PHILIPPE CHOQUETTE

NOUVEAUX ALGORITHMES D'APPRENTISSAGE POUR CLASSIFICATEURS DE TYPE SCM

Mémoire présenté
à la Faculté des études supérieures de l'Université Laval
dans le cadre du programme de maîtrise en informatique
pour l'obtention du grade de maître ès sciences (M. Sc.)

FACULTÉ DES SCIENCES ET DE GÉNIE
UNIVERSITÉ LAVAL
QUÉBEC

OCTOBRE 2007

Résumé

Dans le cadre de l'apprentissage automatique supervisé, un des outils disponibles pour la classification binaire est la *Set Covering Machine* (SCM). Rapidement construite et en général très performante, elle n'est cependant pas systématiquement infaillible. Il existe encore, à ce jour, une marge pour une amélioration.

Ce mémoire présente deux nouvelles façons de construire des SCM. Ces algorithmes sont décrits, expliqués et leur performance est analysée. La première façon est de minimiser une approximation d'une borne sur le risque à l'aide d'un *branch-and-bound*. La deuxième est d'utiliser le *bagging*.

Lors des tests, les nouveaux classificateurs se sont montrés aussi performants que les SCM originales. Nous avons découvert que celles-ci sont soit déjà optimales au sens du critère utilisé pour le *branch-and-bound*, soit aussi performantes que les SCM optimales.

Abstract

In the supervised machine learning field, one of the available tools for binary classification is the Set Covering Machine (SCM). Quickly built and generally having high performance, it's however not proven that they always give optimal results. There is still, to date, a margin for improvement.

This study presents two new ways of building SCM. Theses algorithms are described, explained and their performance is analyzed. The first way is to minimize an approximated bound on the risk with a branch-and-bound. The second is using bagging.

The new classifiers had the same test-set performance than the original SCM. We discovered that the latter are either already optimal according to the branch-and-bound criterion or having the same performance as the optimal SCM.

Avant-propos

Cette maîtrise en informatique m'a permis, en plus d'en apprendre énormément sur l'apprentissage automatique, de mettre au point une méthode de travail et de voir plus clairement dans quelle direction ce travail serait orienté. Dans le futur, il est certain que mes activités continueront de comporter une bonne part d'analyse de données et de programmation.

L'enseignement méthodologique principal des travaux accomplis est, à mon avis, de ne pas oublier de penser et ensuite de coder, et non l'inverse. Cette devise ne m'était pas inconnue avant de commencer, mais parfois l'apprentissage doit se faire par l'expérimentation personnelle.

En ce qui concerne la dimension humaine, mon expérience et mes réflexions ont fait évoluer mes idées sur la rigueur, la confiance et l'intégrité. Voici une mention pour les personnes présentes, à différents degrés et de différentes manières, qui ont alimenté ces réflexions.

Je remercie grandement les acteurs principaux de mon encadrement, soit les Drs Mario Marchand et François Laviolette, mes codirecteurs. Leur disponibilité et leur imagination ont été des atouts sans prix pour l'avancement de mon projet. De plus, leur bourse d'études a été une assistance financière très appréciée.

Je remercie également M. Alexandre Lacasse pour la lecture de ce mémoire et pour les informations techniques ou abstraites, car elles m'ont été très utiles. Je remercie Mme Sigrid Choquette pour la révision de ce document. Je remercie M. Charles-Henri Coulombe et Mme Lucienne Parker pour leur soutien financier et je me dois de mentionner la Dre Nicole Tourigny pour sa contribution pour ma candidature à une bourse. Finalement, merci au Dr Guy Mineau pour avoir laissé libre accès à ses ouvrages de référence.

Table des matières

Liste des tableaux

Table des figures

Chapitre 1

Introduction

Les œuvres de science-fiction présentent souvent des êtres artificiels intelligents. Malgré le niveau de technologie propre à notre société, les machines que nous pourrions qualifier de pensantes sont encore du domaine de la fiction. Ce ne sont pourtant pas les multiples efforts qui manquent. L'intelligence artificielle est une branche de l'informatique qui s'intéresse à l'automatisation du comportement intelligent. Ce domaine regroupe, entre autres, l'apprentissage automatique, le raisonnement à partir de cas, la représentation des connaissances et le traitement automatique du langage naturel. Bien que nous n'ayons pas encore une définition complète de l'intelligence, nous pouvons admettre *a priori* qu'elle inclut raisonner, planifier, penser abstraitement, comprendre des idées et le langage et, surtout, apprendre. L'intelligence artificielle est une terminologie aussi utilisée dans l'industrie du jeu vidéo. Pourtant sur le marché, nous n'observons peu ou pas d'adversaires artificiels qui peuvent effectivement apprendre d'après leur expérience. C'est donc un champ qui demeure ouvert, car les possibilités d'avancées sont encore immenses.

L'apprentissage automatique [1] permet de relever des défis dans une diversité de domaines des plus étendue. Cette discipline consiste à construire des classificateurs qui permettent de déterminer la classe d'un nouvel exemple après avoir appris en traitant des exemples d'entraînement. Un algorithme d'apprentissage automatique reçoit en entrée un jeu de données d'apprentissage et produit en sortie un classificateur. Dans le cas de l'apprentissage supervisé, les exemples d'entraînement sont généralement étiquetés (classifiés) par des humains. Les exemples sont composés d'un objet d'entrée et d'une valeur de sortie. Il est ici question d'apprentissage par induction, car les algorithmes tentent de généraliser à tous les exemples ce qu'ils apprennent avec les exemples d'entraînement. Les algorithmes d'apprentissage dans le cadre de l'intelligence artificielle sont à classer parmi les méthodes non paramétriques ; c'est-à-dire qu'aucune hypothèse

probabiliste n'est faite sur la distribution des objets d'entrée à l'intérieur des classes. L'objectif principal en apprentissage automatique est de produire des classificateurs qui se trompent le moins souvent possible lors de la classification de nouveaux exemples. C'est donc dans cette direction que la recherche avance. Une autre préoccupation importante est que la construction des classificateurs soit faite en un temps raisonnable.

Les applications de diagnostic de maladies à partir de symptômes ou de prédiction de leur développement sont multiples. Désormais, dépister un cancer grâce à un profil d'expression des gènes est une tâche plus exacte et moins coûteuse avec l'aide d'un classificateur nécessitant un nombre réduit de gènes à observer. La prise de décision automatisée est une autre application de l'apprentissage automatique, comme de donner une réponse à la demande de prêt bancaire de la part d'un client sur la base de sa situation personnelle. La reconnaissance de formes est aussi un problème d'apprentissage automatique. Des cas concrets sont la reconnaissance de caractères manuscrits qui peut s'appliquer au classement des lettres par le code postal et la reconnaissance de la parole. Des classificateurs peuvent être aussi utilisés pour déterminer si un courriel est légitime ou non. C'est un exemple de catégorisation automatique de textes qui touche la majorité des gens qui utilisent le courrier électronique, car la guerre contre les courriels non sollicités, les pourriels, est loin d'être gagnée.

L'algorithme sur lequel se porte notre attention tout au long de ce mémoire est celui proposé pour les *Set Covering Machines* [2, 3]. Tous les algorithmes proposés dans ce mémoire tentent de l'améliorer soit en y apportant des modifications, soit en les utilisant dans un contexte où les données subissent un traitement préalable grâce à un algorithme auxiliaire. Les SCM font partie de la famille des algorithmes d'apprentissage automatique qui utilisent la compression de données. C'est-à-dire que des exemples des données d'apprentissage sont enregistrés dans le classificateur pour être réutilisés lors de la classification. Moins il y a d'exemples enregistrés, plus la compression est efficace. L'ensemble de ces exemples est dénommé ensemble de compression. Les SCM sont basées sur un algorithme vorace contenant certains paramètres ajustables permettant à l'utilisateur de choisir le compromis précision (sur l'échantillon d'apprentissage) et compression de données. La version des SCM utilisée au cours des travaux relatifs à cette maîtrise est celle qui construit des hypersphères à partir des exemples d'entraînement. Dans un article traitant des Listes de décision [4], Marchand et Sokolova ont proposé une borne supérieure sur la probabilité de faire une erreur de classification sur un nouvel exemple. Cette borne indique que la probabilité de faire une erreur sera faible lorsque le classificateur comprime beaucoup les données tout en ne faisant pas trop d'erreurs sur l'échantillon d'apprentissage. Cette borne a obtenu un certain succès pour la sélection des paramètres qui mènent au classificateur faisant le moins d'erreurs.

Un des concepts de base est la dimensionnalité d'un problème de classification. Celle-ci est le nombre de variables explicatives dans les jeux de données d'entraînement et des données de test. Les variables explicatives sont les colonnes des tableaux de données excluant la colonne qui contient la classe des exemples, alors que les exemples sont les lignes de ces mêmes tableaux. Les données test regroupent les exemples dont la classe est à déterminer par un classificateur issu d'un algorithme d'apprentissage. Lorsque les données test sont aussi étiquetées, nous nous servons de leur classe réelle pour évaluer la performance d'un classificateur en la comparant avec la classe proposée par celui-ci.

Lors de la résolution de problèmes à faible dimensionnalité (moins de 30 dimensions), les *Set Covering Machines* produisent souvent des classificateurs avec une marge d'erreur plus basse que ceux produits par d'autres algorithmes d'apprentissage automatique eux-mêmes très performants, comme les *Support Vector Machines* [5]. Cependant, les tentatives de résolution de problèmes à dimensionnalité beaucoup plus haute avec les SCM débouchent sur des classificateurs beaucoup moins performants. La reconnaissance de formes à partir d'images est un exemple de problème à très haute dimensionnalité.

L'algorithme SCM-classique donne une solution approximative au problème NP-complet du recouvrement minimal « Minimum Set Cover » [6]. Pour que cet algorithme converge, l'espace des classificateurs possibles n'est que sommairement exploré. Donc, le classificateur obtenu peut être très différent du classificateur optimal. L'objectif principal du travail effectué au cours de cette maîtrise est de concevoir un algorithme d'apprentissage, pour classificateurs de type SCM, qui trouve le classificateur minimisant la borne sur le risque de Marchand et Sokolova. La motivation de cet objectif se trouve dans le fait que la borne croît proportionnellement au nombre d'erreurs sur les données d'entraînement et à la taille de l'ensemble de compression. Lorsque nous construisons un classificateur de type SCM, nous cherchons quelle est sa meilleure taille. Si elle est trop petite, il fera beaucoup d'erreurs sur les données d'entraînement et aussi sur les données test. À l'opposé, si elle est trop grande, il fera peu d'erreurs sur les données d'entraînement, mais en fera beaucoup sur les données test. Dans ce cas, il généralisera mal l'information et ce sera un cas de surlissage ou « overfitting ». Étant donné que le nombre d'erreurs augmente si la taille diminue, minimiser la borne devrait donner l'ensemble de compression de taille et de composition idéale. L'interrogation à laquelle nous cherchions une réponse en appliquant cette technique était de savoir quelle est la performance du classificateur de composition idéale selon la borne, et où se positionne par rapport à celui-ci le classificateur obtenu par l'algorithme SCM-classique. Les résultats ont montré que la sélection de modèle par la borne combinée à l'approche vorace donnait déjà des classificateurs très proches de l'optimalité. Cette information a pour conséquence de donner une valeur supplémentaire aux classificateurs obtenus par

l'approche vorace, puisqu'un classificateur, que nous savons quasi optimal, peut être obtenu en un temps très satisfaisant, alors que la certitude de l'optimalité est obtenue au prix d'une construction beaucoup plus longue.

L'espace des classificateurs de type SCM possibles pour un ensemble d'apprentissage donné peut être représenté par un arbre. À chaque noeud est associé un classificateur potentiel et une valeur de la borne. Pour trouver les classificateurs minimisant la borne dans cet espace, une recherche de type *branch-and-bound* [7] a été effectuée en minimisant une approximation de la borne. Une version algorithmiquement très proche du *branch-and-bound* classique a été implémentée et testée. Par la suite, une version plus restrictive a été élaborée, implémentée et testée.

L'objectif secondaire de cette maîtrise est d'appliquer un vote de majorité aux SCM. L'utilisation d'un vote de majorité pouvait prendre deux formes : le *boosting* et le *bagging* [8]. Ce dernier a semblé plus approprié, car les classificateurs de type SCM sont plutôt sensibles aux variations dans les données d'entraînement. Des fluctuations mineures dans les données d'entraînement peuvent conduire à la construction de classificateurs complètement différents. Le *bagging*, acronyme de « bootstrap aggregating », consiste à piger avec remise des exemples à partir de l'ensemble d'entraînement initial, à construire des classificateurs à partir de chacun de ceux-ci et à regrouper l'information en faisant un vote de majorité. Dans le cas présent, l'algorithme de SCM ne se trouve pas altéré, mais plutôt impliqué dans un méta-algorithme. Pour un exemple donné, un classificateur qui fonctionne avec un vote de majorité va retourner comme classe choisie celle qui est retournée par le plus grand nombre de sous-classificateurs.

Voici un aperçu de la structure de ce mémoire. Celui-ci est amorcé par une révision de concepts de base en apprentissage automatique à l'intérieur du chapitre 2. Les algorithmes des SCM et des SVM y sont décrits, car ils représentent les points de comparaison initiaux. Ensuite vient le chapitre 3 sur le *branch-and-bound* où l'algorithme de base est décrit, de même que les modifications qui y ont été apportées pour le faire fonctionner avec les SCM. Les deux versions implémentées sont décrites en détail, ainsi que les résultats obtenus sur les différents jeux de données d'entraînement utilisés. Le chapitre 4 traite du *bagging* et de la manière dont il a été implémenté. Les résultats obtenus y sont présentés et analysés.

Chapitre 2

L'apprentissage automatique

2.1 Introduction

Dans la discipline de l'apprentissage automatique, nous discernons les méthodes suivantes : les arbres de décision, les réseaux de neurones, les algorithmes génétiques, l'apprentissage par renforcement et bien d'autres. Comme il est mentionné précédemment, l'apprentissage automatique consiste à construire des classificateurs qui permettent de déterminer la classe d'un nouvel exemple après avoir appris en traitant des exemples d'entraînement. Dans ce domaine, les deux tendances principales sont : celle issue de l'intelligence artificielle et qualifiée de symbolique, et celle issue des statistiques et qualifiée de numérique. L'apprentissage symbolique se divise en trois types de raisonnement : par déduction, par induction et par analogie. Deux champs d'applications populaires de l'apprentissage automatique sont la reconnaissance de formes et le forage de données, ou extraction de connaissances à partir de données. Dans ce chapitre, nous verrons les concepts de base utiles en apprentissage automatique. En premier lieu, il sera question de la forme des données traitées. Ensuite, le risque sera présenté, suivi des intervalles de confiance. La validation croisée sera définie et l'apprentissage par compression de données sera expliqué. Une borne sur le risque sera introduite et en dernier lieu, les SCM, SVM et votes de majorité seront présentés.

2.2 Données

Dans cette section, nous introduisons la notation et présentons les données utilisées tout au long de ce mémoire. Les données nous servent à entraîner des classificateurs et à comparer leur performance.

2.2.1 Définitions

Tous les exemples utilisés pour entraîner un classificateur forment un ensemble S de m éléments dénommé échantillon d'apprentissage. Une variable X_j est dite explicative [9] si elle influence une autre variable Y dite d'intérêt, c'est-à-dire qui fait l'objet de l'étude. Comme il y a n variables explicatives, la dimensionnalité du problème est alors de n. En notant chaque exemple par $\mathbf{z}_i$, S est défini par :

$$S = \{\mathbf{z}_i\}_{i=1}^{m} = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_m\}$$

Un exemple $\mathbf{z}$ est composé d'un objet d'entrée $\mathbf{x}$ et d'une valeur de sortie y.

$$\mathbf{z} = (\mathbf{x}, y)$$

Un vecteur $\mathbf{x}$ est une réalisation d'une variable aléatoire multivariée $\mathbf{X} = (X_1, X_2, \ldots X_n)$ dont les valeurs possibles sont regroupées dans l'espace d'entrée $\mathcal{X}$. Un $\mathbf{x}$ est supposé tiré de $\mathcal{X}$ aléatoirement par un tirage indépendant et identiquement distribué (iid) suivant une distribution de probabilité $\mathbf{P}_{\mathbf{X}}(\mathbf{x}) = \mathbf{P}(\mathbf{X} = \mathbf{x})$. Cette distribution est inconnue de l'algorithme qui apprend. Nous aurons par exemple $\mathcal{X} = \mathbb{R}^n$ lorsque chaque X_j est une variable pouvant prendre n'importe quelle valeur dans $\mathbb{R}$ et $\mathcal{X} = \{0, 1\}^n$ lorsque chaque X_j est une variable Booléenne. Chaque variable aléatoire X_j est souvent dénommée par attribut j ou encore la j^e variable explicative. Maintenant que nous avons explicité l'espace d'entrée, mentionnons aussi qu'un y est une réalisation de la variable Y. L'espace de sortie, qui est noté $\mathcal{Y}$, est l'ensemble de toutes les réalisations possibles de Y.

La variable aléatoire pour un exemple complet est $\mathbf{Z} = (\mathbf{X}, Y)$. Il y a une $\mathbf{Z}_i$ pour chaque $\mathbf{z}_i$. $\mathbf{Z}^m = \{\mathbf{Z}_i\}_{i=1}^{m}$ est un ensemble de m variables $\mathbf{Z}_i$, que l'on suppose iid.

Dans le cas des algorithmes d'apprentissage *batch*, les données sont accessibles sous forme de tableaux. Voici la forme d'un tel tableau sans y inscrire de véritables données.

$$\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} & y_1 \\ x_{21} & x_{22} & \cdots & x_{2n} & y_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} & y_m \end{bmatrix}$$

Dans le tableau, chaque ligne correspond à un exemple où, à l'exception de la dernière colonne, l'entrée de la j^e colonne est la valeur pour cet exemple de la j^e variable explicative. La dernière colonne, celle de y, contient la classe de chaque exemple. Étant donné que nous nous arrêtons seulement sur des problèmes de classification binaires, la classe d'un exemple prend comme valeur soit 0, soit 1 (ou encore soit -1, soit 1). Un exemple dont la classe est 0 ou -1 est dénommé exemple négatif et un exemple dont la classe est 1 est dénommé exemple positif. Pour les autres problèmes de classification, ceux à plus de deux classes (où $|\mathcal{Y}| > 2$), les valeurs possibles de la variable réponse sont données par $\mathcal{Y}$. Il est à noter que les problèmes de classification à plus de deux classes peuvent être transformés en problèmes de classification binaires en entraînant un classificateur pour chaque classe versus toutes les autres classes. Il s'agit de la méthode un contre tous qui n'est généralement pas la meilleure. Il peut être avantageux de considérer des partitions plus complexes, mais mieux équilibrées. Par exemple, produire un classificateur qui discrimine les classes (0,1,2,3) de (4,5,6,7), un autre qui discrimine (0,2,4,6) de (1,3,5,7), et ainsi de suite.

La forme du tableau s'applique aussi bien à celle des données d'entraînement qu'à celle des données test. Nous désignons l'ensemble des données test par T. Lors de l'utilisation de celles-ci, la classe de l'exemple est utilisée seulement après qu'un classificateur eut tenté de classifier l'exemple. Les détails de l'utilisation des classes des données test sont donnés à la section 2.4.

Nous sommes dans le domaine de l'apprentissage supervisé, car des étiquettes y des exemples $\mathbf{x}$ sont disponibles. On suppose celles-ci issues d'une distribution conditionnelle $\mathbf{P}_{Y|\mathbf{X}}(y) = \mathbf{P}(Y = y|\mathbf{X} = \mathbf{x})$ de Y selon l'objet d'entrée observé.

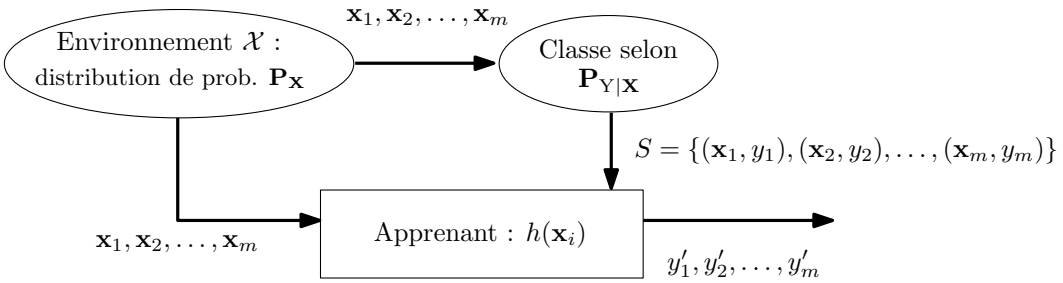

FIG. 2.1 – Apprentissage supervisé

La figure 2.1 montre le scénario classique de l'apprentissage supervisé [10]. Les données issues de l'environnement sont étiquetées selon une distribution $\mathbf{P}_{Y|\mathbf{X}}$ inconnue de l'apprenant. Celui-ci reçoit l'échantillon d'apprentissage S avec lequel il tente d'établir un lien de causalité entre les entrées et les sorties. La fonction h, appelée *hypothèse*, est le classificateur résultant et nous avons $y'_i \stackrel{\text{def}}{=} h(\mathbf{x}_i)$.

Nous avons donc un algorithme d'apprentissage A qui reçoit en entrée un échantillon d'exemples S et produit une hypothèse h. Nous notons cet état de choses par $A(S) = h$.

2.2.2 Données utilisées

Les tests empiriques effectués au cours de cette maîtrise ont pu être menés sur des données provenant de situations de la vie courante. L'école Donald Bren d'information et d'informatique de l'Université de Californie à Irvine offre sur Internet un accès à un dépôt de bases de données dont le but est primordialement de servir à analyser des algorithmes d'apprentissage automatique. Les données proviennent toutes du *UCI Machine Learning Repository* [11]. Leurs descriptions sont présentées dans les paragraphes suivants.

Sonar, Mines vs. Rocks [12] : Les variables explicatives (attributs 1 à 60) sont des mesures sur différentes bandes de fréquences d'échos radar provenant de cylindres métalliques ou de roches enfouies. La classe est 0 si l'objet est une roche ou 1 si c'est un cylindre.

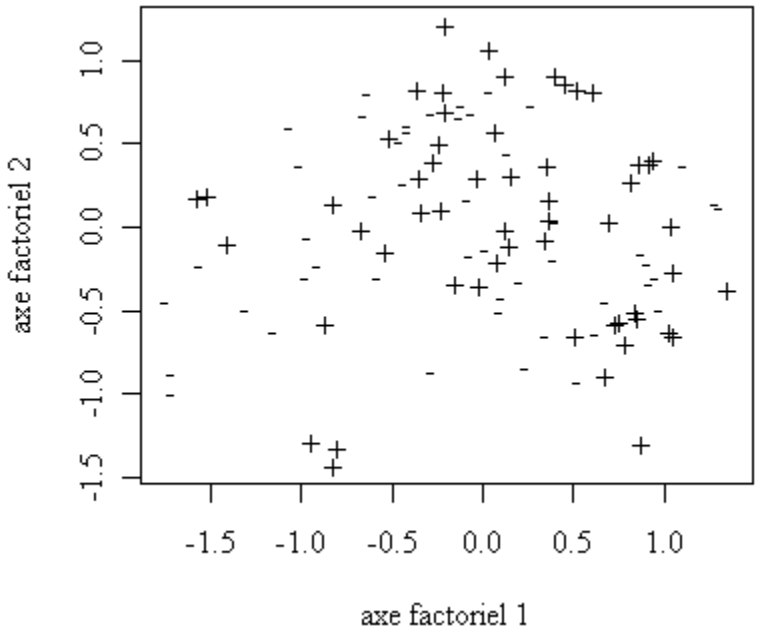

FIG. 2.2 – Axes factoriels principaux de *Sonar, Mines vs. Rocks*

Comme ces données ont été au centre de l'attention au cours de travaux effectués, la figure 2.2 permet sommairement de les visualiser en montrant les deux premiers axes factoriels[1] obtenus en leur appliquant une analyse en composantes principales [13].

Wisconsin Breast Cancer Database [14] : Les variables explicatives (attributs 1 à 9) sont des mesures prises sur des cellules de tumeurs. L'attribut 10 est la classe qui vaut 0 si la tumeur est bénigne ou 1 si elle est maligne.

1984 United States Congressional Voting Records Database [15] : Il y a 16 variables explicatives et chacune correspond au vote de sénateurs américains pour un projet politique. Le vote est +1 « yea », -1 « nay » ou 0 « unknown ». À chaque ligne est associé un sénateur. La classe est -1 s'il est démocrate ou 1 s'il est républicain.

Heart Disease [16] : Les variables explicatives sont l'âge, le sexe, le type de douleur à la poitrine, la pression artérielle au repos, le taux de cholestérol, si le taux de glucose dans le sang est supérieur à 120 mg/dl, le résultat de l'électrocardiogramme au repos, le pouls maximal, si l'exercice provoque de l'angine et d'autres mesures plus pointues d'un individu. La classe vaut 1 si la personne a une maladie du cœur ou elle vaut 0 dans le cas contraire.

Glass Identification Database [17] : La première variable explicative est l'index de réfraction d'un fragment de verre et les suivantes sont le pourcentage de divers éléments (sodium, magnésium, aluminium, etc.) qui le composent. La classe vaut 1 quand le verre provient d'un édifice et a été produit par flottage ou vaut 0 s'il s'agit d'un autre type de verre (provient d'un édifice et n'a pas été produit par flottage, provient d'un véhicule et a été produit par flottage ou non, provient d'un récipient, d'articles de table ou de phares).

Australian Credit [18] : Les variables explicatives sont des caractéristiques (de nature confidentielle) des clients d'une banque et la classe vaut 1 si le crédit est approuvé ou elle vaut -1 dans le cas contraire. Ces données ont été transformées avec la tangente hyperbolique de la valeur centrée réduite.

[1] Axes sur lesquels on projette les points et qui représentent les dimensions dans lesquelles on observe la plus grande variabilité.

2.3 Risque

Dans cette section nous entrons dans les détails de l'évaluation de la performance des classificateurs. Le risque est une mesure de performance communément acceptée. En fait, l'objectif principal des problèmes de classification est de minimiser le risque. Celui-ci est défini comme l'espérance d'une fonction de perte. Pour les problèmes de classification, la fonction de perte la plus couramment utilisée est la fonction de perte zéro-un l_z définie par :

$$l_z(y', y) = \begin{cases} 0 & \text{lorsque} \quad y' = y \\ 1 & \text{lorsque} \quad y' \neq y \end{cases}$$

où $y' \stackrel{\text{def}}{=} h(\mathbf{x})$ est la classe attribuée par le classificateur tandis que y est la classe désirée. En minimisant le risque au sens de la fonction de perte, plus souvent celle-ci vaudra 0, mieux ce sera. Le risque réel est défini ainsi :

$$R(h) \stackrel{\text{def}}{=} \mathbf{E}_{\mathbf{X},Y}[l(h(\mathbf{X}), Y)] = \int l(h(\mathbf{x}, y))\mathbf{P}_{\mathbf{X},Y}(\mathbf{x}, y)d\mathbf{x}dy$$

où $\mathbf{P}_{\mathbf{X},Y}(\mathbf{x}, y) = \mathbf{P}(\mathbf{X} = \mathbf{x} \wedge Y = y)$ est une densité conjointe inconnue définie sur $\mathcal{X} \times \mathcal{Y}$. C'est la distribution selon laquelle sont tirés les exemples des ensembles d'entraînement et de test. De plus, $\mathbf{P}(\mathbf{X} = \mathbf{x} \wedge Y = y) = \mathbf{P}(Y = y|\mathbf{X} = \mathbf{x})\mathbf{P}(\mathbf{X} = \mathbf{x})$.

2.4 Risque empirique

Étant donné que le fait d'ignorer $\mathbf{P}_{\mathbf{X},Y}(\mathbf{x}, y)$ implique que nous ne puissions pas avoir connaissance du risque réel, il faut se rabattre sur son estimateur, le risque empirique, pour pouvoir mesurer la performance d'un classificateur. Le risque empirique du classificateur h sur l'échantillon S est défini par :

$$R_S(h) = \frac{1}{m}\sum_{i=1}^{m} l(h(\mathbf{x}_i), y_i)$$

Pour choisir le classificateur h qui aura le plus petit risque réel en se basant sur l'information contenue dans S, on examine son risque empirique. La minimisation du risque empirique (*empirical risk minimisation*) est probablement le principe inductif le plus utilisé en apprentissage automatique. En considérant $\mathcal{H}$ comme étant l'espace de tous les classificateurs examinables par un algorithme d'apprentissage donné, nous avons la définition :

$$A_{\text{erm}}(S) = \underset{h \in \mathcal{H}}{\text{argmin}} R_S(h)$$

Parmi les autres principes inductifs, il y a le choix de l'hypothèse la plus probable étant donné S. C'est le principe de décision bayésienne. Si on définit *a priori* une distribution de probabilité sur $\mathcal{H}$, l'examen de S permet de modifier cette distribution et de choisir l'hypothèse la plus probable *a posteriori*. Un autre principe inductif consiste à choisir l'hypothèse qui comprime le mieux les informations contenues dans S. Il s'agit du principe de compression d'information. Suivre ce principe consiste à tenter d'éliminer toute redondance de S et d'en extraire les régularités qu'on présume aussi présentes dans $\mathcal{X} \times \mathcal{Y}$.

Le risque empirique est un estimateur non biaisé du vrai risque puisque :

$$\begin{aligned}
\mathbf{E}_S[R_S(h)] &= \mathbf{E}_S\left[\frac{1}{m}\sum_{i=1}^{m} l(h(\mathbf{X}_i), Y_i)\right] \\
&= \frac{1}{m}\mathbf{E}_S\left[\sum_{i=1}^{m} l(h(\mathbf{X}_i), Y_i)\right] \\
&= \frac{1}{m}\sum_{i=1}^{m} \mathbf{E}_{\mathbf{X}_i,Y_i}[l(h(\mathbf{X}_i), Y_i)] \\
&= \frac{1}{m}\sum_{i=1}^{m} \mathbf{E}_{\mathbf{X},Y}[l(h(\mathbf{X}), Y)] \\
&= \mathbf{E}_{\mathbf{X},Y}[l(h(\mathbf{X}), Y)] = R(h)
\end{aligned}$$

Dans les équations précédentes, les $\mathbf{X}_i$ ne doivent pas être confondus avec les X_j énoncés précédemment à la section 2.2.1, car $\mathbf{X}_i$ correspond au vecteur d'entrées du i^e exemple alors que X_j correspond au j^e attribut.

2.5 Intervalles de confiance

Lorsque nous avons obtenu les risques empiriques de différents classificateurs sur des données test communes, nous sommes alors intéressés à savoir si les différences observées sont significatives. Les intervalles de confiance nous donnent une borne inférieure et une borne supérieure entre lesquelles le vrai risque se trouve avec probabilité $1 - \delta$. Lorsque les intervalles de confiance de deux classificateurs se chevauchent, nous déduisons alors que leurs vrais risques ne sont pas significativement différents. Tandis que lorsque les intervalles ne se chevauchent pas, nous déduisons que les vrais risques sont significativement différents (avec un niveau de confiance $1 - \delta$).

Nous savons que la distribution de $mR_T(h)$ est une binomiale à $m = |T|$ tirages avec probabilité d'erreur $R(h)$. Les intervalles utilisés sont ceux produits à partir de l'inverse

de la queue de la binomiale. En théorie, ce sont les intervalles unilatéraux les plus serrés qu'il est possible d'obtenir pour un δ donné. Pour la binomiale, la probabilité de faire k erreurs est donnée par :

$$\mathbf{P}(mR_T(h) = k) = \binom{m}{k} R(h)^k (1 - R(h))^{m-k}$$

Tandis que la probabilité pour un classificateur de faire au plus k erreurs est donnée par la queue de la binomiale :

$$\mathbf{P}(mR_T(h) \leq k) = \text{Bin}(m, k, R_T(h)) \stackrel{\text{def}}{=} \sum_{i=0}^{k} \binom{m}{i} R_T(h)^i (1 - R_T(h))^{m-i}$$

La définition de l'inverse de la queue de la binomiale est donnée par :

$$\overline{\text{Bin}}(m, k, \delta) \stackrel{\text{def}}{=} \max\{r : \text{Bin}(m, k, r) \geq \delta\}$$

Il s'agit de la plus grande valeur de risque r telle que la probabilité est au moins δ de faire au plus k erreurs parmi m exemples. C'est la valeur utilisée comme borne supérieure ainsi qu'indiqué par le théorème suivant :

Proposition 1 (Langford (2005)) [19]
Pour tout classificateur h de risque $R(h)$ et pour tout $\delta \in (0, 1]$, nous avons :

$$\mathbf{P}_{\mathbf{Z}^m}\Big(R(h) \leq \overline{\text{Bin}}(m, mR_T(h), \delta)\Big) \geq 1 - \delta$$

La borne inférieure est calculée ainsi :

Proposition 2 (Langford (2005)) [19]
Pour tout classificateur h de risque $R(h)$ et pour tout $\delta \in (0, 1]$, nous avons :

$$\mathbf{P}_{\mathbf{Z}^m}\Big(R(h) \geq \min\{r : 1 - \text{Bin}(m, R_T(h), r) \geq \delta\}\Big) \geq 1 - \delta$$

2.6 Validation croisée

La validation croisée est utilisée pour évaluer la qualité de l'ajustement d'un modèle. Dans le cadre de cette maîtrise, nous évaluons les paramètres ajustables d'un algorithme d'apprentissage. La première étape de cette méthode consiste à subdiviser les données d'apprentissage en k sous-ensembles disjoints de même taille (autant que faire se peut).

Pour un sous-ensemble, un classificateur est construit à partir des $k-1$ autres sous-ensembles. Un risque empirique est alors calculé à partir du sous-ensemble qui n'a pas servi pour l'entraînement. Ces étapes sont répétées k fois et, à la fin, nous calculons la moyenne des k risques empiriques.

Dans l'algorithme 2.1, la notation $A(S - T_i)$ signifie l'algorithme d'apprentissage appliqué aux données d'apprentissage auxquelles on a enlevé les exemples de T_i. Lorsque k ne divise pas la cardinalité de S, autrement dit, si $|S| \bmod k \neq 0$, nous construisons $|S| \bmod k$ ensembles de taille $\lceil |S| \div k \rceil$ et $k - |S| \bmod k$ ensembles de taille $\lfloor |S| \div k \rfloor$. Pour chaque paramètre essayé, nous obtenons un R^k_{CV}. Le paramètre choisi est celui qui a la plus faible valeur de R^k_{CV}.

Algorithme 2.1 Validation croisée

Entrées : S et k
Sorties : le R^k_{CV}
  Partitionner S en k ensembles disjoints $\{T_1, T_2, \ldots, T_k\}$
  pour tout $i \in \{1, 2, \ldots, k\}$ **faire**
    Construire $h \leftarrow A(S - T_i)$
    Évaluer $R_i \leftarrow R_{T_i}(h)$
  fin pour
  Calculer $R^k_{CV} \leftarrow \frac{1}{k} \sum_{i=1}^{k} R_i$
  retourner R^k_{CV}

2.7 Apprentissage par compression des données

Lorsqu'une partie des données d'apprentissage est utilisée dans la définition même d'un classificateur, nous disons qu'il y a compression des données. L'ensemble des points utilisé pour la construction du classificateur est dénommé ensemble de compression. Pour le représenter, nous avons recours au vecteur d'indices $\mathbf{i}$. Il s'agit des numéros des exemples dans l'ensemble des données d'apprentissage :

$$\mathbf{i} = (i_1, i_2, \ldots, i_d)$$

où $i_j \in \{1, \ldots, m\} \forall j$ et d est la taille de l'ensemble de compression. Nous avons $i_1 < i_2 < \ldots < i_d$. Tel que mentionné à la section 2.2.1, m est le nombre d'exemples dans les données d'apprentissage.

L'ensemble de compression est symbolisé par la notation $\mathbf{z_i}$. Deux exemples concrets d'algorithmes qui utilisent la compression des données sont les SCM et les SVM. Pour

les SCM, il s'agit des centres et des bords des boules et des trous qui composent le classificateur. Les SCM sont décrites de manière complète à la section 2.9.1. Dans le cas des SVM, les points qui composent l'ensemble de compression sont les vecteurs de support. Consulter la section 2.9.2 pour plus de détails.

Nous disons qu'un classificateur $A(S) = h$ est identifié par $\mathbf{z_i}$ et par un message d'information additionnelle σ si et seulement s'il existe une fonction de reconstruction $\mathcal{R}$ qui nous donne le classificateur h lorsque nous l'appliquons à $\mathbf{z_i}$ et $\sigma : \mathcal{R}(\mathbf{z_i}, \sigma) = h$. L'ensemble de compression et le message sont donnés par une fonction de compression $\mathcal{C} : (\mathbf{z_i}, \sigma) = \mathcal{C}(S)$. Ainsi, pour un algorithme d'apprentissage A qui utilise la compression de données, nous avons :

$$A(S) = \mathcal{R}(\mathcal{C}(S)) \quad \forall S$$

2.8 Borne sur le risque pour les classificateurs comprimant les données

Ayant en main un classificateur donné, nous sommes intéressés à quantifier sa capacité à avoir un faible risque réel, en se basant seulement sur l'information provenant des données d'apprentissage. Une façon de faire est de calculer une borne sur le risque [4] avec un certain seuil $1 - \delta$ qui représente la probabilité que la borne soit respectée. Autrement dit, la borne peut être violée avec une probabilité d'au plus δ. Voici tout d'abord la notation utilisée :

- $\mathcal{I}$ dénote l'ensemble de tous les vecteurs $\mathbf{i}$ possibles. Il y a 2^m vecteurs $\mathbf{i}$ possibles lorsque l'échantillon d'apprentissage contient m exemples.
- $\mathcal{M}(\mathbf{z_i})$ dénote l'ensemble de tous les messages σ qu'il est possible de fournir avec un ensemble de compression $\mathbf{z_i}$ pour obtenir un classificateur $\mathcal{R}(\sigma, \mathbf{z_i})$. Nous verrons à la section 2.9.1 l'ensemble $\mathcal{M}(\mathbf{z_i})$ que nous utiliserons pour les SCM.
- $\bar{\mathbf{i}}$ est le complément de $\mathbf{i}$, c'est-à-dire que $\bar{\mathbf{i}}$ dénote la séquence des indices non présents dans $\mathbf{i}$. En d'autres mots, $\bar{\mathbf{i}}$ pointe sur les exemples qui ne font pas partie de l'ensemble de compression $\mathbf{z_i}$.
- $\mathbf{Z}^m$ est la variable aléatoire qui représente l'échantillon d'apprentissage.
- $\mathbf{Z_i}$ est la partie de $\mathbf{Z}^m$ qui est dans l'ensemble de compression.
- $\mathbf{Z}_{\bar{\mathbf{i}}}$ est la partie de $\mathbf{Z}^m$ qui n'est pas dans l'ensemble de compression.
- $R(\mathcal{R}(\sigma, \mathbf{z_i}))$ est le vrai risque du classificateur $\mathcal{R}(\sigma, \mathbf{z_i})$.

Pour calculer la borne, nous utilisons une distribution $P_{\mathcal{M}(\mathbf{z_i})}(\sigma)$ *a priori* sur les messages satisfaisant

$$\sum_{\sigma \in \mathcal{M}(\mathbf{z_i})}^{m} P_{\mathcal{M}(\mathbf{z_i})}(\sigma) \leq 1$$

La distribution spécifique utilisée avec les SCM sera énoncée à la section 2.9.1.

Proposition 3 (Marchand et Sokolova (2005)) [1]
Pour toute fonction de reconstruction $\mathcal{R}$ nous donnant un classificateur à partir d'un ensemble de compression et d'un message d'information supplémentaire, pour toute distribution de messages $P_{\mathcal{M}(\mathbf{z_i})}(\sigma)$, pour tout $\delta \in (0, 1]$:

$$\mathbf{P}_{\mathbf{Z}^m}\Bigg(\forall \mathbf{i} \in \mathcal{I}, \forall \sigma \in \mathcal{M}(\mathbf{Z_i}) : R(\mathcal{R}(\sigma, \mathbf{Z_i})) \leq \epsilon\Big(m, (m - |\mathbf{i}|)R_{\mathbf{Z}_{\bar{\mathbf{i}}}}(\mathcal{R}(\sigma, \mathbf{Z_i})), \sigma, \mathbf{Z_i}, \delta\Big)\Bigg) \geq 1 - \delta$$

où $\epsilon(m, k, \sigma, \mathbf{z_i}, \delta)$ est défini par :

$$\epsilon(m, k, \sigma, \mathbf{z_i}, \delta) = 1 - \exp\left(\frac{-1}{m - |\mathbf{i}| - k}\left[\ln\binom{m}{|\mathbf{i}|} + \ln\binom{m - |\mathbf{i}|}{k} + \ln\left(\frac{1}{P_{\mathcal{M}(\mathbf{z_i})}(\sigma) \cdot \zeta(|\mathbf{i}|) \cdot \zeta(k) \cdot \delta}\right)\right]\right)$$

et où ζ satisfait

$$\sum_{a=0}^{m} \zeta(a) \leq 1$$

Dans la première fraction à l'intérieur de l'exponentielle, $|\mathbf{i}|$ et k ont une influence symétrique sur la borne : plus ils sont petits, plus la borne est petite. Dans le premier logarithme, $|\mathbf{i}|$ fait diminuer la borne lorsqu'il tend vers 0 ou vers m, et il la fait augmenter lorsqu'il tend vers $\frac{m}{2}$. Le comportement conséquent au deuxième coefficient binomial est semblable. La borne diminue lorsque le ratio $\frac{k}{m-|\mathbf{i}|}$ tend vers 0 ou 1, et elle augmente lorsque le ratio tend vers $\frac{1}{2}$. Nous cherchons une borne la plus petite possible, car une borne basse indique que le risque réel doit être bas lui aussi.

La fonction $\zeta(a)$ utilisée pendant les travaux effectués au cours de cette maîtrise est la suivante :

$$\zeta(a) = \frac{6}{\pi^2}(a + 1)^{-2} \;\; \forall a \in \mathbb{N}$$

Pour $|\mathbf{i}| = 0$ et $k = 0$, le produit $\zeta(|\mathbf{i}|) \cdot \zeta(k)$ vaut environ 0.37 et sa valeur va en décroissant vers zéro à mesure que $|\mathbf{i}|$ et k augmentent. L'effet sur la borne est proportionnel à la grandeur de $|\mathbf{i}|$ et k.

Ainsi, la borne sur le risque sera d'autant plus petite que la taille de l'ensemble de compression du classificateur est petite et que son risque empirique[2] est petit. Le δ a typiquement une valeur de 0.05.

2.9 Classificateurs

Dans cette section sont présentés les types de classificateurs qui nous serviront d'éléments de comparaison. Au cours de ce travail, des classificateurs ont été produits avec de nouveaux algorithmes. Nous comparerons leurs performances avec celles de SCM et de SVM. Les SCM donnent de bons résultats pour la classification dans des espaces à faible et moyenne dimensionnalité. L'algorithme pour la construction de SCM a servi de point de départ aux deux versions du *branch-and-bound* (chapitre 3) et au *bagging* (chapitre 4). Les SVM sont des classificateurs performants dont l'usage est répandu.

2.9.1 Les SCM

L'ensemble de classificateurs désigné par *Set Covering Machines* (SCM) fut proposé en 2001 par Mario Marchand et John Shawe-Taylor [2]. La version dont il est ici question a été présentée en 2002 [3]. L'algorithme qui construit une SCM tente de couvrir l'ensemble des exemples de l'une des deux classes. Nous allons d'abord expliquer comment il tente de couvrir l'ensemble des exemples négatifs en utilisant un nombre minimal de partitions de cet ensemble.

Nous avons à notre disposition un ensemble fini $\mathcal{G} = \{g_1, g_2, \ldots, g_{|\mathcal{G}|}\}$ de classificateurs binaires que nous appelons caractéristiques. Chaque caractéristique g_i est une fonction à valeur Booléenne : $g_i(\mathbf{x}) \in \{0, 1\} \quad \forall \mathbf{x} \in \mathcal{X}$ et $\forall g_i \in \mathcal{G}$.

[2] Le risque empirique est ici la proportion d'erreurs que fait le classificateur sur l'ensemble des exemples de l'ensemble d'apprentissage qui ne sont pas dans l'ensemble de compression.

Une SCM est une conjonction ou une disjonction de caractéristiques. Lorsque ce sont les exemples négatifs qui sont couverts, il s'agit d'une conjonction. Pour tout exemple $\mathbf{x} \in \mathcal{X}$, la classe $c(\mathbf{x}) \in \{0, 1\}$ retournée par une conjonction c de caractéristiques est donnée par :

$$c(\mathbf{x}) = g_{i_1}(\mathbf{x}) \wedge g_{i_2}(\mathbf{x}) \wedge \ldots \wedge g_{i_k}(\mathbf{x})$$

Notre échantillon d'apprentissage S est constitué d'un ensemble P d'exemples positifs et d'un ensemble N d'exemples négatifs, de sorte que $P \cup N = S$. Chercher la couverture minimale de N est équivalent à chercher la plus petite conjonction qui ne fait aucune erreur sur S. Les caractéristiques utilisables dans une telle conjonction sont celles qui ne font aucune erreur avec les exemples de P.

Pour une caractéristique g_i, le sous-ensemble de N qu'elle couvre est noté Q_i. L'ensemble des exemples couverts par une conjonction c est donné par $(\cap_{i=i_1}^{i_k} Q_i')'$.

- Q_i' est l'ensemble des exemples $\mathbf{x}$ pour lesquels $g_i(\mathbf{x}) = 1$.
- $\cap_{i=i_1}^{i_k} Q_i'$ est l'ensemble des exemples $\mathbf{x}$ pour lesquels $c(\mathbf{x}) = 1$. Car les « et » logiques entre les g_i se traduisent par des intersections entre les Q_i'.
- $(\cap_{i=i_1}^{i_k} Q_i')'$ est l'ensemble des exemples $\mathbf{x}$ pour lesquels $c(\mathbf{x}) = 0$.

En appliquant la loi de de Morgan, nous avons $(\cap_{i=i_1}^{i_k} Q_i')' = \cup_{i=i_1}^{i_k} Q_i$. Ceci montre pourquoi la recherche de la plus petite conjonction d'erreur nulle sur S équivaut à la recherche du recouvrement de N qui utilise le moins possible de Q_i.

Ce problème NP-complet est appelé *Minimum Set Cover Problem* [6]. Cependant, l'algorithme glouton possède une bonne garantie lorsqu'il existe un petit nombre de sous-ensembles couvrant N. Selon cet algorithme, la caractéristique choisie pour être ajoutée à la conjonction est celle qui couvre le plus d'exemples négatifs qui ne sont pas déjà couverts par la conjonction. Des caractéristiques sont ajoutées jusqu'à ce que tous les exemples négatifs soient recouverts. La garantie de l'algorithme glouton est un maximum pour la taille de la conjonction. La proposition 4 rapporte ce résultat.

Proposition 4 (Marchand) [1]
Si, parmi un ensemble $\mathcal{G}$ de caractéristiques, il existe une conjonction de r caractéristiques classifiant correctement tout l'ensemble $S = P \cup N$ d'exemples positifs et négatifs, alors l'algorithme glouton formera une conjonction d'au plus $r \ln(|N|)$ caractéristiques (peu importe la taille de $\mathcal{G}$).

Généralement, il est possible de trouver une conjonction plus compacte que celle d'erreur nulle sur S et qui a une meilleure garantie de performance. Une première façon de restreindre la taille d'une conjonction est de limiter son nombre de caractéristiques. Dans ce cas, l'hypothèse ainsi créée fait des erreurs de classification sur les exemples négatifs non couverts. L'autre façon est de permettre dans la conjonction des caractéristiques qui font des erreurs sur les exemples positifs. Dans ce cas, la taille de la conjonction est réduite si les caractéristiques choisies peuvent couvrir considérablement plus d'exemples négatifs en faisant des erreurs sur les exemples positifs plutôt qu'en n'en faisant pas.

Pour une caractéristique g_i, le sous-ensemble de P qu'elle classifie incorrectement est noté R_i. Permettre les erreurs sur P requiert une modification de l'algorithme glouton. Une quantité supplémentaire est nécessaire pour évaluer g_i. Il s'agit de l'utilité U_i calculée à partir de l'ensemble des exemples négatifs couverts $Q_i \subset N$ et de l'ensemble des exemples positifs classifiés incorrectement $R_i \subset P$:

$$U_i = |Q_i| - p|R_i|$$

où p est la pénalité de faire une erreur avec un exemple de P. Plus la pénalité est grande, moins les erreurs sur P sont tolérées. Nous avons $p \in \mathbb{R}_+$.

L'algorithme 2.2, ou algorithme glouton modifié, ajoute à la conjonction, à chaque itération, un g_i de U_i maximal au lieu d'ajouter un g_i de $|Q_i|$ maximal. Les Q_i et les R_i ne sont pas reçus en paramètres mais plutôt déduits à partir des données. À chaque étape,

Algorithme 2.2 SCM-classique

Entrées : P l'ensemble des exemples positifs
N l'ensemble des exemples négatifs
p la pénalité
$maxCarac$ le nombre maximal de caractéristiques

Sorties : une SCM C

$C \leftarrow \emptyset$
tant que $N \neq \emptyset$ **et** $|C| < maxCarac$ **faire**
$g_i \leftarrow g_j$ de U_j maximal
$C \leftarrow C \cup \{g_i\}$
$N \leftarrow N - Q_i$
$Q_j \leftarrow Q_j - Q_i \ \forall Q_j$
$P \leftarrow P - R_i$
$R_j \leftarrow R_j - R_i \ \forall R_j$
fin tant que
retourner C

les exemples négatifs couverts par la caractéristique choisie sont enlevés de l'ensemble des exemples négatifs et de chaque Q_j. Pour que les erreurs sur les positifs ne soient pas comptées pour plusieurs caractéristiques, les exemples positifs mal classifiés sont enlevés de l'ensemble des positifs et de chaque R_j. L'algorithme se termine lorsque N est vide ou que le nombre maximal de caractéristiques dans la conjonction a été atteint.

Les paramètres p et $maxCarac$ sont choisis par validation croisée ou par la borne de la section 2.8. Dans ce cas, plusieurs SCM sont construites en utilisant différents paramètres p et $maxCarac$. Celle qui est choisie est celle pour laquelle nous calculons la plus petite valeur de la borne.

Les caractéristiques utilisées dans les SCM sont représentées par un très petit nombre d'exemples de l'échantillon d'apprentissage. Ce type de caractéristique se nomme « caractéristique dépendante des données ». Dans le cadre de cette maîtrise, les seules caractéristiques utilisées sont les boules et les trous. À partir de ce point, $\mathcal{G}$ désigne l'ensemble de ces caractéristiques. Chaque caractéristique $g \in \mathcal{G}$ est identifiée par son centre $\mathbf{x}_c$ et son rayon ρ. Avec celles-ci, nous avons besoin d'une métrique $d : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}_+$. La distance entre les points $\mathbf{x}$ et $\mathbf{x}'$ est donnée par $d(\mathbf{x}, \mathbf{x}')$. La seule métrique utilisée au cours des présents travaux est la métrique euclidienne.

Si g est une boule, on retourne 1 lorsque l'exemple $\mathbf{x} \in \mathcal{X}$ passé en paramètre est à l'intérieur et on retourne 0 lorsqu'il est à l'extérieur.

$$g(\mathbf{x}) = \begin{cases} 1 & \text{lorsque} \quad d(\mathbf{x}, \mathbf{x}_c) \leq \rho \\ 0 & \text{lorsque} \quad d(\mathbf{x}, \mathbf{x}_c) > \rho \end{cases} \tag{2.1}$$

Si g est un trou, on retourne 0 lorsque l'exemple $\mathbf{x} \in \mathcal{X}$ passé en paramètre est à l'intérieur et on retourne 1 lorsqu'il est à l'extérieur.

$$g(\mathbf{x}) = \begin{cases} 0 & \text{lorsque} \quad d(\mathbf{x}, \mathbf{x}_c) \leq \rho \\ 1 & \text{lorsque} \quad d(\mathbf{x}, \mathbf{x}_c) > \rho \end{cases} \tag{2.2}$$

Pour créer une boule à partir des données d'apprentissage, un exemple positif $\mathbf{x}_c$ est choisi comme centre et un autre exemple positif $\mathbf{x}_b$ est choisi comme point frontière, ou bord. Cette paire de points fera partie de l'ensemble de compression de la SCM si la boule y est ajoutée. Le rayon ρ de la boule est défini par $d(\mathbf{x_c}, \mathbf{x}_b) + \epsilon$. Il est légèrement plus grand que la distance entre les deux exemples pour que le point frontière soit classifié positif par la boule. C'est un premier pas pour que la SCM ne fasse pas d'erreur de classification avec son propre ensemble de compression.

Pour créer un trou à partir des données d'apprentissage, un exemple négatif $\mathbf{x}_c$ est choisi comme centre et un exemple positif $\mathbf{x}_b$ est choisi comme point frontière. Le rayon ρ du trou est défini par $d(\mathbf{x_c}, \mathbf{x}_b) - \epsilon$. Il est légèrement plus petit que la distance entre les deux exemples pour que le point frontière soit classifié positif par le trou.

Les SCM utilisées au cours de cette maîtrise ont été construites de manière à ce qu'elles ne fassent pas d'erreur de classification avec les exemples compris dans leurs ensembles de compression. Ce concept est dénommé la « consistance avec l'ensemble de compression ». Pour nous en assurer, nous limitons, à chaque itération de l'algorithme glouton, le choix de caractéristiques à celles qui classifient correctement les exemples positifs déjà présents dans $\mathbf{z_i}$.

Une SCM n'est pas décrite que par son ensemble de compression. Les exemples contenus dans celui-ci donnent une information insuffisante pour reconstruire une conjonction. Il est certain que les exemples négatifs qui en font partie sont des centres. Cependant, nous ne savons pas quels exemples positifs sont des centres ni lesquels sont des bords. C'est à l'aide d'un message d'information supplémentaire σ (section 2.7) que nous les identifions.

Il est possible qu'un exemple positif de l'ensemble de compression soit simultanément le centre d'une caractéristique et le point frontière d'une autre ; de même qu'il est possible qu'il soit le point frontière de plusieurs caractéristiques. Cependant, il ne peut pas être le centre de plusieurs boules. De plus, un exemple négatif ne peut pas être le centre de plusieurs trous. Ne pas forcer les centres uniques irait de pair avec des inconsistances dans l'ensemble de compression.

L'information nécessaire et suffisante pour reconstruire une conjonction à partir d'un ensemble de compression est l'identification de ses exemples positifs qui sont des points frontière sans être des centres. L'ensemble de ces exemples est noté $B(\mathbf{z_i})$. Le sous-ensemble des exemples positifs de $\mathbf{z_i}$ est noté $P(\mathbf{z_i})$ et nous avons $B(\mathbf{z_i}) \subseteq P(\mathbf{z_i})$. Le sous-ensemble des exemples négatifs de $\mathbf{z_i}$ est noté $N(\mathbf{z_i})$. Pour tout centre de boule $\mathbf{z}_c \in P(\mathbf{z_i}) - B(\mathbf{z_i})$, son point frontière est l'exemple $\mathbf{z}_b \in P(\mathbf{z_i})$ qui en est le plus loin selon la métrique d. Pour tout centre de trou $\mathbf{z}_c \in N(\mathbf{z_i})$, son point frontière est l'exemple $\mathbf{z}_b \in P(\mathbf{z_i})$ qui en est le plus près selon la même métrique. Procéder autrement causerait des inconsistances dans l'ensemble de compression.

Nous notons la cardinalité de $B(\mathbf{z_i})$ par $b(\mathbf{z_i})$ et celle de $P(\mathbf{z_i})$ par $p(\mathbf{z_i})$. Le message d'information supplémentaire σ donne le nombre $b(\sigma) = b(\mathbf{z_i})$ de points frontières qui ne sont pas aussi des centres dans $P(\mathbf{z_i})$. Il donne aussi le numéro $g(\sigma)$ du groupe de

points frontières parmi les groupes possibles de $b(\sigma)$ exemples parmi $p(\mathbf{z_i})$. Ainsi, nous savons quels exemples de $P(\mathbf{z_i})$ font partie de $B(\mathbf{z_i})$.

Pour choisir les paramètres p et $maxCarac$ de l'algorithme glouton modifié en utilisant la borne de la section 2.8, la distribution sur les messages d'information supplémentaire σ utilisée est

$$P_{\mathcal{M}(\mathbf{z_i})}(\sigma) = \zeta(b(\sigma)) \binom{p(\mathbf{z_i})}{b(\sigma)}^{-1}$$

où ζ est une fonction satisfaisant $\sum_{b=0}^{p(\mathbf{z_i})} \zeta(b) \leq 1$. Le terme $\zeta(b(\sigma))$ sert à accorder un poids arbitraire aux différentes tailles de $B(\mathbf{z_i})$. Tandis que le terme $\binom{p(\mathbf{z_i})}{b(\sigma)}^{-1}$ fait que la somme de $P_{\mathcal{M}(\mathbf{z_i})}(\sigma)$ sur les σ possibles est inférieure ou égale à 1. En effet :

$$\sum_{\sigma \in \mathcal{M}(\mathbf{z_i})} P_{\mathcal{M}(\mathbf{z_i})}(\sigma) = \sum_{b=0}^{p(\mathbf{z_i})} \zeta(b) \sum_{\sigma : b(\sigma)=b} \binom{p(\mathbf{z_i})}{b(\sigma)}^{-1} \leq 1$$

car

$$\begin{aligned} \sum_{\sigma : b(\sigma)=b} \binom{p(\mathbf{z_i})}{b(\sigma)}^{-1} &= \binom{p(\mathbf{z_i})}{b}^{-1} \sum_{\sigma : b(\sigma)=b} 1 \\ &= \binom{p(\mathbf{z_i})}{b}^{-1} \binom{p(\mathbf{z_i})}{b} \\ &= 1 \end{aligned}$$

Dans l'exemple de la figure 2.3, nous avons une SCM à deux caractéristiques : une boule et un trou. Le centre de la boule est l'exemple numéro 1 et son rayon est déterminé par l'exemple numéro 2. Le centre du trou est l'exemple négatif numéro 3 et son rayon est aussi déterminé par l'exemple numéro 2. Sur la figure, le rayon de la boule a été

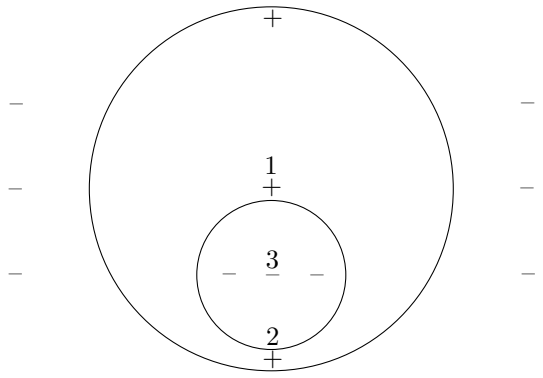

FIG. 2.3 – Exemple de SCM

exagéré pour montrer que l'exemple 2 est à l'intérieur de celle-ci. Le surplus de longueur du rayon provient du ϵ utilisé lors de la création de la boule. De même, le rayon du trou a été exagéré, dans la direction opposée, pour montrer que l'exemple 2 est à l'extérieur de celui-ci. Selon l'algorithme glouton, la boule est ajoutée en premier à la conjonction puisqu'elle couvre les six exemples négatifs à gauche et à droite, soit trois exemples de plus que le trou, qui ne couvre que les trois exemples négatifs du milieu. L'ensemble de compression ne comporte que trois exemples, car l'exemple numéro 2 est réutilisé. Le nombre de points frontières reporté dans σ est 1 et le nombre de groupes possibles est $\binom{2}{1} = 2$.

Comme nous l'avons mentionné au début de cette section, une SCM peut aussi être une disjonction de caractéristiques. Pour tout exemple $\mathbf{x} \in \mathcal{X}$, la classe $d(\mathbf{x}) \in \{0, 1\}$ retournée par une disjonction d de caractéristiques est donnée par :

$$d(\mathbf{x}) = g_{i_1}(\mathbf{x}) \vee g_{i_2}(\mathbf{x}) \vee \ldots \vee g_{i_k}(\mathbf{x})$$

Pour construire une telle SCM, il suffit tout d'abord de permuter la classe des exemples d'entraînement. Ensuite, nous leur appliquons un des algorithmes de construction d'une conjonction. Pour terminer, nous remplaçons chaque $\wedge$ de la conjonction obtenue par un $\vee$ et nous prenons le complément de chaque caractéristique.

L'implémentation de la classification à partir d'une SCM, dans le cas d'une conjonction, examine les caractéristiques une par une, s'arrête dès qu'il y en a une qui donne -1 et retourne cette classe. Si elles donnent toutes 1, alors c'est la classe retournée. Dans le cas d'une disjonction, l'examen des caractéristiques est interrompu dès qu'il y en a une qui donne 1 et c'est la classe retournée. Si elles donnent toutes -1 alors c'est la classe retournée.

2.9.2 Les SVM

Les *Support Vector Machines* (SVM) ont été proposées par Vladimir Vapnik, Bernhard Boser et Isabelle Guyon en 1992 [20]. Elles utilisent pour la classification un hyperplan qui sépare les exemples positifs des exemples négatifs. La construction d'une SVM répond à un problème d'optimisation. La quantité maximisée est la distance entre l'hyperplan et les exemples de classes opposées le plus près de chacun de ses côtés. Cette distance est appelée la marge. Elle est illustrée sur la figure 2.4. L'hyperplan y est représenté par la ligne pointillée.

Les vecteurs de supports (*support vectors*) sont les exemples qui sont à la même distance que la marge de l'hyperplan. Ce sont les exemples encerclés sur la figure 2.4. Ils

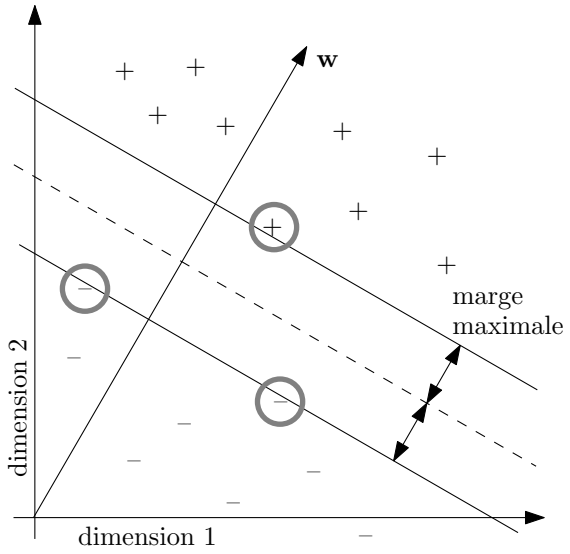

FIG. 2.4 – Exemple de SVM sans noyau

forment un ensemble de compression puisque si nous n'utilisions que ceux-ci comme ensemble d'apprentissage, nous retrouverions le même classificateur qu'en utilisant toutes les données d'apprentissage.

Une notion très utile lors de l'usage de SVM est les noyaux. Les fonctions noyau $k(\mathbf{x}, \mathbf{x}')$ sont utilisées par les SVM pour transformer non linéairement l'espace d'entrée $\mathcal{X}$ en un espace de redescription $\boldsymbol{\phi}(\mathcal{X})$. Cette transformation est notée :

$$\mathbf{x} = (x_1, \ldots, x_n) \;\mapsto\; \boldsymbol{\phi}(\mathbf{x}) = (\phi_1(\mathbf{x}), \ldots, \phi_n(\mathbf{x}), \ldots, \phi_{n'}(\mathbf{x}))$$

Le noyau $k(\mathbf{x}, \mathbf{x}')$ associé à la transformation $\boldsymbol{\phi}$ est donné par le produit scalaire $\boldsymbol{\phi}(\mathbf{x}) \cdot \boldsymbol{\phi}(\mathbf{x}')$. Notez que la dimension n' de l'espace de redescription est généralement grande et possiblement infinie.

$$k(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}(\mathbf{x}) \cdot \boldsymbol{\phi}(\mathbf{x}') = \sum_{i=1}^{n'} \phi_i(\mathbf{x})\phi_i(\mathbf{x}')$$

Les fonctions noyau utilisées dans la cadre de notre recherche sont le noyau polynomial d'ordre p :

$$k(\mathbf{x}, \mathbf{x}') = (\mathbf{x} \cdot \mathbf{x}' + 1)^p$$

et le noyau *Radial Basis Function* (RBF) [21] :

$$k(\mathbf{x}, \mathbf{x}') = \exp(-\gamma||\mathbf{x} - \mathbf{x}'||^2)$$

Avec une version précédente sans noyau, les SVM ne pouvaient que résoudre des problèmes linéairement séparables. Cependant, il existe une formulation du problème d'optimisation dans laquelle les seules manipulations faites avec les objets d'entrée $\mathbf{x}$ et $\mathbf{x}'$ sont des produits scalaires $\mathbf{x} \cdot \mathbf{x}'$. En appliquant $\boldsymbol{\phi}$ à tous les exemples, nous pouvons trouver un hyperplan qui passe par des dimensions absentes des données non transformées. Le produit scalaire entre exemples devient $\boldsymbol{\phi}(\mathbf{x}) \cdot \boldsymbol{\phi}(\mathbf{x}')$. Il est possible de réduire les calculs en substituant ce produit par une fonction noyau. Ainsi, des données non linéairement séparables initialement peuvent le devenir à condition de les redécrire avec suffisamment de dimensions.

La version utilisée au cours des présents travaux est celle avec marges floues publiée en 1995 par Corinna Cortes et Vapnik [5], car elle permet de faire des erreurs sur les exemples des données d'entraînement. Il existe également une version des SVM qui peut résoudre des problèmes de régression.

Nous pouvons contrôler le degré avec lequel l'algorithme qui construit une SVM est intolérant aux erreurs à l'aide d'un paramètre C. Les erreurs sont quantifiées par la distance à laquelle les exemples mal classifiés sont à l'intérieur des marges. Dans la résolution du problème d'optimisation, l'algorithme qui produit une SVM cherche à minimiser les erreurs. Plus C est grand, plus les erreurs « coûtent cher ».

Si nous voulons visualiser ce qui se produit lorsqu'on utilise un noyau, nous pouvons utiliser l'applet disponible à la page `http://svm.dcs.rhbnc.ac.uk/pagesnew/GPat.shtml`.

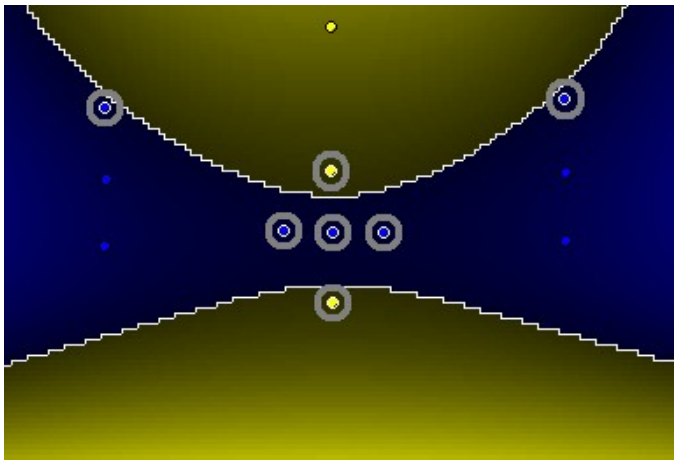

FIG. 2.5 – Exemple de SVM avec noyau

La figure 2.5 montre une SVM avec noyau polynomial appliquée à des données similaires à celle de la figure 2.3. Les vecteurs de support sont les exemples encerclés.

La raison pour laquelle nous voyons des courbes pour représenter l'hyperplan n'est pas que celui-ci est courbé, il est plat dans l'espace transformé par la fonction noyau.

L'implémentation des SVM utilisée est celle donnée par la version 0.4-2 du *package* kernlab du locigiel R [22]. Il existe plus d'une méthode pour implémenter la résolution du problème d'optimisation des SVM [23]. Il y a « sequential minimal optimisation » (SMO), « projected conjugate gradient chunking » et l'algorithme de décomposion d'Osuna. Celle utilisée [21] dans le *package* kernlab est une variante de SMO.

Pour déterminer les meilleurs paramètres de noyau et C, la validation croisée dix fois a été utilisée. Pour le noyau RBF, par exemple, deux boucles imbriquées ont servi pour tester toutes les combinaisons possibles pour $C = 1$, 10, 100, 1000 et 10000 et pour $\gamma = 0.001$, 0.01, 0.1, 1, 10, 100 et 1000. Ensuite, des valeurs intermédiaires ont été essayées pour tenter de trouver un minimum local au R^k_{CV}.

2.9.3 Les votes de majorité

Plus loin dans ce mémoire, nous présenterons certains résultats empiriques issus de votes de majorité. Nous serons alors intéressés à savoir ce qui cause la qualité ou la défaillance d'un vote particulier. Cette section traite des facteurs qui seront examinés et elle introduit les quantités élémentaires utilisées.

Désignons tout d'abord l'ensemble fini des classificateurs pouvant participer à un vote de majorité spécifique par $\mathcal{H}$. Soit Q une distribution sur $\mathcal{H}$ postérieure[3] qui pondère chaque $h \in \mathcal{H}$. La technique que nous utiliserons pour construire la distribution Q aura pour effet de produire une distribution uniforme sur un sous-ensemble de $\mathcal{H}$. Un classificateur par vote de majorité est aussi nommé un classificateur de Bayes. La classe retournée par un tel classificateur B_Q pour un exemple $\mathbf{x}$ est donnée par :

$$B_Q(\mathbf{x}) \stackrel{\text{def}}{=} I\left(\left(\mathop{\mathbf{E}}_{h\sim Q} h(\mathbf{x})\right) \geq \frac{1}{2}\right)$$

où $I(a) = 1$ si a est vrai et 0 sinon. Notez bien que lorsqu'il y a égalité, nous avons décidé que le vote est positif.

Nous pouvons aussi utiliser les classificateurs de $\mathcal{H}$ pour constituer un classificateur de Gibbs G_Q. La classe retournée par celui-ci est la classe déterminée par un h tiré au hasard selon la distribution Q. Le risque $R(G_Q)$ d'un classificateur de Gibbs est donné

[3] Une distribution postérieure utilise l'information présente dans les données d'apprentissage.

par :

$$R(G_Q) \stackrel{\text{def}}{=} \mathop{\mathbf{E}}_{h\sim Q} R(h) = \mathop{\mathbf{E}}_{h\sim Q}\left[\mathop{\mathbf{E}}_{(\mathbf{x},y)\sim P_{\mathbf{X},Y}} I(h(\mathbf{x}) \neq y)\right]$$

tandis que son risque empirique $R_T(G_Q)$, sur un échantillon T de m exemples, est donné par :

$$R_T(G_Q) \stackrel{\text{def}}{=} \mathop{\mathbf{E}}_{h\sim Q} R_T(h) = \mathop{\mathbf{E}}_{h\sim Q}\left[\frac{1}{m}\sum_{i=1}^{m} I(h(\mathbf{x}_i) \neq y_i)\right]$$

Pour un exemple $\mathbf{x}$ de classe y, nous définissons la proportion de classificateurs faisant une erreur comme :

$$W_Q(\mathbf{x}, y) \stackrel{\text{def}}{=} \mathop{\mathbf{E}}_{h\sim Q} I(h(\mathbf{x}) \neq y)$$

Notons par H l'ensemble des classificateurs effectivement utilisés pour un vote de majorité. Nous avons $H \subseteq \mathcal{H}$. La proposition 5 établit un lien entre le risque de Bayes, le risque de Gibbs et $W_Q(\mathbf{x}, y)$ qui ont été définis précédemment :

Proposition 5 (Germain, Lacasse, Laviolette, Marchand et Usunier (2006)) [24]
Pour tout ensemble $\mathcal{H}$ de classificateurs et pour toute distribution Q sur $\mathcal{H}$, si $R(G_Q) \leq 1/2$, alors nous avons :

$$R(B_Q) \leq C_Q \stackrel{\text{def}}{=} \frac{\mathop{\mathbf{Var}}_{(\mathbf{x},y)\sim P_{\mathbf{X},Y}} W_Q(\mathbf{x}, y)}{\mathop{\mathbf{Var}}_{(\mathbf{x},y)\sim P_{\mathbf{X},Y}} W_Q(\mathbf{x}, y) + (1/2 - R(G_Q))^2} = \frac{1}{1 + \frac{(1/2 - R(G_Q))^2}{\mathop{\mathbf{Var}}_{(\mathbf{x},y)\sim P_{\mathbf{X},Y}} W_Q(\mathbf{x},y)}}$$

Nous utiliserons également l'estimation empirique $\widehat{C_Q}$ de C_Q sur un échantillon T de m exemples :

$$\widehat{C_Q} = \frac{1}{1 + \frac{(1/2 - R_T(G_Q))^2}{\frac{1}{m-1}\sum_{i=1}^{m}\left(\overline{W}_Q - \widehat{W}_Q(\mathbf{x}_i, y_i)\right)^2}}$$

où $\widehat{W}_Q(\mathbf{x}, y) = \frac{1}{|H|}\sum_{h\in H} I(h(\mathbf{x}) \neq y)$

et $\overline{W}_Q = \frac{1}{m}\sum_{j=1}^{m} \widehat{W}_Q(\mathbf{x}_j, y_j)$

La figure 2.6 illustre les résultats analysés. La ligne i correspond à un exemple $\mathbf{x}_i \in T$ et la colonne j correspond à un classificateur $h_j \in H$. Donc, la case (i, j) donne $h_j(\mathbf{x}_i)$.

	h_1			$h_{\|H\|}$
exemple 1	1	0	$\cdots$	0
	1	1	$\cdots$	1
	$\vdots$	$\vdots$	$\ddots$	$\vdots$
exemple m	1	0	$\cdots$	1

FIG. 2.6 – Votes des classificateurs

Nous notons un vecteur colonne, ou vecteur de votes, de la figure 2.6 par :

$$h_j(T) = (h_j(\mathbf{x}_1), h_j(\mathbf{x}_2), \ldots, h_j(\mathbf{x}_m))$$

Pour évaluer le degré de similarité entre deux vecteurs de votes, nous calculons un coefficient de corrélation échantillonnal de Pearson r :

$$r(h_{j_1}(T), h_{j_2}(T)) = \frac{m \sum_{i=1}^{m} h_{j_1}(\mathbf{x}_i) h_{j_2}(\mathbf{x}_i) - \sum_{i=1}^{m} h_{j_1}(\mathbf{x}_i) \sum_{i=1}^{m} h_{j_2}(\mathbf{x}_i)}{\sqrt{m \sum_{i=1}^{m} h_{j_1}^2(\mathbf{x}_i) - (\sum_{i=1}^{m} h_{j_1}(\mathbf{x}_i))^2} \sqrt{m \sum_{i=1}^{m} h_{j_2}^2(\mathbf{x}_i) - (\sum_{i=1}^{m} h_{j_2}(\mathbf{x}_i))^2}}$$

Nous obtenons un indicateur global de similarité pour un ensemble test en calculant la moyenne des r de toutes les paires possible de vecteurs de votes.

$$\bar{r} = \frac{1}{\binom{|H|}{2}} \sum_{\substack{h_{j_1}, h_{j_2} \in H \\ j_1 < j_2}} r(h_{j_1}(T), h_{j_2}(T))$$

2.10 Conclusion

Les concepts utiles pour la compréhension des travaux effectués au cours de cette maîtrise sont maintenant frais à notre esprit et facilement accessibles si nécessaire. Pour faciliter la navigation dans ce mémoire, les sections pertinentes seront données en référence lorsque nécessaires. Comme beaucoup du contenu présenté dans ce chapitre est inspiré des notes de cours d'apprentissage automatique [1], un regard sur celles-ci est surérogatoire.

Chapitre 3

Branch-and-bound

3.1 Introduction

Le *branch-and-bound* est une technique utilisée pour résoudre des problèmes d'optimisation. Il fait partie des méthodes d'énumération implicite. Ces méthodes considèrent toutes les solutions pour trouver celle qui est optimale, mais elles enlèvent de la recherche des ensembles de solutions pour lesquels il y a des garanties que la solution optimale ne s'y trouve pas. Les solutions sont représentées dans un graphe implicite orienté. Celui-ci est habituellement acyclique et peut être un arbre. Pour le cas présent, il s'agit d'un arbre.

Le *branch-and-bound* peut être utilisé pour trouver des solutions à des problèmes NP-difficiles tels celui du sac à dos et celui du voyageur de commerce. Pour ce dernier problème, une version hybride du *branch-and-bound*, le *branch-and-cut*, a été utilisée avec succès sur une instance de 33810 points sur une carte de circuits [25]. Pour n villes, ou points, le nombre de combinaisons possibles est $n!$ lorsque l'on visite une seule fois toutes les villes. 33810! est un très grand nombre. C'est 1.529×10^{138446}, tandis que le nombre d'atomes dans l'univers est estimé à 10^{79}.

Nous pouvons représenter avec un arbre toutes les SCM constructibles depuis un ensemble d'apprentissage. Au niveau zéro, il y a la racine et au niveau un, les classificateurs à une caractéristique. Au niveau deux, il y a les classificateurs à deux caractéristiques et ainsi de suite. La suite de nœuds à partir d'un nœud quelconque au niveau i jusqu'à la racine correspond à un classificateur à i caractéristiques. À chaque nœud peut être associée une valeur de la borne sur le risque de la proposition 3 calculée pour la SCM dont les caractéristiques sont le nœud courant et ses parents. Un *branch-and-bound* est

une technique appropriée pour explorer un tel arbre puisque la quantité minimisée, la fonction de coût f présentée ultérieurement, présente les qualités requises. Le nombre de nœuds dans l'arbre à explorer pour le problème d'optimisation de la borne est lui aussi très grand.

Si nous procédions par la force brute et tentions de visiter tous les nœuds, il faudrait se préparer, en étant pessimiste, à attendre un temps $\in O((m^2)!)$. Comme il y a m^2 paires d'exemples possibles dans l'ensemble d'entraînement, le nombre de caractéristiques qu'il est possible de construire à partir de ceux-ci est dans $O(m^2)$. Il n'y en a pas exactement m^2, car même dans le cas où nous pouvons utiliser des exemples positifs comme centres, pour une conjonction, les exemples négatifs ne servent jamais comme bords. Si au premier niveau il y a au plus m^2 nœuds, alors au deuxième niveau, il y en a au plus $m^2 \cdot (m^2 - 1)$, car une caractéristique est utilisée au plus une fois dans une hypothèse. Au troisième niveau, il y aura au plus $m^2 \cdot (m^2 - 1) \cdot (m^2 - 2)$ nœuds. En continuant de cette manière, nous arrivons à la conclusion qu'au dernier niveau il y aura, si on ne limite pas le nombre de niveaux, au plus $(m^2)!$ nœuds. La somme maximale A des nombres de nœuds de chaque niveau est dans $O((m^2)!)$ car :

$$\begin{aligned} A &= m^2 + m^2 \cdot (m^2 - 1) + m^2 \cdot (m^2 - 1) \cdot (m^2 - 2) + \cdots + (m^2)! \\ &= \frac{(m^2)!}{(m^2 - 1)!} + \frac{(m^2)!}{(m^2 - 2)!} + \cdots + \frac{(m^2)!}{(m^2 - (m^2 - 1))!} \\ &= \sum_{i=1}^{m^2-1} \frac{(m^2)!}{(m^2 - i)!} \\ &= (m^2)! \sum_{i=1}^{m^2-1} \frac{1^i}{(i)!} \end{aligned}$$

et nous avons

$$\lim_{m \to \infty} \frac{A}{(m^2)!} = \lim_{m \to \infty} \sum_{i=1}^{m^2-1} \frac{1^i}{(i)!} = \sum_{i=1}^{\infty} \frac{1^i}{(i)!} = e - 1$$

en utilisant

$$\sum_{n=0}^{\infty} \frac{a^n}{n!} = e^a$$

C'est donc dire que l'espace de recherche est immense, bien que les hypothèses ne contiennent jamais toutes les caractéristiques possibles, puisque la profondeur de l'arbre est limitée par le paramètre *maxCarac* présenté un peu plus loin. Dans le cas du *branch-and-bound* conventionnel (voir ci-dessous) le temps de la force brute est dans $O(2^{(m^2)})$.

Tel que mentionné précédemment, l'objectif principal est de concevoir un algorithme d'apprentissage pour classificateurs de type SCM qui trouve le classificateur minimisant la borne sur le risque de Marchand et Sokolova. Nous assumons que la borne varie de manière analogue au vrai risque ; c'est-à-dire qu'un vrai risque élevé implique une borne élevée et qu'un vrai risque bas implique une borne basse. Cette présomption est justifiée par le fait que minimiser la borne a pour conséquence que le classificateur produit ne sera ni trop simple, ni trop complexe.

Le succès d'une technique dérivée du *branch-and-bound* pour des problèmes difficiles nous a encouragé à l'adapter au problème de trouver la SCM minimisant la borne. Pour les données *Sonar, Mines vs. Rocks*, le nombre d'exemples est de 105. Donc, au pire, l'espace à explorer est de taille 11025 !, ce qui est plus petit que la plus grande instance résolue pour le problème du voyageur de commerce. Nous avons réussi à traiter cet espace avec le *branch-and-bound* en trois étapes, mais certains ensembles d'entraînement n'ont pu être analysés au complet en un temps raisonnable avec l'un ou l'autre des algorithmes présentés. Des jeux de données réduits ont alors été utilisés et c'est ainsi que les performances des classificateurs optimaux (selon la borne) ont pu être évaluées et comparées à celles des classificateurs obtenus par l'approche vorace.

Dans les prochaines sections, l'algorithme du *branch-and-bound* sera tout d'abord présenté. Il sera question de l'adapter afin de parcourir l'espace de toutes les SCM possibles pour des données particulières. Les résultats empiriques sur plusieurs ensembles test seront énoncés et discutés. Ensuite, nous présenterons une autre version du *branch-and-bound* qui élimine plus de solutions potentielles. Les résultats pour cette seconde variante seront également énoncés, discutés et comparés.

3.2 Conjonctions ou disjonctions

Rappelons qu'une SCM est soit une conjonction de caractéristiques, soit une disjonction de caractéristiques. Pour chacun des ensembles de données, nous exécuterons un algorithme qui construira uniquement des conjonctions lorsque, pendant la sélection de modèle pour l'algorithme SCM-classique (voir l'algorithme 2.2), nous observerons que nous devons construire une conjonction. À l'opposé, l'algorithme construira des disjonctions lorsque, pendant la sélection de modèle pour l'algorithme SCM-classique, nous observerons que nous devons construire une disjonction. Nous avons procédé de cette façon afin de faciliter la comparaison entre les algorithmes de *branch-and-bound* que nous proposons et l'algorithme SCM-classique.

3.3 Structures de données

Tout au long de ce chapitre, nous nous restreindrons aux SCM qui ont des centres de même type. C'est-à-dire que les conjonctions regrouperont des caractéristiques dont les centres sont tous des exemples négatifs et les disjonctions regrouperont des caractéristiques dont les centres sont tous des exemples positifs. Dans ce chapitre, nous adopterons une des quatre conventions suivantes où la notation $|C|$ pour une liste C nous donne sa taille (c'est-à-dire le nombre d'éléments que la liste C contient). Nous rappelons que l'ensemble d'apprentissage est noté S. Celui-ci est l'union d'un ensemble P d'exemples positifs et d'un ensemble N d'exemples négatifs. Les indices supérieurs des $\mathbf{x}_i$ dans les définitions des SCM pourront être omis lorsqu'il n'y aura pas d'ambiguïté possible entre plusieurs SCM.

Convention 1 *Étant donné un ensemble d'entraînement $S = P \cup N$ et un paramètre de taille maximale de SCM maxCarac.*

- Une caractéristique est nécessairement un trou. Les trous sont introduits dans ce mémoire à l'équation (2.2). Cependant, ici nous représentons un trou par une paire contenant le point centre et le point frontière, ou bord. Tous les trous possibles pour S sont réunis dans l'ensemble suivant :

$$\{(\mathbf{x}_c, \mathbf{x}_b) : \mathbf{x}_c \in N \wedge \mathbf{x}_b \in P\}$$

- Une SCM est nécessairement une conjonction de caractéristiques et est représentée par une liste de la forme :

$$C = \langle (\mathbf{x}^C_{c_1}, \mathbf{x}^C_{b_1}), (\mathbf{x}^C_{c_2}, \mathbf{x}^C_{b_2}), \ldots, (\mathbf{x}^C_{c_{|C|}}, \mathbf{x}^C_{b_{|C|}}) \rangle$$

  où $\mathbf{x}^C_{c_1} < \mathbf{x}^C_{c_2} < \ldots < \mathbf{x}^C_{c_{|C|}}$ dans le même ordre que dans S et où $|C| < maxCarac$.

Convention 2 *Étant donné un ensemble d'entraînement $S = P \cup N$ et un paramètre de taille maximale de SCM maxCarac.*

- Une caractéristique est nécessairement une boule. Les boules sont introduites dans ce mémoire à l'équation (2.1). Nous représentons une boule par une paire contenant le point centre et le point frontière. Toutes les boules possibles pour S sont réunies dans l'ensemble suivant :

$$\{(\mathbf{x}_c, \mathbf{x}_b) : \mathbf{x}_c \in P \wedge \mathbf{x}_b \in N\}$$

- Une SCM est nécessairement une disjonction de caractéristiques et est représentée par une liste de la forme :
$$C = \langle (\mathbf{x}^C_{c_1}, \mathbf{x}^C_{b_1}), (\mathbf{x}^C_{c_2}, \mathbf{x}^C_{b_2}), \ldots, (\mathbf{x}^C_{c_{|C|}}, \mathbf{x}^C_{b_{|C|}}) \rangle$$
où $\mathbf{x}^C_{c_1} < \mathbf{x}^C_{c_2} < \ldots < \mathbf{x}^C_{c_{|C|}}$ dans le même ordre que dans S et où $|C| < maxCarac$.

Convention 3 *Étant donné un ensemble d'entraînement $S = P \cup N$ et un paramètre de taille maximale de SCM $maxCarac$.*

- Une caractéristique est nécessairement un trou. Tous les trous possibles pour S sont réunis dans l'ensemble suivant :
$$\{(\mathbf{x}_c, \mathbf{x}_b) : \mathbf{x}_c \in N \wedge \mathbf{x}_b \in P\}$$
- Une SCM est nécessairement une conjonction de caractéristiques et est représentée par une liste de la forme :
$$C = \langle (\mathbf{x}^C_{c_1}, \mathbf{x}^C_{b_1}), (\mathbf{x}^C_{c_2}, \mathbf{x}^C_{b_2}), \ldots, (\mathbf{x}^C_{c_{|C|}}, \mathbf{x}^C_{b_{|C|}}) \rangle$$
où pour deux bords $\mathbf{x}^C_{b_i}$ et $\mathbf{x}^C_{b_j}$, $i \neq j \Rightarrow \mathbf{x}^C_{b_i} \neq \mathbf{x}^C_{b_j}$ et où $|C| < maxCarac$.

Convention 4 *Étant donné un ensemble d'entraînement $S = P \cup N$ et un paramètre de taille maximale de SCM $maxCarac$.*

- Une caractéristique est nécessairement une boule. Toutes les boules possibles pour S sont réunies dans l'ensemble suivant :
$$\{(\mathbf{x}_c, \mathbf{x}_b) : \mathbf{x}_c \in P \wedge \mathbf{x}_b \in N\}$$
- Une SCM est nécessairement une disjonction de caractéristiques et est représentée par une liste de la forme :
$$C = \langle (\mathbf{x}^C_{c_1}, \mathbf{x}^C_{b_1}), (\mathbf{x}^C_{c_2}, \mathbf{x}^C_{b_2}), \ldots, (\mathbf{x}^C_{c_{|C|}}, \mathbf{x}^C_{b_{|C|}}) \rangle$$
où pour deux bords $\mathbf{x}^C_{b_i}$ et $\mathbf{x}^C_{b_j}$, $i \neq j \Rightarrow \mathbf{x}^C_{b_i} \neq \mathbf{x}^C_{b_j}$ et où $|C| < maxCarac$.

Notre structure de données représentant une SCM diffère légèrement de la forme habituelle (que nous avons présentée à la section 2.9.1). En effet, dans cette section, une SCM C était caractérisée par un ensemble de compression $\mathbf{z_i}$ ainsi que par un message d'information supplémentaire σ. Dans les algorithmes de *branch-and-bound* décrits ci-après, pour une SCM $C = \langle (\mathbf{x}^C_{c_1}, \mathbf{x}^C_{b_1}), (\mathbf{x}^C_{c_2}, \mathbf{x}^C_{b_2}), \ldots, (\mathbf{x}^C_{c_{|C|}}, \mathbf{x}^C_{b_{|C|}}) \rangle$:

- $\mathbf{z_i}(C) = \{\mathbf{x}^C_{c_1}, \mathbf{x}^C_{c_2}, \ldots, \mathbf{x}^C_{c_{|C|}}\} \cup \{\mathbf{x}^C_{b_1}, \mathbf{x}^C_{b_2}, \ldots, \mathbf{x}^C_{b_{|C|}}\}$
- $\sigma(C)$ est l'information minimale requise pour reconstruire C à partir de l'ensemble $\mathbf{z_i}(C)$.

Pour les SCM des quatre conventions, nous avons toujours une hypothèse consistante avec l'ensemble de compression. C'est-à-dire qu'un classificateur SCM ne fait pas d'erreurs avec les points de son propre ensemble de compression. Nous pouvons noter cet état des choses en disant que pour une hypothèse $\mathcal{R}(\mathbf{z_i}(C), \sigma(C))$,

$$(\mathbf{x}_i, y_i) \in \mathbf{z_i}(C) \Rightarrow \mathcal{R}(\mathbf{z_i}(C), \sigma(C))(\mathbf{x}_i) = y_i$$

Nous avons choisi cette structure de données car elle induit implicitement une structure d'arbre avec laquelle il est facile de construire nos algorithmes de *branch-and-bound*. En effet, nous considérerons l'arbre dont la racine est associée au mot vide ϵ et dont les autres nœuds sont chacun associés à une caractéristique satisfaisant une des quatre conventions. Une seule convention, parmi quatre, est utilisée lors de l'exécution du *branch-and-bound*.

Plusieurs nœuds différents peuvent être associés à une même caractéristique. Les arêtes de l'arbre sont ensuite construites de telle sorte qu'à chaque chemin partant de la racine soit associée une SCM satisfaisant la convention choisie et qu'inversement à chaque SCM satisfaisant la convention soit associé un chemin de l'arbre. Étant donné qu'une extrémité de chaque chemin est toujours la racine, nous pouvons désigner chaque chemin par son extrémité qui n'est pas la racine. Donc il est possible à partir d'ici que nous parlions d'une SCM dans l'arbre en utilisant seulement son dernier nœud. Dans ce chapitre, nous ne spécifions pas explicitement la construction de cet arbre, mais elle se retrouve implicitement dans la description de nos algorithmes de *branch-and-bound*.

Convention 5 *Étant donné une liste* $A = \langle a_1, a_2, \ldots, a_{|A|} \rangle$ *et une liste* $B = \langle b_1, b_2, \ldots, b_{|B|} \rangle$

- L'opérateur de troncature $'$ est défini tel que :

$$A' = \langle a_1, a_2, \ldots, a_{|A|-1} \rangle$$

- L'opérateur de concaténation $\|$ est défini tel que :

$$A \parallel B = \langle a_1, a_2, \ldots, a_{|A|}, b_1, b_2, \ldots, b_{|B|} \rangle$$

- L'opérateur de soustraction $-$ est défini tel que la liste $A - B$ contient les éléments de A différents de ceux de B, dans le même ordre que dans A.

3.4 Fonction de coût

À chaque nœud, il y a une fonction de coût qu'il est possible d'évaluer. Dans le cas présent, nous cherchons à minimiser cette fonction. De plus, à chaque nœud, nous pouvons évaluer une borne inférieure à cette fonction. Tout au long du *branch-and-bound* nous gardons en mémoire la plus petite valeur de la fonction de coût trouvée. Ainsi, lorsque nous examinons un nœud, si sa valeur de fonction coût est inférieure au minimum, nous remplaçons le minimum par cette nouvelle valeur.

Au lieu de minimiser directement la borne sur le risque (voir la section 2.8), nous minimisons une valeur plus simple à calculer et à borner. Nous avons mentionné à la page 15 que la borne est proportionnelle au nombre d'exemples dans l'ensemble de compression $|\mathbf{i}|$ et au nombre d'erreurs k sur les exemples de l'ensemble d'apprentissage. Nous minimisons donc la somme de ces deux quantités. Par proportionnelle, nous voulons dire qu'elle augmente quand les deux quantités augmentent et qu'elle diminue lorsque les deux quantités diminuent. Cela est vrai quand $|\mathbf{i}|$ est plus petit que $\frac{m}{2}$, où m est la taille de l'ensemble d'apprentissage.

La définition de la fonction de coût est donnée ci-bas. Nous rappelons que $\mathcal{R}(\mathbf{z_i}(C), \sigma(C))$ est le classificateur identifié par $\mathbf{z_i}(C)$ et $\sigma(C)$. Revoir la section 2.7 pour plus de détails sur la fonction de reconstruction $\mathcal{R}$. Voici donc la fonction dont nous cherchons le minimum :

$$f(\mathbf{z_i}(C), \sigma(C)) \stackrel{\text{def}}{=} |\mathbf{z_i}(C)| + |S| R_S(\mathcal{R}(\mathbf{z_i}(C), \sigma(C))) \tag{3.1}$$

où $|S| R_S(\mathcal{R}(\mathbf{z_i}(C), \sigma(C)))$ est le nombre d'erreurs de classification du classificateur $\mathcal{R}(\mathbf{z_i}(C), \sigma(C))$ sur S.

3.5 Branch-and-bound conventionnel

Pendant la partie « branch » du *branch-and-bound*, que l'on pourrait traduire par séparation, le problème initial est subdivisé en sous-problèmes. Ensuite, et de manière récursive, les sous-problèmes sont subdivisés à leur tour. Ainsi, un arbre est construit, bien que de manière incomplète, car les branches dont on peut s'assurer qu'elles ne donneront pas une solution optimale ne sont pas générées. De manière symbolique, elles sont coupées. La partie « bound » du *branch-and-bound*, que l'on peut traduire

par évaluation, est utilisée pour cette découpe et pour choisir les branches les plus *prometteuses* à explorer.

Afin de rendre la lecture plus facile, $\mathcal{H}$ représente dans cette section l'ensemble de toutes les SCM respectant la convention 1. Nous présenterons l'algorithme uniquement dans le contexte de cette convention. Pour simuler la convention 2, les règles à appliquer sont presque les mêmes que pour la construction d'une disjonction par l'algorithme SCM-classique. Nous inversons les classes de S, nous trouvons des conjonctions et nous inversons les classes des centres et des bords.

3.5.1 Borne pour la fonction de coût

Lors du déroulement du *branch-and-bound*, nous avons partiellement besoin de savoir comment la fonction de coût évoluera pour les descendants d'un nœud en cours de traitement. Comme nous cherchons un minimum à f, il nous faut savoir jusqu'à quelle valeur (une borne) elle peut descendre si nous explorons complètement le sous-arbre ayant comme racine un nœud donné. Si la plus basse valeur trouvée de f lorsque nous traitons un nœud est plus basse que la borne calculée pour le nœud, il est alors inutile de visiter les fils de ce nœud. La borne pour f dans le cadre du *branch-and-bound* conventionnel est :

$$b(C) = |\mathbf{z_i}(C)| + |P| R_P(\mathcal{R}(\mathbf{z_i}(C), \sigma(C))) \tag{3.2}$$

Notons que $f(C)$ est toujours supérieure ou égale à $b(C)$ car :

1. l'algorithme de construction d'une SCM présenté à la section 2.9.1 à son point de départ fait des erreurs sur tous les exemples négatifs et sur aucun exemple positif.
2. À mesure que l'algorithme progresse, le nombre d'erreurs sur les négatifs diminue, tandis que celui sur les positifs reste le même ou augmente.
3. La taille de l'ensemble de compression ne peut qu'augmenter.

En d'autres mots, $b(C)$ est une borne inférieure de $f(\mathbf{z_i}(C_{desc}), \sigma(C_{desc}))$ pour tout nœud C_{desc} descendant du nœud C. Cette propriété est vérifiée puisque le nombre d'erreurs sur les positifs ne peut aller qu'en augmentant à mesure que des caractéristiques sont ajoutées au classificateur, ce qui fait que $b(C)$ est monotone croissante à mesure que nous nous éloignons de la racine de l'arbre.

3.5.2 Algorithme

Dans cette section, nous décrivons en détail le premier algorithme de *branch-and-bound*. Deux sections de l'algorithme principal ont été placées dans des algorithmes secondaires pour améliorer la compréhensibilité et la mise en page.

L'ordre d'exploration de l'arbre est basé sur la borne inférieure b définie à l'équation (3.2). Ainsi, parmi les nœuds du premier niveau, le premier nœud à être exploré est celui ayant la plus petite valeur selon la fonction b et le dernier est celui ayant la plus grande valeur. L'intention derrière cette façon de procéder est de trouver le plus rapidement possible les nœuds ayant une petite valeur de f. De cette manière, nous pouvons réduire le nombre de sous-arbres à explorer. Les autres nœuds du premier niveau sont placés dans une liste $\mathcal{L}$. Une exploration en profondeur est ainsi exécutée récursivement jusqu'à ce que soit rencontré un meilleur nœud qui n'a pas de fils. À cet instant, une recherche est effectuée dans $\mathcal{L}$ pour le prochain nœud à être exploré.

Dans les algorithmes subséquents, nous utiliserons N_C pour désigner l'ensemble des exemples négatifs couverts par C, ou, formulé autrement :

$$N_C = \{\mathbf{x}_i : \mathbf{x}_i \in N \wedge \mathcal{R}(\mathbf{z_i}(C), \sigma(C))(\mathbf{x}_i) = -1\} \tag{3.3}$$

Nous nommons une super-SCM de la SCM C, une SCM qui comporte les caractéristiques C et une de plus. La sous-SCM de C est C sans sa dernière caractéristique $(\mathbf{x}_{c_{|C|}}, \mathbf{x}_{b_{|C|}})$. $f(C)$ est une notation contractée de $f(\mathbf{z_i}(C), \sigma(C))$.

Dans l'algorithme 3.1, la récurrence est émulée avec la conservation, pour exploration immédiate, de la meilleure SCM générée à partir de sa sous-SCM. La boucle principale extrait les SCM de la collection $\mathcal{L}$ tant qu'elle n'est pas vide. La première SCM à être ajoutée à $\mathcal{L}$ est la SCM vide $\langle\rangle$. Cette SCM classifie tous les exemples à 1. Autrement dit, elle classifie correctement tous les exemples positifs et elle classifie incorrectement tous les exemples négatifs.

À la ligne 3 de l'algorithme 3.1, la variable nommée C n'a pas de valeur. C'est la même idée à la ligne 10 pour la variable $C_{meilleur}$. Avant que l'algorithme commence, tous les trous possibles à partir de S sont ordonnés arbitrairement et strictement. Ainsi, une caractéristique $(\mathbf{x}_{c_j}, \mathbf{x}_{b_j})$ peut être positionnée après une caractéristique $(\mathbf{x}_{c_i}, \mathbf{x}_{b_i})$ dans une conjonction seulement si nous avons $(\mathbf{x}_{c_i}, \mathbf{x}_{b_i}) < (\mathbf{x}_{c_j}, \mathbf{x}_{b_j})$ dans l'ordre strict induit par S. Comme deux trous de même centre ne peuvent être présents dans une même SCM, il n'est pas nécessaire de les ordonner entre eux.

Algorithme 3.1 *Branch-and-bound* conventionnel

Entrées : S l'ensemble d'apprentissage
$maxCarac$ le nombre maximal de caractéristiques dans une SCM
Sorties : $\mathcal{B}$ une collection de SCM
Utilise : $\mathcal{L}$ une collection de SCM ou chaque SCM satisfait la convention 1
C, $C_{meilleur}$ et C_{super} des SCM satisfaisant la convention 1
f_{min} la valeur de coût la plus petite trouvée

1: $\mathcal{B} \leftarrow \emptyset$
2: $\mathcal{L} \leftarrow \{\langle\rangle\}$
3: $C \leftarrow$ indéfini
4: $f_{min} \leftarrow \infty$
5: **tant que** $\mathcal{L} \neq \emptyset$ **faire**
6: **si** $C =$ indéfini **alors**
7: $C \leftarrow$ SCM dans $\mathcal{L}$ qui a la b la plus basse.
8: $\mathcal{L} \leftarrow \mathcal{L} - \{C\}$
9: **fin si**
10: $C_{meilleur} \leftarrow$ indéfini
11: **pour tout** $C_{super} \in$ superSCM1(S, C) **faire**
12: **si** $f(C_{super}) < f_{min}$ **alors**
13: $\mathcal{B} \leftarrow \emptyset$
14: $f_{min} \leftarrow f(C_{super})$
15: enlever de $\mathcal{L}$ les SCM dont $b \geq f_{min}$
16: **fin si**
17: **si** $b(C_{super}) \leq f_{min}$ **alors**
18: **si** $f(C_{super}) = f_{min}$ **alors** $\mathcal{B} \leftarrow \mathcal{B} \cup \{C_{super}\}$
19: voir_arrêt(S, $\mathcal{L}$, C_{super}, $C_{meilleur}$, C, $maxCarac$)
20: **fin si**
21: **fin pour**
22: $C \leftarrow C_{meilleur}$
23: **fin tant que**
24: **retourner** $\mathcal{B}$

Ceci explique pourquoi dans la convention 1 nous avons choisi d'ordonner les exemples négatifs (les centres). Ainsi, un trou de centre $\mathbf{x}_j$ peut être après un trou de centre $\mathbf{x}_i$ seulement si nous avons $\mathbf{x}_i < \mathbf{x}_j$ dans S.

L'algorithme 3.2, superSCM1, exploite cette relation pour construire la collection des super-SCM de C. Nous définissons comme équivalents des classificateurs qui regroupent les mêmes caractéristiques, mais dans un ordre différent. L'objectif de suivre un ordre strict est d'éviter que des classificateurs équivalents soient examinés.

Algorithme 3.2 superSCM1

Entrées : S l'ensemble d'apprentissage
C une SCM
Sorties : $\mathcal{U}$ une collection de SCM

$\mathcal{U} \leftarrow \emptyset$
$L_n \leftarrow \{\mathbf{x}_i : \mathbf{x}_i \in N \wedge \mathcal{R}(\mathbf{z_i}(C), \sigma(C))(\mathbf{x}_i) = 1 \wedge \mathbf{x}_i > \mathbf{x}_{c_{|C|}}\}$
pour tout $\mathbf{x}_n \in L_n$ **faire**
  $\rho_{max} \leftarrow \min\{d(\mathbf{x}_i, \mathbf{x}_n) : \mathbf{x}_i \in (P \cap \mathbf{z_i}(C))\}$
  $L_p \leftarrow \{\mathbf{x}_i : \mathbf{x}_i \in P \wedge \mathcal{R}(\mathbf{z_i}(C), \sigma(C))(\mathbf{x}_i) = 1 \wedge d(\mathbf{x}_i, \mathbf{x}_n) \leq \rho_{max}\}$
  pour tout $\mathbf{x}_p \in L_p$ **faire**
    $t \leftarrow$ trou de centre $\mathbf{x}_n$ et de bord $\mathbf{x}_p$
    $\mathcal{U} \leftarrow \mathcal{U} \cup \{C \parallel \langle t \rangle\}$
  fin pour
fin pour
retourner $\mathcal{U}$

Avec superSCM1, les super-SCM possibles sont celles dont la caractéristique supplémentaire possède un centre négatif non couvert par C qui suit, selon l'ordre strict, le centre de la dernière caractéristique de C. C'est la manière dont est défini L_n. Ces caractéristiques supplémentaires sont consistantes avec l'ensemble de compression. L_p est défini selon cette contrainte. De plus, d donne la distance entre deux points et a été introduite à la page 19 de la section 2.9.1 sur les SCM. L'opérateur $\parallel$ est défini dans la convention 5.

Dans l'algorithme du *branch-and-bound*, f_{min} désigne la plus petite valeur de f dans toute l'exploration de l'arbre. Lorsque qu'une SCM C_{super} a une valeur de f plus petite que f_{min}, la collection $\mathcal{B}$ de SCM de f minimale est vidée, f_{min} est mis à jour et un nettoyage de $\mathcal{L}$ est accompli. Il est certain qu'il est inutile de conserver dans $\mathcal{L}$ les SCM dont $b > f_{min}$. Nous pouvons nous persuader qu'il est aussi inutile de conserver celles dont $b = f_{min}$ puisque les SCM incluses dans $\mathcal{L}$ font toutes plus de zéro erreur sur N. En effet, ajouter au moins un exemple négatif dans l'ensemble de compression d'une SCM dont $b = f_{min}$ fera que la nouvelle aura $b^* \geq b + 1 > f_{min}$.

La condition de la ligne 17 sert à ne traiter que les super-SCM présentant un potentiel pour la suite du déroulement de l'algorithme. Nous ne sommes pas intéressés par les super-SCM dont $b > f_{min}$. Il est nécessaire de traiter celles dont $b = f_{min}$ puisqu'il est possible que leur f soit équivalent à leur b (c'est le cas lorsqu'elles ne font pas d'erreur sur N). Les super-SCM qui ont une valeur de f égale à f_{min} sont ajoutées à l'ensemble $\mathcal{B}$. À la ligne 19, la fonction voir_arrêt modifie les valeurs des variables présentes dans l'algorithme 3.1, à la manière d'un passage d'arguments par référence

en C++. Les valeurs des $\mathcal{L}$ et $C_{meilleur}$ peuvent être différentes après cet appel. Après la boucle **pour tout** C_{super}, la variable C est mise à jour selon la super-SCM la plus *prometteuse* trouvée.

L'algorithme 3.3, voir_arrêt, commence par vérifier les conditions d'arrêt pour la construction de C_{super}. Elle est terminée si elle contient le nombre maximal de caractéristiques permis, si elle couvre entièrement N ou si sa valeur de b est trop grande. Dans le cas contraire, ses super-SCM seront générées et évaluées. La taille maximale qui a été utilisée pour les tests est de 25.

Algorithme 3.3 voir_arrêt

Entrées : S l'ensemble d'apprentissage
$\mathcal{L}$ la collection des SCM à visiter
C_{super} la dernière SCM générée
$C_{meilleur}$ la SCM générée la plus *prometteuse*
C la sous-SCM de C_{super}
$maxCarac$ le nombre maximal de caractéristiques dans une SCM

1: **si** $\left(|C_{super}| < maxCarac$ **et** $|N_{C_{super}}| < |N|$ **et** $b(C_{super}) < f_{min}\right)$ **alors**
2: **si** $C_{meilleur}$ = indéfini **alors**
3: $C_{meilleur} \leftarrow C_{super}$
4: **sinon**
5: $n_s \leftarrow |N_{C_{super}}| - |N_C|$
6: $n_m \leftarrow |N_{C_{meilleur}}| - |N_C|$
7: **si** $\left(b(C_{super}) < b(C_{meilleur})\right)$ **ou** $\left(b(C_{super}) = b(C_{meilleur})$ **et** $n_s > n_m\right)$ **alors**
8: $\mathcal{L} \leftarrow \mathcal{L} \cup \{C_{meilleur}\}$
9: $C_{meilleur} \leftarrow C_{super}$
10: **sinon**
11: $\mathcal{L} \leftarrow \mathcal{L} \cup \{C_{super}\}$
12: **fin si**
13: **fin si**
14: **fin si**

La raison d'être de l'algorithme voir_arrêt est de traiter C_{super} seulement lorsque les conditions d'arrêt ne sont pas encore rencontrées. Si elles ne sont pas recontrées, alors C_{super} est évaluée comme candidate pour remplacer $C_{meilleur}$.

Aux lignes 2 à 13 de l'algorithme voir_arrêt, nous conservons dans la variable $C_{meilleur}$ la super-SCM qui a la valeur de b la plus basse et qui couvre le plus d'exemples négatifs supplémentaires. Si $C_{meilleur}$ n'est pas encore définie, elle prend la valeur de C_{super}. Sinon, elle prend sa valeur si la condition de la ligne 7 est respectée. Celle-ci est vraie pour une valeur de b plus basse ou pour une plus grande couverture à b équivalents.

Si toutes les super-SCM de C sont terminées, alors $C_{meilleur}$ demeure indéfinie. Quand cela se produit, la prochaine SCM à être traitée est choisie dans $\mathcal{L}$.

3.5.3 Résultats

Étant donné la taille des espaces à explorer, il n'a pas été possible de traiter la totalité des données pour la plupart des ensembles d'apprentissage. Pour un ensemble d'apprentissage de taille m dont le temps de traitement dépassait 24 heures, m^* exemples, avec $m^* < m$ et m^* pair, ont été sélectionnés. Les $m^*/2$ premiers exemples positifs et $m^*/2$ premiers exemples négatifs ont été utilisés[1]. Les tailles réduites ont été choisies de manière à ce que le traitement soit effectué en moins de 24 heures. Notez que nous avons choisi de construire des ensembles tronqués parfaitement balancés entre nombres d'exemples positifs et négatifs, même si les ensembles d'origine ne l'étaient pas. Ce rebalancement n'est appliqué qu'aux ensembles d'entraînement, les ensembles test n'ont pas été tronqués. Cette façon de procéder a respecté notre priorité, c'est-à-dire avoir des ensembles d'entraînement identiques pour tous les algorithmes. Dans un cadre de production, une plus grande importance est attribuée à la construction d'ensembles d'entraînement les plus représentatifs possible.

Après avoir appliqué le *branch-and-bound* aux données d'apprentissage, plus d'un classificateur de même f minimale ont été trouvés respectivement pour la plupart des ensembles d'apprentissage. L'information contenue dans ces différents classificateurs a été regroupée par des votes de majorité.

Le tableau 3.1 indique la taille m des ensembles d'entraînement originaux, la taille m^* des ensembles réduits, la taille $|T|$ des ensembles test et le nombre de classificateurs

Données	m	m^*	$\|T\|$	Nb classificateurs
Sonar	105	50	103	63
Breast Cancer	343	343	340	3
Australian Credit	353	60	300	2
Glass	107	60	107	5
Heart Disease	150	50	147	4
US Votes	235	130	200	1

TAB. 3.1 – Tailles des données, *branch-and-bound* conventionnel

[1] L'avantage de cette façon de procéder est que les expériences sont reproductibles sans disposer des données tronquées, contrairement au cas d'un choix aléatoire des exemples.

trouvés de même f minimale. Pour l'ensemble *Wisconsin Breast Cancer Database*, tous les exemples ont pu être utilisés, car l'arbre était de faible profondeur. Afin de comparer le *branch-and-bound* conventionnel, les SCM et les SVM sur un pied d'égalité, ils ont tous été entraînés avec les mêmes données et testés sur les mêmes données.

Les SCM produites par l'algorithme SCM-classique pour les comparaisons avec les SCM du *branch-and-bound* ont bénéficié d'une sélection de modèle entièrement menée avec la borne de la section 2.8. Le nombre de caractéristiques, la pénalité et le type (conjonction ou disjonction) sont les paramètres qui ont varié pendant les sélections de modèles. Les paramètres qui donnaient une SCM avec la plus petite borne étaient ceux choisis. Les tailles essayées étaient de 1 jusqu'à 25, les pénalités essayées étaient ∞, 10.0, 9.0, 8.0, 7.0, 6.0, 5.0, 4.0, 3.0, 2.0, 1.5, 1.0, 0.9, 0.8, 0.7, 0.6 et 0.5. La pénalité ∞ signifie qu'aucune erreur sur les positifs n'est permise dans les données d'apprentissage. Les paramètres sélectionnés à l'aide de la borne Marchand-Sokolova sont présentés dans le tableau 3.2. Nous appliquons le qualificatif « simple » à un classificateur individuel qui ne participe pas à un vote de majorité.

Données	Type	Pénalité	Nb caractéristiques
Sonar	disjonction	∞	8
Breast Cancer	disjonction	9	1
Australian Credit	disjonction	5	2
Glass	disjonction	∞	3
Heart Disease	disjonction	2	1
US Votes	conjonction	2	1

TAB. 3.2 – Paramètres des SCM simples

Pour déterminer les meilleurs paramètres de noyau RBF et C_{svm} dans le cas des SVM (voir la section 2.9.2), la validation croisée 10 fois a été utilisée. Pour le noyau, deux boucles imbriquées ont servi pour tester toutes les combinaisons possibles pour

Données	γ	C_{svm}
Sonar	0.005	80
Breast Cancer	0.001	30
Australian Credit	0.005	10
Glass	1.5	80
Heart Disease	0.001	100
US Votes	0.05	10

TAB. 3.3 – Paramètres des SVM

$C_{svm} = 1, 10, 100, 1000$ et 10000 et pour $\gamma = 0.001, 0.01, 0.1, 1, 10, 100$ et 1000. Ensuite, des valeurs intermédiaires ont été essayées pour tenter de trouver un minimum local au R^k_{CV}. Les valeurs retenues sont présentées dans le tableau 3.3.

Le tableau 3.4 contient les risques empiriques sur les ensembles test et les bornes inférieures et supérieures des intervalles de confiance bilatéraux (avec $\delta = 0.1$) pour ces risques empiriques. Les intervalles de confiance sont décrits à la section 2.5. Ce tableau présente aussi les valeurs de f calculées pour les SCM et les valeurs minimales obtenues par le *branch-and-bound*. Les figures 3.1, 3.3, 3.4, 3.5, 3.6 et 3.7 montrent ces mêmes résultats sous une forme plus visuelle. En règle générale, nous n'observons pas de différences significatives entre les vrais risques des différents classificateurs comparés. Dans cette section, l'acronyme BBC signifie « *branch-and-bound* conventionnel ».

Données	Classificateur	$R_T(h)$	I.C. Inf.	I.C. Sup.	$f(\mathbf{z_i}, \sigma)$
Sonar	SCM	0.369	0.290	0.454	14
	SVM	0.330	0.254	0.414	.
	BBC	0.379	0.299	0.464	12
Breast Cancer	SCM	0.035	0.020	0.057	6
	SVM	0.035	0.020	0.057	.
	BBC	0.038	0.023	0.060	6
Australian Credit	SCM	0.203	0.166	0.245	9
	SVM	0.157	0.123	0.195	.
	BBC	0.203	0.166	0.245	9
Glass	SCM	0.168	0.112	0.239	7
	SVM	0.224	0.160	0.301	.
	BBC	0.150	0.096	0.218	7
Heart Disease	SCM	0.184	0.133	0.244	10
	SVM	0.245	0.187	0.310	.
	BBC	0.190	0.139	0.252	10
US Votes	SCM	0.105	0.071	0.148	6
	SVM	0.070	0.043	0.107	.
	BBC	0.105	0.071	0.148	6

TAB. 3.4 – Risques empiriques sur les ensembles test des SCM simples, SVM et votes de majorité construits par des BBC

Sonar

Bien que nous observions une faible différence entre les risques empiriques sur l'ensemble test de la SCM de l'algorithme SCM-classique et de celles du *branch-and-bound*, nous avons de bonnes raisons de croire que le résultat pour le *branch-and-bound* aurait dû être légèrement meilleur. Le fait de faire un vote de majorité pour les SCM résultantes du *branch-and-bound* donne un résultat moins bon que la plupart de ces SCM prises individuellement. Sur les 103 exemples de l'ensemble test, la SCM simple fait 38 erreurs, la SVM en fait 34 et les votants du *branch-and-bound* 39. La figure 3.1 montre les intervalles de confiance pour les $R(h)$ des trois classificateurs comparés.

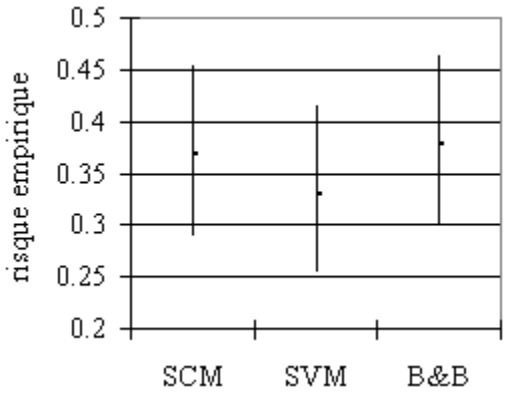

FIG. 3.1 – Intervalles de confiance pour les $R(h)$ de la SCM simple, de la SVM et du vote de majorité construit par un BBC pour les données *Sonar*

Le risque empirique sur l'ensemble test pour le vote de majorité $R_T(B_Q)$ est de 0.379 et le risque empirique de Gibbs $R_T(G_Q)$ est de 0.367. La contribution du vote de majorité est négative, car $R_T(G_Q) < R_T(B_Q)$. Pour être plus précis, il y a 51 des 63 SCM résultantes qui, prises individuellement, ont un risque empirique sur T plus petit que $R_T(B_Q)$.

Nous avons regardé le niveau de similarité entre les SCM produites par le *branch-and-bound*. Pour évaluer celui-ci, les corrélations entre les votes ont été calculées. Les vecteurs $h_j(T)$ qui donnent la classe choisie par les SCM pour chaque exemple de T ont été sauvegardés. Ce sont les colonnes telles que présentées à la figure 2.6. Les corrélations ont été calculées à partir de ces vecteurs. Comme il y a 63 SCM, il y a $\binom{63}{2} = 1953$ corrélations. Les valeurs obtenues sont résumées dans l'histogramme de la figure 3.2. La corrélation moyenne est de 0.841. Nous observons donc que les SCM sont très corrélées. Mais comme nous allons l'observer pour les autres ensembles test dans le cadre du *branch-and-bound* conventionnel, c'est pour *Sonar* que la corrélation moyenne est la plus petite. De plus, nous verrons qu'un mauvais effet de synergie du

vote de majorité semble plus fréquent lorsque les vecteurs de votes sont plutôt corrélés. En effet, les votants produits par le BBC sont tous relativement très corrélés.

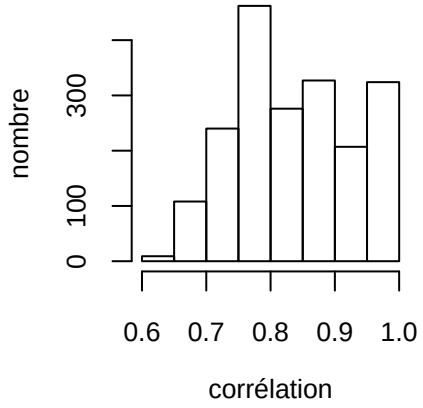

FIG. 3.2 – Corrélations, *Sonar*, BBC

En comparant les vecteurs $h_j(T)$ et en regroupant ceux qui sont identiques, nous nous apercevons que parmi les 63 SCM, il n'y a que 13 groupes. Le tableau 3.5 indique le nombre de vecteurs identiques par groupe. Il y a 9 SCM qui votent exactement de la même manière dans le premier groupe, 12 SCM dans le second groupe et ainsi de suite. Comme il n'y a pas de groupe de taille 1, il y a $\sum_{i=1}^{13} \binom{\text{Taille}[i]}{2} = 181$ corrélations parmi les 1953 qui sont de 1.0, car elles sont calculées sur des vecteurs identiques. Les 13 vecteurs différents ont été comparés deux à deux pour dénombrer leurs composantes différentes. Nous observons entre 1 et 19 différences. Donc, deux SCM qui ne votent pas de la même manière ne diffèrent jamais d'opinion sur plus de 19 exemples de l'ensemble test.

Groupe	1	2	3	4	5	6	7	8	9	10	11	12	13	total
Taille	9	12	3	6	3	3	9	3	3	3	3	2	4	63

TAB. 3.5 – Nombre de SCM de mêmes votes pour l'ensemble T de *Sonar*

Les tableaux 3.6 et 3.7 donnent plus d'information sur les votes de majorité à la suite du *branch-and-bound*. Nous y retrouvons les nombres d'exemples pour lesquels un nombre donné de votants se trompent. Pour l'ensemble test, il y a 51 exemples pour lesquels aucun votant ne se trompe, il y a 13 exemples pour lesquels il y a entre 1 et 31 votants qui se trompent et il y a 25 exemples pour lesquels tous les votants se trompent. Ces nombres sont obtenus en additionnant, pour une ligne donnée, le nombre de composantes des vecteurs $h_j(T)$ tels que dans la figure 2.6 dont la valeur est différente de la classe voulue pour la ligne. Nous pouvons retrouver le risque empirique sur T du vote de majorité en utilisant les dénombrements où le nombre d'erreurs est plus grand

que le nombre de votants divisé par deux : $\frac{14+25}{103} = 0.379$. Ne faire aucune erreur avec 51 exemples sur 103 est très encourageant, par contre, se tromper sur toute la ligne avec 25 exemples nous suggère qu'une bonne part de l'erreur était inévitable.

Nombre de votants erronés	Nombre d'exemples concernés
0	40
3 à 24	7
25 à 31	0
32 à 44	0
45	1
63	2
total	50

TAB. 3.6 – Erreurs par exemple pour l'ensemble S de *Sonar*

Nombre de votants erronés	Nombre d'exemples concernés
0	51
1 à 31	13
32 à 62	14
63	25
total	103

TAB. 3.7 – Erreurs par exemple pour l'ensemble T de *Sonar*

Breast Cancer

Si nous comparons avec les autres données, c'est pour *Breast Cancer* que l'on observe les plus bas risques empiriques sur l'ensemble test. Sur les 340 exemples de l'ensemble test, la SCM simple et la SVM ne font que 12 erreurs, tandis que le vote de majorité en fait 13. La SCM produite par l'algorithme SCM-classique se retrouve dans l'ensemble des SCM trouvées pour le vote de majorité. Il s'agit de la première qui a été trouvée et c'est celle qui a le plus bas risque empirique sur T quand elles sont prises individuelle-

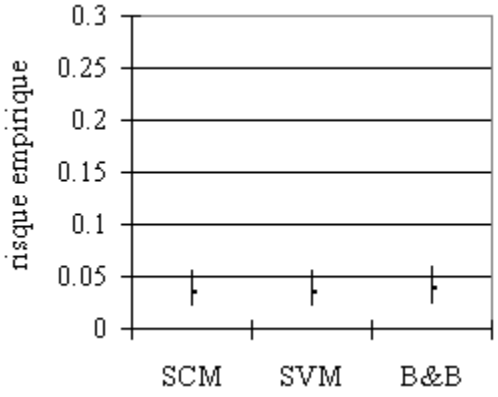

FIG. 3.3 – Intervalles de confiance pour les $R(h)$ de la SCM simple, de la SVM et du vote de majorité construit par un BBC pour les données *Breast Cancer*

ment. L'algorithme SCM-classique parvient donc à atteindre l'optimum déterminé par le *branch-and-bound*. La figure 3.3 montre les intervalles de confiance pour les $R(h)$ des trois classificateurs comparés. Afin d'aider la comparaison, nous avons fixé l'échelle des axes d'ordonnée. Ceci explique l'espace inutilisé dans la figure.

Le tableau 3.8 présente les risques empiriques sur T des SCM du vote de majorité prises séparément. Bien qu'ayant un même risque empirique sur T, les SCM 1 et 2 votent différemment. Pour les données *Breast Cancer*, le vote de majorité donne un résultat pire que la plupart de ses SCM prises individuellement. En effet, le vote de majorité donne un risque empirique sur T de 0.038, tandis que deux sur trois de ses SCM ont un risque empirique sur T moins élevé.

SCM	$R_T(h)$
1	0.035
2	0.035
3	0.038

TAB. 3.8 – $R_T(h)$ individuels, *Breast Cancer*, BBC

Le tableau 3.9 donne le nombre de différences entre les vecteurs de votes des SCM différentes. Nous pouvons constater que le nombre de différences est au plus 4 sur une possibilité de 340. La contribution du vote de majorité n'a pas été influencée positivement par la très grande similarité des SCM. Le tableau 3.10 confirme la similarité, car les corrélations y sont très grandes. L'information du nombre de différences y est reflétée assez fidèlement, car les corrélations y sont inversement proportionnelles.

		SCM	
		3	2
SCM	1	1	4
	2	3	

TAB. 3.9 – Nombre de votes différents, *Breast Cancer*, BBC

		SCM	
		3	2
SCM	1	0.994	0.975
	2	0.981	

TAB. 3.10 – Corrélations, *Breast Cancer*, BBC

Les tableaux 3.11 et 3.12 donnent plus d'information sur les votes de majorité à la suite du *branch-and-bound*. Nous y retrouvons les nombres d'exemples pour lesquels un nombre donné de votants se trompent. Sur l'ensemble test, il y a 326 exemples pour lesquels aucun votant ne se trompe, il y a 1 exemple pour lequel il y a un votant qui se trompe et il y a 10 exemples pour lesquels tous les votants se trompent. Nous pouvons retrouver le risque empirique sur T du vote de majorité en utilisant les dénombrements où le nombre d'erreurs est plus grand que le nombre de votants divisé par deux :

$\frac{3+10}{340} = 0.038$. La proportion d'exemples pour lesquels tous les votants se trompent est très petite.

Nombre de votants erronés	Nombre d'exemples concernés
0	339
1	1
2	0
3	3
total	343

TAB. 3.11 – Erreurs par exemple pour l'ensemble S de *Breast Cancer*

Nombre de votants erronés	Nombre d'exemples concernés
0	326
1	1
2	3
3	10
total	340

TAB. 3.12 – Erreurs par exemple pour l'ensemble T de *Breast Cancer*

Australian Credit

Pour les données d'analyse de crédit, nous remarquons à la figure 3.4 que la SCM simple et le vote de majorité performent de manière équivalente sur l'ensemble test. Les SVM performent un peu mieux que ces deux derniers, mais pas significativement. Dans l'ensemble test, il a 300 exemples. Un risque empirique sur T de 0.203 pour les SCM correspond à 61 erreurs. Pour les SVM, il s'agit de 47 erreurs, soit 14 de moins. La SCM produite par l'algorithme SCM-classique est encore présente dans l'ensemble des SCM trouvées pour le vote de majorité, c'est également la première qui a été trouvée et celle de plus bas risque empirique sur l'ensemble test. La figure 3.4 montre les intervalles de confiance pour les $R(h)$ des trois classificateurs comparés.

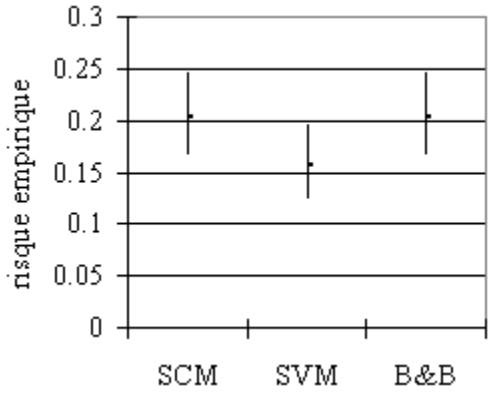

FIG. 3.4 – Intervalles de confiance pour les $R(h)$ de la SCM simple, de la SVM et du vote de majorité construit par un BBC pour les données *Australian Credit*

Parmi toutes les données testées, c'est pour *Australian Credit* que nous retrouvons le deuxième plus petit nombre de SCM résultantes pour le vote de majorité. La première SCM prise individuellement possède un risque empirique sur T de 0.203, tandis que la seconde possède un risque de 0.217. Il est à noter que le risque empirique sur T du vote de majorité est le même que celui de la meilleure SCM qui y participe. L'effet du vote aura donc, sans être exceptionnel, été positif.

La corrélation entre les deux vecteurs de vote est de 0.91 et il y a 14 exemples pour lesquels les votes sont différents, donc 14 égalités. Nous rappelons que dans ce cas le vote est automatiquement positif (voir la section 2.9.3). Ce nombre de différences est relativement faible.

Les tableaux 3.13 et 3.14 nous donnent plus d'information sur les votes de majorité à la suite du *branch-and-bound*. Nous y retrouvons les nombres d'exemples pour lesquels un nombre donné de votants se trompent. Pour l'ensemble test, il y a 239 exemples pour lesquels aucun votant ne se trompe, 5 exemples pour lesquels il y a un seul votant qui se trompe et il y a 56 exemples pour lesquels tous les votants se trompent. Sur les 14 exemples où il y a égalité, il y avait 5 exemples négatifs et 9 positifs. Nous pouvons retrouver le risque empirique sur T du vote de majorité en utilisant les dénombrements où le nombre d'erreurs est plus grand ou égal au nombre de votants divisé par deux : $\frac{5+56}{300} = 0.203$. La convention de classifier positif en cas d'égalité a légèrement aidé. Dans le cas contraire, il aurait été nécessaire de compter 4 erreurs de plus.

Nombre de votants erronés	Nombre d'exemples concernés
0	54
1	2
2	4
total	60

TAB. 3.13 – Erreurs par exemple pour l'ensemble S de *Australian Credit*

Nombre de votants erronés	Nombre d'exemples concernés
0	239
1	5
2	56
total	300

TAB. 3.14 – Erreurs par exemple pour l'ensemble T de *Australian Credit*

Glass

Uniquement pour les données *Glass Identification Database*, nous observons un risque empirique sur l'ensemble test plus petit pour le *branch-and-bound*. Cependant, la différence entre les vrais risques n'est pas significative. La SCM et le vote de majorité performent un peu mieux que la SVM. Ces trois classificateurs font respectivement 18,

16 et 24 erreurs sur les 107 exemples de test. La SCM produite par l'algorithme SCM-classique est encore présente dans l'ensemble des SCM trouvées pour le vote de majorité (et c'est également la première qui a été trouvée). Cependant elle fait partie des deux SCM ayant le plus haut risque empirique sur T lorsqu'elles sont prises individuellement. Cela fut observé uniquement pour ces données. La figure 3.5 montre les intervalles de confiance pour les $R(h)$ des trois classificateurs comparés.

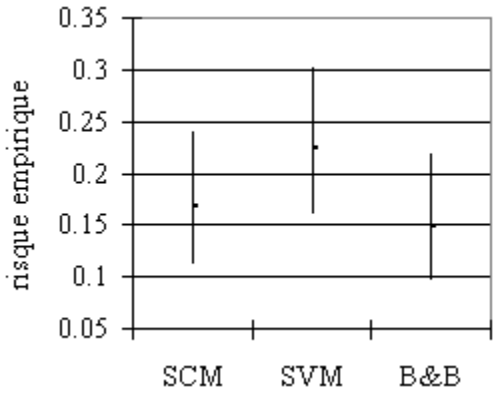

FIG. 3.5 – Intervalles de confiance pour les $R(h)$ de la SCM simple, de la SVM et du vote de majorité construit par un BBC pour les données *Glass*

Le tableau 3.15 présente les risques empiriques sur T des SCM du vote de majorité prises séparément. La première colonne donne un numéro arbitraire pour chaque SCM trouvée. La seconde colonne indique à quelle SCM précédemment trouvée le vecteur de vote $h_j(T)$ est identique. La SCM précédente est référencée par le numéro donné dans la première colonne. Bien qu'ayant un même risque empirique sur T, les SCM 2 et 3 votent différemment. Les seules SCM à voter de la même manière sont les SCM 1 et 4.

SCM	Mêmes votes que la SCM	$R_T(h)$
1	1	0.168
2	2	0.150
3	3	0.150
4	1	0.168
5	5	0.159

TAB. 3.15 – $R_T(h)$ individuels, *Glass*, BBC

Pour ces données, le vote de majorité donne un meilleur résultat que la plupart de ses SCM prises individuellement. En effet, le vote de majorité donne un risque empirique sur T de 0.150, une valeur égale aux deux plus petits risques empiriques sur T observés pour ses SCM.

Le tableau 3.16 donne le nombre de différences entre les vecteurs de votes des SCM différentes. Nous pouvons constater que le nombre de différences est au plus 4 sur une possibilité de 107. La contribution du vote de majorité n'a pas été influencée négativement par la très grande similarité des SCM. Le tableau 3.17 confirme la similarité, car les corrélations y sont grandes.

		SCM		
		5	3	2
SCM	1	3	4	2
	2	1	2	
	3	3		

TAB. 3.16 – Nombre de votes différents, *Glass*, BBC

		SCM		
		5	3	2
SCM	1	0.941	0.920	0.961
	2	0.981	0.961	
	3	0.941		

TAB. 3.17 – Corrélations différentes de 1.0, *Glass*, BBC

Les tableaux 3.18 et 3.19 donnent plus d'information sur les votes de majorité à la suite du *branch-and-bound*. Nous y retrouvons les nombres d'exemples pour lesquels un nombre donné de votants se trompent. Sur l'ensemble test, il y a 87 exemples pour lesquels aucun votant ne se trompe, il y a 2 exemples pour lesquels il y a un votant qui se trompe et il y a 15 exemples pour lesquels tous les votants se trompent.

Nombre de votants erronés	Nombre d'exemples concernés
0	58
1	0
2	0
4	0
5	2
total	60

TAB. 3.18 – Erreurs par exemple pour l'ensemble S de *Glass*

Nombre de votants erronés	Nombre d'exemples concernés
0	87
1	2
2	2
4	1
5	15
total	107

TAB. 3.19 – Erreurs par exemple pour l'ensemble T de *Glass*

Heart Disease

Pour ces données, la SCM et le vote de majorité donnent des résultats très similaires ; tous deux sont un peu meilleurs que celui de la SVM. L'ensemble test compte 147 exemples. Sur ceux-ci, la SCM simple fait 27 erreurs, la SVM 36 et le vote de majorité 28. La SCM produite par l'algorithme SCM-classique est encore présente dans l'ensemble des SCM trouvées pour le vote de majorité ; c'est également la première qui a été trouvée

et elle fait partie des deux SCM de plus petit risque empirique sur l'ensemble test. La figure 3.6 montre les intervalles de confiance pour les $R(h)$ des trois classificateurs comparés.

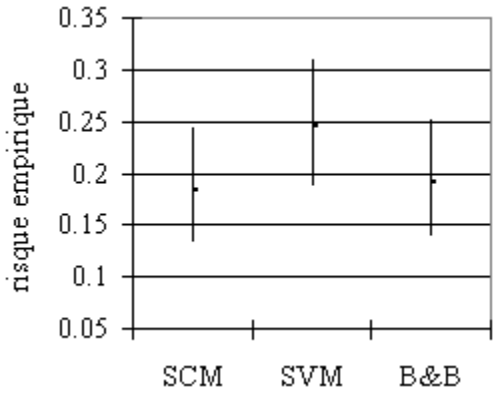

FIG. 3.6 – Intervalles de confiance pour les $R(h)$ de la SCM simple, de la SVM et du vote de majorité construit par un BBC pour les données *Heart Disease*

Les quatre SCM du vote de majorité votent toutes de manière différente. Leurs risques empiriques sur T individuels sont : 0.184, 0.184, 0.224 et 0.231. La contribution du vote de majorité est positive puisque le risque empirique de Bayes sur l'ensemble test $R_T(B_Q)$ (0.190) est plus petit que le risque empirique de Gibbs sur l'ensemble test $R_T(G_Q)$ (0.206).

Les tableaux 3.20 et 3.21 donnent les nombres de différences entre vecteurs de votes pris deux à deux et les corrélations entre ces mêmes vecteurs. Nous pouvons constater que le nombre de différences est au plus 11, sur une possibilité de 147. Un regard sur le tableau 3.21 nous fait également constater que les votes sont plutôt semblables pour la plupart des paires de votants, deux d'entre eux sont en fait très semblables.

		SCM		
		4	3	2
SCM	1	11	10	1
	2	10	11	
	3	1		

TAB. 3.20 – Nombre de votes différents, *Heart Disease*, BBC

		SCM		
		4	3	2
SCM	1	0.8518	0.8646	0.9864
	2	0.8648	0.8504	
	3	0.9865		

TAB. 3.21 – Corrélations, *Heart Disease*, BBC

Les tableaux 3.22 et 3.23 fournissent plus d'information sur les votes de majorité à la suite du *branch-and-bound*. Nous y retrouvons les nombres d'exemples pour lesquels un nombre donné de votants se trompent. Pour l'ensemble test, il y a 119 exemples pour

lesquels aucun votant ne se trompe, 3 exemples pour lesquels il y a deux votants qui se trompent et il y a 25 exemples pour lesquels tous les votants se trompent. Il y a égalité pour 11 exemples. De ceux-ci, il y a 3 exemples négatifs et 8 positifs. Nous pouvons retrouver le risque empirique sur T du vote de majorité en utilisant les dénombrements où le nombre d'erreurs est plus grand ou égal au nombre de votants divisé par deux : $\frac{3+25}{147} = 0.203$. Encore une fois, le choix de classifier positif lorsqu'il y a égalité s'est avéré avantageux. Dans le cas contraire, il aurait été nécessaire de compter 5 erreurs de plus.

Nombre de votants erronés	Nombre d'exemples concernés
0	42
1	0
2	2
3	0
4	6
total	50

TAB. 3.22 – Erreurs par exemple pour l'ensemble S de *Heart Disease*

Nombre de votants erronés	Nombre d'exemples concernés
0	119
1	0
2	3
3	0
4	25
total	147

TAB. 3.23 – Erreurs par exemple pour l'ensemble T de *Heart Disease*

US Votes

Il n'a résulté cette fois de l'optimisation de f qu'un seul classificateur. Il s'agit du même classificateur que celui trouvé par l'algorithme SCM-classique. Dans l'ensemble test, il y a 200 exemples. Donc, un risque empirique de 0.105 sur l'ensemble test corres-

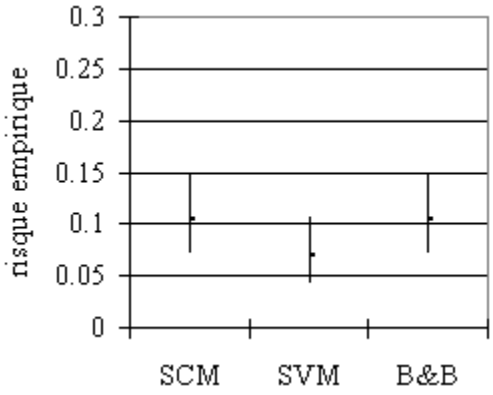

FIG. 3.7 – Intervalles de confiance pour les $R(h)$ de la SCM simple, de la SVM et du vote de majorité construit par un BBC pour les données *US Votes*

pond à 21 erreurs, tandis que pour la SVM, le risque empirique sur T de 0.07 correspond à 14 erreurs. Les intervalles de confiance nous indiquent que le vrai risque de la SCM n'est pas significativement supérieur à celui de la SVM. La figure 3.7 montre les intervalles de confiance pour les $R(h)$ des trois classificateurs comparés.

Facteurs explicatifs

Dans cette section, nous mettons en relation les risques empiriques sur T des votes de majorité avec d'autres quantités calculées à partir des votes des SCM participant à ces votes de majorité. Nous tentons d'expliquer les fluctuations des risques empiriques sur T des votes de majorité par les particularités des SCM qui y participent.

Nous tentons de comprendre les résultats obtenus pour $R_T(B_Q)$ obtenu par le vote de majorité du *branch-and-bound* et nous tentons de trouver une cause à la contribution du vote de majorité, ici quantifiée par $R_T(G_Q) - R_T(B_Q)$. Les facteurs explicatifs dont nous avons tenu compte sont décrits à la section 2.9.3.

La corrélation moyenne $\bar{r}$ donnée dans le tableau 3.24 est la moyenne des corrélations de tous les vecteurs de votes pris deux par deux. Les vecteurs de votes sont illustrés à la figure 2.6. La variance des $\widehat{W_Q}$ est la variance empirique.

Données	$\widehat{C_Q}$	$\bar{r}$	Variance des $\widehat{W_Q}$
Breast Cancer	0.131	0.983	0.032
Glass	0.520	0.953	0.126
Heart Disease	0.632	0.901	0.147
Australian Credit	0.648	0.910	0.155
Sonar	0.931	0.841	0.199

Données	$R_T(G_Q)$	$R_T(B_Q)$	$R_T(G_Q) - R_T(B_Q)$
Breast Cancer	0.036	0.038	-0.00196
Glass	0.159	0.150	0.00935
Heart Disease	0.206	0.190	0.01531
Australian Credit	0.210	0.203	0.00667
Sonar	0.367	0.379	-0.01125

TAB. 3.24 – Facteurs explicatifs et risques empiriques ensemble test, BBC

Nous pouvons observer à la figure 3.8 que les $\widehat{C_Q}$ calculés sur T sont très reliés aux risques empiriques sur T des votes de majorité. Cette constatation semble montrer

que C_Q est, à peu de chose près, directement proportionnel au vrai risque des votes de majorité. D'autre part, il semblerait que le risque empirique sur T d'un vote de majorité soit plus grand lorsque $\bar{r}$ est petit (c'est-à-dire lorsque que les votants s'expriment plutôt différemment).

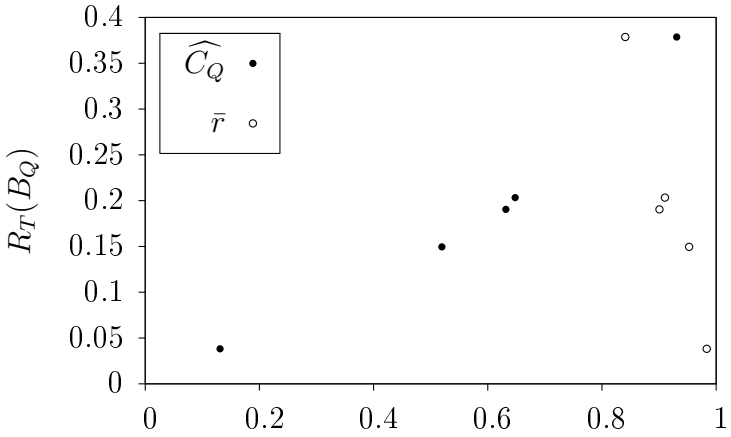

FIG. 3.8 – Facteurs explicatifs explicites pour $R_T(B_Q)$, BBC

3.5.4 Résumé

Rappelons que les vrais risques des classificateurs SCM, SVM et votes de majorité n'ont nulle part été observés significativement différents. Pour toutes les données, sauf *US Votes*, les SCM utilisées sont des disjonctions. Le f (section 3.4) optimal a été atteint par l'algorithme SCM-classique pour la plupart des ensembles d'entraînement. Lorsque c'était le cas, la SCM produite par l'algorithme classique faisait partie des SCM trouvées par le *branch-and-bound* et il s'agissait de la première à être trouvée. De plus, pour ces données, la SCM de l'algorithme SCM-classique a été de celles de plus petit risque empirique sur T, sauf pour *Glass*. Lorsque la SCM produite par l'algorithme SCM-classique n'a pas été de f optimal, donc pour les données *Sonar* uniquement, celle-ci était légèrement plus performante que le vote de majorité. Nous avons observé que le risque empirique sur T du vote de majorité pour les SCM de même f était proportionnel à $\widehat{C_Q}$ et inversement proportionnel à $\bar{r}$.

L'enseignement que nous recevons en comparant empiriquement les SCM de l'algorithme SCM-classique et les classificateurs du vote de majorité suite au *branch-and-bound* conventionnel est qu'il n'est pas justifié d'utiliser le *branch-and-bound* à la place de l'algorithme SCM-classique. Cette observation est la principale découverte effectuée au cours de cette maîtrise.

3.6 Branch-and-bound en trois étapes

La première approche que nous avons décrite pour le *branch-and-bound* était relativement élémentaire. En ajoutant une contrainte sur le nombre d'exemples négatifs supplémentaires couverts, nous pouvons bénéficier d'une borne inférieure sur f plus grande que celle définie à l'équation (3.2) pour le *branch-and-bound* conventionnel. C'est-à-dire qu'en ajoutant cette contrainte, nous serons en mesure d'éliminer plus de nœuds non *prometteurs* dans l'arbre. Par conséquent, il sera alors possible de traiter des ensembles d'entraînement contenant plus d'exemples.

Dans cette section, $\mathcal{H}$ représente l'ensemble de toutes les SCM respectant la convention 3 et nous ne présenterons l'algorithme que dans le contexte de cette convention. Pour simuler la convention 4, les règles à appliquer sont presque les mêmes que pour la construction d'une disjonction par l'algorithme SCM-classique. Nous inversons les classes de S, nous trouvons des conjonctions et nous inversons les classes des centres et des bords.

Pour être considéré comme *prometteur*, un nœud traité doit avoir une borne inférieure b de valeur plus petite que la valeur de f. Si, en général, la borne est plus grande, il y aura donc moins de chemins sans intérêt qui seront explorés. La nouvelle contrainte est un ordre que nous imposons aux nombres d'exemples négatifs supplémentaires couverts. Nous traversons trois phases pour nous rapprocher le plus rapidement possible de la valeur de f minimale. De plus, contrairement au *branch-and-bound* conventionnel, les exemples négatifs couverts ne sont pas enlevés des centres possibles pour les caractéristiques subséquemment ajoutées à une SCM.

Pour ce nouvel algorithme de *branch-and-bound*, nous restreignons l'ensemble des SCM admissibles en ne permettant plus à des points frontière d'être utilisés par plus d'une caractéristique d'une SCM. C'est cette restriction qui nous a permis d'élaborer une nouvelle borne inférieure b à la fonction de coût f qui est plus serrée. De plus, la restriction nous permet de réécrire f ainsi :

$$f(\mathbf{z_i}(C), \sigma(C)) = 2|C| + |S| R_S(\mathcal{R}(\mathbf{z_i}(C), \sigma(C)))$$

3.6.1 Borne pour la fonction de coût

Tout comme à la section 2.9.1 traitant des SCM, $Q_i \subset N$ désigne l'ensemble des exemples négatifs couverts par la caractéristique $(\mathbf{x}_{c_i}, \mathbf{x}_{b_i})$ et $R_i \subset P$ désigne l'ensemble des exemples positifs incorrectement classifiés par cette même caractéristique.

Pour une SCM $C = \langle(\mathbf{x}_{c_1}, \mathbf{x}_{b_1}), (\mathbf{x}_{c_2}, \mathbf{x}_{b_2}), \ldots, (\mathbf{x}_{c_{|C|}}, \mathbf{x}_{b_{|C|}})\rangle$, nous définissons $n_1 = |Q_{C_1}|$, $n_2 = |Q_{C_2} - Q_{C_1}|$ et $n_i = |Q_{C_i} - \cup_{j<i} Q_{C_j}|$. Ainsi, n_i désigne le nombre d'exemples négatifs supplémentaires couverts par une caractéristique i (par rapport aux caractéristiques précédentes). Ainsi, sans perte de généralité nous pouvons imposer la contrainte que : $n_1 \geq n_2 \geq \ldots \geq n_{|C|}$. De plus, définissons $p_1 = |R_{C_1}|$, $p_2 = |R_{C_1} \cup R_{C_2}|$ et $p_i = |\cup_{j=1}^{i} R_{C_j}|$ le nombre d'exemples positifs incorrectement classifiés par la caractéristique i et les précédentes.

- L'opérateur $'$ pour une liste est défini dans la convention 5.
- « C' » désigne la sous-SCM de C.
- N_C désigne le sous ensemble de N classifié négatif par la SCM C (voir l'équation 3.3)

Nous avons à cet instant tous les éléments nécessaires pour introduire la borne inférieure b sur f :

$$b(C) = 2|C'| + p_{|C|} + \min\left(2\left\lceil\frac{|N - N_{C'}|}{n_{|C|}}\right\rceil, 2\left\lfloor\frac{|N - N_{C'}|}{n_{|C|}}\right\rfloor + |N - N_{C'}| \bmod n_{|C|}\right) \quad (3.4)$$

Notez que dans l'expression pour $b(C)$, nous additionnons :

- Deux fois la taille de C'.
- Le nombre d'erreurs sur les exemples positifs par $\mathcal{R}(\mathbf{z_i}(C), \sigma(C))$.
- L'augmentation minimale de la taille de l'ensemble de compression et le nombre d'erreurs sur les exemples négatifs conséquent à cette augmentation.

Le cas couvert par le premier argument du minimum est celui où tous les exemples négatifs restants (et non couverts par C') seront couverts par C et par les caractéristiques subséquentes. En effet, pour couvrir $|N - N_{C'}|$ exemples avec des partitions de taille au plus $n_{|C|}$, nous avons besoin d'au moins $\left\lceil\frac{|N-N_{C'}|}{n_{|C|}}\right\rceil$ caractéristiques.

Le cas couvert par le second argument du minimum est celui où les exemples négatifs restants ne seront pas tous couverts par C et les caractéristiques subséquentes. Il y aura à

ce moment au mieux $\left\lfloor \frac{|N-N_{C'}|}{n_{|C|}} \right\rfloor$ caractéristiques de plus dans l'ensemble de compression. Dans ce cas, il restera alors $|N-N_{C'}| \bmod n_{|C|}$ erreurs supplémentaires sur les exemples négatifs si nous n'ajoutons que $\left\lfloor \frac{|N-N_{C'}|}{n_{|C|}} \right\rfloor$ caractéristiques. Ce cas est réalisé uniquement lorsque le reste de la division entière est 1, car il est plus avantageux de faire une erreur sur N plutôt que d'ajouter une caractéristique à C pour couvrir cet exemple.

3.6.2 Algorithme

La première étape de l'algorithme consiste à trouver une solution par l'algorithme SCM-classique

$$G = \langle (\mathbf{x}^G_{c_1}, \mathbf{x}^G_{b_1}), (\mathbf{x}^G_{c_2}, \mathbf{x}^G_{b_2}), \ldots, (\mathbf{x}^G_{c_{|G|}}, \mathbf{x}^G_{b_{|G|}}) \rangle$$

qui ne fait aucune erreur sur P. Ensuite, nous examinons cette solution avec une approche à rebours. L'objectif de cette seconde étape est de faire diminuer le plus possible la valeur de f avant d'effectuer une recherche dans un arbre plus grand (lors de la troisième étape). G est améliorée graduellement à partir de sa fin avec un *branch-and-bound* pour trouver une des meilleures conjonctions, selon f, qui ne fait pas d'erreur sur P. La conjonction trouvée est notée

$$H = \langle (\mathbf{x}^H_{c_1}, \mathbf{x}^H_{b_1}), (\mathbf{x}^H_{c_2}, \mathbf{x}^H_{b_2}), \ldots, (\mathbf{x}^H_{c_{|H|}}, \mathbf{x}^H_{b_{|H|}}) \rangle$$

Finalement, pour la troisième étape, H est elle aussi améliorée à rebours avec un *branch-and-bound* permettant cette fois les erreurs sur P.

Le but d'utiliser l'approche à rebours est de commencer la recherche par la SCM connue qui donne la valeur de f la plus petite. De plus, les SCM suivantes dans une recherche par une approche à rebours pourraient mener plus rapidement à une faible valeur de f.

Tout comme pour le *branch-and-bound* conventionnel, il peut y avoir plusieurs SCM qui satisfont le critère d'optimalité selon f. Si c'est le cas, elles sont regroupées dans un vote de majorité. Les SCM de même f que H à la fin de la deuxième étape n'ont pas à être examinées puisque H est entièrement revue lors de la troisième étape.

À la deuxième étape de l'algorithme, c'est-à-dire dans le sous-algorithme 3.4, nous commençons par enlever les deux dernières caractéristiques de G. Il serait inutile de commencer une recherche avec G' puisqu'il est impossible d'obtenir une meilleure f qu'avec la caractéristique qui a été enlevée. $f(C)$ est une notation contractée de $f(\mathbf{z_i}(C), \sigma(C))$.

Algorithme 3.4 Étape 2 du *branch-and-bound* en 3 étapes

Entrées : S l'ensemble d'apprentissage
$maxCarac$ le nombre maximal de caractéristiques dans une SCM
G une SCM produite par l'algorithme SCM-classique
Sorties : H une SCM
Utilise : $\mathcal{L}$ une liste de SCM ou chaque SCM satisfait la convention 3
C et C_{super} des SCM satisfaisant la convention 3
f_{min} la valeur de coût la plus petite trouvée

1: $H \leftarrow G$
2: $f_{min} \leftarrow f(G)$
3: $G \leftarrow G'$
4: **tant que** $|G| > 0$ **faire**
5: $\mathcal{L} \leftarrow \text{superSCM2}(S, G') - \langle G \rangle$
6: $G \leftarrow G'$
7: **tant que** $\mathcal{L} \neq \emptyset$ **faire**
8: $C \leftarrow$ dernière SCM de la liste $\mathcal{L}$
9: $\mathcal{L} \leftarrow \mathcal{L}'$
10: **si** $f(C) < f_{min}$ **alors**
11: $H \leftarrow C$
12: $f_{min} \leftarrow f(C)$
13: enlever de $\mathcal{L}$ les SCM dont la valeur de $b \geq f_{min}$
14: **fin si**
15: **si** $|C| < maxCarac$ **et** $|N_C| < |N|$ **alors**
16: **pour tout** $C_{super} \in \text{superSCM2}(S, C)$ **faire**
17: **si** $b(C_{super}) < f_{min}$ **alors** $\mathcal{L} \leftarrow \mathcal{L} \parallel \langle C_{super} \rangle$
18: **fin pour**
19: **fin si**
20: **fin tant que**
21: **fin tant que**
22: **retourner** H

Dans la boucle principale, nous utilisons chacune des sous-SCM de G' pour effectuer une recherche en profondeur. La première opération à l'intérieur de cette boucle est d'enlever de la collection des SCM à visiter la sous-SCM précédente, car elle a servi de point de départ à une autre recherche.

L'algorithme superSCM2 donne les super-SCM d'une SCM C qui respectent la convention 3 et qui ne font pas d'erreur sur P. C'est lors de l'assignation d'une valeur à $\mathbf{x}_p$ que nous nous assurons que les SCM retournées ne font pas d'erreur sur P. Une caractéristique construite à partir d'un nouveau centre négatif et de son exemple positif le plus proche ne fait pas d'erreur sur les autres exemples de P. Notez que d donne la distance entre deux points et a été introduite à la page 19 de la section 2.9.1 sur les SCM. Le nombre d'exemples négatifs supplémentaires couverts par t doit être inférieur ou égal à ceux de la dernière caractéristique de C, tel que spécifié dans la convention. Il doit aussi être supérieur à 2, car sinon il est certain que toutes les super-SCM de C, et les super-SCM de celles-ci, et ainsi de suite, n'auront pas une valeur de f plus petite que $f(C)$. Rappelons que l'opérateur de concaténation $\|$ est défini à la convention 5.

Algorithme 3.5 superSCM2

Entrées : S l'ensemble d'apprentissage
C une SCM ne faisant aucune erreur sur P
Sorties : $\mathcal{U}$ une liste de SCM ne faisant aucune erreur sur P

$\mathcal{U} \leftarrow \langle\rangle$
$L_n \leftarrow \{\mathbf{x}_i : \mathbf{x}_i \in N \wedge \mathbf{x}_i \notin \mathbf{z_i}(C)\}$
pour tout $\mathbf{x}_n \in L_n$ **faire**
  $\rho_{max} \leftarrow \min\{d(\mathbf{x}_i, \mathbf{x}_n) : \mathbf{x}_i \in (P \cap \mathbf{z_i}(C))\}$
  $\mathbf{x}_p \leftarrow \big(\mathbf{x}_i$ qui minimise $d(\mathbf{x}_n, \mathbf{x}_i)$ avec $\mathbf{x}_i \in P$ et $\mathcal{R}(\mathbf{z_i}(C), \sigma(C))(\mathbf{x}_i) = 1$ et
    $\mathbf{x}_i \notin \mathbf{z_i}(C)$ et $d(\mathbf{x}_n, \mathbf{x}_i) \leq \rho_{max}\big)$
  $t \leftarrow$ trou de centre $\mathbf{x}_n$ et de bord $\mathbf{x}_p$
  $C_{super} \leftarrow C \parallel \langle t \rangle$
  si $n_{|C_{super}|} \leq n_{|C|}$ **et** $n_{|C_{super}|} > 2$ **alors** $\mathcal{U} \leftarrow \mathcal{U} \parallel \langle C_{super} \rangle$
fin pour
retourner $\mathcal{U}$

Dans la boucle **tant que** qui commence à la ligne 7 de l'algorithme 3.4, nous effectuons une recherche en profondeur. Pour celle-ci, la liste $\mathcal{L}$ est utilisée comme une pile. Lorsqu'une SCM de f plus petit que f_{min} est trouvée, H est remplacée par C, f_{min} est mis à jour et un nettoyage de $\mathcal{L}$ est effectué. Ensuite, si C n'est pas terminée, ses super-SCM *prometteuses* sont ajoutées dans $\mathcal{L}$. Une SCM est terminée quand elle contient le nombre maximal de caractéristiques ou qu'elle couvre N.

La différence principale entre les algorithmes 3.4 et 3.6 est que dans l'algorithme 3.6 nous retournons une collection $\mathcal{I}$ de SCM et non une seule SCM H. Ensuite, lorsqu'une SCM extraite de $\mathcal{L}$ a une valeur de f équivalente à f_{min}, elle est ajoutée à $\mathcal{I}$. Finalement, les super-SCM potentielles qui ont une valeur de b égale à f_{min} sont ajoutées à $\mathcal{L}$ pour une exploration future. Ceci est nécessaire, puisqu'il est possible que $f(C_{super}) = f_{min}$.

Algorithme 3.6 Étape 3 du *branch-and-bound* en 3 étapes

Entrées : S l'ensemble d'apprentissage
$maxCarac$ le nombre maximal de caractéristiques dans une SCM
H la SCM produite à l'étape 2

Sorties : $\mathcal{I}$ une collection de SCM de f minimale

Utilise : $\mathcal{L}$ une liste de SCM ou chaque SCM satisfait la convention 3
C et C_{super} des SCM satisfaisant la convention 3
f_{min} la valeur de coût la plus petite trouvée

1: $\mathcal{I} \leftarrow \{H\}$
2: $f_{min} \leftarrow f(H)$
3: $H \leftarrow H'$
4: **tant que** $|H| > 0$ **faire**
5: $\quad\mathcal{L} \leftarrow \text{superSCM3}(S, H') - \langle H \rangle$
6: $\quad H \leftarrow H'$
7: $\quad$**tant que** $\mathcal{L} \neq \emptyset$ **faire**
8: $\quad\quad C \leftarrow$ dernière SCM de la liste $\mathcal{L}$
9: $\quad\quad \mathcal{L} \leftarrow \mathcal{L}'$
10: $\quad\quad$**si** $f(C) < f_{min}$ **alors**
11: $\quad\quad\quad \mathcal{I} \leftarrow \emptyset$
12: $\quad\quad\quad f_{min} \leftarrow f(C)$
13: $\quad\quad\quad$ enlever de $\mathcal{L}$ les SCM dont $b > f_{min}$
14: $\quad\quad$**fin si**
15: $\quad\quad$**si** $f(C) = f_{min}$ **alors** $\mathcal{I} \leftarrow \mathcal{I} \cup \{C\}$
16: $\quad\quad$**si** $|C| < maxCarac$ **et** $|N_C| < |N|$ **alors**
17: $\quad\quad\quad$**pour tout** $C_{super} \in \text{superSCM3}(S, C)$ **faire**
18: $\quad\quad\quad\quad$**si** $b(C_{super}) \leq f_{min}$ **alors** $\mathcal{L} \leftarrow \mathcal{L} \parallel \langle C_{super} \rangle$
19: $\quad\quad\quad$**fin pour**
20: $\quad\quad$**fin si**
21: $\quad$**fin tant que**
22: **fin tant que**
23: **retourner** $\mathcal{I}$

La différence majeure entre les étapes 2 et 3 du *branch-and-bound* en trois étapes se trouve dans les algorithmes superSCM2 et superSCM3. Dans superSCM3, nous retrouvons une boucle supplémentaire pour passer en revue les exemples positifs qui peuvent servir de points frontière. Ces exemples doivent être consistants avec la SCM C donnée en entrée. Ils ne doivent pas faire partie de $\mathbf{z_i}(C)$ et ne doivent pas mener à la construction d'une caractéristique qui fait des erreurs avec $\mathbf{z_i}(C) \cap P$.

Algorithme 3.7 superSCM3

Entrées : S l'ensemble d'apprentissage
C une SCM
Sorties : $\mathcal{U}$ une liste de SCM

$\mathcal{U} \leftarrow \langle\rangle$
$L_n \leftarrow \{\mathbf{x}_i : \mathbf{x}_i \in N \wedge \mathbf{x}_i \notin \mathbf{z_i}(C)\}$
pour tout $\mathbf{x}_n \in L_n$ **faire**
  $\rho_{max} \leftarrow \min\{d(\mathbf{x}_i, \mathbf{x}_n) : \mathbf{x}_i \in (P \cap \mathbf{z_i}(C))\}$
  $L_p \leftarrow \{\mathbf{x}_i : \mathbf{x}_i \in P \wedge \mathcal{R}(C)(\mathbf{x}_i) = 1 \wedge \mathbf{x}_i \notin \mathbf{z_i}(C) \wedge d(\mathbf{x}_i, \mathbf{x}_n) \leq \rho_{max}\}$
  pour tout $\mathbf{x}_p \in L_p$ **faire**
    $t \leftarrow$ trou de centre $\mathbf{x}_n$ et de bord $\mathbf{x}_p$
    $C_{super} \leftarrow C \parallel \langle t \rangle$
    si $n_{|C_{super}|} \leq n_{|C|}$ **et** $n_{|C_{super}|} > 2$ **alors** $\mathcal{U} \leftarrow \mathcal{U} \parallel \langle C_{super} \rangle$
  fin pour
fin pour
retourner $\mathcal{U}$

Comme nous n'imposons pas un ordre strict sur les caractéristiques utilisées dans les SCM de ce *branch-and-bound*, il est possible que des SCM équivalentes (qui regroupent les mêmes caractéristiques, mais dans un ordre différent) soient examinées et de surcroît ajoutées dans $\mathcal{I}$ si elles sont optimales. La solution que nous avons retenue pour remédier à cette situation est de filtrer $\mathcal{I}$ *a posteriori*. Un élément possible de solution aurait été d'utiliser une cache peu profonde pour noter les supers-SCM qui ont déjà été générées. Ceci afin de pouvoir vérifier si une super-SCM en cours de génération doit vraiment être examinée.

3.6.3 Résultats

Le fait de disposer d'une borne sur f plus grande que celle du *branch-and-bound* conventionnel nous a permis de traiter des ensembles d'apprentissage plus volumineux en un temps acceptable. Entre autres, nous pouvons maintenant analyser *Sonar* et *Glass* au complet. Avec le *branch-and-bound* conventionnel, nous avions eu à tronquer

l'ensemble d'entraînement de $m = 105$ à $m^* = 50$ pour *Sonar* et de $m = 107$ à $m^* = 60$ pour *Glass* (voir le tableau 3.1).

Il n'a pas été possible de traiter toutes les données d'apprentissage au complet, mais dans chaque cas nous avons réussi à garder la taille de l'ensemble d'entraînement à au moins 100 exemples, ce qui à des fins de classification est une taille raisonnable. Le tableau 3.25 indique le nombre d'exemples utilisés dans les ensembles d'apprentissage. Il y a m exemples dans l'ensemble d'apprentissage original et m^* exemples dans l'ensemble d'apprentissage utilisé. Comme précédemment, les données qui ont été tronquées ont été balancées, autant que possible, en prenant les $m^*/2$ premiers exemples positifs et négatifs. Pour *US Votes*, ce re-balancement a cependant été impossible puisque l'ensemble d'entraînement ne contient que 87 exemples positifs. Nous avons donc conservé tous les exemples positifs et les 100 premiers exemples négatifs. L'avant-dernière colonne du tableau donne la taille des ensembles test. Ensuite, nous avons le nombre de classificateurs différents de même f minimale.

Données	m	m^*	$\lvert T\rvert$	Nb classificateurs
Sonar	105	105	103	102
Breast Cancer	343	343	340	1
Australian Credit	353	110	300	4
Glass	107	107	107	35
Heart Disease	150	100	147	11
US Votes	235	187	200	1

TAB. 3.25 – Tailles des données, *branch-and-bound* en trois étapes

Les paramètres sélectionnées par la minimisation la borne de la proposition 3 pour les SCM simples sont présentés dans le tableau 3.26. Les sélections de modèles se sont déroulées de la même manière que celle décrite à la page 42. Nous avons construit des conjonctions pour les données *Australian Credit* et *US Votes*. Pour les autres données,

Données	Type	Pénalité	Nb caractéristiques
Sonar	disjonction	∞	18
Breast Cancer	disjonction	9	1
Australian Credit	conjonction	5	3
Glass	disjonction	∞	4
Heart Disease	disjonction	2	1
US Votes	conjonction	3	1

TAB. 3.26 – Paramètres des SCM simples

c'était des disjonctions. Nous rappelons que le choix entre conjonction ou disjonction pour le *branch-and-bound* est expliqué à la section 3.2.

C'est pour les données *Sonar* que nous avons le plus grand nombre de SCM dans le vote de majorité. Ceci s'explique en partie par le fait que c'est pour *Sonar* que la SCM simple compte le plus de caractéristiques. Nous pouvons constater que c'est seulement pour des SCM simples à une caractéristique que nous n'avons qu'une SCM optimale selon f.

Nous retrouvons les paramètres choisis pour les SVM par validation croisée dans le tableau 3.27. Les sélections de modèles se sont déroulées de la même manière que celle décrite à la page 42.

Données	γ	C_{svm}
Sonar	0.01	10000
Breast Cancer	0.001	30
Australian Credit	0.001	10
Glass	1	1000
Heart Disease	0.015	5
US Votes	0.8	1

TAB. 3.27 – Paramètres des SVM

Le tableau 3.28 présente les risques empiriques sur l'ensemble test obtenus par trois classificateurs comparés. Nous y retrouvons aussi les limites des intervalles de confiance construits comme indiqué à la section 2.5 avec $\delta = 0.1$. Dans la dernière colonne, il y a la valeur de f pour le classificateur (si applicable). Le classificateur du *branch-and-bound* est un vote de majorité à partir des SCM de même f optimale. Nous n'avons pas reproduit ici les résultats du *branch-and-bound* conventionnel (tableau 3.4) parce que les ensembles d'entraînement ne sont pas les mêmes et que les SCM admissibles proviennent de deux conventions différentes. Dans cette section, l'acronyme BB3E signifie « *branch-and-bound* en trois étapes ».

Données	Classificateur	$R_T(h)$	I.C. Inf.	I.C. Sup.	$f(\mathbf{z_i}, \sigma)$
Sonar	SCM	0.233	0.166	0.312	36
	SVM	0.087	0.046	0.148	.
	BB3E	0.272	0.201	0.353	25
Breast Cancer	SCM	0.035	0.020	0.057	6
	SVM	0.035	0.020	0.057	.
	BB3E	0.035	0.020	0.057	6
Australian Credit	SCM	0.203	0.166	0.245	16
	SVM	0.133	0.102	0.170	.
	BB3E	0.207	0.169	0.249	14
Glass	SCM	0.206	0.143	0.280	13
	SVM	0.215	0.152	0.291	.
	BB3E	0.178	0.120	0.250	13
Heart Disease	SCM	0.156	0.109	0.214	19
	SVM	0.170	0.121	0.229	.
	BB3E	0.177	0.127	0.237	19
US Votes	SCM	0.105	0.071	0.148	10
	SVM	0.190	0.146	0.241	.
	BB3E	0.105	0.071	0.148	10

TAB. 3.28 – Risques empiriques sur les ensembles test des SCM simples, SVM et votes de majorité construits par des BB3E

Les résultats sont abordés ci-bas en procédant par données d'apprentissage.

Sonar

Nous pouvons observer à la figure 3.9 que le risque réel de la SVM sur l'ensemble test est significativement plus faible que celui des deux autres classificateurs. Cependant, nous nous intéressons plus aux différences entre la SCM et le vote de majorité.

Il y a 103 exemples dans l'ensemble test. Sur ceux-ci, la SCM fait 24 erreurs, la SVM en fait 9 et le vote de majorité 28. Nous pouvons constater que, malgré l'écart spectaculaire entre la valeur de f pour la SCM simple et la valeur de f pour les SCM du vote de majorité, le risque empirique sur T du vote de majorité est tout de même plus haut. Ceci démontre que la borne Marchand-Sokolova n'est pas un bon indicateur de performance sur ces données.

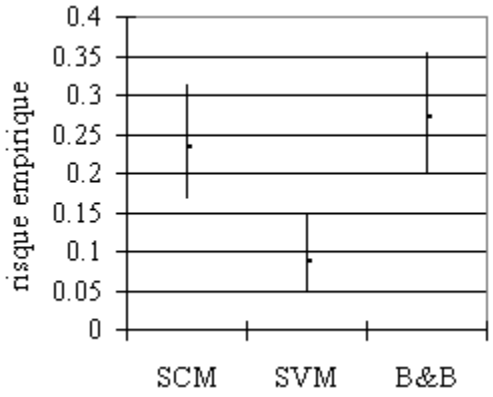

FIG. 3.9 – Intervalles de confiance pour les $R(h)$ de la SCM simple, de la SVM et du vote de majorité construit par un BB3E pour les données *Sonar*

Le risque empirique de Gibbs $R_T(G_Q)$ sur l'ensemble test est de 0.263. La contribution $R_T(G_Q) - R_T(B_Q)$ du vote de majorité est donc très légèrement négative, car $R_T(B_Q) = 0.272$. Il y a 48 votants sur 102 qui ont un risque empirique individuel sur T plus petit que $R_T(B_Q)$, 12 votants dont $R_T(h) = R_T(B_Q)$ et 42 votants dont $R_T(h) > R_T(B_Q)$.

Les tableaux 3.29 et 3.30 résument le nombre de votants qui font une erreur par exemple. Pour l'ensemble test, il y a 59 exemples pour lesquels aucun votant ne se trompe et 16 exemples pour lesquels il y a entre 6 et 48 votants qui se trompent. Ces tableaux montrent les nombres de votants erronés uniques pour lesquels il y a erreur nulle ou totale (0 ou 102). Les relativement grandes valeurs observées aux extrémités dans le tableau 3.30 indiquent qu'il y a une certaine uniformité entre votants.

Nombre de votants erronés	Nombre d'exemples concernés
0	86
6 à 48	7
49 à 53	0
54 à 84	6
102	6
total	105

TAB. 3.29 – Erreurs par exemple pour l'ensemble S de *Sonar*

Nombre de votants erronés	Nombre d'exemples concernés
0	59
6 à 48	16
49 à 53	0
54 à 84	14
102	14
total	103

TAB. 3.30 – Erreurs par exemple pour l'ensemble T de *Sonar*

Parmi les 102 votants, qui ont bel et bien des ensembles de compression différents, nous ne retrouvons que 10 façons de voter différentes sur l'ensemble test. Le tableau

Groupe	1	2	3	4	5	6	7	8	9	10	total
Taille	6	6	12	6	24	12	12	6	6	12	102

TAB. 3.31 – Nombre de SCM de mêmes votes, *Sonar*, BB3E

3.31 donne plus d'information. Il y a, par exemple, 6 votants qui votent identiquement de la manière 1, 6 de la manière 2, 12 de la manière 3 et ainsi de suite. De plus, entre ces 10 vecteurs de votes $h_j(T)$, nous retrouvons de 1 à 27 différences.

Le nombre maximal de différences assez élevé est reflété par la corrélation moyenne entre paires de vecteurs de votes qui est de 0.759. Une valeur qui est relativement basse. Les $\binom{102}{2}$ corrélations calculées sont résumées à la figure 3.10. Nous pouvons observer une distribution bimodale pour celles-ci. Ce qui pourrait nous mener à présumer que les votants sont soit très semblables ou peu semblables.

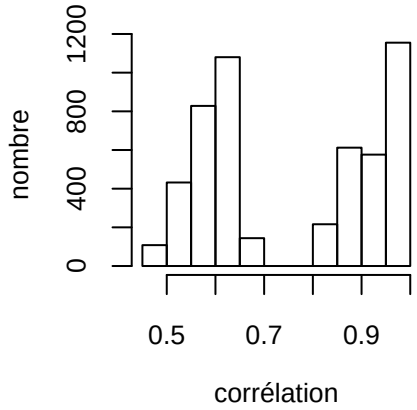

FIG. 3.10 – Corrélations, *Sonar*, BB3E

Breast Cancer

Pour ces données, les risques empiriques sur l'ensemble test sont égaux pour les classificateurs produits par les 3 algorithmes. La seule SCM trouvée par le *branch-and-bound* est la même que celle construite par l'algorithme SCM-classique. Nous n'y retrouvons qu'une seule caractéristique. L'ensemble test compte 340 exemples, donc le risque empirique sur T observé correspond à 12 erreurs.

Australian Credit

Il n'y a pas de différence significative entre le vrai risque de la SCM simple et celui du vote de majorité. Cependant, celui du vote est presque significativement plus grand que celui de la SVM. Les intervalles de confiance se chevauchent, mais seulement par 0.01. Il y a 300 exemples dans l'ensemble test. Sur ceux-ci, la SCM fait 61 erreurs, la SVM 40 et le vote 62. La figure 3.11 montre les intervalles de confiance pour les $R(h)$ des trois classificateurs comparés.

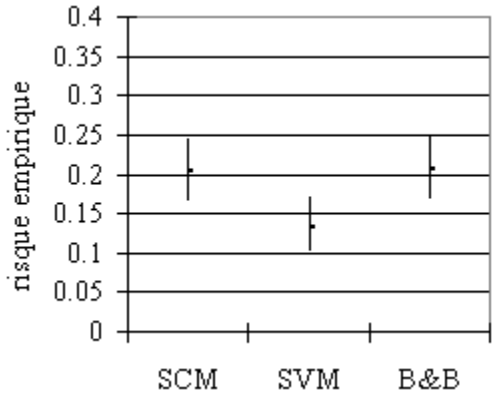

FIG. 3.11 – Intervalles de confiance pour les $R(h)$ de la SCM simple, de la SVM et du vote de majorité construit par un BB3E pour les données *Australian Credit*

Nous observons pour le vote de majorité un risque empirique de Gibbs $R_T(G_Q)$ de 0.207. C'est exactement le même que le risque empirique de Bayes sur T. La contribution du vote de majorité est donc neutre. Il y a un votant qui possède un $R_T(h)$ plus élevé que $R_T(B_Q)$, un votant qui possède un $R_T(h)$ plus bas et les deux autres ont $R_T(h) = R_T(B_Q)$.

Les tableaux 3.32 et 3.33 donnent le nombre d'erreurs par exemple. Pour l'ensemble test, il y a 234 exemples sur lesquels aucun votant ne se trompe, 2 exemples sur lesquels 1 votant se trompe et ainsi de suite. Parmi les exemples où il y a un vote balancé, c'est-à-dire où il y a autant de votants qui votent positif que de votants qui votent négatif, il y a deux exemples de classe réelle positive et deux de classe réelle négative. Donc, pour ces quatre exemples, il n'y a que 2 erreurs de comptées. Les valeurs observées dans le tableau 3.33 dénotent un haut degré de cohésion entre les votants.

Nombre de votants erronés	Nombre d'exemples concernés
0	99
1	4
2	0
3	0
4	7
total	60

TAB. 3.32 – Erreurs par exemple pour l'ensemble S de *Australian Credit*

Nombre de votants erronés	Nombre d'exemples concernés
0	234
1	2
2	4
3	2
4	58
total	300

TAB. 3.33 – Erreurs par exemple pour l'ensemble T de *Australian Credit*

Les quatre votants votent tous de manière différente sur T. Les tableaux 3.34 et 3.35 donnent les nombres de différences entre vecteurs de votes pris deux à deux et les corrélations entre ces mêmes vecteurs. Les différences sont très peu nombreuses et ceci est reflété dans la grandeur des corrélations.

		SCM		
		4	3	2
SCM	1	6	7	1
	2	5	6	
	3	3		

TAB. 3.34 – Nombre de votes différents, *Australian Credit*, BB3E

		SCM		
		4	3	2
SCM	1	0.961	0.954	0.993
	2	0.967	0.960	
	3	0.980		

TAB. 3.35 – Corrélations, *Australian Credit*, BB3E

Glass

Avec ces données, aucune différence significative de vrai risque sur l'ensemble test n'est observée. Celui-ci compte 107 exemples. La SCM simple fait 22 erreurs de classification, la SVM en fait 23 et le vote de majorité 19. C'est la seule occasion où nous observons moins d'erreurs pour le vote. La figure 3.12 montre les intervalles de confiance pour les $R(h)$ des trois classificateurs comparés.

Le risque empirique de Gibbs $R_T(G_Q)$ sur l'ensemble test est de 0.198. La contribution $R_T(G_Q) - R_T(B_Q)$ du vote de majorité est donc positive, car le risque empirique de Bayes sur T est de 0.178. Il y a 4 votants avec $R_T(h) < R_T(B_Q)$, 4 votants avec $R_T(h) = R_T(B_Q)$ et 27 votants avec $R_T(h) > R_T(B_Q)$.

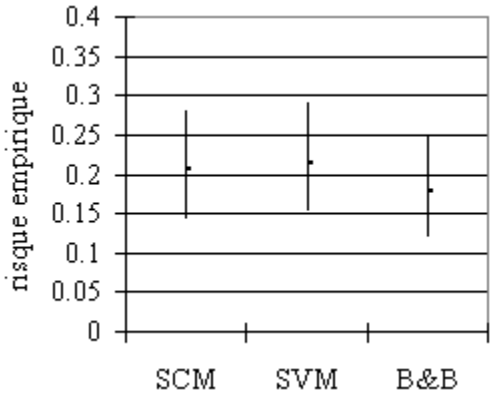

FIG. 3.12 – Intervalles de confiance pour les $R(h)$ de la SCM simple, de la SVM et du vote de majorité construit par un BB3E pour les données *Glass*

Les tableaux 3.36 et 3.37 donnent le nombre d'erreurs par exemple. Pour l'ensemble test, il y a 78 exemples où aucun votant ne se trompe, 10 où de 6 à 17 votants se trompent et ainsi de suite. Les valeurs observées indiquent que les votants présentent une quantité modérée de différences dans leur expression.

Nombre de votants erronés	Nombre d'exemples concernés
0	99
6 à 12	3
18	1
23	1
35	3
total	107

TAB. 3.36 – Erreurs par exemple pour l'ensemble S de *Glass*

Nombre de votants erronés	Nombre d'exemples concernés
0	78
6 à 17	10
18 à 22	0
23 à 24	3
35	16
total	107

TAB. 3.37 – Erreurs par exemple pour l'ensemble T de *Glass*

Parmi les 35 votants, il n'y a pas plus de 18 manières de voter différentes sur T. Il y a une manière de voter qui ne compte qu'un seul votant, tandis que toutes les autres manières de voter comptent 2 votants. Le nombre de différences entre vecteurs de votes différents varie entre 1 et 12 inclusivement.

Les $\binom{35}{2}$ corrélations calculables entre vecteurs de votes sont résumées dans l'histogramme de la figure 3.13. Celles-ci sont réparties sur un spectre plutôt large. La majorité des corrélations est autour de la moyenne qui est de 0.890. Comparée aux autres moyennes, c'est une valeur ni très grande, ni très petite.

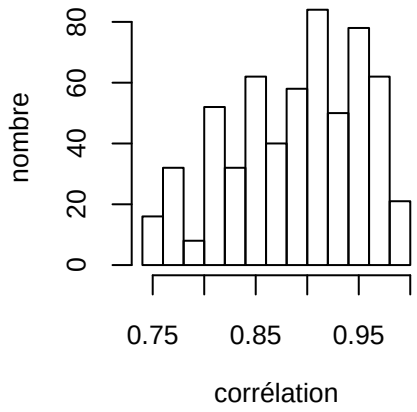

FIG. 3.13 – Corrélations, *Glass*, BB3E

Heart Disease

Aucune différence significative de vrai risque n'a été observée pour ces données. L'ensemble test compte 147 exemples et sur ceux-ci la SCM fait 23 erreurs de classification, la SVM en fait 25 et le vote de majorité en fait 26. La figure 3.14 montre les intervalles de confiance pour les $R(h)$ des trois classificateurs comparés.

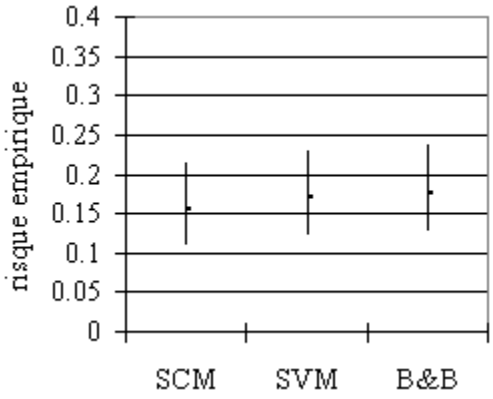

FIG. 3.14 – Intervalles de confiance pour les $R(h)$ de la SCM simple, de la SVM et du vote de majorité construit par un BB3E pour les données *Heart Disease*

Nous avons $R_T(G_Q) = 0.183$. La contribution du vote de majorité est positive, car $R_T(B_Q) = 0.177$. Il y a 7 votants qui ont $R_T(h) > R_T(G_Q)$ et les quatre autres ont $R_T(h) < R_T(G_Q)$.

Les tableaux 3.38 et 3.39 présentent le nombre d'erreurs par exemple. Pour l'ensemble test, il y a 101 exemples où aucun classificateur ne se trompe, 20 exemples où

de 1 à 5 classificateurs se trompent et ainsi de suite. La répartition des nombres de classificateurs erronés indique qu'il y a un bon degré de cohérence entre classificateurs.

Nombre de votants erronés	Nombre d'exemples concernés
0	74
1 et 2	10
5	1
6 et 7	0
8 à 10	6
11	9
total	107

TAB. 3.38 – Erreurs par exemple pour l'ensemble S de *Heart Disease*

Nombre de votants erronés	Nombre d'exemples concernés
0	101
1 à 5	20
6 à 10	12
11	14
total	147

TAB. 3.39 – Erreurs par exemple pour l'ensemble T de *Heart Disease*

Parmi les 11 votants, il n'y en a que deux qui votent de la même manière sur T. Entre les votants différents, nous observons de 1 à 21 différences. Les corrélations calculées entre les $\binom{11}{2}$ paires de vecteurs de votes sont résumées dans l'histogramme de la figure 3.15. Elles sont relativement peu élevées et leur moyenne est de 0.880.

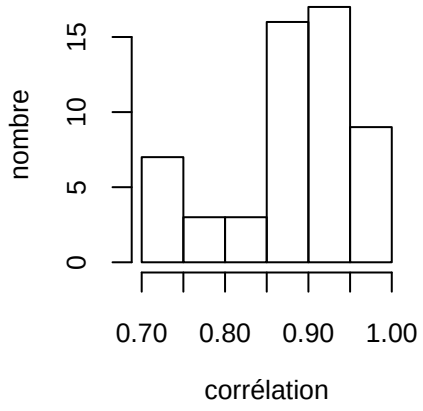

FIG. 3.15 – Corrélations, *Heart Disease*, BB3E

US Votes

En optimisant f, nous n'avons obtenu qu'un seul classificateur. C'est la même SCM qui est obtenue par l'algorithme SCM-classique. Elle ne compte qu'une seule caractéristique. Sur les 200 exemples de l'ensemble test, elle fait 21 erreurs tandis que la SVM

en fait 38. La figure 3.16 montre les intervalles de confiance pour les $R(h)$ des trois classificateurs comparés.

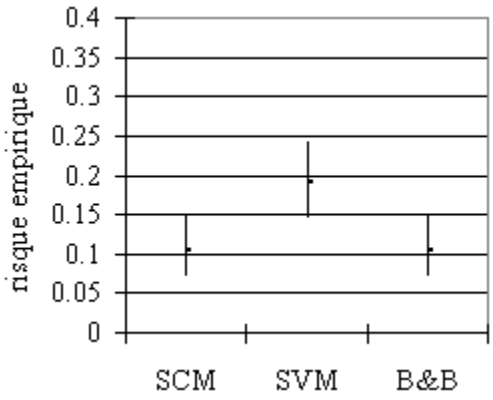

FIG. 3.16 – Intervalles de confiance pour les $R(h)$ de la SCM simple, de la SVM et du vote de majorité construit par un BB3E pour les données *US Votes*

Facteurs explicatifs

Voici réunies dans une sous-section les mesures uniques globales qui nous donnent de l'information sur tous les votants d'un vote de majorité et sur leur synergie. Nous retrouvons dans le tableau 3.40 :

- La borne sur le risque estimée $\widehat{C_Q}$ de la proposition 5.
- La moyenne des corrélations $\bar{r}$ introduite à la page 27 de la section 2.9.3.
- La variance des $\widehat{W_Q}$ présentée dans la même section.
- Le risque empirique de Bayes sur l'ensemble test $R_T(B_Q)$.
- Le risque empirique de Gibbs sur l'ensemble test $R_T(G_Q)$.
- Un indicateur de la contribution du vote $R_T(G_Q) - R_T(B_Q)$.

Si nous regardons les valeurs de $\bar{r}$, les votants les plus semblables seraient produits avec *Australian Credit*, tandis que les moins semblables seraient produits avec *Sonar*. Selon notre indicateur de contribution de vote, le meilleur effet de vote est obtenu avec *Glass*, tandis que le pire, et le seul négatif, est obtenu avec *Sonar*.

À la figure 3.17, nous pouvons observer une tendance en reliant $\widehat{C_Q}$ et $R_T(B_Q)$. D'autre part $R_T(B_Q)$ semble vaguement inversement proportionnel à $\bar{r}$, mais nous avons peu de points et ceux-ci sont loin d'être parfaitement alignés. L'indicateur de contribution de vote est encore une fois donné à titre indicatif, car aucune relation évidente n'a été observée entre celui-ci et la corrélation moyenne ou la variance des $\widehat{W_Q}$.

Données	$\widehat{C_Q}$	$\bar{r}$	Variance des $\widehat{W_Q}$
Sonar	0.706	0.759	0.135
Australian Credit	0.648	0.969	0.159
Glass	0.597	0.890	0.135
Heart Disease	0.550	0.880	0.123

Données	$R_T(G_Q)$	$R_T(B_Q)$	$R_T(G_Q) - R_T(B_Q)$
Sonar	0.263	0.272	-0.009
Australian Credit	0.207	0.207	0.000
Glass	0.198	0.178	0.020
Heart Disease	0.183	0.177	0.006

TAB. 3.40 – Facteurs explicatifs et risques empiriques ensemble test, BB3E

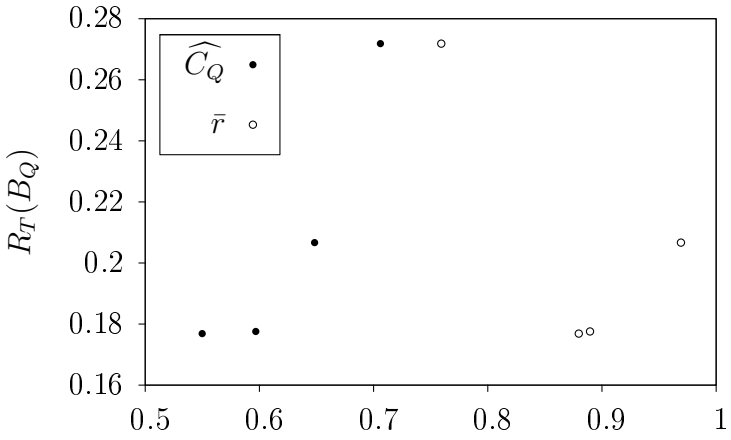

FIG. 3.17 – Facteurs explicatifs pour $R_T(B_Q)$, BB3E

3.6.4 Résumé

Avec le *branch-and-bound* en trois étapes, il y a eu un gain indiscutable en capacité de traitement. Cependant, les conclusions à tirer ne sont pas différentes de celles pour le *branch-and-bound* conventionnel. Nulle part le vrai risque d'une SCM simple est significativement différent de celui du classificateur par vote de majorité. Donc, l'algorithme SCM-classique nous donne des SCM qui sont déjà optimales selon f ou, à tout le moins, des SCM aussi performantes que les SCM optimales selon f. Quoiqu'en apparence négatif, ce résultat est néanmoins important puisqu'il tend à confirmer empiriquement que l'heuristique de l'algorithme SCM-classique est non seulement rapide, mais également performant.

3.7 Conclusion

À performance équivalente sur l'ensemble test, le temps de construction d'un classificateur par *branch-and-bound* est difficilement justifiable comparé à celui d'une SCM ordinaire. Le degré d'optimalité des classificateurs produits par l'algorithme SCM-classique, selon f, est maintenant connu grâce aux travaux effectués dans ce chapitre. Nous savons que l'algorithme SCM-classique nous donne des classificateurs optimaux, ou sinon aussi intéressant que les optimaux selon f. Sans oublier que l'optimalité selon f est très proche de celle selon la borne de la proposition 3. Ce résultat fut une surprise. En effet, rien *a priori* ne laissait supposer que la simple heuristique de pénalité de SCM-classique soit aussi performante. Il était donc important d'établir ce fait (au moins de façon empirique). C'est pour cette raison que les résultats obtenus dans le cadre de cette maîtrise sont importants pour les utilisateurs futurs des SCM. Nous pouvons lire dans Marchand et Shawe-Taylor (2002) [3] que la sélection de modèle par la borne produit des classificateurs aussi performants qu'en effectuant la sélection par validation croisée (en faisant moins de calculs). Maintenant, la sélection de modèle par la borne utilisée en combinaison avec l'heuristique est avantageuse par un aspect supplémentaire.

Chapitre 4

Bagging

4.1 Introduction

L'utilisation d'une méthode d'agrégation pour utiliser toutes les SCM optimales obtenues par un *branch-and-bound* s'est décidée pendant l'implémentation de l'algorithme, car une présomption naturelle, mais erronée, est qu'il ne produira qu'une seule SCM par ensemble d'entraînement. Dans le cas présent, il est clairement défini dans l'objectif que nous voulons combiner plusieurs SCM dans un classificateur.

Comme mentionné lors de l'introduction de ce mémoire, deux possibilités pour faire des votes de majorité étaient le *bagging* et le *boosting*. Le principe du *bagging* est plutôt simple et ne présente pas de difficulté majeure pour l'implémentation. Le défi réside dans la sélection des paramètres optimaux pour construire les votants, car nous avons un grand éventail de possibilités.

Le *bagging* a été présenté par Breiman en 1996 [8]. Le terme est une contraction de « bootstrap aggregating ». Nous utilisons une méthode nommée *bootstrapping* pour générer différents ensembles d'apprentissage à partir de l'ensemble original et un votant est construit à partir de chaque nouvel ensemble. Ensuite, nous regroupons les votants dans un vote de majorité pour décider la classe retournée pour chaque exemple de l'ensemble test.

Le *bootstrapping* est une méthode de rééchantillonnage avec remise. Dans le cas présent, des exemples sont tirés de l'échantillon d'apprentissage S pour créer les nouveaux ensembles. Ceux-ci sont nommés échantillons *bootstrap*. Comme ces échatillons

sont construit avec remise, des exemples de S risquent de s'y retrouver en plusieurs exemplaires. La partie d'agrégation est le vote de majorité.

Étant donné que les SCM sont définies en utilisant des points de l'ensemble d'apprentissage, un échantillon issu de la même distribution mènera à la construction d'une SCM complètement différente. Les zones couvertes par les caractéristiques devraient se trouver en des endroits semblables, mais la dépendance des SCM aux données introduit une certaine instabilité. Le *bagging* a été reconnu comme étant une méthode stabilisatrice, donc son utilisation avec des SCM semble indiquée.

4.2 Algorithme

Afin de bénéficier des votants les mieux adaptés aux données, nous avons décidé de construire les SCM votantes avec des modèles sélectionnés à partir de leurs échantillons *bootstrap* respectifs. Il y a une sélection de modèle effectuée à chaque échantillon. La sélection comprend les choix suivants :

- Le type de la SCM qui est soit une conjonction, soit une disjonction.
- La pénalité utilisée dans l'algorithme SCM-classique. Celle-ci est décrite à la page 18 de la section 2.9.1 sur les SCM.
- Le nombre de caractéristiques présentes dans la SCM. Dans le cadre actuel, lorsqu'il est mention de la taille d'une SCM, il s'agit du nombre de caractéristiques.

La meilleure combinaison de ces paramètres est sélectionnée en utilisant la borne de la proposition 3. Toutes les combinaisons sont essayées et celle qui est choisie minimise la borne. Dans l'algorithme 4.1 cette sélection est effectuée dans la boucle **pour tout**.

Dans les entrées de l'algorithme, nous retrouvons :

- Le taux d'échantillonnage est un nombre $\in (0, 1]$ qui spécifie la fraction de la taille de S à être utilisée pour calculer la taille des échantillons *bootstrap*.
- $maxCarac$ est le nombre commun maximal de caractéristiques pour la construction de tous les votants.
- L_p est un ensemble de pénalités candidates à être essayées lors de la sélection de modèle de chaque votant.
- (1 - δ) est le niveau de confiance tel qu'il est défini à la section 2.8 concernant la borne sur le risque pour les classificateurs comprimant les données.

Algorithme 4.1 Bagger

Entrées : S l'échantillon d'apprentissage
$nbVotants$ le nombre de votants
$tauxÉchan$ le taux d'échantillonnage
$maxCarac$ le nombre maximal de caractéristiques
L_p un ensemble de pénalités
$\delta = 1$ moins le niveau de confiance de la borne

Sorties : une collection $\mathcal{B}$ de SCM

$\mathcal{B} \leftarrow \emptyset$
$m' \leftarrow \lfloor tauxÉchan \times |S| \rfloor$
pour $v = 1$ à $nbVotants$ **faire**
  $S' \leftarrow$ échantillon de taille m' tiré uniformément de S avec remise.
  $\epsilon_{meilleur} \leftarrow \infty$
  pour tout $type \in$ {conjonction, disjonction} **et pour tout** $p \in L_p$ **faire**
    si $type =$ conjonction **alors** $C \leftarrow$ SCM_conj$(P', N', p, maxCarac)$
    sinon $C \leftarrow$ SCM_disj$(P', N', p, maxCarac)$
    pour $taille = 1$ à $|C|$ **faire**
      $C' \leftarrow \cup_{i=1}^{taille} g_{C_i}$
      $borne \leftarrow \epsilon(|S'|, |S'|R_{S'}(C'), \sigma(C'), \mathbf{z_i}(C'), \delta)$
      si $borne < \epsilon_{meilleur}$ **alors**
        $C_{meilleur} \leftarrow C'$
        $\epsilon_{meilleur} \leftarrow borne$
      fin si
    fin pour
  fin pour
  $\mathcal{B} \leftarrow \mathcal{B} \cup \{C_{meilleur}\}$
fin pour
retourner $\mathcal{B}$

L'algorithme nous donne une collection de SCM que nous utiliserons comme votants. Au début, m' est la taille calculée de chaque échantillon bootstrap. Un tirage uniforme des exemples signifie que tous les exemples ont la même probabilité d'être tirés.

La boucle **pour tout** passe en revue toutes les combinaisons possibles de type et de pénalité. Les fonctions « SCM_conj » et « SCM_disj » sont appelées pour la construction d'une SCM $C = \{g_{C_1}, \ldots, g_{C_{|C|}}\}$; C étant une liste de caractéristiques. « SCM_conj » est l'algorithme 2.2 avec, pour la circonstance, un nom plus court. Celui-ci produit une conjonction. « SCM_disj » produit une disjonction en inversant les classes de P' et N', en construisant une conjonction à partir de ces données, en inversant les classes des exemples de l'ensemble de compression et en remplaçant les $\wedge$ par des $\vee$ dans

la conjonction. P' désigne le sous-ensemble des exemples positifs de S' et N' désigne le sous-ensemble des exemples négatifs. Nous avons $P' \cup N' = S'$.

À la ligne 10, nous construisons une sous-SCM de C qui contient ses *taille* premières caractéristiques. Ainsi, pour une combinaison de type et de pénalité donnée, nous préservons la SCM de taille optimale selon la borne. À la ligne suivante, nous calculons la valeur de la borne telle que définie dans la proposition 3.

- $R_{S'}$ est une fonction qui reçoit en entrée une SCM et qui retourne le risque empirique sur S'.
- L'expression $|S'|R_{S'}(C')$ signifie : le nombre d'erreurs sur S' par le classificateur C'.
- $\mathbf{z_i}(C')$ est l'ensemble de compression de la SCM C'.
- $\sigma(C')$ est le message d'information supplémentaire de la SCM C'.

Après avoir calculé la borne, nous la comparons avec la plus basse valeur trouvée. Nous conservons la dernière SCM construite si la nouvelle valeur est encore plus basse. Lorsque nous avons regardé toutes les combinaisons de type et de pénalité, nous ajoutons à $\mathcal{B}$ la meilleure SCM trouvée selon la borne.

4.3 Résultats

Les votants sont construits de telle façon qu'il y a une partie de la sélection de modèle qui est autonome. Cependant, des choix ont dû être faits pour certains paramètres de l'algorithme. Il a été nécessaire de :

- Déterminer le nombre de votants.
- Déterminer le taux d'échantillonnage.
- Décider si nous allions permettre l'utilisation de caractéristiques avec plus d'un type de centre.
- Décider si nous allions utiliser des conjonctions, des disjonctions, ou les deux.

Afin de limiter les essais, les valeurs possibles des paramètres ont été testées séparément pour chaque paramètre. Analyser toutes les combinaisons possibles de valeurs de paramètres aurait été fastidieux. Le nombre de votants a été choisi en observant, pour toutes les données, le risque empirique sur l'ensemble test en fonction du nombre de votants. 501 votants ont été entraînés sur chaque ensemble d'entraînement et les risques empiriques des votes de majorité de 2 à 501 votants ont été calculés. 251 votants semble

être le nombre pour lequel le risque empirique sur l'ensemble test est stabilisé et cesse de diminuer, pour tous les ensembles. Nous n'avons pas observé de relation entre $|S|$ et les points de stabilisation propres à chaque ensemble.

Cette façon de choisir le nombre de votants est la même que dans l'article de Breiman [8]. Une grande différence est observable dans les conclusions puisqu'il mentionne que 10 votants ont été suffisants dans le cadre de ses expérimentations. Dans un article de Freund et Schapire [26], 100 votants sont utilisés pour le *bagging*, mais il ne semble pas y avoir de justification. Quinlan [27], dans un article traitant en partie du *bagging*, se réfère à Breiman et utilise le même nombre.

Le taux d'échantillonnage utilisé est de 1. C'est le taux utilisé par Breiman [8]. Un taux trop petit donnerait des échantillons qui représenteraient mal la distribution ayant produit S et un taux trop grand donnerait des échantillons produisant toujours la même SCM.

Les deux derniers points (de l'énumération ci-haut) produisent des versions annexes de l'algorithme plutôt que d'être des paramètres à proprement parler. Dans l'implémentation, nous pouvons tout de même les sélectionner par validation croisée.

Permettre d'utiliser des boules et des trous dans chaque SCM a semblé systématiquement plus intéressant pour toutes les données. De manière analogue, permettre un mélange de conjonctions et de disjonctions dans les votants a aussi semblé généralement au moins aussi bon que de n'utiliser qu'un seul type. Ceci est cohérent avec l'idée selon laquelle le *bagging* améliore significativement un algorithme lorsque ce dernier a tendance à produire un classificateur différent pour un échantillon d'apprentissage différent.

Finalement, pour le *bagging*, $maxCarac$ a été fixé à 25 et en aucun cas les votants n'ont atteint cette limite. La liste de pénalité contenait ∞, 10.0, 9.0, 8.0, 7.0, 6.0, 5.0, 4.0, 3.0, 2.0, 1.5, 1.0, 0.9, 0.8, 0.7, 0.6 et 0.5. La valeur utilisée de δ était 0.05.

Nous appliquons le qualificatif « simple » à un classificateur individuel qui ne participe pas à un *bagging*. Nous comparons dans cette section le *bagging* de SCM avec les SCM simples et les SVM simples (voir la section 2.9.2 pour une description complète). L'objectif de cette manœuvre est de positionner le *bagging* de SCM relativement à des classificateurs reconnus et performants. Les classificateurs sont respectivement construits à partir des mêmes ensembles d'exemples et testés sur les mêmes ensembles. Pour tous les essais avec le *bagging*, nous n'avons pas eu à tronquer les ensembles d'entraînement.

Le tableau 4.1 donne les paramètres utilisés pour construire les SCM simples. Une sélection de modèle par la borne de la proposition 3 a été effectuée. Les paramètres qui donnaient une SCM avec la plus petite borne étaient retenus. Pour que la comparaison entre les votants et les SCM simples soit équitable, la sélection de modèle pour les SCM simples a inclus la possibilité de mélanger les boules et les trous. Cependant, la borne a été systématiquement plus petite pour les SCM de centre de classe unique. Les pénalités regardées ont encore été ∞, 10.0, 9.0, 8.0, 7.0, 6.0, 5.0, 4.0, 3.0, 2.0, 1.5, 1.0, 0.9, 0.8, 0.7, 0.6 et 0.5. Le nombre maximal de caractéristiques (25) était le même que pour les votants.

Données	Type	Pénalité	Nb caractéristiques
Sonar	disjonction	∞	18
Breast Cancer	disjonction	9	1
Australian Credit	disjonction	3	3
Glass	disjonction	∞	4
Heart Disease	conjonction	1	1
US Votes	disjonction	2	1

TAB. 4.1 – Paramètres des SCM simples

Le tableau 4.2 indique les paramètres choisis pour construire les SVM. La sélection de modèle a été effectuée par validation croisée 10 fois. Les combinaisons de valeurs de γ pour le noyau RBF et de C_{svm} à intervalles réguliers ont été essayées et un minimum local au R^k_{CV} a ensuite été trouvé. Les valeurs de départ pour γ étaient 0.001, 0.01, 0.1, 1, 10, 100 et 1000. Les valeurs de départ pour C_{svm} étaient 1, 10, 100, 1000 et 10000. Si, par exemple, le plus petit R^k_{CV} était observé pour $\gamma = 0.01$ et $C_{svm} = 10$, les nouvelles valeurs de γ à être essayées étaient 0.002, 0.004, 0.006, 0.008, 0.01, 0.02, 0.04, 0.06 et 0.08. Tandis que les nouvelles valeurs de C_{svm} à être essayées étaient 2, 4, 6, 8, 10, 20, 40, 60 et 80. Encore une fois, il y aurait un R^k_{CV} calculé pour chaque combinaison possible. Ensuite, des valeurs intermédiaires plus fines pouvaient être examinées.

Données	γ	C_{svm}
Sonar	0.01	10000
Breast Cancer	0.001	30
Australian Credit	0.001	1400
Glass	1	1000
Heart Disease	0.002	7
US Votes	0.1	1000

TAB. 4.2 – Paramètres des SVM

Le tableau 4.3 donne les tailles des ensembles d'entraînement S et des ensembles test T pour toutes les données utilisées au cours des tests. Une description détaillée des attributs présents dans ces données est disponible à la section 2.2.2.

Données	$\|S\|$	$\|T\|$
Sonar	105	103
Breast Cancer	343	340
Australian Credit	353	300
Glass	107	107
Heart Disease	150	147
US Votes	235	200

TAB. 4.3 – Tailles des données, *bagging*

4.3.1 Analyse

Nous pouvons retrouver tous les risques empiriques sur les ensembles test de tous les classificateurs dans le tableau 4.4. Celui-ci contient aussi les limites inférieures et supérieures des intervalles de confiance bilatéraux pour le vrai risque calculées selon la section 2.5 avec $\delta = 0.1$. Les mêmes valeurs sont affichées ci-dessous sous forme graphique, séparément pour chaque ensemble de données, avec une analyse détaillée.

Données	Classificateur	$R_T(h)$	I.C. Inf.	I.C. Sup.
Sonar	SCM	0.233	0.166	0.312
	SVM	0.087	0.046	0.148
	BAG	0.223	0.158	0.301
Breast Cancer	SCM	0.035	0.020	0.057
	SVM	0.035	0.020	0.057
	BAG	0.032	0.018	0.053
Australian Credit	SCM	0.193	0.157	0.235
	SVM	0.160	0.126	0.199
	BAG	0.150	0.117	0.188
Glass	SCM	0.206	0.143	0.280
	SVM	0.215	0.152	0.291
	BAG	0.187	0.127	0.260
Heart Disease	SCM	0.156	0.109	0.214
	SVM	0.136	0.092	0.192
	BAG	0.177	0.127	0.237
US Votes	SCM	0.115	0.080	0.159
	SVM	0.060	0.035	0.095
	BAG	0.065	0.039	0.101

TAB. 4.4 – Risques empiriques sur les ensembles test des SCM simples, SVM et votes de majorité construits par des *baggings*

Sonar

De toutes les données traitées dans la cadre du *bagging*, c'est uniquement pour *Sonar* que nous avons l'occasion d'observer un vrai risque significativement plus petit pour la SVM. L'ensemble contient 300 exemples sur lesquels la SCM fait 24 erreurs de classification, la SVM 9 erreurs et le vote de majorité 23. À une erreur près, le vote et la SCM simple sont équivalents. La figure 4.1 montre les intervalles de confiance pour les $R(h)$ des trois classificateurs comparés.

Le risque empirique de Gibbs sur l'ensemble test $R_T(G_Q)$ est de 0.306. L'effet du vote de majorité est donc très positif puisque le risque de Bayes sur l'ensemble test $R_T(B_Q)$ est de 0.223. Il a 12 votants avec un risque empirique individuel sur l'ensemble test plus petit que $R_T(B_Q)$, 3 votants avec un risque empirique sur T égal et 236 votants avec un risque empirique sur T plus grand.

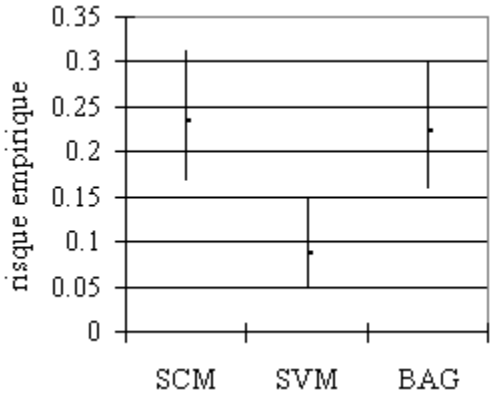

FIG. 4.1 – Intervalles de confiance pour les $R(h)$ de la SCM simple, de la SVM et du vote de majorité construit par *bagging* pour les données *Sonar*

Les tableaux 4.5 et 4.6 présentent les nombres de votants erronés par exemple. Pour l'ensemble test, il n'y a aucun exemple pour lequel aucun votant ne se trompe, il y a 80 exemples pour lesquels il y a entre 6 et 123 votants qui se trompent et ainsi de suite. Il n'y a pas de groupe d'exemples pour lequel nous avons un vote unanime.

Nombre de votants erronés	Nombre d'exemples concernés
0 à 2	0
3 à 125	102
126 à 132	0
133 à 149	3
150 à 251	0
total	105

TAB. 4.5 – Erreurs par exemple pour l'ensemble S de *Sonar*

Nombre de votants erronés	Nombre d'exemples concernés
0 à 5	0
6 à 123	80
124 et 125	0
126 à 188	23
189 à 251	0
total	103

TAB. 4.6 – Erreurs par exemple pour l'ensemble T de *Sonar*

Les vecteurs de votes, c'est-à-dire les $h_j(T)$ tels que définis à la page 27 de la section 2.9.3, sont presque tous différents. Il n'y en a que deux qui sont identiques. C'est un très petit nombre si l'on considère le nombre de votants.

Ces vecteurs sont les moins corrélés de toutes les données. Leur corrélation moyenne $\bar{r}$, telle que définie à la page 27, est la plus petite observée : 0.320. De plus, uniquement pour *Sonar*, nous obtenons des corrélations négatives. Elles sont illustrées dans l'histogramme de la figure 4.2. Il y a donc des votants qui sont en contradiction avec d'autres. Il s'agit de contradictions lourdes, mais non totales.

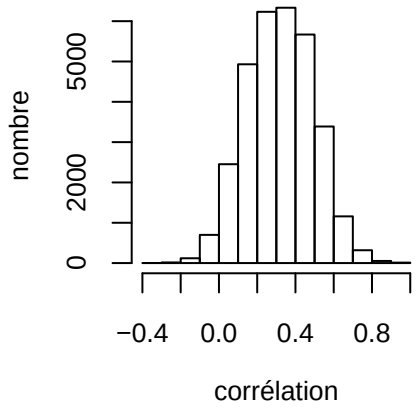

FIG. 4.2 – Corrélations, *Sonar*, *bagging*

Breast Cancer

Pour *Breast Cancer*, aucune différence significative n'est observée. L'ensemble test compte 340 exemples et sur ceux-ci la SCM et la SVM font 12 erreurs de classification, tandis que le vote de majorité en fait 11. Les risques empiriques sur l'ensemble test sont les plus bas observés pour toutes les données. La figure 4.3 montre les intervalles de confiance pour les $R(h)$ des trois classificateurs comparés.

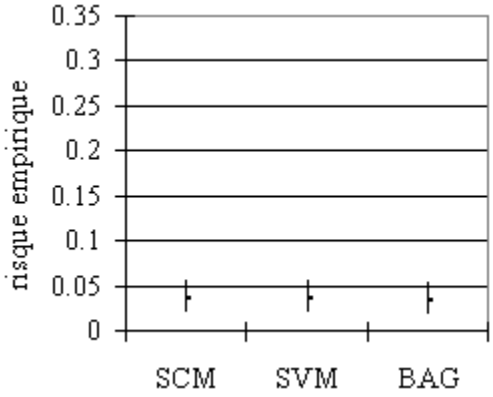

FIG. 4.3 – Intervalles de confiance pour les $R(h)$ de la SCM simple, de la SVM et du vote de majorité construit par *bagging* pour les données *Breast Cancer*

Le risque de Gibbs empirique $R_T(G_Q)$ est de 0.041. L'effet du vote de majorité est positif car le risque de Bayes empirique $R_T(B_Q)$ est de 0.032. Il y a 10 votants qui ont un risque empirique individuel sur T plus petit que $R_T(B_Q)$, 11 votants qui ont un risque empirique sur T égal et 230 votants qui ont un risque empirique sur T plus grand.

Les tableaux 4.7 et 4.8 donnent les nombres de votants erronés par exemple. Pour l'ensemble test, il y a 280 exemples pour lesquels il n'y a aucun votant qui se trompe, il y a 49 exemples pour lesquels il y a de 1 à 122 votants qui se trompent, et ainsi de suite. Ces valeurs dénotent un haut degré de cohérence entre les votants.

Nombre de votants erronés	Nombre d'exemples concernés
0	301
1 à 90	38
91 à 125	0
126 à 223	4
224 à 251	0
total	343

TAB. 4.7 – Erreurs par exemple pour l'ensemble S de *Breast Cancer*

Nombre de votants erronés	Nombre d'exemples concernés
0	280
1 à 122	49
123 à 125	0
126 à 140	0
141 à 248	7
251	4
total	340

TAB. 4.8 – Erreurs par exemple pour l'ensemble T de *Breast Cancer*

Avec les 251 votants, nous retrouvons 105 manières de voter différentes sur l'ensemble test. Formulé autrement, il y a 105 vecteurs de votes $h_j(T)$ différents. Si nous regroupons les votants par manière de voter analogue, nous obtenons des groupes de tailles variant entre 1 et 41. Ces informations sont résumées dans le tableau 4.9. Il y a 72 groupes de taille 1 ; il y a 15 groupes de taille 2 ; il y a 17 groupes de tailles allant de 3 à 12 et il y a un groupe de taille 41.

Nombre de votants de mêmes votes	Nombre de groupes
1	72
2	15
3 à 12	17
41	1
total	105

TAB. 4.9 – Groupes de votants, *Breast Cancer*, *bagging*

La corrélation moyenne $\bar{r}$ entre les paires de vecteurs de votes est 0.943. C'est la plus grande valeur observée pour toutes les données dans le cadre du *bagging*. Les $\binom{251}{2}$ corrélations calculées sont résumées dans l'histogramme de la figure 4.4.

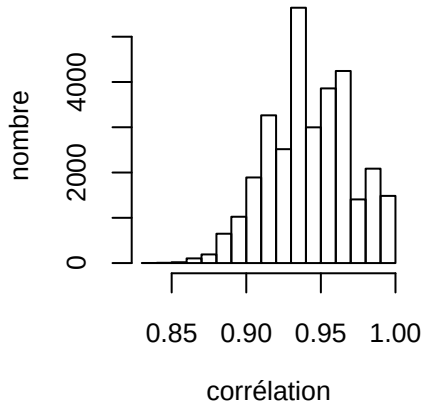

FIG. 4.4 – Corrélations, *Breast Cancer*, *bagging*

Australian Credit

Pour *Australian Credit*, il n'y a pas de différence significative entre les vrais risques. L'ensemble test compte 300 exemples sur lesquels la SCM simple fait 58 erreurs, la SVM 48 erreurs et le vote de majorité 45 erreurs. La différence de nombre d'erreurs entre la SCM simple et le vote est de 13. C'est la plus grande différence absolue observée dans le cadre du *bagging* et la deuxième plus grande différence de risques empiriques sur T. La figure 4.5 montre les intervalles de confiance pour les $R(h)$ des trois classificateurs comparés.

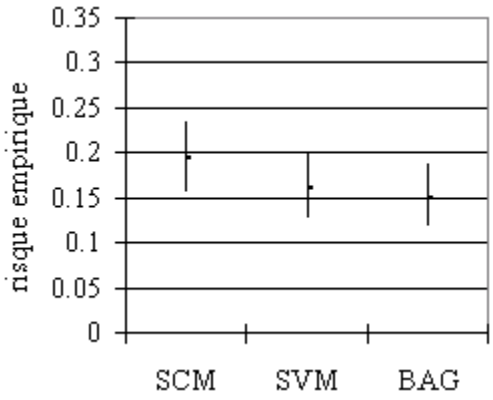

FIG. 4.5 – Intervalles de confiance pour les $R(h)$ de la SCM simple, de la SVM et du vote de majorité construit par *bagging* pour les données *Australian Credit*

Le risque empirique de Gibbs $R_T(G_Q)$ est de 0.216. L'effet du vote de majorité est positif puisque le risque empirique de Bayes $R_T(B_Q)$ est de 0.15. De manière fort surprenante, tous les votants ont un risque empirique individuel sur T plus grand que

$R_T(B_Q)$. Le risque empirique individuel sur T le plus petit est de 0.17. Il y a donc une importante relation de coopération entre les votants.

Les tableaux 4.10 et 4.11 donnent le nombre d'erreurs par exemple. Pour l'ensemble test, il y a 10 exemples pour lesquels aucun votant ne se trompe, 245 exemples pour lesquels il y a de 1 à 124 votants qui se trompent, et ainsi de suite. Les valeurs observées indiquent un faible degré de cohérence entre les votants. Il est cependant encourageant qu'il n'y ait pas d'exemple mal classé par tous les votants, car ceci montre que les votants ne sont pas d'accord non plus pour se tromper souvent aux mêmes endroits.

Nombre de votants erronés	Nombre d'exemples concernés
0	37
1 à 122	298
123 à 125	0
126 à 128	0
129 à 199	18
200 à 251	0
total	353

TAB. 4.10 – Erreurs par exemple pour l'ensemble S de *Australian Credit*

Nombre de votants erronés	Nombre d'exemples concernés
0	10
1 à 124	245
125	0
126 à 250	45
251	0
total	300

TAB. 4.11 – Erreurs par exemple pour l'ensemble T de *Australian Credit*

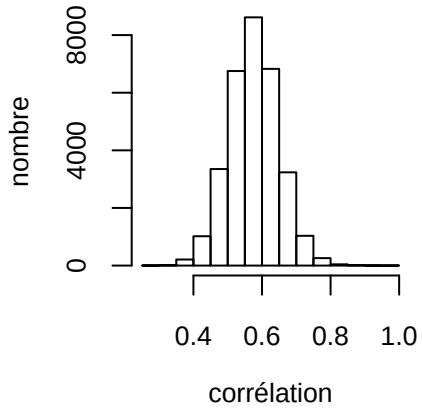

FIG. 4.6 – Corrélations, *Australian Credit*, *bagging*

Une autre particularité des votants, en plus d'avoir un très bon effet de vote sur l'ensemble test, est de voter de manière totalement différente. Tous les vecteurs de votes sont différents. La corrélation moyenne $\bar{r}$ entre ceux-ci est de 0.575. Il s'agit de la deuxième valeur la plus basse. Les corrélations calculées sont résumées dans l'his-

togramme de la figure 4.6. La relative faiblesse de $\bar{r}$ est cohérente avec les erreurs par exemple.

Glass

Il n'y a pas différence notable entre les vrais risques des classificateurs comparés pour *Glass*. L'ensemble test compte 107 exemples et sur ceux-ci la SCM simple fait 22 erreurs, la SVM 23 erreurs et le vote de majorité 20 erreurs. La figure 4.7 montre les intervalles de confiance pour les $R(h)$ des trois classificateurs comparés.

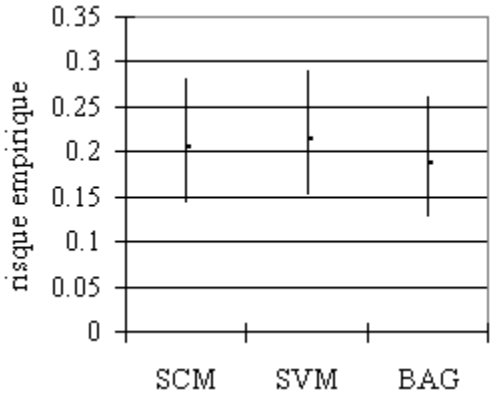

FIG. 4.7 – Intervalles de confiance pour les $R(h)$ de la SCM simple, de la SVM et du vote de majorité construit par *bagging* pour les données *Glass*

Le risque empirique de Gibbs $R_T(G_Q)$ est de 0.2. Comme le risque empirique de Bayes $R_T(B_Q)$ est de 0.187, l'effet du vote est positif. Il y a 66 votants qui ont un risque empirique individuel sur T plus petit que $R_T(B_Q)$, 28 votants qui ont un risque empirique sur T égal à $R_T(B_Q)$ et 157 votants qui ont un risque empirique sur T plus grand.

Les tableaux 4.12 et 4.13 donnent le nombre d'erreurs par exemple. Pour l'ensemble test, il y a 37 exemples pour lesquels aucun votant ne se trompe, il y a 50 exemples pour lesquels il y a de 1 à 123 votants qui se trompent, et ainsi de suite. Le nombre d'exemples majoritairement bien classés (50) plus grand que le nombre d'exemples unanimement bien classés (37) n'est pas une situation idéale. Nous préférerions avoir plus d'exemples pour lesquels tous les votants votent sans erreur.

Nombre de votants erronés	Nombre d'exemples concernés
0	33
1 à 117	70
118 à 125	0
126 à 131	0
132 à 184	4
189 à 251	0
total	107

TAB. 4.12 – Erreurs par exemple pour l'ensemble S de *Glass*

Nombre de votants erronés	Nombre d'exemples concernés
0	37
1 à 123	50
124 et 125	0
126	0
127 à 249	19
251	1
total	107

TAB. 4.13 – Erreurs par exemple pour l'ensemble T de *Glass*

Nous avons 229 vecteurs de votes $h_j(T)$ différents sur une possibilité de 251. Les tailles de groupes de votants ayant des vecteurs de votes identiques sont résumées dans le tableau 4.14. Il y a 210 groupes de taille 1, 16 groupes de taille 2 et 3 groupes de taille 3.

Nombre de votants de mêmes votes	Nombre de groupes
1	210
2	16
3	3
total	229

TAB. 4.14 – Groupes de votants, *Glass*, *bagging*

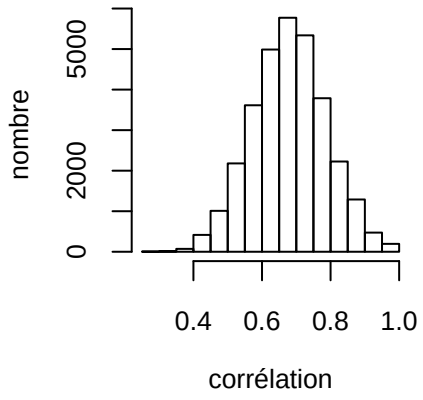

FIG. 4.8 – Corrélations, *Glass*, *bagging*

La corrélation moyenne $\bar{r}$ entre paires de vecteurs de votes est égale à 0.679. Comparée aux autres corrélations moyennes, c'est une valeur qui n'est ni faible ni forte. Les corrélations utilisées dans le calcul sont résumées dans l'histogramme de la figure 4.8.

Heart Disease

Nous n'observons pas de différence significative entre les vrais risques des trois classificateurs comparés. L'ensemble test compte 147 exemples et sur ceux-ci la SCM simple fait 23 erreurs, la SVM 20 erreurs et le vote de majorité 26 erreurs. De toutes les données présentées, c'est la seule occasion où le vote fait plus d'erreurs que la SCM simple. La figure 4.9 montre les intervalles de confiance pour les $R(h)$ des trois classificateurs comparés.

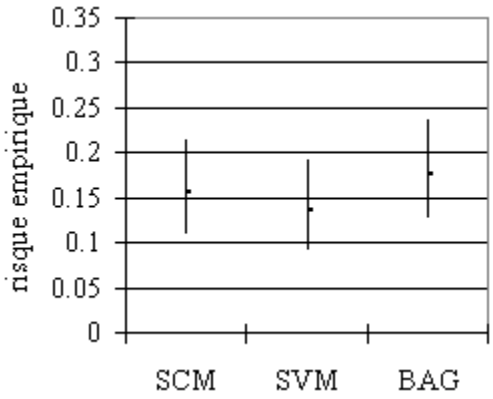

FIG. 4.9 – Intervalles de confiance pour les $R(h)$ de la SCM simple, de la SVM et du vote de majorité construit par *bagging* pour les données *Heart Disease*

Le risque empirique de Gibbs $R_T(G_Q)$ vaut 0.216. L'effet du vote de majorité est positif puisque le risque empirique de Bayes $R_T(B_Q)$ vaut 0.177. Il y 49 votants dont le risque empirique individuel sur T est plus petit que $R_T(B_Q)$, il y a 14 votants dont le risque empirique sur T est égal à $R_T(B_Q)$ et il y a 188 votants dont le risque empirique sur T est plus grand que $R_T(B_Q)$.

Les tableaux 4.15 et 4.16 présentent les nombres d'erreurs par exemple. Pour l'ensemble test, il y a 21 exemples pour lesquels aucun votant ne se trompe, il y a 100 exemples pour lesquels il y a de 1 à 124 votants qui se trompent, et ainsi de suite. Le nombre d'exemples unanimement bien classifiés est plutôt faible. Cependant, le fait qu'il n'y ait qu'un seul exemple unanimement mal classifié est encourageant, car ceci montre que les votants ne sont pas d'accord pour se tromper souvent aux mêmes endroits.

Nombre de votants erronés	Nombre d'exemples concernés
0	24
1 à 111	110
112 à 125	0
126 à 138	0
139 à 207	16
208 à 251	0
total	150

TAB. 4.15 – Erreurs par exemple pour l'ensemble S de *Heart Disease*

Nombre de votants erronés	Nombre d'exemples concernés
0	21
1 à 124	100
125	0
126 à 249	25
251	1
total	147

TAB. 4.16 – Erreurs par exemple pour l'ensemble T de *Heart Disease*

Nous retrouvons cette fois 204 manières de voter différentes sur l'ensemble test. Le tableau 4.17 résume les tailles de groupes de votants ayant des vecteurs de votes $h_j(T)$ identiques. Il y a 189 groupes de taille 1, il y a 9 groupes de taille 2 et il y a 6 groupes dont les tailles varient entre 3 et 10 inclusivement.

Nombre de votants de mêmes votes	Nombre de groupes
1	189
2	9
3 à 10	6
total	204

TAB. 4.17 – Groupes de votants, *Heart Disease*, *bagging*

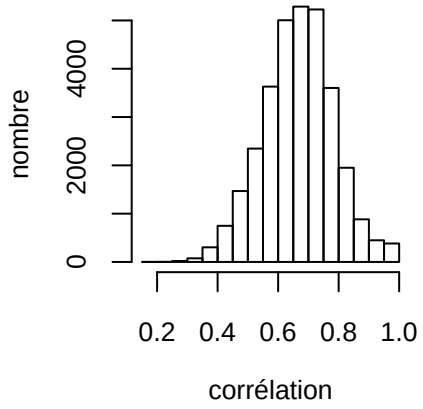

FIG. 4.10 – Corrélations, *Heart Disease*, *bagging*

La corrélation moyenne $\bar{r}$ entre les paires de vecteurs de votes est égale à 0.667. Comparée aux autres corrélations moyennes, c'est une valeur ni faible ni forte. Les corrélations calculées sont résumées dans l'histogramme de la figure 4.10. Leurs valeurs sont présentes dans un intervalle relativement grand.

US Votes

Il n'y a pas de différence significative qui a été observée entre les vrais risques. C'est ici que nous avons la plus grande différence (0.05) entre les risques empiriques sur T de la SCM simple et du vote de majorité. L'ensemble test compte 200 exemples et sur ceux-ci la SCM fait 23 erreurs, la SVM 12 erreurs et le vote 13 erreurs. La différence entre les nombres d'erreurs de la SCM et du vote (10) est la deuxième plus grande dans le cadre du *bagging*. La figure 4.11 montre les intervalles de confiance pour les $R(h)$ des trois classificateurs comparés.

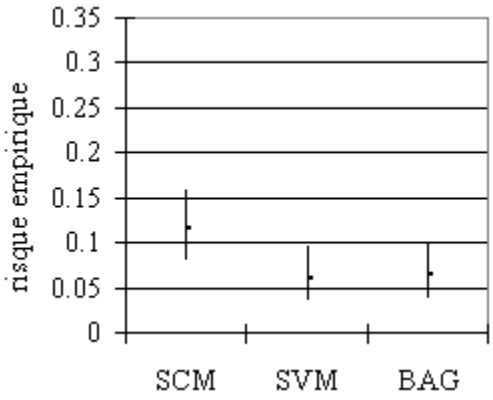

FIG. 4.11 – Intervalles de confiance pour les $R(h)$ de la SCM simple, de la SVM et du vote de majorité construit par *bagging* pour les données *US Votes*

Le risque empirique de Gibbs $R_T(G_Q)$ vaut 0.099. L'effet du vote de majorité est positif puisque le risque empirique de Bayes $R_T(B_Q)$ vaut 0.065. Il y a 13 votants qui ont un risque empirique individuel sur T plus petit que $R_T(B_Q)$, il y a 9 votants qui ont un risque empirique sur T égal à $R_T(B_Q)$ et il y a 229 votants qui ont un risque empirique sur T plus grand que $R_T(B_Q)$.

Parmi les 251 vecteurs de votes $h_j(T)$, il y en a 200 différents. Les tailles des groupes de votants de mêmes votes sont résumées dans le tableau 4.18. Il y a 178 groupes de taille 1, il y a 12 groupes de taille 2 et il y a 10 groupes dont les tailles varient entre 3 et 8.

Nombre de votants de mêmes votes	Nombre de groupes
1	178
2	12
3 à 8	10
total	200

TAB. 4.18 – Groupes de votants, *US Votes*, *bagging*

Les tableaux 4.19 et 4.20 donnent les nombres d'erreurs par exemple. Pour l'ensemble test, il y 86 exemples pour lesquels aucun votant ne se trompe, il y a 101 exemples pour lesquels il y a de 1 à 122 votants qui se trompent et ainsi de suite. Le nombre d'exemples unanimement bien classifiés est raisonnablement élevé. Cependant, le plus haut nombre d'exemples majoritairement bien classifiés (101) dénote tout de même un certain degré d'incohérence entre les votants.

Nombre de votants erronés	Nombre d'exemples concernés
0	140
1 à 114	91
115 à 125	0
126 à 166	0
167 à 220	4
221 à 251	0
total	235

TAB. 4.19 – Erreurs par exemple pour l'ensemble S de *US Votes*

Nombre de votants erronés	Nombre d'exemples concernés
0	86
1 à 122	101
123 à 125	0
126 à 249	13
251	0
total	200

TAB. 4.20 – Erreurs par exemple pour l'ensemble T de *US Votes*

La corrélation moyenne $\bar{r}$ entre les paires de vecteurs de votes est 0.804. Il s'agit de la deuxième valeur la plus élevée, ce n'est cependant pas une valeur que l'on peut qualifier d'extrême. Les corrélations calculées sont résumées dans l'histogramme de la figure 4.12.

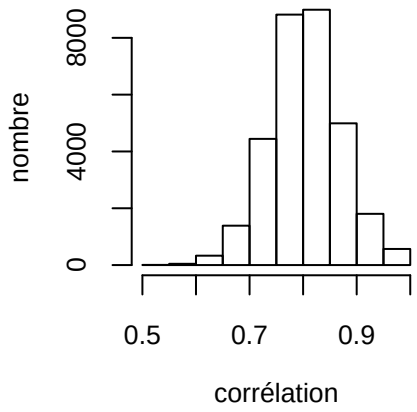

FIG. 4.12 – Corrélations, *US Votes*, *bagging*

Facteurs explicatifs

Dans cette section, nous jetons un regard global sur les résultats obtenus pour toutes les données présentées. Les valeurs du tableau 4.21 résument des informations pour le classificateur par vote de majorité pour chaque ensemble de données. Les mesures présentées sont :

- La borne estimée $\widehat{C_Q}$ avec T sur le risque du vote de majorité. Cette quantité est définie après la proposition 5 à la section 2.9.3.
- La corrélation moyenne entre paires de vecteurs de votes $\bar{r}$ calculée sur T, telle que décrite à la page 27 de la même section.
- La variance des $\widehat{W_Q}$. Les $\widehat{W_Q}$ sont eux aussi définis après la proposition 5. La variance calculée est la variance échantillonnale.
- Le risque empirique de Gibbs $R_T(G_Q)$.
- Le risque empirique de Bayes $R_T(B_Q)$.
- Un indicateur arbitraire de la contribution du vote de majorité $R_T(G_Q) - R_T(B_Q)$. Plus il est grand, meilleure est la contribution.

Nous avons déjà partiellement commenté les valeurs du tableau 4.21 lorsque nous avons examiné les résultats des votes de majorité ensemble de données par ensemble de données. Une description complémentaire pourrait être que la plus grande contribution de vote a été observée pour *Sonar* et que la plus petite a été observée pour *Breast Cancer*. Il est intéressant de noter que le risque empirique de Bayes sur T a toujours été plus petit que le risque empirique de Gibbs sur T.

Données	$\widehat{C_Q}$	$\bar{r}$	Variance des $\widehat{W_Q}$
Sonar	0.531	0.320	0.043
Breast Cancer	0.108	0.943	0.026
Australian Credit	0.442	0.575	0.064
Glass	0.480	0.679	0.083
Heart Disease	0.518	0.667	0.087
US Votes	0.206	0.804	0.042

Données	$R_T(G_Q)$	$R_T(B_Q)$	$R_T(G_Q) - R_T(B_Q)$
Sonar	0.306	0.223	0.083
Breast Cancer	0.041	0.032	0.008
Australian Credit	0.216	0.150	0.066
Glass	0.200	0.187	0.013
Heart Disease	0.216	0.177	0.039
US Votes	0.099	0.065	0.034

TAB. 4.21 – Facteurs explicatifs et risques empiriques ensemble test, *bagging*

La plus grande différence observée entre le $R_T(G_Q)$ d'un vote de majorité et le $R_T(h)$ d'une SCM simple est sur les données *Sonar* et se chiffre à 0.073. Comme le $R_T(G_Q)$ est ce que nous pouvons attendre d'un classificateur SCM typique pour des données générées selon une certaine distribution, nous pouvons déduire que la SCM simple a été plutôt « chanceuse » à cette occasion. La seconde plus grande différence observée concerne les données *Heart Disease* et se chiffre à 0.059.

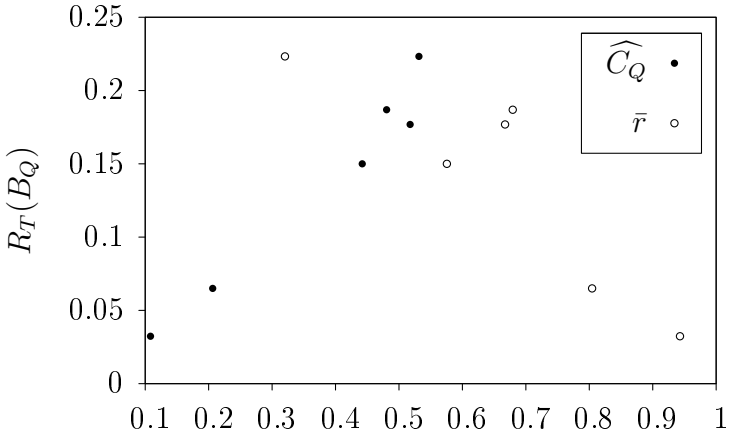

FIG. 4.13 – Facteurs explicatifs pour $R_T(B_Q)$, *bagging*

Le graphique de la figure 4.13 présente le risque empirique de Bayes sur T en fonction de $\widehat{C_Q}$ et de $\bar{r}$. Nous remarquons que $R_T(B_Q)$ est directement proportionnel à C_Q estimée sur l'ensemble test. Nous pouvons également remarquer qu'il semble que plus les vecteurs de votes $h_j(T)$ sont semblables, moins il y a d'erreurs par le vote. Une grande similarité des vecteurs de votes implique une grande moyenne des corrélations.

La seule variable qui semble avoir une relation avec la contribution du vote pour le *bagging* est la corrélation moyenne. Le graphique de la figure 4.14 suggère que la contribution est moindre lorsque $\bar{r}$ est grande. Si une haute corrélation moyenne cause réellement un faible risque pour le vote, il ne peut être réaliste d'espérer avoir simultanément un excellent effet de vote, puisque si tous les vecteurs de votes sont très semblables, l'écart entre le risque empirique de Gibbs sur T et celui de Bayes sur T ne peut être très grand.

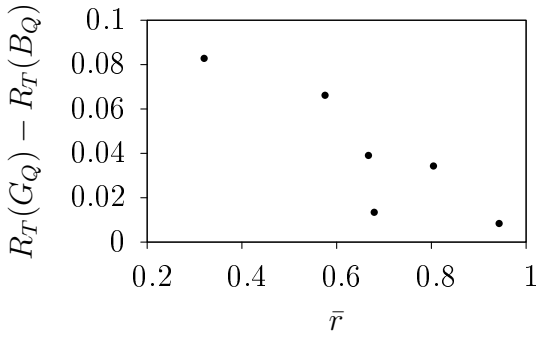

FIG. 4.14 – Contribution du vote, *bagging*

4.3.2 Résumé

Pour toutes les données présentées, il n'y a pas de différence significative entre les vrais risques des SCM simples et des votes de majorité. Partout, sauf pour *Heart Disease*, les risques empiriques sur T sont légèrement plus petits pour les votes que pour les SCM simples. À la lumière des résultats présentés jusqu'ici, le *bagging* de SCM est recommandé si nous pouvons nous contenter d'un gain minime en performance. Les contributions toutes positives des votes de majorité confirment le léger avantage du vote comparé à la SCM simple puisque le risque empirique de Gibbs sur T correspond plus au rendement attendu d'une SCM typique que le risque empirique sur T de la SCM simple selon le modèle choisi.

Lors d'expérimentations sur des données qui n'ont pas été présentées, nous avons observé dans la plupart des cas que les vrais risques des SCM simples et des votes de

majorité n'étaient pas significativement différents. Cependant, il y a eu une occasion où le vrai risque de la SCM simple était significativement inférieur à celui du vote et une où il était significativement plus grand. Les intervalles de confiance ne se chevauchaient pas. Ces cas sont explicables par des solutions de l'algorithme SCM-classique qui ont été favorisées ou victimes de leur sensibilité aux données. Ce sont des exceptions, tandis que ce que nous obtenons avec le *bagging* reflète la tendance que nous observerions en obtenant plus d'exemples de la distribution qui les génère.

4.4 Variantes explorées

L'algorithme 4.1 décrit une version assez simple du *bagging* de SCM. Il y a quelques variantes qui ont été essayées, soit des algorithmes différents, soit des ajustements particuliers. Les résultats ont été soit pires ou équivalents à ceux de l'algorithme présenté. Dans cette section, nous avons comme objectif d'exposer sommairement les avenues qu'il est inutile de revisiter.

Lois normales

Dans l'espoir d'obtenir des échantillons *bootstrap* plus près de la distribution ayant généré S et T, nous avons tenté de perturber les points pigés dans S selon une normale multivariée. Pour chaque exemple, une normale multivariée de vecteur de moyenne $\boldsymbol{\mu} = \mathbf{x}_i$ et de matrice de covariance Σ de dimension $n \times n$ a été utilisée. n est la longueur du vecteur $\mathbf{x}_i$. La valeur de σ pouvait être choisie par validation croisée. La valeur choisie était la même pour tous les tirages de tous les échantillons.

$$\Sigma = \begin{bmatrix} \sigma & 0 & \cdots & 0 \\ 0 & \sigma & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma \end{bmatrix}$$

Les étapes de la construction d'un échantillon *bootstrap* étaient :

1. Piger uniformément, avec remise, $\mathbf{z}_i$ dans S. Nous avons $\mathbf{z}_i = (\mathbf{x}_i, y_i)$ et $i \in \{1, \ldots, m\}$.
2. Centrer la normale sur $\mathbf{x}_i$ par l'opération $\boldsymbol{\mu} = \mathbf{x}_i$.
3. Générer un point $\mathbf{x}'$ selon une normale de paramètres $\boldsymbol{\mu}$ et Σ.

4. $\mathbf{z}' \leftarrow (\mathbf{x}', y_i)$
5. Ajouter $\mathbf{z}'$ dans l'échantillon.
6. Répéter tant qu'il manque des exemples dans l'échantillon.

Sur-lissage ou sous-lissage

En manipulant les paramètres et sans changer l'algorithme 4.1, nous avons tenté de déterminer si la contribution de l'ajustement des votants influençait leur coopération dans le vote. Nous avons produit des votants très ajustés en ne leur permettant pas de faire des erreurs sur leurs ensembles d'entraînement. Nous avons aussi produit des votants peu ajustés en limitant fortement leur nombre de caractéristiques. Nous avons aussi essayé de choisir le nombre maximal de caractéristiques pour tous les votants par validation croisée. Tout cela sans amélioration notable.

Échantillonnage sans remise

Nous avons essayé l'échantillonnage sans remise avec différents taux d'échantillonnage (0.3, 0.5 et 0.7) pour la construction des échantillons *bootstrap*.

4.5 Conclusion

Le fait principal pouvant être dégagé des travaux effectués au cours de ce chapitre est que le *bagging* de SCM n'est pas significativement plus performant qu'une SCM individuelle. Bien que ce ne soit jamais significatif à plus de 90 %, une tendance se dégage : le *bagging* de SCM est presque toujours un peu mieux que la SCM simple. De plus, $R_T(G_Q)$ est toujours pire que $R_T(B_Q)$. En règle générale, la méthode utilisée pour construire une SCM doit produire un classificateur d'une stabilité suffisante. Une procédure qui augmente la stabilité n'aura donc pas un effet remarquable. Nous devons nuancer le terme « suffisante » puisque nous avons observé qu'il était plutôt rare que les vecteurs de votes soient identiques. L'instabilité est donc indéniablement présente.

Chapitre 5

Conclusion

L'itinéraire que nous nous étions fixé pour ce mémoire a maintenant été parcouru. Les travaux effectués ont été décrits le plus complètement et le plus clairement possible. Il ne nous reste qu'à revoir leurs motivations fondamentales, à les considérer globalement et à jeter un regard sur l'avenir. Pendant le regard sur les motivations, nous reverrons les objectifs, nous considérerons ensuite les résultats en les regroupant et enfin nous aborderons les travaux futurs.

5.1 Objectifs

Nous rappelons que l'objectif principal de la maîtrise était de concevoir un algorithme d'apprentissage pour classificateurs de type SCM qui trouve le classificateur minimisant la borne sur le risque de Marchand et Sokolova. Techniquement, cet objectif a été atteint. L'algorithme est cependant accompagné d'une sévère contrainte quant aux entrées possibles. Pour des ensembles d'entraînement de taille restreinte, environ 100 exemples ou moins, le *branch-and-bound* est une alternative valable (pour un utilisateur patient). Il ne faut pas oublier que les votes regroupant les classificateurs optimaux selon la borne ont tous été aussi performants que les SCM simples. Bien que la conception d'algorithmes vise toujours à faire mieux que ce qui existe déjà, c'est rarement possible, et il est certain que c'est possible de moins en moins souvent.

En dehors des considérations d'améliorations algorithmiques, l'absence d'écart entre les votes et les SCM simples pour le *branch-and-bound* nous a révélé un fait important sur l'algorithme glouton utilisé en combinaison avec une sélection de modèle par la borne

de Marchand et Sokolova. Malgré son apparente simplicité, cet algorithme produit des classificateurs qui sont, parfois par transposition, optimaux au sens de la borne. Cette information donne une valeur ajoutée à une utilisation de classificateurs de type SCM.

L'objectif secondaire de cette maîtrise était d'appliquer un vote de majorité aux SCM. Cet objectif a lui aussi été atteint. Le *bagging* de SCM est un algorithme dont nous ne décourageons pas l'utilisation. Il faut cependant garder à l'esprit qu'une SCM simple fait presque un aussi bon travail qu'un vote. Bien que la construction de votants par *bagging* soit moins longue que par *branch-and-bound*, elle l'est tout de même bien plus que la construction d'une seule SCM.

Une motivation pour l'utilisation du *bagging* pourrait être la volonté d'éliminer des SCM l'effet de leur sensibilité aux données. Le prix à payer en temps de calcul n'est pas exorbitant, mais il est peut-être un peu trop élevé considérant que les SCM sont rarement victimes de leur instabilité ou qu'elles sont raisonnablement stables.

5.2 Retour sur les résultats

L'observation clé pour tous les résultats présentés est qu'il n'y a pas de différence significative entre les vrais risques des votes de majorité et des SCM simples. Cependant, bien que non significativement dans chaque cas, le *bagging* de SCM produit un vote de majorité presque toujours meilleur qu'une seule SCM. La conséquence logique de ces observations est de recommander d'utiliser une SCM simple lorsqu'un classificateur produit très rapidement est voulu et de recommander d'utiliser le *bagging* de SCM si un classificateur stable et possiblement légèrement plus performant qu'une SCM simple est voulu. Mais nous rappelons, encore une fois, qu'un *branch-and-bound* ou un *bagging* ne produit pas des classificateurs moins performants que l'algorithme glouton.

Comme il n'a pas semblé approprié de commenter à nouveau les résultats obtenus lors du *branch-and-bound* dans le chapitre traitant du *bagging*, voici une mise en commun de certains résultats observés dans les deux chapitres. Nous ne présentons ici aucun nouvel élément, il ne s'agit que d'un retour sur ce qui a déjà été vu. Nous nous permettons cependant une perspective plus large.

Pendant l'analyse des votes produits lors du *branch-and-bound*, nous n'avions rien pu conclure en observant l'indicateur de la contribution du vote $R_T(G_Q) - R_T(B_Q)$ en fonction de la corrélation moyenne $\bar{r}$. Une tendance était par contre visible pour le *bagging*. Il est intéressant de constater que ce que nous avons observé pour le *branch-*

and-bound n'est pas en contradiction avec ce que nous avons observé en dernier pour le *bagging*. La figure 5.1 présente ce qui inspire ce propos. Les points ajoutés pour le *branch-and-bound* ne semblent pas dévier de la tendance observée pour le *bagging*. L'écart entre le risque empirique de Gibbs et le risque empirique de Bayes semble moindre lorsque les vecteurs de votes sont plus corrélés. Sans oublier qu'une faible corrélation moyenne pourrait causer non seulement un grand effet de vote, mais un effet positif.

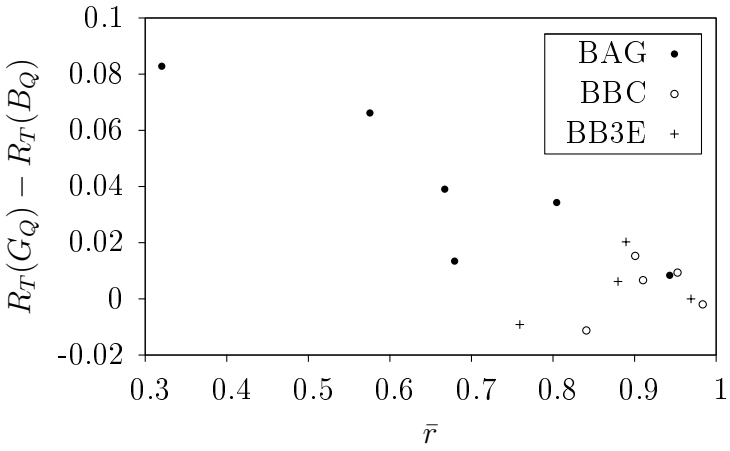

FIG. 5.1 – Contribution du vote, tous les algorithmes

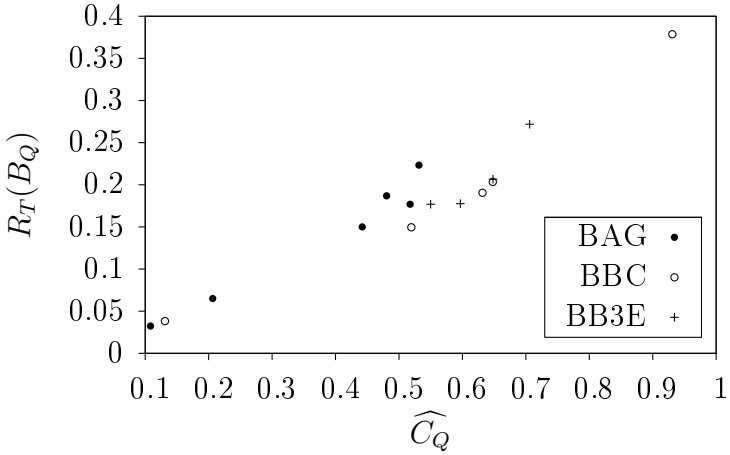

FIG. 5.2 – Borne pour le vote, tous les algorithmes

Nous pouvons observer à la figure 5.2 que la relation est linéaire entre les risques empiriques des votes et les valeurs de la borne sur le risque du vote de majorité estimée. La concordance entre les valeurs pour tous les algorithmes corrobore l'encouragement à l'utilisation de C_Q prônée dans l'article qui la présente [24] car C_Q estimée sur les ensembles d'entraînement est forcément elle aussi linéairement reliée au risque de Bayes.

Lorsque nous relions la corrélation moyenne au risque empirique de Bayes, une tendance est facilement observable algorithme par algorithme. Le regroupement de ces observations nous apprend que la force du lien entre les deux quantités semble varier dépendamment de l'algorithme qui produit les votants. En effet, nous observons, par interpolation, plus d'une droite sur la figure 5.3, mais celles-ci ne sont pas superposées. Bien que de même signe, leurs pentes sont différentes.

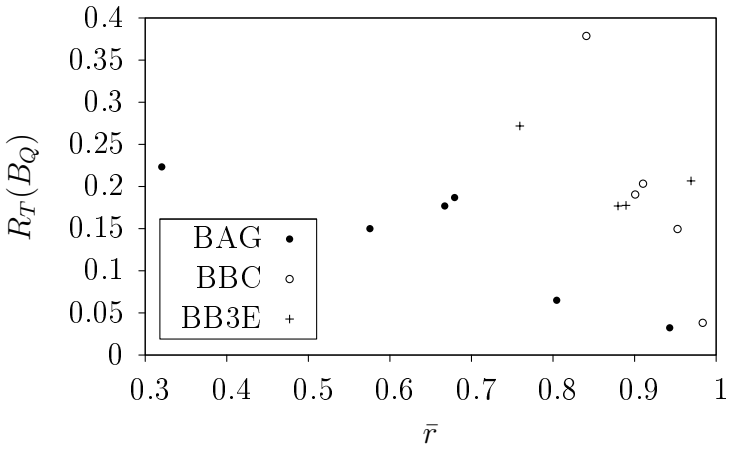

FIG. 5.3 – Corrélation moyenne, tous les algorithmes

Notons que les algorithmes similaires, les deux *branch-and-bounds*, ont produit des votants qui ont engendré des paires corrélation−risque empirique qui semblent plutôt cohérentes entre elles. Elles sont dans une même région du graphique.

5.3 Travaux futurs

Lors des travaux effectués, les objectifs principaux ont été traités sans trop de détours et dans une perspective relativement étroite. Comme le temps alloué était limité et qu'il est difficile de traiter exhaustivement des idées sans prendre un certain recul pour revenir à la charge, voici une section présentant des travaux potentiels. Ils se basent sur ce qui a été accompli ou ils y sont fortement reliés.

Une avenue qui devrait être explorée est un vote de majorité qui s'apparente au *bagging*. Il n'y a pas d'échantillon à proprement parler, mais il y a tirage d'exemples. Le classificateur est un vote de SCM. Lors de l'ajout d'une nouvelle caractéristique à une SCM votante, un exemple négatif est pigé parmi l'ensemble des négatifs non couverts par les caractéristiques déjà présentes dans la SCM en construction. La nouvelle hypersphère

doit couvrir l'exemple pigé de même que le plus de négatifs non couverts possible. Les classificateurs votants pourraient être un mélange de conjonctions et de disjonctions, car cette approche s'est avérée avantageuse pour le *bagging*.

Un ajustement des poids des votants pourrait être envisagé en minimisant C_Q estimée sur S à l'aide d'une descente en gradient. Les poids sont donnés par Q tel que défini à la section 2.9.3. La relation linéaire observée entre C_Q estimée sur T et $R_T(B_Q)$ indique que minimiser C_Q estimée sur S devrait causer un faible risque de Bayes.

La première étape de la procédure est d'entraîner des votants sur S avec l'algorithme 4.1. Ensuite, nous enchaînons avec la descente en gradient. Celle-ci peut être effectuée en sélectionnant aléatoirement sans remise les paires de votants et en ajustant leurs poids de manière à faire diminuer $\widehat{C_Q}$ le plus possible. Les tirages sont répétés jusqu'à ce nous n'observions plus d'amélioration ou jusqu'à ce qu'un nombre maximal d'itérations ait été atteint.

Maximiser la probabilité de désaccord d_Q estimée sur S a aussi été envisagé. Cependant, des résultats préliminaires peu encourageants nous ont fait renoncer à cette alternative. Il peut être démontré que maximiser d_Q est équivalent à minimiser C_Q, pourvu que $R(G_Q)$ ne varie pas en fonction des W_Q. La définition de la probabilité de désaccord d_Q est :

$$d_Q \stackrel{\text{def}}{=} \mathop{\mathbf{E}}_{\mathbf{x}\sim P_{\mathbf{X}}} \left[\mathop{\mathbf{E}}_{h_1,h_2\sim H} I(h_1(\mathbf{x}) \neq h_2(\mathbf{x})) \right]$$

Sans pour autant mener à un nouveau type de classificateur révolutionnaire, les travaux effectués pourraient être poursuivis dans l'optique de clarifier les relations observées dans la description des votes de majorité. Des tests supplémentaires seraient utiles pour mieux cerner la relation entre le risque de Bayes et la corrélation moyenne des paires de vecteurs de votes. Par la même occasion, la relation entre la corrélation moyenne et la contribution du vote pourrait être examinée plus en profondeur. Pour commencer, il faudrait analyser globalement les résultats du *bagging* en incluant des données autres que celles présentées dans ce mémoire.

Dans le cadre du *branch-and-bound*, nous avons travaillé à minimiser une quantité qui varie de la même manière que la borne de Marchand et Sokolova. Pour fins de complétude des travaux, car il serait étonnant que les conclusions soient différentes de celles déjà obtenues, il pourrait être approprié de tenter de minimiser directement la borne avec une approche inspirée du *branch-and-bound* en trois étapes.

Finalement, de nouvelles métriques pourraient être élaborées pour les SCM simples en utilisant des techniques de projection des données, comme les analyses en composantes principales avec ou sans noyau.

5.4 Mot de la fin

Voici qui termine notre première visite du *branch-and-bound* et du *bagging* dans les transformations possibles des SCM. Nous nous sommes rendus où nous voulions aller et en cours de route nous avons pu obtenir des informations capitales sur ces classificateurs encore riches en possibilités. Nous espérons que vous avez apprécié la lecture de ce mémoire et que vous y avez trouvé des éléments qui vous seront utiles.

Bibliographie

[1] Mario Marchand. IFT-65764 Apprentissage Automatique (« Machine Learning »). Notes de cours IFT-65764, Université Laval.

[2] Mario Marchand et John Shawe-Taylor. Learning with the Set Covering Machine. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001)*, pages 345–352, 2001.

[3] Mario Marchand et John Shawe-Taylor. The set covering machine. *Journal of Machine Learning Reasearch*, 3 :723–746, 2002.

[4] Mario Marchand et Marina Sokolova. Learning with Decision Lists of Data-Dependent Features. *Journal of Machine Learning Reasearch*, 6 :427–451, 2005.

[5] Corinna Cortes et Vladimir Vapnik. Support-Vector Networks. *Machine Learning*, 20(3) :273–297, 1995.

[6] Michael R. Garey et David S. Johnson. *Computers and Intractability : A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, 1979.

[7] Gilles Brassard et Paul Bratley. *Fundamentals of Algorithmics*. Prentice Hall, Upper Saddle River NJ, 1996.

[8] Leo Breiman. Bagging Predictors. *Machine Learning*, 24(2) :123–140, 1996.

[9] Henry Aubert. Vocabulaire de la statistique descriptive. En ligne, http://hpa.free.fr/VocabStat.htm (page consultée le 20 septembre 2007).

[10] Antoine Cornuéjols et Laurent Miclet. *Apprentisssage artificiel : Concepts et algorithmes*. Éditions Eyrolles, 2002.

[11] D.J. Newman et S. Hettich et C.L. Blake et C.J. Merz. UCI Repository of Machine Learning Databases. University of California, Irvine, Dept. of Information and Computer Sciences. En ligne, http://www.ics.uci.edu/~mlearn/MLRepository.html (page consultée le 17 novembre 2006).

[12] Terry Sejnowski et R. Paul Gorman. Ensemble de données : Sonar, Mines vs. Rocks. Contribué par Sejnowski et développé en collaboration avec Gorman.

[13] Ludovic Lebart et Alain Morineau et Marie Piron. *Statistique exploratoire multidimensionnelle*. Dunod, Paris, 1997.

[14] William H. Wolberg et Nick Street et Olvi L. Mangasarian. Ensemble de données : Wisconsin Breast Cancer Database. Obtenu de la University of Wisconsin Hospitals, Madison, par Nick Street.

[15] Jeff Schlimmer. Ensemble de données : 1984 United States Congressional Voting Records Database. Provient du Congressional Quarterly Almanac, 98th Congress, 2nd session 1984, vol. XL : Congressional Quarterly Inc. Washington, D.C.

[16] Andras Janosi et William Steinbrunn et Matthias Pfisterer et Robert Detrano. Ensemble de données : Heart Disease.

[17] B. German et Vina Spiehler. Ensemble de données : Glass Identification Database.

[18] J. R. Quinlan. Ensemble de données : Australian Credit.

[19] John Langford. Tutorial on Practical Prediction Theory for Classification. *Journal of Machine Learning Research*, 6 :273–306, 2005.

[20] Bernhard E. Boser et Isabelle Guyon et Vladimir Vapnik. A Training Algorithm for Optimal Margin Classifiers. In *Computational Learing Theory*, pages 144–152, 1992.

[21] Chang Chih-Chung et Lin Chih-Jen. LIBSVM : a library for Support Vector Machines, 2001. En ligne, http://www.csie.ntu.edu.tw/~cjlin/libsvm (page consultée le 2 mars 2006).

[22] Alexandros Karatzoglou et Alex Smola et Kurt Hornik et Achim Zeileis. kernlab : Kernel Methods Lab. En ligne, http://cran.stat.sfu.ca/src/contrib/Descriptions/kernlab.html (page consultée le 2 mars 2006).

[23] John C. Platt. *Fast Training of Support Vector Machines using Sequential Minimal Optimization*, pages 185–208. MIT Press, 1999. Extrait de Advances in Kernel Methods : Support Vector Learning.

[24] Pascal Germain et Alexandre Lacasse et François Laviolette et Mario Marchand et Nicolas Usunier. PAC-Bayes Bounds for the Risk of the Majority Vote and the Variance of the Gibbs Classifier. In *Proceedings of the 2006 conference on Neural Information Processing Systems (NIPS-06)*, 2006. Accepté.

[25] William Cook et Daniel Espinoza et Marcos Goycoolea. Computing with domino-parity inequalities for the TSP. In *INFORMS Journal on Computing*, 2006. Accepté.

[26] Yoav Freund et Robert E. Schapire. Experiments with a New Boosting Algorithm. In *International Conference on Machine Learning*, pages 148–156, 1996.

[27] J. Ross Quinlan. Bagging, Boosting, and C4.5. In *AAAI/IAAI, Vol. 1*, pages 725–730, 1996.