O'REILLY®

Machine Learning avec Scikit-Learn

Mise en œuvre et cas concrets

3e édition

Aurélien Géron

Traduit de l'anglais par Anne Bohy

DUNOD

Authorized French translation of material from the English edition of
Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow, 3E
ISBN 9781098125974

Conception de la couverture : Karen Montgomery
Illustratrice : Kate Dullea

NOUS NOUS ENGAGEONS EN FAVEUR DE L'ENVIRONNEMENT :

Nos livres sont imprimés sur des papiers certifiés pour réduire notre impact sur l'environnement.

Le format de nos ouvrages est pensé afin d'optimiser l'utilisation du papier.

Depuis plus de 30 ans, nous imprimons 70 % de nos livres en France et 25 % en Europe et nous mettons tout en œuvre pour augmenter cet engagement auprès des imprimeurs français.

Nous limitons l'utilisation du plastique sur nos ouvrages (film sur les couvertures et les livres).

© Dunod, 2017, 2019, 2023
11 rue Paul Bert, 92240 Malakoff
www.dunod.com
ISBN 978-2-10-084768-6

Table des matières

Avant-propos

La révolution du Machine Learning

Seuls ceux qui ont vécu dans une caverne depuis une quinzaine d'années ont pu ignorer l'incroyable révolution de l'apprentissage automatique, ou *Machine Learning* (ML). Il ne se passe plus une semaine sans qu'il ne fasse parler de lui : cela a commencé par de formidables progrès en reconnaissance d'images, puis en analyse de la voix, le programme Watson d'IBM est ensuite devenu champion du jeu de Jeopardy, on a vu les premières voitures autonomes de Google sillonner les routes, puis le programme AlphaGo de DeepMind a vaincu le champion du monde du jeu de go, le logiciel Libratus de l'université Carnegie Mellon a écrasé des champions de poker, des patients paralytiques ont pu contrôler le mouvement de leurs membres par la pensée, grâce à un programme qui avait appris à déchiffrer certaines de leurs ondes cérébrales… bref, les succès s'enchaînent et ne se ressemblent pas. Il y a une quinzaine d'années, de telles intelligences artificielles n'existaient que dans les romans de science-fiction.

Le Machine Learning en entreprise

Au-delà de ces exemples qui font la une des journaux, le Machine Learning envahit plus discrètement les systèmes informatiques de toutes les entreprises. D'ores et déjà, lorsque vous parcourez Internet ne serait-ce que quelques minutes, une horde de systèmes d'apprentissage automatique s'activent : certains analysent votre personnalité pour vous proposer les produits qui vous correspondent le mieux, d'autres sélectionnent les publicités qui attireront votre attention, et d'autres encore analysent votre comportement pour s'assurer que vous n'êtes pas un fraudeur. Mais le Machine Learning n'est pas réservé aux géants du web : qu'il s'agisse de prédire des séries temporelles (comme les cours de la Bourse), de détecter des anomalies de production, d'optimiser des centres d'appel, d'analyser les profils des clients, ou encore de classer automatiquement des documents, l'apprentissage automatique s'est révélé d'une grande utilité dans une multitude de domaines. Si votre entreprise n'utilise pas encore le Machine Learning, elle risque fort de se faire devancer rapidement par ses concurrents.

Data Scientist, une espèce rare

Le marché du Machine Learning croît si rapidement que le nombre d'experts en analyse de données (*data scientist*) a bien de la peine à suivre. Malgré un nombre d'étudiants en forte hausse en sciences des données (*data science*), il est aujourd'hui difficile pour une entreprise de trouver suffisamment de profils compétents. Face à cette pénurie d'expertise, la meilleure solution est probablement de former certains employés au Machine Learning. Les mieux placés pour cela sont naturellement les ingénieurs en informatique, car ils maîtrisent déjà la programmation et bien souvent aussi les bases mathématiques que requiert le Machine Learning. En outre, ils sont souvent très demandeurs, à la fois car ils savent que la compétence est rare, mais aussi et surtout car le sujet est absolument passionnant !

Objectif et approche

Ce livre a donc pour objectif de vous former au Machine Learning. Vous apprendrez les concepts fondamentaux et les outils nécessaires pour créer des systèmes capables d'apprendre à partir d'exemples, qu'il s'agisse de classer des images, d'estimer la valeur d'un bien, etc. Nous aborderons un grand nombre de techniques et d'algorithmes, depuis de simples systèmes linéaires, aux arbres de décision et aux forêts aléatoires en passant par les machines à vecteurs de support, et bien d'autres encore. Vous apprendrez aussi à utiliser Scikit-Learn[1], l'une des bibliothèques open source les plus simples et néanmoins les plus puissantes disponibles aujourd'hui, que vous pourrez directement utiliser dans vos systèmes en production.

Hormis le premier chapitre, qui offre une vision d'ensemble du Machine Learning et des principaux concepts, le reste de cet ouvrage est résolument axé sur la pratique : le but n'est pas juste de vous enseigner la théorie, mais aussi que vous sachiez concrètement développer vos propres systèmes de ML. Bien que vous puissiez lire ce livre sans allumer votre ordinateur, vous êtes fortement encouragé(e) à mettre la main à la pâte au cours de la lecture, en particulier en faisant les exercices proposés à la fin de chaque chapitre, et dont les solutions sont disponibles à la fin de l'ouvrage.

Exemples de code

Tous les exemples figurant dans ce livre sont en open source et disponibles sous *https://github.com/ageron/handson-ml3* en tant que notebooks Jupyter : il s'agit de documents interactifs comportant du texte, des images et des fragments de code exécutable (en Python dans notre cas). La façon la plus simple et la plus rapide de commencer est d'exécuter ces notebooks en utilisant Google Colab, un service gratuit vous permettant d'exécuter directement en ligne n'importe quel notebook Jupyter, sans avoir à installer quoi que ce soit sur votre machine. Vous aurez seulement besoin d'un navigateur internet et d'un compte Google.

Dans ce livre, je supposerai que vous utilisez Google Colab, mais j'ai aussi testé les notebooks sur d'autres plateformes en ligne comme Kaggle ou Binder, que vous

1. Cette bibliothèque a été créée par David Cournapeau en 2007, et le projet est maintenant dirigé par une équipe de chercheurs à l'Institut national de recherche en informatique et en automatique (Inria).

pouvez donc utiliser si vous préférez. Sinon, vous pouvez aussi installer les bibliothèques et outils requis (ou l'image Docker de ce livre) et exécuter les notebooks directement sur votre propre ordinateur : reportez-vous aux instructions figurant sur la page *https://homl.info/install*.

Prérequis

Bien que ce livre ait été écrit plus particulièrement pour les ingénieurs en informatique, il peut aussi intéresser toute personne sachant programmer et ayant quelques bases mathématiques. Il ne requiert aucune connaissance préalable sur le Machine Learning mais il suppose les prérequis suivants :

- vous devez avoir un minimum d'expérience de programmation ;
- sans forcément être un expert, vous devez connaître le langage Python, et si possible également ses bibliothèques scientifiques, en particulier NumPy, Pandas et Matplotlib ;
- enfin, si vous voulez comprendre comment les algorithmes fonctionnent (ce qui n'est pas forcément indispensable, mais est tout de même très recommandé), vous devez avoir certaines bases en mathématiques dans les domaines suivants :
  - l'algèbre linéaire, notamment comprendre les vecteurs et les matrices (par exemple comment multiplier deux matrices, transposer ou inverser une matrice),
  - le calcul différentiel, notamment comprendre la notion de dérivée, de dérivée partielle, et savoir comment calculer la dérivée d'une fonction.

Si vous ne connaissez pas encore Python, il existe de nombreux tutoriels sur Internet, que nous vous encourageons à suivre : ce langage est très simple et s'apprend vite. En ce qui concerne les bibliothèques scientifiques de Python et les bases mathématiques requises, le site *github.com/ageron/handson-ml3* propose quelques tutoriels (en anglais) sous la forme de notebooks Jupyter. De nombreux tutoriels en français sont disponibles sur Internet. Le site *fr.khanacademy.org* est particulièrement recommandé pour les mathématiques.

Plan du livre

- Le premier chapitre présente une vue d'ensemble du Machine Learning ainsi que les concepts fondamentaux : qu'entend-on exactement par *apprentissage automatique* ? Quels problèmes le Machine Learning peut-il résoudre ? Quelles sont les principales catégories d'algorithmes et les principales difficultés que l'on peut rencontrer ? Qu'est-ce que le surajustement et le sous-ajustement ?
- Le deuxième chapitre attaque le vif du sujet en présentant un projet de Machine Learning de A à Z (en introduisant Scikit-Learn au passage) : d'abord analyser le problème (en l'occurrence estimer le prix de l'immobilier), puis obtenir les données, les nettoyer, choisir une fonction d'évaluation, sélectionner un modèle, l'entraîner sur les données d'entraînement, évaluer le système sur un jeu de données préalablement mis de côté, rechercher les meilleurs hyperparamètres grâce à la validation croisée, etc.

- Le chapitre 3 analyse les difficultés particulières liées aux problèmes de classification, en particulier les diverses façons d'évaluer un classificateur, et comment le régler au mieux selon ses besoins.
- Le chapitre 4 est le premier à ouvrir les boîtes noires, pour expliquer comment fonctionnent les algorithmes. Il explique en particulier les principaux modèles linéaires et montre comment les entraîner en ajustant progressivement leurs paramètres à l'aide de l'algorithme de descente de gradient. Il montre aussi diverses techniques de régularisation qui permettent d'éviter le piège du surajustement.
- Le chapitre 5 se concentre sur les machines à vecteur de support (SVM), des modèles très puissants, particulièrement prisés pour les problèmes complexes pour lesquels on ne dispose que d'assez peu de données.
- Le chapitre 6 détaille les arbres de décision. Il s'agit de modèle simples et polyvalents, particulièrement faciles à interpréter.
- Le chapitre 7 présente les forêts aléatoires, qui reposent sur une multitude d'arbres de décision, et il explore plus généralement diverses méthodes ensemblistes, qui permettent de combiner plusieurs modèles de Machine Learning.
- Le chapitre 8 détaille plusieurs algorithmes de réduction de dimensionnalité, permettant de réduire le volume d'un jeu de données pour échapper au fléau de la dimensionnalité et ainsi accélérer l'apprentissage, ou encore à des fins de visualisation des données.
- Enfin, le chapitre 9, ayant pour objet l'apprentissage non supervisé, traite notamment du partitionnement avec la méthode des *k*-moyennes ou l'algorithme DBSCAN, des modèles de mélange gaussien, de l'inférence bayésienne et de l'utilisation des modèles de mélange pour le partitionnement, l'estimation de densité, la détection d'anomalies ou de nouveautés. Il fait également un tour d'horizon des algorithmes correspondants.

Le Deep Learning

Pour traiter des problèmes particulièrement complexes, tels que la reconnaissance de la parole ou encore les traductions automatiques, on utilise généralement des réseaux de neurones, qui requièrent souvent de gros volumes de données et une immense puissance de calcul. Presque toutes les applications du Machine Learning mises en avant par la presse reposent sur des réseaux de neurones profonds (c'est-à-dire organisés en de nombreuses couches successives) : on parle alors de *Deep Learning* (apprentissage profond).

Ce livre n'aborde pas le Deep Learning, mais il vous donne les bases nécessaires pour le faire. Si à l'issue de ce livre vous souhaitez poursuivre vers le Deep Learning, nous vous recommandons le livre *Deep Learning avec Keras et TensorFlow*, des mêmes auteur et éditeur. Vous y apprendrez comment utiliser la bibliothèque TensorFlow pour créer diverses architectures de réseaux de neurones profonds – des réseaux de neurones convolutionnels (très utilisés pour l'analyse d'images), des réseaux de neurones

récurrents (utiles pour l'analyse de séquences de tailles arbitraires, telles que des séries temporelles ou la langue naturelle), des auto-encodeurs (capables de reconnaître des motifs et de les imiter) – et comment entraîner et déployer ces réseaux de neurones sur de nombreux serveurs grâce à TensorFlow. Enfin, ce deuxième livre présente le Deep Reinforcement Learning : il s'agit des techniques développées par DeepMind pour créer un système capable d'apprendre tout seul à jouer à des jeux Atari sans en connaître les règles à l'avance.

Les différents types de textes en exergue

Ce symbole indique une astuce ou une suggestion.

Ce symbole indique une précision ou une remarque générale.

Ce symbole indique une difficulté particulière ou un piège à éviter.

Remerciements

Jamais, dans mes rêves les plus fous, je n'aurais imaginé que la deuxième édition de ce livre rencontrerait un public aussi vaste. J'ai reçu de nombreux messages de lecteurs, avec beaucoup de questions, certains signalant gentiment des erreurs et la plupart m'envoyant des mots encourageants. Je suis extrêmement reconnaissant envers tous ces lecteurs pour leur formidable soutien. Merci beaucoup à vous tous ! N'hésitez pas à me contacter si vous voyez des erreurs dans les exemples de code ou simplement pour poser des questions (*https://homl.info/issues3*) ! Certains lecteurs ont également expliqué en quoi ce livre les avait aidés à obtenir leur premier emploi ou à résoudre un problème concret sur lequel ils travaillaient. Ces retours sont incroyablement motivants. Si vous trouvez ce livre utile, j'aimerais beaucoup que vous puissiez partager votre histoire avec moi, que ce soit en privé (par exemple, via *https://linkedin.com/in/aurelien-geron*) ou en public (par exemple dans un tweet *via* @aureliengeron ou par le biais d'un commentaire Amazon).

Grand merci aussi à toutes les personnes merveilleuses qui ont offert de leur temps et leur expertise pour réviser cette troisième édition, en corrigeant des erreurs et en faisant d'innombrables suggestions. Cette édition est tellement meilleure grâce à cela : Olzhas Akpambetov, George Bonner, Francois Chollet, Siddha Ganju, Sam Goodman, Matt Harrison, Sasha Sobran, Lewis Tunstall, Leandro von Werra et mon cher frère Sylvain. Vous êtes tous incroyables !

Je suis également très reconnaissant envers toutes les personnes qui m'ont soutenu tout au long du chemin, en répondant à mes questions, en suggérant des améliorations et en contribuant au code sur GitHub : en particulier, Yannick Assogba, Ian Beauregard, Ulf Bissbort, Rick Chao, Peretz Cohen, Kyle Gallatin, Hannes Hapke, Victor Khaustov, Soonson Kwon, Éric Lebigot, Jason Mayes, Laurence Moroney, Sara Robinson, Joaquin Ruales et Yuefeng Zhou.

Ce livre n'existerait pas sans le personnel fantastique d'O'Reilly, en particulier Nicole Taché, qui m'a fait des commentaires perspicaces, toujours encourageants et utiles : je ne pouvais pas rêver d'un meilleur éditeur. Merci également à Michele Cronin, qui m'a encouragé et m'a permis d'arriver au bout. Merci à toute l'équipe de la production, en particulier Elizabeth Kelly et Kristen Brown. Merci également à Kim Cofer pour la révision minutieuse, et à Johnny O'Toole, qui a géré la relation avec Amazon et répondu à beaucoup de mes questions. Merci à Kate Dullea pour l'amélioration importante de mes illustrations. Merci à Marie Beaugureau, Ben Lorica, Mike Loukides et Laurel Ruma d'avoir cru en ce projet et de m'avoir aidé à le définir. Merci à Matt Hacker et à toute l'équipe d'Atlas pour avoir répondu à toutes mes questions techniques concernant AsciiDoc, MathML et LaTeX, ainsi qu'à Nick Adams, Rebecca Demarest, Rachel Head, Judith McConville, Helen Monroe, Karen Montgomery, Rachel Roumeliotis et tous les autres membres d'O'Reilly qui ont contribué à ce livre.

Je n'oublierai jamais les personnes formidables qui m'ont aidé sur les deux premières éditions du livre : amis, collègues, experts, dont de nombreux membres de l'équipe de TensorFlow. La liste est longue : Olzhas Akpambetov, Karmel Allison, Martin Andrews, David Andrzejewski, Paige Bailey, Lukas Biewald, Eugene Brevdo, William Chargin, François Chollet, Clément Courbet, Robert Crowe, Mark Daoust, Daniel « Wolff » Dobson, Julien Dubois, Mathias Kende, Daniel Kitachewsky, Nick Felt, Bruce Fontaine, Justin Francis, Goldie Gadde, Irene Giannoumis, Ingrid von Glehn, Vincent Guilbeau, Sandeep Gupta, Priya Gupta, Kevin Haas, Eddy Hung, Konstantinos Katsiapis, Viacheslav Kovalevskyi, Jon Krohn, Allen Lavoie, Karim Matrah, Grégoire Mesnil, Clemens Mewald, Dan Moldovan, Dominic Monn, Sean Morgan, Tom O'Malley, James Pack, Alexander Pak, Haesun Park, Alexandre Passos, Ankur Patel, Josh Patterson, André Susano Pinto, Anthony Platanios, Anosh Raj, Oscar Ramirez, Anna Revinskaya, Saurabh Saxena, Salim Sémaoune, Ryan Sepassi, Vitor Sessak, Jiri Simsa, Iain Smears, Xiaodan Song, Christina Sorokin, Michel Tessier, Wiktor Tomczak, Dustin Tran, Todd Wang, Pete Warden, Rich Washington, Martin Wicke, Edd Wilder-James, Sam Witteveen, Jason Zaman, Yuefeng Zhou, et mon frère Sylvain.

Je souhaite également remercier Jean-Luc Blanc, des éditions Dunod, pour avoir soutenu ce projet, et pour la gestion et les relectures attentives des deux premières éditions. Merci également à Matthieu Daniel, qui a pris la suite de Jean-Luc Blanc pour cette troisième édition. Je tiens aussi à remercier vivement Anne Bohy qui a traduit cet ouvrage en français bien mieux que je n'aurais su le faire. Dotée d'une grande expérience en développement informatique et d'un doctorat en mathématiques, elle était sans doute la traductrice idéale pour ce livre. Enfin, je remercie chaleureusement Brice Martin, des éditions Dunod, pour sa relecture extrêmement rigoureuse, ses excellentes suggestions et ses nombreuses corrections.

Pour finir, je suis infiniment reconnaissant à ma merveilleuse épouse, Emmanuelle, et à nos trois enfants, Alexandre, Rémi et Gabrielle, de m'avoir encouragé à travailler dur pour ce livre. Leur curiosité insatiable fut inestimable : expliquer certains des concepts les plus difficiles de ce livre à ma femme et à mes enfants m'a aidé à clarifier mes pensées et à améliorer directement de nombreuses parties de ce livre. J'ai même eu droit à des biscuits et du café, qui pourrait en demander davantage ?

1

Vue d'ensemble du Machine Learning

Il n'y a pas si longtemps, si vous aviez demandé à votre téléphone de vous indiquer comment rentrer chez vous, ce dernier aurait ignoré votre demande et les gens se seraient interrogés quant à votre état mental. Mais l'apprentissage automatique, qu'on désigne volontiers par son appellation anglaise de « Machine Learning » ou ML, n'appartient plus au domaine de la science-fiction : des milliards de personnes l'utilisent chaque jour. À dire vrai, cela fait déjà quelques décennies qu'on le met en œuvre dans certaines applications spécialisées telles que la reconnaissance optique de caractères. Mais la première utilisation à grande échelle qui a amélioré la vie de centaines de millions de personnes date des années 1990 : c'est le filtre de spam. S'il ne s'agit pas exactement d'un robot intelligent, techniquement cela relève bien néanmoins de l'apprentissage automatique : le logiciel a si bien appris qu'il est rare désormais que vous ayez besoin de marquer un message comme indésirable. Par la suite, des centaines d'applications ML ont été peu à peu introduites dans les produits et fonctionnalités que vous utilisez régulièrement : commandes vocales, traduction automatique, recherche d'images, recommandations de produits et bien d'autres encore.

Où commence l'apprentissage automatique, et où s'arrête-t-il ? Lorsqu'on dit qu'une machine *apprend* quelque chose, qu'est-ce que cela signifie ? Si je télécharge l'ensemble des articles de Wikipédia sur mon ordinateur, ce dernier a-t-il vraiment « appris » quelque chose ? Est-il soudainement devenu plus intelligent ? Dans ce chapitre, nous allons commencer par clarifier ce qu'est l'apprentissage automatique et les raisons qui peuvent vous pousser à l'utiliser.

Puis, avant d'entreprendre d'explorer ce vaste territoire, nous en examinerons une vue d'ensemble, en découvrirons les principaux domaines et en étudierons les spécificités les plus notables : apprentissage supervisé / apprentissage non supervisé et leurs variantes, apprentissage en ligne / apprentissage groupé, apprentissage à partir

d'observations / apprentissage à partir d'un modèle. Après quoi, nous nous intéresserons au déroulement d'un projet ML type, nous évoquerons les principales difficultés que vous pourriez rencontrer et expliquerons comment évaluer et régler finement votre système.

Ce chapitre introduit un grand nombre de concepts fondamentaux (et de jargon) que chaque spécialiste du traitement des mégadonnées doit connaître par cœur. Il s'agira d'une présentation très générale (c'est le seul chapitre qui ne comporte pratiquement pas de code) ; l'ensemble est assez simple, mais je tenais à ce que tout soit parfaitement clair pour vous avant de passer à la suite. Préparez-vous un café, et commençons !

Si vous avez déjà des connaissances de base en apprentissage automatique, vous pouvez passer directement au chapitre 2. En cas de doute, essayez de répondre à toutes les questions figurant à la fin de ce chapitre.

1.1 QU'EST-CE QUE L'APPRENTISSAGE AUTOMATIQUE ?

L'apprentissage automatique est la science (et l'art) de programmer les ordinateurs de sorte qu'ils puissent *apprendre à partir de données*.

Voici une définition un peu plus générale :

> « [*L'apprentissage automatique est la*] *discipline donnant aux ordinateurs la capacité d'apprendre sans qu'ils soient explicitement programmés*. »
>
> Arthur Samuel, 1959

En voici une autre plus technique :

> « *Étant donné une tâche T et une mesure de performance P, on dit qu'un programme informatique apprend à partir d'une expérience E si les résultats obtenus sur T, mesurés par P, s'améliorent avec l'expérience E*. »
>
> Tom Mitchell, 1997

Votre filtre anti-spam est un programme d'apprentissage automatique qui peut apprendre à identifier les e-mails frauduleux à partir d'exemples de pourriels ou *spam* (par exemple, ceux signalés par les utilisateurs) et de messages normaux (parfois appelés *ham*). Les exemples utilisés par le système pour son apprentissage constituent le *jeu d'entraînement* (en anglais, *training set*). Chacun de ces exemples constitue ce qu'on appelle en statistique une *observation* (en anglais, *instance*). On parle parfois aussi d'*échantillon* (en anglais, *sample*). On appelle *modèle* la partie du système d'apprentissage automatique qui apprend et effectue des prédictions. Les réseaux de neurones et les forêts aléatoires sont des exemples de modèles. Dans le cas présent, la tâche T consiste à identifier parmi les nouveaux e-mails ceux qui sont frauduleux, l'expérience E est constituée par les *données d'entraînement*, et la mesure de

performance P doit être définie (vous pourrez prendre par exemple le pourcentage de courriels correctement classés). Cette mesure de performance particulière, appelée *exactitude* (en anglais, *accuracy*), est souvent utilisée dans les tâches de classification.

Si vous téléchargez l'ensemble des articles de Wikipédia, votre ordinateur dispose de beaucoup plus de données, mais cela ne le rend pas soudainement meilleur pour quelque tâche que ce soit. Par conséquent, il ne s'agit pas d'apprentissage automatique.

1.2 POURQUOI UTILISER L'APPRENTISSAGE AUTOMATIQUE ?

Voyons comment vous écririez un filtre de spam en utilisant des techniques de programmation traditionnelles (voir figure 1.1) :

1. Tout d'abord, vous examineriez à quoi ressemble en général un courrier frauduleux. Vous remarqueriez peut-être que certains mots (tels que « pour vous », « carte bancaire », « gratuit », « bravo »…) semblent apparaître souvent dans l'intitulé du sujet. Vous remarqueriez peut-être aussi quelques autres détails caractéristiques dans le nom de l'expéditeur, le corps du message et d'autres parties du message.
2. Après quoi, vous écririez un algorithme de détection de chacune des caractéristiques repérées, et votre programme marquerait comme indésirables tous les messages dans lesquels un certain nombre de ces caractéristiques ont été détectées.
3. Vous testeriez alors le programme et répéteriez les étapes 1 et 2 jusqu'à obtenir un résultat satisfaisant.

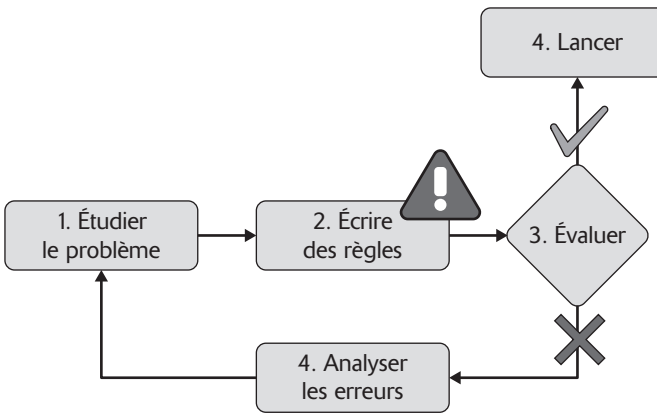

Figure 1.1 – Approche traditionnelle

Le problème étant compliqué, votre programme deviendra vraisemblablement une longue liste de règles complexes assez difficiles à maintenir.

Par contraste, un filtre anti-spam basé sur des techniques de Machine Learning apprend automatiquement quels sont les mots et les phrases qui constituent de bons prédicteurs de courriels frauduleux en détectant des associations de mots aux fréquences inhabituelles dans des exemples de courriels frauduleux comparés à des exemples de courriels licites (voir figure 1.2). Le programme est beaucoup plus court, plus facile à maintenir et a plus de chances de fournir de bons résultats.

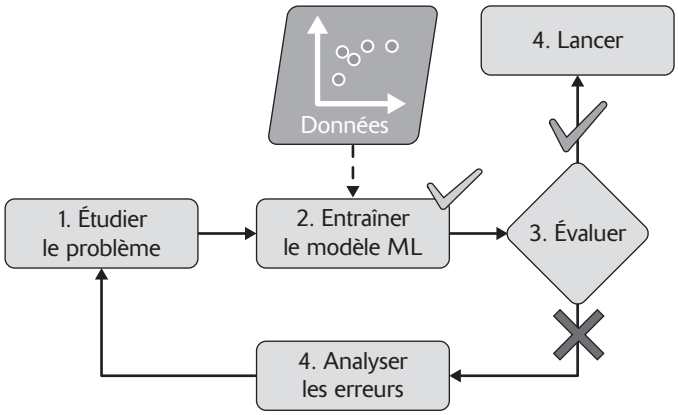

Figure 1.2 – L'approche du Machine Learning

Si les pirates remarquent que leurs e-mails contenant le mot « Pour vous » sont bloqués, ils peuvent décider d'écrire « For you » à la place. Un filtre anti-spam utilisant des techniques de programmation traditionnelles nécessiterait une mise à jour pour pouvoir intercepter les messages « For you ». Si les pirates continuent à tenter de contourner votre filtre anti-spam, vous devrez sans arrêt écrire de nouvelles règles.

Par contre, un filtre anti-spam basé sur des techniques de Machine Learning repère automatiquement que « For you » est devenu inhabituellement fréquent dans les messages indésirables signalés par les utilisateurs et il commence à en tenir compte sans votre intervention (voir figure 1.3).

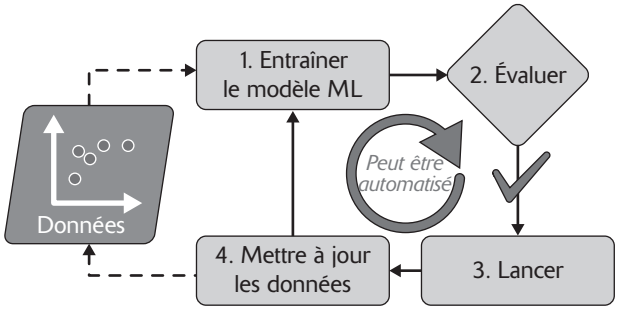

Figure 1.3 – Adaptation automatique aux évolutions

Un autre domaine dans lequel l'apprentissage automatique fait merveille est celui des problèmes pour lesquels il n'existe pas d'algorithme connu ou dont la complexité déroute. Considérons par exemple la reconnaissance vocale. Supposons que vous souhaitiez débuter par quelque chose de simple, en écrivant un programme capable de distinguer les mots « Un » et « Deux ». Vous pourriez remarquer que le mot « Deux » débute par un son de tonalité élevée (« D »), ce qui pourrait vous conduire à coder en dur un algorithme mesurant l'intensité des sons de tonalité élevée et à l'utiliser pour distinguer les « Un » et les « Deux ». Cependant, cette technique s'adaptera manifestement mal à des milliers de mots prononcés par des millions de personnes très différentes dans des environnements bruyants et dans des dizaines de langues distinctes. La meilleure solution (du moins jusqu'à maintenant) consiste à écrire un algorithme qui apprend par lui-même, à partir de nombreux enregistrements pour chaque mot.

Enfin, l'apprentissage automatique peut aider les humains à apprendre (voir figure 1.4) : ses modèles peuvent être inspectés afin de découvrir ce qu'ils ont appris (même si, pour certains d'entre eux, cela peut s'avérer difficile). Ainsi, une fois que le filtre anti-spam a été entraîné sur suffisamment de courriers frauduleux, on peut l'inspecter aisément pour découvrir la liste des mots et combinaisons de mots qu'il considère être les meilleurs prédicteurs de spam. Parfois, cela révélera des corrélations insoupçonnées ou de nouvelles tendances, permettant ainsi d'avoir une meilleure compréhension du problème.

Explorer de gros volumes de données pour y découvrir certains éléments de structuration qui n'étaient pas immédiatement apparents, c'est ce qu'on appelle l'*exploration des données* (en anglais, *data mining*), et l'apprentissage automatique excelle en ce domaine.

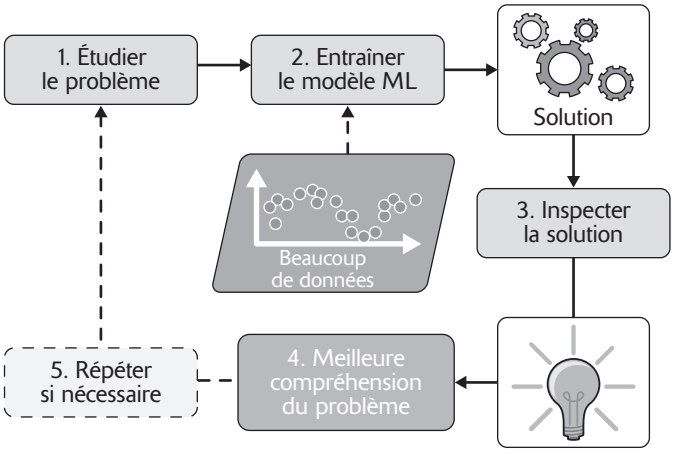

Figure 1.4 – L'apprentissage automatique peut aider les humains à apprendre

En résumé, l'apprentissage automatique est excellent pour :

- les problèmes pour lesquels les solutions existantes requièrent beaucoup d'ajustements fins ou de longues listes de règles (un algorithme d'apprentissage automatique peut souvent simplifier le code et donner de meilleurs résultats que l'approche traditionnelle) ;

- les problèmes complexes pour lesquels il n'existe aucune bonne solution si l'on adopte une approche traditionnelle (les meilleures techniques d'apprentissage automatique peuvent peut-être trouver une solution) ;
- les environnements fluctuants (un système d'apprentissage automatique peut s'adapter à de nouvelles données) ;
- l'exploration des problèmes complexes et des gros volumes de données.

1.3 EXEMPLES D'APPLICATIONS

Voyons maintenant quelques exemples concrets de tâches d'apprentissage automatique, ainsi que les techniques particulières nous permettant de les mener à bien :

- Analyse d'images de produits sur une chaîne de production pour les classifier automatiquement : il s'agit d'une classification d'images, effectuée en général à l'aide de réseaux de neurones convolutifs (en anglais, *convolutional neural networks* ou CNN)[2] ou parfois de transformateurs.
- Détection de tumeurs sur des scanners cérébraux : il s'agit d'une segmentation sémantique dans laquelle chaque pixel de l'image est classifié (étant donné que nous souhaitons déterminer l'emplacement exact et la forme des tumeurs), également à l'aide de CNN ou de transformateurs.
- Classification automatique d'articles de presse : il s'agit de traitement automatique du langage naturel (ou TALN) pour lequel on peut utiliser des réseaux de neurones récurrents (ou RNR) ou des CNN, mais des transformateurs fournissent des résultats encore meilleurs[2].
- Identification automatique de commentaires inappropriés dans les forums de discussion : il s'agit aussi de classification de texte à l'aide des mêmes outils de TALN.
- Synthèse automatique de longs documents : il s'agit d'une branche de TALN appelée *résumé automatique de texte* et utilisant les mêmes outils.
- Création d'un agent conversationnel (alias chatbot) ou d'un assistant personnel : une telle tâche fait appel à de nombreux composants de TALN, parmi lesquels des modules de compréhension de langage naturel (CLN) et de réponse aux questions.
- Prévision des futurs résultats financiers d'une entreprise, en fonction de différentes métriques de performance : il s'agit d'une tâche de régression (consistant à prédire des valeurs) pouvant être effectuée à l'aide d'un modèle de régression linéaire ou polynomiale (voir chapitre 4), à l'aide d'une régression SVM (voir chapitre 5), d'une forêt aléatoire (voir chapitre 7) ou d'un réseau de neurones artificiels[2]. Si vous voulez prendre en compte des séquences de métriques de performance antérieures, vous utiliserez peut-être des réseaux de neurones récurrents (RNR), des réseaux de neurones convolutifs (CNN) ou des transformateurs[2].

2. Pour plus d'informations, voir l'ouvrage *Deep Learning avec Keras et TensorFlow*, A. Géron, Dunod (3e édition).

- Développement d'une interface de commande vocale pour votre appli. Il s'agit de reconnaissance vocale, impliquant le traitement d'échantillons audio : s'agissant de séquences longues et complexes, on utilise en général des RNR, des CNN ou des transformateurs[2].
- Détection d'utilisations frauduleuses de cartes bancaires : il s'agit de détection d'anomalies que l'on peut effectuer à l'aide de forêts d'isolation, de mélanges gaussiens (voir chapitre 9) ou d'autoencodeurs[2].
- Segmentation de clientèle en fonction de leurs achats, de façon à pouvoir concevoir une stratégie marketing différente pour chaque segment : il s'agit de partitionnement, que l'on pourra effectuer à l'aide des k-moyennes, de DBSCAN ou d'autres algorithmes (voir chapitre 9).
- Élaboration d'une représentation graphique claire et explicite à partir d'un jeu de données complexe de grande dimension : il s'agit de visualisation de données mettant fréquemment en œuvre des techniques de réduction de dimension (voir chapitre 8).
- Recommandation d'un produit pouvant intéresser un client, compte tenu de ses achats antérieurs : il s'agit d'un système de préconisation. On peut fournir à un réseau de neurones artificiels[2] les achats antérieurs et autres informations concernant ce client et lui demander en sortie quel sera l'achat suivant le plus probable. Ce réseau de neurones sera en général entraîné sur les séquences d'achats antérieures de tous les clients.
- Réalisation d'un programme-robot intelligent pour un jeu. On utilise souvent pour cela l'apprentissage par renforcement (en anglais, *reinforcement learning* ou RL)[2], une technique d'apprentissage automatique consistant à entraîner des agents (tels que des robots) à choisir les actions maximisant leurs récompenses à long terme (p. ex., un robot peut recevoir une récompense chaque fois qu'un joueur perd des points de vie) dans un environnement donné (tel un jeu). Le fameux programme AlphaGo, qui a battu le champion du monde de go, utilisait des techniques de RL.

On pourrait allonger cette liste, mais cela devrait déjà vous donner un aperçu de l'extrême diversité et complexité des tâches auxquelles l'apprentissage automatique peut s'attaquer, et des diverses techniques utilisables dans chaque cas.

1.4 TYPES DE SYSTÈMES D'APPRENTISSAGE AUTOMATIQUE

Il existe tellement de types de systèmes d'apprentissage automatique différents qu'il est utile de les classer en grandes catégories :

- selon leur mode de supervision durant l'entraînement (apprentissage supervisé, non supervisé, semi-supervisé ou autre),
- selon que l'apprentissage s'effectue ou non progressivement, au fur et à mesure de l'obtention des données (apprentissage en ligne ou apprentissage groupé),

- selon qu'il se contente de comparer les nouvelles données à des données connues, ou qu'il détecte au contraire des éléments de structuration dans les données d'entraînement et construise un modèle prédictif à la façon d'un scientifique (apprentissage à partir d'observations ou apprentissage à partir d'un modèle).

Ces critères ne sont pas exclusifs, vous pouvez les combiner comme vous le souhaitez. Ainsi, un filtre anti-spam dernier cri peut apprendre au fur et à mesure en s'appuyant sur un modèle de réseau de neurones profond dont l'apprentissage s'effectue à partir des exemples de messages indésirables ou non que l'utilisateur lui fournit ; ceci en fait un système supervisé d'apprentissage en ligne à partir d'un modèle. Examinons maintenant plus soigneusement chacun de ces critères.

1.4.1 Supervision de l'apprentissage

Les systèmes d'apprentissage automatique peuvent être classés en fonction de l'importance et de la nature de la supervision qu'ils requièrent durant la phase d'entraînement. Il existe de nombreuses catégories, mais nous ne parlerons que des principales : l'apprentissage supervisé, l'apprentissage non supervisé, l'apprentissage semi-supervisé, l'apprentissage auto-supervisé et l'apprentissage avec renforcement.

Apprentissage supervisé

Dans l'*apprentissage supervisé*, les données d'entraînement que vous fournissez à l'algorithme comportent les solutions désirées, appelées *étiquettes* (en anglais, *labels*), comme sur la figure 1.5.

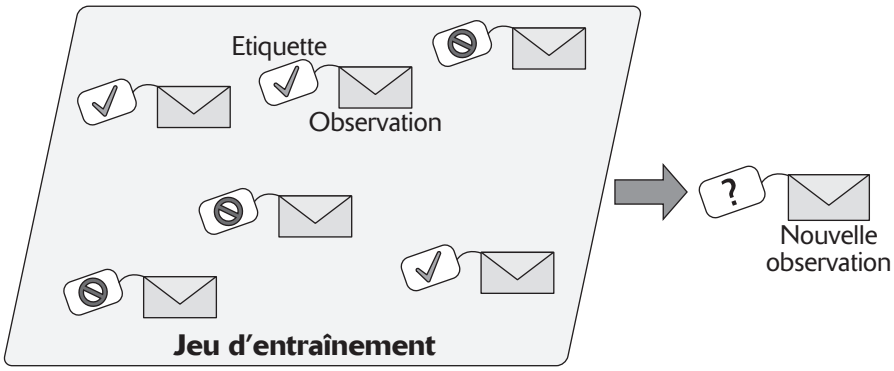

Figure 1.5 – Un jeu d'entraînement étiqueté pour la classification de spam (un exemple d'apprentissage supervisé)

Un exemple classique de tâche d'apprentissage supervisé est la *classification*. Le filtre de spam en constitue un bon exemple : son apprentissage s'effectue à partir de nombreux exemples d'e-mails accompagnés de leur *classe* (spam ou normal), à partir desquels il doit apprendre comment classer les nouveaux e-mails.

Une autre tâche classique consiste à prédire une valeur numérique *cible* (en anglais, *target*) telle que le prix d'une voiture à partir des valeurs d'un certain nombre d'*attributs* ou *variables*. Ces valeurs sont appelées les *caractéristiques* d'une observation. Ces variables, comme le kilométrage, l'âge, la marque, etc., sont appelées *variables explicatives* ou encore *prédicteurs*. Une tâche de ce type est une *régression*[3] (voir figure 1.6). Pour entraîner le système, vous devez lui donner beaucoup d'exemples de voitures, en y intégrant à la fois les variables explicatives et la variable à expliquer (ici, le prix).

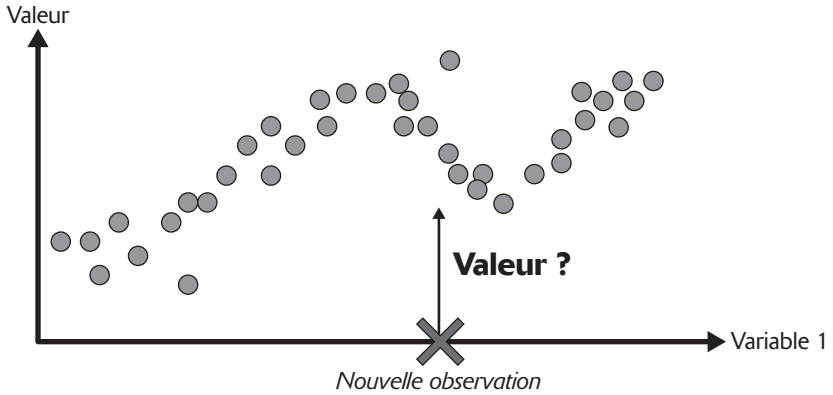

Figure 1.6 – Un problème de régression : prédire une valeur en fonction d'une variable en entrée (il y a généralement plusieurs variables en entrée, et parfois plusieurs valeurs en sortie)

Notez que certains modèles de régression peuvent être utilisés également en classification, et inversement. Par exemple, la *régression logistique* s'utilise couramment en classification, car elle peut fournir une valeur correspondant à la probabilité d'appartenance à une classe donnée (par exemple, 20 % de chances qu'un courriel soit du spam).

Les mots *cible* et *étiquette* sont en général considérés comme synonymes en apprentissage supervisé, mais *cible* est plus couramment employé en régression et *étiquette* est plus commun en classification. De plus, les *variables* sont parfois appelées *prédicteurs* ou *attributs*. Ces termes peuvent s'appliquer aux observations individuelles (« la caractéristique kilométrage de cette voiture vaut 15 000 ») ou à l'ensemble des observations (« la variable kilométrage est fortement corrélée à la variable prix »).

3. Pour la petite histoire, ce nom curieux est un terme statistique introduit par Francis Galton alors qu'il étudiait le fait que les gens de grande taille ont tendance à avoir des enfants plus petits qu'eux. Les enfants étant plus petits, il a nommé cela *régression vers la moyenne*. Ce nom fut alors donné aux méthodes qu'il utilisait pour analyser les corrélations entre variables.

Apprentissage non supervisé

Dans l'*apprentissage non supervisé*, comme vous vous en doutez, les données d'apprentissage ne sont pas étiquetées (voir figure 1.7). Le système essaie d'apprendre sans professeur.

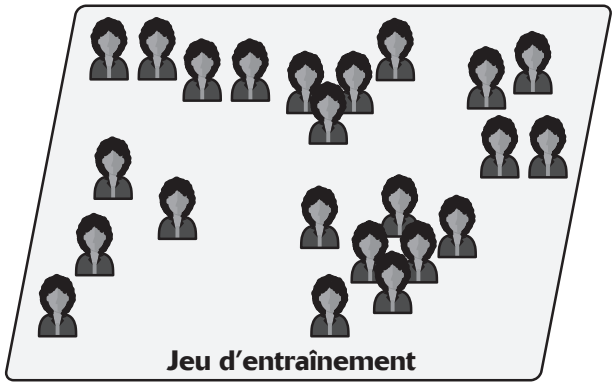

Figure 1.7 – Un jeu d'entraînement non étiqueté pour apprentissage non supervisé

Si par exemple vous disposez de nombreuses données sur les visiteurs de votre blog, vous pourriez effectuer un *partitionnement* (en anglais, *clustering*) pour tenter de détecter des groupes de visiteurs similaires (voir figure 1.8). À aucun moment vous ne dites à l'algorithme à quel groupe un visiteur appartient : il les détermine sans aide de votre part. Il peut par exemple remarquer que 40 % de vos visiteurs sont des hommes adorant les bandes dessinées et lisant en général votre blog le soir, alors que 20 % sont des jeunes passionnés de science-fiction qui le consultent durant le week-end, et ainsi de suite. Si vous utilisez un algorithme de partitionnement de type *classification hiérarchique* (en anglais, *hierarchical clustering)*, il subdivisera éventuellement chaque groupe en groupes plus petits. Ceci pourra vous aider à cibler vos publications pour chaque groupe.

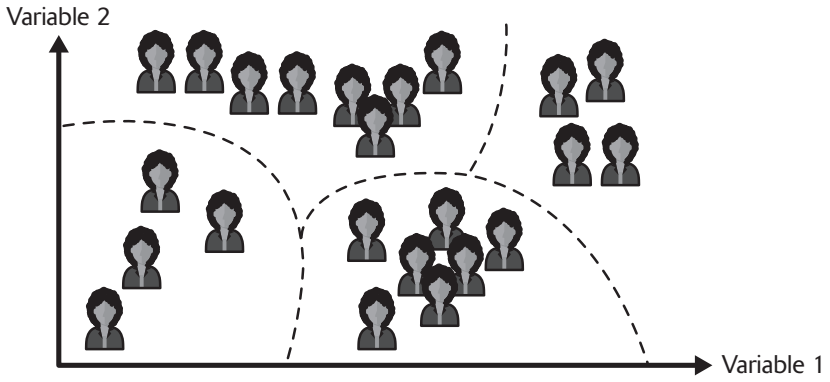

Figure 1.8 – Partitionnement

Les algorithmes de *visualisation* constituent de bons exemples d'apprentissage non supervisé : vous les alimentez avec un grand nombre de données complexes non étiquetées, et ils fournissent en retour une représentation 2D ou 3D de vos données sous forme de nuage de points (voir figure 1.9). Ces algorithmes s'efforcent de préserver la structure aussi bien que possible (par exemple en tentant d'éviter que des classes distinctes dans l'espace de départ se superposent dans la visualisation), afin que vous puissiez comprendre comment les données sont organisées et identifier peut-être une structuration insoupçonnée.

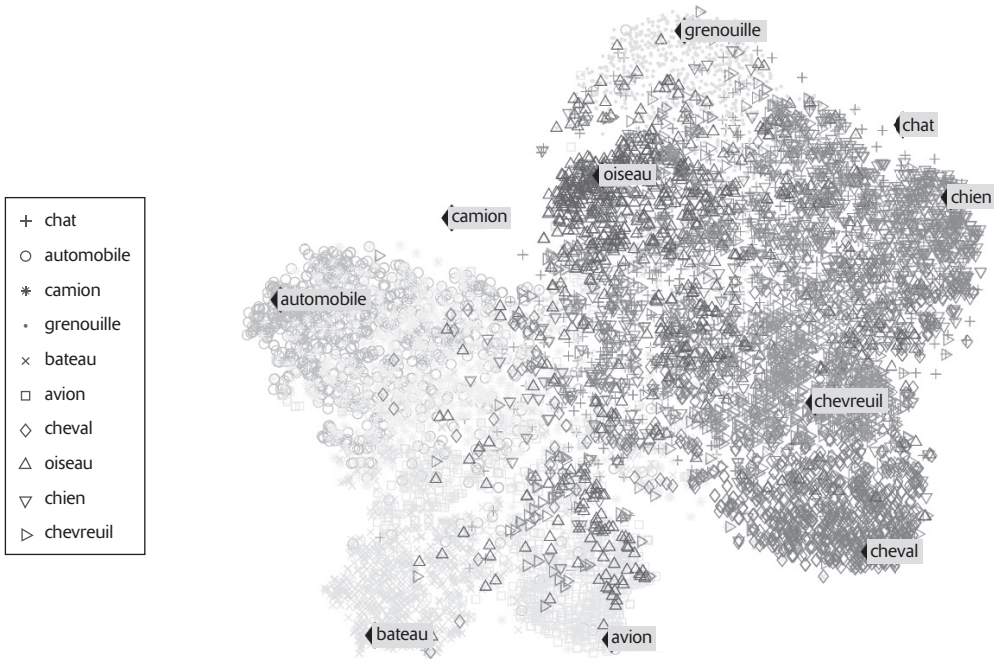

Figure 1.9 – Exemple d'une visualisation t-SNE mettant en évidence des classes sémantiques[4]

Une tâche connexe est la *réduction de dimension* : l'objectif est de simplifier les données sans perdre trop d'informations. Un des moyens d'y parvenir consiste à agréger plusieurs variables corrélées : le kilométrage d'une voiture, par exemple, peut être fortement corrélé à son âge, de sorte que l'algorithme de réduction de dimension les combinera en une seule variable représentant la vétusté de la voiture. C'est ce qu'on appelle l'*extraction de variables* (en anglais, *feature extraction*).

4. Remarquez que les animaux sont bien séparés des véhicules et que les chevaux sont proches des chevreuils mais éloignés des oiseaux. Figure reproduite avec l'autorisation de Richard Socher *et al.*, « Zero-Shot Learning Through Cross-Modal Transfer », *Proceedings of the 26th International Conference on Neural Information Processing Systems*, 1 (2013), 935-943.

Il est souvent judicieux de tenter de réduire le nombre de variables de vos données d'entraînement, c'est-à-dire de réduire la dimension de l'espace sur lequel vous allez travailler, en utilisant un algorithme de réduction de dimension avant de les fournir à un autre algorithme d'apprentissage automatique (tel qu'un algorithme d'apprentissage supervisé). Celui-ci s'exécutera plus vite, les données occuperont moins d'espace sur le disque et en mémoire, et dans certains cas on obtiendra aussi de meilleurs résultats.

Une autre tâche non supervisée d'importance est la *détection d'anomalies*, par exemple pour détecter des transactions de carte bancaire inhabituelles afin de prévenir la fraude, pour repérer des défauts de fabrication ou pour supprimer automatiquement les données aberrantes d'un jeu de données qu'on va fournir à un autre algorithme d'apprentissage. Durant la phase d'apprentissage, le système reçoit essentiellement des observations normales afin qu'il apprenne à les reconnaître et que, lorsqu'il recevra une nouvelle observation, il puisse déterminer si celle-ci est normale ou si c'est vraisemblablement une anomalie (voir figure 1.10). La *détection de nouveautés* est une tâche très similaire : elle vise à détecter les nouvelles observations qui semblent différentes de toutes les observations du jeu d'entraînement. Cela nécessite de disposer d'un jeu d'entraînement très « propre », exempt de toute observation que vous voudriez que l'algorithme détecte. Par exemple, si vous avez des milliers d'images de chiens et que 1 % de ces images représentent des chihuahuas, un algorithme de détection de nouveautés ne doit pas traiter les nouvelles images de chihuahuas comme des nouveautés. En revanche, les algorithmes de détection d'anomalies pourraient considérer ces chiens comme si rares et si différents des autres chiens qu'ils pourraient les classer comme des anomalies (sans vouloir offenser les chihuahuas).

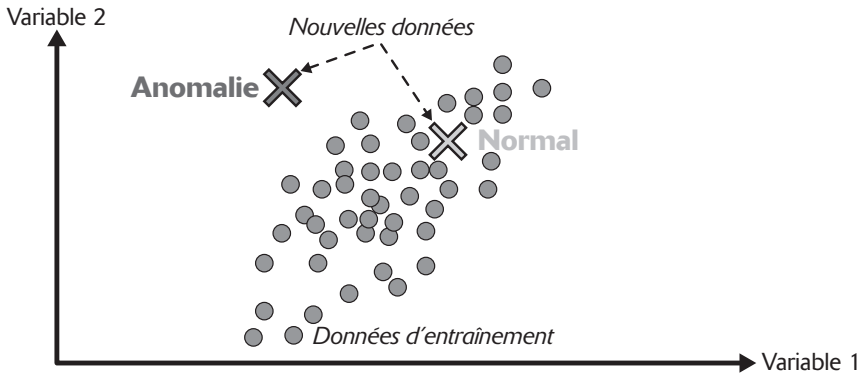

Figure 1.10 – Détection d'anomalies

Enfin, l'*apprentissage par association de règles* est également une tâche non supervisée couramment utilisée : elle a pour but d'explorer de larges ensembles de données et de découvrir d'intéressantes relations entre les variables. En supposant par exemple que vous gériez un supermarché, exécuter une règle d'association sur vos journaux de

vente vous permettrait peut-être de découvrir que les personnes achetant de la sauce barbecue et des chips ont aussi tendance à acheter des grillades. Cela pourrait vous inciter à présenter ces articles à proximité les uns des autres.

Apprentissage semi-supervisé

Étant donné que l'étiquetage des données prend généralement beaucoup de temps et d'argent, vous aurez souvent beaucoup d'observations non étiquetées et peu d'observations étiquetées. Certains algorithmes peuvent s'accommoder de données partiellement étiquetées. C'est ce qu'on appelle l'*apprentissage semi-supervisé* (voir figure 1.11).

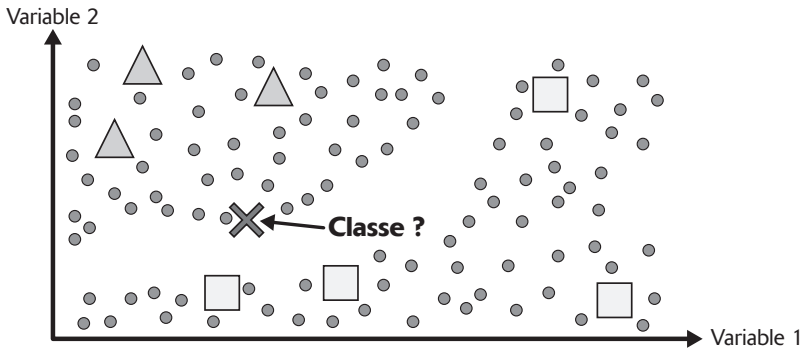

Figure 1.11 – Apprentissage semi-supervisé avec deux classes (triangles et carrés) : les exemples non étiquetés (cercles) permettent de classer une nouvelle instance (la croix) dans la classe triangle plutôt que dans la classe carrée, même si elle est plus proche des carrés étiquetés

Certains services d'hébergement d'images tels que Google Photos en constituent de bons exemples : une fois que vous avez téléchargé toutes vos photos de famille vers ce service, ce dernier reconnaît automatiquement que la personne A apparaît sur les photos 1, 5 et 11, tandis qu'une personne B apparaît sur les photos 2, 5 et 7. C'est la partie non supervisée de l'algorithme (partitionnement). Maintenant, le système a seulement besoin que vous lui disiez qui sont ces personnes : juste une étiquette par personne[5] et il est capable de nommer les personnes figurant sur chaque photo, ce qui est utile pour les recherches ultérieures.

La plupart des algorithmes d'apprentissage semi-supervisé sont des combinaisons d'algorithmes non supervisés et supervisés. Ainsi, on peut utiliser un algorithme d'agrégation pour regrouper des observations similaires, ce qui permet ensuite d'étiqueter les observations qui ne le sont pas à l'aide de l'étiquette la plus commune dans leur agrégat. Une fois que l'ensemble du jeu de données est étiqueté, il est possible d'utiliser n'importe quel algorithme d'apprentissage supervisé.

5. Ceci dans le cas idéal où le système fonctionne parfaitement. En pratique, il crée souvent plusieurs classes par personne et mélange parfois des personnes qui se ressemblent, c'est pourquoi il se peut que vous deviez fournir quelques étiquettes par personne et nettoyer manuellement quelques classes.