france telecom R&D

DTL
Direction des Techniques Logicielles

Note Technique
NT/FTR&D/8444

Mars 2004

MODL: une méthode quasi-optimale de discrétisation supervisée

Marc Boullé (DTL/TIC)

Vu, pour accord le directeur de DTL

V. Robin

Vu, le responsable du laboratoire TIC

O. Perrault

Date : 2 mars 2004

Résumé : Dans le domaine de l'apprentissage supervisé, les méthodes de discrétisation des attributs continus partitionnent un domaine numérique en un nombre fini d'intervalles, en recherchant un compromis entre valeur informationnelle et valeur prédictive de la partition formée. Dans ce papier, nous introduisons une formalisation du problème de la discrétisation supervisée, et dans ce cadre nous proposons un critère d'évaluation d'une discrétisation, dont l'optimisation garantit l'optimalité au sens de Bayes de la discrétisation obtenue. Nous présentons également un algorithme permettant de trouver la discrétisation optimale en $O(n^3)$, où n est le nombre d'instances à discrétiser. Enfin, nous présentons une heuristique d'optimisation en O(n.log(n)) permettant d'obtenir des discrétisations quasi-optimales. L'heuristique d'optimisation se base sur une approche classique ascendante, complétée par une post-optimisation poussée des discrétisations. Des expérimentations intensives ont été menées sur de nombreux jeux de données de nature différente, afin de comparer la méthode MODL avec d'autres méthodes de discrétisation de référence en prenant en compte plusieurs critères tels la performance prédictive, la robustesse ou la taille des discrétisations. Une analyse multi-critères des résultats démontre les bonnes performances de la méthode MODL, tout en apportant un éclairage intéressant sur la problématique générale de la discrétisation.

Mots clés : **analyse intelligente donnée ; apprentissage automatique ; discrétisation**
Domaine : **Traitement de l'information et des connaissances**

France Télécom
Fonction Groupe Technologie et Innovation
France Télécom R&D
2, avenue Pierre-Marzin - 22307 Lannion Cedex
Téléphone : 02 96 05 11 11
Téléphone international : + 33 2 96 05 11 11
SA au capital de 9 609 262 400 € - 380 129 866 RCS Paris

MODL: une méthode quasi-optimale de discrétisation supervisée

MARC BOULLE
France Telecom R&D
2, Avenue Pierre Marzin
22300 Lannion – France
marc.boulle@francetelecom.com

Résumé. Dans le domaine de l'apprentissage supervisé, les méthodes de discrétisation des attributs continus partitionnent un domaine numérique en un nombre fini d'intervalles, en recherchant un compromis entre valeur informationnelle et valeur prédictive de la partition formée. Dans ce papier, nous introduisons une formalisation du problème de la discrétisation supervisée, et dans ce cadre nous proposons un critère d'évaluation d'une discrétisation, dont l'optimisation garantit l'optimalité au sens de Bayes de la discrétisation obtenue. Nous présentons également un algorithme permettant de trouver la discrétisation optimale en $O(n^3)$, où n est le nombre d'instances à discrétiser. Enfin, nous présentons une heuristique d'optimisation en $O(n.\log(n))$ permettant d'obtenir des discrétisations quasi-optimales. L'heuristique d'optimisation se base sur une approche classique ascendante, complétée par une post-optimisation poussée des discrétisations. Des expérimentations intensives ont été menées sur de nombreux jeux de données de nature différente, afin de comparer la méthode MODL* avec d'autres méthodes de discrétisation de référence en prenant en compte plusieurs critères tels la performance prédictive, la robustesse ou la taille des discrétisations. Une analyse multi-critères des résultats démontre les bonnes performances de la méthode MODL, tout en apportant un éclairage intéressant sur la problématique générale de la discrétisation.

Mots clés : Data Mining, Machine Learning, Discretization, Data Analysis

* Dépôt de brevet N° 04 00179

TABLE DES MATIERES

1 Introduction

La discrétisation des attributs numériques est un sujet largement traité dans la bibliographie (Catlett, 1991; Holte, 1993; Dougherty, Kohavi & Sahami, 1995; Zighed & Rakotomalala 2000). Une partie des modèles d'apprentissage est basée sur le traitement des attributs à valeurs discrètes. Il est donc nécessaire de discrétiser les attributs numériques, c'est à dire de découper leur domaine en un nombre fini d'intervalles identifiés chacun par un code. Ainsi par exemple, tous les modèles prédictifs à base d'arbre de décision utilisent une méthode de discrétisation pour traiter les attributs numériques. C4.5 (Quinlan 1993) utilise le gain informationnel basé sur l'entropie de Shannon, CART (Breiman 1984) utilise l'indice de Gini (une mesure de l'impureté des intervalles), CHAID (Kass 1980) s'appuie sur une méthode de type ChiMerge, SIPINA utilise le critère Fusinter (Zighed et al., 1998) basé sur des mesures d'incertitude sensibles aux effectifs.

Parmi les méthodes de discrétisation, il existe des méthodes descendantes et ascendantes. Les méthodes descendantes partent du domaine numérique complet à discrétiser et le coupent en deux récursivement. Les méthodes ascendantes partent des intervalles élémentaires mono-valeur et les fusionnent itérativement. Certaines de ces méthodes nécessitent un paramétrage utilisateur pour modifier le comportement du critère de choix du point de discrétisation ou pour fixer un seuil pour le critère d'arrêt. Le problème de la discrétisation est un problème de compromis entre qualité informationnelle (intervalles homogènes vis à vis de la variable à prédire) et qualité statistique (effectif suffisant dans chaque intervalle pour assurer une généralisation efficace). Les critères de type Khi2 privilégient l'aspect statistique tandis que ceux basés sur la mesure de l'entropie privilégient l'aspect informationnel. D'autres critères (indice d'impureté de Gini, mesure d'incertitude de Fusinter…) essayent de concilier les deux aspects en étant à la fois sensible aux effectifs et à la distribution de la variable à prédire.

Nous présentons dans ce papier une nouvelle méthode de discrétisation appelée MODL. La principale innovation réside dans le choix du critère d'évaluation de la discrétisation résultant d'une approche bayésienne de la discrétisation. Le critère d'évaluation utilisé par la méthode MODL a été élaboré d'une part en fonction de ses qualités d'évaluation fine des discrétisations, y compris sur des ensembles de petite taille, d'autre part en raison de sa décomposabilité sur l'ensemble des intervalles, qui permet d'utiliser des algorithmes d'optimisation performants. Après une formulation du problème de discrétisation, nous démontrons que le critère proposé permet de trouver la discrétisation optimale au sens de Bayes, c'est à dire la discrétisation la plus probable expliquant les données. Contrairement aux résultats de type MDL (Minimum Description Length), la preuve d'optimalité n'est pas asymptotique, et est donc valide y compris pour des échantillons de très petite taille. Nous présentons alors un algorithme optimal en $O(n^3)$ pour le critère utilisé par la méthode MODL. Nous étudions et comparons ensuite les heuristiques gloutonnes classiques ascendantes et descendantes, et proposons une amélioration des ces algorithmes sous la forme d'une post-optimisation des discrétisations, de complexité $O(n.\log(n))$. Nous montrons de façon empirique que la meilleure heuristique d'optimisation proposée aboutit aux discrétisations optimales dans une large majorité des cas. Nous proposons enfin une prise en compte du caractère explicatif des discrétisations, en intégrant dans l'algorithme MODL des contraintes d'effectif minimum par intervalle ou de nombre maximum d'intervalles.

Afin d'évaluer la méthode MODL et de la comparer à plusieurs autres méthodes de discrétisation, nous avons procédé à des expérimentations intensives sur des bases de test provenant de l'UCI Irvine (Blake 1998), ainsi que sur une base France Telecom issue d'une étude opérationnelle. Nous avons pris en compte plusieurs critères d'évaluation, principalement la performance prédictive, la robustesse (dégradation de la performance entre apprentissage et test), la taille des discrétisations et le comportement vis à vis des attributs bruités. Une analyse multi-critères des résultats permet de comparer en détail les différentes méthodes. Cette évaluation en profondeur confirme les résultats théoriques et montre que la méthode MODL est la plus performante sur l'ensemble des critères considérés.

Le reste du document est organisé de la façon suivante. La partie 2 introduit le nouvel algorithme MODL en détaillant son critère d'évaluation puis en étudiant différents algorithmes d'optimisation. La partie 3 procède à des expérimentations comparatives permettant une évaluation multi-critères des méthodes de discrétisation. L'annexe 4 rappelle quelques éléments de théorie de l'information et analyse deux algorithmes de discrétisation existant basés sur l'approche MDL. L'annexe 5 propose plusieurs critères d'évaluation alternatifs découlant comme pour la méthode MODL d'une approche bayésienne. L'annexe 6 présente une application de la méthodologie MODL aux discrétisations non supervisées en largeur égale ou fréquence égale, permettant de trouver le nombre d'intervalles optimaux. L'annexe 7 contient les preuves des théorèmes.

2 Discrétisation MODL

L'approche MDL se focalise sur le problème du codage d'un modèle et des exceptions à ce modèle. Cette approche est fondée théoriquement par son lien fort avec l'approche bayésienne dont elle peut s'approcher asymptotiquement si le choix de codage est correct. Cette approche est détaillée en annexe 4, ainsi que son utilisation dans les méthodes de discrétisation MDLPC (Fayyad & Irani, 1992) et MDL-DISC (Pfahringer, 1995). Nous proposons ici une approche bayésienne de la discrétisation appelée MODL, qui se focalise d'abord sur une définition précise de l'espace des modèles et de la distribution a priori des modèles dans cet espace. L'utilisation d'une définition paramétrique de l'espace des modèle permet alors de calculer exactement les probabilités des modèles et des données connaissant les modèles. Ce calcul débouche sur un critère d'évaluation d'une discrétisation, dont le minimum correspond à la discrétisation optimale au sens de Bayes.

Dans cette partie, on présente dans un premier temps une formalisation du problème de la discrétisation, et un critère d'évaluation permettant d'aboutir aux discrétisations optimales au sens de Bayes. Dans un second temps, on propose des

algorithmes d'optimisation permettant de rechercher les discrétisations optimales, de façon exacte ou approchée.

2.1 Critère d'évaluation

On présente ici une formulation du problème de la discrétisation puis une liste de résultats théoriques, dont la démonstration est donnée en annexe. Dans le cadre des hypothèses présentées, on propose notamment un critère d'évaluation des discrétisations permettant de rechercher la solution optimale au sens de Bayes.

2.1.1 Préliminaires

On se place dans le cadre où les données ayant été triées selon les valeurs de l'attribut descriptif à discrétiser, on recherche une discrétisation en se basant uniquement sur l'ordre des instances. Les données en entrée constituent alors une chaîne *S* de longueur *n*, comportant une séquence de valeurs de l'attribut à prédire.

On suppose que la longueur *n* de la chaîne de symboles, et le nombre de valeurs *J* de l'attribut cible sont connus à l'avance.

Définition: Un modèle de discrétisation est dit *standard* s'il respecte les conditions suivantes:
- il ne repose que sur l'ordre des symboles de la chaîne *S*, sans tenir compte des valeurs de l'attribut descriptif,
- il permet de définir une partition de la chaîne *S* en sous-chaînes (les intervalles),
- la distribution des symboles sur chaque intervalle est définie uniquement par la fréquence des symboles sur cet intervalle.

On dira qu'un tel modèle de discrétisation est de type SDM (Standard Discretization Model).

Prenons par exemple un attribut à discrétiser, dont les valeurs ont été triées par ordre croissant. On s'intéresse alors à la chaîne de symboles '0' ou '1' correspondant à la séquence des classes à prédire. La figure ci-dessous définit une discrétisation SDM en cinq intervalles, si l'on admet que les distributions de classes cibles sur chaque intervalles sont indépendantes les unes des autres. Les distributions par intervalles sont définies par la fréquence des symboles sur chaque intervalle, ce qui correspond ici à une fréquence (du symbole '1') de 2/6 pour l'intervalle 1, 11/14 pour l'intervalle 2, 0/6 pour l'intervalle 3…

010100	11101101111011	000000	110100111011010	1111111

Ces hypothèses sont en fait très générales pour caractériser un modèle de discrétisation. La plupart des méthodes de discrétisation existantes les respectent, au moins implicitement. Par exemple, les méthodes ChiSplit et ChiMerge basées sur le critère statistique du Khi2 recherchent des intervalles qui soient les plus indépendants possibles deux à deux. La méthode MDLPC évalue l'information dans chaque intervalle par comptage du nombre de symboles dans chaque intervalle.

Notations:
n: nombre de symboles de la chaîne à discrétiser,
J: nombre de types de symboles possibles,
I: nombre d'intervalles,
n_i: nombre de symboles de l'intervalle *i*,
n_{ij}: nombre de symboles de type *j* dans l'intervalle *i*.
Une discrétisation de type SDM est entièrement caractérisée par le choix des paramètres $\{ I, \{n_i\}_{1\le i\le I}, \{n_{ij}\}_{1\le i\le I,\, 1\le j\le J} \}$.

Définition: Un modèle SDM est *compatible* avec une chaîne *S* si les sous-chaînes correspondant aux intervalles définis par le modèle ont une distribution de symboles identique à celle définie par le modèle.

Théorème: Un modèle SDM d'une chaîne *S* ne peut-être optimal au sens de Bayes que s'il est compatible avec *S*.

En effet, la probabilité qu'une chaîne *S* non compatible avec un modèle SDM soit conforme à ce modèle est par définition nulle. L'intérêt de ce résultat est que tout algorithme d'optimisation d'une discrétisation SDM d'une chaîne *S* peut se contenter de parcourir les modèles compatibles avec *S*, c'est à dire que l'on peut se limiter au choix des points de coupure des intervalles, le choix des distributions par intervalle étant donné par la chaîne *S*.

Définition: On appelle *a priori d'un modèle de discrétisation* toute distribution de probabilité portant sur les réalisations possibles du modèle.

2.1.2 Discrétisation avec a priori a trois étages

Définition: On appelle *a priori à trois étages* l'a priori de modèle SDM basé sur les hypothèses suivantes:
- le nombre d'intervalles est compris entre 1 et *n*, de façon équiprobable,
- pour un nombre d'intervalles donné, toutes les partitions en intervalles de la chaîne à discrétiser sont équiprobables,
- pour un intervalle donné, toutes les distributions de symboles sont équiprobables,
- les distributions des symboles sur chaque intervalle sont indépendantes les unes des autres.

Cet a priori est essentiellement un a priori uniforme à chaque niveau de la hiérarchie des paramètres des modèles SDM. L'hypothèse d'indépendance sur les distributions des symboles par intervalle est supposée au moins implicitement par la plupart des méthodes de discrétisation existantes. Ainsi par exemple, la méthode ChiMerge (Kerber, 1991) fusionne les intervalles si leur distribution est statistiquement similaire (en utilisant un test d'indépendance du Khi2).
En utilisant la définition formelle des modèles SDM et de leur distribution a priori, la règle de Bayes est applicable pour calculer de manière exacte la probabilité a priori des modèles et des données connaissant un modèle, ce qui nous conduit au théorème suivant.

Théorème: Un modèle de discrétisation standard *M* suivant un a priori à trois étages est optimal au sens de Bayes si son évaluation par la formule suivante est minimale sur l'ensemble de tous les modèles:

$$Value(M) = \log(n) + \log\left(C_{n+I-1}^{I-1}\right) + \sum_{i=1}^{I} \log\left(C_{n_i+J-1}^{J-1}\right) + \sum_{i=1}^{I} \log\left(n_i!/n_{i,1}!n_{i,2}!...n_{i,J}!\right)$$

Le premier terme correspond au choix du nombre d'intervalles et le second terme au choix des bornes des intervalles. Le troisième terme représente le choix des distributions de symboles cibles dans chaque intervalle et le dernier terme le codage de la probabilité des données connaissant le modèle de discrétisation.

Théorème: Dans un modèle de discrétisation SDM optimal suivant un a priori à trois étages il ne peut y avoir de point de coupure entre deux instances ayant même classe cible.

L'intérêt de ce théorème est de permettre d'améliorer le temps de calcul des algorithmes d'optimisation des discrétisations, en restreignant l'espace de recherche des modèles aux modèles ne coupant jamais une sous-chaîne pure (ne comportant qu'un seul type de symbole). Il suffit ainsi dans un prétraitement de constituer des intervalles élémentaires à partir des séquences pures, ce qui limite d'autant le nombre de coupures potentielles.

Théorème: Dans un modèle de discrétisation SDM optimal suivant un a priori à trois étages, il ne peut exister une discrétisation comportant deux intervalles singletons successifs.

Ce théorème permet d'évaluer la pertinence de l'a priori à trois étages de façon intuitive, en vérifiant qu'un modèle isolant deux instances successives ne peut être optimal en généralisation. Afin de mieux appréhender l'a priori a trois étages, on va tenter dans le théorème suivant de qualifier le comportement des discrétisations optimales en présence des chaînes aléatoires. (Li & Vitanyi, 1997) ont démontré qu'asymptotiquement une chaîne de bits aléatoire de longueur n ne peut comporter une sous-chaîne particulière de longueur supérieure à $\log(n) + \log(\log(n))$ et doit comporter de telles sous-chaînes si leur longueur est inférieure à $\log(n) - \log(\log(n))$. Une sous-chaîne particulière est une sous-chaîne ne contenant que des 0 par exemple, ou contenant une séquence prédéterminée de bits. Pour évaluer le critère de discrétisation optimal, on s'intéresse ici à des sous-chaînes "extrêmes" pour le problème de discrétisation, contenant pour moitié des 0 en tête de sous-chaîne et des 1 en fin de sous-chaîne. Le théorème suivant s'intéresse à la longueur maximale d'une sous-chaîne extrême dans une discrétisation optimale pour l'a priori à trois étages.

Théorème: On se place ici dans le cas où le nombre de classes cibles est égal à deux. Dans un modèle de discrétisation SDM optimal suivant un a priori à trois étages, la longueur maximale d'un intervalle extrême (décomposable en deux intervalles purs de même taille) est définie par l'équation suivante:

$$L - 3/2\log_2(L) = \log_2(n/I) + O(1)$$

L'analyse suivante n'est pas une démonstration, mais permet d'apprécier intuitivement le comportement du critère dans le cas d'une chaîne aléatoire. Les petites sous-chaînes (de longueur plus petite que $\log(n) - \log(\log(n))$) sont fréquentes et la plupart ne sont pas extrêmes, ce qui entraîne la constitution d'intervalles de plus en plus gros, et en conséquence une diminution du nombre d'intervalles. Quand *I* devient très petit, L devient proche de n, et la taille maximale d'un intervalle extrême devient bornée par log(n). La longueur maximale d'une sous-chaîne extrême dans une discrétisation est alors du même ordre que dans une chaîne aléatoire. Cela signifie que l'a priori à trois étages implique que le seuil de décision déterminant les discrétisations aboutissant à un seul intervalle terminal correspond intuitivement au modèle de discrétisation que l'on souhaite obtenir pour les chaînes aléatoires.

Le codage à trois étages est également intéressant, parce qu'il permet d'incorporer aisément des contraintes supplémentaires. Si par exemple on souhaite imposer une contrainte d'effectif minimum *Min* par intervalle pour des raisons de lisibilité du modèle pour l'utilisateur final, et que l'on considère une variante de l'a priori à trois étages, où toutes les partitions en intervalles répondant aux contraintes sont équiprobables pour un nombre d'intervalles donné, on vérifie alors que le modèle optimal est obtenu en minimisant cette fois le critère suivant:

$$Value(M) = \log(n) + \log\left(C_{n-I.Min+I-1}^{I-1}\right) + \sum_{i=1}^{I} \log\left(C_{n_i+J-1}^{J-1}\right) + \sum_{i=1}^{I} \log\left(n_i! / n_{i,1}! n_{i,2}! .. n_{i,J}!\right)$$

Il est à noter que dans ce cas, les points de coupures peuvent éventuellement couper les sous-chaînes pures.

2.1.3 Commentaires

La plupart des méthodes de discrétisation sont basées sur un critère d'évaluation et un algorithme d'optimisation, qui implicitement définissent un a priori en favorisant certains modèles soit par le critère, soit par l'heuristique d'optimisation. L'originalité de l'approche MODL est de définir explicitement cet a priori, ce qui permet de proposer un critère dont l'optimisation garantit l'optimalité de la discrétisation au sens de Bayes.

Il y a plusieurs différences entre l'approche MDL et l'approche bayésienne de MODL. Dans MDL, on doit choisir un codage de quantité d'information qui paraisse le plus naturel possible. En fait, un tel choix de codage correspond implicitement aux choix d'un a priori sur les modèles. On a montré en annexe 5 que tous les a priori ne sont pas équivalents, et qu'en particulier, il n'est pas possible d'apprendre en l'absence de biais inductif. Ce problème ne transparaît pas explicitement dans le choix de codage MDL. Ici, il apparaît clairement que l'a priori à trois étages est à la fois le plus "naturel" en terme de codage d'information, et celui dont le comportement induit est le plus satisfaisant vis à vis des chaînes aléatoires. Dans toute la suite de ce papier, c'est l'a priori à trois étages qui sera choisi.

Un autre apport important de l'approche MODL est que le choix du modèle résultant de l'optimisation du critère d'évaluation conduit au modèle optimal pour un a priori donné, sans limite de taille d'échantillon. Dans le cas du MDL, l'optimisation du critère conduit au meilleur modèle uniquement de façon asymptotique, et ce dans le cas où le codage utilisé a été correctement élaboré.

On peut envisager des extensions aux modèles de discrétisation SDM. Par exemple, si l'on tient compte du nombre de valeurs effectives V de l'attribut descriptif, on peut tirer parti du fait que les bornes des intervalles sont limitées au choix de I bornes parmi V (au lieu de I parmi n). Cela raccourcit la longueur de codage de la partie modèle, mais on rend par la même équivalent (du moins pour cette partie du codage) les intervalles réduits à une instance et les intervalles comportant un très grand nombre d'instances, ce qui donne des discrétisations parfois peu intuitives. De plus, on augmente la sensibilité aux problèmes de précision numérique, ce qui risque d'entraîner de grandes variations de discrétisation en changeant la précision numérique avec laquelle les attributs descriptifs sont mesurés.

Un autre axe d'extension des modèles de discrétisation SDM est de permettre des distributions de probabilité continues (et non discrètes) pour les paramètres des modèles. Cela a par exemple été étudié dans le cas d'un modèle simple de Bernouilli dans le codage "Mixture Coding", et finalement fournit un résultat identique au cas du codage discret "Normalized Maximized Likelihood Coding", au prix d'un calcul nettement plus complexe.

On peut également relâcher la contrainte d'indépendance des distributions entre les intervalles. On aborde là une nouvelle famille de modèles, qui vont bien au delà de ce qui est communément appelé discrétisation.

L'a priori à trois étage paraît naturel et possède des propriétés intéressantes. D'autres a priori basés sur des variantes dans la hiérarchie des choix des paramètres d'une discrétisation ont été étudiés et sont présentés en annexe. Il ressort de cette étude comparative que l'a priori à trois étages, qui décompose le plus la hiérarchie des choix des paramètres (nombre d'intervalles, puis bornes des intervalles, puis distribution des classe cible par intervalle), est clairement le plus satisfaisant. On peut noter que l'a priori "uniforme" qui suppose que tous les choix de paramétrage sont équiprobables conduit à un apprentissage "par cœur" inutilisable, avec des discrétisations comportant un intervalle par instance. On retrouve de façon théorique que l'apprentissage n'est pas possible en l'absence d'un biais (a priori) sur les modèles.

On peut également envisager toutes sortes d'a priori, plus ou moins complexes. Par exemple, on peut combiner deux a priori existants en un nouvel a priori pour lequel les deux a priori initiaux sont équiprobables. Il suffit alors de prendre un nouveau codage min(codage A, codage B) avec un bit supplémentaire indiquant le codage retenu pour avoir un codage éventuellement de longueur plus petite en moyenne. Néanmoins, ces variantes de codage complexifient l'algorithme et augmentent son temps de calcul pour des gains probablement mineurs en généralisation. On peut se poser la question de l'existence d'un a priori "universel", qui serait optimal parmi tous les a priori possibles. La complexité algorithmique de Kolmogorov fournit intuitivement un élément de réponse à cette question, en tentant de trouver un codage du modèle qui soit de longueur minimale. Néanmoins, il n'est pas certain qu'il existe un algorithme permettant de trouver le codage de longueur minimale dans le cas des modèles de discrétisation, et de plus l'optimal n'est atteint que de façon asymptotique, ce qui n'est pas le cas pour l'approche MODL. Il s'agit alors de trouver un compromis acceptable entre la qualité de l'a priori retenu et sa facilité de mise en œuvre. Dans ce contexte, l'a priori à trois étages paraît indiqué.

2.2 Algorithmes d'optimisation

On présente dans cette partie différents algorithmes d'optimisation pour la recherche d'une discrétisation de coût minimal. On introduit la notion de critère décomposable sur les intervalles afin de proposer d'une part un algorithme optimal de complexité $O(n^3)$ et d'autre part plusieurs heuristiques d'optimisation de complexité $O(n.\log(n))$.

2.2.1 Critère décomposable sur les intervalles

La discrétisation d'une chaîne S permet de partitionner l'ensemble des instances initiales en I intervalles S_i.

Définition: Un critère d'évaluation d'une discrétisation est décomposable sur les intervalles si:

1. il permet une évaluation globale de la discrétisation,
2. il se décompose de manière additive en une évaluation de la partition, ne dépendant que de S et de I, et une évaluation de chaque intervalle ne dépendant que de S_i,

$$DiscretizationCost(S, I, \{S_i, 1 \le i \le I\}) = PartitionCost(S, I) + \sum_{i=1}^{I} IntervalCost(S_i)$$

3. chaque terme de la décomposition est borné (ce qui permet d'optimiser le critère).

Le premier point permet de comparer deux partitions quelconques et donc de rechercher un optimum global du critère d'évaluation. D'après ce premier point, les algorithmes utilisant un critère d'évaluation local à deux intervalles (MDLPC, ChiMerge, ChiSplit par exemple) n'utilisent pas un critère décomposable sur les intervalles.

Le cas du critère MDL-DISC (cf. annexe 4), qui utilise une évaluation globale, est plus subtil.

DiscretizationCost =

$(I_{max}\text{-}1).ent(I\text{-}1, I_{max}\text{-}1) +$	Codage des points de coupures
$I.ent(I_1, I) +$	Codage des classes majoritaires de chaque intervalle
$\sum_i n_i .ent(n_{imaj}, n_i)$	Codage des exemples de la classe majoritaire dans chaque intervalle

Le premier terme est de type évaluation de partition, alors que le dernier terme est de type évaluation d'intervalle. Par contre, le deuxième terme dépend à la fois du nombre d'intervalles total, qui est une information globale à la partition, et du nombre d'intervalles ayant la première classe cible comme classe majoritaire, qui est une information locale dépendant de chaque intervalle. De ce fait, le critère MDL-DISC n'est pas décomposable sur les intervalles.

On vérifie aisément que les critères d'évaluation MODL sont décomposables sur les intervalles. Par exemple, pour l'a priori à trois étages, on a:

$$PartitionCost(S, I) = \log(C_{n+I-1}^{I-1})$$

$$IntervalCost(S_i) = \log(C_{n_i+J-1}^{J-1}) + \log(n_i! / n_{i,1}! n_{i,2}! ... n_{i,J}!)$$

On peut noter que les critères d'évaluation reposent uniquement sur log(factorielle), qui peut être évalué en O(1), soit en tabulant cette fonction, soit en utilisant la formule de Stirling qui donne une excellente précision numérique.

L'intérêt essentiel des critères d'évaluation décomposables sur les intervalles est de permettre la conception d'algorithmes d'optimisation performants en termes de complexité algorithmique. Dans la suite, on se place dans le cadre de critères d'évaluation décomposables sur les intervalles.

2.2.2 Algorithme OPTD optimal

On présente ici l'algorithme OPTD: Optimal Discretization.

Cet algorithme permet de trouver la discrétisation de coût optimal en $O(n^3)$ pour un critère d'évaluation décomposable sur les intervalles. Cet algorithme s'inspire directement de l'algorithme de programmation dynamique décrit par de nombreux auteurs (Fischer, 1958; Lechevallier, 1990; Fulton, Kasif and Salzberg, 1995; Elomaa & Rousu, 1996) qui pour un critère additif trouve la meilleure partition en moins de I intervalles (I fixés) en $O(I.\ n^2)$. Un critère est additif si pour une partition optimale de S en I intervalles S_1, $S_2 \ldots S_I$, la partition de $(S\text{-}S_1)$ en $(I\text{-}1)$ intervalles est optimale sur $S_2 \ldots S_I$.

On vérifie aisément qu'un critère d'évaluation décomposable sur les intervalles est un critère additif.

$$DiscretizationCost(S, I) = PartitionCost(S, I) + \sum_{i=1}^{I} IntervalCost(S_i)$$

$$DiscretizationCost(S, I) = PartitionCost(S, I) - PartitionCost(S - S_1, I - 1) + IntervalCost(S_1) +$$

$$PartitionCost(S - S_1, I - 1) + \sum_{i=2}^{I} IntervalCost(S_i)$$

Si le coût est optimal pour le découpage de S en I intervalles, alors la formule ci-dessus montre que le coût est optimal pour le découpage de $(S\text{-}S_1)$ en $(I\text{-}1)$ intervalles.

On peut donc appliquer l'algorithme de programmation dynamique, dont on va rappeler ci-dessous les grandes lignes.

Soit S l'ensemble initial composé des n instances.

Soit S^k le sous ensemble de S composé des instances k à n. On a $S = S^1$.

Dans une étape d'initialisation, on cherche la meilleure partition des ensembles S^k en un intervalle.

On a trivialement $S^k = [k, n]$.

A chaque étape suivante, on part d'un état initial où l'on dispose pour chaque ensemble S^k de sa partition en I intervalles, et on cherche la meilleure partition en I+1 intervalles.

Posons $LocalCost(S,I) = DiscretizationCost(S,I) - PartitionCost(S,I) = \sum_{i=1}^{I} IntervalCost(S_i)$

Pour un I donné, optimiser le $DiscretizationCost$ est équivalent à optimiser le $LocalCost$.

$$Min(DiscretizationCost(S^k, I+1)) = PartitionCost(S^k, I+1) + \underset{k'>k}{Min}(IntervalCost([k,k'])) + Min(LocalCost(S^{k'}, I)))$$

Il est alors aisé de calculer la partition optimale en I+1 intervalles pour chacun des ensembles S^k en parcourant les discrétisations optimales en I intervalles des ensembles $S^{k'}$ pour $k'>k$, ce qui correspond à une complexité algorithmique en $O(n^2)$ à chaque étape.

A chaque étape, on a la meilleure partition de $S = S^I$ en I intervalles, et on peut évaluer son coût global. Arrivé à l'étape I, on a ainsi trouvé (en mémorisant la meilleure solution rencontrée) la meilleure discrétisation en moins de I intervalles.

Il y a au plus n étapes, ce qui entraîne une complexité globale en $O(n^3)$ pour la recherche de la partition optimale en moins de n intervalles.

2.2.3 Algorithme GBUD glouton ascendant

On présente ici l'algorithme GBUD: Greedy Bottom Up Discretization.

L'algorithme glouton ascendant est l'algorithme classique, utilisé par exemple dans ChiMerge, qui part des intervalles élémentaires, envisage toutes les fusions d'intervalles possibles, et évalue la meilleure fusion au sens du critère à optimiser. Si le critère d'arrêt n'est pas atteint, la fusion est effectuée et l'algorithme est réitéré.

Une implémentation naïve de cet algorithme implique une complexité en n^2 fois le coût d'évaluation d'une discrétisation.

- Initialisation (tri des valeurs de l'attribut): en $O(n.\log(n))$
- Optimisation de la discrétisation
  - Répéter (au plus n étapes)
    - Evaluer toutes les fusions possibles d'intervalles adjacents : n évaluation discrétisation
    - Chercher la meilleure fusion : en $O(n)$
    - Evaluer la condition d'arrêt : en $O(1)$

On va montrer que pour un critère d'évaluation décomposable sur les intervalles, on peut proposer un algorithme glouton ascendant de complexité algorithmique $n.\log(n)$.

$$DiscretizationCost(S, I, \{S_i, 1 \le i \le I\}) = PartitionCost(S,I) + \sum_{i=1}^{I} IntervalCost(S_i)$$

Suite à la fusion de deux lignes adjacentes i_1 et i_2 ($i_2 = i_1+1$), la nouvelle évaluation de la discrétisation est:

$$DiscretizationCost(Merge_{i_1}) = PartitionCost(S, I-1) + \sum_{i=1}^{i_1-1} IntervalCost(S_i) + IntervalCost(S_{i_1} \cup S_{i_2}) + \sum_{i=i_2+1}^{I} IntervalCost(S_i)$$

La variation du coût suite à la fusion des deux intervalles est:

$$\Delta DiscretizationCost(Merge_{i_1}) = PartitionCost(S, I-1) - PartitionCost(S,I) + IntervalCost(S_{i_1} \cup S_{i_2}) - IntervalCost(S_{i_1}) - IntervalCost(S_{i_2})$$

Soient $\Delta PartitionCost(S,I) = PartitionCost(S, I-1) - PartitionCost(S,I)$,

$\Delta IntervalCost(Merge_{i_1}) = IntervalCost(S_{i_1} \cup S_{i_2}) - IntervalCost(S_{i_1}) - IntervalCost(S_{i_2})$.

On a $\Delta DiscretizationCost(Merge_{i_1}) = \Delta PartitionCost(S,I) + \Delta IntervalCost(Merge_{i_1})$

Cette formule permet de rechercher la meilleure fusion d'intervalles en évaluant uniquement les variations de coût sur les intervalles, puis d'évaluer le critère d'arrêt de l'algorithme en comparant la variation du coût des intervalles à la variation du coût de la partition, qui est elle indépendante du choix des intervalles fusionnés. Il suffit alors à chaque étape de mémoriser pour chaque intervalle de l'algorithme son coût et la variation de son coût suite à fusion avec son successeur. Après une fusion d'intervalles, seuls les intervalles adjacents aux deux intervalles fusionnés doivent être mis à jour pour préparer l'étape suivante. La partie critique de l'algorithme devient alors la recherche de la meilleure fusion à chaque étape. Cette recherche est en $O(n)$. Si l'on trie préalablement la liste des fusions possibles, et que l'on maintient cette liste triée au cours de l'optimisation de la discrétisation, la recherche du meilleur élément est en $O(1)$, au prix du coût de gestion de la liste triée. Les arbres binaires de recherche équilibrés (AVL Binary Search Tree par exemple) permettent de gérer de telles listes triées en maintenant l'ordre dans la liste lors d'insertions/suppressions à un coût logarithmique. En se basant sur la mémorisation des $IntervalCost$ et des $\Delta IntervalCost$, et sur l'utilisation d'une liste triée de type arbre binaire de recherche équilibré, on arrive alors à une complexité globale en $O(n.\log(n))$.

Algorithme GBUD

- Initialisation
  - Tri des valeurs de l'attribut descriptif : en O(n.log(n))
  - Création d'un intervalle élémentaire par valeur de la loi source : en O(n)
  - Calcul des *IntervalCost* initiaux : en O(n)
  - Calcul des $\Delta IntervalCost$: en O(n)
  - Tri des fusions d'intervalles par valeur de $\Delta IntervalCost$: en O(n.log(n))
- Optimisation de la discrétisation
  - Répéter: n étapes
    - Chercher la meilleure fusion : en 1 en prenant le premier élément de la liste triée
    - Evaluer la condition d'arrêt
      - ✓ Arrêter si $\Delta DiscretizationCost = \Delta PartitionCost + \Delta IntervalCost \geq 0$
      - ✓ Continuer sinon (et effectuer la meilleure fusion)
    - Si continuer : effectuer la fusion d'intervalle
      - Calcul du *IntervalCost* pour le nouvel intervalle : en O(1)
      - Calcul des $\Delta IntervalCost$ pour les deux intervalles adjacents au nouvel intervalle
      - Mise à jour de la liste triée des $\Delta IntervalCost$: en O(log(n))
        - ✓ Suppression du $\Delta IntervalCost$ du nouvel intervalle
        - ✓ Suppression des anciens $\Delta IntervalCost$ des intervalles adjacents aux deux sous intervalles sources du nouvel intervalle
        - ✓ Ajout des nouveaux $\Delta IntervalCost$ des intervalles adjacents au nouvel intervalle

On peut noter que l'occupation mémoire nécessaire pour l'algorithme est également en O(n.log(n)). On doit en effet mémoriser n *IntervalCost* lignes, n $\Delta IntervalCost$, et une structure de liste triée de type arbre binaire de recherche équilibré qui a une occupation mémoire de O(n.log(n)).

2.2.4 Algorithme GTDD glouton descendant

On présente ici l'algorithme GTDD: Greedy Top Down Discretization.

L'algorithme glouton descendant est l'algorithme classique, utilisé par exemple dans ChiSplit ou MDLPC, qui part du domaine numérique initialement complet, envisage toutes les coupures en deux intervalles, et évalue la meilleure coupure au sens du critère à optimiser. Si le critère d'arrêt n'est pas atteint, la coupure est effectuée et l'algorithme est réitéré.

Cet algorithme peut être décrit de la façon suivante:

- Initialisation (tri des valeurs de l'attribut)
- Optimisation de la discrétisation
  - Répéter
    - Chercher la meilleure bipartition de l'intervalle en évaluant toutes les coupures possibles
    - Si condition d'arrêt fausse, réitérer sur les deux sous-intervalles

Chaque recherche de bipartition dans un intervalle de taille n est en O(n). Si on effectue cette recherche dans les deux sous-intervalles de taille n_1 et n_2, le coût de la recherche des bipartitions est également en O(n)=O(n_1)+ O(n_2). A chaque niveau de profondeur, la complexité globale reste donc constante. Si l'arbre des bipartitions est équilibré, sa profondeur est $\log_2(n)$, ce qui donne une complexité globale en n.log(n). On n'a cependant aucune garantie d'équilibrage de l'arbre des bipartitions, ce qui implique que la complexité de l'algorithme peut dégénérer en O(n^2). En pratique, on constate néanmoins que les discrétisations sont souvent très rapides, principalement parce que le nombre d'intervalles final est en général très faible devant le nombre d'instances, ce qui entraîne un arrêt de l'algorithme en au plus $I.n$ étapes.

Cet algorithme récursif est particulièrement adapté dans le cas d'un critère d'évaluation de bipartition, local à deux intervalles. On propose ici une adaptation aux critères d'évaluation globaux décomposables par intervalles, qui garde la même complexité algorithmique.

Au tout départ, on recherche la meilleure bipartition en deux sous-intervalles en évaluant tous les points de coupures potentiels, et on effectue la coupure si l'évaluation globale de la bipartition meilleure que l'évaluation de l'intervalle complet initial.

Supposons maintenant qu'à une étape de l'algorithme, on dispose de I intervalles. Tous ces intervalles peuvent être évalués en calculant leur *IntervalCost* , en O(n). Pour un intervalle donné i_1, on va chercher sa meilleure coupure au sens global en deux sous-intervalles i_{1a} et i_{1b}. Suite à cette coupure, le nouveau coût de discrétisation est:

$$DiscretizationCost\left(Split_{i_1}\right) = PartitionCost\left(S, I+1\right) +$$

$$\sum_{i=1}^{i_1-1} IntervalCost\left(S_i\right) + IntervalCost\left(S_{i_{1a}}\right) + IntervalCost\left(S_{i_{1b}}\right) + \sum_{i=i_1+1}^{I} IntervalCost\left(S_i\right)$$

La variation du coût suite à la coupure des deux intervalles est:

$$\Delta DiscretizationCost(Split_{i_1}) = PartitionCost(S, I+1) - PartitionCost(S, I) + IntervalCost(S_{i_{1a}}) + IntervalCost(S_{i_{1b}}) - IntervalCost(S_{i_1})$$

Soient $\Delta PartitionCost(S, I) = PartitionCost(S, I+1) - PartitionCost(S, I)$,

$\Delta IntervalCost(Split_{i_1}) = IntervalCost(S_{i_{1a}}) + IntervalCost(S_{i_{1b}}) - IntervalCost(S_{i_1})$.

On a $\Delta DiscretizationCost(Split_{i_1}) = \Delta PartitionCost(S, I) + \Delta IntervalCost(Split_{i_1})$.

Cette formule permet de rechercher la meilleure coupure d'intervalles en évaluant uniquement les variations des coûts d'intervalles, puis d'évaluer le critère d'arrêt de l'algorithme en comparant la variation du coût des intervalles à la variation du coût de la partition, qui est elle indépendante du choix des intervalles coupés. Il suffit alors à chaque étape de mémoriser pour chaque intervalle de l'algorithme son coût et la variation de son coût suite à sa bipartition. Après une coupure d'intervalles, seuls les deux sous-intervalles issus de la coupure doivent être mis à jour pour préparer l'étape suivante. La partie critique de l'algorithme devient alors la recherche de la meilleure fusion à chaque étape. Cette recherche est en O(n). Si l'on trie préalablement la liste des coupures possibles, et que l'on maintient cette liste triée au cours de l'optimisation de la discrétisation, la recherche du meilleur élément est en O(1), au prix du coût de gestion de la liste triée (en O(log(n)) par insertion/suppression dans la liste triée).

Algorithme GTDD

- Initialisation
  - Tri des valeurs de l'attribut descriptif : en O(n.log(n))
  - Création d'un intervalle initial égal au domaine numérique complet
  - Calcul de son *IntervalCost* initial
  - Calcul du Δ*IntervalCost* de l'intervalle en recherche sa meilleure bipartition: en O(n)
  - Insertion de cette meilleure bipartition initiale dans une liste triée des bipartitions
- Optimisation de la discrétisation
  - Répéter: au plus n étapes
    - Chercher la meilleure bipartition : en 1 en prenant le premier élément de la liste triée
    - Evaluer la condition d'arrêt
      - ✓ Arrêter si $\Delta DiscretizationCost = \Delta PartitionCost + \Delta IntervalCost \geq 0$
      - ✓ Continuer sinon (et effectuer la meilleure bipartition)
    - Si continuer : effectuer la bipartition de l'intervalle
      - Calcul du *IntervalCost* pour chacun des deux sous-intervalles : en O(1) (ils sont mémorisés)
      - Calcul des Δ*IntervalCost* pour les deux intervalles adjacents au nouvel intervalle: en O(taille(intervalle))
      - Mise à jour de la liste triée des Δ*IntervalCost* : en O(log(n))
        - ✓ Suppression du Δ*IntervalCost* de l'intervalle
        - ✓ Ajout des nouveaux Δ*IntervalCost* des deux sous-intervalles

En se basant sur la mémorisation des *IntervalCost* et des Δ*IntervalCost* , et sur l'utilisation d'une liste triée de type arbre binaire de recherche équilibré, on arrive alors à une complexité globale en O(n.log(n)) pour la partie gestion des meilleures bipartitions. Pour la partie calcul des bipartitions, si l'on fait l'hypothèse d'équilibrage des effectifs des intervalles de la discrétisation en cours, l'effectif moyen des intervalles à chaque étape est de n/I. Le nombre total de points de coupure évalués lors du déroulement complet de l'algorithme est alors $\sum_{I=1}^{n} n/I \sim n.\log(n)$. Néanmoins, dans le pire des cas où la discrétisation finale comporte n intervalles et où chaque bipartition intermédiaire consistait à isoler un singleton, la complexité de l'algorithme dégénère en O(n^2). En pratique, cette dégénérescence est d'une part peu probable, et d'autre part la complexité est bornée par $n.I$ ou I est le nombre d'intervalles de la discrétisation finale.

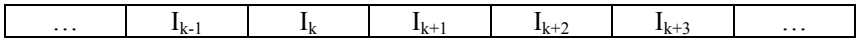
On peut noter que l'occupation mémoire nécessaire pour l'algorithme est en O(n.log(n)). On doit en effet mémoriser n *IntervalCost* lignes, n Δ*IntervalCost* , et une structure de liste triée de type arbre binaire de recherche équilibré qui a une occupation mémoire en O(n.log(n)).

2.2.5 Algorithme OBUD ascendant optimisé

On présente ici l'algorithme OBUD: Optimized Bottom Up Discretization.

L'inconvénient de l'algorithme glouton GBUD est de risquer de tomber dans un minimum local. D'une part, l'algorithme risque de produire trop d'intervalles, d'autre part, les frontières n'étant pas remises en cause risquent d'être mal ajustées. On propose ici une post-optimisation de l'algorithme GBUD, procédant en plusieurs étapes. Chacune de ces étapes est basée sur des opérations élémentaires de fusions d'intervalles adjacents (Merge) ou de coupure d'un intervalles en deux sous-intervalles (Split). Les combinaisons envisagées sont présentées graphiquement ci-dessous.

Intervalles de la discrétisation

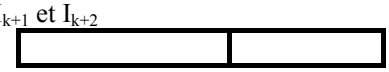


Split de l'intervalle I_k

Merge des intervalles I_k et I_{k+1}

MergeSplit des intervalles I_k et I_{k+1}

MergeMergeSplit des intervalles I_k, I_{k+1} et I_{k+2}



Exhaustive Merge

Cette étape consiste à forcer l'algorithme glouton descendant à accepter tous les fusions (Merges) d'intervalles inconditionnellement jusqu'à obtenir un seul intervalle final, et à mémoriser la discrétisation de coût minimal rencontrée au cours du processus. Cet algorithme permet de sortir d'un minimum local en cumulant plusieurs fusions consécutives, et garde la complexité en O(n.log(n)) de l'algorithme initial.

Greedy Split

Il s'agit cette fois de partir de la meilleure solution rencontrée, et d'évaluer toutes les bipartitions des intervalles de la discrétisation. La phase d'initialisation consiste à évaluer les *IntervalCost* et les Δ*IntervalCost* de chaque intervalle, ce qui repose sur une complexité en O(n). On est alors exactement dans le même cas que pour l'algorithme glouton descendant, cette fois en partant d'une discrétisation initiale comportant plusieurs intervalles.

Greedy MergeSplit

Un MergeSplit est une fusion de deux intervalles suivie de la recherche de la meilleure bipartition. Il s'agit en fait de la recherche de la meilleure frontière entre deux intervalles consécutifs. On peut alors utiliser un algorithme inspiré de l'algorithme Greedy Split, en mémorisant pour chaque intervalle son coût *IntervalCost* et la variation de coût Δ*IntervalCost* suite à un MergeSplit avec son intervalle suivant. La phase d'initialisation consiste à évaluer les *IntervalCost* et les Δ*IntervalCost* de chaque intervalle, ce qui repose sur une complexité en O(n) (en fait, les points de coupures sont tous évalués deux fois, lors d'un MergeSplit entre l'intervalle et son successeur, et entre l'intervalle et son prédécesseur). On est alors dans une situation algorithmique similaire à celle de l'algorithme Greedy Split:

- le Δ*DiscretizationCost* est cette fois nul (il n'y a pas de changement de nombre d'intervalles),
- les MergeSplits sont rangés dans une liste triée,
- chaque MergeSplit effectué entraîne le recalcul des Δ*IntervalCost* de trois intervalles consécutifs.

Greedy MergeMergeSplit

Un MergeMergeSplit est une fusion de trois intervalles consécutifs suivie de la recherche de la meilleure bipartition. Il s'agit d'un redécoupage optimal de trois intervalles en deux intervalles. Le résultat est donc meilleur que le meilleur des deux Merges concernant le premier et le deuxième intervalle, ou le deuxième et le troisième intervalle. Suite à un MergeMergeSplit, le nombre d'intervalles de la discrétisation diminue de 1, mais le Δ*DiscretizationCost* correspondant est indépendant du MergeMergeSplit, ce qui permet de se ramener au cas des deux algorithmes précédents. On peut alors utiliser un algorithme inspiré de l'algorithme Greedy Split, en mémorisant pour chaque intervalle son coût *IntervalCost* et la variation de coût Δ*IntervalCost* suite à un MergeMergeSplit avec ses deux intervalles suivants. La phase d'initialisation consiste à évaluer les *IntervalCost* et les Δ*IntervalCost* de chaque intervalle, ce qui repose sur une complexité en O(n) (en fait, les points de coupures sont tous évalués trois fois par intervalle). On est alors dans une situation algorithmique similaire à celle de l'algorithme Greedy Split:

- les MergeMergeSplit sont rangés dans une liste triée,
- chaque MergeMergeSplit effectué entraîne le recalcul des Δ*IntervalCost* de quatre intervalles consécutifs.

Greedy Post-Optimization

On va cette fois rechercher une amélioration locale d'une discrétisation en évaluant simultanément toutes les améliorations de type Split, MergeSplit et MergeMergeSplit. Les Splits recherchent le meilleur découpage d'un des intervalles et augmentent de un le nombre d'intervalles de la discrétisation. Les MergeSplit recherchent le meilleur changement de frontière entre deux intervalles consécutifs, et laissent invariant le nombre d'intervalles de la discrétisation. Les MergeMergeSplit recherchent le meilleur redécoupage de trois intervalles consécutifs en deux intervalles adjacents, et diminuent de un le nombre d'intervalles de la discrétisation. L'intérêt d'effectuer les trois algorithmes simultanément est d'une part d'améliorer te temps de convergence de l'algorithme, en recherchant la meilleure des améliorations parmi tous les types d'amélioration possibles, d'autre part d'optimiser les mises a jours des structures algorithmiques dès qu'une amélioration est retenue. L'inconvénient de cet

algorithme est sa complexité de mise au point, qui demande une implémentation particulièrement soigneuse. On a ici besoin de trois listes triées, une par type d'amélioration. La recherche de la meilleure amélioration possible se fait en prenant le meilleur $\Delta IntervalCost$ de chaque liste, et en le comparant au $\Delta DiscretizationCost$ correspondant. La mise à jour des structures se fait en se basant sur le tableau d'impacts suivants (qui ne tient pas compte des effets de bord), en préfixant par '-' les améliorations à retirer des listes d'amélioration possibles, et par '+' les améliorations à recalculer et à insérer dans les listes d'améliorations possibles.

Amélioration effectuée	Nombre d'intervalles	Split	MergeSplit	MergeMergeSplit
Split	+1	-1+2	-2+3	-3+4
MergeSplit		-2+2	-3+3	-4+4
MergeMergeSplit	-1	-3+2	-4+3	-5+4

On obtient ici une complexité d'initialisation des structures en $O(n.\log(n))$. Chaque amélioration a une complexité moyenne en $O(n/I)$ où I est le nombre d'intervalles de la discrétisation. Cette complexité peut néanmoins dégénérer en $O(n)$. Au final, on est limité par le nombre A d'améliorations (complexité en $O(A.n)$). Le processus converge car on diminue le coût à chaque étape. En pratique, la convergence est très rapide, car on part d'une solution déjà optimisée par l'algorithme Exhaustive Merge et qu'on choisit à chaque étape la meilleure amélioration possible. On peut néanmoins garantir une post-optimisation en $O(n.\log(n))$ en forçant la post-optimisation à s'arrêter dès que le nombre d'améliorations effectuées dépasse $\log(n)$.

Algorithme OBUD

On récapitule ci-dessous les étapes de l'algorithme OBUD:

- Optimisation gloutonne ascendante GBUD
- Recherche d'une meilleure discrétisation par Exhaustive Merge
- Post-optimisation de la discrétisation par Greedy Post-Optimization
  Recherche simultanée des types d'amélioration suivants
  - Split: bipartition d'un des intervalles
  - MergeSplit: changement de la frontière entre deux intervalles successifs
  - MergeMergeSplit: redécoupage de trois intervalles consécutifs en deux intervalles

2.2.6 Algorithme OTDD descendant optimisé

On présente ici l'algorithme OTDD: Optimized Top Down Discretization.

L'inconvénient de l'algorithme glouton GTDD est de tomber dans un minimum local. D'une part, l'algorithme risque de produire trop peu d'intervalles, d'autre part, les frontières n'étant pas remises en cause risquent d'être mal ajustées. On propose ici une post-optimisation de l'algorithme GTDD, analogue à l'algorithme OBUD en remplaçant uniquement l'étape intermédiaire Exhaustive Merge par une étape Exhaustive Split qui pousse la logique des Splits jusqu'aux intervalles élémentaires, et mémorise la meilleure discrétisation rencontrée.

Algorithme OTDD

On récapitule ci-dessous les étapes de l'algorithme OTDD:

- Optimisation gloutonne descendante GTDD
- Recherche d'une meilleure discrétisation par Exhaustive Split
- Post-optimisation de la discrétisation par Greedy Post-Optimization
  Recherche simultanée des types d'amélioration suivants
  - Split: bipartition d'un des intervalles
  - MergeSplit: changement de la frontière entre deux intervalles successifs
  - MergeMergeMergeSplit: redécoupage de trois intervalles consécutifs en deux intervalles

2.2.7 Commentaires

Il est important de noter que les optimisations présentées sont possibles parce que les critères utilisés évaluent la partition en intervalles de façon globale. De nombreuses méthodes de discrétisation (par exemple: MDLPC, ChiSplit, ChiMerge) utilisent un critère local s'appliquant uniquement à deux intervalles adjacents. Il est alors impossible d'optimiser globalement la partition en intervalles.

En pratique, l'algorithme optimal n'est intéressant que pour évaluer la qualité des heuristiques. Parmi les heuristiques proposées, nous retiendrons l'heuristique OBUD ascendante avec post-optimisation comme algorithme de référence pour les expérimentations. Ce choix est motivé principalement par le fait que la complexité en $O(n.\log(n))$ est garantie pour les heuristiques ascendantes, ce qui n'est pas le cas pour les heuristiques descendantes. De plus, le biais consistant à partir du bas parait plus adapté que le biais consistant à partir du haut si l'on cherche à détecter les structures de données informatives de petites tailles.

Tous les algorithmes précédents peuvent être améliorés en utilisant la propriété des codages MODL, qui ne peuvent être

optimum quand deux instances successives de même classe sont séparées dans deux intervalles. Il suffit alors dans un prétraitement de construire des intervalles initiaux contenant des séquences d'instances pures vis à vis des classes. On obtient alors un nombre M d'intervalles initiaux nettement inférieur en pratique au nombre d'instances. On peut également utiliser le nombre V ($V <= n$) de valeurs effectives de l'attribut descriptif pour optimiser la phase de tri, en utilisant un arbre binaire de recherche pour lequel chaque insertion (maintenant le tri) a un coût de $\log(V)$. Les algorithmes heuristiques ont alors une complexité en $O(n.\log(V) + M.\log(M))$. L'algorithme optimal a lui une complexité en $O(n.\log(V) + M^3)$.

Les algorithmes précédents peuvent également facilement être adaptés pour tenir compte des contraintes d'effectifs minimum par intervalles. Dans les algorithmes heuristiques ascendants où l'on évalue les Merges d'intervalles, il suffit de rendre prioritaire les Merges faisant intervenir au moins un intervalle ne respectant pas la contrainte d'effectif minimum. En ce qui concerne la recherche des Split, il suffit d'interdire les Split résultant en une bipartition dont un intervalle ne respecte pas la contrainte. Les autres types d'améliorations heuristiques MergeSplit et MergeMergeSplit sont des combinaisons de Merge et de Split, et peuvent être gérés de la même façon. Pour l'algorithme optimal, il suffit d'interdire les évaluations de partition dont un intervalle au moins ne respecte pas la contrainte d'effectif minimum.

Il est également facile de prendre en compte la contrainte de nombre maximum d'intervalles. Il suffit de forcer les Merges jusqu'à l'obtention du nombre max d'intervalles désirés, d'interdire les Split au delà du nombre max d'intervalles désirés, et d'arrêter l'algorithme optimal dès que l'on est arrivé à l'étape correspondant au nombre max d'intervalles désirés. De manière générale, la prise en compte des contraintes d'effectif minimum par intervalle ou de nombre maximum d'intervalles améliore la rapidité des algorithmes en diminuant l'espace de recherche des solutions.

On peut se poser la question du choix de l'algorithme (ascendant, descendant ou optimal), et de l'intérêt de proposer une solution optimale, très coûteuse à calculer et d'un intérêt discutable (l'optimisation d'un critère n'apporte pas nécessairement une amélioration significative du taux de bonne prédiction). De façon générale, on peut dire qu'une approche descendante aura tendance à privilégier les grandes structures, mais aura des difficultés à identifier les petites structures intéressantes. En revanche, les premières décisions se faisant sur un grand nombre d'instances seront fiables. A l'inverse, une approche ascendante est adaptée à l'identification des petites structures intéressantes. Elle sera néanmoins d'avantage sujette au sur-apprentissage en basant ses premières décisions sur de très petits nombres d'instances. Ces caractéristiques générales sont notamment illustrées dans le cas de critères d'évaluation locaux à deux intervalles adjacents, comme dans ChiSplit (descendant) et ChiMerge (ascendant).

L'optimisation d'un critère est un problème délicat. D'un côté, une solution optimisée permet d'améliorer la distance entre le modèle et les données apprises, ce qui tend à améliorer l'adéquation du modèle aux données. D'un autre côté, un algorithme optimal explore implicitement un espace de solution plus grand qu'une heuristique qui élague un grand nombre de solutions. La dimension de Vapnik-Chervonenkis implicitement associée à un algorithme optimal étant plus grande, le risque de sur-apprentissage est plus important. Si le critère optimisé ne gère pas correctement le problème de sur-apprentissage, les deux effets (meilleure adaptation aux données, et risque de sur-apprentissage) ont des impacts contradictoires sur la performance prédictive du modèle résultant. En ce qui concerne le codage MODL, le critère utilisé repose sur des bases théoriques fortes en ce qui concerne la gestion du sur-apprentissage, ce qui encourage la recherche d'algorithmes d'optimisation plus poussés.

3 Evaluation de l'algorithme MODL

Nous présentons dans cette partie une évaluation comparative de l'algorithme MODL sur les jeux d'essai de l'UCI et sur une base France Telecom, ainsi qu'une étude du comportement des méthodes de discrétisation en présence de données synthétiques. Nous proposons enfin une analyse spécifique de la méthode MODL, en étudiant les impacts de l'algorithme d'optimisation.

3.1 Expérimentations sur des bases UCI

3.1.1 Présentation

L'évaluation d'une méthode de discrétisation est un problème difficile, abordé de plusieurs façons dans la littérature. (Dougherty & al, 1995) comparent 4 méthodes de discrétisation en tant que prétraitement pour un prédicteur Bayésien Naïf et pour C4.5, et comparent le taux de bonne prédiction par cross-validation sur une quinzaine de jeux de données de la base UCI. (Liu et al., 2002) comparent 8 méthodes de discrétisation en tant que prétraitement pour C4.5 sur une douzaine de jeux de données de l'UCI, et mesurent le temps de discrétisation, le temps de construction de l'arbre de décision avant et après discrétisation, le nombre de nœuds de l'arbre et le taux de bonne prédiction. (Zighed & Rakotomalala, 2000) comparent 7 méthodes de discrétisation en mesurant le taux de bonne prédiction des prédicteurs univariés pour les 21 attributs du jeu de données waveform. Cette dernière approche a le mérite d'isoler l'apport de la discrétisation, indépendamment de son emploi dans un prédicteur. Les conclusions générales sont que les différences de performances prédictives sont considérées comme minimes, mais que la méthode MDLPC (Fayyad & Irani, 1992) est communément reconnue comme étant une des méthodes les plus performantes.

Dans ce document, nous proposons une méthodologie d'évaluation des méthodes de discrétisation qui permet d'effectuer des comparaisons fiables sur plusieurs critères, et nous montrons que les différences observées sont significatives. Nous avons évalué sept méthodes de discrétisation en utilisant quinze jeux de données standards extraits de la base UCI, comportant un

nombre total de 181 attributs continus à discrétiser. Toutes les mesures sont effectuées en utilisant la procédure de validation croisée stratifiée en 10 étapes. Toutes les méthodes testées ont été réimplémentées, afin d'effectuer les tests sans biais potentiel dû au choix des coupures dans la validation croisée. Afin de déterminer si les différences de résultats entre méthodes sont significatives, les tests de Student de comparaisons par paires au seuil de confiance de 5% ont été évalués.

L'objet des expérimentations étant l'évaluation des méthodes de discrétisation, nous n'avons pris en compte que les attributs continus des jeux de données utilisés. L'expérimentation repose sur l'utilisation de prédicteurs univariés pour chaque attribut discrétisé afin d'évaluer les performances intrinsèques des méthodes de discrétisation sur un très grand nombre d'attributs. Les critères testés sont :

- taux de bonne prédiction des prédicteurs univariés basés sur les discrétisations,
- nombre d'intervalles des discrétisations.

Nous détaillerons les procédures de tests utilisées lors de la présentation des résultats.

Les méthodes évaluées sont :

- MODL: la méthode présentée dans ce document, en utilisant l'a priori à trois étages et l'algorithme OBUD (ascendant avec post-optimisation),
- MDLPC : Minimum Description Length Principal Cut (Fayyad & Irani, 1992),
- Fusinter: méthode ascendante basée sur une mesure d'incertitude (Zighed et al., 1998),
- Khiops : méthode ascendante basée sur le test du Khi2 appliqué de façon globale sur tous les intervalles, avec garantie de résistance au bruit (Boullé, 2003),
- ChiMerge : méthode ascendante basée sur le test du Khi2 (Kerber, 1991),
- ChiSplit : méthode descendante basée sur le test du Khi2 (Bertier & Bouroche, 1981),
- EqualWidth : découpage en intervalle de largeur égale,
- EqualFrequency : découpage en intervalles de fréquence égale.

La méthode MDLPC est une méthode descendante qui découpe récursivement les intervalles en partant du domaine initial complet. Le critère MDLPC évalue la quantité d'information contenue à la fois dans le modèle (la coupure) et les exceptions au modèle, et accepte la coupure si cette quantité d'information globale diminue après la coupure. Cette méthode ne nécessite aucun paramétrage.

La méthode Fusinter est une méthode ascendante. Le critère utilisé est une mesure d'incertitude sensible aux effectifs des intervalles. Cette mesure est composée d'un terme d'entropie quadratique évaluant l'information des intervalles, régularisé par un terme inversement proportionnel aux effectifs des intervalles. L'évaluation étant globale à l'ensemble des intervalles, l'algorithme s'arrête quand la valeur du critère ne décroît plus. Le critère comporte deux paramètres, dont nous avons fixé la valeur en suivant les recommandations de l'auteur ($\lambda = 1$ et $\alpha = 0.975$). Pour cette méthode, on a utilisé le même algorithme optimisé que pour la méthode MODL, afin d'axer la comparaison sur le critère d'évaluation. Par la suite, on se référera à cette version optimisée sous le nom de FusinterO.

La méthode Khiops est une méthode ascendante. Le critère utilisé est le critère du Khi2 appliqué globalement à l'ensemble des intervalles. Le critère d'arrêt est contrôlé par un test statistique, qui permet de garantir que la discrétisation aboutira à un intervalle unique dans le cas d'un attribut descriptif indépendant de l'attribut cible.

La méthode ChiMerge est comme Khiops une méthode ascendante. Le critère utilisé est le critère du Khi2 appliqué localement à deux intervalles à fusionner. La fusion est acceptée si les deux intervalles sont suffisamment semblables pour un seuil de vraisemblance donné. Nous avons utilisé un seuil de 95% pour les expérimentations.

La méthode ChiSplit est une méthode descendante. Le critère utilisé est le critère du Khi2 appliqué localement aux deux sous-intervalles d'un intervalle à couper. La coupure est acceptée si les deux sous-intervalles sont suffisamment différents pour un seuil de vraisemblance donné. Nous avons utilisé un seuil de 95% pour les expérimentations.

Les méthodes EqualWidth et EqualFrequency sont des méthodes de discrétisation non supervisées, paramétrées par le nombre d'intervalles désiré. Nous avons adopté un découpage en 10 intervalles.

Seules les méthodes MODL et FusinterO bénéficient d'un algorithme optimisé. Les méthodes MDLPC, ChiMerge et ChiSplit ne peuvent être post-optimisées car elles utilisent un critère d'évaluation local à deux intervalles. L'heuristique de recherche de la méthode Khiops ne peut pas être transformée non plus, car le critère d'arrêt de Khiops est lié à la statistique de l'algorithme ascendant mis en œuvre.

Les jeux de données extraits de la base UCI comportent tous au moins un attribut continu et au moins quelques dizaines d'instances par modalité cible. Il s'agit exactement des même jeux de données que ceux utilisés dans (Boullé, 2003). Ces jeux de données sont présentés dans le tableau 1, dont la dernière colonne représente le taux de bonne prédiction de la modalité cible majoritaire.

Tableau 1 : Jeux de données

Dataset	Continuous Attributes	Nominal Attributes	Size	Class Values	Majority Accuracy
Adult	7	8	48842	2	76,07
Australian	6	8	690	2	55,51
Breast	10	0	699	2	65,52
Crx	6	9	690	2	55,51
German	24	0	1000	2	70,00
Heart	10	3	270	2	55,56
Hepatitis	6	13	155	2	79,35
Hypothyroid	7	18	3163	2	95,23
Ionosphere	34	0	351	2	64,10
Iris	4	0	150	3	33,33
Pima	8	0	768	2	65,10
SickEuthyroid	7	18	3163	2	90,74
Vehicle	18	0	846	4	25,77
Waveform	21	0	5000	3	33,92
Wine	13	0	178	3	39,89

3.1.2 Performance prédictive

Nous avons mesuré la performance prédictive des discrétisations pour chacun des attributs continus des 15 jeux de données précédemment utilisés. Cela correspond à l'utilisation de 181 jeux de données mono-attributs, ce qui d'une part apporte une fiabilité statistique aux résultats obtenus, et d'autre part permet d'évaluer la performance prédictive intrinsèque des méthodes de discrétisation, sans le biais dû à l'utilisation d'un prédicteur agrégé.

La table de résultats est trop volumineuse pour être reproduite dans ce document. Elle est résumée dans le tableau 2, qui indique pour chaque jeu de données la moyenne des taux de bonne prédiction par attribut et le nombre de gains et de pertes significatives dans les comparaisons avec MODL.

Tableau 2 : Moyenne des taux de bonne prédiction, nombre de gains et pertes par jeu de données, pour les prédicteurs élémentaires univariés

	MODL	MDLPC			FusinterO			Khiops			ChiMerge			ChiSplit			Eq. Width			Eq. Freq.		
			+	-		+	-		+	-		+	-		+	-		+	-		+	-
Adult	77.4	77.3	0	0	77.3	2	0	77.3	3	0	75.8	2	2	77.3	1	2	76.8	2	0	76.6	2	0
Australian	64.8	65.0	0	1	64.5	0	1	64.8	0	1	64.7	0	0	65.1	0	0	61.4	3	0	65.8	0	0
Breast	85.9	86.1	0	1	85.8	0	1	85.8	0	0	85.6	0	1	85.9	0	1	86.0	0	1	85.8	1	1
Crx	65.0	65.2	0	0	63.5	3	0	65.0	0	0	63.8	2	0	65.3	0	0	61.1	3	0	65.7	0	1
German	70.1	70.0	0	0	70.1	0	0	70.1	0	0	70.0	0	0	70.1	0	0	70.1	0	1	70.0	0	0
Heart	64.3	64.0	0	0	63.6	0	0	64.4	0	0	64.0	0	0	63.9	0	0	63.9	2	0	64.5	1	0
Hepatitis	79.0	79.3	0	0	79.0	1	1	79.6	0	0	78.1	3	1	79.3	0	0	79.5	0	0	80.0	0	0
Hypothyroid	96.0	96.1	0	0	96.1	0	0	96.1	1	1	96.0	1	0	96.1	0	1	95.4	3	0	95.2	3	0
Ionosphere	80.0	77.6	11	0	79.4	1	2	79.7	2	2	76.0	21	0	79.6	3	1	73.8	21	1	74.9	21	0
Iris	79.5	75.5	2	0	80.5	0	0	78.8	0	0	77.0	0	0	78.8	0	0	76.3	1	0	76.2	0	0
Pima	66.2	66.1	0	0	66.5	1	2	66.3	0	0	66.1	0	1	66.5	0	0	66.8	2	1	66.3	0	1
SickEuthyroid	91.3	91.3	0	0	91.3	0	0	91.3	0	0	91.3	1	0	91.3	0	0	90.7	2	0	91.0	1	0
Vehicle	41.3	40.5	4	1	41.6	3	2	41.5	2	3	41.5	2	2	42.0	0	2	40.9	3	2	40.3	5	1
Waveform	49.3	49.3	0	2	49.3	2	1	49.3	0	0	48.8	5	0	49.2	3	0	49.2	3	1	49.5	1	2
Wine	60.4	60.1	1	0	62.6	0	1	60.0	1	0	59.6	2	0	60.1	2	0	60.8	1	1	60.6	0	1
Synthesis	68.6	68.0	18	5	68.6	13	11	68.6	9	7	67.5	39	7	68.6	9	7	67.1	46	8	67.5	35	7
Average rank	2.8	3.4			3.2			2.9			4.5			3.4			4.1			3.8		

L'analyse des résultats montre que les méthodes supervisées (sauf ChiMerge) obtiennent des résultats nettement meilleurs que les méthodes non supervisées. La méthode ChiMerge est légèrement meilleure que la méthode EqualWidth, mais moins performante que la méthode EqualFrequency. La méthode MDLPC est nettement meilleure que la méthode EqualFrequency. Les méthodes MODL, FusinterO, Khiops et ChiSplit se détachent en tête, en surpassant la méthode MDLPC.

Afin de mieux comprendre l'importance des différences entre les méthodes, on va s'attacher à comparer en détail les résultats des méthodes MODL et MDLPC. L'écart entre les valeurs moyennes synthétiques (68,6 pour MODL et 68,0 pour MDLPC) peut paraître faible (0,6%), mais il faut le comparer à la valeur moyenne du prédicteur majoritaire qui est de 57,4. L'amélioration relative par rapport au prédicteur majoritaire est alors supérieure à 5%, ce qui n'est pas négligeable. D'autre

part, cet écart absolu de 0,6% est une moyenne sur une grande variété d'attributs, et il est intéressant d'examiner en détail comment se répartissent les différences de taux de bonnes prédictions sur l'ensemble des 181 attributs ayant servi à l'évaluation. La figure 1 représente les fonctions de répartition des différences de performance entre MODL et toutes les autres méthodes.

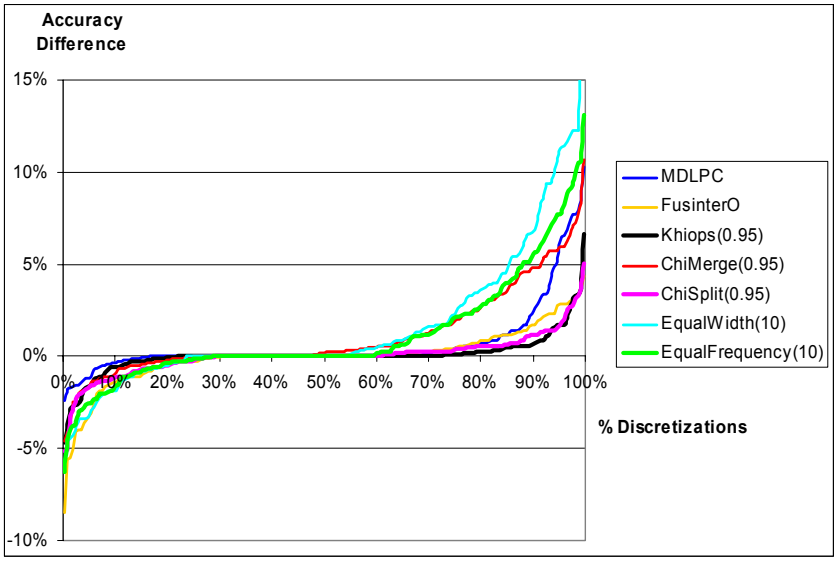


Figure 1 : Répartition des différences de taux de bonne prédiction entre MODL et les autres méthodes

La partie gauche de la figure représente les cas où MODL est moins performant que les autres méthodes, et la partie droite les cas où MODL est meilleur. Dans 40% des cas environ (entre les abscisses 20% et 60%), toutes les méthodes sont équivalentes. Cela correspond à des attributs ayant une seule valeur ou très peu de valeurs, qui ne sont pas utilisables en fait pour comparer les discrétisations. Si l'on compare MODL à MDLPC, MODL est de 0 à 2% moins bon que MDLPC dans environ 10% des cas de discrétisation, mais est de 3 à 10% meilleur dans environ 10% des cas. On voit alors clairement que la différence moyenne de 0,6% entre MODL et MDLPC est significative, et qu'elle se traduit en fait par des différences beaucoup plus importantes.

On peut dès lors considérer que les différences observées entre les méthodes sont significatives, et que la valeur moyenne synthétique (sur 181 attributs discrétisés chacun 10 fois) est un bon indicateur global permettant de comparer les performances des méthodes.

3.1.3 Taille des discrétisations

La taille des discrétisations est simplement le nombre d'intervalles obtenus sur l'ensemble d'apprentissage par les différentes méthodes de discrétisation. Le tableau 3 résume les résultats obtenus lors de l'évaluation de la taille des discrétisations sur l'ensemble des attributs des jeux de données.

Cette mesure est sans intérêt pour les méthodes non supervisées, dont on a paramétré le nombre d'intervalles. On remarque que ces méthodes aboutissent à un peu moins que les 10 intervalles demandés, à cause des attributs pour lesquels il n'y a pas assez de valeurs différentes pour trouver un découpage effectif en 10 intervalles. Les méthodes MODL, Khiops et MDLPC produisent les discrétisations les plus petites, sont faiblement discernables pour ce critère. Le nombre important de différences significatives observées est un peu artificiel, et provient du fait que la valeur mesurée est le nombre d'intervalles (qui ne prend que des valeurs entières). La méthode FusinterO produit envrion 50% d'intervalles en plus, et la méthode ChiSplit plus de deux fois plus d'intervalles. La méthode ChiMerge génère des nombres d'intervalles extrêmement importants. Il est à noter que pour la base Adult, ce nombre d'intervalles est considérable (plus de 1000 intervalles en moyenne pour 50000 individus). On remarque que cette base nécessite apparemment un nombre d'intervalles assez important, y compris pour les méthodes les plus performantes. C'est la seule base pour laquelle MODL, Khiops ou MDLPC produisent plus d'intervalles que EqualFrequency (avec un paramètre de 10 intervalles demandés). On observe également un comportement particulier pour la base Ionosphere, pour laquelle les différences tant en performance prédictive qu'en nombre d'intervalles sont très importantes entre la méthode MODL et les autres méthodes. Un examen détaillé des discrétisations produites par MODL révèle des intervalles d'effectifs d'équilibrés, et un pattern fréquent consistant en un intervalle "parlant" isolé au milieu de deux intervalles plus "standard". Ce type de pattern est bien isolé par les approches ascendantes comme MODL, FusinterO ou Khiops, mais est difficile à identifier pour MDLPC qui utilise une approche descendante: deux coupures successives sont nécessaires pour identifier l'intervalle intéressant, la première étant peu significative. L'approche ChiSplit, bien que descendante, a tendance à produire de nombreux intervalles, ce qui lui permet d'identifier néanmoins les patterns intéressant au prix de nombreux intervalles inutiles.

Tableau 3 : Moyenne des nombres d'intervalles, nombre de gains et pertes par jeu de données, pour les prédicteurs élémentaires univariés

	MODL	MDLPC			FusinterO			Khiops			ChiMerge			ChiSplit			Eq. Width			Eq. Freq.		
			+	-		+	-		+	-		+	-		+	-		+	-		+	-
Adult	9.4	8.8	1	1	8.4	2	4	8.5	3	2	1264	0	7	28.2	0	6	9.4	2	5	6.6	3	3
Australian	2.2	2.0	1	0	6.6	0	6	2.1	0	0	16.1	0	6	5.2	0	6	8.1	0	6	8.8	0	6
Breast	2.7	2.9	0	2	4.3	0	6	2.6	3	0	11.6	0	9	4.9	0	9	9.2	0	10	5.9	0	10
Crx	2.2	2.1	1	0	6.6	0	6	2.1	0	0	15.8	0	6	5.1	0	6	8.2	0	6	8.7	0	6
German	1.3	1.2	1	0	2.2	0	16	1.3	0	2	2.4	0	13	2.0	0	13	3.8	0	24	3.4	0	22
Heart	1.7	1.7	0	0	3.6	0	6	1.7	0	0	5.0	0	5	2.5	0	5	5.9	0	8	6.1	0	8
Hepatitis	1.6	1.4	1	0	3.5	0	5	1.7	0	0	6.4	0	6	2.8	0	6	8.6	0	6	9.2	0	6
Hypothyroid	2.8	3.1	0	2	2.9	1	2	3.6	0	5	15.3	0	7	6.0	0	7	9.6	0	7	8.3	0	7
Ionosphere	4.7	3.9	14	1	6.2	0	27	4.3	11	0	30.0	0	32	8.0	0	30	9.4	0	32	8.9	0	32
Iris	3.0	2.8	2	0	3.2	0	1	2.8	1	0	3.7	0	3	3.6	0	2	9.7	0	4	9.5	0	4
Pima	2.3	2.1	2	0	5.1	0	8	2.3	0	0	13.2	0	8	5.0	0	8	9.5	0	8	9.3	0	8
SickEuthyroid	3.0	3.0	0	0	3.5	0	3	3.4	0	2	17.2	0	6	5.8	0	6	9.6	0	7	8.3	0	6
Vehicle	4.4	3.9	8	0	5.9	1	16	4.0	8	3	9.7	0	18	8.0	0	18	9.6	0	18	9.6	0	18
Waveform	4.8	4.9	1	5	7.8	0	20	4.5	10	2	49.0	0	21	13.6	0	21	10.0	0	21	10.0	0	21
Wine	2.8	2.8	2	2	4.0	0	11	2.6	5	0	6.7	0	13	4.8	0	12	9.8	0	13	10.0	0	13
Synthesis	3.4	3.2	34	13	5.1	4	137	3.3	41	16	66.1	0	160	7.2	0	155	8.5	2	175	8.0	3	170
Average rank	1.8	1.5			3.9			1.7			6.3			4.9			6.3			6.0		

3.1.4 Analyse multi-critères des résultats

La comparaison des méthodes à l'issue des expérimentations permet de comparer et de classer les méthodes sur chacun des critères étudiés. Il est intéressant de procéder à une analyse multi-critères des résultats, afin de mieux comprendre les liens entre ces critères, notamment pour la performance prédictive, la robustesse et la taille des discrétisations.

Rappelons dans un premier temps quelques notions de l'analyse multi-critères. On dit qu'une solution *domine* (ou est *non inférieure* à) une autre solution si elle est meilleure sur tous les critères. Une solution ne peut être dominée si toute amélioration sur un des critères entraîne une détérioration sur un autre critère. Une telle solution est un *optimum de Pareto*. La *surface de Pareto* (*courbe de Pareto* pour 2 critères) est l'ensemble de tous les optima de Pareto.

On a repris ici les résultats des discrétisations sur les attributs élémentaires des jeux de données et utilisé la valeur moyenne (sur les 1810 discrétisations effectuées) comme indicateur de performance pour le taux de bonne prédiction et le nombre d'intervalles des méthodes de discrétisation. La figure 2 présente l'ensemble des résultats sur le plan des critères de taux de bonne prédiction et de nombre d'intervalles.

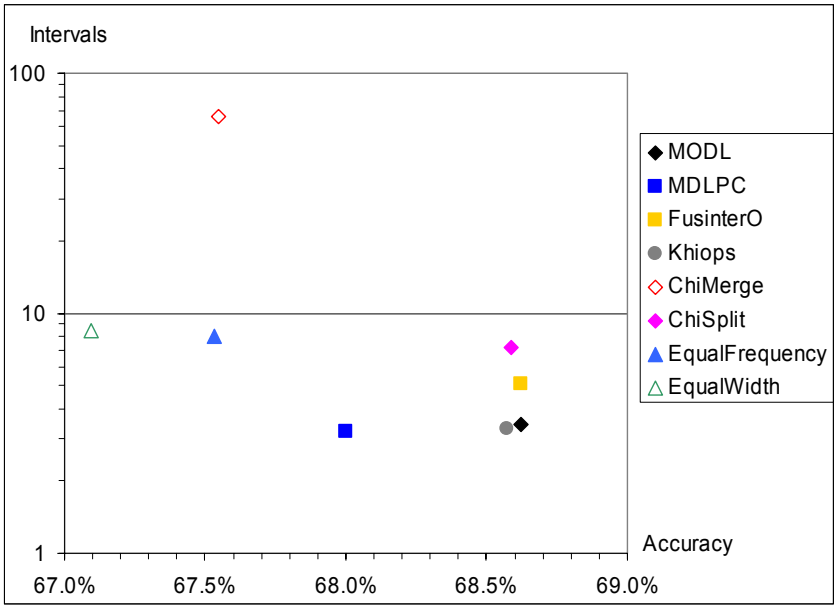


Figure 2 : Evaluation des méthodes de discrétisation pour les critères de taux de bonne prédiction et de taille des discrétisations

Les méthodes non supervisées EqualWidth et EqualFrequency sont très largement dominées par l'ensemble des autres

méthodes (sauf ChiMerge) sur les deux critères étudiés.

La méthode ChiMerge est la moins performante des méthodes supervisées testées, et obtient notamment des nombres d'intervalles très élevés. Ainsi, pour son paramétrage par défaut de 0,95, elle obtient en moyenne 66 intervalles.

La méthode MDLPC obtient de très bons résultats en termes de nombre d'intervalles, mais est dépassée en taux de bonne prédiction par les méthodes MODL, FusinterO, Khiops et ChiSplit.

ChiSplit obtient de très bonnes performances, mais au prix d'un nombre d'intervalles très élevé par rapport à MDLPC, Khiops ou MODL. C'est également le cas pour FusinterO, mais avec nombre d'intervalles intermédiaire.

Les méthodes MODL et Khiops obtiennent les meilleurs résultats sur l'ensemble des deux critères, et sont très faiblement discernables l'une de l'autre. La méthode MODL produit légèrement plus d'intervalles que la méthode Khiops pour un taux de bonne prédiction légèrement meilleur. En fait, les deux méthodes sont toutes deux basées sur des principes statistiques, mais alors que Khiops se base sur le test du Khi2 dont la validité est asymptotique, MODL produit des discrétisation optimales sans contraintes d'effectif minimum par intervalle ou d'effectif minimal par cellule d'un tableau de contingence. Les jeux d'essai de l'UCI sont en fait assez faiblement sélectifs comme l'avait noté (Holte, 1993), ce qui permet aux méthodes alternatives d'obtenir souvent des résultats très proches des résultats optimaux. On peut également noter qu'à la différence de Khiops, la méthode MODL ne nécessite aucun paramétrage, et qu'elle n'a pas le problème asymptotique de Khiops qui impose un effectif minimum par intervalle de façon à pouvoir utiliser de façon fiable le critère du Khi2.

En conclusion, les méthodes MODL et Khiops réalisent les meilleurs compromis entre nombres d'intervalles et taux de bonne prédiction. Il est à noter que la méthode MODL n'a besoin d'aucun paramètre.

3.1.5 Exemple illustratif: la base Adult

L'expérimentation précédente montre que les méthodes MODL et Khiops se détachent clairement des autres méthodes pour leur performances en taux de bonne précision et taille des discrétisations. Les deux méthodes sont comparables dans le sens où elles procèdent à une évaluation globale de tous les intervalles de la discrétisation, et utilisent un algorithme glouton ascendant. Alors que Khiops minimise la probabilité d'indépendance entre la discrétisation et l'attribut à prédire, MODL recherche le modèle de discrétisation de probabilité maximale connaissant les données. Les deux approches sont duales, ce qui explique les performances similaires. En revanche, alors que MODL évaluent les probabilités de façon exacte, Khiops s'appuie sur le test du Khi2, qui a une validité asymptotique. Khiops doit donc trouver un compromis entre des effectifs minimum par intervalle suffisant pour garantir la validité du test du Khi2, mais pas trop important pour permettre la détection de niches intéressantes. Les performances de Khiops trouvent alors leurs limites dans les cas asymptotiques, c'est-à-dire pour les échantillons de petite taille, les cas de classe cible de très faible proportion, ou les cas d'intervalles pertinents de petite taille. MODL ne souffre pas de ces problèmes, et ses performances restent optimales sans limites asymptotiques. Afin d'illustrer ces comportements, nous allons comparer les performances des méthodes MODL et Khiops sur les attributs continus de la base Adult.

Le tableau 4 compare les tailles des discrétisations des attributs continus de la base Adult pour les méthodes MODL et Khiops. La plupart des attributs sont discrétisés de manière quasiment identique avec des discrétisations de même taille. Par contre, MODL produit 20 intervalles pour l'attribut capital_gain contre 11 seulement pour Khiops. A l'inverse, MODL ne produit qu'un seul intervalle pour l'attribut fnlwgt contre 7 pour Khiops. Nous allons utiliser l'ensemble de test (les 30% de la base Adult non utilisés pour apprendre les discrétisations) afin d'évaluer les deux méthodes.

Tableau 4 : Nombres d'intervalles des discrétisations des attributs continus de la base Adult, après apprentissage sur 70% de la base

Attribut	Nombre d'intervalles MODL	Nombre d'intervalles Khiops
capital_gain	20	11
age	7	7
education_num	6	6
hours_per_week	6	6
capital_loss	15	15
fnlwgt	1	7
Label	1	1

Le tableau 5 présente la table de contingence de la discrétisation Khiops de l'attribut fnlwgt, en apprentissage et en test. On voit clairement que les intervalles identifiés en apprentissage traduisent un phénomène de sur-apprentissage des données. En test, les proportions de classes cibles estimées dans les intervalles de petite taille donnent raison à la discrétisation MODL, qui n'avait produit qu'un seul intervalle.

Tableau 5 : Discrétisation Khiops de l'attribut fnlgwt de la base Adult

	Table de contingence en apprentissage			Table de contingence en test		
Intervalle	%less	%more	Fréquence	%less	%more	Fréquence
]-inf; 111479.5[	77	23	7585	77	23	3281
[111479.5; 111578[	38	62	39	83	17	12
[111578; 123001.5[	76	24	1677	75	25	709
[123001.5; 123081.5[	27	73	22	50	50	8
[123081.5; 207538.5[	75	25	12936	75	25	5677
[207538.5; 207672.5[	29	71	21	75	25	16
[207672.5; inf[	77	23	11819	77	23	5040

Le tableau 6 présente la table de contingence de la discrétisation MODL de l'attribut capital_gain, en apprentissage et en test. En apprentissage, on remarque que MODL a produit de nombreux intervalles de très petite taille (moins d'une dizaine d'instance pour un échantillon d'apprentissage d'environ 33000 instances). Il est notable de constater que les proportions de classes cibles estimées en apprentissage sont également observées en test, ce qui justifie le grand nombre d'intervalles produits par la méthode MODL.

Tableau 6 : Discrétisation MODL de l'attribut capital_gain de la base Adult

	Table de contingence en apprentissage			Table de contingence en test		
Intervalle	%less	%more	Fréquence	%less	%more	Fréquence
]-inf; 57[	79	21	31283	80	20	13524
[57; 3048[	100	0	495	100	0	206
[3048; 3120[	4	96	103	12	88	49
[3120; 4243.5[	100	0	302	100	0	133
[4243.5; 4401[	15	85	72	14	86	36
[4401; 4668.5[	100	0	87	100	0	23
[4668.5; 4826[	0	100	29	0	100	10
[4826; 4932.5[	100	0	19	100	0	10
[4932.5; 4973.5[	0	100	7	0	100	3
[4973.5; 5119[	100	0	84	100	0	35
[5119; 5316.5[	0	100	95	0	100	51
[5316.5; 6667.5[	62	38	52	42	58	26
[6667.5; 7055.5[	100	0	33	100	0	20
[7055.5; 7436.5[	0	100	254	0	100	126
[7436.5; 7565.5[	100	0	6	100	0	1
[7565.5; 10543[	1	99	423	0	100	175
[10543; 10585.5[	100	0	4	100	0	4
[10585.5; 30961.5[	0	100	570	0	100	239
[30961.5; 70654.5[	100	0	7	100	0	2
[70654.5; inf[	0	100	174	0	100	70

En conclusion, les deux discrétisations précédentes illustrent les limites de Khiops, qui dans certains cas auraient avantage à augmenter l'effectif minimum par intervalle afin de fiabiliser les discrétisations, et dans d'autre cas aurait intérêt à diminuer cette contrainte d'effectif minimum par intervalles, afin d'identifier les petits intervalles pertinents. La méthode MODL ne souffre pas de ce compromis et obtient des discrétisations quasi-optimales dans toutes les situations.

3.2 Expérimentation sur une base de France Telecom

3.2.1 Présentation

L'expérimentation dont est extrait le jeu d'essai utilisé porte sur une base d'environ 8000 clients, qu'une enquête individuelle a permis de catégoriser en quatre segments marketing: les célibataires, les jeunes foyers, les foyers comportant au moins un adolescent et les séniors. L'enquête étant très coûteuse, le but de l'étude consiste à prédire les segments marketing à partir de données issues du système d'information France Telecom, de façon à pouvoir généraliser la segmentation marketing à l'ensemble des clients France Telecom sans payer le prix de l'enquête. Les données descriptives au nombre de 206 sont principalement des données agrégées de trafic (moyennes par tranche horaire et jour de la semaine), et des données concernant le parc de produits et services de télécommunication des clients.

L'évaluation des méthodes de discrétisation est basée sur le même protocole que pour les bases UCI. Néanmoins, les segments à prédire étant déséquilibrés et le problème étant peu séparable, les prédicteurs univariés obtiennent très rarement des performances supérieures au simple prédicteur majoritaire. Afin de pouvoir comparer simplement les performances entre les méthodes, la base a été réduite à 3516 instances, avec des segments représentant chacun exactement 25% des instances.

3.2.2 Résultats

On a ici utilisé quatre critères afin de comparer les méthodes:
- Test Accuracy: taux de bonne prédiction en test
- Ratio Accuracy: rapport de taux de bonne prédiction en test et en apprentissage; ce critère mesure la robustesse de la méthode
- Mono-intervals: pourcentage des attributs pour lesquels la discrétisation n'a produit qu'un seul intervalle terminal
- Interval Number: nombre d'intervalles produits

Le tableau 7 présente l'ensemble des résultats obtenus.

Tableau 7 : Moyenne, nombre de gains et pertes pour quatre critères d'évaluations, pour les prédicteurs élémentaires univariés

	MODL	MDLPC			FusinterO			Khiops			ChiMerge			ChiSplit			Eq. Width			Eq. Freq.		
			+	-		+	-		+	-		+	-		+	-		+	-		+	-
Test Accuracy	28,8	28,7	17	1	28,8	15	29	28,9	3	8	28,3	45	27	28,9	14	30	26,3	131	22	29,0	13	48
Ratio Accuracy	98,7	98,8	3	4	96,7	50	2	98,7	2	2	87,1	138	2	96,2	88	2	98,9	19	4	97,7	31	5
Mono-intervals	30,1	33,8	0	9	21.5	21	0	29,1	5	2	10,6	45	0	12,9	39	0	1,1	62	0	1,1	62	0
Interval Number	2,0	1,9	41	0	4.0	0143		2,1	4	15	43,0	0178		6,5	0168		7,0	0201		6,0	0191	

Dans un premier groupe de méthodes, on trouve les méthodes MODL, MDLPC et Khiops, qui ont pour caractéristiques communes d'obtenir une bonne performance prédictive moyenne, une très grande robustesse, est un très faible nombre d'intervalles. La méthode FusinterO obtient les mêmes performances prédictives que celle du premier groupe au prix d'un nombre d'intervalle deux fois plus élevé. Dans un deuxième groupe, on trouve les méthodes ChiSplit et EqualFrequency, qui ont une performance prédictive moyenne légèrement meilleure, au prix dune robustesse inférieure et d'un nombre d'intervalles trois fois plus important. On trouve ensuite la méthode ChiMerge caractérisée principalement par un sur-apprentissage très important, qui se manifeste par une très mauvaise robustesse et un très grand nombre d'intervalles. Enfin, la méthode EqualWidth obtient de très mauvaises performances en prédiction. Cela est dû principalement à la nature des données, principalement les agrégats de trafic, dont la loi de distribution entre les valeurs extrêmes est proche d'une exponentielle décroissante.

Une analyse détaillée du tableau de résultats montre que même la méthode ChiMerge obtient des résultats significativement meilleurs que la méthode MODL dans 27 cas, ce qui associé aux meilleures performances prédictives moyennes de ChiSplit et EqualFrequency paraît surprenant. Pourquoi une méthode produisant des discrétisations optimales (au sens de Bayes) est-elle surpassée en moyenne par la méthode non supervisée EqualFrequency en ce qui concerne la performance prédictive? Afin de mieux comprendre ce phénomène, nous avons collecté pour chacune des méthodes l'ensemble des 206 performances prédictives par attributs (moyenne sur 10 discrétisation pour chaque attribut), et tracé les courbes de résultats en apprentissage sur la figure 3 et en test sur la figure 4. En abscisse, on a le pourcentage de discrétisation (sur les 206 résultats), et en ordonnée, on a le taux de bonne prédiction, les résultats ayant été triés par performance croissante, indépendamment pour chaque méthode. On retrouve les mêmes groupes de méthodes qu'à l'issue de l'analyse du tableau synthétique de résultats, mais les figures présentées permettent d'affiner l'analyse.

Les trois méthodes performantes et robustes MODL, MDLPC et Khiops sont très fiables, et leur résultat en test est très proche du résultat en apprentissage. En ce sens, ces méthodes produisent de bons modèles de discrétisation, les plus appropriés pour expliquer les donnés observées. Cette robustesse se paye par une moindre performance prédictive dans les cas où la performance prédictive potentielle est très faible (sur la partie gauche des figures). En effet, dans ces cas, les méthodes robustes ne constituent qu'un seul intervalle, en estimant que les micro-régularités détectées en apprentissage sont le fait du hasard. On remarque d'ailleurs dans ce cas que la performance des méthodes peu robustes (de l'ordre de 26%) est environ 1% au dessus de la performance minimale du prédicteur majoritaire (25%), ce qui est de l'ordre de grandeur de la variance de la loi binomiale pour cette taille d'échantillon.

La méthode FusinterO est clairement sujette a sur-apprentissage, mais sa performance prédictive moyenne reste identique à celle des méthodes robuste en raison des meilleurs prédiction dans les zones des très faibles taux de bonne prédiction (à gauche de la figure).

Les deux méthodes très performantes et peu robustes ChiSplit et EqualFrequency montrent un net comportement de sur-apprentissage. La décroissance de performance entre apprentissage et test est nettement plus importante que pour les trois méthodes robustes. Néanmoins, les risques pris en apprentissage restent limités et s'avèrent globalement payant en permettant de dépasser en moyenne la performance prédictive des méthodes robuste. On assiste ici a un biais de l'évaluation des méthodes par le taux de bonne prédiction. Dans le cas d'un attribut indépendant de l'attribut à prédire, le meilleur modèle de discrétisation est clairement le modèle mono-intervalle. Néanmoins, dans le cas de classes cibles équilibrées, toute

discrétisation multi-intervalles aura une prédiction aléatoire pour chaque intervalle (due aux régularités apparentes en apprentissage), mais sa performance en test ne sera pas pénalisée par rapport à celle d'une discrétisation mono-intervalle. Si l'on s'éloigne très légèrement de l'hypothèse d'indépendance des attributs, la stratégie consistant à produire trop d'intervalle s'avère en moyenne payante, par rapport à des méthodes plus conservatrices qui ne vont se "risquer" au multi-intervalles que si les régularités observées en apprentissage paraissent statistiquement significatives.

La méthode ChiMerge a un comportement de sur-apprentissage excessif. Alors que le sur-apprentissage "contrôlé" de FusinterO, ChiSplit et EqualFrequency permet d'obtenir de bons résultats en test, la dégradation est telle pour ChiMerge que la performance prédictive est nettement moins bonne que pour la plupart des autres méthodes. Dans les cas où il n'y a presque rien à apprendre, ChiMerge parvient néanmoins à dépasser en test les méthodes robustes, qui jusqu'à un certain seuil décident qu'il n'y a strictement rien à apprendre. Ce cas est illustré dans les 30% de discrétisations sur la gauche des figures, pour lesquelles les méthodes robustes produisent des discrétisations mono-intervalles.

La méthode EqualFrequency sous-apprend, en obtenant de très mauvais résultats tant en apprentissage qu'en test.

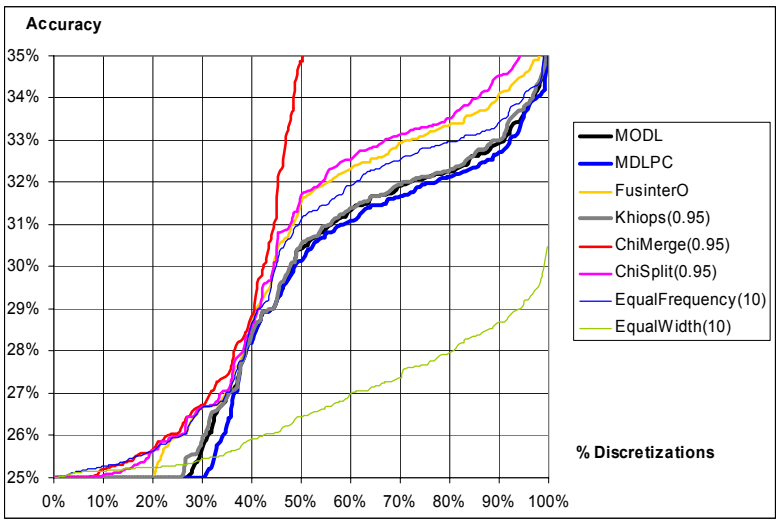


Figure 3 : Fonction de répartition sur 206 attributs des performances prédictives mesurées en apprentissage pour chaque méthode de discrétisation

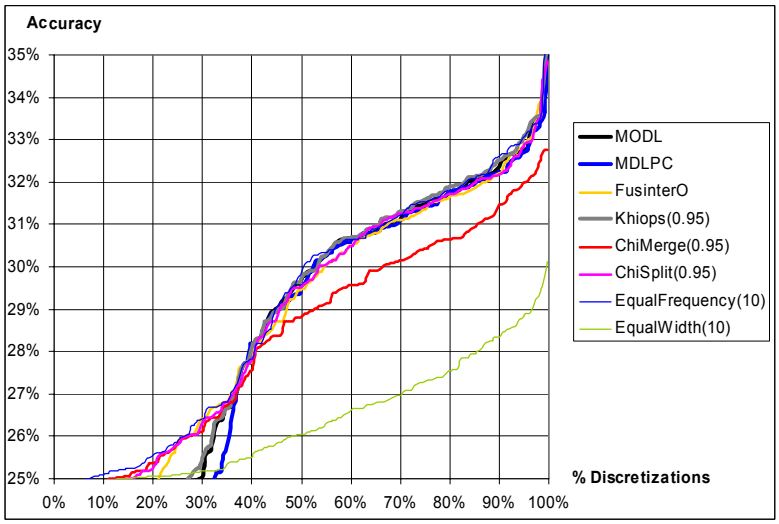


Figure 4 : Fonction de répartition sur 206 attributs des performances prédictives mesurées en test pour chaque méthode de discrétisation

En résumé, les méthodes robustes MODL, MDLPC et Khiops produisent des discrétisations très fiables, qui obtiennent de très bons résultats en test et sont les plus pertinentes pour expliquer les données. Les méthodes ChiSplit et EqualFrequency sont légèrement plus performantes en moyenne, mais l'apport est dû principalement à la contribution des attributs les moins intéressants du jeu de données, et les discrétisations sont peu pertinentes pour expliquer les données. La méthode FusinterO à un comportement intermédiaire entre les deux groupes de méthodes. Les deux dernières méthodes ChiMerge et EqualWidth obtiennent de mauvais résultats.

3.3 Expérimentation sur des jeux d'essai synthétiques

Dans cette section, on étudie le comportement global des méthodes de discrétisation face à un attribut complètement bruité, puis une étude plus complète portant sur une famille de jeux synthétique est présentée. Enfin, on analyse le seuil de détection d'un intervalle pur isolé au milieu d'un attribut bruité

3.3.1 Discrétisation de bruit

Dans cette expérimentation, on a évalué le comportement des différentes méthodes en présence d'un attribut continu indépendant de l'attribut cible. Pour cela, on a utilisé des jeux de données artificiels de taille comprise entre 100 et 100000, comportant un attribut continu descriptif ayant ses valeurs uniformément distribuées sur [0; 1], et un attribut cible à deux modalités équiréparties, choisies de façon aléatoire. Pour chaque méthode, on a mesuré le nombre moyen d'intervalles obtenus. L'expérimentation a été effectuée 10000 fois pour obtenir un résultat fiable. L'expérimentation ne concerne que les méthodes de discrétisation supervisées (pour lequel on ne paramètre pas le nombre d'intervalles à obtenir). La figure 5 présente les résultats de l'expérimentation obtenus par les différentes méthodes.

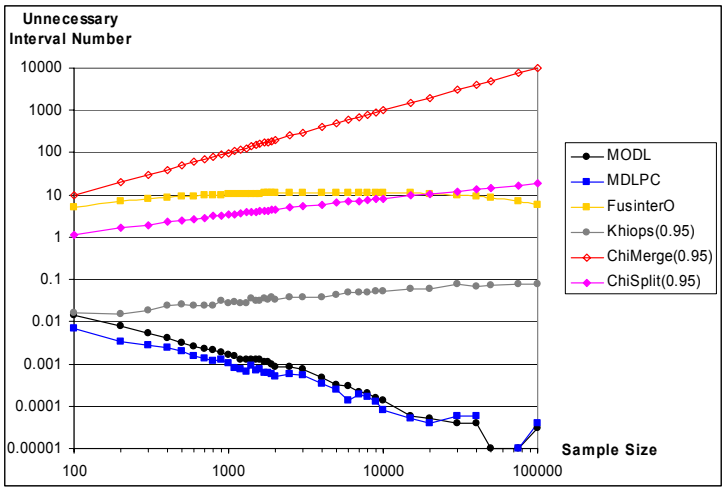


Figure 5 : Nombre moyen d'intervalles surnuméraires lors de la discrétisation d'un attribut indépendant de l'attribut cible

Les méthodes ChiMerge et ChiSplit produisent systématiquement plus de un intervalle, et le nombre d'intervalles produit croît de façon quasiment linéaire avec la taille de l'échantillon. La méthode FusinterO produit entre 5 et 10 intervalles avec un maximum pour les échantillons de taille 5000 environ. La méthode Khiops produit un seul intervalle dans environ 95% des cas quelle que soit la taille de l'échantillon, ce qui correspond au "cahier des charges" de son algorithme (le nombre moyen d'intervalles surnuméraires est compris entre 1.02 et 1.08 en raison des discrétisations multi-intervalles qui comportent la plupart 3 intervalles (et non 2)). Les méthodes MODL et MDLPC sont très résistantes au bruit et aboutissent presque systématiquement à un seul intervalle. Des expérimentations plus poussées réalisées 100000 fois ont montré que le pourcentage de discrétisations multi-intervalles est de l'ordre de 1% pour les échantillons de taille 100, de 0.1% pour les échantillons de taille 1000 et que cette proportion décroît environ en 1/n avec la taille de l'échantillon. Il est notable que les rares cas de discrétisations multi-intervalles, MDLPC aboutit à 2 intervalles (un seul Split a été effectué), alors que MODL produit également des discrétisation comportant 3 intervalles (un pattern émergeant du « bruit » a été isolé au milieu de deux intervalles standard). Ce comportement correspond à la différence d'approche descendante ou ascendante utilisée par les deux méthodes.

On peut remarquer le côté paradoxal de cette expérimentation. Il paraît évident que les méthodes de discrétisation supervisées sont d'autant meilleures qu'elles génèrent peu d'intervalles en présence d'un attribut indépendant de l'attribut cible. Pourtant, une méthode produisant systématiquement un seul intervalle n'aurait aucun intérêt pour la prédiction. Inversement, une méthode produisant de nombreux intervalles n'est pas pénalisée quand on mesure son taux de bonne prédiction. Sur chaque intervalle, la prédiction est en fait très proche de celle du prédicteur majoritaire (optimale dans le cas de l'expérimentation), et les petites fluctuations de bonne prédiction entre les intervalles ont tendance à se compenser globalement. Ce critère est alors très intéressant pour éclairer la stratégie d'apprentissage des différentes méthodes. Les comportements observés vont de MODL ou MDPLC qui produisent presque toujours un seul intervalle à ChiMerge et ChiSplit qui en produisent d'autant plus que la taille du jeu de données augmente, en passant par Khiops qui contrôle la probabilité de produire un seul intervalle et FusinterO qui produit une dizaine d'intervalles selon une dépendance atypique vis-à-vis de la taille de l'échantillon.

3.3.2 Discrétisation d'une fonction "créneaux"

Cette expérimentation utilise une famille de jeux d'essai plus complexe, basé sur une fonction créneaux où l'attribut continu à discrétiser est distribué uniformément sur l'intervalle [0; 1], et la classe à prédire comporte deux valeurs équidistribuées "+"et

"-" variant alternativement sur l'axe des x suivant la fonction $Class = Signe(Sinus(100\pi x))$. En l'absence de bruit, la discrétisation optimale consiste en 100 intervalles aux bornes équiréparties sur [0; 1]. Pour des échantillons de petite taille, où le nombre d'instances est de l'ordre du nombre d'intervalles optimal, la distribution des classes sur le domaine de valeur à discrétiser s'apparente à du bruit, et les méthodes de discrétisation ne devraient produire qu'un seul intervalle (ce qui correspond à une erreur en test de 50%). En accroissant la taille des échantillons, chaque intervalle optimal se peuple de suffisamment d'instances pour être identifié par les méthodes de discrétisation (entraînant alors asymptotiquement une erreur en test de 0%). L'expérimentation permet d'observer le moment de la transition à partir duquel tous les intervalles sont correctement identifiés.

Le protocole de l'expérimentation consiste à générer des jeux d'essai aléatoires pour la fonction créneaux, à discrétiser l'attribut continu et à évaluer le nombre d'intervalles des discrétisations. L'expérimentation est effectuée pour différentes tailles d'échantillon comprises entre 100 et 100000, et répétée 1000 fois pour chacune des méthodes de discrétisation étudiées.

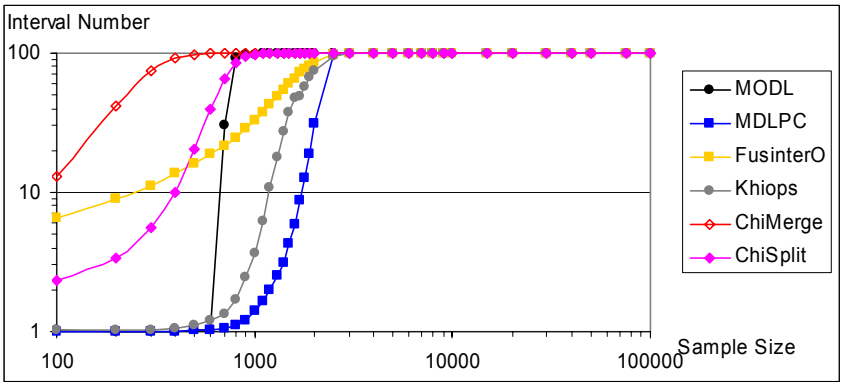


Figure 6 : Nombre d'intervalles en fonction de la taille de l'échantillon pour la discrétisation de la fonction créneaux $Class = Signe(Sinus(100\pi x))$ sur le domaine [0; 1], dans le cas non bruité

La figure 6 montre le nombre d'intervalles en fonction de la taille de l'échantillon dans le cas non bruité. La méthode ChiMerge qui produit beaucoup d'intervalles (cf. expérimentations sur données UCI) en tire parti en l'absence de bruit, et identifie très rapidement les intervalles pertinents. ChiMerge est ainsi la seule méthode identifiant pratiquement tous les intervalles pour des échantillons de 500 instances. Les trois méthodes FusinterO, ChiMerge et ChiSplit sur-apprennent en produisant trop d'intervalles pour les petites tailles d'échantillon. A l'inverse, les méthodes robustes MODL, MDLPC et Khiops produisent un seul intervalle pour les petits échantillons. La transition entre les discrétisations mono-intervalle et les discrétisations à 100 intervalles intervient vers 700 instances pour la méthodes MODL de façon très abrupte. Les méthodes MDLPC, FusinterO et Khiops n'identifient correctement tous les intervalles que pour des échantillons compris entre 2000 et 3000 instances. A partir de cette taille d'échantillon, toutes les méthodes évaluées identifient correctement les 100 intervalles et deviennent indiscernables. A ce niveau, l'erreur en test est optimale, et son amélioration ne provient plus que de la meilleure estimation des bornes exactes des intervalles optimaux quand le nombre d'instances continue à augmenter.

De toutes les méthodes testées, MODL est la méthode qui produisent le moins d'intervalles pour les échantillons de petite taille et qui parvient pratiquement la première à identifier correctement les 100 intervalles.

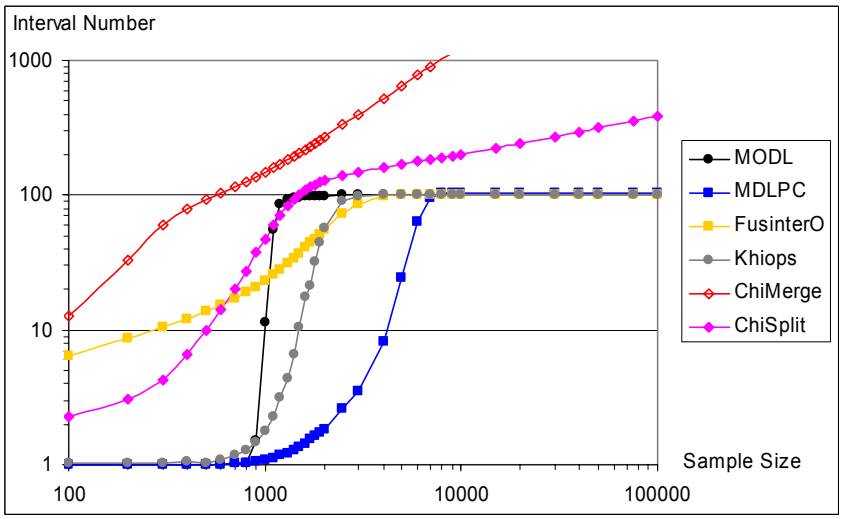


Figure 7 : Nombre d'intervalles en fonction de la taille de l'échantillon pour la discrétisation de la fonction créneaux $Class = Signe(Sinus(100\pi x))$ sur le domaine [0; 1], dans le cas bruité (10% de bruit)

L'expérimentation a également été menée dans un cas bruité, où 10% des instances choisies aléatoirement sont mal classées. Les résultats sont présentés sur la figure 7. Là encore, les trois méthodes FusinterO, ChiMerge et ChiSplit sur-apprennent en produisant trop d'intervalles, mais la méthode FusinterO ne va pas au delà des 100 intervalles. Les méthodes robustes MODL, MDLPC et Khiops ne produisent qu'un intervalle pour les petits échantillons. Là encore, la transition intervient de façon très abrupte pour la méthode MODL, pour des échantillon d'environ 1000 instances. La transition est beaucoup plus douce pour les autres méthodes, qui ont besoin de plus d'instance pour identifier correctement les 100 intervalles: environ 300O pour Khiops, 4000 pour FusinterO et 7000 pour MDLPC. La méthode MDLPC sur-apprend très légèrement (environ 103 intervalles pour les échantillons de 100000 instances) en raison de son critère d'évaluation local à deux intervalles.

En résumé, les méthodes MODL et Khiops (et dans une moindre mesure MDLPC et FusinterO) sont les seules méthodes qui asymptotiquement identifient correctement les intervalles optimaux. Parmi ces méthodes, la méthode MODL est de loin la plus sensible, en détectant bien avant les autres méthodes robustes les intervalles optimaux, tant en situation non bruitée qu'en situation bruitée.

3.3.3 Seuil de détection d'un intervalle pur

On a ici évalué le seuil de détection d'un intervalle pur isolé au milieu d'un attribut bruité. Pour cela, on a utilisé des jeux d'essai artificiels de taille comprise environ 20 et 1000000, avec deux modalités cibles exactement équi-réparties. Une première expérience consiste à évaluer le seuil de détection d'un intervalle pur (ne contenant que des instances d'une même classe cible), suivi d'un intervalle contenant le reste du jeu d'essai (mélange quasi-homogène des deux classes cibles). Pour chaque taille de jeu d'essai, on part d'un intervalle pur initial réduit à une instance, dont on incrémente progressivement la taille jusqu'à ce que la méthode de discrétisation testée aboutisse à plus de un intervalle (c'est à dire détecte l'intervalle pur en refusant de le fusionner avec l'autre intervalle). La figure 8 présente les résultats de l'expérimentation. La taille du jeu d'essai est donnée en abscisse sur une échelle logarithmique. En ordonnée, on trouve la taille minimum d'un intervalle pur en tête de jeu d'essai, nécessaire pour qu'une discrétisation aboutisse à deux intervalles. On a également tracé la courbe du logarithme base 2 de la taille du jeu d'essai, qui correspond aux sous-chaînes pures dont la probabilité d'apparition dans une chaîne aléatoire est de l'ordre de 0,5.

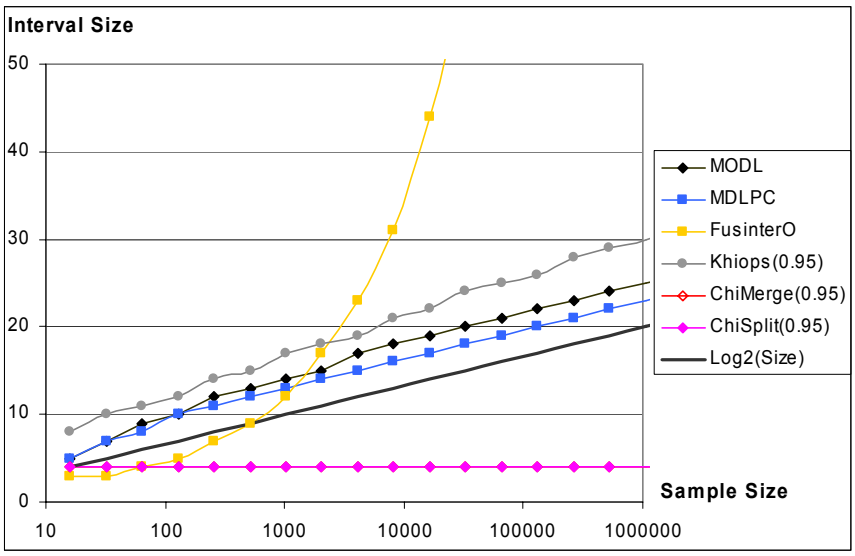


Figure 8 : Taille minimum d'un intervalle pur en tête d'un jeu d'essai équidistribué détecté par différentes méthodes de discrétisation

Les résultats montrent que les méthodes ChiMerge et ChiSplit détectent les intervalles purs dès que leur effectif dépasse 4 instances, quelle que soit la taille du jeu d'essai. Ce comportement est beaucoup trop "agressif", ce qui explique pourquoi ces méthodes sur-apprennent et fabriquent trop d'intervalles. Les trois méthodes robustes MODL, MDLPC et Khiops ont un comportement plus réaliste, en ayant une courbe de détection légèrement au dessus de la courbe du logarithme de la taille de l'échantillon, qui est une borne inf de la taille des intervalles purs pouvant apparaître par hasard. La méthode FusinterO souffre de sa contrainte de régularisation inversement proportionnelle à la taille des intervalles, ce qui l'empêche de découvrir des petits patterns intéressant en tête de l'échantillon.

On a effectué une seconde expérience consistant à détecter un intervalle pur situé cette fois au milieu d'un jeu d'essai équidistribué, entre deux intervalles quasi homogènes vis à vis des classes cibles. La figure 9 présente les résultats de cette seconde expérience.

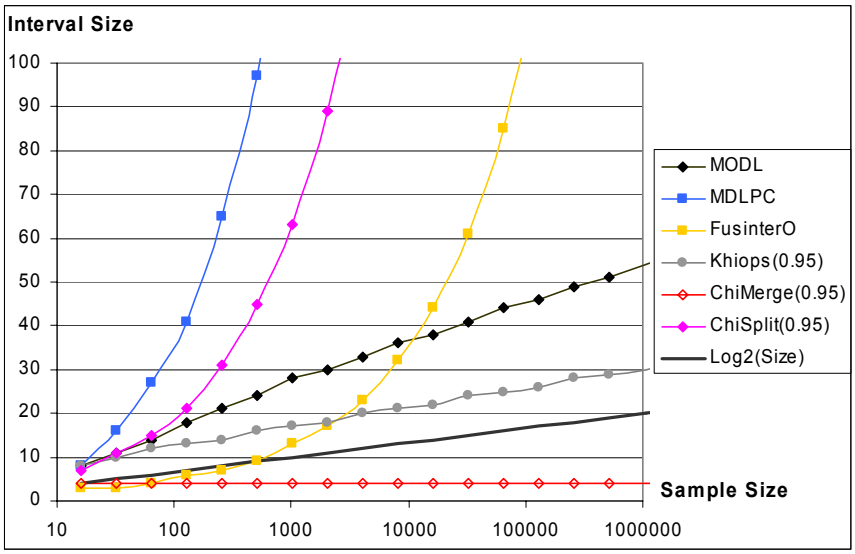


Figure 9 : Taille minimum d'un intervalle pur au milieu d'un jeu d'essai équidistribué détecté par différentes méthodes de discrétisation

La méthode ChiMerge obtient le même seuil de détection (beaucoup trop faible) de 4 instances pour l'intervalle pur. Etant une méthode ascendante, la méthode a le même comportement quelle que soit la position de l'intervalle. En revanche, la méthode ChiSplit est une méthode descendante, qui doit effectuer deux coupures successives pour détecter l'intervalle pur, la première n'étant pas très intéressante. En conséquence, la taille des intervalles purs détectables par ChiSplit devient beaucoup trop importante (de l'ordre de 60 pour un jeu d'essai de taille 1000). En raison de son algorithme, cette méthode est à la fois trop laxiste sur les bords du jeu d'essai, et trop conservatrice vers le centre du jeu d'essai. La méthode MDLPC, elle aussi basée sur un algorithme descendant, est également incapable de détecter des intervalles purs de petite taille s'ils sont au centre du jeu d'essai. La méthode FusinterO bénéficie d'un algorithme ascendant lui donnant un net avantage sur les méthodes descendantes MDLPC et ChiSplit, mais le handicap de sa technique de régularisation l'empêche de trouver des petits patterns intéressants au milieu de l'échantillon. Les deux méthodes MODL et Khiops ont un comportement plus adéquat, en permettant de détecter des intervalles purs dont la taille reste de l'ordre de grandeur du logarithme de la taille du jeu d'essai. Il est à noter que pour Khiops, la taille des intervalles purs ne dépend pas de leur position dans le jeu d'essai. En effet, cette méthode basée sur un algorithme ascendant utilise un critère assurant un contrôle statistique des fusions d'intervalles, indépendamment de leur position ou du nombre d'intervalles de la discrétisation. En revanche, la méthode MODL utilise un critère global prenant en compte le nombre total d'intervalles et la quantité d'information nécessaire pour les coder, ce qui explique que son comportement varie selon que l'intervalle pur soit en début (deux intervalles à coder) ou au milieu (trois intervalles à coder).

En conclusion de ces expérimentations sur la résistance au bruit, on observe que les méthodes ChiMerge et ChiSplit exhibent un comportement de sur-apprentissage, ChiSplit ne dominant ChiMerge qu'en raison du biais dû à son algorithme descendant. La méthode FusinterO ne parvient pas à isoler de petits intervalles intéressants quelle que soit leur position dans l'échantillon. Les trois méthodes MODL, MDLPC et Khiops sont robustes face aux données bruitées, mais MDLPC a un comportement trop conservateur en raison de son algorithme descendant et de son critère local à deux intervalles adjacents. Les méthodes MODL et Khiops parviennent à être à la fois résistantes au bruit et sensibles à la présence de petits patterns pertinents quelle que soit leur position dans le jeu d'essai.

3.4 Etude comparative des algorithmes d'optimisation

Afin d'étudier les performances des différentes heuristiques d'optimisation, nous les avons comparées à l'algorithme optimal sur 1810 discrétisations effectuée lors de l'expérimentation portant sur les jeux d'essai de l'UCI. Les algorithmes étudiés sont:

- OPT: algorithme optimal (en $O(n^3)$)
- GBUD: algorithme glouton ascendant
- OBUD: algorithme ascendant avec post-optimisation
- GTDD: algorithme glouton descendant
- OTDD: algorithme descendant avec post-optimisation

Pour mémoire, c'est l'algorithme OBUD qui est l'algorithme de référence pour la méthode MODL, principalement parce que l'approche ascendante garantit une complexité en $O(n.\log(n))$ dans le pire des cas, et parce qu'il est post-optimisé.

La figure 10 présente les résultats synthétiques en termes de taux de bonne prédiction et taille de discrétisation, en situant les performances des algorithmes pour MODL par rapport à celle des méthodes aux performances les plus proches Khiops, MDLPC et ChiSplit. On a également rapporté les résultat de la méthode Fusinter avec les mêmes algorithmes que ceux utilisés pour MODL, afin d'évaluer l'impact des algorithme pour un deuxième critère.

Pour chacun des deux critères MODL et Fusinter, les algorithmes post-optimisés tant ascendant (OBUD) que descendant (OTDD) obtiennent des performances très proches de l'algorithme optimal, tant en taux de bonne prédiction qu'en nombre d'intervalles. Pour le critère MODL, la post-optimisation améliore significativement les performances prédictives de l'algorithme glouton initial. Ce n'est pas le cas pour le critère Fusinter, pour lequel la version gloutonne descendante a même de meilleure performance que la version optimisée. Le passage d'un algorithme glouton a un algorithme optimisé n'est intéressant qu'avec le critère MODL, donc l'optimalité est fondée sur le plan théorique.

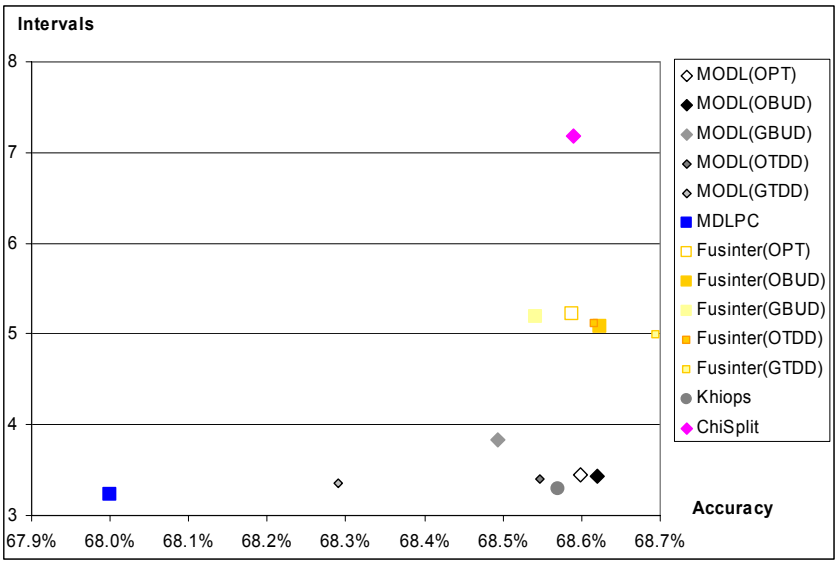


Figure 10 : Evaluation des différents algorithmes de discrétisation MODL pour les critères de taux de bonne prédiction et de taille des discrétisations, et comparaison avec les méthodes Khiops, MDLPC et ChiSplit.

Afin de mieux étudier les apports de la post-optimisation pour le critère MODL, le nombre moyen d'améliorations effectué par chaque type d'algorithme a été collecté dans le tableau 8. Par exemple, l'algorithme OBUD effectue en moyenne 275,02 Merges, puis dans sa phase de post-optimisation commence par effectuer en moyenne 0,41 ExtraMerges, et finalise l'optimisation en effectuant des Splits, MergeSplits et MergeMergeSplits. Le tableau 8 indique également le pourcentage de discrétisation ayant abouti à la discrétisation optimale (sur les 1810 discrétisations), ainsi que l'écart relatif au critère optimal, dans les cas où la discrétisation obtenue n'est pas optimale. On remarque que les algorithmes gloutons parviennent à la solution optimale dans environ 60% des cas, et que la post-optimisation permet d'obtenir la solution optimale dans environ 95% des cas. Dans les 5% des cas restant, l'écart relatif au critère optimal est très faible, de l'ordre de 0,1 à 0,2%.

On peut noter que le nombre d'améliorations apportées en post-optimisation est non négligeable par rapport au nombre d'intervalles final, et qu'il permet ainsi une réelle amélioration des algorithmes gloutons. Le nombre d'ExtraMerges (resp. Extra Splits) effectué lors de la phase d'ExhaustiveMerge (resp. ExhaustiveSplit) des algorithmes gloutons peut être assez important, jusqu'à plusieurs dizaines d'améliorations pour une discrétisation. Néanmoins, cette partie de l'algorithme de post-optimisation est contrôlé par une complexité algorithmique en $O(n.\log(n))$. La phase finale de la post-optimisation utilisant des Splits, MergeSplits et MergeMergeSplits n'utilise jamais plus qu'une demi-douzaine d'améliorations par type d'amélioration, ce qui signifie que l'algorithme de post-optimisation converge extrêmement rapidement. Il est à noter qu'un prétraitement a été effectué, consistant à initialiser les intervalles en identifiant les séquences pures de classes cibles. Ce prétraitement a permis de diminuer d'un facteur 10 en moyenne le nombre de coupures potentielles. Cela implique que pour les heuristiques de discrétisation, le temps de calcul est dominé par le tri des valeurs initiales de l'attribut descriptif à discrétiser.

Tableau 8 : Moyenne, nombre de gains et pertes pour quatre critères d'évaluations, pour les prédicteurs élémentaires univariés, pour différents algorithmes d'optimisation appliqué au critère MODL

Algorithme	Merges	Splits	ExtraMerges	ExtraSplits	MergeSplits	MMSplits	Intervals	%Optimal	DeltaCost
OPT							3,44	100,00%	0,00%
GBUD	275,02						3,84	55,41%	0,63%
OBUD	275,02	0,06	0,41		0,58	0,05	3,43	95,14%	0,14%
GTDD		2,35					3,35	68,95%	1,39%
OTDD		2,36		0,17	0,25	0,13	3,40	93,26%	0,26%

Le tableau 9 rapporte le même type de résultat pour le critère Fusinter. Là encore, la post-optimisation améliore significativement la performance de l'algorithme glouton ascendant, en permettant d'atteindre la solution optimale (minimum du critère Fusinter) dans de nombreux cas. Néanmoins, le critère Fusinter étant basé sur une technique de régularisation ajustée de façon non optimale, l'apport de l'optimisation en termes de performance prédictive ou de nombre d'intervalles est

nettement moindre que pour le critère MODL.

Tableau 9 : Moyenne, nombre de gains et pertes pour quatre critères d'évaluations, pour les prédicteurs élémentaires univariés, pour différents algorithmes d'optimisation appliqué au critère Fusinter

Algorithme	Merges	Splits	ExtraMerges	ExtraSplits	MergeSplits	MMSplits	Intervals	%Optimal	DeltaCost
OPT							5.22	100.00%	0.00%
GBUD	273.67						5.19	40.28%	0.54%
OBUD	273.67	0.18	0.16		1.17	0.12	5.09	86.19%	0.07%
GTDD		3.98					4.98	56.19%	0.44%
OTDD		4.01		0.26	0.38	0.15	5.11	85.97%	0.16%

Conclusion

La méthode MODL repose sur une approche bayésienne de la discrétisation. Un modèle standard de discrétisation est déterminé, en préalable à une distribution a priori de ces modèles. On a défini un a priori à trois étages, pour lequel on choisit de façon uniforme d'abord le nombre d'intervalles, puis la partition en intervalles, et enfin la distribution des classes cibles sur chaque intervalle. On a en plus supposé l'indépendance des distributions entre les intervalles. Sur la base de ces hyptothèses, on a montré qu'il existe un critère d'évaluation permettant de rechercher la discrétisation optimale au sens de Bayes. Un algorithme optimal en $O(n^3)$ permet de trouver l'optimum global de ce critère, et donc d'aboutir à la discrétisation optimale. Nous avons également proposé une nouvelle heuristique en $O(n.\log(n))$, basée sur une méthode gloutonne ascendante et intégrant une phase de post-optimisation. Lors d'évaluations intensives, cette heuristique a permis d'aboutir à la solution optimale dans environ 95% des cas.

Des expérimentations comparatives intensives ont été menées sur de nombreux jeux de données de l'UCI, sur une base France Telecom ainsi que de nombreux jeux d'essai théoriques.. Les résultats ont montré que les discrétisations optimales issues de la méthode MODL lui permettaient de dominer les méthodes alternatives testées sur l'ensemble des critères évalués, à savoir le taux de bonne prédiction, la robustesse, le faible nombre d'intervalles produits, la résistance au bruit, le seuil de détection des intervalles pertinents, l'absence de paramétrage et l'absence de problèmes asymptotiques dus à de très petits effectifs.

Références

Bertier, P. & Bouroche, J.M. (1981). *Analyse des données multidimensionnelles*. Presses Universitaires de France.

Blake, C.L. & Merz, C.J. (1998). UCI Repository of machine learning databases [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science.

Boullé, M. (2003). Khiops: a Discretization Method with Guaranteed Resistance to Noise. In *Proc. of the Third Conference on Machine and Data Mining in Pattern Recognition*, 50-64.

Breiman, L., Friedman, J.H., Olshen, R.A. & Stone, C.J. (1984). Classification and Regression Trees. California: Wadsworth International.

Catlett, J. (1991). On Changing Continuous Attributes into ordered discrete Attributes. In *Proceedings of the European Working Session on Learning.* Springer-Verlag, 87-102.

Cover, T.M. & Thomas, J.A. (1991). Elements of Information Theory. Wiley Series in Telecommunications.

Dougherty, J., Kohavi, R & Sahami, M. (1995). Supervised and Unsupervised Discretization of Continuous Features. *Proceedings of the Twelf International Conference on Machine Learning*, Los Altos, CA : Morgan Kaufmann, 194-202.

Elomaa, T. & Rousu, J. (1996). Finding optimal multi-splits for numerical attributes in decision tree learning. Technical report, NeuroCOLT Technical Report NC-TR-96-041, Royal Holloway, University of London.

Fayyad, U. & Irani, K. (1992). On the handling of continuous-valued attributes in decision tree generation. *Machine Learning, 8*: 87-102.

Fischer, W.D. (1958). On grouping for maximum of homogeneity. *Jounal of the American Statistical Association.* 53. 789-798.

Fulton, T., Kasif, S. & Salzberg, S. (1995). Efficient algorithms for finding multi-way splits for decision trees. In *Proc. of the Twelfth International Conference on Machine Learning.*

Hansen, M.H. & Yu, B. (1998). Model Selection and the Principle of Minimum Description Length. *Journal of the American Statistical Association*.

Holte, R.C. (1993). Very simple classification rules perform well on most commonly used datasets. *Machine Learning, 11*: 63-90.

Kass, G.V. (1980). An exploratory technique for investigating large quantities of categorical data. *Applied Statistics, 29(2)*: 119-127.

Kerber, R. (1991). Chimerge discretization of numeric attributes. *Proceedings of the 10th International Conference on Artificial Intelligence*, 123-128.

Lanterman, A.D. (2001). Schwarz, Wallace, & Rissanen: Interwinning Themes in Theories of Model Selection. *International Statistical Review*, Vol. 69, No. 2, , 185-212.

Langley, P., Iba, W., & Thompson, K. (1992). An analysis of bayesian classifiers. *Proceedings of the 10th national conference on Artificial Intelligence*, AAAI Press, 223-228.

Lechevallier, Y. (1990). Recherche d'une partition optimale sous contrainte d'ordre total. Technical report N° 1247, INRIA.

Li, M and Vitanyi, P.M.B. (1997). An Introduction to Kolmogorov Complexity and Its Applications. Second Edition. Springer Verlag.

Liu, H., Hussain, F., Tan, C.L. & Dash, M. (2002). Discretization: An Enabling Technique. *Data Mining and Knowledge Discovery 6(4)*: 393-423.

Pfahringer, B. (1995). Compression-Based Discretization of Continuous Attributes, In *Proc. of the Twelfth International Conference on Machine Learning*.

Quinlan, J.R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

Rissanen, J. (1978). Modeling by shortest data description. *Automatica, 14*, 465-471.

Rissanen, J. (1983). A universal prior for integers and estimation by minimum description length. *Ann. Statist., 11*, 416-431.

Vitanyi, P.M.B. & Li, M. (2000). Minimum Description Length Induction, Bayesianism, and Kolmogorov Complexity. *IEEE Trans. Inform. Theory*, IT-46:2, 446-464.

Zighed, D.A., Rabaseda, S. & Rakotomalala, R. (1998). Fusinter: a method for discretization of continuous attributes for supervised learning. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 6(33)*: 307-326.

Zighed, D.A. & Rakotomalala, R. (2000), *Graphes d'induction*. HERMES Science Publications, 327-359.

4 Annexe: principe MDL et discrétisation

Dans cette annexe, on présente quelques résultats de théorie de l'information, le principe MDL (Minimum Description Length) et son application dans deux méthodes de discrétisation existantes.

4.1 Quelques résultats de théorie de l'information

On résume ci-dessous quelques résultats de la théorie de l'information qui permettront de situer les principes utilisés pour la méthode de discrétisation MODL. Pour une introduction à ces éléments de théorie de l'information, on peut se référer à (Hansen & Yu, 1998; Lanterman, 2001; Vitanyi & Li, 2000), ou aux ouvrages de référence de (Cover & Thomas, 1991; Li & Vitanyi, 1997) pour des approfondissements.

4.1.1 Longueur de code optimale et probabilité

On se pose ici la question de l'optimisation de la transmission d'une chaîne de symbole.

Soit *S* une chaîne de symboles prenant ses valeurs sur des symboles A_i dont on connaît la distribution. On cherche à attribuer à chaque symbole A_i un code binaire sous forme d'une séquence de bits, de façon à minimiser le nombre total de bit à transmettre pour la chaîne S. Intuitivement, on a intérêt à utiliser les codes les plus courts pour les symboles les plus fréquents, et les plus longs pour les symboles les plus rares. Le théorème de Shannon du codage à la source affirme que la longueur de codage de la chaîne de symbole est minimisée en moyenne quand on choisit un codage des symboles pour lequel le nombre de bits utilisé pour chaque symbole est déduit de sa probabilité d'apparition par la formule suivante:

$$\text{LongueurCodage}(A_i) = -\log_2(P(A_i)).$$

Le codage de Huffman est un algorithme permettant de trouver des codes effectifs pour une distribution de symboles donnée. Ce codage est quasi optimal, et (Cover & Thomas, 1991) ont démontré que la différence de longueur entre le code de Huffman et le code optimal est au plus 1.

4.1.2 Complexité algorithmique de Kolmogorov

Kolmogorov a définit la complexité algorithmique d'une chaîne de n bits comme la longueur du plus petit algorithme permettant de générer cette chaîne. Il a été démontré que cette longueur ne dépend du choix du langage de l'algorithme qu'à une constante près, ce qui permet d'affirmer que la complexité algorithmique est une propriété intrinsèque de la chaîne, mesurant la quantité d'information qu'elle contient. Cette formulation correspond intuitivement au fait qu'une chaîne ne contenant que des 0 parait plus simple à décrire qu'une chaîne aléatoire.

La complexité algorithmique d'une chaîne correspond donc à sa compression maximale. Elle permet également de caractériser les chaînes aléatoires. Une chaîne aléatoire est une chaîne dont la complexité algorithmique est voisine de la longueur de la chaîne. En effet, aucune régularité ne permet de trouver un algorithme significativement plus efficace que le codage exhaustif de tous les bits de la chaîne. Enfin, si l'on considère une chaîne comme un historique d'évènements pour un problème où il faut prédire la valeur du prochain bit, alors la complexité algorithmique permet de définir le prédicteur dont la capacité de généralisation sera maximisée.

La complexité algorithmique de Kolmogorov est un optimum théorique. Il a été montré qu'il n'existe pas d'algorithme permettant de la calculer dans le cas général.

4.1.3 Principe (MDL) de description de longueur minimale

Le principe MDL est une technique proposée par (Rissanen, 1978) permettant de choisir le meilleur modèle possible pour représenter des données. Le principe MDL permet de formaliser et généraliser de façon rigoureuse le principe du rasoir d'Occam, consistant à choisir le modèle le plus simple expliquant les données. En se référant à la complexité de Kolmogorov (inatteignable en théorie), il s'agit de trouver le codage des données de longueur minimale en utilisant un modèle. On décompose alors le codage en deux parties, en codant d'une part le choix du modèle parmi les modèles possibles, d'autre part en codant les données connaissant le modèle. Cela fournit une borne sup de la complexité de Kolmogorov, et en minimisant la longueur totale de codage, on maximise la capacité de prédiction du modèle. On doit donc choisir le modèle qui minimise:

$$\text{LongueurCodage}(\text{Modèle}) + \text{LongueurCodage}(\text{Données/Modèle})$$

Intuitivement, le problème du choix du modèle réalise un compromis entre un modèle simple qui a une bonne capacité de généralisation, mais décrit approximativement les données, et un modèle complexe qui décrit précisément les données, mais risque de sur-apprendre. Le principe MDL permet de choisir le meilleur compromis possible.

Il y a lien fort avec l'approche bayesienne, qui dit que le modèle optimal est le modèle le plus probable connaissant les données. Dans cette approche, il faut maximiser la probabilité suivante:

$$\text{Prob}(\text{Modèle/Données})$$

D'après la formule de Bayes, cela revient à maximiser:

$$\text{Prob}(\text{Modèle})\text{Prob}(\text{Données/Modèle})/\text{Prob}(\text{Données})$$

Donc à maximiser

$$\text{Prob}(\text{Modèle})\text{Prob}(\text{Données/Modèle})$$

En se basant sur le codage optimum de Shannon (LongueurCodage(A) = -log2(P(A)), on retrouve le principe MDL de minimisation de l'expression:

LongueurCodage(Modèle) + LongueurCodage(Données/Modèle)

Le problème de l'approche Bayesienne est celui du choix d'une distribution de probabilité a priori des modèles. Le principe MDL fournit alors un guide pour choisir implicitement ces probabilités a priori, en permettant de se focaliser sur le codage des paramètres du modèle, plus facile à définir que le choix d'une distribution de probabilités.

(Vitanyi & Li, 2000) ont établi le lien entre l'approche "ideal MDL" et la complexité de Kolmogorov, en codant au moyen d'un algorithme de longueur minimale le modèle d'une part et les données sachant le modèle d'autre part. Cette approche "ideal MDL" permet de choisir un modèle optimal, qui coïncide presque toujours avec celui déterminé par une approche bayesienne pour un "prior universel". Le problème de l'approche "ideal MDL" est que la complexité de Kolmogorov est non calculable, et que l'approche est valide asymptotiquement. Ce premier problème est résolu en recherchant une approximation de la complexité de Kolmogorov, par exemple au moyen d'un codage de Shannon (valide asymptotiquement).

4.2 Exemple: codage d'une chaîne de bits avec un modèle binomial

Dans cette section, on passe en revue les schémas classiques de codage d'une chaîne de bits de longueur n et comportant k bits de valeur 1, basée sur l'utilisation d'un modèle binomial. Ces schémas de codage sont décrits et analysés de façon détaillée dans (Hansen & Yu, 1998).

4.2.1 Uniform coding

Ce schéma de codage sert de référence, et n'utilise aucun modèle. Il s'agit donc de transmettre tels quels les n bit de la chaîne. On a ici:

$Cost = n$

4.2.2 Two-stage coding

On utilise ici un modèle de Bernouilli de paramètre p, ou $p=k/n$ est la proportion de 1 dans la chaîne à coder. Il faut d'abord coder p, qui pouvant prendre n+1 valeurs possibles, nécessite $\log_2(n+1)$ bits. D'après Shannon, on peut trouver un code dont la longueur sera $-\log_2(k/n)$ pour les bits à 1 et $-\log_2(1-k/n)$ pour les bits à 0. Il faudra coder k bits à 1 et $(n-k)$ bits à 0, ce qui donne une longueur totale de code égale à $\log_2(n+1) + [-k \log_2(k/n) -(n-k) \log_2(1-k/n)]$.

On reconnaît l'entropie de Shannon ent(p) = $-p.\log_2(p) - (1-p).\log_2(1-p)$, on a donc:

$$Cost = \log_2(n+1) + n.entropy(k/n)$$

4.2.3 Mixture coding

On considère ici a priori que p a une distribution uniforme sur l'intervalle [0, 1]. Un calcul d'intégration sur la probabilité p entre 0 et 1 montre que dans ce cas, la longueur du code de transmission de k bits à 1 résultant de cette hypothèse est:

$$Cost = \log_2((n+1)!) - \log_2(k!) - \log_2((n-k)!)$$

$$Cost = \log_2(n+1) + \log_2\left(C_n^k\right)$$

4.2.4 Predictive coding

On imagine ici que l'on transmet le premier bit avec un paramètre de Bernouilli a priori p=1/2. Il faut alors $-\log_2(1/2)$ = bits pour ce premier bit transmis. Le paramètre de Bernouilli est alors réactualisé en tenant compte de la valeur du premier bit, et le second bit est transmis sur la base de ce nouveau paramètre. Ce schéma est réitéré jusqu'à la transmission du dernier bit. Un calcul montre que dans ce cas, le coût de codage est exactement le même que dans le cas du Mixture coding, à savoir:

$$Cost = \log_2(n+1) + \log_2\left(C_n^k\right)$$

4.2.5 Normalized Maximized Likelihood Coding

On utilise ici un schéma analogue à celui du two-stage coding. On transmet dans un premier temps le nombre exact de k dans la chaîne, ce qui réclame $\log_2(n+1)$. Suivant le modèle de Bernouilli, toutes les chaînes de n bits comportant k bits à 1 sont équiprobables. Il suffit d'indiquer l'index de la chaîne parmi les chaînes possibles. On a donc en définitive une fois de plus le même coût de codage que pour le mixture coding, à savoir:

$$Cost = \log_2(n+1) + \log_2\left(C_n^k\right)$$

4.2.6 Commentaires

Tous les schémas de codage se révèlent ici finalement équivalent. La seule variante au final réside dans le codage des exceptions au modèle. Le schéma du two-stage coding utilise l'entropie de Shannon. Le Normalized Maximized Likelihood Coding se base sur le fait que toutes les chaînes sont équiprobables pour un k donné, et ce codage se ramène à un problème combinatoire.

Il est a noter que l'hypothèse du mixture coding qui prend en compte une distribution de probabilité p continue et uniforme sur [0, 1] aboutit après des calculs relativement complexes exactement au même résultat qu'un simple calcul de dénombrement (basé sur une distribution discrète de probabilité).

4.3 Méthodes de discrétisation basée sur MDL

Dans cette partie, nous présentons deux méthodes de discrétisation basées sur le principe MDL: la méthode MDLPC (Fayyad & Irani, 1992) et la méthode MDL-DISC (Pfahringer, 1995).

4.3.1 MDLPC

La méthode MDLPC est une méthode descendante recherchant récursivement la meilleure bipartition d'un attribut, en optimisant un critère local à deux attributs adjacents. Le critère mesure la quantité d'information d'un intervalle sans coupure, et celle d'un intervalle avec coupure. La méthode de discrétisation part de l'intervalle complet, évaluent toutes les coupures potentielles, et retient celle dont la quantité d'information résultante est minimum. Si cette quantité d'information est inférieure à celle de l'intervalle initial, la coupure est retenue, et l'algorithme est appliqué récursivement aux deux sous-intervalles.

On présente ci-dessous le codage utilisé par la méthode MDLPC.
Soient S_0 l'intervalle initial d'effectif n, S_1 et S_2 deux sous-intervalles de S_0 d'effectif n_1 et n_2.
Soient J le nombre de classes cibles, $n_{0,j}$, $n_{1,j}$ et $n_{2,j}$ le nombre d'instances de S_0, S_1 et S_2 pour la classe j.
La longueur moyenne de codage d'une instance d'un ensemble est choisie en utilisant l'entropie de l'ensemble:

$$ent(S_i) = -\sum_{j=1}^{J} \left(n_{i,j}/n_i\right) \log_2 \left(n_{i,j}/n_i\right)$$

Pour coder un intervalle, la méthode MDLPC définit dans un premier temps la table des codes utilisés, puis code toutes les instances en se référant à cette table de code. Le coût de codage de l'intervalle initial S est:

$InitialCost = J.\text{ent}(S) + n.\text{ent}(S)$.

Pour coder la version avec deux intervalles, la méthode MDLPC code la position de coupure, avec $\log_2(n\text{-}1)$ bits (une position parmi n-1 possibilités), puis les instances de chaque sous-intervalle. Le codage MDLPC introduit une subtilité supplémentaire en tenant compte du fait que les sous intervalles S_1 et S_2 peuvent utiliser des nombres de classes cibles J_1 et J_2 inférieurs à J. Le nombre potentiel de couples (J_1, J_2) a été évalué à $3^J\text{-}2$. En définitive, le coût de codage des deux intervalles résultant d'une coupure de l'intervalle initial est:

$SplitCost =$

$\log_2(n\text{-}1) +$	Codage du point de coupure
$\log_2(3^J\text{-}2) +$	Codage de la répartition des classes effectivement utilisées entre S_1 et S_2
$J_1.\text{ent}(S_1) + n_1.\text{ent}(S_1) +$	Codage des instances de S_1
$J_2.\text{ent}(S_2) + n_2.\text{ent}(S_2).$	Codage des instances de S_2

L'utilisation de l'entropie pour coder les exceptions assure un bon comportement asymptotique dès que les effectifs par intervalles sont non négligeables. Par contre, quand le coût de la table de code devient non négligeable devant le coût des instances, le codage MDLPC a pour conséquence de privilégier les intervalles purs de très petite taille. Par exemple, dans le cas de deux instances de deux classes cibles différentes, on a $InitialCost = 4$ et $SplitCost = \log_2(7)$, ce qui fait que l'option a deux singletons est préférée. Un rapide calcul montre que ce comportement se manifeste jusqu'au cas d'un premier intervalle pur de taille 5 suivi d'un singleton de l'autre classe. Au delà, l'option à un seul intervalle est préférée. Ce comportement, gênant si l'algorithme MDLPC utilisait une approche ascendante, ne pose pas de problème dans le cadre de l'approche descendante de MDLPC. En effet, en partant des intervalles les plus peuplés, MDLPC reste dans le cadre "asymptotique" de bon comportement du critère utilisé, et arrête les coupures d'intervalles avant de rencontrer les zones de comportement limite du critère.

En revanche, l'utilisation d'une approche descendante limite la qualité de l'optimisation de la discrétisation. Les premières coupures ayant identifié les grandes structures du domaine numérique peuvent éventuellement entraver la découverte de "patterns" de plus petite taille. L'exemple le plus flagrant est le pattern consistant en un intervalle pur ne comportant qu'une classe cible (donc très intéressant), isolé au milieu de deux intervalles équilibrés vis à vis des classes cibles (donc sans intérêt). La stratégie descendante recherche une bipartition alors qu'ici deux coupures sont nécessaires pour isoler l'intervalle intéressant. Cet intervalle intéressant risque ne pas être détecté si la première coupure entraîne un surcoût de codage local, provoquant l'arrêt de l'algorithme.

4.3.2 MDL-DISC

La méthode MDL-DISC utilise un critère d'évaluation global d'une discrétisation. L'auteur renonce explicitement à utiliser les heuristiques classiques ascendantes ou descendantes en raison de leur complexité algorithmique en $O(n^2)$. L'heuristique d'optimisation utilisée procède en deux étapes. Dans un premier temps, la méthode D-2 (Catlett, 1991) est utilisée pour générer un ensemble de points de coupure potentiellement intéressants. Cette méthode est une méthode descendante cherchant

récursivement la meilleure bipartition d'un intervalle en maximisant le critère de gain d'information (également utilisé dans les algorithmes ID3/C4.5 (Quinlan, 1993)). La méthode est ici appliquée jusqu'à un niveau de profondeur 5 pour obtenir 32 intervalles initiaux. La méthode MDL-DISC utilise alors un algorithme de type best first search pour rechercher la meilleure discrétisation en optimisant le critère MDL dans l'espace des 31 points de coupures potentiels.

On présente ci-dessous le codage utilisé par la méthode MDL-DISC (qui est présente dans le cas de deux classes cibles).
Soit I le nombre d'intervalles, I_{max} le nombre max d'intervalles (on a ici I_{max}=32 après le prétraitement par l'algorithme D-2).
Soit I_1 le nombre d'intervalles pour lesquels la classe majoritaire est la classe 1.
Soit S_i l'intervalle i d'effectif n_i, se répartissant en $n_{i,1}$ et $n_{i,2}$ instances de chaque classe cible.
Soit $n_{imaj} = \max(n_{i,1}, n_{i,2})$ le nombre d'instances de l'intervalle i de la classe majoritaire de l'intervalle.

Le codage s'appuie principalement sur le codage du choix de k possibilités parmi n en utilisant l'entropie avec $p=k/n$:

$$Cost(choose(k,n)) = n.ent(k,n) \text{ avec } ent(k,n) = -(k/n)\log_2(k/n) - (1-k/n)\log_2(1-k/n)$$

Le coût total de la discrétisation est défini de la façon suivante:
Coût de la discrétisation:

$DiscretizationCost =$

$(I_{max}\text{-}1).\text{ent}(I\text{-}1, I_{max}\text{-}1) +$	Codage des points de coupures
$I.\text{ent}(I_1, I) +$	Codage des classes majoritaires des intervalles
$\sum_i n_i .ent(n_{imaj}, n_i)$	Codage des exemples de la classe majoritaire dans chaque intervalle

Contrairement à MDLPC, le codage MDL-DISC ne code pas les tables de code, et se contente de coder les classes majoritaires de chaque intervalle, en se passant de coder la proportion des classes. Cela peut poser des problèmes y compris dans certains cas avec des intervalles comportant des effectifs importants. Prenant l'exemple de deux intervalles purs de cardinaux égaux $n_1=n_2=n$.

Dans le cas où les deux intervalles sont fusionnés, le codage des exemples est de $2n$, alors qu'il est nul dans le cas où les deux intervalles restent disjoints (l'entropie est nulle dans le cas d'un intervalle pur). La différence réside alors dans les coûts de codage du modèle. Calculons alors la différence de coût suite à la fusion des deux intervalles purs.

$$DeltaCost = (I_{max}\text{-}1).(\text{ent}(I\text{-}2, I_{max}\text{-}1) - \text{ent}(I\text{-}1, I_{max}\text{-}1)) + (I\text{-}1).\text{ent}(I_1, I\text{-}1) - I.\text{ent}(I_1, I) + 2n$$

$$DeltaCost = DeltaModelCost + 2n$$

L'étude numérique de la partie codage du modèle du *DeltaCost*, en faisant varier I entre 2 et I_{max} et I_1 entre 0 et I, montre que *DeltaModelCost* varie entre environ –8 et plus 8. Cela signifie que dès que n est plus grand que 4, le codage privilégie la solution avec deux intervalles purs par rapport à la solution avec un seul intervalle fusionné. Par exemple, pour I=16 et I_{max}=8, on a $DeltaCost = -1{,}23 + 2n$, ce qui entraîne que la discrétisation comportant deux intervalles singletons successifs est préférée à la solution comportant un intervalle de deux instances. Cela est dû au codage qui donne un poids trop faible au modèle (en codant uniquement la classe majoritaire de chaque intervalle, au lieu de coder la proportion exacte de la classe majoritaire de l'intervalle). Néanmoins, le pré-traitement par l'algorithme D-2 qui consiste à identifier en préalable 32 intervalles initiaux peut éventuellement compenser ce problème en se basant sur un faible nombre d'intervalles ayant probablement des effectifs suffisants pour échapper aux problèmes de comportement aux limites.

4.3.3 Commentaires

Dans le cas particulier d'un seul intervalle avec deux classes cibles, on se ramène en fait au cas du codage d'une chaîne de bits avec une loi binomiale. Cela permet de positionner les codages MDL utilisés dans les algorithmes de discrétisation par rapport aux codages classiques d'une chaîne de bits utilisant un modèle binomial. On s'aperçoit alors que la partie codage des exceptions est correctement gérée (utilisation de l'entropie de Shannon dans MDLPC et MDL-DISC, comme dans le schéma two-stage coding), mais que la partie codage du modèle semble sous-estimée. Ainsi, MDLPC code le modèle (codebook dans sa terminologie) en utilisant 2 fois l'entropie de Shannon du paramètre de Bernouilli, c'est à dire en moins de deux bits, quelle que soit la longueur de l'intervalle. De même, MDL-DISC code uniquement la classe majoritaire de chaque intervalle, c'est à dire que MDL-DISC utilise au plus 1 bit pour coder le modèle d'un intervalle, indépendamment de sa longueur. Cela semble largement sous-estimé en regard des codages classiquement proposés dans la communauté MDL, ce qui explique la tendance de ces codages à favoriser plus de découpages d'intervalles dès que l'on a atteint des effectifs faibles par intervalle. Il semble néanmoins que ces problèmes soient compensés par les heuristiques d'optimisation utilisées, qui ont un biais naturel en faveur des discrétisations à faible nombre d'intervalles.

5 Annexe: autres a priori de discrétisation supervisée

5.1 Discrétisation avec a priori à deux étages du premier type

Définition: On appelle *a priori à deux étages du premier type* l'a priori de modèle SDM basé sur les hypothèses suivantes:
- le nombre d'intervalles est compris entre 1 et n, de façon équiprobable,

- pour un nombre d'intervalles donné, toutes les partitions en intervalles de la chaîne à discrétiser et toutes les distributions de symboles pour ces intervalles sont équiprobables,
- les distributions des symboles sur chaque intervalle sont indépendantes les unes des autres.

Théorème: Un modèle de discrétisation standard *M* suivant un a priori à deux étages du premier type est optimal au sens de Bayes si son évaluation par la formule suivante est minimale sur l'ensemble de tous les modèles:

$$Value(M) = \log(n) + \log\left(C_{n+I.J-1}^{I.J-1}\right) + \sum_{i=1}^{I} \log\left(n_i! / n_{i,1}! n_{i,2}! .. n_{i,J}!\right)$$

On a alors des résultats analogues à ceux présentés pour l'a priori à trois étages.

Théorème: Dans un modèle SDM optimal suivant un a priori à deux étages du premier type, il ne peut y avoir de point de coupure entre deux instances ayant même classe cible.

Théorème: On se place ici dans le cas où le nombre de classes cibles est égal à deux. Dans un modèle SDM optimal suivant un a priori à deux étages du premier type, il ne peut exister une discrétisation comportant deux intervalles singletons successifs.

Théorème: On se place ici dans le cas où le nombre de classes cibles est égal à deux. Dans un modèle SDM optimal suivant un a priori à deux étages du premier type, la longueur maximale d'un intervalle extrême (décomposable en deux intervalles purs de même taille) est définie par l'équation suivante:

$$L - 1/2 \log_2(L) = 2 \log_2(n/I) + O(1)$$

On remarque que les intervalles extrêmes peuvent être de longueur deux fois plus importantes que les sous-chaînes d'une chaîne aléatoire, ce qui rend l'a priori à deux étages du premier type moins attractif que l'a priori à trois étages.

5.2 Discrétisation avec a priori à deux étages du second type

Définition: On appelle *a priori à deux étages du second type* l'a priori de modèle SDM basé sur les hypothèses suivantes:
- toutes les partitions en intervalles sont équiprobables quel que soit le nombre d'intervalles (compris entre 1 et *n*),
- pour un intervalle donné, toutes les distributions de symboles sont équiprobables,
- les distributions des symboles sur chaque intervalle sont indépendantes les unes des autres.

Théorème: Un modèle de discrétisation standard *M* suivant un a priori à deux étages du second type est optimal au sens de Bayes si son évaluation par la formule suivante est minimale sur l'ensemble de tous les modèles:

$$Value(M) = \sum_{i=1}^{I} \log\left(C_{n_i+J-1}^{J-1}\right) + \sum_{i=1}^{I} \log\left(n_i! / n_{i,1}! n_{i,2}! .. n_{i,J}!\right)$$

Théorème: Dans un modèle SDM optimal suivant un a priori à deux étages du second type, il ne peut y avoir de point de coupure entre deux instances ayant même classe cible.

Théorème: Dans un modèle SDM optimal suivant un a priori à deux étages du second type, il ne peut y avoir d'intervalle séparable en deux intervalles purs.

Dans la discrétisation optimale, il faut donc exactement un intervalle par séquence pure de classe cible. On retrouve ici l'algorithme de discrétisation 1R (Holte, 1993), que Holte avait présenté pour montrer que les bases de l'UCI sont suffisamment simples pour que même des méthodes basiques obtiennent de bons résultats. Holte avait néanmoins rajouté une contrainte empirique d'effectif minimum de 6 par intervalle, afin de prévenir le sur-apprentissage.

On constate ici que l'a priori à deux étages du second type n'est plus du tout compatible avec le comportement des chaînes aléatoires.

Théorème: La quantité d'information contenue dans le choix d'un modèle SDM suivant un a priori à deux étages du second type est supérieure à la taille de la chaîne *S*.

Il paraît clair d'après ce théorème que l'a priori à deux étages du second type n'a aucun intérêt pour discrétiser une chaîne. Il est en effet plus efficace de transmettre la chaîne directement.

5.3 Discrétisation avec a priori uniforme

Définition: On appelle *a priori uniforme* l'a priori de modèle SDM basé sur les hypothèses suivantes:

- tous les modèles de discrétisation sont équiprobables, quel que soit le nombre d'intervalles (compris entre 1 et n), la partition en intervalles et la distribution de symboles par intervalles,
- les distributions des symboles sur chaque intervalle sont indépendantes les unes des autres.

Théorème: Un modèle de discrétisation standard *M* suivant un a priori uniforme est optimal au sens de Bayes si son évaluation par la formule suivante est minimale sur l'ensemble de tous les modèles:

$$Value(M) = \sum_{i=1}^{I} \log\left(n_i! / n_{i,1}! n_{i,2}! ... n_{i,J}!\right)$$

Corollaire: Un modèle SDM suivant un a priori uniforme ayant un intervalle par symbole de la chaîne est optimal au sens de Bayes.

En effet, l'évaluation d'un tel modèle de discrétisation est nulle, donc minimale.

L'a priori uniforme correspond en fait à une absence de biais inductif, et ne permet pas de réellement apprendre un modèle de discrétisation. Toute discrétisation comprise entre la discrétisation (non contrainte) 1R de Holte et la discrétisation élémentaire avec une instance par intervalle est optimale dans ce cas, ce qui est évidemment absurde en généralisation. On peut néanmoins construire des jeux d'essai pour lesquels cette stratégie est la plus efficace. Par exemple, tout jeu d'essai où la taille de l'ensemble d'apprentissage est supérieure au cardinal de l'espace des instances possibles va favoriser une stratégie d'apprentissage "par cœur".

5.4 Etude expérimentale de l'impact de l'a priori

Une discrétisation SDM est caractérisée par le choix du nombre d'intervalles, de la partition du jeu d'essai en intervalles et du choix des distributions des classes cibles par intervalle. L'a priori à trois étages (3S) organise ces trois choix de façon hiérarchique, et utilise une distribution uniforme à chaque étage. Cet a priori semble le plus naturel et possède de bonnes propriétés théoriques, qui ont conduit à le retenir pour la méthode MODL. Les trois autres a priori "naturels" sont:

- a priori a deux étages du premier type (2S1): choix(I), puis choix($\{n_i\}$, $\{n_{ij}\}$)
- a priori a deux étages du second type (2S2): choix(I, $\{n_i\}$), puis choix($\{n_{ij}\}$)
- a priori un uniforme (U): choix(I, $\{n_i\}$, $\{n_{ij}\}$)

L'étude théorique a montré que seul l'a priori à deux étages du premier type était également pertinent, les deux derniers a priori conduisant à un apprentissage "par cœur". On a montré de façon théorique qu'asymptotiquement, l'a priori 3S permet de détecter certains patterns pertinents de façon plus fine que l'a priori 2S1. Nous avons effectué l'expérience de détection d'un intervalle pur en tête d'un jeu d'essai afin de comparer finement les deux a priori. La figure 11 présente les résultats et confirme que l'a priori 3S est clairement plus sensible que l'a priori 3S pour la détection des petits patterns.

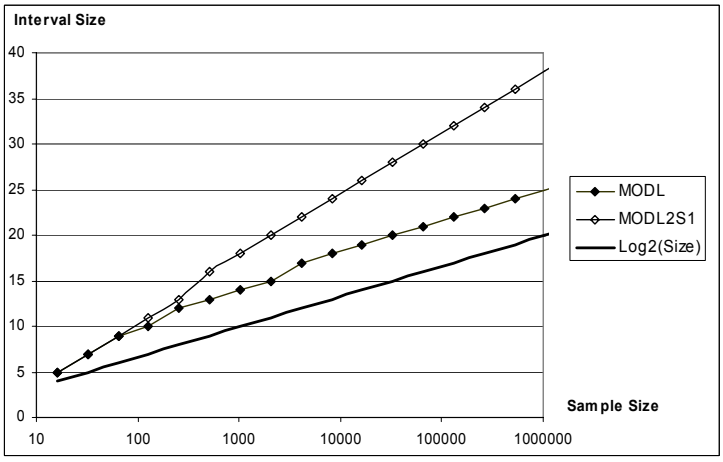


Figure 11 : Taille minimum d'un intervalle pur en tête d'un jeu d'essai équidistribué détecté par la méthode MODL pour l'a priori à trois étages et pour l'a priori à deux étages du premier type (MODL2S1)

Les deux a priori ont également été comparés sur l'ensemble des jeux d'essai issus des bases UCI, en utilisant l'algorithme optimal afin d'éviter tout biais dû à une heuristique d'optimisation. Un troisième a priori hybride a été testé. Cet a priori consiste à supposer que les deux a priori 3S et 2S1 sont équiprobables, ce qui se traduit par un nouveau critère égal au minimum des deux critères utilisés. L'idée est de tester si l'on peut accroître la performance de la méthode, même au prix d'un algorithme plus complexe, permettant de se rapprocher de la complexité algorithmique de Kolmogorov. Le tableau 10 présente une synthèse des résultats pour quatre critères d'évaluation.

Tableau 10 : Moyenne, nombre de gains et pertes pour quatre critères d'évaluations, pour les prédicteurs élémentaires univariés MODL utilisant trois types d'a priori

	Min(3S,2S1)	MODL(3S)			MODL(2S1)		
			+	-		+	-
Test Accuracy	68,6	68,6	1	0	68,6	1	0
Ratio Accuracy	98,4	98,4	0	0	98,5	0	0
Mono-intervals	18,6	18,8	0	0	18,7	0	0
Interval Number	3,5	3,4	0	0	3,4	4	0

On observe des résultats quasiment identiques, à la fois en moyenne et individuellement. En effet, sur 181 attributs discrétisés (10 fois), le nombre de différences significatives (testé au seuil de 95%) est quasiment nul. Cela signifie que bien que théoriquement préférable, l'a priori 3S ne se distingue pas de l'a priori 2S1 lors de cette expérimentation. Les bases de l'UCI ne sont sans doutes pas assez sélectives pour différencier les deux a priori, entre eux et vis à vis de l'a priori hybride.

6 Annexe: a priori de discrétisation non supervisé

On présente dans cette annexe une utilisation de l'approche MODL pour les méthodes EqualFrequency et EqualWidth dans le cas supervisé, permettant de déterminer le nombre optimal d'intervalles.

6.1 Codage "universel" des entiers naturels

Lorsque l'on dispose d'un nombre fini d'entiers naturels compris entre 1 et *n*, il parait "naturel" de supposer une distribution uniforme de ces entiers, chacun ayant une probabilité 1/n d'apparition. Cette approche ne se généralise évidement pas au cas où tous les nombre entiers sont possibles (en nombre infini donc).

L'a priori "universel" des entiers naturels présenté par (Rissanen, 1983) permet de construire un code pour les entiers naturels {1, 2, 3,…} en partant de la propriété de décroissance de la longueur de code avec les entiers. Le taux de décroissance de cette longueur de code est le plus faible possible, en étant contraint par les inégalités de Kraft (le code doit correspondre à une distribution de probabilités). Ce codage est "universel" dans le sens où il est le plus compact possible pour coder les grands entiers naturels. Ce codage peut également être intéressant même dans le cas fini, si l'on désire un a priori où les petits entiers sont plus probables que les grands entiers.

La longueur du codage universel des entiers est donnée par la formule suivante:

$$L(n) = \log_2(c_0) + \log_2^*(n) = \log_2(c_0) + \sum_{j>1} \max\left(\log_2^{(j)}(n), 0\right)$$

où $\log_2^{(j)}(.)$ est la j^{ème} composition de $\log_2$ (par exemple: $\log_2^{(2)}(n.) = \log_2(\log_2(n.))$) et $c_0 = \sum_{n>1} 2^{-\log_2^*(n)} = 2.865...$.

6.2 Discrétisation avec a priori de fréquence égale par intervalle

6.2.1 Critère d'évaluation

Définition: On appelle *a priori de fréquence égale par intervalle* l'a priori de modèle SDM basé sur les hypothèses suivantes:

- le nombre d'intervalle est compris entre 1 et n, distribué suivant l'a priori "universel" des entiers naturels,
- tous les intervalles ont la même fréquence,
- pour un intervalle donné, toutes les distributions de symboles sont équiprobables,
- les distributions des symboles sur chaque intervalle sont indépendantes les unes des autres.

Théorème: Un modèle de discrétisation standard *M* suivant un a priori de fréquence égale par intervalle est optimal au sens de Bayes si son évaluation par la formule suivante est minimale sur l'ensemble de tous les modèles de discrétisation en intervalles de fréquence égale:

$$Value(M) = \log_2^*(n)\log(2) + \sum_{i=1}^{I} \log\left(C_{n_i+J-1}^{J-1}\right) + \sum_{i=1}^{I} \log\left(n_i!/n_{i,1}!n_{i,2}!..n_{i,J}!\right)$$

Preuve:

La preuve est similaire aux preuves données pour les a priori MODL en annexe 7. On se contente ici d'apporter quelques précisions.

Le premier terme correspond au choix du nombre d'intervalle. Il est basé sur le codage universel des entiers, normalisé par un facteur log(2) pour être homogène au codage des intervalles

Il n'y a pas de termes pour le choix des bornes des intervalles. Ces bornes sont en effet exactement déterminées par le choix du nombre d'intervalles et la contrainte de fréquence égale par intervalle. Il ne reste alors que le codage de la distribution des

classes cibles dans chaque intervalle et des instances connaissant cette distribution.

Le théorème suivant correspond à une propriété "intuitive" de comportement pour un attribut cible indépendant de l'attribut descriptif. Il est à noter que la preuve de ce théorème résulte du choix de l'a priori sur le nombre des intervalles de la discrétisation.

Théorème: Dans un modèle SDM suivant un a priori de fréquence égale par intervalle, dans le cas de deux classes cibles de fréquence égale, la discrétisation en un seul intervalle est asymptotiquement plus probable que la discrétisation en *n* intervalles.

Preuve:

Il suffit de calculer le coût de codage dans ces deux cas extrêmes.

Dans le cas avec un seul intervalle, on a:

$Value(M(I=1)) = \log_2^*(1)\log(2) + \log(C_{n+1}^1) + \log(n!/(n/2)!(n/2)!)$.

En utilisant l'approximation de $C_n^{n/2} \sim 2^n / \sqrt{\pi n/2}$

$Value(M(I=1)) \sim \log(c_0) + \log(n+1) + n\log(2) - 1/2\log(n) - 1/2\log(\pi/2)$.

Dans le cas avec autant d'intervalles que d'instances, on a:

$Value(M(I=n)) = \log_2^*(n)\log(2) + \sum_{i=1}^{n}\log(C_2^1) + \sum_{i=1}^{I}\log(1!/1!0!)$.

$Value(M(I=n)) = \log_2^*(n)\log(2) + n\log(2)$.

En utilisant l'égalité $\log_2^*(n) = \sum_{j>1}\max(\log_2^{(j)}(n), 0)$., on déduit que $Value(M(I=n)) \geq \log(n) + n\log(2)$.

Le codage en 1 intervalle est moins coûteux que le codage en n intervalles (le coût est environ $1/2\log(n)$ moins cher), et correspond par conséquent à une discrétisation plus probable.

Il est à noter que cette propriété est fausse si l'on choisit un a priori uniforme entre 1 et *n* pour le nombre d'intervalles. En effet, dans ce cas, le coût de codage en 1 intervalle est environ $1/2\log(n)$ plus cher que le codage en *n* intervalles.

6.2.2 Algorithme d'optimisation

Un algorithme élémentaire de recherche de la discrétisation optimale consiste à évaluer toutes les discrétisations non supervisées en intervalles de fréquence égale, et à retenir celle minimisant le critère d'évaluation. Cet algorithme est en O(n^2). On va montrer que l'on peut ramener cette complexité algorithmique à O(n.log(n)).

Tout d'abord, on va fixer clairement le déroulement de l'algorithme EqualFrequency pour un nombre d'intervalles *I* fixé. On calcule d'abord la fréquence moyenne des intervalles $n_i = \lceil n/I \rceil$. Les instances étant triées par valeurs descriptives croissantes, on constitue le premier intervalle en collectant les n_i premières instances. Les instances ayant la même valeur que la dernière instance collectée doivent également être rajoutées pour définir la borne sup de ce premier intervalle (qui peut donc au final contenir plus de n_i instances). On procède de la même façon avec les instances suivantes jusqu'à rangement de toutes les instances dans les intervalles. Dans le cas où le nombre de valeurs descriptives est plus petit que le nombre d'instances, certains intervalles seront surpeuplés, et le nombre d'intervalles constitués pourra être inférieur à *I*. En particulier, le dernier intervalle pourra contenir strictement moins de n_i instances. Si le dernier intervalle contient moins de la moitié des instances nécessaires, on le fusionnera avec l'avant dernier intervalle.

La complexité de cet algorithme EqualWidth élémentaire est en O(n.log(n)) pour la phase de tri des instances, puis en O(n) pour la passe de constitution des intervalles.

On peut optimiser la phase de constitution et d'évaluation MODL des intervalles en O(*I*) de la façon suivante. Il suffit dans une première étape de mémoriser pour chaque index d'instance les fréquence cumulées des classes cibles. Ainsi, le calcul des fréquences des classes cibles par intervalle peut se faire par différence des fréquences cumulées entre l'index de la première instance et de la dernière instance de l'intervalle.

Au total, l'algorithme MODLEqualFrequency doit évaluer les discrétisations en *I* intervalles pour *I* variant de 1 à *n*, correspondant à des fréquences minimum par intervalles n_i distinctes (deux valeurs de *I* peuvent correspondre à la même fréquence n_i). Le nombre total d'intervalles évalués correspond au nombre de valeurs n_i distinctes, pour n_i variant de *n* à 1. Comme $n_i = \lceil n/I \rceil$, on obtient l'inégalité suivante: $I \leq n/(n_i - 1)$.

Le nombre total d'intervalles évalués est alors majoré par $n + \sum_{n_i=2}^{n} n/(n_i - 1) \sim n(1 + \log(n))$.

Au total, la complexité algorithmique de l'algorithme MODLEqualFrequency est donc en O(n.log(n)).

Algorithme MODLEqualFrequency

- Initialisation
  - Tri des valeurs de l'attribut descriptif : en O(n.log(n))
  - Calcul des fréquences des classes cibles, cumulées par instances : en O(n)
- Optimisation de la discrétisation
  - Répéter pour I = 1 à n
    - Calculer $n_i = \lceil n/I \rceil$: si n_i est égale au n_i de l'étape précédente, passer directement à l'étape suivante
    - Evaluer le coût de la discrétisation en I intervalles: en O(I)
      - ✓ Recherche des index des instances correspondant aux bornes de l'intervalle: en O(1)
      - ✓ Calcul des fréquences cibles sur chaque intervalle par différence des fréquences cumulées entre les bornes de l'intervalle, pour l'évaluation du critère MODL: en O(I)
    - Si amélioration de l'évaluation , mémorisation du meilleur I

6.3 Discrétisation avec a priori de largeur égale par intervalle

6.3.1 Critère d'évaluation

On obtient ici des résultats identiques à ceux données pour l'a priori de fréquence égale par intervalle. Il est à noter que l'on utilise ici une variante de la définition des modèles SDM, puisque l'on tient compte des valeurs de l'attribut descriptif pour le choix des bornes d'une discrétisation (contrairement à la première hypothèse des modèles SDM.

Définition: On appelle *a priori de largeur égale par intervalle* l'a priori de modèle SDM basé sur les hypothèses suivantes:
- le nombre d'intervalle est compris entre 1 et n, distribué suivant l'a priori "universel" des entiers naturels,
- tous les intervalles ont la même largeur,
- pour un intervalle donné, toutes les distributions de symboles sont équiprobables,
- les distributions des symboles sur chaque intervalle sont indépendantes les unes des autres.

Théorème: Un modèle de discrétisation standard M suivant un a priori de largeur égale par intervalle est optimal au sens de Bayes si son évaluation par la formule suivante est minimale sur l'ensemble de tous les modèles de discrétisation en intervalles de largeur égale:

$$Value(M) = \log_2^*(n)\log(2) + \sum_{i=1}^{I} \log\left(C_{n_i+J-1}^{J-1}\right) + \sum_{i=1}^{I} \log\left(n_i!/n_{i,1}!n_{i,2}!...n_{i,J}!\right)$$

Théorème: Dans un modèle SDM suivant un a priori de largeur égale par intervalle, dans le cas de deux classes cibles de fréquence égale, la discrétisation en un seul intervalle est asymptotiquement plus probable que la discrétisation en n intervalles.

6.3.2 Algorithme d'optimisation

Le nombre de quantiles en discrétisation EqualWidth peut a priori être quelconque entre 1 et l'infini. Un nombre très important (I>>n) de quantiles peut ainsi être nécessaire pour isoler certaines instances en présence d'outliers. On se limitera volontairement aux nombres d'intervalles correspondant à une fréquence moyenne par intervalle au moins égale à 1. On retombe alors également dans le même nombre de cas évalués que pour MODLEqualFrequency, dont on peut alors réutiliser l'algorithme. Pour garder une complexité algorithmique en O(n.log(n)), il suffit juste de faire attention à l'étape de recherche des index des instances correspondant aux bornes de l'intervalle. Il faut ici rechercher l'index d'une instance à partir de la valeur de sa borne (et non de son rang comme précédemment), ce qui peut se faire par une recherche dichotomique en O(log(n)). Si de façon préliminaire on procède à une discrétisation EqualWidth en n intervalles, les bornes de la recherche dichotomique sont alors restreintes au minimum, ce qui permet de trouver les index des instances bornes des intervalles en O(1), comme dans le cas MODLEqualFrequency.

Au total, la complexité algorithmique de l'algorithme MODLEqualWidth est également en O(n.log(n)).

6.4 Expérimentation

On reprend ici les principales expérimentations réalisées pour la méthode MODL standard, en se contentant d'exposer rapidement les résultats.

6.4.1 Bases UCI

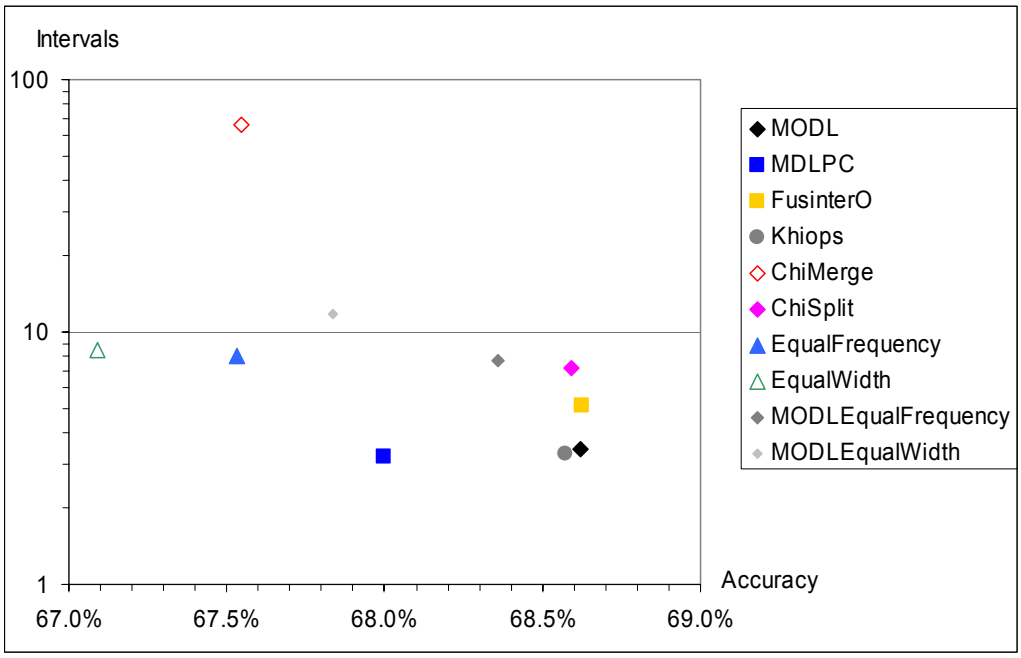


Figure 12 : Evaluation des méthodes de discrétisation pour les critères de taux de bonne prédiction et de taille des discrétisations

Alors que le nombre d'intervalle moyen obtenu avec MODLEqualFrequency et MODLEqualWidth est environ égal à 10, comme pour le paramètre imposé pour EqualFrequency et EqualWidth, on note une amélioration très importante de la qualité prédictive de ces deux méthodes. Les discrétisations en intervalles de fréquence égale s'avèrent clairement plus performantes que les discrétisation en intervalles de largeur égale, et deviennent même compétitives parmi les méthodes de discrétisation supervisée. Ainsi, la performance prédictive moyenne de MODLEqualFrequency est meilleure que celle de MDLPC.

6.4.2 Base France Telecom

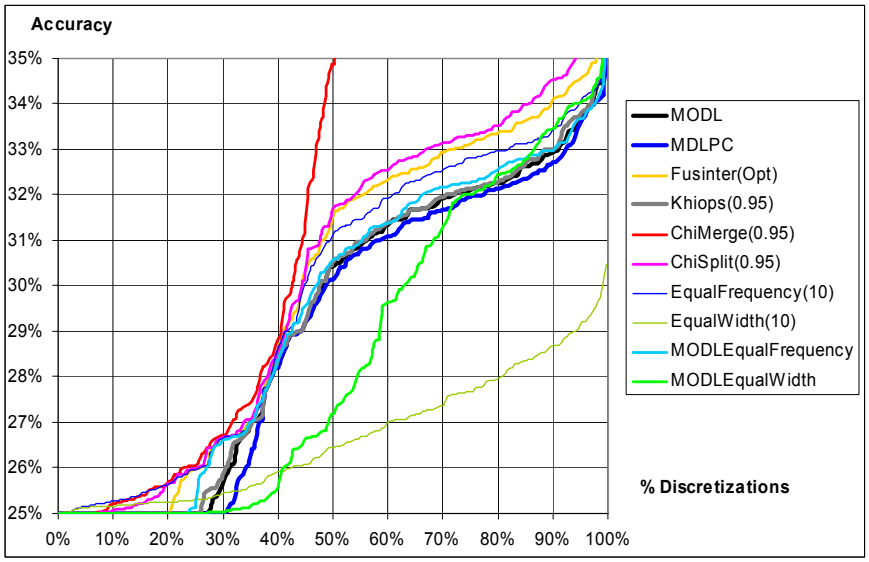


Figure 13 : Fonction de répartition sur 206 attributs des performances prédictives mesurées en apprentissage pour chaque méthode de discrétisation

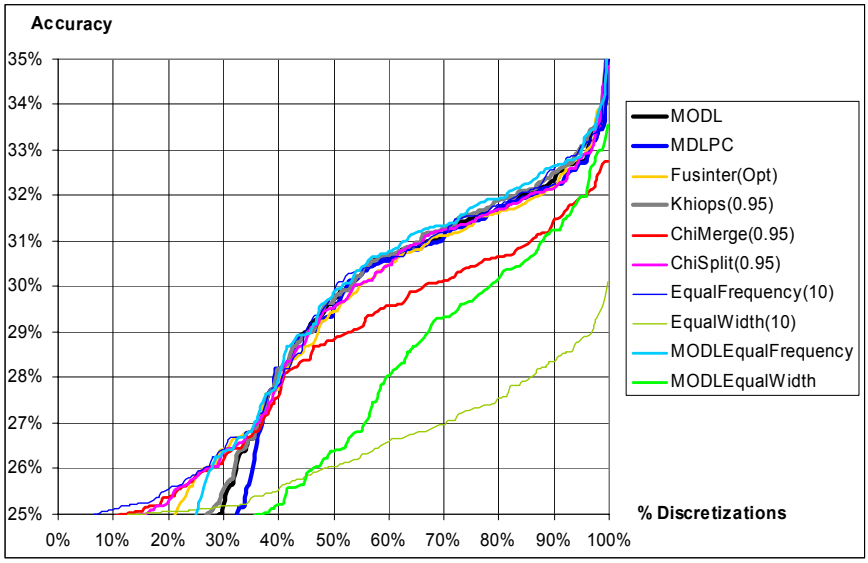


Figure 14 : Fonction de répartition sur 206 attributs des performances prédictives mesurées en test pour chaque méthode de discrétisation

Là encore, l'ajustement automatique du bon nombre d'intervalles par MODLEqualFrequency et MODLEqualWidth améliore significativement la performance des méthodes originales. Notamment, MODLEqualFrequency sur-apprend moins que EqualFrequency(10), et MODLEqualWidth sous-apprend moins que EqualWidth(10). Sur ce jeux d'essai, la méthode MODLEqualFrequency peut être considérée comme la plus performante en taux de bonne prédiction parmi toutes les méthodes testées.

6.4.3 Discrétisation de bruit

On n'a ici évalué que MODLEqualFrequency, qui ne dépend que de l'ordre des valeurs de l'attribut descriptif (alors que MODLEqualWidth dépend des valeurs effectives).

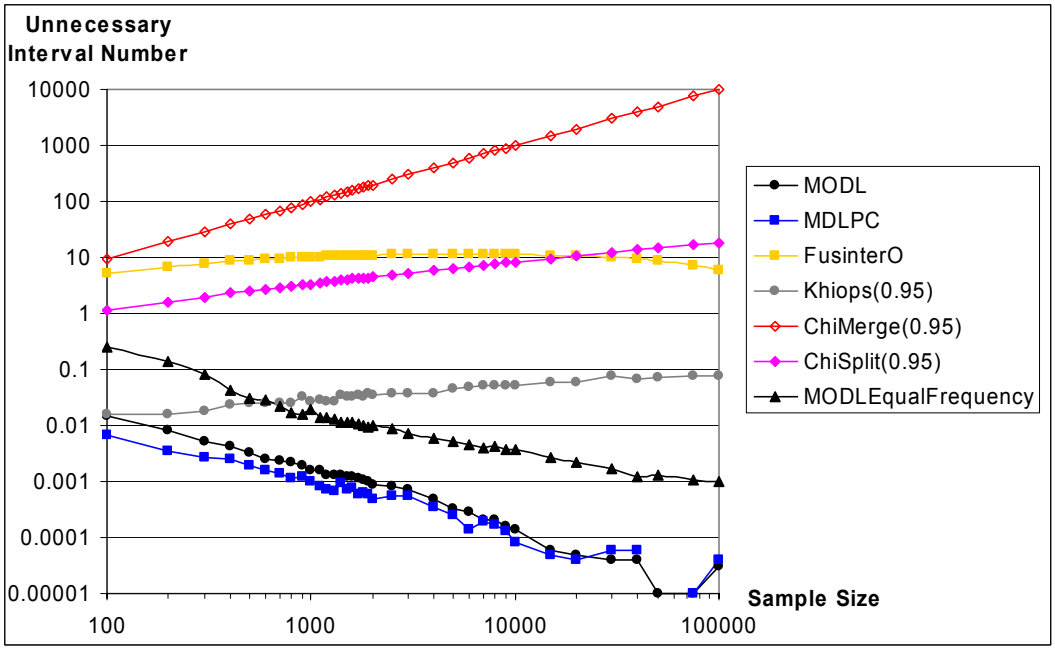


Figure 15 : Nombre moyen d'intervalles surnuméraires lors de la discrétisation d'un attribut indépendant de l'attribut cible

La méthode MODLEqualFrequency résiste très bien au bruit, en produisant asymptotiquement un seul intervalle lors de la discrétisation d'un attribut descriptif indépendant de l'attribut cible. Elle est cependant moins robuste que les méthodes MODL et MDLPC, en produisant un nombre d'intervalles excédentaires de l'ordre de 10 à 100 fois supérieurs (ce qui reste très faible).

7 Annexe: preuves des théorèmes

Soit n le nombre total d'instances, J le nombre de classes.
Soit I le nombre d'intervalles, $n_1, \ldots n_I$ le nombre d'instance de chaque intervalles.
Pour un intervalle i, soit $n_{i,1}, \ldots n_{i,J}$ le nombre d'instances de la classe 1,… J dans l'intervalle i.

7.1 Préliminaires

On va énoncer ci-dessous quelques résultats de combinatoire, qui seront utilisés par la suite.

Lemme 1: Soit une urne contenant n boules indiscernables, et k urnes vides. Le nombre de partitions des n boules dans les k urnes est égal à C_{n+k-1}^{k-1}.

Preuve:
Soit $n_1, n_2, \ldots n_k$ le nombre de boules dans la $k^{\text{ième}}$ urne.
On s'intéresse ici uniquement au nombre de solutions distinctes de $n_1+n_2+ \ldots +n_k = n$.
L'expression précédente montre que ce problème revient au choix de (k-1) signes '+' dans une urne contenant n boules et k-1 signes '+'. Le nombre de répartitions recherché est donc C_{n+k-1}^{k-1}.

On rappelle également un résultat sur la loi multinomiale. Le coefficient multinomial $n!/n_1!n_2!..n_k!$ correspond au nombre de répartitions distinguables de n boules dans k urnes contenant respectivement $n_1, n_2, \ldots n_k$ boules.

Lemme 2: Soit une urne contenant n boules indiscernables, k urnes vides de contenance minimale $min_1, min_2,\ldots,min_k$. On se place dans le cas où $Min = min_1+ \ldots +min_k <= n$. Le nombre de partitions des n boules dans les k urnes est égal à $C_{n-Min+k-1}^{k-1}$.
Preuve:
Soit $n_1, n_2, \ldots n_k$ le nombre de boules dans la $k^{\text{ième}}$ urne.
On s'intéresse ici uniquement au nombre de solutions distinctes de $n_1+n_2+ \ldots +n_k = n$, avec $min_i <= n_i <= n$.
Soit $n'_i = n_i - min_i$.
Soit $n'=n-Min$.
On a alors:
$n'_1+ \ldots +n'_k = n'$
$0 <= n'_i <= n'$
On se ramène alors au problème du lemme 1, ce qui donne un nombre de partitions respectant les contraintes d'effectif minimum par urne de $C_{n-Min+k-1}^{k-1}$.

Corollaire: Dans le cas où toutes les contraintes d'effectif minimum $\min_i$ sont égales à une unique valeur min, le nombre de partitions est égal à $C_{n-k.\min+k-1}^{k-1}$.

7.2 Discrétisation avec a priori a trois étages

Théorème: Un modèle de discrétisation standard M suivant un a priori à trois étages est un modèle de prédiction optimal au sens de Bayes si son évaluation par la formule suivante est minimale sur l'ensemble de tous les modèles:

$$Value(M) = \log(n) + \log\left(C_{n+I-1}^{I-1}\right) + \sum_{i=1}^{I} \log\left(C_{n_i+J-1}^{J-1}\right) + \sum_{i=1}^{I} \log\left(n_i!/n_{i,1}!n_{i,2}!..n_{i,J}!\right)$$

Preuve:
La probabilité a priori d'un modèle de discretization M peut être décomposée sur l'ensemble de des paramètres du modèle $\{ I, \{n_i\}_{1\le i\le I}, \{n_{ij}\}_{1\le i\le I,\ 1\le j\le J} \}$.

Introduisons quelques notations:
- $p(I)$: probabilité a priori du nombre d'intervalles I ,
- $p(\{n_i\})$: probabilité a priori des paramètres $\{n_1,...,n_I\}$,
- $p(n_i)$: probabilité a priori du paramètre n_i ,
- $p(\{n_{ij}\})$: probabilité a priori des paramètres $\{n_{11},...,n_{ij},...,n_{IJ}\}$,
- $p(\{n_{ij}\}_i)$: probabilité a priori des paramètres $\{n_{i1},...,n_{iJ}\}$.

L'objectif est de trouver le modèle de discrétisation M qui maximise la probabilité $p(M/S)$ pour une chaîne S données de classes cibles. En utilisant la formule de Bayes et en remarquant que la probabilité $p(S)$ est constante quelque soit le modèle, cela revient à maximiser $p(M)p(S/M)$.

Dans un premier temps, on va calculer la probabilité a priori $p(M)$ du modèle. On a

$$p(M) = p(I, \{n_i\}, \{n_{ij}\})$$
$$= p(I)\,p(\{n_i\}/I)\,p(\{n_{ij}\}/I, \{n_i\}).$$

La première hypothèse de l'a priori à trois étages est que le nombre d'intervalles est uniformément distribué entre 1 et n. On obtient donc

$$p(I) = \frac{1}{n}.$$

Selon la seconde hypothèse, toutes les divisions de S en I intervalles sont équiprobables pour un I donné. Evaluer la probabilité d'un ensemble d'intervalles revient au calcul combinatoire du nombre des ensembles d'intervalles possibles. Diviser la chaîne S en I intervalles est équivalent à décomposer un entier n comme la somme des effectifs n_i des intervalles. En utilisant la combinatoire, on montre que le nombre de choix des ensembles de fréquences $\{n_i\}_{1\le i\le I}$ est égal à C_{n+I-1}^{I-1}. On obtient alors

$$p(\{n_i\}/I) = \frac{1}{C_{n+I-1}^{I-1}}.$$

Le dernier terme à évaluer peut être réécrit comme un produit en utilisant l'hypothèse d'indépendance des distributions de classes cibles entre les intervalles. On a

$$\begin{aligned} p(\{n_{ij}\}/I, \{n_i\}) &= p(\{n_{ij}\}_1, \{n_{ij}\}_2, ..., \{n_{ij}\}_I / I, \{n_i\}) \\ &= \prod_{i=1}^{I} p(\{n_{ij}\}_i / I, \{n_i\}) \\ &= \prod_{i=1}^{I} p(\{n_{ij}\}_i / n_i). \end{aligned}$$

Pour un intervalle i donné d'effectif n_i, toutes les distributions de classes cibles sont équiprobables. Le calcul de la probabilité d'une distribution est à nouveau un problème combinatoire dont la solution est

$$p(\{n_{ij}\}/n_i) = \frac{1}{C_{n_i+J-1}^{J-1}}.$$

Donc,

$$p(\{n_{ij}\}/I, \{n_i\}) = \prod_{i=1}^{I} \frac{1}{C_{n_i+J-1}^{J-1}}.$$

La probabilité a priori d'un modèle est alors

$$p(M) = \frac{1}{n} \frac{1}{C_{n+I-1}^{I-1}} \prod_{i=1}^{I} \frac{1}{C_{n_i+J-1}^{J-1}}.$$

Evaluons maintenant la probabilité d'observer une chaîne S pour une modèle M donné. On découpe d'abord la chaîne S en I sous-chaînes S_i d'effectif n_i puis on utilise à nouveau l'hypothèse d'indépendance entre les intervalles. On obtient

$$\begin{aligned} p(S/M) &= p(S/I, \{n_i\}, \{n_{ij}\}) \\ &= p(S_1, S_2, ..., S_I / I, \{n_i\}, \{n_{ij}\}) \\ &= \prod_{i=1}^{I} p(S_i / I, \{n_i\}, \{n_{ij}\}) \\ &= \prod_{i=1}^{I} \frac{1}{(n_i! / n_{i,1}! n_{i,2}! ... n_{i,J}!)}, \end{aligned}$$

car l'évaluation de la probabilité d'observer une sous-chaîne S_i avec un a priori uniforme correspond à la distribution multinomiale.

En prenant les opposés des logarithmes des probabilités, le problème de maximisation se transforme en problème de minimisation pour le critère du théorème

$$Value(M) = \log(n) + \log(C_{n+I-1}^{I-1}) + \sum_{i=1}^{I} \log(C_{n_i+J-1}^{J-1}) + \sum_{i=1}^{I} \log(n_i! / n_{i,1}! n_{i,2}! ... n_{i,J}!).$$

Remarque: si l'on se base sur une variante d'a priori on l'on impose que chaque intervalle ait un effectif minimum égal à Min, il suffit dans l'expression précédente de remplacer le premier terme par $\log(C_{n-I.Min+I-1}^{I-1})$.

Théorème: Dans un modèle SDM optimal suivant un a priori à trois étages il ne peut y avoir de point de coupure entre deux instances ayant même classe cible.

Preuve:

Supposons au contraire qu'il y ait un tel point de coupure entre deux instances de même classe cible.

Soient A1 et B1 les intervalles correspondants. On suppose, sans perte de généralité, que la classe cible concernée est la classe d'index 1.

On peut construire deux intervalles alternatifs A0 et B2 en déplaçant la dernière instance de A1 vers B1.

La variation de coût de discrétisation est dans ce cas:

$$\begin{aligned}\Delta Cost1 &= \log\left((n_{A0}+J-1)!/n_{A0}!(J-1)!\right)+\log\left(n_{A0}!/n_{A0,1}!n_{A0,2}!...n_{A0,J}!\right)\\ &+\log\left((n_{B2}+J-1)!/n_{B2}!(J-1)!\right)+\log\left(n_{B2}!/n_{B2,1}!n_{B2,2}!...n_{B2,J}!\right)\\ &-\log\left((n_{A1}+J-1)!/n_{A1}!(J-1)!\right)-\log\left(n_{A1}!/n_{A1,1}!n_{A1,2}!...n_{A1,J}!\right)\\ &-\log\left((n_{B1}+J-1)!/n_{B1}!(J-1)!\right)-\log\left(n_{B1}!/n_{B1,1}!n_{B1,2}!...n_{B1,J}!\right)\end{aligned}$$

Les effectifs par classe cible sont les mêmes, sauf pour la classe cible d'index 1.

$$\Delta Cost1 = \log\left((n_{A1}+J-2)!/(n_{A1}+J-1)!\right)+\log\left((n_{B1}+J)!/(n_{B1}+J-1)!\right)+\log\left(n_{A1,1}!/(n_{A1,1}-1)!\right)+\log\left(n_{B1,1}!/(n_{B1,1}+1)!\right)$$

$$\Delta Cost1 = \log\left((n_{B1}+J)/(n_{A1}+J-1)\right)+\log\left(n_{A1,1}/(n_{B1,1}+1)\right)$$

$$\Delta Cost1 = \log\left((n_{B1}+1)/(n_{B1,1}+1)\right)-\log\left(n_{A1}/n_{A1,1}\right)$$
$$+\log\left((n_{B1}+J)/(n_{B1}+1)\right)-\log\left((n_{A1}+J-1)/n_{A1}\right)$$

Pour $1 \le x < y$, $(y+1)/(x+1) < y/x$

Donc $\Delta Cost1 < \log\left(n_{B1}/n_{B1,1}\right)-\log\left(n_{A1}/n_{A1,1}\right)+\log\left((n_{B1}+J)/(n_{B1}+1)\right)-\log\left((n_{A1}+J)/(n_{A1}+1)\right)$

De même, on peut construire deux intervalles alternatifs A2 et B0 en déplaçant la première instance de B1 vers A1.

$$\Delta Cost2 < \log\left(n_{A1}/n_{A1,1}\right)-\log\left(n_{B1}/n_{B1,1}\right)+\log\left((n_{A1}+J)/(n_{A1}+1)\right)-\log\left((n_{B1}+J)/(n_{B1}+1)\right)$$

On remarque que les bornes de $\Delta Cost1$ et $\Delta Cost2$ sont exactement l'opposée l'une de l'autre.

Donc, la variation de coût est strictement négative quand on fait passer une instance de A1 vers B1 ou de B1 vers A1, ce qui rend impossible une coupure entre deux instances de même classe cible dans la solution de coût minimal.

Théorème: Dans un modèle SDM optimal suivant un a priori à trois étages, il ne peut exister une discrétisation comportant deux intervalles singletons successifs.

Preuve:

Soient deux intervalles successifs A et B, AuB l'intervalle résultant de la fusion de A et B.

Soient n_A, n_B et n_{AUB} les effectifs de ces intervalles.

Soient $n_{A,j}$, $n_{A,j}$ et $n_{AUB,j}$ les effectifs des ces intervalles par classe cible.

Calculons la variation du coût de discrétisation suite à la fusion de A et B, faisant passer le nombre d'intervalles de I à I-1.

$$\begin{aligned}\Delta Cost &= \log\left(C_{n+I-2}^{I-2}\right)-\log\left(C_{n+I-1}^{I-1}\right)+\left(\log\left(C_{n_{A\cup B}+J-1}^{J-1}\right)+\log\left(n_{A\cup B}!/n_{A\cup B,1}!n_{A\cup B,2}!...n_{A\cup B,J}!\right)\right)\\ &-\left(\log\left(C_{n_A+J-1}^{J-1}\right)+\log\left(n_A!/n_{A,1}!n_{A,2}!...n_{A,J}!\right)\right)-\left(\log\left(C_{n_B+J-1}^{J-1}\right)+\log\left(n_B!/n_{B,1}!n_{B,2}!...n_{B,J}!\right)\right)\end{aligned}$$

$$\Delta Cost = \log\left(I-1/n+I-1\right)+\log\left((n_{A\cup B}+J-1)!(J-1)!/(n_A+J-1)!(n_B+J-1)!\right)-\sum_{j=1}^{J}\log\left(C_{n_{A\cup B,j}}^{n_{A,j}}\right)$$

On a ici $n_A = n_B = 1$ et $n_{AUB} = 2$.

$$\Delta Cost = \log\left(I-1/n+I-1\right)+\log\left(J+1/J\right)$$

$$\Delta Cost \le 0 \Leftrightarrow I \le nJ+1$$

Donc la variation du coût est toujours négative suite à la fusion de deux intervalles.

Une discrétisation de coût minimal ne peut donc pas contenir deux intervalles singletons successifs.

Théorème: On se place ici dans le cas où le nombre de classes cibles est égal à deux. Dans un modèle de discrétisation SDM optimal suivant un a priori à trois étages, la longueur maximale d'un intervalle extrême (décomposable en deux intervalles purs de même taille) est définie par l'équation suivante:

$$L - 3/2\log_2(L) = \log_2(n/I) + O(1)$$

Preuve:

On reprend les même notations que pour le théorème précédent, et on repart du calcul de la variation de coût de la discrétisation.

Soit l=L/2.

On a ici J=2, $n_A = n_B = l$, n_{AUB} =2l.

$$\Delta Cost = \log\left(I-1/n+I-1\right)+\log\left((2l+1)!/(l+1)!(l+1)!\right)$$

En utilisant la formule de Stirling, on a:

$$\log\left(C_{2l}^{l}\right) = 2l.\log(2)-1/2\log(l)+O(1)$$

$$\Delta Cost = -\log(n/I)+2l.\log(2)-3/2\log(2l)+O(1)$$

$$\Delta Cost \le 0 \Leftrightarrow L-3/2\log_2(L) \le \log_2(n/I)+O(1)$$

7.3 Discrétisation avec a priori à deux étages du premier type

Théorème: Un modèle de discrétisation standard *M* suivant un a priori à deux étages du premier type est un modèle de prédiction optimal au sens de Bayes si son évaluation par la formule suivante est minimale sur l'ensemble de tous les modèles:

$$Value(M) = \log(n) + \log\left(C_{n+I.J-1}^{I.J-1}\right) + \sum_{i=1}^{I} \log\left(n_i!/n_{i,1}!n_{i,2}!...n_{i,J}!\right)$$

Preuve:

On utilise exactement le même type de preuve que dans le cas de l'a priori à trois étages. La variante est ici dans le choix du codage du modèle. Ici, toutes les partitions en intervalles et toutes les distributions de symboles pour les intervalles correspondants sont équiprobables, pour un nombre d'intervalle *I* donné. Choisir un modèle revient alors à choisir l'ensemble des valeurs n_{ij} vérifiant les conditions suivantes:

$$\sum_{i,j} n_{i,j} = n$$

$$\forall i, j, 0 \le n_{i,j} \le n$$

Il s'agit donc du nombre de répartitions des *n* boules d'une urne dans *I.J* urnes, ce qui donne $C_{n+I.J-1}^{I.J-1}$ possibilités.

D'où le critère d'évaluation du théorème.

Théorème: Dans un modèle SDM optimal suivant un a priori à deux étages du premier type, il ne peut y avoir de point de coupure entre deux instances ayant même classe cible.

Preuve:

Supposons au contraire qu'il y ait un tel point de coupure entre deux instances de même classe cible.

Soient A1 et B1 les intervalles correspondants. On suppose, sans perte de généralité, que la classe cible concernée est la classe d'index 1.

On peut construire deux intervalles alternatifs A0 et B2 en déplaçant la dernière instance de A1 vers B1.

La variation de coût de discrétisation est dans ce cas:

$$\Delta Cost1 = \log\left(n_{A0}!/n_{A0,1}!n_{A0,2}!...n_{A0,J}!\right) + \log\left(n_{B2}!/n_{B2,1}!n_{B2,2}!...n_{B2,J}!\right)$$
$$- \log\left(n_{A1}!/n_{A1,1}!n_{A1,2}!...n_{A1,J}!\right) - \log\left(n_{B1}!/n_{B1,1}!n_{B1,2}!...n_{B1,J}!\right)$$

Les effectifs par classe cible sont les mêmes, sauf pour la classe cible d'index 1.

$$\Delta Cost1 = \log\left((n_{A1}-1)!/(n_{A1,1}-1)!\right) + \log\left((n_{B1}+1)!/(n_{B1,1}+1)!\right) - \log\left(n_{A1}!/n_{A1,1}!\right) - \log\left(n_{B1}!/n_{B1,1}!\right)$$

$$\Delta Cost1 = \log\left((n_{B1}+1)/(n_{B1,1}+1)\right) - \log\left(n_{A1}/n_{A1,1}\right)$$

De même, on peut construire deux intervalles alternatifs A2 et B0 en déplaçant la première instance de B1 vers A1.

La variation de coût de discrétisation est dans ce cas:

$$\Delta Cost2 = \log\left((n_{A1}+1)/(n_{A1,1}+1)\right) - \log\left(n_{B1}/n_{B1,1}\right)$$

Pour $1 \le x \le y$, $(y+1)/(x+1) \le y/x$

Donc $\Delta Cost1 \le \log\left(n_{B1}/n_{B1,1}\right) - \log\left(n_{A1}/n_{A1,1}\right)$

$$\Delta Cost2 \le \log\left(n_{A1}/n_{A1,1}\right) - \log\left(n_{B1}/n_{B1,1}\right)$$

On remarque que les bornes de $\Delta Cost1$ et $\Delta Cost2$ sont exactement l'opposée l'une de l'autre.

Donc, la variation de coût est négative quand on fait passer une instance de A1 vers B1 ou de B1 vers A1, ce qui rend impossible une coupure entre deux instances de même classe cible.

Théorème: On se place ici dans le cas où le nombre de classes cibles est égal à deux. Dans un modèle SDM optimal suivant un a priori à deux étages du premier type, il ne peut exister une discrétisation comportant deux intervalles singletons successifs.

Preuve:

Soient deux intervalles successifs A et B, AuB l'intervalle résultant de la fusion de A et B.

Soient n_A, n_B et n_{AUB} les effectifs de ces intervalles.

Soient $n_{A,j}$, $n_{A,j}$ et $n_{AUB,j}$ les effectifs des ces intervalles par classe cible.

Calculons la variation du coût de discrétisation suite à la fusion de A et B, faisant passer le nombre d'intervalles de *I*+1 à *I*.

$$\Delta Cost = \log\left(C_{n+I.J-1}^{I.J-1}\right) - \log\left(C_{n+I.J+J-1}^{I.J+J-1}\right)$$
$$+ \log\left(n_{A\cup B}!/n_{A\cup B,1}!n_{A\cup B,2}!...n_{A\cup B,J}!\right) - \log\left(n_A!/n_{A,1}!n_{A,2}!...n_{A,J}!\right) - \log\left(n_B!/n_{B,1}!n_{B,2}!...n_{B,J}!\right)$$

$$\Delta Cost = \log\left((n+I.J-1)!(I.J+J-1)!/(n+I.J+J-1)!(I.J-1)!\right) + \log\left(C_{n_{A\cup B}}^{n_A}\right) - \sum_{j=1}^{J} \log\left(C_{n_{A\cup B,j}}^{n_{A,j}}\right)$$

On a ici *J*=2, $n_A = n_B = 1$ et $n_{AUB} = 2$.

$$\Delta Cost = \log\left((2I+1)2I/(n+2I+1)(n+2I)\right) + \log\left(C_2^1\right)$$
$$\Delta Cost \le 0 \Leftrightarrow (2I+1)4I/(n+2I+1)(n+2I) \le 1$$
$$\Delta Cost \le 0 \Leftrightarrow 4I^2 + 2I(1-2n) - n(n+1) \le 0$$

Les racines de cette équation sont $I_1 = \left(2n-1-\sqrt{8n^2+1}\right)\Big/4$ et $I_2 = \left(2n-1+\sqrt{8n^2+1}\right)\Big/4$

On vérifie que $I_1 \le 0$ et $N \le I_2$.

Donc la variation du coût est toujours négative suite à la fusion de deux intervalles.

Une discrétisation de longueur de codage A minimal ne peut donc pas contenir deux intervalles singletons successifs.

Théorème: On se place ici dans le cas où le nombre de classes cibles est égal à deux. Dans un modèle SDM optimal suivant un a priori à deux étages du premier type, la longueur maximale d'un intervalle extrême (décomposable en deux intervalles purs de même taille) est définie par l'équation suivante:

$$L - 1/2\log_2(L) = 2\log_2(n/I) + O(1)$$

Preuve:

On reprend les même notations que pour le théorème précédent, et on repart du calcul de la variation de coût de la discrétisation.

Soit $l=L/2$.

On a ici $J=2$, $n_A = n_B = l$, $n_{AUB} = 2l$.

$$\Delta Cost = \log\left((2I+1)2I/(n+2I+1)(n+2I)\right) + \log\left(C_{2l}^{l}\right)$$

En utilisant la formule de Stirling, on a:

$$\log\left(C_{2l}^{l}\right) = 2l.\log(2) - 1/2\log(l) + O(1)$$
$$\Delta Cost = -2\log(n/I) + 2l.\log(2) - 1/2\log(l) + O(1)$$
$$\Delta Cost \le 0 \Leftrightarrow L - 1/2\log_2(L) \le 2\log_2(n/I) + O(1)$$

7.4 Discrétisation avec a priori à deux étages du second type

Théorème: Un modèle de discrétisation standard M suivant un a priori à deux étages du second type est un modèle de prédiction optimal au sens de Bayes si son évaluation par la formule suivante est minimale sur l'ensemble de tous les modèles:

$$Value(M) = \sum_{i=1}^{I} \log\left(C_{n_i+J-1}^{J-1}\right) + \sum_{i=1}^{I} \log\left(n_i!/n_{i,1}!n_{i,2}!...n_{i,J}!\right)$$

Preuve:

On utilise exactement le même type de preuve que dans le cas de l'a priori à trois étages. La variante est ici dans le choix du codage du modèle. Ici, toutes les partitions en intervalles sont équiprobables quel que soit le choix du nombre d'intervalle I. Ce terme est par conséquent constant pour le problème d'optimisation de la discrétisation, et peut être retiré du critère d'évaluation. D'où le critère d'évaluation du théorème, qui ne comprend plus que le choix de la distribution des symboles dans chaque intervalle pour la partie codage du modèle.

Théorème: Dans un modèle SDM optimal suivant un a priori à deux étages du second type, il ne peut y avoir de point de coupure entre deux instances ayant même classe cible.

Preuve:

La preuve est exactement la même que pour l'a priori à trois étages. En effet, à nombre d'intervalle constant, la variation de coût est exactement la même.

Théorème: Dans un modèle SDM optimal suivant un a priori à deux étages du second type, il ne peut y avoir d'intervalle séparable en deux intervalles purs.

Preuve:

Soient deux intervalles successifs A et B, AuB l'intervalle résultant de la fusion de A et B.

Soient n_A, n_B et n_{AUB} les effectifs de ces intervalles.

Soient $n_{A,j}$, $n_{A,j}$ et $n_{AUB,j}$ les effectifs des ces intervalles par classe cible.

Calculons la variation du coût de discrétisation suite à la fusion de A et B, faisant passer le nombre d'intervalles de I+1 à I.

$$\Delta Cost = \left(\log\left(C_{n_{A\cup B}+J-1}^{J-1}\right) + \log\left(n_{A\cup B}!/n_{A\cup B,1}!n_{A\cup B,2}!...n_{A\cup B,J}!\right)\right)$$
$$-\left(\log\left(C_{n_A+J-1}^{J-1}\right) + \log\left(n_A!/n_{A,1}!n_{A,2}!...n_{A,J}!\right)\right) - \left(\log\left(C_{n_B+J-1}^{J-1}\right) + \log\left(n_B!/n_{B,1}!n_{B,2}!...n_{B,J}!\right)\right)$$

$$\Delta Cost = \log\left((n_{A\cup B}+J-1)!(J-1)!/(n_A+J-1)!(n_B+J-1)!\right) - \sum_{j=1}^{J}\log\left(C_{n_{A\cup B,j}}^{n_{A,j}}\right)$$

Si les intervalles A et B sont purs vis à vis des classes cibles, le terme de droite disparaît.

$$\Delta Cost = \log\left((n_{A\cup B}+J-1)!(J-1)!/(n_A+J-1)!(n_B+J-1)!\right)$$

$$\Delta Cost = \log\left(C_{n_{A\cup B}+2J-2}^{n_A+J-1} \Big/ C_{n_{A\cup B}+2J-2}^{J-1}\right) = \log\left(C_{n_{A\cup B}+2J-2}^{n_B+J-1} \Big/ C_{n_{A\cup B}+2J-2}^{J-1}\right)$$

Les termes du monôme étant plus grand vers le centre que vers les bords, la variation du coût est toujours positive suite à la fusion de deux intervalles purs.

Théorème: La quantité d'information contenue dans le choix d'un modèle SDM suivant un a priori à deux étages du second type est supérieure à la taille de la chaîne *S*.

Preuve:

La preuve est ici analogue à la preuve d'optimalité du critère d'évaluation.

On repart de l'évaluation de la probabilité a priori d'un modèle $p(M) = p\left(I, \{n_i\}, \{n_{ij}\}\right)$.

$p(M) = p\left(I, \{n_i\}\right) p\left(\{n_{ij}\} / I, \{n_i\}\right)$.

Ici, toutes les partitions en intervalles sont équiprobables quel que soit le nombre d'intervalles *I*.

Le nombre de partition en *I* intervalles est égal à C_{n+I-1}^{I-1}.

Le nombre total de partition en intervalles est égal à $NbPartitions = \sum_{I=1}^{n} C_{n+I-1}^{I-1}$.

Montrons que $NbPartitions = C_{2n}^{n-1}$.

En se basant sur la formule du triangle de Pascal $C_n^k = C_{n-1}^k + C_{n-1}^{k-1}$, on a:

$C_{2n}^{n-1} = C_{2n-1}^{n-1} + C_{2n-1}^{n-2}$

$C_{2n}^{n-1} = C_{2n-1}^{n-1} + C_{2n-2}^{n-2} + C_{2n-2}^{n-3}$

…

Par récurrence, on obtient le résultat annoncé, à savoir $C_{2n}^{n-1} = \sum_{I=1}^{n} C_{n+I-1}^{I-1}$.

Toutes les partitions étant équiprobables, on a $p(M) = p\left(1/NbPartitions\right) p\left(\{n_{ij}\} / I, \{n_i\}\right)$.

La quantité d'information est égale à l'opposé du logarithme base 2 de la probabilité. Donc la quantité d'information dans le choix d'un modèle est ici supérieure à $\log_2\left(C_{2n}^{n-1}\right)$.

En utilisant la formule de Stirling, on a:

$\log\left(C_{2n}^{n}\right) = 2n.\log(2) - 1/2\log(n) + O(1)$

$\log_2\left(C_{2n}^{n-1}\right) = \log_2\left(C_{2n}^{n}\, n^2/(n-1)(n+1)\right) = 2n - 1/2\log_2(n) + O(1)$

La quantité d'information dans le choix du modèle est donc beaucoup plus importante que la longueur de la chaîne *S*.

7.5 Discrétisation avec a priori uniforme

Théorème: Un modèle de discrétisation standard *M* suivant un a priori uniforme est un modèle de prédiction optimal au sens de Bayes si son évaluation par la formule suivante est minimale sur l'ensemble de tous les modèles:

$$Value(M) = \sum_{i=1}^{I} \log\left(n_i! / n_{i,1}! n_{i,2}! ... n_{i,J}!\right)$$

Preuve:

On utilise exactement le même type de preuve que dans le cas de l'a priori à trois étages. La variante est ici dans le choix du codage du modèle. Ici, tous les modèles étant équiprobables, le terme correspond au codage du modèle est par conséquent constant pour le problème d'optimisation de la discrétisation, et peut être retiré du critère d'évaluation. D'où le critère d'évaluation du théorème, qui ne comprend plus que la partie codage des exceptions au modèle.