Louise Colcombet Cyril Simon. Attaque au couteau à paris : Azimov identifié grâce à la reconnaissance faciale. `https://www.leparisien.fr/faits-divers/reconnaissance-faciale-comment-a-ete-identifie-l-assaillant-khamzat-azimov-14-05-2018-7715343.php`, 2018.

[17] Braun Elisa. 40pas d'intelligence artificielle. `https://www.lefigaro.fr/secteur/high-tech/2019/03/06/32001-20190306ARTFIG00121-40-des-start-up-europeennes-d-intelligence-artificielle-n-utilisent-pas-d-intelligence-artificielle.php`, 2019.

[18] Cour européenne des droits de l'Homme. Affaire taxquet c. belgique. `http://hudoc.echr.coe.int/fre?i=001-90517`, 2009.

[19] Pratik Gajane and Mykola Pechenizkiy. On formalizing fairness in prediction with machine learning, 2017.

[20] Renaud Gardette. Un logiciel de reconnaissance faciale utilisé lors d'un procès à lyon fait débat. `https://france3-regions.francetvinfo.fr/auvergne-rhone-alpes/rhone/lyon/logiciel-reconnaissance-faciale-utilise-lors-proces-lyon-1724157.html`, 2019.

[21] Google. Google's ai principles. `https://ai.google/responsibilities/`, 2020.

[22] Gouvernement Français. Décret n 2012-652 du 4 mai 2012 relatif au traitement d'antécédents judiciaires. `https://www.legifrance.gouv.fr/loda/id/JORFTEXT000025803463/2020-12-30/`, 2012.

[23] Pam Greenberg. Spotlight | facial recognition gaining measured acceptance. `https://www.ncsl.org/research/telecommunications-and-information-technology/facial-recognition-gaining-measured-acceptance-magazine2020.aspx`, 2020.

[24] Rebecca Heilweil. Big tech companies back away from selling facial recognition to police. that's progress. `https://www.vox.com/recode/2020/6/10/21287194/amazon-microsoft-ibm-facial-recognition-moratorium-police`, 2020.

[25] Kashmir Hill. Wrongfully accused by an algorithm. `https://www.nytimes.com/2020/06/24/technology/facial-recognition-arrest.html`, 2020.

[26] Ahmed Hosny, Chintan Parmar, John Quackenbush, Lawrence H. Schwartz, and Hugo J. W. L. Aerts. Artificial intelligence in radiology. *Nature Reviews Cancer*, 18(8) :500–510, May 2018.

[27] Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. Adversarial attacks on neural network policies, 2017.

[28] Anne Hobson Ian Adams. Plus de voitures autonomes, c'est aussi moins d'organes pour les greffes. `http://www.slate.fr/story/133544/voitures-autonomes-organes-greffes`, 2017.

[29] IBM. Ibm's multidisciplinary, multidimensional approach to ai ethics. `https://www.ibm.com/artificial-intelligence/ethics`, 2020.

[30] iihs. Fatality facts 2019. `https://www.iihs.org/topics/fatality-statistics/detail/state-by-state`, 2019.

[31] Patrick Lin. *Why Ethics Matters for Autonomous Cars*, pages 69–85. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

[32] LQN. La reconnaissance faciale des manifestant est déjà autorisée. `https://www.laquadrature.net/2019/11/18/la-reconnaissance-faciale-des-manifestants-est-deja-autorisee/`, 2020.

[33] Kim Lyons. Timnit gebru's actual paper may explain why google ejected her. `https://www.theverge.com/2020/12/5/22155985/paper-timnit-gebru-fired-google-large-language-models-search-ai`, 2020.

[34] Jean-Luc Dugelay Marie Lena Eckert, Neslihan Kose. Facial cosmetics database and impact analysis on automatic face recognition. `https://www.eurecom.fr/en/publication/4044/download/mm-publi-4044.pdf`, 2013.

[35] B. Mehlig. Machine learning with neural networks, 2019.

[36] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning, 2019.

[37] Cade Metz. Who is making sure the a.i. machines aren't racist ? `https://www.nytimes.com/2021/03/15/technology/artificial-intelligence-google-bias.html`, 2021.

[38] Cade Metz and Kate Conger. Uber, after years of trying, is handing off its self-driving car project. `https://www.nytimes.com/2020/12/07/technology/uber-self-driving-car-project.html`, 2020.

[39] Richard Van Noorden. The ethical questions that haunt facial-recognition research. `https://www.nature.com/articles/d41586-020-03187-3`, 2020.

[40] Lara C. Pullen. Doctor AI. *American Journal of Transplantation*, 19(1) :1–2, December 2018.

[41] Maddie ROSE. Thousands of mathematicians join boycott against police collaboration. `https://shadowproof.com/2020/10/01/thousands-of-mathematicians-join-boycott-against-police-collaboration/`, 2020.

[42] sae. Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles. `https://www.sae.org/standards/content/j3016_201806/`, 2019.

[43] F. Shi, J. Wang, J. Shi, Z. Wu, Q. Wang, Z. Tang, K. He, Y. Shi, and D. Shen. Review of artificial intelligence techniques in imaging data acquisition, segmentation, and diagnosis for covid-19. *IEEE Reviews in Biomedical Engineering*, 14 :4–15, 2021.

[44] Jacob Snow. Amazon's face recognition falsely matched 28 members of congress with mugshots. `https://www.aclu.org/blog/privacy-technology/surveillance-technologies/amazons-face-recognition-falsely-matched-28`, 2018.

[45] Jean Souyris, Virginie Wiels, David Delmas, and Herve Delseny. *Formal Verification of Avionics Software Products*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[46] Eliza Strickland. How ibm watson overpromised and underdelivered on ai health care. `https://spectrum.ieee.org/biomedical/diagnostics/how-ibm-watson-overpromised-and-underdelivered-on-ai-health-care`, 2019.

[47] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks, 2014.

[48] Synced. Adversarial patch on hat fools sota facial recognition. `https://medium.com/syncedreview/adversarial-patch-on-hat-fools-sota-facial-recognition-82e8c4f83498`, 2019.

[49] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2016.

[50] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, 2013.

[51] U.S Department of transportation. Critical reasons for crashes investigated in the national motor vehicle crash causation survey. `https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812115`, 2015.

[52] Xiangjun Wang, Junxiao Song, Penghui Qi, Peng Peng, Zhenkun Tang, Wei Zhang, Weimin Li, Xiongjun Pi, Jujie He, Chao Gao, Haitao Long, and Quan Yuan. Scc : an efficient deep reinforcement learning agent mastering the game of starcraft ii, 2020.

[53] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial attacks on neural networks for graph data. 2018.