Rpublique Algrienne Dmocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Abderrahmane Mira de Béjaïa

Faculté des Sciences Exactes
Département d'Informatique

Mémoire de fin de cycle

En vue de l'obtention du diplôme de master en informatique
Spécialité : Réseaux et Systèmes distribués ReSyD

Thème

Classification des informations contextuelles dans un environnement intelligent ubiquitaire

---

Réalisé par :

Wissam AZNI
Faïza BOUDJEMIL

Devant le jury composé de :

Président :	M^{elle}	Amel	KOUICEM	Université de Béjaïa
Examinateur :	M^{elle}	Souhila	GHANEM	Université de Béjaïa
Examinateur :	M^{elle}	Nassima	BOUADEM	Université de Béjaïa
Encadreur :	M^{elle}	Salima	SABRI	Université de Béjaïa
CoEncadreur :	M^{me}	Malika	YAICI	Université de Béjaïa

Promotion 2014.

Remerciements

Nous tenons à remercier sincèrement nos encadreurs Salima SABRI et Malika YAICI, pour la pertinence de leur encadrement, leurs valeureux conseils et encouragements, et surtout pour leurs qualités humaines. Leur rigueur scientifique, leurs conseils judicieux et leur disponibilité tout au long de ce mémoire ont joué un rôle déterminant et nous ont permis de mener à terme ce travail de recherche avec les qualités escomptées. Avec nos encadreurs, nous avons appris la flexibilité, l'efficacité et la patience. Ils demeurent des professeurs envers qui nous avons beaucoup de respect.

Nous tenons aussi à remercier les membres de jury pour avoir accepté de juger notre travail.

Á

la mémoire de mes grands parents
mon père
ma mère
ma soeur Kahou
mes deux frères Hamza et Ali
tous les autres membres de ma famille
tous mes amis sans exception

Wissam.

Á

mon père
ma mère
la mémoire de mes défunts grands parents
ma grand mère
mes soeurs et frères
mes belles soeurs et mes beaux frères
tous les autres membres de ma famille
tous mes amis

Faïza.

Abréviations

AVQ Activités de la Vie Quotidienne

CART Classification And Regression Tree

CC/PP Composite Capability/Preference Profiles

CML Context Modeling Language

GPS Global Positioning System

HMM Hidden Markov Model

ID3 Iterative Dichotomiser 3

KPPV K Plus Proches Voisins

LMS Least Mean Square

OMG Object Management Group

ORM Object Role Modeling

PAM Partitionning Around Medoids

PDA Personal Digital Assistant

RAH Reconnaissance d'Activité Humaine

RDF Ressource Description Framework

RFID Radio Frequency IDentification

ROC Receiver Operating Characteristic

SVM Support Vector Machine

UML Unified Modeling Language

W3C World Wide Web Consortium

WGS84 World Geodetic System 84

XML eXtensible Markup Language

Table des matières

Table des figures

Liste des tableaux

Introduction générale

Grâce aux progrès réalisés, ces dernières années, dans la miniaturisation de composants électroniques et à l'émergence des technologies réseaux sans fil, un nombre croissant d'objets de notre quotidien intègrent des dispositifs électroniques grâce auxquels ils deviennent communicants. Ces objets sont, selon le cas, soit mobiles ou immobiles. Tout objet physique de l'environnement de l'utilisateur peut être potentiellement communicant et être ainsi capable d'interagir avec l'utilisateur et/ou avec d'autres objets de façon autonome. Finalement, l'utilisateur se retrouve au centre d'un espace composé d'objets physiques hétérogènes, dotés de capacités de communication filaire ou non. Un tel environnement qui forme d'une fédération spontanée et dynamique d'objets communicants sera dit "Ubiquitaire".

L'un des éléments centraux de l'informatique ubiquitaire est la capacité à reconnaître et à comprendre ses utilisateurs dynamiquement. Cette capacité peut être produite par l'obtention des informations courantes et mises à jour de l'utilisateur qui représentent son contexte. Le contexte des utilisateurs peut être n'importe quelle information qui concerne leur statut, leur localisation, leur environnement, leur activité, et ainsi de suite. L'informatique sensible au contexte est ainsi introduite pour adresser les chalenges de leur reconnaissance et de leur compréhension.

La reconnaissance du contexte correspond à l'utilisation des techniques d'apprentissage automatique pour déterminer des descriptions de contexte de niveau supérieur à partir des données provenant généralement de capteurs. Dans le cadre de ce travail, nous nous intéressons à un contexte relatif à la personne, il s'agit de l'activité de cette dernière.

La reconnaissance d'activité humaine à partir de données de capteurs dans les systèmes ubiquitaires est un domaine scientifique en pleine effervescence avec de nombreuses applications dans le domaine de la sécurité et de la vidéosurveillance de lieux publics. Cette recherche s'est récemment étendue à l'assistance à domicile, à l'assistance et à la surveillance dans les unités de soins spécialisés, ainsi qu'à la surveillance et à l'évaluation de procédés dans l'industrie.

Les techniques appliquées pour la reconnaissance des activités humaines se rapportent en majorité aux techniques développées dans le domaine de l'intelligence artificielle et data mining. Il s'agit des techniques de classification de données. Ces techniques rencontrent quelques difficultés pour faire face aux problèmes de bruit, d'annotation de données et de leur incertitude, d'apprentissage en fonction de la classe d'appartenance et de la difficulté de l'implémentation des modèles de classification.

Dans le domaine de la reconnaissance des activités dans les environnements intelligents, la problématique de ce mémoire de recherche consiste à proposer une technique de classification permettant de remédier aux limites des méthodes existantes.

Le présent rapport est composé de quatre chapitres, le premier chapitre dresse un tour d'horizon sur les systèmes ubiquitaire, il pourvoit une vue d'ensemble sur les environnements ubiquitaires. Le deuxième chapitre met en évidence la notion de contexte et souligne les éléments de contexte que nous allons utiliser. Le troisième chapitre concerne les techniques de classification pour la reconnaissance des activités humaines existantes. Le quatrième et dernier chapitre décrit et explique notre contribution. Nous clôturons ce mémoire avec une conclusion générale et des perspectives.

1

Introduction à l'informatique ubiquitaire

1.1 Introduction

Smartphones, tablettes, 3G et autres technologies ont bouleversé la manière dont on utilise les différents systèmes qui prennent place dans notre quotidien. Cette informatique pervasive est devenue invisible à nos yeux, puisque nous utilisons continuellement des ressources informatiques sans forcément les percevoir en tant qu'ordinateurs [28]. Ce chapitre introduit les concepts liés à l'*informatique ubiquitaire*, un paradigme récent de l'*informatique personnelle*. Cette informatique ubiquitaire permet la mobilité de l'utilisateur, et a pour objectif la possibilité d'accéder à tout moment, de tout endroit et avec tout média à toute information et à tout service. Ainsi, les objectifs des systèmes ubiquitaires ne sont plus restreints à l'exécution de tâches commandées par l'utilisateur, mais à faire communiquer plusieurs systèmes mobiles ou fixes pour fournir des services personnalisés à un seul usager.

1.2 Historique

Le concept d'informatique ubiquitaire ou informatique ambiante a été développé par Mark Weiser[52] au cours des années 90. Il revient sur l'évolution de l'informatique depuis les années 60, caractérisée par trois ères :

- Les mini-ordinateurs : une unité centrale partagée par un ensemble d'utilisateurs.
- Les ordinateurs personnels : une unité centrale par utilisateur.
- La mobilité : plusieurs unités centrales par utilisateur, ces unités centrales pouvant le suivre dans ses déplacements.

La miniaturisation des unités centrales, la réduction de la consommation d'énergie et la généralisation des réseaux (Wifi, 3G etc) conduit à une omniprésence de dispositifs informatiques nous accompagnant dans notre vie quotidienne : Smartphones, console de jeux portables, PDA, ordinateurs portables.... Les applications ne seront plus associées à une machine physique et surtout à un écran associé. Elles vont pouvoir migrer et nous suivre au gré de nos déplacements. Les interactions vont être plus naturelles et l'ordinateur va se fondre dans notre environnement et disparaître.

Une des premières réalisations associées à l'informatique ubiquitaire sont les trois dispositifs `tabs`[1] (1), `pads`[2] (2) et `boards`[3] (3) développés entre 1988 et 1994, la figure 1.1 illustre les dispositifs Tabs , Pads et Boards en situation de travail.

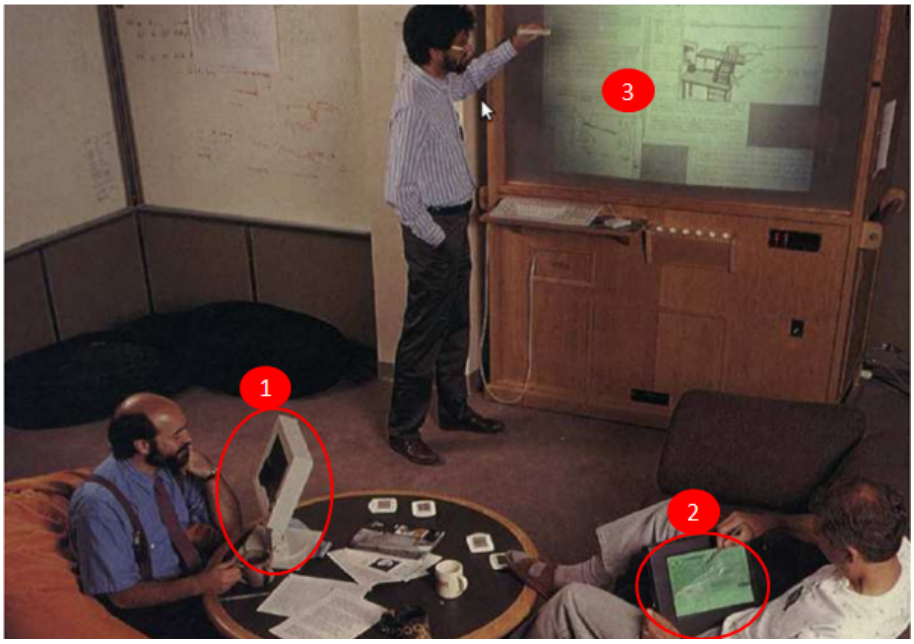

FIGURE 1.1 – Tabs, Pads, Boards en situation de travail [61]

On peut également citer les travaux du consortium Things That Think du laboratoire Media Lab du `M.I.T`[4]. Ce consortium cherche à inventer les objets du futur en embarquant en particulier l'informatique dans les objets de tous les jours. Le projet Sixth Sense par exemple, de l'équipe Fluid Interfaces Group, augmente les objets quotidiens en projetant dessus des informations supplémentaires. Cela peut servir par exemple à guider ses achats dans un supermarché.

1. Badges personnels de la taille d'un post-it
2. support mobile personnel de la taille d'un livre
3. support non mobile de la taille d'un tableau
4. Massachusetts Institute of Technology

1.3 Définition de l'informatique ubiquitaire

Le concept d'informatique ubiquitaire à été introduit par Weiser en 1991 dans son article The computer for the 21st Century [52]. concept qu'il a raffiné par la suite [53][55][54].

Dans ses articles, Weiser propose une révolution étonnante dans les technologies de l'information afin qu'elles développent complètement leur potentiel : les ordinateurs doivent devenir invisibles. La façon de les rendre invisibles consiste à les intégrer avec les objets et les endroits que nous utilisons dans la vie quotidienne. En effet, Weiser affirme : *Les technologies les plus profondes sont celles qui disparaissent. Elles se mêlent dans le tissu de la vie de tous les jours jusqu'au point où elles en deviennent indiscernables.*

Il considère l'informatique omniprésente comme la troisième vague de l'informatique, après la première, celle de l'ordinateur central et la deuxième, celle de l'ordinateur personnel.

Dans l'informatique ubiquitaire, les capacités informatiques s'intègrent dans la vie de tous les jours, dans les différents objets qui nous entourent, Un exemple d'un tel environnement est représenté par la figure 1.2 où l'on peut voir un bureau équipé de divers appareils (ordinateurs fixes (1) et portables (2), téléphones portables (3), haut-parleurs (4), vidéo-projecteurs (5), caméras (6), murs de microphones (7), antennes wifi, Bluetooth et RFID (8), etc.).

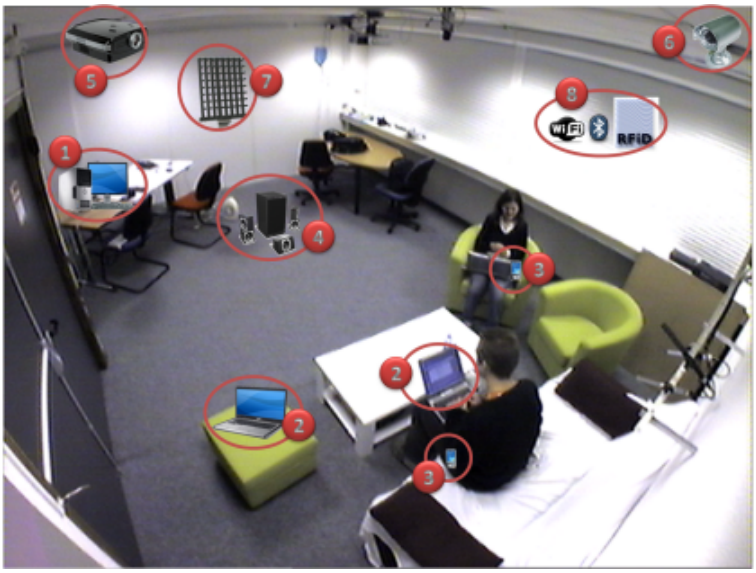

FIGURE 1.2 – Un exemple de bureau ubiquitaire [61]

Le but de cette nouvelle tendance en informatique est de faciliter et augmenter la qualité des interactions d'un utilisateur, que ce soit avec des objets informatiques ou avec les autres personnes (les collègues de bureau, les amis, etc.) à proximité ou à distance.

Le système ubiquitaire se voit ajouter des capacités de perception et d'analyse de l'environnement, des utilisateurs et de leur activité, ce qui lui permet de réagir en fonction du contexte (dont il est question dans le chapitre suivant). Pour ceci, il est nécessaire de formaliser et structurer les informations perçues par les différents capteurs qui vont équiper cet environnement. De cette façon, ces dispositifs divers vont travailler ensemble, de façon coordonnée, pour atteindre un but commun. La figure 1.3 illustre cette idée de réseau de dispositifs hétérogènes.

FIGURE 1.3 – Illustration des appareils hétérogènes intégrés dans l'informatique ubiquitaire

1.4 Caractéristique de l'informatique ubiquitaire

l'informatique ubiquitaire est aussi appelé dans la littérature académique *informatique diffuse*, *informatique pervasive*, *intelligence ambiante*, *informatique omniprésente*, ou encore *l'internet des objets*. Malgré toutes ces dénominations, l'idée centrale reste la même, l'informatique est omniprésente dans notre environnement, avec de nouveaux avantages mais aussi de nouveaux défis. les principales caractéristiques de l'informatique ubiquitaire sont résumées dans les points suivants :

- Ce sont des systèmes décentralisés, basés sur des réseaux décentralisés ;
- Le matériel informatique et les logiciels sont embarqués dans des objets de tous les jours, disparaissant dans l'environnement ;
- Les systèmes sont sensibles au contexte et s'adaptent aux besoins d'information ;
- L'utilisateur reçoit de l'assistance mobile à travers des services informatiques présents n'importe où, n'importe quand ;

- Les systèmes font une reconnaissance automatique et un traitement autonome des tâches répétitives ;
- Les systèmes sont hautement distribués, hétérogènes et mobiles.

1.5 Objectifs de l'informatique ubiquitaire

Les principaux objectifs de l'informatique ubiquitaire sont résumés ci-dessous :
- Offrir aux utilisateurs nomades un accés à l'information en fonction de leurs dispositifs
- Adapter l'information au contexte d'utilisation :
  - Les caractéristiques du dispositif d'accés ;
  - La localisation ;
  - Le moment de connexion ;
  - Les activités de l'utilsateur ;
- Adapter l'information au profil de l'utilisateur (préférences).

1.6 Défis de l'informatique ubiquitaire

L'hétérogénéité des objets communicants, la dynamicité des environnements physiques constituent deux des principaux défis des systèmes ubiquitaires.

1.6.1 Hétérogénéité

Par rapport aux premiers systèmes distribués, les systèmes ubiquitaires sont composés d'objets communicants hétérogènes dont les domaines applicatifs étaient jusqu'à récemment disjoints[25]. Ces objets communicants sont, par exemple, des terminaux téléphoniques, des ordinateurs, des appareils électroménagers, des appareils électroniques grand public ou des appareils médicaux. Ils hébergent une variété de services accessibles par des bus matériels et logiciels hétérogènes. Ces services héritent ainsi de l'hétérogénéité de leur objet communicant hôte. Pour assurer l'interopérabilité entre les objets communicants, leur hétérogénéité doit être abstraite par des outils capables d'intégrer un flot constant de nouveaux objets communicants.

1.6.2 Dynamicité

Les systèmes ubiquitaires requièrent de considérer des environnements physiques dont l'évolution est imprévisible. Notamment, la mobilité des utilisateurs et la volatilité des objets communicants rendent les systèmes ubiquitaires difficiles à appréhender. Les deux formes de dynamicité

dont les systèmes ubiquitaires sont dépendants sont la dynamicité des objets communicants et la dynamicité du contexte.

Objets communicants

La disponibilité des objets communicants varie au cours du temps. Ils sont mobiles car connectés sur des réseaux sans fil, ont des ressources limitées, ont des problèmes matériels ou des bogues logiciels les rendant inaccessibles, ou bien sont ajoutés ou supprimés des environnements physiques pour des raisons de maintenance. Les systèmes ubiquitaires doivent donc considérer la dynamicité des objets communicants en plus de leur hétérogénéité.

Contexte

Les systèmes ubiquitaires permettent l'automatisation de tâches auparavant attribuées aux utilisateurs. À l'instar d'un utilisateur, les systèmes ubiquitaires doivent être sensibles à leur environnement physique pour prendre les décisions appropriées. Ils doivent alors non seulement prendre en compte la dynamicité des objets communicants, mais aussi l'évolution des paramètres physiques de l'environnement physique. Ces paramètres physiques sont les phénomènes observables de l'environnement physique et inclut, par exemple, le température ambiante, la lumière du soleil ou la présence d'un utilisateur. Ils sont généralement mesurés par des objets communicants appelés capteurs. L'analyse de ces paramètres constitue le contexte des environnements. Par exemple, des valeurs élevées des paramètres physiques de température et de fumée peuvent constituer un contexte d'incendie. Les systèmes ubiquitaires analysent les changements de contexte pour décider du comportement à adopter et ainsi s'adapter.

1.7 Domaines d'application

Plusieurs domaines s'orientent de plus en plus vers l'adoption des applications ubiquitaires pour gagner la fidélité de leurs clients en leur proposant des services mobiles et intelligents qui nécessitent de moindres efforts de la part de l'utilisateur. Ces applications considèrent le contexte de la personne cible pour proposer des services personnalisés concernant différents domaines comme la santé, les loisirs, les voyages, les banques et le transport.

Parmi ces domaines nous citons par exemple le domaine médical où le patient est assisté en permanence par des capteurs qui détectent les informations sur sa santé et permettent grâce à des dispositifs mobiles d'accéder à des informations de diverses sources concernant la santé du patient, de conclure à son état et de détecter les situations d'urgence.

La recherche d'informations touristiques ou de loisir peut également trouver dans les systèmes ubiquitaires un champ d'application large puisque l'utilisateur souhaite dans la plupart des cas être informé, dans un contexte mobile, des lieux intéressants ou des hôtels près de sa localisation.

Il y a aussi le domaine de l'information voyageur, dans ce domaine un service peut être qualifié d'ambiant, s'il renseigne le voyageur en fonction de son environnement direct. Il s'agit donc de service capable d'adapter l'information en fonction des spécificités de l'utilisateur, de sa localisation selon le besoin afin de l'aider à réaliser son voyage. Dans cette catégorie, nous pouvons citer par exemple des services aidant les voyageurs handicapés, ou les voyageurs dans un lieu de transit. Dans ces différents systèmes, l'objectif est de confronter des informations sur le voyageur avec une information locale qui peut l'aider dans son déplacement. Ainsi, la personne aveugle prenant le bus est alertée dès que son bus arrive à proximité de son arrêt. Dans cet exemple, il s'agit d'une interaction purement locale entre le PDA du voyageur et l'arrêt. Le voyageur dans un aéroport est localisé grâce à des puces qui peuvent donner une information sur son activité et les services sont adaptés en conséquence. Un prototype de service tirant parti de l'interaction entre une gare et l'équipement nomade du voyageur a été testé (Smartphone...). A son arrivée en gare, le voyageur pénétre dans une zone couverte par le wifi. L'équipement entre en relation avec la gare. Ayant préalablement stocké le billet électronique, l'équipement "sait" qu'il entre dans un lieu signifiant, un lieu de service. La gare, de son côté, reconnaît le voyageur au travers de son équipement, reconnaît son profil et son besoin au travers des informations stockées dans cet équipement, et peut lui délivrer, de manière proactive, un service adapté, par exemple, lui indiquer sur quel quai se trouve son train. L'objectif est de modifier l'affichage des informations sur les panneaux donnant les horaires en fonction des voyageurs à proximité.

1.8 Conclusion

Ce chapitre à été consacré à la mise en exergue des concepts liés à la notion de l'informatique ubiquitaire qui est souvent définie comme étant un système basé sur les technologies de l'information et de la communication pouvant nous soutenir dans notre vie quotidienne tout en devenant pratiquement invisible pour nous. Nous ne pouvons pas parler de l'informatique ubiquitaire sans aborder la notion de contexte, ces deux concepts sont liés, pour cela, le chapitre suivant abordera la notion de contexte et les modèles associés.

2

Définition et représentation du contexte

2.1 Introduction

Le contexte est une notion importante pour caractériser les informations sur l'utilisateur et son environnement. Ce chapitre propose une réflexion sur cette notion, il dresse un tour d'horizon sur le contexte. le but est de mettre l'accent sur les éléments de contexte que nous allons prendre en considération dans le cadre de ce travail. Ainsi, après en avoir introduit les définitions accordées à la notion de contexte, nous passons au sujet de la sensibilité au contexte. Ensuite, nous énumérons quelques exemples d'utilisation du contexte. Enfin, nous abordons la problématique de la représentation du contexte, auquel niveau, nous allons mettre en évidence les différents modèles de représentation du contexte.

2.2 Définition formelle du contexte

Le contexte étant une notion complexe et abstraite, il est difficile de trouver une définition détaillée.

Schilit et al.[42] étaient parmi les premiers à essayer de définir le contexte, selon eux, le contexte est constitué de la localisation de l'utilisateur, ainsi que des identités et des états des personnes et des objets qui l'entourent.

Brown et al. [7] ajoutent à cette définition des données telles que l'orientation ou la température.

La définition du concept qui nous parait la plus pertinente est celle donnée par Dey [16] : *Le contexte est n'importe quelle information qui peut être utilisée pour caractériser la situation d'une entité*. Une entité peut être une personne, un lieu ou un objet considéré comme pertinent pour l'interaction entre l'utilisateur et l'application.

Cependant, Abowd et al. [1] et Ryan et al. [39] ont retenu comme éléments les plus importants pour caractériser la situation d'une entité :

- Le temps,
- La localisation,
- L'activité,
- l'identité de la personne.

En outre, une application qui utilise le contexte doit connaître le *quand*, et le *quoi* (ce que la personne est en train de faire) pour déterminer *pourquoi* une situation a eu lieu. Dans le cadre de ce travail, nous utilisons les mêmes éléments pour composer le contexte, sauf l'identité de la personne qui n'est pas importante puisque nous assumons la présence d'une seule personne dans l'environnement.

2.3 Sensibilité au contexte

Les systèmes sensibles au contexte ou *Context Aware Systems* adaptent leur comportement à la fois selon plusieurs facteurs qui conditionnent leur environnement à un instant donné, selon les utilisateurs, leur localisation, les dispositifs accessibles, mais aussi selon les changements dans le temps que subissent ces facteurs [42]. L'ensemble de ces éléments constitue le contexte. Pour être sensible au contexte, un système doit donc être capable d'obtenir ces informations, de les analyser, et de réagir de la façon la plus convenable pour l'utilisateur.

Selon Dey [16], un système est sensible au contexte s'il utilise le contexte pour fournir des informations ou des services pertinents pour l'utilisateur.

La sensibilité au contexte concerne l'acquisition de contexte (par exemple au travers de capteurs), l'abstraction et la compréhension du contexte (en faisant le lien entre des événements détectés et des éléments du contexte), et l'adaptation de la réponse du système basée sur le contexte reconnu.

Un des problèmes les plus importants pour la recherche en systèmes sensibles au contexte est l'aspect spatio-temporel des informations qui composent le contexte. Par exemple, une

consommation éléctrique importante, que l'on considérerait normale à midi avec la présence de l'habitant, deviendrait a priori anormale la nuit alors que l'habitant est dans sont lit. Dans cet exemple, la temporalité du contexte peut être analysée à plusieurs échelles, si l'évènement a lieu de jour ou nuit (quand la personne est censée dormir).

Les systèmes sensibles au contexte peuvent être classifiés en trois groupes selon l'utilisation du contexte selon Schiele et al.[40] :

- **Présentation sensible au contexte :** l'application change d'apparence selon l'information de contexte. Il s'agit par exemple des systèmes de navigation qui changent le niveau de détail en fonction de la vitesse.

- **Exécution automatique de services :** les applications sont déclenchées en fonction du contexte. C'est le cas des services de rappel qui notifient leurs utilisateurs différemment selon la localisation de ceux-ci, par exemple quand ils se trouvent dans un magasin, ou une salle de classe. La plupart des applications pour les maisons intelligentes appartiennent à cette catégorie, de même que les services qui détectent des circonstances où il y a un risque pour la sécurité ou le confort de l'habitant, et se servent du contexte (l'activité ou l'heure) pour mesurer le risque.

- **Marquage d'information de contexte pour une utilisation ultérieure :** ces applications permettent d'enrichir l'interaction des utilisateurs avec leurs environnements au travers d'objets virtuels. Des messages associés à certaines positions et à certains objets environnant l'utilisateur peuvent alors apparaître sur les interfaces dans certaines conditions. Par exemple, un agenda électronique qui rappelle l'utilisateur d'une activité qui aura lieu dans le l'endroit où l'utilisateur est localisé actuellement.

2.4 Catégorisation du contexte

Plusieurs catégories de contexte existent, elles catégorisent les différents types de données contextuelles. Ces catégorisations permettent de lister les paramètres que l'on peut prendre en compte dans un système sensible au contexte.

La première catégorisation est celle de Schilit [41] qui propose de répondre aux questions *où l'on se trouve, avec qui* et *quelles sont les ressources à proximité* afin de caractériser le contexte d'une entité.

Ryan et al[39] proposent les catégories de *position, identité, environnement* et *temps*.

Chen et Kotz [13] citent une taxonomie alternative qui divise le contexte en trois catégories principales : le contexte informatique, le contexte utilisateur et le contexte physique. Le contexte informatique contient toutes les informations relatives aux ressources matérielles qui permettent l'exécution de l'application. Le contexte utilisateur contient des informations relatives à l'identité de l'utilisateur, ses préférences, ses relations sociales, etc. Finalement, les données du contexte physique caractérisent l'environnement physique dans lequel se trouvent les personnes et les applications.

La catégorisation la plus acceptée est celle proposée par Abowd, Dey et al. [1] , qui est constituée de deux niveaux, avec des paramètres primaires et secondaires. Les paramètres primaires sont les mêmes que ceux de Ryan et al., sauf que la notion d'activité (c'est-à-dire, ce qu'on est en train de faire) remplace celle d'environnement. Les paramètres secondaires sont des attributs des entités qui se trouvent dans le contexte primaire. Ces paramètres peuvent être retrouvés en utilisant les données primaires pour les indexer. Par exemple, le numéro de téléphone d'une personne peut être retrouvé en utilisant l'identité de la personne comme index dans une base de données qui contient des numéros de téléphone.

Dans notre travail, nous avons utilisé cette dernière catégorisation, parceque, nous nous intéressons à l'activité de l'utilisateur.

2.5 Exemples d'utilisation du contexte

2.5.1 Active-Badge

Expérimenté en 1990 au laboratoire de recherche Olivetti en Angleterre, Ce projet est généralement considéré comme le premier exemple d'application sensible au contexte. Initié par Want et al[49], le projet vise à une meilleure localisation et coordination d'un large groupe de personnes dans une entreprise. Active Badge est un badge personnalisé porté par chaque employé qui va permettre de le localiser avec précision dans les locaux et ainsi de diriger les appels destinés à ce dernier vers le poste téléphonique le plus proche.

2.5.2 Téléphone cellulaire sensible au contexte : Sensay

Le téléphone portable est un objet qui aurait beaucoup à gagner d'une meilleure utilisation du contexte. Siewiorek et al.[43] proposent avec Sensay d'ajouter du contexte à un téléphone portable. Sensay est un téléphone portable qui modifie son comportement en prenant en compte

l'état de l'utilisateur et de son environnement. Il s'adapte dynamiquement à un contexte en évolution et sait aussi prévenir l'interlocuteur (la personne qui appelle) de l'état de l'utilisateur. Pour déterminer le contexte, Sensay va utiliser des capteurs de lumière, de son, de mouvement qui sont placés à des endroits clé sur l'utilisateur. Tous ces capteurs sont reliés à la "sensor box", sorte d'unité centrale portable qui va interpréter les données des capteurs et calculer l'état dans lequel doit être mis le téléphone.

2.5.3 Cyberguide

Cyberguide est un dispositif informatique qui utilise les variations de localisation de l'utilisateur, en intérieur comme en extérieur pour fournir aux utilisateurs d'un guide touristique numérique des informations relatives à leur position[26]. Le système enregistre aussi les parcours passés des utilisateurs et leur suggére des visites en fonction des sites auxquels ils se sont précédemment attachés. Pour Cyberguide, le contexte se compose des informations de localisations courantes et passées.

2.5.4 Conference Assistant

Conference Assistant est un prototype d'ordinateur porté (ordinateurs qui suivent leurs utilisateurs dans leur mobilité) qui utilise un profil des préférences des utilisateurs (dans ce cas des chercheurs en congrés) pour leur proposer d'assister à des séminaires selon leur localisation et leur intérêt scientifique[17].

2.5.5 Office Assistant

Office Assistant est un agent numérique qui gére l'agenda d'un employé de bureau en fonction de la personne se présentant à la porte, de l'activité du propriétaire du bureau, de l'opportunité du rendez-vous, etc.[59] Ce dernier projet dépasse le niveau de la localisation pour proposer des interprétations de la mobilité, notamment de celle du propriétaire du bureau (déductions contextuelles de la forme : S'il n'est pas au bureau, alors son activité ne peut pas être..., etc.).

2.6 Modélisation du contexte

La modélisation a pour objectif de représenter une idée ou un phénomène du monde réel dans une description simplifiée[5]. L'objectif de la modélisation du contexte est de fournir des représentations abstraites des concepts utilisés pour prendre en compte le contexte.

Un modèle de contexte identifie un sous-ensemble du contexte qui est accessible de façon réaliste à travers des capteurs, des applications et des utilisateurs et qui peut être exploité pour l'exécution de la tâche.

Plusieurs exigences doivent être prises en compte lors de la modélisation des informations contextuelles :

Hétérogéneité et mobilité

La modélisation des informations contextuelles doit faire face à une large variété de sources d'informations contextuelles qui différent dans leur niveau sémantique et leur taux de mise à jour. Les capteurs peuvent observer certains états du monde physique et produire des accès temps réel rapide, fournir des données brutes (par exemple. position GPS ou flux caméra) qui doivent être interprétées avant d'être utilisées par les applications. L'information produite par l'utilisateur -comme son profil ou ses préférences- est mise à jour moins rarement et ne nécessite pas une interprétation additionnelle. Finalement, les données contextuelles obtenues des bases de données ou des librairies digitales -comme les données d'une carte géographique- sont souvent statiques. Beaucoup d'applications sensibles au contexte sont également mobiles (c.à.d s'éxecutant sur un dispositif mobile) ou dépendant d'une source d'informations contextuelles mobile (par exemple, sondes mobiles). Ceci s'ajoute au problème de l'hétérogéneité comme le contexte doit être adaptable à un environnement en permanente évolution. En outre, l'endroit et la disposition spatiale d'information de contexte jouent des rôles importants dûs à cette condition.

Rapports et dépendances

Il existe différentes liaisons entre les différentes catégories des informations contextuelles qui sont capturées afin d'assurer un comportement correct des applications. Un tel rapport est la dépendance par laquelle les entités d'informations contextuelles peuvent dépendre d'autres informations contextuelles. Par exemple, un changement dans la valeur d'une propriété (portée d'un réseau) peut avoir un impact sur les valeurs d'autres propriétés (le restant de l'énergie dans la batterie)

historique du contexte

Les applications sensibles au contexte peuvent avoir besoin d'accéder aux états antérieurs et courants. Par conséquent, l'historique du contexte est une autre caractéristique d'une information de contexte qui à besoin d'être prise en compte par le modèle de contexte. La gestion

des histoiriques des informations de contexte est difficile si le nombre des mises à jours est trop élevé. Il peut être impossible de sauvegarder chaque valeur pour un futur accès.

Rentabilité des formalismes de modélisation.

Les modèles des informations contextuelles sont crées par les concepteurs des applications sensibles au contexte et sont aussi utilisés par ces applications pour manipuler les données contextuelles. Donc les caractéristiques importantes des formalismes de modélisation est la facilité avec laquelle les concepteurs peuvent traduire les concepts réels du monde physique à un modèle et la facilité avec laquelle les applications peuvent utiliser et manipuler la donnée contextuelle.

2.7 Approches de modélisation

Hoareau et al. font un état de l'art dans[22] sur les différents modèles de contexte, ils présentent une vue d'ensemble sur une partie des problèmes liés au contexte, notamment la modélisation, en se basant également sur le travail de Chen et al.[13]. D'après ces deux travaux, les structures de données utilisées pour encoder et transmettre le contexte appartiennent le plus souvent à l'une des cinq catégories suivantes :

2.7.1 Modèle clef-valeur

Ce modèle consiste à associer à un utilisateur un ensemble de mots clés pondérés. Ces mots clés sont extraits du domaine de l'application. Les poids associés à chaque mot clé sont une représentation numérique de l'intérêt de l'utilisateur par rapport au mot concerné. Ces mots clés sont généralement utilisés pour représenter des préférences ou des centres d'intérêt qui constituent une dimension du profil de l'utilisateur [9].

2.7.2 Modèle à balise

Les modèles basés sur les langages à `balises`[1] emploient une variété de langages à balises comprenant XML. L'introduction de la norme W3C pour la description des terminaux et des préférences utilisateur, *Composite Capabilities / Preference Profile(CC/PP)* est parmi les premières approches utilisant *Ressource Description Framwork (RDF)* pour modéliser le contexte. CC/PP utilise des contraintes élémentaires et des relations entre les types de données contextuelles. Cette approche permet de représenter typiquement les profils, composant essentiel pour la construction du contexte.

1. En anglais : Markup model

2.7.3 Modèle graphique

Le modèle graphique le plus répandu et le plus utilisé est le standard de l'Object Management Group (OMG) : UML. Les diagrammes de classes d'UML sont utilisés pour modéliser les concepts qui constituent le contexte et les relations qui les connectent. Les classes UML sont utilisées pour représenter des éléments du contexte avec leurs attributs et les associations représentent les liens entre les concepts. Cette modélisation profite des avantages des langages orientés objets tels que la possibilité de représenter des associations d'héritage, et d'encapsulation.

Le deuxième exemple de langage graphique est le modèle orienté rôle Context Modeling Language (CML) qui est une extension de Object-Role Modeling (ORM) et qui est proposé par [21]. Ce langage représente les avantages suivants :

- capturer les différentes classes et les sources de contexte et les qualifier comme : statiques, capturées (par capteur), dérivées ou fournies par l'utilisateur (ou profilées) ;
- permettre de saisir de l'information imparfaite à l'aide de métadonnées de qualité et mettre en évidence les données contradictoires ;
- capturer les dépendances entre les types de données de contexte ;
- sauvegarder l'historique de certaines données contextuelles et définir des contraintes sur les données de l'historique.

2.7.4 Modèle basé sur la logique

Dans le modèle basé sur la logique, les données de contexte peuvent être exprimées comme des faits dans un système à base de règles. Des méthodes de la logique de premier ordre peuvent être appliquées pour raisonner sur le contexte.

2.7.5 Modèle basé ontologies

Les ontologies constituent un moyen puissant pour représenter les concepts et les relations entre eux. En effet, elles possèdent un pouvoir expressif élevé bénéficiant de techniques de raisonnement bien définies et supportées par des outils automatisés. Plusieurs systèmes sensibles au contexte utilisent les ontologies pour représenter le contexte. Les ontologies permettent de fournir une sémantique formelle des données de contexte en vue de partager le contexte entre différentes sources. Des outils de raisonnement peuvent être utilisés à la fois pour vérifier la cohérence de l'ensemble des relations décrivant un scénario de contexte et pour construire une vue plus abstraite du contexte de haut niveau.
Une des premières approches utilisant les ontologies pour la modélisation du contexte a été

proposée par Ötztürk et al. dans[30]. Puis plusieurs modèles d'ontologies ont été proposés pour représenter des descriptions de données contextuelles tels que SOUPA [14] pour modéliser le contexte dans des environnements ubiquitaires.

2.7.6 Modèle de localisation

Depuis que l'informatique est devenue mobile, la localisation est l'un des aspects du contexte les plus importants et aussi l'un des plus utilisés. Nous pouvons mieux intéragir avec l'environnement en sachant où l'on est, qui est autour de nous et quels objets sont à proximité. On peut faire deux types d'utilisations de la localisation : d'un côté, la localisation du terminal et d'un autre côté, le suivi des entités environnantes. Becker et al.[4] font un état de l'art sur les principaux modèles de localisation utilisés pour des applications mobiles.

Modèle géométrique.

La position des entités est représentée par des coordonnées géométriques, permettant de calculer des distances entre les entités. Le plus souvent, la position est exprimée par rapport au World Geodetic System (WGS84) (le système géodésique mondial), en utilisant les satellites GNSS (Global Navigation Satellite System).

Modèle symbolique (sémantique)

Les positions sont définies en donnant des noms symboliques, plus proches de l'intuition humaine (par exemple, nom et numéro de rue, étage et bureau dans un bâtiment, etc.)

2.8 Conclusion

Depuis les premières propositions de définitions jusqu'aux différents domaines d'application du conexte, ce chapitre à introduit et illutré les fondamentaux de la notion de contexte. Face aux difficultés rencontrées pour établir une définition consensuelle du contexte, plusieurs travaux se concentrent sur la représentation du contexte. Le chapitre suivant concernera les techniques de classification des informations contextuelles qui est une étape cruciale dans la chaine de reconnaissance du contexte.

3

Reconnaissance d'activités et techniques de classification

3.1 Introduction

La Reconnaissance d'Activité Humaine (RAH)) à partir de capteurs est un domaine scientifique en pleine effervescence et qui a émergé dans la recherche sur les intéractions hommes-machines et sur l'informatique ubiquitaire. Dans ce chapitre, nous introduirons la chaîne de reconnaissance d'activités qui décrit les phases de la reconnaissance d'activité humaine, nous allons nous intéresser à la phase *classification* qui est une étape cruciale dans la chaîne de reconnaissance d'activités. Ensuite, nous procéderons à la présentation des méthodes de classification les plus utilisées dans ce domaine. Enfin, nous allons présenter quelques travaux dans la littérature qui utilisent ces méthodes de classification pour inférer les activités.

3.2 Reconnaissance d'activité

3.2.1 Concept de reconnaissance d'activités

Les activités quotidiennes sont des données contextuelles concernant les utilisateurs et leur usage des équipements essentiels si l'on souhaite les assister et leur permettre d'économiser de l'énergie.

La reconnaissance d'activités dans un espace donné, nécessite l'interaction des systèmes informatiques vis-à-vis de leurs environnements. En fait, cette notion est basée sur le concept

de vision par ordinateur (une branche de l'intelligence artificielle) permettant à une machine d'intégrer des informations lorsqu'on la connecte à un ou plusieurs capteurs (caméra, capteurs environnementaux, etc.). D'ailleurs, l'installation des capteurs, dans un environnement donné, permet à ce dernier d'être surveillé en fournissant des informations précises. Le traitement de ces informations donne un rapport de toutes les activités des personnes ou des objets mobiles dans cet environnement.

Il est important de mentionner que la reconnaissance d'activité se fait généralement en suivant un certain nombre d'étapes, ces étapes sont réunies dans le cadre de ce qu'on appelle une chaîne de reconnaissance d'activités introduite dans la section suivante.

3.2.2 Chaîne de reconnaissance d'activité

La chaîne de reconnaissance d'activités(CRA) est l'ensemble des principaux traitements que suivent généralement la plupart des chercheurs pour inférer des activités à partir des données brutes des capteurs [50, 2, 19], comme montrés dans la figure 3.1.

Le traitement sub-symbolique (sub-symbolic processing) transforme les données de bas-niveau issues de capteurs (par exemple. l'accélération d'un membre du corps) en des actions primitives significatives (par exemple : tenir). Un sens est attribué aux données de capteurs en les comparant à des prototypes d'activités connus.
Les sorties du traitement sub-symbolique sont des événements indiquant l'occurence d'une action. La chaîne de reconnaissance d'activités se termine à ce stade lorsque les activités qui nous interesse sont des gestes simples.

Le traitement symbolique (symbolic processing) transforme les séquences d'actions primitives (par exemple : tenir, couper) en des activités de haut niveau (par exemple : cuisiner,...).

Les étapes du traitement sub-symbolique sont [50, 2, 19](figure 3.1) :

- *Acquisition des données de capteurs (sensor acquisition)* : Un flux de données de capteurs S est obtenu ;

- *prétraitement du signal (preprocessing) :* Le flux de données recueilli des capteurs subit un prétraitement. ce prétraitement a pour but la réduction des `bruits`[1] de capteurs ou

1. Distorsion du signal de capteur

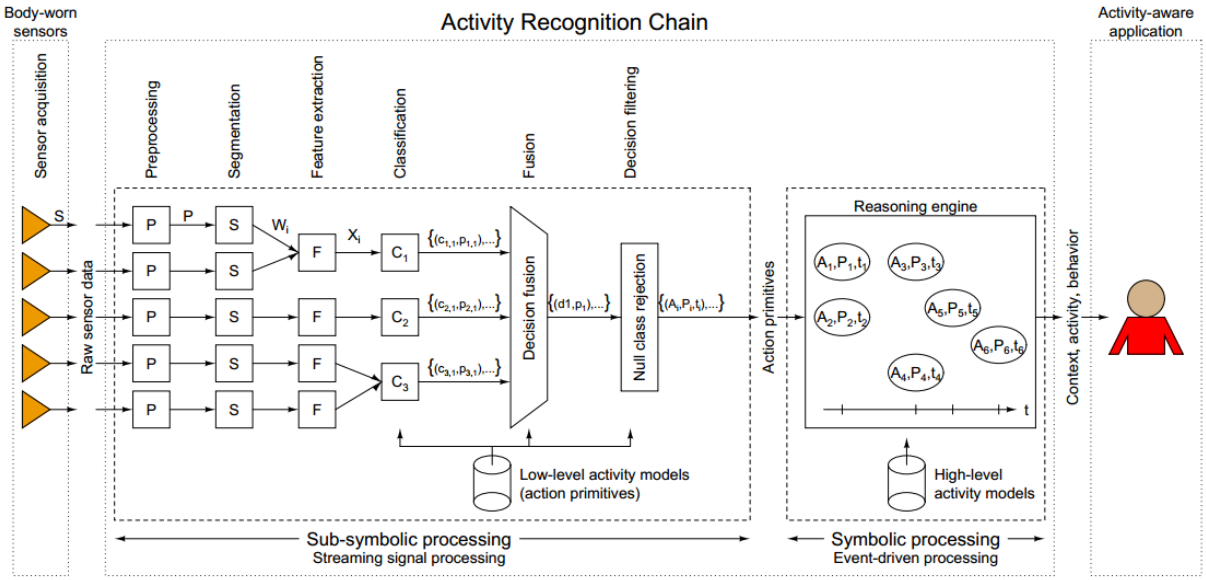

FIGURE 3.1 – Chaîne de reconnaissance d'activités (CRA) [38]

inhérent au signal ;

– *Segmentation du flux de données (segmentation) :* Le flux de données est ségmenté en des sections **W** qui contiennent probablement un "geste", les ségments sont identifiés par leur instant de début et leur instant de fin dans le flux de données. L'une des techniques de ségmentation les plus utilisées est la *fenêtre coulissante* ;

– *Extraction de caractéristiques (feature extraction) :* Les caractéristiques sont estimées sur les ségments identifiés pour réduire leur dimension, obtenant ainsi un vecteur de caractéristiques X ;

– *Classification :* Un classifieur transforme le vecteur de caractéristiques en un ensemble prédéfini de classes (activités, gestes)

– *Décision de fusion (decision fusion) :* Combiner les informations de sources multiples (capteurs multiples ou classifieurs multiples opérant sur un seul capteur) en une décision sur l'activité qui a eu lieu ;

– *Rejet de classes nulles (null class rejection) :* Dans le cas où la confiance en résultats de classification est trop basse (le système peut se débarrasser de l'activité classifiée basée sur sa probabilité).

À cette étape (rejet de classes nulles), le résultat est la détection d'une action primitive A_i avec une probabilité p_i et à l'instant t_i.

Les classifieurs utilisés dans la chaîne de reconnaissance d'activités (CRA) sont obtenus en utilisant un ensemble d'apprentissage contenant les instances de données (vecteurs de caractéristiques) $\mathbf{X}$ et leurs labels d'activités correspondants y.

Dans ce qui suit, nous allons mettre le point sur la phase classification dans cette chaîne de reconnaissance qui est l'objet de cette étude.

3.3 Définition de la classification

La classification est la tâche qui permet d'apprendre une fonction objectif f pour assigner un ensemble d'attributs x à l'une des classes prédéfinies étiquettée y. La fonction f peut aussi être appelée un modèle de classification. Ce modèle de classification peut être utilisé dans différentes situations [11].

L'objectif principal de la classification est d'identifier les classes auxquelles appartiennent des objets à partir des traits descriptifs, appelés aussi attributs, caractéristiques, ou en anglais, "features". Les attributs dans notre cas représentent les événements et les classes représentent les activités.

3.4 Les méthodes de classification

Les méthodes de classification possèdent un point commun : la nécessité de réaliser un apprentissage et d'évaluer leurs performances. Elles nécessitent donc deux ensembles d'échantillons :

1. **Base d'apprentissage** : doit contenir un nombre suffisant d'échantillons x_i de chaque classe. Cet ensemble doit être le plus représentatif possible d'un point de vue qualitatif (en vue de l'application visée) et quantitatif (un nombre important d'échantillons tout en respectant les probabilités à priori de chacune des classes).

2. **Base de test :** doit aussi être choisie de manière rigoureuse puisqu'elle permet d'évaluer l'algorithme.

Classiquement, le processus se déroule en deux phases principales : apprentissage (hors-ligne) et décision (en-ligne) comme illustré dans la figure 3.2. Le module d'apprentissage permet, lorsqu'il est supervisé, c'est à dire à partir de la connaissance à priori de la classe des échantillons fournie par un expert, de construire un modèle (ensemble de règles ou critères de décision). Ce modèle est ensuite utilisé par le classifieur pour fournir la décision concernant l'appartenance d'un échantillon inconnu, représenté par ses attributs, à l'une des classes prédéfinies.

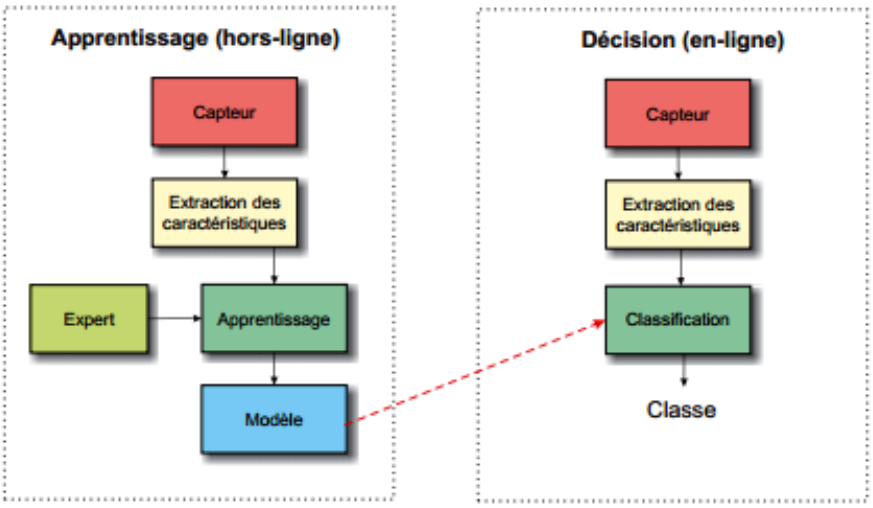

FIGURE 3.2 – Processus classique de reconnaissance par apprentissage [11]

Suivant que la classe des échantillons d'apprentissage est connue ou non, on obtient deux grandes familles de méthodes de classification et par conséquent deux principales approches d'apprentissage :

– **Approches supervisées :** La procédure générale pour apprendre et tester un algorithme d'apprentissage supervisé pour la reconnaissance d'activités se déroule en général suivant les cinq étapes suivantes :

1. Acquérir les données de capteurs des activités en incluant l'annotation de ce que fait l'utilisateur au moment de l'acquisition de ces données ;
2. Transformer ces données en des caractéristiques, par exemple, en calculant des propriétés spécifiques, réduire la dimensionnalité,... ;
3. Diviser cet ensemble de caractéristiques en un ensemble d'apprentissage et un ensemble de test ;

4. Apprendre l'algorithme en utilisant l'ensemble d'apprentissage ;
5. Tester la performance de la classification de l'algorithme en utilisant l'ensemble de données de test.

– **Approches non-supervisées :** La procédure d'apprentissage non-supervisé consiste à :

1. Acquérir les données non labelisées ;
2. Transformer ces données en des caractéristiques ;
3. Modéliser les données en utilsant un algorithme de partitionnement (clustering).

Durant le partitionnement, par exemple, chaque point de données est assigné à un ou N groupes de points qui lui sont proches avec respect d'une mesure de distance prédéfinie.

Dans la suite du chapitre, nous allons détailler quelques une des méthodes de classification utilisées dans le domaine de la reconnaissance des activités humaines.

3.4.1 Les arbres de décision

Les arbres de décision [33][34] représentent l'une des techniques les plus connues et les plus utilisées en classification. Leur succès est notamment dû à leur aptitude à traiter des problèmes complexes de classification. En effet, ils offrent une représentation facile à comprendre et à interpréter ainsi qu'une capacité à produire des règles logiques de classification.

Un arbre de décision est composé de :

– noeuds de décision contenant chacun un test sur un attribut ;

– branches correspondant généralement à l'une des valeurs possibles de l'attribut sélectionné ;

– feuilles comprenant les objets qui appartiennent à la même classe.

Exemple 1 :

La figure 3.3 représente un arbre de décision pour le concept "acheter ordinateur", indiquant quand est ce qu'un client dans un magasin de produits éléctroniques est susceptible d'acheter

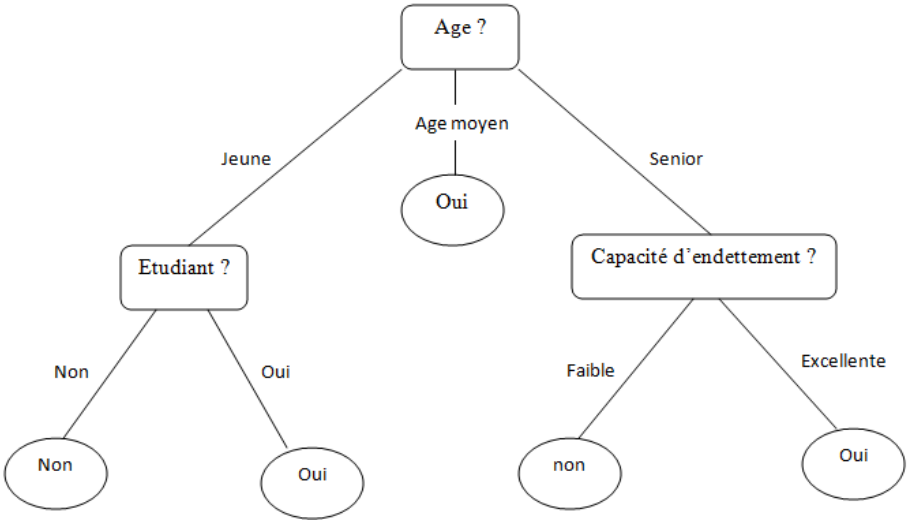

FIGURE 3.3 – Exemple d'arbre de décision

un ordinateur.
Chaque noeud de décision (en forme de réctangle) représente un test sur un attribut et chaque noeud feuille (en forme ronde) représente une classe
(si acheter-ordinateur = "oui" ou acheter-ordinateur = "non").

L'utilisation des arbres de décision dans les problèmes de classification se fait en deux étapes principales :

- la construction d'un arbre de décision à partir d'une base d'apprentissage ;
- la classification ou l'inférence consistant à classer une nouvelle instance à partir de l'arbre de décision construit dans la première étape.

La construction d'un arbre de d'ecision se base sur un ensemble d'apprentissage donné. Elle consiste à sélectionner pour un noeud de décision le test d'attribut approprié et puis de définir la classe relative à chaque feuille de l'arbre induit.

Plusieurs algorithmes ont été développés afin d'assurer la phase de construction. Parmi les algorithmes `non-incrementaux` [2], nous citons `ID3` [3] et C4.5 développés par Quinlan [33] [34] et l'algorithme `CART` [4] de Breiman et al.[29] qui sont probablement les plus populaires.

2. L'apprentissage non-incrémental correspond à un système qui n'est pas capable de recevoir et d'intégrer de nouveaux exemples sans devoir réaliser un apprentissage complet.
3. Iterative Dichotomizer 3
4. Classification And Regression Tree

Les algorithme développés suivent généralement le même principe suivant :

Soit T un ensemble d'apprentissage contenant des objets qui peuvent appartenir à l'une des classes $C_1, C_2, \ldots, C_n$. Si tous les objets appartiennent à la même classe, l'arbre de décision relatif à l'ensemble d'apprentissage sera une feuille libellée par cette classe, autrement T contient des objets appartenant à différentes classes. Soit A_k un attribut avec comme valeurs possibles $(v_1, v_2, \ldots, v_f)$. L'ensemble d'apprentissage sera partitionné en sous-ensembles $T_1, T_2, \ldots, T_f$ où chaque ensemble T_t $(t = 1, \ldots, f)$ correspond à une valeur v_t du test A_k. Puis, la même procédure sera appliquée récursivement à chaque sous-ensemble.

Exemple 2 :

Afin d'illustrer les différentes notions concernant les arbres de décision, nous allons considérer un exemple de base d'apprentissage donné dans le tableau 3.1. Composé de quelques lignes sélectionnées aléatoirement de la base de données *client* de *ALLElectronics*, (Les données sont adaptées de [33]). Dans cet exemple chaque attribut a une valeur discrète. l'attribut classe : acheter-ordi a deux valeurs distinctes (oui, non) ; donc, il y a deux classes distinctes.

age	**salaire**	**etudiant**	**capacité-endettement**	**class :acheter-ordi**
jeune	élevé	non	faible	non
jeune	élevé	non	excellent	non
age-moyen	élevé	non	faible	oui
senior	moyen	non	faible	oui
senior	bas	oui	faible	oui
senior	bas	oui	excellent	non
age-moyen	bas	oui	excellent	oui
jeune	moyen	non	faible	non
jeune	bas	oui	faible	oui
senior	moyen	oui	faible	oui
jeune	moyen	oui	excellent	oui
age-moyen	moyen	non	excellent	oui
age-moyen	élevé	oui	faible	oui
senior	moyen	non	excellent	non

TABLE 3.1 – Ensemble d'apprentissage

Les différents algorithmes de construction d'arbres de décision se basent généralement sur trois paramètres principaux : (1) Mesure de sélection d'attributs, (2) stratégie de partitionnement, et (3)critère d'arrêt.

1. Mesure de sélection d'attributs

La mesure de sélection d'ttributs permet de choisir l'attribut qui sera la racine de l'arbre. En outre, elle permet le choix des racines des sous-arbres de décision. En fait, la mesure de sélection d'attributs permet de classer les attributs entre eux et de choisir celui qui partitionne l'ensemble d'apprentissage de manière optimale réduisant par conséquent la taille de l'arbre. Ainsi, une bonne mesure doit permettre de limiter la taille de l'arbre et de donner une cohérence sémantique aux noeuds qui le composent.

La mesure de sélection d'attributs est généralement basée sur la théorie de l'information. Nous citons comme mesures celles proposées par Quinlan :*le gain d'information*[33] et *le ratio de gain*[34].

L'une des mesures de sélection d'attributs les plus utilisées est le critère de gain d'information de Quinlan [33] qui est appliqué dans ID3 et C4.5 et qui est défini comme suit :

$$Gain(T, A_k) = Info(T) - Info_{A_k}(T) \tag{3.1}$$

où

$$Info(T) = -\sum_{i=1}^{n} \frac{Freq(C_i, T)}{|T|} \log_2 \frac{Freq(C_i, T)}{|T|} \tag{3.2}$$

et

$$Info_{A_k}(T) = \sum_{v \in D(A_k)} \frac{|T_v^{A_k}|}{|T|} Info(T_v^{A_k}) \tag{3.3}$$

où $Freq(C_i, T)$ représente le nombre d'objets dans l'ensemble appartenant à la classe C_i et $T_v^{A_k}$ est le sous-ensemble d'objets pour qui l'attribut A_k prend la valeur v.

Ainsi, l'attribut ayant le gain d'information le plus élevé est sélectionné comme étant la racine de l'arbre (ou du sous-arbre) de décision.

Bien que le critère de gain d'information donne de bons résultats, il présente un sérieux inconvénient. En effet, il favorise les attributs ayant un nombre élevé de valeurs [34]. Pour remédier à cet inconvénient, Quinlan propose une sorte de normalisation connue sous le nom de *ratio de gain* :

$$Gain\_ratio(T, A_k) = \frac{Gain(T, A_k)}{Split\_info(A_k)} \tag{3.4}$$

où $Split\_info(A_k)$ est définie comme étant l'information contenu dans l'attribut A_k[34].

2. Stratégie de partitionnement

La stratégie de partitionnement permet de partitionner l'ensemble d'apprentissage courant en tenant compte de l'attribut test. Dans le cas d'attributs discrets, la stratégie consiste à tester toutes les valeurs possibles de l'attribut, alors que dans le cas d'attributs numériques, une étape de discrétisation est généralement nécessaire[18][51].

3. Critère d'arrêt

Le critère d'arrêt permet de traiter les conditions d'arrêt du développement d'une partie d'un arbre ou d'un sous-arbre. Ce critère se déclenche généralement si tous les objets de l'ensemble d'apprentissage relatifs à un sous arbre de décision appartiennent à une seule classe. Ainsi, une partie de l'arbre vérifiant le critère d'arrêt peut donner naissance à une feuille.

Exemple 3 :

Continuons l'exemple précédent et calculons le gain d'information des quatre attributs afin de choisir la racine de l'arbre de décision. Calculons, tout d'abord, la valeur de $Info(T)$ relative à l'ensemble d'apprentissage :

$$\begin{aligned} Info(T) &= -\frac{Freq(oui, T)}{|T|} \log_2 \frac{Freq(oui, T)}{|T|} - \frac{Freq(non, T)}{|T|} \log_2 \frac{Freq(non, T)}{|T|} \\ &= -\frac{9}{14} \log_2 \frac{9}{14} - \frac{5}{14} \log_2 \frac{5}{14} \\ &= 0.940 \end{aligned} \tag{3.5}$$

Le ratio de gain relatif à l'attribut age est calculé comme suit :

$$Info_{age}(T) = \frac{5}{14}Info(T^{age}_{jeune}) + \frac{4}{14}Info(T^{age}_{age-moyen}) + \frac{5}{14}Info(T^{age}_{senior})$$

$$= \frac{5}{14}(-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}) + \frac{4}{14}(-\frac{4}{4}\log_2\frac{4}{4} - \frac{0}{4}\log_2\frac{0}{4}) + \frac{5}{14}(-\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5})$$

$$= 0.694$$

Donc $Gain(T, age) = Info(T) - Info_{age}(T) = 0.940 - 0.694 = 0.246bits$
De façon similaire, nous trouvons :
$Gain(T, salaire) = 0.029bits$, $Gain(T, etudiant) = 0.151bits$ et $Gain(T, capacit - endettement) = 0.048bits$

Nous remarquons que l'attribut *age* présente le gain d'information le plus élevé, donc, il sera choisi comme la racine de l'arbre de décision, et l'arbre obtenu par conséquent est représenté par la figure 3.4

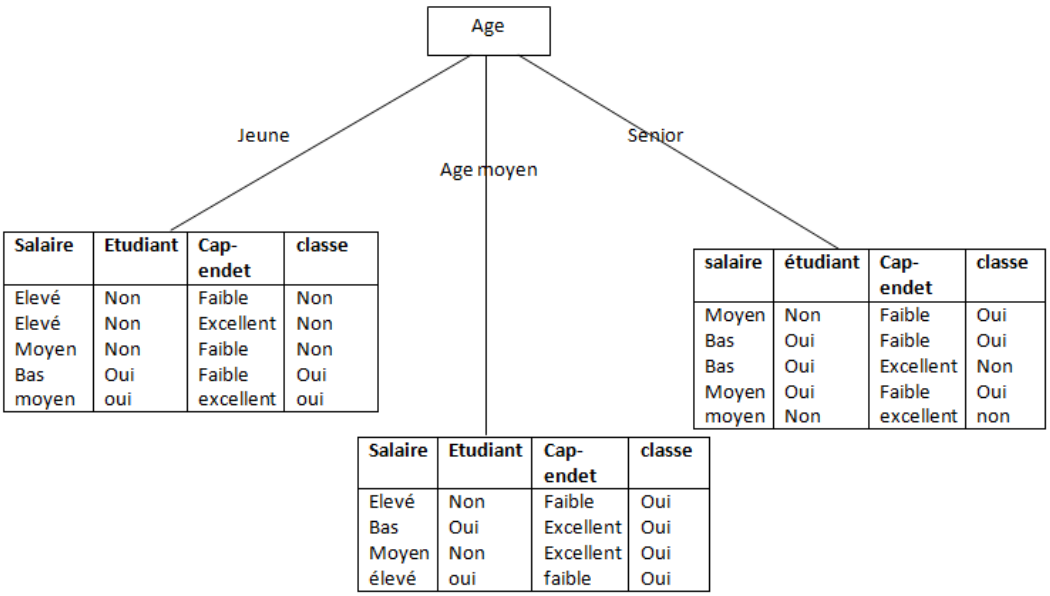

FIGURE 3.4 – arbre obtenu en sélectionnant l'attribut "age" comme racine

Nous remarquons que le sous-ensemble issue de la branche portant la valeur "age-moyen" respecte l'un des critères d'arrêt, puisque pour la valeur "age-moyen", tous les exemples appar-

tiennent à la même classe (acheter-ordi = oui), par conséquent, le noeud issue de cette branche sera déclaré comme étant une feuille.

En suivant le même raisonnement, l'arbre final est représenté par la figure 3.3 (vue précédement).

Une fois l'arbre construit, il peut être utilisé pour l'étape de classification. Pour cela, il suffit de suivre le chemin en partant de la racine jusqu'aux feuilles en effectuant les différents tests à chaque noeud selon les valeurs des attributs de l'objet à classer.

3.4.2 Le classifieur naïf de Bays

De manière générale, l'application de l'analyse statistique de Bayes [3] aux problèmes de classification est basée sur une modélisation probabiliste des classes [12]. En phase d'exploitation, les modèles de classes, étant supposés connus, un classifieur de Bayes détermine la classe d'un exemple $X = [x_1, \ldots, x_n]$ selon l'hypothèse du maximum a posteriori (MAP). Dans ce contexte, la classe d'affectation de X, C_{MAP} , est obtenue par maximisation de la probabilité conditionelle de classe, c'est-à-dire :

$$C_{MAP} = argmax_{C_i \in C} P(X/C_i), \tag{3.6}$$

ou encore par application de la règle de Bayes :

$$C_{MAP} = argmax_{C_i \in C} \frac{P(X/C_i).P(C_i)}{P(X)} = argmax_{C_i \in C} P(X/C_i).P(C_i), \tag{3.7}$$

où

- $P(C_i)$ est la probabilité a priori de la classe C_i .
- *argmax* : est l'argument du maximum, l'ensemble des classes en lesquelles P atteint son maximum.

L'objectif de la phase d'apprentissage d'un classifieur de Bayes est d'estimer les probabilités nécessaires à la résolution de 3.7, à savoir $P(X/C_i)$ et $P(C_i)$, à partir de l'ensemble d'exemples disponibles.

Afin de pouvoir appliquer le principe de Bayes sur un problème réel de classification, quelques hypothèses de simplification sont couramment utilisées. Une de ces hypothèses est l'indépendance conditionelle des différents attributs observés. Si l'on prend en considération deux at-

tributs x_1 et x_2 avec les probabilités conditionelles $P(x_1/C)$ et respectivement $P(x_2/C)$, où $P(A/B)$ est la probabilité de l'événement A, sachant que l'événement B s'est produit et que C est une classe, alors x_1 et x_2 sont probabilistiquement indépendants si la probabilité de leur apparition simultanée est égale au produit des deux probabilités conditionelles :

$$P((x_1, x_2)/C) = P(x_1/C).P(x_2/C) \tag{3.8}$$

Soit $X = [x_1, x_2, \ldots, x_n]$ l'exemple observé, où n est le nombre d'attributs des objets à classifier. Si l'hypothèse d'indépendance conditionelle est satisfaite, la classe C_{MAP} qui minimise l'erreur de classification pour l'exemple X est donnée par l'équation 3.9

$$C_{MAP} = argmax_{C_i \in C} \prod_{k=1}^{n} P(x_k/C_i).P(C_i) \tag{3.9}$$

Pour tout nouvel exemple X , le système de classification, dit classifieur naïf de Bayes, doit alors trouver la classe C_{MAP} qui est la solution de l'équation 3.9 [37].

La probabilité a priori d'une classe, $P(C_i)$; $i \in \{1, \ldots, |C|\}$ est la proportion des points d'apprentissage qui appartiennent à la classe C_i. La probabilité conditionelle $P(x_k/C_i)$ est donnée par la proportion de l'événement x_k parmi tous les exemples qui composent la classe C_i.

Exemple 3

Dans cet exemple, nous illustrons comment le modèle bayésien naïf peut être utilisé dans un problème de classification. Reprenons la base de données présentée dans la table 3.1.

Soit un nouvel enregistrement présenté dans la table 3.2 pour lequel nous cherchons la classe à laquelle appartient cet enregistrement :

age	**salaire**	**etudiant**	**capacité-endettement**	**class :acheter-ordi**
jeune	moyen	oui	faible	?

TABLE 3.2 – Exemple d'enregistrement

Soit :

– A : l'ensemble des attributs

- O : La classe acheter-ordi = oui

- N : La classe acheter-ordi = non

- $P(A/O) = \frac{2}{9}.\frac{4}{9}.\frac{6}{9}.\frac{6}{9} = 0.044$

- $P(A/N) = \frac{3}{5}.\frac{2}{5}.\frac{1}{5}.\frac{2}{5} = 0.019$

- $P(A/O)P(O) = 0.044.\frac{9}{14} = 0.028$

- $P(A/N)P(N) = 0.019.\frac{5}{14} = 0.007$

Puisque $P(A/O)P(O) > P(A/N)P(N)$ donc le classifieur bayésien naïf prédit que la classe du nouvel enregistrement est acheter-ordi = oui

3.4.3 les machines à vecteurs supports

La machine à vecteurs supports[6] (en anglais SVM : Support vector Machine) est une nouvelle méthode prometteuse de classification. La machine à vecteurs supports est un algorithme qui utilise une fonction non-linéaire pour transformer les données originales en une dimension elevée. Dans cette nouvelle dimension, il cherche l'hyperplan séparateur optimal (qui est la *frontière de décision*) qui sépare les exemples entre deux classes.

Pour expliquer le mystère des SVMs, prenons le cas simple d'un problème de deux classes, où les classes sont linéairement séparables. Soit l'ensemble de données D donné comme $(X_1, y_1), (X_2, y_2), \ldots, (X_{|D|}, y_{|D|})$ où X_i est l'ensemble des exemples avec leurs labels de classes y_i. Chaque y_i peut prendre une des deux valeurs $+1$ ou -1 (c'est à dire $y_i \in \{-1, +1\}$). Considérons un exemple avec deux attributs, A_1 et A_2, comme illustré sur la figure 3.5, on constate que les données $2D$ sont **linéairement séparables** parce qu'une ligne droite peut être tracée pour séparer les exemples de la classe $+1$ de la classe -1. Il existe un nombre infini de lignes séparatrices qui peuvent être tracées. L'objectif est de trouver la ligne séparatrice optimale. Si nos données étaient en $3D$ (c'est à dire avec trois attributs), dans ce cas, c'est pas une ligne séparatrice qu'on cherchera mais un *plan* séparateur. Si on généralise pour n dimensions, nous aurons à chercher le meilleur *hyperplan* qui sépare nos données. Nous utiliserons le terme

"hyperplan" pour désigner la frontière de décision indépendemment du nombre d'attributs.

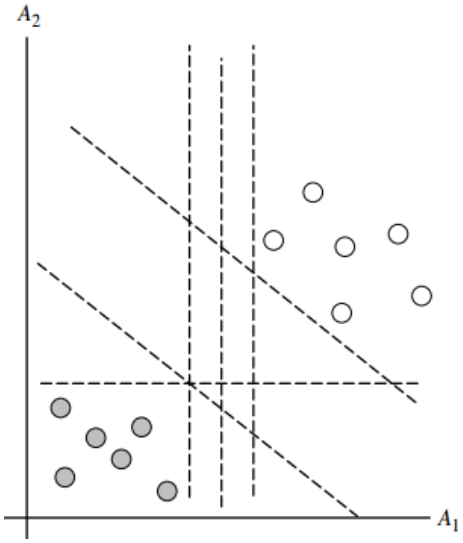

FIGURE 3.5 – Exemple de SVM

Un SVM aborde le problème de trouver l'hyperplan optimal par chercher **l'hyperplan de marge maximale**. Considérons la figure 3.6 qui montre deux hyperplans séparateurs possibles et leurs marges associées, avant de définir la marge, observons cette figure, les deux hyperplans peuvent classifier tous les exemples de données. Toutefois, intuitivement, nous nous attendons à ce que l'hyperplan avec une grande marge sera plus exacte pour la classification d'exemples de données futurs que l'hyperplan avec une petite marge. Pour cela, durant la phase d'apprentissage, le SVM cherche l'hyperplan qui possède la plus grande marge. La marge est la plus courte distance de l'hyperplan aux exemples les plus proches de chacune des deux classes.

Un hyperplan séparateur peut être donné par :

$$\mathbf{W}.\mathbf{X} + b = 0 \tag{3.10}$$

où :

- W est le vecteur de poids, W=$\{w_1, w_2, \ldots, w_n\}$, n est le nombre d'attributs ;
- b est un scalaire.

Considérons deux attributs, A_1 et A_2, comme le montre la figure 3.6, les exemples sont $2D$, $X = (x_1, x_2)$ où x_1 et x_2 sont les valeurs des attributs A_1 et A_2 respectivement pour X. Si nous pensons que b est un poids additionnel, w_0, nous pouvons réecrire l'hyperplan séparateur

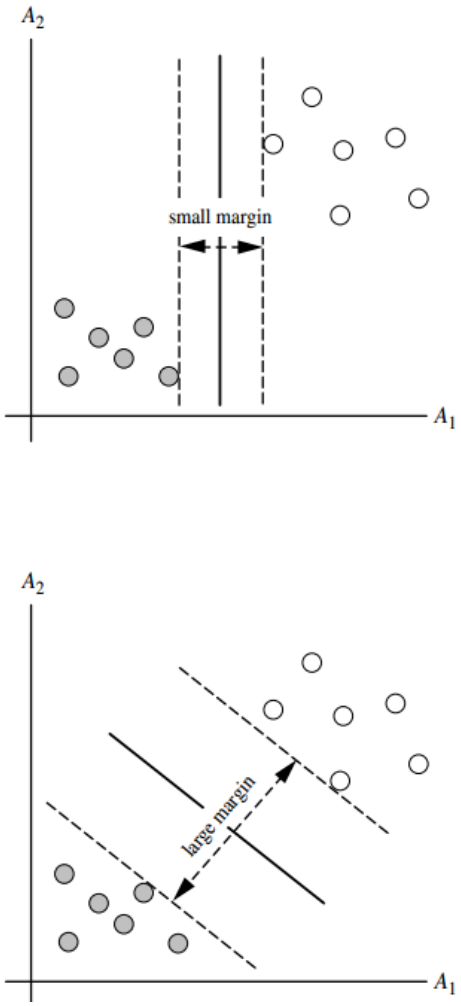

FIGURE 3.6 – Exemple de deux hyperplans avec deux marges différentes

ci-haut comme suit :

$$w_0 + w_1.x_1 + w_2.x_2 = 0 \tag{3.11}$$

Ainsi, chaque point en-dessus de l'hyperplan séparateur satisfait :

$$w_0 + w_1.x_1 + w_2.x_2 > 0 \tag{3.12}$$

De manière similaire, chaque point en-dessous de l'hyperplan séparateur satisfait :

$$w_0 + w_1.x_1 + w_2.x_2 < 0 \tag{3.13}$$

Les poids peuvent être ajustés de sorte que les hyperplans qui définissent les côtés de la marge peuvent être écris comme :

$$H_1 : w_0 + w_1.x_1 + w_2.x_2 \geq 1 \quad pour \quad y_i = +1 \tag{3.14}$$

et

$$H_2 : w_0 + w_1.x_1 + w_2.x_2 \leq -1 \quad pour \quad y_i = -1 \tag{3.15}$$

Donc, chaque exemple qui tombe en-dessus de H_1 appartiendra à la classe $+1$, et chaque exemple qui tombe en-dessous de H_2 appartiendra à la classe -1. En combinant les deux inégalité (3.14) et (3.15), nous obtenons :

$$y_i(w_0 + w_1.x_1 + w_2.x_2) \geq 1, \forall i \tag{3.16}$$

N'importe quel exemple qui tombe sur les hyperplans H_1 ou H_2 (c.à.d. les côtés qui définissent la marge) qui satisfait l'équation 3.16 sont appelés *vecteurs supports*. Dans la figure 3.7, les vecteurs supports sont montrés avec des bordures épaisses.

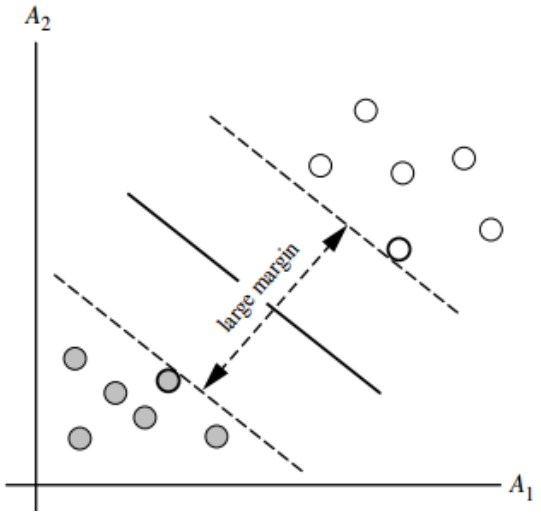

FIGURE 3.7 – Illustration des vecteurs supports (bordures épaisses)

3.4.4 Le classifieur K plus proches voisins - kPPV

L'algorithme K plus proches voisins est un algorithme qui se base sur le concept de proximité. L'apprentissage (construction du modèle de classification) par cet algorithme est considéré comme un apprentissage paresseux car il consiste seulement à stocker l'ensemble d'entrainement, cela veut dire, qu'il ne comporte pas d'étape d'apprentissage qui permette d'apprendre un modèle à partir d'un ensemble d'échantillons.

L'algorithme K plus proches voisins définit une fonction de distance entre les vecteurs de caractéristiques pour faire la classification d'une nouvelle entrée.

Le choix de cette fonction de distance est primordial au bon fonctionnement de la méthode. Les distances les plus simples permettent d'obtenir des résultats satisfaisant. L'une des distances utilisées dans l'algorithme K plus proches voisins est la distance Euclidienne, elle est définie comme suit : Soit $X = (x_1, \ldots, x_n)$ et $Y = (y_1, \ldots, y_n)$ deux exemples, la distance euclidienne entre X et Y est :

$$D(X, Y) = \sqrt{\sum_{i=1}^{n} (X_i - Y_i)^2} \tag{3.17}$$

L'algorithme procède en identifiant un ensemble de K plus proches voisins pour chaque classe. Cet ensemble est obtenu en faisant une comparaison entre la nouvelle entrée et chaque exemple de l'ensemble d'entrainement à l'aide d'une mesure de similarité. Une fois que l'ensemble des K plus proches voisins est trouvé, l'algorithme cherche la classe qui a le plus de représentants dans cet ensemble afin d'associer une étiquette à la nouvelle entrée. La figure suivante (figure 3.8) illustre le principe de fonctionnement de ce classifieur. Sur la figure de droite qui est une explication de la figure de gauche, nous remarquons que le nombre de voisins de l'exemple que nous souhaitons classifier (en forme ronde) est $K = 3$ (3-NN : 3 Nearest Neighbors), l'algorithme associe à cet exemple la classe des exemples triangulaires parce que ces exemples ont le plus de représentants que les exemples de forme réctangulaire dans l'ensembles des voisins.

3.4.5 Les approches de classification neuronales

De nombreux réseaux de neurones sont connus et exploités actuellement, mais les principes basiques peuvent être illustrés sur deux concepts "classiques", les perceptrons et les réseaux multi-couches.

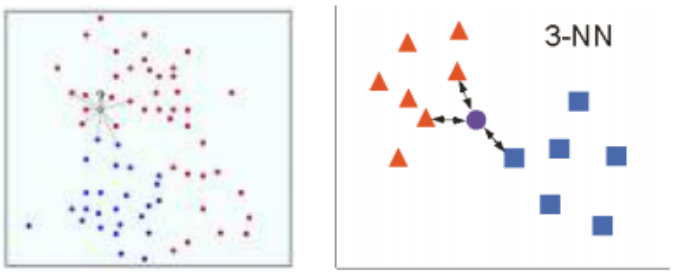

FIGURE 3.8 – Principe de fonctionnement de KPPV [47]

Les perceptrons linéaires

Le problème de classification le plus simple consiste à choisir parmi deux classes (C_0 *et* C_1). Le perceptron est un système qui choisit une de ces deux classes comme classe d'appartenance d'un objet-entrée. Dans ce cas, le perceptron implémente en fait un discriminant linéaire et construit la ligne de séparation donnée par la formule 3.18 suivante :.

$$w_1.x_1 + w_2.x_2 + \cdots + w_n.x_n + w_0 \tag{3.18}$$

où les x_i sont les valeurs du vecteur d'observations à classifier et les w_i les poids qui doivent être déterminés à partir des données d'apprentissage.

Le principe de fonctionnement du perceptron linéaire est illustré dans la figure 3.9.
Les x_i ; $i = \{1, \ldots, n\}$ sont les n composantes de l'exemple analysé et les pondérations w_i ; $i = \{1, \ldots, n\}$ sont les coefficients qui définissent la ligne de séparation et qui doivent être calculés. Le coefficient w_0 s'appelle "seuil" ou "biais" du perceptron[56].

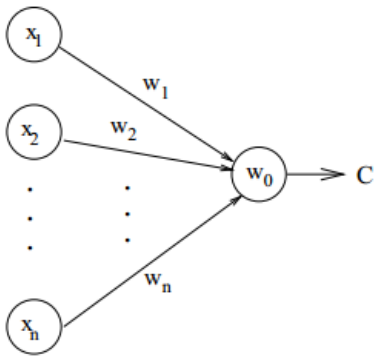

FIGURE 3.9 – Principe du percepteron linéaire

La sortie du perceptron sera alors donnée par l'équation :

$$C = \begin{cases} C_1 & \text{si } \sum_{i=1}^{n} w_i.x_i + w_0 > 0 \\ C_0 & \text{autrement.} \end{cases} \tag{3.19}$$

Afin d'allouer l'exemple à la classe C_1, il faut que la somme pondérée des attributs soit supérieure à la valeur $-w_0$.

La tâche la plus importante et qui définit le perceptron est le calcul des pondérations w_i, $i = \overline{1,n}$

La mise en oeuvre du perceptron contient une étape d'apprentissage, lorsque des exemples connus sont utilisés pour déterminer les valeurs des poids w_i. Le principe est davantage basé sur un apprentissage séquentiel et répétitif. Les objets d'apprentissage sont présentés au système selon un ordre prédéfini et les pondérations sont modifiées afin de corriger les erreurs de sortie. Si la sortie est correcte, les pondérations ne changent pas. La procédure d'ajustement des pondérations est présentée comme suit :

Algorithme : ajustement des pondérations

Début

1. Faire entrer un nouvel exemple
2. Si la classe de sortie C_s est différente de la classe correcte d'appartenance C_v recalculer les pondérations de chaque composante
3. Répéter 1 et 2 jusqu'à la fin de l'ensemble d'apprentissage
4. Si l'erreur globale obtenue pour l'ensemble des points d'apprentissage est supérieure à un seuil, répéter 1 - 3.

Fin

Typiquement, le cycle des opérations 1 - 3 s'appelle "epoch" d'apprentissage.

Les pondérations w_i sont initialisées de manière aléatoire (typiquement avec des valeurs $\in [0;1]$). Si l'on note les pondérations courantes $w_i(t)$, où t est un numéro utilisé afin d'identifier l'itération, le calcul des pondérations actualisées $w_i(t+1)$ se fait conformément à l'équation 3.20. La tâche du perceptron est alors de déterminer pour chaque itération la valeur Δw_i

$$\begin{aligned} w_0(t+1) &= w_0(t) + \Delta w_0(t) \\ w_i(t+1) &= w_i(t) + \Delta w_i(t), \forall i = \{1, \ldots, n\} \end{aligned} \tag{3.20}$$

Si l'on considère un problème de classification binaire (On ne dispose que de deux classes), une manière assez simple de construire le Δw_i est donnée par l'équation 3.21, où s est l'indice de la classe de sortie du perceptron et v la vraie classe d'appartenance de l'exemple. Par exemple, dans le cas d'une classification "false positive", où le système classifie de manière erronée l'exemple dans la classe C_1 ($s = 1; v = 0$), le biais est diminué d'une unité et les pondérations décrémentées proportionnellement à la valeur de l'attribut concerné. De même dans le cas d'une classification "false negative" ($s = 0; v = 1$), le biais est augmenté d'une unité et les valeurs des pondérations en proportion des valeurs d'attributs.

$$\begin{aligned} \Delta w_0(t) &= v - s \\ \Delta w_i(t) &= (v - s).x_i \forall i = 1, \ldots, n \end{aligned} \tag{3.21}$$

Pour le cas des classes non-séparables par une surface linéaire, des variantes, comme les systèmes d'apprentissage de type LMS (Least Mean Square) ont été proposées[31]. Ces systèmes reposent sur le fait que la sortie ne sera pas une classe, donc une décision claire, mais des "degrés d'appartenance" de l'exemple à chacune des classes apprises.La figure 3.10 montre le principe du perceptron linéaire LMS. Chaque sortie du système est un "degré d'appartenance" à une des classes apprises ($m = |C| - 1$). Les algorithmes d'apprentissage et d'utilisation sont simplement une répétition pour chaque classe des algorithmes décrits pour le perceptron linéaire. La sortie de chaque neurone, qui donne un "degré d'appartenance" à la classe correspondante, s'appelle "activation" du neurone. Elle est obtenue par l'équation 3.22 suivante :

$$A = \sum_{i=1}^{n} w_i.x_i + w_0 \tag{3.22}$$

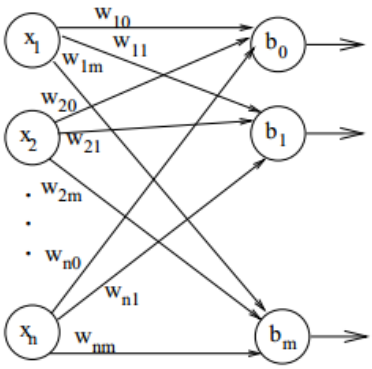

FIGURE 3.10 – Principe du perceptron linéaire LMS

Les réseaux multi-couches

Le principe de base des réseaux de neurones multi-couches est de combiner plusieurs perceptrons afin de raffiner la sortie. La combinaison la plus intuitive est de fournir les activations des neurones d'une première couche comme entrées d'une deuxième et ainsi de suite. Le principe général est présenté dans la figure 3.11.

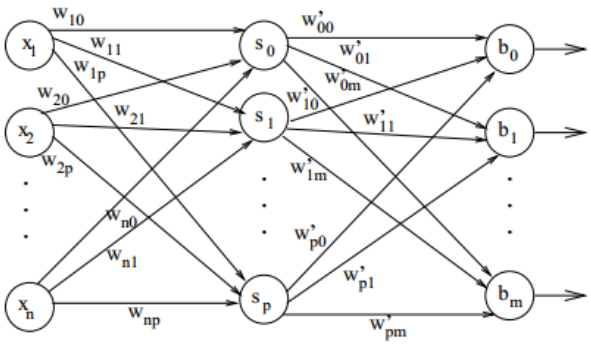

FIGURE 3.11 – Principe des réseaux de neuronnes multi-couches

Une couche intermédiaire de neurones, caractérisée par le biais s_i ;$i = 0, \ldots, p$, est intercalée entre les entrées et la couche de sortie. Les activations des neurones de la couche intermédiaire (habituellemnt appellée "couche cachée" -hidden layout) forment les entrées de la couche finale.

Pour le cas de réseaux de neurones multi-couches, l'activation est calculée d'une manière plus complexe (non linéaire) en comparaison au calcul de l'activation d'un perceptron monocouche Soit le neurone j , situé dans une couche quelconque du réseau. La valeur d'entrée caractéristique à ce neurone, In_j, est donnée par la somme entre le biais qui lui correspond et les sorties pondérées de toutes les unités de la couche antérieure auxquelles il est connecté. Les unités de la couche antérieure peuvent être des neurones cachés ou des attributs de l'exemple à analyser. L'activation du neurone j ; A_j, est une fonction qui dépend de cette valeur d'entrée :

$$A_j = \frac{1}{1 + x^{\ln j}} \tag{3.23}$$

L'apprentissage du réseau multi-couches se fait, comme pour le cas du perceptron linéaire, en plusieurs "epochs". Chaque "epoch" consiste à analyser successivement tous les vecteurs d'entrée qui forment l'ensemble d'apprentissage et à raffiner les poids des différents neurones des différentes couches si l'activation des neurones de la couche finale n'est pas celle attendue.

3.4.6 Modèle de Markov caché

Un modèle de Markov caché ou HMM (Hidden Markov Model) [35] est caractérisé par un double processus stochastique : un processus interne qui modélise les états non observables $X = \{x_1, x_2, \ldots, x_N\}$ d'un système (qui est sensé être un processus Markovien) et un processus externe qui modélise les observations $Y = v_1, v_2, \ldots, v_N$ du système dont les états sont cachés [10].
Cette méthode a été testée dans le domaine de la classification d'activités en environnements `perceptifs` [5] en partant du principe qu'une activité humaine est une séquence d'actions atomiques se déroulant les unes à la suite des autres [57, 44, 45] Dans ce cas, les états du système sont les activités définies et les observations sont les données issues des différents capteurs.
Un modèle HMM noté $\lambda = (A, B, \pi)$ est défini par :

1. L'ensemble de ses états $X = \{x_1, x_2, \ldots, x_N\}$;
2. L'ensemble des observations $Y = \{v_1, v_2, \ldots, v_N\}$;
3. Une matrice A de probabilités de transition entre les états de la chaîne : a_{ij} représente la probabilité de passer de l'état i à l'état j ;
4. Une matrice B de probabilités d'observation : b_{jk} représente la probabilité d'observer le symbole v_k quand le modèle est dans l'état j ;
5. Un vecteur π de densité de probabilité initiale, π_i représente la probabilité que l'état de départ du modèle soit à l'état i . La distribution de l'état initial $\pi = \{\pi_i\}$.

X prend ses valeurs dans un espace fini discret qui constitue les états du modèle. Y prend ses valeurs dans un espace qui peut être discret ou continu suivant la nature des séquences de données à modéliser. L'observation Y est une fonction probabiliste de l'état. La probabilité d'émission est notée $p(y|x)$, elle est donnée par la matrice B . Pour illustrer le principe, La figure 3.12 montre un exemple de chaîne de Markov cachée. Á chaque instant t le processus se trouve dans un des états X_i et émet une observation Y_i avec une certaine probabilité, à l'instant suivant $t+1$, soit il se produit une transition non déterministe vers un autre état, soit le processus reste dans le même état. Dans les 2 cas, une nouvelle observation est générée.

Les deux tâches de classification les plus courantes dans un HMM consistent à : trouver la séquence d'états la plus probable sachant une séquence d'observations, et calculer la probabilité qu'une certaine séquence soit générée.

La figure 3.13 montre un exemple de modèle de Markov caché pour l'activité 'préparer un repas', l'ensemble des états cachés sont "Entrer','Frigo','Cuisiner','sortir", les pondérations sur les

5. Un environnement que l'on peut observer

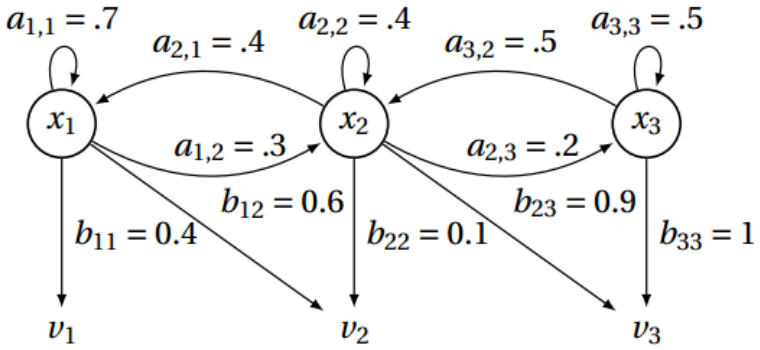

FIGURE 3.12 – Exemple de modèle de Markov caché [10]

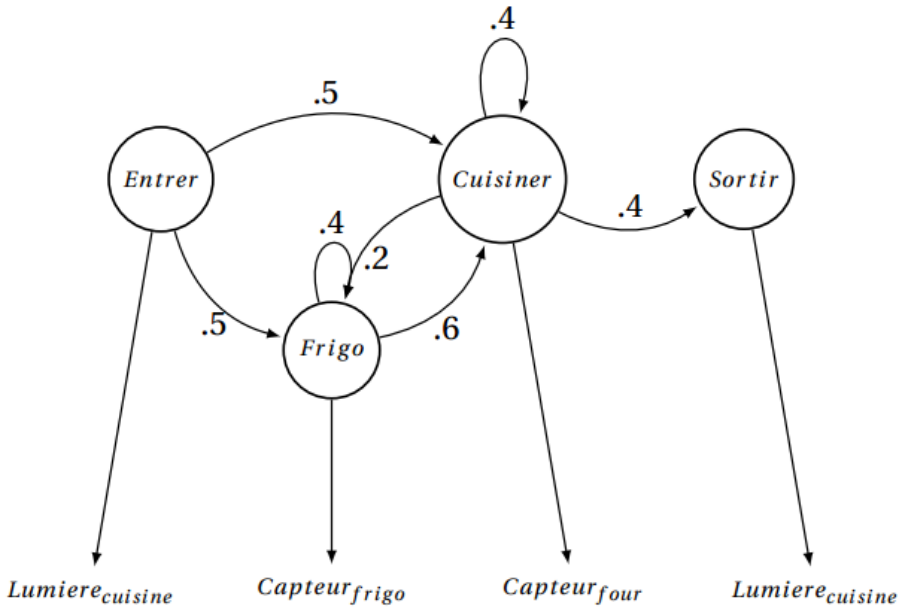

FIGURE 3.13 – Modèle de Markov caché pour l'activité "préparer un repas" [10]

flèches représentent les probabilités de passer d'un état à un autre, par exemple la probabilité de passage de l'état 'entrer' à l'état 'cuisiner' est de 0.5. L'ensemble des états observables de cet exemple de HMM est {'$Lumiere_{cuisine}$','$Capteur_{frigo}$','$Capteur_{four}$','$Lumiere'_{cuisine}$}. L'observation de cette séquence et ensuite sa modélisation avec le modèle de Markov caché va permettre de déduire que l'activité en cours est 'préparer le repas'.

3.5 Évaluation de performances d'un classifieur

Apprendre un classifieur revient à entrainer un modèle sur un ensemble d'apprentissage qui minimise le taux d'erreur, étant donné que le taux d'erreur est le nombre d'exemples mal classés. Le but est d'entrainer un classifieur qui fait moins d'erreurs si on lui présente de nou-

veaux cas qu'on n'a pas regardés pendant la phase d'entrainement. Dans ce cas, on parle de l'erreur de généralisation. Il existe plusieurs techniques permettant d'évaluer la performance d'un classifieur.

3.5.1 La validation croisée

Dans ce type de validation, les méthodes de classification sont évaluées sur une base de test. La procédure d'évaluation leave-one-out consiste à diviser la base d'exemples de taille n en n sous-bases[32]. Ensuite à entrainer le modèle sur $n-1$ sous-bases, puis le tester sur la base restante. Le processus se répéte un certain nombre de fois. On distingue d'autres techniques de la validation croisée :

- Méthode holdout : la procédure d'évaluation consiste à séparer toutes les données dont on dispose en deux ensembles, un ensemble d'entrainement et un ensemble de test. Ensuite à entrainer le modèle sur l'ensemble d'entrainement, puis l'évaluer sur l'ensemble de test.

- K-fold-cross-validation : la procédure d'évaluation consiste à diviser k fois l'ensemble de données de taille n en k sous-bases. Ensuite à entrainer le modèle sur $k-1$ sous-bases, puis le valider sur la base k. La procédure se répéte k fois. On note que cette procédure est appelée leave-one-out dans le cas où $k = n$.

3.5.2 Les courbes ROC

Une courbe ROC est un outil graphique qui vise à représenter, et à comparer la performance d'un classifieur par rapport à un autre en fonction du risque associé à chaque classe [58] (cf. figuer 3.14) La courbe ROC est inspirée de l'analyse de la matrice de confusion. Cette matrice permet de fournir deux quantités, la sensibilité et la spécificité. L'idée de la courbe ROC est de faire varier un seuil et, pour chaque cas, calculer la spécificité et la sensibilité que l'on reporte dans un graphique où l'inverse de la spécificité se place en abscisse et la sensibilité se trouve en ordonnée, et étant donné les deux mesures :

- La sensibilité = TP/TP+FN
- La spécificité= FN/FN+TP

	positif	négatif
positif	TP	FP
négatif	FN	TN

Table 3.3 – Matrice de confusion

où TP, FP, TN, FN représentent respectivement le nombre de positifs correctement reconnus, le nombre de négatifs détectés par erreur, le nombre de négatif correctement reconnus, le nombre de positifs détectés par erreur.

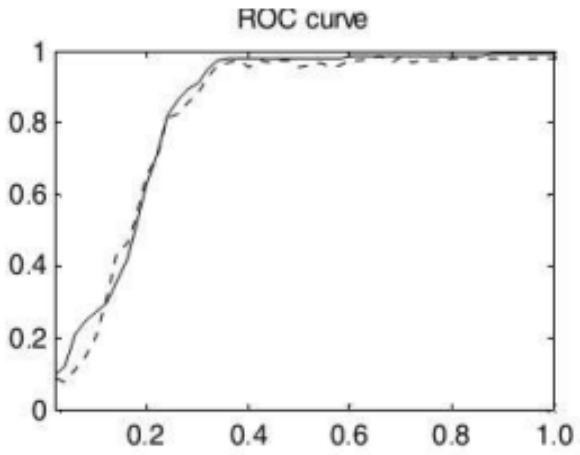

FIGURE 3.14 – Courbe ROC montrant la comparaison d'un classifieur SVM par rapport à un réseau de neuronnes [58]

3.6 Revue des travaux existants

Dans cette section nous allons présenter les principaux travaux publiés dans la littérature dans le domaine de la reconnaissance d'activités, dans lesquels on a utilisés les approches de classification présentées dans ce chapitre.

Bao L. et al.[2] utilisent plusieurs classifieurs tels que les arbres de décision, le modèle Bayésien naïf pour reconnaître des activités. Dans ce cas, les données sont obtenues à partir des `accelerometres`[6] placés dans différents endroits du corps humain à savoir les bras, la hanche, les cuisses , les mains, etc. La figure 3.15 montre un exemple d'un accéléromètre et les endroits sur lesquels il peut être placé. Dans ce travail, des descripteurs tels que la moyenne, l'entropie, et la corrélation des données ont été calculés et utilisés pour la classification. Les arbres de décision ont obtenu des meilleurs performances que le modèle Bayésien.

D'autres travaux utilisent d'autres classifieurs pour la reconnaissance d'activités comme les réseaux de neurones [60], KNN (K nearest neighbor),HMM et SVM [23], et le classifieur Bayésien [46].

6. Instruments éléctromécaniques qui mesurent l'accélération appliquée en agissant sur le long de leurs axes sensibles

(a) Accéléromètre (b) Un usager porte des accéléromètres

FIGURE 3.15 – Exemple d'un accéléromètre et les endroits sur lesquels il peut être placé

Dans le travail de [23], les accéléromètres sont utilisés pour collecter les activités réalisées par les usagers. La moyenne et la variance d'un signal ont été choisies comme des attributs (features), et sont calculées pour chaque signal ensuite L'algorithme SVM est utilisé pour effectuer la classification en attribuant à un échantillon de test, l'étiquette.

Panduranga et al. [36] ont utilisé un modèle HMM de base pour la reconnaissance d'activités dans un habitat intelligent. Les données dans ce cas sont issues de capteurs `RFID`[7]. Les capteurs RFID sont installés sur tout objet utilisé dans la vie quotidienne afin de récupérer l'information sur l'utilisation de cet objet par l'usager. Le même modèle est utilisé pour prédire les événements dans un habitat intelligent. La prédiction de l'événement à apparaître dans un instant t se fait en prenant l'historique des observations $y_1, y_2, \ldots, y_{t-1}$.

Kasteren et al.[48] utilisent le modèles bayésien naïf pour la reconnaissance d'activités basiques dans les résidences pour personnes âgées.

Buettner et al.[8] ont utilisé le modèle de Markov caché comme technique de classification supervisée pour la reconnaissance des activités de la vie quotidienne (AVQ) comme 'regarder TV', 'lire', 'dormir', les attributs en entrée du classifieur sont les noms des objets et les données RFID.

3.7 Conclusion

Les méthodes de classification sont utilisées pour inférer des classes (dans notre cas des activités), les méthodes que nous avons détaillé dans ce chapitre nous ont permis d'élaborer une proposition d'une méthode de classification. Le chapitre suivant concernera les détails de notre contribution dans le domaine de la classification des informations contextuelles.

7. Radio Frequency IDentification

4
Proposition

4.1 Introduction

Ce chapitre dresse les détails de notre proposition d'une technique de classification pour la reconnaissance des activités humaines. Tout d'abord, nous décrivons le jeu de données que nous utiliserons pour la validation. Ensuite, nous présentons et en détail les différentes étapes de construction et d'utilisation du modèle proposé. Enfin nous montrons les résultats de validation.

4.2 Jeu de données

Génération du jeu de données

Le jeu de données utilisé pour entraîner et valider la technique de classification proposée pour la reconnaissance d'activité est un jeu de données généré en 2008 pendant une expérimentation au laboratoire DOMUS [15] où des personnes ont dû :

- Se réveiller
- aller aux toilettes
- préparer le déjeuner
- manger
- faire la vaisselle

Dans le cadre de ce mémoire, nous n'allons considérer que les deux activités *(se réveiller)* et *(manger)* pour des raisons de simplification.

L'expérimentation a été réalisée avec six personnes différentes et a généré 433 instances d'activités, une instance correspond à une réalisation, par une personne, d'une activité particulière. Le nombres d'instances correspondant aux deux activités que nous avons choisi est de 158 instances. ces 158 instances recueillies ont été divisées en deux séries. La première série (contient 109 instances) correspond à la première fois que les personnes ont réalisé ces deux activités, elle a été utilisée pour l'apprentissage car 109 instances réparties sur deux activités différentes est largement suffisant pour faire l'apprentissage. La deuxième série (contient 49 instances d'activités) correspond à la deuxième fois que les utilisateurs ont réalisés ces deux activités, elle a été utilisée pour la validation.

Description du jeu de données

Le jeu de données permettant d'entraîner la technique de classification est composé de deux fichiers : le premier est généré automatiquement par le serveur et contient tous les évènements des capteurs générés avec l'heure précise de leur déclenchement. Le second est un fichier d'annotations, rempli "à la main" par une personne qui a analysé les activités réalisées et a indiqué pour chacune d'elles son heure de début et de fin.
Par exemple, d'après le fichier d'annotations, l'activité 1, qui correspond à l'activité "se réveiller", a été réalisée entre 09 :17 :46 et 09 :18 :00 (figure 4.1) En se référant au deuxième

```
Start time           End time             ID
-----------------------------------------
11-Aug-2008 09:17:46 11-Aug-2008 09:18:00 1
11-Aug-2008 09:18:04 11-Aug-2008 09:18:51 2
```

FIGURE 4.1 – Exemple d'un fichier d'annotation

fichier, le fichier d'évènements des capteurs (figure 4.2), on peut connaître quels évènements ont été générés lors de la réalisation de cette activité, soit les capteurs 5105, 0215 et 5101.

```
   Date  Time  ID   Type    Room   State
-------------------------------------------------
2008-08-11 9:17:46  5105  Lampe  Bedroom  Open
2008-08-11 9:17:48  0215  Door   Bedroom  Open
2008-08-11 9:17:53  5101  Lampe  Kitchen  Open
2008-08-11 9:17:57  5105  Lampe  Bedroom  Close
2008-08-11 9:18:0   0215  Door   Bedroom  Close
2008-08-11 9:18:4   0216  Door   Bathroom Open
2008-08-11 9:18:5   5106  Lampe  Bathroom Open
2008-08-11 9:18:9   0216  Door   Bathroom Close
2008-08-11 9:18:12  F101  Water  Bathroom Open
```

FIGURE 4.2 – Exemple d'un fichier d'évènements de capteurs

Ainsi, on est en mesure de dire que la génération des évènements 5105, 0215 et 5101 correspond à l'activité de se réveiller. Grâce à cette connaissance, il est possible d'entraîner notre algorithme de classification pour la reconnaissance d'activité.

4.3 Description de la proposition

L'analyse de l'état de l'art en matière de techniques de classification pour la reconnaissance d'activité, nous a permis de constater que les techniques les plus utilisées sont supervisées. Cette supervision nécessite l'utilisation de collections coûteuses et labelisées de larges ensembles d'apprentissage. Ensuite, elles présentent l'inconvénient de modélisation en fonction de la classe d'appartenance de chaque échantillon, ce qui nécessite de se référencer à l'ensemble d'apprentissage durant le processus de construction du modèle de classification.

Des travaux précédents ont montré qu'il est possible de reconnaître les mouvements ou les activités humaines en se basant sur des modèles appris de manière non supervisée ou semi-supervisée [27, 24]. Notre proposition découle de cette idée afin de remédier au problème de la modélisation en fonction de la classe d'appartenance, nous utilisons l'algorithme *K-Médoïdes* comme algorithme simple mais efficace de regroupement de données afin de partitionner l'ensemble des échantillons en groupes en fonction de la similarité que ces échantillons présentent entre eux et sans la nécessité de connaître la classe d'appartenance de chaque échantillon. Le résultat de ce regroupement est un ensemble de partitions, chacune d'entre elles est représentée par l'échantillon le plus central appelé *échantillon représentatif*, ensuite, il suffit d'attribuer une étiquette à une partition.

L'utilisation de la technique de classification que nous proposons se fait en deux étapes principales : l'apprentissage et la classification dont un petit aperçu est donné ci-dessous avant d'entrer dans les détails de chaque phase :

- **L'apprentissage :** La phase d'apprentissage ou de construction est l'étape qui consiste à construire un modèle qui sera exploité pour la classification, elle est à son tour constituée de deux étapes :

  1. **Clustérisation de l'ensemble des vecteurs d'activités** x **:** la clustérisation consiste à diviser l'ensemble des vecteurs d'activités en partitions appellées clusters dont chacun contient les vecteurs qui présentent une grande similarité l'un envers l'autre, l'algorithme choisi pour réaliser la clustérisation est l'algorithme *K-Médoïdes*, le résultat de la clustérisation est un ensemble de clusters, chacun d'entre eux est représenté par son vecteur représentatif

  2. **Annotation des clusters :** Elle survient après le partitionnement et elle consiste à attribuer une étiquette pour chaque cluster obtenu (pour chaque vecteur représentatif) dans la première phase (c.à.d. la clustérisation) ;

– **Classification :** Á ce niveau, il est question de l'exploitation du modèle construit lors de l'étape d'apprentissage, la classification est réalisée par l'algorithme k plus proches voisins.

4.4 Apprentissage

Les étapes de l'apprentissage d'une activité A sont :

1. Construction des vecteurs d'activités x à partir du jeux de données. ils contiennent des valeurs binaires indiquant si un capteur est activé ou non pendant la réalisation d'une activité donnée A
2. Partitionnement de l'ensemble de ces vecteurs d'activités en k partitions en utilisant l'algorithme *K-Médoïdes*.
3. L'annotation : consiste à attribuer au cluster (vecteur représentatif) une étiquette.

4.4.1 Construction des vecteurs d'activités

Les données d'entrée de notre algorithme, obtenues à partir du jeu de données discuté précédemment, sont des vecteurs indiquant si un capteur est activé ou non. À chaque fois qu'une activité est réalisée, on obtient une instance de cette activité, ce qui donne un vecteur d'activité. On représente un vecteur d'activité par x dont la dimension d correspond au nombre de capteurs.
Un grand nombre de capteurs entraîne donc une grande dimension. Pour éviter une dimension trop élevée, il faut donc sélectionner les capteurs les plus significatifs et importants, c'est-à-dire les mieux situés et les plus différentiables. Dans notre cas, nous allons nous intéresser à seulement cinq d'entre eux, ces capteurs sélectionnés correspondent aux capteurs qui s'activent lors de réalisation des activités que nous avons considéré dans le cadre de ce travail a savoir : *se réveiller* et *manger*.

$$x = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} \quad \text{Ce qui signifie} \quad \begin{bmatrix} \text{capteur capt1} & : & \text{non activé} \\ \text{capteur capt2} & : & \text{activé} \\ \text{capteur capt3} & : & \text{non activé} \\ \text{capteur capt4} & : & \text{activé} \\ \text{capteur capt5} & : & \text{activé} \end{bmatrix}$$

où

– capt1 : est un capteur de pression placé sur la porte de la chambre à coucher, on obtient grâce à lui une information sur l'ouverture et la fermeture de porte ;

– capt2 : est un capteur d'interrupteur des lumières placé sur l'interrupteur de lumière de la chambre à coucher, il indique si la lumière est allumée ou non ;

– capt3 : est un capteur infrarouge placé dans la cuisine, il permet de détecter la présence d'une personne dans la cuisine ;

– capt4 : est un contacteur pour placard placé dans le placard de la cuisine, il s'active lors de l'ouverture ou la fermeture de ce placard ;

– capt5 : est un débimètre placé sur le robinet du lévier de cuisine qui fournit une information sur l'ouverture ou la fermeture du robinet.

4.4.2 Partitionnement de l'ensemble des vecteurs d'activités

Pour mieux comprendre l'algorithme de clustérisation que nous avons choisi pour partitionner l'ensemble des vecteurs d'activités (*K-Médoïdes*), nous allons d'abord mettre en exergue quelques notions sur la clustérisation et nous expliquerons aussi l'algorithme des *K-Moyennes*, il est nécessaire de comprendre ce dernier pour comprendre le fonctionnement de l'algorithme que nous avons choisi.

Définition de la clustérisation

Le processus de grouper un ensemble d'objets en classes d'objets similaires est appelé *partitionnement* ou *clustérisation*. Un cluster est une collection de données qui présentent une similarité à l'intérieur du même cluster et une dissimilarité vis-a-vis des données des autres clusters. Un cluster de données peut être traité collectivement comme un seul groupe et donc la clustérisation peut être considérée comme une forme de compression de données.

La dissimilarité

Le calcul de la dissimilarité entre les objets de données pour la clustérisation dépend fortement des types de données que nous souhaitons clustériser. Dans notre cas, il s'agit de données binaires. les variables binaires ont seulement deux états : 0 ou 1, où 0 signifie que la variable est absente et 1 signifie que la variable est présente. Par exemple, étant donnée la variable capt1,

1 indique que le capteur capt1 est activé tandis que 0 indique que le capteur n'a pas été activé. La dissimilarité entre deux vecteurs i et j est obtenue par l'équation suivante [20] :

$$d(i,j) = \frac{r+s}{q+r+s+t} \tag{4.1}$$

où

- r : est le nombre de variables égales à 1 pour le vecteur i et 0 pour le vecteur j ;
- s : est le nombre de variables égales à 0 pour le vecteur i et 1 pour le vecteur j ;
- q : est le nombre de variables égales à 1 pour les deux vecteurs i et j ;
- t : est le nombre de variables égales à 0 pour les deux vecteurs i et j ;

Algorithme des K-moyennes

K-Moyennes est un algorithme qui prend k comme paramètre en entrée et partitionne un ensemble de n exemples en k partitions (clusters) de sorte que la similarité intracluster (similarité des exemples dans le même cluster) est grande mais la similarité inter-clusters (similarité des exemples appartenant à des clusters différents) est petite.

La similarité est mesurée par rapport à la valeur moyenne des objets dans le cluster, cette moyenne est considérée comme le centroïde du cluster.

L'algorithme K-Moyennes procéde comme suit :

- Premièrement, il sélectionne aléatoirement k exemples initiaux, chacun d'entre eux est considéré comme la moyenne ou le centre du cluster, pour chacun des exemples restants, un exemple est affecté au cluster pour lequel, il est le plus similaire, en se basant sur la distance entre l'exemple considéré et la moyenne du cluster.
- Deuxièmement L'algorithme calcule la nouvelle moyenne de chaque cluster.

Ce processus se répéte jusqu'à ce que la fonction de critère converge. la fonction utilisée est celle du critère de l'erreur quadratique, définie comme suit [20] :

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} |p - m|^2 \tag{4.2}$$

où

- E est la somme de l'erreur quadratique de tous les exemples dans l'ensemble de données ;
- p est un point dans l'espace représentant un exemple donné ;
- m_i est la moyenne du cluster C_i (p et m_i sont multi-dimensionnels).

Ce critère tente de rendre les k clusters résultants aussi compactes que possible et le mieux séparés.

L'algorithme k-moyennes est donné comme suit [20] :

Algorithme : K-moyennes.

□ **Entrées :**

$-K$: le nombre de clusters

$-D$: l'ensemble de données contenant n exemple

□ **Sortie :** un ensemble de k clusters

□ **Méthode :**

(1) choisir arbitrairement k exemples dans D comme les centres initiaux des clusters

(2) **répéter**

(a) (re)affecter chaque exemple au cluster pour lequel l'exemple est le plus similaire en se basant sur la valeur moyenne des exemples dans le cluster

(b) Mettre à jour les moyennes des clusters, c'est à dire la valeur moyenne des objets dans le cluster

(3) **jusqu'à** aucun changement dans les clusters

L'algorithme K-Medoïdes : Présentation générale

Au lieu de prendre la valeur moyenne des exemples dans un cluster comme point de référence comme dans le cas de l'algorithme K-Moyennes, on peut prendre les exemples pour représenter les clusters en utilisant un exemple représentatif par cluster, chacun des exemples restants

est clustérisé avec l'exemple représentatif pour lequel il est le plus similaire. La méthode de partionnement est ainsi effectuée en se basant sur le principe de minimisation de la somme des dissimilarités entre chaque objet et son point de référence correspondant. C'est le critère de l'erreur absolue qui est utilisé et qui est défini comme[20] :

$$E = \sum_{j=1}^{k} \sum_{p \in C_j} |p - o_j| \tag{4.3}$$

où

1. E est la somme des erreurs absolues de tous les exemples dans l'ensemble de données ;

2. p est un point dans l'espace qui représente un exemple donné dans un cluster C_j ;

3. o_j est l'exemple représentatif de C_j.

En général, l'algorithme se réitère jusqu'à, éventuellement, chaque exemple représentatif est rèellement le médoïde, ou l'exemple le plus central de son cluster. C'est le principe de base de la méthode K-Médoïdes pour grouper n exemples en K clusters.

Regardons de plus près le clustering K-Medoïdes. Les exemples représentatifs initiaux sont choisis arbitrairement, le processus itératif de remplacement des exemples représentatifs par des exemples non-représentatifs continue tant que le clustering résultant est amélioré. Cette qualité est estimée en utilisant la fonction de coût qui mesure la dissimilarité moyenne entre un exemple et l'exemple représentatif de son cluster, la fonction de coût est obtenue en effectuant la différence entre l'erreur absolue de l'itération i et l'erreur absolue de l'itération $i+1$

Pour déterminer si un exemple non-représentatif o_{random} est un bon remplacement pour l'exemple représentatif courant, o_j, les quatre cas suivants sont examinés pour chacun des exemples non-représentatifs, p , comme illustré sur la figure 4.3.

1. Cas1 : p appartient actuellement à l'exemple représentatif o_j. Si o_j est remplacé par o_{random} comme exeemple représentatif et p est le plus proche d'un des autres exemples représentatifs, o_i, $i \neq j$, alors p est réaffecté à o_i
2. Cas2 : p appartient actuellement à l'exemple représentatif o_j. Si o_j est remplacé par o_{random} comme exemple représentatif et p est le plus proche de o_{random}, alors p est réaffecté à o_{random}

3. Cas3 : p appartient actuellement à l'exemple représentatif o_i, $i \neq j$. Si o_j est remplacé par o_{random} comme exemple représentatif et p est toujours plus proche de o_i alors l'affectation ne change pas
4. Cas4 : p appartient actuellement à l'exemple représentatif o_i, $i \neq j$. Si o_j est remplacé par o_{random} comme exemple représentatif et p est plus proche de o_{random} alors p est réaffecté à o_{random}

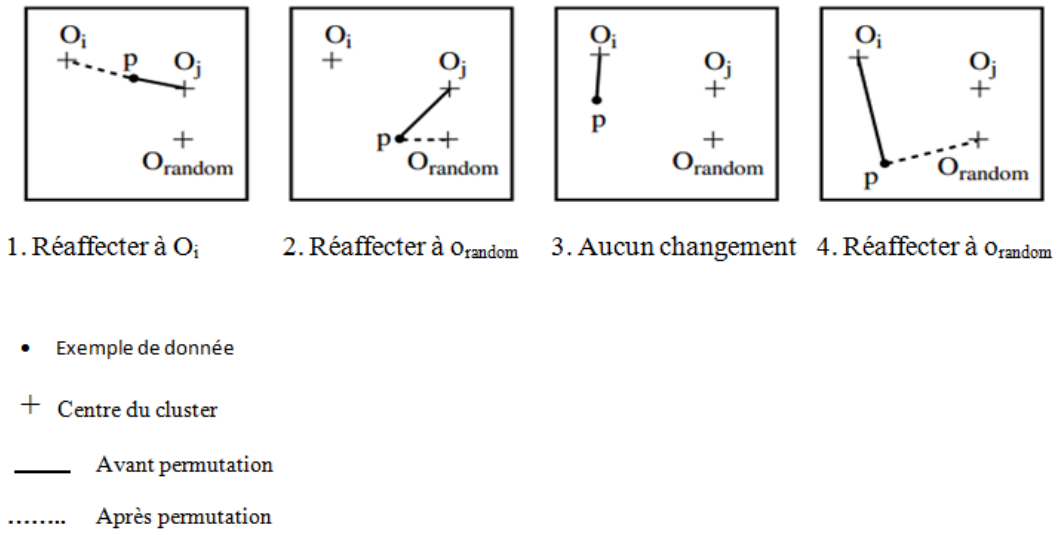

FIGURE 4.3 – Les quatre cas de la fonction de coût dans le clustering k-Medoïdes

Á chaque fois qu'une réaffectation a lieu, une différence dans l'erreur absolue est ajoutée à la fonction de coût. Donc, la fonction de coût calcule la différence dans la valeur de l'erreur absolue si l'exemple représentatif courant est remplacé par un exemple non représentatif.

Le coût total des permutations est la somme des coûts encourus par tous les exemples non-représentatifs. Si le coût est négatif alors l'exemple est remplacé ou permuté avec o_{random} de telle sorte que l'erreur absolue E est réduite. Si le coût total est positif alors l'exemple représentatif courant o_j est considéré acceptable, et aucun changement dans cette itération ne sera apporté.

PAM (Partitionning Around Medoids) était un des premiers algorithmes K-Medoïdes introduit, il tente de déterminer K partitions pour n exemples. Après une selection initiale aléatoire de K exemples représentatifs, l'algorithme tente de manière répététive de rendre le choix des exemples représentatifs meilleur. Toutes les paires possibles des exemples sont analysées, ou un exemple dans chaque paire est considéré comme l'exemple représentatif et l'autre non. Un exemple o_j est replacé avec l'exemple qui cause une grande réduction dans

l'erreur. l'ensemble des meilleurs exemples pour chaque cluster dans une itération forme les exemples représentatifs de la prochaine itération.
L'ensemble final des exemples représentatifs sont les médoïdes respectifs des clusters.
L'algorithme est comme suit :

Algorithme : K-Medoïdes.

□ **Entrées :**
- $-K$: le nombre de clusters
- $-D$: l'ensemble de données contenant n exemples

□ **Sortie :** un ensemble de k clusters

□ **Méthode :**

(1) choisir arbitrairement k exemples dans D comme des exemples représentatifs initiaux
(2) **répéter**
  (a) affecter chaque exemple restant à l'exemple représentatif le plus proche
  (b) Sélectionner aléatoirement un exemple non-représentatif o_{random}
  (c) Calculer le coût total S des permutations des exemples représentatifs o_j avec o_{random}, $S = E_{i+1} - E_i$ (la différence entre l'erreur absolue de l'itération $i+1$ et l'erreur absolue de l'itération i)
  (d) Si $S < 0$ alors permuter o_j avec o_r pour former le nouvel ensemble des k exemples représentatifs
(3) **jusqu'à** aucun changement dans les clusters

Application

L'objectif est d'utiliser cet algorithme (K-Médoïdes) afin de partitionner l'ensemble des vecteurs d'activités x, donc l'ensemble de ces vecteurs sera fourni comme entrée à cet algorithme, et il retournera les clusters correspondants et leurs vecteurs représentatifs.

4.4.3 Annotation des clusters

L'annotation est la tâche qui consiste à attribuer à chaque cluster une étiquette.

l'étiquette est en fait attribuée à l'exemple représentatif du cluster, puisque cet exemple est le plus central, ou exprimé d'une autre manière, il représente les autres exemples se trouvant à l'intérieur de ce cluster. Pour attribuer une étiquette à un cluster, on se référe à l'ensemble d'apprentissage qui contient les vecteurs d'activités que nous avons partitionner et leurs labels associés, le tableau 4.1 montre un exemple d'une base d'apprentissage.

Capt1	**Capt2**	**Capt3**	**Capt4**	**Capt5**	label
1	1	0	0	0	se réveiller
0	0	1	1	0	manger
1	1	0	1	0	manger
0	1	1	1	1	se réveiller
1	1	0	1	1	se réveiller
0	1	1	0	1	se réveiller
1	1	0	0	0	se réveiller
1	1	0	1	1	manger
0	1	1	1	1	manger
1	1	1	0	0	se réveiller

TABLE 4.1 – Exemple de base d'apprentissage

L'algorithme de l'annotation est comme suit :

Algorithme : Annotation des clusters.

□ **Entrées :**

- K vecteurs représentatifs
- La base d'apprentissage

□ **Sortie :** l'ensemble de vecteurs labélisés

□ **Méthode :**

Pour chaque vecteur représentatif

- attribuer l'étiquette appropriée en se basant sur l'ensemble d'apprentissage.

Fin Pour

4.4.4 Fonctionnement global de l'apprentissage

La figure suivante 4.4 montre le schéma général du fonctionnement de l'apprentissage.

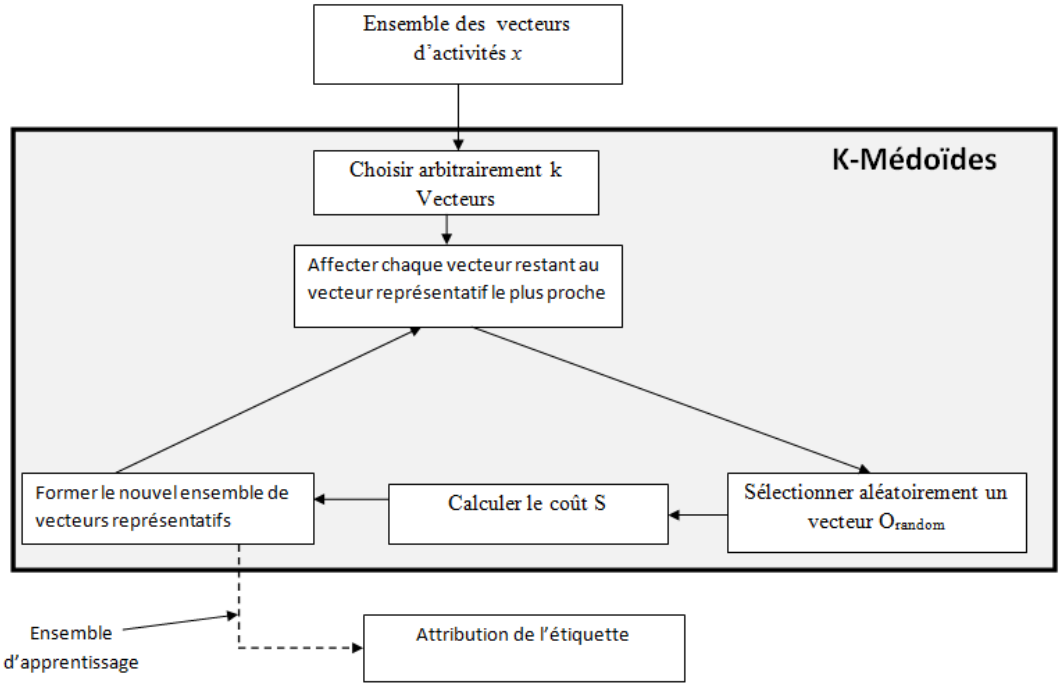

FIGURE 4.4 – Fonctionnement global de l'apprentissage

4.5 Classification

La classification consiste à utiliser le modèle appris durant l'apprentissage pour classer un nouvel élément (vecteur d'activité) dont on ignore le label.

La classification est effectuée en affectant au vecteur de test l'étiquette du vecteur représentatif le plus proche. La distance utilisée est la distance Euclidienne défine comme suit :

$$D(X,Y) = \sqrt{\sum_{i=1}^{n}(X_i - Y_i)^2} \tag{4.4}$$

où :

- $X = (x_1, x_2, \ldots, x_n)$ est le vecteur test dont on souhaite trouver le label ;
- $Y = (y_1, y_2, \ldots, y_n)$ est le vecteur représentatif.

L'algorithme de classification, par conséquent se présente comme suit :

Algorithme : Classification.

□**Entrées :**

- K vecteurs représentatifs
- Un vecteur de test $vect\_test_i$

□**Sortie :** Un vecteur $vect\_test_i$ labelisé

□**Méthode :**

pour chaque vecteur représentatif $vect\_rep_i$

- Calculer la distance $dist_i$ du vecteur test $vect\_test_i$ par rapport à $vect\_rep_i$

Fin pour

- choisir le vecteur représentatif qui a la distance minimale
- Affecter l'étiquette de ce vecteur a $vect\_test_i$

L'algorithme aura comme entrées les vecteurs représentatifs des clusters obtenus en appliquant l'algorithme K Médoïdes et le vecteur de test pour lequel on cherche l'étiquette, ces vecteurs représentatifs sont les k plus proches voisins, il procéde ensuite au calcul des distances entre chaque vecteur représentatif et le vecteur de test. Ensuite, il attribue au vecteur de test l'étiquette du vecteur représentatif qui présente la distance minimale.

La figure 4.5 montre l'organigramme de la méthode de classification correspondante à l'algorithme précédent.

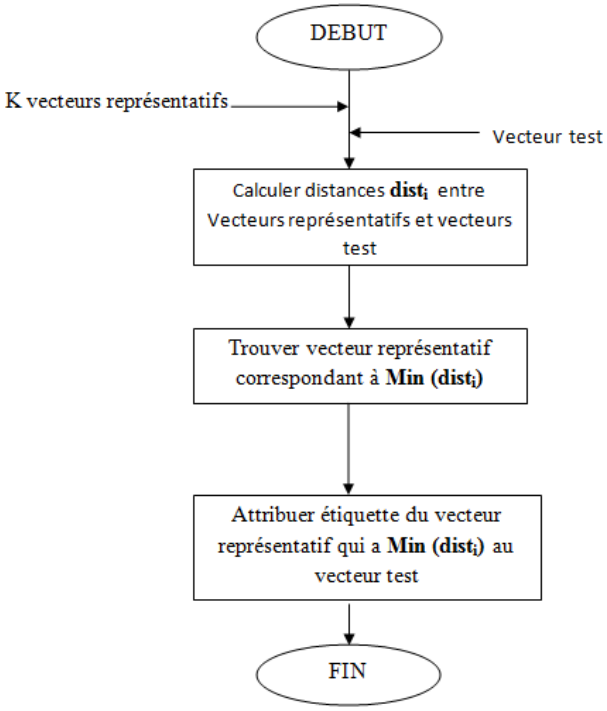

FIGURE 4.5 – Organigramme de la méthode de classification proposée

4.6 Validation

Lorsqu'on applique la technique de classification proposée sur le jeu de données présenté dans la section 4.2 et plus précisemment sur l'ensemble des vecteurs test qui contient 49 instances d'activités labélisées, on obtient les résultats présentés dans le tableau 4.2.

Activités	**Nombre total d'instances**	**Nombre d'instances reconnues**	**pourcentage**
Se réveiller	27	20	74.07
Manger	22	22	100

TABLE 4.2 – Résultats de validation de la technique de classification proposée

Les résultats sont intéressants, puisque 85.71% des activités sont reconnues correctement.

Le taux de reconnaissance représente le pourcentage des vecteurs test correctement classifiés. La matrice de confusion est l'outil utilisé pour analyser à quel point la technique de classification peut reconnaître les vecteurs test de différentes classes.

La matrice de confusion associée à notre technique de classification est comme suit :

Activités	Se réveiller	manger	taux de reconnaissance
Se réveiller	20	7	74.07
Manger	0	22	100

TABLE 4.3 – Matrice de confusion des deux activités *manger* et *se réveiller*

4.7 Conclusion

Dans ce chapitre qui clos ce mémoire, nous avons présenté notre contribution d'une technique de classification pour la reconnaissance d'activité en soulignant les lacunes qu'elle comble constatées dans les l'état de l'art dans le chapitre précédent, et en détaillant les différentes phases qui la compose. Les résultats de validation sont aussi présentés.

Conclusion générale

Dans cette conclusion, il nous a semblé tout à fait indiqué de repréciser la problématique traitée ainsi que la nature de notre contribution par rapport à cette dernière.

Tout d'abord, nous avons montré au travers de l'état de l'art dans le troisième chapitre que d'une part les méthodes de classification existantes sont supervisées, elles utilisent l'information de la classe d'appartenance de chaque échantillon de données pour construire le modèle de classification, et d'autre part, les modèles existants sont difficiles à implémenter. Ceci nous a permis dans la perspective de réduire la supervision, de souligner la nécessité d'aborder ces problèmes de façon générale.

Dans le quatrième chapitre, nous avons détaillé notre contribution, en présentant les différentes phases qui la compose. Le sens de cette contribution n'était pas seulement de montrer qu'il était important de réduire la supervision, mais aussi de pourvoir une méthode de classification facile à implémenter et à mettre en oeuvre.

Il nous paraît également judicieux de décrire les étapes par lesquelles nous sommes passés pour aboutir à l'achèvement de ce mémoire :

La première étape de notre recherche à permis de faire des investigations en profondeur concernant la problématique abordée dans ce mémoire, a savoir, la classification des données pour la reconnaissance d'activité. Cette étape a servi à faire une étude ciblée des différentes méthodes existantes afin de bien comprendre les caractéristiques de chaque méthode et les limites qu'elles présentent.

La deuxième étape consistait à proposer une solution afin de faire face aux limites soulevées dans la première étape.

La dernière étape consistait à valider le modèle développé dans l'étape précédente de façon à vérifier sa fonctionnalité et sa performance dans un contexte réel avec des données réelles. Pour

ce faire, nous avons utilisé une base de données contenant différents types d'activités, nous nous somme focalisé sur deux activités.

Ce travail ouvre de nombreuses perspectives, entre autres :

- L'amélioration de notre technique de classification en se débarrassant complètement de la supervision.
- Pouvoir implémenter cette technique sur une plate forme mobile (Smartphone) disposant de caractéristiques de traitement réduites (microprocesseur de capacité limitée).
- Concevoir un calendrier électroniques qui à partir de reconnaissance d'activités par la technique proposée va pouvoir rappeler les activités, il peut être utilisé par les personnes âgées atteintes de maladies cognitives pour les rappeler des activités qu'ils oublient.

Bibliographie

[1] G. D Abowd, A. K Dey, P. J Brown, N. Davies, M. Smith, and P. Steggles. Towards a better understanding of context and context-awareness. In *Handheld and ubiquitous computing*, pages 304–307. Springer, 1999.

[2] L. Bao and S. S Intille. Activity recognition from user-annotated acceleration data. In *Pervasive computing*, pages 1–17. Springer, 2004.

[3] J. Baron. *Thinking and deciding*. Cambridge University Press, 2000.

[4] C. Becker and F. Dürr. On location models for ubiquitous computing. *Personal and Ubiquitous Computing*, 9(1) :20–31, 2005.

[5] C. Bettini, O. Brdiczka, K. Henricksen, J. Indulska, D. Nicklas, A. Ranganathan, and D. Riboni. A survey of context modelling and reasoning techniques. *Pervasive and Mobile Computing*, 6(2) :161–180, 2010.

[6] B. E Boser, I. M Guyon, and V. N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

[7] P. J Brown, J. D Bovey, and X. Chen. Context-aware applications : from the laboratory to the marketplace. *Personal Communications, IEEE*, 4(5) :58–64, 1997.

[8] M. Buettner, R. Prasad, M. Philipose, and D. Wetherall. Recognizing daily activities with rfid-based sensors. In *Proceedings of the 11th international conference on Ubiquitous computing*, pages 51–60. ACM, 2009.

[9] C. CAUVET. *E-CARe : une méthode d'ingénierie des Systèmes d'Information ubiquitaires*. Thèse de doctorat, Université Pierre Mendès France, 2012.

[10] P. Chahuara Quispe. *Contrôle intelligent de la domotique à partir d'informations temporelles multi sources imprécises et incertaines*. Thèse de doctorat, Grenoble, 2013.

[11] I. Charfi. *Détection automatique de chutes de personnes basée sur des descripteurs spatio-temporels : définition de la méthode, évaluation des performances et implantation temps-réel*. Thèse de doctorat, Université de Bourgogne, 2013.

[12] P. Cheeseman, J. Kelly, M. Self, J. Stutz, W. Taylor, and D. Freeman. Autoclass : a bayesian classification system. In *Readings in knowledge acquisition and learning*, pages 431–441. Morgan Kaufmann Publishers Inc., 1993.

[13] G. Chen, D. Kotz, et al. A survey of context-aware mobile computing research. Rapport technique, Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College, 2000.

[14] H. Chen, F. Perich, T. Finin, and A. Joshi. Soupa : Standard ontology for ubiquitous and pervasive applications. In *Mobile and Ubiquitous Systems : Networking and Services,*

2004. MOBIQUITOUS 2004. The First Annual International Conference on, pages 258–267. IEEE, 2004.

[15] B. Chikhaoui, S. Wang, and H. Pigot. A frequent pattern mining approach for adls recognition in smart environments. In *Advanced Information Networking and Applications (AINA), 2011 IEEE International Conference on*, pages 248–255. IEEE, 2011.

[16] A. K Dey. Understanding and using context. *Personal and ubiquitous computing*, 5(1) :4–7, 2001.

[17] A. K Dey, D. Salber, G. D Abowd, and M. Futakawa. The conference assistant : Combining context-awareness with wearable computing. In *Wearable Computers, 1999. Digest of Papers. The Third International Symposium on*, pages 21–28. IEEE, 1999.

[18] U. M Fayyad and K. B Irani. On the handling of continuous-valued attributes in decision tree generation. *Machine learning*, 8(1) :87–102, 1992.

[19] D. Figo, P. C Diniz, D. R Ferreira, and J. MP Cardoso. Preprocessing techniques for context recognition from accelerometer data. *Personal and Ubiquitous Computing*, 14(7) :645–662, 2010.

[20] J. Han, M. Kamber, and J. Pei. *Data mining : concepts and techniques*. Morgan kaufmann, 2006.

[21] K. Henricksen and J. Indulska. Developing context-aware pervasive computing applications : Models and approach. *Pervasive and mobile computing*, 2(1) :37–64, 2006.

[22] C. Hoareau and I. Satoh. Modeling and processing information for context-aware computing : A survey. *New Generation Computing*, 27(3) :177–196, 2009.

[23] T. Huỳnh, U. Blanke, and B. Schiele. Scalable recognition of daily activities with wearable sensors. In *Location-and context-awareness*, pages 50–67. Springer, 2007.

[24] T. Huỳnh and B. Schiele. Unsupervised discovery of structure in activity data using multiple eigenspaces. In Springer, editor, *2nd International Workshop on Location and Context-Awareness (LoCA)*, volume 3987 of LNCS, pages citée dans les pages 6,10,15,19,20,56, Dublin, Ireland, Mai 2006.

[25] W. Jouve. *Approche déclarative pour la génération de canevas logiciels dédiés à l'informatique ubiquitaire*. Thèse de doctorat, Université Sciences et Technologies-Bordeaux I, 2009.

[26] S. Long, R. Kooper, G. D Abowd, and C. G Atkeson. Rapid prototyping of mobile context-aware applications : The cyberguide case study. In *Proceedings of the 2nd annual international conference on Mobile computing and networking*, pages 97–107. ACM, 1996.

[27] D. Minnen, T. Starner, I. Essa, and C. Isbell. Discovering characteristic actions from on-body sensor data. In 16 19 72 citée dans les pages 10, 15, editor, *Proc. ISWC*, Octobre 2006.

[28] S. Najar, M. K. Pinheiro, B. Le Grand, C. Souveyet, et al. Systèmes d'information pervasifs et espaces de services : Définition d'un cadre conceptuel. *UbiMob 2013 : 9èmes journées francophones Mobilité et Ubiquité*, 2013.

[29] L B. JH Friedman RA Olshen and C. J Stone. Classification and regression trees. *Wadsworth International Group*, 1984.

[30] P. Öztürk and A. Aamodt. Towards a model of context for case-based diagnostic problem solving. In *Context-97; Proceedings of the interdisciplinary conference on modeling and using context*, pages 198–208. Citeseer, 1997.

[31] S. K Pal and S. Mitra. Multilayer perceptron, fuzzy sets, and classification. *IEEE Transactions on Neural Networks*, 3(5) :683–697, 1992.

[32] K. L Priddy and P. E Keller. *Artificial neural networks : an introduction*, volume 68. SPIE Press, 2005.

[33] J. R. Quinlan. Induction of decision trees. *Machine learning*, 1(1) :81–106, 1986.

[34] J. R. Quinlan. *C4. 5 : programs for machine learning*, volume 1. Morgan kaufmann, 1993.

[35] L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2) :257–286, 1989.

[36] S. Panduranga Rao and D. J Cook. Predicting inhabitant action using action and task models with application to smart homes. *International Journal on Artificial Intelligence Tools*, 13(01) :81–99, 2004.

[37] I. Rish. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46, 2001.

[38] D. Roggen, S. Magnenat, M. Waibel, and G. Troster. Designing and sharing activity-recognition systems across platforms. *IEEE ROBOTICS & AUTOMATION MAGAZINE*, 2011.

[39] N. S Ryan, J. Pascoe, and D. R Morse. Enhanced reality fieldwork : the context-aware archaeological assistant. In *Computer applications in archaeology*. Tempus Reparatum, 1998.

[40] G. Schiele, M. Handte, and C. Becker. Pervasive computing middleware. In *Handbook of Ambient Intelligence and Smart Environments*, pages 201–227. Springer, 2010.

[41] B. Schilit, N. Adams, and R. Want. Context-aware computing applications. In *Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on*, pages 85–90. IEEE, 1994.

[42] B. N Schilit and M. M Theimer. Disseminating active map information to mobile hosts. *Network, IEEE*, 8(5) :22–32, 1994.

[43] D. Siewiorek, A. Krause, N. Moraveji, A. Smailagic, J. Furukawa, K. Reiger, Fei L. Wong, and J. Shaffer. Sensay : A context-aware mobile phone. In *2012 16th International Symposium on Wearable Computers*, pages 248–248. IEEE Computer Society, 2003.

[44] G. Singla, D. J Cook, and M. Schmitter-Edgecombe. Incorporating temporal reasoning into activity recognition for smart home residents. In *Proceedings of the AAAI workshop on spatial and temporal reasoning*, pages 53–61, 2008.

[45] M. Stikic, T. Huynh, K. Van Laerhoven, and B. Schiele. Adl recognition based on the combination of rfid and accelerometer sensing. In *Pervasive Computing Technologies for Healthcare, 2008. PervasiveHealth 2008. Second International Conference on*, pages 258–263. IEEE, 2008.

[46] E. Munguia Tapia, Stephen S Intille, and K. Larson. *Activity recognition in the home using simple and ubiquitous sensors*. Springer, 2004.

[47] `http://www.vias.org/tmdatanaleng/cc\_classif\_knn.html` et `http://www.nysaes.cornell.edu/fst/faculty/siebert/FS608/syllabus.html`. date dernière consultation : 21 juin 2014.

[48] T. van Kasteren and B. Krose. Bayesian activity recognition in residence for elders. 2007.

[49] R. Want, A. Hopper, V. Falcao, and J. Gibbons. The active badge location system. *ACM Transactions on Information Systems (TOIS)*, 10(1) :91–102, 1992.

[50] J. A Ward, P. Lukowicz, G. Troster, and T. E Starner. Activity recognition of assembly tasks using body-worn microphones and accelerometers. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(10) :1553–1567, 2006.

[51] L. Wehenkel. Discretization of continuous attributes for supervised learning. variance evaluation and variance reduction. In *Proceedings of the Seventh International Fuzzy Systems Association World Congress*, volume 1, pages 413–418, 1997.

[52] M. Weiser. The computer for the 21st century. *Scientific american*, 265(3) :94–104, 1991.

[53] M. Weiser. Hot topics-ubiquitous computing. *Computer*, 26(10) :71–72, 1993.

[54] M. Weiser. Some computer science issues in ubiquitous computing. *Communications of the ACM*, 36(7) :75–84, 1993.

[55] M. Weiser. The world is not a desktop. *interactions*, 1(1) :7–8, 1994.

[56] S. M Weiss and C. A Kulikowski. Computer systems that learn : Classification and prediction methods from statistics, neural nets, machine learning and exp. 1990.

[57] D. H Wilson, D. Wyatt, and M. Philipose. Using context history for data collection in the home. *COGNITIVE SCIENCE RESEARCH PAPER-UNIVERSITY OF SUSSEX CSRP*, 577 :17, 2005.

[58] W. Xu-hui, S. Ping, C. Li, and W. Ye. A roc curve method for performance evaluation of support vector machine with optimization strategy. In *Computer Science-Technology and Applications, 2009. IFCSTA'09. International Forum on*, volume 2, pages 117–120. IEEE, 2009.

[59] H. Yan and T. Selker. Context-aware office assistant. In *Proceedings of the 5th international conference on Intelligent user interfaces*, pages 276–279. ACM, 2000.

[60] J. Yang, J. Wang, and Y. Chen. Using acceleration measurements for activity recognition : An effective learning algorithm for constructing neural classifiers. *Pattern recognition letters*, 29(16) :2213–2220, 2008.

[61] S. Zaidenberg. *Apprentissage par renforcement de modeles de contexte pour l'informatique ambiante*. Thèse de doctorat, Institut National Polytechnique de Grenoble-INPG, 2009.

Résumé

Dans l'informatique ubiquitaire, les systèmes de reconnaissance d'activité peuvent être utilisés pour étiqueter de gros ensembles de données. La reconnaissance des activités à partir de données de capteurs est un paradigme clé de l'informatique ubiquitaire. La variabilité dans les activités humaines, les caractéristiques de déploiement de capteurs, et les domaines d'application, ont conduit à l'élaboration de meilleures pratiques et des méthodes pour améliorer la robustesse des systèmes de reconnaissance d'activité.

La classification constitue une des étapes les plus importantes qui vise à rendre le processus de reconnaissance plus expressif et réduire l'incertitude, ainsi de minimiser la représentation.

Dans ce mémoire, nous abordons les méthodes de classification et nous explicitons comment elles sont utilisées pour la reconnaissance d'activité. Nous mettons en avant une méthode qui utilise l'algorithme de partitionnement k-Médoïdes. Les différentes phases de construction du modèle de classification y sont présentées en détail. Enfin sont présentés les résultats de validation.

Mots clés : informatique ubiquitaire, contexte, classification, k-moyennes, k-médoïdes.

Abstract

In ubiquitous computing, activity recognition systems can be used to label large sets of data. Recognition of activities from data sensors is a key paradigm in ubiquitous computing. Variability in human activities, deployment sensors characteristics, and application domains leads to the elaboration of best practices and methods to improve robustness of activity recognition systems.

Classification is an important step who aims to yield the recognition process more expressive and reduce uncertainty, therefore minimize representation.

In this report, we address classification methods and showed explicitly how they are used for activity recognition. We propose a method which uses the clustering algorithm k-Medoids. Different stages of construction of the model of classification are showed in detail. Finally validation results are presented.

Key words : ubiquitous computing, context, classification, k-means, k-medoids.