Sélection automatique de variables pertinentes

Pierre Dangauthier

Sélection automatique de variables pertinentes

Vers la découverte de nouvelles modalités sensori-motrices.

Encadrants :
Pierre Bessière
Anne Spalanzani

Pierre Dangauthier
DEA Image Vision Robotique
26 avril 2005

Résumé

Pour être qualifié d'autonome, un robot mobile doit être capable d'évoluer dans un environnement non contrôlé, changeant et sur lequel il ne dispose que d'informations incertaines. La *programmation bayésienne des robots* est adaptée à ce problème, et dans ce cadre, notre objectif consiste à **automatiser la découverte de nouveaux comportements**. Nous désirons que le robot trouve par lui-même des corrélations entre ses variables sensori-motrices afin d'apprendre à résoudre sa tâche d'une nouvelle manière. Notre approche consiste à proposer et à comparer différentes méthodes pour trouver ces corrélations. Nous validerons les méthodes sur des données issues d'un simulateur logiciel et des données réelles provenant d'un petit robot autonome : Le Koala.

Abstract

An autonomous mobile robot should be able to evoluate in an uncontrolled environment. Therefore its knowledge about the world is fuzzy and incomplete. Bayesian programming is designed to tackle this problem. In this context, our goal is to able the robot to **find out new behaviors** automatically and autonomously by finding correlations in his sensors set. By a analysing of these correlations, it will be able to complete its task in different ways. Our approach consists in a comparison of several feature selection methods tuned for this issue. We will validate our solutions on simulated and real robotic data sets.

Remerciements

Je remercie tout d'abord mes deux co-tuteurs, Pierre Bessière et Anne Spalanzani pour leur aide technique et leur réelle sympathie. Leurs conseils et leurs idées ont été crutiaux pour mener ce stage à son terme.

Je tiens aussi à adresser ma gratitude aux membres de mon jury : Sabine Coquillart, Augustin Lux et plus particulièrement à M.Roland Siegwart pour m'avoir fait l'honneur de venir de si loin afin d'assister à ma soutenance.

Mes remerciements vont également à J. Crowley, A. Lux et bien sûr à Anne Pierson pour leur travail dans l'organisation du cursus du DEA Image Vision Robotique.

Je n'oublerai pas Guillaume Gillet qui a travaillé pour me permettre de relever des données sur le robot Koala.

Finalement, je tiens à manifester ma sympathie envers toute l'équipe du laboratoire CYBERMOVE, qui a su entretenir une atmosphère de travail conviviale durant ces quelques mois de stage.

Table des matières

Chapitre 1

Introduction

1.1 Cadre général

Ce projet s'inscrit dans le thème de recherche du proceljet CyberMove, laboratoire Gravir de l'INRIA Rhône-Alpes. Il s'agit d'étudier les systèmes sensori-moteurs du point de vue de l'**inférence** et de **l'apprentissage bayésien**. Les systèmes sensori-moteurs qu'ils soient des animaux, des humains, des robots autonomes ou des personnages virtuels, sont destinés à évoluer dans un environnement non contrôlé. Cela signifie que les informations qu'ils possèdent sur leur environnement sont perpétuellement changeantes, incomplètes et parfois inconsistantes. Alors le modèle qu'ils construisent de leur environnement sera nécessairement incomplet (mais "modèle" et "complet" ne sont-ils pas deux termes antinomiques ?).

Il apparaît alors qu'une approche basée sur une programmation déterministe classique pour créer des artefacts évoluant dans de tels environnements sera rapidement mise en échec. La solution proposée par les chercheurs de CyberMove est de transformer l'**incomplétude** du modèle en **incertitude** en modélisant le monde grâce aux **probabilités bayésiennes**. Il sera ensuite possible de prendre des décisions motrices en utilisant la théorie des probabilités comme une extension de la logique. Cette approche est nommée : **programmation bayésienne des robots**.

1.2 Contexte du projet

1.2.1 Objectifs

Dans ce cadre, notre objectif consiste à **automatiser la découverte de nouveaux comportements**. Nous voulons que notre artefact puissent par lui-même trouver des corrélations dans son domaine sensori-moteur, et qu'il les exploite pour apprendre de nouvelles façons d'interagir avec son environnement et de réaliser sa tâche.

Par exemple, l'homme sait se focaliser sur les signaux sensori-moteurs utiles pour accomplir une action. Il ne prend pas en compte toutes ses informations sensorielles et il ne monopolise plus que certains de ses sens. Par exemple, on ne regarde plus le clavier pour taper du texte : on a fait progressivement un transfert de la vue vers le toucher et la localisation spatiale. Cette découverte d'une nouvelle manière de résoudre le problème à été apprise lors d'un processus de transfert d'information d'une modalité sensorielle à

une autre. L'apprenti informaticien a compris qu'il y avait une corrélation entre l'image visuelle du clavier et du doigt, et sa sensation de touché. On aimerait faire de même avec un robot, on aimerait faire émerger de nouvelles manières de résoudre un problème.

Dans ce travail, nous avons choisi de nous intéresser au cas dit "*de la balle rouge*". Le comportement préxistant est le suivi par un robot équipé d'une caméra mobile d'une balle rouge qui se déplace dans son champ de vision. Le robot connaît en permanance l'angle θ formé entre son axe pricipal et la balle. Mais ce robot est aussi équipé d'un grand nombre de capteurs en tout genre, dont certains comme les proximètres apportent aussi de l'information sur θ. Nous voulons que le robot puisse déterminer par lui-même quels sont les capteurs qui sont pertinents pour cette tâche de suivi. Ainsi il pourra se passer de la caméra et suivre d'autres objets, un cube vert par exemple (on peut supposer que le programme de suivi vidéo était spécifique aux objets sphériques de couleur rouge).

1.2.2 La programmation Bayésienne des robots

Dans toute la suite, nous utiliserons les conventions suivantes : $\forall i \in [0 \dots N-1]$ X_i est une variable qui peut prendre C valeurs $x_{i,k}$, $\forall k \in [0 \dots C-1]$. On notera $P(X_i = x_{i,k})$ la probabilité de l'évènement $X_i = x_{i,k}$. Quand il n'y a pas d'ambiguïté, on la notera simplement $P(X_i)$. Par exemple on écrira :

$$\sum_{k=0}^{k=C} P(X_i = x_{i,k}) = \sum_{X_i} P(X_i) = 1$$

.

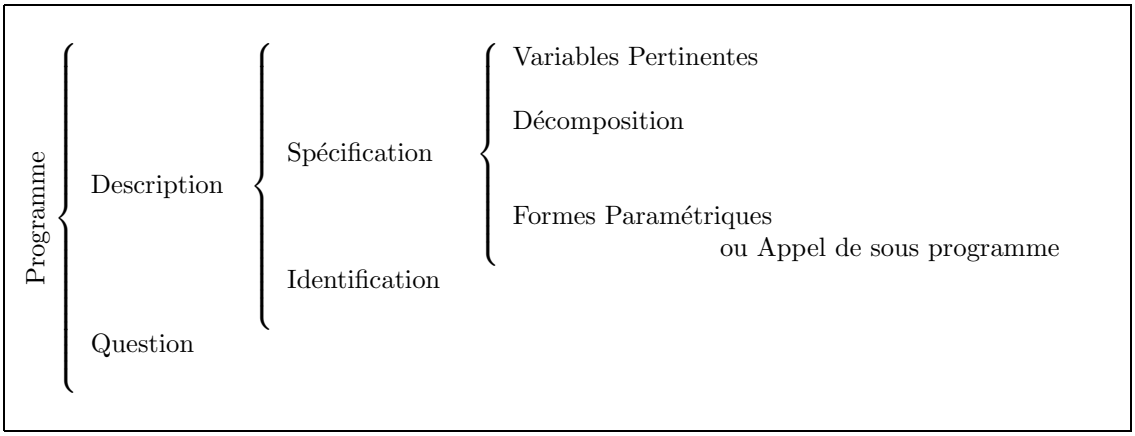

Fig. 1.1: Structure d'un Programme Bayésien

Pour faire de la programmation bayésienne, et conformément à la vision bayésienne des probabilités, le programmeur commence par exprimer explicitement toutes ses connaissances *a priori*. C'est la phase de **description**, d'apport des **connaisssances préalables**.

Ensuite on pose une **question** qui est par exemple de la forme : "Quelle est la probabilité pour que $X_i = x_{i,k}$ sachant que $X_{j \neq i} = x_{j,k_j}$?".

Pour décrire le programme, on commence par spécifier la forme de la distribution de probabilité conjointe $P(X_0 \dots X_{N-1})$ en fonction de ses paramètres, puis on **identifie** ces paramètres par un apprentissage sur des exemples.

La **spécification** se décompose en : le **choix des variables pertinentes** $X_0 \dots X_{N-1}$, le choix d'une **décomposition** où le programmeur introduit ses *a priori* sur les indépendances conditionnelles entre variables, et le choix des **formes paramétriques** pour les distributions partielles.

Pour plus de détails sur les programmes bayésiens, on peut se référer à l'article [LOE03] et à la thèse [O.99].

1.3 Problème

Notre robot est programmé avec le paradigme bayésien pour réaliser une tâche particulière. Il possède un grand nombre de capteurs (variables) et nous voulons qu'il détermine lesquelles de ces variables sont corrélées avec la tâche courante. Nous supposons ici que cette tâche se ramène à l'estimation d'une valeur. Par exemple, pour le suivi de balle rouge, la tâche se ramène à l'estimation de sa direction (θ). Cette hypothèse est peu restrictive car de nombreux problèmes équivalent à l'estimation d'une ou plusieurs grandeures.

Déterminer les variables corrélées avec la tâche courrante revient alors à déterminer les variables permettant une bonne prédiction de θ. Comme nous travaillons sur des variables discrètes, faire une estimation consiste à classer une sensation (*i.e.* un ensemble de valeurs des capteurs) parmi les l classes $\theta_1 \dots \theta_l$ possibles pour la variable θ.

Ainsi pour le suivi de balle rouge, on veut trouver le plus petit nombre de capteurs qui permettent de prévoir θ.

Dans le cadre de la programmation bayésienne, cela revient à déterminer une partie des connaissances préalables au problème, connaissances habituellement données par le programmeur. On veut donc automatiser la sélection de variables pertinentes.

Par conséquent, il s'agit de **sélectionner les variables qui maximisent les performances d'une classification bayésienne**. Ce problème est connu sous l'appellation anglaise : *feature selection for machine learning*.

1.4 Approche

Notre approche consiste à étudier, tester et comparer un certain nombre de méthodes de *feature selection* dans le cadre de notre problème. Nos hypothèses sont les suivantes :

- nous traitons des variables discrètes,
- la tâche à accomplir se ramène à une classification,
- notre méthode de classification est un apprentissage bayésien naïf (voir 1.4.2),
- on estime la probabilité *a posteriori* d'un évènement par son estimateur de maximum de vraissemblance (*i.e.* sa fréquence observée),
- les probabilités *a priori* sont uniformes,
- nous nous plaçons dans le *modèle capteur* que nous décrivons ci-dessous.

1.4.1 Le modèle capteur

Le modèle capteur est une hypothèse que le programmeur de robots bayésiens est souvent amené à utiliser. C'est une hypothèse simplificatrice très forte dans le cas général,

mais qui modélise bien la réalité dans le cas où plusieurs capteurs observent le même phénomène. L'idée est de dire que les variables associées aux capteurs sont indépendantes entre elles, conditionnellement au phénomène observé. Si X_i et X_j sont deux variables associées à deux capteurs et θ le phénomène mesuré par ces capteurs, alors on a :

$$P(X_i|X_j,\theta) = P(X_i|\theta)$$

Ceci signifie que, une fois la valeur de θ connue, X_j n'apporte pas de nouvelle information sur X_i. Autrement dit, l'information apportée à X_i par X_j est déjà présente dans θ. La représentation sous forme de réseau bayésien des dépendances entre variables devient un soleil dont le centre est θ. Dans ses conditions et en appliquant N fois la définition de la probabilité conditionnelle et $\frac{N(N-1)}{2}$ fois le modèle capteur, la distribution conjointe de probabilité devient :

$$P(X_0 \dots X_{N-1},\ \theta) = P(\theta) \prod_{i=0}^{N-1} P(X_i|\theta)$$

Dans ce cadre, la seule couverture markovienne (voir **??**) de X_i est l'ensemble $\{\theta\}$, *i.e.* si X_i est indépendante de θ, alors il n'existe pas de sous-ensemble $R = \{X_{r_0} \dots X_{r_n}\}$ tel que X_i devienne indépendante de θ conditionnellement à R.

Ceci simplifie grandement le travail de la *feature selection* : **il n'y a pas besoin d'étudier les indépendances conditionnelles**.

1.4.2 Apprentissage bayésien naïf

Nous décrivons ici notre méthode de classification car elle se base aussi sur le modèle capteur. Le programme d'apprentissage n'étant pas le centre du projet, nous avons opté pour une méthode d'apprentissage simple mais néanmoins efficace : le classificateur bayésien. Combiné au modèle capteur, il se réduit à ce que l'on appelle un **classificateur bayésien naïf**. Il est optimal dans le sens ou il minimise la probabilité de faire une mauvaise classification sachant les données. Il a été montré qu'il demeurait optimal dans le cas de dépendances conditionnelles modérées [DP97].

Cette méthode consiste à choisir la classe θ_i qui maximise la probabilité conditionnelle :

$$P(\theta_i|X_0 \dots X_{N-1}) = \frac{P(\theta_i X_0 \dots X_{N-1})}{P(X_0 \dots X_{N-1})}$$

Comme $P(X_0 \dots X_{N-1})$ est indépendante de θ_i, elle maximise alors :

$$P(\theta_i X_0 \dots X_{N-1}) = P(\theta_i) \prod_{j=0}^{N-1} P(X_j|\theta_i)$$

Les valeurs de $P(X_j|\theta_i)$ et $P(\theta_i)$ sont issues de l'apprentissage sur les exemples, elles sont stockées sous la forme d'histogrammes suivant une *loi de succession de Laplace*. Une *loi de succession de Laplace* est une représentation d'une distribution de probabilité très proche d'un simple histogramme. La seule différence est que l'on suppose *a priori* que la variable suit une loi uniforme, avant même de remplir l'histogramme avec les valeurs issues des exemples. Ainsi, on approxime la probabilité P qu'une variable X_i prenne la

valeur $x_{i,k}$ par $\hat{P}$. En notant T le nombre d'exemples et $T_{X_i=x_{i,k}}$ le nombre d'exemples tels que $X_i = x_{i,k}$, on pose :

$$\hat{P}(X_i = x_{i,k}) = \frac{1 + T_{X_i=x_{i,k}}}{C + T}$$

Ainsi, quand on n'a aucune réalisation $X_i = x_{i,k}$, on évite d'avoir un zéro qui pose toujours problème. En effet, supposer qu'une probabilité est parfaitement nulle est une hypothèse très forte qui aura de lourdes conséquences par la suite. De plus, cette hypothèse forte est peu justifiée lorsqu'on ne possède qu'un faible nombre d'exemples. D'autre part on constate que, quand le nombre d'exemples est grand on a :

$$\lim_{T \to +\infty} \hat{P}(X_i = x_{i,k}) = \frac{T_{X_i=x_{i,k}}}{T} = P(X_i = x_{i,k})$$

ce qui montre que la *loi de succession de Laplace* tend vers la "vraie" loi de probabilité, au sens fréquenciel du terme.

1.5 Plan de lecture

État de l'art

Le chapitre 2 décrit les méthodes existantes pour résoudre des problèmes similaires au nôtre. Nous verrons que les méthodes de sélection de caractéristiques sont diverses et variées. Cependant, nous verrons qu'il n'existe pas de cadre théorique unificateur ou du moins, que ces méthodes sont des approximations de ce que la théorie de l'information suggère.

Néanmoins, nous exhiberons un schéma général dans lequel s'inscrivent la majorité de ces méthodes et nous en distinguerons deux branches principales dont nous conserverons la dénomination anglophone : celle des *wrappers* et celle des *filters*.

Algorithmes

Nous décrivons notre contribution dans le chapitre 3 : la création et l'implémentation de plusieurs algorithmes de sélection de variables adaptés à notre problème. Nous avons entrepris de tester 10 méthodes dont trois *wrappers* et 7 *filters*. Ces méthodes se basent sur des algorithmes génétiques, des parcours heuristiques, la notion d'information mutuelle, et la statistique du χ^2.

Évaluation

Le chapitre 4 décrit le simulateur logiciel et le robot qui nous ont permis de générer des fichiers de données de test. De plus, nous y évoquons notre façon de noter les algorithmes et de déterminer le taux de réussite de la classification.

Résultats

Finalement, nous discuterons des résultats des algorithmes au vu des critères suivants : nombre de variable éliminées, performance du sous-ensemble généré pour la classification, temps de calcul, et pouvoir explicatif des dépendances entre variables.

Chapitre 2

Etat de l'art

La sélection de caractéristiques (*feature selection*) est un domaine très actif depuis quelques années, en particulier dans le cadre du *data mining*. En effet, la "fouille de données" dans de très grandes bases devient un enjeu crutial pour des applications telles que le génie génétique, la finance, les études de marché, les processus industriels complexes, etc.. Il s'agit en fait de résumer et d'extraire intelligement de la "connaissance" à partir de données brutes. La fouille de données est un domaine basé sur la statistique, l'appprentissage automatique et la théorie des bases de données.

La sélection de variables joue un rôle important dans le *data mining*, en particulier dans la préparation des données avant leur traitement. En effet, les intérêts de la sélection de variables sont les suivants :

- Lorsque le nombre de variables est vraiment trop grand (il peut aller jusqu'à plusieurs dizaines de milliers dans certaines applications), l'algorithme d'apprentissage ne peut pas terminer en un temps convenable. La sélection réduit la dimension de l'espace des caratéristiques.
- D'un point de vue intelligence artificielle, créer un classificateur revient à créer un modèle pour les données. Or, une attente légitime pour un modèle est d'être le plus simple possible (principe du Razoir d'Occam [BEHW87]). La réduction de la dimension de l'espace des caractéristiques permet alors de réduire le nombre de paramètres nécessaires à la description de ce modèle.
- Elle améliore la performance de la classification, sa vitesse et son pouvoir de généralisation.
- Elle augmente la compréhensibilité des données : on voit mieux quels sont les processus qui leur ont donné naissance.

Cette sélection opère par le bias de :

- l'élimination de variables indépendantes de la classe,
- l'élimination de variables redondantes.

Dans cette partie, nous commencerons par formaliser le problème de sélection de variables, puis nous classifierons différentes méthodes trouvées dans la littérature.

2.1 Un problème de dimension

Notre formalisme bayésien nous amène naturellement à décrire le problème de sélection de variables dans les termes de la théorie des probabilités.

Soit $\mathbf{X} = (X_0, X_1, \ldots, X_{N-1})$ le vecteur des N variables. Soit $\mathbf{x} = (x_0, x_1, \ldots, x_{N-1})$ le vecteur d'une réalisation de ces variables. Nous supposons que $\mathbf{X}$ suit une loi de probabilité unique, et donc qu'une réalisation $\mathbf{x}$ se produit avec la probabilité $P(\mathbf{X} = \mathbf{x})$.

Une classification est une procédure qui va associer à chaque réalisation $\mathbf{x}$ une des L classes θ_i $\forall i \in [0 \ldots L-1]$. Comme les données sont soumises à l'incomplétude et au bruit du monde réel [BtLrG03], la classification ne pourra pas être déterministe, mais probabiliste. On décidera que l'instance $\mathbf{x}$ appartiendra à la classe θ_i si θ_i maximise $P(\theta = \theta_i | \mathbf{X} = \mathbf{x})$. Cette façon de classer sera apprise sur une base d'exemples de couples $(\mathbf{x}, \theta_i(x))$.

Le but de ce travail est alors de trouver le meilleur sous-ensemble $\mathbf{Y} \subset \mathbf{X}$, *i.e.* le sous-ensemble $\mathbf{Y}$ qui maximise les performances de la classification. Pour cela, il faut trouver un sous-ensemble vérifiant que la distribution $P(\theta|\mathbf{Y})$ soit peu différente de la distribution $P(\theta|\mathbf{X})$, et que :

$$Card(\mathbf{Y}) < Card(\mathbf{X})$$

Par exemple, si l'on suppose que $X = \{X_1, X_2\}$ et que X_1 et X_2 soient deux variables parfaitement redondantes (*i.e.* $X_1 = X_2$), alors $Y = X_1$ vérifie ces deux propriétés car :

$$P(\theta|\mathbf{X}) = P(\theta|X_1X_2) = \frac{P(\theta X_1 X_2)}{P(X_1X_2)} = \frac{P(\theta X_1)}{P(X_1)} = P(\theta|X_1) = P(\theta|\mathbf{Y})$$

et

$$Card(\mathbf{Y}) = 1 < 2 = Card(\mathbf{X})$$

Dans le cas général, il est aisé de mesurer $Card(\mathbf{X})$ et $Card(\mathbf{Y})$, mais il est beaucoup plus hardu de déterminer si la distribution $P(\theta|\mathbf{Y})$ est "peu différente" de la distribution $P(\theta|\mathbf{X})$. En effet, il existe des fonctions donnant une idée de la "distance" entre deux distributions de probabilité (comme la KL-distance, voir chapitre suivant). Mais on ne peut pas les appliquer directement car elles nécessitent de faire des sommes sur tout l'espace, c'est-à-dire sur un espace de dimension $N+1$ pour $P(\theta|\mathbf{X})$ (ou pour la densité conjointe $P(\theta\mathbf{X})$).

En plus de ce problème de temps de calcul, une autre limitation nous empêche de chiffrer une telle "distance". Dans les cas pratique de robotique ou de *data mining*, nous n'avons jamais accès à la "vraie" distribution $P(\theta, \mathbf{X})$, (quoi que cela signifie...). Nous pouvons seulement estimer les probabilités par leur estimateur de maximum de vraissemblance en construisant des histogrammes à partir de fichiers de données. Mais à cause de la haute dimentionnalité de l'espace, nous n'aurons jamais assez d'exemples pour estimer correctement les probabilités, beaucoup trop de "cases" de l'espace resteront vides. Et utiliser des modèles paramétriques ou des *a priori* uniformes ne changera rien car il n'y a tout simplement pas assez d'information pour approximer correctement $P(\theta, \mathbf{X})$.

Le nombre et la diversité des méthodes de *feature selection* viennent de la diversité des manières de contourner ce problème, en opérant des approximations plus ou moins importantes.

2.2 Structure de la *feature selection*

Une structure générale des algorithmes de sélection de caractéristiques peut être proposée de la façon de la figure 2.1 ([LY02]). Jusqu'à ce qu'un certain critère soit satisfait, des sous-ensembles **Y** sont générés en parcourant l'espace des sous-ensembles, et ils sont évalués. Nous décrivons successivement chaque phase.

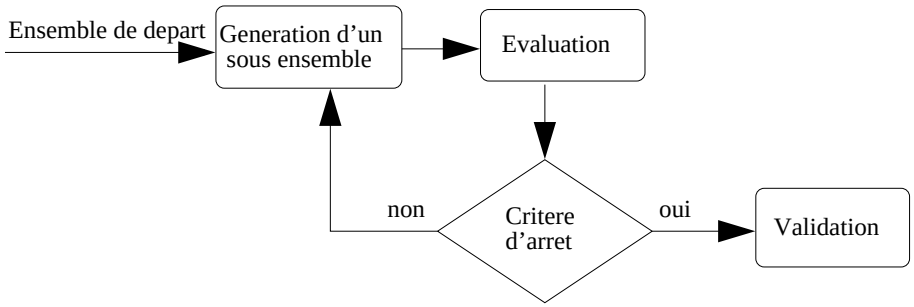

FIG. 2.1: *Procédure générale de sélection d'un sous-ensemble de caratéristiques*

2.2.1 Génération de sous-ensembles

La génération de sous-ensembles est une procédure de recherche dans l'espace des sous-ensembles de cardinal 2^N. Tous les méthodes de parcours classiques sont utilisables. Nous en présentons trois familles.

1. **Parcours exhaustif :** On génère et teste tous les sous-ensembles. Ceci est prohibitif dès que le nombre de variables est supérieur à 5. Un tel parcours nous garantirait l'optimalité de la solution. De par des propriétés de monotonie, cette optimalité pourrait quand même être atteinte sans tester tous les sous-ensembles. Mais [NF77] montre que le coût est toujours $O(2^N)$.
2. **Parcours Heuristique :** On utilise une heuristique pour guider la recherche. Le coût devient $O(N^2)$. Dans cette catégorie, on peut commencer par un ensemble vide et ajouter des variables une à une (*forward addition*), ou bien commencer avec l'ensemble de toutes les variables et en enlever (*backward elimination*). [KS96] avance que la méthode *backward elimination* est mieux adaptée mais les résultats expérimentaux de [JKP94] montrent que les performances sont équivalentes. La méthode *forward addition* sélectionne néanmoins moins de variables. D'autres types de recherche sont possibles (*greedysearch, best first* voir [KJ97] pour une nomenclature).
3. **Parcours non déterministe :** Les sous-ensembles sont générés lors d'un processus comprenant une part de hasard. Typiquement, [Rea02] et [YH98] utilisent des algorithmes évolutionnistes.

2.2.2 Évaluation

L'évaluation d'un sous-ensemble est traitée de facons très diverses. Il existe deux grandes classes d'algorithmes :

1. Les *wrappers* : qui utilisent l'algorithme de classification pour évaluer les sous-ensembles générés,
2. Les *filters* : qui sont complètement indépendants de cet algorithme, mais se basent sur des considerations statistiques, entropiques, de cohérence, de distance,etc.

Les wrappers

Bien que conceptuellement plus simples que les filtres, les *wrappers* ont été introduits plus récemment par John, Kohavi et Pfleger en 1994 [JKP94]. Leur principe est de générer des sous-ensembles candidats et de les évaluer grâce à l'algorithme de classification. Le score d'un ensemble sera par exemple un compromis entre le nombre de variables éliminées et le taux de réusite de la classification sur un fichier de test. Ainsi la phase "évaluation" du cycle de sélection est constituée par un appel à l'algorithme de classification. En fait, celui-ci est appellé plusieurs fois à chaque évaluation car un mécanisme de validation croisée (voir section 4.3.1) est fréquement utilisé.

De par son principe même, cette méthode génère des sous-ensembles bien adaptés à l'algorithme de classification. Les taux de reconnaissance sont hauts car la sélection prend en compte le biais intrinsèque de l'algorithme de classification. Un autre avantage est sa simplicité conceptuelle : il n'y a nul besoin de comprendre comment l'induction est affectée par la sélection de variables, il suffit de générer et de tester.

Cependant, trois raisons font que les *wrappers* ne constituent pas une solution parfaite. D'abord, ils n'apportent pas vraiment de justification théorique à la sélection et ils ne nous permettent pas comprendre les relations de dépendances conditionnelles (voir des compléments sur cette notion en annexe 7.1) qu'il peut y avoir entre les variables. D'autre part la procédure de sélection est spécifique à un algorithme de classification particulier et les sous-ensembles trouvés ne sont pas forcement valides si on change de méthode d'induction. Finalement, et c'est le défaut principal de la méthode, les calculs deviennent vite très longs, voir irréalisables lorsque le nombre de variable croît.

Les filters

Les filtres n'ont pas les défauts des *wrappers*. Ils sont beaucoup plus rapides, ils reposent sur des considérations plus théoriques, ils nous permettent de mieux comprendre les relations de dépendance entre variables. Mais, comme ils ne prennent pas en compte les biais de l'algorithme de classification, les sous-ensembles de variables générés donnent un taux de reconnaissance plus faible.

Pour donner un score à un sous-ensemble, une première solution est de donner un score à chaque variable indépendemment des autres et de faire la somme de ces scores. Pour évaluer une variable, l'idée est de déterminer sa corrélation avec la variable de classe. Mais [IG03] propose des exemples simples montrant que cette approche nommée *feature ranking* pose des problèmes dans le cas général. En effet, cette approche n'élimine pas les variables redondantes, d'autre part il est possible que des variables peu corrélées avec la classe deviennent utiles lorsqu'on les considère dans le contexte des autres variables. Mais ce dernier reproche n'est pas valide dans le modèle capteur.

L'autre solution est d'évaluer un sous-ensemble dans sa globalité. On se raproche ici de l'apprentissage de la structure de réseau bayésien décrite dans [HGC94]. Une méthode plus spécifique au problème est décrite dans [KS96]. Koller et Sahami y proposent

d'éliminer une variable si elle possède une couverture markovienne, c'est-à-dire si elle est indépendante de la classe, sachant les autres variables. Ils proposent un algorithme utilisant une approximation sommaire de cette idée car elle n'est pas implémentable en pratique.

Il existe un intermédiaire entre *feature ranking* et *subset ranking* basé sur une idée de Ghiselli [Ghi64] et utilisé avec de bons résultats dans le cadre de la CFS (*correlation based feature selection*) par M. Hall [Hal98]. Le score d'un sous-ensemble est construit en fonction des corrélations variable-classe et des corrélations variable-variable selon la formule suivante :

$$r_{\theta S} = \frac{k\overline{r_{\theta i}}}{\sqrt{k + k(k-1)\overline{r_{ij}}}}$$

avec $r_{\theta S}$ le score du sous-ensemble de cardinal k, $\overline{r_{\theta i}}$ la moyenne arithmétique des corrélations entre θ et les variables i, et $\overline{r_{ij}}$ la moyenne des k^2 intercorrélations entre variables. Cette équation exprime que le score du sous-ensemble augmente si les variables sont fortement corrélées avec θ et diminue si elles sont fortement corrélées entre elles. De plus cette formule est valable dans le cadre de scores normalisés (*i.e.* de variance unitaire). L'idée est de dire qu'un bon sous-ensemble est constitué de variables hautement corrélées avec la classe (pour ne pas garder les indépendantes), et peu corrélées entre elles (pour éviter la redondance). Il s'agit d'une approximation car on ne prend en compte que les interactions d'ordre 1.

La corrélation ou dépendance entre deux variables peut être définie de plusieurs façons. Utiliser le coefficient de corrélation statistique comme dans [Hal00] est trop restrictif car il ne capture que les dépendances linéaires. On peut, en revanche ,utiliser un test d'indépendance statistique comme le test du χ^2 [KS77][LS95][Gau02]. Il est aussi possible d'utiliser la notion d'information mutuelle définie par :

$$I(X_i, X_j) = \sum_{X_i} \sum_{X_j} P(X_i, X_j) log(\frac{P(X_i, X_j)}{P(X_i)P(X_j)})$$

D'autres méthodes d'évaluation de sous-ensembles sont possibles. Par exemple, il est intéressant de se baser sur la vision bayésienne des probabilités pour construire des mesures d'indépendances strictement bayésiennes [MT01] [ZH02] [Wol94]. Certains auteurs [AD94] utilisent la notion de consistance pour juger un sous-ensemble. Des méthodes récentes combinant *wrapper* et *filter* sont présentées dans [IG03].

2.2.3 Critère d'arrêt

Le critère d'arrêt peut être de diverses natures : un temps de calcul, un nombre de générations (pour un algorithme génétique), un nombre de variables sélectionnées, une évaluation heuristique de la "valeur" du sous-ensemble...

2.2.4 Validation

Une fois le "meilleur" sous-ensemble déterminé, les performances de la classsification sont alors évaluées comme nous le décrivons dans le section 4.3.1.

2.3 Bilan

Ainsi les méthodes de sélection de caractéristiques sont diverses et variées, et nombre d'entre elles ont été déployées avec succès dans des applications réelles comme le *data mining*. Cependant il n'existe pas de cadre théorique unificateur, et ces méthodes sont souvent constituées d'approximations grossières de ce que la théorie de l'information suggère.

Deux branches principale se distinguent : *wrappers* et *filters*. Les *filters* qui sont moins performant mais beaucoup plus rapides, se basent la plupart du temps sur des mesures de dépendance entre variables.

Dans le chapitre suivant, nous présentons une dizaine d'algorithmes de *feature selection* que nous avons implémentés.

Chapitre 3

Algorithmes

Dans ce chapitre, nous présentons rapidement les 10 algorithmes de sélection de variables que nous avons définis, implémentés et comparés.

Les notations utilisées sont : X_i les variables de capteurs, θ la variable de classe, **C=L** le nombre moyen de valeurs possibles pour toutes ces variables, **T** le nombre d'exemples (nombre de pas de temps lors de la simulation) et **k** le nombre de variables sélectionnées parmi les **N** initiales.

Remarque : tous ces algorithmes ont été construits et écrits spécifiquement pour ce projet, sauf la classe Algorithme Génétique qui a été reprise de la thèse d'Anne Spalanzani [Spa99]. Le dernier algorithme **ModCapt** est totalement original alors les autres sont des recombinaisons de méthodes inspirées de l'état de l'art.

3.1 AGBN

Nom : AGBN = Algorithme Génétique, évaluation par apprentissage Bayésien Naïf.
Description :

1. Parcours de l'espace : algorithme génétique.
2. Évaluation des sous-ensembles : *wrapper*.
3. Critère d'arrêt : nombre de générations.

Le parcours de l'espace des sous-ensembles de variables est réalisé par un algorithme génétique décrit en annexe (7.2). Un individu représente un sous-ensemble et la fonction d'évaluation retourne le taux de reconnaissance de l'algorithme de classification Bayésien naïf pour ce sous-ensemble.

Complexité :
Pour chaque individu de chaque génération, on effectue un apprentissage bayésien naïf sur un fichier de T données de taille $\overline{k} = \frac{N}{2}$ (en moyenne) et une classification sur un fichier similaire. Si **G** est le nombre de générations et **I** le nombre d'individus, le coût est :

$$O(GI(\overline{k}c^2 + TC\overline{k})) = O(GITC\overline{k})$$

opérations basiques (affectation, addition, multiplication...)

Remarques :
C'est le seul *wrapper*. L'ajustement des paramètres de l'algorithme génétique est toujours

délicat. Ces paramètres sont notamment le taux de mutation, le taux de *crossover*, le mode de sélection, la taille de la population et le nombre de générations.

3.2 AGXH

Nom : AGXH = Algorithme Génétique, évaluation par l'entopie croisée (cross-H) .
Description :

1. Parcours de l'espace : algorithme génétique.
2. Évaluation des sous-ensembles : Formule de Ghiselli (voir état de l'art 2.2.2)

$$r_{\theta S} = \frac{k\overline{r_{\theta i}}}{\sqrt{k + k(k-1)\overline{r_{ij}}}}$$

3. Évaluation de corrélation : Entropie croisée. C'est une mesure de la "distance" entre deux distributions. L'entropie **H** d'une variable mesure l'incertitude pour cette variable. Elle est maximale pour une distribution uniforme et nulle pour une distribution en pic de Dirac.

$$H(X) = \sum_X P(X) log \frac{1}{P(X)} = -\sum_X P(X) log P(X)$$

L'entropie croisée ou KL-distance (pour Kullback-Leibler [KL51]) entre deux distributions de probabilité μ et σ est :

$$D_{KL}(\mu, \sigma) = \sum_x \mu(x) log \frac{\mu(x)}{\sigma(x)}$$

On retrouve alors que l'information mutuelle entre deux variables X_i et X_j

$$I(X_i, X_j) = \sum_{X_i} \sum_{X_j} P(X_i, X_j) log(\frac{P(X_i, X_j)}{P(X_i)P(X_j)})$$

est en fait la KL-distance entre les distributions $P(X_i, X_j)$ et $P(X_i)P(X_j)$. En effet, elle est nulle si les variables sont indépendantes et d'autant plus grande que les variables sont dépendantes.
On peut monter que :

$$I(X_i, X_j) = I(X_j, X_i) = H(X_i) - H(X_i|X_j) = H(X_i) + H(X_j) - H(X_i, X_j)$$

Comme cette information mutuelle est biasée en faveur des variables avec un grand nombre de valeurs possibles, nous utilisons le coefficient d'incertitude symétrique CIS défini par :

$$CIS = 2[\frac{I(X_i, X_j)}{H(X_i) + H(X_j)}]$$

Le CIS vaut 0 si les variables sont indépendantes (l'information mutuelle est nulle) et 1 si $H(X_i|X_j) = 0$, c'est-à-dire si l'incertitude sur X_i connaissant X_j est nulle, *i.e.* si elles sont parfaitement liées.

4. Critère d'arrêt : nombre de générations.

Complexité :
A première vue $O(GI\overline{k}^2C^2)$, mais $O(GI\overline{k}^2)$ grace à nos optimisations.

Remarques :
La formule de Ghiselli favorise les ensembles fortement corrélés avec la classe et peu intra-corrélés. Cependant, elle ne renseigne pas sur les dépendances d'ordre supérieures à 1.

3.3 FAXH

Nom : FAXH = *Forward Addition*, évaluation par l'entopie croisée (XH).
Description :

1. Parcours de l'espace : *Forward Addition*.
2. Évaluation des sous-ensembles : Formule de Ghiselli.
3. Évaluation de corrélation : Entropie croisée.
4. Critère d'arrêt : On s'arrête quand il n'est plus possible de faire croître le score en ajoutant une variable.

C'est la même méthode que AGXH pour évaluer les sous-ensembles, mais le parcours de l'espace est différent. On part de l'ensemble vide et on ajoute les variables une à une. Soit S le sous-ensemble courant. Pour chaque variable $X_i \notin S$, on évalue tous les $S_i = S \bigcup X_i$. On retiendra le sous-ensemble de plus grand score.
Complexité : $O(\sum_{k=0}^{N}(N-k)kC^2) = O(N^3C^2)$
Remarques
Comme pour les AG, le critère d'arrêt est basé sur le score du sous-ensemble et non sur son nombre de variables.

3.4 FAχ^2

Nom : $FA\chi^2$ = *Forward Addition* évaluation par la statistique du χ^2
Description :

1. Parcours de l'espace : *Forward Addition*.
2. Évaluation des sous-ensembles : Formule de Ghiselli.
3. Évaluation de corrélation : statistique du χ^2.
4. Critère d'arrêt : idem FAXH.

FAχ^2 est identique à FAXH sauf que la corrélation entre deux variables est mesurée par la statistique du χ^2 [LS95] qui mesure un sorte de "distance" à l'indépendance (dans ce cas $\chi^2 = 0$)

$$\chi^2(X,Y) = \sum_X \sum_Y \frac{A_{ij} - E_{ij}}{E_{ij}}$$

avec A_{ij} le nombre d'exemples tels que $X = x_i$ et $Y = y_j$, et E_{ij} ce même nombre en supposant l'indépendance *i.e.* $E_{ij} = \frac{1}{T}(\sum_l A_{il}) * (\sum_l A_{lj})$.
Complexité : $O(N^3C^2)$ car il coûte autant de calculer un CIS que $\chi^2(X,Y)$

Remarques :
Pour être statistiquement valide, il faut avoir suffisamment d'exemples, en fait il faut qu'il y ait au moins 5 exemples dans au moins 95% des cases de la table de contingence. On suppose une distribution *a priori* uniforme pour se placer dans ces conditions, sans perte de généralité dans un cadre bayésien (hypothèse de loi de succession de Laplace).

3.5 Rank XH et Rank χ^2

Description :

1. Parcours de l'espace : non défini.
2. Évaluation des sous-ensembles : non défini (*feature ranking*)
3. Évaluation de corrélation : XH et χ^2
4. Critère d'arrêt : non défini.

Il s'agit simplement de 2 tris des variables selon leur corrélation avec la variable de classe.

Complexité : $O(NC^2)$

Remarques :
Ceci nous sert à interpréter les résultats des autres algorithmes.

3.6 χ^2Test

Description :

1. Parcours de l'espace : non défini.
2. Évaluation des sous-ensembles : non défini.
3. Évaluation de corrélation : test d'indépendance du χ^2
4. Critère d'arrêt : non défini.

Pour chaque variable, on réalise un test d'indépendance avec la classe (test du χ^2).

Complexité :
$O(NC^2)$ + le calcul de la p-valeur en fonction du nombre de degrés de liberté et du niveau de signification α.

Remarques :
Voir [Gau02] pour plus de détails sur les tests d'hypothèses. Il s'agit d'une variation des méthodes de *feature ranking* dans laquelle le seuil est défini par le niveau de signification et ne dépend donc pas du nombre de valeurs possibles C pour les variables.

3.7 EX et EX2

Description :

1. Parcours de l'espace : on teste tous les sous-ensembles.
2. Évaluation des sous-ensembles : *wrapper*.
3. Évaluation de corrélation : *wrapper*.
4. Critère d'arrêt : on s'arrête quand tous les sous-ensembles ont été testés.

EX est un wrapper EXhaustif qui essaie systématiquement tous les sous-ensembles et trouve le meilleur. Ainsi nous obtiendrons le sous-ensemble optimal (au sens des *wrappers*). EX2 ne diffère que dans le sens où il ne teste "que" tous les ensembles de cardinal $\frac{N}{2}$. L'idée est d'avoir également un sous-ensemble optimal de $\frac{N}{2}$ variables pour pouvoir mieux étudier l'algorithme ModCapt.

Complexité : $O(2^N TC)$

Remarques :
Ils ne sont utilisables que pour les très petits N et servent alors de référence optimale.

3.8 ModCapt

Nom : ModCapt car c'est le seul à prendre en compte explicitement l'hypothèse de MODèle CAPTeur.

Description

1. Parcours de l'espace : *backward elimination*
2. Évaluation des sous-ensembles : non défini,
3. Évaluation de corrélation : non défini,
4. Critère d'arrêt : on arrête quand on a éliminé un nombre prédéfini de variables ($\frac{N}{2}$ dans nos tests).

On suppose l'hypothèse du modèle capteur. On part de l'ensemble plein, et on enlève des variables une à une. A chaque essai, on quantifie la distance entre la densité de probabilité originelle et la densité en considérant la variable indépendante des autres conditionnellement à la classe. La distance utilisée est la KL-distance symétrisée Δ.

$$\Delta(\mu, \sigma) = D_{KL}(\mu, \sigma) + D_{KL}(\sigma, \mu)$$

On éliminera les $\frac{N}{2}$ variables qui rendent cette distance minimale. Soit P la distribution sur N variables en supposant le modèle capteur et P_i la même distribution en supposant de plus que X_i est indépendante de la classe θ. On a :

$$\begin{aligned} P(X_0 \dots X_{N-1}\theta) &= P(\theta) \prod_{k=0}^{N-1} P(X_k|\theta) \\ P_i(X_0 \dots X_{N-1}\theta) &= P(\theta)P(X_i) \prod_{k=0\ k\neq i}^{N-1} P(X_k|\theta) \end{aligned} \tag{3.1}$$

et alors :

$$\begin{aligned}
\Delta(P,P_i) &= \sum_{\theta,X_0,\ldots,X_{N-1}} P(X_0\ldots X_{N-1}\theta) log(\frac{P(X_0\ldots X_{N-1}\theta)}{P_i(X_0\ldots X_{N-1}\theta)}) \\
&+ \sum_{\theta,X_0,\ldots,X_{N-1}} P_i(X_0\ldots X_{N-1}\theta) log(\frac{P_i(X_0\ldots X_{N-1}\theta)}{P(X_0\ldots X_{N-1}\theta)}) \\
&= \sum_{\theta,X_0,\ldots,X_{N-1}} \prod_{j\neq i}^{N-1} P(X_j|\theta)[log(\frac{P(X_i)}{P(X_i|\theta)}) - (P(X_i) - P(X_i|\theta))] \\
&= \sum_{\theta} P(\theta) \sum_{X_i} log(\frac{P(X_i)}{P(X_i|\theta)}) - (P(X_i) - P(X_i|\theta)) \sum_{X_{j\neq i}} \prod_{j\neq i}^{N-1} P(X_j|\theta) \\
&= \sum_{\theta} P(\theta) \sum_{X_i} log(\frac{P(X_i)}{P(X_i|\theta)}) - (P(X_i) - P(X_i|\theta))
\end{aligned} \tag{3.2}$$

car

$$\sum_{X_{j\neq i}} \prod_{j\neq i}^{N-1} P(X_j|\theta) = \sum_{X_{j\neq i}=0}^{N-1} P(X_0\ldots X_{i-1}, X_{i+1}\ldots X_{N-1}|\theta) = 1$$

Complexité : $O(NC^2)$

Remarques

On a ainsi retrouvé que, dans le cadre du modèle capteur, il n'est pas nécessaire de considérer les autres variables pour décider si X_i est indépendante de θ. On retrouve qu'il suffit de regarder "l'écart" entre $P(X_i)$ et $P(X_i|\theta)$. On pourrait alors penser qu'il suffit de classer les variables selon leur corrélation avec θ et de prendre les meilleures (*feature ranking*). Mais, comme nous l'avons vu dans la section 2.2.2, cette approche n'élimine pas les variables redondantes. L'indépendance conditionnelle n'a rien à voir avec la redondance. Par exemple si on condidère deux fois la même variable on a $P(X|\theta, X) = P(X|\theta)$ alors que X est évidemment redondante avec elle-même.

On remarque d'autre part que l'hypothèse du modèle capteur permet de passer d'une complexité astronomique en $O(C^{N+1})$ à une complexité linéaire en N.

3.9 Bilan

Dans ce chapitre, nous avons présenté succintement 10 algorithmes de sélection de variables permettant de contourner le problème de l'explosion combinatoire. Dans les chapitres suivants nous les comparons sur différents fichiers de données.

Chapitre 4

Évaluation

Dans ce chapitre, nous présentons notre approche d'évaluation et de validation de nos algorithmes. Pour cela nous décrivons nos méthodes de génération de fichiers de données , puis nous présentons notre estimateur du taux de réussite de la classification.

4.1 Génération de données

Nous avons deux sources de données : un simulateur logiciel et un vrai robot.

4.1.1 Simulateur

Dans un premier temps, il nous a paru nécessaire de disposer de données artificielles complètement maitrisées. En effet cela nous a permis de mieux comprendre les relations de dépendances entre les variables pour mieux étudier les réponses de nos algorithmes. De plus, il est très commode de pouvoir générer les données qui nous intéressent *a priori*, afin de tester un aspect particulier d'une méthode. Ainsi, nous avons codé une plateforme expérimentale simulant les sorties des capteurs d'un robot stimulés par différentes sources.

Techniquement, le simulateur est un programme C++ objet modélisant un ou plusieurs robots ponctuels et fixes dans un univers en 2D (Voir figure 4.1). Dans cet univers, divers objets peuvent se mouvoir selon des trajectoires précalculées ou définies par l'utilisateur. Les objets de l'environnement sont perçus par le robot par l'intermédiaire de ses capteurs. Un robot peut posséder un nombre arbitraire de capteurs. Les capteurs peuvent être sensibles à différentes grandeures (lumière, son, proximité ...).

Face à ces différents capteurs, les objets se mouvant dans le monde simulé sont source de stimuli. Par exemple, ils sont toujours visibles par les capteurs de proximétrie. D'autre part, on peut leur attribuer une émission de son ou de lumière.

Le principe d'une simulation est de définir un intervalle de temps discrétisé, puis, pour chaque pas de temps, le simulateur calcule les sorties de chaqun des capteurs en fonction des sources présentes dans le monde et de leur position. Les capteurs sont additifs dans le sens où leur sortie face à plusieurs sources est la somme de leurs sorties pour chacune des sources.

Détaillons les différents types de capteurs :

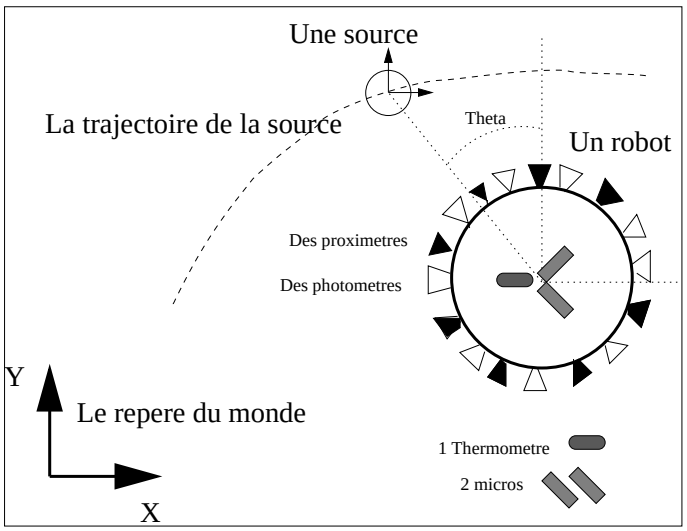

FIG. 4.1: *Représentation du monde simulé, avec robots, sources et capteurs.*

1. **Lumière :** Un capteur de lumière est directionnel. Il est sensible à la distance de la source lumineuse et à son angle par raport à l'axe propre du capteur. La sortie du capteur est toujours positive. Elle decroit linéairement avec la distance, elle admet un maximum lorsque la source est dans l'axe du capteur. Elle décroit comme une gaussienne lorsque la source s'éloigne de cet axe. De plus, pour mieux simuler les incertitudes du monde réel, nous avons troublé cette sortie par un bruit gaussien de variance paramétrable (voir figure 4.2).

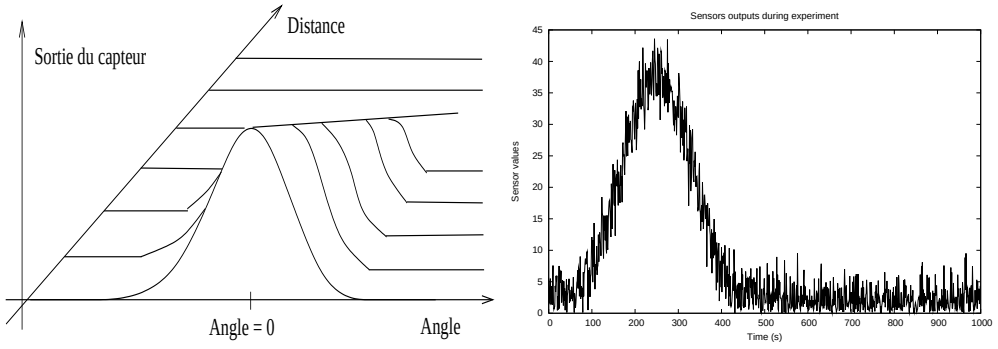

FIG. 4.2: **A gauche** *: Allure de la réponse d'un photomètre en fonction de l'angle et de la distance de la source lumineuse.* **Remarque :** *le bruit n'est pas représenté.* **A droite** *: Réponse d'un photomètre lorsqu'une source de lumière tourne autour du robot.*

2. **Proximité d'un autre objet :**
   De même, les proximètres sont directionnels. En dehors d'une certaine plage d'angles paramétrable, leur sortie est nulle. Si la distance de la source est supérieure à un seuil, la sortie est toute aussi nulle. Dans leur domaine de valeur, les proximètres retournent une valeure constante, mais légèrement bruités comme

précédement (voir figure 4.3).

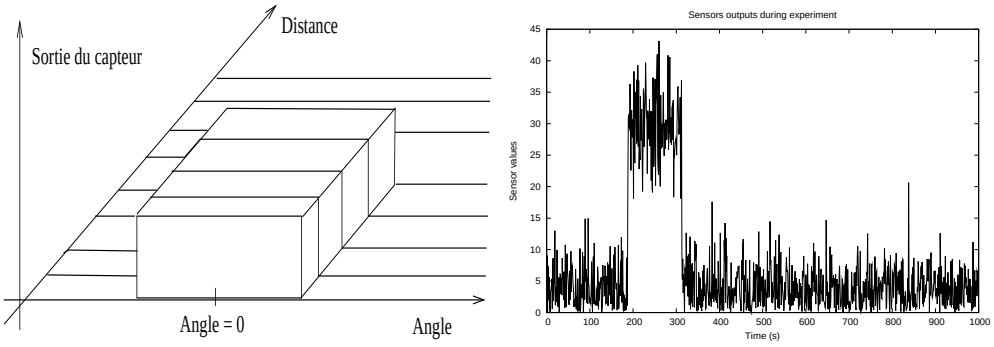

FIG. 4.3: **A gauche** *: Allure de la réponse d'un proximètre en fonction de l'angle et de la distance de l'objet.* **Remarque :** *le bruit n'est pas représenté..* **A droite** *: Réponse d'un proximètre lorsqu'un objet tourne autour du robot.*

3. **Son :**

   Les capteurs de son ont été modélisés de façon à mettre en évidence la nécessité de fusionner des données pour déterminer la position d'une source. La sortie d'un microphone ne dépend pas de la distance mais seulement de l'angle relatif de la source. Nous avons choisi la fonction de réponse du capteur pour qu'avec un seul microphone, on ait 2 candidats équiprobables pour l'angle de la source. Par contre, si un autre microphone est présent et s'il n'est pas dans le même axe, alors il devient possible de déterminer l'angle sans ambiguïté (figure 4.4). De plus, les valeurs sont encore une fois bruitées.

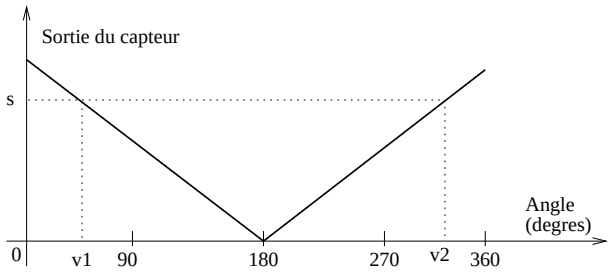

FIG. 4.4: *Réponse d'un micro en fonction de l'angle relatif de la source sonore. On voit que pour une valeur s de sortie du capteur, deux valeurs sont possibles pour l'angle* v1 *et* v2*. Mais si on ajoute un deuxième micro, alors sur les 4 valeurs, deux seront égales et ce sera l'angle recherché.*

4. **Température :**

   Nous avons considéré la température comme une variable indépendante de toutes les autres. Pour cela nous l'avons modélisée comme une sinusoïde du temps de faible amplitude. Elle est aussi bruitée.

5. **Charge de la batterie :**

C'est une variable maximale au début de l'expérience qui décroit linéairement au cours temps. Nous avons fixé un faible coefficient directeur à cette droite, et nous l'avons bruitée.

6. **Capteurs totalement bruités :**
   Afin de tester nos algorithmes avec des données artificielles révélatrices, nous avons ajouté des capteurs qui retournent des valeurs uniformément réparties dans leur domaine. Ces capteurs devront être facilement écartés par un bon algorithme de sélection de caractéristiques.

Ainsi, ce simulateur nous permet de générer de nombreux fichiers de données, chacun testant un aspect particulier des algorithmes. De plus, nous sommes maîtres du nombre de variables, et donc de la complexité des calculs à effectuer. La figure 4.5 présente un relevé avec de nombreux capteurs.

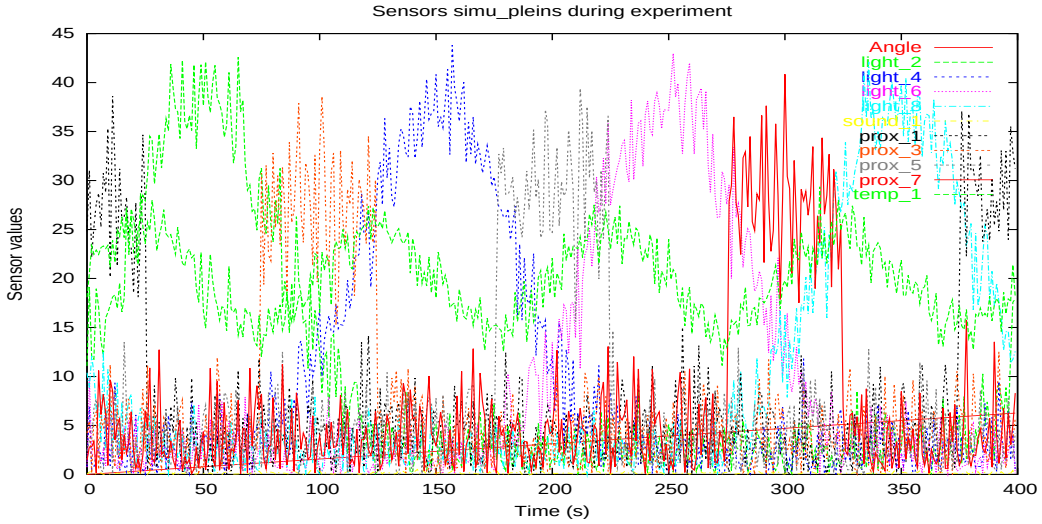

FIG. 4.5: *Un relevé avec de nombreux capteurs lors du déplacement circulaire d'une source de lumière autour du robot. On voit des photomètres, des proximètres, la température...*

4.1.2 Robot

La deuxième phase de l'évaluation est réalisée sur des données robotiques réelles provenant de plusieurs expériences. Le robot utilisé est un **Koala** commercialisé par la société **K-Team S.A.**. Il s'agit d'un robot mobile de taille moyenne (70 cm par 40 cm) monté sur 6 roues. Il est équipé de nombreux capteurs dont 16 proximètres à ultrasons, 16 photomètres, des odomètres et une caméra vidéo mobile selon deux axes (figure 4.6).

L'expérience réalisée est celle précédemment décrite comme *suivi de balle rouge*. Le Koala est immobile pendant toute l'expérience. grace à sa camera il peut reconnaître des objets colorés. Ce suiveur (*tracker*) est une implémentation classique de la règle de Bayes. Un modèle de l'objet coloré est construit sur la base d'histogrammes et une

FIG. 4.6: *Le Koala et ses deux caméras mobiles (le robot utilisé dans les expériences ne possède qu'une seule caméra).*

image de probabilité pour chaque pixels de faire partie de l'objet est générée. Le centre de gravité de cette image de probabilité est considéré comme le centre de l'objet. Ensuite la caméra est asservie à fixer le centre de l'objet mobile. Nous choisissons de ne déplacer l'objet (la balle rouge) que dans le plan horizontal. Tout ceci nous donne accès à une image de l'angle entre l'axe pricipal du robot et la balle rouge. Durant l'expérience, on déplace la balle autour du robot en prenant soin que sa présence stimule les proximètres. Toutes les 100ms, on recueille 59 valeurs sur le robot.

Ces 58 (59 - l'angle θ) variables seront soumises à nos algorithmes de sélection. Nous cherchons quelles sont les variables parmis elles qui sont nécessaires et suffisantes pour prévoir la position de la balle rouge (ou d'un autre objet) sans utiliser la caméra. Le but est de trouver que seuls les proximètres sont porteurs d'information sur la position de la balle. La figure 4.7 représente la mesure de l'angle de la balle rouge et de quelques autres capteurs pendant l'expérience.

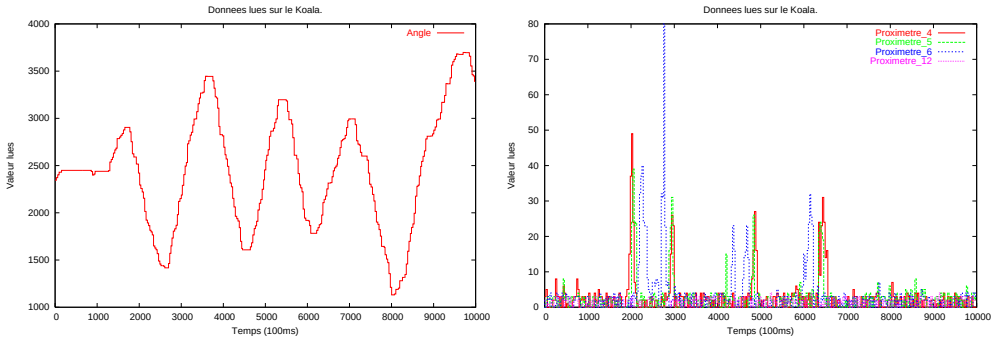

FIG. 4.7: **A gauche** *: Angle entre le Koala et la balle. Plusieurs arcs de cercles sont décrits autour du robot.* **A droite** *: Quelques capteurs du Koala durant l'expérience de la balle rouge, dont des proximètres*

4.2 Plan d'expériences

Pour valider et comparer les méthodes implémentées, un protocole de test a été mis en place. Il commence avec des expériences très simples avec le simulateur et finit par l'utilisation de données robotiques.

Dans les six premières expériences, une source de lumière repérée par son angle θ tourne autour d'un robot équipé de différents capteurs. Le but est de trouver un sous-ensemble de capteurs suffisant pour prédire θ.

1. **Expérience** *2thetas*
   Les capteurs sont deux images identiques de θ. On veut tester par cette expérience si nos algorithmes peuvent détecter une redondance si évidente. En effet, un seul de ces capteurs est suffisant pout prévoir l'angle θ .
2. **Expérience** *theta_et_3bruits*
   Capteurs : un copie de θ et 3 bruits totalement décorrélés avec θ.
   But : tester l'élimination de variables trivialement indépendantes (le bruit)
3. **Expérience** *8proxis_et_3bruits*
   Capteurs : 8 proximètres équirépartis autour du robot et 3 bruits.
   But : déterminer si les algorithmes trouvent que les proximètres sont porteurs de plus d'information sur θ que le bruit. Si oui, quel sera alors le taux de reconnaissance du classificateur n'utilisant que les données proximétriques ?
4. **Expérience** *2fois8proxis*
   Capteurs : les 8 proximètres et leur copie.
   But : étudier l'élimination de redondances lorsque qu'un seul capteur n'est pas suffisant pour prédire θ.
5. **Expérience** *8proxis8lumis*
   Capteurs : les 8 proximètres et 8 photomètres de même emplacement.
   But : étudier l'élimination de redondances plus fine. Il y a ici deux fois l'information sous deux formes différentes, car on peut prévoir θ grace aux proximètres seuls ou au luminomètes seuls. Il sera intéressant de voir si les algorithmes font un mélange des ces capteurs ou bien s'ils n'en conservent qu'une seule sorte.
6. **Expérience** *les23*
   Capteurs : 8 proximètres, 8 photomètres, 2 micros, 3 bruits, 1 témoin de la charge de la batterie et 1 capteur de température.
   But : étudier les algorithmes dans un contexte plus réaliste, mesurer leur temps de calculs avec un nombre intermédiaire de variables.
7. **Expérience** *2sound_3bruits*
   Source : sonore et non plus lumineuse,
   Capteurs : deux micros formant un angle de $\frac{\pi}{2}$ et 3 bruits.
   But : En théorie, le déphasage angulaire entre les micros permet de remonter sans ambiguïté à θ. Les résultats le confirmeront-ils dans un contexte bruité ?
8. **Expérience** *sound19_TrajV_et_V*
   Source : sonore,
   Trajectoire : un mouvement de 4 va-et-vien en arc de cercle d'ouverture angulaire $\frac{5\pi}{6}$ devant le robot

Capteurs : deux micros et 17 autres capteurs de toute sorte.
But : voir si la sélection saura retrouver les micros au milieu des 19 capteurs. De plus on change la trajectoire de la source pour se rapprocher de notre expérimentation robotique.

9. **Expérience** *23_TrajV_et_V*
   Source : lumineuse,
   Trajectoire : le même va-et-vien que précédement.
   Capteurs : les 23.
   But : simuler au mieux l'expérience robotique.
10. **Expérience** *robotik*
    Environnement : réel, on utilise le Koala et non plus le simulateur,
    Source : une balle rouge,
    Trajectoire : un va-et-vien proche de celui simulé et représenté en figure 4.7
    Capteurs : les 59 du Koala.
    But : valider sur des données du monde réel. Dans cette expérience, la sélection d'un sous-ensemble et la validation se fait sur deux jeux de données correspondant à deux expériences similaires mais distinctes.
11. **Expérience** *robotik2*
    C'est la même que *robotik* sauf que la validation se fera sur les mêmes données que l'apprentissage, ce qui donnera un meilleur taux de reconnaissance.

4.3 Validation

Dans la littérature, la sélection de caractéristiques est souvent un prétraitement pour faire de la classification. Les auteurs valident donc leurs sous-ensemble de variables par les performances du programme d'apprentissage utilisé ensuite. En général, ils comparent les taux de réussite de la classification, avant et après la sélection.

Dans notre cas, cette démarche est tout à fait pertinente car nous voulons aussi prévoir la valeur d'une variable avec moins d'informations. De plus, dans le contexte de la programmation bayésienne des robots, le choix de la méthode d'apprentissage à été très naturel : on utilise un apprentissage bayésien. En effet, c'est ce type d'apprentissage qui est utilisé sur les robots de l'équipe CYBERMOVE.

Comme nous l'avons vu dans l'introduction, nous supposons l'indépendance des variables conditionnellement à la classe. Ceci nous amène à utiliser un apprentissage bayésien naïf.

4.3.1 Validation croisée

Cette section répond au problème de l'estimation du taux de reconnaissance de l'algorithme de classification bayésien naïf muni du sous-ensemble de variables sélectionnées. Il est sensé d'estimer ce taux à l'aide d'un ensemble de données n'ayant pas servi pour l'apprentissage, afin déviter l'apprentissage "par cœur". Toutefois, utiliser un grand ensemble test parmi nos exemples peut apparaître commme une perte d'information lorsque le nombre de données est restreint. Un petit ensemble test conduirait en revanche à des conclusions peu fiables. Une solution, appelée *validation croisée* [Rip96], consiste à diviser l'ensemble des données en m parties. Pour chaque partie, on fait l'apprentissage

sur les données complémentaires de cette partie, et on teste le classificateur sur la partie en question. On obtient ainsi m taux de reconnaissance dont on fait la moyenne. Ceci permettant de diminuer la variance du taux de prédiction estimé.

De plus, on peut montrer que si $m = T$ et tend vers l'infini, alors le taux estimé s'approche du taux de reconnaissance théorique fondé sur la vraissamblance (voir [coi01]). Ceci nous amène à une méthode : la validation croisée *leave-one-out* qui consiste donc à apprendre sur tous les exemples sauf un, à faire la prédiction sur le dernier exemple et à calculer le taux de reconnaissance en réitérant ceci pour chaque exemple. Cette méthode est assez couteuse car on réalise T apprentissages bayésiens coûtant chacun NC^2 opérations simples.

Pour cette raison, et parceque nous avons à notre disposition un assez grand nombre de données (le simulateur en produit à la demande et le robot en génère dix par seconde), nous avons choisit une méthode de **validation plus simple**. On apprend sur un fichier de données et on mesure le taux de reconnaissance sur un autre fichier, similaire mais différent. Dans le cas du simulateur le fichier de test correspond au même mouvement de la même source, mais il diffère de par le bruit des capteurs qui n'est pas le même (en fait le générateur de hasard a été initialisé différemment). Ceci nous rapproche des conditions réelles dans lesquelles le robot fait face à de nouveaux stimulis. Pour le cas des données robotiques, on a validé avec le même fichier et avec un fichier relevé lors d'une autre expérience.

4.4 Bilan

Ainsi nous utilisons deux sources de données pour tester nos algorithmes : un simulateur logiciel et un robot Koala. Une description des expériences a été donnée et une méthode de notation des algorithmes proposée.

Chapitre 5

Résultats

Pour interpréter les résultats et comparer nos algorithmes, nous nous sommes basés sur les critères suivants :

- Nombre de variables retenues,
- Taux de reconnaissance sur un fichier similaire mais différent de celui ayant servi à l'apprentissage,
- Temps de calcul sur PC 400 MHz,
- Pouvoir explicatif du sous-ensemble, capacité de généralisation (sont-ce les "bonnes" variables ?),
- Nombre de paramètres à fixer à la main.

Nous commencerons par exhiber les chiffres bruts des expérimentations, puis nous nous lancerons dans des analyses de plus en plus fines des résultats.

5.1 Résultats bruts

Le tableau 5.1 ci-dessous présente les résultats numériques des expériences simulées (F de 1 à 9) et sur le robot (10 et 11). Les fichiers sont donnés dans le même ordre que dans la section 4.2, soit *2thetas*, *theta_et_3bruits*, *8proxis_et_3bruits*, *2fois8proxis*, *8proxis8lumis*, *les23*, *2sound_3bruits*, *sound19_TrajV_et_V*, *23_TrajV_et_V*, *robotik* et *robotik2*. Les algorithmes sont ceux définis dans le chapitre 3, les résultats de classement de variables par Rank XH et Rank χ^2 ne sont pas présentés ici. La dernière colonne (**ALL**) donne le nombre total de variables du fichier d'exemples, le taux de réussite de la classification en éliminant aucune de ces variables et le temps de calcul de la classification.

D'autre part F désigne les fichiers de données, k le nombre de variables retenues par l'algorithme, Tx le taux de réussite de la classification avec le sous-ensemble trouvé, et t est le temps de calcul en secondes (3 k = 3 000 s.). Pour les algorithmes EX et EX2, X signifie que le temps de calcul était trop grand pour permettre une expérimentation.

F		AGBN	AGXH	FAXH	FAχ^2	$\chi^2 Test$	EX	EX2	MoCa	ALL
1	k	2	1	2	2	2	1	1	1	2
	Tx	1	1	1	1	1	1	1	1	1
	t	2.7	0.31	0.0028	0.0026	0.0017	0.27	0.14	0.0029	0.0014
2	k	3	1	1	1	1	1	2	2	4
	Tx	1	0.18	1	1	1	1	1	1	1
	t	3.9	0.18	0.0035	0.0031	0.0016	2.05	0.72	0.0031	0.0024
3	k	11	8	8	1	4	9	5	5	11
	Tx	0.81	0.79	0.76	0.24	0.56	0.79	0.62	0.62	0.81
	t	13	0.42	0.14	0.0072	0.0043	664	236	0.0089	0.006
4	k	12	15	16	1	9	13	8	8	16
	Tx	0.81	0.78	0.77	0.25	0.74	0,64	0.57	0.65	0.77
	t	13	0.49	0.72	0.012	0.0098	15 k	2 k	0.018	0.01
5	k	14	8	8	1	13	9	8	8	16
	Tx	0.93	0.88	0.9	0.32	0.93	0.91	0.90	0.9	0.92
	t	14	0.33	0.34	0.012	0.01	17 k	3 k	0.015	0.0088
6	k	11	9	9	1	15	X	X	11	23
	Tx	0.93	0.86	0.9	0.5	0.91	X	X	0.92	0.89
	t	15	0.39	0.55	0.014	0.012	X	X	0.024	0.012
7	k	4	2	2	1	2	X	X	2	5
	Tx	0.95	0.68	0.94	0.58	0.94	X	X	0.94	0.94
	t	4.8	0.18	0.0077	0.0037	0.0022	X	X	0.0036	0.003
8	k	6	2	2	1	5	X	X	9	19
	Tx	0.81	0.37	0.83	0.78	0.81	X	X	0.78	0.74
	t	7.5	0.25	0.028	0.011	0.0063	X	X	0.018	0.01
9	k	10	5	5	1	8	X	X	11	23
	Tx	0.82	0.79	0.8	0.42	0.79	X	X	0.78	0.74
	t	12	0.32	0.16	0.014	0.0088	X	X	0.023	0.013
10	k	35	15	15	1	45	X	X	29	58
	Tx	0.33	0.25	0.25	0.05	0.19	X	X	0.27	0.19
	t	1500	19	390	2.1	2.8	X	X	2.6	1.2
11	k	37	15	15	1	45	X	X	29	58
	Tx	0.91	0.8	0.8	0.4	0.89	X	X	0.87	0.89
	t	3 k	29.4	535	3.4	3.1	X	X	1.03	1.3

TAB. 5.1 – *Résultats bruts présentés par algorithme (colonne), et par fichier (ligne).* **k** *est le nombre de variables retenues,* **Tx** *le taux de reconnaissance du classificateur et* **t** *le temps de calcul. On n'a pas détaillé les sous-ensembles de variables sélectionnés.*

5.2 Première discussion

On commence par analyser rapidement les résultats afin d'éliminer tout de suite les mauvais algorithmes.

Moyenne sur tous les fichiers

%	AGBN	AGXH	FAXH	FAχ^2	$\chi^2 Test$	ModCap	ALL
NbVar	62	34	35	5	63	45	100
TxReco	105	83	101	62	99	98	100
TpsCalc	4586	51	927	6	6	4	1

TAB. 5.2 – *Moyennes des résultats. Les valeurs sont exprimées relativement à la dernière colonne qui présente* **k=N**, **Tx** *et* **t** *pour l'ensemble de toutes les variables.*

Le tableau 5.2 présente une moyenne sur les 11 fichiers de tests des résultats bruts. De plus, ils ont été exprimés par rapport aux résultats sans élimination de variables. Par exemple, l'algorithme AGBN a, en moyenne, gardé 62% des variables tout en améliorant faiblement le taux de réussite de la classification d'un facteur 1,04 (104 % du taux originel avec ALL). On voit par contre qu'il est très couteux de l'utiliser ($\frac{4586}{51} = 90$ fois plus lent que AGXH). Remarque : les moyennes ont été arrondies à l'entier le plus proche.

Moyenne sur les "gros" fichiers

%	AGBN	AGXH	FAXH	FAχ^2	$\chi^2 Test$	ModCap	ALL
NbVar	52	25	25	3	59	41	100
TxReco	106	87	102	64	104	103	100
TpsCalc	2273	23	401	3	2	1	1

TAB. 5.3 – *Moyennes des résultats sur les fichiers les23, sound19_TrajV_et_V, 23_TrajV_et_V et robotik2.*

Le tableau 5.3 à été constuit sur le même principe que le tableau 5.2 sauf qu'on n'a considéré que les fichiers d'expériences réalistes : *les23*, *sound19_TrajV_et_V*, *23_TrajV_et_V* et *robotik2*.

Conclusion

On peut d'emblée mettre de côté les 3 *wrappers* qui sont vraiment trop lents pour de la robotique autonome. EX et EX ne sont pas représentés dans les tableaux car ils n'auraient pas eu le temps de terminer dans la durée du stage. Dans l'expérience *robotik2* AGBN a mis près d'une heure. Ceci nous pousse à le rejeter car ce calcul est beaucoup trop lourd pour un processeur embarqué. Cependant, dans le cadre d'une autre application *off line*, il aurait été pertinent de le garder car, comme on l'attendait il maximise le taux de bonnes classifications.

On élimine aussi FAχ^2 qui, sur les gros exemples, insiste pour finir trop tôt l'exploration de l'espace des sous-ensembles, et donc qui amène un trop petit TxReco. Cet arrêt prématuré (3% = 1 variable retenue) s'explique par le fait que la statistique du χ^2 n'est pas adaptée à la formule de Ghiselli, la moyenne des intra-corrélations l'emporte toujours sur la moyenne des corrélations avec θ.

5.3 Discussion fine

%	AGXH	FAXH	$\chi^2 Test$	ModCap
NbVar	25	25	59	41
TxReco	87	102	104	103
TpsCalc	23	401	2	1

TAB. 5.4 – *Moyennes des résultats sur les fichiers les23, sound19_TrajV_et_V, 23_TrajV_et_V et robotik2 et pour les algorithmes AGXH, FAXH, $\chi^2 Test$ et ModCap.*

$\chi^2 Test$ et ModCap donnent un TxReco équivalent, mais $\chi^2 Test$ garde trop de variables, donc on l'élimine. De plus $\chi^2 Test$ nécessite l'ajustement du seuil α qui est délicat à réaliser.

Comme l'estimation de complexité l'avait prévu ($O(N^3)$), FAXH est assez lent. Ceci est dû à sa méthode de parcours de l'espace des sous-ensembles qui est très répétitive. Elle pourrait être optimisée en rendant certains calculs incémentaux.

5.3.1 AGXH et ModCapt

À ce stade, nos meilleurs candidats sont AGXH et ModCap. ModCap donne un meilleur taux, mais il garde plus de variables. Cependant il faut rappeller que le critère d'arrêt de cet algorithme est justement de sélectionner $\frac{N}{2}$ variables. Pour mieux le comparer on a refait des tests en changeant ce critère d'arrêt par $\frac{N}{4}$, puis 8 variables. Les résultats sont les suivants, sur les "gros" fichiers seulement :

%	AGXH	ModCap$\frac{N}{2}$	ModCap$\frac{N}{4}$	ModCap8
NbVar	25	41	23	26
TxReco	87	103	99	97
TpsCalc	23	1	1	1

TAB. 5.5 – *Moyennes des résultats sur les "gros" fichiers pour AGXH et ModCap avec comme critères d'arrêts $\frac{N}{2}$, $\frac{N}{4}$ et 8 variables.*

On contate alors qu'avec un nombre de variables retenues équivalent ($\frac{N}{4}$), ModCap reste plus performant que AGXH, et beaucoup plus rapide. Cet algorithme semble donc le meilleur, si l'on considère qu'on sait *a priori* le nombre de variables à garder. Or, ce n'est pas du tout le cas dans le cadre de notre application robotique dans lequel on veut obtenir un maximum d'autonomie.

Etude de AGXH

Les paramètres de l'algorithme génétique lors de ces tests étaient les suivants :
- 100 générations,
- 50 individus,
- 60 % de probabilité de *crossover*,

– 2 % de probabilité de mutation.

On lit dans le tableau 5.5 que le TxReco avec AGXH est de 87% alors qu'avec FAXH il est de 102%. C'est étonnant car ces deux algorithmes ne diffèrent que dans leur méthode d'exploration de l'espace des sous-ensembles ; leur fonction d'adaptation est strictement la même. AGXH devrait pouvoir trouver le même sous-ensemble que FAXH. Par exemple, sur le fichier *2sound_3bruits*, AGXH donne TxReco = 68% alors

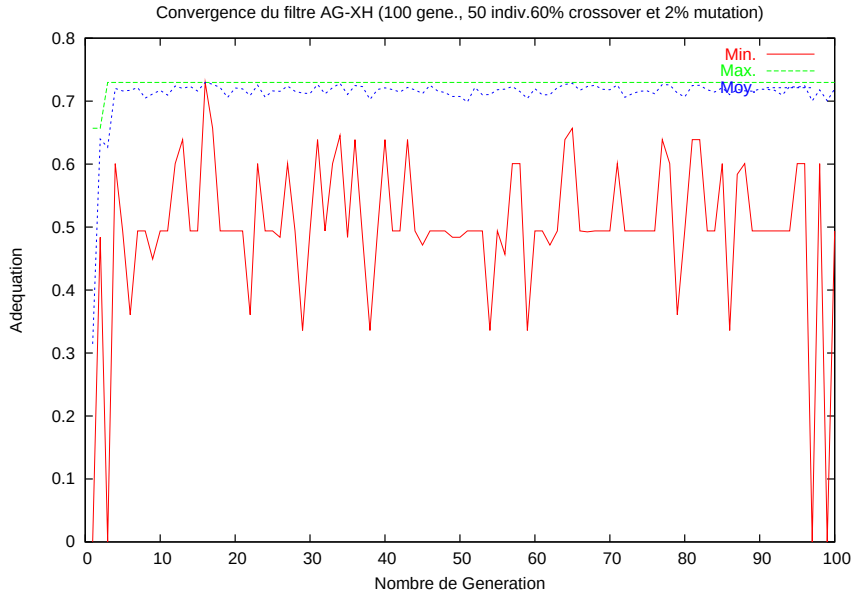

FIG. 5.1: *Allure de la convergence de AGXH sur le fichier 2sound_3bruits. La courbe du haut représente l'évaluation du meilleur individu de chaque génération, celle du milieu la moyenne, et la dernière le pire individu.*

que FAXH 94 %. En effet, FAXH sélectionne bien les deux micros, mais AGXH retient un micro et un bruit. En analysant la convergence de l'algorithme génétique, on voit (figure 5.1) qu'elle se fait très rapidement et que plus rien ne bouge ensuite. Nous pouvons raisonnablement émettre deux hypothèses :

1. Soit l'algorithme génétique n'a pas trouvé l'individu optimal et il faut alors modifier ses paramètres.
2. Soit l'individu maximisant l'adéquation (calculée par la formule de Ghiselli et par l'entropie) ne maximise pas le taux de réussite de la classification. Dans ce cas la conclusion est que FAXH a trouvé les deux micros par chance.

Notre parcours exhaustif de l'espace nous permet de trancher : en testant tous les sous ensembles, on constate que c'est bien l'ensemble des deux micros qui maximise l'adaptation **et** le taux de reconnaissance. On conclue que l'algorithme génétique n'a pas bien exploré l'espace et qu'il faut changer ses paramètres.

Ce problème s'est produit seulement sur trois fichiers : *theta_et_3bruits*, *2sound_3bruits* et *sound19_TrajV_et_V*). Sur les autres le taux de réussite est correct.

Etude de MoCap

A part le problème du critère d'arrêt, l'algorithme MoCap s'est bien comporté. Il est rapide et les variables sélectionnées sont les bonnes dans le sens où elles entrainent un taux de reconnaissance souvent supérieur à celui donné par l'ensemble des N variables. Dans les expériences *theta_et_3bruits* et *8proxis_et_3bruits*, ses résultats sont mauvais car il est contraint à sélectionner un nombre insuffissant de variables. En revanche dans *2fois8proxis* il garde bien les proximètres et dans *8proxis8lumis* les photomètres. On voit d'ailleurs dans cette expérience que les photomètres sont plus corrélés avec θ que les proximètres. AGXH les a aussi retenus avec en plus un des proximètres.

Dans les "grosses" expériences, on constate que MoCap est moins performant qu'AGXH dans l'élimination des variables redondantes, mais qu'il est sait mieux débusquer les variables indépendantes de θ.

5.4 Comparatif

Après une étude détaillée des sous-ensembles générés, nous pouvons résumer nos conclusions dans le tableau 5.6.

Critère	AGBN	AGXH	FAXH	$\chi^2 Test$	ModCap
Temps	—	+	—	++	+++
TxReco	++++	+	++	+++	+++
Nombre variables	–	+	+	–	paramètre
Explication	–	+	+	+	++
Paramètres	-	-	++	-	—

TAB. 5.6 – *Forces et faiblesses des différents algorithmes. Le nombre de variables retenues par ModCap est l'un de ses paramètres. La ligne "explication" exprime si l'algorithme a bien sélectionner les variables "logiques", par exemple des photomètres si la source est lumineuse. La ligne "paramètres" exprime la difficulté à trouver les bons paramètres de l'algorithme.*

5.5 Retour sur le changement de modalité

En regardant de plus près les résultats de l'expérience de la balle rouge, on constate d'abord que sur les 58 variables initiales le test du chi^2 et AGBN ont sélectionné presque toutes les variables non constantes. Ce défaut pourrait être corrigé en modifiant leurs paramètres (baisser les seuil de signification du test et modifier la fonction d'évaluation de l'algorithme génétique).

D'autre part on lit sur le tableau 5.7 que les taux de reconnaissance sont faibles, même avec toutes les variables. Ceci est du à l'expérience elle même et non au méthodes de sélection. Etant donné que, lors de l'expérience, la balle ne faisait pas le tour entier du robot, tous les proximètres n'ont pas été stimulés.

On constate alors que AGXH, FAXH et ModCap$\frac{N}{4}$ ont des performances équivalentes. Mais comme les deux premiers sont plus lents et comme il faut dire ex-

	AGXH	FAXH	ModCap$\frac{N}{2}$	ModCap$\frac{N}{4}$	AGBN	ALL
Proximètres retenus (sur 16)	6	7	16	6	13	16
Autres cap. retenus (sur 42)	9	8	13	8	22	42
TxReco	0.256	0.28	0.27	0 .23	0.33	0.19

TAB. 5.7 – *Nombre de proximètres, d'autres capteurs retenus et taux de reconnaissance lors de l'expérience de la balle rouge.*

plicitement à ModCap de garder $\frac{N}{4}$ variables, nous considérons que AGXH est le plus adapté à la découverte d'un nouvelle modalité sensorielle pour cette expérience.

Chapitre 6

Conclusion

6.1 Contribution

Dans le but de permettre la découverte de nouvelles modalités sensori-motrices, nous avons comparé plusieurs méthodes de sélection de variables. Pour cela nous avons testé différents algorithmes selon des critères objectifs de performances et de coût. Nous en avons conclu que bien que le choix final dépende de l'application visée, deux algorithmes sortent du lot : MoCapt et AGXH. Tous deux sont basés sur une mesure entropique, mais il diffèrent de par leur méthode de parcours de l'espace des sous-ensembles de variables.

6.2 Perspectives

6.3 Perspectives techniques

Il serait judicieux de poursuivre l'étude de l'algorithme génétique afin d'optimiser ses nombreux paramètres pour le rendre plus rapide et plus sélectif. Il serait possible de modifier sa fonction d'adaptation en ajoutant une pénalité dépendant du nombre de variables retenues afin de le forcer à en éliminer plus. On pourrait aussi complexifier la formule de Ghiselli afin de détecter les dépendances conditionnelles d'ordre supérieure à deux, mais ceci suppose une amélioration pronfonde de la théorie. On pourrait en outre étudier le cas de variables continues ; ce qui est plus difficile car les calculs de sommes se transforment en intégrales. La dernière voie à explorer en priorité serait de trouver un procédé pour changer le critère d'arrêt de l'algorithme ModCap sans en changer la complexité. Alors celui-ci serait un excellent candidat pour une application embarquée.

6.4 Perspectives générales

A un plus haut niveau d'abstraction, nous n'avons réalisé qu'un premier pas vers la découverte entièrement automatique de nouveaux comportements. Des questions ouvertes sont : comment le robot indentifiera-t-il les périodes pendant lesquelles il doit tenter une telle découverte ? Doit-il le chercher en permanence et en parallèle avec ses autre tâches ?

C'autre part, nous ne nous sommes intéressé qu'au variables sensorielles, mais il est tout aussi pertinent de rechercher des corrélations dans le domaine moteur. Quels seraient les enjeux d'une telle approche ?

Notre approche trouverait une application intéressante sur le robot BIBA de l'équipe. Ce robot se localise dans un couloir grâce à un capteur laser très performant. Cependant, la localisation devient presque impossible lorsqu'il passe devant des portes vitrées transparentes. Comme ce robot est aussi équipé de proximètres, on pourrait intégrer nos méthodes afin qu'il comprenne, lorsque tout va bien, que ses proximètres donnent une information sur la distance aux murs, et, quand il se perd à cause de vitres, il passe automatiquement en mode proximétrique pour combler la faiblesse du capteur laser.

Bibliographie

[AD94] Hussein Almuallim and Thomas G. Dietterich. Learning boolean concepts in the presence of many irrelevant features. *Artificial Intelligence*, 69(1-2) :279–305, 1994.

[BEHW87] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Occam's razor. *Information Processing Letters*, 24(6) :377–380, 1987.

[BtLrG03] Pierre Bessière and the LAPLACE research Group. Survey : Probabilistic methodology and techniques for artefact conception and development, 2003.

[coi01] Olivier François. Notes de cours de réseaux de neurones. grenoble, france, 2001.

[DP97] Pedro Domingos and Michael J. Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, 29(2-3) :103–130, 1997.

[Gau02] Olivier Gaudoin. Méthodes statistiques pour l'ingénieur. grenoble, france, 2002.

[Ghi64] Edwin E. Ghiselli. *Theory of Psychological Measurement*. McGraw-Hill Book Company, 1964.

[Hal98] M. Hall. Correlation-based feature selection for machine learning, 1998.

[Hal00] Mark A. Hall. Correlation-based feature selection for discrete and numeric class machine learning. In *Proc. 17th International Conf. on Machine Learning*, pages 359–366. Morgan Kaufmann, San Francisco, CA, 2000.

[HGC94] David Heckerman, Dan Geiger, and David Maxwell Chickering. Learning bayesian networks : The combination of knowledge and statistical data. In *KDD Workshop*, pages 85–96, 1994.

[Hol75] John Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, 1975.

[IG03] André Elisseeff Isabelle Guyon. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 2003.

[JKP94] George H. John, Ron Kohavi, and Karl Pfleger. Irrelevant features and the subset selection problem. In *International Conference on Machine Learning*, pages 121–129, 1994. Journal version in AIJ, available at http ://citeseer.nj.nec.com/13663.html.

[KJ97] Ron Kohavi and George H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2) :273–324, 1997.

[KL51] S. Kullback and Leibler. On information and suoeciency. page 22 :7986, 1951.

[KS77] Sir Maurice Kendall and Alan Stewart. *The Advanced Theory of Statistics, Volume 1, 4th Edition*. Mcmillan Publishing, New York, 1977.

[KS96] Daphne Koller and Mehran Sahami. Toward optimal feature selection. In *International Conference on Machine Learning*, pages 284–292, 1996.

[LOE03] Diard J. Lebeltel O., Bessière P. and Mazer E. Bayesian robots programming. *Autonomous Robot 2003 (in press)*, 2003.

[LS95] H. Liu and R. Setiono. Chi2 : Feature selection and discretization of numeric attributes, 1995.

[LY02] H. Liu and L. Yu. Feature selection for data mining, 2002.

[MT01] Dimitris Margartitis and Sebastian Thrun. A bayesian multiresolution independence test for continuous variables. In *Uncertainty in Artificial Intelligence : Proceedings of the Seventeenth Conference (UAI-2001)*, pages 346–353, San Francisco, CA, 2001. Morgan Kaufmann Publishers.

[NF77] P. Narendra and K. Fukunaga. A branch and bound algorithm for feature subset selection. In *IEEE Transactions on Computers*, pages 917–922, 1977. 26(9).

[O.99] Lebeltel O. Programmation bayésienne des robots. thèse de l'institut national polytechnique de grenoble, france, 1999.

[Rea02] Oliver Ritthoff and et al. A hybrid approach to feature selection and generation using an ea, 2002.

[Rip96] Brian D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, Cambridge, United Kingdom, 1996.

[Spa99] Anne Spalanzani. Algorithmes évolutionnaires pour l'étude de la robustesse des systèmes de reconnaissance de la parole, thèse de l'université joseph fourier, 1999.

[Wol94] David R. Wolf. Mutual information as a bayesian measure of independence, 1994.

[YH98] Jihoon Yang and Vasant Honavar. Feature subset selection using a genetic algorithm. *IEEE Intelligent Systems*, 13 :44–49, 1998.

[ZH02] Marco Zaffalon and Marcus Hutter. Robust feature selection by mutual information distributions, 2002.

Chapitre 7

Annexes

7.1 Notion d'indépendance

Dans le cadre de la programmation bayésienne comme dans toute autre modélisation, il faut choisir les variables ou grandeures dignes d'intérêt et écarter les autres, en particulier celles qui sont indépendantes. Mais bien que la notion d'indépendance soit bien définie dans le cadre de la théorie des probabilités ($P(X,Y) = P(X)P(Y)$), cette notion n'est pas aussi intuitive qu'il n'y paraît.

Par exemple la seule particularité en cas d'indépendance de la distribution conjointe $P(X,Y)$ est d'être un produit tensoriel. Si X et Y sont des grandeures continues, il existe des fonctions h et g telles que $P(X = x, Y = y) = h(x)g(y)$. En ce sens, l'espace des variables indépendantes est de mesure nulle dans l'espace des variables dépendantes. En prenant deux variables au hasard, la probabilité qu'elles soient indépendantes est donc nulle. Ainsi il faut supposer la dépendance et démontrer l'indépendance.

La figure 7.1 présente 15 densités à 2 variables. Quelles sont les densités de variables indépendantes ? La réponse est : 1, 2, 5, 6, 8, 11 (qui est représenter être une gaussienne), 12 et 15.

Mais la notion la moins intuitive est celle de dépendance conditionnelles. X est indépendante de Y conditionnellemnent à Z si $P(XY|Z) = P(X|Z)P(Y|Z)$. Il est surprenant de constater que des variable indépendantes peuvent devenir conditionnellement indépendantes. Et vice-versa. Par exemple, soit X l'événement "mon gazon est mouillé" et Y : "le gazon du voisin est mouillé". X et Y sont dépendants mais deviennent indépendants conditionnellement à Z : "il a plut cette nuit". En effet Y n'apporte pas de nouvelle information sur X quand Z est déjà connu. Ce cas de figure se produit lorsque Z est une cause, *i.e.* une source dans le réseau bayésien correspondant. Le cas contraire se produit lorsque Z est un puit.

7.2 Les algorithmes génétiques

Les algorithmes génétiques ont été introduits et développés vers 1975 par John Holland [Hol75], ses collègues et ses étudiants. Informellement, un algorithme génétique est un algorithme heuristique d'optimisation globale qui permet d'explorer de très grands espaces. Ils peuvent être appliqués à tout problème pour lequel on dispose d'une fonction

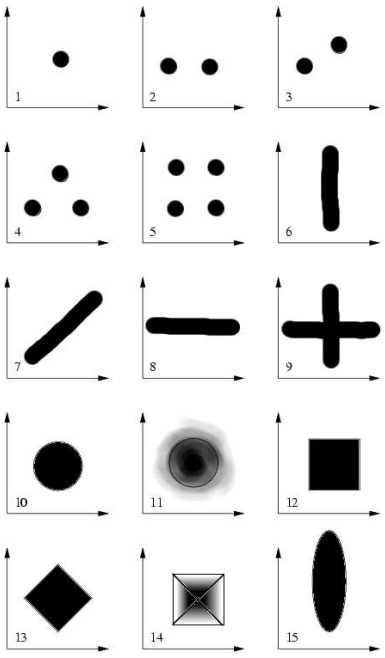

FIG. 7.1: **Quizz** *: Parmi ces 15 distributions, 8 représentent des couples de variables indépendantes. Lesquelles ?*

d'évaluation des solutions potentielles.

Le principe d'un tel algorithme est de créer au hasard une population de solutions potentielles, puis de la faire évoluer. La population de solutions est soumise à une imitation de l'évolution des espèces naturelles : mutations et reproductions par hybridation. En favorisant la survie des individus les plus "aptes" (les solutions dont l'adaptation est la plus grande), on provoque l'apparition d'individus meilleurs que leurs parents.

Le tableau 7.1 présente les caractéristiques d'un algorithme génétique en général et dans notre application.

Un algorithme génétique a la structure suivante :

```
t=0;
Initialisation de P(t)
Evaluation de P(t)
Tant que non(critere d'arret)
   P'(t) = Selectionner( P(t) )
P"(T) = Evoluer ( P'(t) )
P(t+1) = P"(t)
t = t+1
Fin tantque
```

7.3 Code source

Le programme se compose de trois parties indépendantes :

Caractéristiques	**Application**
Nature d'un individu	Un sous-ensemble de variables
Espace de recherche	sous-ensembles de variables (2^N éléments)
Codage d'un individu	Ensemble de N bits
Taille de la population	Variable
Nombre de générations	Variable
Fonction d'évaluation	Voir chapitre 3
Type d'hybridation	Cross-over et mutations
Type de sélection	Tournoi

TAB. 7.1 – *Caractéristiques d'un algorithme génétique en général et dans notre application.*

- Une partie *Simulateur* détaillée dans la section 4.1.1 (voir le diagramme de classes figure 7.2.
- Une partie *Analyseur* qui contient tous les algorithmes présentés.
- Une partie *Validateur* qui implémente les algorithmes de validation et du classificateur bayésien naïf. Ceux-ci nous permettent de mesurer les performances des algorithmes de sélection de variables.

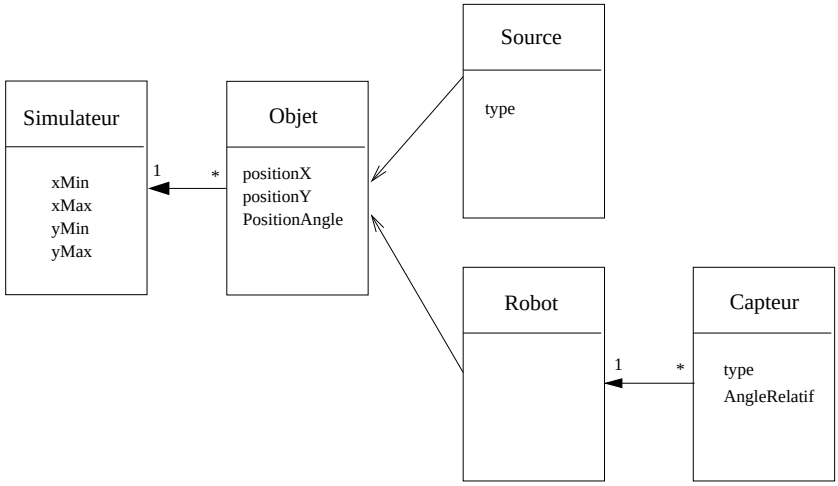

FIG. 7.2: *Diagramme de classes simplifié du simulateur.*

Le langage utilisé est C++ objet.

Les sources sont disponibles dans le répertoire **CVS** de l'équipe **CYBERMOVE**, dans le dossier **PierreD**. Voir *http ://indigo/cgi-bin/cvsweb/PierreD/.*