THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)*

Présentée et soutenue le *5 juillet 2017* **par :**

Nicolas Hug

Contributions to the use of analogical proportions for machine learning: Theoretical properties and application to recommendation.

JURY

Antoine Cornuéjols	Professeur des universités	Rapporteur
Hélène Fargier	Directeur de recherche CNRS	Examinateur
Aurélien Garivier	Professeur des universités	Membre invité
Jean Lieber	Maître de conférences HDR	Rapporteur
Laurent Miclet	Professeur honoraire	Membre invité
Henri Prade	Directeur de recherche CNRS	Examinateur
Gilles Richard	Professeur des universités	Directeur de thèse
Agnès Rico	Maître de conférences	Examinateur
Mathieu Serrurier	Maître de conférences HDR	Directeur de thèse
François Yvon	Professeur des universités	Président du jury

École doctorale et spécialité :

MITT : Domaine STIC : Intelligence Artificielle

Unité de Recherche :

Institut de Recherche en Informatique de Toulouse

Directeur(s) de Thèse :

Gilles Richard et *Mathieu Serrurier*

Rapporteurs :

Antoine Cornuéjols et *Jean Lieber*

Abstract

Analogical reasoning is recognized as a core component of human intelligence. It has been extensively studied from philosophical and psychological viewpoints, but recent works also address the modeling of analogical reasoning for computational purposes, particularly focused on analogical proportions. We are interested here in the use of analogical proportions for making predictions, in a machine learning context.

In recent works, analogy-based classifiers have achieved noteworthy performances, in particular by performing well on some artificial problems where other traditional methods tend to fail. Starting from this empirical observation, the goal of this thesis is twofold. The first topic of research is to assess the relevance of analogical learners on real-world, practical application problems. The second topic is to exhibit meaningful theoretical properties of analogical classifiers, which were yet only empirically studied.

The field of application that was chosen for assessing the suitability of analogical classifiers in real-world setting is the topic of recommender systems. A common reproach addressed towards recommender systems is that they often lack of novelty and diversity in their recommendations. As a way of establishing links between seemingly unrelated objects, analogy was thought as a way to overcome this issue. Experiments here show that while offering sometimes similar accuracy performances to those of basic classical approaches, analogical classifiers still suffer from their algorithmic complexity.

On the theoretical side, a key contribution of this thesis is to provide a functional definition of analogical classifiers, that unifies the various pre-existing approaches. So far, only algorithmic definitions were known, making it difficult to lead a thorough theoretical study. From this functional definition, we clearly identified the links between our approach and that of the nearest neighbors classifiers, in terms of process and in terms of accuracy. We were also able to identify a criterion that ensures a safe application of our analogical inference principle, which allows us to characterize analogical reasoning as some sort of linear process.

Le raisonnement par analogie est reconnu comme une des principales caractéristiques de l'intelligence humaine. En tant que tel, il a pendant longtemps été étudié par les philosophes et les psychologues, mais de récents travaux s'intéressent aussi à sa modélisation d'un point de vue formel à l'aide de proportions analogiques, permettant l'implémentation de programmes informatiques. Nous nous intéressons ici à l'utilisation des proportions analogiques à des fins prédictives, dans un contexte d'apprentissage artificiel.

Dans de récents travaux, les classifieurs analogiques ont montré qu'ils sont capables d'obtenir d'excellentes performances sur certains problèmes artificiels, là où d'autres techniques traditionnelles d'apprentissage se montrent beaucoup moins efficaces. Partant de cette observation empirique, cette thèse s'intéresse à deux axes principaux de recherche. Le premier sera de confronter le raisonnement par proportion analogique à des applications pratiques, afin d'étudier la viabilité de l'approche analogique sur des problèmes concrets. Le second axe de recherche sera d'étudier les classifieurs analogiques d'un point de vue théorique, car jusqu'à présent ceux-ci n'étaient connus que grâce à leurs définitions algorithmiques. Les propriétés théoriques qui découleront nous permettront de comprendre plus précisément leurs forces, ainsi que leurs faiblesses.

Comme domaine d'application, nous avons choisi celui des systèmes de recommandation. On reproche souvent à ces derniers de manquer de nouveauté ou de surprise dans les recommandations qui sont adressées aux utilisateurs. Le raisonnement par analogie, capable de mettre en relation des objets en apparence différents, nous est apparu comme un outil potentiel pour répondre à ce problème. Nos expériences montreront que les systèmes analogiques ont tendance à produire des recommandations d'une qualité comparable à celle des méthodes existantes, mais que leur complexité algorithmique cubique les pénalise trop fortement pour prétendre à des applications pratiques où le temps de calcul est une des contraintes principales.

Du côté théorique, une contribution majeure de cette thèse est de proposer une définition fonctionnelle des classifieurs analogiques, qui a la particularité d'unifier les approches préexistantes. Cette définition fonctionnelle nous permettra de clairement identifier les liens sous-jacents entre l'approche analogique et l'approche par k plus-proches-voisins, tant au plan algorithmique de haut niveau qu'au plan des propriétés théoriques (taux d'erreur notamment). De plus, nous avons pu identifier un critère qui rend l'application de notre principe d'inférence analogique parfaitement certaine (c'est-à-dire sans erreur), exhibant ainsi les propriétés linéaires du raisonnement par analogie.

Remerciements

Mes remerciements s'adressent d'abord naturellement à mes trois directeurs de thèse, Messieurs Henri Prade, Gilles Richard et Mathieu Serrurier. Jusqu'au moindre résultat, les contributions apportées dans ce manuscrit sont le fruit d'une étroite collaboration avec mes directeurs de thèse, sans qui rien de tout ça ne serait arrivé. Je leur suis très reconnaissant pour leur précieuse expertise, ainsi que pour l'énergie et la patience qu'ils m'ont accordées, en me laissant toujours le degré d'autonomie dont j'avais besoin.

Je remercie également Mesdames et Messieurs :

- Antoine Cornuéjols,
- Hélène Fargier,
- Aurélien Garivier,
- Jean Lieber,
- Laurent Miclet,
- Agnès Rico,
- et François Yvon

pour avoir accepté de prendre part au jury de cette thèse. Parmi eux, ma gratitude va particulièrement à Antoine Cornuéjols et Jean Lieber pour leurs critiques éclairées et leurs rapports minutieux, qui m'ont permis d'améliorer significativement la qualité de ce manuscrit.

Je remercie également chaleureusement Miguel Couceiro avec qui nous avons dernièrement beaucoup collaboré, et sans qui le chapitre 6 de cette thèse n'aurait jamais vu le jour.

Je terminerai enfin en envoyant de gros bisous à Lauriane et Audren, pour avoir enduré la (première !) relecture. De toute ma vie, je n'avais jamais vu autant de rouge.

Table of Contents

Introduction

Analogical reasoning is widely recognized as a powerful ability of human intelligence. It can lead to potential conclusions for new situations by establishing links between apparently unrelated domains. One well-known example is the Rutherford-Bohr model of atom where electrons circle around the kernel, which was analogically inspired by the model of planets running around the sun. Unsurprisingly, this kind of reasoning has generated a lot of attention from the artificial intelligence community.

In this work we will focus on a particular model of analogical reasoning, based on **analogical proportions**. An analogical proportion is a statement of the form *a is to b as c is to d*, and expresses the fact a differs from b in the same manner as c differs from d. For example, one could say that *France is to Paris as Germany is to Berlin*, or that *electrons are to the kernel as planets are to the sun*. In some cases, when the element d of the proportion is not known[1], it can be **inferred** from the three other elements a, b, c. It seems indeed natural that even a child with basic geographic knowledge could answer the question *France is to Paris as Germany is to what?* with the correct answer: *Berlin*. And even in the event that our subject does not know that the name of the correct answer is *Berlin*, they could still **imagine** a city which is the capital of Germany, and suppose that this hypothetical city corresponds to the correct answer. We witness here two key cognitive processes that can be triggered using analogical proportions: **inference**, and **creativity**.

As a tool that allows inductive inference, analogical reasoning has been proposed for plausible reasoning and for machine learning purposes. Indeed, analogical proportion-based learning has been addressed in recent works: we may cite for instance [SY05a] for an application in linguistics (verb conjugation task), and [BMD07] for classification problems in a Boolean setting. Our investigations follow on from these two works, and the goal of this thesis is twofold. The first topic is to **apply analogical reasoning to practical problems**, and the second one is to **exhibit some theoretical properties of analogical classifiers**.

The application of analogical reasoning to concrete tasks is motivated by the fact that the previous empirical investigations were only led in standard artificial settings. To our knowledge, the analogical learners were still to be experienced in real-world problems to assess their suitability in concrete applications, where the requirements for success are often significantly

[1] Or any other element, actually.

different. We have chosen here to apply analogical learning to the field of recommender systems.

Also, our theoretical investigations are motivated by the fact that so far, analogical classifiers were only known from their algorithmic descriptions. In fact, each implemented classifier provides a clean description of *how to compute*, but we definitely miss a clean description of *what do we actually compute*. The previous experimental investigations were not directed to provide an analytical view, and as a result analogical classifiers were yet quite poorly understood: we had no clear knowledge about their strengths and weaknesses.

Road map and contributions

Chronologically, we first focused on the applicative side. Our first contributions were indeed devoted to applying analogical reasoning to the task of recommendation. Later, we led our theoretical investigations on analogical classification.

In this document, we will choose to present our work somehow differently. After having recalled the necessary background on analogical proportions in Chapters 1 and 2, we will present some of our theoretical results in Chapter 3. Then, we will fully describe our contributions to analogical recommendation in Chapters 4 and 5, and finally we will address more theoretical considerations in Chapter 6. Simply put, our contributions to the theoretical side will be split up by the topic of analogical recommendation.

This choice is in fact motivated by pedagogical reasons. It will indeed become clear that the topic of analogical recommendation (Chapters 4 and 5) is a lot easier to introduce and to understand once we have enough background on analogical classification (Chapter 3), as our methods for analogical recommendation are strongly inspired by previous works on analogical classification. Moreover, whereas the motivations for Chapter 6 could have been directly derived from the results of Chapter 3, it will be more interesting to account for Chapter 6 in the light of the results of Chapter 5.

We now provide the detailed structure of this document.

The first chapter will provide the necessary background on existing models of analogical reasoning, with a strong emphasis on models that allow to perform computational inference. It will appear that these models are, for the most part, motivated by cognitive and psychological aspects, and depart from our main computational tool: formal analogical proportions.

Formal analogical proportions are the subject of the second chapter. We will provide their definitions in various algebraic settings, focusing on the two proportions that we will mostly use: the arithmetic proportion (dealing with real numbers), and the Boolean proportion. In addition, we will try to give a geometrical insight on these two proportions, which to the best of our knowledge had never been done before. We will also go through a toy classification

problem that will allow us to describe the process of **analogical equation solving**[2], and the **analogical inference principle** that underlies all of our investigations. In a classification context, this principle states that if a is to b as c is to d, then we should also have that $f(a)$ is to $f(b)$ as $f(c)$ is to $f(d)$, where $f(x)$ is the function that determines the class of x. It is obviously an unsound inference principle, in that the conclusion does not follow from the premise. But as we will see, it can still be used for classification tasks, which is the subject of the third chapter.

The study of analogical classification is indeed the object of Chapter 3, which corresponds to a detailed version of our ECAI paper: [HPRS16a]. In this chapter, we will start to address one of the two main objectives of this thesis: identifying theoretical properties of analogical classifiers. Our first contribution will be to provide a functional definition of analogical classifiers, that will unify the two pre-existing approaches mentioned earlier ([SY05a] and [BMD07]). This new definition will bring more insight into these classifiers, enabling us to derive results related to their Vapnik-Chervonenkis dimension, as well as their error rate. Our functional definition will also reveal the close links between the analogical classifiers and the k-nearest neighbors (k-NN) techniques. We will show indeed that the analogical classification process can be viewed as a two-step procedure that first consists in extending the training set (by **generating** new examples), and then applying a k-NN classifier. Quite remarkably, the two key cognitive processes related to analogical proportions (inference and creativity), are here blended together. From these results, a natural question arises: how can we ensure a training set extension that is completely error-free? This question will be addressed later in Chapter 6, while the following two chapters (4 and 5) will be devoted to our second main objective: applying analogical reasoning to the task of recommendation.

Chapter 4 will be dedicated to the necessary background on recommender systems. We will briefly review the three main families of recommender systems, namely content-based techniques, collaborative filtering, and knowledge-based systems. We will also formally define the problem that we plan to address, which is that of rating prediction: given a database of user-item interactions taking the form of ratings, the goal is to predict all the ratings for the pairs (user, item) that are not in the database. The different measures allowing to assess the quality of the predictions will be presented. Finally, as our contributions will be of a collaborative nature, we will thoroughly detail two popular methods for collaborative filtering: the neighborhood-based techniques, and the matrix-factorization-based methods. These two families of algorithms will serve as benchmarks to compare the performance of our own algorithms.

In Chapter 5, we will present our contributions to the topic of recommender systems. We will first describe our preliminarily investigations, which are a direct adaptation of the analogi-

[2]Very roughly, analogical equation solving is the process of finding the unknown x in the proportion *a is to b as c is to x*, e.g. *France is to Paris as Germany is to What?*

cal classifier described in the previous chapter. These first investigations were the object of a paper published at ISMIS [HPR15]. It will become clear that while offering similar accuracy to the traditional approaches, analogical recommenders suffer from their cubic complexity. Acknowledging this fact, we developed another view of analogical recommendation, taking into account the fact that some users may have different interpretation of the rating scale. The algorithms that we derived were much more scalable than the previous ones, and these investigations were described in a Fuzz-IEEE paper [HPRS16b]. Finally, we will address the problem of mining analogical proportions between users (or items) in a rating database [HPRS16c]. Our results will help us to retrospectively interpret the modest performances of our analogical recommenders: as bland as it may seem, it turns out that there were just not enough decent analogies to be found in the available databases.

But this down-to-earth observation will lead us to the very questioning of the analogical inference principle that had been underlying all of our investigations so far. In the last chapter, we will go back to our theoretical considerations, and provide a criterion that allows us to apply analogical inference in a sound way. We have briefly described that the analogical inference principle states that if four elements a, b, c, d are in proportion, then their classes $f(a), f(b), f(c), f(d)$ should also be in proportion. We will provide a complete characterization of the functions f such that when four elements are in proportion, then their image by f are also in proportion: these functions ensure a safe use of the analogical inference principle. It will be clear that these functions are also the ones that allow to extend a training set by analogy in a perfectly safe way. We will call these functions the **analogy preserving** functions, and they will turn out to be the well-known affine functions, both in Boolean and real settings. These investigations led to an IJCAI paper [CHPR17], and provide various future research tracks.

We will now start with the first chapter, dedicated to the description of previous (and current) attempts at formalizing analogical reasoning.

Introduction in French

Le raisonnement par analogie est largement considéré comme une caractéristique clef de l'intelligence humaine. Il permet d'arriver à de potentielles conclusions pour de nouvelles situations, en établissant des liens entre des domaines en apparence très différents. Un exemple connu est celui du modèle de l'atome de Rutherford-Bohr, où les électrons gravitent autour du noyau, comme les planètes du système solaire gravitent autour du soleil. Ce type de raisonnement a fait l'objet de nombreuses études de la part des psychologues et des philosophes, mais il intéresse aussi la communauté de l'intelligence artificielle.

Ce travail s'intéresse à un mode particulier de raisonnement par analogie, à savoir le raisonnement par proportions analogiques. Une proportion analogique est une proposition de la forme *a est à b comme c est à d*, et exprime le fait que a diffère de b comme c diffère de d. On pourrait par exemple affirmer que la *France est à Paris ce que l'Allemagne est à Berlin*, ou encore que *les électrons sont au noyau ce que les planètes sont au soleil*. Dans certains cas, lorsque l'élément d n'est pas connu, on pourra l'inférer à partir des trois autres éléments a, b, et c. Il semble en effet naturel qu'un enfant doté de connaissances géographiques basiques pourrait répondre à la question *La France est Paris ce que l'Allemagne est à quoi ?* Et même dans l'éventualité où l'enfant ne connaîtrait pas le nom de la ville *Berlin*, il pourrait tout de même s'imaginer une hypothétique ville au nom inconnu, mais qui serait la capitale de l'Allemagne. L'utilisation des proportions analogiques nous permet ici de toucher du doigt deux processus cognitifs clefs : l'inférence, et la créativité.

En tant qu'outil permettant une forme d'inférence inductive, le raisonnement par analogie a déjà été envisagé pour des problèmes d'apprentissage artificiel. On pourra citer par exemple [SY05a] pour une application en linguistique, ainsi que [BMD07] pour des problèmes de classification dans des domaines booléens. Nos recherches s'inscrivent dans la lignée de ces deux précédents travaux, et le but de cette thèse est double. Le premier sujet de recherche est d'appliquer le raisonnement analogique à des problèmes concrets, et le second est d'étudier les classifieurs analogiques d'un point de vue théorique, afin d'en exhiber des propriété théoriques intéressantes.

L'application du raisonnement analogique à des problèmes concrets est motivée par le fait que les précédentes recherches ont principalement été menées d'un point de vue expérimental, dans des environnements assez artificiels. A notre connaissance, aucun travail n'avait été mené pour évaluer la pertinence des classifieurs analogiques sur des tâches concrètes, où les critères de succès sont souvent différents. Nous avons ici choisi d'appliquer le raisonnement analogique au problème de la recommandation, et plus particulièrement à celui de la prédiction de notes.

D'autre part, nos recherches théoriques sont motivées par le fait que jusqu'alors, les classifieurs analogiques n'étaient connus que via leurs descriptions algorithmiques. On avait une connaissance claire de comment calculer la sortie d'un classifieur analogique, mais la nature de ce qui était effectivement calculé restait encore assez floue. On manquait alors d'outils pour

caractériser les forces et les faiblesses de tels classifieurs.

Annonce du plan et contributions

Chronologiquement, nous nous sommes d'abord intéressés à la partie applicative : nos premières contributions concernaient l'application du raisonnement analogique à la recommandation. Ensuite, dans un second temps, nous nous sommes intéressés à la partie théorique.

Nous avons choisi de présenter nos travaux d'une manière quelque peu différente. Nous allons d'abord exposer les connaissances requises sur les proportions analogiques dans les chapitres 1 et 2, puis nous présenterons une partie de nos résultats théoriques dans le chapitre 3. Ensuite, nous nous attaquerons à la partie applicative dans les chapitres 4 et 5. Enfin dans le chapitre 6, nous reviendrons sur des aspects théoriques.

Ce choix est en fait motivé par des raisons pédagogiques. En effet, il apparaîtra clair que le sujet de la recommandation analogique (chapitres 4 et 5) s'introduit et se comprend beaucoup plus facilement une fois que l'on a déjà étudié le chapitre 3 qui traite de classification analogique. De plus, bien que l'on aurait pu s'attaquer au chapitre 6 directement après le chapitre 3, il sera d'autant plus intéressant d'étudier le chapitre 6 à la lumière des conclusions des chapitres 4 et 5.

Ce document est organisé comme suit.

Le premier chapitre donnera les prérequis nécessaires sur différents modèles de raisonnement par analogie, en insistant particulièrement sur les modèles permettant l'inférence. Il apparaîtra clair que ces modèles sont, pour la plupart, motivés par des recherches en sciences cognitives ou psychologiques, ce qui les distingue de notre principal outil : les proportions analogiques.

Les proportions analogiques seront l'objet du deuxième chapitre. Nous donnerons leurs définitions dans différents domaines algébriques, en insistant sur les deux proportions que nous utiliserons le plus : la proportion arithmétique et la proportion booléenne. De plus, nous essaierons d'aborder ces proportions d'un point de vue géométrique afin d'en mieux saisir les différentes caractéristiques. Nous étudierons aussi un problème de classification *jouet*, ce qui nous permettra d'introduire la notion de résolution d'équation analogique, ainsi que le principe d'inférence analogique qui est sous-jacent à toutes nos recherches. Dans un contexte de classification, ce principe stipule que si a est à b ce que c est à d, alors on devrait aussi avoir que $f(a)$ est à $f(b)$ comme $f(c)$ est à $f(d)$, où $f(x)$ est la fonction qui détermine la classe de l'élément x. Ce principe est évidemment non correct, dans le sens où la conclusion (les classes sont en proportion) n'est pas logiquement déductible de la prémisse (les éléments sont en proportion). Comme nous le verrons, il reste néanmoins utilisable dans des tâches de classification, ce qui sera l'objet du troisième chapitre.

L'étude des classifieurs analogiques sera en effet le sujet du chapitre 3. Ce chapitre correspond à une version étendue d'un papier publié à l'ECAI [HPRS16a]. Nous commencerons ici à nous attaquer à un des deux thèmes principaux de cette thèse : identifier des propriétés théoriques des classifieurs analogiques. Notre première contribution sera de proposer une définition fonctionnelle des classifieurs analogiques, qui unifie les deux approches préexistantes déjà mentionnées ([SY05a] et [BMD07]). Cette nouvelle définition nous permettra de dériver des résultats relatifs à la VC-dimension des classifieurs, ainsi que leur taux d'erreur. Cette définition fonctionnelle permettra aussi de mettre clairement en lumière les liens qui relient les classifieurs de type plus-proches-voisins (k-NN) et les classifieurs analogiques. En effet, on montrera que la classification analogique peut se concevoir comme la successions de deux étapes. Dans un premier temps, on étend l'ensemble d'apprentissage en ajoutant de nouveaux exemples, puis on utilise ensuite un classifieur k-NN sur cet ensemble étendu (appelé extension analogique). On remarquera que les deux processus cognitifs mentionnés précédemment au sujet des proportions analogiques (créativité et inférence) sont ici intimement liés. De ces résultats, une question naturelle se pose : comment assurer que l'extension analogique soit saine, c'est-à-dire comment être certain que les exemples générés ont été associés à la bonne classe ? Cette question sera résolue dans le chapitre 6, mais les deux chapitres suivants (4 et 5) s'intéresseront à l'application du raisonnement par analogie au problème de la recommandation.

Le chapitre 4 sera dédié aux prérequis nécessaires sur les systèmes de recommandation. Nous décrierons brièvement les trois principales familles de systèmes de recommandation, à savoir les techniques *content-based*, le filtrage collaboratif, et les techniques *knowledge-based*. Nous décrirons aussi formellement le problème que l'on se propose d'étudier, qui est celui de la prédiction de notes : étant donné un ensemble de notes données par des utilisateurs pour des items, le but est de prédire d'autres notes qui ne sont pas dans la base de données. Les différentes mesures qui permettent d'évaluer la qualité d'un système de recommandation seront présentées. Enfin, comme nos algorithmes seront de type *filtrage collaboratif*, nous décrirons en détail deux techniques de prédiction collaborative qui nous serviront à comparer les performances de nos algorithmes : les méthodes de voisinage, et les méthodes fondées sur la factorisation de matrice.

Dans le chapitre 5, nous présenterons nos contributions au problème de la recommandation. Nous décrirons d'abord nos premières investigations, qui sont une adaptation directe des classifieurs analogiques décrits dans le chapitre 3. Ces recherches ont fait l'objet d'un papier publié à ISMIS [HPR15]. Il apparaîtra que ces algorithmes, bien que proposant une précision semblable à celle des méthodes par voisinage, se révèlent d'une complexité calculatoire telle qu'il n'est pas raisonnable de les envisager comme des candidats potentiels à une implémentation dans un système effectif. Fort de ce constat, nous envisagerons une autre version de la recommandation analogique, cette fois fondée sur le fait que les utilisateurs ont tous une différente interpré-

tation de l'échelle de note. Les algorithmes qui en découleront proposeront une complexité algorithmique tout à fait raisonnable et sont l'objet d'un papier Fuzz-IEEE [HPRS16b]. Enfin, nous nous intéresserons au problème de la fouille de proportions analogiques dans une base de données partiellement renseignée, telle que celles dont on dispose lorsque l'on s'attaque à la prédiction de notes. Les résultats obtenus nous permettront de rétrospectivement interpréter les modestes performances de nos premiers algorithmes de recommandation analogique : même si cela semble tristement banal, il apparaîtra qu'il n'existait probablement tout simplement pas de bonnes analogiques dans les bases de données utilisées.

Mais cette observation quelque peu terre-à-terre ne nous empêchera pas de remettre en question le principe d'inférence analogique même. Dans le dernier chapitre, nous reviendrons à nos considérations théoriques, et nous proposerons un critère qui permet d'appliquer ce principe d'inférence de manière correcte. Nous avons brièvement décrit ce principe, qui stipule que si quatre élément a, b, c, d sont en proportion, alors leurs classes $f(a), f(b), f(c), f(d)$ doivent aussi être en proportion. Nous donnerons une caractérisation complète des fonctions f telles que lorsque quatre éléments sont en proportion, alors leurs images par f le sont aussi. Ces fonctions permettent d'utiliser le principe d'inférence analogique de manière sûre, et assurent une extension analogique saine (c'est-à-dire sans erreur). Nous les appellerons les fonctions qui préservent l'analogie, et nous montrerons qu'elles correspondent aux fonctions affines dans les domaines booléens et réels. Ces travaux ont abouti à la publication d'un papier IJCAI [CHPR17], et ouvrent la voie à différentes pistes de recherche.

Chapter 1

Computational models of analogical reasoning

Content

Roughly speaking, analogy is a matter of establishing parallels between two different situations. Reasoning by analogy allows to infer properties about one of the two situations based on our knowledge of the other. Closely related to analogical reasoning is the idea of analogical proportion, which is a statement of the form *a is to b as c is to d*, often written $a : b :: c : d$. Here also, one of the four elements may sometimes be inferred if the three

other are known: this is called the *solving* of the analogical equation $a : b :: c : \ ?$.

The first two chapters of this document will provide the necessary background on formal analogical models and especially analogical proportions. In this first chapter, we provide an overview of various attempts to formalize analogical reasoning, with a strong emphasis on computational models (the interested reader may refer to [PR14b] for an historical overview of analogical reasoning models). In the next chapter we will describe in more details the modeling of analogical proportions, especially in the Boolean and real settings and their application to machine learning tasks. This chapter is structured as follows.

Section 1.1 will be devoted to the description of existing models that do not make explicit use of analogical proportions, and Section 1.2 will describe models that make use of analogical proportions in one way or another, but that still do not use analogical proportions in a formal way (these models will be the object of the next chapter).

1.1 Models without analogical proportions

In this first section, we will review some past attempts at modeling analogical reasoning, where no use is made of analogical proportions.

1.1.1 First attempts by Pólya and Hesse

In his famous book *How to Solve It* [Pol45], the mathematician George Pólya suggests to his readers various ways of reaching the solution of mathematical problems. Among the different heuristics that are proposed, analogy has a prominent place. Considering the problem of finding the center of gravity of a homogeneous tetrahedron, Pólya suggests to observe that the tetrahedron and the triangle have many similar features, and to first find the center of gravity of the triangle: a somewhat simpler problem. He writes:

> *Knowing that the triangle and the tetrahedron are alike in many respects, we conjecture that they are alike in one more respect. It would be foolish to regard the plausibility of such conjectures as certainty, but it would be just as foolish, or even more foolish, to disregard such plausible conjectures.*

As the center of gravity of a triangle is the meeting point of the three medians, the analogical argument suggests that the center of gravity of the tetrahedron is the meeting point of the six median planes. The work of Pólya is probably the most elaborated analysis of analogical reasoning as a tool for problem solving in the modern era. Later in [Pol54], Pólya formalizes

Bovine	Equid
Is mammal	Is mammal
Has hoofs	Has hoofs
Eats grass	Eats grass
$\sim$ 1.6 meters high	$\sim$ 1.6 meters high
Harmless	Harmless
Domesticated	*Domesticable*?

Table 1.1: The tabular representation of an analogy between cows and horses.

this idea in what he calls a *pattern of plausible inference*:

$$\frac{\begin{array}{c}a \text{ is analogous to b} \\ a \text{ is true}\end{array}}{\text{b is more credible}}$$

Another early attempt at formalizing analogical reasoning is due to the philosopher of science Mary Hesse. In [Hes66], Hesse proposes to represent an analogy in a tabular form, as illustrated in Table 1.1. We already recognize here a mapping between two domains: a source domain (the bovine domain) and a target domain (the equid domain). In Hesse's theory, an analogy should follow three general principles to be considered seriously. The first (questionable) requirement is the **requirement of material analogy**: for Hesse, material similarities (i.e. *observable* similarities) between the two domains are more important than formal similarities, which emerge when the two domains can be seen as two different interpretations of a unifying model. In that sense, Hesse rejects the deep underlying similarities that may bond the two domains. The second requirement is that when it comes to inferring a property Q (e.g. *is a horse domesticable?*), there should be a **causal relationship** between the known properties P and the one we need to infer. For example in our example of Table 1.1, the fact that the animals are harmless and that they eat grass probably helps us to domesticate them. Whether they have hoofs is however fairly irrelevant for the matter. Finally, the **no essential difference condition** states that if a property P is causally related to the hypothetical property Q in the source domain, then it should also be the case in the target domain. For example if the horse was not harmless, or if it had a very special dietary regime, then we should probably not infer that it is easily domesticable.

Let us also briefly mention that in [Hes59], Mary Hesse outlined a formal framework allowing to define analogical proportions in a modular lattice, anticipating quite remarkably other future works that will be reviewed in Chapter 2.

1.1.2 Analogy as a structural mapping

The role of a structural mapping between a source domain and a target domain has been recognized for a long time as a key component of analogical reasoning: see for example Pólya in [Pol54], or Hesse's theory that we have just described. As such, this principle has led to numerous theories and computational models that we briefly (and non-exhaustively) recall here.

Gentner's Structure Mapping Theory

Probably the most influential model of analogical reasoning is the Structure Mapping Theory (SMT), introduced by the American cognitive scientist Dedre Gentner in [Gen83]. The main feature of SMT is to consider that good analogies are those that result from strong, deep, relations and dependencies between the source and the target domains, rather than on some superficial characteristics. In this regard, SMT departs from Hesse's theory in a significant way.

The point of Gentner is that superficial similarities are often irrelevant, while what matters in an analogy are the underlying **structural**, high order relations between the objects at play. To exemplify, Gentner argues that when one says that *a battery is like a reservoir*, the analogy stands because at some abstract level, a battery and a reservoir serve the same purpose: to release some potential energy that has been stored for some time. The fact that batteries come in different shapes, colors and sizes than reservoirs does not play any role in the relevance of the analogy. This principle is called the **systematicity principle**, for which we now give some technical details.

The world is assumed to be represented by objects (belonging either to the source domain S or to the target domain T), along with some **predicates** that deal with one or more objects of the same domain. The distinction is made between predicates that only take one argument (**attributes** of objects), and those that take at least two arguments (**relations**). Higher-order relations are relations for which arguments are themselves relations, instead of simple objects. To illustrate these syntactic distinctions, *TALL(Bob)* and *BLONDE(Alice)* are attributes over the objects *Bob* and *Alice*. *ARE_FRIENDS(Bob, Alice)* and *HAVE_DINNER(Bob, Alice)* are first-order relations, and *CAUSE[ARE_FRIENDS(Bob, Alice), HAVE_DINNER(Bob, Alice)]* is a second-order relation.

In SMT, an analogy is defined as a one-to-one mapping M from S to T that maps relations (and only relations) between the two domains. The systematicity principle mentioned earlier states that during the mapping, attributes of objects (considered to be superficial features) are discarded and not taken into account, while higher-order relations are given priority over lower-order ones. Also, out of two relations of the same order, the one that is the most involved into other (higher-order) relations is the most likely to be mapped in the target domain. This last requirement gives an implicit rule to somehow assess the relevance of a relation in an analogy.

Note that this definition of analogy involves purely structural and syntactical features. The

semantic underlying the relations (or the objects) are completely out of concern. In our example, the fact that *Alice* and *Bob* are actually friends is of no importance: for SMT this relation is nothing but a first-order relation, with no particular meaning. As far as SMT is concerned, *Alice* and *Bob* could just as well be arch-enemies, it would not make any difference during a potential mapping process with a target domain (which would, for example, involve two other individuals with a similar relationship).

While SMT is a purely theoretical framework for analogy, these ideas have been practically implemented in a software called the Structure Mapping Engine (SME) [FFG89] written in LISP, leading to numerous applications. A recent example is that of a program that is able to solve Raven's Progressive Matrices [LFU10], a typical kind of visual geometric puzzles. The SME algorithm, in complete accordance with SMT, can be conceptually summarized as follows:

1. Look for all potential matches between relations in the source domain and the target domain.
2. Try to group matches into maximally consistent collections of matches.
3. From each collection, infer some relations that might stand in the target domain.

In its most simple form, the SME algorithm can be viewed as the finding of a maximum common subgraph between two graphs, namely those representing the source and the target domains. As such, SME is part of the connectionist approaches.

In [CFH92], Chalmers, French, and Hofstadter point out various concerns about SMT and SME. Among them is the fact that SME is too reliant on the (human-made) description inputs of the source and target domains, and that the intelligence mostly comes from these descriptions:

> *when the program's discovery of the correspondences between the two situations is a direct result of its being explicitly given the appropriate structures to work with, its victory in finding the analogy becomes somewhat hollow. Since the representations are tailored (perhaps unconsciously) to the problem at hand, it is hardly surprising that the correct structural correspondences are not difficult to find.*

Also, while the systematicity principle is undoubtedly at the core of many analogies, it should seem natural to challenge it in some other situations. It is indeed quite easy to find analogies were superficial features are the most decisive ones [Bar10, Bar16].

Prior to SMT: Winston's theory

We should mention that SMT and SME are not the first theory/engine couple that models analogical reasoning based on the mapping of two situations. Indeed in [Win80], Patrick H. Winston already presents a theory along with a program that also rely on this general view. Winston uses a propositional representation of the two situations that are extracted from natural language sentences.

Contrary to SMT, Winston's view of analogy is fairly close to that of Hesse. For Winston, the matching between the two situations should not only involve high order relations, but also the attributes of the objects. Moreover, he acknowledges the importance of **causal** relations to guide the mapping process, which is compliant with Hesse's theory. Winston also considers that the mapping process is *constraint-based*, which may have inspired the model of Holyoak and Thagard that we describe now.

The Constraint Satisfaction Theory

As some of the most influential theories of analogical reasoning, SMT and Winston's model have opened the way to various other models such as that of Holyoak and Thagard [HT89]. Here as well, an analogy is considered to be a mapping between two domains S and T[1]. Taking over Gentner's systematicity principle (in a relaxed form), Holyoak and Thagard exhibit two additional dimensions of importance in an analogical process. First, the semantics behind the objects at hand, i.e. the meaning that human agents associate with these objects, are taken into account. In this theory, the two relations *ARE_FRIENDS* and *ARE_ENEMIES* are not the same. This clearly stands in contrast with SMT, where all object attributes are simply discarded (along with their meanings), and seems to be in accordance with Hesse's requirement of material analogy. Second, this theory also involves the pragmatic considerations of the human agent: the goal and purpose of the analogist should somehow guide the mapping process in some direction or another. Mappings that serve the purpose of the agent are therefore given higher priority than others.

Another main difference with SMT, where the systematicity principle is a fixed, inflexible rule, is that here the three dimensions (systematicity, semantic similarity and pragmaticity) are interpreted as **constraints** and not as rigid directions. These constraints are only here to guide the mapping process.

Holyoak and Thagard's theory has been implemented in a LISP software called ACME (Analogical Constraint Mapping Engine), in a similar fashion as the Copycat program (detailed later in Section 1.2.3) in that they are both cooperative algorithms. Much like SME, ACME is part of the connectionist approaches.

Heuristic-Driven Theory Projection

Another framework where structural mapping is considered at the core of an analogical process is the so-called Heuristic-Driven Theory Projection proposed by Gust, Kühnberger and Schmid [GKS06]. While in SME and ACME the main algorithm boils down to finding a maximum common subgraph between the two domain representations, HDTP banks on a more formal approach. The two domains are formally described in first-order logic as a set of facts (variable-less formulas such as *TALL(Bob)*) and a set of laws (quantified formulas, such as $\exists x$,

[1] with the exception that the mapping goes here from T to S.

TALL(x)). Using an anti-unification (generalization) process, the two domains (or theories) S and T are mapped through a generalized theory G described in a second-order logic. As expected from the name of the framework, the way the mapping is performed is heuristic-based. From this generalized theory, a transfer of knowledge can be applied to the target domain, thus allowing the inference of new facts and laws, provided that they are in accordance with the already known formulas. This process of generalization followed by a transfer phase is what is called a **theory projection**.

1.1.3 Logical view of analogical reasoning

We have seen so far various models of analogical reasoning that mostly rely on some heuristic. In contrast with this tendency, Davies and Russel [DR87] proposed first order description of the analogical inference process, and most importantly a set of conditions required for this inference process to be sound. Concretely, they provide some sufficient conditions that must hold on two properties P and Q for the following inference rule to be valid:

$$\frac{\begin{array}{c} P(S) \wedge Q(S) \\ P(T) \end{array}}{Q(T)},$$

where S and T are the source and target objects. Naturally, this framework can perfectly be generalized with many properties $P_1, P_2, \cdots P_n$. It is clear that for now, this inference rule is not sound and the problem is now to add a premise such that the conclusion can be logically deduced.

A first obvious option would be to add as a premise the following implication:

$$\forall x, P(x) \implies Q(x).$$

But this is unsatisfactory, because in this case the inference simply reduces to

$$\frac{\begin{array}{c} P(T) \\ \forall x, P(x) \implies Q(x) \end{array}}{Q(T)},$$

where no use of the source S is made. It is clear that this inference cannot be reasonably considered as an analogy: the additional premise must not lead to a conclusion by only involving the target T. Some knowledge about the source S must be taken into account.

To solve this issue, Davies and Russel introduce what they call the **determination rule** i.e. the fact that the value of P determines that of Q:

$$(\forall x \; P(x) \implies Q(x)) \vee (\forall x \; P(x) \implies \neg Q(x)) \,.$$

This reads as *all P's are Q's, or none of them are*. More generally, this relation can be considered as being a functional dependency between P and Q, i.e. $Q = f(P)$. This determination rule has the two required properties: it allows a sound deduction of $Q(T)$, and forces to inspect the source S to rule out one of the two parts of the disjunction. In the end, the complete analogical inference principle of Davies and Russel can be described as:

$$\frac{\begin{array}{c} P(S) \wedge Q(S) \\ P(T) \\ (\forall x\ P(x) \implies Q(x)) \vee (\forall x\ P(x) \implies \neg Q(x)) \end{array}}{Q(T)}$$

It is clear that when we have $P(S) \wedge Q(S)$, the right disjunct $\forall x\ P(x) \implies \neg Q(x)$ cannot be true so the inference principle is still reduced to the simpler form mentioned above. The determination rule still has the merit of requiring to inspect the source S, at least in the first step of the process.

We have to mention here that the contribution of Davies and Russel is in fact a bit more sophisticated than the one we have described here for the sake of understanding. Indeed, the more complex determination rule that they propose is as follows:

$$\forall (y, z),\ [\exists x\ P(x, y) \wedge Q(x, z)] \implies [\forall x\ P(x, y) \implies Q(x, z)]\,.$$

If $P(x, y)$ expresses that the car x has the color yellow and if $Q(x, z)$ expresses that the car x has the price \$10,000, then the expression $[\exists x\ P(x, y) \wedge Q(x, z)] \implies [\forall x\ P(x, y) \implies Q(x, z)]$ means that if we can find a yellow car that costs \$10,000, then **all** yellow cars cost \$10,000. Because of the universal quantifier $\forall (y, z)$, this determination rule should have to be checked for every possible color, and every possible price. Unfortunately, this quickly becomes impractical because of the computational complexity, and because it is fairly unlikely to observe data satisfying these conditions.

1.2 Models with analogical proportions

In this section, we will describe models that have made explicit use of analogical proportions.

1.2.1 Solving geometric problems

The ANALOGY program of Thomas Evans [Eva64] is one of the pioneer works in the design of programs capable of analogical reasoning. ANALOGY, written in LISP[2], is able to solve analogical equations in the form of geometrical problems, such as that of Figure 1.1: given

[2] And according to its author, the largest LISP program at the time (1963)!

three geometrical patterns, choose the fourth among a list of candidates that leads to the best proportion.

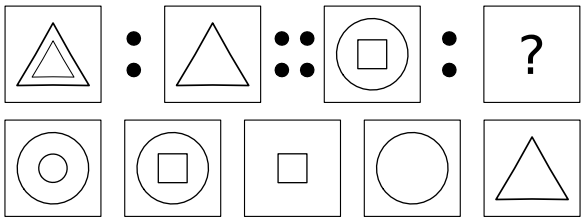

Figure 1.1: A geometrical analogy problem: figure A is to figure B as figure C is to which candidate?

The inputs to the program are rough, hand-built low-level descriptions of the figures. For example, simple geometric shape such as circles, rectangles or triangles are all described with one single primitive: *SIMPLE_CLOSED_CURVE(…)*, where the arguments are the start and end of the (multiple) lines, and their curvatures. The program is then divided into two distinct parts.

The role of the first part is to build high level representations of the figures from these raw descriptions. After identifying coherent shapes (such as triangle, square, etc.) as independent objects, the program tries to find relations of the kind *INSIDE(Obj1, Obj2)* or *ABOVE(Obj1, Obj3)* between figures. Finally, similarities between every pair of objects are computed. The similarity measure is based on the transformability of the first object into the second one, using traditional geometrical transformations (rotation, scale, reflections, etc.). All the information computed by the first part are given as input (on punched cards!) to the second part.

The second part relates to the analogical solving process per se. Using the output of the first part, ANALOGY will try to find a set of rules that transform figure A into figure B. A rule would for example state *REMOVE(Obj1), ROTATE(Obj2, 45°)*, etc. Then, by generalizing each rule, it tries to establish a correspondence between these rules and those transforming C into one of the candidate solutions. The chosen solution is the one that maximizes the resemblance between the two sets of rules.

An interesting fact is that ANALOGY analyses the way to go from A to B and applies it to go from C to D, but it does not make use of *central permutation*[3]: it may just as well analyze the way to go from A to C and transpose it to go from B to D. Note also that in ANALOGY the semantics behind the relations and the geometrical objects are not taken into account. In this respect, ANALOGY is close to Gentner's Structure Mapping Theory (but preexisting by far).

[3]The central permutation axiom, described in the next chapter, states that $A : B :: C : D \iff A : C :: B : D$.

1.2.2 Analogical reasoning as distance minimization on a vector space

At a time where most models of analogy used to take the form of complex computer programs (such as that of Evans), the two cognitive scientists David Rumelhart and Adele Abrahamsen proposed a simple theoretical model of analogical reasoning [RA05]. Their model is of great interest for us because as it will become clear, their view of analogy is in total accordance with our use of analogy in machine learning (or should we say more humbly that our use of analogy is fully compliant with their earlier model).

For Rumelhart and Abrahamsen, a human reasoning process can be defined by two components: a memory structure in which the remembered objects are stored, and an algorithm that manipulates these data to produce a result. In their paper, they define these two components for the analogical reasoning.

With regards to the memory structure, their assumption is that it can be considered as an m-dimensional Euclidean space. This assumption is originally that of Henley [Hen69] who showed, with the help of social experiments, that a set of 30 mammals could be fairly well represented in the 3-dimensional space $\mathbb{R}^3$ with the three axes *ferocity*, *humanness*, and *size*. It is supposed that the semantic similarity that people associate with two concepts is inversely proportional to their distance in the Euclidean space $\mathbb{R}^3$. For example, *cow* and *horse* should be fairly close, but *rat* would likely be far from both.

Rumelhart and Abrahamsen are interested in the problem of solving an analogical equation (although they never state it in these terms): given three concepts A, B, C and a set of candidate solutions $D_1, D_2, \cdots, D_n$, which D_i is the best candidate for $A : B :: C : D_i$? Their assumption is the following: **for a human agent, the best solution is the closest D_i to the (potentially hypothetical) *perfect* solution I, defined as $I = C - A + B$.** This principle is illustrated in figure 1.2. To assess the soundness of this hypothesis, diverse social experiments are led by the authors. For example, students are asked to choose the best candidates for various randomly generated analogies, e.g. rat is to pig as goat is to [chimpanzee, cow, rabbit, sheep].

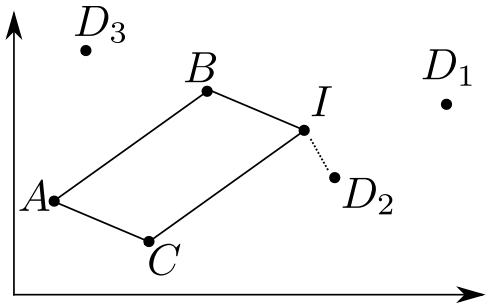

Figure 1.2: Analogical equation solving process as in [RA05]. The solution is here D_2.

As it will become clear, this idea that an analogy $A : B :: C : D$ is better when D is **close** (as measured by a distance on a metric space) to the true solution I is entirely compatible with

the notion of *analogical dissimilarity* that will be defined in Chapter 3.

More recently, various techniques have been proposed to generate a vector-space embedding for words from a large corpus of text documents, for example Word2Vect [MCCD13]. In these continuous spaces also, some interesting analogies between words can be found that give even more power to the works of Rumelhart and Abrahamsen. The most classical example of word-analogy in such a space is given by *man* is to *woman* as *king* is to *queen*, where the vector *queen* was found to be the one that is the closest to the vector *king* - *man* + *woman*, which is in perfect accordance to the idea of Rumelhart and Abrahamsen. Other meaningful analogies can be found, and some of them are illustrated in Figure 1.3. See [MYZ13] for more details on finding linguistic regularities in word space embeddings.

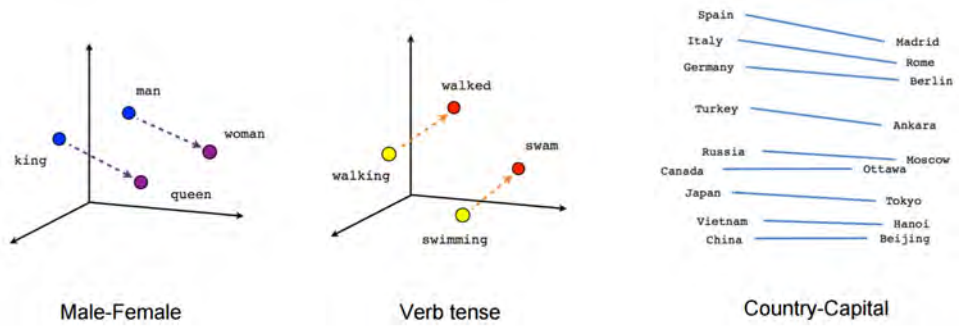

Figure 1.3: Analogies between words in the Word2Vect model. Image source: `https://www.tensorflow.org/versions/master/tutorials/word2vec`.

1.2.3 The Copycat program

Copycat [Mit93] (see also [HM$^+$94]) is another famous program that performs analogical reasoning tasks, introduced by Melanie Mitchell and Douglas Hofstadter. The problems considered by Copycat are the solving of analogical equations in a *microworld* of strings of letters. A typical Copycat problem looks as follows:

$$\mathbf{abc} : \mathbf{abd} :: \mathbf{ijk} : x.$$

What should the value of x be? Various answers may be relevant here, such as **ijd** (replace the right-most letter by **d**), but the most natural answer probably is **ijl**: replace the right-most letter by its successor. How about **zrq**? Well, not really convincing. The point of the authors, however, is that *a priori* every single option should be given equal chances of success.

This principle is strongly reflected in the Copycat program which is probabilistic in nature: for the same problem, various solutions can be found depending on the initial condition. For example the equation **abc** : **abd** :: **mrrjjj** : x leads to **mrrkkk** in 70% of the cases (over

1000 experiments), and to **mrjjk** 20% of the time. Other less plausible solutions make up the remaining 10%.

At the beginning of the program, each option is equally available. Then, on the basis of their validity, some hypotheses are given a stronger chance to *survive* till the end of the program, while others are discarded. Conceptually, this process emerges from the interoperability of three main components: the workspace, the codelets, and the temperature.

- The workspace, is the place where objects and relations live. In the workspace, diverse conceptual structures are built-in: *successor*, *predecessor*, *left-most*, *right-most*, *orientation*, etc. When the program starts, none of these structures are activated: this will be the role of the codelets.
- The codelets, are competing agents trying to explore and build perceptual structures and relations between the objects in the workspace. Their behaviour depends on the temperature.
- The temperature could be also defined as the entropy of the system. When relations in the workspace are strong and well established, the temperature is low. At the beginning of the program, the temperature is the highest, leading to a multitude of codelets being run in every single *direction*. This concept of temperature can be viewed as the trade-off between exploration (trying every possible hypothesis) and exploitation (the actual use of a well established hypothesis). When the temperature goes below a given threshold, the program stops and the current hypothesis is outputted.

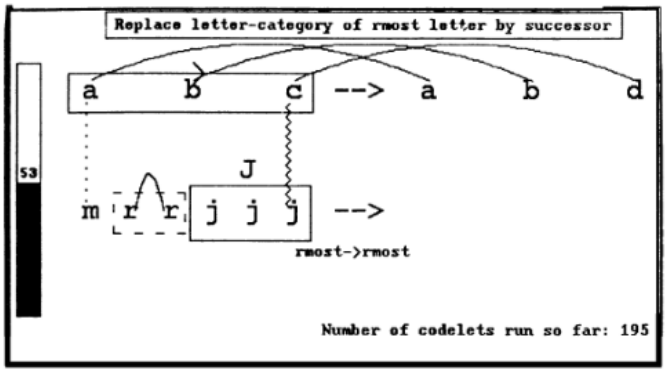

Figure 1.4: Snapshot of Copycat during an equation solving process. Image taken from [Mit01].

Figure 1.4 illustrates the internal state of Copycat during the solving of **abc** : **abd** :: **mrjjj** : x. 195 codelets have run so far, so the temperature is average and some structure have been properly established, such as the *successor group* **abc** and the *sameness group* **jjj**. The **rr** group is also being defined, though still weaker than the others at this stage. Also, a rule describing the change from **abc** to **abd** has been established: *replace the right-most letter by its successor*. Depending on the future behaviour of the codelets, this rule with either stick till the end of the program or change to lead to the most common prediction $x =$ **mrrkkk**.

Copycat is clearly a complex adaptative system where a global behaviour emerges from small, independent parts. According to its authors, *Copycat's architecture is neither symbolic nor connectionist, nor a hybrid of the two; rather, the program has a novel type of architecture situated somewhere in between these extremes.*

1.2.4 Analogy and the Minimum Description Length Principle

Ockham's razor (also Occam), due to the Franciscan philosopher William of Ockham (1285 - 1347), is a well known principle in machine learning theory. The main and most useful interpretation of the original Latin version states that when trying to explain a situation, if two hypotheses give the same answer then the best one is probably the **simplest** one. In practice, what makes an hypothesis simple remains quite vague, at least from a computational point of view. Yet this principle has been formalized into Rissanen's Minimum Description Length Principle (MDLP) [Ris78], which is based on Kolmogorov complexity. Despite the difficulty to build inferential models from this principle (Kolmogorov complexity is often intractable and impossible to compute, and can only be estimated), it has shown to be quite influential in the field of machine learning, at least from a theoretical point of view.

With this in mind, Antoine Cornuéjols proposed a framework for assessing the quality of an analogy [Cor96b] (see also [Cor96a]). In these papers, Cornuéjols hypothesizes that the best analogy between a source and a target model is the *simplest* one, meaning that its description length is minimal, in terms of Kolmogorov complexity.

Let us first recall some basic knowledge about Kolmogorov complexity, before diving into more technical details. The Kolmogorov complexity of a string of characters x, denoted $K(x)$, is the length of the shortest computer program capable of outputting x on a universal Turing machine. $K(x)$ is supposed to capture the intrinsic complexity of x. Intuitively, $K('aaaaabbbbb')$ is supposed to be lower than $K('abaabbabab')$, because a clear pattern emerges in the first string, leading to a simple program: first print a five times, then do the same for b. The second string seems more or less random, which makes it difficult to factorize into a concise program. In some sense, the Kolmogorov complexity captures how well a string x can be *compressed*. The conditional complexity $K(x \mid y)$ is the size of the shortest program that outputs x when given y as an input.

Now, let's get back to our analogical concerns. As illustrated in figure 1.5, Cornuéjols considers an analogy as a process involving an object x_S in a source domain, an object x_T in a target domain, and two functions f_S and f_T transforming x_S and x_T into y_S and y_T respectively: $y_S = f_S(x_S)$ and $y_T = f_T(x_T)$. Each domain S and T *lives* inside a theory or model (namely M_S and M_T), that can describe their corresponding objects.

For Cornuéjols, the best analogy is the one that minimizes the following sum of Kolmogorov

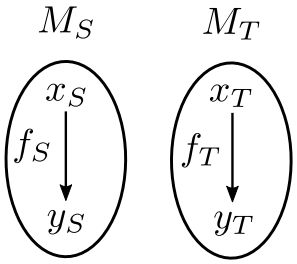

Figure 1.5: The two domains S and T in Cornuéjols' model.

complexities:

$$K(M_S) + K(x_S \mid M_S) + K(f_S \mid M_S) + K(M_T \mid M_S) + K(x_T \mid M_T) + K(f_T \mid M_T),$$

where:

- $K(M_S)$ is the cost associated with the source theory,
- $K(x_S \mid M_S)$ is the cost of x_S as described in the source theory,
- $K(f_S \mid M_S)$ is the cost of f_S as described in the source theory,
- $K(M_T \mid M_S)$ is the cost of describing the target theory from the source theory,
- $K(x_T \mid M_T)$ is the cost of x_T as described in the target theory,
- and finally $K(f_T \mid M_T)$ is the cost of f_T as described in the target theory.

Notice that the terms y_i are not considered in this cost function, because they are entirely defined by their corresponding x_i and f_i.

Cornuéjols illustrates the plausibility of his model using experiments in the microworld of Copycat. After defining the Kolmogorov complexity of the built-in relations (*direction*, *length*, etc.), and those of the representations of the inputs, the solutions of the analogical equations are set as those minimizing the above criterion. The empirical results show that this model, in addition to its theoretical appeal due to the proximity with MDLP, is at the very least an interesting option that deserves further investigations. In a recent work, this model of analogy has been applied to a task of transfer learning [MC16].

Adopting a different (but close) approach, the association between Kolmogorov complexity and the evaluation of the quality of an analogy has also been recently advocated in [BPR12]. In this work, the authors propose that if four objects a, b, c, d are in proportion, then it should be as *complicated* to go from a to b as to go from c to d, and it should also be as *complicated* to go from b to a as to go from d to c. More formally:

$$a : b :: c : d \implies K(b \mid a) = K(d \mid c) \text{ and } K(a \mid b) = K(c \mid d).$$

This criterion was used to automatically discriminate *good* analogies from *bad* analogies, where objects were concepts represented as words of the natural language, and where their Kolmogorov

complexity was roughly approximated using online search-engine queries. The experiments showed that the system was able to differentiate credible analogies from fishy ones in about 75% of the time.

1.2.5 Case-Based Reasoning

We will here describe the field of Case-Based Reasoning (CBR). First of all, let us note that CBR is not literally a computational model of analogy, but is rather a general problem solving paradigm that makes use of some particular form of analogical reasoning. As discussed in [AP94], CBR and analogical reasoning are often mixed up in the literature. We choose to briefly describe CBR now, because we need to clarify how our approach to problem solving will differ from that of CBR.

CBR is a very general paradigm for problem solving, where a new problem is solved using the knowledge of some similar past experiences. This model can be motivated by its psychological appeal: it is indeed well acknowledged that, when confronted with an unknown situation, humans tend to make use of their past knowledge about some other similar situation to solve the problem at hand. As an example, the recipe of a strawberry pie could be easily inferred using the past knowledge on the recipe of other fruit pies (e.g. apple pie, pear pie, etc.).

In its most general form, CBR is a way of reaching the unknown solution S to a problem P using the known solutions S_i to problems P_i that are **similar** to P. Each pair (S_i, P_i) is called a **case**, hence the name of the paradigm. The term *problem* is here used in its most general sense. In [AP94], a CBR process is described by the succession of the following four steps:

1. **Retrieve**: this first step is to retrieve all cases (S_i, P_i) where P_i is similar to the initial problem P. The way these cases are chosen is entirely dependent of the current setting, and greatly differ depending on the application. Sometimes, only a single case is selected.
2. **Reuse**: the solutions S_i associated to the most relevant P_i are aggregated in some way, and adapted if needed to build up the final solution S.
3. **Revise**: the relevance and the adequacy of the chosen solution S are evaluated, and S is modified if needed.
4. **Retain**: if the revised solution S ended up being a good choice, then the solution is retained and the pair (P, S) is bound to be used in the solving of some future problems.

Before moving further to the next chapter, we would like to point out a significant difference between the CBR approach and the one we will address in this document. In a way, the CBR process can be viewed as building the following analogical proportions:

$$P_i : S_i :: P : S,$$

where S is unknown and inferred from all the S_i. We can indeed consider that *problem* P_i *is to solution* S_i *as problem* P *is to solution* S. Each of these analogical proportions is built on the

basis that P_i and P are **similar**, so S should be similar to S_i.

But as we will see in the next chapter, an analogical proportion $a : b :: c : d$ is not restricted to having a close to c and b close to d! It is quite the contrary actually: all elements are allowed to be extremely distinct. And indeed, if we are able to *capture* how a given P_i (that is not close to P) differs from P, then applying the same transformation to S_i should give us a relevant potential solution S. Acknowledging this fact, our approach to analogical learning will rather follow another conceptual scheme: we will try to retrieve the proportions of the form $P_{i_1} : P_{i_2} :: P_{i_3} : P$, and the solution S will be inferred by *extrapolation*[4] from the associated equations: $S_{i_1} : S_{i_2} :: S_{i_3} : S$. This allows us to consider problems P_i that are both similar **and** different from P, which may allow to find more interesting solutions S.

Conclusion

This first chapter was devoted to the description of past works on the modeling of analogical reasoning. We chose here to distinguish between two groups of models: those that make explicit use of analogical proportions (either as a key component of the model or as a simple use-case example), and those that do not. Among the models that use analogical proportions, we will see that the model of Rumelhart and Abrahamsen (derived from a psychological perspective) will be in perfect accordance to our view of analogical classification, and analogical learning in general.

It is fairly clear that these past works are, with a few exceptions, mostly motivated by cognitive and psychological investigations. It is only recently that some authors [FPY95, LS96] have provided a formal framework to deal with analogical proportions in general algebraic settings, which opened the door to many computational applications. These formal analogical proportions are a key concept for our work. We will now review this topic in the next chapter.

Résumé du chapitre D'une façon générale, une analogie établit des parallèles entre deux situations différentes. Le raisonnement par analogie permet alors d'inférer des propriétés sur l'une des situations, en utilisant nos connaissances de l'autre situation. Un type particulier de raisonnement par analogie est celui lié aux proportions analogiques, qui sont des expressions de la forme *a est à b ce que c est à d*, souvent écrit $a : b :: c : d$. Là aussi, on peut parfois déduire la valeur d'un des quatre éléments sur la base de notre connaissance des trois premiers. Ce processus s'appelle la *résolution d'équation analogique*.

Les deux premiers chapitres de ce document seront dédiés à fournir les prérequis nécessaires concernant les modèles de raisonnement par analogie, et en particulier ceux relatifs aux proportions analogiques. Dans ce premier chapitre nous décrivons brièvement différentes tentatives de

[4]The way S is *extrapolated* will be described in the next chapter as the **solving** of the analogical equation.

formaliser le raisonnement par analogie, accordant une attention particulière aux modèles qui peuvent s'utiliser à des fins inférentielles.

Nous avons choisi ici de faire la distinction entre deux types de modèles : ceux qui ne font jamais référence aux proportions analogiques, et ceux qui recourent aux proportions analogiques d'une manière ou d'une autre : les proportions sont alors soit une composante clef du modèle (par exemple sections 1.2.1 et 1.2.2), soit une simple possibilité d'application (sections 1.2.3 et 1.2.4).

Ces différentes propositions sont, en grande partie, inspirées par des travaux provenant des sciences cognitives et psychologiques. Ce n'est qu'assez récemment que certains auteurs [FPY95, LS96] ont proposé divers modèles formels permettant l'usage des proportions analogiques au sein de structures algébriques. Ces proportions analogiques sont le principal objet de nos travaux, et nous les décrivons en détail dans le prochain chapitre.

Chapter 2

Formal analogical proportions

Content

After a first introduction of past works on analogical reasoning in Chapter 1, we will now provide all the necessary background on formal analogical proportions, the most important concept of this document.

An **analogical proportion** is a statement of the form "a is to b as c is to d". It is a relation between the two pairs (a, b) and (c, d), which expresses that what is common to a and b is also common to c and d, and what is different between a and b is also different between c and d. There are numerous examples of such statements, with which everybody will more or less

agree, such as "a calf is to a cow as a foal is to a mare", or "Paris is to France as Berlin is to Germany". As a relation involving both similarities **and** dissimilarities between four objects, an analogical proportion is able to model complex associations.

The three axioms of analogical proportions

It has been agreed since Aristotle time that an analogical proportion A is a quaternary relation satisfying the **three following axioms**, for any a, b, c, d:

1. $A(a, b, a, b)$ always holds (Reflexivity)
2. $A(a, b, c, d) \implies A(c, d, a, b)$ (Symmetry)
3. $A(a, b, c, d) \implies A(a, c, b, d)$ (Central permutation)

When there is no ambiguity over A and its domain, the infix notation $a : b :: c : d$ is often used. Considering again the farm example, the symmetry axiom states that if a calf (a) is to a cow (b) as a foal (c) is to a mare (d), then a foal (c) is to a mare (d) as a calf (a) is to a cow (b), which seems perfectly sound. The central permutation axiom leads to the natural consequence that a calf (a) is to a foal (c) as a cow (b) is to a mare (d).

This chapter is structured as follows. In the first section of this chapter, we will provide the definition of analogical proportions in various settings, with a strong emphasis on the arithmetic and Boolean proportions, which will turn out to be very close. We will further investigate this bond in the second section, which will be devoted to describe these two proportions from a geometrical perspective. In the last section, we will consider a toy classification problem, which will help us introduce two other key concepts: the **analogical inference principle**, and the process of **analogical equation solving**. This last section will also serve as a transition towards the next chapter, which will be devoted to describe our contributions to analogical classification.

2.1 Formal definitions

The goal of this first section is to provide the definitions of analogical proportions in various settings. We will particularly focus on the arithmetic proportion that deals with real numbers, and on the Boolean proportion.

The use and definition of formal analogical proportions is not new and dates back[1] to works such as those of Federici, Pirelli and Yvon [FPY95, PY99], who proposed definitions of proportions between strings and between symbolic features (thus implicitly providing a definition for the Boolean proportion). Almost in parallel, the works of Lepage [LS96, Lep98] explore similar concerns with a distance-based definition of analogy between words.

[1] We ignore for now the works of Klein and Piaget, which will be describe later in Section 2.1.3.

2.1.1 The geometric and arithmetic proportions

A factorization-based definition of analogical proportions

We will start here by a general factorization-based definition of analogical proportions in semigroups, provided in [SY05a] and [SY05b]:

Definition 2.1 (Proportion in semigroups)**.** Let $(U, \oplus)$ be a semigroup, i.e. U is a set and $\oplus$ is an associative binary operation. Four elements $a, b, c, d \in U$, are in proportion if there exists some factorization

$$\begin{aligned} a &= a_1 \oplus a_2 \oplus \cdots \oplus a_n \\ b &= b_1 \oplus b_2 \oplus \cdots \oplus b_n \\ c &= c_1 \oplus c_2 \oplus \cdots \oplus c_n \\ d &= d_1 \oplus d_2 \oplus \cdots \oplus d_n, \end{aligned}$$

such that for all $i \in [1, n]$,

$$\begin{cases} a_i = b_i \text{ and } c_i = d_i \\ \text{or} \\ a_i = c_i \text{ and } b_i = d_i. \end{cases}$$

> **Example:**
> A use of this definition can be found in [SY05b], where the authors define an analogy between words of a finite alphabet using the concatenation operator, which allows to build proportions such as:
>
> eat : eating :: look : looking.
>
> This analogy was used in an automatic verb conjugation task that we will briefly describe in the next chapter.

Analogical proportions between real numbers

To further illustrate the use of Definition 2.1, it will be insightful to instantiate U as the set of natural number $\mathbb{N}$ and $\oplus$ as the usual multiplication $\times$.

> **Example:**
> In this setting, let's consider the prime factorization of:

$$a = 30 = 1 \times 2 \times 3 \times 5$$
$$b = 60 = 2 \times 2 \times 3 \times 5$$
$$c = 25 = 1 \times 1 \times 5 \times 5$$
$$d = 50 = 2 \times 1 \times 5 \times 5.$$

For each i, we either have $a_i = b_i$ and $c_i = d_i$ $(i = 2, 3, 4)$ or $a_i = c_i$ and $b_i = d_i$ $(i = 1, 4)$. We can then say that a, b, c, d are in proportion, i.e. $30 : 60 :: 25 : 50$. We recognize here the **geometric proportion**: $\frac{30}{60} = \frac{25}{50}$.

Definition 2.2 (Geometric proportion)**.** Four elements a, b, c, d in $\mathbb{R}^*$ are in **geometric proportion** if

$$\frac{a}{b} = \frac{c}{d},$$

or equivalently if

$$a \times d = b \times c.$$

It now becomes clear why analogical proportions are effectively called *proportions*: it is precisely because they generalize the well-known numerical geometric proportion to more complex structures. Actually, Aristotle stated the three aforementioned axioms on the basis of the geometric proportion.

In fact, when $\mathbb{R}$ is equipped with the multiplication $\times$, the pair $(\mathbb{R}, \times)$ is a group (i.e. a set equipped with an operation that satisfies closure, associativity, identity and invertibility). When dealing with a group (instead of a semigroup), Definition 2.1 can be greatly simplified:

Proposition 2.1 (Proportion in groups)**.** *Let $(U, \oplus)$ be a group. Four elements $a, b, c, d \in U$, are in proportion if:*

$$a \oplus d = b \oplus c.$$

The geometric proportion is clearly a direct instance of this definition. Another proportion can be defined over real numbers, using this time the addition operator $+$ for the group $(\mathbb{R}, +)$:

Definition 2.3 (Arithmetic proportion)**.** Four elements a, b, c, d in $\mathbb{R}$ are in **arithmetic proportion** if:

$$a + d = b + c,$$

or equivalently if

$$a - b = c - d.$$

Analogical proportions over vectors

An analogical proportion in $\mathbb{R}$ is great, but what will be more useful is a proportion in $\mathbb{R}^m$, so

that we can deal with vectors. Let us make a brief notation note: in the rest of this document, vectors will be denoted by boldface letters and their components will be denoted by regular letters. For example, a vector $\mathbf{x} \in X^m$ will be defined as $\mathbf{x} = (x_1, x_2, \ldots, x_m)$.

It is actually fairly easy to define a proportion in a given set X^m provided that we already have a proportion in X. Indeed, a natural extension to X^m is to require that each of the m dimensions make up valid proportions in X:

Definition 2.4 (Proportion for vectors)**.** Let A be an analogy over a set X. We can define an analogy A^m over X^m in a component-wise fashion by:

$$A^m(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}) \text{ if } A(a_i, b_i, c_i, d_i) \text{ for all } i \in [1, m],$$

where $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}$ are vectors in X^m. More often than not, A^m will simply be denoted A for the sake of brevity.

Thanks to Definition 2.4, we now have the definition of two numerical proportions in $\mathbb{R}^m$: one that generalizes the geometric proportion and one that generalizes the arithmetic proportion. The interpretation of the geometric proportion in $\mathbb{R}^m$ can be opaque, but that of the arithmetic proportion is very clear: four vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}$ of $\mathbb{R}^m$ are in arithmetic proportion if $\mathbf{a}-\mathbf{b} = \mathbf{c}-\mathbf{d}$, i.e. **if they are the four vertices of a parallelogram**, as illustrated in Figure 2.1.

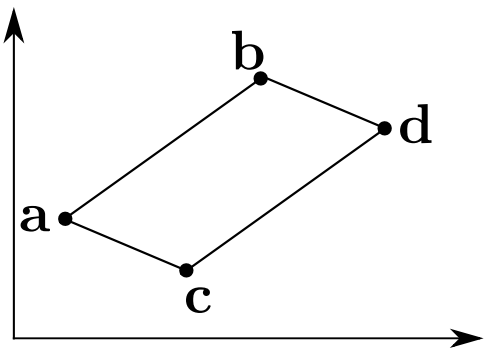

Figure 2.1: $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}$ are in arithmetic proportion iff they are the four vertices of a parallelogram.

Let us mention that even if they did not explicitly mention the parallelogram view of analogy, Rumelhart and Abrahamsen (Section 1.2.2) clearly used the arithmetic proportion for their psychological model of analogical reasoning.

Starting from the general lines of Definition 2.1, other proportions have been defined in various domains, e.g. analogies between trees, lattices, or matrices [MD04, SY05b, MBD08]. However, we will not detail them further here and rather focus on other domains of interest, namely analogies between sets. Defining analogies between sets will indeed allow us to derive the definition of an analogical proportion for Boolean values, which is the main subject of this document.

2.1.2 Analogical proportions between sets

In [Lep03], Lepage defines an analogy between sets as follows: four subsets A, B, C, D of a universal set X are in proportion if A can be transformed into B by adding and deleting the same elements as to transform C into D. A more formal definition has been given in [SY05b]:

Definition 2.5 (Proportion between sets (i))**.** Let $A, B, C, D \subseteq X$. A, B, C, D are in proportion if there exist four subsets U, V, W and Z (not necessarily disjoint) such that:

$$\begin{cases} A = U \cup V \\ B = U \cup W \\ C = Z \cup V \\ D = Z \cup W. \end{cases}$$

The requirements of Lepage are clearly respected here: to go from A to B, one needs to add W and remove V. To go from C to D, one needs to do the exact same thing: add W and remove V. This definition is completely compliant with our informal introduction at the beginning of this chapter, where we explained that an analogical proportion $a : b :: c : d$ expresses that *what is common to a and b is also common to c and d, and what is different between a and b is also different between c and d.*

An equivalent definition of analogy between sets has been given in [MP09]:

Definition 2.6 (Proportion between sets (ii))**.** Let $A, B, C, D \subseteq X$. A, B, C, D are in proportion if:

$$A \setminus B = C \setminus D \text{ and } B \setminus A = D \setminus C.$$

Definition 2.6 reads as *A differs from B as C differs from D, and B differs from A as D differs from C*, which is naturally equivalent to the statement of Definition 2.5: $A \setminus B = C \setminus D = V$, and $B \setminus A = D \setminus A = W$. Finally, a last equivalent definition can be given as follows (not without using various ingenious tweaks):

Definition 2.7 (Proportion between sets (iii))**.** Let $A, B, C, D \subseteq X$. A, B, C, D are in proportion if:

$$A \cap D = B \cap C \text{ and } A \cup D = B \cup C.$$

These three equivalent definitions are illustrated in Figure 2.2.

Example:
For a more concrete example, let us consider $A = \{a, b, c, g\}, B = \{a, b, d, e, g\}, C = \{c, f, g\}$ and $D = \{d, e, f, g\}$, as summed-up in Table 2.1. Setting $U = \{a, b, g\}, V = \{c\}, W = \{d, e\}$ and $Z = \{f, g\}$, the conditions of Definition 2.5 are clearly satisfied. Also, $A \setminus B = C \setminus D = \{c\} = V$, and $B \setminus A = D \setminus C = \{d, e\} = W$. Finally, $A \cup D = B \cup C = \{a, b, c, d, e, f, g\}$ and $A \cap D = B \cap C = \{g\}$.

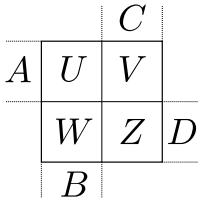

Figure 2.2: The four sets A, B, C, D are in proportion. In general, U, V, W, Z do not need to be disjoint.

	a	b	c	d	e	f	g
A	×	×	×				×
B	×	×		×	×		×
C			×			×	×
D				×	×	×	×

Table 2.1: Four sets A, B, C, D in analogical proportion.

Analogies between sets are important because they allow us to derive the definition of the Boolean proportion, which is the object of the next subsection.

2.1.3 The Boolean proportion

Formal definition

We are now in a position to give a formal definition of the Boolean proportion, which can be directly derived from Definitions 2.6 or 2.7 by considering the set $\mathbb{B} = \{0, 1\}$ [MP09]:

Definition 2.8 (Boolean proportion (i))**.** Four elements a, b, c, d in $\mathbb{B} = \{0, 1\}$ are in **Boolean proportion** if

$$[(a \wedge \neg b) \leftrightarrow (c \wedge \neg d)] \wedge [(\neg a \wedge b) \leftrightarrow (\neg c \wedge d)]\,, \text{ or equivalently}$$
$$(a \wedge d \leftrightarrow b \wedge c) \wedge (a \vee d \leftrightarrow b \vee c),$$

where $\leftrightarrow$ stands for the equivalence connective.

The first expression is the direct counterpart of Definition 2.6, and states that a **differs from** b **as** c **differs from** d**, and conversely** b **differs from** a **as** d **differs from** c. The second expression directly comes from Definition 2.7.

In a Boolean setting, there are exactly $2^4 = 16$ different valuations of a, b, c, d. Definition 2.8 tells us that exactly 6 of these valuations (or **patterns**) lead to valid Boolean proportions. Table 2.2 illustrates the 6 valid patterns and the 10 invalid ones. We recognize in Table 2.2 that all the valid patterns express that we need to perform the same operation to go from a to b as

a	b	c	d	$A(a,b,c,d)$
0	0	0	0	**1**
1	1	1	1	**1**
0	0	1	1	**1**
1	1	0	0	**1**
0	1	0	1	**1**
1	0	1	0	**1**

a	b	c	d	$A(a,b,c,d)$
0	0	0	1	**0**
0	0	1	0	**0**
0	1	0	0	**0**
1	0	0	0	**0**
1	1	1	0	**0**
1	1	0	1	**0**
1	0	1	1	**0**
0	1	1	1	**0**
0	1	1	0	**0**
1	0	0	1	**0**

Table 2.2: The six valid patterns of the Boolean proportion (left), and the ten invalid patterns (right).

to go from c to d, and that this is not the case for the invalid patterns. This is compliant with what we expect from an analogical proportion, as discussed earlier. With regards to this, the last two invalid patterns (namely $0:1::1:0$ and its counterpart $1:0:0:1$) have a particular status: we could indeed claim that they express the fact that b is the negation of a just like c is the negation of d, so in some sense we **still** could say that a differs from b as c differs from d. We will discuss this point further when we will mention the works of Sheldon Klein, who came up with a different definition of the Boolean analogical proportion.

We observe from Table 2.2 that some sort of *code independence axiom* is satisfied, which guarantees that 0 and 1 play symmetric roles:

Property 2.1. *Let a, b, c, d in $\mathbb{B}$. Then*

$$a:b::c:d \iff \neg a:\neg b::\neg c:\neg d,$$

where $\neg x$ is the negation of x.

This code independence property is actually extremely important for representational purposes, because it means that we do not have to care about the coding convention of the Boolean vectors that we use [PR13b].

We gave the definition of the Boolean proportion as a consequence of the definition of proportions between sets, but let us note that in fact, the 6 valid patterns of Table 2.2 could have been derived directly from the 3 axioms of an analogical proportion: the first axiom tells us that the proportion $a:b::a:b$ holds for any value of a and b in $\mathbb{B}$. This means that the four following proportions are valid:

- $0:1::0:1$;
- $1:0::1:0$;

- $0 : 0 :: 0 : 0$;
- $1 : 1 :: 1 : 1$.

Using the central permutation axiom ($a : b :: c : d \iff a : c :: b : d$), we can then derive two additional proportions, which makes a total of 6 proportions that are indeed those of Table 2.2:

- $1 : 1 :: 0 : 0$;
- $0 : 0 :: 1 : 1$.

The Boolean proportion as a particular case of the arithmetic proportion

By paying attention to Table 2.2, we notice that the Boolean proportion can actually be defined in a simpler way:

Definition 2.9 (Boolean proportion (ii))**.** Four elements a, b, c, d in $\mathbb{B}$ are in **Boolean proportion** if:

$$\begin{cases} a = b \text{ and } c = d \\ \text{or} \\ a = c \text{ and } b = d, \end{cases}$$

or equivalently if

$$a - b = c - d.$$

Naturally, the values $a - b$ and $c - d$ do not necessarily belong to $\mathbb{B}$, but it does not really matter: we just have to temporarily consider that a, b, c, d are in $\mathbb{R}$. We are here witnessing a close bond that links the Boolean proportion and the arithmetic proportion: **when restricted to $\mathbb{B}$, the arithmetic proportion and the Boolean proportion are equivalent**. The connections between these two proportions will be more and more obvious as we go through this document.

If the Boolean and the arithmetic proportion are equivalent in $\mathbb{B}$, the geometric proportion only offers a necessary condition to define the Boolean proportion: $a \times d = b \times c$ is satisfied by all of the valid patterns in Table 2.2, but also for the pattern $0 : 0 : 0 : 1$ which is not a valid analogy. In that sense, the Boolean proportion is closer to the arithmetic proportion than to the geometric proportion.

The Boolean proportion of Sheldon Klein

We cannot talk about formal Boolean proportions without going back to the 80s and mention the pioneering work of Sheldon Klein who, to some extent, defined his *own* Boolean analogical proportion [Kle83]. His definition states that a, b, c, d in $\mathbb{B}$ are in proportion if $a \oplus b = c \oplus d$, where $\oplus$ is the XOR operator. This definition is less restrictive than Definition 2.8, and amounts to considering that in addition to the 6 valid patterns from Table 2.2, two other patterns are also valid, namely $0 : 1 :: 1 : 0$ and its counterpart $1 : 0 :: 0 : 1$.

These two patterns seem appealing at first sight, because they capture the fact that $a : \neg a :: b : \neg b$. However, they also lead to the fact that $a : b :: c : d \iff b : a :: c : d$, which seems quite unnatural for an analogy. Indeed, if a calf (a) is to a cow (b) as a foal (c) is to a horse (d), it seems unnatural to say that a cow (b) is to a calf (a) as a foal (c) is to a horse (d). This notion will become clear in the next section when we will present the equivalence classes of analogical proportion (here, (a, b, c, d) and (b, a, c, d) are not in the same equivalence class). In practice, it is often observed that using the modeling of Klein leads to less accurate classifiers than using the standard modeling.

In fact, Klein's definition of the Boolean proportion $a : b :: c : d \iff a \oplus b = c \oplus d$ is equivalent to $a \oplus d = b \oplus c$, due to the singular properties of the XOR operator. This definition is actually a direct instance of Proposition 2.1 (analogies in groups), which is natural because of the fact that when the set $B = \{ 0, 1 \}$ is equipped with the XOR operator, we obtain a group.

The logical proportion of Jean Piaget and its generalization

Another remarkable precursor work is that of Jean Piaget who, in the annex of a French book from 1952 [Pia52], defined what he called a *logical proportion* as follows: four propositions a, b, c, d are in proportion if $(a \wedge d \leftrightarrow b \wedge c) \wedge (a \vee d \leftrightarrow b \vee c)$. This definition is clearly equivalent to what we refer to now as the Boolean **analogical** proportion, and was likely derived by Piaget when he was trying to give a Boolean counterpart to the geometric proportion, where the exchange of extreme and means plays a key role. In fact, it turns out that Piaget never made any reference to analogy in this work.

This concept of **logical proportion** has been generalized in recent works (see e.g. [PR13b]). A logical proportion is here defined as a pair of equivalences of the form:

$$I_1(a, b) \leftrightarrow I_2(c, d) \quad \wedge \quad I_3(a, b) \leftrightarrow I_4(c, d),$$

where a, b, c, d are Boolean variables and each I_i is either an **indicator of similarity** or an **indicator of dissimilarity**. An indicator of similarity $I(x, y)$ is either of the form $x \wedge y$ or of the form $\neg x \wedge \neg y$: it captures what is common to x and y, or what is both not in x and not in y. Conversely, a dissimilarity indicator $I(x, y)$ will be of the form $\neg x \wedge y$ or $x \wedge \neg y$, and captures what is in y and not in x, or what is in x and not in y. A logical proportion then expresses a semantic equivalence between the two pairs of variables (a, b) and (c, d), in terms of similarities and/or dissimilarities. The Boolean analogical proportion that we are studying here is a particular instance of logical proportion.

There is a total 120 possible logical proportions (see [PR12] for a derivation), and quite interestingly each of these logical proportions are true for exactly 6 valuations of a, b, c, d. This makes them quite rare, because there are in fact $\binom{16}{6} = 8008$ propositions that are true for 6 valuations only, out of the 16 possible valuations. Out of these 120 logical proportions, only 8 of them support the code independency property (Property 2.1) which makes the remaining

112 proportions unsuitable for representational purposes.

Out of these 8 code-independent proportions, 4 proportions exhibit interesting cognitive properties, that we name the **homogeneous** proportions. These proportions are characterized by the fact that all the expressions $I_i(x, y)$ are different, and by the fact that I_1 and I_2 are of the same kind (i.e. they are both similarity indicators, or both dissimilarity indicators), and the same goes for I_3 and I_4. These four proportions are called the paralogy, the reverse analogy, the inverse paralogy, and naturally the **analogy**. We will however not make use of these logical proportions in this document (except the Boolean analogical proportion of course!), so we will not detail them further.

Extension to multiple-valued logics

In [PR13a], the authors propose an extension of the Boolean proportion to a multiple-valued logic where the elements can take real values in the interval $[0, 1]$. This extension was proposed on the basis of the definition $A(a, b, c, d) \iff [(a \wedge \neg b) \leftrightarrow (c \wedge \neg d)] \wedge [(\neg a \wedge b) \leftrightarrow (\neg c \wedge d)]$, and using the following mapping between operators:

- The central $\wedge$ becomes the min operator.
- $x \leftrightarrow y$ becomes $1 - |x - y|$.
- $x \wedge \neg y$ becomes $\max(0, x - y)$.

The graded view of the analogical proportion then becomes:

$$A(a, b, c, d) = \begin{cases} 1 - |(a - b) - (c - d)| & \text{if } a \geq b \text{ and } c \geq d, \text{ or } a \leq b \text{ and } c \leq d \\ 1 - \max(|a - b|, |c - d|) & \text{if } a \geq b \text{ and } c \leq d, \text{ or } a \leq b \text{ and } c \geq d. \end{cases}$$

The conditions may seem complicated but they actually express simple facts: the first one states that a differs from b in the same direction as c differs from d, and the second one states the reverse. It is clear that this definition is quite closely linked to that of the arithmetic proportion.

Let us note that there are other ways to transform the Boolean operators into multiple-valued logic operators. If we had started from a different expression of the Boolean proportion, or if we had used alternative choices of operators, we would have ended up with a different potential definition of this analogical proportion. However, some combinations lead to definitions that do not completely suit the requirements that one can expect from an analogical proportion. So in practice, the choice of operators is dictated by the plausibility of the final expression (see [PR13a] for a discussion). We here chose to describe this definition because of its links to the arithmetic proportion, and because we will make use of this graded view of analogy later in Chapter 5.

Conclusion

This first section was devoted to the description of various definitions of analogical proportions in different settings: we have reviewed analogies between real numbers, between sets, between

Boolean numbers, and an extension to multiple-valued logics. In this document, the two kinds of proportions that we will deal with are the Boolean proportion and the arithmetic proportion, which we have shown to be extremely close. As having a deep understanding of how these two proportions work will be very insightful for us, the next section is devoted to give a geometrical interpretation of the Boolean and arithmetic proportions.

2.2 Geometrical insights of the Boolean and arithmetic proportions

The goal of this section is to demystify the Boolean and arithmetic proportions, which will be used in all of the investigations described in this document. To this end, we shall provide geometrical insights of how these proportions connect the four elements a, b, c, d. We have already seen in the previous section that the arithmetic proportion is a relation between four elements that build up the vertices of a parallelogram. In this section, we will show that this parallelogram-based view of analogical proportions can be extended to the Boolean setting.

Before moving on to the geometric insights, we will first make a quick detour in the first subsection, and describe what we call the **equivalence classes** of analogical proportions. We have chosen to introduce these general concepts here, because they are best illustrated by examples from the arithmetic and Boolean proportions, and because they are paramount to properly understand the geometrical properties of these proportions.

2.2.1 Equivalence classes of analogical proportions

We place ourselves here in a very general setting, and consider four elements of a given set X that are in proportion. Starting from the assertion $A(a, b, c, d)$ and by successive application of the symmetry and central permutation axioms[2], we arrive to the following 8 equivalent proportions:

$$
\begin{aligned}
& A(a,b,c,d) \\
\iff & A(c,d,a,b) && \text{Symmetry} \\
\iff & A(c,a,d,b) && \text{Central permutation} \\
\iff & A(d,b,c,a) && \text{...} \\
\iff & A(d,c,b,a) \\
\iff & A(b,a,d,c) \\
\iff & A(b,d,a,c) \\
\iff & A(a,c,b,d).
\end{aligned}
$$

[2]Recall that the symmetry axiom states that $A(a,b,c,d) \iff A(c,d,a,b)$, and the central permutation axiom states that $A(a,b,c,d) \iff A(a,c,b,d)$.

If we applied one more time the central permutation axiom, we would end up with our initial proportion $A(a,b,c,d)$: we say that these 8 proportions constitute an **equivalence class**. There are exactly $4! = 24$ ways to order the four elements a, b, c, d, and we have just seen that 8 of these orderings form an equivalence class. In fact, the 16 remaining orderings are also part of two other equivalence classes, each of 8 proportions. Indeed, if we had started with the proportion $A(a,b,d,c)$, by successive applications of the central permutation and symmetry axiom, we would have ended up again with 8 equivalent forms that are all different from those that are *generated* by $A(a,b,c,d)$. The third equivalence class is generated by $A(a,c,d,b)$. In total, we end up with $24 = 3 \times 8$ distinct proportions, which correspond to the $24 = 4!$ possible orderings of a, b, c, d.

Example:
The point here that if we say that a calf (a) is to a cow (b) as a foal (c) is to a mare (d) ($A(a,b,c,d)$), then we can actually say it in 7 other equivalent ways. Also, we **cannot** say that a calf (a) is to a cow (b) as a mare (d) is to a foal (c) (second equivalence class $A(a,b,d,c)$), because such a statement would implicitly express that *a young is to an adult as an adult is to a young*, which seems wrong: we cannot go from a *young* to an *adult* in the same way as we go from an *adult* to *young*. We **cannot** say either that a calf (a) is to a foal (c) as a mare (d) is to a cow (b) (third equivalence class $A(a,c,d,b)$), because this would suggest that *a bovine is to an equid as an equid is to a bovine*: here again, we cannot transform the concept *bovine* into *equid* in the same way as we would transform the concept *equid* into *bovine*.

Another illustration of the three classes of proportions is illustrated in Figure 2.3, using the arithmetic proportion.

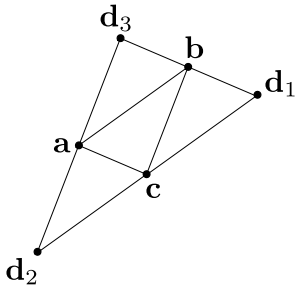

Figure 2.3: The three equivalence classes: $A(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}_1)$, $A(\mathbf{a},\mathbf{b},\mathbf{d}_2,\mathbf{c})$ and $A(\mathbf{a},\mathbf{c},\mathbf{d}_3,\mathbf{b})$. See also [PR13b].

Let us mention an important point: if a, b, c, d are not all distinct, the equivalence classes contain less than 8 proportions. This is natural, because there are no longer 24 different orderings of a, b, c, d. For example if $a = c$ and $b = d$, then the equivalence class that is generated by $A(a,b,c,d)$ is only made of 4 proportions:

$$A(a,b,c,d) \stackrel{\text{def}}{=} A(a,b,a,b) \iff A(a,a,b,b) \iff A(b,b,a,a) \iff (b,a,b,a).$$

Although these notions may seem shallow at first sight, they will actually turn out to be paramount to grasp the geometrical view of Boolean proportions that we now address.

2.2.2 The Boolean and arithmetic proportions as parallelograms

What we know so far is that the arithmetic proportion in $\mathbb{R}^m$ involves the four vertices of a parallelogram. As for the Boolean proportion, we know from Definition 2.4 that four Boolean vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}$ are in proportion if each component build up a valid proportion, i.e. that each of the (a_i, b_i, c_i, d_i) follows one of the 6 valid patterns of Table 2.2 on page 34.

Example:
For example in $\mathbb{B}^2$, the four vectors $\mathbf{a} = (0,0),\ \mathbf{b} = (1,0),\ \mathbf{c} = (0,1)$ and $\mathbf{d} = (1,1)$ are in proportion because the two component-wise proportions $a_1 : b_1 :: c_1 : d_1$ and $a_2 : b_2 :: c_2 : d_2$ are valid: we have $0 : 1 :: 0 : 1$ and $0 : 0 :: 1 : 1$.

The four vectors $\mathbf{a} = (0,0),\ \mathbf{b} = (1,0),\ \mathbf{c} = (0,1)$ and $\mathbf{d} = (1,0)$ are **not** in proportion because $a_2 : b_2 :: c_2 : d_2$ is not a valid pattern.

We have also seen that in $\mathbb{B}$, the Boolean proportion and the arithmetic proportion are equivalent. This is also naturally the case in $\mathbb{B}^m$! This means that a proportion in $\mathbb{B}^m$ can also be represented as a (potentially *flat*) parallelogram. By *flat* parallelogram we mean that the four vertices are not all distinct: this will all become clear by looking at the two following illustrations, in $\mathbb{B}^2$ and in $\mathbb{B}^3$.

The 36 proportions of $\mathbb{B}^2$

Figure 2.4 illustrates the proportions that one can build in $\mathbb{B}^2$:

- The proportion $\mathbf{a} : \mathbf{b} :: \mathbf{c} : \mathbf{d}$ and its 7 other equivalent forms, making up the parallelogram **abcd** (remember that when all elements are distinct, the equivalence class represented by $A(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d})$ is composed of 8 elements).
- The proportions that we can build using any pair of vertices, for example $\mathbf{a} : \mathbf{d} :: \mathbf{a} : \mathbf{d}$. Each of these proportions has 3 other equivalent forms: in this case $\mathbf{a} : \mathbf{a} :: \mathbf{d} : \mathbf{d}$, $\mathbf{d} : \mathbf{a} :: \mathbf{d} : \mathbf{a}$ and $\mathbf{d} : \mathbf{d} :: \mathbf{a} : \mathbf{a}$. As there are $\binom{4}{2} = 6$ pairs of vertices in $\mathbb{B}^2$, there are exactly 6×4 proportions of this kind, all of which are flat[3].
- The 4 proportions involving each vertex independently, for example $\mathbf{b} : \mathbf{b} :: \mathbf{b} : \mathbf{b}$. These proportions are also flat, naturally.

All in all, this makes up to $36 = 8 + 6 \times 4 + 4$ proportions: 8 proportions from the parallelogram **abcd**, 6×4 proportions from the pairs of vertices, and 4 proportions from the 4 vertices. In fact, this number of 36 proportions could have been derived directly by remembering that in $\mathbb{B}^2$ the proportions are defined in a component-wise fashion, and that there are exactly 6 valid

[3]The *flat* proportion are those that make up flat parallelograms. This naming convention is actually fortunate, because these proportions are in practice absolutely useless, as it will soon become clear in the next section.

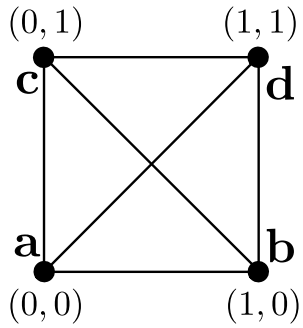

Figure 2.4: The 36 proportions in $\mathbb{B}^2$.

patterns of proportions in $\mathbb{B}$: as a proportion in $\mathbb{B}^2$ is the concatenation of two proportions in $\mathbb{B}$, the number of valid proportions in $\mathbb{B}^2$ necessarily is $6 \times 6 = 36$.

Let us also note though that only $1 + 6 + 4 = 11$ of them can be considered *unique* (up to equivalence) and only one is non-flat. In the next section, we will further investigate the number of unique proportions that can be built in $\mathbb{B}^m$.

The proportions of $\mathbb{B}^3$

Going a dimension further can still be insightful. Figure 2.5 illustrates the 12 non-flat proportions that exist in $\mathbb{B}^3$. These proportions are the parallelograms making up the 6 faces of the 3-cube, and six other *diagonal* parallelograms passing through the center of the cube. The flat proportions are not shown for the sake of clarity.

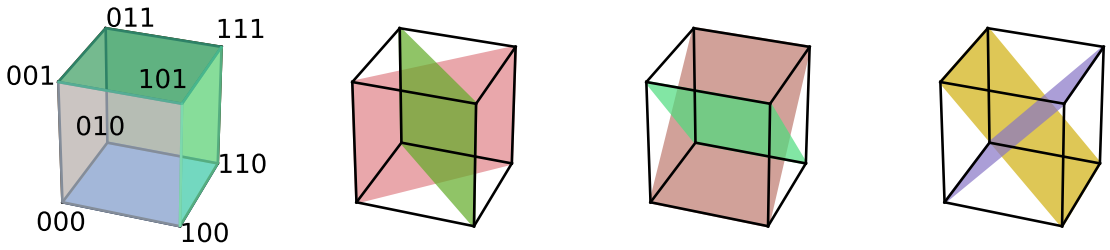

Figure 2.5: The 12 *non-flat* parallelograms in $\mathbb{B}^3$: the six faces of the cube, and 3×2 *diagonal* parallelograms, e.g. $(000) : (010) :: (101) : (111)$.

Let us finally note the following remarkable property, which will be useful to us later in Chapter 6:

Property 2.2. *Let* $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}$ *in* $\mathbb{B}^m$*, and let* $H(\mathbf{x}, \mathbf{x}')$ *be the Hamming distance between* $\mathbf{x}$ *and* $\mathbf{x}'$*, i.e. the number of components we need to flip to transform* $\mathbf{x}$ *into* $\mathbf{x}'$ *(or the reverse).*
If $\mathbf{a} : \mathbf{b} :: \mathbf{c} : \mathbf{d}$*, then these three equalities hold:*

$$\begin{cases} H(\mathbf{a},\mathbf{b}) = H(\mathbf{c},\mathbf{d}) \\ H(\mathbf{a},\mathbf{c}) = H(\mathbf{b},\mathbf{d}) \\ H(\mathbf{a},\mathbf{d}) = H(\mathbf{b},\mathbf{c}). \end{cases}$$

We can easily verify this property in $\mathbb{B}$ from Table 2.2, and the general case in $\mathbb{B}^m$ immediately follows from Definition 2.4. Sadly, Property 2.2 only offers a necessary condition for a proportion to hold, and not a sufficient one: indeed taking $a = d = 1$ and $b = c = 0$ in $\mathbb{B}$, the three equalities are clearly respected but $1 : 0 :: 0 : 1$ is not a valid proportion. The first two properties $H(\mathbf{a},\mathbf{b}) = H(\mathbf{c},\mathbf{d})$ and $H(\mathbf{a},\mathbf{c}) = H(\mathbf{b},\mathbf{d})$ are classical properties for parallelograms, but interestingly the third one $H(\mathbf{a},\mathbf{d}) = H(\mathbf{b},\mathbf{c})$ is only true for rectangles. In fact, the parallelograms that we can build in $\mathbb{B}^m$ are **always** rectangles.

This closes this discussion on the geometric interpretation of Boolean proportions. In the next section we will apply the previously described notions to a task of classification, which will help us to introduce new concepts, and also to make a smooth transition towards the next chapter.

2.3 Analogical inference principle and equation solving

The goal of this last section is to introduce two other fundamental notions that will be used in our investigations: the **analogical inference principle**, and the process of **analogical equation solving**. Once combined, these two concepts allow us to perform inference, and to apply analogical reasoning to machine learning tasks such as classification.

We will here consider a toy classification problem in a Boolean setting, which will help us introduce these two fundamental notions. This toy problem will also serve as an introduction to the next chapter, where we will study the theoretical properties of analogical classifiers. Because of the nature of the problem, this section will be a mix of particular and general considerations.

Problem setting

We now have enough background on Boolean proportions to start doing some machine learning. Here is our problem. We consider the set $\mathbb{B}^m$ and its 2^m elements. For various $\mathbf{x} \in \mathbb{B}^m$, we know the value of $f(\mathbf{x})$, where f is a function from $\mathbb{B}^m$ to $\mathbb{B}$. The value $f(\mathbf{x})$ is called the **class** of $\mathbf{x}$, or its **label**. The set $S \subsetneq \mathbb{B}^m$ of elements for which $f(\mathbf{x})$ is known is called the **training set**. For any element $\mathbf{x} \notin S$, $f(\mathbf{x})$ is unknown and our goal is to guess it: this is a binary **classification problem**.

Let us suppose that we are in $\mathbb{B}^3$ and consider Figure 2.6. We know the values of $f(\mathbf{x})$ for every single element $\mathbf{x}$ but $\mathbf{h}$, so $S = \{\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d},\mathbf{e},\mathbf{f},\mathbf{g}\}$. There are numerous methods at hand to go guess the value of $f(\mathbf{h})$. One of the most famous one is the nearest neighbors heuristic,

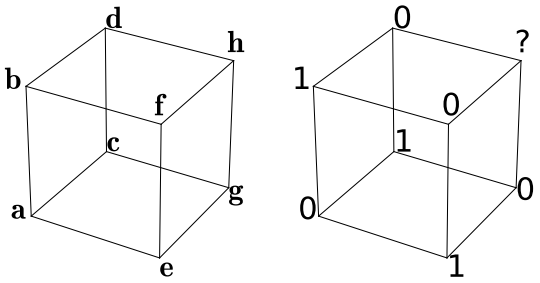

Figure 2.6: A classification problem in $\mathbb{B}^3$. Labels $f(\mathbf{x})$ are on the right cube.

which would lead us to consider that the label of $\mathbf{h}$ is the same as that of its *neighbors*, i.e. the elements that are close to $\mathbf{h}$ with respect to a given distance function. Using the Hamming distance, we would assign to $\mathbf{h}$ the label 0, because the 3 closest neighbors of $\mathbf{h}$, namely $\mathbf{d}, \mathbf{f}, \mathbf{g}$, are all labeled with 0.

The analogical inference principle

Our point is here to describe the process of analogical classification, so we will forget the nearest neighbor heuristic and we will instead make use of the so-called **analogical inference principle**, which states that if four elements $\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{t}$ are in proportion, then their labels should also be in proportion:

Analogical inference principle

$$\frac{\mathbf{x} : \mathbf{y} :: \mathbf{z} : \mathbf{t}}{f(\mathbf{x}) : f(\mathbf{y}) :: f(\mathbf{z}) : f(\mathbf{t})}$$

This is obviously an unsound principle, in that the conclusion does not logically follows from the premise. But as we will see in this document, it can still be useful.

The key point here is that if the value of $f(\mathbf{t})$ is unknown, it can be **recovered** from the values of $f(\mathbf{x}), f(\mathbf{y}), f(\mathbf{z})$ by the process of **analogical equation solving** that we explain now.

We here want to guess the value of $f(\mathbf{h})$. The analogical inference principle leads us to look for all 3-tuples $(\mathbf{x}, \mathbf{y}, \mathbf{z}) \in S^3$ such that $\mathbf{x} : \mathbf{y} :: \mathbf{z} : \mathbf{h}$ holds. The principle then states that for all of these 3-tuples, we should have $f(\mathbf{x}) : f(\mathbf{y}) :: f(\mathbf{z}) : f(\mathbf{h})$. Figure 2.5 tells us that there are 6 (non-flat) parallelograms involving $\mathbf{h}$ as a vertex. The six unique corresponding proportions are:

1. $\mathbf{a} : \mathbf{b} :: \mathbf{g} : \mathbf{h}$
2. $\mathbf{a} : \mathbf{d} :: \mathbf{e} : \mathbf{h}$
3. $\mathbf{a} : \mathbf{c} :: \mathbf{f} : \mathbf{h}$
4. $\mathbf{b} : \mathbf{d} :: \mathbf{f} : \mathbf{h}$
5. $\mathbf{e} : \mathbf{f} :: \mathbf{g} : \mathbf{h}$
6. $\mathbf{c} : \mathbf{g} :: \mathbf{d} : \mathbf{h}$.

Applying the analogical inference principle to the first proportion $\mathbf{a} : \mathbf{b} :: \mathbf{g} : \mathbf{h}$ leads to

$f(\mathbf{a}) : f(\mathbf{b}) :: f(\mathbf{g}) : f(\mathbf{h})$, which is equivalent to:

$$0 : 1 :: 0 : f(\mathbf{h}).$$

By referring to Table 2.2 on page 34, we notice that $f(\mathbf{h})$ should be equal to 1 for the proportion $f(\mathbf{a}) : f(\mathbf{b}) :: f(\mathbf{g}) : f(\mathbf{h})$ to be a valid one, because $0 : 1 :: 0 : 1$. So we keep 1 in the back of our head as a possible candidate for $f(\mathbf{h})$: we say that 1 is a **candidate solution** for $f(\mathbf{h})$. What we have just done is the **solving of an analogical equation**.

Analogical equation solving in general

Generally speaking, an analogical equation is a proportion $a : b :: c : x$ in a context where x is unknown. Determining the value of x is then called the *solving* of this equation. Depending on the nature and values of a, b, c, there may or may not exist a solution, and it might not be unique.

In fact, we have all been solving analogical equations since our childhood, because in the case of the geometric proportion it reduces to the well known rule of three: $\frac{5}{2} = \frac{10}{x} \implies x = \frac{2\times 10}{5} = 4$, and generally $a : b :: c : x \implies x = \frac{b\times c}{a}$ $(a \neq 0)$. The solution of the arithmetic proportion $a - b = c - x$ is also extremely simple and is given by $x = c - a + b$.

In a Boolean setting, an equation is solvable for 6 patterns of a, b, c (the 6 patterns of Table 2.2, naturally):

Proposition 2.2. *Let a, b, c in $\mathbb{B}$. The analogical equation $a : b :: c : x$ is solvable if and only if $a = b$ or $a = c$. The solution denoted* $\mathrm{sol}(a, b, c)$ *is always unique and is given by:*

$$\begin{cases} \mathrm{sol}(a, b, c) \stackrel{\mathsf{def}}{=} c & \text{if } a = b, \\ \mathrm{sol}(a, b, c) \stackrel{\mathsf{def}}{=} b & \text{if } a = c, \end{cases}$$

or more generally:

$$\mathrm{sol}(a, b, c) \stackrel{\mathsf{def}}{=} c - a + b,$$

because the Boolean proportion is a particular case of the arithmetic proportion. The values of $\mathrm{sol}(a, b, c)$ *for all the Boolean* 3*-tuples* (a, b, c) *are given in Table 2.3.*

Back to the estimation of $f(\mathbf{h})$

Let's get back to the estimation of $f(\mathbf{h})$. Applying the analogical inference principle to the second proportion $\mathbf{a} : \mathbf{d} :: \mathbf{e} : \mathbf{h}$ leads $f(\mathbf{a}) : f(\mathbf{d}) :: f(\mathbf{e}) : f(\mathbf{h})$, and to the solving of $0 : 0 :: 1 : f(\mathbf{h})$. Here again, the solution is 1, just like the candidate of the first proportion. The third proportion $\mathbf{a} : \mathbf{c} :: \mathbf{f} : \mathbf{h}$ leads to the solving of $0 : 1 :: 0 : f(\mathbf{h})$, also claiming that 1 is a good candidate.

a	b	c	$\text{sol}(a,b,c)$
0	0	0	0
1	1	1	1
0	0	1	1
1	1	0	0
0	1	0	1
1	0	1	0
1	0	0	?
0	1	1	?

Table 2.3: The solutions $\text{sol}(a,b,c)$ for every Boolean 3-tuple (a,b,c). The question mark ? means that $\text{sol}(a,b,c)$ is undefined. We can verify that $\text{sol}(a,b,c) = c - a + b$ when the solution is defined.

And now the fourth proportion $\mathbf{b} : \mathbf{d} :: \mathbf{f} : \mathbf{h}$, which leads to $1 : 0 :: 0 : f(\mathbf{h})$. **This equation is not solvable**: neither $1 : 0 :: 0 : 1$ nor $1 : 0 :: 0 : 0$ are valid proportions. We can also check this fact from Table 2.3. We thus simply discard the proportion $\mathbf{b} : \mathbf{d} :: \mathbf{f} : \mathbf{h}$ as a potential source of information about $f(\mathbf{h})$, because we are not in a position to apply the analogical inference principle. We can easily verify that the fifth and sixth proportions also lead to non-solvable equations.

All in all, we are left with three candidates coming from the first three proportions, all of which are equal to 1. Our guess will thus be that $f(\mathbf{h})$ should be equal to 1, so we will set the **estimation** of $f(\mathbf{h})$ as $\hat{f}(\mathbf{h}) = 1$. Notice that we have so far completely ignored all the flat proportions, i.e. proportions where the four elements are not all distinct. This is because none of these proportions allow us to derive a solvable equation. Considering for example $\mathbf{d} : \mathbf{h} :: \mathbf{d} : \mathbf{h}$, we're led to the solving of $f(\mathbf{d}) : f(\mathbf{h}) :: f(\mathbf{d}) : f(\mathbf{h})$, or equivalently of $0 : f(\mathbf{h}) :: 0 : f(\mathbf{h})$. Both values 0 and 1 could lead to equally valid proportions here, so there is simply no predictive power in the flat proportions. We have also only considered unique proportions, up to equivalence. The first proportion $\mathbf{a} : \mathbf{b} :: \mathbf{g} : \mathbf{h}$ for example is equivalent to $\mathbf{a} : \mathbf{g} :: \mathbf{b} : \mathbf{h}$ (using central permutation), which we could have used instead. As any two equivalent proportions lead to the same solution for their associated class equations, we usually choose to ignore all other equivalent proportions, which allows to divide the number of equation solving processes by a factor of 2.

The analogical classification process: summary

Here is a small outline of the classification process we have just followed, that was entirely governed by the analogical inference principle. Our goal was to guess the value of $f(\mathbf{h})$:

- We first looked at all the 3-tuples $(\mathbf{x}, \mathbf{y}, \mathbf{z}) \in \mathbf{B}^3$ such that:
  - $f(\mathbf{x}), f(\mathbf{y})$ and $f(\mathbf{z})$ are known;

– $\mathbf{x} : \mathbf{y} :: \mathbf{z} : \mathbf{h}$ holds;
– the **class equation** $f(\mathbf{x}) : f(\mathbf{y}) :: f(\mathbf{z}) : s$ is **solvable**. s is called a **candidate solution**.

Each of the three candidate solutions s_1, s_2, s_3 agreed on the same prediction: $s_i = 1$ for all i.

- We thus estimated that $f(\mathbf{h})$ should indeed be equal to 1.

Open questions

This minimal example of analogical classification raises a few concerns, which will be thoroughly addressed in the rest of this document:

1. All of the three candidate solutions for $f(\mathbf{h})$ agreed on the same prediction: 1. What if one of them predicted 0? A first drastic option is to refuse to classify $\mathbf{h}$, on the basis that if the candidates cannot agree on their predictions, then we should not trust any of them. As it will become clear, in practice the candidates never really completely agree, even though sometimes a clear majority emerges. So this strategy would make the prediction impossible for most elements $\mathbf{x} \notin S$. A wiser option is to consider an aggregation of the candidate solutions: the most common one, for example. This is the option that has been taken on so far, as we will see in the next chapter.
2. Here, by chance, the set of candidate solutions for $f(\mathbf{h})$ was not empty: we found three candidate solutions. What happens if we cannot find any candidate? This case arises when there are no 3-tuple $(\mathbf{x}, \mathbf{y}, \mathbf{z}) \in S^3$ such that $\mathbf{x} : \mathbf{y} :: \mathbf{z} : \mathbf{h}$ holds and such that the associated equation $f(\mathbf{x}) : f(\mathbf{y}) :: f(\mathbf{z}) : f(\mathbf{h})$ is solvable. In the works of Stroppa and Yvon ([SY05a]), such an $\mathbf{h}$ cannot be classified. The notion of **analogical dissimilarity** as used in [BMD07] will allow to bypass this issue. In the next chapter, we will detail these two versions of analogical learning, and one of the contributions of our work is to provide a unifying view of the two techniques.
3. We have entirely relied on the analogical inference principle to classify $f(\mathbf{h})$. How *safe* is this inference principle? Can we find some theoretical guarantees that could allow us to use it on a sound basis? In Chapter 6, we will provide a complete characterization of the Boolean functions f that allow to derive sound conclusions.

Before moving further to the next section, let us consider a seemingly trivial question: how many proportions can we build in $\mathbb{B}^m$? As we will see, this question will provide us with some further insight about the Boolean proportion. This last discussion is, however, not as important as the previous ones and can safely be ignored by the impatient reader.

How many proportions can we build in $\mathbb{B}^m$?

Exactly 6^m! And it is fairly easy to derive: there are 6 proportions in $\mathbb{B}$. As a proportion in

$\mathbb{B}^2$ is the concatenation of any two proportions in $\mathbb{B}$, there are exactly $6^2 = 36$ proportions in $\mathbb{B}^2$, as seen in the previous section. Analogical proportions in $\mathbb{B}^m$ are defined component-wise, i.e. they are the concatenation of m proportions in $\mathbb{B}$. Equivalently, a proportion in $\mathbb{B}^m$ is the concatenation of a proportion in $\mathbb{B}^{m-1}$ and a proportion in $\mathbb{B}$. By trivial induction, we can build 6^m proportions in $\mathbb{B}^m$.

But let's now ask a more relevant and challenging question: **how many *useful* proportions can we build in $\mathbb{B}^m$?** By *useful*, we mean proportions that could be used for classification purposes. We have seen in this section that the only proportions $\mathbf{a} : \mathbf{b} :: \mathbf{c} : \mathbf{d}$ that are useful for classification purposes are those where all elements $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}$ are distinct, i.e. the non-flat proportions. We thus want to exclude from the 6^m proportions those that comply with one of the following patterns:

- $\mathbf{a} : \mathbf{a} :: \mathbf{a} : \mathbf{a}$;
- $\mathbf{a} : \mathbf{b} :: \mathbf{a} : \mathbf{b}$, and its three equivalent forms $\mathbf{a} : \mathbf{a} :: \mathbf{b} : \mathbf{b}$, $\mathbf{b} : \mathbf{a} :: \mathbf{b} : \mathbf{a}$, and $\mathbf{b} : \mathbf{b} :: \mathbf{a} : \mathbf{a}$.

Now, let's count them.

- Each of the 2^m vertices $\mathbf{a}$ will generate a proportion of the form $\mathbf{a} : \mathbf{a} :: \mathbf{a} : \mathbf{a}$, so there are exactly 2^m proportions of this kind.
- Every pair $(\mathbf{a}, \mathbf{b})$ of vertices will generate four proportions:
  - $\mathbf{a} : \mathbf{b} :: \mathbf{a} : \mathbf{b}$;
  - $\mathbf{a} : \mathbf{a} :: \mathbf{b} : \mathbf{b}$;
  - $\mathbf{b} : \mathbf{a} :: \mathbf{b} : \mathbf{a}$;
  - $\mathbf{b} : \mathbf{b} :: \mathbf{a} : \mathbf{a}$.

  There are $\binom{2^m}{2}$ distinct pairs of vertices, so in total this makes $4 \cdot \binom{2^m}{2}$ proportions of the form $\mathbf{a} : \mathbf{b} :: \mathbf{a} : \mathbf{b}$ with its equivalent forms.

We are then left with the number of $6^m - 2^m - 4 \cdot \binom{2^m}{2}$ useful proportions. We know that for each of these proportions, all elements $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}$ are distinct. For each of them, there are thus 7 other equivalent forms, which reduces the number of useful and unique (up to equivalence) proportions to:

$$P_m = \frac{1}{8}\left[6^m - 2^m - 4 \cdot \binom{2^m}{2}\right].$$

Table 2.4 gives the values of P_m for the first 10 values of m. Naturally, P_2 and P_3 are in accordance with the empirical results from Figures 2.4 and 2.5. The fact that $P_1 = 0$ means that no inference can be done with analogical proportion in $\mathbb{B}$, which is perfectly normal: in $\mathbb{B}$, all proportions are flat (i.e. the pattern is always $a : b :: a : b$ or $a : a :: b : b$).

In fact, it turns out that this sequence of numbers $0, 1, 12, 100 \ldots$ is already known as the A016283 sequence from the OEIS[4]. This sequence actually describes *the number of rectangles*

[4]The On-line Encyclopedia of Integer Sequences (`http://oeis.org/A016283`).

m	P_m
1	0
2	1
3	12
4	100
5	720
6	4816
7	30912
8	193600
9	1194240
10	7296256

Table 2.4: Number of unique non-flat proportions in an m-dimensional cube.

that can be formed from the vertices of an m-dimensional cube, which is exactly equivalent to the number of *useful* proportions that we wanted to compute. The formula given by the OEIS is:

$$P_m = \frac{6^m}{8} - 4^{m-1} + 2^{m-3}.$$

Using the fact that $\binom{n}{k} = \frac{n}{k}\binom{n-1}{k-1}$, it can be shown that the two formulas $\frac{1}{8}\left[6^m - 2^m - 4 \cdot \binom{2^m}{2}\right]$ and $\frac{6^m}{8} - 4^{m-1} + 2^{m-3}$ are, fortunately, equivalent. It would be interesting to find out how the OEIS formula was derived, but in all fairness we have no idea of the process that was used.

Conclusion

In this chapter, we have given the necessary background on formal analogical proportions, mostly focusing on Boolean proportions. We have seen that an analogical proportion A is a quaternary relation following three simple axioms that were already known in Aristotle times.

In the first section, we formally defined various analogical proportions in different settings, starting from the most general definitions for semigroups and groups, until we reached the Boolean proportion. We also presented an extension of the Boolean proportion to multiple-valued logics, which we will use much later in Chapter 5.

In the second section, we extended our knowledge and understanding of the Boolean proportion. We illustrated the close bonds that exist between the Boolean proportion and the arithmetic proportion, and we showed that both can be seen as a relation between the four vertices of a parallelogram in $\mathbb{B}^m$ or in $\mathbb{R}^m$.

The third section was devoted to the description of the analogical inference principle and of the analogical equation solving process. By solving an analogical equation, we are able to infer some information about any of the four elements at play in an analogy, provided that the three others are known. This process can be seen as a simple generalization of the rule of three that

we have been using since childhood: $\frac{4}{2} = \frac{6}{x} \implies x = 3$. Using this powerful tool and applying the analogical inference principle, we were able to go trough a simple classification problem in a Boolean setting.

The aim of these first two chapters was to provide an overview of past works in analogical reasoning, and to provide the necessary background to understand the contributions made in this thesis. In the next chapter, we will describe our contributions to the task of analogical classification.

Résumé du chapitre Ce chapitre est dédié à la présentation du principal outil utilisé dans ces travaux : les proportions analogiques. Une proportion analogique est une expression de la forme *a est à b ce que c est à d*. Cette expression exprime le fait que ce qui est commun à a et b l'est aussi à c et d, et ce qui diffère entre a et b diffère aussi entre c et d. On pourra trouver de nombreux exemples de proportion analogique, par exemple "*le veau est à la vache comme le poulain est à la jument*".

Selon Aristote, une proportion analogique A est une relation quaternaire qui satisfait les trois axiomes suivant, pour tout élément a, b, c, d :

1. $A(a, b, a, b)$ est toujours vrai (réflexivité)
2. $A(a, b, c, d) \implies A(c, d, a, b)$ (symétrie)
3. $A(a, b, c, d) \implies A(a, c, b, d)$ (permutation centrale)

Lorsqu'il n'y a pas d'ambiguité sur A et son domaine, on se permettra d'utiliser la notation infixe $a : b :: c : d$. Considérant à nouveau notre exemple fermier, l'axiome de symétrie indique si le veau (a) est à la vache (b) ce que le poulain (c) est à la jument (d), alors le poulain (c) est à la jument (d) ce que le veau (a) est à la vache (b), ce qui semble tout à fait naturel. L'axiome de permutation centrale indique qu'une autre conséquence est que le veau (a) est au poulain (c) ce que la vache (b) est à la jument (d).

Dans la première section, nous avons formellement défini les proportions analogiques dans divers domaines algébriques, partant de la définition générale dans les semi-groupes pour arriver aux proportions booléennes. Nous avons aussi brièvement présenté une extension multi-valuée de la proportion booléenne qui nous sera utile bien plus tard au chapitre 5.

Dans la deuxième section, nous avons illustré les liens étroits qui relient la proportion booléenne et la proportion arithmétique via diverses considérations géométriques. En particulier, nous avons illustré le fait que ces deux proportions mettent en jeu les quatre sommets d'un parallélogramme soit dans $\mathbb{B}^m$, soit dans $\mathbb{R}^m$.

Dans la troisième section, nous nous sommes intéressés à la résolution d'équations analogiques, ainsi qu'au principe d'inférence analogique. En résolvant une équation analogique, nous sommes capables d'inférer des propriétés relatives à l'un des quatre éléments de la proportion. Ce processus est en fait une sorte de généralisation de la fameuse règle de trois : $\frac{4}{2} = \frac{6}{x} \implies x = 3$. Nous avons appliqué cet outil inférentiel à un problème de classification, en se reposant sur

le principe d'inférence analogique qui stipule que lorsque quatre éléments sont en proportion, alors cela doit aussi être le cas pour leurs labels respectifs.

Chapter 3

A functional definition of analogical classifiers

Content

In the previous chapter, we briefly described a basic process of analogical classification in an informal way. In this chapter, we will clearly and formally define analogical classification, which will help us clarify some of the grey areas that we have glimpsed before. Also, we will describe our contributions to this problem.

A first form of analogical classifiers has been defined in the works of Stroppa and Yvon [SY05a], which we will refer to as **conservative classifiers**. These works extend some previous proposal on learning by analogy in the setting of natural language processing [PY99]. Another form of analogical classifiers has later been proposed in the works of Bayoudh, Delhay and Miclet [MBD08, BMD07], referred-to here as **extended classifiers**.

Though the theoretical groundings of the conservative and extended classifiers have a lot in common (they are both analogical classifiers after all), we will see that their practical implementations are actually quite different. We will show however that the extended classifier is a generalization of the conservative one, and most importantly that these two approaches can be factorized into a single, unifying framework.

So far, only algorithmic descriptions of both methods were available. Our unifying framework will allow us to give a general and **functional** definition of analogical classifiers, opening the door to some theoretically grounded research.

This chapter is an extended version of [HPRS16a] and it structured as follows. In Section 3.1, we will review and detail the two previous approaches to analogical classification, that we will name here the conservative and the extended classifiers. We will then get to the heart of the matter and start describing our first contribution, that consists in factorizing these two approaches into a single functional description. Thanks to this functional definition of analogical learners[1], we will be able in Section 3.2 to derive some theoretical properties about analogical learners. We will indeed show that their VC-dimension is infinite, and that their accuracy is closely related to that of the k-NN classifier. Finally in Section 3.3, we will empirically verify our results and provide some insights that will motivate the topic of Chapter 6.

3.1 Analogical classification

This first section will be devoted to the unification of two seemingly different analogical approaches to classification. Let us first define the problem of classification, one of the main subfields of machine learning. The aim is simple: a classifier has to estimate the class of any element given as input on the basis of some other elements for which the class is known. Formally, we denote X^m our universe which is usually a Cartesian product: $X^m = X_1 \times X_2 \times \ldots \times X_m$. We have at our disposal a subset $S \subsetneq X^m \times Y$ of n pairs $(\mathbf{x}, f(\mathbf{x}))$, called the **training set**.

$$S = \left\{ \left(\mathbf{x}^{(i)}, f(\mathbf{x}^{(i)})\right) \in X^m \times Y \;\middle|\; i \in [1, n] \right\}.$$

For any $\mathbf{x} \in X^m$, the value $f(\mathbf{x})$ is called its **class** or **label**. The functional notation $f(\mathbf{x})$ suggests that all labels are defined by an underlying function $f\colon X^m \to Y$, which is the usual general view point of machine learning (usually, $\mathbf{x}$ is also considered as a random variable). In the case of classification, Y is a finite set of size C where C is the number of different classes.

[1] In what follows, we will abuse language and use *analogical learner* and *analogical classifier* as synonyms.

The goal of a classifier is to **learn** the function f on the basis of the elements in S, called the **examples**. The output of a classifier is a function $\hat{f}\colon X^m \to Y$ that is *as close as possible* the ground truth function f.

3.1.1 Conservative classifier

We will here describe what we call a **conservative** classifier. Without further ado, let us consider Algorithm 1 which describes such classifiers.

Algorithm 1 The Conservative classifier.

Input: A training set S and an element $\mathbf{x} \in X^m \setminus S$ for which $f(\mathbf{x})$ is unknown.
Output: $\hat{f}(\mathbf{x})$, an estimation of $f(\mathbf{x})$.
Init: $\mathbf{C}(\mathbf{x}) \leftarrow \varnothing$ // A multiset of candidate labels.
for all $(\mathbf{a}, \mathbf{b}, \mathbf{c}) \in S^3$ such that $\mathbf{a} : \mathbf{b} :: \mathbf{c} : \mathbf{x}$ **do**
  if $f(\mathbf{a}) : f(\mathbf{b}) :: f(\mathbf{c}) : y$ is solvable **then**
    $y \leftarrow \mathrm{sol}\,(f(\mathbf{a}), f(\mathbf{b}), f(\mathbf{c}))$
    $\mathbf{C}(\mathbf{x}) \leftarrow \mathbf{C}(\mathbf{x}) \cup y$
  end if
end for
$\hat{f}(\mathbf{x}) \leftarrow \mathrm{Mode}(\mathbf{C}(\mathbf{x}))$ // The most common value in $\mathbf{C}(\mathbf{x})$ (undefined if $\mathbf{C}(\mathbf{x}) = \varnothing$).
// Ties broken at random.

It should be easy to see that the process of Algorithm 1 is extremely similar to the one we followed on our first introduction to classification with Boolean proportions in Section 2.3. Here again the key underlying process is the analogical inference principle, which states that if four elements are in proportion, then their classes should also be in proportion. But Algorithm 1 is slightly more general than what we have glimpsed in Section 2.3: it allows to deal with cases where some of the candidate solutions y do not agree with each other. The fix here is to consider that the prediction $\hat{f}(\mathbf{x})$ will be the most common candidate solution among all the predictors. This definition of analogical classification comes from the work of Stroppa and Yvon [SY05a, SY05b], and to our knowledge it is the first instance of analogical learner that has ever been proposed, following ideas that can be already found in [PY99]. We note however that the authors did not explicitly state that the class equations must be solvable to be useful.

Note that if the multiset $\mathbf{C}(\mathbf{x})$ is empty, i.e. if we cannot find 3-tuples in S^3 such that they are in proportion with $\mathbf{x}$ and such that the related class equation is solvable, then the value $\hat{f}(\mathbf{x})$ is undefined and $\mathbf{x}$ cannot be classified (hence the name *conservative*).

At first sight, we are here in a setting where the examples in S are stored for future use, without any generalization process. We will thus consider (for now) that the conservative classifier ranges among the category of **lazy learners**, i.e. learners that only rely on the training set S without any actual *learning* stage, or without building any model of the data. The most popular instance of lazy learner certainly is the famous k-NN algorithm.

In the following, we will give a functional definition of $\hat{f}(\mathbf{x})$ as outputted by a conservative classifier. To this end, we will define two crucial concepts: the **analogical extension** of a training set S, and the **analogical root** of an element $\mathbf{x}$. Both the analogical extension and the analogical root were first defined in [SY05b], with some minor differences with our definitions.

Definition 3.1 (Analogical extension)**.** Denoting S a training set and f the ground truth function of the labels, the **analogical extension** of S using f is:

$$\mathbf{E}_S^f \stackrel{\text{def}}{=} \left\{ \mathbf{x} \in X^m \;\middle|\; \exists (\mathbf{a}, \mathbf{b}, \mathbf{c}) \in S^3,\ \mathbf{a} : \mathbf{b} :: \mathbf{c} : \mathbf{x} \text{ and } f(\mathbf{a}) : f(\mathbf{b}) :: f(\mathbf{c}) : y \text{ is solvable} \right\}.$$

Intuitively, $\mathbf{E}_S^f$ can be regarded as the set of all $\mathbf{x} \in X^m$ that are in proportion with at least one 3-tuple in S, provided that the equation related to the associated labels is also solvable. For any S and any f, $\mathbf{E}_S^f$ satisfies some elementary properties:

1. $S \subseteq \mathbf{E}_S^f$, since $\mathbf{x} : \mathbf{x} :: \mathbf{x} : \mathbf{x}$ always holds;
2. $\mathbf{E}_\varnothing^f = \varnothing$, and $\mathbf{E}_{X^m}^f = X^m$;
3. $S_1 \subseteq S_2 \implies \mathbf{E}_{S_1}^f \subseteq \mathbf{E}_{S_2}^f$.

We will denote by $\mathbf{E}_S^{f*}$ the set of elements in $\mathbf{E}_S^f$ that are not in S:

$$\mathbf{E}_S^{f*} \stackrel{\text{def}}{=} \mathbf{E}_S^f \setminus S.$$

In a setting where for any $\mathbf{a}, \mathbf{b}, \mathbf{c}$ the equation $\mathbf{a} : \mathbf{b} :: \mathbf{c} : \mathbf{x}$ is always solvable (as is the case for the arithmetic proportion) and such that the associated class equation is also solvable, we can show that the size of $\mathbf{E}_S^{f*}$ is exactly $\frac{1}{2}n^2(n-1)$ [BMMA07]. This value is thus a loose upper bound for $|\ \mathbf{E}_S^{f*}\ |$, because in practice equations are not always solvable.

Example:
To polish our understanding of the analogical extension, let us consider Figure 3.1. The training set S is $\{\, \mathbf{a}, \mathbf{b}, \mathbf{d}, \mathbf{e} \,\}$. Taking $(\mathbf{b}, \mathbf{d}, \mathbf{a}) \in S^3$, we notice that $\mathbf{b} : \mathbf{d} :: \mathbf{a} : \mathbf{c}$, and the associated class equation $0 : 1 :: 0 : y$ is solvable. So $\mathbf{c}$ is in $\mathbf{E}_S^f$. The same goes for $\mathbf{f}$ with the 3-tuple $(\mathbf{a}, \mathbf{b}, \mathbf{e}) \in S^3$. However, even if we can find $(\mathbf{b}, \mathbf{d}, \mathbf{e}) \in S^3$ such that $\mathbf{b} : \mathbf{d} :: \mathbf{e} : \mathbf{g}$, the associated class equation $0 : 1 :: 1 : y$ is not solvable, so $\mathbf{g} \notin \mathbf{E}_S^f$. At last, vertex $\mathbf{h}$ will suffer the same fate because the class equation $f(\mathbf{a}) : f(\mathbf{e}) :: f(\mathbf{d}) : y$ is not solvable either. In the end, $\mathbf{E}_S^f = \{\, \mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{e}, \mathbf{f} \,\}$ and $\mathbf{E}_S^{f*} = \{\, \mathbf{c}, \mathbf{f} \,\}$.

The dual concept of the analogical extension is what we call the **analogical root** of a given element $\mathbf{x} \in X^m$:

Definition 3.2 (Analogical root)**.** Denoting S a training set and f the ground truth function of the labels, the **analogical root** of an element $\mathbf{x} \in X^m$ is:

$$\mathbf{R}_S^f(\mathbf{x}) \stackrel{\text{def}}{=} \left\{ (\mathbf{a}, \mathbf{b}, \mathbf{c}) \in S^3 \;\middle|\; \mathbf{a} : \mathbf{b} :: \mathbf{c} : \mathbf{x} \text{ and } f(\mathbf{a}) : f(\mathbf{b}) :: f(\mathbf{c}) : y \text{ is solvable} \right\}.$$

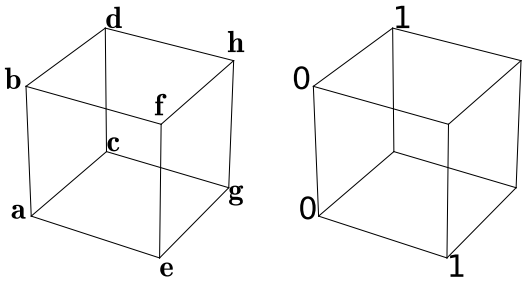

Figure 3.1: With $S = \{ \mathbf{a}, \mathbf{b}, \mathbf{d}, \mathbf{e} \}$, we have $\mathbf{E}_S^{f*} = \{ \mathbf{c}, \mathbf{f} \}$.

$\mathbf{R}_S^f(\mathbf{x})$ is simply the set of 3-tuples in S which are analogically linked to $\mathbf{x}$.

Example:
Considering again the example of Figure 3.1, we have $\mathbf{R}_S^f(\mathbf{c}) = \{ (\mathbf{b}, \mathbf{d}, \mathbf{a}), (\mathbf{b}, \mathbf{a}, \mathbf{d}) \}$. However, the two corresponding proportions $\mathbf{b} : \mathbf{d} :: \mathbf{a} : \mathbf{c}$ and $\mathbf{b} : \mathbf{a} :: \mathbf{d} : \mathbf{c}$ are equivalent, and we will follow the convention that the analogical root should only contain non-equivalent 3-tuples, so technically $\mathbf{R}_S^f(\mathbf{c})$ is entirely described by $\{ (\mathbf{b}, \mathbf{d}, \mathbf{a}) \}$. We also have $\mathbf{R}_S^f(\mathbf{f}) = \{ (\mathbf{a}, \mathbf{b}, \mathbf{e}) \}$, $\mathbf{R}_S^f(\mathbf{a}) = \{ (\mathbf{a}, \mathbf{a}, \mathbf{a}), (\mathbf{b}, \mathbf{b}, \mathbf{a}), (\mathbf{d}, \mathbf{d}, \mathbf{a}), (\mathbf{e}, \mathbf{e}, \mathbf{a}) \}$, and $\mathbf{R}_S^f(\mathbf{h}) = \varnothing$.

It is clear that in general,

$$\mathbf{R}_S^f(\mathbf{x}) = \varnothing \iff \mathbf{x} \notin \mathbf{E}_S^f.$$

It should also be clear that even if we only consider unique 3-tuples (up to equivalence) in the analogical root, $\mathbf{R}_S^f(\mathbf{x})$ may contain more than one 3-tuple: for example in $\mathbb{R}^m$ or $\mathbb{B}^m$ with the arithmetic (or Boolean) proportion, $\mathbf{x}$ may be the involved in more than one parallelogram, as illustrated in Figure 3.2.

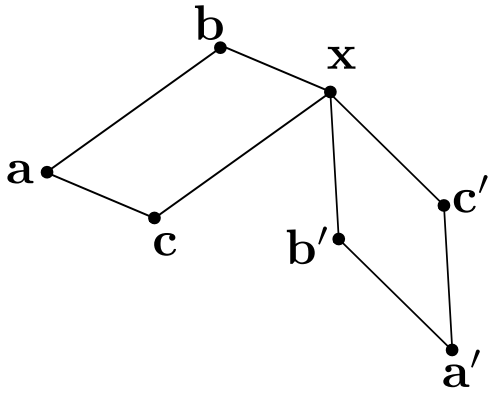

Figure 3.2: With $S = \{ \mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{a}', \mathbf{b}', \mathbf{c}' \}$, we have $\mathbf{x} \in \mathbf{E}_S^f$ and $\mathbf{R}_S^f(\mathbf{x}) = \{ (\mathbf{a}, \mathbf{b}, \mathbf{c}), (\mathbf{a}', \mathbf{b}', \mathbf{c}') \}$. Class equations are all assumed to be solvable.

We need to introduce one last definition before we can go on with the functional view of conservative classifiers:

Definition 3.3 (Analogical label)**.** For any element $\mathbf{x}$ of $\mathbf{E}_S^f$, we denote $\mathbf{C}(\mathbf{x})$ the multiset of solutions to the class equations associated with $\mathbf{x}$:

$$\mathbf{C}(\mathbf{x}) \stackrel{\text{def}}{=} \left\{ y \;\middle|\; f(\mathbf{a}) : f(\mathbf{b}) :: f(\mathbf{c}) : y, \quad \text{with } (\mathbf{a}, \mathbf{b}, \mathbf{c}) \in \mathbf{R}_S^f(\mathbf{x}) \right\}$$

We then define the **analogical label** of $\mathbf{x}$, denoted $\overline{f}(\mathbf{x})$, as:

$$\overline{f}(\mathbf{x}) \stackrel{\text{def}}{=} \begin{cases} f(\mathbf{x}) \text{ if } \mathbf{x} \in S \\ \text{Mode}\left(\mathbf{C}(\mathbf{x})\right) \text{ if } x \notin S \end{cases}$$

where Mode($\mathbf{C}$) returns the most frequent element of the multiset $\mathbf{C}$. In case of a tie, the returned element is chosen at random between the most frequent elements.

We are now in position to give a functional definition of the conservative classifier:

Definition 3.4 (Conservative classifier)**.** Given a training set S with an underlying ground truth function f, a **conservative classifier** sets the prediction of an element $\mathbf{x}$ as follows:

$$\text{If } \mathbf{x} \in \mathbf{E}_S^f, \;\; \hat{f}(\mathbf{x}) \stackrel{\text{def}}{=} \overline{f}(\mathbf{x}). \text{ Else, } \hat{f}(\mathbf{x}) \text{ is undefined.}$$

As previously mentioned, a conservative classifier is not able to output a prediction if $\mathbf{x}$ does not belong to the analogical extension. We have already made this observation when describing Algorithm 1: for a given $\mathbf{x}$, the set of candidate solutions $\mathbf{C}(\mathbf{x})$ is empty if and only if $\mathbf{x} \notin \mathbf{E}_S^f$. Also, we now understand why $\overline{f}(\mathbf{x})$ is simply set to $f(\mathbf{x})$ when the $\mathbf{x} \in S$: for such an $\mathbf{x}$ it is natural to want $\hat{f}(\mathbf{x}) = f(\mathbf{x})$!

Example:
Going back to our example with Figure 3.1, we will have $\hat{f}(\mathbf{c}) \stackrel{\text{def}}{=} \overline{f}(\mathbf{c}) = 1$, because $\mathbf{R}_S^f(\mathbf{c}) = \{ (\mathbf{b}, \mathbf{d}, \mathbf{a}) \}$ and $\mathbf{C}(\mathbf{c}) = \{ 1 \}$. Also, $\hat{f}(\mathbf{f}) \stackrel{\text{def}}{=} \overline{f}(\mathbf{f}) = 1$, because $\mathbf{R}_S^f(\mathbf{f}) = \{ (\mathbf{a}, \mathbf{b}, \mathbf{e}) \}$ and $\mathbf{C}(\mathbf{f}) = \{ 1 \}$. Here, we did not need to use the majority-vote procedure as there was only one candidate solution for each prediction. As $\mathbf{g}$ and $\mathbf{h}$ do not belong to $\mathbf{E}_S^f$, both $\overline{f}(\mathbf{g})$ and $\overline{f}(\mathbf{h})$ stay undefined.

Let us note that in Algorithm 1, the analogical extension $\mathbf{E}_S^f$ is never explicitly computed. Instead, for a given $\mathbf{x}$, we look for every 3-tuple in S^3 and check if they belong to $\mathbf{R}_S^f(\mathbf{x})$. One drawback is that the search in S^3 has to be done for every new element $\mathbf{x}$ that we want to classify, and can prove to be quite expensive in computation time, even for small sizes of S. However, taking advantage of our functional definition, we can reformulate the learning process of a conservative classifier as follows:

- First, compute $\mathbf{E}_S^f$. While $\mathbf{E}_S^f$ is being computed, it is trivial to also compute the set of candidates $\mathbf{C}(\mathbf{x})$ for each $x \in \mathbf{E}_S^f$.

- Then, for all $x \in \mathbf{E}_S^f$, compute $\overline{f}(\mathbf{x}) \stackrel{\text{def}}{=} \text{Mode}(\mathbf{C}(\mathbf{x}))$.

Whenever we are asked to classify a given $\mathbf{x}$, all we need now is to check whether $\mathbf{x}$ belongs to $\mathbf{E}_S^f$ or not. The output $\hat{f}(\mathbf{x})$ is then immediate. This way to proceed has the undeniable advantage to perform the search in S^3 only once. Obviously, this saving in computation time is done at the expense of memory consumption, because the size of $\mathbf{E}_S^f$ can be drastically greater than that of S. This alternative view also leads us to reconsider our previous statement about conservative classifiers belonging to the class of lazy learners. Here, the computation of the analogical extension can be viewed as a generalization process, even if not all elements of the universe X^m are affected by this generalization.

We will end this discussion on conservative classifier by sketching a use-case described in [SY05b], where an analogical proportion is defined on the set of words over a finite alphabet, and the task is to learn the conjugation of English verbs. This setting is slightly different from that of classification that we have settled-in, but all definitions and concepts remain valid. Here, solving the analogical equation $view : reviewer :: search : y$ leads to *researcher*, using the definition of the analogy and the concatenation operator.

Example:
We have a training set $S = \left\{ \mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \mathbf{x}^{(3)}, \mathbf{x}^{(4)}, \mathbf{x}^{(5)} \right\}$ where $\mathbf{x}^{(1)} = (read, 3, reads), \mathbf{x}^{(2)} = (read, G, reading), \mathbf{x}^{(3)} = (view, 3, views)$, $\mathbf{x}^{(4)} = (view, G, viewing), \mathbf{x}^{(5)} = (eat, 3, eats)$.
$(read, 3, reads)$ means that *read* is transformed into *reads* when conjugated at the third person singular, and G stands for the gerund form.

Given a new element $\mathbf{x} = (eat, G, ?)$, $\mathbf{R}_S^f(\mathbf{x}) = \left\{ \left(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \mathbf{x}^{(5)}\right), \left(\mathbf{x}^{(3)}, \mathbf{x}^{(4)}, \mathbf{x}^{(5)}\right) \right\}$. The prediction is as expected $\hat{f}(\mathbf{x}) = eating$, which is the solution of the equations $reads : reading :: eats : y$ and $views : viewing :: eats : y$. There is no solution for $\mathbf{x} = (absurd, G, y)$, just because $\mathbf{R}_S^f(\mathbf{x}) = \varnothing$.

The fact that conservative learners cannot output a prediction for any $\mathbf{x}$ can here be considered a good thing: it would not be natural to look for the gerund form of *absurd*. However, in many use-cases we do want our classifier to be able to predict the label of any element. This is why other options have been implemented to overcome this problem, and to extend in some sense the generalization ability of analogical learners. This is the purpose of **extended classifiers** described in the next section.

3.1.2 Extended classifier

We now want our analogical classifier to be able to predict the label of any input element $\mathbf{x} \in X^m$, and not just the ones in $\mathbf{E}_S^f$. The main bottleneck of Algorithm 1 is that we require the 3-tuples in S^3 to be in perfect proportion with the element $\mathbf{x}$ we want to classify. Such 3-tuples are not always available in the training set, so necessarily some elements of X^m are excluded from $\mathbf{E}_S^f$.

The key concept that will help us overcome this issue is what we call an **analogical dissimilarity**, first introduced in [BMD07]. The analogical dissimilarity is a measure that quantifies in some sense *how far* a relation $\mathbf{a} : \mathbf{b} :: \mathbf{c} : \mathbf{d}$ is from being a perfectly valid proportion.

We keep the initial notation $\mathrm{AD}(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d})$ to denote the analogical dissimilarity between 4 elements. Some minimal properties have to be satisfied by such a dissimilarity $\mathrm{AD}\colon (X^m)^4 \longrightarrow \mathbb{R}^+$ to fit with the intuition. For any $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{e}, \mathbf{f} \in X^m$, the following properties should hold:

- $\mathrm{AD}(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}) = 0 \iff \mathbf{a} : \mathbf{b} :: \mathbf{c} : \mathbf{d}$ (Consistency with the Analogy)
- $\mathrm{AD}(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}) = \mathrm{AD}(\mathbf{c}, \mathbf{d}, \mathbf{a}, \mathbf{b})$ (Symmetry)
- $\mathrm{AD}(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}) = \mathrm{AD}(\mathbf{a}, \mathbf{c}, \mathbf{b}, \mathbf{d})$ (Central permutation)
- $\mathrm{AD}(\mathbf{a}, \mathbf{b}, \mathbf{e}, \mathbf{f}) \leq \mathrm{AD}(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}) + \mathrm{AD}(\mathbf{c}, \mathbf{d}, \mathbf{e}, \mathbf{f})$ (Triangle inequality)
- $\mathrm{AD}(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}) \neq \mathrm{AD}(\mathbf{b}, \mathbf{a}, \mathbf{c}, \mathbf{d})$ (except if $\mathbf{a} = \mathbf{b}$)

Naturally, the symmetry and central permutation properties lead to the same three classes of equivalence that we have for the analogical proportion. In particular, the equivalence class that is *generated* by $\mathrm{AD}(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d})$ is:

$$\mathrm{AD}(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}) = \mathrm{AD}(\mathbf{c}, \mathbf{d}, \mathbf{a}, \mathbf{b}) = \mathrm{AD}(\mathbf{c}, \mathbf{a}, \mathbf{d}, \mathbf{b}) = \mathrm{AD}(\mathbf{d}, \mathbf{b}, \mathbf{c}, \mathbf{a}) = \mathrm{AD}(\mathbf{d}, \mathbf{c}, \mathbf{b}, \mathbf{a}) = \mathrm{AD}(\mathbf{b}, \mathbf{a}, \mathbf{d}, \mathbf{c}) = \mathrm{AD}(\mathbf{b}, \mathbf{d}, \mathbf{a}, \mathbf{c}) = \mathrm{AD}(\mathbf{a}, \mathbf{c}, \mathbf{b}, \mathbf{d}).$$

The definition of an analogical proportion strongly relies on the structure and operators available on X^m, and the same goes for the definition of the analogical dissimilarity: there are a lot of possibilities. For the arithmetic proportion in $\mathbb{R}^m$, the analogical dissimilarity $\mathrm{AD}(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d})$ should capture *how far* are $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}$ from making up a perfect parallelogram. This can be achieved using the following definition:

Definition 3.5 (Analogical dissimilarity for real vectors)**.** Given four vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}$ of $\mathbb{R}^m$ equipped with the standard p-norm $\|\cdot\|_p$, the analogical dissimilarity related to the arithmetic proportion is defined as:

$$\mathrm{AD}(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}) = \|(\mathbf{a} - \mathbf{b}) - (\mathbf{c} - \mathbf{d})\|_p.$$

Of course, we will need an analogical dissimilarity in $\mathbb{B}^m$, but we will start small and consider $\mathbb{B}$ first:

Definition 3.6 (Analogical dissimilarity for Boolean values)**.** Given four elements a, b, c, d of $\mathbb{B}$, the analogical dissimilarity $\mathrm{AD}(a, b, c, d)$ is defined as the number of values that have to be flipped to get a proper analogy.

Table 3.1 shows the values of $\mathrm{AD}(a, b, c, d)$ for 8 patterns. The 8 remaining patterns have the same values of AD as their negated versions and have thus been ignored. The natural extension to $\mathbb{B}^m$ is again to consider component-wise proportions. The analogical dissimilarity in $\mathbb{B}^m$ is

a	b	c	d	$\mathrm{AD}(a,b,c,d)$
0	0	0	0	**0**
0	1	0	1	**0**
0	0	1	1	**0**
0	0	0	1	**1**
0	0	1	0	**1**
0	1	0	0	**1**
1	0	0	0	**1**
0	1	1	0	**2**

Table 3.1: The values of $\mathrm{AD}(a,b,c,d)$ for 8 patterns of a,b,c,d in $\mathbb{B}$. The 8 remaining patterns are the negated versions of these.

thus defined as the sum of the m individual ADs:

$$\mathrm{AD}(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = \sum_{i=1}^{m} \mathrm{AD}(a_i,b_i,c_i,d_i).$$

We get an analogical dissimilarity whose co-domain is $[0,2m]$. In fact, if we use the 1-norm defined as $\|\mathbf{x}\|_1 = \sum_{i=1}^{m} |x_i|$, Definition 3.5 is equivalent to Definition 3.6. This means that for $\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}$ in $\mathbb{B}^m$, $\mathrm{AD}(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = \|(\mathbf{a}-\mathbf{b})-(\mathbf{c}-\mathbf{d})\|_1$. Note that the expression $\|(\mathbf{a}-\mathbf{b})-(\mathbf{c}-\mathbf{d})\|_1$ is nothing but the Hamming distance between the two vectors[2] $(\mathbf{a}-\mathbf{b})$ and $(\mathbf{c}-\mathbf{d})$. Actually, we could use any p-norm in $\mathbb{B}^m$, so Definition 3.5 could also be used to give a more general definition of the analogical dissimilarity in $\mathbb{B}^m$. Here again, we are witnessing the close bond that links the Boolean proportion and the arithmetic proportion.

As a measure of *how poorly an analogical proportion holds*, the analogical dissimilarity will help us to define more flexible classifiers. The main underlying idea is to consider *approximate* analogies which are not valid stricto sensu, but not too far to be valid. In [BMD07], after defining analogical dissimilarity, the authors build an extended classifier allowing classification of elements that do not belong to $\mathbf{E}_S^f$. Algorithm 2 gives a description of their classifier.

This algorithm is similar to the conservative one, but instead of looking for pure, flawless proportions, we allow for some analogies not to be perfect when we need to. This translates into the fact that we now only look for 3-tuples in S^3 such that the class equation is solvable. We do not care if they are in perfect proportion with $\mathbf{x}$. Such 3-tuples **always** exist in S^3, simply because we can choose any 3-tuple $(\mathbf{a},\mathbf{a},\mathbf{a})$ with $\mathbf{a} \in S$. Hence, Algorithm 2 is able to predict the label of any element $\mathbf{x} \in X^m$.

For the sake of simplicity, we ignored here a small but relevant detail: in their implementation

[2] $\mathbb{B}^m$ is not closed under addition (or subtraction), so technically it may happen that $(\mathbf{a}-\mathbf{b}),(\mathbf{c}-\mathbf{d})$, or their difference are not in $\mathbb{B}^m$ but in $\mathbb{R}^m$.

Algorithm 2 The extended classifier.

Input: A training set S, an element $\mathbf{x} \in X^m$ for which $f(\mathbf{x})$ is unknown, and a constant $k > 0$.
Output: $\hat{f}(\mathbf{x})$, an estimation of $f(\mathbf{x})$.
Init: $\mathbf{C}(\mathbf{x}) \leftarrow \varnothing$ // A multiset of candidate labels.
for all $(\mathbf{a}, \mathbf{b}, \mathbf{c}) \in S^3$ such that $f(\mathbf{a}) : f(\mathbf{b}) :: f(\mathbf{c}) : y$ is solvable **do**
  compute $\mathrm{AD}(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{x})$ and store it
end for
for all k least values of $\mathrm{AD}(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{x})$ **do**
  $y \leftarrow \mathrm{sol}\,(f(\mathbf{a}), f(\mathbf{b}), f(\mathbf{c}))$
  $\mathbf{C}(\mathbf{x}) \leftarrow \mathbf{C}(\mathbf{x}) \cup y$
end for
$\hat{f}(\mathbf{x}) \leftarrow \mathrm{Mode}(\mathbf{C}(\mathbf{x}))$ // The most common value in $\mathbf{C}(\mathbf{x})$

[BMD07], the authors actually look for all the 3-tuples that have the same analogical dissimilarity as the k^{th} one. For example if the k^{th} 3-tuple has an AD of 5, they will actually consider all the 3-tuples that have an AD less than or equal to 5, bringing the number of candidates to k' with $k' \geq k$. This is fortunate because it allows their extended classifier to fit with the previous conservative approach in the case of a perfect proportion (or equivalently, in the case where $\mathbf{x} \in \mathbf{E}_S^f$). Indeed, when the k^{th} 3-tuple $(\mathbf{a}, \mathbf{b}, \mathbf{c})$ has an AD of 0, i.e. $\mathbf{a} : \mathbf{b} :: \mathbf{c} : \mathbf{x}$ stands, they will consider all other 3-tuples $(\mathbf{a}', \mathbf{b}', \mathbf{c}')$ such that $\mathbf{a}' : \mathbf{b}' :: \mathbf{c}' : \mathbf{x}$, which is consistent with the conservative approach. Clearly, the extended classifier is a generalization of the conservative one.

Example:
To better grasp the behaviour of the extended classifier, we will consider again Figure 3.1 that we repeat here for the sake of convenience on page 61. Let us go through the classification process of $\mathbf{c}, \mathbf{f}, \mathbf{g}$ and $\mathbf{h}$. We will here consider k to be equal to 1. For $\mathbf{c}$, the 3-tuple in S^3 with the least value of AD is $(\mathbf{b}, \mathbf{d}, \mathbf{a})$ with $\mathrm{AD}(\mathbf{b}, \mathbf{d}, \mathbf{a}, \mathbf{c}) = 0$. The set $\mathbf{C}(\mathbf{c})$ is thus $\{\, 1 \,\}$, and leads to the prediction $\hat{f}(\mathbf{c}) = 1$. For $\mathbf{f}$, the best 3-tuple is $(\mathbf{a}, \mathbf{b}, \mathbf{e})$ and leads to the prediction $\hat{f}(\mathbf{f}) = 1$. For these two elements in $\mathbf{E}_S^f$, the extended classifier outputs the same predictions as the conservative one, and most importantly, the underlying process is the same (to some extent, this is due to the fact that we have chosen $k = 1$).

Consider now $\mathbf{g} \notin \mathbf{E}_S^f$. We know that we won't find any 3-tuple in perfect proportion with $\mathbf{g}$ (else the conservative classifier would have been able to output $\hat{f}(\mathbf{g})$), but the 3-tuple $(\mathbf{e}, \mathbf{e}, \mathbf{e}) \in S^3$ has $\mathrm{AD}(\mathbf{e}, \mathbf{e}, \mathbf{e}, \mathbf{c}) = 1$. We have $k = 1$, so strictly following Algorithm 2 would lead us to stick to this only 3-tuple. However, considering the above remark we are allowed to look for all other 3-tuples that also have an AD of 1. Actually, the 3-tuple $(\mathbf{b}, \mathbf{d}, \mathbf{a})$ also has $\mathrm{AD}(\mathbf{b}, \mathbf{d}, \mathbf{a}, \mathbf{g}) = 1$, as can be verified on table 3.2. We thus have $\mathbf{C}(\mathbf{g}) = \{\, 1, 1 \,\}$: one solution comes from the solving of $f(\mathbf{e}) : f(\mathbf{e}) :: f(\mathbf{e}) : y$ and the other solution comes from the solving of $f(\mathbf{b}) : f(\mathbf{d}) :: f(\mathbf{a}) : y$. Finally, we have $\hat{f}(\mathbf{g}) = 1$. As for $\mathbf{h}$, we have $\mathbf{C}(\mathbf{h}) = \{\, 1, 1 \,\}$: one

b	**d**	**a**	**g**	$\mathrm{AD}(b_i, d_i, a_i, g_i)$
0	0	0	1	**1**
0	1	0	1	**0**
1	1	0	0	**0**

Table 3.2: $\mathrm{AD}(\mathbf{b}, \mathbf{d}, \mathbf{a}, \mathbf{g}) = 1$.

solution comes from $f(\mathbf{a}) : f(\mathbf{b}) :: f(\mathbf{e}) : y$ and the other comes from $f(\mathbf{d}) : f(\mathbf{d}) :: f(\mathbf{d}) : y$, and we end up with $\hat{f}(\mathbf{h}) = 1$.

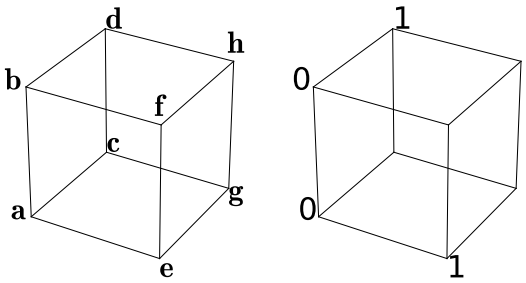

Figure 3.1: With $S = \{ \mathbf{a}, \mathbf{b}, \mathbf{d}, \mathbf{e} \}$, we have $\mathbf{E}_S^{f*} = \{ \mathbf{c}, \mathbf{f} \}$. (Repeated from page 55).

In [BMD07], the authors evaluated this classifier on a Boolean setting $\mathbb{B}^m$ over 8 benchmarks from the UCI repository. This approach led to remarkable results in terms of accuracy, when compared to off-the-shelf standard classifiers. Nonetheless, this algorithmic description of the extended classifier does not allow us to grasp its inherent working behaviour, and it is difficult to extract theoretical properties. The aim of the next subsection is to give a functional translation of this algorithmic description.

3.1.3 Analogical classifier: a functional definition

As we have previously seen, we can generally assume that $\mathrm{AD}(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}) \stackrel{\mathsf{def}}{=} \|(\mathbf{a} - \mathbf{b}) - (\mathbf{c} - \mathbf{d})\|_p$, whether we are dealing with the arithmetic proportion or the Boolean proportion. A key observation is that we can formulate this expression with a distance function on X^m. Indeed, a simple rewriting leads to the following expression:

$$\mathrm{AD}(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}) = \|\mathbf{d} - (\mathbf{c} - \mathbf{a} + \mathbf{b})\|_p = \|\mathbf{d} - \mathbf{d}'\|_p,$$

where $\mathbf{d}' \stackrel{\mathsf{def}}{=} \mathbf{c} - \mathbf{a} + \mathbf{b}$. Actually, $\mathbf{d}'$ is nothing but the 4^{th} vertex of the parallelogram $\mathbf{abcd}'$, or equivalently, $\mathbf{d}'$ is the solution of the equation $\mathbf{a} : \mathbf{b} :: \mathbf{c} : \mathbf{x}$. All this means that $\mathrm{AD}(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d})$ simply is the distance between $\mathbf{d}$ and this 4^{th} vertex $\mathbf{d}'$: $\mathrm{AD}(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}) = \delta(\mathbf{d}, \mathbf{d}')$, where δ is

the distance defined by the p-norm. Figure 3.4 schematically sums up all the aforementioned properties. Note that when $(\mathbf{a}, \mathbf{b}, \mathbf{c}) \in S^3$ and when the associated class equation is solvable, we naturally have $\mathbf{d}' \in \mathbf{E}_S^f$.

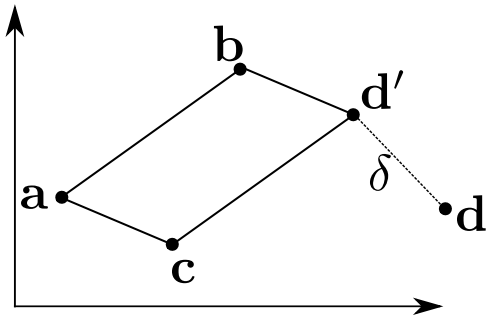

Figure 3.4: The analogical dissimilarity $\mathrm{AD}(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d})$ is equal to the distance $\delta(\mathbf{d}, \mathbf{d}')$.

Example:
Let us consider (again) Figure 3.1 on page 61. We have already seen that $\mathrm{AD}(\mathbf{b}, \mathbf{d}, \mathbf{a}, \mathbf{g}) = 1$, as proved by Table 3.2 on page 61. We now can reinterpret this statement: $\mathrm{AD}(\mathbf{b}, \mathbf{d}, \mathbf{a}, \mathbf{g})$ should be equal to the distance between $\mathbf{g}$ and another vector that makes up a perfect parallelogram with the 3-tuple $(\mathbf{b}, \mathbf{d}, \mathbf{a})$. This vector is naturally $\mathrm{sol}(\mathbf{b}, \mathbf{d}, \mathbf{a}) = \mathbf{c}$ and have indeed $H(\mathbf{c}, \mathbf{g}) = \mathrm{AD}(\mathbf{b}, \mathbf{d}, \mathbf{a}, \mathbf{g}) = 1$, where H is the hamming distance! The same can be observed for $\mathbf{h}$: $\mathrm{AD}(\mathbf{a}, \mathbf{b}, \mathbf{e}, \mathbf{h}) = H(\mathbf{h}, \mathbf{f})$, where $\mathbf{f} = \mathrm{sol}(\mathbf{a}, \mathbf{b}, \mathbf{e})$.

We are now finally ready for our unifying functional definition of analogical classifiers. As we have seen, for a given $\mathbf{x} \in X^m$, Algorithm 2 tries to minimize $\mathrm{AD}(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{x})$ over all the 3-tuples $(\mathbf{a}, \mathbf{b}, \mathbf{c}) \in S^3$. In the light of what has just been explained, we see that this is equivalent to finding the closest vertex $\mathbf{d}' \in \mathbf{E}_S^f$ from $\mathbf{x}$ for any $(\mathbf{a}, \mathbf{b}, \mathbf{c}) \in S^3$. In other words, **looking for the k 3-tuples in S^3 that have the smallest value of AD with $\mathbf{x}$ amounts to finding the k closest elements to $\mathbf{x}$ in $\mathbf{E}_S^f$**. We will call these elements the k **nearest analogical neighbors** of $\mathbf{x}$:

Definition 3.7 (The k nearest analogical neighbors)**.** For any $\mathbf{x} \in X^m$, and any training set $S \subsetneq X$, we define the k **nearest analogical neighbors of $\mathbf{x}$** as the set:

$$k\text{-nan}\left(\mathbf{x}, S\right) \stackrel{\mathsf{def}}{=} \left\{ \underset{\mathbf{d}' \in A_E^Y(S)}{\overset{k}{\arg\min}}\ \delta(\mathbf{x}, \mathbf{d}') \right\},$$

where $\arg\min_{\Sigma}^{k} \delta(\mathbf{x}, \cdot)$ returns the k elements in Σ with the least value of δ. Another equivalent way of defining k-nan is:

$$k\text{-nan}\left(\mathbf{x}, S\right) \stackrel{\mathsf{def}}{=} k\text{-nn}\left(\mathbf{x}, \mathbf{E}_S^f\right),$$

where $k\text{-nn}\left(\mathbf{x}, \Sigma\right)$ is the set of k nearest neighbors of $\mathbf{x}$ in the set Σ. Ties are broken at random.

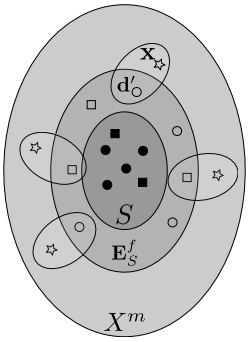

Figure 3.5: Classification process of an extended analogical learner. $\hat{f}(\mathbf{x}) = \overline{f}(\mathbf{d}')$.

For any $\mathbf{x} \in X^m$, the elements in k-nan$(\mathbf{x}, S)$ all belong to $\mathbf{E}_S^f$, so they all have an analogical label. The prediction $\hat{f}(\mathbf{x})$ of an extended classifier is thus nothing but the most frequent analogical label:

Definition 3.8 (Extended classifier)**.** Given a training set S with an underlying ground truth function f, an **extended classifier** with parameter k sets the prediction of an element $\mathbf{x} \in X^m$ as follows:

$$\hat{f}(\mathbf{x}) \stackrel{\text{def}}{=} \text{Mode}\left\{ \overline{f}(\mathbf{d}') \;\middle|\; \mathbf{d}' \in k\text{-nan}\,(\mathbf{x}, S) \right\}.$$

In some sense, an analogical classifier behaves as a NN classifier[3] but on an extended training set. The above definitions lead us to understand the process of analogical classification as follows:

1. First, extend the training set S to its analogical extension $\mathbf{E}_S^f$. $\mathbf{E}_S^f$ can be viewed as an extended training set that has **class noise**: the labels associated with elements in $\mathbf{E}_S^{f*}$ are their analogical labels as defined in Definition 3.3, and they may not be equal to the ground truth value $f(\mathbf{x})$.
2. Then, simply apply a classical k-NN strategy over this extended training set.

Figure 3.5 gives an illustration of the classification process for $k = 1$: the label of $\mathbf{x} \in X$ is unknown, and we set it to that of $\mathbf{d}' \in \mathbf{E}_S^f$ (a circle), which is its nearest analogical neighbor. To show that the analogical label of $\mathbf{d}'$ has itself been inferred, it is depicted as transparent instead of plain black. Let us note that topologically speaking, Figure 3.5 is not representative of a real case: even if we always have $S \subseteq \mathbf{E}_S^f \subseteq X^m$, this does not mean that these sets are embedded into one another as shown in the drawing. Actually, if the underlying probability distribution of X^m is uniform, the elements of S (and thus of $\mathbf{E}_S^f$) are scattered over the whole universe.

[3] In the following, NN and NAN will denote classifiers. k-nn and k-nan denote sets, as in Definition 3.7.

Example:
Going back (for the last time!) to Figure 3.1, our new functional view allows us to consider the following steps:

1. First extend $S = \{ \mathbf{a}, \mathbf{b}, \mathbf{d}, \mathbf{e} \}$ to $\mathbf{E}_S^f = \{ \mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{e} \}$ with $\hat{f}(\mathbf{c}) = \overline{f}(\mathbf{c}) = 1$ and $\hat{f}(\mathbf{f}) = \overline{f}(\mathbf{f}) = 1$.
2. Then, to classify $\mathbf{g}$, we simply look to its nearest neighbors in $\mathbf{E}_S^f$, which are $\mathbf{c}$ and $\mathbf{e}$. The aggregation of their analogical labels leads to $\hat{f}(\mathbf{g}) = 1$. The nearest neighbors of $\mathbf{h}$ are $\mathbf{d}$ and $\mathbf{f}$, which also leads to $\hat{f}(\mathbf{h}) = 1$.

As far as we know, this is the first time a functional definition of analogy-based classifiers is given. This definition clearly fits with the known algorithms but obviously, some implementation details cannot be exactly caught up by such a high level description. It is indeed possible to find a few edge cases where this functional definition may not output the same result as algorithm 2: this is the case for example when the nearest analogical neighbor (nan) of $\mathbf{x}$ is not unique. It is also the case with the Boolean proportion when the closest vertex $\mathbf{d}'$ does not belong to $\mathbb{B}^m$. However, as we will see later, these cases are not likely to occur and the two approaches (that of Algorithm 2 and that of Definition 3.8) produce very similar results, thus empirically validating this functional definition.

Since we now have a clear functional definition of analogical classifiers, we are in position to examine some general theoretical properties of analogical learners. This is the purpose of the next section.

3.2 Theoretical properties of analogical classifiers

The functional definition of analogical learners opens the door to the study of various theoretical properties of analogical classifiers, which had only been described algorithmically so far. Just like for the k-NN literature, theoretical results are easier to derive when $k = 1$. We will therefore mostly focus on the properties of the 1-NAN classifier.

3.2.1 Study of convergence

Let us consider the case where $X^m = \mathbb{R}^m$, and let δ be any distance function defined from a norm. Simply because $S \subseteq \mathbf{E}_S^f$ for any S, the following inequality holds for any $\mathbf{x} \in X^m$:

$$\delta(\mathbf{x}, \text{1-nan}(\mathbf{x}, S)) \leq \delta(\mathbf{x}, \text{1-nn}(\mathbf{x}, S)).$$

Now, let us consider $\mathbf{x}^{(i)}$ an i.i.d. sequence of random variables in $\mathbb{R}^m$, where $\mathbb{R}^m$ is equipped with a probability measure denoted P. The training set will be denoted $S_n = \left\{ \left(\mathbf{x}^{(i)}, f(\mathbf{x}^{(i)}) \right), \ i \in [1, n] \right\}$. As S_n is random, then 1-nan$(\mathbf{x}, S_n)$ can also be considered as a

random element of X^m. We then are exactly in the same context as the work of Cover & Hart who studied the 1-NN algorithm ([CH67]), and we obtain a similar result:

Property 3.1. *For any* $\mathbf{x}$*, its nan* 1-nan$(\mathbf{x}, S_n)$ *converges almost surely towards* $\mathbf{x}$ *as* n *tends to infinity:*

$$P\left(\lim_{n\to\infty} \text{1-nan}(\mathbf{x}, S_n) = \mathbf{x}\right) = 1.$$

Proof. The proof can be easily derived from the results of Cover & Hart in [CH67]. Indeed, they have already showed that the nearest neighbor of $\mathbf{x}$ converges almost surely towards $\mathbf{x}$ as n tends to infinity:

$$P\left(\lim_{n\to\infty} \text{1-nn}(\mathbf{x}, S_n) = \mathbf{x}\right) = 1.$$

To prove this result, they simply start from the fact that $\delta(\mathbf{x}, \text{1-nn}(\mathbf{x}, S_n))$ converges in probability towards 0:

$$\forall \varepsilon > 0, \quad \lim_{n\to\infty} P\Big(\delta(\mathbf{x}, \text{1-nn}(\mathbf{x}, S_n)) > \varepsilon\Big) = 0.$$

Using the previous fact that $\delta(\mathbf{x}, \text{1-nan}(\mathbf{x}, S_n)) \leq \delta(\mathbf{x}, \text{1-nn}(\mathbf{x}, S_n))$, the convergence in probability of $\delta(\mathbf{x}, \text{1-nan}(\mathbf{x}, S_n))$ towards 0 is immediate. Then, it naturally follows that 1-nan$(\mathbf{x}, S_n)$ converges almost surely towards $\mathbf{x}$, following the same steps as Cover & Hart. □

Let us note the following points:

1. The result of Cover and Hart was actually proved in a less restrictive setting than ours. They have proven their result for any separable metric space, without any additional structure. Unfortunately, we are not able to follow these lines here, because there is no known way to define an analogical dissimilarity on a metric space, without the help of any other structure or operator (see [MBD08] for a detailed discussion on this issue).
2. Cover & Hart proved that when n tends to infinity, the probability of error of the 1-NN classifier is less than twice that of the Bayes probability of error, and therefore less than twice the probability of error of any other classifier. We have not been able to prove such a strong result for the 1-NAN classifier, but its probability of error will be thoroughly investigated in Section 3.2.3.
3. We have to be careful about the interpretation of Property 3.1 in terms of machine learning. Indeed, another key property is proved in [CH67]: for an integrable function f over $\mathbb{R}^m$ with regards to the probability measure P, the expectation of $f(\text{1-nn}(\mathbf{x}, S_n)) - f(\mathbf{x})$ converges to 0 when n tends to infinity. This means that asymptotically, the nearest neighbor of $\mathbf{x}$ has the same properties as $\mathbf{x}$, and particularly: **they have the same label**, which is a very powerful property when we are dealing with a classification task. Sadly, such a property has not been proven for 1-nan$(\mathbf{x}, S_n)$.
4. Finally, it is clear that when n goes to infinity, the behavior of an analogical classifier tends to that of a nearest neighbors classifier. Indeed, when S_n is very big, the nearest analogical neighbor of an element $\mathbf{x}$ simply is its nearest neighbor, in most cases. Moreover, when

the nan and the nn are too close, paying the price of the noise related to the nan may not be worth it. This supports the common acknowledgement that analogical reasoning is mostly useful when very few data are available. In this latter case, extending a small training set with its analogical extension may be particularly beneficial, as we will see in Chapter 6.

To conclude this discussion, even if the convergence result is interesting in itself, it does not say a lot in terms of machine learning. A much more interesting and powerful theoretical result of analogical classifiers will be proven in Chapter 6. Let us now focus on their VC-dimension.

3.2.2 Vapnik Chervonenkis (VC) dimension of analogical classifiers

The notion of VC-dimension was originally defined by Vapnik and Chervonenkis [Vap98], and introduced into learnability theory by Blumer et al. [BEHW89]. Roughly speaking, the VC-dimension (denoted VC) of a class of learners is a numerical measure of their discrimination power. One of the main results about the VC-dimension is that it allows to bound the true risk R of a learner. With probability $1-\eta$, and under some theoretical conditions that we omit here, we have:

$$R \leq R_{\text{emp}} + \sqrt{\frac{1}{n} \cdot [\text{VC}(\log(2n/\text{VC}) + 1) - \log(\eta/4)]},$$

where R_{emp} is the probability of error on a training set S_n, and VC is the VC-dimension of the learner. In our setting,

- $R \stackrel{\text{def}}{=} P\left(\hat{f}(\mathbf{x}) \neq f(\mathbf{x}) \mid \mathbf{x} \in X^m\right)$, and
- $R_{\text{emp}} \stackrel{\text{def}}{=} P\left(\hat{f}(\mathbf{x}) \neq f(\mathbf{x}) \mid \mathbf{x} \in S_n\right)$.

This theoretical upper bound on the true risk R is what makes the VC-dimension of a class of learners an essential element of their theoretical study. The VC-dimension is defined as follows:

Definition 3.9 (VC-dimension)**.** We consider a class of learners $\mathcal{C}$ and a set S_n of n labeled elements in X^m. There are exactly 2^n ways to label each of the n elements. If for any labeling, there exists a learner in $\mathcal{C}$ that correctly labels all the elements in S_n, we say that $\mathcal{C}$ **shatters** S_n. The **VC-dimension** of $\mathcal{C}$ is the biggest n such that there exists a set of n points S_n that can be shattered by $\mathcal{C}$. When there exists a set S_n that can be shattered for any size n, we say that the VC-dimension is infinite.

Let's first consider the class of nearest neighbors classifiers: $\mathcal{C}_{\text{NN}} = \{k\text{-NN classifiers}, k \in \mathbb{N}^*\}$. It is clear that the VC-dimension of $\mathcal{C}_{\text{NN}}$ is infinite: taking $k = 1$, any training set of any size can be perfectly labeled by 1-NN, because every element is its own nearest neighbor. Actually, we can easily show the same result for the analogical learners.

Proposition 3.1. *The class $\mathcal{AC}_k$ of analogical learners has infinite VC-dimension.*

Proof. Considering Definition 3.8 on page 63 with $k = 1$, we see that the predicted label of any element $\mathbf{x}$ is that of its nan. For any element $\mathbf{x}$ in S_n, $\mathbf{x}$ is its own nan (just like $\mathbf{x}$ is its own nearest neighbor). Hence, any training set S_n of size n is correctly labeled by 1-NAN, and thus can be shattered by $\mathcal{AC}_k$. Therefore, the VC-dimension of $\mathcal{AC}_k$ is infinite. □

Clearly, the upper bound of the true risk is a decreasing function in n, and an increasing function in the VC dimension of the learner. As such, we usually want the VC-dimension of our classifiers to be small. However, we have previously seen that the nearest neighbors classifiers, despite their infinite VC-dimension, enjoy many interesting theoretical properties, and their practical efficiency is well-acknowledged. In fact, the upper bound on the true risk does not even hold for classes of learner whose VC-dimension is infinite [Bur98]. We then need to keep in mind that even though the analogical learners have an infinite VC-dimension, which is an interesting result in its own right, this does not necessarily prevent them from being useful classifiers.

3.2.3 Accuracy analysis

In this section, we study the accuracy of an analogical classifier, and more particularly that of the NAN classifier. As explained in Section 3.1.3, for a given $\mathbf{x} \in X^m$ and a training set $S \subsetneq X^m$, the prediction $\hat{f}(\mathbf{x})$ is set as:

$$\hat{f}(\mathbf{x}) \stackrel{\text{def}}{=} \overline{f}(\text{1-nan}(\mathbf{x}, S)) \stackrel{\text{def}}{=} \overline{f}(\text{1-nn}(\mathbf{x}, \mathbf{E}_S^f)).$$

We now equip the set X^m with a probability distribution denoted P. The accuracy of the NAN_S classifier[4] over all the elements of X^m is defined as:

$$\text{Acc}(\text{NAN}_S, X^m) \stackrel{\text{def}}{=} P\left(\hat{f}(\mathbf{x}) = f(\mathbf{x}) \mid \mathbf{x} \in X^m\right).$$

Our main result is given by Proposition 3.2:

Proposition 3.2. *The accuracy of NAN_S over X^m can be seen as the weighted sum of the accuracy of NN over A and B, using a different training set each time (respectively S and $\mathbf{E}_S^{f*}$):*

$$\begin{aligned} Acc(NAN_S, X^m) = \; & Acc(NN_S, A) \cdot \alpha + \\ & Acc(NN_{\mathbf{E}_S^{f*}}, B) \cdot (1 - \alpha), \end{aligned}$$

where:

- $A \stackrel{\text{def}}{=} \{\, \mathbf{x} \in X^m \mid \text{1-}nan(\mathbf{x}, S) \in S \,\}$*: the elements that have their nan in S.*
- $B \stackrel{\text{def}}{=} \{\, \mathbf{x} \in X^m \mid \text{1-}nan(\mathbf{x}, S) \in \mathbf{E}_S^{f*} \,\}$*: the elements that have their nan in $\mathbf{E}_S^{f*}$.*

[4]The S subscript is here to specify that the training set of the NAN algorithm is S. The same notation is used for the *nearest neighbor* algorithm: NN_Σ is the NN algorithm trained on the set Σ.

- $\alpha \stackrel{\text{def}}{=} P(\mathbf{x} \in A)$ *and* $1 - \alpha = P(\mathbf{x} \in B)$.
  Naturally, $A \cup B = X^m$ *and* $A \cap B = \varnothing$.

Proof. By observing that for any $\mathbf{x}$, its 1-nan either belongs to S or to $\mathbf{E}_S^{f*}$, the accuracy $\text{Acc}(\text{NAN}_S, X^m)$ can be split into two distinct parts as follows:

$$\begin{aligned}
& \text{Acc}(\text{NAN}_S, X^m) \stackrel{\text{def}}{=} P\left(\hat{f}(\mathbf{x}) = f(\mathbf{x}) \mid \mathbf{x} \in X^m\right) = P\left(\overline{f}(\text{1-nan}(\mathbf{x}, S)) = f(\mathbf{x})\right) \\
= \; & P\left([\overline{f}(\text{1-nan}(\mathbf{x}, S)) = f(\mathbf{x})] \wedge [\text{1-nan}(\mathbf{x}, S) \in S]\right) \; + \\
& P\left([\overline{f}(\text{1-nan}(\mathbf{x}, S)) = f(\mathbf{x})] \wedge [\text{1-nan}(\mathbf{x}, S) \in \mathbf{E}_S^{f*}]\right) \\
= \; & P\left([\overline{f}(\text{1-nn}(\mathbf{x}, \mathbf{E}_S^f)) = f(\mathbf{x})] \wedge [\text{1-nn}(\mathbf{x}, \mathbf{E}_S^f) \in S]\right) \; + \\
& P\left([\overline{f}(\text{1-nn}(\mathbf{x}, \mathbf{E}_S^f)) = f(\mathbf{x})] \wedge [\text{1-nn}(\mathbf{x}, \mathbf{E}_S^f) \in \mathbf{E}_S^{f*}]\right) \\
= \; & P\left([\overline{f}(\text{1-nn}(\mathbf{x}, \mathbf{E}_S^f)) = f(\mathbf{x})] \mid [\text{1-nn}(\mathbf{x}, \mathbf{E}_S^f) \in S]\right) \times P\left([\text{1-nn}(\mathbf{x}, \mathbf{E}_S^f) \in S]\right) \; + \\
& P\left([\overline{f}(\text{1-nn}(\mathbf{x}, \mathbf{E}_S^f)) = f(\mathbf{x})] \mid [\text{1-nn}(\mathbf{x}, \mathbf{E}_S^f) \in \mathbf{E}_S^{f*}]\right) \times P\left([\text{1-nn}(\mathbf{x}, \mathbf{E}_S^f) \in \mathbf{E}_S^{f*}]\right)
\end{aligned}$$

Let us denote $\alpha \stackrel{\text{def}}{=} P\left(\text{1-nn}(\mathbf{x}, \mathbf{E}_S^f) \in S\right)$. Obviously, we also have $\alpha \stackrel{\text{def}}{=} P\left(\text{1-nan}(\mathbf{x}, S) \in S\right)$. The above formula becomes:

$$\begin{aligned}
& \text{Acc}(\text{NAN}_S, X^m) = \\
& P\left([\overline{f}(\text{1-nn}(\mathbf{x}, \mathbf{E}_S^f)) = f(\mathbf{x})] \mid [\text{1-nn}(\mathbf{x}, \mathbf{E}_S^f) \in S]\right) * \alpha \; + \\
& P\left([\overline{f}(\text{1-nn}(\mathbf{x}, \mathbf{E}_S^f)) = f(\mathbf{x})] \mid [\text{1-nn}(\mathbf{x}, \mathbf{E}_S^f) \in \mathbf{E}_S^{f*}]\right) * (1 - \alpha).
\end{aligned}$$

Let us focus on the first term, temporarily discarding the factor α:

$$P\left([\overline{f}(\text{1-nn}(\mathbf{x}, \mathbf{E}_S^f)) = f(\mathbf{x})] \mid [\text{1-nn}(\mathbf{x}, \mathbf{E}_S^f) \in S]\right).$$

It is easy to see that the event $[\text{1-nn}(\mathbf{x}, \mathbf{E}_S^f) \in S]$ is equivalent to the event $[\text{1-nn}(\mathbf{x}, \mathbf{E}_S^f) = \text{1-nn}(\mathbf{x}, S)]$. As a result, we can transform the first term to get a better grasp of its meaning:

$$\begin{aligned}
& P\left([\overline{f}(\text{1-nn}(\mathbf{x}, \mathbf{E}_S^f)) = f(\mathbf{x})] \mid [\text{1-nn}(\mathbf{x}, \mathbf{E}_S^f) \in S]\right) \\
= \; & P\left([\overline{f}(\text{1-nn}(\mathbf{x}, \mathbf{E}_S^f)) = f(\mathbf{x})] \mid [\text{1-nn}(\mathbf{x}, \mathbf{E}_S^f) = \text{1-nn}(\mathbf{x}, S)]\right) \\
= \; & P\left([\overline{f}(\text{1-nn}(\mathbf{x}, S)) = f(\mathbf{x})] \mid [\text{1-nn}(\mathbf{x}, \mathbf{E}_S^f) \in S]\right).
\end{aligned}$$

In this form, the first term is just the accuracy of the NN_S algorithm over the elements that have their nearest analogical neighbor in S. As for the second term, the same process can be applied by observing that the event $[\text{1-nn}(\mathbf{x}, \mathbf{E}_S^f) \in \mathbf{E}_S^{f*}]$ is equivalent to the event

$[\text{1-nn}(\mathbf{x}, \mathbf{E}_S^f) = \text{1-nn}(\mathbf{x}, \mathbf{E}_S^{f*})]$. This leads to

$$\begin{aligned} & P\left([\overline{f}(\text{1-nn}(\mathbf{x}, \mathbf{E}_S^f)) = f(\mathbf{x})] \mid [\text{1-nn}(\mathbf{x}, \mathbf{E}_S^f) \in \mathbf{E}_S^{f*}]\right) \\ = \; & P\left([\overline{f}(\text{1-nn}(\mathbf{x}, \mathbf{E}_S^{f*})) = f(\mathbf{x})] \mid [\text{1-nn}(\mathbf{x}, \mathbf{E}_S^f) \in \mathbf{E}_S^{f*}]\right). \end{aligned}$$

This second term is then the accuracy of the $\text{NN}_{\mathbf{E}_S^{f*}}$ algorithm over the elements that have their nearest analogical neighbor in $\mathbf{E}_S^{f*}$.

In the light of these interpretations, we can directly rewrite the accuracy formula in the concise form given in Proposition 3.2. □

The value $\text{Acc}(\text{NN}_S, A)$ is the accuracy of NN_S over all the elements in A. A theoretical study of this accuracy has been done in [LI93] when the size of A is known. Regarding $\text{Acc}(\text{NN}_{\mathbf{E}_S^{f*}}, B)$, this is the accuracy of 1-NN when the training set is noisy, and has been studied in [OY97]. This last formula leads to the consistent facts:

1. The smaller $\mathbf{E}_S^{f*}$ (i.e. analogical reasoning does not bring much more labels), the closer α is to 1, the closer A is to X^m and the more the accuracy of NAN_S tends towards the accuracy of NN_S over X^m.
2. In return, if $\mathbf{E}_S^f$ is much bigger than S, α is then small, B is close to X^m and the accuracy of NAN_S greatly depends on the quality of $\mathbf{E}_S^f$:

Definition 3.10 (Quality of the analogical extension ω_S^f)**.** Given a training set $S \subsetneq X^m$ and a ground truth function f, the **quality** of the analogical extension $\mathbf{E}_S^f$ is defined as:

$$\omega_S^f \stackrel{\text{def}}{=} P\left(\overline{f}(\mathbf{x}) = f(\mathbf{x}) \mid \mathbf{x} \in \mathbf{E}_S^{f*}\right).$$

Note that the value $1 - \omega_S^f$ corresponds to the class noise of $\mathbf{E}_S^{f*}$.

This closes our contributions to the theoretical properties of analogical classifiers. In the next section, we will empirically investigate and illustrate the soundness of our results.

3.3 Experiments and empirical validation

In order to get an empirical validation of our formulas, we have developed a set of experiments that we now describe.

3.3.1 Evaluation protocol

Working with Boolean vectors, we have computed the accuracies of the 1-NAN and 1-NN algorithms over $X^m = \mathbb{B}^m$ for $m = 8$. Other values of m have been investigated, leading to very similar results. The ground truth label of the elements of X^m is defined by different Boolean functions f:

- $f_1(\mathbf{x}) = x_i$ for some i. These functions are called projections, or dictators. They lead to perfectly linearly separable problems, i.e. there exists an hyperplane that can perfectly discriminate both classes. As such, they are expected to be fairly easy to learn.
- $f_2(\mathbf{x}) = \bigoplus_{i=1}^{k} x_i$ for some $k \in [1, m]$, where $\oplus$ is the XOR operator (which is associative). These functions lead to highly non-linearly-separable problems: remember the example of Figure 2.6 on page 43, who was an instance of a XOR problem. XOR problems are traditionally difficult to learn.
- $f_3(\mathbf{x}) = \bigwedge_{i=1}^{k} x_i$ for some $k \in [1, m]$.
- $f_4(\mathbf{x}) = \bigvee_{i=1}^{k} x_i$ for some $k \in [1, m]$.
- $f_5(\mathbf{x}) = [\sum_{i=1}^{m} x_i \geq k]$ for some $k \in [1, m]$, where the function [cond] is equal to 1 if cond is verified and 0 else. Simply put, $f(\mathbf{x}) = 1$ iff at least k components are equal to 1.
- $f_6(\mathbf{x}) = [\sum_{i=1}^{m} x_i = k]$ for some $k \in [1, m]$. Simply put, $f(\mathbf{x}) = 1$ iff exactly k components are equal to 1.

Regarding the size of the training set, to be sure to fit with the size of the universe, we have investigated various sizes between 3 and 100. When dealing with a training set of size 100, the cubic complexity of the analogical classifier leads to explore a set of approximately 100^3 elements: as a consequence, we limit our investigation to a maximum of 100 elements in the training set in order to get a realistic execution time.

All the accuracy (and other metrics) computations are averaged over a set of 100 experiments. We have made sure that the class distributions of the training sets are as close as possible to that of the whole universe X^m. Note that our implementation of the NAN algorithm is not that of algorithm 2, but is instead that of the functional definition of the analogical classifier as described in Definition 3.8: we first construct the analogical extension set of S, and then proceed to a nearest neighbor strategy over this noisy extended training set. We have estimated probabilities by frequencies, thus implicitly assuming a uniform distribution on X^m.

For each experiment, we have computed the following measures to empirically investigate the behaviour of the analogical classifier, and assess our theoretical results:

- The empirical accuracy of the 1NAN classifier, defined[5] as $\text{Acc}(\text{NAN}_S) \stackrel{\text{def}}{=} P\left(\hat{f}(\mathbf{x}) = f(\mathbf{x}) \mid \mathbf{x} \in X^m \setminus S\right)$.
- The empirical accuracy of the 1NN classifier, whose definition is of course the same but this time $\hat{f}$ is the output of the 1NN classifier.
- The **theoretical** accuracy of 1NAN, as defined in Proposition 3.2. The probability $\alpha \stackrel{\text{def}}{=} P(\mathbf{x} \in A)$ has been estimated by the frequency $\frac{|A|}{|X^m|}$. Both sets A and B are easy to determine.
- The quality of $\mathbf{E}_S^f$, measured by ω_S^f as previously defined in Definition 3.10. ω_S^f has been

[5]We notice here that $\mathbf{x}$ is only required to belong to $X^m \setminus S$ and not to X^m, because classifying elements in S is pointless.

estimated by the frequency $\frac{\left|\left\{\, \mathbf{x}\in\mathbf{E}_S^{f*} \;\middle|\; \overline{f}(\mathbf{x})=f(\mathbf{x}) \,\right\}\right|}{\left|\mathbf{E}_S^{f*}\right|}$.

- Finally, the quantity $\gamma_S^f \stackrel{\text{def}}{=} P(\mathbf{x} \in \mathbf{E}_S^f)$, estimated by $\frac{|\mathbf{E}_S^f|}{|X^m|}$: the size of the analogical extension with respect to that of the whole universe.

It is fairly easy to see that both functions f_1 and f_2 split the universe X^m into two disjoint sets of size 2^{m-1}: those for which $f(\mathbf{x})$ is equal to 1, and those with labels equal to 0. We say that the datasets are balanced. However, the other functions may lead to very unbalanced datasets. An extreme case is for example f_3 with $k = m$, which leads to $f_3(\mathbf{x}) = x_1 \wedge x_2 \wedge \cdots \wedge x_m$. The only element whose label is 1 is the vector $\mathbf{x} = (1, 1, \ldots, 1)$. For these very unbalanced datasets, the accuracy of a classifier is not a relevant measure: predicting 0 all the time would lead to an accuracy of almost 1, but would systematically assign the wrong label to $(1, 1, \ldots 1)$. Therefore, as accuracy is our main concern here, we will focus on functions for which datasets are as balanced as possible.

In addition to these Boolean functions, we have also run the NAN algorithm over the Monk datasets over the UCI repository[6]. They are datasets of 432 binarized elements, among which exactly 169 of them have been used for training.

3.3.2 Comments and discussion

Our experiments are summed-up by Figure 3.6 on page 72. As they are the result of multiple averaged experiments, the measures ω_S^f and γ_S^f are here denoted ω and γ. Paying careful attention to these plots will allow us to draw interesting conclusions about the behaviour of the 1-NAN algorithm.

First, the theoretical accuracy seems to fit perfectly with the empirical accuracy of the NAN algorithm, thus validating our theoretical study that led to proposition 3.2. Actually, the maximal difference we observed between the theoretical accuracy and its actual value is of about 10^{-10}.

An interesting observation is that the value of ω seems to always converge to that of the theoretical accuracy (and therefore to the actual accuracy) of NAN. This can be easily explained by paying attention to the value of γ, the proportion of elements of X^m that belong to $\mathbf{E}_S^f$. We see that in any setting, γ converges to 1 as $|\, S \,|$ grows. This means that when $|\, S \,|$ is big enough (but not necessarily *that* big with respect to X^m), the analogical extension of S covers the whole universe X^m (obviously, the bigger the dimension m, the slower the convergence occurs): every element $\mathbf{x}$ is then its own nearest analogical neighbor and $\hat{f}(\mathbf{x}) = \overline{f}(\mathbf{x})$. It is therefore straightforward to see that in this case,

$$\omega \stackrel{\text{def}}{=} P\left(\overline{f}(\mathbf{x}) = f(\mathbf{x}) \,|\, \mathbf{x} \in \mathbf{E}_S^{f*}\right) = P\left(\hat{f}(\mathbf{x}) = f(\mathbf{x}) \,|\, \mathbf{x} \in \mathbf{E}_S^{f*}\right) = \text{Acc}(\text{NAN}_S, \mathbf{E}_S^{f*}).$$

[6] `https://archive.ics.uci.edu/ml/datasets/MONK's+Problems`

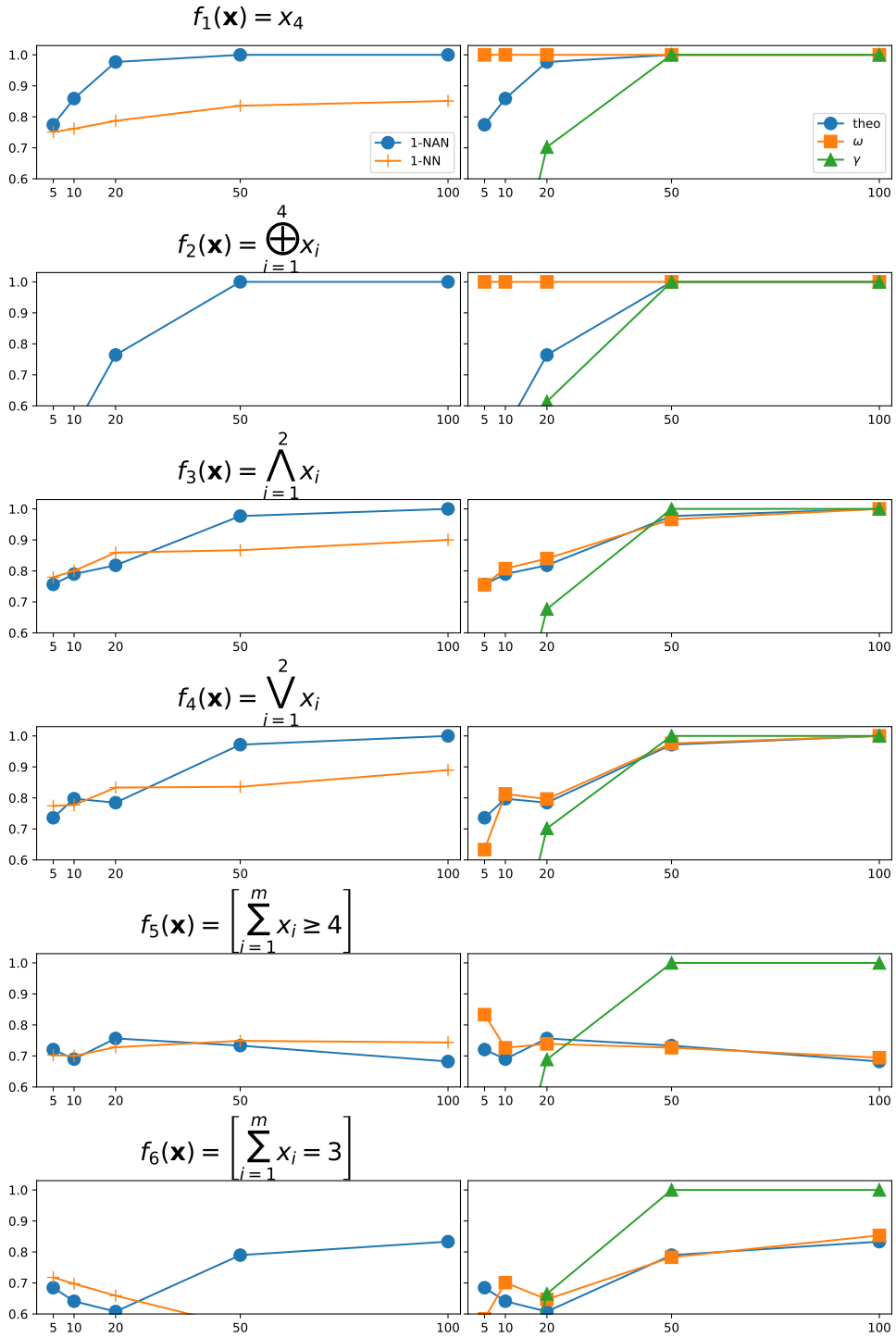

Figure 3.6: Accuracies (left) of the 1-NAN and 1-NN algorithms over different Boolean functions ($m = 8$) and training set sizes, with corresponding values of ω, γ, and theoretical accuracy (right). The x axis corresponds to the size of the training set.

	1-NAN	APC	1-NN	ω_S^f	γ_S^f
Monk 1	.961	.98	.787	.961	1
Monk 2	.998	1	.738	.998	1
Monk 3	.963	.96	.829	.963	1

Table 3.3: Accuracies of the 1-NAN, APC and 1-NN algorithms over the Monk datasets.

When $\gamma = 1$, the only elements $\mathbf{x}$ we want to classify belong to $\mathbf{E}_S^{f*}$ (otherwise they would be in S), so this last term exactly corresponds to the accuracy of the classifier. Another way to see it is to observe that the first term of the expression of Proposition (3.2) $\text{Acc}(\text{NN}_S, A) \cdot \alpha$ is null because $\alpha = 0$. Only the second term $\text{Acc}(\text{NN}_{\mathbf{E}_S^{f*}}, B) \cdot (1 - \alpha)$ is of importance, and its value corresponds to ω. As expected, this observation allows us to state that estimating the value of ω is paramount to have a precise idea of the accuracy of an analogical classifier.

A striking result of the experiments is that for the two functions $f_1(\mathbf{x}) = x_4$ and $f_2(\mathbf{x}) = x_1 \oplus \cdots \oplus x_4$, the analogical labels are always correctly predicted: $\omega = 1$, i.e. there is no class noise. However, the accuracy of 1-NAN is not necessarily equal to 1 especially for f_2 with small training set sizes. The reason is simple: even if all the elements in $\mathbf{E}_S^f$ are correctly classified, $\mathbf{E}_S^f$ does not cover the whole space ($\gamma < 1$) so the accuracy of 1-NAN still depends on that of 1-NN. While the accuracy of 1-NN over f_1 is not too bad, the one over f_2 is absolutely terrible and actually below 0.3 (a random prediction would have done better!). This is one of the distinguishing features of the XOR functions: the nearest neighbor of an element is almost always in a different class. So, sadly, the 1-NN algorithm can do nothing but wrong on these kind of functions. In Chapter 6, we will prove a strong result showing that both the projections (like f_1) and the XOR functions (like f_2) **always** lead to perfectly sound extensions, for any training set of any size.

Table 3.3 shows the same metrics for the Monk datasets and also reports the results of the Analogical Proportion Classifier (APC) from [MBD08], which corresponds to algorithm 2 with $k = 100$. We note that the functional NAN approach (almost) achieves the same results as the somewhat more complex algorithm described in Section 3.1.2, and that here again the analogical extension set covers the whole universe: this means that a conservative approach would have been sufficient! Actually, this raises the following question: why would we want to look for more than one analogical neighbor when every element of the universe is already in $\mathbf{E}_S^f$, and therefore *analogically linked* to those in S? Our experiments tend to show that this becomes superfluous, provided that the training set is big enough.

Conclusion

In this chapter, we have provided a functional and unifying definition of analogical learners, which turned out to be closely related to the k-NN classifiers. Starting from this definition,

we were in a position to prove an analytical convergence result, similar to that of the nearest neighbor algorithm. Obviously, this is not enough to conclude regarding the predictive ability of analogy-based classifiers. We have also shown that their VC-dimension is infinite. It should not come as a surprise, as a very particular case of analogical rule (when the analogical proportion is trivial) is the k-NN rule.

This fact obviously has a consequence on the way we can work to improve analogy-based classifiers. Indeed, learning by analogy is prone to over-fitting and a relevant learning strategy should take this fact into account. Nevertheless, we have to keep in mind that the VC-dimension alone is not a perfect marker of the quality of a machine learning algorithm, especially when it comes to practical implementation.

In terms of accuracy in a Boolean setting, we have found a strong link between the accuracy of the NAN algorithm and that of the NN algorithm. Our functional definition tells us that we can consider the NAN algorithm as a NN strategy on an extended and noisy training set: the analogical extension $\mathbf{E}_S^f$. This leads us to consider analogical classification as a two-steps process:

- First, the training set is extended to the analogical extension.
- Then, a k-NN algorithm is used on this extended training set.

Obviously, the accuracy of the NAN algorithm depends on the quality of the analogical extension. In Chapter 6, we will entirely describe the conditions needed for this extension to be perfectly sound.

We have to remember that analogical reasoning brings its whole power in the case where few data are available. If a lot of data are available, it is very likely that we have elements similar to the one at hand and, in that case, a k-NN style reasoning is natural. In the opposite case, when we only have a few relevant cases at hand, applying analogical proportion-based predictions appears to be a meaningful option.

This concludes for now our investigations on analogical classifiers. We have seen that they have so far mostly been used for natural language processing tasks, or classification in Boolean settings. One of the main goals of this thesis was to apply analogical proportion-based reasoning to concrete, real-world applications. We have chosen to investigate the suitability of analogical proportions in the design of recommender systems. This will be the object of the next two chapters.

Résumé du chapitre Ce chapitre s'intéresse au problème de la classification analogique, et décrit nos contributions à ce sujet.

Une première forme de classifieur analogique a été proposée dans les travaux de Stroppa et Yvon [SY05a] (classifieurs conservatifs). Une autre forme de classifieur analogique est due aux travaux de Bayoudh, Delhay et Miclet [MBD08, BMD07] (classifieurs étendus). En pratique, ces deux types de classifieurs ont des implémentations relativement distinctes (même s'ils reposent

sur les mêmes fondements théoriques).

Notre première contribution est de proposer une définition fonctionnelle des classifieurs analogiques, qui a la particularité d'unifier ces deux approches préexistantes. Cette définition nous permet de dériver diverses propriétés théoriques des classifieurs analogiques telles que leur VC-dimension (infinie), et leur taux d'erreur qui s'exprime en fonction de celui d'un classifieur de type plus-proches-voisins (k-NN). En effet, un des autres apports majeurs de cette définition fonctionnelle, est de révéler les liens étroits qui existent entre les classifieurs analogiques et les classifieurs k-NN. Aussi, on peut considérer le processus de classification comme la succession des deux étapes suivantes :

- D'abord, l'ensemble d'apprentissage est étendu pour devenir ce que l'on appelle *l'extension analogique*.
- Ensuite, un classifieur k-NN est utilisé sur cette extension analogique, qui fait office d'ensemble d'apprentissage étendu (et potentiellement bruité).

Naturellement, le taux d'erreur du classifieur analogique dépend de la qualité de l'extension. Dans le chapitre 6, nous décrirons dans quelles conditions cette extension est non-bruitée.

Jusqu'à présent, l'utilisation des proportions analogiques sur des problèmes d'apprentissage était principalement cantonnée à des applications artificielles (domaine booléen par exemple). Un des deux enjeux de cette thèse était d'évaluer la pertinence de l'usage des proportions analogiques sur des problèmes plus concrets. Cet axe de recherche est l'objet des deux chapitres suivants, où nous nous intéresserons au problème de la recommandation.

Chapter 4

Background on recommender systems

Content

In a world of information overload, automatic filtering tools are essential to extract relevant information from basic noise [RRSK11, AT05]. In the field of e-commerce, recommender systems play the role of search engines when surfing the entire web: they filter available items to provide relevant suggestions to customers. They help to answer questions as diverse as "which movie to rent?", "which car to buy?" or "which restaurant to try?". They do so by

providing the user with a list of recommendations.

We have seen in the previous chapter that analogical learners exhibit promising results when compared to the k-NN algorithms in a Boolean setting. In chapter 5, we will try to apply analogical learning to a recommendation task to see if analogical learners can prove useful in these more challenging contexts.

In this chapter, we will provide some necessary background on recommender systems, before diving into Chapter 5 where we will describe our contributions and experiments. This chapter is structured as follows. We first provide a taxonomy of current research in the recommender systems area in Section 4.1, and set ourselves within the collaborative filtering framework. In Section 4.2, we formally define the specific recommendation problem that we will address in this document, and we explain how to assess the quality of a recommender system in this context. In Section 4.3, we extensively describe two collaborative filtering methods: neighborhood-based algorithms, and matrix factorization-based techniques. These two families of algorithms will serve as baselines with which we will compare our own prediction algorithms in Chapter 5.

4.1 Taxonomy of recommender systems

Undoubtedly, there are many different tasks that range among the general problem of recommendation. If they all share the ultimate goal of providing users with relevant recommendations, they greatly differ in nature by the items that are recommended, and by the technical means that are involved.

To provide a clear view of the vast recommendation landscape, various taxonomies for recommender systems have been proposed: see for example [Bur02, AT05, Bur07], and more recently [BR11]. Using that of [BR11], we will successively describe **content-based**, **collaborative** and **knowledge-based** approaches in the next three sections, and show that they generally apply to different problems.

Rest assured that in real systems, the frontiers between these three families are not that sharp, and practical recommendation solutions usually fall into more than one single category. We refer to such systems as **hybrid** recommenders.

4.1.1 Content-based techniques

The general idea of content-based algorithms is as follows: they try to recommend to a user some items that are **similar** to those that the user has liked in the past. For example, if the renting history of a user mainly contains science fiction movies, the aim of the system is to learn these preferences and to recommend some other science fiction movies to the user. We recommend [LdGS11] for a recent overview of content-based methods.

To find out similar items to those that a user has liked, these systems usually rely on a similarity metric whose nature strongly depends on the representation of the items. In large-

scale online shopping systems, were items are extremely diverse and abundant, a similarity measure between two items would for example be the number of web sessions on which the two item pages have jointly been visited (yes, this is the reason why some online systems will try to sell you a fridge even though you just bought a brand new one). In systems were items are more homogeneous, i.e. in a movie store system, more sophisticated approaches can be built relying on some metadata about the items (hence the name *content*-based). In the case of movies, the metadata could be for example the genre, main actors, film director, etc.

As content-based techniques usually do not rely on a set of ratings to compute item similarities, a nice resulting feature is that new items that have not yet been rated by anybody can still be recommended provided that the items are properly described (which is not necessarily easy, as we will see). This is not the case for a new user though: as we have no information about the items this user may like, a content-based recommender would not be able to output any prediction. These techniques also stand out by their explanatory potential: the motto "*here are some items that are akin to those you liked*" is perfectly understandable and seems sound. Recommender systems usually strive for explanatory power, because it is recognized that when confronted with a given recommendation, users are more likely to accept the recommendation if they understand and acknowledge the underlying process.

However, content-based systems are prone to various behaviours that tend to make them less competitive than collaborative filtering methods. The first obvious drawback, known as the problem of *limited content analysis*, is precisely the need to describe items with metadata (remember our movie example with genre, actors, etc.). Such descriptions can be extremely costly to acquire, and can only capture a very limited subset of item features, which are not necessarily the most important ones when it comes to the users personal tastes. Going back to our science fiction fan, it could be plausible that the user has a strong preference for the steampunk sub-genre, and yet is perfectly indifferent to the superhero fiction movies, and both can still be considered as sub-genres of science fiction. A system that could not distinguish these two kinds of movies would fail to provide our steampunk fan with relevant recommendations.

Another well-known drawback of content-based recommenders is their tendency to overfit and recommend only items that users may already know or do not need anymore (such as a fridge!). The recommendations tend to lack in novelty, surprise and diversity. Also, content-based systems tend to output much less accurate predictions than the collaborative filtering methods (the way accuracy is computed will be described in Section 4.2.2). This is why collaborative filtering methods have gained in popularity.

4.1.2 Collaborative filtering

The main idea behind collaborative filtering algorithms is to recommend items to a user that other users with similar tastes have liked in the past. In its most general form, collaborative filtering methods (also called social filtering) try to model the social environment of users and

to output recommendations based on the behaviour of their peers.

Probably the most common way of modeling the social environment of a user is to use historical transactions stored in the system, for example using user-item ratings: two users are peers if their ratings are similar. If Bob and Alice usually agree on their movie ratings, i.e. they like the same movies and dislike the same movies, and if Alice has not seen a movie that Bob has liked, then it will be recommended to Alice.

A main advantage of collaborative filtering methods over content-based ones is that they do not require any item or user knowledge: only user-item feedback is needed to output a prediction. Thanks to this, they also allow to build systems that can deal with items of different natures. It is indeed plausible to imagine a collaborative filtering system recommending both books and movies, on the basis that users that love the same books usually like the same movies. As books and movies cannot be described using the same set of features (even though they are close), the use of content-based techniques would make the task much more difficult, or would again require largely hand-crafted work.

The main advantage of collaborative filtering methods is also their main weakness. Before they can become effective, these systems require a serious amount of data (i.e. user-item interactions) to train a model or to use heuristics efficiently. A side-effect is known as the cold-start problem: there is no way to yield recommendations for new users that have no known ratings, and the same goes for new items that nobody has rated.

All the recommendation algorithms that we will propose in this document are of a collaborative nature. Section 4.3 will be devoted to an overview of two main collaborative techniques: the neighborhood approach and matrix factorization-based methods. They are typical examples of the two main approaches to collaborative filtering: heuristic-based systems, and model-based systems.

4.1.3 Knowledge-based systems

The two families we have described so far (content-based and collaborative methods) are only suitable for certain kinds of items. While movies, books, music or news can all fit quite well within these two frameworks, this is not the case for some other items such as cars, bank loans, or real estates. Take for example the problem of car recommendation. A content-based approach would need to know what kind of cars the user has owned in the past, and would recommend some new cars that are similar to those past vehicles. This method is obviously bound to fail, not only because in practice such information about past vehicles is never available, but also because the recommendations would be based on some years-old preferences. As for collaborative methods, they would require the system to have a massive amount of past transactions at its disposal, and people do not buy as many cars as they watch movies: only very few information would be available, leading to disastrous results.

In broad terms, we refer to knowledge-based systems as any method that is neither content-

based nor collaborative. In general, these systems are used in settings where the domain and contextual knowledge (i.e. the knowledge about the items that are available in the system) prevails over any other source of information, such as social information for example.

A practical instance of knowledge-based recommendation takes the form of a recommendation session, where users indicate their needs in an interactive manner, and the systems tries to match these needs as best as possible to provide useful recommendations. According to [FB08], there exist two kinds of knowledge-based systems, that differ in the way recommendations are computed. The first category is made up of the **case-based** systems. Case-based recommenders obviously draw from research in case-base reasoning described at the end of the first chapter, and computations are based on similarities: the system will try to recommend items that are similar to an ideal item expressed by the user's needs. In this respect, case-based recommenders are close to the aforementioned content-based systems. The second kind of knowledge-based recommenders are the so-called **constraint-based** systems (see [FFJZ11] for a complete review). In such systems, users and items are described by a set of properties, along with a set of constraints. In the car recommendation example, an item constraint could for example express that a convertible car should not possess a sunroof, and user constraint could state that the car color should not be green. The problem of recommending an item to a user then takes the form of a classical constraint satisfaction problem.

We are well aware that these two definitions of case-based and constraint-based systems are quite general and vague, but the fact remains that practical implementation of such systems are so domain-dependent that it is difficult to exhibit a clear unifying framework that would still be accurate.

Now that we have a clear overview of the various tasks that range among the wide topic of recommendation, we will detail the recommendation problem that will be addressed in the next chapter.

4.2 Recommendation by rating prediction

In this section, we formally define the recommendation problem that we plan to address later in Chapter 5. We also describe how to assess the quality of a recommender system.

4.2.1 Problem formalization and notation

Let U be a set of users and I a set of items. For some pairs $(u, i) \in U \times I$, a rating r_{ui} is supposed to have been given by user u to express their preferences towards the item i. The way the preferences are expressed depends on the rating scale that is used, which can be of different nature. According to [SFHS07], we can distinguish four different kinds of rating scales:

- It is quite common that r_{ui} belongs to a **numerical** rating scale. A typical numerical scale is $[1, 5]$, where 5 means a strong preference for item i, 1 means a strong rejection,

and 3 means indifference, or just an average.

- A sub-case of the previous scale is the **binary** rating scale, where ratings belong to $[0, 1]$. In this case, we associate the value 0 with the meaning *dislike* and the value 1 with the meaning *like*.
- An extreme case is the **unary** rating scale, where a value of 1 translates as a preference, but rejections (or dislikes) are not addressed.
- Sometimes, the rating scale can be considered as **ordinal**, where the elements of the scale have no absolute meaning and are only meaningful when compared to others, e.g. [*rejection, indifference, agreement*].

In practice, these semantic distinctions are purely contextual: the $[1, 5]$ rating scale could perfectly be interpreted as purely numerical, or conversely as purely ordinal. This observation has been addressed for example in [KS11], and will also be discussed here in the next chapter. However, the rating scale is a decisive component of a recommender system, and algorithms are usually designed to work with only one single rating scale.

Ratings may be collected in different ways, and we commonly distinguish explicit and implicit ratings. **Explicit** ratings are those for which users have explicitly expressed their preference towards the items. **Implicit** ratings belong to a binary or unary scale, and correspond to the result of collected data about users behaviour, without any active involvement. For example, if a user has listened to a given track more than 10 times, we might consider this as an implicit rating of 1. The ability to deal with both explicit and implicit ratings can lead to significant improvements for a recommender system. In [KS11], a model using only explicit numerical ratings was extended to deal with implicit ratings, which notably improved the RMSE of the system (defined later). When no implicit ratings are available, the same authors showed that the simple modeling of implicit ratings as $r'_{ui} = 1 \iff r_{ui} \in R$ led to significant improvements. Here, r'_{ui} is an implicit rating stating that u has rated i, regardless of the rating value.

In our work, we will mostly focus on the explicit and numerical rating scheme. Also, the algorithms we will design will be of a collaborative nature, and therefore they will rely on a dataset containing user-item interactions (in our case, user-item ratings). Let us denote by R the set of known ratings recorded in the system. In real systems, the size of R is very small with regard to the potential number of ratings which is $|U| \times |I|$, as a lot of ratings are missing. The set U_i will denote the set of users that have rated item i, and $U_{ij} \stackrel{\text{def}}{=} U_i \cap U_j$ is the set of users that have rated both items i and j. Similarly, I_u is the set of items that user u has rated, and $I_{uv} \stackrel{\text{def}}{=} I_u \cap I_v$ is the set of items that both users u and v have rated.

A very common way of providing personalized recommendations to a target user is to estimate their tastes regarding the items that the system provides. The taste of a user u for a given item i is usually represented as the rating that u would give to i. Once these estimations are made, a simple option is to recommend the items with the highest ratings among all the estimated scores for the considered user (using the implicit assumption that a user should be

interested in the items with high scores).

More formally, a recommender system usually proceeds as follows:

1. Using a prediction algorithm A, estimate the unknown ratings r_{ui} (i.e. $r_{ui} \notin R$). This estimation $A(u, i)$ will here be denoted $\hat{r}_{ui}$.
2. Using a recommendation strategy S and in the light of the previously estimated ratings, recommend items to users. For instance, a basic yet common strategy is to suggest to user u the items $i \notin I_u$ with the highest estimation $\hat{r}_{ui}$.

When using a unary or binary rating scale, these two stages are usually indistinguishable: it is natural to recommend an item to a user if the estimated predicted rating is 1. In practice, the recommendation strategy depends a lot on the pragmatic constraints of the system: a store manager may want to push forward a given item to boost its sales, regardless of the rating predictions outputted by the algorithm. As a result, most of the research has focused on the prediction algorithms, and so will we. Note also that this view is only meaningful for non-ordinal rating scale. When the output of prediction algorithm A are ranks, the recommendation strategy S is obvious and the distinction between A and S is superfluous.

Prediction algorithm implementations are strongly influenced by the fields of data mining and of course machine learning. However, we want to emphasize that the general setting of recommender systems is actually quite different from that of classification or regression, the two main prediction tasks of machine learning. A naive reasoning could lead us to consider that the rating prediction problem with the rating scale $[1, 5]$ is nothing but a classification task with five classes $1, 2, 3, 4, 5$. But this is forgetting that the values $1, 2, 3, 4, 5$ are actually ordered, and that the difference between *class* 1 and *class* 2 is not the same as the difference between *class* 1 and *class* 4. But most importantly, in classification or regression all the instances belong to the same space (that space was X^m in our previous chapters), and have the same features. This is not the case here! In a recommendation problem, if we chose our instances to be the users and their feature space to be the items, we would end up with instances that are only **very partially described**, because of course no user has rated the whole set of items. All the more, as the number of items is usually very high, we would end up with a very high dimensional problem, where traditional learning algorithms tend to fail due to the so-called curse of dimensionality. Our point here is that recommender systems problems do not really fit within the traditional machine learning setting, and represent a new setting in their own right. It can be insightful to consider our prediction problem as that of matrix completion: the dataset R can be viewed as a sparse matrix $R \in \mathcal{M}^{|I|\times|U|}$ where columns are users and rows are items. Our prediction task is to fill-in the missing entries of this matrix.

We now describe how the quality of a prediction algorithm and of a recommendation strategy can be evaluated.

4.2.2 Recommender system evaluation

Providing an accurate measure of the overall quality of a recommender system is not a simple task, and diverse viewpoints have to be considered. We can distinguish three main evaluation settings. The first one is to perform **user studies**, where actual users are asked to interact with a given system and to provide some feedback. This evaluation process is probably the one that allows to best capture users need, because feedback is explicitly given. Naturally, this is also the most expensive kind of experiment to conduct, because it requires a lot of user time. The second evaluation setting is to perform **on-line studies**. In its most simple form, this is equivalent to A/B testing: to see which of the two versions of our recommender performs better, we provide some users with the first version (A) and some other users with the second version (B). We can then assess the change in performance between the two versions and decide which one is the most suitable for our needs.

Finally, the last available option to evaluate recommender systems is to perform **off-line evaluation**: in this setting, we dispose of a dataset of past transactions (the set of all ratings R), and try to simulate user behaviour to compute different measures. In general, we will use k-folds cross-validation procedures to reliably evaluate each of the performance measures. The set of all ratings R is divided into k (typically 5) disjoint sets of equal sizes; at each of the k iterations, the test set R_{test} is set to the k^{th} subset and the training set R_{train} is set as the union of the $k-1$ remaining subsets: the system will be trained on R_{train} and tested on R_{test}. The reported performances are then averaged over the k folds.

The first two evaluation settings (users studies and online studies) require the use of an actual working recommender system. Thus, we will here only focus on measures that can be calculated in an off-line evaluation process.

Accuracy

The performance of the algorithm A is usually evaluated in terms of accuracy, which measures how close the rating predictions $\hat{r}_{ui}$ are to the true ratings r_{ui}, for every possible prediction. The Root Mean Squared Error (RMSE) is probably the most common indicator of how accurate an algorithm is, and is calculated as follows:

$$\text{RMSE}(A) = \sqrt{\frac{1}{|R_{\text{test}}|} \cdot \sum_{r_{ui} \in R_{\text{test}}} (\hat{r}_{ui} - r_{ui})^2}.$$

Another common indicator for accuracy is the Mean Absolute Error (MAE), where important errors are not penalized more than small ones:

$$\text{MAE}(A) = \frac{1}{|R_{\text{test}}|} \cdot \sum_{r_{ui} \in R_{\text{test}}} |\hat{r}_{ui} - r_{ui}|.$$

As we said, RMSE probably is by far the most popular measure for evaluating the per-

formance of a recommender system. The simple fact that \$1 million was awarded for a 10% improvement of RMSE during the Netflix competition illustrates the supremacy of RMSE. Yet this measure is also criticized, if only because it is not very easy to interpret it in a meaningful way: if you end up with an RMSE of 0.5, how do you know if it is good or not? You would need to compare multiple algorithms before knowing what can be thought of as a good RMSE, which by the way highly depends on the rating scale.

Another issue with RMSE is that they do not properly reflect the way users interact with a recommendation system [JRTZ16]. Ultimately, the most important feature of a recommender system probably is how well it can **order** the different items for a given user, with regards to their preferences. Indeed, it would be harmful to rank a disliked item higher than an item for which the user has a strong preference, and conversely predicting a rating $\hat{r}_{ui}$ of 2.5 while the true rating is 1 (i.e. with a potentially high error) is not as serious, because the corresponding item would not have been recommended anyway. The point here is that providing accurate predictions is crucial, but only on a particular subset of predictions.

A related concern is that when using a suitable loss function, some algorithms relying on an optimization process can have a great RMSE score while still having no recommendation power. As we will see, in Section 5.2.3, a simple baseline predictor can outperform the RMSE of many other approaches, and yet this algorithm has absolutely no recommendation power: the recommendations that it outputs are the same for all the users.

To better reflect the user-system interaction, other precision-oriented metrics are sometimes used in order to provide a more informed view.

Precision and recall

Precision and recall help measure the ability of a system to provide relevant recommendations, and are therefore indicators of the performance of the recommendation strategy S. They are defined by means of the number of true positives, true negatives, false positives and false negatives. As such, they are mostly used in an implicit rating scheme (or at least with binary rating scales), but they can also be used when the rating scale is gradual. In this latter case, the recommendation strategy has to be clearly defined.

In the following, we denote by I^S the set of items that the strategy S will suggest to the users using the predictions coming from A. For ratings in the interval $[1, 5]$, a simple strategy could be for example to recommend an item i to user u if the estimation rating $\hat{r}_{ui}$ is greater than 4:

$$I^S \stackrel{\text{def}}{=} \{\, i \in I \mid \exists u \in U,\ \hat{r}_{ui} \geq 4,\ r_{ui} \in R_{\text{test}} \,\}\,.$$

We also define I^* as the set of items that are **actually** relevant to the users, i.e. the set of items that would have been recommended to the users if all the predictions made by A were exact:

$$I^* \stackrel{\text{def}}{=} \{\, i \in I \mid \exists u \in U,\ r_{ui} \geq 4,\ r_{ui} \in R_{\text{test}} \,\}\,.$$

The **precision** of the system is defined as the fraction of recommended items that are relevant to the users, and the **recall** is defined as the fraction of relevant recommended items over all relevant items:

$$\text{Precision} \stackrel{\text{def}}{=} \frac{\mid I^S \cap I^* \mid}{\mid I^S \mid},$$
$$\text{Recall} \stackrel{\text{def}}{=} \frac{\mid I^S \cap I^* \mid}{\mid I^* \mid}.$$

Precision and recall are complementary metrics: it would not make sense to evaluate an algorithm performance by only looking at the precision, without considering the recall. It is indeed quite easy to obtain a high recall (by simply recommending all of the items), but that would lead to a terrible precision. Precision and recall are commonly summarized into the F-measure, which is their harmonic mean:

$$\text{F} \stackrel{\text{def}}{=} 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}.$$

If accurate predictions are crucial, it is widely agreed that it is insufficient for deploying an effective recommendation engine. Indeed, other dimensions are worth estimating in order to get a complete picture of the performance of a system [MRK06, HKTR04, KB14]. For instance, one may naturally expect from a recommender system not only to be accurate, but also to be surprising, and to be able to recommend a large number of items.

Coverage

In its simplest form, coverage is used to measure the ability of a system to recommend a large amount of items: it is quite easy indeed to create a recommender system that would only recommend very popular items. Such a recommender system would drop to zero added value. We here define the **coverage** as the proportion of recommended items out of all existing items (I):

$$\text{Coverage} \stackrel{\text{def}}{=} \frac{\mid I^S \mid}{\mid I \mid}.$$

Surprise

Users expect a recommender system to be surprising: recommending an extremely popular item is not really helpful. Be warned though: surprise is one of the most difficult notions to grasp, especially when it comes to assess it with numerical measures. Following the works of [KB14], we will here define the **surprise** associated with a recommendation with the help of the pointwise mutual information (PMI). The PMI between two items i and j is defined as follows:

$$\text{PMI}(i,j) \stackrel{\text{def}}{=} -\log_2 \frac{P(i,j)}{P(i)P(j)} / \log_2 P(i,j),$$

where $P(i)$ and $P(j)$ represent the probabilities for the items to be rated by any user, and $P(i,j)$ is the probability for i and j to be rated together. They are estimated by $P(i) \approx \frac{|U_i|}{|U|}$ and $P(i,j) \approx \frac{|U_i \cap U_j|}{|U|}$. PMI values fluctuate between the interval $[-1,1]$, -1 meaning that i and j are never rated together and 1 meaning that they are always rated together. To estimate the surprise of recommending an item i to a user u, the authors propose two definitions:

- either to take the maximum of the PMI values for i and all other items rated by u, with $\text{Surp}^{\max}(u,i) \stackrel{\text{def}}{=} \max_{j \in I_u} \text{PMI}(i,j)$,
- or to take the mean of these PMI values with $\text{Surp}^{\text{avg}}(u,i) \stackrel{\text{def}}{=} \frac{\sum_{j \in I_u} \text{PMI}(i,j)}{|I_u|}$.

The overall capacity of a recommender to surprise its users can be defined as the mean of all the surprise values for each prediction. Because they are defined using the pointwise mutual information, the **lower** the values of $\text{Surp}^{\max}$ and Surp^{avg}, the most *surprising* the recommendations.

Other dimensions

There are many other measures that can assess the performances of a recommender system (see [SG11] for an extensive survey). We can cite for example **trust** which characterizes the confidence that a user would put into a recommendation, or also **diversity** that evaluates how the recommendations are distinct from each other. Unfortunately these dimensions are actually a lot harder to assess in an off-line setting, and there are no popular quantitative measure that can evaluate them in a satisfactory way.

Now that we know how to evaluate the performances of a recommender system, we will detail two very popular collaborative filtering algorithms that usually lead to good results.

4.3 Two collaborative filtering techniques: neighborhood-based, and matrix factorization

We will here present two families of collaborative filtering algorithms: the neighborhood approach based on the well known k-NN algorithm, and the matrix factorization techniques whose groundings come from linear algebra and that lead to elegant and accurate models. These two families of algorithms will serve as baselines against which we will compare the performances of our own algorithms in the next chapter, so it is important that we understand their functioning in detail.

4.3.1 The neighborhood approach

4.3.1.1 Prediction framework

The motto of collaborative filtering methods is to recommend some items that are appreciated by other users having the same tastes. This principle is carried out to the letter in neighborhood

methods. Neighborhood approaches are instances of the general k-NN scheme. To estimate the rating r_{ui} of a user u for an item i, the most basic method consists in computing the set of k users that are most similar to u and that have rated i. We will denote this set $N_i^k(u)$. The computation of $N_i^k(u)$ depends of course on a similarity measure between users, which is based on their respective ratings. The estimation $\hat{r}_{ui}$ of r_{ui} is then computed as an aggregate of the ratings r_{vi}, where v is one of the neighbors of u in $N_i^k(u)$. Usually, the aggregation is simply a mean weighted by the similarity between u and v:

Definition 4.1 (Neighborhood approach)**.** The estimation of a rating r_{ui} using the **neighborhood approach** (also denoted k-NN here) is:

$$\hat{r}_{ui} = \frac{\sum\limits_{v \in N_i^k(u)} r_{vi} \cdot \text{sim}(u, v)}{\sum\limits_{v \in N_i^k(u)} \text{sim}(u, v)},$$

where $N_i^k(u)$ is the set of users having rated i and having the highest k values of sim with u:

$$N_i^k(u) \stackrel{\text{def}}{=} \left\{ v \in U \;\middle|\; r_{vi} \in R,\ v \in \arg\max_{u' \in U}^{k} \left[\text{sim}(u, u')\right] \right\}$$

The estimation process of the neighborhood approach is described in Algorithm 3. Obviously, the training stage only needs to be done once. Then, the similarities can be reused for future predictions.

Algorithm 3 The neighborhood recommender.

Input: A set of ratings R, and a pair (u, i) for which r_{ui} is unknown.
Output: $\hat{r}_{ui}$, an estimation of r_{ui}.
Training stage:
for all $(u, v) \in U^2$ **do**
  Compute $\text{sim}(u, v)$.
end for
Prediction stage:
$\text{denum} \leftarrow 0,\ \text{num} \leftarrow 0$
$U_i^s \leftarrow \text{Sorted}(U_i)$ *// Sort by value of sim with u, in decreasing order.*
for all $v \in U_i^s[:k]$ **do**
  I.e. for the k first users in U_i^s
  $\text{denum} \leftarrow \text{denum} + \text{sim}(u, v)$
  $\text{num} \leftarrow \text{num} + r_{vi} \cdot \text{sim}(u, v)$
end for
$\hat{r}_{ui} \leftarrow \frac{\text{num}}{\text{denum}}$

We need here to make an important point: instead of recommending an item to a user, we could perfectly recommend a user to an item, which is a symmetric and equivalent point of

view. The prediction $\hat{r}_{ui}$ would be the exact counterpart of the one we described above, but instead of considering similarities between users, we would compute similarities between items:

$$\hat{r}_{ui} = \frac{\sum\limits_{j \in N_u^k(i)} r_{uj} \cdot \text{sim}(i,j)}{\sum\limits_{j \in N_u^k(i)} \text{sim}(i,j)}.$$

The set $N_u^k(i)$ is defined as the counterpart of the set $N_i^k(u)$. We can interpret this prediction as an edge-case of content-based recommendation: the items we are recommending to u are close to some other items that u has already rated in the past, which is exactly what a content-based recommender does. In this case though, we did not use any metadata about the items, and items where represented as vectors of ratings.

The choice between a user-based or an item-based recommendation mainly depends on the system at hand. When the number of users is high with regard to the number of items, computing similarities between items can help saving memory and computation time, but it is also important to assess the potential number of neighbors for any given u or i: it is always more desirable to compute similarities over a high number of ratings. Also, item-based methods are more easily justified: when presenting a recommendation to a user u, it is easier to explain how two items relate to each other rather than involving other users that u does not even know.

4.3.1.2 The similarity measures

There are many ways to define the similarity metric between two users (or items). One of the most common measure is the Cosine similarity.

Definition 4.2 (Cosine similarity)**.** The **Cosine similarity** between two users u and v is defined as:

$$\text{Cosine sim}(u,v) \stackrel{\text{def}}{=} \frac{\sum\limits_{i \in I_{uv}} r_{ui} \cdot r_{vi}}{\sqrt{\sum\limits_{i \in I_{uv}} r_{ui}^2} \cdot \sqrt{\sum\limits_{i \in I_{uv}} r_{vi}^2}}.$$

Here, users u and v are considered as vectors in a vector space defined by the items they have both rated (the set of common items is I_{uv}). Their Cosine similarity simply is the cosine of the angle between the two vectors. An unnatural feature of this metric is that two vectors with (potentially different) constant values will always have a similarity of 1, because they are collinear. For example if $u = (2,2)$ and $v = (5,5)$, Cosine $\text{sim}(u,v) = \frac{20}{\sqrt{8}\sqrt{50}} = 1$, while one would expect u and v to be quite different with regards to their tastes.

Such a flaw can be overcome using the Pearson similarity, which can be viewed as a mean-centered version of the Cosine similarity.

Definition 4.3 (Pearson similarity)**.** The **Pearson similarity** between two users u and v is defined as:

$$\text{Pearson sim}(u,v) \stackrel{\text{def}}{=} \frac{\sum\limits_{i \in I_{uv}} (r_{ui} - \mu_u) \cdot (r_{vi} - \mu_v)}{\sqrt{\sum\limits_{i \in I_{uv}} (r_{ui} - \mu_u)^2} \cdot \sqrt{\sum\limits_{i \in I_{uv}} (r_{vi} - \mu_v)^2}},$$

where μ_u and μ_v are the average rating of users u and v respectively.

One last similarity metric that we will use is the Mean Squared Difference (MSD)[1]:

Definition 4.4 (Mean squared difference)**.** The **Mean Squared Difference** between two users u and v is defined as:

$$\text{MSD}(u,v) \stackrel{\text{def}}{=} \frac{1}{|I_{uv}|} \cdot \sum_{i \in I_{uv}} (r_{ui} - r_{vi})^2.$$

Notice that none of these metrics take into account the support between the two users u and v, i.e. the number of items that they have commonly rated. It would be foolish to have the same faith in a similarity computed over hundreds of common items as in a similarity computed only over a few items. This is why in practice, it is common to give a higher weight to the similarities that have a high support. Another option is to *shrink* the similarity measure of two users toward zero if their support is low:

$$\text{shrunk\_sim}(u,v) \stackrel{\text{def}}{=} \frac{\mid I_{uv} \mid -1}{\mid I_{uv} \mid -1 + \lambda} \cdot \text{sim}(u,v),$$

where λ is a sort of regularization constant. Such shrinkage technique can be motivated by a Bayesian perspective, and usually leads to significant improvements in the system performances. It only makes sense however for similarity measures that are centered around zero, such as the Pearson similarity. We will not consider shrinkage in any of our experiments.

4.3.1.3 Similarity computation

We will now describe how to practically compute these similarity metrics. We see that all of the presented metrics rely on the set of common items I_{uv}. In practice, the explicit computation of this set for every single pair of users can be very expensive, and naive algorithms lead to poor performances. Consider for example the naive similarity computation of Algorithm 4. In the worst case, we can consider that $\mid I_u \mid = \mid I_v \mid = \mid I \mid$ for all u and v, so the complexity of this naive algorithm is $\mathcal{O}(\mid U \mid^2 \mid I \mid^2)$.

However, the complexity can be taken down by a great deal if we proceed in a MapReduce fashion [DG04], as described in Algorithm 5 which exemplifies how to compute the Cosine similarity. Actually, any other similarity measure that relies on the set I_{uv} can be computed

[1] Strictly speaking MSD is actually a distance rather than a similarity metric, so we could take its inverse.

Algorithm 4 A general naive algorithm for similarity computation

Input: A set of ratings R.
Output: The similarity between all pairs of users.
for all $u \in U$ **do**
  for all $v \in U$ **do**
    for all $i \in I_u$ **do**
      if $i \in I_v$ **then**
        We know now that $i \in I_{uv}$*, so we can use* r_{ui} *and* r_{vi} *as we please, depending on the measure that is computed.*
      end if
    end for
  end for
end for

Algorithm 5 Computation of the Cosine similarity.

Input: A set of ratings R.
Output: Cosine sim(u, v) for all pairs of users.
Initialization (n is defined as the number of users, i.e. $n =\mid U \mid$*):*
sim = null_array[n][n]
sum_prod = null_array[n][n]
sum_squ = null_array[n][n]
sum_sqv = null_array[n][n]
for all $i \in I$ **do**
  for all $u \in U_i$ **do**
    for all $v \in U_i$ **do**
      sum_prod(u, v) += sum_prod$(u, v) + r_{ui} \cdot r_{vi}$
      sum_squ(u, v) += sum_squ$(u, v) + {r_{ui}}^2$
      sum_sqv(u, v) += sum_sqv$(u, v) + {r_{vi}}^2$
    end for
  end for
end for
for all $u \in U_i$ **do**
  for all $v \in U_i$ **do**
    $\text{sim}(u, v) = \frac{\text{sum\_prod}(u,v)}{\sqrt{\text{sum\_squ}(u,v) \cdot \text{sum\_sqv}(u,v)}}$
  end for
end for

this way (see [SBM12]). Considering again the worst case where $\mid U_i \mid = \mid U \mid$, the complexity of Algorithm 5 is $\mathcal{O}(\mid I \mid \cdot \mid U \mid^2 + \mid U \mid^2) = \mathcal{O}(\mid U \mid^2 \cdot [\mid I \mid +1])$, which is far less than that of Algorithm 4. Another undeniable advantage of Algorithm 5 is that it follows the MapReduce framework, and as such it can be easily parallelized for another significant performance gain. Indeed, the first three embedded `for` loops could be dispatched in various clusters, each dealing with a given set of items (this is the Map stage). Each of the clusters would have its own three sum matrices `sum_prod`, `sum_squ` and `sum_sqi`, which could then all be merged to rebuild

the entire sum matrices (Reduce stage). Let us note however that, even if the similarities computation can be optimized in various ways, these methods still become intractable when the number of users is too high, so other methods are usually deployed.

That's all for the neighborhood methods. In the next section, we describe another very popular collaborative filtering technique, which models the data in a significantly different (but meaningful!) way: matrix factorization techniques.

4.3.2 Matrix factorization techniques

About ten years ago, the Netflix company organized a competition where the goal was to improve the RMSE of their standard prediction algorithm by 10%. The challenge quickly became very popular, not only because the winning prize was of $1 million, but also because the dataset was orders of magnitude larger than any other available dataset at the time: about 100 million ratings from 480.000 users and 17.700 movies. Unfortunately, the dataset is no longer publicly available, but the challenge led to the emergence of many successful recommendation techniques, among which matrix factorization methods clearly stand out. Let's be honest: the author of this document has a crush on matrix factorization techniques, and as a consequence they will be thoroughly detailed. All the more, just like neighborhood approaches, they have become one of the main baselines for benchmarking.

The matrix factorization approach we will describe here is heavily inspired by the Singular Value Decomposition (SVD) of a matrix, one of the highlights of linear algebra. Because having a basic understanding of SVD will be very insightful for us, we will briefly review it now.

4.3.2.1 Background on SVD

Let's first dive into the theory (but not for long):

Proposition 4.1. *Any real-valued matrix $R \in \mathcal{M}^{m \times n}$ of rank r can be decomposed as the product of three matrices*[2]*:*

$$R = U \Sigma I^t,$$

where $U \in \mathcal{M}^{m \times r}$, $\Sigma \in \mathcal{M}^{r \times r}$ is a diagonal matrix, and $I \in \mathcal{M}^{n \times r}$. U and I are both ***orthogonal*** *matrices, and this factorization is called the* ***Singular Value Decomposition*** *(SVD) of R.*

In practice, we know how to compute the two matrices U and I: their columns are the eigenvectors[3] of the two matrices $R^t R$ and $R R^t$, and their associated eigenvalues are the squared entries of Σ, called the singular values (hence the name of the factorization). As Σ is a diagonal

[2]The matrix I is **not** the identity matrix!

[3]For this reason, SVD and Principal Component Analysis are strongly related.

matrix, it only acts as a scaling factor for either U or I. For the sake of simplicity, we will consider that the decomposition can be written $R = UI^t$, where Σ has been merged into either U or I.

The key point is that the columns of U are actually an **orthonormal basis** for the column space of R, and the columns of I are an **orthonormal basis** for the row space of R. Maybe this statement is worth some explanation. The column space of R is the vector space that is spanned by the n columns or R, i.e. the set of vectors that are a linear combination of the columns of R. As some of the columns of R may be linearly dependent, this vector space is of dimension $r \leq n$: the rank of a matrix is defined as the number of independent columns, and equivalently as the number of independent rows. Our statement says that the columns of U span the columns space or R, and particularly that they form an orthonormal basis of r vectors for this vector space. The same goes for the r orthogonal columns of I, which span the row space of R.

A particular case of this general statement is that **any column of R can be expressed as a unique linear combination of the columns of U**. As the columns of U are orthonormal, each column has a unique contribution that cannot be compensated by the others. Symmetrically, **any row of R is a linear combination of the orthogonal columns of I.** What does it have to do with our recommendation problem? Imagine for a moment that R is a dense rating matrix, where the rows represent items and the columns represent users:

$$R = \begin{array}{c} \begin{array}{ccc} \text{Alice} & \text{Bob} & \text{Charlie} \end{array} \\ \left(\begin{array}{ccc} 4 & 5 & 5 \\ 2 & 1 & 1 \\ 1 & 2 & 2 \\ 3 & 3 & 3 \\ 4 & 2 & 2 \end{array}\right) \end{array} \begin{array}{c} \text{Titanic} \\ \text{Toy Story} \\ \text{LOTR} \\ \text{Mad Max} \\ \text{E.T.} \end{array} = \begin{pmatrix} | & | & & | \\ U_1 & U_2 & \cdots & U_r \\ | & | & & | \end{pmatrix} \cdot \begin{pmatrix} — & I_1 & — \\ — & I_2 & — \\ & \vdots & \\ — & I_r & — \end{pmatrix},$$

where $U_1, \ldots, U_r$ are the columns of U and $I_1, \ldots, I_r$ are the columns of I (i.e. the lines of I^t).

When we compute the SVD of R, we find in the columns of U some *prototype* users that have no real existence, but that can be combined (linearly) to build up any of the users: Alice, Bob and Charlie are linear combinations of $U_1, U_2, \ldots, U_r$. Also, any linear combination of Alice, Bob and Charlie can be expressed as a linear combination of the U_i. Similarly, the columns of I are *prototype* movies that do not properly exist but that can be combined to build up Titanic, Toy Story, etc., and any of their linear combinations. We can consider each vector I_i to be some sort of idealized movie type, for example I_1 is a typical action movie, while I_2 would be a typical romantic movie, etc. Take the example of Titanic, which is an *action-packed romance* movie. Thanks to the SVD, we can express Titanic as a linear combination $\alpha_1 \cdot I_1 + \alpha_2 \cdot I_2 + \cdots + \alpha_r \cdot I_r$, and as I_1 and I_2 represent typical action and romantic movies, the values α_1 and α_2 will be

high. Also, if I_3 is a typical science-fiction movie, α_3 should be fairly low (or even negative) for Titanic. The magic of SVD is that all these coefficients (or factors) are automatically derived from the matrix R.

Similarly, the U_i's can be viewed as some idealized critics, with specific tastes toward some particular kind of movies. Obviously in practice, there is no way to associate a clear semantic meaning to the different U_i or I_i, and the way they are defined only depends on the matrix R. We have here chosen to interpret the prototype movies as *action movie* or *romantic movie* for the sake of simplicity.

Moving further, let us now focus on the modeling of a single rating r_{ui}. Each $r_{ui} \in R$ is defined as a dot product $r_{ui} = {q_i}^t \cdot p_u$, where $q_i \in \mathbb{R}^r$ is a **row** in U and represents the item i, and $p_u \in \mathbb{R}^r$ is a **column** in I^t and represents the user u. We can now understand that p_u **represents how well** u **agrees with each of the** r **prototype movies, and** q_i **represents how well** i **suits the tastes of the** r **prototype users**.

Example:
As an example, let us consider the rating of Charlie for Titanic. We will assume that the rank r is 3 and that the three movies types are *Action*, *Romance*, and *Science fiction*. Charlie is a guy who loves action movies and likes romantic movies, but does not like science fiction at all. The two vectors p_u and q_i are described in Table 4.1, and the rating r_{ui} is high because Titanic has all the attributes (or factors) that Charlie likes[a].

[a]These values are completely arbitrary and obviously are not necessarily those that would appear in the true SVD of R.

	Action	Romance	Science fiction
Titanic q_i	1	3	0
Charlie p_u	2	1	-1

Table 4.1: Charlie's rating for Titanic is ${q_i}^t \cdot p_u = 1 \cdot 2 + 3 \cdot 1 + 0 \cdot (-1) = 5$.

Here, each rating r_{ui} is a dot product between two vectors of dimension r, because U and I are made of r orthogonal column vectors. A strong result related to SVD if we *restrict* U and I to U' and I' with $f \leq r$ columns vectors (choosing the f vectors associated with the highest corresponding values in Σ), the matrix product $U'{I'}^t$ is still a good approximation of R. It is, in fact, the **best** approximation of R of rank f with regards to the Forbenius norm[4].

4.3.2.2 Application to recommendation

The above statement is the principal motivation for the use of SVD in recommendation settings. It postulates the existence of f factors/criteria (whose nature is not necessarily known) that

[4]The Forbenius norm of a matrix is the square root of sum of its squared entries.

determine the value of any rating r_{ui}. A user u is modeled as a vector $p_u \in \mathbb{R}^f$, where each component of p_u models the importance of the corresponding factor for u. Similarly, an item i is modeled as a vector $q_i \in \mathbb{R}^f$, where each component of q_i models how well i fits the corresponding criterion. From then, a rating prediction $\hat{r}_{ui}$ is calculated as the dot product of the two vectors p_u and q_i:

$$\hat{r}_{ui} = {q_i}^t \cdot p_u.$$

Once the number of factors f is set[5], the problem is here to estimate the vectors p_u and q_i for every possible user and item. We saw that when the matrix R is dense (i.e. when there is no unknown entry), there exists an analytic solution for computing the two matrices U and I. An equivalent solution is to solve the following minimization problem[6], with the constraint that the vectors q_i must be orthogonal, as well as the vectors p_u:

$$\sum_{r_{ui} \in R} \left(r_{ui} - {q_i}^t \cdot p_u \right)^2.$$

But what happens in a recommendation setting, where the matrix R is extremely sparse? The SVD of R is not even properly defined! One of the main contributions during the Netflix competition was made by Simon Funk in a blog post[7], who empirically showed that **we should actually just not care** that the matrix R is sparse, and still solve the same optimization problem only using the known ratings. When R is sparse, the problem is not convex in both p_u and q_i so this optimization will only lead to a local minimum, but this method turned out to be very efficient in practice. Before that, techniques usually involved a preliminary step where the matrix R was filled using some heuristic, in order to have a fully defined problem [SKKR00]. Funk's contribution led him to the top 10 competitors of the Netflix competition at the time, and was heavily used by other competing teams, including the BellKor team that ended up winning the contest [Kor09]. It also turns out that the orthogonality constraint is useful for interpretation purposes, but in practice it usually leads to a lower generalization power, so we will simply not bother and allow the factor vectors to be non-orthogonal.

The appeal of this basic approach does not only rest upon its link to SVD, but also because it is easily extendable. It was later improved and theoretically studied in many ways, for example by incorporating user and items biases [Kor10] (which we will further detail in Section 5.2.2), by adding the ability to incorporate implicit ratings along with explicit ones [Pat07, Kor10], or by adding some time-dependent features, taking into account items popularity over time [Kor09]. Still in [Kor10], a model combining matrix factorization techniques and neighborhood approaches was proposed. For a complete overview mixing all previously mentioned methods,

[5] f must be lower than or equal to the rank r. When $f < r$ we end up with a **low-rank** approximation of the matrix R.

[6] From then on we will abuse notation and use R both as a matrix (sparse or dense) and as a rating dataset as defined in Section 4.2.

[7] `http://sifter.org/~simon/journal/20061211.html`

see [KB11] or [KBV09].

In [SM08], the matrix factorization we just saw was studied from a Bayesian perspective, and was named Probabilistic Matrix Factorization (PMF), leading to the following regularized least squares problem:

$$\min_{p_u,q_i} \sum_{r_{ui}\in R} \left(r_{ui} - {q_i}^t \cdot p_u\right)^2 + \lambda \left(\|q_i\|^2 + \|p_u\|^2\right),$$

where λ is a regularization term. Not surprisingly, Stochastic Gradient Descent (SGD) tends to perform really well when it comes to solving this problem, and is quite simple to implement. We have already talked too much about matrix factorization, but a few lines more won't hurt so we now propose to derive the associated SGD algorithm.

4.3.2.3 Optimization by Stochastic Gradient Descent

When we need to minimize a loss function L of the form $L(\theta) = \sum_k L_k(\theta)$, where θ is a (potentially multidimensional) parameter, SGD comes up as a very popular optimization technique. While a vanilla gradient descent would require to compute the derivative of L with respect to θ, in SGD we just assume that the derivative of each L_k is enough. SGD is an iterative optimization process where the value of θ is updated using the following rule: $\theta \leftarrow \theta - \alpha\frac{\partial L_k}{\partial \theta}$, where α is the learning rate (or stepsize). This update is performed over all k's for a given number of times. Various theoretical guarantees about the convergence of this algorithm have been derived, that obviously depend on the stepsize and on some regularity conditions on the loss function L: see [BCN16] for a recent overview.

As far as we are concerned, our loss function is:

$$L(p_u, q_i) \stackrel{\text{def}}{=} \sum_{r_{ui}\in R} \left(r_{ui} - {q_i}^t \cdot p_u\right)^2 + \lambda \left(\|q_i\|^2 + \|p_u\|^2\right).$$

Let L_{ui} be the cost associated with a single prediction r_{ui}:

$$L_{ui} \stackrel{\text{def}}{=} \left(r_{ui} - {q_i}^t \cdot p_u\right)^2 + \lambda \left(\|q_i\|^2 + \|p_u\|^2\right).$$

Denoting err_{ui} the prediction error ($\text{err}_{ui} \stackrel{\text{def}}{=} r_{ui} - \hat{r}_{ui} \stackrel{\text{def}}{=} r_{ui} - {q_i}^t \cdot p_u$), the derivative of L_{ui} with respect to q_i is (remember that q_i is a vector):

$$\frac{\partial L_{ui}}{\partial q_i} = -\text{err}_{ui} \cdot p_u + 2\lambda q_i.$$

Similarly, the derivative of L_{ui} with respect to p_u is:

$$\frac{\partial L_{ui}}{\partial p_u} = -\text{err}_{ui} \cdot q_i + 2\lambda p_u.$$

With these derivatives in mind, the optimization problem can then be easily solved by following Algorithm 6. Another optimization technique can be used, by noting that if either p_u or q_i is

Algorithm 6 Stochastic Gradient Descent for matrix factorization.

Input: A set of ratings R, a learning rate α, a regularization penalty λ and a number of iterations n.
Output: p_u and q_i, the user and item factors.
Initialization:
Randomly initialize p_u and q_i
for iteration $\in [1, n]$ **do**
  for all $r_{ui} \in R$ **do**
    $\text{err}_{ui} = r_{ui} - {q_i}^t \cdot p_u$
    $q_i = q_i - \alpha(\text{err}_{ui} \cdot p_u - \lambda q_i)$
    $p_u = p_u - \alpha(\text{err}_{ui} \cdot q_i - \lambda p_u)$
  end for
end for

fixed, we obtain a convex problem that can be solved using classical least squares methods. The idea is then to first consider all q_i as constants and solve the associated linear problem. Then, the p_u are considered constant and the associated problem is solved. Repeating these steps an arbitrary number of time will also converge to a local solution, and this method is known as Alternating Least Squares [BK07b].

Conclusion

The purpose of this chapter was to provide the necessary background and tools, in preparation for the next chapter where we will describe our contributions on analogical recommendation.

We have seen that the three main families of recommender systems are the content-based, collaborative and knowledge-based techniques. Collaborative filtering is by far the most popular family of techniques today, so we will try to address it in our contributions. In particular, we will try to build algorithms that are capable of predicting the rating r_{ui} that a user u has given to an item i, on the basis of some past rating history.

We thoroughly detailed two popular collaborative filtering techniques. The first one is the neighborhood-based method, which is a direct implementation of the general k-NN strategy. The neighborhood of each user is computed using a similarity metric that is based on the ratings: two users are considered close if they like the same items and dislike the same items. Then, the prediction $\hat{r}_{ui}$ is set as an aggregate (usually a weighted mean) of the ratings of the neighbors of u for the item i. The neighborhood-based methods tend to model local relationship in the data. In contrast, the matrix factorization-based techniques, whose popularity has grown during the Netflix Prize, tend to model global effects in the data. These techniques define ratings in terms of user and item factors. Each user u is modeled as a vector of factors which

indicates how much the user likes the given factors. Each item i is also modeled as a vector of factors which indicates how well i corresponds to the given factors. The prediction r_{ui} is then set as the scalar product between these two vectors, while the vectors are estimated using a global estimation procedure (usually) based on stochastic gradient descent. These two techniques (neighborhood and matrix factorization) will serve as baselines in the next chapter to compare our algorithms performances.

We now have enough background on the recommender systems field to apply analogical learning to a recommendation task. This is the purpose of the next chapter.

Résumé du chapitre Nous avons vu dans le chapitre précédent que les classifieurs analogiques proposaient des résultats prometteurs dans des domaines booléens, comparativement aux classifieurs k-NN. Nous nous intéressons, dans ce chapitre et dans le suivant, à l'usage de l'apprentissage par analogie à des tâches plus concrètes, telles que celle de la recommandation.

Les systèmes de recommandation sont des outils de filtrage automatique qui permettent de fournir des suggestions d'items aux utilisateurs d'un système. Ils permettent de répondre à des questions aussi variées que "dans quel restaurant manger?", "quel film visionner?", etc. Pour cela, les systèmes de recommandation fournissent des listes de suggestions aux utilisateurs. Le but de ce chapitre est de donner les prérequis nécessaires au sujet de systèmes de recommandation, en vue du chapitre suivant où nous décrirons nos contributions à la recommandation analogique.

Nous avons vu qu'il existe trois familles principales de systèmes de recommandation : les méthodes *content-based*, les méthodes par filtrage collaboratif, et les méthodes dites *knowledge-based*. Le problème que nous nous proposons d'étudier ici est celui de la prédiction de notes : étant donné un ensemble de notes passées caractérisant les interactions d'un groupe d'utilisateurs avec un groupe d'items, le but d'un algorithme de prédiction est d'estimer les notes manquantes. Pour ce genre de problèmes, les systèmes collaboratifs sont généralement nettement plus performants que les méthodes *content-based*. Nos contributions seront donc de nature collaborative.

Nous avons décrit en détail deux techniques populaires de filtrage collaboratif. La première est la technique par voisinage, qui est une généralisation directe des méthodes k-NN. Pour prédire la note d'un utilisateur pour un item, le *voisinage* de l'utilisateur est estimé, puis on procède à une agrégation des notes des voisins pour l'item cible. Le voisinage est généralement estimé à l'aide d'une mesure de similarité qui caractérise à quel point deux utilisateurs ont tendance à noter les items avec les mêmes notes. Ce genre de méthode a tendance à modéliser des interactions locales sous-jacentes aux données, contrairement aux méthodes par factorisation de matrice qui modélisent des effets globaux. Les techniques par factorisation de matrice modélisent les utilisateurs et les items comme des vecteurs de *facteurs latents*, lesquels sont estimés via un problème d'optimisation. La note d'un utilisateur pour un item est alors donnée par le produit scalaire entre leurs deux vecteurs respectifs. Ces deux familles de méthodes (voisinage et factorisation de matrice) nous serviront à comparer les performances de nos algorithmes dans le prochain chapitre.

Chapter 5

Analogical recommendation

Content

Now that we have enough background on recommender systems, we are in a position to present our contribution to analogical recommendation.

This chapter is structured as follows. In Section 5.1, we will describe our first approach which is directly inspired from extended classifiers, and compare it to the neighborhood approach and matrix factorization-based techniques. In Section 5.2 we will devise a new algorithmic framework that is still inspired by analogy, but that does not rely on the search of 3-tuples which will appear to be very costly. Finally in Section 5.3, we will address the problem of mining analogical proportions in partially described databases, and describe how the solving of this problem can be applied to a recommendation task. This will also help us to retrospectively analyze the results obtained in Section 5.1.

5.1 An algorithm using arithmetic proportion

We will here describe our first attempt to use an analogical proportion-based algorithm for recommender systems. Let us note that applying analogical reasoning to the task of recommendation has already been addressed in [SAO+11], though with a significantly different concern. The main idea of our algorithm is quite simple and is directly inspired by the analogical classifiers that we have extensively studied in Chapter 3.

5.1.1 Algorithm

Here again, we will make use of the analogical inference principle, but we will state it in slightly different terms. In chapters 2 and 3, the analogical inference principle stated that if four vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}$ are in proportion, then their labels should also be in proportion. Here, we will say that if an analogical proportion stands between four users a, b, c, d, meaning that for each item j that they have commonly rated, the analogical proportion $r_{aj} : r_{bj} :: r_{cj} : r_{dj}$ holds, then it should also hold for an item i that a, b, c have rated but d has not (i.e. r_{di} is the missing component):

$$\frac{r_{aj} : r_{bj} :: r_{cj} : r_{dj} \quad \forall j \in I_a \cap I_b \cap I_c \cap I_d}{r_{ai} : r_{bi} :: r_{ci} : r_{di} \quad \forall i \in I_a \cap I_b \cap I_c, \text{ and } i \notin I_d}$$

This leads us to estimate r_{di} as the solution y of the following analogical equation:

$$r_{ai} : r_{bi} :: r_{ci} : y.$$

Naturally, we may find many 3-tuples (a, b, c) that are in proportion with u, so we will need to aggregate all the candidate solutions y. Given a pair (u, i) such that $r_{ui} \notin R$, the main procedure to estimate r_{ui} is as follows:

1. Find the set of 3-tuples of users a, b, c such that an analogical proportion stands between a, b, c, and u and such that the equation $r_{ai} : r_{bi} :: r_{ci} : y$ is solvable.
2. Solve the equation $r_{ai} : r_{bi} :: r_{ci} : y$ and consider the solution y as a candidate rating for r_{ui}.

3. Set $\hat{r}_{ui}$ as an aggregate of all candidate ratings.

This pseudo-algorithm is exactly that of a conservative classifier described in Section 3.1.1, except that here, the instance space (i.e. the space of a, b, c, u) changes with every pair (u, i), and every 3-tuple (a, b, c).

In our implementation, we have used the arithmetic proportion in $\mathbb{R}^m$: $a : b :: c : u \iff a - b = c - u$. The four users a, b, c, u are here considered as vectors of ratings in a space defined by their common items. For the sake of clarity though, we will not write them with boldface letters and a will denote both the user a as well as its vectorial representation. Let us consider Figure 5.1: the four users a, b, c and u are in proportion, i.e. they make up a parallelogram in the 2-dimensional space of their common items, namely the two movies Titanic and Toy story. If the three users a, b, c have rated a movie i that u has not rated, the estimation of r_{ui} will be such that a, b, c and u still make up a parallelogram, but this time in a 3-dimensional space. This process is also explained in Table 5.1.

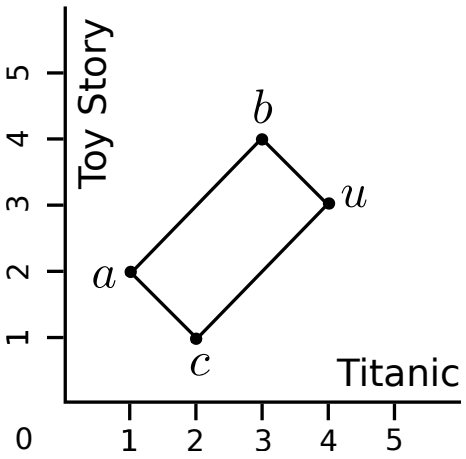

Figure 5.1: Four users a, b, c, u that are in proportion.

	j_1	j_2	j_3	$\cdots$	i
a	1	4	3	$\cdots$	2
b	5	2	3	$\cdots$	4
c	1	5	3	$\cdots$	3
u	5	3	3	$\cdots$	?

Table 5.1: The four users a, b, c, u are in proportion for every item j that they have commonly rated. For an item i that u has not rated, the prediction $\hat{r}_{ui}$ is set as the solution of the analogical equation $2 : 4 :: 3 :?$, i.e. $\hat{r}_{ui} = 3 - 2 + 4 = 5$, using the arithmetic proportion.

Just like with the conservative classifier, we will face the problem that in practice, we may not find any 3-tuple (a, b, c) such that a perfect proportion stands between a, b, c and u. We will thus allow some distortion of shape for the parallelogram $abcu$ by choosing another condition,

for example that $\|(a-b)-(c-u)\| \leq \lambda$, where λ is a suitable threshold and $\|\cdot\|$ is any p-norm. Let us notice that this relaxing of the definition of an analogical proportion exactly corresponds to the usage of an analogical dissimilarity, so technically our algorithm is very close to that of an extended classifier.

Our analogical proportion-based algorithm for recommendation is described by Algorithm 7.

Algorithm 7 Analogical proportion-based algorithm for recommendation.

Input: A set of known ratings R, a user u, and an item i such that $r_{ui} \notin R$.
Output: $\hat{r}_{ui}$, an estimation of r_{ui}.
Init:
$C \leftarrow \varnothing$ // The set of candidate ratings
for all users a, b, c in U_i such that:

- $\|(a-b)-(c-d)\| \leq \lambda$
- $r_{ai} : r_{bi} :: r_{ci} : y$ is solvable

do
$y \leftarrow r_{ci} - r_{ai} + r_{bi}$
$C \leftarrow C \cup \{ y \}$ // Add y as a candidate rating
end for
$\hat{r}_{ui} \leftarrow \text{mean}_{y \in C}\ y$

We will note here an important point: while an extended classifier would look for the k 3-tuples with the least values of analogical dissimilarity, we here look for all 3-tuples whose AD is below some given threshold. This is a common variant of the k-NN algorithm: you can either look for the k nearest neighbors, or look for all instances that are within a given distance.

We have considered a strict condition for the solvability of the equation $r_{ai} - r_{bi} = r_{ci} - y$: as the exact arithmetic result $y = r_{ci} + r_{bi} - r_{ai}$ does not necessarily belong to the rating scale used in our experiments (which is $[1, 5]$), we have considered that the equation is solvable only when $r_{ai} = r_{bi}$ or $r_{ai} = r_{ci}$. In both cases, we ensure that the solution y is in $[1, 5]$. We also tried another option where an equation is considered solvable when the solution belongs to $[1, 5]$. This option led to extremely close results, where differences where not significant.

Just like for the neighborhood approach described earlier, it is perfectly possible to apply this algorithm in an item-based way rather than in a user-based way. That is to say, instead of looking for 3-tuples of users, we may look for 3-tuples of items. As both methods lead to similar results when compared to the performances of the others recommendation algorithms, we will only focus on the user-based approach described here.

In the next section, we will extensively study the performances of our analogical recommender, and compare it to other standard recommendation algorithms.

5.1.2 Experiments and results

The performances of our analogical recommender are summed up in Table 5.2. Various options were considered, that we will describe in a moment. We also report the performances of other standard recommendation algorithms that we have already mentioned:

- The basic neighborhood approach (denoted k-NN) with different similarity metrics, namely Mean Squared Difference, Cosine similarity and Pearson similarity. For these three algorithms, the size of the neighborhood was set to $k = 40$.
- The matrix-factorization algorithm (denoted PMF), as described in section 4.3.2. The number of factors was set to $f = 100$, and the optimization problem was solved by a stochastic gradient descent of 20 iterations with a constant learning rate of 0.005 and a regularization penalty of 0.02. These values are those recommended by the authors in [KB11] for the Netflix dataset, and turned out to be quite efficient for our experiments.
- For the sake of completeness we also report results for an algorithm that always predicts the average of all ratings ($\hat{r}_{ui} = \mu$ for all u, i). In addition, we also give the performances of a random algorithm that predicts random ratings based on the distribution of the dataset, which is assumed to be normal.

The reported metrics are RMSE, precision, recall, F-measure, coverage, and surprise. Precision, recall, F-measure and coverage are reported as percentages. Remember that for these dimensions high values mean high performance, while for RMSE and surprise low values are better. RMSE is the only measure that only depends on the prediction algorithm A. All the other dimensions depend on the items that we actually choose to recommend, and rely on the recommendation strategy S. Here, we have chosen to recommend i to u if $\hat{r}_{ui} \geq 4$.

All reported results are averaged over a 5-folds cross-validation procedure. Obviously, the 5 folds are the same for all of the algorithms, to allow for meaningful comparisons. The dataset that we used is the MovieLens-100K dataset[1], composed of 100,000 ratings from 1000 users and 1700 movies. Each rating belongs to the interval $[1, 5]$, and the sparsity of this dataset is of about 94%, i.e. only 6% of all possible ratings are actually known. We also report the computation time of each algorithm (roughly estimated), which is not an average but the total over the 5 folds. All our experiments have been carried out using Surprise [Hug17], a Python recommendation library that we have developed specifically for the occasion. The Surprise package will be described more in depth in the conclusion of this chapter.

We have considered various alternatives for our analogical recommender. In fact, when trying to predict a single rating r_{ui}, looking for **all** the 3-tuples of users in U_i as described in Algorithm 7 is simply impractical[2]: there are often too many users in U_i. As the complexity of this search is in $\mathcal{O}(\mid U_i \mid^3)$ and as $\mid U_i \mid$ can be quite large for some items, strictly sticking to

[1] http://grouplens.org/datasets/movielens/

[2] Remember that U_i is the set of users that rated item i.

Algorithm	Details		RMSE	Cov	Prec	Rec	F	Surp^{max}	Surp^{avg}	Time
Parall	Sample	$n = 100$	1.18	57	95	68	79	.419	.166	10m
		$n = 1000$	1.04	23	95	27	42	.418	.168	1h
	k-NN	$k = 20$	1.00	31	97	38	54	.421	.176	6h
		$k = 30$	0.99	25	96	31	47	.422	.177	19h
k-NN	MSD	$k = 40$	0.98	23	96	28	43	.423	.175	30s
	Cos	$k = 40$	1.02	21	96	26	41	.426	.180	30s
	Pears	$k = 40$	1.01	25	95	30	46	.425	.170	30s
PMF		$f = 100$	0.95	38	99	47	64	.422	.178	45s
Mean			1.13							1s
Random			1.52	81	86	89	88	.432	.155	1s

Table 5.2: Performances of recommendation algorithms on the MovieLens-100k dataset.

Algorithm 7 would lead to even worse computation times. We thus have chosen two different strategies:

- The first one is to randomly sample n times a 3-tuple in U_i^3. As in the strict version of Algorithm 7 the prediction is an average from all the 3-tuples in U_i^3, choosing a large value of n should lead to a fairly good estimate. We have reported the results for $n = 100$ and $n = 1000$. Note though that $\mid U_i \mid^3$ is usually much larger than 1000, so we only have a very rough estimation here. This option is referred-to as **Parall Sample**.
- The second strategy is to consider the 3-tuples (a, b, c) in the neighborhood of u, using the assumption that the neighbors of u are probably more reliable to yield a prediction where u is involved. We have used the MSD metric to compute the neighborhood, and we have considered various sizes of neighborhood, namely $k = 20$ and $k = 30$. Unfortunately, values of k greater than 30 lead to basically never-ending algorithms. This option is referred-to as **Parall k-NN**.

RMSE analysis

Out of all the algorithms, the PMF algorithm is by far the most accurate, with an RMSE of 0.95. Matrix factorization models were extremely popular during the Netflix competition precisely because of their high accuracy. Note however that even though the matrix factorization algorithms cannot be overlooked when it comes to performance comparison, their philosophy remains quite different from that of other classical neighborhood approaches. They tend to model data structures at a high-level of abstraction, while neighborhood methods tend to model local features of the data. As such, it makes more sense to compare the performances of analogical recommenders to those of neighborhood-based techniques, rather than use matrix factorization-based models as a baseline.

As expected, the RMSE of the Parall Sample algorithm gets better while n grows, but still

remains quite far from that of the k-NN algorithms. By paying attention to the Parall k-NN algorithms, we see that looking for the 3-tuples in the neighborhood of the users significantly improves the accuracy of our analogy-based algorithms. Their RMSE is better than that of the neighborhood approaches when using Cosine and Pearson similarity, and it may be expected that using a neighborhood size of $k = 40$ would lead to an RMSE close to that of the k-NN algorithm with MSD similarity.

Coverage

Let us now consider the coverage measure. At first sight, it may appear that the Random and Parall Sample 100 algorithm have the best recall values. But this would be missing a very important detail: any measure that evaluates the fitness of the recommendation strategy S (such as the coverage) greatly depends **also** on the fitness of the prediction algorithm A, simply because S highly depends on A. Therefore, the coverage of the Random algorithm cannot be taken very seriously, because its accuracy (RMSE) is disastrous: it is only by chance that some items were estimated with a rating greater than 4, and were then recommended. The same goes for the Parall Sample 100 algorithm, which has a quite bad accuracy. Actually, the most reasonable choice would probably be the PMF algorithm again, which has the best RMSE and very decent coverage. With respect to coverage, our other analogy-based algorithms yield comparable performances to the neighborhood-based methods, with a slight advantage for analogy-based methods.

Precision and Recall

As for precision, recall and F-measure, we somehow have the same situation: the Random algorithm seems to be the best one, but this needs to be taken very carefully. Here again, the PMF algorithm probably yields the best actual trade-off. Comparatively, analogy-based algorithms have a slightly better F-measure than the neighborhood approaches. This may describe the fact that our Parall algorithms tend to produce more diverse recommendations, as would also suggest the results on the coverage.

Surprise

The surprise measures are a bit more tricky to analyze. Looking at $\text{Surp}^{\max}$, which assesses the least surprising recommendation (averaged over all users), we see that the analogy-based algorithms tend to perform better, but the Surp^{avg} measure counterbalances this observation. Undoubtedly, the surprise associated to a recommendation remains an extremely tricky concept to grasp and to model, and the measures presented here only allow to assess a very rough aspect of this complex dimension.

Other remarks

As a side note, we have reported only the RMSE for the Mean algorithm, because this

algorithm always outputs a prediction of $\hat{r}_{ui} = \mu = 3.53$ which is lower than the threshold we have chosen for our recommendation strategy S. Therefore, not a single item is recommended with this algorithm, and the precision, recall, etc. are not defined (or are null).

We are now led to computation time... The nightmare of every analogical proportion-based algorithm. All of the other algorithms can manage through the 5-folds cross-validation procedure in less than a minute, but our analogy-based algorithms need hours to yield only decent performances. This issue is linked to the cubic complexity of the analogy-based learners, which cannot cope with big amounts of data. For now, this limitation prevents analogy-based learners to be relevant in real-world applications such as recommender systems, where computation time is one of the most decisive factor.

Nonetheless, it is still possible to design analogy-inspired solutions for recommendation. In the next section, we describe other algorithms that rely on the notion of ***clones*** and that generalize the classical neighborhood approach.

5.2 A "clone"-based view of analogical recommendation

Considering analogies between four users has shown to be computationally intensive, thus not really suitable for recommendation purposes, where time is a highly critical dimension. Yet, other forms of analogy can be addressed in the recommendation task, based on the observation that some users may be more inclined to give good (or bad) ratings than others. Indeed, ratings are in no way absolute and greatly depend on the subjective appreciation each user has about the rating scale. In the $[1, 5]$ scale for example, two users u and v might semantically agree on an item i describing it as *bad*, but there is a chance that this agreement is not perfectly reflected in the ratings: u might have rated i with $r_{ui} = 1$ and v with $r_{vi} = 3$, simply because from v's point of view 3 is a *bad* rating, while for u a rating of 3 would simply mean *decent* or *good enough*. In the following, we refer to such users that *semantically* agree on their common items (but not necessarily *numerically*) as **clones**, as illustrated in Figure 5.2. Please note that the word *clone* is not used here to mean *strictly identical*, but rather in the sense that two clones are two users following parallel paths.

It is obvious that in collaborative filtering, clones are of great interest when it comes to predicting a user's ratings, and yet the information they provide is often discarded. Indeed, in Figure 5.2, Alice and Bob would definitely not be considered as neighbors, so Bob would not be used to predict Alice's ratings, and Alice would not be used to predict Bob's ratings. The principle underlying the analogical clone-based view is the following: for predicting a missing rating for u, we not only look at its nearest neighbors but also at the users v whose ratings are such that $r_{ui} = r_{vi} + t_{vu}$ where t_{vu} is a more or less constant **correction term** that can be either positive or negative. This correction term is the difference between Bob's ratings

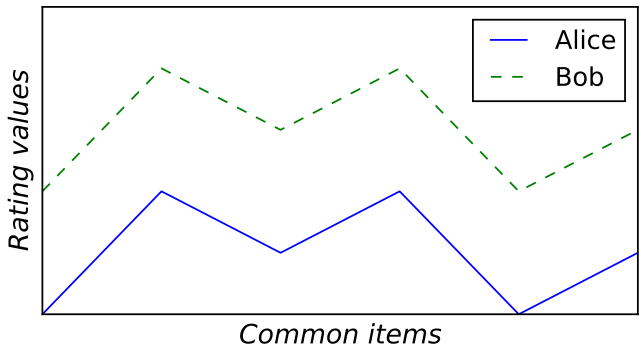

Figure 5.2: Bob is a perfect clone of Alice, and Alice is a perfect clone of Bob.

and those of Alice. When two users u and v are clones, we can come back to an analogical proportion-based viewpoint by noticing that we have:

$$r_{ui} : r_{vi} :: r_{uj} : r_{vj}, \quad r_{uj} : r_{vj} :: r_{uk} : r_{vk}, \ldots$$

where $i, j, k, \cdots$ are the common items rated by u and v. The algorithms we will derive will however be much more efficient than those of the previous section, because they will not rely on an extensive search of 3-tuples of users.

5.2.1 Two new analogical algorithms

In the following, $C_i(u)$ will denote the set of users that are clones of u and that have rated item i. From the previous informal definitions, one can easily derive a very general collaborative filtering framework for predicting a user's rating by taking into account its clones:

$$\hat{r}_{ui} = \underset{v \in C_i(u)}{\text{aggr}} \ (r_{vi} + t_{vu}),$$

where t_{vu} is a *correction term* that we need to add to v's ratings so that they correspond to those of u. We clearly have a generalization of the neighborhood approach defined in Section 4.3.1, which could be rewritten as:

$$\hat{r}_{ui} = \underset{\substack{v \in C_i(u), \\ t_{vu}=0}}{\text{aggr}} \ (r_{vi} + t_{vu}).$$

Following this general framework, one can construct a great variety of algorithms with various level of complexity. In the next subsections, we will propose a very straightforward algorithm, and a more efficient one.

5.2.1.1 A straightforward prediction algorithm

We will here design a very basic clone-based algorithm, to show that even a basic method can outperform the classical neighborhood approach.

Let us first introduce the notion of t-clone. In its most simple form, a user v can be considered to be a t-clone of u if the ratings of v exactly differ from those of u by a constant term t:

$$t\text{-}C(u) \stackrel{\text{def}}{=} \{\, v \in U \mid \forall i \in I_{uv},\ r_{ui} = r_{vi} + t \,\}.$$

From then on, computing $\hat{r}_{ui}$ amounts to finding all the users v that satisfy this criterion, and computing an aggregation of their ratings for i, which can simply be an average. We implemented this basic algorithm described by Algorithm 8, and referred to as **brute-force**.

Algorithm 8 A brute-force algorithm for clone-based recommendation.

Input: A set of known ratings R, a user u, an item i such that $r_{ui} \notin R$.
Output: $\hat{r}_{ui}$, an estimation of r_{ui}.
Init:
$C \leftarrow \varnothing$ // list of candidate ratings
for all users $v \in U_i$ **do**
  for all t **do**
    if $v \in t$-Clones(u) **then**
      $C \leftarrow C \cup \{r_{vi} + t\}$ // add $r_{vi} + t$ as a candidate rating
    end if
  end for
end for
$\hat{r}_{ui} \leftarrow \underset{c \in C}{\text{aggr}}\ c$

Of course, one may want to relax the definition of a t-clone, as the current one is too strict and only very few users will satisfy this criterion. In our implementation, we chose the following condition:

$$t\text{-}C(u) \stackrel{\text{def}}{=} \left\{ v \in U \;\middle|\; \sum_{i \in I_{uv}} |(r_{ui} - r_{vi}) - t| \leq |\, I_{uv} \,| \right\},$$

which amounts to accept v as a t-clone of u if on average, the difference $|r_{ui} - r_{vi}|$ is equal to t with a margin of 1. The values of t clearly depend on the rating scale. The datasets on which we tested our algorithms use the $[1, 5]$ interval, so possible values for t that we have considered are integer values in $[-4, 4]$.

This is obviously a very rough algorithm, to which one could point out numerous flaws. The first obvious one is its time complexity which is very high, but the purpose of this brute-force algorithm is simply to show that even such a basic clone-based approach can lead to better results than a basic neighborhood method, as we will see in the experiments section.

5.2.1.2 Modeling clones with the similarity measure

Another option to consider clones is to use the well known neighborhood-based formula, and capture their effects using an appropriate similarity measure. Recall that the general neighborhood formula is as follows:

$$\hat{r}_{ui} = \frac{\sum\limits_{v \in N_i^k(u)} r_{vi} \cdot \text{sim}(u, v)}{\sum\limits_{v \in N_i^k(u)} \text{sim}(u, v)}.$$

We have seen that this formula is commonly used with classical similarity metrics such as Pearson similarity, Cosine similarity, or inverse of MSD. However, these similarities are not fully satisfactory when it comes to clones. Indeed with these metrics, two users are considered to be close if their common ratings are often the same, but two perfect clones u and v with a significant correction term t_{vu} would be considered as being far from each other, thus involving a loss of information.

We propose the following simple choice of metric to measure how two users relate as clones:

$$\text{Clone\_dist}(u, v) \stackrel{\text{def}}{=} \frac{1}{\mid I_{uv} \mid} \cdot \sum_{i \in I_{uv}} \left[(r_{ui} - r_{vi}) - \mu_{uv} \right]^2,$$

where μ_{vu} is the mean difference between ratings of u and v:

$$\mu_{uv} \stackrel{\text{def}}{=} \frac{1}{\mid I_{uv} \mid} \sum_{i \in I_{uv}} (r_{ui} - r_{vi}).$$

We can understand this distance in two ways:

- it can be regarded as the variance of the difference of ratings between u and v,
- or it can be regarded as a simple MSD measure (defined in Section 4.3.1) to which the mean difference of ratings between u and v has been subtracted.

As our measure Clone_dist is a distance, it needs to be turned into a similarity measure. A common choice is to take its inverse (while accounting for zero division):

$$\text{Clone\_sim}(u, v) = \frac{1}{\text{Clone\_dist}(u, v) + 1}.$$

Once we know how to find the clones of a user, it is easy to output a prediction using the classical neighborhood approach:

$$\hat{r}_{ui} = \frac{\sum_{v \in N_i^k(u)} (r_{vi} + \mu_{uv}) \cdot \text{Clone\_sim}(u, v)}{\sum_{v \in N_i^k(u)} \text{Clone\_sim}(u, v)}.$$

This algorithm will be referred to as **Clone**. For the sake of completeness, we also tried the same formula but with a more basic similarity metric that does not care about clones: MSD.

5.2.2 Current advances in neighborhood-based techniques

What we have seen so far in terms of neighborhood methods are the rough, basic techniques that have existed for a long time. Actually, more sophisticated approaches have been developed, in particular during the Netflix competition. The one we will describe here has been popularized in [BK07a], and makes use of **baseline predictors**:

$$\hat{r}_{ui} = b_{ui} + \frac{\sum\limits_{v \in N_i^k(u)} (r_{vi} - b_{vi}) \cdot \text{sim}(u, v)}{\sum\limits_{v \in N_i^k(u)} \text{sim}(u, v)},$$

where b_{ui} is a baseline (or bias) related to user u and item i. Its expression is $b_{ui} = \mu + b_u + b_i$, where b_u is supposed to model how u tends to give higher (or lower) ratings than the average of all ratings μ, and b_i is supposed to model how i tends to be rated higher or lower than μ. For example, if the mean of all ratings is $\mu = 3$, and the ratings of a user are $(2, 2, 1)$, its bias b_u would be close to -1.

Baselines are computed by solving a regularized least squares problem:

$$\min_{b_u, b_i} \sum_{r_{ui} \in R} \left[r_{ui} - (\mu + b_u + b_i) \right]^2 + \lambda \left(b_u^2 + b_i^2 \right),$$

which can be achieved efficiently by stochastic gradient descent, or alternating least squares. The regularization terms are here to avoid overfitting: they allow to give more confidence to biases that are computed on a high number of ratings. In our previous example, the user had only rated 3 items so we cannot reliably say that its real bias is close to -1. The regularization term will allow to give a value closer to 0 for this user bias.

As a side note, notice that this optimization problem has the same look as the one we used for the PMF algorithm of Section 4.3.2. In fact, the use of baselines can be easily combined with the matrix factorization model, leading to the following optimization problem:

$$\min_{b_u, b_i, p_u, q_i} \sum_{r_{ui} \in R} \left[r_{ui} - \hat{r}_{ui} \right]^2 + \lambda \left(\|q_i\|^2 + \|p_u\|^2 + b_u^2 + b_i^2 \right),$$

with $\hat{r}_{ui} = \mu + b_u + b_i + q_i^t p_u$. This prediction algorithm is called SVD [KB11], and is nothing but the PMF algorithm with user and items biases. Its extension to handling implicit ratings is called SVD++ [KB11], but we will not describe it here.

In their work, the authors have used this particular similarity metric, that is in perfect accordance with their prediction formula:

$$\text{sim}(u, v) = \frac{\sum\limits_{i \in I_{uv}} (r_{ui} - b_{ui}) \cdot (r_{vi} - b_{vi})}{\sqrt{\sum\limits_{i \in I_{uv}} (r_{ui} - b_{ui})^2} \cdot \sqrt{\sum\limits_{i \in I_{uv}} (r_{vi} - b_{vi})^2}}.$$

It is simply a Pearson correlation coefficient, except that instead of centering ratings with their averages, they are centered with the baseline predictors. This seemingly simple tweak actually has various consequences that are very interesting. An intuitive and illuminating way to look at this algorithm as a whole is to see that it conceptually follows these steps:

1. Compute R', the set of all ratings normalized by the corresponding baselines: $r'_{ui} = r_{ui} - b_{ui}$. R' can be regarded as the set where all ratings are given from the same frame of reference (in this case 0), thus discarding any bias coming from the users or from the items. In R', ratings can then be considered as absolute, in the sense that they are not spoiled by the users' moods or the items inherent popularity.
2. Using R', compute similarities between users using the Cosine similarity (the Cosine similarity is the same as the Pearson correlation coefficient, except that quantities are not centered).
3. Output a prediction using the basic neighborhood formula. As this prediction belongs to the same space of R' where ratings have no bias, it needs to be transposed back to the space of R for performance evaluation purposes. This is why b_{ui} and b_{vi} are added (or subtracted) in the prediction formula of $\hat{r}_{ui}$.

In what follows, this algorithm will be referred to as k-NN*.

It is very clear that the use of the baseline predictors and the use of clone-based recommendation are motivated by the same reason: they both come from the fact that users (and items) tend to interpret the rating scale differently. This means that k-NN* implicitly takes the idea of clones into account, and thus a form of analogical reasoning. Differences and resemblances of these two approaches will be discussed in the next section.

5.2.3 Experiments and discussion

To assess the suitability of our clone-based view of recommendation, we have evaluated the accuracy of the brute-force and Clone algorithms, and compared them to the previously mentioned approaches: the basic neighborhood method (k-NN), and the neighborhood method taking into account user and item biases (k-NN*). For reasons that will become clear at the end of this discussion, we also evaluated the Bsl algorithm, whose prediction is simply the baseline predictor, i.e. $\hat{r}_{ui} = b_{ui}$. The evaluation protocol is the same as that of Section 5.1.2, i.e. results are averaged over a 5-folds cross-validation procedure. In addition to the MovieLens-100k dataset presented earlier, we also used the MovieLens-1M dataset containing 1 million ratings with 6000 users and 4000 movies. For each of these algorithms, the number of neighbors or clones used to output a prediction is $k = 40$, except for the brute-force algorithm where the number of clones cannot be controlled. The RMSE and MAE of the algorithms are reported on Table 5.3.

It is very clear that even a very straightforward approach of the clone-based recommendation principle significantly outperforms the most basic k-NN algorithm, and thus validates

Algorithm	Details		ML-100k RMSE	ML-100k MAE	ML-1M RMSE	ML-1M MAE
k-NN	MSD	$k = 40$	.979	.773	.921	.725
Brute-Force			.948	.737		
Clone	Clone_sim	$k = 40$	.936	.733	.899	.705
	MSD	$k = 40$	.931	.732	.897	.707
k-NN*		$k = 40$	.921	.721	.869	.680
Bsl			.945	.749	.909	.719

Table 5.3: RMSE and MAE of our clone-based algorithms on the MovieLens-100k and 1M datasets.

the need to take into account biases between users. The brute-force is however a lot heavier to compute, and thus not very suitable for real world recommendation purposes (its performances on the MovieLens-1M dataset simply could not be computed). The two other clone-based algorithms, however, have the exact same complexity of any k-NN-based algorithm, which is a significant improvement from the algorithm described in Section 5.1.

Our two Clone algorithms output (almost) exactly the same accuracies. This may seem a bit surprising, because the Clone MSD algorithm does not take into account clones in the similarity measure, and we would expect it to yield a lower accuracy. But this result still gives a further reason to consider clones as useful predictors: the only difference between the Clone MSD algorithm and the k-NN MSD algorithm (whose accuracy is much worse) is that in the prediction of Clone MSD, the mean differences μ_{uv} between the ratings of u and its neighbors are taken into account. If this (seemingly) simple change can make such a significant difference in the accuracies, this means that the way users relate as clones is an important feature of the dataset and should not be ignored.

Performances of the Clone algorithms are close to those of the state of the art k-NN* algorithm, yet the difference is more striking on the MovieLens-1M dataset. It is however important to understand that these algorithms differ on the following points:

- The Clone algorithms do not address item bias, which is a significant drawback. It is not unreasonable to believe that incorporating item bias in the prediction would lead to better results.
- There is a subtle yet meaningful difference of interpretation between the biases induced by both algorithms. In the Clone algorithm, biases are all pairwise, meaning that they involve two users, and they are computed on items that both users have rated. As for the k-NN* algorithm, there is no such thing as a pairwise bias. Bias for a given user is computed using only its own ratings, and is a result of a global optimization problem involving the global mean of all ratings, which means that every single rating in R has an

impact on the bias. As baselines are computed on the whole training set, they tend to capture most of the noise when the training set gets bigger. This may explain why the difference between the Clone algorithms and k-NN* is more striking on the MovieLens-1M dataset.

As previously mentioned in Section 4.3.1.2, it is recommended to perform a shrinkage on the similarity measure of algorithm k-NN*, in order to take into account the number of common items between two users: the more items they share, the more confident we are when computing their similarity [Kor10]. Such techniques can further improve both RMSE and MAE of the algorithm. Similarly, in the clone-based approach, it might be of interest to discount clones that rely on a too small number of common items.

Leaving clones for a moment, let us focus now on the Bsl predictor. Bsl outperforms by far the RMSE of the k-NN algorithm, and in fact if we go back to table 5.2 on page 104, we see that it outperforms also the algorithms mentioned there. Yet, this algorithm is unable to output personalized recommendations: all the recommendations would be the same for **any** user! This is simply due to the fact that in the prediction $\hat{r}_{ui} = \mu + b_u + b_i$, the only user-dependent factor is the term b_u, which is the **same** in all the predictions regardless of the item. The point of this remark is to exhibit one of the drawbacks of RMSE that we already mentioned before: RMSE is not an appropriate measure when it comes to assess user-system interactions.

Indeed, what ultimately matters is how well a recommender system can **rank** the preferences of a user for all the items. Also, rather than defining clones with the general numerical criterion ($r_{ui} = r_{vi} + t_{vu}$), we may instead consider that two users are clones if they order their common items in the same way. Back to Figure 5.2, we see that Alice and Bob are perfect clones and we actually do not need to know the real rating values: only the order matters. The goal of the next section is precisely to investigate this ordinal view of ratings, and to see how it can be embedded in our clone-based framework.

5.2.4 Towards an ordinal view of ratings

We will here investigate if we can devise a counterpart of the numerical clone-based approach, which would be compatible with an ordinal view of the ratings. Indeed, an extreme way for unbiasing and comparing two sets of ratings is to forget about their numerical values, and only consider their respective rankings. The idea of viewing ratings in a ordinal manner is not new, and has been advocated for example in [KS11]. In this section, we will discuss an ordinal counterpart of the analogical approach previously presented. Analogical reasoning with ordinal data has first been proposed in [BM09], yet with a different concern.

5.2.4.1 An algorithm for rank prediction

Indeed, the idea that "*the rating of user u for item i is to the rating of user v for item i as the rating of user u for item j is to the rating of user v for item j*" may be understood as well in an ordinal manner. This leads to state that "*the relative ranking of item i among the ratings given by user u is to the relative ranking of item i among the ratings given by user v as the relative ranking of item j among the ratings given by user u is to the relative ranking of item j among the ratings given by user v*".

This means that we need to compare the rankings given by two users u and v on their common items. In the following, ρ_{ui} denotes the relative ranking of item i out of all the items rated by u. Our goal is to estimate all values of ρ_{ui}, for any user and any item. The main steps of a possible algorithm would be as follows:

1. Compute similarities between users, based on their rankings. A very popular similarity ranking measure is the Spearman's rank correlation coefficient (or Spearman's rho), that will be described later.
2. Compute an estimated rank $\hat{\rho}_{ui}$ as an aggregation of all the rankings ρ_{vi} extracted from the k nearest neighbors (using Spearman's rho as similarity):

$$\hat{\rho}_{ui} = \frac{\sum\limits_{v \in N_i^k(u)} \rho_{vi} \cdot \mathrm{sim}(u, v)}{\sum\limits_{v \in N_i^k(u)} \mathrm{sim}(u, v)}.$$

This is obviously very similar to the numerical clone-based approach described before, but instead of predicting a rating, we output the prediction of a rank. This approach is denoted as **RankAnlg**.

The computation of Spearman's rho takes into account the relative rankings of the ratings, and is computed as follows:

$$\mathrm{Spearman}(u, v) \stackrel{\mathsf{def}}{=} \frac{\underset{i \in I_{uv}}{\mathrm{Cov}}\left(\rho_{ui}, \rho_{vi}\right)}{\underset{i \in I_{uv}}{\mathrm{std}}\left(\rho_{ui}\right) \cdot \underset{i \in I_{uv}}{\mathrm{std}}\left(\rho_{vi}\right)},$$

where Cov naturally stands for covariance and std is the standard deviation. Spearman's rho is usually defined as a statistic for random variables, and we here have translated it as a realization of this statistic in our recommendation setting. A common issue when dealing with rankings is that there may be ties in the data. Consider for example user u from Table 5.4, whose ratings are $1, 1, 2, 3, 3, 3$ for items i_1, i_2, i_3, i_4, i_5 and i_6. As i_1 and i_2 have the same ratings, there is no reason to rank i_1 before i_2, or i_2 before i_1. The same goes for i_4, i_5 and i_6, and this is why the first ranking option (raw rankings) is not a suitable one. Instead, it is common to consider that the ranking of a given item is the mean of the raw rankings for the items that share the same rating value. In our case, $\rho_{ui_1} = \rho_{ui_2} = \frac{1+2}{2}$, and $\rho_{ui_4} = \rho_{ui_5} = \rho_{ui_6} = \frac{4+5+6}{3}$. This process

	i_1	i_2	i_3	i_4	i_5	i_6
r_{ui}	1	1	2	3	3	3
Raw rankings	1	2	3	4	5	6
Processed rankings ρ_{ui}	1.5	1.5	3	5	5	5

Table 5.4: Values of ρ_{ui} for the user u.

leads to a better estimation of Spearman's rho, and is equivalent to averaging over all possible permutations between the raw ranks of items with equal ratings.

5.2.4.2 Experiments

We evaluated the performance of our algorithm and compared it to other previously described approaches, using the exact same evaluation protocol as in the previous sections. The MovieLens-1M dataset was not benchmarked, because the computation of Spearman's rho was too computationally intensive on this bigger dataset.

RMSE and MAE are good measure for evaluation rating prediction accuracy, but are not suitable when it comes to evaluate rankings. A better measure is the Fraction of Concordant Pairs (FCP), which evaluates the probability that given any two items i and j rated by any user u, the system has correctly estimated whether u prefers i over j, or the reverse. To compute the FCP, we need to use intermediate measures. Following the notation of [KS11], c_u defines the number of concordant pairs for user u, and d_u is the number of discordant pairs. The FCP is then computed over all users as the proportion of concordant pairs.

$$
\begin{aligned}
c_u &\stackrel{\text{def}}{=} \Big| \Big\{ (i,j) \in I^2 \quad \Big| \quad \hat{r}_{ui} > \hat{r}_{uj} \text{ and } r_{ui} > r_{uj} \Big\} \Big|, \\
d_u &\stackrel{\text{def}}{=} \Big| \Big\{ (i,j) \in I^2 \quad \Big| \quad \hat{r}_{ui} \geq \hat{r}_{uj} \text{ and } r_{ui} < r_{uj} \Big\} \Big|, \\
\text{FCP} &\stackrel{\text{def}}{=} \frac{\sum\limits_{u \in U} c_u}{\sum\limits_{u \in U} c_u + \sum\limits_{u \in U} d_u}.
\end{aligned}
$$

Here, $\hat{r}_{ui}$ may represent either a rating prediction or a ranking prediction $\hat{\rho}_{ui}$.

Results are reported in table 5.5. Unfortunately, even a basic algorithm such as k-NN that

	RankAnlg	k-NN	k-NN*
FCP	.7063	.7096	.7163

Table 5.5: FCP of our rank prediction algorithm on the MovieLens-100k dataset.

was not designed for ranking prediction performs better in terms of FCP. To explain this difference, one may look at the distribution of average support over all the predictions, as shown

on figure 5.3. Between two users u and v, the support is defined as the number of common items $|I_{uv}|$ which was used to compute the similarity between u and v. For a given prediction $\hat{r}_{ui}$, the average support is the average of all the supports $|I_{uv}|$ over all users $v \in N_i^k(u)$.

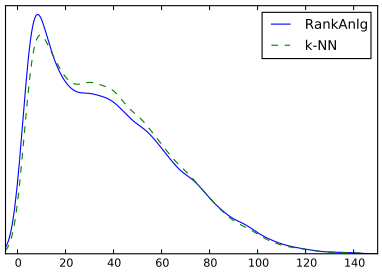

Figure 5.3: Distribution of average support for Spearman's rho (RankAnlg) and MSD (k-NN).

The use of Spearman's rho tends to provide with neighbors that have smaller support values, thus leading to a less significant and less accurate estimation of the neighborhood, which may explain the differences in performance.

This closes for now our contributions to analogical recommendation. We will come back to it in the conclusion of this chapter, and we will now focus on a somewhat unrelated problem: analogical proportion mining in databases. As we will see though, analogical recommendation can in fact be viewed as both a motivation **and** a purpose for the mining of analogical proportions.

5.3 Mining analogical proportions

Let us recap a bit the first section of this chapter, where we described our first attempt to design an analogical-based recommendation algorithm. As it is a direct application of the analogical inference principle, our algorithm suffers from the same drawbacks as the conservative and extended learners introduced in Chapter 3: the time complexity is cubic, which makes it impossible to use in actual systems and hard to evaluate accurately. Also, the performances of this algorithm were quite modest, even in its most elaborated forms.

Our analogical inference principle used in the first section states that if four users a, b, c, d are in analogy, then their ratings for an item i that d has not rated should also be in analogy:

$$\frac{r_{aj} : r_{bj} :: r_{cj} : r_{dj} \quad \forall j \in I_a \cap I_b \cap I_c \cap I_d}{r_{ai} : r_{bi} :: r_{ci} : r_{di} \quad \forall i \in I_a \cap I_b \cap I_c, \text{ and } i \notin I_d}$$

But we have so far neglected a central question: do such proportions actually exist in our database? Can we find enough 4-tuples of users such that their common ratings are in proportion? And if we can, how good are these proportions? We already acknowledged that absolutely

perfect proportions are actually hard to find, and this is why we have relaxed the condition for an analogy to hold in Algorithm 7.

In this section, we will design an algorithm that is able to answer all of the above questions, by extracting all the analogical proportions underlying a database that satisfy some given quality criterion. We will apply this algorithm to the MovieLens database, and see how this relates to our recommendation task. We will here look for analogies between items (movies in our case) instead of looking for analogies between users, because the interpretation of a movie-based proportion is actually easier, and because the parallel with association rules (described below) will be more natural. Both problems are symmetric though, and our solution can be adapted to user proportions in a straightforward manner.

Because our method for mining analogical proportions is inspired from the mining of association rules, we will first review this topic in the next subsection.

5.3.1 Association rules

Association rules are pieces of information that one can extract from a database, and that reveal dependencies between items. Recommendation is one of the principal application of association rules. Starting from the association rule $i \implies j$, which means that users that buy i tend to buy j with high probability, a recommendation system can suggest j as soon as we bought i (but not yet j). A well-known example of association rule is the famous beer $\implies$ diapers association, which was revealed after mining association rules in a supermarket selling history, suggesting that people tend to buy beer and diapers altogether. We now formally introduce the problem of **association rule mining**.

Let $I = \{ i_1, i_2, ..., i_m \}$ be a set of m items, and let $T = \{ t_1, t_2, ..., t_n \}$ be a multiset of transactions, where each transaction is a subset of I: $\forall i,\ t_i \subseteq I$. A transaction simply is a set of items that are purchased together. An association rule is expressed in the form $X \implies Y$, where $X \subsetneq I,\ Y \subsetneq I$, and $X \cap Y = \varnothing$. X and Y are sets of items that we will call **itemsets**, and usually Y is restricted to a single item. We define the **support** $\text{supp}(X)$ of an itemset X as the proportion of transactions that contain it:

$$\text{supp}(X) \stackrel{\text{def}}{=} \frac{|\ \{\ t \in T \mid X \subseteq t\ \}\ |}{|\ T\ |}.$$

Sometimes, the support is not defined as a proportion but rather as the absolute number $|\ \{\ t \in T \mid X \subseteq t\ \}\ |$. Various measures can be used to evaluate the quality of an association rule, such as the **confidence** which can be expressed as:

$$\text{Conf}(X \implies Y) \stackrel{\text{def}}{=} \frac{\text{supp}(X \cup Y)}{\text{supp}(X)}.$$

As could be naturally expected, $\text{Conf}(X \implies Y)$ is equal to 1 when the items of X and Y are

systematically bought together, and decreases if the set X is sometimes found in a transaction that does not contain Y.

The mining of association rules is a popular research topic, and has been extensively studied. In a perfect world with unlimited computational resources, a naive association rules mining algorithm would be as follows:

1. First, generate the power set of I, i.e. the set of all subsets of I that we denote 2^I. We know that the elements in 2^I are partially ordered with respect to inclusion, and 2^I can be represented as a lattice where the *join* is the union and the *meet* is the intersection.
2. Let I_S be any of the 2^m itemsets in 2^I. Then for each I_S, compute all partitions $\{ X, Y \}$ of I_S and calculate the confidence associated with the rule $X \implies Y$. If the confidence is above a given threshold, then keep the association rule, else discard it.

The second step remains reasonable and cannot really be optimized, but the first one is obviously impossible to perform in practice due to the terrific size of 2^I, when real-world databases usually contain hundreds or thousands of items. The most famous algorithm for association rule mining probably is the **Apriori** algorithm introduced in [AS94], that we will briefly review. Using Apriori allows to scan the itemset lattice in an efficient way, avoiding many useless nodes.

Ultimately, we are only interested in association rules where the support of the involved itemsets is high. An itemset whose support is above some given threshold α is called a **frequent** itemset. The downward-closure property of support states that if I_S is a frequent itemset, then all of its subsets are also frequent itemsets. For example, if the itemset { apple, banana, orange } is found in more than 30 transactions, then the three subsets { apple, banana }, { apple, orange } and { banana, orange } must also be found in **at least** 30 transactions. Conversely, if the two sets { kiwi, pear } and { kiwi, strawberry } are found in less than 30 transactions, we are sure that their union { kiwi, pear, strawberry } will also be found in less than 30 transactions. So if we are only interested in itemsets whose support is higher than 30, there is no point in building the set { kiwi, pear, strawberry }, or any of its supersets.

Taking advantage of this fact, a basic version of the Apriori can be described in the following steps:

1. Consider all itemsets of size 1 whose support is above α.
2. By joining these 1-itemsets, build all possible 2-itemsets and only keep those whose support is above α.
3. By joining these 2-itemsets, build all possible 3-itemsets and only keep those whose support is above α.
4. Repeat the process: all the frequent k-itemsets are built by joining the frequent $k-1$ itemsets.
5. Once all the frequent itemsets have been computed, the second step of the above naive algorithm is used to assess the confidence of the association rules.

In the end, we are provided with a set of association rules than comply with some quality requirements, and that give us insightful information that link the elements of our database.

Just like association rules, the identification of analogical proportions in a database is an additional information source, and deserves to be addressed. We will now get to the heart of the matter.

5.3.2 Looking for analogies in an incomplete database

In order to avoid any ambiguity, we will formally define our problem first.

5.3.2.1 Problem definition and links to association rules

We will be looking for analogies between items in the MovieLens database, but our method can naturally be extended to any other database with the same structure. We dispose of a set of users U and a set of items I. Each user u has rated (or purchased) a given set of movies $I_u \subseteq I$. In this setting, each user u can be considered as a transaction t, as defined in the previous section. The set of all the ratings r_{ui} is denoted R and can be viewed as a sparse matrix where users are columns and items are rows (or the reverse). For now, we do not care about the rating values and just focus on whether the rating exists or not (just like we only care about the fact that an item has been bought in the case of association rules). We will only make use of the rating values when we will evaluate how good the proportions are, in later subsections.

We need to insist here on the fact that we are dealing with a **sparse** database. In the case where all values in R are known, the problem of mining analogical proportion is pretty trivial: all we need is to look for all 4-tuples of items, and evaluate how good the proportion $a : b :: c : d$ is. This search is in $\mid I \mid^4$ which is extremely expensive, but there is probably no shortcut. This problem is not really interesting in the sense that there is no need to come up with an elaborated algorithm. Its parallel in the association rules world corresponds to the case where each transaction t_i contains all the items ($\forall i, t_i = I$): the very idea of association rule does not make sense anymore. The problem we propose to address here is different: the matrix R is sparse and the set of users that have rated (purchased) the items is different each time: for all items i and j (or at least for most), we have $U_i \neq U_j$. We thus cannot just look at all 4-tuples of items and check if they are in analogy. Or more accurately we could, but as we are ultimately interested in good analogies (i.e. analogies involving a sufficient number of ratings), we can make use of the downward-closure property just like the Apriori algorithm, which will allow us to avoid many 4-tuples that could not lead to good analogies.

Let's now consider four users a, b, c, d, and our task is to find out if these four users make up a valid analogy. For now, we do not know in which order we need to consider them. Just like for association rules, the notion of support still makes sense here because these four users

make up a 4-itemset:

$$\text{supp}(a, b, c, d) \stackrel{\text{def}}{=} \frac{\mid I_{abcd} \mid}{\mid I \mid},$$

where I_{abcd} is the set of items that a, b, c and d have all rated. Here again, we will only be interested in proportions whose support is greater than some threshold α: a proportion built on only two components is a lot less meaningful than a proportion built on dozens of components.

When given four items a, b, c, d, the question is now to find out which is the proportion that actually holds. It could be $a : b :: c : d$, but it could just as well be $a : b :: d : c$ or $a : c :: d : b$, or any of the 24 (4!) combinations of these four elements. Fortunately, we do not have to test all the 24 orderings. We know from Chapter 2 that there are exactly 3 equivalent classes of analogies, which are represented by:

- $a : b :: c : d$,
- $a : b :: d : c$,
- $a : d :: c : b$.

Thus, testing these three orderings is enough to find out about the 24 possible forms of analogies. To assess the quality of a proportion, we will use a function f that plays a similar role to the confidence function for association rules. Then, it will be natural to only consider analogies that are above some given quality threshold. Simply put, we would consider $a : b :: c : d$ as a valid analogy if $f(a : b :: c : d) \geq \beta$. In the end we are left with a set of proportions that represent analogical relations between four items.

5.3.2.2 Assessing the quality of a proportion

We will here describe various functions f that can assess the quality of a proportion. The four items a, b, c, d are considered as vectors of ratings in the space of their common users, and we will consider two cases: that of the binary rating scale $[0, 1]$, and that of the gradual rating scale $[1, 5]$. Note that here, in case of the binary rating scale, the value 1 is associated with *like* and the value 0 is associated with *dislike*, but a value of 0 **still** means that the user u has rated the item i. In some settings (e.g. with the unary rating scale), a value of 0 can be interpreted as the absence of rating, but this view is not compatible with our problem: if $r_{ui} = 0$ means $r_{ui} \notin R$, the four items a, b, c, d would only be represented as vectors of constant value 1, where analogies are all trivial because their only pattern is $1 : 1 :: 1 : 1$. Instead, when 0 still means that the user has rated the item, the items can be represented as Boolean vectors, which is fortunate because we know how to deal with Boolean proportions.

Naturally, the quality functions that we can define will depend on the nature of the rating scale.

- When we have a binary rating scale, the obvious choice for assessing the quality (or rather the *badness*) of a proportion is the analogical dissimilarity defined in Section 3.1.2. For Boolean vectors, the analogical dissimilarity is defined as the number of components that

need to change in order to have a perfect proportion. Note however that we ultimately want to **compare** the quality of different proportions, and the fact is that no two 4-tuples of items will have the same common users, so the item vectors of the two different 4-tuples will likely have different dimensions. Therefore, it might be wise to consider the **relative** analogical dissimilarity, which is the classical AD divided by the dimension of the vectors.

- Another obvious quality measure in $\mathbb{B}^m$ simply is the number of components where a Boolean proportion perfectly holds. It is a slightly less conservative approach than the above one but still very similar. In practice, this means that the two proportions $0:1::1:0$ and $1:0::0:1$ will be given an AD of 1 instead of 2. Here again, considering a fraction rather than an absolute number may be more meaningful, because of the different dimensions.
- When the rating scale is numerical (e.g. $[1,5]$), the analogical dissimilarity is defined as $\|(a-b)-(c-d)\|_p$, and indicates how well the parallelogram *abcd* holds. Clearly, this can also be used as a measure of quality of the proportion.
- Another option for the gradual rating scale is to use the definitions of the fuzzy analogy evaluation described at the end of Section 2.1. As the quality of each proportion is evaluated in a component-wise fashion, we can choose various aggregation functions to assess the overall quality: mean, max, min, or also compare two proportions by lexicographic order. We will give further details in the experiments section.
- Finally, by adopting a statistical point of view, we can try to evaluate the probability of observing the proportion $a:b::c:d$, and consider it a meaningful proportion if it is unlikely that we could have observed this proportion by random chance alone. This method is highly linked to statistical test theory.

5.3.2.3 Algorithm

Our algorithm for mining analogical proportions imitates an association rule mining process: we preliminarily set a threshold α for the support, and a quality evaluation f along with a threshold β.

After having built all the 4-itemsets whose support is above the threshold α with the Apriori algorithm, we compute the quality of the proportions associated with the three equivalent classes, and keep those that satisfy our criterion. These steps are described in Algorithm 9.

In theory, it is possible to end up with two non-equivalent proportions in $\mathcal{P}$ that still relate to the same four items, i.e. we could find in $\mathcal{P}$ the proportion $a:b::c:d$ as well as the non-equivalent proportion $a:b::d:c$. It should not seem natural to have these two proportions in $\mathcal{P}$, so if this happens, it is probably because the quality function f is too permissive or because the threshold β is not correctly tuned.

We also want to stress the point that the actual rating values are only used in the second part of the algorithm, i.e. when we evaluate the quality of the proportions. In the first part

Algorithm 9 Analogical proportion mining.

Input: A set of known ratings R, a quality function f, and two thresholds α and β.
Output: A set $\mathcal{P}$ of analogical proportions between items.
$\mathcal{P} \leftarrow \varnothing$
Candidate retrieval:
Using Apriori, derive all the 4-itemsets whose support is greater than α.
Quality evaluation:
for all (a, b, c, d) in the set of 4-itemsets **do**
  for all prop $\in \{ (a : b :: c : d), (a : b :: d : c), (a : d :: c : b) \}$ **do**
    if $f(\text{prop}) \geq \beta$ **then**
      $\mathcal{P} \leftarrow \mathcal{P} \cup \{ \text{prop} \}$
    end if
  end for
end for

involving the Apriori algorithm, the only information that matter are that a rating exists. Its value is not taken into account.

5.3.3 Experiments and discussion

As previously indicated, we have considered the MovieLens-100k dataset for our experiments: 100,000 ratings in $[1, 5]$ from 1000 users and 1700 movies. We will first treat ratings as numerical values, and in the second part of this investigation we will binarize them.

5.3.3.1 Numerical ratings

For our purpose, we actually do not need to set a quality threshold β, which would by the way be quite arbitrary. Instead, we will only be interested in **comparing** the quality of the proportions. We will not make use of the analogical dissimilarity because the AD is dependent on the dimensions of the vectors, and we will necessarily compare proportions that have different dimension so this would not make sense. Instead, we have chosen to compare two proportions by first computing the truth value of each of their component-wise proportions using the graded-view of analogical proportions described in Section 2.1, and then by comparing these truth values in lexicographic order. This allows to compare proportions that have different dimensions in a more meaningful way.

This fuzzy view of analogical proportion is recalled here: for four real elements $a, b, c, d \in [0, 1]$, the quality of the analogical proportion $a : b :: c : d$ is evaluated by:

$$A(a, b, c, d) = \begin{cases} 1 - |(a - b) - (c - d)| & \text{if } a \geq b \text{ and } c \geq d, \text{ or } a \leq b \text{ and } c \leq d \\ 1 - \max(|a - b|, |c - d|) & \text{if } a \geq b \text{ and } c \leq d, \text{ or } a \leq b \text{ and } c \geq d. \end{cases}$$

Example:
Consider for example the two 4-itemsets of Table 5.6, where ratings $[1, 2, 3, 4, 5]$ have been mapped to $[0, .25, .5, .75, 1]$. We will consider that the first 4-tuple i_1, i_2, i_3, i_4 (on the left) is a better proportion than the second one because once their truth values are sorted, it is the one that comes first. Had we chosen to compare them with the mean of the truth values, the best proportion would have been the second one. In the remaining of this discussion, a **component-proportion** will denote a proportion between ratings (for example the proportion $0 : 1 :: 0 : 1$ is the first component-proportion of $i_1 : i_2 :: i_3 : i_4$), and in general a proportion will refer to an analogy between items (i.e. it is an aggregation of component-proportions).

i_1	i_2	i_3	i_4	A	i_5	i_6	i_7	i_8	A
0	1	0	1	**1**	0	0	0	0	**1**
0	.5	.25	.75	**1**	0	.25	1	.75	**.75**
1	.5	1	.25	**.75**	1	.5	1	.25	**.75**
0	.5	1	.25	**.25**	1	1	1	.75	**.75**

Table 5.6: Two 4-itemsets and their related proportions, with corresponding truth value (A).

Using this comparison procedure, we have looked for the 10 best proportions with a minimum support of 200 common users. The results are reported on Table 5.7.

	i_1	i_2	i_3	i_4
1	Star Wars	The Emp. Strikes Back	Raid. of the Lost Ark	Return of the Jedi
2	Star Wars	Return of the Jedi	Raid. of the Lost Ark	The Emp. Strikes Back
3	Star Wars	The Emp. Strikes Back	Return of the Jedi	Raid. of the Lost Ark
4	Star Wars	Raid. of the Lost Ark	Return of the Jedi	I.J. and the Last Crus.
5	Star Wars	Return of the Jedi	Raid. of the Lost Ark	The Fugitive
6	Star Wars	Raid. of the Lost Ark	Return of the Jedi	Back to the Future
7	Star Wars	The Emp. Strikes Back	Raid. of the Lost Ark	I.J. and the Last Crus.
8	Star Wars	The Emp. Strikes Back	Return of the Jedi	I.J. and the Last Crus.
9	Star Wars	The Emp. Strikes Back	I.J. and the Last Crus.	Return of the Jedi
10	Star Wars	Raid. of the Lost Ark	The Emp. Strikes Back	The Fugitive

Table 5.7: The ten best item proportions with a support of more than 200 common ratings, using A.

The first obvious observation is that only a few movies (exactly 7) make up these 10 proportions. This is not really surprising, as we have set the support threshold quite high, and only a few movies are rated by more than 200 users. In fact, we found 331 4-itemsets with more than 200 common users, but they only contain 37 unique movies. Clearly here the involved movies (Star Wars, Indiana Jones…) are extremely popular, and people tend to give them high ratings. Using notions introduced in Section 5.2.2, we can say that these movies have a high

item bias b_i. To confirm this claim, we can check that in average, the mean rating of these 7 movies is of 4.09, while the average rating of all movies is only 3.53.

Some of the analogies are actually quite good though. For example the fourth one *Star Wars (1977)* is to *Raiders of the Lost Ark (1981)* as *Return of the Jedi (1983)* is to *Indiana Jones and the Last Crusade (1989)*, or the seventh one which is similar: *Star Wars (1977)* is to *The Empire Strikes Back* as *Raiders of the Lost Ark (1981)* is to *Indiana Jones and the Last Crusade (1989)*

Looking at the first and third proportions, we see that the two forms $i_1 : i_2 :: i_3 : i_4$ and $i_1 : i_2 :: i_4 : i_3$ are present in the table. These two forms are non-equivalent, and it does not seem natural to consider these two proportions as (almost) equally valid. But this result can be explained by looking at the actual rating values: a great number of them are of the form $r : r :: r : r$ (where r is usually a high rating, given that these movies are popular). In the two cases, a switch between i_3 and i_4 has absolutely no effect on the truth values of the analogies.

This leads us to another point: all these analogies are, after all, quite trivial. They all concern quite similar movies. We do not mean to offend any cinema fan, but we believe it is still fair to say that the differences between a Star Wars movie and an Indiana Jones movie are quite shallow. And these similarities are reflected in the ratings, where the most common pattern is $r : r :: r : r$ (or close to it), suggesting that people like all these movies the same.

To try to discover some more surprising analogies, we basically have two options. The first one is to tweak a bit our quality function f (and in our case, our comparison function) by penalizing proportions with the pattern $r : r :: r : r$, or inversely by promoting the proportions $r : r' :: r : r'$ where r and r' are quite different. The second option is to lower the minimum support threshold, to allow movies that are rated by a reasonable (and not necessarily very high) number of users. As we have seen, movies with a very high number of ratings are likely to lead to trivial proportions precisely because they are popular, and because people like them all equally well in general. If we really want to ban popular movies, we can also set a **maximum** support threshold. This will also have the benefit to limit the number of itemsets that we consider, leading to a significant improvement in computation time. These two options will be explored.

Promoting *disagreement* proportions

Table 5.8 shows the 10 best proportions with a minimum support of 200, with a tweaked comparison function. Here, the truth value of a component-proportion $r_1 : r_2 :: r_3 : r_4$ is defined by:

$$A'(r_1, r_2, r_3, r_4) = \frac{1}{2}A(r_1, r_2, r_3, r_4) + \frac{1}{4}|r_1 - r_2| + \frac{1}{4}|r_3 - r_4|.$$

We are here promoting proportions where rating values tend to disagree, which should lead to different proportions from that of Table 5.7. Proportions are still compared using the lexicographic order of all the A' values.

	i_1	i_2	i_3	i_4
1	Star Wars	Pulp Fiction	The Empire Strikes Back	The Fugitive
2	Star Wars	Pulp Fiction	Return of the Jedi	The Fugitive
3	Star Wars	Pulp Fiction	The Empire Strikes Back	The Silence of the Lambs
4	Star Wars	Twister	Return of the Jedi	Independence Day
5	Star Wars	Return of the Jedi	The Silence of the Lambs	Fargo
6	Star Wars	The Terminator	Return of the Jedi	Pulp Fiction
7	Star Wars	Pulp Fiction	Return of the Jedi	The Terminator
8	Star Wars	The Fugitive	The Empire Strikes Back	The Silence of the Lambs
9	Star Wars	Pulp Fiction	Return of the Jedi	The Silence of the Lambs
10	Star Wars	Pulp Fiction	Raiders of the Lost Ark	The Silence of the Lambs

Table 5.8: The ten best item proportions with a support of more than 200 common ratings, using A'.

We still have a lot of Star Wars movies (and a few Indiana Jones), but the movies seem more diverse: we now have 11 unique movies, instead of just 7 before. 9 of these proportions comply with the pattern *Star Wars movie* is to *Other Star Wars movie* what *movie A* is to *movie B*. This suggests that we should find strong links underlying the connection between *Movie A* and *Movie B*, because the two Star Wars movies are deeply related. Is *Pulp Fiction* related to *The Fugitive*? Is *Twister* related to *Independence Day*? Is *Fargo* related to *The Silence of the Lambs*? We will not venture to answer these questions, and leave them to the reader's appreciation. All we can say is that some forms of analogy seem to emerge from their respective ratings. One thing we can note however, is that even though all these movies are very popular just like in Table 5.7, their genres are however quite different: while we have considered Star Wars and Indiana Jones to be of the same kind, we can fairly claim that a Star Wars movie is very far from Fargo, Pulp Fiction or The Silence of the Lambs.

Lowering support threshold

Table 5.9 illustrates the ten best proportions we have found after setting the minimum support at 10, and a maximum support at 50. This drastically reduces the number of 4-tuples that are explored, and avoids any highly popular movie. Now, we find up to 20 unique movies building up these 10 proportions, which is quite an improvement with respect to the previous settings. But let's be honest, we have come to a point where the author is unable to further hide his lack of cinematographic culture. He has never heard of any of these movies, and will not pretend to.

Still, we shall try to make sense out of this. It seems that our proportions exhibit some sort of clustering: movies of proportions 1, 3, 8 and 9 are all movies from the 40'-50's. Clearly the users that rated the corresponding movies must be some fans of the old movie-making era. Similarly, proportions 2 and 10 are all movies from 1994 or 1995, suggesting a niche. Here again

	i_1	i_2	i_3	i_4
1	To Catch a Thief	Laura	Gigi	An American in Paris
2	Nick of Time	It Could Happen to You	Milk Money	Only You
3	To Catch a Thief	An American in Paris	Gigi	Meet John Doe
4	Dangerous Minds	Money Train	Higher Learning	With Honors
5	Judge Dredd	Under Siege 2: Dark Territory	The Shadow	Mortal Kombat
6	Terminal Velocity	Under Siege 2: Dark Territory	Money Train	Drop Zone
7	Judge Dredd	Under Siege 2: Dark Territory	Mortal Kombat	Coneheads
8	To Catch a Thief	An American in Paris	Gigi	Laura
9	To Catch a Thief	Meet John Doe	Gigi	An American in Paris
10	Nick of Time	It Could Happen to You	Only You	Milk Money

Table 5.9: The ten best item proportions with a support between 10 and 50.

it is not really surprising that we observe the two patterns $a : b :: c : d$ and $a : b :: d : c$. This comes from the fact that these movies are often given similar (and high) ratings. The other proportions involve movies that are still from the 90's and that are given pretty bad reviews from critics: they seem to belong to the *so bad it's good* kind.

5.3.3.2 Boolean ratings

To complete this investigation, we have led the same experiments with binary ratings. The ratings of the MovieLens dataset were binarized by associating the value 1 if the rating r_{ui} is above the user's average rating μ_u (i.e. the user liked the movie) and 0 otherwise (the user disliked the movie):

$$\text{binary}(r_{ui}) = \begin{cases} 1 \text{ if } r_{ui} \geq \mu_u \\ 0 \text{ else.} \end{cases}$$

The quality of a movie-based proportion is here the fraction of perfect component-proportions. When setting a minimal support of 200 without any other constraint, it should not come as a surprise that the movie-proportions that we find are extremely similar to that of Table 5.7. They involve the same 7 movies and are so alike that we chose not to show them here to avoid redundancy. Indeed, the set of candidate 4-itemsets are necessarily the same in the two settings: recall that on the first stage of the algorithm, we do not care about the rating values so the fact that ratings are binary or numerical does not change anything. The fact that the best proportions are still the same is an argument in support of the coherence of the two quality functions.

There are in total 993 candidate 4-itemsets. Figure 5.4 shows the quality of these 993 proportions in descending order. For the best proportions, a perfect analogy stands on more than 70% of the components. This is quite significant as we are measuring this fraction over more than 200 components. However, we have seen that much of these proportions are not

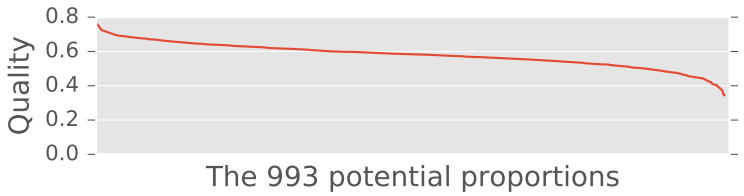

Figure 5.4: Quality of the 993 proportions. The quality of a proportion is defined as the fraction of component-proportions that stand perfectly.

really interesting as all the common users tend to agree on the popular movies, and the four movies of a proportion tend to be clustered into the same genre.

To try to exhibit disagreements in the ratings, we used another quality function. We assign a truth value of 1 to an analogy $r_1 : r_2 :: r_3 : r_4$ if and only if the proportion complies with one of the following patterns:

- $0 : 1 :: 0 : 1$,
- $1 : 0 :: 1 : 0$,
- $0 : 0 :: 1 : 1$,
- $1 : 1 :: 0 : 0$.

Simply put, we have forbidden the *agreeing analogies* $0 : 0 :: 0 : 0$ and $1 : 1 :: 1 : 1$. Figure 5.5 gives us the fraction of correct proportions for the 993 potential analogies, using this custom quality function. Obviously, the quality drops significantly for the mere reason that we have

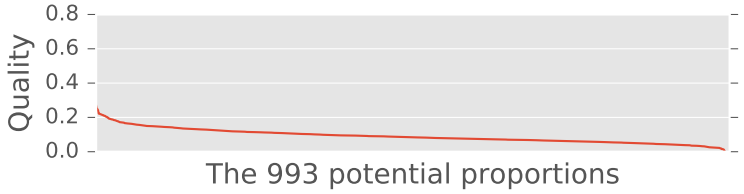

Figure 5.5: Quality of the 993 proportions. The quality of a proportion is defined as the fraction of components that make up perfect proportions, forbidding patterns $0 : 0 :: 0 : 0$ and $1 : 1 :: 1 : 1$.

forbidden patterns like $r : r :: r : r$. This tells us that in fact, **there are simply not a lot of interesting analogies**: the best proportion that we found is *interesting* on less than 25% of its components, which is quite small. This confirms our previous observation that sadly, most of the analogies we find tend to relate four movies that are actual **neighbors**. We will see in the conclusion of this chapter how this fact can retrospectively explain the performances of our basic analogy-based recommender from the first section.

Conclusion

In this chapter, we have detailed our contributions to the field of recommender systems. We have proposed two frameworks based on analogy for the design of rating prediction algorithms, and also proposed a method for mining analogical proportions in an incomplete database.

Section 5.1 described our first attempt at building an analogy-based prediction algorithm for recommendation. This algorithm is a direct application of the analogical inference principle, and is strongly related to the extended classifier that we have described in Chapter 3. The basic underlying idea is that if four users (a, b, c, d) are in analogy regarding their common items (using the arithmetic proportion), then it should also be the case for other items that d has not yet rated. We presented a user-based view where analogies were considered between users, but it is perfectly possible to consider analogies between items.

We have compared this algorithm with the two main approaches of collaborative filtering algorithms, namely neighborhood methods and matrix factorization-based techniques. Our algorithm yields decent accuracy compared to neighborhood methods, but is extremely expensive in terms of computational resources. It was clear that this simple and straightforward application of the analogical inference principle could not be used in real-world settings, where computation time is a scarce resource.

These observations led us to consider another framework for analogical recommendation in Section 5.2. Acknowledging the fact that each user tends to have a very personal idea of what a good (or bad) rating should be, we introduced the notion of clones which captures how two users tend to agree semantically on a rating. This semantic agreement is not necessarily reflected in the numerical rating values: for a user u, the value 3 may be a good rating, while for v it may be a bad one. We derived two algorithms relying on this clone-based view of the ratings. We also described a quite recent technique in collaborative filtering, that allows to address user and item biases. This method is based on the fact that some users tend to give higher ratings than others, and some items tend to be rated higher than others. We discussed how this view (using biases) is related to our view (using clones), and pointed out the similarities and differences of the two approaches. It turned out that these two techniques are really close, and the resemblance is reflected in the algorithms performances. Yet, as the bias-based method relies on a global optimization problem, it tends to be better at capturing the overall noise in the data, and still outperforms our method. Our last contribution to analogical recommendation was to consider an extreme case of clone-users, considering that the rating scale $[1, 5]$ could be interpreted as purely ordinal. Unfortunately, this track of research did not lead to promising results.

In Section 5.3, we proposed an algorithm for the mining of analogical proportions in a partially described database. This general method perfectly applies to the problem of mining analogies between users or items in a rating dataset, such as the one that was used in the first two sections. Based on the Apriori algorithm, our method is inspired by the mining of

association rules and allows to extract the potential analogies in an efficient way. We proposed many different quality functions allowing to assess the quality of a proportion, and naturally some of the quality functions are strongly related to the Analogical Dissimilarity as defined in Chapter 3. We gave examples of some of the analogies that we found using different variations of the general mining algorithm, treating ratings as numerical values and as Boolean values. The main take-away message was that the best analogies that could be extracted between items were involving users with similar tastes: the four movies of a proportion were mostly neighbors, in the sense that users tend to agree on their respective ratings. Also, when forcing proportions to display a disagreement in the ratings of the four movies, the best proportions that were revealed still had a pretty bad quality.

This allows us to retrospectively interpret the results of our first analogy-based prediction algorithm. We have seen that its performances were close to that of a neighborhood-based technique, but in the light of what has just been explained we can fairly say that there were no analogies to find in the database, except for those involving four neighbors. So after all, our analogy-based inference principle was reduced to that of the classical k-NN approach. Clearly in that case, paying the price of the cubic search in the set of all users is really not worth it. This is entirely dependent on the database that was used (in our case, the MovieLens dataset), and we may still presume that there should exist a dataset where the use of analogical proportion is beneficial over a simple neighborhood approach.

Even if our investigations on analogical recommendation did not completely live up to our expectations, there still is a contribution of ours that is worth mentioning. As pointed out earlier in the first section, all of our recommendation experiments were carried out using Surprise[3] [Hug17], an open source Python recommendation engine that we have developed. Quite surprisingly, when we started our experiments a few years ago, there was no Python library that would allow to easily run cross-validation procedures, or that would make the implementation of a recommendation algorithm easy. But most importantly, it seemed that most of the implementation details of the available algorithms were never explicitly mentioned, so it made comparing performances of the algorithms difficult, if not meaningless. So we decided to write our own package. We started out by writing our own custom (and hacky) scripts, which grew up to become a much more complete and fairly popular recommendation engine. As of April 2017, the main features of Surprise are the following ones:

- Users can have perfect control over their experiments, and to this end a strong emphasis was laid on documentation, which we have tried to make as clear and precise as possible by pointing out every detail of the algorithms.
- Dataset handling is made easy. Custom datasets can be used, and the MovieLens datasets are natively supported.
- Various ready-to-use prediction algorithms are available, such as baseline algorithms,

[3] `http://surpriselib.com`

neighborhood methods, matrix factorization (SVD, PMF, SVD++, NMF), and many others. Also, various similarity measures (Cosine, MSD, Pearson …) are built-in.

- The implementation of new algorithm ideas is made extremely simple.
- Cross-validation procedures can be run very easily, as well as exhaustive search over a set of parameters for optimization (much like the Grid Search feature of the excellent machine learning library scikit-learn [PVG+11]).

With that, we now close our discussion on analogical recommendation and on practical applications of analogy. In the next chapter, having these findings in mind, we will go back to our original theoretical considerations and try to challenge the analogical inference principle that we have so far "blindly" followed.

Résumé du chapitre Ce chapitre décrit nos contributions à la recommandation analogique.

Nous avons d'abord décrit nos premières expérimentations relatives à l'implémentation d'un algorithme de prédiction de notes fondé sur les proportions analogiques. Cet algorithme est une application directe du principe d'inférence analogique, et il est très étroitement lié aux classifieurs étendus que nous avons décrits dans le chapitre 3. L'idée principale derrière cet algorithme est que si quatre utilisateurs (a, b, c, d) sont en proportion vis-à-vis des items qu'ils ont conjointement notés (en utilisant la proportion arithmétique), alors cela devrait aussi être le cas pour d'autres items que a, b, c ont noté mais que d n'a pas encore noté. La prédiction de la note de d pour ces items est alors le résultat de la résolution de plusieurs équations analogiques. Notons qu'il est parfaitement possible de considérer des proportions entre items plutôt que des proportions entre utilisateurs.

Nous avons comparé cet algorithme avec les deux principales approches de filtrage collaboratif : les méthodes par voisinage et les méthodes par factorisation de matrice. Les performances de notre algorithme sont raisonnables du point de vue de la précision, comparativement aux méthodes par voisinage. Cependant, sa complexité algorithmique leur est bien supérieure et le rend inadapté à des usages où le temps de calcul est une des contraintes principales.

Ces observations nous ont conduit à proposer une autre façon d'envisager la recommandation analogique. Pour cela, nous avons introduit la notion de *clones* : deux utilisateurs sont des clones s'ils ont tendance à donner des notes *sémantiquement* équivalentes. Cependant, cette équivalence sémantique n'est pas forcément reflétée dans les notes : pour un utilisateur u, une note de 3 peut représenter une bonne note, alors que pour l'utilisateur v cela représenterait une mauvaise note, ou juste une note moyenne. Nous avons proposé deux algorithmes qui suivent cette idée. Nous avons aussi présenté une contribution récente de technique par filtrage collaboratif motivée par le fait que certains utilisateurs ont tendance à donner des notes plus élevées que d'autres, et que certains items ont tendance à être notés avec plus de clémence. Nous avons expliqué par quels aspects ces deux approches se rapprochent et diffèrent l'une de

l'autre. Du point de vue des performances, les deux méthodes restent assez similaires avec tout de même un léger avantage pour les méthodes qui ne reposent pas sur les *clones*.

Enfin, nous avons proposé un algorithme pour la fouille de proportions analogiques dans des bases de données partiellement complètes. Cette méthode générale s'applique parfaitement au problème de la fouille de proportions entre utilisateurs ou entre items, dans une base de notes telle que celle utilisée dans nos précédentes expérimentations. Notre algorithme est inspiré de la fouille de règles d'associations et repose sur l'algorithme Apriori pour extraire des analogies potentielles de manière efficace. Nous avons proposé diverses fonctions de qualité permettant d'évaluer la qualité d'une proportion, lesquelles sont très proches de la dissimilarité analogique décrite dans le chapitre 3. Nous avons donné divers exemples de proportions trouvées dans la base de notes, révélant des proportions entre films. Une des principales observations est que les meilleures analogies que l'on pouvait extraire de cette base ne mettaient en jeu que des utilisateurs avec des goûts similaires : les quatre films d'une même proportion étaient généralement également aimés, ou également rejetés. En ce sens, on peut considérer que les quatre items d'une même proportion sont en fait des voisins.

Ceci nous permet d'interpréter rétrospectivement les résultats modestes de nos premières expérimentations sur la prédiction de note. Nous avons vu que les résultats de notre algorithme étaient proches de ceux des méthodes par voisinage, ce qui s'explique maintenant par le fait que les meilleures proportions dont l'algorithme disposait mettaient en jeu des voisins. En quelque sorte, notre algorithme analogique était réduit à une approche par voisinage classique. Il est clair que dans ce cas là, il n'est pas nécessaire d'avoir recours à la recommandation analogique, puisque l'on est obligé de payer le prix d'une complexité cubique. Il reste à noter que ces résultats ont été obtenus sur la base de données MovieLens : on peut toujours supposer que sur une autre base, les proportions potentielles restent significatives et qu'alors l'usage de la recommandation analogique reste intéressant.

Pour finir, permettons-nous de mentionner que nos expérimentations sur la recommandation nous ont conduit à développer un package Python appelé Surprise [Hug17] consacré aux algorithmes de prédiction de notes, dont les principales fonctionnalités sont les suivantes :

- Les utilisateurs peuvent avoir un contrôle total sur le protocole expérimental mis en oeuvre. A cet effet, une attention particulière a été attachée à la documentation, que l'on a essayé de rendre la plus claire et détaillée possible.
- La gestion des jeux de données est facilitée. Certaines bases de données classiques sont pre-intégrées, telle que les bases MovieLens.
- Divers algorithmes de prédiction sont implémentés, tels que les méthodes de voisinage, des méthodes par factorisation de matrice, et d'autres. D'autre part, plusieurs mesures de similarité sont disponibles d'office.
- La création de nouveaux algorithmes de prédiction est extrêmement simple.
- Les procédures de validation croisée sont faciles à implémenter. Une autre fonctionnalité

intéressante est la possibilité d'effectuer des recherches exhaustives sur l'espace des paramètres d'un algorithme, pour déterminer la combinaison de paramètres qui produit les meilleurs résultats (*grid search*).

Chapter 6

Analogy-preserving functions

Content

We first introduced the analogical inference principle in Chapter 2. This principle states that if four elements are in analogy, then their classes (or any relevant attribute) should also be in proportion. This is of course an unsound inference principle, in that the conclusion does not always follow from the premises. Nonetheless, we have seen in Chapter

3 that it can still be used to design some classifiers that we have named conservative and extended classifiers. Both in [BMD07] and in our experiments in Section 3.3, these classifiers showed promising results, but were still only used in artificial Boolean classification problems. Analogical classifiers still had to prove their worth in real-world applications, and this was precisely the purpose of the previous chapter, where we designed various techniques to apply analogical inference on a recommendation task.

Unfortunately, the results of a direct application of the analogical inference principle did not quite live up to our expectations. The conclusion of the last chapter offered a partial explanation to these modest results, stating that there were just too few decent analogies to be found in the datasets that we used. Even if this argument is perfectly admissible, we cannot spare ourselves the very questioning of the analogical inference principle. We say that if four elements are in proportion then so are their classes... But why should it even be true? Actually, this question of *why* will probably be best answered by philosophers, rather than by computer scientists. We can however answer the question of *when*: when is the analogical inference principle sound? Said differently, what are the conditions on the function f so that the inference

$$\frac{\mathbf{a} : \mathbf{b} :: \mathbf{c} : \mathbf{d}}{f(\mathbf{a}) : f(\mathbf{b}) :: f(\mathbf{c}) : f(\mathbf{d})}$$

is sound? The current chapter will be devoted to this question, in a Boolean setting.

It is actually pretty easy to build up simple problems where the application of our analogical inference principle leads to disastrous results. Take for example the problem of Figure 6.1. We have a set S of points (on the left) that belong to two classes that are linearly separable: an instance is a square if it is on the left side of the space, else it is a circle. A naive application of the analogical inference principle would lead us to believe that for any element that is in proportion with a 3-tuple in S^3 (i.e. the element is the fourth vertex of a parallelogram that we can build with three elements in S), then the classes of the four vertices are in proportion. In the right figure, we show all these elements, along with their estimated classes. The result is absolutely terrible, because the whole space is covered with instances belonging to both classes. In this example, an analogical classifier would lead to very poor accuracy results.

But if it is so easy for our analogical inference principle to fail, then why does it work well in some other cases? Back to Figure 3.6 on page 72, the accuracies of the NAN algorithm were definitely not as bad as that of the problem we have just seen, and some of them were actually really good. A first obvious difference between the two experiments is that the one we just described deals with the arithmetic proportion with real-valued instances, while the other one deals with Boolean attributes. Does it mean that the analogical inference principle is bound to fail with the arithmetic proportion? We will probably not be able to completely answer this question, even though we will provide some insights. We will instead focus on the Boolean setting which has seemed to be the most promising for analogical inference so far, and from

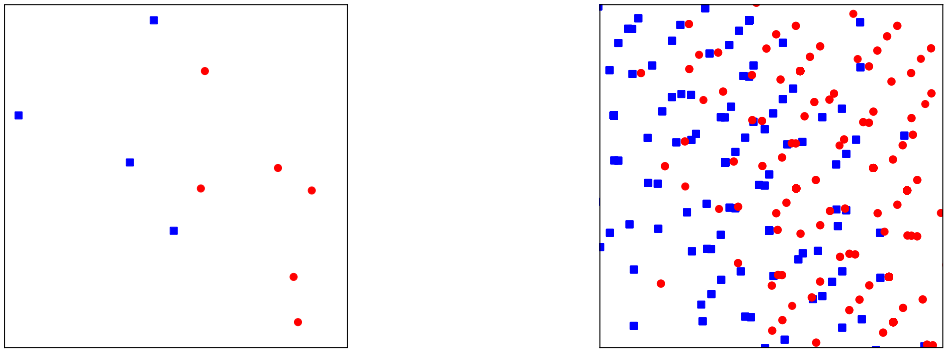

Figure 6.1: A naive application of the analogical inference principle in $\mathbb{R}^2$.

there we will derive some links with the real-valued case.

In this chapter, we will provide a complete characterization of functions that are fully compatible with the analogical inference principle in a Boolean setting. Let us restate the inference principle in this context to explain what we mean: for any four elements $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}$ in $\mathbb{B}^m$, the analogical inference principle states that if $\mathbf{a} : \mathbf{b} :: \mathbf{c} : \mathbf{d}$ holds, then $f(\mathbf{a}) : f(\mathbf{b}) :: f(\mathbf{c}) : f(\mathbf{d})$ also holds, where f is a Boolean function from $\mathbb{B}^m$ to $\mathbb{B}$. We have said it before: this principle is unsound. The goal of this chapter is to identify all of the functions f that lead to a valid conclusion, i.e. the functions such that $\mathbf{a} : \mathbf{b} :: \mathbf{c} : \mathbf{d} \implies f(\mathbf{a}) : f(\mathbf{b}) :: f(\mathbf{c}) : f(\mathbf{d})$ (with some minimal and natural requirements). We will say that these functions are **analogy preserving** (AP).

We will tackle this problem by adopting the general point of view of the training set extension. In Section 6.1, we will first define the problem of training set extension, describe some previous works and explain how this problem can motivate the research of analogy preserving functions. In Section 6.2, we will give a complete and accurate depiction of analogy preserving functions by showing that the class of analogy preserving functions is the class of (Boolean) affine functions. This strong theoretical result will be complemented in Section 6.3, where we will study some functions that derive from AP in various ways. We also address the identification of AP functions in a nominally valued setting and in a real setting, where we will once again witness the close links between the Boolean proportion and the arithmetic proportion. This chapter is an extended version of [CHPR17].

6.1 Extending a training set

We have proved in Chapter 3 that the analogical classification process can be formalized via two new conceptual steps:

1. First, the analogical extension $\mathbf{E}_S^f$ of the training set is computed. This extension is the set of all elements in the instance space that are in analogy with one 3-tuple in the training

set, provided that the associated class equation is solvable. Each element of the extension is associated with an estimated label: the analogical label denoted $\overline{f}(x)$.

2. Then, a k-NN algorithm is run, considering that the training set is now the analogical extension. This k-NN classifier allows to classify all the elements of the universe that are not in the analogical extension.

But then a natural question arises: why should we stick to a k-NN classifier? The reason why the k-NN algorithm is used is because in their algorithmic description, extended learners rely on an analogical dissimilarity, which is strongly related to a distance on the instance space. But now that we have an equivalent definition of extended classifiers that does not rely explicitly on an analogical dissimilarity, nothing actually prevents us from using any other classifier such as a decision tree, a support vector machine, or any of the dozens of learners available out there. This would lead us to a different paradigm for analogical learning: you first extend the training set by building the analogical extension, and then you are free to do how you please and use this extension as a bigger training set with any classifier.

In the next subsection, we will formally describe the problem of training set extension, and present some previous works on this subject. Then in subsection 6.1.2, we will introduce the class of analogy preserving functions, which lead to a perfectly sound training set extension in all cases.

6.1.1 Previous works on training set extension

Machine learning algorithms usually require training on sufficiently large datasets. The problem of learning from few examples is not new and is referred-to as one-shot learning, where we rely on a transfer of knowledge from known classes to unknown classes: see for example [LFFP06] in a pattern recognition context. But other options are available when the number of training instances is small, and training set extension naturally comes in as a handy tool. Indeed, the extension of the training set is a simple idea to improve the generalization power of the learner. The more examples you have, the better you learn! The point is here to add to the training set S some new examples, and to try do that in a way that preserves the *quality* of S.

Formally, we start with a set $S = \{\mathbf{x}^{(i)} \in X^m | i \in [1, n]\}$ of examples (n is supposed to be small), where $\mathbf{x}^{(i)}$ is an element of a Cartesian product $X^m = X_1 \times \ldots \times X_m$. For each element $\mathbf{x}^{(i)} \in S$, we associate a target $f(\mathbf{x}^{(i)}) = y^{(i)} \in Y$. In the case of regression $y^{(i)} \in \mathbb{R}$, and in the case of classification $y^{(i)}$ belongs to a finite set.

Actually, only a few methods have been proposed for extending a training set with new examples, and most of them are specific to a particular application. For example in [CPCAL02], some techniques are proposed but they are only relevant for character recognition tasks. In the general case, we may consider that we can build a new example starting from 1, 2 or 3 known examples.

1. With one example, a natural way to proceed is to use the classical neighborhood approach: given one example $(\mathbf{a}, f(\mathbf{a}))$, we can generate a new example $(\mathbf{b}, f(\mathbf{b}))$ where $\mathbf{b}$ is not too far from $\mathbf{a}$ and $f(\mathbf{b})$ is not too far from $f(\mathbf{a})$. In the classification case, $f(\mathbf{b})$ may be chosen as $f(\mathbf{a})$. This is the setting in which [CPCAL02] operates, where new characters are generated by slanting, shrinking, and dilating of training instances.
2. With two examples, the previous neighborhood option is still available and leads to interpolate the new example from the two given ones. A somehow different option is the Feature Knockout procedure [WM04], which amounts to build a third example obtained by modifying a randomly chosen feature of the first example with that of the second one. This way to proceed enjoys nice properties and appears to be equivalent to the popular Tikhonov regularization technique in the case of linear regression (also known as Ridge regression). A related idea is used in a recent proposal [BPR16] which introduces a measure of oddness with regard to a class that is computed on the basis of pairs made of two nearest neighbors in the same class; this is equivalent to replacing the two neighbors by a fictitious representative of the class.
3. With three examples $(\mathbf{a}, f(\mathbf{a})), (\mathbf{b}, f(\mathbf{b})), (\mathbf{c}, f(\mathbf{c}))$, the previous options remain available and lead to build a fourth example which is somehow in-between the three other ones: we still have some kind of interpolation. A quite different idea is to extrapolate the fourth item on the basis of analogical proportions. In this perspective, this fourth element is not necessarily in the neighborhood of the three others. In fact, this idea has been addressed in [BMMA07], where the authors generated new examples that had the least analogical dissimilarity with some 3-tuples in S^3. The analogical dissimilarity was an AD between sequences, and they used it to generate new examples of handwritten characters modeled using the Freeman coding, which is a sequence of symbols indicating the successive directions of the pen that are needed to write the character. They used their extended training set to train an SVM and a Radial Basis Function Network for a classification task, and results showed that the classifiers had better results when they were trained with the analogically-extended training set than when they were trained with an extended training set using other generation techniques (such as those of [CPCAL02]). Note though that just like the algorithmic description of the extended classifier using analogical dissimilarity, we here only have an algorithmic description of this training set extension procedure. The authors in [BMMA07] never explicitly refer to the analogical extension set $\mathbf{E}_S^f$, and this is normal because the unifying framework between the works in [BMD07] and the works in [SY05a] were only made later in [HPRS16a], as described in Chapter 3.

As far as we know, the problem of generation of new examples has never been addressed from a theoretical point of view (except for the Knockout procedure, which is only relevant in some particular settings), and this is actually quite understandable. Before they can be used in an extended training set, the new generated examples have to be assigned an estimated

label... But how do we estimate these labels? Well there is a name for such problems: it is called classification, and classification was the actual reason why we wanted to extend the training set in the first place, so we have a vicious circle here. In some sense, extending a training set with potentially noisy data amounts to nothing but performing classification on a specific set of instances.

However, training set extension can still be extremely useful if we are perfectly sure that the new added examples are sound, i.e. that their estimated labels are the correct ones. In the next subsection, we will formally define this problem in the context of analogical learning. Section 6.2 will then be devoted to the identification of the cases where such a perfect extension is feasible.

6.1.2 Analogy preserving functions: a safe way to extend a training set

In the rest of this chapter, we will address the problem of extending a training set using analogical proportions in a Boolean setting. Instead of using an analogical dissimilarity like in [BMMA07], we will use our equivalent functional framework described in Chapter 3, i.e. we will consider that the extension of the training set S is the analogical extension $\mathbf{E}_S^f$, where f is the function underlying the ground truth labels. Algorithmically, the generation of $\mathbf{E}_S^f$ and the computation of the analogical labels goes as follows:

1. Add every $\mathbf{x} \in S$ to $\mathbf{E}_S^f$. Then, for every $\mathbf{a}, \mathbf{b}, \mathbf{c} \in S$ such that $f(\mathbf{a}) : f(\mathbf{b}) :: f(\mathbf{c}) : y$ is solvable and such that there is $\mathbf{x} \in \mathbb{B}^m \setminus S$ with $\mathbf{a} : \mathbf{b} :: \mathbf{c} : \mathbf{x}$, add $\mathbf{x}$ to $\mathbf{E}_S^f$ and save the solution $\text{sol}(f(\mathbf{a}), f(\mathbf{b}), f(\mathbf{c}))$ as a candidate for $\overline{f}(\mathbf{x})$.
2. Then for every $\mathbf{x} \in \mathbf{E}_S^{f*} \stackrel{\text{def}}{=} \mathbf{E}_S^f \setminus S$, run a **majority-vote procedure**: set $\overline{f}(\mathbf{x})$ as the most common candidate among all solutions $\text{sol}(f(\mathbf{a}), f(\mathbf{b}), f(\mathbf{c}))$ where $(\mathbf{a}, \mathbf{b}, \mathbf{c})$ is in the analogical root of $\mathbf{x}$. In case of a tie between two labels, then one of the values is chosen at random. For elements in S, $\overline{f}(\mathbf{x})$ is simply set to $f(\mathbf{x})$.

The extension $\mathbf{E}_S^f$ is then considered to be an extended training set, where the labels are the analogical labels, **which are not necessarily correct**. Indeed for some elements $\mathbf{x} \in \mathbf{E}_S^{f*}$, it may happen that $\overline{f}(\mathbf{x}) \neq f(\mathbf{x})$, simply because the analogical inference principle may not be *suitable* for the function f. In this chapter, we are precisely interested in the identification of all Boolean functions f that lead to perfectly sound extensions, i.e. extensions where the analogical labels $\overline{f}(\mathbf{x})$ are always equal to the ground truth values $f(\mathbf{x})$. These functions will be called **Analogy preserving** functions:

Definition 6.1 (Analogy preserving functions)**.** We say that $\mathbf{E}_S^f$ is **sound** if $\overline{f}(\mathbf{x}) = f(\mathbf{x})$ for every $\mathbf{x} \in \mathbf{E}_S^{f*}$. In other words, $\mathbf{E}_S^f$ is sound if $\omega_S^f \stackrel{\text{def}}{=} P\left(\overline{f}(\mathbf{x}) = f(\mathbf{x}) \mid \mathbf{x} \in \mathbf{E}_S^{f*}\right) = 1$.
Also, if $\mathbf{E}_S^f$ is sound for all $S \subseteq \mathbb{B}^m$, we say that f is **Analogy Preserving** (AP).

Proposition 6.1 gives an equivalent definition of AP functions, that will be more useful to convey our proofs:

Proposition 6.1. *A function $f\colon \mathbb{B}^m \to \mathbb{B}$ is AP iff for every $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d} \in \mathbb{B}^m$, f suits the following requirement:*

$$\begin{cases} \mathbf{a} : \mathbf{b} :: \mathbf{c} : \mathbf{d} \text{ and} \\ \text{solvable}(f(\mathbf{a}), f(\mathbf{b}), f(\mathbf{c})) \end{cases} \implies \text{sol}\,(f(\mathbf{a}), f(\mathbf{b}), f(\mathbf{c})) = f(\mathbf{d}).$$

Proof. If f fulfills this requirement, then it is clear from the algorithmic description given above that for any $S \subseteq \mathbb{B}^m$ and $\mathbf{x} \in \mathbf{E}_S^{f*}$, all the candidates for $\overline{f}(\mathbf{x})$ are equal to $f(\mathbf{x})$, so $\overline{f}(\mathbf{x})$ will be invariably set to $f(\mathbf{x})$ by the majority-vote procedure, which makes f AP.

If f does not suit this requirement, then there exist $\mathbf{a}, \mathbf{b}, \mathbf{c}$, and $\mathbf{d} \in \mathbb{B}^m$ such that $\mathbf{a} : \mathbf{b} :: \mathbf{c} : \mathbf{d}$ but the solution $\text{sol}(f(\mathbf{a}), f(\mathbf{b}), f(\mathbf{c}))$ is not equal to $f(\mathbf{d})$. Taking $S_0 = \{\mathbf{a}, \mathbf{b}, \mathbf{c}\}$ we obtain $\mathbf{E}_{S_0}^{f*} = \{\mathbf{d}\}$, and since $\overline{f}(\mathbf{d}) = \text{sol}(f(\mathbf{a}), f(\mathbf{b}), f(\mathbf{c})) \neq f(\mathbf{d})$, then $\mathbf{E}_{S_0}^{f}$ is not sound so f is not AP. □

It should be clear from Proposition 6.1 that looking for the functions that lead to a perfect extension (i.e. the AP functions) is equivalent to looking for the functions that are compatible with the analogical inference principle, which was the first purpose of this chapter. Indeed, saying that $\text{sol}\,(f(\mathbf{a}), f(\mathbf{b}), f(\mathbf{c})) = f(\mathbf{d})$ is equivalent to saying that $f(\mathbf{a}) : f(\mathbf{b}) :: f(\mathbf{c}) : f(\mathbf{d})$.

Let us note that many natural functions are not AP. Consider for example the binary function $f(x_1, x_2) = x_1 \wedge x_2$, along with $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d} \in \mathbb{B}^2$ in Table 6.1. We have $\mathbf{a} : \mathbf{b} :: \mathbf{c} : \mathbf{d}$ and $f(\mathbf{a}) : f(\mathbf{b}) :: f(\mathbf{c}) : y$ is solvable, yet the solution is $\text{sol}(f(\mathbf{a}), f(\mathbf{b}), f(\mathbf{c})) = 0$, which is different from $f(\mathbf{d}) = 1$ so f is not AP. This actually comes from the fact that analogical proportions are not stable by conjunction combination. It is also the case for the disjunction [PR14a].

	x_1	x_2	$f(\cdot)$
$\mathbf{a}$	0	0	0
$\mathbf{b}$	0	1	0
$\mathbf{c}$	1	0	0
$\mathbf{d}$	1	1	1

Table 6.1: $f(x_1, x_2) = x_1 \wedge x_2$ is not AP.

The following section will be devoted to a complete characterization of AP functions in a Boolean setting.

6.2 A complete characterization of analogy preserving functions

We will show in this section that the class of AP functions is the class of affine functions, which will be defined later. To complete our proofs, we will first need to recall some background

knowledge on Boolean algebra and Boolean functions, which is the purpose of the next subsection.

6.2.1 Preliminary background on Boolean functions

Our first step will be to use a unifying notation for all of the Boolean functions, and to adopt an algebraic point of view. In the following, the AND operator '$\wedge$' will be denoted '$\cdot$', and '$+$' will now denote the modulo-2 addition, equivalent to the XOR operator. We shall make use of the polynomial representation of Boolean functions, that we now describe. A **monomial** is a term of the form:

$$\mathbf{x}_I = \prod_{i \in I} x_i,$$

for some possibly empty finite set of positive integers I, where $|I|$ is called the **degree** of $\mathbf{x}_I$. Simply put, a monomial is a conjunction of variables. We take the convention that 1 is the empty monomial $\mathbf{x}_\emptyset$. A **polynomial** is a sum of monomials and its degree is the largest degree of its monomials. It is well-known [Sto36, Zhe27] that **any function $f : \mathbb{B}^m \to \mathbb{B}$ is uniquely represented by a polynomial**, also called the **Algebraic Normal Form** (ANF) of f:

$$f(x_1, \ldots, x_m) = \sum_{I \subseteq \{1,\ldots,m\}} a_I \cdot \mathbf{x}_I,$$

where each a_I belongs to $\mathbb{B}$. The degree of a function $f : \mathbb{B}^m \to \mathbb{B}$, denoted $d(f)$, is defined as the degree of the unique polynomial representing f. In the following, we will often refer to f as both a function object and as the associated ANF.

Example:
To illustrate this concept, here are the ANF of some common Boolean functions in $\mathbb{B}^2$ or $\mathbb{B}$:

- $\mathrm{AND}(x_1, x_2) = x_1 \cdot x_2$;
- $\mathrm{OR}(x_1, x_2) = x_1 + x_2 + x_1 \cdot x_2$;
- $\mathrm{XOR}(x_1, x_2) = x_1 + x_2$;
- $\mathrm{NEG}(x_1) = x_1 + 1$;
- The constant function $f(x_1, x_2) = 0$ is represented by $a_\emptyset \cdot \mathbf{x}_\emptyset$ with $a_\emptyset = 0$. The constant function $f(x_1, x_2) = 1$ is also represented by $a_\emptyset \cdot \mathbf{x}_\emptyset$, but this time $a_\emptyset = 1$.

Another key concept of our proof is the notion of essential and inessential variables. For $k \in [1, m]$, $\boldsymbol{\alpha} \in \mathbb{B}^m$, and $c \in \mathbb{B}$, let $\boldsymbol{\alpha}_k^c$ be the tuple in $\mathbb{B}^m$ whose i-th component is set to c if $i = k$, and to α_i otherwise. A variable x_i is said to be **inessential** in $f : \mathbb{B}^m \to \mathbb{B}$ if for all $\boldsymbol{\alpha} \in \mathbb{B}^m$ and $c \in \mathbb{B}$, $f(\boldsymbol{\alpha}_i^c) = f(\boldsymbol{\alpha}_i^{\neg c})$. Otherwise, x_i is said to be **essential** in f, or that f depends on x_i. In simple terms, an essential variable is a variable that has the *ability* to change the value of f. In fact, it can be shown that if x_i is an inessential variable for a function f, then x_i does not appear in the ANF of f. For example in $f(x_1, x_2, x_3) = x_1 \cdot x_3$, x_1 and x_3 are

essential variables while x_2 is inessential. We denote by $\mathrm{ess}(f)$ the number of essential variables of f (or **essential arity**).

Two functions $f\colon \mathbb{B}^m \to \mathbb{B}$ and $g\colon \mathbb{B}^n \to \mathbb{B}$ are said to be **equivalent** if there exist two mappings $\sigma\colon [1,n] \to [1,m]$ and $\sigma'\colon [1,m] \to [1,n]$ such that:

$$f(x_1,\ldots,x_m) = g(x_{\sigma(1)},\ldots,x_{\sigma(n)}) \text{ and}$$
$$g(x_1,\ldots,x_n) = f(x_{\sigma'(1)},\ldots,x_{\sigma'(m)}).$$

In other words, f and g are equivalent if one can be obtained from the other by permutation of variables, addition of inessential variables, or identification of inessential variables. Another point of view is to consider that two functions are equivalent if their ANF is the same, up to the renaming of variables. For example, $f(x_1,x_2,x_3) = x_1 \cdot x_3$ and $g(x_1,x_2) = x_1 \cdot x_2$ are equivalent functions. Two equivalent functions necessarily have the same number of essential variables. For further background on the theory of essential variables of functions, see [CP08, CL09, Sal63, Wil96]. In our demonstrations, we will use the following property:

Property 6.1. *Let $f\colon \mathbb{B}^m \to \mathbb{B}$ and $g\colon \mathbb{B}^n \to \mathbb{B}$ be equivalent functions. Then f is AP if and only if g is AP.*

This can be verified by noting that as the analogy in $\mathbb{B}^m$ is defined in a component-wise fashion, the permutation of variables has no effect on the equation and its solution. Also, manipulation of inessential variables does not change the value of the function f, and thus the AP property still holds.

We now define the concept of **section** of a function, also known as a *restriction*, or equivalently as the result of *partial application* in computer science. Let f be a function $\mathbb{B}^m \to \mathbb{B}$, and (I,J) be a partition of $[1,m]$. With $\mathbf{x} \in \mathbb{B}^m$ and $\boldsymbol{\alpha} \in \mathbb{B}^{|I|}$, the I-section (or simply section) $f_I^{\boldsymbol{\alpha}}\colon \mathbb{B}^{|J|} \to \mathbb{B}$ is the function that is obtained after setting all variables in I to the components of $\boldsymbol{\alpha}$. More formally, let us define $(\mathbf{x}_I^{\boldsymbol{\alpha}}) \in \mathbb{B}^m$ as follows:

$$\begin{cases} (\mathbf{x}_I^{\boldsymbol{\alpha}})_i \stackrel{\mathsf{def}}{=} x_i \text{ if } i \notin I \\ (\mathbf{x}_I^{\boldsymbol{\alpha}})_i \stackrel{\mathsf{def}}{=} \alpha_j \text{ where } j \text{ is the index of } i \text{ in } I. \end{cases}$$

Concretely, $\mathbf{x}_I^{\boldsymbol{\alpha}}$ is nothing but the vector $\mathbf{x}$ where the components at the indices in I have been replaced by the successive components of $\boldsymbol{\alpha}$. For instance, with $m=5, \mathbf{x}=(0,1,1,0,0), I=\{1,3,4\}$ and $\boldsymbol{\alpha}=(1,0,0)$, we have $(\mathbf{x}_I^{\boldsymbol{\alpha}})=(1,1,0,0,0)$. The I-section $f_I^{\boldsymbol{\alpha}}$ is the function from $\mathbb{B}^{|J|} \to \mathbb{B}$ defined as:

$$\forall \mathbf{y} \in \mathbb{B}^{|J|}, \quad f_I^{\boldsymbol{\alpha}}(\mathbf{y}) \stackrel{\mathsf{def}}{=} f((\mathbf{x}_I^{\boldsymbol{\alpha}})_J^{\mathbf{y}}).$$

The arity of $f_I^{\boldsymbol{\alpha}}$ is $|J|$ and is always less than the arity of f, and $\mathrm{ess}(f_I^{\boldsymbol{\alpha}}) \leq \mathrm{ess}(f)$. As an

example, let us consider the function $f\colon \mathbb{B}^3 \to \mathbb{B}$ such as:

$$f(x_1, x_2, x_3) = x_1 \cdot x_2 \cdot x_3 + x_1 \cdot x_2 + x_3.$$

The section $f^{(1,0)}_{\{1,3\}}$ is a function from $\mathbb{B}$ to $\mathbb{B}$ and is defined by $f^{(1,0)}_{\{1,3\}}(x_2) = x_2$. A main result about sections that will be used in other proofs is stated in Property 6.2:

Property 6.2. *If $f\colon \mathbb{B}^m \to \mathbb{B}$ is AP, then every section of f is also AP.*

Proof. This statement is actually pretty obvious but sadly, its proof is quite ugly. Let f be an AP function from $\mathbb{B}^m$ to $\mathbb{B}$, (I, J) a partition of $[1, m]$, and $\boldsymbol{\alpha} \in \mathbb{B}^{|I|}$. We will consider the section $f^{\boldsymbol{\alpha}}_I$. Let us now assume that we have $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d} \in \mathbb{B}^{|J|}$ with the two following properties:

$$\begin{cases} \mathbf{a} : \mathbf{b} :: \mathbf{c} : \mathbf{d} \\ \text{solvable}(f^{\boldsymbol{\alpha}}_I(\mathbf{a}), f^{\boldsymbol{\alpha}}_I(\mathbf{b}), f^{\boldsymbol{\alpha}}_I(\mathbf{c})). \end{cases}$$

We want to prove that this implies that $\text{sol}(f^{\boldsymbol{\alpha}}_I(\mathbf{a}), f^{\boldsymbol{\alpha}}_I(\mathbf{b}), f^{\boldsymbol{\alpha}}_I(\mathbf{c})) = f^{\boldsymbol{\alpha}}_I(\mathbf{d})$. First, let us note that the following property holds:

$$\forall \mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d} \in \mathbb{B}^{|J|},\ J \subsetneq [1, m],\ \mathbf{x} \in \mathbb{B}^m,\ \ \mathbf{a} : \mathbf{b} :: \mathbf{c} : \mathbf{d} \implies \mathbf{x}^{\mathbf{a}}_J : \mathbf{x}^{\mathbf{b}}_J :: \mathbf{x}^{\mathbf{c}}_J : \mathbf{x}^{\mathbf{d}}_J,$$

simply due to the fact that $\mathbf{x} : \mathbf{x} :: \mathbf{x} : \mathbf{x}$ always holds.

Regarding the first condition $\mathbf{a} : \mathbf{b} :: \mathbf{c} : \mathbf{d}$, since $\forall \mathbf{x} \in \mathbb{B}^m$, we have that $\mathbf{x}^{\boldsymbol{\alpha}}_I : \mathbf{x}^{\boldsymbol{\alpha}}_I :: \mathbf{x}^{\boldsymbol{\alpha}}_I : \mathbf{x}^{\boldsymbol{\alpha}}_I$, and we deduce: $(\mathbf{x}^{\boldsymbol{\alpha}}_I)^{\mathbf{a}}_J : (\mathbf{x}^{\boldsymbol{\alpha}}_I)^{\mathbf{b}}_J :: (\mathbf{x}^{\boldsymbol{\alpha}}_I)^{\mathbf{c}}_J : (\mathbf{x}^{\boldsymbol{\alpha}}_I)^{\mathbf{d}}_J$. By definition $f^{\boldsymbol{\alpha}}_I(\mathbf{y}) = f((\mathbf{x}^{\boldsymbol{\alpha}}_I)^{\mathbf{y}}_J)$, and the second condition is just:

$$\text{solvable}(f((\mathbf{x}^{\boldsymbol{\alpha}}_I)^{\mathbf{a}}_J), f((\mathbf{x}^{\boldsymbol{\alpha}}_I)^{\mathbf{b}}_J), f((\mathbf{x}^{\boldsymbol{\alpha}}_I)^{\mathbf{c}}_J)).$$

Since f is AP, we have that $\text{sol}(f((\mathbf{x}^{\boldsymbol{\alpha}}_I)^{\mathbf{a}}_J), f((\mathbf{x}^{\boldsymbol{\alpha}}_I)^{\mathbf{b}}_J), f((\mathbf{x}^{\boldsymbol{\alpha}}_I)^{\mathbf{c}}_J)) = f((\mathbf{x}^{\boldsymbol{\alpha}}_I)^{\mathbf{d}}_J) \overset{\text{def}}{=} f^{\boldsymbol{\alpha}}_I(\mathbf{d})$, which is precisely the condition for $f^{\boldsymbol{\alpha}}_I$ to be AP. □

This section was quite dense so let us recap what we have seen. We described functions by their ANF, which is a polynomial representation. This ANF allowed us to define the degree of a function, which has a similar meaning to the notion of degree for real valued polynomials. We also defined the notion of equivalent functions, and showed that two equivalent functions are either both AP, or none of them are. Finally, we defined the section of a function which is a function of lower arity, and saw that sectioning an AP function leads to another AP function.

We are now in a position to examine some examples of AP functions. We will first show that any affine function is AP, and we will then prove that these functions are actually the only AP functions.

6.2.2 The class of Analogy Preserving functions

Let us first define the class of affine functions:

Definition 6.2 (Affine functions)**.** The class L of **affine** functions is the set of functions f of the form:

$$f(x_1, \ldots, x_m) = \alpha_1 \cdot x_1 + \ldots + \alpha_m \cdot x_m + \alpha_0,$$

where $\alpha_0, \ldots, \alpha_m$ are elements of $\mathbb{B}$. We also write:

$$f(\mathbf{x}) = \boldsymbol{\alpha} \cdot \mathbf{x} + \alpha_0,$$

where $\boldsymbol{\alpha} \in \mathbb{B}^m$ and $\cdot$ is here a scalar product. If $\alpha_0 = 0$, we say that f is **linear**.

We used the polynomial representation of the function f, and we can see that L is precisely the set of functions whose degree is less than or equal to 1. Informally, a linear function is a function that computes the exclusive OR between any subset of variables. The class L of affine functions is then the set of all linear functions along with their negations (because $f + 1 = \neg f$). A particular case of affine functions are the **projections** or **dictator** functions, defined by $f(\mathbf{x}) = x_i + \alpha_0$. These functions will be of interest in the next subsection.

Now, we will prove that every affine function is AP:

Proposition 6.2. *Any affine function f is AP.*

Proof. Let $f\colon \mathbb{B}^m \to \mathbb{B} \in L$. Using the obvious fact that f is AP iff $f + 1 = \neg f$ is AP, we may assume without loss of generality that $\alpha = 0$. Also, considering that f essentially depends on $n \leq m$ variables (n is then the number of α_i equal to 1), f is equivalent to the function $g\colon \mathbb{B}^n \to \mathbb{B}$ defined by $g(x_1, \cdots, x_n) = x_1 + \cdots + x_n$. Using Property 6.1, we just need to prove that g is AP to show that f is also AP.

This function g has the remarkable property[1] that changing the value of any x_i changes the value of g: $\forall_i, \quad g(x_1, \cdots, x_i, \cdots, x_n) = \neg g(x_1, \cdots, \neg x_i, \cdots, x_n)$, as illustrated on Figure 6.2. From this property, it is easy to see that:

$$\forall \mathbf{x}, \mathbf{x}' \in \mathbb{B}^n, g(\mathbf{x}) = g(\mathbf{x}') \iff H(\mathbf{x}, \mathbf{x}') \text{ is even},$$

where H is the Hamming distance function.

Let $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d} \in \mathbb{B}^n$ such that the two hypotheses in the definition of AP are satisfied, i.e.

$$\mathbf{a} : \mathbf{b} :: \mathbf{c} : \mathbf{d} \quad \text{and} \quad g(\mathbf{a}) : g(\mathbf{b}) :: g(\mathbf{c}) : y \quad \text{is solvable.}$$

[1] This is the reason why affine functions lead to classification problems that are, in fact, highly **non** linearly separable.

This equation is solvable in two possible cases: either $g(\mathbf{a}) = g(\mathbf{b})$, or $g(\mathbf{a}) = g(\mathbf{c})$. This can be verified in Table 2.2 on page 34, or by referring to Proposition 2.2. Also, we know from Property 2.2 that as $\mathbf{a} : \mathbf{b} :: \mathbf{c} : \mathbf{d}$, we have $H(\mathbf{a}, \mathbf{b}) = H(\mathbf{c}, \mathbf{d})$ and $H(\mathbf{a}, \mathbf{c}) = H(\mathbf{b}, \mathbf{d})$. Therefore, we either are in one of the two following cases:

1. $g(\mathbf{a}) = g(\mathbf{b})$, and in this case the solution is $\text{sol}(g(\mathbf{a}), g(\mathbf{b}), g(\mathbf{c})) = g(\mathbf{c})$. As $g(\mathbf{a}) = g(\mathbf{b})$, then $H(\mathbf{a}, \mathbf{b})$ is even, and so is $H(\mathbf{c}, \mathbf{d})$. Then, $g(\mathbf{c}) = g(\mathbf{d})$, and thus $\text{sol}(g(\mathbf{a}), g(\mathbf{b}), g(\mathbf{c})) = g(\mathbf{d})$.
2. $g(\mathbf{a}) = g(\mathbf{c})$, and in this case the solution is $\text{sol}(g(\mathbf{a}), g(\mathbf{b}), g(\mathbf{c})) = g(\mathbf{b})$. As $g(\mathbf{a}) = g(\mathbf{c})$, then $H(\mathbf{a}, \mathbf{c})$ is even, and so is $H(\mathbf{b}, \mathbf{d})$. Then $g(\mathbf{b}) = g(\mathbf{d})$, and thus $\text{sol}(g(\mathbf{a}), g(\mathbf{b}), g(\mathbf{c})) = g(\mathbf{d})$.

In both cases we have $\text{sol}(g(\mathbf{a}), g(\mathbf{b}), g(\mathbf{c})) = g(\mathbf{d})$, thus showing that g is AP. As g and f are equivalent functions, then any function $f \in L$ is AP. □

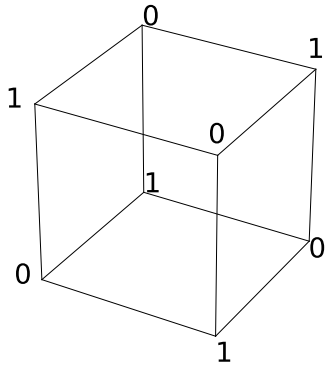

Figure 6.2: The labels as determined by the affine function $g(x_1, x_2, x_3) = x_1 + x_2 + x_3$. Each 1 is surrounded by three 0, and each 0 is surrounded by three 1. Note that solving any (solvable) analogical equation leads to the correct solution: this is because g is AP. Also, we see that we have here a classification problem that is highly non-linearly separable.

Proposition 6.2 can now retrospectively explain the results of Figure 3.6 on page 72 for the two functions f_1 and f_2, which are both instances of affine functions. We saw that for these two functions, ω_S^f (defined as the proportion of elements $\mathbf{x}$ in $\mathbf{E}_S^{f*}$ for which $\overline{f}(x) = f(x)$) was always equal to 1. This result is now obvious and directly comes from the fact that both f_1 and f_2 are actually analogy preserving functions: f_1 was a projection function while f_2 was a general affine function.

We now know that every affine function is AP. We will here give a stronger result: the affine functions are the **only** AP functions. In the previous subsection, we introduced the ANF of Boolean functions and also defined their degree. We also mentioned that the class of functions with degree at most 1 is exactly the class L of affine functions, which are AP. We will show that the class of AP functions is the class of affine functions by proving that if a function f is AP, then $d(f) \leq 1$. Let us first consider the case where $d(f) = 2$:

Property 6.3. *Let $f : \mathbb{B}^m \to \mathbb{B}$ with $d(f) = 2$. f is not AP.*

Proof. Let us consider f with $d(f) = 2$ and $\text{ess}(f) \geq 2$. We denote $\mathbf{x}_I$ one of the monomial of f of degree 2. We consider the section $f^{\mathbf{0}}_J$, where $J = [1, m] \setminus I$, and $\mathbf{0}$ denotes the constant 0 vector in $B^{|J|}$. All variables that are not part of the monomial $\mathbf{x}_I$ have been set to 0. This section $f^{\mathbf{0}}_J$ has a unique monomial of degree 2 (namely $\mathbf{x}_I$) and $\text{ess}(f) = 2$. $f^{\mathbf{0}}_J$ is necessarily equivalent to one of the following functions:

$$\begin{cases} f_1(x_1, x_2) = x_1 \cdot x_2 + \alpha_0 \\ f_2(x_1, x_2) = x_1 \cdot x_2 + x_1 + \alpha_0 \\ f_3(x_1, x_2) = x_1 \cdot x_2 + x_1 + x_2 + \alpha_0. \end{cases}$$

To see that none of these functions are AP, consider Table 6.2 which gives the values of f_1, f_2 and f_3 (with $\alpha_0 = 0$) for the 4-tuple $(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d})$. It is clear that the counter-example $(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d})$

	x_1	x_2	f_1	f_2	f_3
a	0	0	0	0	0
b	0	1	0	0	1
c	1	0	0	1	1
d	1	1	1	0	1

Table 6.2: Examples for f_1, f_2, f_3 showing that they are not AP.

shows that f_1 is not AP because we have $\mathbf{a} : \mathbf{b} :: \mathbf{c} : \mathbf{d}$ and the equation $f_1(\mathbf{a}) : f_1(\mathbf{b}) :: f(\mathbf{c}) : y$ is solvable, but the solution is 0 while $f(\mathbf{d})$ is 1. Similarly, $(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d})$ is also a counter-example for f_2 and $(\mathbf{d}, \mathbf{c}, \mathbf{b}, \mathbf{a})$ is a counter-example for f_3.

As f_1, f_2, f_3 are not AP, $f^{\mathbf{0}}_J$ cannot be AP either because it is equivalent to one of the f_i. As $f^{\mathbf{0}}_J$ is a section of f, f cannot be AP. □

All we need now is a property that could allow us to decrease the degree of a function without changing the fact that it is AP. This is the purpose of Property 6.4.

Property 6.4. *Let $f : \mathbb{B}^m \to \mathbb{B}$ be a function with $d(f) = k \geq 2$. Then there is a section g of f with $d(g) = k - 1$.*

Proof. Suppose that $d(f) = k \geq 2$, and let $\mathbf{x}_I$ be a monomial of f of maximum degree, i.e. $|I| = k$. Here again, consider the section $g = f^{\mathbf{0}}_J$ where $J = [1, m] \setminus I$ and $\mathbf{0}$ denotes the constant 0 vector in $\mathbb{B}^{|J|}$. It is clear that g is represented by a polynomial that has a unique monomial of maximal degree k, namely $\mathbf{x}_I$, and maybe some other monomials of degree strictly less than k. Let us choose any $i \in I$: then $g' = g^1_{\{i\}}$ is a section of g of degree (and arity) $k - 1$. As g' is a section of g, it is also a section of f which completes the proof. □

We are now in position to prove our main result.

Proposition 6.3. *The class of AP functions is the class L of affine functions.*

Proof. We have seen that every affine function is AP, i.e. if $d(f) \leq 1$, then $f \in AP$. On the other hand, Property 6.3 tells us that if $d(f) = 2$, then $f \notin AP$. So suppose that $d(f) \geq 3$. By successive applications of Property 6.4, it follows that there is a section g of f with $d(g) = 2$.

As g is not AP, then f is not AP either. All in all, if $d(f) \geq 2$ then f is not AP, so the class of AP functions is L. □

In the end, we have proved that **the class of functions that ensure a sound analogical extension of any training set $S \subseteq \mathbb{B}^m$ is the class of affine functions L**. If the function is not affine, then there exists a training set $S \subseteq \mathbb{B}^m$ for which $\mathbf{E}_S(f)$ is unsound. We can now finally give a definite answer to our initial problem: **a function f is *compatible* with the analogical inference principle if and only if f is affine.**

In the next subsection, we will focus on some AP functions that are still *compatible* with the analogical inference principle, but in a stronger sense.

6.2.3 The class of Strongly Analogy Preserving functions.

Let us come back to the analogical inference principle for a moment, which states that if we have $\mathbf{a} : \mathbf{b} :: \mathbf{c} : \mathbf{d}$, then we should also have $f(\mathbf{a}) : f(\mathbf{b}) :: f(\mathbf{c}) : f(\mathbf{d})$:

$$\frac{\mathbf{a} : \mathbf{b} :: \mathbf{c} : \mathbf{d}}{f(\mathbf{a}) : f(\mathbf{b}) :: f(\mathbf{c}) : f(\mathbf{d})}$$

Something we have overlooked so far but that is of importance, is that implicitly this inference principle requires that the class equation $f(\mathbf{a}) : f(\mathbf{b}) :: f(\mathbf{c}) : y$ is solvable as soon as we have $\mathbf{a} : \mathbf{b} :: \mathbf{c} : \mathbf{d}$. Indeed if the equation is not solvable, then we simply cannot have $f(\mathbf{a}) : f(\mathbf{b}) :: f(\mathbf{c}) : f(\mathbf{d})$. Therefore, a more explicit writing of the analogical inference principle is as follows:

Inference Principle A

$$\frac{\mathbf{a} : \mathbf{b} :: \mathbf{c} : \mathbf{d}}{\begin{array}{c}\text{solvable}(f(\mathbf{a}), f(\mathbf{b}), f(\mathbf{c})) \\ f(\mathbf{a}) : f(\mathbf{b}) :: f(\mathbf{c}) : f(\mathbf{d})\end{array}}$$

Yet, in the end, the ultimate use of this principle is to infer the value of $f(\mathbf{d})$, and we can only do that by the actual solving of the equation $f(\mathbf{a}) : f(\mathbf{b}) :: f(\mathbf{c}) : y$ (remember that when we build the analogical extension, we always require the class equation to be solvable). This was also the case when we were dealing with the algorithmic description of the conservative classifier (see e.g. Algorithm 1). So if the equation is not solvable, we are unable to infer anything about $f(\mathbf{d})$. This is why the AP functions are defined as the functions such that we have $f(\mathbf{a}) : f(\mathbf{b}) :: f(\mathbf{c}) : f(\mathbf{d})$ provided that $\mathbf{a} : \mathbf{b} :: \mathbf{c} : \mathbf{d}$ stands **and that** $f(\mathbf{a}) : f(\mathbf{b}) :: f(\mathbf{c}) : y$ is

solvable. With regard to this, we can say that the class of AP functions is the class of functions that allow the following inference principle to be sound:

$$\text{Inference principle B}$$
$$\frac{\mathbf{a} : \mathbf{b} :: \mathbf{c} : \mathbf{d} \qquad \text{solvable}(f(\mathbf{a}), f(\mathbf{b}), f(\mathbf{c}))}{f(\mathbf{a}) : f(\mathbf{b}) :: f(\mathbf{c}) : f(\mathbf{d})}$$

This one is the inference principle that is actually useful in practice: if the equation is not solvable then we simply try another 4-tuple. The first one is too restrictive on the function f because it requires that $\mathbf{a} : \mathbf{b} :: \mathbf{c} : \mathbf{d} \implies \text{solvable}(f(\mathbf{a}), f(\mathbf{b}), f(\mathbf{c}))$, and in practice this requirement is not needed.

We have shown that the AP functions are those that lead the inference principle B to be sound. For the sake of completeness however, we will also identify the functions that lead the inference principle A to be sound. These functions will be called Strongly Analogy Preserving functions (SAP):

Definition 6.3 (Strongly analogy preserving functions)**.** A function f is **Strongly Analogy Preserving** (SAP) if:

$$\forall \mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d} \in \mathbb{B}^m, \quad \mathbf{a} : \mathbf{b} :: \mathbf{c} : \mathbf{d} \implies \begin{cases} \text{solvable}(f(\mathbf{a}), f(\mathbf{b}), f(\mathbf{c})) \\ f(\mathbf{a}) : f(\mathbf{b}) :: f(\mathbf{c}) : f(\mathbf{d}). \end{cases}$$

We will show that the SAP functions are the dictator functions, i.e. functions of the form $f(\mathbf{x}) = x_i + \alpha_0$ for some $i \in [1, m]$. Naturally, these are a subclass of affine functions, because every SAP function is necessarily AP. In Property 6.5, we prove that every dictator function is SAP.

Property 6.5. *Let f be a dictator function, i.e. an affine function of the form:*

$$f(x_1, x_2, \cdots, x_m) = x_i + \alpha_0,$$

where $i \in [1, m]$ and $\alpha_0 \in \mathbb{B}$. Then f is SAP.

Proof. Let f be a dictator function: $f(\mathbf{x}) = x_i$ (we consider α_0 to be 0 for now). Consider $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d} \in \mathbb{B}^m$ such that $\mathbf{a} : \mathbf{b} :: \mathbf{c} : \mathbf{d}$.

As the analogy is defined component-wise, this means that we have in particular $a_i : b_i :: c_i : d_i$, which is strictly equivalent to $f(\mathbf{a}) : f(\mathbf{b}) :: f(\mathbf{c}) : f(\mathbf{d})$, so f is SAP. The case where $\alpha_0 = 1$ is derived in the same obvious way. □

It is also easy to show that the two constant functions are SAP. We now show that the dictators and the constant functions are the only SAP functions:

Proposition 6.4. *The class of SAP functions is the class of dictators and constant functions.*

Proof. We already know that a SAP function is an AP function, so a SAP function necessarily is affine. Consider a function f that is affine but that is neither a dictator function nor a constant, i.e. f is of the form:

$$f(\mathbf{x}) = \alpha_1 \cdot x_1 + \cdots + \alpha_m \cdot x_m + \alpha_0,$$

with at least $\alpha_i = 1$ and $\alpha_j = 1$ for two non-zero indices i and j with $i \neq j$. Without loss of generality, we will assume that $\alpha_0 = 0$. Consider now the 4-tuple $(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d})$ such that:

- $a_k = 0$ for all $k \neq i$ and $k \neq j$ $(a_i = a_j = 1)$
- $b_k = 0$ for all $k \neq i$ $(b_i = 1)$
- $c_k = 0$ for all $k \neq j$ $(c_j = 1)$
- $d_k = 0$ for all k.

The 4-tuple is summarized in Table 6.3, where we clearly see that we have $\mathbf{a} : \mathbf{b} :: \mathbf{c} : \mathbf{d}$, but the equation $f(\mathbf{a}) : f(\mathbf{b}) :: f(\mathbf{c}) : y$ is not solvable, so f is not SAP, QED.

	x_i	x_j	$f(\cdot)$
$\mathbf{a}$	1	1	0
$\mathbf{b}$	1	0	1
$\mathbf{c}$	0	1	1
$\mathbf{d}$	0	0	0

Table 6.3: f is not SAP because $f(\mathbf{a}) : f(\mathbf{b}) :: f(\mathbf{c}) : y$ is not solvable.

□

This section was quite dense, so let us recap a bit what we have seen so far. We have given two different instances of the analogical inference principle:

$$\frac{\begin{array}{c}\textsc{Inference Principle A}\\ \mathbf{a} : \mathbf{b} :: \mathbf{c} : \mathbf{d}\end{array}}{\begin{array}{c}\text{solvable}(f(\mathbf{a}), f(\mathbf{b}), f(\mathbf{c}))\\ f(\mathbf{a}) : f(\mathbf{b}) :: f(\mathbf{c}) : f(\mathbf{d})\end{array}}$$

and

$$\frac{\begin{array}{c}\textsc{Inference principle B}\\ \mathbf{a} : \mathbf{b} :: \mathbf{c} : \mathbf{d}\\ \text{solvable}(f(\mathbf{a}), f(\mathbf{b}), f(\mathbf{c}))\end{array}}{f(\mathbf{a}) : f(\mathbf{b}) :: f(\mathbf{c}) : f(\mathbf{d})}$$

It is obvious that the second version (B) is less restrictive on the function f, and it is the one that is useful in practice. In Section 6.2.2, we showed that the inference principle B is sound if and only if f is an affine Boolean function. We also showed in Section 6.2.3 that the inference principle A is sound for dictator functions (a particular case of affine functions).

These are strong theoretical results, but obviously real-world problems are not limited to the Boolean case, and the function f underlying the labels of a classification problem is almost never completely affine. We will address these issues in the next section.

6.3 Beyond Boolean Analogy Preserving functions

The aim of this section will be to complement the theoretical results of Section 6.2. We will first stay in the Boolean setting in Section 6.3.1 and empirically study the behaviour of functions that are *close* from being AP. In Sections 6.3.2 and 6.3.3, we will derive some more theoretical results about AP functions when dealing with nominal values and with real values.

6.3.1 Approximately AP functions and experiments

While the theoretical result of Proposition 6.3 is interesting on its own, it is obvious that purely affine functions are not representative of what would be encountered in a real-world Boolean environment. This leads to the following question: what remains of the quality of the analogical extension $\mathbf{E}_S(f)$ when f deviates from being AP in different ways? We will here investigate this question.

ε-close functions from L'

Recall that for a given training S, we defined ω_S^f as the quality of the extension $\mathbf{E}_S^f$ with $\omega_S^f \stackrel{\text{def}}{=} P\left(\overline{f}(\mathbf{x}) = f(\mathbf{x}) \mid \mathbf{x} \in \mathbf{E}_S^{f*}\right)$. By definition, $\omega_S^f = 1$ for all S iff f is AP. What we want to know now is how ω_S^f is affected when f is not a purely affine function. What we mean by *not purely affine* is that f is at a distance at most ε from the set of affine functions. We will define these notions now.

Given two Boolean functions f and g, we define their distance $\text{dist}(f,g) \stackrel{\text{def}}{=} P\left(f(\mathbf{x}) \neq g(\mathbf{x})\right)$, where P is the uniform distribution over $\mathbb{B}^m$. Here, $\mathbf{x} \in \mathbb{B}^m$ is also considered as a random variable.

Definition 6.4 (ε-close functions)**.** We say that f is ε-close to g if $\text{dist}(f,g) \leq \varepsilon$, and that f is ε-close to a set of functions Σ if $\exists g \in \Sigma$ such that f is ε-close to g.

We want to study here the value of ω_S^f when f is ε-close from L, the set of affine functions. Note however that thanks to the code-independency property of the Boolean proportion (1 and 0 play symmetric roles), we have that for any S and any f, $\omega_S^f = \omega_S^{\neg f}$. As the set of affine

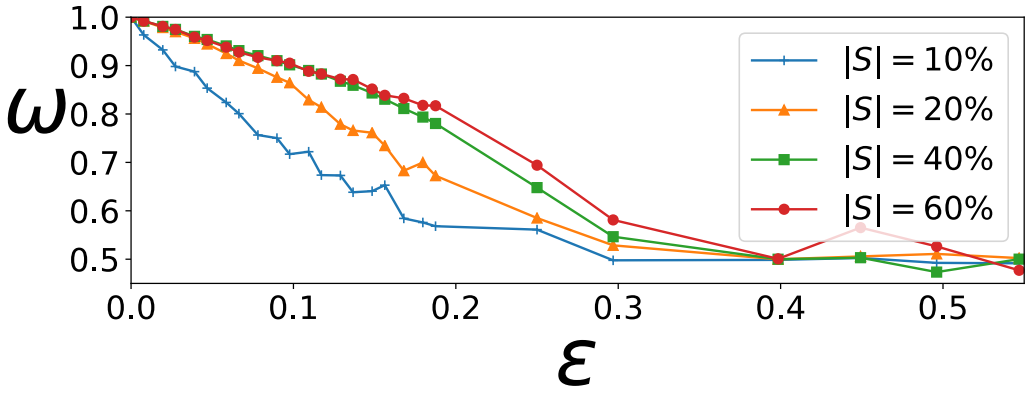

Figure 6.3: ω for ε-close functions to L.

functions is the set of linear functions and their negations, **it is enough to study the value of ω_S^f when f is ε-close to L', the set of linear functions**.

This question is clearly of interest, because in practice there exists some statistical test allowing the practitioner to query a partially-known function f to find out if it is ε-close to L'. This test is the BLR test [BLR93], whose steps are as follows:

- Choose randomly and independently $\mathbf{x}$ and $\mathbf{y}$ in $\mathbb{B}^m$.
- Compute $f(\mathbf{x}+\mathbf{y})$ and $f(\mathbf{x})+f(\mathbf{y})$.
- If $f(\mathbf{x}+\mathbf{y}) = f(\mathbf{x})+f(\mathbf{y})$, return *Accept*.

The main useful result is given in Property 6.6:

Property 6.6. *If BLR accepts f with a probability at least $1-\varepsilon$, then f is ε-close to the set of linear functions L'.*

So if we can exhibit a clear probabilistic dependence between ω_S^f and ε, this would allow the practitioner to have guarantees about ω_S^f because we can know ε from the BLR test.

Let us first consider the following experiment. Starting from a linear function $g\colon \mathbb{B}^8 \to \mathbb{B}$ defined as $g(\mathbf{x}) = x_1 + \cdots + x_8$, we introduce some noise by negating its output $g(\mathbf{x})$ with probability ε. We thus obtain some functions f_ε that are ε-close to L', and report the value of $\omega_S^{f_\varepsilon}$ averaged over 50 experiments. Figure 6.3 gives an illustration of the variation of $\omega_S^{f_\varepsilon}$ (simply denoted ω) against ε for different sizes of S (as a percentage of $|\mathbb{B}^m|$). Other experiments have been carried out with other affine functions (i.e. with less essential variables or different arity), leading to very similar results. As a side note, ε only needs to be taken in $[0, \frac{1}{2}]$, because $f_\varepsilon = \neg f_{1-\varepsilon}$ and for any f, we have $\omega_S^f = \omega_S^{\neg f}$.

When $\varepsilon = 0$, i.e. when f_ε is AP, we get $\omega = 1$ as expected from Proposition 6.3. We observe an almost linear decrease in ω as ε grows to 0.3 then leading to a plateau where $\omega = \frac{1}{2}$, indicating that the analogical labels $\overline{f}(\mathbf{x})$ are more or less random. Moreover, ω appears to

	ω	β_S
Monk 1	.96	.73
Monk 2	.96	.69
Monk 3	.98	.87

Table 6.4: ω and β over the Monk's problems.

decrease faster for small training set S. This is due to the fact that the analogical labels $\overline{f}(\mathbf{x})$ are the result of a majority-vote procedure among the candidate solutions that one can build from S, and the number of candidates becomes smaller as $\mid S \mid$ decreases, thus altering the quality of the prediction.

We only have an empirical study here. Ultimately, it would be very interesting to prove a result like

$$P(\omega_S^f \geq \eta) > 1 - \delta,$$

where $\eta \in [0, 1]$ and $\delta \in [0, \frac{1}{2}]$ are parameters depending on ε and $\mid S \mid$. Such a result would be strongly linked to the PAC learning theory [Val84], and is the object of current research.

Other non-AP functions

We have studied so far the variation of ω_S^f when f deviates from the set of affine functions. We now focus on another point: the variation of ω_S^f when f deviates from being AP in a different way. Let us note the following point: even though a function f is far from being AP, the quality ω_S^f of the extension $\mathbf{E}_S^f$ may still be very high. To illustrate this, let us define the value β which is an indicator of how far is f from being completely AP. For each $\mathbf{x} \in \mathbf{E}_S^{f*}$, we define $\beta_{\mathbf{x}}$ as the proportion of candidates $\mathrm{sol}(f(\mathbf{a}), f(\mathbf{b}), f(\mathbf{c}))$ (with $(\mathbf{a}, \mathbf{b}, \mathbf{c}) \in \mathbf{R}_S^f(\mathbf{x})$) that led to the correct label, i.e. the proportion of candidate solutions that are equal to $f(\mathbf{x})$. β is defined as the average of all the $\beta_{\mathbf{x}}$. Obviously, a function f is AP iff $\beta = 1$ for all S, i.e. iff $\beta_{\mathbf{x}} = 1$ for all $\mathbf{x} \in \mathbf{E}_S^{f*}$ and for all S.

Table 6.4 reports the values of ω and β over three datasets from the UCI repository, namely the three Monk's problems[2] [Lic13]. Results are averaged over 100 experiments, where the training set S is each time randomly sampled with a size of 30% of the universe of possible instances.

We observe that for each dataset, β is significantly lower than 1. This suggests that the Boolean functions underlying these datasets are highly not AP, because on average, there is a high proportion (around 20%) of candidates that predicted the wrong label. However, ω is not lower than 96%, implying extensions of very high quality. **This is where the majority-vote comes into play**: in some cases, it may be able to compensate for the predictors that were

[2]As these datasets are nominally-valued, they have been binarized.

wrong. This is what happens here in 96%, 96% and 98% of the cases respectively. Here again, obtaining theoretical guarantees about the majority-vote procedure is currently investigated.

In the next subsection, we will extend our theoretical results about Boolean AP functions to a more general setting where instances are nominally valued.

6.3.2 Extension to nominal attributes

When it comes to real life applications, it is quite rare to get a purely Boolean dataset. It is generally the case that the dataset is a mix between Boolean attributes and nominal attributes taking their values in finite sets. For instance, an attribute *color* typically takes its values in, let us say, $\{ red, green, blue \}$. Unfortunately, on this set of values for attribute *color*, there is no structure, no order and no known operator. Nevertheless, there is still a way to deal with such attributes within the analogical framework, and to characterize AP functions in such settings. This is what we will investigate in this subsection.

We have already seen in Chapter 2 that we can define an analogical proportion on a set X by restricting the accepted analogical patterns to $a : b :: a : b$ and $a : a :: b : b$. Formally, an analogy A over X is a subset of X^4 defined as:

$$A \stackrel{\text{def}}{=} \{ (a, a, b, b) \mid a, b \in X \} \cup \{ (a, b, a, b) \mid a, b \in X \} .$$

Back to the *color* attribute, it means that we consider $red : blue :: red : blue$ as a valid analogical proportion but $red : blue :: red : green$ is not valid. We can easily extend componentwise this analogy to a Cartesian product $X^m = \prod_{i=1}^{m} X_i$ which is now equipped with an analogical relation. In classification, this allows to deal with any kind of nominal attributes. In this setting, a binary classification function f is a function from $X^m = \prod_{i=1}^{m} X_i$ to $\mathbb{B}$. The question we want to address here is the same as that of Section 6.2: we want to characterize all the functions f that lead to a perfectly sound analogical extension. Unfortunately, we cannot turn to the previous framework because the Cartesian product X^m is made of distinct sets and is not a power of $\mathbb{B}$ anymore. So we will not be able to entirely identify these functions, simply because we do not even know how to *write* them. When dealing with Boolean functions we used an algebraic point of view that allowed us to write the functions in their Algebraic Normal Form, and we identified the AP functions as those that have a particular ANF. But this will not be possible here, because there is no algebraic structure on the Cartesian product that is the domain of our functions.

Nonetheless, we will still find close links between the AP Boolean functions and the AP functions of our current setting, and these links are based on the *binarization* of the nominal attributes. Indeed, any nominal attribute with k values can be binarized into k Boolean attributes. We can also choose $k-1$ attributes, or some other fancy binarization procedure, but we will stick to the simplest one: binarizing the *color* attribute leads to three binary attributes

$(is\_red, is\_green, is\_blue)$ so that red is coded as $(1,0,0)$, $green$ is coded as $(0,1,0)$ and $blue$ is coded as $(0,0,1)$. Obviously, the k Boolean attributes are not independent: for example $(1,1,1)$ is not a valid value. More formally, let us denote bin this injective embedding going from $X_i = \{v_1, \ldots, v_k\}$ to $\mathbb{B}^k$:

$$\text{bin}(v_i) \stackrel{\text{def}}{=} (0, \ldots, 0, 1, 0, \ldots 0),$$

where 1 is at position i. So if we have a set of nominal attributes $x_1, \ldots, x_m$ where x_i takes its values in a finite set X_i, the Cartesian product $\prod_{i=1}^m X_i$ can be mapped into the set $\prod_{i=1}^m \mathbb{B}^{|X_i|}$ by applying the injective embedding bin in a component-wise fashion:

$$\text{bin}(\prod_{i=1}^{m} X_i) \stackrel{\text{def}}{=} \prod_{i=1}^{m} \text{bin}(X_i).$$

From the very definition of this embedding, we have the following immediate and obvious property:

Property 6.7. *For any* $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d} \in X^m = \prod_{i=1}^m X_i$,

$$\mathbf{a} : \mathbf{b} :: \mathbf{c} : \mathbf{d} \iff bin(\mathbf{a}) : bin(\mathbf{b}) :: bin(\mathbf{c}) : bin(\mathbf{d}).$$

Note that the proportion $\mathbf{a} : \mathbf{b} :: \mathbf{c} : \mathbf{d}$ holds in X^m while the proportion $\text{bin}(\mathbf{a}) : \text{bin}(\mathbf{b}) :: \text{bin}(\mathbf{c}) : \text{bin}(\mathbf{d})$ is a usual Boolean proportion that holds in $\prod_{i=1}^m \mathbb{B}^{|X_i|}$. Now, every function f from $\prod_{i=1}^n X_i$ to $\mathbb{B}$ can be associated to a **partial** Boolean function f^b from $\prod_{i=1}^m \mathbb{B}^{|X_i|}$ to $\mathbb{B}$ where f^b is defined as follows:

$$f^b(\text{bin}(\mathbf{x})) \stackrel{\text{def}}{=} f(\mathbf{x}).$$

As the initial nominal universe $\prod_{i=1}^m X_i$ has been compiled into a Boolean universe, it is tempting to consider the framework that has been developed in Section 6.2. Unfortunately, the whole set $\prod_{i=1}^m \mathbb{B}^{|X_i|}$ is not the exact image of $\prod_{i=1}^m X_i$: it is bigger, and some elements in $\prod_{i=1}^m \mathbb{B}^{|X_i|}$ are not the image of an element of $\prod_{i=1}^m X_i$ using bin. Ultimately, the domain of f^b is just $\text{bin}(\prod_{i=1}^m X_i)$, which is included in $\prod_{i=1}^m \mathbb{B}^{|X_i|}$. This is why f^b is considered to be a partial function.

Nevertheless, the AP property is still relevant for functions that are only partially defined: a partially defined function that always leads to a perfectly sound analogical extension for any training set S will also be said to be Analogy Preserving, and will be called PAP for Partial AP. Just like in Section 6.2, a natural way to characterize PAP functions is given in Definition 6.5:

Definition 6.5 (Partial analogy preserving functions)**.** A **partial** function f^b is **Analogy**

Preserving (PAP) if for all $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d} \in \mathrm{dom}(f^b)$:

$$\begin{cases} \mathbf{a} : \mathbf{b} :: \mathbf{c} : \mathbf{d} \text{ and} \\ \mathit{solvable}(f^b(\mathbf{a}), f^b(\mathbf{b}), f^b(\mathbf{c})) \end{cases} \implies \mathrm{sol}(f^b(\mathbf{a}), f^b(\mathbf{b}), f^b(\mathbf{c})) = f^b(\mathbf{d}).$$

Proposition 6.5 will provide us with the link between f and f^b, when one of them is AP:

Proposition 6.5. *Let f be a function from $\prod_{i=1}^{n} X_i$ to $\mathbb{B}$. Then f is AP if and only if f^b is PAP.*

Proof. Let us assume that f is AP and let be 4 elements belonging to the domain of f^b: $\mathrm{bin}(\mathbf{a}), \mathrm{bin}(\mathbf{b}), \mathrm{bin}(\mathbf{c}), \mathrm{bin}(\mathbf{d}) \in \mathrm{bin}(\prod_{i=1}^{m} X_i)$ such that:

$$\begin{cases} \mathrm{bin}(\mathbf{a}) : \mathrm{bin}(\mathbf{b}) :: \mathrm{bin}(\mathbf{c}) : \mathrm{bin}(\mathbf{d}) \text{ and} \\ \mathrm{solvable}(f^b(\mathrm{bin}(\mathbf{a})), f^b(\mathrm{bin}(\mathbf{b})), f^b(\mathrm{bin}(\mathbf{c}))). \end{cases}$$

Thanks to Property 6.7, we have seen that in this case we also have $\mathbf{a} : \mathbf{b} :: \mathbf{c} : \mathbf{d}$. Since by definition $f(\mathbf{x}) = f^b(\mathrm{bin}(\mathbf{x}))$, the second condition simply is $\mathrm{solvable}(f(\mathbf{a}), f(\mathbf{b}), f(\mathbf{c}))$. As f is AP, $\mathrm{sol}(f(\mathbf{a}), f(\mathbf{b}), f(\mathbf{c})) = f(\mathbf{d})$, and $f(\mathbf{d}) = f^b(\mathrm{bin}(\mathbf{d}))$. So in the end, we have that $\mathrm{sol}((f^b(\mathbf{a}), f^b(\mathbf{b}), f^b(\mathbf{c})) = f^b(\mathrm{bin}(\mathbf{d}))$, which proves that f^b is PAP.

Let us now focus on the reverse implication. Consider a PAP function f^b and 4 elements $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d} \in \prod_{i=1}^{m} X_i$ such that:

$$\begin{cases} \mathbf{a} : \mathbf{b} :: \mathbf{c} : \mathbf{d} \text{ and} \\ \mathrm{solvable}(f(\mathbf{a}), f(\mathbf{b}), f(\mathbf{c})). \end{cases}$$

Thanks again to Property 6.7, $\mathrm{bin}(\mathbf{a}) : \mathrm{bin}(\mathbf{b}) :: \mathrm{bin}(\mathbf{c}) : \mathrm{bin}(\mathbf{d})$ holds. The second condition is just $\mathrm{solvable}(f^b(\mathrm{bin}(\mathbf{a})), f^b(\mathrm{bin}(\mathbf{b})), f^b(\mathrm{bin}(\mathbf{c})))$ because by definition $f(\mathbf{x}) = f^b(\mathrm{bin}(\mathbf{x}))$. Because f^b is PAP, we have that $\mathrm{sol}(f^b(\mathrm{bin}(\mathbf{a})), f^b(\mathrm{bin}(\mathbf{b})), f^b(\mathrm{bin}(\mathbf{c}))) = f^b(\mathrm{bin}(\mathbf{d}))$. As $f^b(\mathrm{bin}(\mathbf{d})) = f(\mathbf{d})$ by definition, we have that f is AP. □

In the end, the notion of AP function is smoothly transferable to the notion of PAP function. This result is only purely theoretical, and sadly there is not much more we can extract from it for practical use. Proposition 6.5 tells us that when dealing with nominal attributes, if the function underlying the ground truth labels is affine once the dataset has been binarized, then we can safely extend our training set. As there are no common operators on the set $\prod_{i=1}^{n} X_i$, we are not able to give a more accurate characterization of AP functions in such settings.

In the next subsection, however, we will see that in a real setting the AP functions are also precisely the affine (real) functions.

6.3.3 Extension to real-valued functions

Section 6.2 was devoted to the identification of Boolean functions that were fully compatible with the analogical inference principle, and we saw that these functions are the Boolean affine functions. One may wonder what would these functions be in a real setting, i.e. using functions from $\mathbb{R}^m$ to $\mathbb{R}$ with the arithmetic proportion?

We already know that the Boolean proportion and the arithmetic proportion are really close. Once again, we will witness this close bond: in a real setting, the affine functions are also AP. Luckily, the proof is a lot easier to derive. Let us first (re)define the arithmetic proportion:

Property 6.8. *Four elements* $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d} \in \mathbb{R}^m$ *are in arithmetic proportion if* $\mathbf{d} = \mathbf{c} - \mathbf{a} + \mathbf{b}$.

Also, for any $a, b, c \in \mathbb{R}$*, the equation* $a : b :: c : y$ *is always solvable and the solution is* $\mathrm{sol}(a, b, c) = c - a + b$.

Please note that here, $+$ is the regular addition operator (not the modulo-2 addition as in the previous sections).

Definition 6.6. Real affine functions A real function f from $\mathbb{R}^m$ to $\mathbb{R}$ is affine if f is of the form:

$$f(\mathbf{x}) = \boldsymbol{\alpha} \cdot \mathbf{x} + \alpha_0,$$

where $\boldsymbol{\alpha} \in \mathbb{R}^m$, $\alpha_0 = f(\mathbf{0}) \in \mathbb{R}$, and $\cdot$ is here the usual dot product.

Proposition 6.6. *All real affine functions are AP (w.r.t. the the arithmetic proportion)*

Proof. A real function f is AP iff for all $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d} \in \mathbb{R}^m$,

$$\mathbf{a} : \mathbf{b} :: \mathbf{c} : \mathbf{d} \implies \mathrm{sol}(f(\mathbf{a}), f(\mathbf{b}), f(\mathbf{c})) = f(\mathbf{d}).$$

Equivalently, thanks to property 6.8, f is AP iff:

$$\mathbf{d} = \mathbf{c} - \mathbf{a} + \mathbf{b} \implies f(\mathbf{c}) - f(\mathbf{a}) + f(\mathbf{b}) = f(\mathbf{c} - \mathbf{a} + \mathbf{b}).$$

Let f be an affine function. $\forall \mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbb{R}^m$,

$$\begin{aligned} f(\mathbf{c} - \mathbf{a} + \mathbf{b}) &= \boldsymbol{\alpha} \cdot (\mathbf{c} - \mathbf{a} + \mathbf{b}) + \alpha_0 \\ &= \boldsymbol{\alpha} \cdot \mathbf{c} - \boldsymbol{\alpha} \cdot \mathbf{a} + \boldsymbol{\alpha} \cdot \mathbf{b} + \alpha_0 \\ &= f(\mathbf{c}) - f(\mathbf{a}) + f(\mathbf{b}), \end{aligned}$$

which means that f is AP. □

Proposition 6.6 is illustrated in Figure 6.4.

In contrast with the Boolean setting, the equivalence between AP functions and affine functions does not hold in the real case. It is indeed possible to find functions $f : \mathbb{R} \to \mathbb{R}$

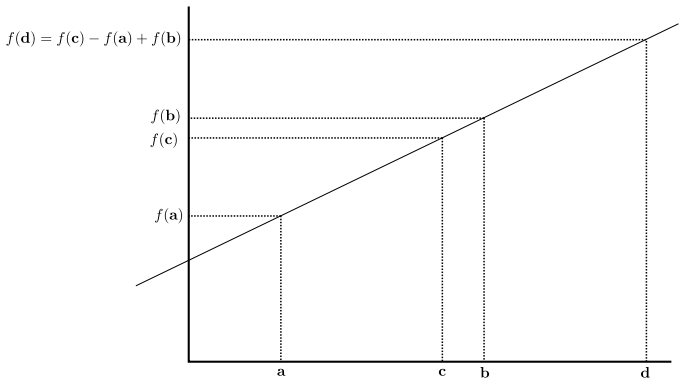

Figure 6.4: When f is affine, $\text{sol}(f(\mathbf{a}), f(\mathbf{b}), f(\mathbf{c})) = f(\mathbf{d})$.

satisfying $f(c-a+b) = f(c) - f(a) + f(b)$ that are not affine. Using a result from Darboux [RRA09] about Cauchy's functional equation[3], we can show that these functions are highly pathological: they are for instance discontinuous at every point, and clearly they would never be encountered in a regression problem. To our knowledge, no such result has been proved for functions $f : \mathbb{R}^m \to \mathbb{R}$ $(m > 1)$, but we may still conjecture that AP functions that are not affine would be quite pathological.

In any case, let us note that the fact that real affine functions are AP is not really significant: when the function is (supposed to be) affine, there are much more efficient techniques such as least-squares regression. Using analogical learning in such settings would not be reasonable. This property is mainly interesting because it stresses once again the close links between the arithmetic and Boolean proportion.

Conclusion

The main contribution of this chapter was to provide a complete characterization of Boolean functions that are fully compatible with the analogical inference principle that we have used so far in all the previous chapters of this document. Using an appropriate representation for Boolean functions (namely their algebraic normal form), we have been able to prove that the class of AP functions is the class of affine functions, i.e. functions whose degree is less than 1.

We have chosen to tackle the problem of finding the class of AP functions from the angle of training set extension. It became clear that the class of AP functions can be seen as the class of functions which allow a perfectly sound extension, or equivalently as the class of functions which allow the analogical inference principle to be sound.

We also provided a more explicit version of the analogical inference principle which is in fact less restrictive than the one we had claimed so far. The distinction between the strict version and the relaxed version is that the class equation $f(\mathbf{a}) : f(\mathbf{b}) :: f(\mathbf{c}) : y$ is not always

[3]Cauchy's functional equation is: $f(x+y) = f(x) + f(y)$ with $x, y \in \mathbb{R}$

solvable, and the relaxed version takes this fact into account. The class of AP functions is fully compatible with the relaxed version, but we have also identified the functions that are compatible with the strict version, which turned out to be the class of projection (or dictator) functions.

In real-world application though, purely affine functions are extremely rare. This means that using the analogical inference principle will necessarily produce errors in the label predictions. We have empirically investigated the quality of the analogical extension (which measures in some sense how *adequate* was the use of the analogical inference principle) when a function deviates from the class of purely affine functions. We have observed that the quality of the extension linearly decreases, but a strong statistical result still needs to be derived. We also noted that in some particular cases, a function that is strongly not AP can still lead to a high extension quality. Indeed, as the analogical labels are the result of a majority-vote procedure, it may happen that the most common candidate label actually corresponds to the ground truth label $f(\mathbf{x})$. In this case, the majority-vote procedure is able to compensate for the candidates that were wrong in their predictions, because it will choose the most common predictor that happens to be the correct label. Ultimately, even if only $50\% + \varepsilon$ ($\varepsilon > 0$) of the candidates are correct, it is still enough to have a perfectly sound extension. The class of AP functions ensures that exactly 100% of the candidates are correct, but it is clear that such a requirement is too strong to be really useful in practice. The derivation of a theoretical result involving the role of the majority-vote procedure is still a topic of current research.

Our main goal was to identify the class of AP functions in a Boolean setting. We have extended our results to two other domains. It is often the case that the classification problem does not involve purely Boolean attributes, but rather a more general domain where instances are multi-valued. Sadly in these domains, there is no known structure or operators, so it was not possible to give a clear identification of AP functions such as that of Boolean AP functions. We were however able to provide a clear link between these AP functions and the Boolean AP functions by embedding the multi-valued space into a partially defined Boolean space. We also proved that in a real setting, affine functions are also AP (the reverse implication does not necessarily hold). This result for real functions was much easier to prove and in the end quite trivial, but it should be noted that proportion-based learning in regression problems is not very reasonable.

Résumé du chapitre Nous avons introduit le principe d'inférence analogique au chapitre 2. Ce principe stipule que si quatre éléments sont en proportion, alors cela doit aussi être vrai pour leurs classes (ou n'importe quel autre attribut). C'est évidemment un principe non-correct, puisque la conclusion ne se déduit pas logiquement de la prémisse. Néanmoins, on a pu observer au chapitre 3, ainsi que dans [BMD07], que les classifieurs analogiques semblaient afficher des résultats prometteurs (au moins sur certains problèmes). Le but de ce chapitre est d'identifier clairement et de caractériser les types de problèmes sur lesquels le principe d'inférence analogique

est correct. En d'autres termes, nous proposons de caractériser les fonctions booléennes f telles que l'inférence suivante soit correcte :

$$\frac{\mathbf{a} : \mathbf{b} :: \mathbf{c} : \mathbf{d}}{f(\mathbf{a}) : f(\mathbf{b}) :: f(\mathbf{c}) : f(\mathbf{d})}$$

Il est en effet assez facile de construire des problèmes simples où l'application du principe d'inférence analogique mène a des résultats désastreux. Considérons par exemple celui de la figure 6.1 à la page 135 : l'ensemble d'apprentissage S (à gauche) est linéairement séparable par une droite verticale. Cependant, lorsqu'il est étendu pour former son extension analogique (à droite), on constate que l'espace entier est criblé de points appartenant à l'une ou l'autre des deux classes. Utiliser cette extension analogique comme ensemble d'apprentissage mènera nécessairement à de très mauvais résultats.

Les fonctions f qui sont compatibles avec le principe d'inférence analogique sont les fonctions dites *qui préservent l'analogie* (fonctions AP), c'est-à-dire des fonctions qui satisfont $f(\mathbf{a}) : f(\mathbf{b}) :: f(\mathbf{c}) : f(\mathbf{d})$ dès lors que $\mathbf{a} : \mathbf{b} :: \mathbf{c} : \mathbf{d}$ est vrai (en fait, une définition moins restrictive et plus utile en pratique consiste à ajouter la prémisse selon laquelle l'équation $f(\mathbf{a}) : f(\mathbf{b}) :: f(\mathbf{c}) : y$ est soluble).

Dans un contexte booléen, nous avons prouvé que les fonctions AP sont les fonctions affines. Notre preuve repose sur l'utilisation d'outils algébriques tels que la forme normale algébrique des fonctions booléennes. Toute fonction booléenne peut être exprimée sous la forme d'un polynôme, et notre preuve consiste d'abord à montrer que les polynômes de degré 1 (c'est-à-dire les fonctions affines) sont AP, puis à montrer que tout polynôme de degré supérieur ou égal à 2 n'est pas AP.

Évidemment, dans des problèmes pratiques, les fonctions f rencontrées sont très variées et ne se cantonnent pas à l'ensemble des fonctions affines. Aussi, nous avons expérimentalement étudié la qualité de l'extension analogique lorsque la fonction f s'éloigne de l'ensemble des fonctions AP de différentes manières. D'abord, nous avons observé les variations de la qualité de l'extension en fonction de la distance ε de f à l'ensemble des fonctions affines. Ensuite, nous avons observé que même lorsqu'une fonction n'est vraiment pas AP (car une partie des prédicteurs se trompent régulièrement), il est toujours possible d'obtenir une très bonne qualité de l'extension analogique grâce à la procédure de vote majoritaire qui permet de compenser les erreurs des mauvais prédicteurs.

Enfin, nous avons partiellement étendu nos résultats au cas où la fonction f est une fonction réelle, et au cas où les attributs ne sont plus booléens mais plutôt nominaux.

Conclusion

It is now time to conclude this document, whose purpose was twofold: exhibit clear theoretical properties of analogical classifiers, and apply analogical inference to the field of recommender systems. Let us first recap our main results and contributions (we will not necessarily follow the order of the chapters here).

Contributions

The first chapter was dedicated to give the necessary background on existing models of analogical reasoning, with a strong emphasis on models that allow to build computer programs. In the second chapter, we thoroughly described various definitions of analogical proportions in different settings, with a particular focus on the arithmetic and the Boolean proportions. We have tried to provide the reader with different geometrical insights on these proportions, which were hopefully useful to gain a better intuition. Finally, we considered a toy classification problem in a Boolean setting that allowed us to introduce the analogical equation solving process and most importantly, our analogical inference principle. This principle states that if we have $a : b :: c : d$, then we should also have $f(a) : f(b) :: f(c) : f(d)$, where $f(x)$ is the label of x, or any characteristic of interest. When $f(d)$ is unknown, it can be **inferred** from the values of $f(a)$, $f(b)$ and $f(c)$, allowing us to apply this form of analogical reasoning to classification tasks, or more general problems such as that of rating prediction for recommendation tasks.

Analogical recommendation was addressed in Chapters 4 and 5. Chapter 4 was a background chapter, where we clearly defined the problem we planned to address, and described two popular collaborative filtering families (neighborhood methods and matrix-factorization techniques) that will be compared to our custom algorithms in the experiments.

We then developed in Chapter 5 an algorithm for rating prediction [HPR15], on the basis that if four users are in proportion (i.e. their respective ratings are in arithmetic proportion), then it should also be the case for any item that one of the four users has not rated. This approach is directly inspired from past works on analogical classification. The experiments showed that this algorithm offers reasonable performance compared to neighborhood techniques, but its cubic complexity makes it simply impossible to use in real-world scenarios, due to enormous computation time.

This led us to consider another kind of analogical recommendation, that does not rely on

the search of 3-tuples of users. This "clone"-based view [HPRS16b] is motivated by the fact that some users may have different interpretations of the rating scale that is used. Our results show that the concept of "clone" is extremely relevant, but it must be noted that this kind of bias in the user ratings had already been addressed in previous works. As a side note, these works on the rating prediction task have led us to build a Python library [Hug17] that helps building and analyzing such prediction algorithms.

Finally, we provided an algorithm for the mining of analogical proportions in incomplete databases [HPRS16c], strongly inspired from the mining of association rules. We used this algorithm to extract analogical proportions between items in a rating database, which actually corresponds to the database that we had been using for our recommendation experiments. Our results showed that the analogies we found were in fact rather uninteresting, in that they related four items that were either equally appreciated, or equally disliked. There was no discrepancy in the ratings. In a way, this fact can retrospectively explain the modest results of our first analogical recommendation algorithm, and its performances that were close to those of neighborhood methods.

In Chapter 3, we described our contributions to the field of analogical classification [HPRS16a]. Our first key contribution was to provide a unifying functional definition of analogical classifiers, which were yet only known from their algorithmic descriptions. From this definition, we were able to derive various theoretical properties. In particular, we showed that the VC-dimension of analogical classifiers is infinite, and that their error rate is closely related to that of the k-NN algorithms, as can be seen on the analytical formula that we derived. In fact, our functional definition establishes clear links between analogical classification and nearest-neighbors classification. We have showed indeed that analogical classification can be viewed as a two-steps procedure: first the training set is extended by analogy, where new examples are generated and assigned a potentially noisy label called the analogical label. Then, using this extended training set, all the remaining elements can be labeled using the classical k-NN procedure. This new point of view is quite interesting, because it clearly binds together the two processes triggered by analogical proportions (inference and creativity) as the two sides of the same coin.

In Chapter 6, we investigated a question that naturally followed from the results of Chapter 3: how can we ensure that the analogical extension is completely error-free? In other words, is there a criterion that tells us that the new examples that are generated are associated to the correct label? We have been able to answer this question in a Boolean setting: the analogical extension is perfectly sound if and (only if) the Boolean function f underlying the label is an affine function. This strong result [CHPR17] is reminiscent of that of Davies and Russel (see Section 1.1.3), who provided a side condition that allowed to safely use an analogical inference principle. Their analogical inference principle was actually quite different from ours

(although we could claim that ours is a particular case of theirs), but the two approaches can be considered similar from a general point of view. We extended our results to the real setting and in the case where attributes are nominally-valued. These results open the door to various speculative research topics, that we will now mention.

Future work

In Section 6.3.1, we presented the concept of approximate AP functions. Indeed in practice, there is no way to know for sure if the function f that we want to learn is completely affine. So a clear topic of interest is to obtain statistical guarantees about the quality of the analogical extension, depending on how far the function f is from the set of affine functions. As previously mentioned, a result of the following form would be extremely interesting:

$$P(\omega_S^f \geq \eta) > 1 - \delta,$$

where $\eta \in [0, 1]$ and $\delta \in [0, \frac{1}{2}]$ are functions depending on ε and $\mid S \mid$. The value ε tells us that f is ε-close from the set of affine functions. We are currently investigating a somewhat simpler question:

$$P(\omega_S^f < 1) < \delta,$$

where δ is still a function of ε. In other words, we want to have an upper bound of the probability of the event "$\mathbf{E}_S^{f*}$ *is unsound*".

We have also seen during our experimentations that even though some functions are highly not AP, the quality of the analogical extension may still be very high. This is because of the majority-vote procedure that is applied when we compute the analogical label of the elements. This aspect has been left out of the discussion so far, but it is clear that it is a decisive step of our inference process, and it has to be thoroughly studied to fully understand the analogical classification process. We may for example suppose that some 3-tuples should be given a higher weight in the aggregation, on the basis of some particular static criterion that still needs to be identified.

Another track of research can be motivated by looking at Figure 6.5. The function f is not affine, but is instead **piecewise affine**. We know from our results that for any elements $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d} \in X^m$ that are on the same simplex (a simplex is here defined as a subset of X^m where f is affine), then we can correctly predict the value of $f(\mathbf{d})$ from those of $f(\mathbf{a}), f(\mathbf{b}), f(\mathbf{c})$. Well in $\mathbb{R}^m$ this result is not really useful, but we can actually show that every Boolean function is piecewise affine! So theoretically, if we can identify all the simplices, this means that we should be able to produce a sound extension in any case, for any function f, and for any training set. At this point, only preliminary experiments have been carried out, and a more thorough

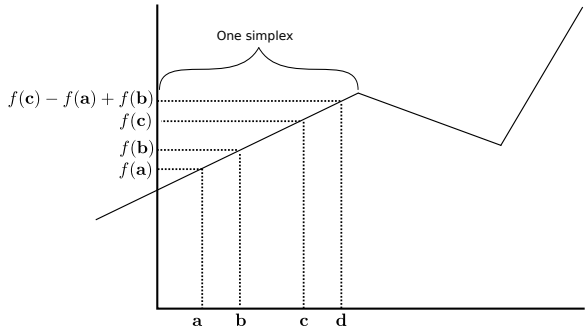

Figure 6.5: A real function f that is piecewise affine.

investigation deserves to be performed.

Finally, we will close this discussion by opening to transfer learning. Transfer learning is a current trend in the machine learning community, whose purpose is to use the knowledge of a source problem S to apply it to a less known target problem T. Undoubtedly, transferring knowledge is the essence of analogy. Current techniques in transfer learning usually involve statistical tools (see e.g. [PY10] for a survey), and the use of analogical proportions for such purposes is still entirely to be explored.

Conclusion in French

Les deux principaux objectifs de cette thèse étaient les suivants : exhiber des propriétés théoriques des classifieurs analogiques, et appliquer le raisonnement analogique au problème de la recommandation. Commençons par rappeler succinctement les principaux résultats et contributions apportées dans ce document (on ne suivra pas nécessairement l'ordre des chapitres).

Contributions

Le premier chapitre était dévoué aux prérequis nécessaires sur les modèles de raisonnement analogique existants, en accordant une attention particulière aux modèles qui permettent l'implémentation de programmes informatiques. Dans le deuxième chapitre, nous avons décrit en détail différentes définitions de proportions analogiques, en insistant particulièrement sur les proportions booléennes et arithmétiques. Nous avons essayé d'apporter au lecteur quelques intuitions géométriques sur les propriétés de ces proportions. Enfin, nous avons considéré un problème de classification booléenne basique, qui nous a permis d'introduire le processus de résolution d'équation analogique, ainsi que le principe d'inférence analogique. Ce principe stipule que si l'on a $a : b :: c : d$, alors on devrait aussi avoir $f(a) : f(b) :: f(c) : f(d)$, où $f(x)$ détermine la classe de x. Lorsque $f(d)$ est inconnu, on peut l'inférer à partir des valeurs de $f(a), f(b)$ et $f(c)$, ce qui nous permet d'appliquer ce type d'inférence à des tâches de classification ou des problèmes plus généraux encore comme celui de la prédiction de notes.

La recommandation analogique fut l'objet des chapitres 4 et 5. Le chapitre 4 était un chapitre de prérequis, où nous avons clairement défini le problème de la prédiction de notes, et décrit deux familles d'algorithmes (les méthodes par voisinage et par factorisation de matrice) qui nous servent de base pour comparer nos propres algorithmes.

Nous avons ensuite développé dans le chapitre 5 un premier algorithme pour la prédiction de notes [HPR15], fondé sur l'idée que si quatre utilisateurs sont en proportion (c'est-à-dire que leurs notes respectives sont en proportion géométrique), alors cela doit aussi être vrai pour un item que l'un des utilisateurs n'a pas encore noté. Cette approche est directement inspirée des travaux en classification analogique. Nos expériences ont montré que les performances de ces algorithmes sont proches de celles des techniques par voisinage, mais leur complexité cubique conduit à des temps de calculs prohibitifs.

Cela nous a conduit à considérer une autre approche pour la recommandation analogique, qui ne repose pas sur la recherche de triplets d'utilisateurs. Cette approche [HPRS16b], qui repose sur la notion de "*clones*", prend en compte le fait que chaque utilisateur a une appréciation bien particulière de l'échelle de note qui est utilisée. Nos résultats montrent que le concept de clone est très pertinent et qu'il est primordial de l'intégrer d'une manière ou d'une autre dans un algorithme de prédiction de notes. Il est cependant nécessaire de noter que ce type de biais dans les notes avait déjà été considéré dans de précédents travaux. Nous noterons enfin que nos

travaux sur les systèmes de recommandation nous ont conduit à développer une bibliothèque Python [Hug17] qui permet de facilement tester et développer des algorithmes de prédiction de notes.

Enfin, nous avons proposé un algorithme pour la fouille de proportions analogiques dans des bases de données incomplètes [HPRS16c], en nous inspirant fortement des travaux en fouille de règles d'association. Nous avons utilisé cet algorithme pour extraire des proportions analogiques entre films dans la base de données MovieLens, qui était celle que nous utilisions pour nos algorithmes de prédiction de notes. Les résultats suggèrent qu'en fait, les analogies potentielles restent d'un intérêt relativement faible, en ceci qu'elle mettaient généralement en jeu des items soit également appréciés, soit également désapprouvés. En d'autre termes, les notes relatives aux différents utilisateurs pour les quatre films d'une même proportion étaient souvent les mêmes. Ceci nous permet de rétrospectivement interpréter les résultats modestes de notre premier algorithme pour la prédiction de notes : cela suggère que finalement, l'algorithme ne pouvait trouver que trop peu de bonnes proportions pour produire des estimations de qualité.

Dans le chapitre 3, nous avons décrit nos contributions au problème de la classification analogique. Notre première contribution a été de proposer une définition fonctionnelle des classifieurs analogiques, qui a la particularité d'unifier les approches préexistantes. A partir de cette définition, nous avons été capables de dériver différentes propriétés théoriques. En particulier, nous avons montré que la VC-dimension des classifieurs analogiques était infinie, et que leur taux d'erreur est intimement lié à celui des algorithmes k-NN. En fait, notre définition fonctionnelle établie clairement le lien qui relie la classification analogique et la classification par plus-proches-voisins. Nous avons montré en effet que la classification analogique peut se concevoir en deux étapes. La première étape consiste à étendre l'ensemble d'apprentissage en ajoutant de nouveaux exemples dont la classe est estimée en utilisant le principe d'inférence analogique. Ensuite, sur la base de cette extension analogique, un algorithme k-NN est utilisé pour classer le reste des éléments qui ne sont pas dans l'extension. Ce nouveau point de vue est assez intéressant car il fait clairement intervenir deux des principales caractéristiques des proportions analogiques, à savoir la créativité et l'inférence.

Dans le chapitre 6, nous nous sommes intéressés à une question qui découle naturellement des conclusions du chapitre 3 : comment assurer que l'extension analogique soit saine ? En d'autres termes, existe-t-il un critère qui nous permette d'être sûr que les éléments de l'extension analogique sont correctement classés ? Nous avons été capables de répondre à cette question pour les domaines booléens : l'extension analogique est parfaitement saine si (et seulement si) la fonction booléenne f qui détermine les labels est une fonction affine. Nous avons étendu ces résultats aux domaines réels, et au cas où les attributs ont des valeurs nominales. Cette série de travaux ouvre la porte à diverses pistes de recherches futures, que nous mentionnons maintenant.

Axes de recherches futures

Dans la section 6.3.1, nous avons présenté le concept de fonction approximativement AP (fonction qui préserve approximativement l'analogie). En effet en pratique, il n'y a aucun moyen de savoir si la fonction cible f est vraiment affine. Ainsi, il est d'un intérêt crucial d'obtenir des garanties statistiques relatives à la qualité de l'extension analogique lorsque la fonction f s'éloigne de l'ensemble des fonctions affines. Un résultat comme le suivant serait très intéressant :

$$P(\omega_S^f \geq \eta) > 1 - \delta,$$

où $\eta \in [0, 1]$ et $\delta \in [0, \frac{1}{2}]$ sont des fonctions qui dépendent de ε et de $\mid S \mid$. La valeur ε indique que f est ε-proche de l'ensemble des fonctions affines. Pour le moment, nous nous intéressons à un problème un peu plus simple. On se propose en effet de chercher une borne δ telle que :

$$P(\omega_S^f < 1) < \delta,$$

où δ est toujours fonction de ε. En d'autres termes, on cherche à borner la probabilité de l'événement "$\mathbf{E}_S^{f*}$ *contient des erreurs*".

Nous avons aussi remarqué lors de nos expérimentations que même lorsqu'une fonction ne préserve pas du tout l'analogie, la qualité de l'extension analogique peut tout de même être très bonne. Cela est dû au fait que la procédure de vote majoritaire permet parfois de contrebalancer les erreurs de certains predicteurs. Il est clair qu'une compréhension profonde du rôle de ce vote majoritaire reste une étape clef dans l'étude des classifieurs analogiques.

Un autre axe de recherche possible serait de tirer partie du fait que toute fonction booléenne est en fait affine par morceaux. Si l'on peut être sûr que quatre éléments sont dans un domaine où f est affine, alors on peut appliquer de manière correcte le principe d'inférence analogique. Pour l'instant, nous n'avons mené que des expérimentations très préliminaires à ce sujet.

Enfin, nous terminerons cette discussion en évoquant l'apprentissage par transfert (*transfer learning*). L'apprentissage par transfert est un sujet assez populaire de nos jours dans la communauté de l'apprentissage artificiel, dont le but est de transférer des connaissances d'un problème source S pour les appliquer à un problème moins bien connu T. Indubitablement, l'essence même de l'analogie est d'opérer un transfert de connaissances. L'usage des proportions analogiques appliquées à un tel domaine reste encore entièrement à explorer.

Bibliography

[AP94] A. Aamodt and E. Plaza.
Case-based reasoning: Foundational issues, methodological variations, and system approaches.
AI communications, 7(1):39–59, 1994.

[AS94] R. Agrawal and R. Srikant.
Fast algorithms for mining association rules in large databases.
In *Proc. 20th Int. Conf. on Very Large Data Bases*, VLDB'94, pages 487–499, San Francisco, 1994. Morgan Kaufmann Pub, 1994.

[AT05] G. Adomavicius and A. Tuzhilin.
Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions.
IEEE Trans. on Knowl. and Data Eng., 17(6):734–749, June 2005.

[Bar10] P. Bartha.
By parallel reasoning.
Oxford University Press, 2010.

[Bar16] P. Bartha.
Analogy and analogical reasoning.
In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy.* Metaphysics Research Lab, Stanford University, winter 2016 edition, 2016.

[BCN16] L. Bottou, F. E. Curtis, and J. Nocedal.
Optimization Methods for Large-Scale Machine Learning.
ArXiv e-prints, June 2016.

[BEHW89] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth.
Learnability and the Vapnik-Chervonenkis dimension.
J. ACM, 36(4):929–965, 1989.

[BK07a] R. Bell and Y. Koren.
Lessons from the Netflix prize challenge.
SIGKDD Explor. Newsl., 9(2):75–79, December 2007.

[BK07b] R. Bell and Y. Koren.

Scalable collaborative filtering with jointly derived neighborhood interpolation weights.
In *Data Mining, 2007. ICDM'07*, pages 43–52. IEEE, 2007.

[BLR93] M. Blum, M. Luby, and R. Rubinfeld.
Self-testing/correcting with applications to numerical problems.
J. Comput. Syst. Sci., 47(3):549–595, December 1993.

[BM09] N. Barbot and L. Miclet.
La proportion analogique dans les groupes. Applications à l'apprentissage et à la génération.
In *Proc. Conf. francophone sur l'apprentissage artificiel (CAP), Hammamet, Tunisia*, 2009.

[BMD07] S. Bayoudh, L. Miclet, and A. Delhay.
Learning by analogy: A classification rule for binary and nominal data.
Proc. Inter. Joint Conf. on Artificial Intelligence IJCAI07, pages 678–683, 2007.

[BMMA07] S. Bayoudh, H. Mouchère, L. Miclet, and E. Anquetil.
Learning a classifier with very few examples: Analogy based and knowledge based generation of new examples for character recognition.
In *Proc. 18th Europ. Conf. on Machine Learning (ECML'07)*, pages 527–534. Springer, 2007.

[BPR12] M. Bayoudh, H. Prade, and G. Richard.
Evaluation of analogical proportions through Kolmogorov complexity.
Knowl.-Based Syst., 29:20–30, 2012.

[BPR16] M. Bounhas, H. Prade, and G. Richard.
Not being at odds with a class: A new way of exploiting neighbors for classification.
In G. A. Kaminka, M. Fox, P. Bouquet, E. Hüllermeier, V. Dignum, F. Dignum, and F. van Harmelen, editors, *Proc. 22nd Europ. Conf. on Artificial Intelligence (ECAI'16)*, volume 285, pages 1662–1663. IOS Press, 2016.

[BR11] R. Burke and M. Ramezani.
Matching Recommendation Technologies and Domains, pages 367–386.
In Ricci et al. [RRSK11], 2011.

[Bur98] C. J.C. Burges.
A tutorial on support vector machines for pattern recognition.
Data Mining and Knowledge Discovery, 2:121–167, 1998.

[Bur02] R. Burke.
Hybrid recommender systems: Survey and experiments.
User Modeling and User-Adapted Interaction, 12(4):331–370, 2002.

[Bur07] R. Burke.
The adaptive web.
chapter Hybrid Web Recommender Systems, pages 377–408. Springer, Berlin, Heidelberg, 2007.

[CFH92] D. Chalmers, R. French, and D. Hofstadter.
High-level perception, representation, and analogy: A critique of artificial intelligence methodology.
Journal of Experimental and Theoretical Artificial Intelligence, 4:185–211, 1992.

[CH67] T. M. Cover and P. E. Hart.
Nearest neighbor pattern classification.
In *IEEE Transactions on Information Theory*, volume 13, pages 21–27. IEEE, 1967.

[CHPR17] M. Couceiro, N. Hug, H. Prade, and G. Richard.
Analogy-preserving functions: A way to extend boolean samples.
In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-26*, pages 1575–1581, 2017.

[CL09] M. Couceiro and E. Lehtonen.
Generalizations of Swierczkowski's lemma and the arity gap of finite functions.
Discrete Math., 309(20):5905–5912, 2009.

[Cor96a] A. Cornuéjols.
Analogie, principe d'économie et complexité algorithmique.
Actes des 11èmes Journées Françaises de l'Apprentissage, 1996.

[Cor96b] A. Cornuéjols.
Analogy as minimization of description length.
In G. Nakhaeizadeh and C. Taylor, editors, *Machine Learning and Statistics: The interface*, pages 321–336. Wiley and Sons, 1996.

[CP08] M. Couceiro and M. Pouzet.
On a quasi-ordering on Boolean functions.
Theor. Comput. Sci., 396(1-3):71–87, 2008.

[CPCAL02] J. Cano, J-C. Perez-Cortes, J. Arlandis, and R. Llobet.
Training Set Expansion in Handwritten Character Recognition, pages 548–556.
Springer, 2002.

[DG04] J. Dean and S. Ghemawat.
MapReduce: Simplified data processing on large clusters.

In *Proceedings of the 6th Conference on Symposium on Operating Systems Design & Implementation - Volume 6*, OSDI'04, pages 10–10, Berkeley, CA, USA, 2004. USENIX Association.

[DR87] T. R. Davies and S. J. Russell.
A logical approach to reasoning by analogy.
In *Proceedings of the 10th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'87, pages 264–270, San Francisco, CA, USA, 1987. Morgan Kaufmann Publishers Inc.

[Eva64] T. G. Evans.
A heuristic program to solve geometric-analogy problems.
In *Proceedings of the April 21-23, 1964, Spring Joint Computer Conference*, AFIPS'64 (Spring), pages 327–338, New York, NY, USA, 1964. ACM.

[FB08] A. Felfernig and R. Burke.
Constraint-based recommender systems: Technologies and research issues.
In *Proceedings of the 10th International Conference on Electronic Commerce*, ICEC'08, pages 3:1–3:10, New York, NY, USA, 2008. ACM.

[FFG89] B. Falkenhainer, K. Forbus, and D. Gentner.
The structure-mapping engine: Algorithm and examples.
Artificial intelligence, 41(1):1–63, 1989.

[FFJZ11] A. Felfernig, G. Friedrich, D. Jannach, and M. Zanker.
Developing Constraint-based Recommenders, pages 187–215.
In Ricci et al. [RRSK11], 2011.

[FPY95] S. Federici, V. Pirrelli, and F. Yvon.
A dynamic approach to paradigm-driven analogy.
In *Learning for Natural Language Processing*, volume 1040 of *Lecture Notes in Computer Science*, pages 385–398. Springer, 1995.

[Gen83] D. Gentner.
Structure-mapping: A theoretical framework for analogy.
Cognitive Science, 7(2):155 – 170, 1983.

[GKS06] H. Gust, K. Kühnberger, and U. Schmid.
Metaphors and heuristic-driven theory projection (HDTP).
Theoretical Computer Science, 354(1):98 – 117, 2006.

[Hen69] N. M. Henley.
A psychological study of the semantics of animal terms.
Journal of Verbal Learning and Verbal Behavior, 8(2):176 – 184, 1969.

[Hes59] M. Hesse.

On defining analogy.
In *Proceedings of the Aristotelian Society*, volume 60, pages 79–100. JSTOR, 1959.

[Hes66] M. Hesse.
Models and Analogies in Science.
University of Notre Dame Press, 1966.

[HKTR04] J. L. Herlocker, J. A. Konstan, Loren G. Terveen, and J. T. Riedl.
Evaluating collaborative filtering recommender systems.
ACM Trans. Inf. Syst., 22(1):5–53, 2004.

[HM+94] D. Hofstadter, M. Mitchell, et al.
The copycat project: A model of mental fluidity and analogy-making.
Advances in connectionist and neural computation theory, 2(31-112):29–30, 1994.

[HPR15] N. Hug, H. Prade, and G. Richard.
Experimenting analogical reasoning in recommendation.
In F. Esposito, O. Pivert, M.-S. Hacid, Z. W. Ras, and S. Ferilli, editors, *Proc. 22nd Int. Symp. on Foundations of Intelligent Systems (ISMIS'15), Lyon, Oct. 21-23*, volume 9384 of *LNCS*, pages 69–78. Springer, 2015.

[HPRS16a] N. Hug, H. Prade, G. Richard, and M. Serrurier.
Analogical classifiers: A theoretical perspective.
In *Proc. 22nd Europ. Conf. on Artificial Intelligence (ECAI'16)*, Frontiers in Artificial Intelligence and Applications, pages 689–697. IOS Press, 2016.

[HPRS16b] N. Hug, H. Prade, G. Richard, and M. Serrurier.
Analogy in recommendation. Numerical vs. ordinal: A discussion.
In *2016 IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 2016, Vancouver, BC, Canada, July 24-29, 2016*, pages 2220–2226. IEEE, 2016.

[HPRS16c] N. Hug, H. Prade, G. Richard, and M. Serrurier.
Proportions analogiques : créativité et fouille de données.
In *Rencontres francophones sur la Logique Floue et ses Applications*, 2016.

[HT89] K. J Holyoak and P. Thagard.
Analogical mapping by constraint satisfaction.
Cognitive science, 13(3):295–355, 1989.

[Hug17] N. Hug.
Surprise, a Python library for recommender systems.
`http://surpriselib.com`, 2017.

[JRTZ16] D. Jannach, P. Resnick, A. Tuzhilin, and M. Zanker.
Recommender systems: beyond matrix completion.
Communications of the ACM, 59(11):94–102, 2016.

[KB11] Y. Koren and R. Bell.
Advances in Collaborative Filtering, pages 145–186.
In Ricci et al. [RRSK11], 2011.

[KB14] M. Kaminskas and D. Bridge.
Measuring surprise in recommender systems.
In P. Adamopoulos et al., editor, *Procs. of the Workshop on Recommender Systems Evaluation: Dimensions and Design (Workshop Programme of the 8th ACM Conference on Recommender Systems)*, 2014.

[KBV09] Y. Koren, R. Bell, and C. Volinsky.
Matrix factorization techniques for recommender systems.
Computer, 42(8):30–37, August 2009.

[Kle83] S. Klein.
Analogy and mysticism and the structure of culture (and Comments & Reply).
Current Anthropology, 24 (2):151–180, 1983.

[Kor09] Y. Koren.
The BellKor solution to the Netflix grand prize, 2009.

[Kor10] Y. Koren.
Factor in the neighbors: Scalable and accurate collaborative filtering.
ACM Trans. Knowl. Discov. Data, 4(1):1:1–1:24, January 2010.

[KS11] Y. Koren and J. Sill.
Ordrec: An ordinal model for predicting personalized item rating distributions.
In *Proceedings of the Fifth ACM Conference on Recommender Systems*, RecSys'11, pages 117–124, New York, NY, USA, 2011. ACM.

[LdGS11] P. Lops, M. de Gemmis, and G. Semeraro.
Content-based Recommender Systems: State of the Art and Trends, pages 73–105.
In Ricci et al. [RRSK11], 2011.

[Lep98] Y. Lepage.
Solving analogies on words: An algorithm.
In *COLING-ACL*, pages 728–735. Morgan Kaufmann Publishers / ACL, 1998.

[Lep03] Y. Lepage.
De l'analogie rendant compte de la commutation en linguistique.
Habilitation à diriger des recherches, Université Joseph-Fourier - Grenoble I, May 2003.

[LFFP06] R. Fergus L. Fei-Fei and P. Perona.
One-shot learning of object categories.
IEEE Trans. on Pattern Analysis and Machine Intelligence, 28(4):594–611, 2006.

[LFU10] A. Lovett, K. Forbus, and J. Usher.
A structure-mapping model of Raven's progressive matrices.
In *Proceedings of CogSci*, volume 10, pages 2761–2766, 2010.

[LI93] P. Langley and W. Iba.
Average-case analysis of a nearest neighbor algorithm.
In *Proceedings of the 13th Int. Joint Conf. on Artificial Intelligence. Chambéry, France, August 28 - September 3*, pages 889–894. Morgan Kaufmann, 1993.

[Lic13] M. Lichman.
UCI machine learning repository, 2013.

[LS96] Y. Lepage and A. Shin-ichi.
Saussurian analogy: a theoretical account and its application.
In *COLING*, pages 717–722, 1996.

[MBD08] L. Miclet, S. Bayoudh, and A. Delhay.
Analogical dissimilarity: Definition, algorithms and two experiments in machine learning.
J. Artif. Intell. Res. (JAIR), 32:793–824, 2008.

[MC16] P-A. Murena and A. Cornuéjols.
Minimum description length principle applied to structure adaptation for classification under concept drift.
In *2016 International Joint Conference on Neural Networks, IJCNN 2016, Vancouver, BC, Canada, July 24-29, 2016*, pages 2842–2849, 2016.

[MCCD13] T. Mikolov, K. Chen, G. Corrado, and J. Dean.
Efficient estimation of word representations in vector space.
arXiv preprint arXiv:1301.3781, 2013.

[MD04] L. Miclet and A. Delhay.
Relation d'analogie et distance sur un alphabet défini par des traits.
Technical Report 1632, IRISA, July 2004.

[Mit93] M. Mitchell.
Analogy-making as perception: A computer model.
MIT Press, 1993.

[Mit01] M. Mitchell.
Analogy making as a complex adaptive system.
In *Design Principles for the Immune System and Other Distributed Autonomous Systems*, 2001.

[MP09] L. Miclet and H. Prade.
Handling analogical proportions in classical logic and fuzzy logics settings.

In *European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, pages 638–650. Springer, 2009.

[MRK06] S. M. McNee, J. Riedl, and J. A. Konstan.
Being accurate is not enough: how accuracy metrics have hurt recommender systems.
In G. M. Olson and R. Jeffries, editors, *Extended Abstracts Proceedings of the 2006 Conference on Human Factors in Computing Systems (CHI'06), Montréal, Québec, Canada, April 22-27*, pages 1097–1101, 2006.

[MYZ13] T. Mikolov, W.-T. Yih, and G. Zweig.
Linguistic regularities in continuous space word representations.
In *HLT-NAACL*, volume 13, pages 746–751, 2013.

[OY97] S. Okamoto and N. Yugami.
An average-case analysis of the k-nearest neighbors classifier for noisy domains.
In *Proceedings of the Fifteenth Int. Joint Conf. on Artificial Intelligence, IJCAI 97, Nagoya, Japan, August 23-29*, pages 238–245. Morgan Kaufmann, 1997.

[Pat07] A. Paterek.
Improving regularized singular value decomposition for collaborative filtering.
In *Proceedings of KDD cup and workshop*, volume 2007, pages 5–8, 2007.

[Pia52] J. Piaget.
Essai sur les transformations des opérations logiques: les 256 opérations ternaires de la logique bivalente des propositions.
Presses Universitaires de France, 1952.

[Pol45] G. Polya.
How to Solve It.
Princeton University Press, 2nd ed. 1957, 1945.

[Pol54] G. Polya.
Mathematics and Plausible Reasoning- Vol.1: Induction and analogy in Mathematics, Vol.2: Patterns of Plausible Inference.
Princeton Univ. Press, 2nd ed. 1968, 1954.

[PR12] H. Prade and G. Richard.
Homogeneous logical proportions: Their uniqueness and their role in similarity-based prediction.
In *KR*. AAAI Press, 2012.

[PR13a] H. Prade and G. Richard.
Analogical proportions and multiple-valued logics.

In *European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, pages 497–509. Springer, 2013.

[PR13b] H. Prade and G. Richard.
From analogical proportion to logical proportions.
Logica Universalis, 7(4):441–505, 2013.

[PR14a] H. Prade and G. Richard.
From analogical proportion to logical proportions: A survey.
In *Computational Approaches to Analogical Reasoning: Current Trends*, pages 217–244. 2014.

[PR14b] H. Prade and G. Richard.
A short introduction to computational trends in analogical reasoning.
In *Computational Approaches to Analogical Reasoning: Current Trends*, pages 1–22. 2014.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay.
Scikit-learn: Machine learning in Python.
Journal of Machine Learning Research, 12:2825–2830, 2011.

[PY99] V. Pirrelli and F. Yvon.
The hidden dimension: a paradigmatic view of data-driven NLP.
J. Exp. Theor. Artif. Intell., 11(3):391–408, 1999.

[PY10] S. J. Pan and Q. Yang.
A survey on transfer learning.
IEEE Trans. on Knowl. and Data Eng., 22(10):1345–1359, 2010.

[RA05] D. E. Rumelhart and A. A. Abrahamsen.
A model for analogical reasoning.
Cognitive Psychol., 5:1–28, 2005.

[Ris78] J. Rissanen.
Modeling by shortest data description.
Automatica, 14(5):465 – 471, 1978.

[RRA09] Teodora-Liliana Radulescu, Vicentiu D Radulescu, and Titu Andreescu.
Problems in Real Analysis: advanced calculus on the real axis.
Springer Science & Business Media, 2009.

[RRSK11] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors.
Recommender Systems Handbook.
Springer, 2011.

[Sal63] A. Salomaa.
On essential variables of functions, especially in the algebra of logic.
Ann. Acad. Sci. Fenn. Ser. A I. Math., 339:3–11, 1963.

[SAO+11] T. Sakaguchi, Y. Akaho, K. Okada, T. Date, T. Takagi, N. Kamimaeda, M. Miyahara, and T. Tsunoda.
Recommendation system with multi-dimensional and parallel-case four-term analogy.
In *Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on*, pages 3137–3143. IEEE, 2011.

[SBM12] S. Schelter, C. Boden, and V. Markl.
Scalable similarity-based neighborhood methods with MapReduce.
In *Proceedings of the Sixth ACM Conference on Recommender Systems*, RecSys'12, pages 163–170, New York, NY, USA, 2012. ACM.

[SFHS07] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen.
The adaptive web.
chapter Collaborative Filtering Recommender Systems, pages 291–324. Springer, Berlin, Heidelberg, 2007.

[SG11] G. Shani and A. Gunawardana.
Evaluating Recommendation Systems, pages 257–297.
In Ricci et al. [RRSK11], 2011.

[SKKR00] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl.
Application of dimensionality reduction in recommender system-a case study.
Technical report, DTIC Document, 2000.

[SM08] R. Salakhutdinov and A. Mnih.
Probabilistic matrix factorization.
In *Advances in neural information processing systems*, pages 1257–1264, 2008.

[Sto36] M. H. Stone.
The theory of representation for Boolean algebras.
Trans. of the American Mathematical Society, 40(1):37–111, 1936.

[SY05a] N. Stroppa and F. Yvon.
An analogical learner for morphological analysis.
In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages 120–127. Association for Computational Linguistics, 2005.

[SY05b] N. Stroppa and F. Yvon.
Analogical learning and formal proportions: Definitions and methodological issues.
Technical Report D004, ENST-Paris, 2005.

[Val84] L. G. Valiant.
A theory of the learnable.
Commun. ACM, 27(11):1134–1142, November 1984.

[Vap98] V. Vapnik.
Statistical Learning Theory.
Wiley, 1998.

[Wil96] R. Willard.
Essential arities of term operations in finite algebras.
Discrete Math., 149(1-3):239–259, 1996.

[Win80] P. H. Winston.
Learning and reasoning by analogy.
Communications of the ACM, 23(12):689–703, 1980.

[WM04] L. Wolf and I. Martin.
Regularization through feature knock out.
MIT Computer Science and Artificial Intelligence Laboratory, (CBCL Memo 242), 2004.

[Zhe27] I. I. Zhegalkin.
On the technique of calculating propositions in symbolic logic.
Mat. Sb, 43:9–28, 1927.

List of Tables

List of Figures

List of Definitions