

Université Pierre et Marie Curie

École doctorale des Sciences mécaniques,
acoustique, électronique et robotique de Paris

Laboratoire de Vision et Ingénierie des Contenus (CEA, LIST, DIASI)

Segmentation supervisée d'actions à partir de primitives haut niveau dans des flux vidéos

Par Adrien CHAN-HON-TONG

Thèse de doctorat de Génie informatique, automatique et traitement du signal

Dirigée par Catherine ACHARD
et encadrée par Laurent LUCAT

Présentée et soutenue publiquement le 29/09/2014

Devant un jury composé de :

CAPLIER Alice, Professeur, Rapporteur
CANU Stéphane, Professeur, Rapporteur
CORD Matthieu, Professeur, Examinateur
EL YACOUBI Mounim A., Maitre de conférences, Examinateur
ACHARD Catherine, Maitre de conférences, Directeur de thèse
LUCAT Laurent, Ingénieur – Chercheur, Encadrant de thèse

A mes parents.

A ma femme et mes enfants.

Résumé

Cette thèse porte sur la segmentation supervisée de flux vidéo dans un contexte applicatif lié à la reconnaissance d'actions de la vie courante. La méthode de segmentation proposée est dérivée la méthode des *modèles de formes implicites* (*Implicit Shape Model*) et s'obtient en optimisant les votes présents dans cette méthode d'élection. Nous démontrons que cette optimisation (dans un contexte de fenêtre temporelle glissante) peut être exprimée de manière équivalente dans le formalisme des SVM en imposant une contrainte de cohérence temporelle à l'apprentissage, ou, en représentant la fenêtre glissante selon une *décomposition pyramidale dense*. Tout ce processus est validé expérimentalement sur un jeu de données de la littérature de segmentation supervisée. Il y surpasse les autres méthodes de type *modèles de formes implicites* et le *SVM* linéaire standard.

La méthode proposée est ensuite mise en œuvre dans le cadre de la segmentation supervisée d'actions. Pour cela, des primitives dédiées sont extraites des données *squelettes* de la personne d'intérêt obtenues grâce à des logiciels standards. Ces primitives sont ensuite quantifiées puis utilisées par la méthode d'élection. Ce système de segmentation d'actions obtient les meilleurs scores de l'état de l'art sur un jeu de données de la littérature de reconnaissance d'actions, ce qui valide cette combinaison des primitives et de la méthode d'élection.

Dans un dernier temps, nous montrons la faisabilité du déploiement de la méthode présentée dans un contexte applicatif réel. D'une part, un ensemble d'expérimentations soutient la disponibilité des primitives *squelettes* lors de la reconnaissance d'activités de la vie courante. D'autre part, un ensemble de techniques permet de réduire considérablement les ressources matérielles nécessaires au système. Cela est rendu possible par la faible complexité calculatoire de la méthode d'élection mais n'est réalisé que grâce à un travail spécifique sur l'architecture du système.

mots clés : segmentation supervisée, apprentissage statistique, traitement de vidéos, reconnaissance d'actions, actions de la vie courante, primitives *squelettes*.

Abstract

This thesis focuses on the supervised segmentation of video streams within the application context of daily action recognition. A segmentation algorithm is obtained from *Implicit Shape Model* by optimising the votes existing in this polling method. We prove that this optimisation can be linked to the sliding windows plus *SVM* framework and more precisely is equivalent with a standard training by adding temporal constraints, or, by encoding the data through a dense pyramidal decomposition. This algorithm is evaluated on a public database of segmentation where it outperforms other *Implicit Shape Model* like methods and the standard linear *SVM*.

This algorithm is then integrated into an action segmentation system. Specific features are extracted from the *skeleton* obtained from the video by standard software. These features are then clustered and given to the polling method. This system, combining our feature and our algorithm, obtains the best published performance on a human daily action segmentation dataset.

Finally, we highlight the algorithm capacity to deal with real application. First, we perform a set of experiments where skeleton based features are found relevant for human daily action. Then, we dramatically reduce the amount of machine resource needed by the system. This reduction is possible thanks to specific properties of the polling method but is realized only thanks to a work on the system structure.

keywords : supervised segmentation, machine learning, computer vision, action recognition, human daily actions, *skeleton* based features.

Remerciements

Je voudrais remercier mes encadrants de thèse Laurent LUCAT et Catherine ACHARD pour le soutien qu'ils m'ont apporté tout au long de cette thèse, ainsi que pour la préparation mon avenir professionnel après la thèse.

Merci pour le temps passé à comprendre mes travaux ; et ce n'était pas toujours facile entre les longues formules mathématiques et la clarté de mon expression !

Merci pour les lettres de recommandation demandées le samedi matin pour le lundi et pour les relectures d'articles nocturnes.

Merci pour les critiques et les encouragements à ma démarche, tour à tour nécessaires.

Merci de m'avoir aidé à dépasser mes intuitions pour me permettre de donner une cohérence à mon travail qu'il n'aurait sinon pas eu.

Merci pour m'avoir (presque) convaincu qu'il y a généralement un pourquoi derrière un savoir.

Je tiens également à remercier les membres du jury pour leur relecture du manuscrit, pourtant travaillé, mais encore bien améliorable. Je remercie en particulier Mounim A. EL YACOUBI pour sa relecture attentive, ainsi que les rapporteurs Alice CAPLIER et Stéphane CANU pour leur travail d'évaluation du manuscrit.

Je tiens aussi à remercier toutes les personnes du Laboratoire de Vision et d'Ingénierie des Contenus et en particulier les thésards et anciens thésards du laboratoire pour la bonne ambiance de travail et pour les nombreux bons moments passés ensemble.

Je tiens enfin à remercier toute ma famille pour le soutien qu'elle m'a apporté tout au long de cette thèse.

Table des matières

Résumé **3**

Abstract **4**

Introduction générale **11**

I L'apprentissage statistique **17**

1 Définition de la classification supervisée **17**
1.1 Description du problème de classification supervisée 17
1.1.1 Les objets manipulés 17
1.1.2 L'introduction de probabilités 17
1.1.3 L'objectif de l'apprentissage statistique 18
1.2 *No free lunch theorem* : la non existence d'une solution 18
1.3 Lignes directrices et taxonomie des contextes 19

2 Algorithmes asymptotiquement universels **21**
2.1 Plus proche voisin 21
2.2 K plus proches voisins 21
2.3 Estimation de densité 22
2.4 Malédiction de la dimension 22

3 La minimisation du risque structurel **23**
3.1 Borne probabiliste sur l'erreur 23
3.2 SVM : la maximisation de la marge 24
3.3 Généralisation aux frontières non linéaires 26

4 Le SVM à plus de deux classes **27**
4.1 *1 contre tous* 27
4.2 *1 contre 1* 29
4.3 Associer un score à chaque classe 29
4.4 Créer une hiérarchie 30

5 Classification de suites **32**
5.1 Intérêts des méthodes spécifiques 32
5.2 DTW : association élastique d'instants 33
5.3 Utilisation de la DTW 34

6 Classification de multi ensembles de mots **34**
6.1 *CNB* : la classification naïve bayésienne 35
6.2 Décision par élection 36
6.3 La *CNB* comme solution d'une optimisation 36
6.4 Cas du suffrage universel 37
6.5 Lien avec l'application d'un *SVM* 37

7 Apprendre sur les zones homogènes **39**
7.1 Segmentation construite sur les zones homogènes 39
7.2 Suppression des non maximaux 40
7.3 Pavage 41

8 Apprendre sur des configurations locales **42**
8.1 Segmentation construite sur des termes locaux 42
8.2 Utilisation d'états supplémentaires 43
8.3 Optimisation des termes 43

9 Apprendre des votes temporels **44**

II Les systèmes de reconnaissance d'actions **46**

10 Comparer les informations **47**
10.1 Primitives et taxonomie des systèmes vis à vis de la structure de l'extraction des primitives 47
10.2 Technique de quantification 48
10.2.1 K moyennes 49
10.2.2 Codage et agglomération 50
10.3 Taxonomie des systèmes vis à vis de la nature de l'extraction des primitives 51

11 Extraction de formes **52**
11.1 Descripteurs de voisinage 52
11.2 Extraction de points d'intérêts 54
11.2.1 Détection des points 54
11.3 Description d'une image 55
11.4 Description d'une vidéo 55

12 Extraction de formes grâce aux mouvements **55**
12.1 La silhouette 56
12.2 Estimation la silhouette 56
12.2.1 Différence d'images 56
12.2.2 Soustraction de fond 57

13 Extraction de mouvements grâce aux formes **58**
13.1 Extraction de trajectoire de points d'intérêts 58
13.2 Flot optique 58
13.3 Descripteurs de trajectoires 59

14 Les systèmes *en un seul tenant* **59**
14.1 *Machine de Boltzmann réduite* 60
14.2 Arbre de décision et *forêt de Hough* 60
14.3 Fusion de critères 63

15 Adaptation générique de la dimension **65**
15.1 Analyse en composante principale 65
15.2 Sélection de dimensions et de mots 66
15.2.1 Sélection a priori 66
15.2.2 Sélection a posteriori 67
15.2.3 Sélection simultanée 68
15.3 Construction de dimensions ou de mots 69
15.3.1 Sélection vs construction 69
15.3.2 L'exemple des *JEP* : 69
15.4 Projection sur des variétés 70

16 Les systèmes étagés : Squelettes, actoms... **71**

III Optimiser les votes en segmentation supervisée par élection **74**

17 Intérêt des méthodes d'élection en segmentation **74**
17.1 Besoins en ressource machine 74
17.2 Taxonomie des méthodes d'élection 75

18 Optimiser le modèle de formes implicites **76**
18.1 Transformer *ISM* 76
18.1.1 Remplacer le seuillage par une élection 77
18.1.2 Supprimer le lissage 77
18.1.3 Mise en équation de la méthode d'élection en segmentation 78
18.2 Optimisation naïve de la méthode d'élection 78
18.3 Forcer une cohérence temporelle 79
18.4 Forcer une régularité 80
18.5 Optimisation de la méthode d'élection en segmentation : *DOHT* 80

19 Évaluation du *DOHT* **81**
19.1 État de l'art des méthodes de type *ISM* 82
19.2 Évaluation de *DOHT* pour la segmentation de comportement d'abeilles 82
19.2.1 Traitement du signal 83
19.2.2 Résultats 84
19.3 Évaluation sur des données de synthèse 85
19.3.1 Premier cas d'école 85
19.3.2 Deuxième cas d'école 86
19.3.3 Résultats sur les deux cas d'école 87

20 Approximation du *DOHT* **87**
20.1 Reformulation de *DOHT* 88
20.2 Décomposition pyramidale dense 89
20.3 Accélération de l'apprentissage 91
20.3.1 Principe 91
20.3.2 Expériences 91
20.3.3 Résultats 92

IV Segmenter des actions à partir de *primitives squelettes* **93**

21 Choisir le cœur du traitement du signal **93**
21.1 L'adéquation des primitives aux applications visées 93
21.2 Intérêt de primitives *squelettes* pour la segmentation d'actions 94
21.3 Traitement des vidéos *squelettes* 95

22 Évaluations de *primitives squelettes* **95**
22.1 Jeux de données 95
22.2 Normalisation 99
22.2.1 Taxonomie des normalisations 100
22.2.2 Descriptions des normalisations 100
22.2.3 Évaluation des normalisations 101
22.3 Positions des articulations : individuelles ou combinées 102
22.4 Positions des articulations ou sous trajectoires 103
22.5 Différentes primitives temporelles 104
22.6 Différentes implémentations de quantification 104
22.7 Descripteur de trajectoires et de vidéos 105
22.8 Tableau récapitulatif et résultats 106

23 Comparaison à l'état de l'art **106**

V Augmenter la capacité de déploiement du système de segmentation d'actions **109**

24 Évaluation des primitives *squelettes* dans des vidéos corrompues **109**
24.1 L'état de l'art de l'impact du bruit sur les primitives *squelette* 110
24.2 Protocole de mesure de l'influence du bruit 111
24.3 Résultats : mesure de l'impact des données manquantes . . . 113

25 Dictionnaires parcimonieux **115**
25.1 Sélection a priori 115
25.2 Dictionnaire initial 116
25.3 Sélection 117
25.4 Extraction de super mots 118
25.5 Bilan et impact sur les ressources machines 118

26 Extraction creuse **119**
26.1 Échantillonnage régulier 119
26.2 Sous extractions considérées 119
26.3 Performances des différentes extractions 120
26.4 Bilan et impact sur les ressources machines 122

Conclusion générale et perspectives **123**

Références **125**

Annexe **132**

Introduction générale

De la détection de visage à la conduite automatique

Ces dernières années ont vu naître des systèmes, destinés à une commercialisation à large échelle, fondés sur l'apprentissage statistique et la vision par ordinateur. En 2011, Microsoft a vendu en quelques mois des millions de *kinect*, caméras capables, par apprentissage statistique, d'extraire le *squelette* de la personne observée dans un contexte de jeu vidéo, supprimant tout contact matériel entre le joueur et la console. En 2012, *Google* a activé un processus d'annotation automatique des images de *Google+* à partir de leurs contenus (et non à partir des annotations émises par les auteurs des images). En 2013, *Facebook* a créé un centre de recherche en intelligence artificielle avec comme objectif affiché d'implémenter un système de type *réseau de neurones* à grande échelle afin d'améliorer l'offre de service de *Facebook*. Fin 2014, *mobileeye* (système d'aide à la conduite) espère être disponible sur 160 véhicules de 18 constructeurs automobiles différents. La *voiture Google* (voiture sans conducteur), autorisée à rouler de façon expérimentale sur les routes du Nevada depuis 2011, entend, de son coté, révolutionner les modes de déplacement dans les années à venir.

Avant cette récente explosion technologique, les principaux exemples de tels systèmes étaient les moteurs de recherche classiques et la détection de visage dans les appareils photo. Mais, ces exemples ne peuvent pas être considérés comme similaires aux développements plus récents. Par exemple, l'indexation automatique de contenu multimédia (comme celle de *Google+*) introduit des problèmes beaucoup plus aigus que l'analyse de texte classique. D'une part la taille totale de données *html* d'internet est estimée à seulement 50 000 000 000 ko contre 2 000 000 000 000 ko pour la taille totale des images et vidéos. D'autre part, l'information *html* est déjà symbolique à la différence de l'information contenue dans une image. L'indexation automatique de contenu multimédia doit donc gérer 40 fois plus de données que l'indexation de site *web* (et l'écart devrait croître dans les années à venir) ainsi que des données plus complexes. De même, dans la *kinect*, *mobileeye* et la *voiture Google*, l'apprentissage statistique n'est plus un argument de vente comme dans le cas de l'appareil photo mais bien le cœur même du produit. Ainsi, ces nouveaux systèmes n'ont pas réellement d'équivalents antérieurs.

L'apparition de ces systèmes est permise par la déconnexion entre intelligence artificielle et apprentissage statistique. Entremêlées, ces deux notions furent un sujet très à la mode dans les années 50 mais délaissées durant les années 80 après qu'il ait été démontré que les systèmes d'apprentissage statistique simples ne permettraient pas d'obtenir des systèmes d'intelligence artificielle [67]. Dédiés à des tâches réalistes, ce sont pourtant ces mêmes systèmes simples qui permettent, aujourd'hui, de répondre aux besoins listés

plus haut : extraction de *squelette*, indexation automatique de document, et détection de piéton.

C'est dans ce contexte que cette thèse s'intéresse spécifiquement à l'application d'apprentissage statistique pour la reconnaissance d'actions dans des vidéos.

La reconnaissance d'actions

Le besoin en reconnaissance d'actions est en pleine explosion. Il se décompose en trois parties principales : l'indexation de vidéos, l'interaction homme machine par mouvement, et la vidéo surveillance. L'indexation de vidéos touche le grand public à travers la construction de moteurs de recherche capables de chercher une vidéo pertinente au regard d'une requête dans des ensembles de vidéos de très grandes tailles (comme *YouTube*). L'interaction homme machine par mouvement touche aussi le grand public, principalement à travers le jeu vidéo sans manette.

La vidéo surveillance a, quant à elle, une position paradoxale vis à vis du grand public. Elle renvoie à quelque chose de négatif. Mais, pourtant, le nombre de caméra de surveillance de la voie publique a triplé en France entre 2007 à 2011. Cette ambiguïté de la demande en vidéo surveillance existe pour la vidéo surveillance classique : détection d'infraction et/ou détection de comportement violent. Elle devrait aussi apparaître pour d'autres types d'applications parfois qualifiées de vidéo protection. Par exemple, le vieillissement de la population laisse prévoir une forte demande pour des systèmes de surveillance pour le maintien à domicile de personnes fragiles. Plutôt que de faire accompagner la personne fragile par un personnel médical présent en permanence, on peut imaginer installer des caméras chez elle et les relier à un système qui utiliserait de l'apprentissage statistique pour garantir l'appel automatique du personnel médical quand leur présence est utile. Cela permettrait ainsi à ces personnes de rester autonomes. Mais surtout, de tels systèmes seront vraisemblablement plus faciles à mettre en place qu'un accompagnement constant des personnes fragiles par un personnel médical (même si cela répond sans doute plus aux souhaits des personnes). Un tel système devrait détecter les signes engageant le pronostic vital de la personne (chute, étouffement, crise cardiaque, perte de coordination) pour déclencher des alertes d'urgences. Il devrait aussi détecter des tendances à long terme dans l'activité globale de la personne (perte de dynamisme, augmentation de la segmentation des activités) dans le cadre d'un protocole de soin. L'acceptation de tels systèmes n'est pas évidente mais elle parait possible si le système est entièrement automatisé (personne ne visionne les vidéos).

Cette thèse vise, en lointaine perspective, ce contexte applicatif de surveillance pour le maintien à domicile de personnes fragiles. Notamment, le travail de cette thèse se concentre sur le traitement de certaines caractéris-

tiques avérées ou supposées de ce contexte applicatif.

Spécificités prises en compte

Le travail de cette thèse se concentre principalement sur le traitement des trois points suivants : la nécessité de prendre en compte un important contexte temporel pour reconnaître les actions, la possibilité de travailler sur des vidéos dont les modalités d'acquisition sont maîtrisées et la nécessité de traiter des flux vidéo continus en maintenant une latence de traitement constante.

Ces trois caractéristiques sont au cœur des choix structurant la thèse.

La prise en compte d'un important contexte temporel nécessiterait au mieux de récolter de trop grandes bases d'apprentissage (via l'utilisation de techniques standards d'apprentissage statistique). Cela justifie de traiter les données à l'aide de techniques moins standards. C'est pourquoi la *reconnaissance d'actions* devient de la *segmentation supervisée d'actions*.

La possibilité de travailler sur des données dont les modalités d'acquisition sont maîtrisées permet de ne traiter que des vidéos de bonne qualité, et donc, d'en extraire *des primitives haut niveau*, trop fragiles pour être extraites de vidéos quelconques, mais apportant plus d'information que des primitives proches des données brutes.

Enfin, la nécessité de travailler sur des *flux vidéos* impose que le traitement puisse garantir une latence constance, et donc, il soit peu exigeant en terme de ressources machines.

Réponses apportées à ces spécificités

Pour répondre à ces spécificités, la thèse a conduit à trois contributions. La première contribution porte sur la conception d'un algorithme de segmentation. Cet algorithme peut être vu comme un algorithme de classification classique de la littérature auquel est ajouté des contraintes supplémentaires. Ces contraintes supplémentaires permettent de rendre plus robuste l'algorithme d'origine à la prise en compte d'un important contexte temporel (première spécificité).

Cet algorithme est ensuite appliqué à la segmentation d'actions à travers un système. Ce système utilise la disponibilité de primitives haut niveau (deuxième spécificité) pour faciliter l'application de l'algorithme de segmentation. Pour cela, il considère un certain nombre de primitives dédiées dont la présentation est une seconde contribution de cette thèse.

Enfin, compte tenu de la nécessité de travailler sur des flux vidéos (troisième spécificité), un travail montrant la flexibilité du système vis à vis des ressources machines disponibles est présentée, et forme la troisième contribution de cette thèse.

Plan du manuscrit

Le manuscrit est structuré de la façon suivante. Les parties 1 et 2 portent sur la littérature scientifique dans laquelle s'inscrit cette thèse. La partie 1 présente la littérature d'apprentissage statistique, et notamment, précise la spécificité de la segmentation par rapport à la reconnaissance. La partie 2 porte sur la littérature générale de reconnaissance d'actions. Les contributions apportées par la thèse font l'objet des parties 3, 4 et 5. La partie 3 décrit la conception d'un algorithme de segmentation adapté à la nécessité de prendre en compte un important contexte temporel. La partie 4 présente l'utilisation de cet algorithme pour la segmentation d'actions, et précise les primitives haut niveau considérées. Enfin, la partie 5 porte sur des travaux de maturation du système, et notamment, sur des techniques permettant d'adapter le système aux ressources machines disponibles.

Cette thèse s'appuie sur un certain nombre de résultats mathématiques nouveaux. Les démonstrations de ces résultats, étant secondaires du point de vue de ce manuscrit, sont ajoutées aux annexes.

Liste des notations

Ce manuscrit tente d'utiliser les notations standards de la littérature pour désigner les objets étudiés. Ces notations sont introduites au fur et à mesure de leur utilisation. Elles sont aussi regroupées ci dessous à toutes fins utiles.

Quelques notations mathématiques générales :

- le cardinal d'un ensemble fini, la valeur absolue, la dimension d'un espace vectoriel, et la longueur d'une suite fini : $|.|$
- une norme : $\|.\|$
- le produit scalaire : $\langle ., . \rangle$ ou la simple juxtaposition
- la restriction d'une fonction f à un sous ensemble $\mathcal{D}$: $f_{|\mathcal{D}}$
- les intervalles discrets : $[., .]$ - cette notation n'est pas utilisée pour les intervalles continus

Les notations d'apprentissage statistique :

- une donnée : x
- l'ensemble des données : X
- un concept, une classe : y
- le concept associé à une donnée x : $y^*(x)$
- l'ensemble des classes : Y
- une probabilité : P
- une base d'apprentissage : B
- l'ensemble des éléments de classe y de la base d'apprentissage B : B_y
- une mesure d'erreur : e
- un algorithme d'apprentissage : L
- l'indice n est utilisé pour désigner les données, l'indice d pour les composantes d'une donnée
- la dimension de l'espace des données : D
- le nombre d'exemples d'apprentissage : N
- le vecteur normal à un hyperplan : w
- un mot : m
- un dictionnaire (ou vocabulaire) de mots : M
- un vote : v
- un score : S
- la fonction de pénalisation standard : $h(u) = \frac{1}{2}(|u| - u)$

Les notations associées à la segmentation temporelle :

- un temps : t
- le concept associé à l'instant t dans la donnée x : $y^*(x, t)$
- une durée : T
- une latence (une durée minimale pour obtenir l'information nécessaire) : $\triangleright$
- un décalage temporel (différence entre deux temps) : $\triangle$
- un intervalle discret : I
- l'ensemble des sous intervalles discret de $[a, b]$ contenant 0 : $I([a, b])$

L'état de l'art

Première partie
L'apprentissage statistique

1 Définition de la classification supervisée

1.1 Description du problème de classification supervisée

1.1.1 Les objets manipulés

Le problème de la classification est celui de l'estimation d'une fonction y^*, d'un ensemble de données X vers un ensemble de concepts Y, à partir d'exemples $(x, y^*(x)) \in X \times Y$, c'est à dire de données $x \in X$ pour lesquelles le concept $y^*(x)$ est connu.

Plus précisément, un algorithme de prédiction (appelé aussi critère de décision) est un algorithme qui prend en entrée une donnée $x \in X$ et qui prédit un concept $y(x) \in Y$. Un ensemble d'apprentissage B est un ensemble d'exemples (donc un ensemble de couples $(x, y^*(x)) \in X \times Y$). Un algorithme d'apprentissage L est un algorithme qui prend en entrée un ensemble d'apprentissage B (ou base d'apprentissage) et qui produit en sortie un algorithme de prédiction L_B.

Étant donné un exemple $(x, y^*(x))$, l'algorithme de prédiction L_B fait une erreur sur x si $L_B(x) \neq y^*(x)$. On introduit une fonction e pour exprimer la présence ou non d'une erreur sur un exemple : $e((x, y^*(x)), L_B) = 1$ si $L_B(x) \neq y^*(x)$ et 0 sinon. On note également e la moyenne de l'erreur sur tous les exemples d'un ensemble de test B_{test} (un ensemble d'exemples généralement différent de B) : $e(B_{test}, L_B) = \frac{1}{|B_{test}|} \sum_{x \in B_{test}} e((x, y^*(x)), L_B)$ où $|B_{test}|$ désigne le cardinal de B_{test}.

1.1.2 L'introduction de probabilités

Pour modéliser le lien entre l'ensemble de test et l'ensemble d'apprentissage, on suppose que les exemples des deux ensembles sont tirés selon une même probabilité P. Dès lors, plutôt que de calculer $e(B_{test}, L_B)$, on peut directement calculer l'espérance de l'erreur vis à vis de B_{test} c'est à dire l'erreur moyenne sur X entier : $e((P, y^*), L_B) = \int_X e((x, y^*(x)), L_B) \, \mathrm{d}_P x$ où d_P est la mesure associée à P (par exemple, si P possède une densité de probabilité f alors $\mathrm{d}_P x = f(x) \, \mathrm{d}x$).

De même, on peut introduire l'espérance de l'erreur vis à vis de B_{test} et de B. On la note $e((P, y^*), L, N) = \int_{\text{bases taille N}} e((P, y^*), L_B) \, \mathrm{d}_P B$. Ici, on s'autorise à noter d_P la mesure associée à P étendue aux tuples de points (ainsi si B contient deux points x_1, x_2, $\mathrm{d}_P B = f(x_1) f(x_2) \, \mathrm{d}x_1 \mathrm{d}x_2$).

Remarquons que cette dernière erreur est un objet intéressant car elle ne

fait apparaître que les objets intrinsèques au problème c'est à dire P, y^*, L, N et non pas des objets intermédiaire comme B et B_{test} représentant des tirages particuliers.

1.1.3 L'objectif de l'apprentissage statistique

La question de l'apprentissage statistique est de savoir s'il existe L tel que pour tous P et y^*, on ait une formule de la forme $e\left(\left(P, y^*\right), L, N\right) \leq \frac{1}{N}$ (plus généralement, il faudrait remplacer $\frac{1}{N}$ par une quantité ϵ_N). Cette question équivaut à l'existence d'une technique permettant, étant donné L, d'estimer pour tous P et y^*, une borne sur l'erreur $e\left(\left(P, y^*\right), L, N\right)$ à partir de B (qu'on peut éventuellement séparer en deux pour simuler une base de test B_{test}). En effet, si on peut borner $e\left(\left(P, y^*\right), L, N\right)$ connaissant B, il suffit de considérer l'algorithme qui évalue la borne de tous les algorithmes L jusqu'à en trouver un tel que de façon sure $e\left(\left(P, y^*\right), L, N\right) \leq \frac{1}{N}$.

1.2 *No free lunch theorem* : la non existence d'une solution

Malheureusement, la réponse à cette question est *grosso modo* négative.

Introduisons la notion d'erreur d'apprentissage correspondant à l'erreur si la base d'apprentissage est la base de test. Cette erreur est $e\left(B, L_B\right)$ avec les définitions de e précédemment introduites. Introduisons de même la notion d'erreur hors apprentissage : c'est l'erreur hors des points de la base d'apprentissage (cela suppose une distribution discrète). Notons son espérance : $e'_N\left(\left(P, y^*\right), L\right)$. Cette espérance s'écrit formellement $e'_N\left(\left(P, y^*\right), L\right) = \int_{\text{bases taille N}} \int_{\text{bases tailles N disjointes de B}} e\left(B_{test}, L_B\right) \mathrm{d}_P B_{test} \mathrm{d}_P B$.

Alors, un des plus importants résultats du domaine [97] est que, quelque soit l'algorithme L, l'erreur de L hors de la base d'apprentissage (c'est à dire $e'_N\left(\left(P, y^*\right), L\right)$) moyennée sur toutes les situations (i.e. (P, y^*)) est constante.

Un corollaire de ce résultat est que pour tous les couples d'algorithmes (L, L') il existe P tel que L a une erreur $e'_N\left(\left(P, y^*\right), L\right)$ plus faible que L', et, le nombre de probabilités telles que L est meilleur que L' hors de la base d'apprentissage est le même que le nombre de probabilité pour lesquelles l'inverse est vrai [97]. Ce corollaire reste vrai même si un des deux algorithmes consiste à prédire une classe aléatoirement ou constante. Il reste également vrai même si un des deux algorithmes cherche parmi une banque d'algorithmes à utiliser celui qui parait plus adapté vis à vis de la base d'apprentissage alors que l'autre cherche à utiliser le moins adapté.

Ainsi, non seulement il n'y a pas d'algorithme solution (si on veut maintenir l'universalité vis à vis de P et y^*), mais surtout, cela signifie que même si $e\left(B_{test}, L_B\right)$ vaut 0 (pour un couple B, B_{test} de taille N), cela ne donne *aucune* information sur $e'_N\left(\left(P, y^*\right), L\right)$. L'inverse aboutirait à une contradiction : si à partir d'une mesure d'erreur sur un nombre fini d'exemples, on pouvait de façon déterministe prédire une borne sur l'erreur globale, alors

il suffirait d'évaluer cette borne pour tous les algorithmes et d'en retenir un dont la borne est suffisamment faible. Cela contredirait le *No free lunch theorem*.

La seule alternative à ce théorème est de relâcher la condition de vouloir traiter tous les problèmes (i.e. tous les couples P et y^*). Cela est d'ailleurs acceptable en remarquant que les *problèmes qui répondent à des besoins* (de la société) ne forment qu'une partie de l'ensemble des problèmes. Mais, en l'absence d'une description explicite de cet ensemble des probabilités, la reconnaissance par ordinateur n'est pas un problème mathématique. Aujourd'hui, la reconnaissance par ordinateur est, en un sens, une science expérimentale avec tous les biais que cela entraîne dont les lois inconnues sont les familles de probabilités associées aux *problèmes qui répondent à des besoins*.

Cependant, la reconnaissance par ordinateur est surtout un ensemble de techniques spécifiques car la discipline est très jeune. Cela explique la difficulté de répondre, aujourd'hui, clairement à la question du choix d'un algorithme pour un problème donné et surtout le manque de théories unifiant les observations.

1.3 Lignes directrices et taxonomie des contextes

L'absence de cadre formel pour l'apprentissage statistique ne signifie cependant pas que le domaine n'est pas structuré : il existe des grandes lignes directrices dans le domaine sur lesquelles est construite cette présentation. Notamment, s'il n'existe aucun algorithme tel que pour tous P et y^*, on ait une formule de la forme $e\left(\left(P, y^*\right), L, N\right) \leq \frac{1}{N}$, il existe néanmoins des algorithmes tels que pour tous P et y^*, $e\left(\left(P, y^*\right), L, N\right)$ tendent vers 0 quand N tend vers l'infini [28]. Ces algorithmes dits asymptotiquement universels sont l'objet de la section 2.

De plus, même s'il ne peut exister de technique pour borner l'erreur de façon déterministe, il existe des bornes probabilistes dont l'exemple le plus classique [89] est l'objet de la section 3.

Simultanément il existe différents contextes d'apprentissage résumés dans la figure 1 : les sections 2 et 3 s'intéressent à la classification binaire de points i.e. $X = \mathbb{R}^D$ et $Y = \{-1, 1\}$. Mais ce contexte est réducteur. Notamment, les sections 4 à 6 portent sur les problèmes où les données ne sont plus des points i.e. $X \neq \mathbb{R}^D$ (sections 5 et 6) et Y est fini mais de cardinal strictement supérieur à 2 (section 4). La section 6 introduit, de plus, un exemple de classes de probabilités pour lesquelles l'apprentissage statistique se ramène à un simple apprentissage unidimensionnel. Les sections 7,8,9 s'intéressent au problème de la segmentation qui y est défini.

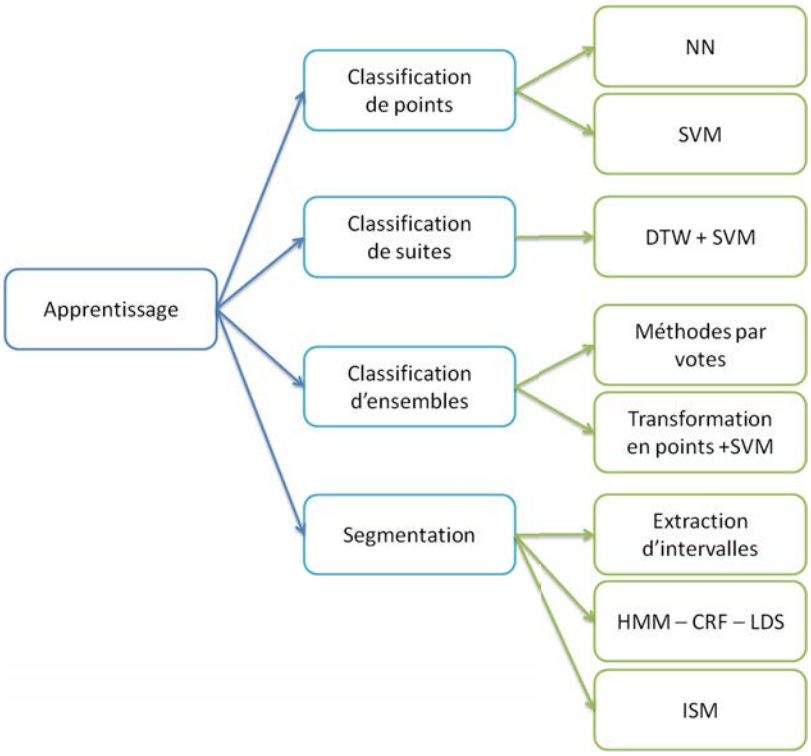


FIGURE 1 – Les principaux contextes d'apprentissage statistique

La classification binaire de points

2 Algorithmes asymptotiquement universels

2.1 Plus proche voisin

Historiquement, le premier algorithme asymptotiquement universel dans le cas $X = \mathbb{R}^D$ est l'algorithme du plus proche voisin (*NN* pour *Nearest Neighbor*). Cet algorithme consiste à décider que x est de classe 1 si x est *plus proche* d'un des points de la classe 1 que de tous les points de la classe -1 (et inversement). Mathématiquement, cet algorithme s'écrit :

$$y\left(x\right) = y^* \left(\arg\min_{x' \in B} \left\| x - x' \right\| \right)$$

Il est asymptotiquement universel pour toutes les probabilités P et fonctions y^* [28].

Il est indispensable ici que les données soient des points, à la fois fonctionnellement et pour que la propriété soit vraie. Dit autrement, il est indispensable pour utiliser la distance classique que les exemples soient des points c'est à dire $X = \mathbb{R}^D$. Mais de plus, si on étendait la distance pour permettre par exemple d'accepter des valeurs manquantes ($X = \left(\mathbb{R} \cup \{\emptyset\}\right)^D$) alors le théorème ne s'applique plus.

2.2 *K* plus proches voisins

Il existe une variante, qui est en réalité le problème générique, du problème de la classification où l'on ne considère non pas une probabilité P et une fonction y^* mais uniquement une probabilité P sur $\mathbb{R}^D \times \{-1, 1\}$ ($X \times Y$ dans le cas général). Ainsi on tire directement des couples (x, y) selon P et un même point x est associé aux différents $y \in Y$ avec des probabilités $P\left(y|x\right)$. Or, dans ce cas, l'erreur minimale atteignable n'est plus nécessairement nulle. Il suffit de penser au cas où x et y sont indépendants, on ne peut pas faire mieux qu'une erreur de $\frac{|Y|-1}{|Y|}$. Mais la question se pose de savoir si on peut converger asymptotiquement vers l'erreur minimale même si elle n'est plus nulle et *NN* n'est plus une réponse car il n'est plus asymptotiquement universel dans ce cas.

L'algorithme des K plus proches voisins (*KNN*) consiste à choisir la classe majoritaire parmi les K plus proches voisins. D'un point de vue algorithmique, cet algorithme peut être implémenté simplement en stockant les K plus proches voisins lors d'un parcours des exemples :

```
//xtest : le point dont on veut prédire la classe
//xapp : les exemples d'apprentissage
//y : les classes correspondantes aux exemples
//K : nombre de voisins considérés
```

```
//pred : classe prédite pour xtest
pred = predictionNN(xtest,xapp,y,K)
   nn une pile de priorité
   pour n de 1 à N
      nn.ajouter xapp[n] avec priorité ||xtest-xapp[n]||
   si n>K
      dépiler l'élément le plus prioritaire de nn
   y=0
   pour k de 1 à K
      y += y[nn[k]]
   retourner le signe de y
```

Si K est adapté au nombre d'exemples d'apprentissage N (typiquement $K = \log(N)$, l'algorithme *KNN* est asymptotiquement universel pour le problème générique [28, 33]. Plus précisément, *KNN* est asymptotiquement universel si et seulement si $K(N) \underset{N\to\infty}{\to} \infty$ et $\frac{K(N)}{N} \underset{N\to\infty}{\to} 0$.

2.3 Estimation de densité

KNN apprend implicitement la probabilité P sur $X \times Y$ selon laquelle sont tirés les exemples. Il existe aussi des algorithmes qui cherchent à apprendre explicitement cette probabilité. Le plus classique de ces algorithmes est la méthode *des fenêtres de Parzen*. Soit ϕ une fonction noyau de $\mathbb{R}$ (cet algorithme est généralisable à $\mathbb{R}^D$) i.e. $\phi \geq 0$ et $\int_{\mathbb{R}} \phi(u)\,du = 1$, étant donné $\alpha_1, ..., \alpha_N \in \mathbb{R}$ tiré selon une densité de probabilité P (on note indistinctement P et sa densité), on construit $\psi_{N,\delta}(u) = \sum_{n=1}^{N} \frac{1}{N\delta}\phi\left(\frac{u-\alpha_n}{\delta}\right)$. Pour toutes fonctions noyaux ϕ et pour toutes fonctions $\delta(N)$ telles que $\delta(N) \underset{N\to\infty}{\to} 0$ mais $\sqrt{N}\delta(N) \underset{N\to\infty}{\to} \infty$, il est démontré [68] que $\psi_{N,\delta(N)}$ converge uniformément vers P quand N tend vers l'infini.

Cette méthode peut directement être utilisée pour capturer $P(x, y)$ et donc décider selon le critère $\arg\max_y (P(y|x))$ qui atteint l'erreur minimale.

2.4 Malédiction de la dimension

KNN est en un sens l'algorithme d'apprentissage idéal : il est asymptotiquement universel, il n'a aucun paramètre (si on fixe K au logarithme du nombre d'exemples), il traite naturellement un contexte à plus de deux classes (bien que présenté ici dans le cas binaire) et les ressources machines nécessaires à son utilisation n'augmentent que linéairement (à des facteurs logarithmiques près) même dans une implémentation naïve.

Cependant, il apparaît dans la littérature que les méthodes de type *NN* convergent trop lentement : [10, 52] estiment que le nombre d'exemples né-

cessaires à *NN* pour obtenir une erreur fixée croit exponentiellement avec la dimension de l'espace des points. Ce phénomène dit de malédiction de la dimension dont on attribue la première observation à [47] est appuyé à la fois expérimentalement et par des arguments de combinatoire. L'argument combinatoire généralement invoqué est que le nombre de points nécessaires pour avoir un plus proche voisin à moins d'une distance fixée augmente exponentiellement avec la dimension de l'espace.

De plus, si un concept dispose de certains invariants vis à vis des données, *KNN* finira asymptotiquement par atteindre l'erreur minimale mais sans *comprendre* ces invariants. Par exemple, considérons des points du plan qui contiennent leur classe dans leur première composante et un bruit uniforme d'amplitude 10000 dans l'autre composante (c'est à dire $X = \{-1,1\} \times \{u \in \mathbb{R}, -10000 \leq u \leq 10000\}$ avec $y^*(x) = x_1$ et x_2 tiré uniformément), *KNN* ne finira par atteindre l'erreur minimale qu'à l'infini alors que ce problème est trivial.

Pour ces raisons *KNN* n'est pas souvent pertinent pour traiter les *problèmes qui répondent à des besoins.*

3 La minimisation du risque structurel

Un des plus grand théorème d'apprentissage statistique a été introduit dans [89]. Il porte sur une borne probabiliste sur l'espérance de l'erreur.

3.1 Borne probabiliste sur l'erreur

Cette borne introduit la notion de dimension de *Vapnik Chervonenkis* (*VC*). Soit L un algorithme. Soit $\mathcal{L}$ l'ensemble des critères qui peuvent résulter de l'application de L à une base B. Soit $x'_1, ..., x'_\Omega$ tels que pour tous $y'_1, ..., y'_\Omega$, il existe $l \in \mathcal{L}$ tel que pour tous $\omega \in \{1, ..., \Omega\}$, $l(x'_\omega) = y'_\omega$, alors la dimension *VC* de L est supérieure à Ω. La dimension *VC* de L est soit infinie soit correspond à la valeur maximale de ces nombres Ω. Typiquement la dimension *VC* de *NN* est infinie. Par contre, la dimension *VC* de l'algorithme renvoyant un critère constant est 1.

Il est démontré dans [89] que pour tous les algorithmes L, si B est une base de taille N tirée selon P, et pour tous $0 < \eta < 1$, alors avec une probabilité $1 - \eta$, l'inégalité suivante est vérifiée :

$$e(P, L_B) \leq e(B, L_B) + \sqrt{\frac{\left(VC(L)\left(\log\left(\frac{2N}{VC(L)}\right) + 1\right) - \log\left(\frac{\eta}{4}\right)\right)}{N}}$$

Remarquons que cette borne n'est pas incompatible avec le *No free lunch theorem* notamment parce qu'elle porte sur l'espérance de l'erreur totale c'est à dire qu'elle porte autant sur les nouveaux points que sur ceux de la base

d'apprentissage. À l'opposé, le *No free lunch theorem* ne porte que sur l'erreur hors de la base d'apprentissage. Ainsi, on ne peut pas dire : un algorithme de faible dimension VC qui fait peu d'erreur d'apprentissage fait probablement peu d'erreur sur la base de test disjointe de la base d'entraînement. Mais on peut bien dire qu'un algorithme de faible dimension VC qui fait peu d'erreur d'apprentissage fait probablement peu d'erreur sur l'ensemble des points.

Mais surtout, elle n'est pas incompatible avec le *No free lunch theorem* car c'est une borne probabiliste : minimiser le terme de droite de cette borne n'a aucun sens car il suffit de prendre un algorithme qui se contente de reconnaître exactement et seulement B (ainsi cette borne ne permet pas de répondre à la question de l'apprentissage statistique). Dit autrement, la notion de *probabilité* dans l'expression *un algorithme de faible dimension VC qui fait peu d'erreur d'apprentissage fait probablement peu d'erreur sur l'ensemble des points* porte justement sur le tirage de la base d'apprentissage. Ainsi, il faut comprendre la borne par *si l'espérance de l'erreur d'apprentissage est faible et que la VC est faible alors l'espérance de l'erreur est faible* (ainsi, il n'y a plus de *probabilité* dans cette phrase). Cependant, il n'est pas possible de calculer l'espérance de l'erreur d'apprentissage sans connaître P.

L'intérêt de cette borne est de démontrer que si un algorithme a une VC finie, l'écart entre l'erreur d'apprentissage et l'erreur totale diminue rapidement (comme $\frac{VC(L)}{\sqrt{N}}$ en fonction du nombre N d'exemples d'apprentissage). Dit autrement, l'erreur est *stable* et il suffit d'avoir un nombre d'exemples proportionnel à la dimension VC pour maintenir un écart constant. Forcément, il n'existe que certaines probabilités pour lesquelles l'erreur converge vers l'erreur minimale mais il n'est pas exclus que ce soit le cas pour les *problèmes qui répondent à des besoins.* D'ailleurs, les algorithmes de *VC minimale* se sont révélés très pertinents pour beaucoup de problèmes.

La dimension VC est *minimale* (vis à vis de la dimension des points) par la recherche d'une frontière *linéaire* i.e. sous la forme d'un hyperplan séparant les exemples des deux classes. Dans le cas 2D cela conduit à une frontière sous la forme d'une ligne (figure 2).

3.2 *SVM* : la maximisation de la marge

Un tel hyperplan peut se paramétrer par un couple $b \in \mathbb{R}$ et $w \in \mathbb{U}_D$ où $\mathbb{U}_D$ est l'ensemble des vecteurs unitaires de dimension D (i.e. $ww = 1$).

La distance algébrique ou *marge* d'un exemple $x, y^*(x)$ au plan w, b est définie comme $y^*(x) \times (\langle w, x \rangle + b)$. La marge d'un ensemble de points à un plan est le minimum de la marge des points de l'ensemble (appelé aussi nuage) au plan. S'il existe un plan ayant une marge positive aux deux ensembles de points, ceux-ci sont dits *séparables.*

Dans ce cas, l'hyperplan qui maximise la marge introduit dans [15] est justifié spécifiquement vis à vis des autres hyperplans (par exemple dans [88]). Il est appelé *SVM* (pour *Support Vector Machine*).

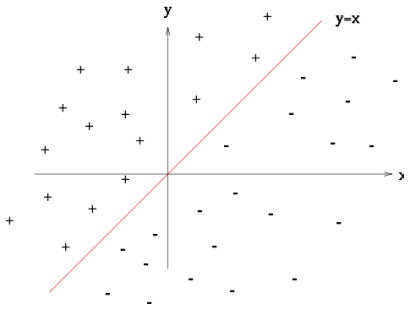


FIGURE 2 – algorithme d'apprentissage à faible dimension VC

On note $h(u) = \frac{1}{2}(|u| - u)$ la fonction de perte standard : h pénalise les nombres négatifs linéairement par rapport à leur valeur et ne pénalise pas les positifs. Il est démontré dans [27] qu'il existe une valeur C' telle que pour tous les $C \geq C'$, le plan SVM est solution de :

$$\underset{w \in \mathbb{R}^D, b \in \mathbb{R}}{\arg\min} \left(\|w\|_2^2 + C \sum_{n=1}^{N} h\left(y^*(x_n) \times (wx_n + b) - 1\right) \right) \tag{1}$$

Remarquons qu'avec cet algorithme, il y a une vraie différence entre l'algorithme d'apprentissage et celui de prédiction (dit autrement entre *apprentissage* et *test*) : l'algorithme d'apprentissage calcule un hyperplan, l'algorithme de prédiction ne calcule que des produits scalaires. Ainsi, le calcul de la classe prédite d'un point nécessite seulement très peu de ressource machine (une quantité constante vis à vis de la taille de la base d'apprentissage) ce qui n'est par exemple pas le cas avec NN.

Dans le cas où les nuages ne sont pas séparables, la formulation (1) représente une quantité approximant l'erreur d'apprentissage [27]. Ainsi, même en présence de nuages non séparables, le SVM conserve un sens à travers la minimisation de (1). La différence est que $\sum_{n=1}^{N} h\left(y^*(x_n) \times (wx_n + b) - 1\right)$ n'est plus égale à 0. D'ailleurs, la littérature s'autorise à prendre C quelconque (ce que l'on s'autorise aussi dans cette thèse) car il est convenu que cela stabilise l'erreur. C devient un paramètre qui influe sur l'équilibre entre une approximation de l'erreur d'apprentissage et une mesure de la régularité de la solution.

En lissant légèrement la fonction h, on peut se ramener à un problème dérivable et convexe. La solution de ce problème peut alors être obtenue par une descente de gradient comme dans [23].

Cependant, la solution est généralement obtenue par résolution du problème dual $\underset{\alpha \in \mathbb{R}^N}{\arg\max} \left(\sum_{n=1}^{N} \alpha - \sum_{i,j=1}^{N} y^*(x_i)\, y^*(x_j)\, \alpha_i \alpha_j \phi(x_i, y_j) \right)$ en respectant

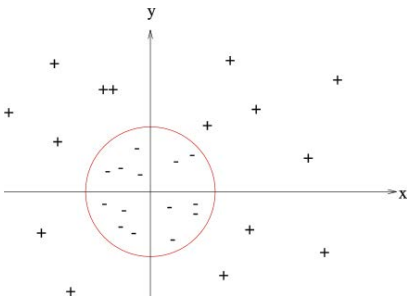


FIGURE 3 – Situation où deux nuages (+ et -) ne sont pas linéairement séparables

les contraintes que $\sum_{n=1}^{N} y^*(x_n)\,\alpha_n = 0$, que $0 \leq \alpha_n \leq C$, et où $\phi(x_i, x_j)$ est la fonction noyau du produit scalaire i.e. $\phi(x_i, x_j) = \langle x_i, x_j \rangle$ (Utiliser une autre fonction noyau correspond à chercher une surface séparatrice non linéaire). Cette résolution peut s'effectuer par descente de gradient [16] ou par des techniques dédiées comme [48].

3.3 Généralisation aux frontières non linéaires

Les hyperplans ne permettent pas de séparer certains types de nuages de points. La figure 3 illustre ce phénomène.

De tels nuages sont cependant parfois linéairement séparables dans un espace plus grand. Soit ψ une application de $\mathbb{R}^D$ vers $\mathbb{R}^{D'}$. Il est possible que $\psi(x_1), ..., \psi(x_N)$ soit séparable même si $x_1, ..., x_N$ ne l'est pas. C'est le cas dans la figure 3 avec $\psi((u, v)) = \left(u, v, \sqrt{u^2 + v^2}\right)$.

Dans le cas où $D' \gg D$, l'application de ψ est problématique. Cependant, la formulation duale du *SVM* ne fait apparaître cette dimension qu'à travers la fonction $\phi(v, w)$. Notons $\Phi(v, w) = \phi(\psi(v), \psi(w))$. La connaissance de Φ est suffisante pour pouvoir exprimer le dual. Cette astuce des noyaux [15] permet même l'utilisation d'une projection vers un espace de dimension infinie.

Le *SVM* linéaire et cette technique des noyaux est très importante en pratique car les *SVM* forment la plus grande partie de l'état de l'art des méthodes de classification de points et ont permis d'obtenir des avancées significatives sur de nombreux problèmes dont [17, 64].

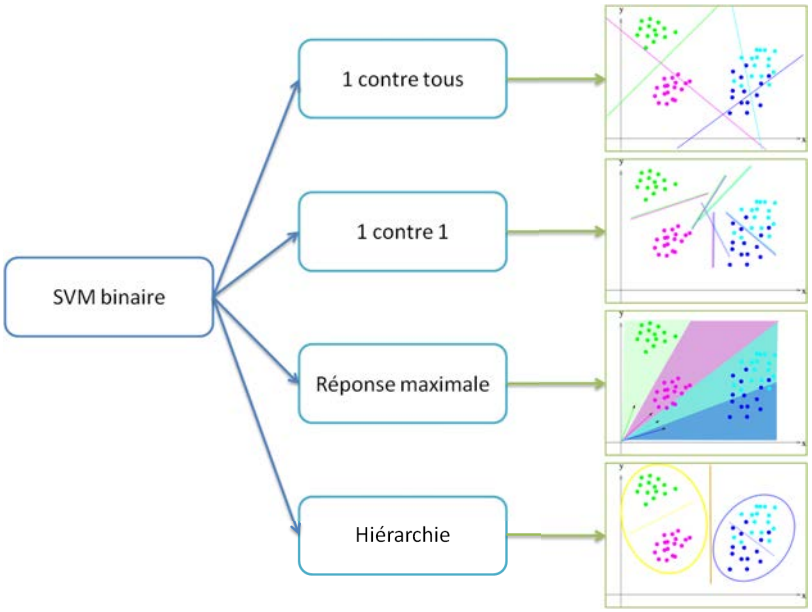


FIGURE 4 – Résumé des quatre principales méthodes de fusion pour le *SVM* à plus de deux classes

D'autres contextes de classification

4 Le *SVM* à plus de deux classes

Les sections précédentes s'intéressent à la classification binaire de points c'est à dire au contexte où $X = \mathbb{R}^D$ et $Y = \{-1, 1\}$. Certains des algorithmes présentés peuvent être trivialement étendus au contexte où Y est un ensemble fini mais de cardinal quelconque comme *NN* (on prend la classe du plus proche voisin) et *KNN*. Par contre, *SVM* qui est plus pertinent pour les *problèmes qui répondent à un besoin* est pour l'instant strictement binaire.

Dans la section suivante, on présente les quatre méthodes pour permettre au *SVM* de traiter plus de deux classes dont une visualisation est donnée dans les figures 4 et 5. Ces méthodes sont notamment différentes d'un point de vue fonctionnel et conduisent à d'importantes différences en terme de nombre d'hyperplans appris et testés sur chaque nouveau point. Ces différences sont résumées dans la table 4.

4.1 *1 contre tous*

L'approche la plus courante pour appliquer *SVM* dans un contexte à plus de deux classes est la méthode *1 contre tous* qui consiste à apprendre un *SVM*

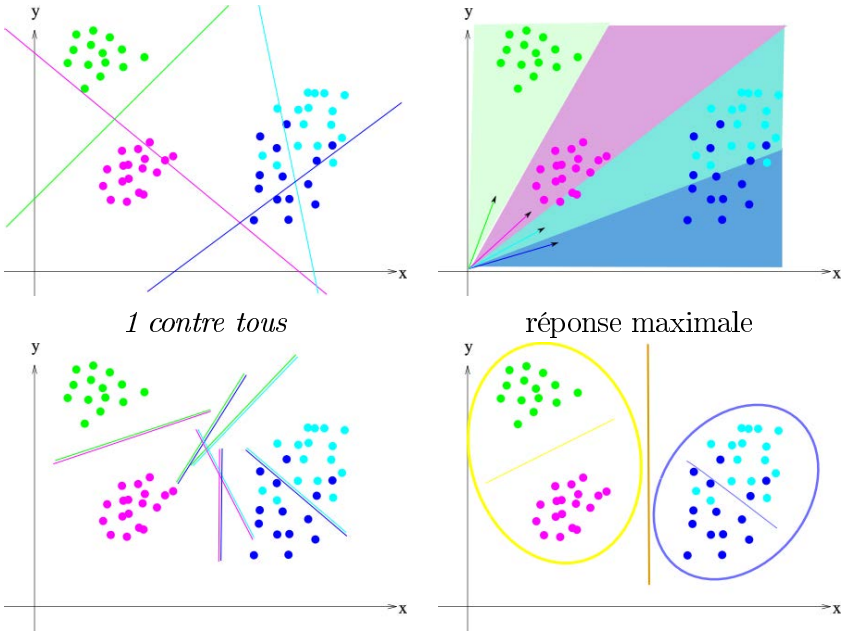


FIGURE 5 – Illustrations des quatre principales méthodes de fusion pour le *SVM* à plus de deux classes

	appris	testés	spécifiques	comparables
1 contre tous	$\|Y\|$	$\|Y\|$	oui	non
réponse maximale	$\|Y\|$	$\|Y\|$	non	oui
1 contre 1	$\|Y\|^2$	$\|Y\|^2$	oui	non
hiérarchie	$\|Y\|$	$\log(\|Y\|)$	oui	non

caractéristiques des quatre processus de fusion en termes de modèles

pour chaque classe en considérant les exemples de cette classe comme positifs et tous les exemples des autres classes comme négatifs.

L'inconvénient majeur de cette méthode est qu'étant donné un point, on a alors $|Y|$ réponses binaires (une par plan) et non pas une réponse dans Y. Dans le cas où un seul *SVM* binaire répond positivement, il suffit de décider la classe associée à ce plan. Mais, il n'est pas trivial de décider une seule classe si plusieurs *SVM* binaires répondent positivement. Dans ce cas, une possibilité est de prendre en compte la distance à chaque plan pour fusionner les réponses. Cependant, ces plans étant appris indépendamment les uns des autres, il n'y a aucune raison pour que les distances avec chacun des plans soient comparables (les plans pouvant même appartenir à des espaces différents) même si des normalisations heuristiques existent [69].

Par contre, un avantage de cette méthode est de pouvoir apprendre un modèle spécifique par classe et même de pouvoir effectuer la classification dans un espace spécifique à la classe : par exemple, on peut utiliser un noyau différent pour chaque *SVM* binaire.

4.2 *1 contre 1*

La méthode *1 contre 1* consiste à apprendre un *SVM* binaire pour chaque paire de classes (ce qui permet de se ramener naturellement à une classification binaire). Étant donné un point, la classe décidée est alors celle ayant obtenu le plus de réponses positives parmi toutes les réponses (une par paire). Cette méthode possède un processus non ambigu (on décide la classe ayant reçu le plus de votes) à la différence de la méthode *1 contre tous*.

Tout comme la méthode *1 contre tous*, cette méthode a l'avantage de permettre d'utiliser une représentation spécifique par paire de classes.

Par contre, cette méthode dispose toujours de moins d'exemples que la méthode *1 contre tous*. Mais *SVM* converge rapidement du moment qu'il est capable de capturer la distribution des concepts. Or, la distribution des concepts est intuitivement plus simple à capturer pour 2 classes que pour 1 classe contre les autres. Cet argument est confirmé expérimentalement : cette méthode est considérée comme plus performante que *1 contre tous* [22].

L'inconvénient majeur de cette méthode est qu'elle implique de calculer le produit scalaire avec $\frac{1}{2}|Y| \times (|Y| - 1)$ vecteurs pour donner une réponse, ce qui n'est pas possible pour par exemple $|Y| = 100000$ [31] car $\frac{1}{2}|Y| \times (|Y| - 1) = 4999950000$.

4.3 Associer un score à chaque classe

La méthode *SVM* est naturellement binaire : il s'agit de trouver un hyperplan séparant deux nuages de points i.e. trouver w tel que $y^*(x_n)\langle w, x_n\rangle > 0$ (le produit scalaire est noté $\langle ., .\rangle$ pour éviter les confusions ici). Cependant ce problème est équivalent à chercher w tel que $\langle w, (y(x_n)\, x_n)\rangle > 0$.

Ce nouveau problème dit du *Perceptron* [74] consistant à chercher un vecteur représentatif est *semblable* à celui de trouver un hyperplan séparateur. L'algorithme du *Perceptron* introduit dans [74] pour résoudre ce problème est :

```
//v ensemble des vecteurs d'apprentissage
//w représentatif des v
//ps le produit scalaire
w = apprendrePerceptron(v)
   w = 0;
   tant qu'il existe n dans [1,N] tel que ps(v[n],w) <= 0
      w += v[n];
   retourne w;
```

Il est démontré [12] que cette algorithme renvoie (si elle existe) une solution approchée de la formulation $\min_{v\in\mathbb{R}^D}\left(\|v\|_2^2 + C_\infty \sum_{n=1}^{N} h\left(vx_n - 1\right)\right)$ où C_∞ symbolise que les termes $vx_n - 1$ sont contraints d'être positifs. Or, cette dernière formulation est très similaire au problème (1) (section 3 associé au SVM) : la seule différence est l'absence d'un terme constant b.

Ainsi, la recherche d'un hyperplan séparateur est semblable à la fois sémantiquement et fonctionnellement à la recherche d'un vecteur représentatif. Or, chercher un vecteur représentatif est facilement extensible à plus de deux classes en ne cherchant pas un vecteur mais $|Y|$ vecteurs. Il suffit simplement d'imposer que pour chaque exemple, son vecteur représentatif lui soit plus proche que ceux des autres classes.

Étant donné les vecteurs $x_1, ..., x_N \in \mathbb{R}^D$, cela conduit à la formulation :

$$\min_{w\in\mathbb{R}^{Y\times D}}\left(\|w\|_2^2 + C\sum_{n,y} h\left(\left(w_{y^*(x_n)} - w_y\right)x_n - 1\right)\right) \tag{2}$$

On peut remarquer que ce problème est un cas particulier du problème du *perceptron* à une classe où on veut w tel que $\langle w, v_n\rangle > 0$: en notant $z_{(n,y)}$ le vecteur tel que pour tous les α, $z_{(n,y),\alpha d} = v_d$ si $\alpha = y^*(x_n)$, $z_{(n,y),\alpha d} = -v_d$ si $\alpha = y$ et 0 sinon, résoudre (2) revient à résoudre

$$\min_{w\in\mathbb{R}^{Y\times D}}\left(\|w\|_2^2 + C\sum_{(n,y)} h\left(wz_{(n,y)} - 1\right)\right)$$

Cependant, rien que pour écrire les vecteurs $z_{(n,y)}$ il faut $N\times D\times Y$ nombres contre $N\times D$ pour la formulation (2) ce qui amène à traiter spécifiquement le problème (2).

4.4 Créer une hiérarchie

Pour présenter les méthodes par hiérarchie, introduisons dans ce contexte les arbres binaires.

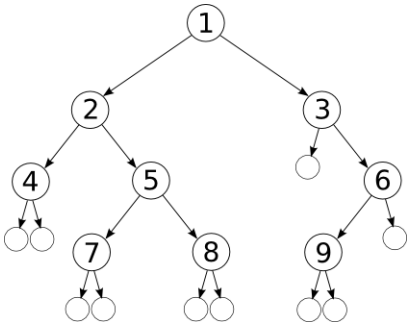


FIGURE 6 – Un arbre binaire

Un arbre binaire est une structure récursive faite de nœuds et de feuilles, dans laquelle chaque nœud à deux fils (un fils droit et un fils gauche) étant eux même des arbres binaires. Le plus simple des arbres est une simple feuille. La figure 6 donne un exemple d'arbre binaire où les nœuds sont représentés par un numéro dans un cercle et où les feuilles sont représentées par un cercle vide. Le nœud 1 est la racine et a deux fils qui sont les arbres de racines 2 et 3. Les nœuds 4,5,8,9 on chacun deux fils qui sont justes des feuilles comme le fils gauche de 3 et le fils droit de 6.

Dans les méthodes par *hiérarchie* comme [8], on construit un arbre binaire dont les feuilles sont associées à une classe de Y et dont les noeuds sont associés à un SVM binaire. Étant donné un point x arrivant sur un nœud et donc sur un SVM binaire, on fait descendre le nœud vers le fils gauche si la réponse du SVM de ce nœud sur ce point est négative. Sinon on fait descendre le point vers le fils droit. Ainsi, le point x parcours un chemin de la racine vers une feuille qui est la classe décidée.

Dans ces méthodes, on sépare en deux en chaque nœud l'ensemble des classes atteignant ce nœud. Typiquement, le SVM du nœud est appris avec les éléments du premier sous ensemble comme positifs et les autres comme négatifs.

Dit autrement, pour tous les sous-ensembles $\mathcal{U}, \mathcal{V}$ disjoints de Y, on note $SVM_{\mathcal{U},\mathcal{V}}$ le SVM binaire résultant de l'apprentissage avec les exemples de classe $\mathcal{U}$ comme positifs et les exemples de classe $\mathcal{V}$ comme négatifs. Notons Γ l'ensemble des algorithmes constructibles avec des arbres binaires dans notre contexte. Alors d'une part, pour tous les $y \in Y$, ψ_y la fonction constante valant y, $\psi_y \in \Gamma$. D'autre part, à partir de $\mathcal{G}, \mathcal{D} \in \Gamma$ tel que $\mathrm{Im}(\mathcal{G}) \cap \mathrm{Im}(\mathcal{D}) = \emptyset$, on peut construire une fonction ψ appartenant à Γ telle que $\psi(x) = \begin{cases} \mathcal{D}(x) & SVM_{\mathrm{Im}(\mathcal{G}),\mathrm{Im}(\mathcal{D})}(x) = 1 \\ \mathcal{G}(x) & SVM_{\mathrm{Im}(\mathcal{G}),\mathrm{Im}(\mathcal{D})}(x) = -1 \end{cases}$.

Durant l'étape d'apprentissage à la fois les SVM et la structure de l'arbre doivent être appris. L'approche la plus standard consiste à apprendre les

SVM 1 contre tous puis à former une matrice de similarité S où $S_{y,y'}$ mesure le taux de bonne réponse du SVM appris avec comme positifs les exemples de y sur les exemples de classe y'. Il suffit alors de former deux groupes dans S pour obtenir la racine puis récursivement chacun des nœuds [8] (par exemple avec des techniques de regroupement spectral).

L'avantage de cette approche est que dans le cas où elle sépare chaque ensemble en deux sous-ensembles de taille identique, elle apprend $|Y|$ vecteurs mais n'en teste que $\log(|Y|)$ pour décider de la classe de chaque nouveau point.

5 Classification de suites

Les sections précédentes s'intéressent à la classification de points d'un même espace. Ce contexte est le plus étudié dans la littérature d'apprentissage par ordinateur. Cependant, ce contexte est réducteur. Par exemple, ce contexte ne permet pas de traiter des points dont certaines composantes pourraient être manquantes ce qui conduit à $X = (\mathbb{R} \cup \{\emptyset\})^D$ et non à $X = \mathbb{R}^D$. Il ne permet de plus de formaliser ni les situations où les exemples sont des points qui peuvent être de taille différentes ni le cas où les exemples sont des ensemble finis ou des suites finies.

Parmi l'ensemble des contextes imaginables de classification, deux sont particulièrement étudiés par la littérature : la classification de suites finies, objet de cette section, et la classification de multi ensembles, objet de la section suivante.

Formellement, soit $\mathcal{E}$ un ensemble (par exemple $\mathbb{R}^D$). Une suite finie est un élément de $\mathcal{E}^+ = \bigcup_{n \in \mathbb{N}-\{0\}} (\mathcal{E}^n)$. Cette section présente une méthode pour classer les suites finies. Ainsi, dans cette section $X = \mathcal{E}^+$. On utilise les termes suites ou suites temporelles ou suites finies indistinctement quand cela ne nuit pas à la compréhension.

5.1 Intérêts des méthodes spécifiques

Ce nouveau contexte de classification de suite pourrait se ramener à celui de la classification de points sous de faibles hypothèses : s'il n'existe qu'un ensemble fini de taille pour les suites, il est possible d'apprendre un algorithme de classification de points par taille. Plus généralement, cette réduction ramène le problème de la classification de points de tailles différentes à la classification de points.

Cependant, utiliser un algorithme de classification de points par taille de points divise le nombre d'exemples par le nombre de tailles, ce qui freine d'autant la convergence. Dit autrement, si $X = \bigcup_{D=1}^{D_{max}} \mathbb{R}^D$, utiliser un KNN ou SVM sur chaque $\mathbb{R}^D$ revient à diviser N par D_{max}.

Inversement, si l'on disposait d'une méthode permettant de comparer des points des différents espace entre eux, on pourrait utiliser la totalité des exemples au sein d'un même apprentissage qui serait plus complexe mais qui disposerait de D_{max} fois plus d'exemples.

Si les composantes des points des différents espaces n'ont aucun lien entre elles, une telle comparaison est impossible. Cependant, dans le cas de suite temporelle, les points correspondent à l'observation d'un phénomène sur une durée plus ou moins longue. Ainsi, chaque composante correspond à un instant dans l'observation d'un phénomène de vitesse variable. Dans ces conditions, il parait acceptable de comparer différentes composantes entre elles (même dans le cas de points de taille identique). Une telle méthode pourra ainsi à la fois éviter la division des exemples et utiliser l'information supplémentaire disponible sur les exemples à savoir que les composantes des exemples sont de même nature.

Une méthode pour cela consiste à introduire une *distance* (alors dite élastique) sur les suites temporelles.

5.2 *DTW* : association élastique d'instants

La méthode dite d'association élastique d'instants (*DTW* pour *Dynamic Time Wrapping*) [75] est la distance élastique la plus classique sur les suites temporelles à valeur dans un espace continue. Elle consiste à se ramener à une distance simple en s'autorisant à associer plusieurs instants d'une suite à un seul instant de l'autre.

Formellement, soient $x, x' \in X$ deux suites temporelles de taille T, T', une association croissante de taille Ω des instants de $[1, T]$ avec ceux de $[1, T']$ est une fonction $\phi = (\mu, \nu)$ de $[1, \Omega]$ vers $[1, T] \times [1, T']$ telle que pour tous les $\omega \in [1, \Omega - 1]$, $\mu(\omega) \leq \mu(\omega + 1) \leq \mu(\omega) + 1$ et $\nu(\omega) \leq \nu(\omega + 1) \leq \nu(\omega) + 1$ et telle que $\phi(1) = (1, 1)$ et $\phi(\Omega) = (T, T')$ (on utilise la notation des intervalles pour désigner les intervalles discrets). Le coût d'une telle association est naturellement $\sum_{\omega=1}^{\Omega} \|x(\mu(\omega)) - x'(\nu(\omega))\|$. La distance *DTW* $\mathcal{D}(x, x')$ entre x et x' est le minimum des coûts sur l'ensemble des associations croissantes (de taille quelconque).

Cette distance peut se calculer en $O(T \times T') = O(|x|\,|x'|)$ opérations par programmation dynamique. Soit $x \in X$, on note $x_{[1,t]}$ l'élément de X correspondant aux t premiers instants de x pour $t \in [1, T]$ avec $T = |x|$. Notons $\mathcal{D}_{t,t'} = \mathcal{D}\left(x_{[1,t]}, x'_{[1,t']}\right)$. Ces quantités vérifient les relations suivantes :

$$\mathcal{D}_{t+1,t'+1} = \left\|x(t+1) - x'\left(t'+1\right)\right\| + \min\left(\mathcal{D}_{t+1,t'}, \mathcal{D}_{t,t'}, \mathcal{D}_{t,t'+1}\right)$$

et, de plus, les quantités $\mathcal{D}_{1,t'}$ et $\mathcal{D}_{t,1}$ se calculent trivialement : $\mathcal{D}_{1,t'} = \sum_{t''=1}^{t'} \|x(1) - x'(t'')\|$ et $\mathcal{D}_{t,1} = \sum_{t''=1}^{t} \|x(t'') - x'(1)\|$. Ces deux propriétés permettent de calculer la distance *DTW* (i.e. $\mathcal{D}_{T,T'}$) suivant l'algorithme :

```
//U et V deux suites de taille a et b
dist = DTW(U,V,a,b)
  initialise D //stocke les distances DTW partielles
  pour k de 2 à min(a,b)
    Dprec = min(D[k+1][k],D[k][k],D[k][k+1]);
    D[k+1][k+1] = ||U[k+1]-V[k+1]|| + Dprec;
    pour t de k+2 à a
      Dprec = min(D[k+t-1][k],D[k+t][k-1],D[k+t-1][k-1]);
      D[k+t][k] = ||U[k+t]-V[k]|| + Dprec;
    pour t de k+2 à b
      Dprec = min(D[k-1][k+t],D[k][k+t-1],D[k-1][k+t-1]);
      D[k][k+t] = ||U[k]-V[k+t]|| + Dprec;
  return D[a-1][b-1];
```

La commande `initialise D` de cet algorithme alloue un tableau à deux dimensions et assure que la première ligne et la première colonne du tableau `D` soient initialisées convenablement (en utilisant l'expression triviale de ces valeurs) c'est à dire :

```
  D[1][1] = ||U[1]-V[1]||;
  pour t de 2 à a
    D[t][1] = D[t-1][1] + ||U[t]-V[1]||;
  pour t de 2 à b
    D[1][t] = D[1][t-1] + ||U[1]-V[t]||;
```

5.3 Utilisation de la DTW

À l'aide d'une métrique sur les suites temporelles, il est possible d'appliquer directement KNN sur les données ou d'appliquer un SVM par l'intermédiaire d'un noyau utilisant une distance élastique par exemple si $\phi(x, x') = \mathcal{D}(x, 0)^2 + \mathcal{D}(x', 0)^2 - \mathcal{D}(x, x')^2$ alors on se rapproche sémantiquement du produit scalaire classique (car si x, x' deux vecteurs, $2\langle x, x'\rangle = \|x\|_2^2 + \|x'\|_2^2 - \|x - x'\|_2^2$).

Ainsi, à la différence de la réduction naïve où un KNN ou SVM spécifique est utilisé pour chaque taille de suite séparément, ici un seul KNN ou SVM est utilisé sur l'ensemble des tailles. Cette méthode permet donc de ne pas diviser les exemples par le nombre de tailles.

D'autres techniques spécifiques aux suites temporelles existent notamment des techniques utilisant l'hypothèse de *Markov*. Elles sont plus pertinemment décrites en section 8.

6 Classification de multi ensembles de mots

Cette section présente le contexte de la classification de multi ensembles de mots inclus à un dictionnaire c'est à dire un ensemble de mots M. Ainsi,

dans cette section X est l'ensemble des multi ensembles de mots sur M. L'ensemble d'apprentissage est donc constitué d'un ensemble d'ensembles de mots.

Cela dit, il existe une bijection directe entre les multi ensembles et les points de $\mathbb{N}^M$: si x est un multi ensemble de mots sur M, on peut voir x comme un point de dimensions $|M|$ avec $x_{\{m\}}$ le nombre de mots m inclus dans x vue comme un multi ensemble. Formellement, on a la bijection : $x_{\{m\}} = |\{m \in x\}|$ dans laquelle à gauche x est vu comme un point et à droite x est vu comme un multi ensemble.

Ainsi, il est possible de considérer que $X = \mathbb{N}^M \subset \mathbb{R}^M$. La classification de multi ensembles de mots peut donc directement se ramener à la classification binaire de points. D'ailleurs, la littérature considère qu'appliquer un *KNN* ou un *SVM* sur les points correspondants est pertinent (surtout pour un *SVM* [81]).

Cependant, il existe des méthodes spécifiques à la classification de multi ensembles de mots et notamment une méthode qui permet de ramener l'apprentissage initialement en dimension M à un apprentissage en dimension 1 et cela de façon exacte sous certaines conditions.

6.1 *CNB* : la classification naïve bayésienne

Soit une donnée $x \in X$ dont on cherche la classe $y^*(x) \in Y$. Sous des hypothèses (principalement d'indépendances sur la présences des mots), on peut écrire :

$$P(y = y^*(x)|x) \propto P(y) \times \prod_{m \in x} P(m|y) = P(y) \times \prod_{m \in M} P(m|y)^{x_{\{m\}}}$$

où $x_{\{m\}}$ est le nombre d'occurrences du mot m dans la donnée x et où $P \propto f$ signifie qu'il existe une constante Z telle que $P = Zf$. Cette décomposition n'est que l'application des hypothèses d'indépendance.

Or, chacun des termes dans le produit de droite correspond à une probabilité *simple* notamment finie et discrète. Or, pour une probabilité *simple*, il est pertinent de considérer l'approximation de cette probabilité par les fréquences empiriques qu'on peut calculer étant donné des observations (c'est à dire, ici, $B = x_1, ..., x_N$ l'ensemble d'apprentissage). Notons B_y l'ensemble des n tels que $y^*(x_n) = y$, Cette approximation s'écrit :

$$P(y = y^*(x)|x) \approx \frac{|B_y|}{N} \times \prod_{m \in M} \left(\frac{\sum_{x' \in B_y} x'_{\{m\}}}{\sum_{x' \in B_y} |x'|} \right)^{x_{\{m\}}}$$

où $|x|$ est le nombre de mots extraits de x.

Dans cette équation, seul $x_{\{m\}}$ dépend de la nouvelle donnée, tous les autres termes peuvent être calculés explicitement grâce à la base d'apprentissage. Cette décomposition dite de Classification Naïve Bayésienne *CNB*

permet donc de transformer l'estimation de $P(y|x)$ en $M+1$ estimations de probabilités simples comme $P(m|y)$. C'est un exemple de famille de probabilités pour lesquelles on a un algorithme qui permet de ramener l'apprentissage de dimension quelconque à des apprentissages de dimension 1 (au sens de l'espace).

Malheureusement, le *no free lunch theorem* implique que les hypothèses de la *CNB* sont rarement vérifiées dans l'ensemble des problèmes (et il n'est pas clair de savoir si ces hypothèses sont souvent vérifiées dans les problèmes répondant à des besoins).

6.2 Décision par élection

Cependant, même dans le cas où la *CNB* ne s'applique pas, on peut voir le terme de droite de la décomposition *CNB* comme le score résultant du vote de chaque mot m appartenant à x selon un poids $\left(\frac{\sum_{x\in B_y} x_{\{m\}}}{\sum_{x\in B_y} |x|}\right)$. Cela introduit les méthodes d'élection.

Dans une méthode d'élection, chaque mot vote individuellement pour une classe ou plus généralement avec un poids différent pour chaque classe. Le nombre de votes (ou plus généralement, la somme des votes) reçus pour chaque classe est calculé et la classe avec le plus de votes est élue i.e. décidée.

Formellement, chaque mot m vote selon $v_{y,m}$ pour la classe y, ce qui amène aux scores $S(y,x) = \prod_{m\in x} v_{y,m}$ (ou de façon équivalent via le logarithme $S(y,x) = \sum_{m\in x} v_{y,m}$) et donc à la décision $y(x) = \arg\max_{y\in Y} S(y,x)$.

Il convient alors, à l'apprentissage, de choisir des votes $v_{y,m}$ adaptés au problème. Typiquement, au plus les hypothèses de la *CNB* sont vérifiées, au plus il est pertinent que les votes $v_{y,m}$ soient proches de ceux de la *CNB*. Mais réciproquement, dans le cas où les hypothèses sont fortement non vérifiées, il peut être plus adapté de choisir d'autres votes.

6.3 La *CNB* comme solution d'une optimisation

Si l'on mesure la qualité des votes à travers la quantité $\prod_{n=1}^{N} S(y^*(x_n), x_n)$ et que l'on impose la contrainte $\sum_{m\in M} v_{y,m} = 1$, cela conduit à résoudre $\max_v \prod_{n=1}^{N} \left(\prod_{m\in x_n} v_{y^*(x_n),m}\right)$ sous les contraintes que $\sum_{m\in M} v_{y,m} = 1$ pour tous les $y \in Y$. Alors, les votes optimaux pour ces critères sont $v_{y,m} = \frac{\sum_{x\in B_y} x_{\{m\}}}{\sum_{x\in B_y} |x|}$, c'est à dire $v_{y,m} = P(m|y)$ (Annexe 1).

On retrouve ainsi exactement la décomposition de la *CNB* comme la

conséquence d'un critère de qualité assez intuitif ($\prod_{n=1}^{N} S\left(y^*\left(x_n\right), x_n\right)$) et d'un choix de contrainte sur les votes ($\sum_{m \in M} v_{y,m} = 1$).

Cette contrainte sur les votes est pertinente dans le cas où la *CNB* s'applique (puisque elle permet dans ce cas d'obtenir directement $P(y|x)$). Mais dans le cas où la *CNB* ne s'applique pas, d'autres contraintes peuvent être plus pertinentes.

6.4 Cas du suffrage universel

Par exemple, on peut se demander ce que deviennent les scores si on impose les contraintes $\sum_{y \in Y} v_{y,m} = 1$. Cette contrainte est plus proche de ce que l'on attend d'une élection : chaque mot a globalement 1 voix c'est à dire l'élection s'effectue au suffrage universel.

Le *suffrage universel* des mots conduit (Annexe 1) à $v_{y,m} = \frac{\sum_{x \in B_y} x_{\{m\}}}{\sum_{x \in B} x_{\{m\}}}$, c'est à dire $v_{y,m} = P(y|m)$.

On voit ainsi qu'avec ces nouvelles contraintes sur les votes, on obtient des votes tout aussi pertinents sémantiquement. Les meilleurs votes dépendent de P. Plus la *CNB* s'applique, plus les votes $P(m|y)$ sont pertinents mais moins la *CNB* s'applique plus les votes $P(y|m)$ ont de chance d'être meilleurs que $P(m|y)$.

6.5 Lien avec l'application d'un *SVM*

Pour les deux méthodes d'élection précédentes, l'objectif est de maximiser $\prod_{n=1}^{N} S\left(y^*\left(x_n\right), x_n\right)$ c'est à dire le produit des scores que donnerait l'algorithme aux exemples d'apprentissage. Cependant, une grande ligne directrice de la littérature est qu'à formalisme égal, il est pertinent de chercher à minimiser l'erreur d'apprentissage. Dans le cas des méthodes de votes, cela permet de voir sous un angle différent l'approche très employée dans la littérature qui consiste à utiliser la bijection entre les multi ensembles et les points pour utiliser un *SVM* sur les points correspondants [81].

L'utilisation d'un *SVM* linéaire (sans noyau) est une méthode par élection : pour décider de la classe d'un point $x \in \mathbb{R}^D$ le *SVM* calcule $\langle w, x \rangle$ (dans le cas sans biais) i.e. $wx = \sum_{d=1}^{D} w_d x_d$. Or, dans le cas où x est un multi ensemble, pour tous les d, il existe un mot $m \in M$ tel que $x_d = x_{\{m\}}$ (c'est à dire le nombre d'occurrence du mot m dans x). Ainsi $\langle w, x \rangle = \sum_{m \in M} w_m x_{\{m\}} = \sum_{m \in x} w_m$. Il s'agit donc d'une méthode d'élection dans laquelle le mot m vote pour la classe 1 avec le vote $v_{1,m} = w_m$ (et $-w_m$ pour la classe -1). Cette

équivalence avec une méthode d'élection n'est vraie que pour le simple *SVM* linaire sans transformation sur les points : pour *NN* ou pour des *SVM* à noyau ou si les points sont transformés, les mots ont un effet dans leur ensemble et non plus individuellement.

Dans cette méthode, les votes v sont donc la solution du problème d'optimisation suivant : $\arg\min_{v} \left(vv + C \sum_{n=1}^{N} h\left(\left\langle v_{y^*(x_n)}, x_n \right\rangle - 1 \right) \right)$. Or, le produit scalaire $\left\langle v_{y^*(x_n)}, x_n \right\rangle$ est exactement le score $S\left(y^*\left(x_n\right), x_n\right)$ et le terme $h\left(S\left(y^*\left(x_n\right), x_n\right) - 1\right)$ mesure si une erreur est commise sur l'exemple d'apprentissage x_n.

Ainsi, la méthode consistant à appliquer un *SVM* linéaire sur les points revient simplement à effectuer une élection au suffrage censitaire (les votes ayant un coût en vv) en faisant voter chaque mot m pour la classe y avec un vote $v_{y,m}$ de sorte à minimiser l'erreur sur la base d'apprentissage : $\arg\min_{v} \left(vv + C \sum_{n=1}^{N} h\left(S\left(y^*\left(x_n\right), x_n\right) - 1\right) \right)$.

Cette dernière sous section tend plutôt à diminuer l'intérêt des méthodes spécifiques à la classification d'ensemble puisque l'application d'un simple *SVM* linéaire revient à appliquer une méthode d'élection utilisant les votes minimisant l'erreur d'apprentissage et donc des votes déjà pertinents (bien que forcément moins pertinents que ceux de la *CNB* si elle s'applique).

Cependant, cette autre façon de voir le *SVM* linéaire dans ce contexte lui donne notamment une justification vis à vis d'un *SVM* à noyau ou de *NN* mais surtout cette autre vision est au cœur de la contribution principale de cette thèse qui s'intéresse à l'utilisation d'une méthode d'élection en segmentation (partie 3). Cela amène aussi à introduire le problème de la segmentation dans les sections suivantes.

La segmentation supervisée

Les sections précédentes s'intéressent à la classification : chaque exemple est associé à un et un seul concept. On s'intéresse maintenant au contexte de segmentation supervisée dans lequel un exemple est associé à plusieurs concepts *localisés à des positions strictement différentes*. Typiquement, on s'intéresse au contexte où $X = \mathcal{E}^+$ (par exemple $X = \left(\mathbb{R}^D\right)^+$ ou $X = \left(\mathbb{N}^M\right)^+$) et où y^* va de X vers Y^+ avec $|y^*(x)| = |x|$). Ainsi chaque instant t de $x \in X$ est associé à une classe $y^*(x,t) \in Y$ (dans les sections suivantes Y peut contenir 3 éléments).

Comme pour la classification de suites, ce problème se ramène à la classification sous de faibles hypothèses. Considérons l'hypothèse qu'il existe un nombre $\triangleright$ tel que l'information contenue dans $x_{|[t-\triangleright, t+\triangleright]}$ est suffisante pour connaître $y^*(x,t)$. Dit autrement, cette hypothèse est que l'erreur minimale

associée à la nouvelle probabilité $P\left(y^*\left(x,t\right)|x_{|[t-\triangleright,t+\triangleright]}\right)$ est la même que l'erreur minimale du problème de départ. Sous cette faible hypothèse, il existe un moyen de ramener la segmentation supervisée à la classification supervisée.

Quel que soit $x \in X$, on considère le point $x_{t'}$ tel que $\forall t \in [-\triangleright, \triangleright]$, $x_{t'}\left(t\right) = x\left(t'+t\right)$. Étant donnés $x_1, ..., x_N$ des exemples dans X, on peut créer des exemples de classification afin d'appliquer un *KNN* : il suffirait de considérer tous les points $x_{n,t'}$ chacun respectivement de classe $y^*\left(x_n, t'\right)$. Cela permet directement d'apprendre $P\left(y^*\left(x,t\right)|x_{|[t-\triangleright,t+\triangleright]}\right)$. Ainsi, étant donnée une nouvelle donnée $x \in X$, il suffirait pour chaque instant t' de construire $x_{t'}$ et d'utiliser le *KNN* appris pour obtenir $y^*\left(x,t\right)$.

Remarquons que cette réduction n'est pas acceptable sur *les bords* des exemples c'est à dire pour $t = 1, ..., \triangleright - 1$ et pour $t = T - \triangleright + 1, ..., T$. Cependant, en segmentation on suppose que $T \gg \triangleright$, il n'est donc pas nécessaire de prendre en compte *les bords*.

Cependant cette réduction naïve est intuitivement problématique. On a l'intuition qu'un algorithme appliqué dans ce contexte ne peut être performant que si implicitement il comprend qu'il doit faire jouer un rôle particulier au temps, ce que l'on sait déjà de part la construction des données. La réduction naïve impose donc de comprendre simultanément les classes et le temps, alors qu'il est possible de comprendre les classes à travers une modélisation explicite du temps.

La section suivante présente des algorithmes qui cherchent des intervalles de classes homogènes. La section 8 présente des algorithmes qui découplent l'information apportée par chaque image individuellement et l'information apportée par le séquencement (avec une réduction de dimension) La section 9 présente l'extension de la *CNB* au problème de la segmentation ce qui est ensuite repris plus en détail dans la partie 3.

7 Apprendre sur les zones homogènes

Dans le contexte de segmentation $y^*\left(x,t\right)$ peut changer à chaque instant t. Cependant, on a l'intuition que $y^*\left(x,t\right)$ varie peu dans les problèmes qui répondent à des besoins. Cela conduit à introduire les zones homogènes : ce sont les différents intervalles maximaux sur lesquelles $y^*\left(x,t\right)$ est constante. Ainsi, cette méthode permet de fournir à l'apprentissage des intervalles homogènes dont la distribution peut être plus facile à capturer. Notamment en présence de mots forts, on peut s'autoriser à supprimer l'information temporelle lors de l'apprentissage sur les intervalles homogènes [46] (ce qui réduit fortement la dimension).

7.1 Segmentation construite sur les zones homogènes

La première méthode de segmentation consiste à apprendre un algorithme sur ces zones homogènes. On apprend donc dans un contexte de classification

(mais comme pour classer des suites et non pas en effectuant la réduction naïve). Ce regroupement en intervalles homogènes est possible à l'apprentissage car les classes sont connues. Formellement, soit $B = x_1, ..., x_N \in X$ les données d'apprentissage. Il est possible de regrouper les instants en intervalles de classes homogènes : soit les instants $t_{n,i}$ tels que $y^*(x_n, t')$ ait une valeur constante pour $t' \in [t_{n,i}, t_{n,i+1} - 1]$ et que $y^*(x_n, t_{n,i}) \neq y^*(x_n, t_{n,i+1})$. Dans ce contexte, on note $\triangleright$ la durée maximale d'une zone homogène. On apprend alors L un algorithme de classification de suites sur les exemples $x_{n|[t_{n,i},t_{n,i+1}-1]}$ associés aux classes $y^*(x_n, t_{n,i})$. Cet algorithme doit retourner un score de confiance envers chacune des classes plutôt que simplement une classe (ce qui ne restreint pas significativement la généralité).

Sur une nouvelle donnée (dont la suite de classes est inconnue) on ne peut pas directement extraire les intervalles homogènes. Il convient alors d'appliquer l'algorithme appris sur tous les intervalles de la nouvelle donnée et de ne conserver que ceux qui semblent homogènes. Formellement, étant donnée une nouvelle suite $x \in X$ avec $|x| = T$, on forme l'ensemble des scores $S(t, t', y) = L_B\left(x_{|[t,t']}\right)$ où $[t, t']$ appartient à un ensemble dense d'intervalles de taille inférieure à $\triangleright$, et où $S(t, t', y)$ est le score pour la classe y sur l'intervalle $[t, t']$. La construction de S est possible en $O\left(|Y| \times \triangleright^2 \times T\right)$ opérations.

L'idée de ce type d'algorithmes est alors de former une segmentation à l'aide des maximaux de S. Il existe pour cela deux techniques classiques.

7.2 Suppression des non maximaux

La suppression des non maximaux consiste à conserver en priorité les intervalles sur lesquels on estime avoir une réponse fiable. Typiquement, un critère de fiabilité d'un intervalle est soit l'intensité du score maximal soit la différence entre les deux scores maximaux.

Ce post traitement consiste donc à trier par ordre décroissant l'ensemble des intervalles vis à vis d'un critère de fiabilité puis de sélectionner le *meilleur* intervalle (et décider sur cet intervalle la classe ayant le score maximal) puis de sélectionner le *meilleur* intervalle suivant ne chevauchant aucun intervalle déjà choisi et ainsi de suite :

```
//S(t,t',y) score prédit pour y sur l'intervalle [t,t']
segmentation = SNM(S)
   L = trier (t,t',y) par ordre de S(t,t',y) décroissant;
   segmentation[1..T] = vide;
   tant que L est non vide
      (t,t',y) = L.dépile();
      collision = faux;
      pour u de t à t'
         si (segmentation[u]!=vide)
            collision=vrai;
```

```
    si (collision==faux)
      pour u de t à t'
        segmentation[u] = y;
  retourne segmentation;
```

Cette méthode présente cependant un inconvénient majeur : la nature gloutonne de l'algorithme sélectionne des intervalles très pertinents au début mais potentiellement très peu pertinents à la fin, ce qui peut conduire à de faibles performances.

7.3 Pavage

La méthode de pavage consiste à sélectionner globalement un ensemble d'intervalles qui recouvrent sans chevauchement le support de la suite x. Pour cela, on définit le score d'un pavage (un ensemble d'intervalles non chevauchants, remplissant l'intervalle $[1, T]$). Le score du pavage en R parties d'instants pivots $t_2, ..., t_R$ et associé aux classes $y_1, ..., y_R$ est par exemple $\left(\sum_{r=1}^{R} S\left(t_r, t_{r+1}, y_r\right)\right)$.

La méthode par pavage [46, 73] consiste à choisir comme segmentation, le pavage de score maximal. Cela revient à résoudre le problème suivant : $\max_{R, y_1, ..., y_R, t_2, ..., t_R} \left(\sum_{r=1}^{R} S\left(t_r, t_{r+1}, y_r\right)\right)$ avec la convention que $t_1 = 1$ et $t_{R+1} = T$. Ce type de problème peut se résoudre efficacement par programmation dynamique (i.e. avec le même type de technique que pour le DTW). Pour cela, on introduit pour tous les τ (correspondant au temps final) les quantités :

$$S_\tau = \max_{y_1, ..., y_R, t_2, ..., t_R} \left(\sum_{r=2}^{R-1} S\left(t_r, t_{r+1}, y_r\right) + S\left(t_R, \tau, y_R\right)\right)$$

Ces quantités vérifient la récurrence :

$$S_\tau = \max\left\{\max_y \left(S\left(1, \tau, y\right)\right), \max_{y, \nu} \left(S_\nu + S\left(\nu + 1, \tau, y\right)\right)\right\}$$

où par convention $S\left(t, t', y\right) = -\infty$ quand la configuration t, t', y est impossible (par exemple si la $t' - t \gg \triangleright$). Cette dernière équation permet de calculer le pavage optimal (dont le score est S_T).

Avec ce post traitement, il est possible de calculer une segmentation sans trou et homogène en score avec peu de ressources machines (précisément en $O\left(|Y| \times \triangleright \times T\right)$ opérations).

8 Apprendre sur des configurations locales

8.1 Segmentation construite sur des termes locaux

La deuxième méthode est notamment justifiée par une diminution du nombre de variables de l'apprentissage dont la mécanique est de n'apprendre que des influences locales. Ces influences locales conduisent ensuite à une segmentation par un traitement global.

Intuitivement, la probabilité d'observer deux classes y et y' à t et $t+1$ n'est pas uniforme vis à vis des classes y, y'. Compte tenu de cette non uniformité, il convient de pénaliser les segmentations qui contiendraient des transitions peu probables. De même, la probabilité d'observer à un instant donné une certaine classe y et une certaine valeur x_t n'est pas non plus uniforme et invite à pénaliser les segmentations qui contiendraient des appariements locaux peu probables.

Cela introduit un deuxième exemple de familles de probabilité pour laquelle on dispose d'algorithmes spécifiques : il s'agit des probabilités qui admettent une forme $P\left(y\left(x\right)|x\right) \propto \prod_{t=1}^{T} P\left(y_t|x_t\right) \times \prod_{t=1}^{T-1} P\left(y_{t+1}|y_t\right)$ avec $P\left(y_t|x_t\right)$ la probabilité que la classe à l'instant t soit y sachant qu'on observe à l'instant t les données x_t et $P\left(y_{t+1}|y_t\right)$ la probabilité que la classe à l'instant $t+1$ soit y' sachant que la classe à l'instant t est y. Cette famille de probabilités est associée à l'hypothèse de *Markov* : l'état de l'instant t ne dépend que des entrées de l'instant t et de l'état de l'instant $t-1$.

Comme pour la *CNB*, on peut utiliser cette forme de décomposition même dans le cas où les hypothèses de *Markov* ne sont pas vérifiées. On introduit deux termes. Le premier est un terme d'attache aux données évaluant la pertinence d'un couple classe valeur à un instant donné. Ce terme est instantané car il ne s'intéresse qu'à un instant. Le deuxième évalue la pertinence de la classe en un instant étant données les classes des instants voisins. Ce terme est local car il ne s'intéresse qu'aux voisins. Formellement, on introduit $Q\left(y = y^*\left(x,t\right)|x_t\right)$ abréviée $Q\left(y|x_t\right)$ qui se veut représentative de la probabilité que la classe à l'instant t soit y sachant qu'on observe à l'instant t les données x_t. On introduit de même $Q\left(y' = y^*\left(x,t+1\right)|y = y^*\left(x,t\right)\right)$ abréviée $Q\left(y'|y\right)$ qui se veut représentative de la probabilité que la classe à l'instant $t+1$ soit y' sachant que la classe à l'instant t est y.

Chaque segmentation peut alors être associée à un score dépendant uniquement de Q. Par exemple, le score de la segmentation $y\left(x\right)$ sachant x peut être : $S\left(y\left(x\right)|x\right) = \prod_{t=1}^{T} Q\left(y_t|x_t\right) \times \prod_{t=1}^{T-1} Q\left(y_{t+1}|y_t\right)$. Cette décomposition permet néanmoins de réduire l'apprentissage à l'apprentissage de Q seulement. Dit autrement, cette méthode réduit le nombre de variables de l'apprentissage de $M \times \triangleright$ (pour un *SVM*) à $M + \triangleright$. Le nombre de variables n'est en toute généralité pas corrélé avec la dimension de l'apprentissage au sens de

Vapnik Chervonenkis mais on sous entend que c'est implicitement le cas ici.

Étant donnée une nouvelle suite x, et étant fixé $Q(y|x_t)$ et $Q(y'|y)$, il convient alors de décider la segmentation de plus haut score pour x c'est à dire décider $y(x)$ tel que $S(y(x)|x)$ soit maximal.

En pratique, cette segmentation peut se calculer par programmation dynamique en introduisant :

$$S_{y,[1,t']} = \max_{y_1,\ldots,y_{t'}=y} \left(\prod_{t=1}^{t'-1} Q(y_{t+1}|y_t) \prod_{t=1}^{t'} Q(y_t|x_t) \right)$$

Ces quantités vérifient la récurrence :

$$S_{y',[1,t'+1]} = \max_{y} \left(S\left(y, \left[1, t'\right]\right) \times Q\left(y'|y\right) \times Q\left(y'|x_{t'+1}\right) \right)$$

Cette récurrence (initialisée par $S_{y,1} = Q(y|x_1)$) permet de calculer l'ensemble des valeurs $S_{y,[1,t']}$ (et donc $\max_y \left(S_{y,[1,T]}\right)$ le score de la segmentation optimale).

La méthode de calcul précédente utilise $O\left(|Y|^2 \times T\right)$ opérations, Y étant l'ensemble des classes. Cependant, cette méthode de calcul nécessite d'effectuer toutes ces opérations en une seule fois. Ainsi, on ne peut connaître la classe à l'instant $t = 1$ qu'après avoir parcouru les T instants. Dit autrement, cette méthode de calcul a une latence infinie et ne permet pas de traiter des flux continus. Il est possible de n'utiliser que $[1, t + \triangleright]$ pour décider de la classe à l'instant t. Mais, la complexité devient $O\left(|Y|^2 \times T + |Y| \triangleright \times T\right)$.

8.2 Utilisation d'états supplémentaires

La méthode décrite précédemment peut fonctionner exactement de la même façon si on remplace l'ensemble des classes Y par un produit cartésien $Y \times H$ où H est un ensemble de symboles permettant de représenter la structure interne des zones homogènes d'une classe.

L'algorithme produit alors deux suites y, h correspondant aux classes et aux états au cours du temps qui donnent la probabilité maximale (où seul y est conservé). Alternativement, il peut produire une suite y correspondant à la suite qui maximise la marginalisation selon h (on utilise alors l'algorithme dit *progressif rétrograde* [71]).

Il est classique dans la littérature de représenter ces états comme des nœuds intermédiaires entre les données et les classes mais tous les modèles classiques peuvent se formaliser comme l'utilisation de $Y \times H$ à la place de Y.

8.3 Optimisation des termes

Lors de l'utilisation de ce type de méthode sur une nouvelle donnée, les termes $Q(y'|y)$ et $Q(y|x)$ sont fixés. L'apprentissage consiste justement à les

choisir à la vue des exemples. La littérature est vaste (citons par exemple [71, 54, 5, 9]) sur cette dernière question et on présente ici une taxonomie des différent types d'apprentissage.

Tout d'abord, ce type de méthodes est couramment utilisé en classification [71], la modélisation du temps n'étant alors qu'un moyen de mieux comprendre la structure des différentes classes. Il est aussi utilisé pour faire de la segmentation dans [9] ou dans [54].

D'autre part, l'apprentissage est différent selon que X est continu [5] ou symbolique [71] c'est à dire selon que $X = \left(\mathbb{R}^D\right)^+$ ou $X = (M)^+$.

Ensuite, l'apprentissage peut être dit *génératif* s'il cherche à se rapprocher de $P\left(y\left(x\right), x\right)$ [71] ou dit *discriminatif* s'il cherche à se rapprocher de $P\left(y\left(x\right)|x\right)$ [54].

Enfin, l'apprentissage est différent selon la présence d'un ensemble d'états supplémentaires H (à la différence de l'utilisation du modèle qui est globalement identique). Sans H (dans le cas discret), $S\left(y\left(x\right)|x_n\right)$ n'est qu'une expression des différentes valeurs de Q. Fixer un objectif sur les valeurs $Q\left(y|x\right)$ et $Q\left(y'|y\right)$ définit alors directement des valeurs optimales pour Q (comme pour les votes).

Par contre, l'utilisation de H implique que l'apprentissage mélange l'inférence des suites h_n et de Q.

9 Apprendre des votes temporels

La troisième méthode de segmentation consiste à étendre la *CNB* et les méthode d'élection à la segmentation. Cette méthode est donc spécifique au contexte $X = \left(\mathbb{N}^M\right)^+$.

La *CNB* s'applique sur des ensembles de mots sur un dictionnaire M via l'égalité $P\left(y = y^*\left(x\right)|x\right) \propto p\left(y\right) \prod_{m \in x} P\left(m|y\right)$. Or, rien ne change si les mots sont localisés dans le temps (relativement à un instant central dont on cherche la classe) excepté que le dictionnaire n'est plus M mais $M \times [-\triangleright, \triangleright]$. On peut dans ce contexte écrire $P\left(y = y^*\left(x\right)|x\right) \propto P\left(y\right) \prod_{(m,t) \in x} P\left((m,t)|y\right)$.

Dit autrement en segmentation, la *CNB* (ainsi que le *suffrage universel*) réduit la dimension de l'apprentissage de $O\left(M\right)$ à 2 (au sens de l'espace).

Cependant, il n'est pas clair que la dimension, au sens de *Vapnik Chervonenkis*, des méthodes d'élections soit différente de celle d'un *SVM* $O\left(M\right)$. Cela serait possible (bien que peu de résultat existe sur ce point à ma connaissance) si, par exemple, la composante temporelle était traitée différemment de la composante dictionnaire.

Cela nous amène à introduire [56] qui propose la méthode des *modèles de formes implicites* (*ISM*) proche d'une méthode d'élection utilisant la position des mots. Cette méthode est aussi proche d'une méthode de segmentation.

L'entrée de cette méthode est un ensemble de mots localisés x. Cette méthode se décompose en 4 étapes.

- chaque mot localisé $(m, t') \in x$ vote pour l'hypothèse qu'un intervalle de classe y est centré en t dans la donnée x avec un poids $P(y, t' - t|m)$
- l'ensemble des votes est aggloméré : $S(y, t, x)$ est la quantité de votes obtenue par l'hypothèse qu'un intervalle de classe y est centré en t dans la donnée x c'est à dire : $S(y, t, x) = \sum_{(m,t') \in x} P(y, t - t'|m)$ (contrairement à l'intuition, il s'agit bien ici de sommer les $P(y, t - t'|m)$ et non pas les multiplier)
- les scores S sont lissés temporellement
- Un intervalle de classe y centré en t est détecté si $S(y, t, x)$ est supérieur à un seuil.

Dans ce dernier algorithme, il y a une claire asymétrie entre le dictionnaire et le temps notamment visible dans l'utilisation d'un lissage temporel. Une présentation plus complète de cette idée d'utiliser une méthode d'élection en segmentation est présentée ultérieurement, associée avec des contributions de la thèse (sections 18,19,20).

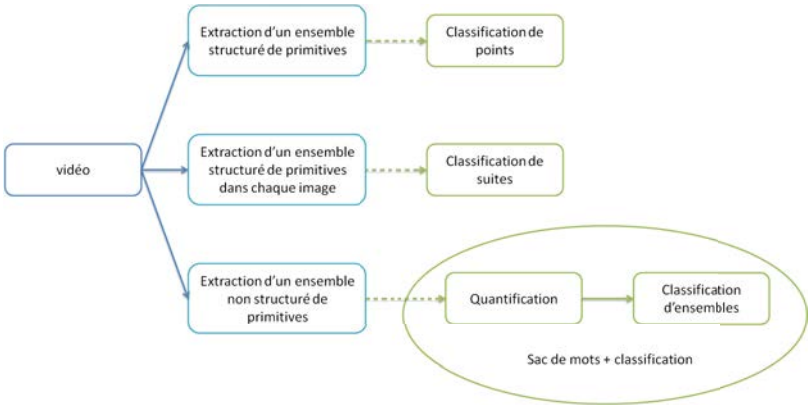


FIGURE 7 – Une décomposition en trois ensembles de la façon dont on compare l'information dans un système *à primitives* de reconnaissance d'actions

Deuxième partie

Les systèmes de reconnaissance d'actions

Les algorithmes d'apprentissage présentés comme (*NN* et *SVM*) peuvent d'un point de vue fonctionnel directement être appliqués à la reconnaissance d'actions dans des vidéos : il est possible de considérer une vidéo standard comme un point. Cependant, utiliser directement un de ces deux algorithmes n'est pas pertinent. Typiquement, apprendre à classer des actions dans des vidéos couleurs (3 canaux) d'une seconde échantillonnée à 25Hz (de taille d'image standard : 640x480 pixels) revient à faire un apprentissage en dimension 23040000 (3x640x480x25), ce qui est un problème pour *NN*. De même, les vidéos brutes ne sont pas directement linéairement séparables (changer une valeur d'un seul pixel ne change pas la sémantique) ce qui est un problème pour les *SVM*.

Ainsi, la réponse à un problème de reconnaissance d'actions est généralement un système de reconnaissance d'actions plus qu'un algorithme d'apprentissage statistique (bien que la différence soit en partie arbitraire). La présentation de ces systèmes est l'objet de cette partie.

La section 10 se focalise sur les différentes façons d'utiliser l'information et les différences que cela implique. Cela introduit la notion de primitives, ainsi que le lien entre les types de représentations des données et les algorithmes d'apprentissage résumé en figure 7. Les sections 11 à 13 présentent les primitives de la littérature qui sont résumées dans la figure 8.

Cependant, il existe des systèmes (objets de la section 14 et 15) qui

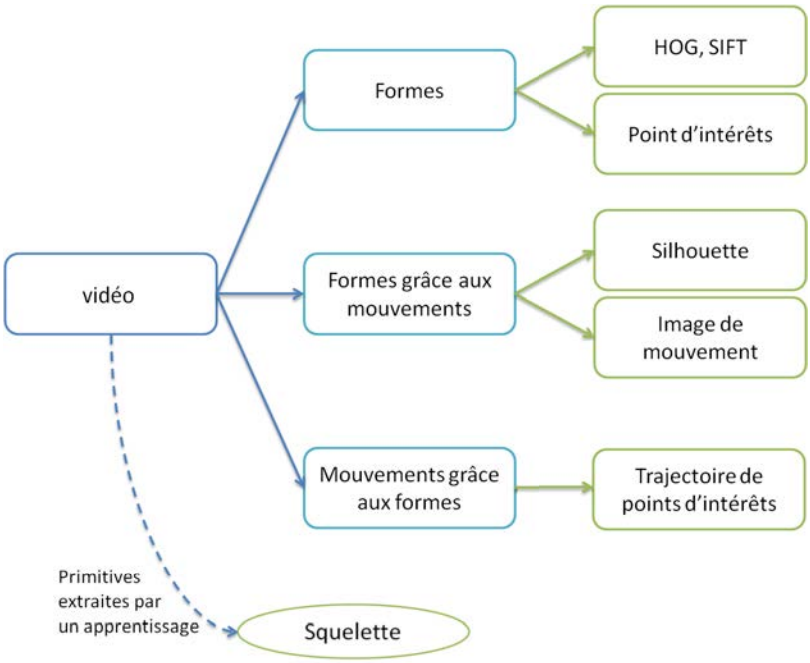


FIGURE 8 – Les primitives de reconnaissances d'actions

n'utilisent pas explicitement des primitives. De même, la section 16 introduit des systèmes dans lesquels les primitives sont construites sur les résultats d'un premier algorithme de prédiction.

Systèmes *à primitives*

10 Comparer les informations

10.1 Primitives et taxonomie des systèmes vis à vis de la structure de l'extraction des primitives

Une primitive est une information spécifique extraite d'une vidéo. Par exemple, si f est une fonction qui à une vidéo $\mathcal{V}$ associe une valeur alors $f(\mathcal{V})$ peut être considérée comme une primitive.

Il existe alors une différence immédiate entre les systèmes qui à partir des vidéos extraient des vecteurs de primitives et ceux qui extraient des ensembles de primitives. Dit autrement, l'extraction d'information au sein du système peut prendre la forme d'une fonction ϕ telle qu'à une vidéo $\mathcal{V}$ est associé un point $\phi(\mathcal{V}) = (o_1, ..., o_D)$. Mais, cette extraction peut aussi prendre la forme d'une fonction ψ telle qu'à une vidéo $\mathcal{V}$ est associé un

ensemble $\psi(\mathcal{V}) = \left\{o'_1, ..., o'_{D_\mathcal{V}}\right\}$.

La nuance peut paraître dérisoire mais il suffit d'essayer d'appliquer l'algorithme *NN* dans les deux situations pour s'apercevoir d'une différence : *NN* attend des points et non des ensembles.

Aussi, on se propose de classer chaque système selon la façon dont il utilise l'information extraite en trois classes :

- *les systèmes globaux :* l'extraction d'information aboutit à représenter chaque vidéo comme un point (d'un espace commun à toutes les vidéos).
- *les systèmes irréguliers :* l'extraction d'information aboutit à représenter chaque vidéo comme un ensemble d'objets.
- *les systèmes image par image :* l'extraction d'information aboutit à représenter chaque image comme un point mais chaque vidéo comme un ensemble.

Par exemple, un système qui extrait un mot symbolique de chaque image et qui utilise ensuite un algorithme de type *HMM* est classé dans *image par image*, alors qu'un système qui extrait un mot dans certaines images seulement puis fait voter ces mots est classé dans *irrégulier* car la structure de l'algorithme n'utilise pas le séquencement entre images. Par contre, un système qui consisterait à extraire une (et une seule) image clé est *global* car la représentation de la vidéo est directement un point.

Un des intérêts de cette décomposition est qu'elle est grossièrement corrélée avec les contextes d'apprentissages.

- les systèmes globaux sont majoritairement associés à de la classification de points et donc à des algorithmes comme *NN* ou *SVM*.
- les systèmes image par image sont majoritairement associés à des algorithmes de type *HMM*.
- les systèmes irréguliers sont majoritairement associés à de la classification d'ensembles.

10.2 Technique de quantification

La sous section précédente associe *les systèmes irréguliers* à des contextes de classification d'ensembles. Cependant, tous les algorithmes de classification d'ensembles associés aux systèmes standards traitent la classification d'ensembles discrets (c'est à dire d'ensemble de mots sur un dictionnaire M) et non pas d'ensembles continus (à ma connaissance, le seul système proposant une classification d'ensembles continus est [14]).

Or, l'extraction de primitives des systèmes irréguliers conduit naturellement à des ensembles continus. Il est donc nécessaire d'utiliser la technique dite de *quantification* consistant à extraire des mots à partir d'un ensemble de primitives. La technique de quantification permet de passer d'un ensemble de primitives à un ensemble de mots et donc d'utiliser des algorithmes de classification d'ensembles comme *ISM* (*Implicite Shape Model* section 9).

10.2.1 K moyennes

Dans sa forme simple, la quantification consiste à regrouper des primitives considérées comme proches. L'ensemble des primitives passe ainsi d'un ensemble continu (ou de très grand cardinal) à un ensemble discret : le vocabulaire associé aux primitives (i.e. l'ensemble de mots M). Dans une quantification, les primitives ne sont regroupées que vis à vis de leur valeur et non vis à vis de leur processus d'extraction. Ainsi, plutôt que de détecter des similarités par la présence de valeurs similaires *au même endroit* (systèmes globaux), on détecte des similarités par la présence de valeurs similaires dans l'espace quantifié des primitives.

La méthode la plus simple pour construire ce vocabulaire consiste à regrouper les primitives en un nombre fixé de mots en fonction des distances. Cela conduit naturellement au problème des K moyennes, qui étant données $\lambda_1, ..., \lambda_\Omega$ des primitives de $\mathbb{R}^\mathcal{U}$, et K un entier (>0), consiste à trouver K points $\mu_1, ..., \mu_K \in \mathbb{R}^\mathcal{U}$ tels que la somme des distances des points λ_ω aux points μ soit minimale c'est à dire : $\min_{\mu_1,...,\mu_K \in \mathbb{R}^\mathcal{U}} \left(\sum_\omega \left(\min_{k \in [1,K]} \|\lambda_\Omega - \mu_k\|_2^2 \right) \right)$. Comme ce problème est NP-complet (quand K est un paramètre même si $\mathcal{U} = 2$ [62] et quand $\mathcal{U}$ est un paramètre, même si $K = 2$ [30]), on se contente d'obtenir une solution approchée en appliquant *l'algorithme des K moyennes* qui consiste à répéter un certain nombre de fois la succession d'opérations suivantes : trouver à $\mu_1, ..., \mu_K$ fixés, le plus proche voisin de chaque point λ_ω dans $\mu_1, ..., \mu_K$; puis ; placer chaque μ_k au centre de l'ensemble des points dont il était le plus proche voisin. Typiquement le pseudo code est le suivant :

```
//q : l'ensemble des primitives qu'on souhaite quantifiées
//c : des centres à améliorer
//chaque primitive sera associée à son centre le plus proche
c = Kmoyenne(q,c)
  allouer un tableau v
  //c[v[n]] est le centre le plus proche de q[n]
  changement = vrai;
  tant que changement == vrai
    changement = faux;
    pour tous n
      pour tous k
        si ||q[n] - c[k]|| < ||q[n] - c[v[n]]||
          v[n] = k;
          changement = vrai;
  allouer un tableau s
  //s[k] est le nombre de points q[n] de plus proche centre c[k]
  pour tous k
    pour tous n
      si v[n]==k
```

```
        c[k] += q[n];
        s[k]++;
    c[k] = c[k]/(s[k]+1);
  retourner c;
```

L'algorithme *des K moyennes* converge très rapidement en pratique et si besoin un nombre fixé à l'avance d'itérations peut être utilisé (le nombre maximal d'itérations avant d'arriver à un optimum local de l'algorithme ne semble pas clairement établi dans la littérature).

Par contre, cet algorithme suppose d'initialiser des points μ. Des expérimentations [1] soutiennent que l'initialisation est importante. Une procédure standard [1] consiste à tirer le prochain point μ parmi les λ avec une probabilité décroissante vis à vis de la distance aux points μ déjà tirés. Pour obtenir une procédure déterministe, il est possible de prendre μ_1 comme le centre du nuage puis μ_2 comme le point le plus éloigné du centre (donc μ_1), puis μ_3 comme le point le plus éloigné des deux précédents et ainsi de suite.

Cette algorithme des K *moyennes* permet de construire les K centres. La quantification à proprement parler consiste (dans sa forme simple) à associer chaque primitive à son centre le plus proche. Ainsi, chaque primitive devient un mot dans $[1, K]$. Chaque mot représente la cellule de Voronoi d'un centre particulier. L'ensemble des cellules représente le vocabulaire M.

Une vidéo $\mathcal{V}$ dont on extrait l'ensemble $\lambda_1, ..., \lambda_\Omega$ de primitives est alors transformée en un ensemble $i_1, ..., i_\Omega$ de mots (où μ_{i_ω} est le plus proche voisin de λ_ω dans μ) et éventuellement en le point $x \in \mathbb{R}^K$ tel que x_k est le nombre de primitives λ_ω telles que $i_\omega = k$. L'ensemble de ce processus popularisé dans [81] est appelé la méthode *sac de mots* (*BOW* pour *Bag Of Words*) et constitue le schéma de la majorité des systèmes irréguliers standards. De plus, même si le point x n'est pas explicitement construit dans les méthodes de type *ISM*, il est équivalent de manipuler des ensembles de mots localisés et un point du moment qu'on définit une bijection de l'un à l'autre (comme discuté en section 6).

10.2.2 Codage et agglomération

Il est possible de généraliser le processus *BOW* précédemment présenté en introduisant deux étapes : le codage et l'agglomération.

Codage : Le codage consiste, étant donné le vocabulaire M (i.e. les K centres μ) et la primitive λ_ω, à construire un vecteur θ_ω de taille K tel que θ_k mesure la ressemblance de λ avec μ_k.

Codage dur : La version historique dite *codage dur* se modélise alors par un vecteur θ où tous les θ_k sont nuls sauf un θ_k qui vaut 1 pour le k tel que μ_k est le plus proche voisin de λ_ω parmi les $\mu_1, ..., \mu_K$.

Codage doux : Le codage doux consiste par opposition au *codage dur* à avoir un vecteur θ où θ_k est décroissant avec la distance entre λ_ω et μ_k par exemple [87] $\theta_k = \exp\left(-\alpha \left\|\lambda_\omega - \mu_k\right\|_2^2\right)$

Codage creux : Le codage creux consiste à chercher un compromis entre le nombre de cases non nulles dans θ et la perte d'information commise en remplaçant la primitive λ_ω par θ (connaissant les vecteurs μ), par exemple θ est la solution de [100] $\min_\theta \ \|\theta\|_0 + \beta \left\|\lambda_\omega - \sum_k \theta_k \mu_k\right\|_2^2$

Agglomération : L'agglomération consiste à partir de l'ensemble des vecteurs θ_ω provenant du codage des primitives λ_ω à construire un unique vecteur de taille K encodant l'ensemble des primitives (c'est à dire la donnée x).

Agglomération additive : La version historique correspond à sommer les θ : $x = \sum_\omega \theta_\omega$. Alors x_k contient $\sum_\omega \theta_{\omega,k}$ qui correspond au nombre de fois qu'une primitive de plus proche voisin μ_k parmi les μ a été extraite. On retrouve ainsi exactement le point associé à l'ensemble des mots correspondant (comme dans la section 6).

Agglomération maximale : Cependant, il est possible de considérer d'autres agglomérations comme $x_k = \max_\omega \theta_{\omega,k}$ [100].

Seule la combinaison codage dur et agglomération additive conduit sémantiquement à une méthode d'élection. Les autres, plus efficaces expérimentalement [100, 2], conduisent à une classification de points mais sémantiquement différente de celles des systèmes globaux.

10.3 Taxonomie des systèmes vis à vis de la nature de l'extraction des primitives

La classification précédente des systèmes de reconnaissance d'actions dépend de la façon dont l'information est structurée lors de son extraction à partir de la vidéo. Schématiquement, cela consiste à se demander si le système extrait un ensemble structuré d'informations (conduisant à représenter la vidéo comme un point et donc à appliquer un *SVM*) ou s'il extrait un ensemble non structuré d'informations (conduisant à représenter la vidéo comme un ensemble d'éléments qui après quantification devient un ensemble de mots sur lequel on peut appliquer *ISM* ou *SVM*).

Indépendamment, une autre question se pose. Elle consiste à se demander si le système effectue explicitement la recherche de certaines informations ou si l'information pertinente est sélectionnée par l'apprentissage lui même.

Pour répondre à cette question, on propose une autre taxonomie (simultanée à la précédente) :

- *les systèmes à primitives :* des informations spécifiques sont recherchées dans les vidéos
- *les systèmes en un seul tenant :* L'apprentissage sélectionne les informations à rechercher. Dit autrement, le système utilise fonctionnellement les vidéos brutes.
- *les systèmes étagés :* les primitives ne sont pas extraites de façon déterministe des vidéos brutes mais des prédictions d'autres algorithmes d'apprentissage.

Le continuum entre les *systèmes à primitives* et les *systèmes en un seul tenant* est présenté dans les section 14 et 15. Enfin les *systèmes étagés* sont présentés en section 16. Les sections 11 à 13 présentent les *systèmes à primitives*.

Plus précisément, la principale caractéristique des *systèmes à primitives* est que l'on peut décrire leurs primitives qui sont les objets des trois prochaines sections.

11 Extraction de formes

Le premier type de primitives de la littérature vise à faire ressortir les formes. L'intérêt des formes est qu'elles sont généralement plus invariantes que la couleur (qui correspond à l'information brute) pour les objets qu'on cherche à reconnaître.

Par exemple, deux voitures peuvent avoir des couleurs très différentes. Ainsi, pour apprendre à reconnaître la classe voiture, il faut soit avoir une base d'apprentissage suffisamment riche pour avoir des exemples de voiture de toutes les couleurs soit trouver une solution pour qu'une voiture d'une nouvelle couleur soit bien reconnue. Une solution est de ne tenir compte que de la forme de la voiture. Cette solution est ici très pertinente car une voiture est un objet rigide dont la forme est très caractéristique. Bien entendu, dire que la couleur n'est pas très pertinente pour la reconnaissance n'est vrai que dans certains contextes. L'exemple opposé est celui de l'identification où l'objectif est d'apprendre à reconnaître une voiture parmi d'autres (afin de ne pas les confondre par exemple), situation où la couleur joue un rôle plus que prépondérant.

11.1 Descripteurs de voisinage

Une avancée importante de la vision par ordinateur consiste à modéliser le gradient d'intensité dans les images. Une méthode emblématique est celle dite des histogrammes des gradients orientés (*HOG* pour *Histogram of Oriented Gradient*) [29]. Cette méthode permet de faire ressortir l'informa-

tion de forme contenue dans un patch. Vu autrement, cette méthode permet de décrire les formes du voisinage du centre du patch.

Dans [29], l'orientation et la norme du gradient sont calculées au niveau de chaque pixel d'un patch de 8x8 pixels. Les orientations des différents gradients sont quantifiées de façon fixe en 8 orientations (0°, 45°, 90°, 135°, 180°, 225°, 270°, 315°). Chaque patch est représenté par un histogramme de 8 colonnes (chaque colonne est associée à une des orientations). La valeur de chaque colonne est la somme des normes des gradients ayant l'orientation correspondante.

Mathématiquement si f est une fonction *suffisamment régulière* de $\mathbb{R}^D$ dans $\mathbb{R}$, en tous points $u \in \mathbb{R}^D$, il existe un unique vecteur ∇f_u (le gradient) tel que pour tous vecteurs $\delta \in \mathbb{R}^D$, la propriété $f(u+\delta) - f(u) = \delta \nabla f_u + o(\delta)$ est vérifiée. Dans le contexte d'une image [29] $\mathcal{I}$, $D = 2$ et $u = (i,j)$ est un pixel et $\mathcal{I}(i,j)$ est l'intensité (en niveau de gris) associée au pixel (i,j). De plus, le gradient $\nabla \mathcal{I}_{(i,j)}$ est remplacé dans [29] par $(\mathcal{I}(i-1,j) - \mathcal{I}(i+1,j), \mathcal{I}(i,j-1) - \mathcal{I}(i,j+1))$. Le calcul du descripteur d'un patch s'effectue donc via le pseudo code :

```
//I : patch de taille 8x8 pixel
descripteur=hog(I)
  H[1..8] = 0
  pour tous i
    pour tous j
      di=I[i-1][j]-I[i+1][j]
      dj=I[i][j-1]-I[i][j+1]
      norme = sqrt(di*di + dj*dj)
      arg = arg(di,dj)
      orientation = ceil(arg/45)
      H[orientation] += norme
  retourner H
```

Dans [99], la boite englobante de la personne d'intérêt est calculée sur toute la vidéo et redimensionnée à une image de 64x128 pixels (elle peut être extraite par un détecteur de personne éventuellement aidé par un suivi d'objet ou par des techniques spécifiques de traitement de vidéos présentées dans la section suivante). La boite englobante dans chaque image est pavée de patchs 8x8 décrits par leur histogramme *HOG*. L'ensemble des histogrammes dans chaque image est concaténé pour former un descripteur d'image. Puis, l'ensemble des descripteurs d'image de la vidéo sont concaténés pour former un descripteur de vidéo $x_{t,i}$. Un *SVM* est appliqué sur des fenêtres à moitié chevauchantes de ce vecteur vis à vis du temps. Chaque fenêtre conduit à un vote et l'agglomération des votes conduit à la classe prédite.

Cette méthode est plutôt *globale* car elle aboutit à l'extraction d'un vecteur x à partir d'une vidéo $\mathcal{V}$. Elle n'est cependant pas emblématique des méthodes globales car la taille du vecteur x dépend de la taille de la vidéo (cependant, ce n'est pas sur le vecteur x que le *SVM* est appris mais sur

des fenêtres de x de taille constante, ainsi du point de vue de l'apprentissage c'est une méthode globale).

Le descripteur *HOG* peut s'étendre en 3D [53]. L'extension du gradient en 3D est triviale. La quantification des orientations en 2D peut quant à elle être assimilée à une approximation du cercle par un polygone [53]. Suivant cette idée, une quantification utilisant sur l'approximation de la sphère par l'icosaèdre (le polyèdre de Platon avec le plus de face) est proposée dans [53].

11.2 Extraction de points d'intérêts

Le *HOG* est un descripteur de voisinage. Ainsi, étant donnée une image, il convient de choisir des points puis de décrire leur voisinage.

Typiquement, pour obtenir une représentation globale, il faut que chaque point choisi soit défini de façon robuste vis à vis de l'image pour qu'on puisse, étant donné deux images, coupler les points des deux images un à un. C'est le cas dans [29] où pour chaque image de 64x128 pixels, le système extrait 128 points sur une grille régulière de sorte que les voisinages de 8x8 pixels forment un pavage de l'image.

Alternativement, pour conserver un maximum de l'information présente dans l'image, on pourrait vouloir extraire les meilleurs points de l'image, la qualité d'un point étant la spécificité de son voisinage. Par exemple, on voudrait les K meilleurs points ou tous les points dont le voisinage a une spécificité estimée supérieure à une seuil. Cette alternative est pertinente d'un point de vue de l'information conservée. Malheureusement, d'un point de vue système, avec une telle extraction, il est impossible de prédire où seront les points considérés.

Cela conduit à une extraction *irrégulière* puisqu'il n'est alors plus possible de coupler les points un à un à partir de leur position d'extraction. La seule information pour comparer des points n'est alors plus que l'apparence de leur voisinage (ce qui peut s'effectuer, par exemple, à travers une quantification des apparences).

Cela implique de se donner un critère pour sélectionner des points de façon stable. Cela introduit une autre avancée importante de la vision par ordinateur : la technique des points d'intérêts.

11.2.1 Détection des points

Une des conditions pour qu'une méthode d'extraction de points d'intérêts soit pertinente est que les points extraits soient peu nombreux mais surtout qu'ils soient invariants vis à vis de certaines transformations de l'image. Typiquement, il est nécessaire que les points d'intérêts soient robustes à de petites variations d'illuminations, des rotations, des changements d'échelles.

La principale méthode dite des points de Harris [44] pour répondre à ces exigences consiste à extraire des points caractérisés par une configura-

tion locale elle même robuste aux petites transformations. Ces points sont caractérisés par les valeurs des dérivées secondes de l'image. Mathématiquement, la matrice hessienne d'une fonction en un point u est telle que $\mathcal{I}(u+\delta)-\mathcal{I}(u)=\delta\nabla\mathcal{I}_u+\frac{1}{2}\delta\mathcal{H}\mathcal{I}_u\delta+o\left(\delta^2\right)$. Soient α et β les valeurs propres (avec $|\alpha|>|\beta|$) de cette matrice. La variation d'ordre deux d'intensité au niveau du point $u=(i,j)$ peut être quantifiée par $|\alpha_{i,j}|$ et $|\beta_{i,j}|$: un coin est caractérisé par $|\beta_{i,j}|\gg 1$, un bord par $|\alpha_{i,j}|\gg 1\gg|\beta_{i,j}|$ et une zone uniforme par $1\gg|\alpha_{i,j}|$. Ces valeurs sont stables vis à vis des petites transformations globales.

11.3 Description d'une image

Chacun des points extraits représente une zone de l'image supposée être pertinente. Chacune de ces zones peut alors être encodée par un descripteur comme *HOG* : les descripteurs les plus classiques dans ce contexte sont les *SIFT* [59] ou les *SURF* [7] dont le principe général est similaire au *HOG*.

Ainsi, la méthode [29] peut être assimilée à une extraction uniforme de points dont on encode le voisinage avec un *HOG* avec l'avantage d'une taille fixe, alors que la méthode [59] consiste à prendre les voisinages qui paraissent les plus robustes et les plus pertinents avec l'inconvénient d'obtenir une méthode d'extraction irrégulière.

[81] est emblématique de ce type de méthodes : ces descripteurs sont utilisés par la méthode *BOW+SVM* (quantification, construction de l'histogramme des occurrences application d'un *SVM* sur les points correspondants) pour classer les images.

11.4 Description d'une vidéo

En reconnaissance d'objets dans des images, les approches globales et les approches irrégulières sont toutes autant présentes. Ce n'est pas le cas en reconnaissance d'actions car la flexibilité apportée par l'approche irrégulière est très pertinente pour traiter des vidéos de tailles différentes.

[81] s'étend directement à la reconnaissance d'activités dans des vidéos [77]. Les points d'intérêts des vidéos (*STIP* pour *Spatio temporal Interested Point*) sont extraits similairement à ceux des images [77]. Leur voisinage est encodé similairement à *HOG* et l'approche *BOW+SVM* permet directement de classer les vidéos.

12 Extraction de formes grâce aux mouvements

L'extraction de formes de la section précédente est principalement une extraction dans l'image (même si les *STIP* utilisent les images voisines dans le temps). Le deuxième type de primitives utilisées dans des vidéos est l'ex-

traction de formes permise par les mouvements c'est à dire l'évolution des images dans le temps. Ce type de primitive est donc spécifique aux vidéos.

12.1 La silhouette

La principale information de forme extractible grâce à la vidéo est la silhouette de la personne d'intérêt. C'est à dire la segmentation (pixel à pixel) de la personne dans la vidéo ou dans une version moins forte la boite englobante de la personne d'intérêt.

L'ensemble des silhouettes de la vidéo (une par image) peut alors être projeté dans une seule image [13] (pour se ramener à de la reconnaissance d'objets dans une image) ou considérer comme un volume 3D descriptible à l'aide d'un certain nombre d'intégrales semblables aux moments du volume [11].

Ce système [11] (mais aussi [13]) est emblématique des systèmes *globaux*. Ce système extrait Φ le volume 3D correspondant à la segmentation de la silhouette dans la vidéo $\mathcal{V}$. Il le transforme en un autre volume Φ' qui capture la saillante des points de Φ. Il construit alors un vecteur x de taille fixe tel que $x_{\alpha,\beta,\gamma}$ est $\int\int\int \left(\Phi'(i,j,t)\, i^\alpha j^\beta t^\gamma\right) \mathrm{d}i\mathrm{d}j\mathrm{d}t$. Dit autrement, x est un vecteur dans lequel chaque composante est simplement la valeur d'une primitive certes non triviale mais clairement identifiée. L'apprentissage repose ensuite sur *NN*.

Il est possible d'extraire des primitives de la vidéo segmentée comme dans [49] (ce qui peut conduire à un système *image à image* ou *irrégulier*).

12.2 Estimation la silhouette

La silhouette d'une personne est la segmentation de la personne dans la vidéo. Cette segmentation peut s'obtenir image par image par un apprentissage statistique. Mais, ce dernier problème est potentiellement plus dur que celui de la reconnaissance d'actions. Cependant, cette segmentation peut s'obtenir directement dans certaines vidéos en détectant de façon non supervisée les mouvements.

Il existe principalement deux méthodes pour capturer de façon non supervisée les mouvements dans une vidéo : le calcul de la différence avec l'image précédente et la soustraction du fond.

12.2.1 Différence d'images

Deux images sont typiquement séparées par 0.04 seconde. Dans ces conditions, la lumière ambiante naturelle peut être considérée comme constante. Ainsi, si la caméra est fixe et qu'elle observe un objet fixe, l'objet doit se projeter de la même façon dans l'image courante que dans l'image précédente. Donc, si la caméra est fixe et qu'un pixel a une valeur différente dans deux images successives, c'est que l'objet qui se projette sur ce pixel a bougé. Cette

remarque permet directement de détecter des mouvements dans des vidéos. Certes, la présence de bruit, de changement d'illumination et d'imperfections dans les caméras introduisent des erreurs. Ces erreurs peuvent être modérées par exemple en supprimant les détections isolées et/ou en utilisant un seuil sur la différence de valeur entre deux pixels consécutifs. Bien qu'il persiste toujours des erreurs résiduelles, cette méthode permet directement de segmenter un objet mobile dans une vidéo si sa couleur est différente de celle du fond.

Le principal problème est que si un objet se projette dans le pixel α dans l'image précédente puis dans le pixel β dans l'image courante alors à la fois α et β ont changé de valeur. Ainsi, cette méthode segmente l'objet mobile et *sa trace* dans l'image. On parle d'effets fantôme.

12.2.2 Soustraction de fond

La soustraction de fond permet une segmentation sans effet fantôme de l'objet. L'idée est de remarquer que si un objet se projette dans le pixel α dans l'image précédente puis dans le pixel β dans l'image courante alors à la fois α et β ont changé de valeur mais que β prend une valeur nouvelle alors que α retourne à sa valeur *standard*.

L'idée de la soustraction de fond est donc d'apprendre l'image de fond de la vidéo c'est à dire l'image qu'on aurait s'il n'y avait pas l'objet mobile.

Dans l'image courante, il suffit donc de faire la différence entre l'image courante et l'image du fond pour détecter les pixels qui ne sont pas du fond c'est à dire qui appartiennent à l'objet mobile (avec toujours l'hypothèse que l'objet mobile a une couleur différente du fond). Cette méthode permet ainsi une segmentation directe d'un objet mobile dans une vidéo.

Bien entendu, il reste toujours une erreur résiduelle car si la vidéo est longue, l'hypothèse de constance de l'illumination ne tient plus. De plus, l'existence de bruit et de défaut reste un problème dans cette méthode. Enfin, comme on *apprend* l'image du fond, on est dans un contexte d'apprentissage avec tout ce que cela implique (notamment le *no free lunch theorem*). Cependant, on est dans un contexte d'apprentissage en dimension 1 si on apprend la luminosité standard des pixels indépendamment. Dans ces conditions, la simple moyenne est justifiée surtout si la présence de l'objet mobile est un événement rare dans la vidéo [34]. Cette moyenne peut se compléter par l'estimation d'une variance utilisable à travers la distance de *Mahalanobis*.

Le pseudo code de ce calcul du fond est :

```
imgfond = calculFond(video)
   pour i et j
      imgfond[i][j] = 0
      pour t
         imgfond[i][j]+=video[t][i][j]
      imgfond[i][j] = imgfond[i][j]/T
```

```
   retourner imgfond
```
La segmentation s'effectue ensuite suivant le pseudo code :
```
segm = soustractionFond(video,imgFond)
   pour i,j et t
      segm[t][i][j] = |video[t][i][j]-imgFond[i][j]|>seuil
   retourner segm
```

13 Extraction de mouvements grâce aux formes

Le troisième type de primitives consiste à estimer finement les mouvements. La section précédente permet de détecter des mouvements mais des mouvements globaux c'est à dire qu'on sait ce qui a bougé entre l'image courante et l'image précédente. Ici, l'objectif est d'estimer des mouvements fins : on veut savoir pour chaque point quel est son mouvement entre l'image courante et l'image précédente.

13.1 Extraction de trajectoire de points d'intérêts

Une avancée majeure de la reconnaissance d'actions est d'estimer ces mouvements locaux en suivant un point d'intérêt à travers plusieurs images.

Cela est possible car les points d'intérêts sont extraits avec des méthodes robustes aux petites transformations, et typiquement aux transformations entre une image et l'image précédentes. Ainsi, la position et l'aspect d'un point d'intérêt mais aussi l'aspect de son voisinage ne devraient que peu changer d'une image sur l'autre. Cela permet d'identifier (bien qu'il puisse y avoir des erreurs) un point à travers un ensemble d'images et donc d'extraire sa trajectoire dans le temps.

Ces primitives ont contribué à un saut majeur dans les performances des systèmes de reconnaissance d'actions [83, 92]. Une explication possible est que ces trajectoires sont corrélées aux gestes et donc porteuses d'information spécifiquement pertinente pour la reconnaissance d'activités. L'expérience de [51] montre d'ailleurs que nous sommes capables de reconnaître les gestes d'un autre humain uniquement à partir du mouvement d'un certain nombre de points à la surface de ce dernier.

13.2 Flot optique

Alternativement à chercher la trajectoire de certains points il est possible de chercher à calculer la trajectoire de tous les points simultanément.

Cependant, comme on cherche à calculer le mouvement de tous les points, on doit considérer des points dont l'aspect et le voisinage est quelconque. Si par exemple une personne portant un habit uniforme se déplace, alors pour chacun des points à l'intérieur de l'habit, il n'est pas possible de distinguer

grâce à l'aspect, le point dans l'image suivante : il peut être associé à chacun des autres points de l'intérieur de l'habit.

Ainsi, le mouvement des points quelconques doit être guidé par les mouvements des points remarquables. Typiquement, l'association du pixel α de l'image précédente avec le pixel β de l'image courante dépend à la fois de l'adéquation des couleurs de α et de β mais aussi de l'adéquation des associations des pixels voisins de α et de β (dans leur image respective). Par exemple, la méthode du flot optique [6] se base sur un terme d'attache aux données et un terme de régularité. Le terme d'attache aux données ρ pénalise la distance entre deux couleurs. Le terme de régularité ϱ pénalise la différence entre deux déplacements. Le flot optique ω optimal est celui qui minimise la pénalité totale et le calcul du flot se ramène à un problème d'optimisation de $\int\limits_{\text{pixel}} \left(\rho\left(I_{t+1}\left(u\right), I_t\left(u+\omega\left(u\right)\right)\right) + \sum\limits_{v \ voisin \ u} \varrho\left(\omega\left(u\right), \omega\left(v\right)\right) \right) \mathrm{d}u$ vis à vis de ω.

13.3 Descripteurs de trajectoires

Tout comme les points d'intérêts, après avoir extrait des trajectoires, il convient de décrire à la fois la forme de la trajectoire mais aussi le voisinage du point le long de la trajectoire.

Dans [83, 92], les trajectoires sont coupées et tronquées afin d'être de taille fixe. Ensuite dans [92], le voisinage de chaque trajectoire est décrit par la concaténation des histogrammes *HOG* et d'histogrammes liés au flot optique. Cela permet d'encoder à la fois l'aspect visuel et le mouvement. La forme de la trajectoire est quant à elle normalisée en extrayant le vecteur des vitesses. L'ensemble des descripteurs est ensuite quantifié.

Dans [83], l'aspect est décrit pas la moyenne des *HOG* le long de la trajectoire et la forme est quantifiée image à image (le mouvement d'un point entre deux images successives est quantifié). Puis un vecteur capturant la distribution des co occurrences des mouvements est calculé. Ces vecteurs sont de nouveau quantifiés. Enfin, un descripteur compte le nombre de trajectoires de chaque type autour de chaque trajectoire donnée afin de capturer les interactions entre trajectoires.

Rapprocher l'apprentissage des données brutes

14 Les systèmes *en un seul tenant*

Les systèmes *à primitives* considèrent des primitives spécifiques afin de prendre une décision. On s'intéresse ici à des systèmes-algorithmes qui ne

considèrent pas explicitement de primitives.

14.1 *Machine de Boltzmann réduite*

L'exemple le plus emblématique est la classe des algorithmes de type *Machine de Boltzmann réduite* (*DL* pour *Deep Learning*) [76, 42, 43] (appliqué à la reconnaissance d'actions par exemple dans [3]).

Lors de l'entraînement de ce type d'algorithme, l'exemple brut est donnée en entrée à une pyramide de nœuds et la classe à la dernière couche de nœuds de la pyramide. Des règles paramétrées par des variables définissent l'influence entre les nœuds de deux couches successives. L'apprentissage peut s'effectuer à l'aide d'une descente de gradient qui trouvent une affectation des variables pour minimiser localement l'erreur d'apprentissage.

À travers l'empilement des couches de nœuds cet algorithme apprend en un sens des primitives hiérarchiquement pertinentes. Cependant, du point de vue utilisateur l'algorithme peut s'utiliser comme une simple boite noire quasiment indépendamment du type d'exemples.

Par exemple, l'implémentation *Torch-convnet*[1] permet d'apprendre à classer des objets dans des images en ne précisant que les tailles (profondeur et largeur) de la pyramide.

Dans le cadre de la thèse, ce type d'algorithme n'a pas été particulièrement étudié. Cependant, il parait important de mentionner cette classe d'algorithmes très différentes des algorithmes à primitives.

14.2 Arbre de décision et *forêt de Hough*

Un autre exemple réside dans les méthodes de type *forêt de Hough* (*HF* pour *Hough Forest*) [102, 101] ou forêt d'arbres de décision (*C4.5* [70]).

La méthode de la *forêt de Hough* consiste à se donner une probabilité sur des fonctions binaires *simples* de patchs de vidéos brutes. Dit autrement, on se donne un ensemble de fonctions qui prennent en entrée un patch de vidéos brutes (par exemple dans [102], un patch 8x8x5 pixels c'est à dire un patch 8x8 pixels dans 5 images successives) et on se donne une règle pour tirer aléatoirement une fonction dans cet ensemble.

Cette probabilité est la seule information spécifique au problème. Tout le reste de la méthode est identique à tous les problèmes.

Structure de l'algorithme : L'apprentissage consiste à trouver une quantification des données brutes (i.e. un processus qui transforme les données brutes en ensemble de mots) adaptée aux votes *ISM*. Plus précisément, l'apprentissage consiste à trouver des quantifications. Chaque quantification prend la forme d'un arbre binaire de fonctions binaires. L'ensemble des quantifications prend la forme d'une forêt.

1. http ://torch.cogbits.com

Cependant, chaque arbre binaire est appris indépendamment : l'apprentissage de plusieurs arbres accroît simplement la stabilité du système [86].

Chaque arbre est constitué de nœuds et de feuilles (comme dans la section 4 et la figure 6). Chaque nœud contient une fonction binaire simple. Chaque feuille correspond à une partie de l'ensemble des patchs : chaque patch suit un chemin dans l'arbre (en fonction de la valeur de la fonction binaire sur le patch) et atteint une feuille et l'ensemble des patchs atteignant une feuille forme une partie de l'espace des patchs. Le pseudo code permettant de déterminer la feuille associée à un patch (et donc la partie de l'espace associé à un patch) est :

```
numFeuille = chemin(patch,arbre)
  si arbre est une feuille
    retourner arbre.numéro
  //sinon arbre est un noeud
  //un noeud possède une fonction binaire et deux fils
  si arbre.fonctionBinaire(patch) == 1
    retourner chemin(patch,arbre.filsDroit())
  sinon //arbre.fonctionBinaire(patch) == 0
    retourner chemin(patch,arbre.filsGauche())
```

La présence d'une forêt et non d'un seul arbre signifie simplement que de chaque patch on extrait un mot par arbre (le nombre de mots extraits est donc le nombre de patchs utilisés multiplié par le nombre d'arbres).

Utilisation en test : Chaque feuille correspond donc à un mot m et l'ensemble des feuilles au vocabulaire M.

Ainsi, une fois la forêt apprise, chaque patch passe dans chaque arbre, tombe sur une feuille par arbre et est donc transformé en un mot par arbre. Étant donné l'ensemble des mots, le système se comporte exactement comme l'algorithme *ISM* présenté en section 9.

On peut remarquer la proximité entre l'approche de la *forêt de Hough* et une approche plus classique d'extraction de primitives puis quantification puis *ISM*. Dans un système utilisant *ISM*, on décrit les patchs par exemple avec un *HOG* puis on les quantifie avec un algorithme de K moyennes, puis on les fait voter. Dans *forêt de Hough*, on fait passer les patchs dans l'arbre, les fonctions binaires choisies font office de description et le regroupement des éléments touchant une même feuille de quantification, puis on les fait voter. Cette proximité rend discutable la distinction entre les *systèmes à primitives* et les *systèmes en un seul tenant*. Cependant, cette distinction est pertinente d'une part parce que les performances de *forêt de Hough* surpassent celles des systèmes utilisant *ISM* [40] et sémantiquement car la structure de *forêt de Hough* permet de choisir les mots en tenant compte des données, des classes connues à l'apprentissage et des votes qu'on va utiliser là où ceux d'un système utilisant *ISM* ne dépendent que des données.

Apprentissage : Un arbre est appris de manière incrémentale : pour choisir la fonction de la racine, on tire un certain nombre de *fonctions binaires simples*, on mesure l'entropie qui serait associée aux votes *ISM* si on séparait en deux les patchs à l'aide de chacune des fonctions et on prend la fonction qui minimise l'entropie. Le processus est itéré pour chaque fils (racine de leur sous arbre) jusqu'à atteindre un critère d'arrêt. L'apprentissage correspond au pseudo code suivant :

```
//Q un ensemble de H patchs
//Y l'action correspondante au patch
//D le décalage temporel avec le centre de l'action
//G un générateur aléatoire de fonction binaire
//nombre de tirages en chaque nœud
//seuil : critère d'arrêt de la décomposition d'une partie
arbre = apprentissage(Q,Y,D,G,R,seuil)
   entropieMin = infini
   fMin = null
   pour r de 1 à R
      f = G()
      estimer P(y,t|f(q)==-1) et P(y,t|f(q)==1) avec Q,Y,D
      ent = entropie(P)
      si ent< entropieMin
         entropieMin = ent
         fMin = f
   séparer Q,Y,D en Qm,Ym,Dm et Qp,Yp,Dp en fonction de fMin(q)
   si entropieMin < seuil
      filsGauche = feuille
      filsDroit = feuille
   sinon
      filsGauche = apprentissage(Qm,Ym,Dm,G)
      filsDroit = apprentissage(Qp,Yp,Dp,G)
retourner Arbre(fMin, filsGauche, filsDroit)
```

L'intérêt de cette extraction implicite de mots est d'introduire l'entropie de P la probabilité d'un couple classe décalage au centre sachant un mot. P est estimé à l'aide du code :

```
   pour h de 1 à H
      P[y[h],D[h],f(Q[h])]++
      Ptot[f(Q[h])]++
   pour h de 1 à H
      P[y[h],D[h],1] /= Ptot[1]
      P[y[h],D[h],-1] /= Ptot[-1]
```

La calcul de l'entropie d'une probabilité P' sur Ω s'effectue suivant la définition $\sum_{\omega\in\Omega} -P'(\omega)\log_2(P'(\omega))$. La principale propriété de l'entropie est qu'elle est maximale pour P' uniforme et minimale s'il existe ω tel que $P'(\omega) = 1$

Dans le cas de votes (ici $P = P'$), l'entropie est maximale si les mots votent indistinctement pour chaque classe (c'est à dire s'ils n'amènent aucune information) et minimale si un mot est systématiquement corrélé à un couple classe position. Cette quantité estime donc l'inutilité des votes. Ainsi, cet algorithme d'apprentissage cherche de façon approchée l'extraction de mots qui, couplée à *ISM*, minimise l'erreur d'apprentissage. En ce sens, ce système est un système *en un seul tenant*, même si la probabilité utilisée pour tirer des fonctions binaires constitue un a priori sur le problème. Typiquement dans [102], les patchs sont préalablement décrits avec un *HOG* (et l'entropie est séparée en une entropie de classes et une entropie de décalage temporel).

Cette probabilité sur les fonctions et le nombre de tirages sont deux paramètres cruciaux de cet algorithme. Typiquement, un trop grand nombre de tirages conduit à sélectionner des fonctions trop spécifiques aux données d'apprentissage et donc décroit la stabilité de l'erreur. Inversement, un trop petit nombre de tirages conduit à des mots peu adaptés et donc à une forte erreur.

Les arbres de décision : L'algorithme d'apprentissage présenté conduit à la *forêt de Hough* si on utilise l'entropie des votes *ISM* et aux algorithmes d'arbres de décision (dont *C4.5* est la forme la plus emblématique [70]) si on mesure l'entropie de la probabilité $P(y|m)$ (probabilité d'une classe sachant un mot).

14.3 Fusion de critères

La *fusion de critères* [90] (*Boosting* en anglais) est un autre exemple de système *en un seul tenant*.

Structure de l'algorithme : La structure de l'algorithme consiste à former un critère de décision *fort* en pondérant des algorithmes de classification *faibles*.

Ces algorithmes de classification *faibles* peuvent être de simples fonctions binaires sur les vidéos. Remarquons que dans la méthode de la *forêt de Hough*, on utilise des fonctions binaires simples de patchs (typiquement un patch fait 8x8x5 pixels) et non de vidéos (typiquement de 64x128x250 pixels). Ici, il s'agit bien de fonctions binaires globales simples des vidéos par exemple des souches de décision (*decision stump* abrégé en *stump* dans le manuscrit).

La *fusion de critères* [90] est aux méthodes globales comme *HOG+SVM* [29] ce que la *forêt de Hough* [101] est aux méthodes irrégulières comme *SIFT+BOW+SVM* [81]. Dans les deux cas, l'apprentissage est rapproché des vidéos brutes.

Utilisation en test : La *fusion de critères* étant en un sens une méthode globale et aboutissant à l'apprentissage d'un critère, il suffit en test d'évaluer

le critère sur la nouvelle donnée.

Apprentissage : L'apprentissage de la *fusion de critères* consiste à former un critère pertinent à l'aide d'une somme pondérée de *stump*. Cependant, l'ensemble de ces *stump* n'est pas représenté explicitement : comme dans l'algorithme de *forêt de Hough*, on se donne une probabilité pour tirer aléatoirement des *stump*.

L'apprentissage est incrémental. Il consiste à chaque étape à choisir la meilleure *stump* (parmi celles tirées) vis à vis de l'erreur d'apprentissage courante pondérée. À l'aide de cette nouvelle *stump*, on réactualise à la fois la somme pondérée de *stump* courante et l'erreur d'apprentissage courante pondérée. Dit autrement, à chaque étape on choisit une fonction binaire globale simple qui sépare bien les exemples qu'on n'arrive pour l'instant pas à séparer et on l'ajoute au critère courant. Le pseudo code correspondant est :

```
//x les N points d'apprentissages
//y les classes correspondantes aux points de x
//G un générateur d'algorithme de classification faible
//nbTirage nombre de tirage à chaque étape
//nbIter nombre d'itérations
classifFort = boosting(x,y,G,R,nbTirage,nbIter)
  initialiser P[1..N] à 1/N
  //P mesure la difficulté des exemples
  initialiser a[1..T]
  pour t de 1 à nbIter
    emin = infini
    hmin = null
    pour r de 1 à nbTirage
      h = G()
      e = somme sur n : P[n]*|y[n]-h(x[n])|;
      //e : erreur d'apprentissage pondérée de h
      si e < emin
        emin = e
        hmin = h
    a[t] = 1/2*log((1-emin)/emin);
    initialise coeff[1..N]
    pour n de 1 à N
     coeff[n] = y[n]==hmin(x[n])?-a[t]:a[t];
     P[n] ∝ P[n] exp(coeff[n]);
    H += a[t]hrmin;
  retourne H;
```

Dans le cas où l'ensemble de tous les algorithmes de classification *faibles* considérés $h_1, ..., h_D$ est petit, on peut explicitement transformer les vidéos

$x_1, ..., x_N$ en les points $x'_1, ..., x'_N$ avec $x'_n = (h_1(z_n), ..., h_D(z_n))$. Appliquer un SVM, sur les points $x'_1, ..., x'_N$ revient à trouver une pondération des algorithmes de classification faibles qui minimise l'erreur d'apprentissage (cas C grand). Cependant, même dans ce cas, il peut être plus pertinent d'utiliser l'algorithme du *fusion de critères* car ce processus de sélection introduit une régularité (notamment à travers le tirage aléatoire) que n'a pas le SVM. Dit autrement, dans ce cas, un SVM aurait nécessairement une erreur d'apprentissage plus faible mais la *fusion de critères* aurait une erreur plus stable. Ainsi, il n'est pas trivial de savoir lequel des deux aurait la plus petite erreur sur une base de test disjointe de la base d'apprentissage (dans le contexte d'un *problème répondant à un besoin*).

15 Adaptation générique de la dimension

La section précédente présente des systèmes *en un seul tenant* qui apprennent directement sur les données brutes à l'opposé des systèmes *à primitives* qui nécessitent une phase explicite d'extraction de primitives.

Cependant, parmi ces systèmes *en un seul tenant* seul le *deep learning* apprend vraiment directement à partir des vidéos brutes. Par exemple, avec *Hough forest*, il reste intéressant de changer la représentation des patchs (ce qui peut être fait implicitement à travers le choix de la probabilité sur les fonctions binaires simples).

L'objectif de cette section est de présenter quelques systèmes dans le continuum entre les méthodes qui apprennent directement à partir des vidéos brutes et celles qui nécessitent une extraction explicite de primitives, notamment, les grandes méthodes de sélection génériques de dimensions.

L'intérêt de la sélection de dimension est visible dans la borne VC [89] objet de la section 3. La borne supérieure de l'erreur comprend d'une part l'erreur d'apprentissage et d'autre part un terme qui dépend de la dimension VC. Ainsi, diminuer la dimension VC fait décroître le deuxième terme, mais au prix d'une perte d'information qui peut faire accroître le premier. Bien que cette règle ne permette pas de régler un algorithme au cas par cas, pour les *problèmes qui répondent à des besoins*, il est pertinent d'utiliser un algorithme de réduction de dimension parallèlement à un algorithme d'apprentissage.

15.1 Analyse en composante principale

L'analyse en composante principale (ACP) est la méthode standard de réduction de la dimension dans le contexte de la classification de points. Le principe de l'ACP est de trouver la projection de plus forte variance des données $x_1, ..., x_N$ de X vers $x'_1, ..., x'_N$ dans X' de dimension plus faible.

Soit $\mathcal{M}$ la matrice correspondant à la concaténation des $x_1, ..., x_N$. Soit $I_{|X'|}$ la matrice telle que $I_{r,i,j} = 1$ si $i = j \leq |X'|$ 0 sinon. L'ACP consiste à

trouver une matrice orthogonale $\mathcal{O}$ telle que $\left\|\mathcal{M} - I_{|X'|}\mathcal{O}\mathcal{M}\right\|_2^2$ soit minimal.

Le principal problème de l'ACP est qu'elle suppose que l'information est corrélée avec la variance. Si on reprend l'exemple de points qui ont dans leur première composante leur classe et un bruit uniforme de grande amplitude sur toutes les autres, l'ACP va immédiatement détruire toute l'information pertinente. Par contre l'avantage de l'ACP est de permettre une réduction de la dimension et non de sélection. Dans la sélection, on conserve les dimensions sélectionnées à l'identique et on supprime directement les dimensions non sélectionnées. Dans la réduction, on transforme les données (par exemple en formant des combinaisons linéaires de dimensions). Typiquement une réduction de dimensions selon un critère de variance consisterait à minimiser $\left\|\mathcal{M} - I_{|X'|}\mathcal{S}\mathcal{M}\right\|_2^2$ où $\mathcal{S}$ est une matrice de permutation alors que l'ACP consiste à minimiser $\left\|\mathcal{M} - I_{|X'|}\mathcal{O}\mathcal{M}\right\|_2^2$ où $\mathcal{O}$ est une matrice orthogonale.

En fonction du problème, l'inconvénient de l'ACP peut surpasser son avantage et il peut devenir plus pertinent d'effectuer une sélection de dimension en tenant compte des classes.

15.2 Sélection de dimensions et de mots

15.2.1 Sélection a priori

La sélection a priori (dans un contexte de sélection de mots) consiste à n'utiliser que certains des mots du vocabulaire en fonction d'un critère ϕ calculable à partir des données d'apprentissage (par exemple une entropie comme dans le contexte de la *forêt de Hough*). Le code permettant de passer d'un dictionnaire de M mots à un dictionnaire de M' mots est :

```
//M dictionnaire de départ
//taille : taille du dictionnaire en sortie
//critère de qualité a priori d'un mot
M' = selectionAPriori(M,phi,taille)
  soit Q une file de priorité
  pour m de 1 à M
    insérer m dans Q avec priorité phi(m)
  insérer dans M' les taille premiers éléments de Q
  retourner M'
```

Le premier avantage de cette sélection est sa simplicité. De plus, cette méthode détecte la présence systématique d'un mot donnant la classe.

Cependant, cette méthode sélectionne les mots indépendamment de leur utilisation. Par exemple, on peut utiliser le critère d'entropie à la fois pour les votes naïfs $P(y|m)$ ou les votes optimisés par un SVM. Or, si l'entropie est une mesure pertinente de la qualité d'un mot vis à vis des votes naïfs, ce n'est pas le cas pour les votes optimisés. Pour cette raison, cette méthode est dite *a priori* par opposition à la sélection *a posteriori* utilisant l'algorithme de classification lui même.

15.2.2 Sélection a posteriori

La sélection a posteriori consiste à sélectionner les D' dimensions (ou mots) à partir des D dimensions (ou mots) de départ qui minimisent l'erreur d'apprentissage ou une estimation de l'erreur totale (une telle estimation peut être construite, par exemple, par la méthode de validation croisée [33]). Cependant, comme cette minimisation n'est pas toujours réalisable, il est classique de minimiser ce critère de façon gloutonne. Cela aboutit au code suivant :

```
//x,y les points et les classes
//e : la procédure d'évaluation de l'algorithme d'apprentissage
//cette procédure retourne une mesure de l'erreur
//elle prend en entrée une base comme x,y
//ou comme x',y avec x'=x privé de certaines dimensions
dim = selection(x,y,e)
   pour n de 1 à N
         xr = []
   pour d de 1 à D
         dim[d] = infini
   pour i de 1 à D'
      ei = infini
      di = null
      pour d de 1 à D tel que dim[d] != infini
         pour n de 1 à N
            xr[n].ajouter(x[n][d])
         eid = e(xr,y)
         si eid < ei
            ei = eid
            di = d
         pour n de 1 à N
            xr[n].supprimer()
      dim[di] = i
      pour n de 1 à N
         xr[n].ajouter(x[n][di])
   retourner dim
```

Le principal inconvénient de cette méthode est la complexité nécessaire à l'apprentissage : notons $f(N, D)$ la quantité de ressources machines nécessaire pour apprendre avec N points de dimension D, alors la quantité de ressources machines nécessaire pour faire une sélection a posteriori de D' éléments est en $O(D' \times D \times f(N, D))$. Cette méthode est donc difficilement acceptable pour D et D' grands alors qu'elle n'est intéressante que dans cette circonstance.

15.2.3 Sélection simultanée

Dans la formulation du SVM (équation (1) de la section 3), le terme $\|w\|_2^2$ force à maximiser la marge entre les deux nuages de points. Cependant, ce terme s'apparente aussi à un simple terme de régularisation complémentant le terme d'attache aux données $\sum_{n=1}^{N} h\left(y^*\left(x_n\right)\left(wx_n+b\right)-1\right)$, avec un coefficient C équilibrant ces deux termes.

Considérer ce terme $\|w\|_2^2$ comme un simple terme de régularisation invite à le remplacer par d'autres régulations possibles comme par exemple $\|w\|_1 = \sum_d |w_d|$ [38].

Typiquement, la dimension VC est la même quelle que soit la fonction de régularisation dans (1). Certes, l'hyperplan qui maximise la marge a des justifications qui lui sont propres [88] mais cela n'exclut pas d'explorer d'autres termes de régularité. Ainsi, la littérature a exploré le choix d'autres formes pour le terme de régularité en fonction du contexte et de la possibilité de résoudre effectivement l'optimisation.

Par exemple, les régularisations L_1 et L_2 ont des avantages et des inconvénients *opposés*.

Une régularisation en norme L_2 force l'unicité de la solution [89], elle permet la commutation de la résolution avec les rotations (la solution du problème après rotation est la rotation de la solution initiale) enfin elle augmente la redondance, ce qui augmente la robustesse à la disparition d'une partie de l'information. Cette redondance s'illustre dans le cas où deux dimensions d,d' des vecteurs x sont identiques, alors $w_d = w'_d$: le terme d'attache aux données est identique pour tous les couples w_d, w'_d de même somme ($w_d + w'_d = cst$) mais pour $w_d + w'_d = cst$, ww est minimal pour $w_d = w'_d$. De même, généralement toutes les dimensions sont utilisées ($|w_d| \neq 0$) car quand $|w_d| \ll 1$ le coût associé à w_d^2 est négligeable.

À l'opposé pour une régularisation en norme 1, si deux dimensions d,d' sont identiques alors la solution *est invariante* pour tous les couples w_d, w'_d avec $w_d + w'_d = cst$ (il y a une infinité de solutions) mais la façon de résoudre ce type de problème conduit généralement à des solutions en coin, c'est à dire avec $w_d \times w'_d = 0$. Cela rend la solution sensible aux rotations des exemples d'apprentissage mais donne un sens aux différentes dimensions. De plus, même si $|w_d| \ll 1$, le coût associé à $|w_d|$ n'est pas négligeable, ce qui implique qu'il peut être préférable de choisir $|w_d| = 0$. Ainsi, la régularisation en norme 1 se concentre sur les dimensions les plus pertinentes aux vues des exemples d'entraînement. Cela rend la solution plus robuste au bruit (mais plus sensible à la perte d'information).

Ainsi, utiliser une norme 1 plutôt qu'une norme 2 permet de sélectionner de façon implicite des dimensions. La *norme* 0 permet de donner un coût explicite à l'utilisation d'une dimension, cependant l'optimisation n'est plus réalisable dans ce cas.

Ces trois méthodes de sélection *a priori*, *a posteriori*, et la méthode de cette sous sous section dite *simultanée* (*embedded* en anglais) forment la taxonomie des méthodes de sélection de dimension. Cependant, un des inconvénients de ces méthodes (que partage l'ACP) est que l'ensemble des dimensions doit déjà être construit.

15.3 Construction de dimensions ou de mots

15.3.1 Sélection vs construction

À l'opposé des méthodes de sélection de dimensions qui travaillent sur des points dont on connaît toutes les dimensions, il existe des méthodes de construction de dimension. D'une certaine manière, la méthode *Hough Forest* est une méthode de construction de dimension.

Vu autrement, la méthode de sélection a priori par entropie suivi de *ISM* consiste à construire les mots (par exemple avec K moyennes) puis extraire ceux qui sont adaptés à *ISM* puis utiliser *ISM*. Alors que, la méthode de la *forêt de Hough* consistent à construire directement des mots adaptés à *ISM* puis utiliser *ISM*.

Sous une contrainte de ressources machines limitées, ces deux méthodes ne sont pas équivalentes. Il semble difficile de choisir chaque fonction binaire dans un apprentissage d'arbres de décision en utilisant à chaque étape le score obtenu par les mots courants dont les votes sont optimisés avec un *SVM*. Aussi, les méthodes de construction (comme la *forêt de Hough*) sont limitées à une construction a priori. Inversement, les méthodes de sélections ne peuvent pas générer l'ensemble de tous les mots possibles ce qui empêche d'appliquer une sélection sur tous les mots possibles. Ainsi, même si virtuellement une méthode de sélection a priori par entropie sur *ISM* partant de l'ensemble des mots devrait aboutir au même résultat que *Hough Forest*, il est plus pertinent dans ce contexte de construire plutôt que de sélectionner.

Afin de donner un exemple éventuellement plus parlant que celui des *Hough forest*, intéressons nous à l'exemple de la construction de super mots introduit dans [57] les *JEP* pour *Jump Emerging Pattern*.

15.3.2 L'exemple des *JEP* :

Dans le contexte de classification d'ensemble de mots, un super mot est un mot qui correspond à la présence d'un ensemble de mots. Or, faire voter un super mot n'est pas équivalent à faire voter indépendamment chacun des mots qui le constitue. Cela invite à s'intéresser à la construction de super mots.

Cependant, si M est le dictionnaire, il y a 2^M super mots. Une sélection est donc impossible car il faudrait construire les 2^M super mots puis en sélectionner certains. Mais, [57] (repris par [91]) est un exemple emblématique d'algorithme qui rend possible de construire les supers mots qui

correspondraient à une sélection construite sur la discrimination (présence dans les exemples de la classe 1 divisée par la présence dans la classe -1) et la représentativité (présence dans les exemples de la classe 1).

Définition des *JEP* : Les *JEP* sont des ensembles de mots spécifiques à une classe (c'est à dire jamais présents dans les ensembles de l'autre classe).

Formellement, $Y = \{-1, 1\}$ et les exemples sont des ensembles de mots $x_1, ..., x_N$ sur M (pour simplifier ici les x_n sont des ensembles au sens strict et non des multi ensembles : un même élément ne peut pas être inclus plusieurs fois). Les *JEP* associés à la classe 1 sont les ensembles $\mathcal{M} \subset M$ tels que : $\exists n, y^*(x_n) = 1, \mathcal{M} \subset x_n \subset M$ et $\forall n, y^*(x_n) = -1, \mathcal{M} \nsubseteq x_n$.

Parmi ces *JEP*, les minimaux au sens de l'inclusion sont des ensembles de mots infiniment discriminant et localement les plus représentatifs. Construire ces *JEP* minimaux se rapproche d'appliquer une sélection a priori sur 2^M.

Construction des *JEP* : La construction des *JEP* repose sur l'idée centrale de bordure d'ensembles d'ensembles et d'ensembles des ensembles intermédiaires. Si $\mathcal{A}, \mathcal{C}$ sont deux ensembles alors on note $\mathcal{G}(\mathcal{A}, \mathcal{C})$ l'ensemble des ensembles *intermédiaires* entre $\mathcal{A}$ et $\mathcal{C}$ défini comme $\mathcal{G}(\mathcal{A}, \mathcal{C}) = \{\mathcal{B} | \mathcal{A} \subset \mathcal{B} \subset \mathcal{C}\}$. Inversement, si $\mathcal{E}$ est un ensemble d'ensembles clos par inclusion (c'est à dire $\mathcal{A} \subset \mathcal{B} \subset \mathcal{C}$ et $\mathcal{A}, \mathcal{C} \in \mathcal{E}$ alors $\mathcal{B} \in \mathcal{E}$) alors la bordure de $\mathcal{E}$ est notée $\mathcal{B}(\mathcal{E})$ et est constituée de deux ensembles $\mathcal{A}, \mathcal{C}$ tels que $\mathcal{G}(\mathcal{A}, \mathcal{C}) = E$ (ces deux définitions s'étendent aux ensembles d'ensembles d'ensembles).

Il est démontré dans [57] que l'ensemble des *JEP* est exactement la différence (ensembliste) entre d'une part les ensembles intermédiaires entre $\emptyset$ et les x_n avec $y^*(x_n) = 1$ et d'autre part les ensembles intermédiaires entre $\emptyset$ et les x_n avec $y^*(x_n) = -1$.

Or, il est aussi démontré dans [57] qu'il est possible de construire les bordures de la différence de deux ensembles représentés par des bordures efficacement (sans construire les ensembles intermédiaires). C'est à dire exactement ce qu'il faut pour construire directement les *JEP* minimaux à partir des exemples.

15.4 Projection sur des variétés

L'*ACP* consiste grossièrement à se placer dans l'espace dans lequel les points semblent les mieux repartis. Dit autrement, l'*ACP* consiste à chercher s'il existe un sous espace sur lequel les points seraient répartis plongé dans un espace plus grand.

La projection sur des variétés consiste à se poser la même question en s'autorisant des surfaces qui ne sont pas nécessairement des sous espaces. L'exemple le plus emblématique est le cas de points distribués sur une sphère. Fondamentalement, les points sont répartis sur une surface de dimension 2 plongée dans un espace de dimension 3. Utiliser cette propriété permet de

diminuer la dimension avec peu voire pas de perte d'information. Typiquement dans cet exemple, il est très pertinent d'utiliser un *NN* en dimension 2 avec des distances propres à la sphère plutôt qu'un *NN* en dimension 3.

Cette technique est utilisée en reconnaissance d'actions par exemples dans [60].

16 Les systèmes étagés : Squelettes, actoms...

Les sections précédentes présentent un continuum de systèmes de décisions. Ces systèmes consistent schématiquement à changer explicitement ou implicitement la représentation des données puis à appliquer un algorithme d'apprentissage. La diversité des systèmes vient principalement de la façon de changer la représentation des données : construire directement une représentation adaptée ou extraire une représentation adaptée d'une première représentation grossière, mesurer la qualité de la représentation indépendamment de l'algorithme d'apprentissage ou en utilisant directement le score de cet algorithme etc.

Cependant, un autre type de systèmes existe, qu'on se propose d'appeler *les systèmes étagés*. Dans ces systèmes, on ne fait pas une décision mais plusieurs en se servant des décisions intermédiaires pour prendre les décisions suivantes.

Cette idée de concevoir un système où les décisions sont étagées n'est pas nouvelle [37] mais elle est souvent présente en reconnaissance d'actions car une part importante des systèmes de reconnaissance d'actions utilisent comme données brutes non plus des vidéos mais la trajectoire du *squelette* de la personne d'intérêt c'est à dire la position de chacune de ses articulations au cours du temps. Ce signal, de faible dimension et très haut niveau, a permis un accroissement spectaculaire des performances des systèmes d'interaction homme machine utilisant la vision dont un exemple représentatif est [72]. Or, ce signal est lui même extrait de la vidéo par un algorithme d'apprentissage comme [41, 80, 78].

D'autres exemples sont l'apprentissage de motifs (appelés *actom*) sémantiquement proches du geste [39] et les techniques de modèles déformables [36] (quoique l'apprentissage des parties y est non supervisé).

Formellement, on peut voir ce double apprentissage ainsi : $x \in X$ n'est plus associé à une classe $y^*(x) \in Y$ mais à deux types de classes $y_1^*(x) \in Y_1$ (les actions) et $y_2^*(x) \in Y_2$ (le squelette) (avec ici Y_1 discret mais Y_2 continu). Le système apprend donc deux critères : un premier $y_2(x)$ pour prédire $y_2^*(x)$ (prédire le *squelette*) puis un critère $y_1(x)$ pour prédire non pas $y_1^*(x)$ mais $P(y_1^*(x) = y | y_2^*(x) = z)$ c'est à dire l'action sachant le *squelette*.

Un des désavantages d'un tel système est que l'erreur minimale associée à $P(y_1^*(x) = y | y_2^*(x) = z)$ peut ne pas être nulle (ou plus généralement est forcément plus grande que l'erreur associée à $P(y_1^*(x) = y|x)$). Mais, il y a

au moins deux types d'intérêts à construire de tels systèmes : Le premier est que capturer $P\left(y_1^*\left(x\right)=y|y_2^*\left(x\right)=z\right)$ peut être plus simple que capturer $P\left(y_1^*\left(x\right)=y|x\right)$. Il est donc possible de faire moins d'erreur en capturant la première probabilité même si l'erreur minimale atteignable est plus grande. C'est cet avantage qui est recherché dans [36, 39] ou encore [91, 82].

Cependant, s'il est plus simple de capturer $P\left(y_1^*\left(x\right)=y|y_2^*\left(x\right)=z\right)$ que $P\left(y_1^*\left(x\right)=y|x\right)$, il est probable que capturer $P\left(y_2^*\left(x\right)=y|x\right)$ soit plus difficile que de capturer $P\left(y_1^*\left(x\right)=y|x\right)$. Mais rien n'impose de chercher à apprendre $P\left(y_2^*\left(x\right)=y|x\right)$ avec uniquement les exemples x pour lesquels on connaît à la fois $y_1^*\left(x\right)$ et $y_2^*\left(x\right)$. C'est typiquement le cas en reconnaissance d'action : [72] apprend des actions uniquement en utilisant le squelette mais l'algorithme d'extraction de squelette n'est pas appris uniquement avec les mêmes exemples, au contraire l'algorithme d'extraction de squelette [41, 80] est appris avec des millions d'exemples alors que les actions ne sont apprises qu'avec quelques milliers d'exemples. Les systèmes étagés touchent en ce sens un des points clés de l'intelligence artificielle : la possibilité de partager des exemples et donc des connaissances entre des taches d'apprentissage différentes.

Néanmoins, le domaine a encore assez peu de recul sur ce type de systèmes. Dans les articles de la littérature [93, 72, 98, 101, 84], le squelette est généralement considéré comme la donnée brute. C'est d'ailleurs aussi la position adoptée dans cette thèse.

Contributions

Troisième partie

Optimiser les votes en segmentation supervisée par élection

17 Intérêt des méthodes d'élection en segmentation

Cette thèse porte sur la segmentation supervisée d'actions dans des vidéos. Il convient donc de choisir un algorithme de segmentation et si besoin un algorithme de traitement du signal. Cependant, comme discuté dans l'introduction, cette thèse se concentre sur certaines spécificités dont la prise en compte d'un important contexte temporel et la nécessité de traiter des flux vidéos. Ces caractéristiques conduisent à privilégier les systèmes ayant besoin de peu de ressource machine et peu sensibles à la taille du contexte temporel. Or, il existe dans la littérature trois principales méthodes de segmentation qui ne sont pas équivalentes en terme de ressource machine et en sensibilité à la taille du contexte. Cela justifie d'étudier le choix de l'algorithme de segmentation.

17.1 Besoins en ressource machine

Tout d'abord, étant donné x une suite de taille T (ici $X = \left(\mathbb{N}^M\right)^+$), vouloir la segmenter implique de vouloir décider d'une classe Y pour chaque instant. Cela ne semble pas pouvoir se faire en moins de $O\left(T\left|Y\right|\right)$ opérations. Cependant, si on suppose qu'il est nécessaire de collecter l'information de l'intervalle $[t-\triangleright, t+\triangleright]$ pour décider de la classe de l'instant t, on a au moins $O\left(T\left(\triangleright+\phi+\left|Y\right|\right)\right)$ opérations avec ϕ le nombre d'opérations nécessaires pour prendre une décision sur l'intervalle $[t-\triangleright, t+\triangleright]$. Ainsi, une approche *SVM* avec suppression de l'information temporelle sur chaque intervalle conduit à une complexité en $O\left(T\left(\triangleright+\left|MY\right|\right)\right)$ où M est le nombre de mots. Mais cette approche supprime l'information temporelle, ce qui rend certaines classes indistinguables.

Pour conserver l'information temporelle, il faut que chaque temps puisse influer sur chaque classe. Cela semble donc conduire à une complexité minimale de $O\left(T\triangleright\left|Y\right|\right)$ c'est à dire exactement la complexité de *ISM*. Ainsi, *ISM* fait partie au vu de ces arguments des algorithmes utilisant le moins de ressources possibles.

Par contre, la méthode de type pavage impose de tester tous les intervalles d'une nouvelle vidéo, c'est à dire que la complexité est au moins de $O\left(T\triangleright\left|Y\right|+T\triangleright^2\right)$. Dans [46], elle est de $O\left(T\triangleright\left(M\left|Y\right|+\triangleright\right)\right)$.

De même, les méthodes de type *HMM* imposent de mettre à jour à chaque instant la probabilité que la classe à t soit y sachant qu'elle est y' à $t-1$. De plus, pour décider de la classe à $t-\triangleright$, il faut remonter dans le temps

et attribuer une classe temporaire à tous les instants de t à $t - \triangleright$. Ainsi, la complexité est au moins de $O\left(T \triangleright |Y| + T |Y|^2\right)$.

Ainsi, dans un contexte de ressources machines limitées, il est nécessaire d'utiliser une méthode d'élection comme *ISM* (ou un algorithme de même complexité). Remarquons que cette analyse de ressources néglige le nombre de mots extraits dans chaque image. Cela est pertinent dans un contexte standard et même en tenant compte de cette variable l'avantage reste à *ISM*.

Dans le contexte de la thèse, il aurait été possible d'utiliser *ISM* comme une boite noire en travaillant particulièrement le traitement du signal associé. Cependant, *ISM* est originalement un algorithme de classification qui nécessite des transformations pour être appliqué en segmentation (comme *HMM* est originalement un algorithme de classification même si le paradigme est propice à une utilisation en segmentation). Or, le travail sur les transformations, nécessaire à *ISM*, a permis de mettre en lumière la possibilité de proposer un nouvel algorithme d'élection adapté à la segmentation qui réunit deux extensions existantes de la méthode d'élection standard.

17.2 Taxonomie des méthodes d'élection

Il existe trois principales méthodes d'élection dans la littérature : les méthodes d'élection de type *classification naïve bayésienne* (*CNB*), la méthode *SVM* (i.e. l'application d'un *SVM* sur les points représentant les ensembles), et la méthode des modèles de formes implicites.

La *CNB* consiste à faire voter chaque mot $m \in M$ selon $v_{y,m} = P(y|m)$. La classe élue est alors celle qui a obtenu la plus grande quantité de votes.

La méthode *SVM* est identique excepté que chaque mot m vote pour la classe y comme $v_{y,m}$ solution de $\arg\min_{v} \left(vv + C \sum_{n=1}^{N} h\left(S\left(y^*\left(x_n\right), x_n\right) - 1\right)\right)$. Ainsi, on peut dire que la méthode *SVM* correspond à ajouter une forme d'optimisation dans la méthode de la *CNB*.

D'un autre coté, dans la méthode *ISM* chaque mot m localisé à l'instant t vote pour l'hypothèse qu'un intervalle de classe y est centré à l'instant t' selon $v_{y,t'-t,m} = P(y, t' - t|m)$. Ainsi, la méthode *ISM* correspond en un sens à ajouter le temps dans la *CNB*.

Ainsi, il existe deux extensions de la *CNB* : ajouter de l'optimisation (ce qui conduit à *SVM*) et ajouter le temps dans les votes (ce qui conduit à *ISM*). Or, ces deux extensions ne sont pas conceptuellement exclusives. Cela invite à considérer les deux simultanément.

Il existe dans la littérature des travaux allant dans le sens de rajouter de l'optimisation à *ISM* [63, 95, 104].

Indépendamment, le *SVM* peut être appliqué sur $\mathbb{R}^{M \times [-\triangleright, \triangleright]}$. Cependant, dans la littérature, le *SVM* est généralement appliqué sur des points où la dimension temporelle a été supprimée i.e. sur $\mathbb{R}^M$. Afin, d'éviter la confusion

on qualifiera ce dernier *SVM* de statique (sans aspect temporel) et le précédent de *SVM*. Or, il existe aussi des travaux allant dans le sens d'ajouter une dimension temporelle au *SVM* statique [55] : la méthode la plus classique dite de *décomposition pyramidale* consiste à construire plusieurs sacs de mots (statique) sur des intervalles décomposant la donnée (en supprimant sur chacun d'entre eux l'information temporelle) puis de les concaténer avant d'appliquer un *SVM*.

La principale contribution de cette thèse est d'aller plus loin que ces travaux en optimisant totalement les votes temporels de *ISM*, ou de façon équivalente, à utiliser un *SVM* contraint ou encore à utiliser le *SVM* statique avec une forme spécifique de *décomposition pyramidale* dense. Cette contribution est à la fois algorithmique et mathématique. Elle est algorithmique car elle conduit à un algorithme permettant d'effectuer une segmentation à plusieurs classes. Mais, elle est aussi mathématique car elle explicite les équivalences mentionnées : l'équivalence entre optimiser *ISM*, et, contraindre *SVM* à avoir une cohérence temporelle, et, utiliser *SVM* avec une *décomposition pyramidale* dense.

18 Optimiser le modèle de formes implicites

La première contribution de la thèse consiste à présenter une méthode d'élection pour la segmentation temporelle utilisant des votes temporels optimisés vis à vis de l'erreur.

Il est naturel de rapprocher cette méthode de *ISM*. Mais *ISM* n'est directement ni une méthode d'élection, ni une méthode de segmentation. Aussi, il convient de présenter les transformations successives de *ISM* nécessaires pour obtenir une méthode d'élection pour la segmentation temporelle utilisant des votes temporels.

18.1 Transformer *ISM*

Le modèle de formes implicites introduit dans [56] est construit sur 4 étapes :

- chaque mot localisé $(m, t') \in x$ vote pour l'hypothèse qu'un intervalle de classe y est centré en t dans la donnée x avec un poids $P(y, t' - t|m)$
- l'ensemble des votes est aggloméré : $S(y, t, x) = \sum_{(m,t') \in x} P(y, t - t'|m)$
- les scores S sont lissés temporellement
- un intervalle de classe y centré en t est détecté si $S(y, t, x)$ est supérieur à un seuil.

Les deux principales différences avec une méthode d'élection sont la présence d'un seuillage et d'un lissage.

18.1.1 Remplacer le seuillage par une élection

Le seuillage peut directement être remplacé par l'élection de la classe obtenant le plus de votes à chaque instant comme dans [101]. La méthode devient ainsi :

- chaque mot localisé $(m, t') \in x$ vote pour l'hypothèse qu'un intervalle de classe y est centré en t dans la donnée x avec un poids $P(y, t' - t|m)$
- l'ensemble des votes est aggloméré : $S(y, t, x) = \sum_{(m,t') \in x} P(y, t - t'|m)$
- les scores S sont lissés temporellement
- la classe décidée à l'instant t est celle ayant obtenu le plus de votes i.e. $y(x, t) = \arg\max_{y \in Y} S(y, t, x)$.

18.1.2 Supprimer le lissage

Choisir $y(x, t) = \arg\max_{y \in Y} S(y, t, x)$ impose que chaque instant t reçoive une quantité non négligeable de votes. Or dans *ISM*, la plupart des instants ne reçoit aucun vote.

Cela est compensé dans [101] en augmentant l'impact du lissage temporel qui prend la forme de : $S'(y, t, x) = \sum_{t'} S(y, t', x)\, \mathcal{G}(t - t')$ où $\mathcal{G}$ est typiquement une gaussienne.

À première vue, rapprocher *ISM* d'une élection renforce le lissage qu'on veut aussi supprimer. Cependant, comme $S(y, t, x)$ est une simple somme i.e. $S(y, t, x) = \sum_{(m,t') \in x} P(y, t - t'|m)$, il est équivalent de lisser S pour former S', ou, de lisser chacun des votes : si on introduit $P'(y, \triangle_t |m) = \sum_{\triangle_{t'}} P(y, \triangle_{t'} |m)\, \mathcal{G}(\triangle_t - \triangle_{t'})$ alors S' peut s'obtenir directement en agglomérant les votes P'.

Mais à première vue, P a une sémantique alors que P' n'en a pas. Cependant, si on introduit $P''(\triangle |y)$ la probabilité que l'instant relatif $\triangle$ soit compris dans un intervalle de classe y et que $\mathcal{G} = P''$ alors P' retrouve une sémantique. $P'(y, \triangle |m)$ correspond alors à la probabilité que l'instant $\triangle$ (relativement à l'instant d'extraction de m) soit de classe y sachant l'extraction de m. On passe ainsi d'une probabilité de centrage à une probabilité de présence ce qui est sémantiquement cohérent avec la volonté d'obtenir une segmentation.

ISM devient alors :

- chaque mot localisé $(m, t) \in x$ vote pour l'hypothèse que la classe de l'instant t' est y avec un poids $P'(y, t' - t|m)$ (et non P)
- l'ensemble des votes est aggloméré : $S(y, t, x) = \sum_{(m,t') \in x} P'(y, t' - t|m)$
- la classe décidée à l'instant t est celle ayant obtenu le plus de votes i.e. $y(x, t) = \arg\max_{y \in Y} S(y, t, x)$.

Cette dernière version de *ISM* est à la fois une méthode d'élection et une méthode de segmentation.

Introduire une forme d'optimisation dans *ISM* revient alors simplement à remplacer $P'(y, t'-t|m)$ par des votes génériques $v(y, \triangle, m)$. Toutes ces transformations conduisent à une formalisation générique de la méthode d'élection en segmentation.

18.1.3 Mise en équation de la méthode d'élection en segmentation

La méthode d'élection en segmentation est donc fondée sur les votes $v(y, \triangle, m)$ qui se veulent une mesure de la confiance dans le fait que la classe de l'instant t' est y sachant l'extraction de $m \in M$ en t avec $\triangle = t' - t \in [-\triangleright, \triangleright]$. D'autre part, cette méthode est régie par les deux équations :

$$S(y, t, x) = \sum_{(m,t') \in x} v\left(y, t - t', m\right) \tag{3}$$

$$y(x, t) = \arg\max_{y \in Y} S(y, t, x) \tag{4}$$

Dit autrement, ce type de méthode d'élection en segmentation est résumé par le pseudo code :

```
//Soit T la taille de la donnée
//Soit Y l'ensemble des classes
//Soit L la latence
//Soit M le dictionnaire
//soit x[t=1..T] la donnée
//x[t] l'ensemble des mots de M extrait en t
//soit v[y=1..Y,dt=-L..L,m=1..M] les votes
//alors pred[t'=1..T] est la prédiction de y à l'instant t'
pred=prediction(T,Y,L,M,x,v)
  S[y,t']=0
  pour t de 1 à T
    pour t' de t-L à t+L
      pour y de 1 à Y
        pour m dans x[t]
          S[y,t']+=v[y,t'-t,m]
  pour t' de 1 à T
    pred[t'] = arg max sur y de S[y,t']
```

L'apprentissage consiste alors à sélectionner des votes pertinents.

18.2 Optimisation naïve de la méthode d'élection

Optimiser les votes pour minimiser l'erreur se traduit alors par la volonté que pour chaque exemple d'apprentissage x_n et chaque temps $t \in [1, T_n]$ (avec $T_n = |x_n|$), la classe décidée $y(x_n, t)$ soit la classe réelle $y^*(x_n, t)$. Cela

conduit à l'objectif : $\arg\min_{v} \sum_{n,t} \delta\left(y^*\left(x_n,t\right)-y\left(x_n,t\right)\right)$ avec $\delta\left(0\right)=1$ et 0 sinon. Compte tenu de l'équation (4), cela implique d'avoir une marge strictement positive typiquement de 1 entre la vraie classe et les autres classes : $\arg\min_{v} \sum_{n,t,y} h\left(S\left(y^*\left(x_n,t\right),t,x_n\right)-S\left(y,t,x_n\right)-1\right)$ où h est la fonction introduite dans la formulation SVM (section 3 équation (1)) c'est à dire $h\left(u\right)=0$ si $u\geq 0$ et $-u$ sinon.

Compte tenu de l'équation (3), cela conduit au problème d'optimisation : $\arg\min_{v} \sum_{n,t,y} h\left(\sum_{(m,t')\in x_n} v\left(y^*\left(x_n,t\right),t-t',m\right)-v\left(y,t-t',m\right)-1\right)$.

Cependant, toute la méthode présentée n'est alors que l'application brutale d'un SVM linéaire (à plusieurs classes) sur une fenêtre glissante représentée par un point dans l'espace $\mathbb{R}^{M\times[-\triangleright,\triangleright]}$. Le problème est que présentée ainsi la méthode considère $v\left(y,\triangle,m\right)$ et $v\left(y,\triangle+1,m\right)$ comme deux variables totalement différentes.

Le lissage temporel qui a été supprimé de la méthode ISM pour la ramener à une méthode d'élection était justement une façon de contraindre les votes à être lisses. Cette régularité temporelle est à la fois sémantiquement souhaitable et pourrait réduire la dimension VC de l'apprentissage. Un argument allant dans ce sens est détaillé en annexe 2. Il convient donc de réintroduire une cohérence temporelle tout en conservant le formalisme de l'élection.

18.3 Forcer une cohérence temporelle

La sémantique de $v\left(y,\triangle,m\right)$ est le vote du mot m pour la présence de la classe y au temps $\triangle$ (relatif à celui de l'extraction de m).

Dit autrement, $v\left(y,\triangle,m\right)$ mesure une information apportée par m à $\triangle$ de sa position d'extraction. Il peut paraître pertinent dans ce contexte de forcer cette information à avoir 0 comme unique point de maximum non nul c'est à dire en un sens à décroître avec $|\triangle|$.

Ces arguments invitent ainsi à rajouter des contraintes de consistance temporelle. Pour cela, on impose premièrement aux votes d'être positifs i.e. $v\geq 0$ afin que la valeur du vote soit sémantiquement cohérente avec une information. Ensuite, on impose à l'information de décroître ce que l'on traduit par les contraintes de décroissance suivantes :

$$\forall y,m,\triangle,\triangle',\begin{cases} 0\leq\triangle\leq\triangle' \\ \triangle'\leq\triangle\leq 0 \end{cases} \Rightarrow v\left(y,\triangle,m\right)-v\left(y,\triangle',m\right)\geq 0$$

Ces contraintes de décroissance sont soutenues par l'argument de l'annexe 2 et l'étude des cas extrêmes : dans le cas de données parfaitement structurées, s'il existe un mot tel que systématiquement on observe une classe y pour $t'=t-\alpha,...,t+\beta$ avec t l'instant d'extraction du mot alors, le

vote du mot est constant sur l'intervalle $[-\alpha, \beta]$. De même, dans le cas de données parfaitement non structurées, chaque mot ne vote qu'à son instant d'extraction.

Ici, contrairement à la section 6, il y a une différence de nature (pas seulement une différence en terme de paramètres) entre un *SVM* linéaire sur l'espace $\mathbb{R}^{M\times[-\triangleright,\triangleright]}$ et cette méthode d'élection dont les votes minimisent l'erreur d'apprentissage tout en respectant des contraintes de décroissance. Toute cette différence (dit autrement toute l'intelligence ajoutée à la méthode pour lui permettre de modéliser le temps simplement) correspond exactement à ces contraintes de décroissance.

Ainsi, la méthode *SVM* de $\mathbb{R}^M$ est de même nature que la méthode *CNB* sur les multi ensemble sur M mais la méthode *SVM* de $\mathbb{R}^{M\times[-\triangleright,\triangleright]}$ n'est pas de même nature que la *ISM* ou *DOHT* pour les suites de multi ensemble sur M car ces dernières forcent une cohérence temporelle.

18.4 Forcer une régularité

En plus de minimiser l'erreur d'apprentissage tout en respectant cette nouvelle contrainte de décroissance, il est possible d'ajouter un terme de régularité $\|v\|$ à la formulation. Le terme de régularité standard de la littérature est $\|v\|_2^2$ [88]. Cependant, ce terme rend difficile l'optimisation du problème (5). Notamment, les contraintes de décroissance de (5) ne sont pas standards et un grand nombre de ces contraintes peuvent être *saturées* c'est à dire influer sur la solution. Ces deux caractéristiques (généricité et saturation des contraintes) conduisent à utiliser des algorithmes d'optimisations génériques. Ainsi, l'utilisation de $\|v\|_2^2$ conduit à un problème semi défini alors que $\|v\|_1$ conduit à un programme linéaire. Au vu de la disponibilité d'implémentation pour résoudre des programmes semi définis et de celles pour résoudre des programmes linéaires et au vu de l'historique de la résolution des programmes linéaires vis à vis des programmes semi définis, il a paru pertinent d'opter pour la norme $\|v\|_1$.

18.5 Optimisation de la méthode d'élection en segmentation : *DOHT*

La formulation complète de l'apprentissage qui a été nommée *DOHT* pour *Deeply Optimized Hough Transform* est :

$$\min_{v\geq 0} \left(\begin{array}{c} \|v\|_1 + C \sum\limits_{n,t,y} h\left(\sum\limits_{(m,t')\in x_n} \left(\begin{array}{c} v\left(y^*\left(x_n,t\right), t-t', m\right) \\ -v\left(y, t-t', m\right) \end{array} \right) - 1 \right) \\ +C_\infty \sum\limits_{m,y,0\leq\triangle_t\leq\triangle_{t'}\vee\triangle_{t'}\leq\triangle_t\leq 0} h\left(v\left(y,\triangle_t,m\right) - v\left(y,\triangle_{t'},m\right)\right) \end{array} \right) \quad (5)$$

Dans cette formulation, C équilibre la régularisation et l'attache aux données et C_∞ est un grand nombre (comme pour le problème *SVM* séparable, il

existe une valeur à partir de laquelle toutes les valeurs de C_∞ conduisent au respect des contraintes de décroissance).

Il convient de rappeler que ce problème est le problème de l'apprentissage du *DOHT* de la même façon que l'apprentissage *ISM* consiste à dire $v(y, t-t', m) = P(y, t-t'|m)$. La phase de test est la même pour toutes les méthodes de type *ISM* i.e. régie par les équations (3) (4).

19 Évaluation du *DOHT*

D'un point de vue mathématique, *DOHT* se situe entre d'une part *ISM* (apprentissage avec une dimension de l'espace de 2) ou le *SVM* avec suppression de l'information temporelle (dimension $VC = M \times Y$), et d'autre part le *SVM* (en dimension $VC = M \times (2 \triangleright +1) \times Y$). La suppression de l'information temporelle dans le *SVM* permet de réduire la dimension VC mais au prix d'une importante perte d'information. À l'inverse, les contraintes de décroissance sont sémantiquement pertinentes. En un sens, on peut espérer qu'elles ne réduisent pas l'information pour les *problèmes qui répondent à des besoins*. Avec cette hypothèse implicite, démontrer que la dimension VC de l'apprentissage du *DOHT* est inférieur à $M \times (2 \triangleright +1) \times Y$ aurait été une importante justification de la méthode vis à vis du *SVM*. Mais, bien qu'un argument allant dans ce sens soit présenté en annexe 2, il n'a pas été possible d'établir un tel résultat.

Alternativement, trouver un *problème qui réponde à des besoins* pour lequel l'algorithme est efficace permet de le justifier. Aussi, cette section présente les résultats de deux campagnes d'expériences pour évaluer le *DOHT* sur des problèmes reconnus et de synthèse.

Il n'existe pas à ma connaissance de problème reconnu par la communauté qui soit à la fois adapté au contexte de segmentation supervisée et dont les données brutes rentrent directement dans le formalisme d'une méthode d'élection. Aussi, pour évaluer *DOHT* sur un problème de segmentation supervisée reconnu par la communauté, il n'y a pas d'alternative à effectuer un traitement du signal qui formate les données brutes pour la méthode d'élection. Or, les performances dépendent alors de ce traitement : [97] tend à rappeler qu'il peut exister d'autres traitements qui font que le premier algorithme d'apprentissage devienne le dernier.

À l'opposé, un problème de synthèse fournit des données brutes directement utilisables par le *DOHT*, et donc, permet une évaluation non biaisée par un traitement du signal. Mais, le *no free lunch theorem* rappelle qu'on peut toujours trouver un problème sur lequel un algorithme est meilleur qu'un autre. Un problème de synthèse qui ne répond pas à une situation pratique n'a donc aucun intérêt.

Aussi, après la présentation des autres méthodes de type *ISM*, on présente, à la fois, une expérience sur le jeu de données *honeybee* reconnu par

la communauté, et à la fois, des problèmes de synthèse qui nous semblent expliquer les résultats sur *honeybee*.

19.1 État de l'art des méthodes de type *ISM*

Il existe à ma connaissance trois méthodes de type *ISM* en plus de *DOHT* et de *ISM*. Chacune de ces autres méthodes rajoute de l'optimisation pour influencer les votes de *ISM* sans aller jusqu'à une optimisation *complète* des votes. Ces méthodes sont les suivantes.

***Max-Margin Hough Transform* :** Dans [63], un coefficient est introduit pour chaque mot afin de pondérer les votes *ISM*. Cet algorithme *Max-Margin Hough Transform* (*MMHT*) conduit aux votes : $v(y, \triangle, m) = \omega_m \times P(y, \triangle | m)$. Les poids ω_m donnent plus ou moins d'importance aux différents mots. Ces poids sont appris par un *SVM*.

***Implicit Shape Kernel* :** Dans [104], les votes restent les votes de *ISM* i.e. $P(y, \triangle | m)$ mais l'estimation de P est effectuée individuellement sur chaque exemple d'apprentissage et les différents exemples sont pondérés. Cela permet de donner plus de poids aux exemples représentatifs. Cette méthode *Implicit Shape Kernel* (*ISK*) conduit ainsi aux votes : $v(y, \triangle, m) = \sum_n (\omega_n \times P_n(y, \triangle | m))$ où P_n est l'estimation de la probabilité P associée à l'exemple n.

***ISM* + *SVM* :** Dans [95], un *SVM* est appliqué sur le score S associé à l'agglomération des votes dans un contexte binaire de détection. Cela est équivalent à ajouter un coefficient pour chaque déplacement : $v(y, \triangle, m) = \omega_\triangle \times P(y, \triangle | m)$

***HF* :** Il convient de remarquer qu'il est normal que les méthodes *Hough forest* ne soient pas présentées. Ces méthodes consistent à sélectionner un ensemble de mots adaptés aux votes *ISM*. Mais, du point de vue des votes i.e. à dictionnaire fixé, ces méthodes correspondent à un simple *ISM*.

Résumé : Le tableau 19.1 résume les différentes méthodes de type *ISM*.

Comme *ISK* est très spécifique au problème de la classification, il n'est pas pris en compte dans les tests suivants.

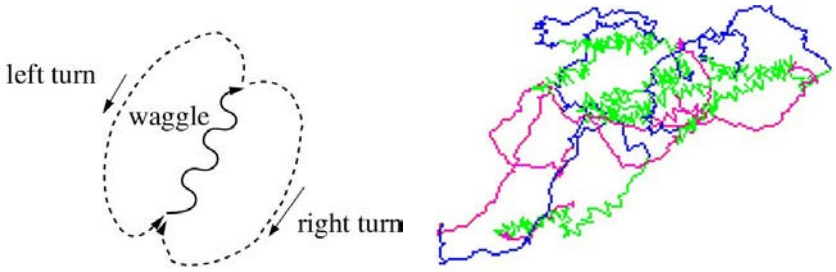
19.2 Évaluation de *DOHT* pour la segmentation de comportement d'abeilles

L'article [66] introduit le jeu de données *honeybee* très adapté pour évaluer les performances de méthodes de segmentation supervisée. Ce jeu de

méthode	forme des votes	variables optimisées
ISM [56]	$P(y, \triangle \mid m)$	-
MMHT [63]	$\omega_m \times P(y, \triangle \mid m)$	ω_m
ISK [104]	$\sum_n (\omega_n \times P_n(y, \triangle \mid m))$	ω_n
ISM+SVM [95]	$\omega_\triangle \times P(y, \triangle \mid m)$	$\omega_\triangle$
DOHT	$v_{y,\triangle,m}$	$v_{y,\triangle,m}$

P_n est la probabilité estimée uniquement à l'aide de l'exemple d'apprentissage n

TABLE 1 – Les différentes méthodes de type *ISM*



(a) types de comportements (b) exemple de trajectoire

Le vert correspond au comportement *waggle*, le magenta à *right turn* et le bleu à *left turn*.

FIGURE 9 – le jeu de données *honeybee*

donnée consiste en 6 longues trajectoires d'abeilles. La trajectoire de chaque abeille autour d'un instant dépend du comportement de l'abeille à cet instant. Il y a trois types de comportements dans les trajectoires du jeu de données (figure 9). Une annotation manuelle du comportement dans chaque image de chaque vidéo est fournie.

19.2.1 Traitement du signal

Le signal source pour chaque trajectoire est la suite des positions et orientations dans l'image de l'abeille au cours du temps (u_t, v_t, ϕ_t). Mais, même ce signal pourtant haut niveau doit être traité : par exemple, le comportement est invariant aux translations et rotations globales des trajectoires. Un traitement du signal produisant une description également invariante est donc souhaitable.

Soit, $R(\mu)$ la matrice de rotation de l'angle opposé à μ (i.e. l'angle $-\mu$) et notons $p(t) = (u_t, v_t)$. On propose d'extraire à l'instant t les primitives $(R(\phi_t)(p_{t-\tau} - p_t), ..., R(\phi_t)(p_{t+\tau} - p_t))$ avec τ des tailles faibles devant la taille de la trajectoire T.

Ces primitives sont ensuite transformées en mots par l'algorithme des K

moyennes indépendamment pour chaque taille τ.

Un ensemble de tailles τ et de nombres de centres dans l'algorithme des K moyennes a été considéré. Le paramétrage consistant à choisir 3 tailles $\tau \in \{1, 3, 6\}$ et $K = 10$ a été considéré comme représentatif (on a donc trois mots par image).

19.2.2 Résultats

Afin de présenter des résultats comparables à ceux de [66], la pertinence est mesurée suivant un processus de *leave-one-out cross validation* ($LOOCV$). Les algorithmes réalisent l'apprentissage sur 5 des 6 vidéos et appliquent leurs modèles sur la dernière. L'erreur $e\left(B_{test}, L\left(B\right)\right)$ est calculée pour chacun des quatre algorithmes. Plus précisément, on compte le nombre d'instants où l'algorithme prédit la bonne classe divisé par le nombre total d'instants. Le taux de consistance est 1 moins l'erreur et le pourcentage de ce taux mesure alors la performance. Le problème d'optimisation (5) est résolu en utilisant *CPLEX*[2]. Les votes de *ISM* sont directement obtenus par des comptages sur B. Les apprentissages des autres méthodes de type *ISM* sont résolus en rajoutant la forme des votes correspondante comme contrainte dans le programme linéaire de (5). *SVM* est évaluée, afin de valider la pertinence de l'ajout des contraintes de décroissance vis à vis de *SVM*. Remarquons qu'ici *SVM* réfère bien à un *SVM* travaillant sur $\mathbb{R}^{M \times [-\triangleright, \triangleright]}$ et non sur $\mathbb{R}^M$. Ce dernier est désigné par *SVM statique* pour souligner la suppression de l'information temporelle. Pour *SVM*, *MMHT*, *ISM+SVM* et *DOHT*, l'apprentissage comporte des méta paramètres : typiquement un coefficient C d'attache aux données. Différentes valeurs des méta paramètres sont considérées et chaque algorithme est paramétré de façon à optimiser ses résultats. L'ensemble des 6 décompositions possibles des 6 vidéos en deux ensembles disjoints de 5 et 1 vidéos est utilisé pour chaque algorithme. Le taux de consistance moyen sur les 6 décompositions mesure la performance finale. Les résultats des différentes chaînes de traitement de type *ISM* sont présentés dans le tableau 2 avec les résultats de la littérature sur ce jeu de données.

Dans cette expérience, l'algorithme *DOHT* surpasse *ISM*, *MMHT* ainsi que *ISM+SVM* (pour un traitement du signal identique). Ce qui justifie l'optimisation des votes. De plus, *DOHT* a des résultats meilleurs que le *SVM*, ce qui valide la pertinence des contraintes de décroissance et surpasse le *SVM statique* ce qui valide que la suppression simple de l'information temporelle conduit à une forte perte d'information pertinente.

DOHT obtient de plus des résultats légèrement meilleurs que *HDP-HMM*, *SLDS*, *pavage naïf* (pour un traitement du signal différent) alors que ces algorithmes ne sont pas des algorithmes de faible complexité (ils pourraient être inutilisables en test sur des bases plus grandes). *DOHT* obtient

2. www.ibm.com

Méthode	taux de consistance moyen
SVM statique	62.3
SVM	82.4
ISM [56]	71.9
MMHT [63]	78.8
ISM + SVM [95]	77.5
DOHT	**86.5**
approximation DOHT	85.1
HDP-HMM [94]	83.3
SLDS [66]	85.9
PS-SLDS [66]	**87.7**
pavage naïf [46]	84.5
pavage [46]	**89.3**

TABLE 2 – Résultat sur *honeybee*
Les résultats pour *SVM statique*, *SVM*, *ISM*, *MMHT*, *ISM+SVM* et *DOHT* (et *approximation DOHT*) correspondent à des implémentations naïves et partagent le même traitement du signal. Les autres résultats sont extraits des articles correspondants qui utilisent un traitement du signal différent.

TABLE 3 – Résultats sur *honeybee*

par contre des résultats inférieurs à ceux de l'état de l'art avec un taux de consistance de 86.5% contre 87.7% et 89.3% pour [66] et [46]. Cependant, ces deux algorithmes ne sont pas des algorithmes de faible complexité, *PS-SLDS* a, par exemple, une complexité en $O\left(T \triangleright^2 |Y|^2\right)$.

Aussi, cette expérience soutient quand même la pertinence de *DOHT* au moins pour un contexte applicatif impliquant l'utilisation de peu de ressources machines.

L'amélioration apportée par *DOHT* sur *honeybee* (parmi les algorithmes de faible complexité) a motivé la publication [18].

19.3 Évaluation sur des données de synthèse

19.3.1 Premier cas d'école

L'expérience suivante illustre qualitativement les limitations de *ISM*. Considérons trois classes $\mathcal{A}, \mathcal{B}, \mathcal{C}$ et deux mots $\mathcal{U}, \mathcal{V}$. Une zone homogène de classe $\mathcal{A}$ est associée à une suite de $\mathcal{U}$ (i.e. $\mathcal{UUUUUU}$...). De même, $\mathcal{B}$ est associée à une suite de $\mathcal{V}$ (i.e. $\mathcal{VVVVVV}$...) et $\mathcal{C}$ est associée à une alternance régulière de $\mathcal{U}$ et de $\mathcal{V}$ (i.e. $\mathcal{UVUVUVUV}$...). Dans les trois cas, on a un mot par image.

Pour faire le lien avec *honeybee*, on peut voir $\mathcal{A}$ et $\mathcal{B}$ comme *right turn*

et *left turn* et $\mathcal{C}$ comme *waggle* avec $\mathcal{U}, \mathcal{V}$ qui sont des mouvements vers la droite ou vers la gauche. Ainsi, même si ce problème de synthèse est abstrait, il parait pertinent pour pouvoir expliquer pourquoi *DOHT* surpasse *ISM* sur *honeybee*.

Notons $P(\mathcal{A}|\mathcal{U})$ la probabilité que la classe soit $\mathcal{A}$ en un instant où un mot $\mathcal{U}$ est extrait. Alors, $P(\mathcal{A}|\mathcal{U}) = P(\mathcal{B}|\mathcal{V}) = \frac{2}{3}$, $P(\mathcal{C}|\mathcal{U}) = P(\mathcal{C}|\mathcal{V}) = \frac{1}{3}$ (les autres probabilités étant nulles).

Supposons maintenant que chaque mot vote également pour son instant d'extraction et pour les deux instants voisins : c'est à dire formellement que $P(y|x_{-1}x_0x_1) = P(y|x_{-1}) + P(y|x_0) + P(y|x_1)$. Au milieu d'un motif $\mathcal{UUU}$, le score *ISM* envers $\mathcal{A}$ est de $P(\mathcal{A}|\mathcal{UUU}) = 3 \times P(\mathcal{A}|\mathcal{U}) = 2$ alors que celui de $\mathcal{C}$ est de $3 \times P(\mathcal{C}|\mathcal{U}) = 1$. Ainsi, la classe élue est la classe correcte. Mais, au centre d'un motif $\mathcal{UVU}$ le score *ISM* envers $\mathcal{C}$ est toujours de 1 alors que celui pour $\mathcal{A}$ est encore de $P(\mathcal{A}|\mathcal{UVU}) = 0 + 2 \times P(\mathcal{A}|\mathcal{U}) = \frac{4}{3}$. Ainsi, $\mathcal{A}$ est toujours préféré à $\mathcal{C}$.

Ainsi *ISM* n'est pas capable de segmenter correctement la classe $\mathcal{C}$. Pourtant il est possible de segmenter correctement les trois classes avec ce formalisme : il suffit que $v(\mathcal{A},\mathcal{U}) \geq v(\mathcal{C},\mathcal{U})$ mais que $3v(\mathcal{C},\mathcal{U}) \geq 2v(\mathcal{A},\mathcal{U})$. Par exemple, il suffit que $v(\mathcal{A},\mathcal{U}) = 2$ et $v(\mathcal{C},\mathcal{U}) = \frac{5}{3}$. Ainsi, $S(\mathcal{A},\mathcal{UUU}) - S(\mathcal{C},\mathcal{UUU}) = S(\mathcal{C},\mathcal{UVU}) - S(\mathcal{A},\mathcal{UVU}) = 1$. Or, ces dernières équations sont exactement les contraintes de marge de 1 associées à ce problème. Ainsi, les votes solutions donnés en exemple sont ceux sélectionnés par *DOHT* (quand le C du *DOHT* est infini).

19.3.2 Deuxième cas d'école

Considérons maintenant le problème où $\mathcal{A}$ est associé à une suite qui commence par $\mathcal{U}$ et finit par $\mathcal{V}$, $\mathcal{B}$ est associé à l'extraction simultanée au centre de $\mathcal{U}$ et $\mathcal{V}$, et $\mathcal{C}$ est associé à l'extraction au premier quart de $\mathcal{U}$ et au dernier quart de $\mathcal{V}$.

Ce cas d'école est motivé par l'objectif de reconnaître des actions dans des vidéos. Imaginons une personne qui boit, téléphone ou se brosse les dents $(\mathcal{B}, \mathcal{A}, \mathcal{C})$. Si l'on n'est ni capable de reconnaître les objets ni de savoir si la main est au niveau de l'oreille ou du visage (ce qui est le cas en faible résolution), alors les seules choses qui distinguent les actions sont les durées entre le fait de lever la main et de descendre la main $\mathcal{U}, \mathcal{V}$. Certes, cet exemple est moins intuitif que le précédent connecté à *honeybee*. Cependant, des situations où seules les durées permettent de distinguer des actions semblent réellement possibles en reconnaissance d'actions. Or, dans ces cas, la dimension temporelle devient primordiale, ce que veut souligner ce cas d'école.

cas 1	cas 2
$\mathcal{AAAAAA} \leftrightarrow \mathcal{UUUUUU}$	$\mathcal{A}...\mathcal{A} \leftrightarrow \mathcal{U}$ $\mathcal{V}$
$\mathcal{BBBBBB} \leftrightarrow \mathcal{VVVVVV}$	$\mathcal{B}...\mathcal{B} \leftrightarrow$ $\mathcal{UV}$
$\mathcal{CCCCCC} \leftrightarrow \mathcal{UVUVUV}$	$\mathcal{C}...\mathcal{C} \leftrightarrow$ $\mathcal{U}$ $\mathcal{V}$

Les formes schématiques des exemples segmentés emblématiques des deux cas d'école sont présentés ci dessus. Les classes i.e. y^* sont à gauche de la double flèche (une classe par instant) et les mots i.e. x à droite de la double flèche (un mot par instant dans le cas 1 et un mot dans certains instants dans le cas 2). Les résultats correspondants à ces cas d'école sont détaillés ci dessous.

Méthode	forme du votes	cas 1	cas 2
ISM [56]	$P(y, \triangle \mid m)$	67	66
MMHT [63]	$\omega_m \times P(y, \triangle \mid m)$	69	67
ISM+SVM [95]	$\omega_\triangle \times P(y, \triangle \mid m)$	67	77
-	$\omega_{m,y} \times P(y, \triangle \mid m)$	85	69
DOHT	$v_{y,\triangle,m}$	**86**	**88**

TABLE 4 – taux de consistances des différentes méthodes de types *ISM* sur des cas d'école

19.3.3 Résultats sur les deux cas d'école

Pour chacun de ces deux cas d'école, une base d'apprentissage B et une base de test B_{test} ont été formées à partir des règles présentées précédemment auxquelles on ajoute du bruit (substitutions aléatoires de mots, de classes, longueurs des zones homogènes aléatoires et choix aléatoires des classes des zones homogènes successives). L'expérience est résumée dans le tableau 4.

L'algorithme *DOHT* surpasse les autres algorithmes de type *ISM* sur ces cas d'école. Cela soutient à nouveau la pertinence de *DOHT*.

20 Approximation du *DOHT*

La pertinence de *DOHT* justifiée expérimentalement invite à s'en servir pour la reconnaissance d'activités dans des vidéos. Cependant, la solution de (5) bien que facilement calculable sur *honeybee* nécessite trop de ressources machines lors de l'apprentissage sur de grandes bases d'apprentissage. Il est important de noter que les ressources machines nécessaires après l'apprentissage sont faibles. Les ressources nécessaires à l'apprentissage sont généralement jugées non pertinentes dans la communauté car l'apprentissage peut être effectué *une fois pour toutes*. Cependant, cela suppose quand même de pouvoir l'effectuer, ce qui n'est pas trivial pour le problème (5). Aujourd'hui, l'état de l'art des méthodes d'optimisation de programmes linéaires

est capable de traiter quelques millions de variables et contraintes pour des problèmes creux. Or, la formulation $DOHT$ contient $O(NT|Y|)$ contraintes avec $O(|Y| \triangleright |M|)$ coefficients non nuls. Typiquement pour 20 vidéos de 4 minutes (i.e. avec $T = 4 \times 60 \times 25$) et pour 10 classes, on atteint déjà le million de contraintes.

Il est donc indispensable d'accélérer l'apprentissage. La deuxième contribution de cette thèse est de proposer une telle accélération construite sur une reformulation et une approximation de (5).

Les démonstrations centrales au cœur de cette reformulation sont présentés en annexe 3.

Il en découle que dans la formulation (5), il est équivalent de chercher une fonction v qui respecte les contraintes de décroissance ou de chercher une fonction v décomposable sur l'ensemble des fonctions caractéristiques d'intervalles inclus dans $[-\triangleright, \triangleright]$ contenant 0 noté $I([-\triangleright, \triangleright])$. Il est donc possible de substituer la recherche de v à la recherche de ω les coefficients de cette décomposition, et dès lors, les contraintes de décroissance sont automatiquement vérifiées.

20.1 Reformulation de *DOHT*

En utilisant la remarque précédente, il est possible de reformuler le $DOHT$ (on omet temporairement le terme de régularité) :

$$\min_{\omega \geq 0} \sum_{n,t,y} h\left(\sum_{(m,t') \in x_n} \left(\begin{array}{c} \sum\limits_{I \in I([-\triangleright,\triangleright])} \omega(y^*(x_n,t), I, m) \times \chi_I(t-t') \\ - \sum\limits_{I \in I([-\triangleright,\triangleright])} \omega(y, I, m) \times \chi_I(t-t') \end{array} \right) - 1 \right)$$

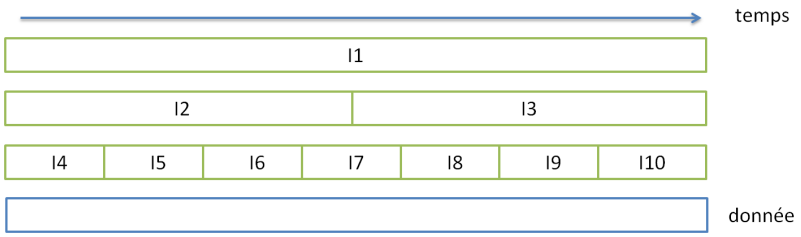
Cette reformulation consiste simplement à remplacer $v(y, \triangle, m)$ par sa décomposition c'est à dire $v(y, \triangle, m) = \sum\limits_{I \in I([-\triangleright,\triangleright])} \omega(y, I, m) \times \chi_I(\triangle)$.

Afin de voir cette équation plus simplement, on introduit le point $x_{n,t}$ tel que pour tout $m \in M$ et tout $I \in I([-\triangleright, \triangleright])$, $x_{n,t,m,I}$ est le nombre d'occurrences du mot m dans x_n localisées dans l'intervalle I translaté de t c'est à dire formellement $x_{n,t,m,I} = |\{(m,t') \in x_n, t - t' \in I\}|$. Alors, la somme $\sum\limits_{(m,t') \in x_n} \sum\limits_{I \in I([-\triangleright,\triangleright])} \omega(y, I, m) \times \chi_I(t-t')$ est simplement $\langle \omega_y, x_{n,t} \rangle$.

Ainsi, l'équation (5) se ramène au problème d'optimisation :

$$\min_{\omega \geq 0} \left(\|\omega\|_2^2 + C \sum_{n,t,y} h\left(\langle \omega_{y^*(x_n,t)}, x_{n,t} \rangle - \langle \omega_y, x_{n,t} \rangle - 1 \right) \right) \tag{6}$$

Or, ce dernier problème ressemble fortement à l'équation (2) (section 4).



Décomposition pyramidale classique :
un sac de mots est construit
sur chacun des intervalles d'une *grille*



Décomposition pyramidale dense :
Pour chaque instant, on forme l'ensemble des sacs de mots
correspondant à l'ensemble des intervalles centrés en cet instant

Dans les deux cas, l'ensemble des sacs est concaténé pour former un descripteur. Celui ci est associé à la donnée dans le premier cas et à un instant de la donnée dans le second.

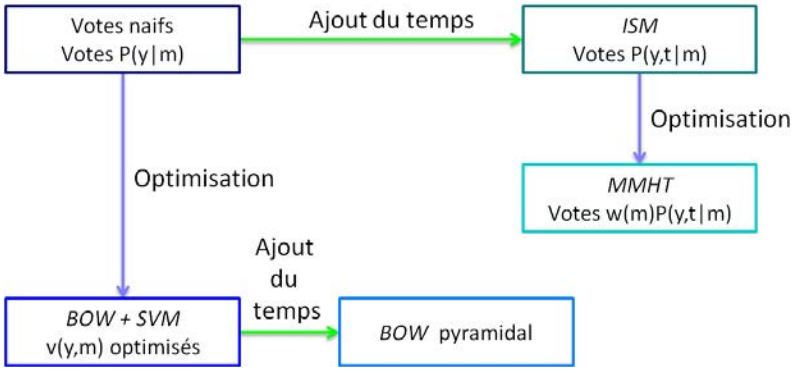
FIGURE 10 – La décomposition pyramidale dense

20.2 Décomposition pyramidale dense

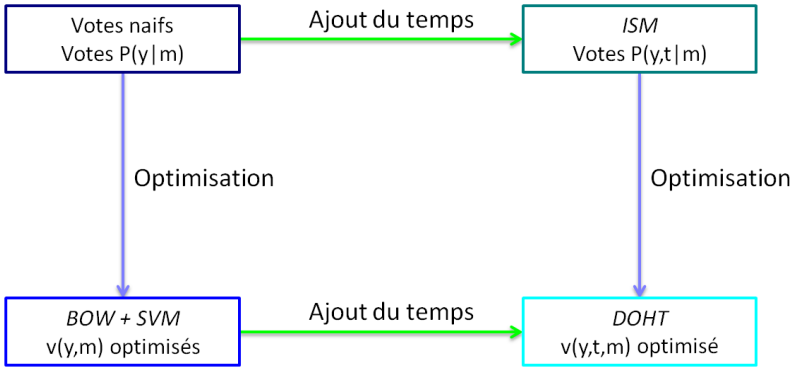
L'équation (6) correspond à l'apprentissage d'un simple *SVM* (si on omet les contraintes de positivité). Pourtant les équations (5) et (6) sont équivalentes : dans cette dernière, la cohérence temporelle est cachée dans la construction même du point sur lequel on applique le *SVM statique*.

On appelle cette décomposition la décomposition pyramidale dense (illustrée par la figure 10). L'équivalence entre les équations (5) et (6) donne une nouvelle façon de voir la méthode de décomposition pyramidale [55].

Ainsi, la figure 11 résume la taxonomie présente dans la littérature et les premières contributions de cette thèse. Cette thèse va plus loin que les travaux précédents en optimisant totalement les votes temporels de l'approche *ISM* et/ou en ajoutant des contraintes dans la méthode *SVM* et/ou en utilisant la décomposition pyramidale dense avec un *SVM statique*.



L'état de l'art



L'unification proposée par *DOHT*

FIGURE 11 – La taxonomie des méthodes d'élection

20.3 Accélération de l'apprentissage

20.3.1 Principe

L'objectif premier de la reformulation est de permettre d'accélérer le *DOHT*. Naïvement, la formulation (6) est pire que la formulation initiale : le nombre de votes est passé de $O(\triangleright \times |M| \times |Y|)$ à $O\left(\triangleright^2 \times |M| \times |Y|\right)$ puisqu'il y a $O\left(\triangleright^2\right)$ intervalles dans $I([-\triangleright, \triangleright])$.

Cependant, cette deuxième formulation s'approxime beaucoup plus simplement. Sur *honeybee*, on observe notamment une quasi conservation des performances $85,1\%$ contre $86,5\%$ pour *DOHT* (ce score est présenté dans le tableau 2) malgré la suppression des contraintes de positivité et la substitution d'un petit sous ensemble d'intervalles à l'ensemble des intervalles $I([-\triangleright, \triangleright])$.

L'avantage de cette double approximation est de transformer le problème *DOHT* (eq. (6)) en un simple problème *SVM* (eq. (2)). Cela conduit à une importante accélération (d'un facteur 100 dans l'exemple précédent) du à la fois à la simplification de l'expression de l'apprentissage mais aussi à la possibilité d'utiliser des logiciels standards pour résoudre l'apprentissage notamment *svm multiclass* extention de *svm ligth* [50].

L'inconvénient de cette approximation est que le réglage du petit sous ensemble d'intervalles utilisé devient un réglage supplémentaire lors de l'apprentissage dont le comportement n'est pas trivial. Bien sur, trop peu d'intervalles conduit à perdre toute information temporelle et donc à de mauvaises performances. Mais trop d'intervalles conduit aussi à de mauvaises performances : la conservation de tout les intervalles combinée avec la suppression de la positivité revient à supprimer les contraintes de décroissance qui sont la principale information ajoutée (par rapport à un *SVM*) explicitement pour modéliser le temps et qui permettent de mieux contraindre l'optimisation. Afin que la modélisation du temps soit apportée par l'obligation pour les mots de voter de façon cohérente dans le temps, il convient d'utiliser un ensemble d'intervalles de taille adaptée (et dans ce contexte, la suppression de la contrainte de positivité a un sens).

Ainsi, sur chaque problème, il convient de régler l'ensemble d'intervalle à utiliser. Cependant, l'algorithme est relativement peu sensible aux variations de l'ensemble, ce qui permet d'obtenir rapidement des valeurs pertinentes. Ce dernier point est soutenu par les expériences suivantes sur *honeybee*.

20.3.2 Expériences

Deux ensembles d'intervalles est évalués : des intervalles dichotomiques formellement $\mathcal{J}_\times = \left\{\left[-2^{-\alpha}\triangleright, 2^{-\beta}\triangleright\right], 0 \leq \alpha, \beta \leq 4\right\}$, des intervalles arithmétiques formellement $\mathcal{J}_+ = \left\{\left[\frac{1}{\alpha}\triangleright, \frac{1}{\beta}\triangleright\right], 1 \leq \alpha, \beta \leq 4\right\}$. De plus, des intervalles où la partie positive et négative sont découplées sont aussi évalués

	$\triangleright = 50$	$\triangleright = 100$	$\triangleright = 200$
$\mathcal{J}_\times$	17	17	20
$\mathcal{J}_+$	18	20	26
$\mathcal{J}_\times^\cup$	19	17	19
$\mathcal{J}_+^\cup$	19	20	27

TABLE 5 – Pourcentage d'erreur de l'approximation de *DOHT* sur *honeybee* en fonction des intervalles utilisées

$\mathcal{J}_\times^\cup = \{[-2^{-\alpha}\triangleright, 0], 0 \leq \alpha \leq 4\} \cup \{[0, 2^{-\alpha}\triangleright], 0 \leq \alpha \leq 4\}$ (équivalent à $\mathcal{J}_\times$) et $\mathcal{J}_+^\cup = \left\{\left[\frac{1}{\alpha}\triangleright, 0\right], 0 \leq \alpha \leq 4\right\} \cup \left\{\left[0, \frac{1}{\alpha}\triangleright\right], 0 \leq \alpha \leq 4\right\}$ (équivalent à $\mathcal{J}_+$).

Ces quatre ensembles sont évalués avec $\triangleright = 50$, 100, ou 200.

Indépendamment, il est possible de n'utiliser non pas tous les instants mais seulement une partie des instants durant l'apprentissage. De plus, s'il n'est pas possible de conserver les contraintes de positivité avec *svm ligth*, il est possible de biaiser l'optimisation vers des valeurs positives (heuristique sans garantie théorique : cela consiste simplement à rajouter des exemples associés à une nouvelle classe formés de points à composantes négatives). Afin de permettre de découpler l'influence de ces modifications, elles sont indépendamment évaluées sur *honeybee*.

20.3.3 Résultats

Sous échantillonnage et bais de positivité : Pour les 12 ensembles d'intervalles présentés, le sous échantillonnage (1 image sur 2 contre toutes les images) ne modifie que très peu les performances. L'erreur augmente en moyenne de 0,08%. Il est donc pertinent sur *honeybee* de n'utiliser qu'une image sur 2.

De même, l'ajout du terme biaisant l'optimisation vers des valeurs positive ne diminue l'erreur que de 0,04%. Il est donc pertinent sur *honeybee* de ne pas rajouter ce biais de positivité.

Ensembles d'intervalles : Le pourcentage moyen d'erreur des intervalles est présenté dans le tableau 5.

Excepté pour $\mathcal{J}_+$ (et $\mathcal{J}_+^\cup$) avec une trop grande latence, on peut observer que tous les résultats sont proches (et relativement proche de la version non approximée). Ces résultats montrent la grande stabilité de la version approximée de (6) vis à vis du sous échantillonnage, de l'absence de tout terme de positivité et vis à vis de l'ensemble d'intervalles utilisé.

L'équivalence entre les formulations (5) et (6) ainsi que l'efficacité de l'approximation de la formulation (6) (en terme de stabilité vis à vis des paramètres et en terme de performance) a motivé la publication [19].

Quatrième partie

Segmenter des actions à partir de *primitives squelettes*

21 Choisir le cœur du traitement du signal

La partie précédente présente un algorithme de segmentation. Cependant, cet algorithme n'est pas capable de traiter directement des vidéos brutes à la manière d'un algorithme de type *Deep Learning*. Premièrement, cet algorithme est conçu pour traiter des ensembles de *mots* localisés (c'est à dire des symboles) alors que les données brutes sont des vidéos. Ensuite, dans cette méthode, l'apprentissage a une dimension (au sens de l'espace) linéaire en la taille du dictionnaire mais avec un coefficient de linéarité élevé et difficilement réglable : il convient donc de maîtriser la taille du dictionnaire, ce qui impose d'extraire des mots portant une forte information. Ainsi, cet algorithme de segmentation ne peut être pertinent qu'inclus dans un *système à primitives*. Cela nous amène à nous intéresser aux primitives.

21.1 L'adéquation des primitives aux applications visées

Chaque système de reconnaissance d'actions dépend de manière importante du type de vidéos traitées et donc du contexte applicatif visé. Par exemple, la reconnaissance d'actions dans des corpus vidéos de type *youtube* doit gérer des vidéos extrêmement hétérogènes en terme de taille d'images, de longueur, de résolution et de nombres de personnes filmées. De plus, ces vidéos sont généralement filmées depuis des caméras mobiles et à zoom variable, ce qui rend difficile l'utilisation des méthodes de type soustraction de fond.

L'exemple opposé est la reconnaissance de gestes dans des vidéos d'interaction homme machine. Ces vidéos sont généralement acquises en intérieur, par un capteur dédié et fixe, et sont focalisées sur une personne d'intérêt qui s'est positionnée de façon adéquate vis à vis du capteur. De plus, dans ce contexte, la personne essaye d'effectuer des gestes prédéfinis par l'usage de l'interaction, c'est à dire qu'elle essaye de faciliter la communication avec la machine.

Dans ces deux exemples, l'algorithme d'apprentissage doit tenir compte des atouts et des inconvénients du type de vidéos traitées. Typiquement, l'hétérogénéité des vidéos *web* impose aux algorithmes de traiter de grandes bases de données, tout en considérant que, même grandes, ces bases de données ne représentent qu'un petit ensemble de l'ensemble des vidéos. Inversement, dans un contexte d'interaction homme machine, la stabilité de la configuration entre la personne et la machine permet même avec une base de données

plus petite de capturer une part représentative de l'ensemble des actions. De plus, dans un contexte d'interaction homme machine, le système doit répondre immédiatement. Cela supprime l'intérêt d'utiliser un algorithme sophistiqué utilisant le contexte temporel puisque cela signifie que l'information à un instant donnée devra être suffisante pour déterminer le geste associé à cet instant. Ainsi, dans ce type de contexte, segmenter et classer sont deux choses identiques puisque l'information d'un instant n'est pas utilisée aux autres instants.

De même, et surtout, les primitives doivent elles aussi tenir compte des atouts et des inconvénients du type de vidéos traitées. Typiquement dans un contexte de vidéos *web*, il convient que les primitives soient les plus robustes possibles à la qualité de l'image et au mouvement de la caméra. L'utilisation de points d'intérêt est alors adaptée pour utiliser au mieux le peu d'information disponible.

Inversement, le contexte d'interface hommes machines permet l'utilisation de la soustraction de fond et l'utilisation de capteurs spécifiques comme la *Kinect* qui fournit directement une carte de profondeur et éventuellement le *squelette* de la personne d'intérêt.

21.2 Intérêt de primitives *squelettes* pour la segmentation d'actions

Dans [72], un détecteur est spécifiquement mis au point pour la reconnaissance de mouvements de danse dans un contexte d'un jeu vidéo. L'avantage de ne pas avoir à tenir compte du contexte temporel combiné avec l'avantage de disposer du *squelette* permet d'utiliser directement le signal *squelette* avec un algorithme de type plus proche voisin (très pertinent en faible dimension).

Cet exemple illustre l'intérêt de l'extraction de *squelettes* pour la reconnaissance de gestes dans un contexte d'interface homme machine. Mais, la question de savoir si ce type de primitive est intéressant pour les applications visées de cette thèse (par exemple la surveillance pour le maintien à domicile de personnes fragiles) mérite d'être posée.

Tout d'abord, se pose la question de la disponibilité d'une telle information. Dans ce dernier contexte, les vidéos sont acquises en intérieur et de façon maîtrisée en terme de résolution, taille d'image, et stabilité du fond. Ainsi, l'utilisation de capteur dédié et de soustraction de fond est possible. Mais, à la différence du contexte d'interaction homme machine, la personne d'intérêt est d'une part positionnée indépendamment de la caméra, et d'autre part ne cherche pas à faciliter la communication. Ces deux aspects rendent beaucoup plus difficile l'extraction du squelette.

Ensuite, même dans l'hypothèse où le *squelette* est disponible, il est indispensable pour notre objectif, de prendre en compte un important contexte temporel pour décider de l'action en un instant. Or, cette nécessité dégrade le premier avantage de l'extraction du *squelettes* à savoir sa faible dimension.

Ainsi, même en disposant du *squelette*, il n'est plus pertinent d'utiliser directement un algorithme *NN*. Dit autrement, la nécessité de tenir compte du contexte temporel amène à utiliser un algorithme de segmentation sachant qu'aucun n'est capable de profiter pleinement de l'incroyable réduction de dimension offerte par l'extraction de squelette.

Cependant, au vu de l'engouement académique et grand public pour l'extraction de squelette, il parait acceptable de supposer que les données squelettes seront bientôt disponibles dans un contexte de surveillance à domicile. Mais sur ces vidéos *squelettes* une nouvelle extraction de primitives (qu'on appelle des *primitives squelettes*) a lieu. À partir de ces *primitives squelettes*, des mots localisés sont extraits afin de se placer dans le contexte d'utilisation du *DOHT*. Le système présenté est donc un système *irrégulier*. Il est aussi en un sens un système étagé puisque utilisant sur des données squelette. Mais, comme l'extraction de squelette est utilisée comme un boite noire, il est plus pertinent de la considérer comme une extraction *irrégulière* sur les vidéos *squelettes*.

21.3 Traitement des vidéos *squelettes*

La littérature sur le traitement des données *squelettes* est peu structurée (car jeune : la *kinect* date de 2008) mais pourtant relativement importante. De plus, *DOHT* impose peu de contraintes fonctionnelles : typiquement, avec *DOHT*, il est possible d'utiliser des primitives qui ne capturent aucune information temporelle puisqu'il prend déjà partiellement cette information en compte. La seule contrainte de *DOHT* est que le dictionnaire soit de taille adaptable car *DOHT* est plus pertinent en faible dimension (par contre le nombre de mots par image peut être quelconque, ce qui permet de mélanger des primitives).

Aussi, cette partie présente une campagne expérimentale d'évaluation de différentes primitives *squelettes*.

22 Évaluations de *primitives squelettes*

22.1 Jeux de données

Afin d'évaluer différentes implémentations de primitives *squelettes*, il convient d'utiliser des jeux de données *squelettes* ou à partir desquels on sait extraire les *squelettes*. Mais, l'écrasante majorité des jeux de données squelettes est dédiée (de façon cohérente avec la visée de la *kinect*) à la reconnaissance de gestes, c'est à dire une reconnaissance de mouvements cherchant à répéter un motif préétabli ([24] présente une liste exhaustive des jeux de données de reconnaissance d'actions publiés avant le 2/3/2012). Ces jeux de données ne sont donc que partiellement adaptés pour évaluer des primitives squelettes avec comme objectif applicatif la surveillance à domicile c'est à

dire un contexte dans lequel les mouvements sont hétérogènes. [26] présente une liste (consensuelle vis à vis de la littérature) de jeux de données *kinect* relativement proches du contexte applicatif visé : *MSR-Action3D dataset [58], RGBD-HuDaAct [65], kitchen scene action dataset [79], ReadingAct dataset [25], LIRIS human activity dataset [96], CAD60 [84], MSRDailyActivity3D dataset [93], ChaLearn Gesture dataset [35]* . À ces jeux de données, on peut ajouter *UTKAD* (*University of Texas Kinect-Action Dataset*) [98] et *TUM* [85].

Tous ces jeux de données contiennent des vidéos de profondeur et certains directement des données squelettes. Plus précisément, seuls *TUM, CAD60, MSRDailyActivity3D dataset, ChaLearn* (ainsi que *MSR-Action3D dataset, UTKAD* bien que les données *squelettes* fournies soient de mauvaise qualité) contiennent des données squelettes. Pour les autres jeux de données, l'extraction de squelette de la *kinect* à partir des vidéos de profondeur échoue fréquemment car les vidéos sont éloignées de vidéos de type interaction homme machine (personne face au capteur, à une distance adaptée, sans occultation). Ainsi, *RGBD-HuDaAct, kitchen scene action dataset, ReadingAct dataset, LIRIS human activity dataset* ne sont pas directement adaptés pour notre campagne d'évaluation. De plus, *ChaLearn Gesture dataset* est un jeu de données contenant aussi du son dont l'exploitation est nécessaire afin d'atteindre de bonnes performances, ce qui le rend difficilement utilisable dans notre contexte.

Enfin, seul *TUM* est adapté pour évaluer de la segmentation supervisée, donc l'utilisation de *DOHT*. Au bilan, seuls *TUM, CAD60, MSRDailyActivity3D dataset,MSR-Action3D dataset, UTKAD* sont réellement adaptés à notre campagne de tests et seul *TUM* permet d'évaluer *DOHT* en segmentation supervisée. Aussi, les évaluations ont principalement lieu dans un contexte de classification avec un algorithme de type *SVM* et secondairement avec *ISM* et *DOHT* à la fois sur *TUM*, mais aussi sur d'autres jeux de données bien qu'alors dans un contexte non adapté au *DOHT*.

Une description plus précise des jeux de données et des protocoles associés est introduite ci dessous.

***TUM* :** *TUM* est un jeu de données introduit dans [85]. Les vidéos correspondent à une personne dressant une table (10 actions de types *ouvrir un placard, poser un objet* sont considérées). *TUM* contient environ 20 vidéos de 2 mins (5 acteurs différents). Les vidéos ne sont pas segmentées (plusieurs actions se succèdent dans 1 vidéo) et l'action présente dans chaque image est fournie. Le nombre d'images pour chaque classe d'action varie significativement notamment une action *divers* est présente dans 55% des images. La durée de chaque type d'action varie aussi significativement avec des actions très courtes (de l'ordre du geste). Les successions d'actions sont structurées mais restent non triviales.

Dans cette thèse, ce jeu de données est principalement utilisé avec 3 protocoles d'évaluations :

TUM-S-27 : [101] propose un protocole d'évaluation : les vidéos sont séparées en 2 groupes (test et apprentissage). Le taux de consistance sur la base de test mesure la performance : le nombre d'images des vidéos de test bien classées divisé par le nombre total d'images de test.

TUM-S-13 : Le squelette est très détaillé dans *TUM* (27 articulations). Cependant, [101] propose aussi d'utiliser seulement 13 articulations.

TUM-C-13 : *TUM* permet d'évaluer un algorithme de classification en utilisant la vérité terrain (i.e. les classes connues) pour découper les vidéos en sous vidéos de classes homogènes.

UTKAD : *UTKAD* est un jeu de données, introduit dans [98], qui contient des vidéos *kinect* et les squelettes associés. Les vidéos restent proches d'un contexte interface homme machine : la personne filmée accomplit des mouvements prédéfinis. 10 actions sont considérées (*applaudir, tirer, pousser*). La base contient environ 20 vidéos de 2 mins (10 acteurs différents). Les vidéos ne sont pas segmentées (plusieurs actions se succèdent dans 1 vidéo) mais une transition (un retour au repos) est présente entre chacune des actions. Le squelette n'est pas présent dans toutes les images, et de plus, l'échantillonnage est irrégulier et notamment plus concentré hors des zones de repos. Ici aussi, la variance des durées des actions et du nombre d'image d'actions par classes est significative.

Dans cette thèse, ce jeu de données est principalement utilisé avec 2 protocoles d'évaluations :

UTKAD-C : [98] propose un protocole d'évaluation pour évaluer des algorithmes de classification. Les zones homogènes de vidéos ne correspondant pas à du repos sont extraites directement des classes données. Il en résulte 200 petites vidéos segmentées qui sont la véritable entrée du système. La performance est mesurée par le taux de consistance moyen à travers une *LOOCV*. Mais, ici on compte le nombre de vidéos correctement classées (divisé par 200) et non pas le nombre d'images correctement classées.

UTKAD-S : Pour utiliser le *DOHT* sur *UTKAD*, on utilise la même procédure que pour *honeybee*. Cependant, certaines images sont associées à plusieurs actions. Dans ce cas, ces images ne sont pas utilisées à l'apprentissage et ces images sont comptées comme des erreurs en test.

CAD60 : *CAD60* est un jeu de données introduit dans [84] qui contient des vidéos *kinect* et les squelettes associés. Les vidéos sont très intéressantes vis à vis de l'objectif de maintient à domicile de personnes fragiles : les 12 actions sont des actions de la vie courante *boire, cuisiner, utiliser un ordinateur, se brosser les dents, téléphoner, se reposer*. Cependant, les vidéos sont déjà segmentées. Pire, les vidéos sont segmentées grossièrement : la base contient 60 vidéos de 2 mins environ et chaque vidéo est associée dans sa globalité à une et une seule action. Typiquement, dans une vidéo *boire*, la personne ne boit pas sans discontinuité pendant toute la vidéo. Il serait par exemple pertinent d'étiqueter *boire* les images correspondant effectivement à l'action *boire* et *neutre* les images sans action particulière. L'algorithme présenté dans [84] est d'ailleurs paradoxalement réglé pour étiqueter *neutre* un maximum d'image alors que chacune de ces décisions est nécessairement une erreur puisque cela n'est pas pris en compte dans l'annotation associée au jeu de données.

Dans cette thèse, ce jeu de données est principalement utilisé avec 2 protocoles d'évaluations :

CAD60-C : [84] propose d'évaluer des algorithmes de classification à travers le protocole *LOSO* (*leave one subject out*) qui consiste à apprendre avec 3 des 4 acteurs, tester sur le 4ème acteur et moyenner les scores sur les 4 découpes possibles. Comme le jeu de données ne contient qu'un seul gaucher les vidéos d'apprentissage sont symétrisées pour simuler le mouvement du droitier et du gaucher. Enfin, les actions possibles sont regroupées par lieux (*cuisine, salle de bain, salon*). La matrice de confusion des prédictions pondérées par les nombres d'images est calculée et la performance est mesurée par $F_{0,5}$-mesure.

CAD60-S : Pour utiliser le *DOHT* sur *CAD60*, on utilise une procédure assez *synthétique*. Cette procédure est toujours fondée *LOSO* (donc avec une personne en test et trois en apprentissage). À partir des 45×2 séquences d'apprentissage (après symétrisation droitier gaucher), on construit 10 très longues séquences en concaténant aléatoirement les 90 séquences d'apprentissage. On construit de même 5 séquences de test à partir des 30 séquences restantes. On compte alors le nombre d'images correctement étiquetées dans les grandes séquences de test divisé par le nombre d'images de test (comme pour *TUM-S*).

MSRAction3D : *MSRAction3D* est un jeu de données introduit dans [58] qui contient des vidéos de profondeur *kinect* à partir desquels des squelettes de qualité assez variable ont été extraits (par d'autres algorithme que ceux de la *kinect* et/ou *NITE*). Les 567 vidéos correspondent à un contexte interface homme machine. Ce sont des vidéos courtes, segmentées. 20 actions sont

considérées (*applaudir, tirer, pousser*). La taille de ce jeu de données en fait un jeu de données incontournable de la reconnaissance d'actions à visée interface homme machine.

Dans cette thèse, ce jeu de données est principalement utilisé avec le protocole d'évaluation suivant :

MSR1-C **:** [58] propose d'évaluer des algorithmes de classification à travers plusieurs protocoles dont un protocole de type *LOSO* comme pour *CAD60-C*. Le nombre de vidéos correctement étiquetées divisé par le nombre de vidéos de test mesure la performance. Par contre, les actions sont séparées en 3 groupes [58]. Cette séparation est importante : dans une de nos expériences, nous observons un écart de 30% entre l'évaluation avec et sans séparation. Cependant, par exemple, [93] ne tient pas compte de cette séparation à en juger par leur matrice de confusion.

MSRDailyAction **:** *MSRDailyAction* est un jeu de données introduit dans [93] qui contient des vidéos *kinect* et les squelettes associés. Les 320 vidéos se veulent proches d'un contexte de surveillance à domicile notamment pour les actions choisies *boire, manger, lire, téléphoner, écrire, utiliser un ordinateur, passer l'aspirateur, se réveiller, se lever, jeter des objets à la poubelle, jouer à un jeu vidéo, s'asseoir, marcher, jouer de la guitare, se coucher*. Cependant, la durée et la richesse sémantique des vidéos (segmentées) reste proche d'un contexte interface homme machine. Ce jeu de données devient de plus en plus incontournable en reconnaissance d'actions.

Dans cette thèse, ce jeu de données est principalement utilisé avec le protocole d'évaluation suivant :

MSR2-C **:** [93] ne propose pas explicitement de protocole mais on peut imaginer (même laboratoire, format des données, format des fichiers) que celui utilisé dans [93] est le même que pour *MSR1-C* (sans former de groupe d'actions). C'est en tout cas celui qui est utilisé dans la thèse.

22.2 Normalisation

Comme pour *honeybee*, bien que les données *squelettes* soient haut niveau, toutes les approches de la littérature (à ma connaissance) normalisent ces données pour qu'elles deviennent invariantes aux translations et rotations globales de la personne d'intérêt. Cela se justifie dans la mesure où la plupart des actions sont invariantes à ces translations et rotations. Cela permet dans [72] de projeter le squelette entier (en dimension 48 pour la *kinect*) sur une variété de dimension 16. Il faut noter une limite pour certaines actions : typiquement l'action *ouvrir un tiroir* de *TUM* a nécessairement lieu à proximité d'un tiroir. Mais, ces informations peuvent être éventuellement

rajoutées de façon découplée au squelette. Cela explique l'omniprésence de normalisations.

22.2.1 Taxonomie des normalisations

Il existe deux types de normalisations. La première consiste à utiliser uniquement des distances entre articulations [101, 93]. Le désavantage de cette approche est que naïvement la dimension du squelette passe de 48 à 120, et cette normalisation n'est pas invariante à l'échelle.

La deuxième normalisation consiste à exprimer les positions des articulations dans le repère (i.e. système de coordonnées) de la personne [72]. Le repère de la personne est celui construit sur les 3 axes égocentriques *gauche vers droite*, *bas vers haut* et *derrière vers devant*. Dans le cas simple, ce repère est construit sur les axes épaule gauche, épaule droite et pieds tête.

Ce changement de repère permet de rendre la représentation invariante aux translations et rotations sans changer la dimension. Cependant, il est possible d'aller plus loin en remarquant que la distance entre deux articulations liées est constante dans le temps. Cela amène à remplacer les positions normalisées par des angles : les positions normalisées du torse sont conservées mais les positions des autres articulations sont remplacées par deux angles encodant la direction de l'articulation vis à vis de l'articulation qui la connecte au torse.

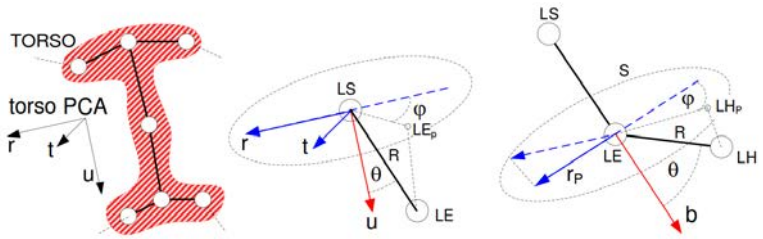
22.2.2 Descriptions des normalisations

Formellement, introduisons quelques notations : les données brutes sont un ensemble de positions 3D $\rho_{\alpha,t}$ avec α désignant une articulation (16 articulations avec la *kinect*) et t désignant un instant. La normalisation par distance consiste à remplacer $\rho_{\alpha,t}$ par $\sigma_{\alpha,\beta,t} = \|\rho_{\alpha,t} - \rho_{\beta,t}\|$. Alternativement, le squelette est remplacé par $\sigma_{\alpha,\beta,t,t'} = \left\|\rho_{\alpha,t} - \rho_{\beta,t'}\right\|$ dans [101] avec $t - t'$ petit : cela encode la distance entre l'articulation α de l'instant t avec l'articulation β de l'instant t'.

La normalisation par repère consiste schématiquement à se donner une fonction qui au squelette ρ_t de l'instant t associe un point 3D central $\gamma(\rho_t)$ et une matrice de similitude (rotation + homothétie) $\eta(\rho_t)$ puis remplacer chaque valeur $\rho_{\alpha,t}$ par $\varrho_{\alpha,t} = \eta(\rho_t) \times (\rho_{\alpha,t} - \gamma(\rho_t))$. Alternativement, le squelette est remplacé par $\varrho_{\alpha,t,t'} = \eta(\rho_{t'}) \times (\rho_{\alpha,t} - \gamma(\rho_{t'}))$ dans [72] avec $t - t'$ petit : cela encode la position de l'articulation α de l'instant t dans le repère associé au squelette de l'instant t'. Dans [72], normaliser le squelette grâce aux instants voisins permet de stabiliser les variations du repère associé au squelette.

Quand la normalisation utilise uniquement l'image courante, on parle de normalisation image par image sinon on parle de normalisation centrée.

La normalisation par angle, consiste premièrement à se donner un repère



LS, LE, LH correspondent aux articulations de niveau 1, 2, 3 par exemple épaule, coude, main.

FIGURE 12 – La normalisation par angles [72]

(comme ci dessus). Ensuite, il convient de considérer les dépendances des articulations du squelette. Il y a trois niveaux d'articulations : le premier niveau contient les épaules, les hanches et le cou, le deuxième niveau les genoux, les coudes et la tête, le troisième niveau les pieds et les mains. Les articulations du niveau 1 restent représentées par leurs positions normalisées (ou ne sont pas considérées : par exemple si ces articulations servent à construire le repère, il est possible que leurs positions soient constantes après normalisation). Chaque articulation du niveau 2 (respectivement 3) est associée à son articulation d'attache du niveau 1 (respectivement 2). Les articulations sont associées aux deux angles de la représentation sphérique du vecteur entre elles et leur articulation d'attache. Formellement, soit μ et ν les axes bas-haut et gauche-droite du squelette, et, soit ι le vecteur entre une articulation de niveau 2 et son attache de niveau 1, alors, l'articulation (du niveau 2) est associée à l'angle entre ι et μ d'une part et à l'angle entre ν et la projection de ι dans le plan orthogonal à μ. De même, soit ι le vecteur entre une articulation de niveau 3 et son attache de niveau 2 et κ le vecteur entre l'articulation de niveau 2 et celle de niveau 1, alors, l'articulation (du niveau 2) est associée à l'angle entre κ et ι d'une part et à l'angle entre la projection de ι dans le plan orthogonal à κ et la projection de ν dans ce plan. Tout cela est résumé dans la figure 12.

22.2.3 Évaluation des normalisations

Afin de valider la nécessité d'une telle normalisation, la performance de *DOHT* sur les mots directement obtenus à partir d'une quantification du squelette avec ou sans normalisation a été évaluée sur *UTKAD* et *TUM*. La quantification est effectuée par l'algorithme des K moyennes. Les méta paramètres de chaque chaîne sont donc K (K moyennes) et C (pour l'attache aux données dans l'équation (6)). Différentes valeurs des méta paramètres sont considérées et chaque algorithme est paramétré de façon à optimiser ses

Normalisation image à image	*UTKAD-S*	*TUM-S-13*
sans normalisation	58	53
distance	69	60
repère	70	**62**
angle	**72**	61

TABLE 6 – Résultats de *DOHT* appliqué sur les mots obtenus par quantification directe du squelette en fonction de la normalisation

résultats. Les résultats sont présentés dans la table 6.

Cette évaluation valide bien l'intérêt de normaliser le squelette. Notamment sur *TUM-S-13*, *DOHT* n'arrive même pas à dépasser la barre des 55% sans normalisation (score de *ISM*). Par contre, les performances des normalisations sont assez similaires.

TUM-S-13 étant plus proche de l'objectif applicatif de la thèse, la normalisation choisie pour les expériences de cette campagne est la normalisation repère.

22.3 Positions des articulations : individuelles ou combinées

Dans [98], les articulations sont combinées : le squelette est quantifié, ainsi à chaque instant on extrait un mot *squelette*. Dans [93], les articulations sont au contraire considérées indépendamment. Cela conduirait typiquement à 13 quantifications et 13 mots par image (1 pour chaque articulation).

Plus généralement, de nombreux facteurs interviennent dans l'impact d'une quantification individuelle ou combinée. La somme minimale des distances d'un point à son centre est plus petite dans une quantification combinée que dans une quantification individuelle. Cependant, comme l'algorithme des K moyennes ne donne qu'une approximation de cette somme minimale, avec K moyennes, il est possible que la quantification individuelle soit meilleure que la combinée. Indépendamment, la quantification optimale du point du vue du signal n'est pas forcément optimale du point de vue des classes : c'est tout le problème d'un traitement du signal découplé de l'apprentissage.

Formellement, la quantification combinée consiste ici à transformer le squelette complet ϱ_t en un seul mot $m_{squelette,t}$. Inversement, la quantification individuelle (articulation par articulation) consiste ici à transformer chaque position $\varrho_{\alpha,t}$ en un mot $m_{\alpha,t}$.

Afin, de choisir entre ces deux options, les performances de *DOHT* appliqué aux deux types de mots sont comparées comme dans l'évaluation des normalisations. Différentes valeurs des méta paramètres sont considérées et chaque algorithme est paramétré de façon à optimiser ses résultats. Les résultats sont présentés dans la table 7.

quantification	*UTKAD-S*	*TUM-S-13*
squelette	70	62
articulations	**74**	**77**

TABLE 7 – Résultats de *DOHT* en fonction de la quantification des positions des squelettes

primitives	*TUM-S-13*
position	77
sous trajectoire ($\tau = 8$)	**80**

TABLE 8 – Résultats de *DOHT* avec des primitives instantanées ou temporelles

Cette expérience souligne l'intérêt de quantifier individuellement les articulations vis à vis d'une quantification totalement combinée.

22.4 Positions des articulations ou sous trajectoires

Le choix entre la quantification individuelle ou combinée des positions vis à vis des articulations existe aussi vis à vis du temps. Formellement, quantifier image à image consiste à transformer chaque position $\varrho_{\alpha,t}$ en un mot $m_{\alpha,t}$ et quantifier des sous trajectoires consiste à transformer chaque vecteur $\varrho_{\alpha,t-\tau,t}, ..., \varrho_{\alpha,t+\tau,t}$ en un mot $m_{\alpha,t,\tau}$. $\varrho_{\alpha,t-\tau,t}, ..., \varrho_{\alpha,t+\tau,t}$ est simplement $\rho_{\alpha,t-\tau}, ..., \rho_{\alpha,t+\tau}$ normalisé dans le repère lié à l'instant central t (un peu comme dans *honeybee*). Ici les positions sont normalisées image à image mais les sous trajectoires subissent une normalisation centrée (on utilise le repère du squelette de l'instant central).

Afin, de choisir entre ces deux options, les performances de *DOHT* appliqué aux deux types de mots sont comparées comme pour les normalisations (excepté que *UTKAD* ne peut pas être utilisé car il présente un échantillonnage irrégulier). Différentes valeurs des méta paramètres sont considérées et chaque algorithme est paramétré de façon à optimiser ses résultats. Les résultats sont présentés dans la table 8.

Dans cette expérience, les sous trajectoires amènent à de meilleurs résultats que les positions seules. Cela est consistant avec la littérature : l'extraction de primitives capturant une partie du contexte temporel, à la fois comme [93, 101] qui extrait des primitives dans des données squelettes mais surtout comme dans [92] avec les techniques de trajectoires de points d'intérêts, est utilisée par l'état de l'art de la reconnaissance d'actions dans des vidéos. Cette expérience indique l'intérêt de s'intéresser aux sous trajectoires (vis à vis de primitives instantanées) et plus généralement aux primitives temporelles, objets du reste de cette section.

Une part significative des primitives, techniques de quantification et descripteurs suivants (principalement les primitives locales, la décomposition sphérique, l'histogramme d'occurrence et l'apprentissage à multiples noyaux) ont été conçus et implémentés pas Nicolas BALLAS ancien doctorant du CEA, LIST, DIASI, laboratoire de vision et d'ingénierie des contenus.

22.5 Différentes primitives temporelles

Rappelons les notations : $\varrho_{\alpha,t+t',t}$ est la position de l'articulation α à l'instant $t+t'$ normalisée vis à vis de l'instant t. Les primitives considérées ici sont :

Position : les primitives *position* (déjà introduites dans l'expérience précédente) sont les $\varrho_{\alpha,t+t',t}$.

Vitesse : les primitives *vitesse* [4] sont les $\varrho_{\alpha,t+1,t} - \varrho_{\alpha,t-1,t}$.

Accélération : les primitives *accélération* [4] sont les $\varrho_{\alpha,t+1,t} - 2\varrho_{\alpha,t,t} + \varrho_{\alpha,t-1,t}$.

Suite de positions : les primitives *suite de positions* (déjà introduites dans l'expérience précédente) sont les $\varrho_{\alpha,t-\tau,t}, ..., \varrho_{\alpha,t+\tau,t}$.

Suite de vitesse : les primitives *suite de vitesse* [92] sont les $\varrho_{\alpha,t-\tau+1,t} - \varrho_{\alpha,t-\tau,t}, ..., \varrho_{\alpha,t+1,t} - \varrho_{\alpha,t,t}, ..., \varrho_{\alpha,t+\tau,t} - \varrho_{\alpha,t+\tau-1,t}$.

22.6 Différentes implémentations de quantification

Chaque type de primitives est quantifié pour former des mots (les mots sont disjoints entre les différents types de primitives).

Algorithme des K moyennes : La quantification classique est l'algorithme des K moyennes qui a l'avantage d'être générique vis à vis de la dimension. Étant donné un nuage de points $\vartheta_1, ..., \vartheta_\Omega$, l'algorithme des K moyennes consiste à trouver des centres $\theta_1, ..., \theta_K$ tels que la somme des distances de chaque point ϑ_ω à son plus proche voisin parmi les θ_k soit faible.

Chaque point ϑ (y compris un nouveau point du même espace mais n'appartenant pas au nuage d'apprentissage) est alors remplacé par son plus proche voisin θ_k parmi les θ et donc symboliquement par le mot k. Ainsi, l'ensemble des points est décomposé en K parties qui forment le dictionnaire, chaque partie étant un mot.

Décomposition sphérique : [83] introduit une autre quantification pour les points 2D du disque unité construite sur une décomposition polaire. L'ensemble des angles est décomposé en 8 parties de 45 degrés (donc de 0 à 45 puis de 45 à 90 etc) et l'ensemble des rayons est décomposé en 3 parties. Enfin, une partie est rajoutée pour représenter le vecteur nul. Chaque point du cercle unité est alors associé à sa partie.

Cette décomposition s'étend en dimension 3. Afin de limiter le nombre de parties (et de réutiliser des implémentations préexistantes), nous avons choisi de décomposer les rayons en 2 parties et les deux angles en 6 parties de 60 degrés (ce qui conduit à 73 parties). Cette décomposition permet directement de quantifier les primitives 3D centrées et réduites (positions, vitesses et accélérations sachant que vitesses et accélérations sont déjà centrées).

22.7 Descripteur de trajectoires et de vidéos

Le *DOHT* s'utilise directement à partir de l'extraction de primitive et de la quantification. Mais avec d'autres algorithmes d'apprentissage (comme *SVM*), il est possible de construire un descripteur global pour chaque trajectoire et chaque vidéo.

Histogramme d'occurrences : Le descripteur global de trajectoire le plus simple est l'histogramme d'occurrences, typique de la méthode sac de mots. La vidéo est transformée en un point x où chaque composante x_d correspond au nombre d'occurrences du mot d dans la trajectoire.

Dans notre campagne d'évaluation, s'il est différent de 0, x est normalisé en norme L_1 : $\sum_d x_d = 1$.

Histogramme de co occurrence : L'histogramme d'occurrences ne prend pas en compte le séquencement. Une façon (alternative à l'ajout explicite du temps) de le prendre en compte est de former l'histogramme de co occurrences. Soit $\mathcal{M}$ la matrice dans laquelle chaque case i, j correspond au nombre de fois qu'un mot i est suivi par un mot j. Si on normalise les lignes de $\mathcal{M}$ en norme 1, on peut voir $\mathcal{M}$ comme une matrice de *Markov* : on considère un système dont l'état change à chaque instant et $\mathcal{M}_{i,j}$ encode alors la probabilité que l'état change de i à j. Alors, sous de faibles hypothèses, on peut considérer la distribution des états. Cette distribution est la combinaison d'états temporellement invariante (en espérance). C'est donc aussi le vecteur propre de $\mathcal{M}$ associé à la valeur propre 1 i.e. le vecteur π tel que $\mathcal{M}\pi = \pi$. Ce vecteur, de taille fixe indépendamment de la taille de la trajectoire, peut être utilisé comme descripteur de trajectoire.

Remarquons que ce descripteur peut aussi être vu comme une primitive (qui sera quantifiée [83] à l'aide d'un algorithme de type K moyennes).

Agglomération : Les histogrammes précédents fournissent des descripteurs globaux de trajectoires (notamment de taille fixe). Il convient ensuite de former des descripteurs globaux de vidéos. La solution la plus simple consiste à concaténer les différents histogrammes (dont le nombre est constant puisque ici le nombre d'articulations est constant). Ainsi, de la vidéo n, on extrait un point $q_{\lambda,n}$ pour chaque λ représentant un couple articulation - type de primitives. Puis, on forme le point x_n en concaténant les $q_{\lambda,n}$.

Apprentissage à plusieurs noyaux : Alternativement, il est possible d'utiliser la structure particulière des données pour apprendre un noyau différent par λ. Cela revient globalement à changer la fonction de régularité du *SVM* notamment en y rajoutant des paramètres qu'il convient d'optimiser simultanément à ceux du *SVM*. On parle alors d'apprentissage à plusieurs noyaux *MKL* (pour *Multiples kernel learning*). Ici, à la fois le *MKL* et le *SVM* traitent le cas non binaire par la méthode *1 contre 1* et sont statiques (suppression des positions des primitives extraites). Le *SVM* utilise l'implémentation *LIBSVM* et le *MKL* l'implémentation *SHOGUN*.

22.8 Tableau récapitulatif et résultats

Dans toutes ces expériences, on s'intéresse au résultat correspondant au réglage donnant la performance optimale. Les principales évaluations effectuées sont présentées dans le tableau 9 avec des libellés abrégés. Les abréviations[3] sont : P pour position, V pour vitesse, A pour accélération, SV pour suites de vitesses, SP pour suites de positions, K pour quantification par K moyennes, F pour quantification fixe, *short* pour *P-F + V-F + A-F + V-M*, *SPVP* correspond à l'ensemble *SP+V+P*. Enfin, *MKL* pour l'utilisation d'un noyau par articulation et *SVM* pour une agglomération suivie d'un *SVM statique* linéaire (*SVM* avec suppression des positions des primitives extraites).

La principale conclusion de ces résultats est que, malgré toutes les primitives envisagées, les suites de positions qui sont les plus simples des primitives combinées sont soutenues par leurs performances sur l'ensemble des expériences. Cela soutient ainsi l'intérêt des primitives temporelles et particulièrement des suites de positions.

23 Comparaison à l'état de l'art

La section précédente permet de fixer quelques chaînes de décision. Cette section présente dans le tableau 10 une évaluation plus poussée des chaînes conservées et compare les résultats obtenus avec l'état de l'art.

3. Se reporter au texte pour les abréviations

chaîne de décision[3]	*CAD60-C*	*TUM-C-13*
P-F-MKL	81	76
V-F-MKL	72	**84**
A-F-MKL	69	68
V-M-MKL	80	81
short-MKL	70	-
SP-K-MKL	**88**	**84**
SP-K-SVM	84	80
SV-K-SVM	73	-
P-K-SVM	74	78
V-K-SVM	68	78

Principales expériences de classification

primitive[3]	*P*	*V*	*SP*	*P+V*	*P+SP*	*V+SP*	*P+V+SP*
consistance	77	77	80	79	80	80	**81**

Principales performances de *DOHT* sur *TUM-S-13*

TABLE 9 – Résumé des principales expériences effectuées pour comparer les primitives *squelettes*

La comparaison des différentes méthodes avec l'état de l'art montre que *DOHT* a amélioré l'état de l'art sur *TUM* en segmentation (81% contre 80% pour [101] sur *TUM-S-13* et 83% contre 81% pour [101] sur *TUM-S-27*). *DOHT* surpasse *ISM* en segmentation (vu les résultats sur *TUM-S-13*, *TUM-S-27*, *UTKAD-S*, *CAD60-S*).

Ces deux résultats ont motivé la publication [20] dans le journal *Pattern recognition*.

La comparaison des différentes méthodes soutient, de plus, la pertinence de l'utilisation de primitives squelettes et notamment des suites de positions (vue les résultats sur *CAD60-C*, *TUM-C*).

L'amélioration sur l'état de l'art apportée par *SP+MKL* a motivé la publication [21].

L'utilisation de ces primitives dédiées, de la quantification et de l'algorithme *DOHT* forme un système de segmentation d'actions. C'est en un sens une réponse à l'intitulé de la thèse et donc une réponse au différentes spécificités considérées.

méthode[2]	*CAD60-C*	*TUM-C-13*	*UTKAD-C*	*MSR1-C*	*MSR2-C*	*TUM-S-13*	*TUM-S-27*	*UTKAD-S*	*CAD60-S*
SP-MKL	**88**	84	-	-	-	np	np	np	np
SP-SVM	84	80	np	76	70	np	np	np	np
P-SVM	69	72	59	62	59	np	np	np	np
P-ISM	86	62	64	-	-	55	55	62	43
P-DOHT	71	82	82	-	-	79	79	**74**	59
SP-ISM	**88**	65	np	-	-	55	55	np	43
SP-DOHT	71	85	np	-	-	80	81	np	**69**
SPVP-ISM	86	65	np	-	-	55	55	np	42
SPVP-DOHT	70	*88*	np	-	-	**81**	**83**	np	68
DTW [61]	86	76	-	54	54	np	np	np	np
HF [101]						80	81		
MMEM [84]	64								
SVM [58]	np	np	np	74	np	np	np	np	np
MKL [93]				88	**85**	np	np	np	np
HMM [98]			90	90		np	np	np	np
NN [32]			**91**			np	np	np	np
EP [91]				90		np	np	np	np
NN [103]				**91**	73	np	np	np	np

TABLE 10 – Performances obtenues vis à vis de l'état de l'art
Les résultats avant la double barre sont des implémentations naïves. Les résultats après sont simplement extraits des articles correspondants. Un tiret indique un résultat manquant et la mention np indique que l'expérience est peu pertinente (il est peu pertinent pour des algorithmes de classification de traiter des vidéos non segmentées, il est impossible d'utiliser des vitesses sur des vidéos non régulièrement échantillonnées...).

Cinquième partie

Augmenter la capacité de déploiement du système de segmentation d'actions

Les sections 21 à 23 présentent un système de segmentation d'actions couplant *primitives squelettes*, quantification par K moyennes et $DOHT$ dont les performances sont meilleures que celles de l'état de l'art sur certains jeux de données (principalement TUM). Les sections suivantes présentent un ensemble de travaux ayant comme ligne directrice l'objectif de déployer ce système dans le contexte de la surveillance pour le maintien à domicile de personnes fragiles. Ces travaux ne modifient donc pas structurellement le système présenté mais visent à permettre au système de respecter des contraintes spécifiques à un tel déploiement.

La première contrainte qui s'impose est celle d'une latence constante : le système doit garantir de pouvoir donner à l'instant $t + \delta$ l'action prédite pour l'image de l'instant t avec δ fixé à l'avance. Notamment, cela implique que le système met moins de 40 ms pour traiter une nouvelle image (sur une vidéo standard à 25 images/s). Les paramètres influant cette capacité de traitement étant multiples, cette contrainte est l'objet des sections 26 et 27.

Une deuxième contrainte qui s'impose est celle de la robustesse aux défaillances de l'algorithme d'extraction de *squelette* qui est l'objet de la section suivante.

24 Évaluation des primitives *squelettes* dans des vidéos corrompues

Dans les jeux de données de la littérature (*TUM, CAD, MSR1, MSR2, UTKAD*) les *squelettes* sont de bonne qualité : ils sont corrigés manuellement dans *TUM* et dans une moindre mesure dans *MSR1*, et les vidéos associées à un *squelette* de trop mauvaise qualité ne sont pas utilisées dans *CAD, MSR2, UTKAD*. De façon cohérente, le système présenté considère donc le *squelette* de la personne observée comme la donnée brute sur ces jeux de données.

Cependant dans une perspective de déploiement, le squelette devient lui même le simple résultat d'un algorithme d'apprentissage (comme discuté en section 16) dont les prédictions comportent des erreurs. Ces erreurs sont négligeables dans un contexte d'interface homme machine par mouvement, mais pas forcément dans le contexte désiré (comme discuté en section 21).

Cela amène à se demander comment se comporterait le système en pré-

sence de *squelettes* de mauvaise qualité et/ou de *squelettes* non extraits. La présence de *squelettes* manquants doit être considérée car les algorithmes d'extraction de *squelette* préfèrent ne pas donner de prédiction dans le cas où la confiance qu'il estime envers leur propre prédiction est trop faible.

24.1 L'état de l'art de l'impact du bruit sur les primitives *squelette*

Cette question est déjà présente dans [101] où les performances d'une *forêt de Hough* apprise sur des vidéos non bruitées sont mesurées sur des vidéos de test artificiellement corrompues par un bruit gaussien de variance croissante. Le résultat de cette expérience est encourageant vis à vis de l'utilisation de vidéos *squelettes*. Pour toutes les variances de 0 à 75mm, la *forêt de Hough* apprise a des performances stables (exceptée celle utilisant des fonctions binaires construites sur des informations de vitesse dont la performance décroit dramatiquement avec la variance). De plus, pour 100mm de variance, les performances de la *forêt de Hough* apprise sur les vidéos *squelettes* rejoignent seulement celles de la *forêt de Hough* apprise sur les vidéos classiques. Ainsi, l'utilisation de l'algorithme [101] permet d'être robuste a un bruit gaussien de forte variance. Une étude assez similaire est réalisée dans [91].

Cependant, cette étude [101] n'est pas suffisante vis à vis de notre besoin, ce qui conduit à définir un nouveau protocole qui renforce l'influence du bruit mais aussi celui de l'apprentissage. Le protocole de [101] est réducteur car il consiste à apprendre sans bruit puis à évaluer avec du bruit. Il est envisageable que la *forêt de Hough* apprise sur les données bruitées accepte une variance encore plus forte. Le nouveau protocole doit permettre d'évaluer l'algorithme quand le bruit est présent à la fois en test et en apprentissage. De plus, dans [101], le bruit ajouté est gaussien. Aussi, malgré sa forte variance il reste relativement lisse et surtout affecte autant les primitives non discriminantes que les discriminantes. Le nouveau protocole doit permettre d'évaluer l'effet d'un bruit non artificiel et donc éventuellement corrélé avec les primitives les plus discriminantes. Cela tend à renforcer l'influence du bruit.

Il n'est pas trivial de prédire le comportement du *DOHT* ou de la *forêt de Hough* avec ce nouveau protocole. Le renforcement de l'influence de l'apprentissage tend à favoriser le *DOHT*. Ce dernier est construit sur un apprentissage beaucoup plus intense, et, est donc a priori plus adapté pour apprendre la structure du bruit. Inversement, le système *DOHT* est plus fragile que celui de la *forêt de Hough*. D'une part, les primitives utilisées par la *forêt de Hough* sont beaucoup plus robustes que les *suites de positions* utilisées par le système. D'autre part, la normalisation construite sur les épaules est moins robuste que celle construite sur les distances dans [101] vis à vis du bruit. Ensuite, la quantification apprise par la *forêt de Hough* comporte plus

d'un million de parties contre une centaine pour la quantification résultante des K moyennes. En un sens, une quantification grossière va plutôt dans le sens d'une robustesse au bruit. Mais, la quantification apprise par la *forêt de Hough* n'est pas construite à partir des distances brutes entre primitives. Ainsi, parmi le million de parties de la *forêt de Hough*, il est possible que certaines capturent des informations robustes au bruit. Enfin, *DOHT* peut conduire à autoriser un mot spécifique à avoir un impact décisif dans la décision d'une classe alors que *ISM* donne la même importance à chaque mot. Ainsi, une élection utilisant des votes *ISM* est a priori plus robuste qu'une élection utilisant des votes *DOHT*.

Tout cela nous amène à faire une étude complémentaire à celle de [101] et [91] sur l'impact du bruit avec un nouveau protocole.

24.2 Protocole de mesure de l'influence du bruit

L'objectif étant de mesurer l'influence d'un bruit non synthétique, il est exclu de procéder comme dans [101]. Afin d'obtenir un bruit non synthétique, la solution la plus directe est d'utiliser la suite logicielle fournie avec la *kinect* directement sur des vidéos *kinect* (sans squelette). Cela permet d'obtenir des vidéos squelettes dans les mêmes conditions que si le système était déployé.

Cependant, on voudrait malgré tout pouvoir mesurer l'influence d'un bruit croissant. Une solution simple consiste à mesurer la qualité de chaque vidéo par le pourcentage d'images de la vidéo où le squelette est extrait. Ainsi, considérer plusieurs pourcentages ou ratios permet de considérer des bruits croissants. Le ratio des images avec squelette dans l'ensemble des images d'une vidéo est abrégé *SER* (pour *Skeleton Extraction Ratio*).

Plus précisément, l'extraction de squelette est réalisée avec la suite logicielle *NITE* qui, pour chaque articulation, indique la confiance du système envers la position estimée. Un squelette est considéré comme effectivement extrait dans une image si et seulement si pour les 13 articulations principales (mains, coudes, épaules, pieds, genoux, hanches et tête) cette confiance est considérée comme acceptable.

Jeu de données : Afin, de disposer d'un large spectre de *SER*, on utilise le jeu de données *RGBD-HuDaAct* [65]. Ce jeu de données comprend 12 classes (plus une classe de fond) d'activités humaines de la vie courante réalisées par 30 acteurs différents. Il en résulte 1189 vidéos d'environ 1 minute chacune. Chaque vidéo est associée à une et une seule classe (les vidéos sont segmentées).

Distribution de *SER* : Le nombre de vidéos ayant un *SER* donné (présentée dans la figure 13) est relativement constant dans le jeu de données (si on exclut les 160 vidéos dans lesquelles aucun squelette n'est extrait).

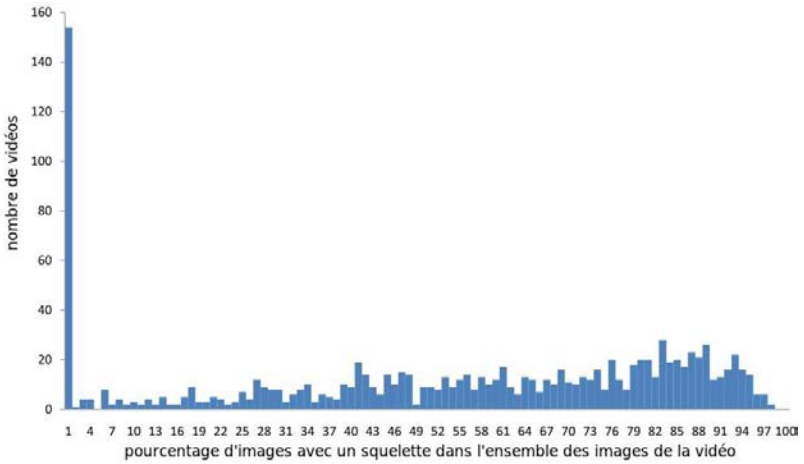


FIGURE 13 – Distribution du bruit dans les vidéos de *RGBD-HuDaAct*

La corrélation entre *SER* moyen et classes présentée en figure 14 est significative sans être prépondérante.

Ces deux distributions rendent ce jeu de données compatible avec l'objectif de l'expérience. Dans ce jeu de données homogène, il est possible de construire des sous jeux de données utilisant des vidéos d'une certaine qualité. La performance sur ces différents jeux de données permet de mesurer l'influence de la qualité des données.

Formellement, pour tous les λ de 0 à 87, on considère le sous jeu de données dans lequel on ne conserve que les vidéos dont le *SER* est supérieur à λ. Le jeu de données correspondant à $\lambda = 0$ est le jeu de donnée entier puisque l'on conserve toutes les vidéos. À l'opposé $\lambda = 87$ correspond aux 100 vidéos les moins corrompues.

Mesure de performance : Sur chacun de ces 88 jeux de données, une procédure d'évaluation du système *SP-SVM* (primitives *suite de points* + *BOW+SVM* voir section 23) est mise en œuvre. Cette procédure est la procédure *LOSO* comme pour *CAD60-C* : on apprend sur toutes les personnes sauf une sur laquelle on teste. La performance du système est mesurée par le nombre moyen (quand on fait varier la personne testée) de vidéos correctement classées sur le nombre de vidéos.

L'utilisation de *BOW+SVM* plutôt que de *DOHT* est pertinente d'une part compte tenu du nombre important de vidéos à traiter mais surtout compte tenu de la pré segmentation des vidéos. Une primitive *suite de points* est extraite uniquement quand toutes les images nécessaire à son extraction contiennent un squelette.

À la différence des expériences du tableau 10, le *SVM* prend en compte

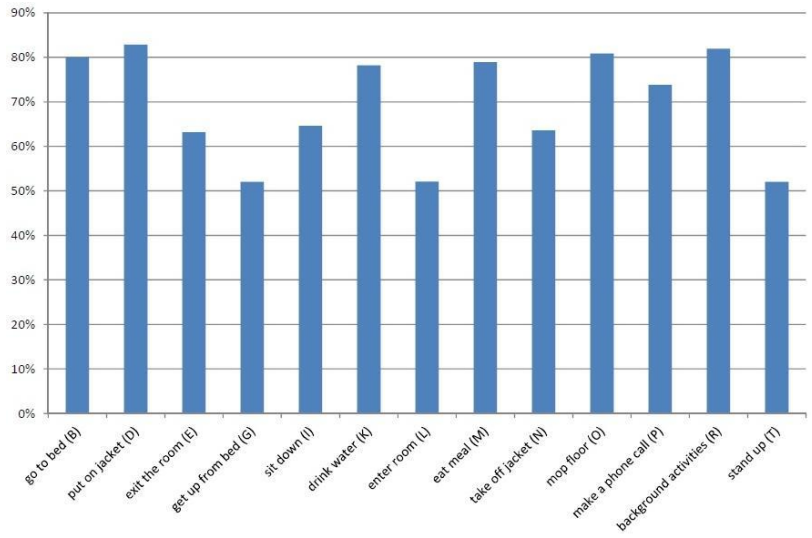


FIGURE 14 – Intensité moyenne du bruit selon les classes dans le jeu de données *RGBD-HuDaAct*

dans cette expérience le nombre d'exemples par classes (qui varie significativement pour les vidéos à fort *SER*) comme recommandé par [45] (ce qui est directement permis dans l'implémentation *LIBSVM*).

24.3 Résultats : mesure de l'impact des données manquantes

La performance du système en fonction de λ (c'est à dire du *SER* minimal imposé aux vidéos) est présentée dans la figure 15.

Performances du système : L'évolution du taux de consistance vis à vis de λ est d'une part fortement bruitée et d'autre part étonnamment décroissante pour λ grand. La décroissance pour λ grand parait interprétable comme l'effet de la réduction du nombre d'exemples.

Jusqu'à $\lambda = 69$ (correspondant à 442 vidéos), le taux de consistance suit une évolution prévisible : croissante avec $\lambda = 69$ c'est à dire qu'au moins les vidéos sont corrompues, au plus la performance est élevée. Cependant, la courbe est étonnamment stable à part pour λ proche de 0 où elle décroit brutalement. Cette stabilité du taux de consistance montre que le système est robuste aux données manquantes.

Comparaison à l'état de l'art : Pour $\lambda = 69$, l'algorithme a un taux de consistance de 82%. Afin de relier partiellement ce résultat à l'état de l'art, on peut noter que le système introduit dans [65] (utilisant principalement des *STIP* et donc non gêné par l'absence de *squelette*) obtient un taux de

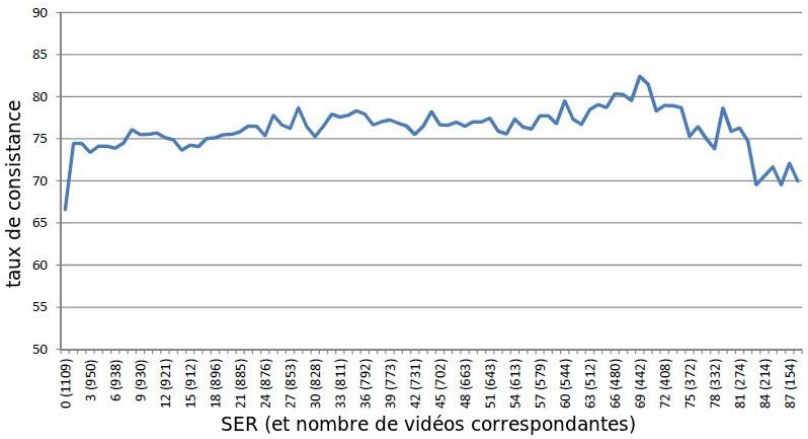


FIGURE 15 – Influence du bruit dans *RGBD-HuDaAct* sur le système

consistance de 81% sur 59% des vidéos (655 vidéos tirées aléatoirement). Ces deux résultats sont difficilement comparables. Le système présenté dans [65] n'est pas évalué sur l'intégralité des vidéos compte tenu de contraintes matérielles. Il est évalué sur des vidéos tirées aléatoirement. D'un autre coté, le système *SP-SVM* est évalué sur un certain nombre de sous ensembles de vidéos liés avec un taux de *SER*. Ainsi, ces deux systèmes n'ont pas été évalués sur des vidéos communes. Cependant, il semble que l'ordre de grandeur des performances soit proche. Cela soutient que même en présence de données manquantes, du moment que la corruption de la vidéo est modérée (typiquement de l'ordre de 30%), l'approche *SP-SVM* a une performance au niveau de l'état de l'art.

Intérêt d'apprendre sur des vidéos corrompues : Qualitativement, la robustesse de *SP-SVM* vient de la capacité de l'apprentissage de modéliser la présence de données manquantes : le système *SP-SVM* appris sur les 300 meilleures vidéos (en terme de *SER*), et testé sur les 600 meilleures vidéos suivantes, n'obtient qu'un très faible taux de consistance. Il en est de même pour le système appris sur les 600 meilleures vidéos et testé sur les 300 suivantes. Dans les deux cas, le taux de consistance est au moins de 8% en dessous de l'algorithme évalué sur les 900 meilleures vidéos. Cette expérience soutient l'intérêt d'apprendre sur une base d'apprentissage représentative de la base de test même en terme de bruit.

De plus, lors de l'évaluation *LOSO* sur les 300 meilleures vidéos, il est intéressant d'ajouter les 600 vidéos suivantes dans la base d'apprentissage. Cependant, il n'est pas clair que le gain apporté corresponde à l'ajout de bruit dans la base d'apprentissage (qui force à ne pas trop coller aux données d'apprentissage) ou à la simple augmentation de la taille de la base

d'apprentissage.

Bilan : Ainsi, malgré l'utilisation de primitives *suite de positions* et malgré une quantification grossière, le système *SP-SVM* est robuste à des vidéos présentant des *squelettes* manquants.

Ce résultat soutient la robustesse a priori du système de segmentation proposée vis à vis des échecs de l'algorithme d'extraction de *squelette* et soutient sa capacité de déploiement.

25 Dictionnaires parcimonieux

Cette section et la suivante traitent principalement de la réduction des ressources nécessaire à l'utilisation du système en apprentissage et après déploiement. Comme détaillé en section 17, la méthode d'élection nécessite en un sens le minimum possible de ressources : chaque mot donne simplement une information pour chaque action et à chaque instant voisin de son instant d'extraction (voisin au sens de la latence d'observation qu'on s'autorise).

Ainsi, le principal gain qu'on peut espérer consiste à n'extraire ou ne faire voter qu'une partie des mots. Pour faire le lien avec une approche par points d'intérêts, n'extraire qu'une partie des mots revient à utiliser une extraction creuse (c'est l'objet de la prochaine section) et n'utiliser qu'une partie des mots revient à n'utiliser qu'une partie des descripteurs (typiquement des descripteurs *SIFT*). Cela revient à n'utiliser qu'une partie du dictionnaire ou dit autrement à construire un dictionnaires parcimonieux en n'utilisant qu'une partie des dimensions (comme discuté en section 15). C'est l'objet de cette section.

25.1 Sélection a priori

Compte tenu des ressources machines nécessaires à l'apprentissage du *DOHT*, il n'est pas envisageable d'utiliser une sélection a posteriori (utilisant les résultats du détecteur comme critère de confiance en un mot). De même, afin d'être le plus proche possible d'une formulation *SVM* classique, il n'est pas possible d'utiliser une sélection simultanée (rajout d'un terme dans l'apprentissage qui vise à n'utiliser que peu des mots). Cela implique que seule une sélection a priori est envisageable (sélection des mots par un critère indépendant du score du détecteur).

Une telle sélection implique de choisir un critère pour évaluer a priori la pertinence d'un mot. Au regard de la littérature (principalement de [101]), un critère construit sur l'entropie est utilisé.

$\tau = 4$

K :	5	10	15
$C = 0.25$	70	69	67
$C = 0.5$	72	72	70
$C = 1$	74	74	73
$C = 2$	75	76	75
$C = 4$	74	76	76
$C = 8$	74	77	75

$\tau = 6$

K :	5	10	15
$C = 0.25$	70	70	704
$C = 0.5$	72	73	73
$C = 1$	75	75	76
$C = 2$	77	78	77
$C = 4$	77	79	78
$C = 8$	76	78	77

$\tau = 8$

K :	5	10	15
$C = 0.25$	70	71	70
$C = 0.5$	72	76	73
$C = 1$	74	77	76
$C = 2$	76	**80**	78
$C = 4$	77	79	78
$C = 8$	77	78	77

$\tau = 10$

K :	5	10	15
$C = 0.25$	71	72	70
$C = 0.5$	73	76	74
$C = 1$	76	78	76
$C = 2$	77	79	78
$C = 4$	76	79	78
$C = 8$	76	79	78

TABLE 11 – taux de consistance de *SP-DOHT* sur *TUM-S-13* en fonction des paramètres K, C, τ (nombre de parties pour la quantification, coefficient d'attache aux données et taille des suites de positions)

25.2 Dictionnaire initial

Avant d'appliquer l'algorithme de sélection, il convient de construire le dictionnaire initial. Dans le système proposé, le dictionnaire initial est construit simplement par l'algorithme des K moyennes sur chaque type de primitives. Par exemple pour la configuration *SP-DOHT*, pour chaque articulation, toutes les suites de position de taille τ de la base d'apprentissage sont regroupées dans un ensemble de vecteur sur lequel on applique l'algorithme des K moyennes.

Le paramètre K est donc un paramètre extrêmement important du système. Cependant, le système est relativement robuste à de petites variations de K. Le tableau 11 présente le taux de consistance de *SP-DOHT* sur *TUM-S-13*

Le paramètre K qui conduit aux meilleures performances est $K = 10$. Cette valeur de K est très faible par rapport à la valeur utilisée dans [92] ($K = 2000$) pour des trajectoires de points d'intérêts. Cet avantage semble être apporté par les primitives *squelettes* plus fragiles mais porteuses de plus d'information que les trajectoires de points d'intérêts.

Néanmoins, la quantification a lieu articulation par articulation. Ainsi, cela revient déjà (eq. (6)) à faire un apprentissage en dimension 3250 (*SP*, 25 intervalles dans le *DOHT approximé*, $K = 10$ et 13 articulations) ce qui



FIGURE 16 – Taux d'erreur en fonction de l'intensité de la sélection a priori des mots du dictionnaire dans le système présenté

La courbe jaune correspond au système *SP-DOHT* sur *TUM-S-13* avec $K = 10$. Les courbes rouge et bleu correspondent au système *SPVP-DOHT* sur *TUM-S-27* avec respectivement $K = 15$ et $K = 20$.

justifie de vouloir réduire le dictionnaire. Cela est d'autant plus pertinent avec *SPVP-DOHT* sur *TUM-S-27* où la dimension de l'apprentissage est de 30375 (*SP*, *P*, *V*, 25 intervalles dans le *DOHT approximé*, $K = 15$ et 27 articulations).

25.3 Sélection

Le résultat de la sélection a priori est résumé dans la figure 16.

Cette sélection de mots permet d'améliorer les performances du système *SP-DOHT* sur *TUM-S-13* avec $K = 10$. Cela constitue un argument envers l'hypothèse que le système soit en situation de sur apprentissage : accroître la stabilité au prix d'une erreur d'apprentissage peut conduire à une erreur globale plus faible. Dans les deux autres expériences, les performances sont croissantes avec la taille du dictionnaire alors même que le dictionnaire est beaucoup plus important que dans l'expérience précédente. Cela indiquerait une situation de sous apprentissage. Cependant, dans ces expériences, les mots sélectionnés par le critère d'entropie sont d'abord des mots liés à des primitives de position et de vitesse moins performantes que les mots liés aux suites de positions pour *DOHT*. Ainsi, cette croissance des performances est biaisée par l'utilisation croissante de primitives plus adaptées au *DOHT*.

25.4 Extraction de super mots

La sélection de mots à partir du dictionnaire initial est intéressante mais le dictionnaire initial ne contient dans la section précédente que peu des mots qu'on voudrait considérer. Notamment, il parait pertinent de s'intéresser aux positions conjointes de plusieurs articulations (déjà des paires) à un même instant. Cependant, naïvement considérer l'ensemble des paires d'articulations pour appliquer une extraction de primitives puis une quantification accroît drastiquement le nombre de mots car pour 13 articulations il y a déjà 78 paires d'articulations. Considérant que l'expérience de sélection indique un sur apprentissage, il est exclu d'essayer de sélectionner des mots provenant de paires d'articulations.

Mais, comme discuté en section 15, il est éventuellement possible de construire des super mots à partir d'un dictionnaire réduit plutôt que de sélectionner des mots à partir d'un dictionnaire vaste. D'autant que si la sélection est a priori, le résultat peut être équivalent.

Cette approche est présentée dans [91] et conduit à une augmentation significative de performances par rapport au dictionnaire initial. Cependant, malgré un important effort d'implémentation, il nous est apparu difficile de maîtriser le nombre de super mots finalement considérés (en l'occurrence *JEP* comme présenté en section 15), ce qui n'est pas acceptable dans une perspective de déploiement.

25.5 Bilan et impact sur les ressources machines

La difficulté à maîtriser finement le nombre de super mots a conduit à ne pas explorer plus loin cette idée. Par contre, la sélection a priori est intéressante dans les trois expériences sur *TUM* : dans une perspective système, utiliser une fraction (typiquement 80%) du dictionnaire initial semble pertinent étant donné que les performances sont relativement stables et que cela réduit les ressources machines nécessaires.

Cependant, utiliser 80% du dictionnaire ne revient pas à réduire de 20% le temps de calcul. Notamment, l'extraction et la quantification restent les mêmes. De plus, les mots exclus peuvent être les mots les moins souvent extraits ce qui diminue l'intérêt en test de la réduction du dictionnaire. Typiquement, dans l'expérience *SP-DOHT* sur *TUM-S-13* avec $K = 10$, la sélection de 80% du dictionnaire augmente légèrement les performances mais ne réduit que de 10% le nombre de mots extraits et donc de votes effectifs.

Cette diminution des ressources machines requises est intéressante mais relativement restreinte. Aussi, une autre approche (éventuellement complémentaire) consistant à n'extraire que certaines des primitives a été explorée.

26 Extraction creuse

Dans le système standard, un primitive est extraite chaque fois qu'elle peut l'être. Cette extraction est qualifiée de *dense*. Elle conduit typiquement pour *SP-DOHT* à extraire une suite de positions à chaque image pour chaque articulation. À l'opposé, il peut être intéressant de se donner un critère pour n'extraire que certaines des primitives.

26.1 Échantillonnage régulier

Le critère le plus simple est d'extraire une primitive non pas toutes les images mais seulement sur une fraction fixe des images. Par exemple, au lieu d'extraire une primitive par image et par articulation, il est possible d'extraire une primitive par articulation une image sur quatre. Or, cela réduit immédiatement le nombre de votes (quasiment équivalent aux temps de calcul) de 75%. Ainsi, le gain (en terme de ressources) qu'on peut espérer d'un simple sous échantillonnage des extractions est de très loin supérieur à celui qu'on peut espérer d'une réduction spécifique du dictionnaire. Cependant, la baisse de performance qu'on peut attendre est elle aussi supérieure, ce qui explique que cette approche n'ait pas été la première explorée.

Cependant, il s'est révélé que le système *SP-DOHT* sur *TUM-S-13* permet sans perte de performances significatives un sous échantillonnage de 1 primitive toutes les 8 images c'est à dire une réduction de 87% des ressources nécessaires. Ce résultat extrêmement intéressant d'un point de vue système a motivé une exploration plus poussée des modalités de sous extraction.

26.2 Sous extractions considérées

Réintroduisons certaines notations. ϱ_t est la position normalisée d'une articulation à l'instant (ou l'image) t. Le traitement étant identique pour toutes les articulations l'indice de l'articulation est omis. De même, la position peut être normalisée par rapport au repère d'une autre image mais cela est indépendant des positions utilisées ce qui nous intéresse dans cette expérience. Avec ces notations, les primitives P (pour positions) sont simplement les points ϱ_t alors que les primitives SP (pour suites de positions) sont les vecteurs $(\varrho_{t-\tau}, ..., \varrho_{t+\tau})$. ϱ_t et $(\varrho_{t-\tau}, ..., \varrho_{t+\tau})$ sont considérés extraits à l'instant t.

Les différents types d'extractions considérés sont les suivants :

Extraction dense : Tous les ϱ_t ou $(\varrho_{t-\tau}, ..., \varrho_{t+\tau})$ pour t de 1 à T (la taille de la vidéo) sont utilisés

Extraction régulière : Tous les $\varrho_{\mu t}$ ou $(\varrho_{\mu t-\tau}, ..., \varrho_{\mu t+\tau})$ pour t de 1 à $\frac{T}{\mu}$ sont utilisés. $\mu = 1$ correspond à une extraction dense, $\mu = 2$ à une

extraction toutes les deux images. Avec les primitives P, seule une image sur μ est utilisée. Avec les primitives SP, chaque image est utilisée plus d'une fois tant que $\mu < 2\tau + 1$ puisque l'image t' est utilisée par toutes les primitives extraites entre $t' - \tau$ et $t' + \tau$. Pour $\mu = 2\tau + 1$ chaque image est utilisée une et une seule fois. Pour $\mu > 2\tau + 1$ certaines images ne sont pas utilisées.

Extraction par la vitesse : Le vecteur $(\varrho_{t-\tau}, ..., \varrho_{t+\tau})$ n'est utilisé que si la moyenne des carrés des vitesses est supérieure à un seuil. Dit autrement, si $\sum_{t'=t-\tau}^{t+\tau-1} \|\varrho_{t'+1} - \varrho_{t'}\|_2^2$ est supérieure à un seuil, la primitive est extraite sinon non. Ce critère de vitesse n'est utilisé que pour l'extraction mais pas pour la description (et donc la quantification). Ce critère de vitesse tend à être corrélé avec les changements d'actions. Cette extraction (ainsi que les suivantes) n'est pas utilisée avec les primitives P.

Extraction inverse par la vitesse : Cette extraction consiste à extraire tous les vecteurs rejetés par l'extraction précédente.

Extraction construite sur les changements de gestes : Cette extraction consiste à extraire un vecteur à chaque fois que le geste de la personne d'intérêt change entre l'instant courant et un instant voisin. Les changements de gestes sont manuellement annotés. Cette méthode d'extraction n'est donc pas algorithmique car on ne connaît pas les changements de gestes à l'avance mais elle permet de donner une borne à la performance de l'action par la vitesse.

Extraction inverse construite sur les changements de gestes : Cette extraction consiste à extraire les primitives rejetées par l'extraction précédente.

Genèse de l'extraction par vitesse :
La corrélation entre vitesse et changement de gestes est un thème récurent de la segmentation non supervisée, et en particulier, a donné lieu à un stage de M2 réalisé au CEA, LIST, DIASI, laboratoire de vision et d'ingénierie des contenus par Lingagbé Perside GBEHOUNOU. Ce stage observe une adéquation pertinente (bien que bruitée) entre le critère de vitesse sélectionné et le changement de gestes et donc le changement d'actions.

26.3 Performances des différentes extractions

Le taux d'erreur de *SP-DOHT* sur *TUM-S-13* pour les différentes extractions est présenté en figure 17. Les extractions construites sur la vitesse et les gestes ne sont pas paramétrées et conduisent à une seule configuration.

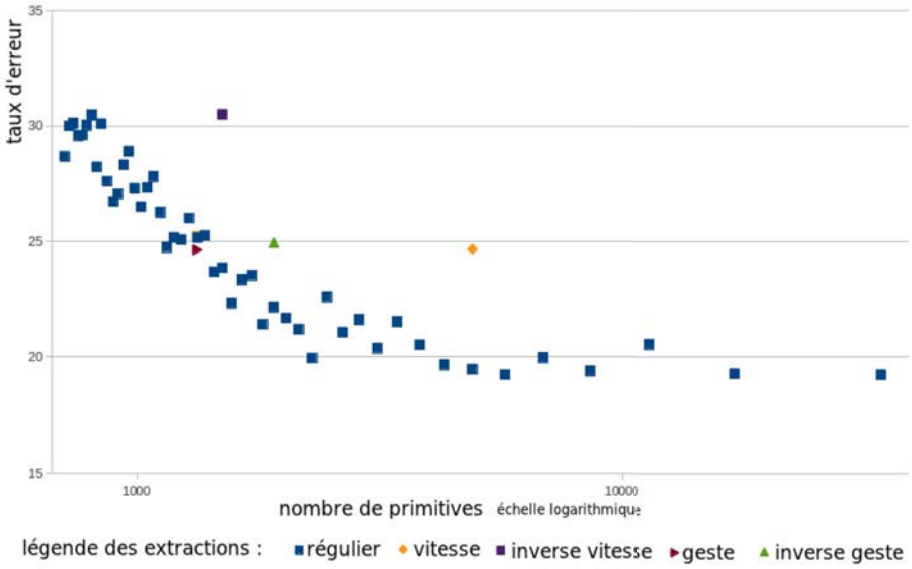


FIGURE 17 – Taux d'erreur et nombre de primitives en fonction de l'extraction sélectionnée pour le système *SP-DOHT* sur *TUM-S-13*

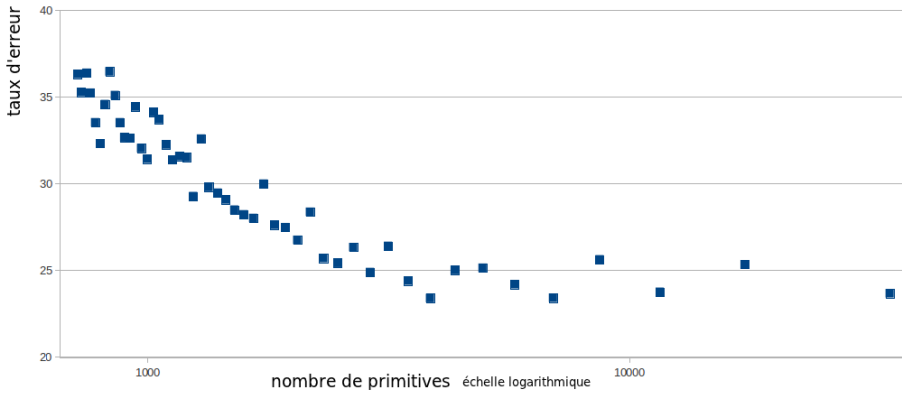


FIGURE 18 – Taux d'erreur en fonction du nombre de primitives extraites pour le système *P-DOHT* sur *TUM-S-13*

L'extraction régulière conduit à autant de configurations que de valeurs de μ considérées (la distance fixe entre 2 extractions).

Le résultat de cette expérience est que l'extraction régulière est non seulement la meilleure pour un même nombre de primitives mais surtout conduit à des performances stables pour de petites valeurs de μ. Cette stabilité des performances est aussi observable avec *P-DOHT* sur la figure 18.

Les performances de l'extraction des primitives associées à des gestes (extraits manuellement par un humain) sont étonnamment faibles vis à vis des performances de l'extraction régulière pour un même nombre de primitives. Une explication de ce phénomène est que le principe de fonctionnement du *DOHT* qui consiste à faire voter dans le temps les différents mots est sensible à la distribution temporelle des mots.

26.4 Bilan et impact sur les ressources machines

La principale conclusion de l'expérience précédente est la robustesse du système vis à vis d'un sous échantillonnage des instants d'extractions de primitives. En remarquant que le sous échantillonnage conduit à une réduction directe des ressources nécessaires au système, cette robustesse est un argument fort en faveur de la capacité d'adaptation du système aux ressources machines disponibles et donc dans sa capacité de déploiement.

Conclusion générale et perspectives

Les travaux réalisés au cours de cette thèse sont complémentaires. Premièrement, une méthode d'élection adaptée à la segmentation supervisée est proposée. Le principal intérêt de ce type de méthodes d'élection est la faible quantité de ressources machines nécessaire à son utilisation. Cependant, ce type de méthodes est fortement dépendant de la façon dont votent les éléments. La principale contribution de cette thèse est de montrer la faisabilité et l'intérêt d'utiliser les votes qui d'une part conservent une cohérence temporelle, et d'autre part, minimisent l'erreur d'apprentissage. Les performances obtenues en utilisant ces votes surpassent les votes naïfs sur des jeux de données du domaine.

Cette méthode de segmentation supervisée est ensuite appliquée au problème de la segmentation supervisée d'actions dans des vidéos. Pour cela, l'algorithme est inclus dans un système travaillant sur la position des articulations (le *squelette*) de la personne d'intérêt à chaque instant. Des primitives sur ces vidéos *squelettes* sont extraites puis quantifiées et utilisées par l'élection. Une sélection de primitives spécifiques est présentée. Le système complet améliore l'état de l'art sur un jeu de données du domaine. Cette deuxième contribution de la thèse a donné lieu à la publication d'un article dans le journal *pattern recognition* [20].

Enfin, un travail de préparation d'un possible déploiement du système de segmentation d'actions est présenté. Ce travail est constitué de deux axes. D'une part, une évaluation de la sensibilité du *squelette* soutient la pertinence de son utilisation pour déploiement. D'autre part, des approches sont mises en œuvre pour accélérer le système de segmentation supervisée d'actions. Un résultat de ce dernier travail est une réduction de 87% des ressources machines nécessaires au fonctionnement du système sans perte notables de performance sur le principal jeu de données utilisé pour les expérimentations.

Ces travaux conduisent à de nombreuses perspectives. Tout d'abord, il parait pertinent de chercher à étendre l'algorithme de segmentation temporelle présenté, à de la segmentation spatiale (pixel à pixel dans des images) ou spatio-temporelle (voxel à voxel dans des vidéos). Il n'y a aucun obstacle démontré à l'application de l'algorithme présenté à ces contextes. C'est presque l'inverse, les méthodes d'élection (et la notre en particulier) ne semblent pas avoir de rivales en segmentation spatiale ou spatio temporelle. L'écart en terme de ressources machines entre ces méthodes et leurs concurrentes, déjà favorable en segmentation temporelle, explose en segmentation spatiale (ou spatio temporelle) : les méthodes d'élection restent polynomiales alors que les méthodes concurrentes conduisent à des explosions combinatoires (sauf pour la méthode dite du *graph cut* à deux classes calculable par un simple flot).

Une autre perspective est la justification mathématique des contraintes

de cohérence temporelle proposée. On conjecture que ces contraintes font diminuer une dimension équivalente à celle de *Vapnik Chervonenkis* par rapport à la simple minimisation de l'erreur d'apprentissage (à modèle équivalent). Cette conjecture est alimentée par l'annexe 2 qui ne fait qu'ébaucher l'étude mathématique de ces contraintes.

Cette thèse invite aussi à construire un système plus complet pour la reconnaissance d'actions. Cela passe principalement par l'utilisation de l'image en plus du squelette. Pour cela aussi, il n'y a aucun obstacle démontré à l'utilisation de primitives images conjointement à l'utilisation de primitives *squelettes* : la méthode d'élection est directement capable de faire de la fusion de plusieurs modalités. Ensuite, au vu de nos expériences, le système semble gagner à utiliser (comme les forets de *Hough*) des primitives plus pertinentes que celles construites avec un algorithme des K moyennes. Ce dernier point est une perspective qui demande à être approfondie.

Une perspective plus anecdotique s'ouvre en se demandant si le système ne gagnerait pas à chercher à reconnaître des gestes élémentaires à partir du *squelette* puis les actions à partir des gestes élémentaires extraits.

Enfin, la (peut être) plus importante des perspectives ouvertes par les expériences de cette thèse est l'observation qu'extraire seulement quelques primitives très pertinentes ne permet pas de bien segmenter (expérience des super mots). Cependant, ce résultat n'est pas si paradoxal : si on est sûr que l'objet a des yeux, cela permet facilement de savoir si c'est un piéton ou un non piéton (voiture, panneau, route, paysage) mais par contre, savoir où sont les yeux ne donne que peu d'information sur la position précise des pieds. Dans un contexte de segmentation, une information de faible qualité peut être utile pour placer correctement les objets. Ainsi, l'utilisation d'un algorithme spécifique à la segmentation devrait être conjointe avec l'utilisation d'une stratégie d'extraction d'informations spécifiques à la segmentation, et notamment non parcimonieuse.

Références

[1] David Arthur and Sergei Vassilvitskii. k-means++ : The advantages of careful seeding. In *Discrete algorithms*, 2007.

[2] Sandra Avila, Nicolas Thome, Matthieu Cord, Eduardo Valle, and Arnaldo De A AraúJo. Pooling in image representation : The visual codeword point of view. *Computer Vision and Image Understanding*, 2013.

[3] Moez Baccouche, Franck Mamalet, Christian Wolf, Christophe Garcia, and Atilla Baskurt. Sequential deep learning for human action recognition. In *Human Behavior Understanding*, pages 29–39. Springer, 2011.

[4] Nicolas Ballas, Bertrand Delezoide, and Françoise J. Prêteux. Trajectory signature for action recognition in video. In *ACM Multimedia*, 2012.

[5] Yaakov Bar-Shalom and Xiao-Rong Li. *Estimation and tracking : principles, techniques, and software*. 1993.

[6] John L Barron, David J Fleet, and Steven S Beauchemin. Performance of optical flow techniques. *Journal of computer vision*, 1994.

[7] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf : Speeded up robust features. In *Europeen Conference on Computer Vision*. 2006.

[8] Samy Bengio, Jason Weston, and David Label embedding trees for large multi-class tasks. In *NIPS*, 2010.

[9] Y. Bengio and P. Frasconi. An input output hmm architecture. *Advances in neural information processing systems*, 1995.

[10] Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When is "nearest neighbor" meaningful ? In *Database Theory*. 1999.

[11] Moshe Blank, Lena Gorelick, Eli Shechtman, Michal Irani, and Ronen Basri. Actions as space-time shapes. In *Computer Vision*, 2005.

[12] Avrim Blum and John Dunagan. Smoothed analysis of the perceptron algorithm for linear programming. In *ACM-SIAM symposium on Discrete algorithms*, 2002.

[13] Aaron F. Bobick and James W. Davis. The recognition of human movement using temporal templates. *Pattern Analysis and Machine Intelligence*, 2001.

[14] Oren Boiman, Eli Shechtman, and Michal Irani. In defense of nearest-neighbor based image classification. In *Computer Vision and Pattern Recognition*, 2008.

[15] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Computational learning theory*, 1992.

[16] Christopher JC Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 1998.

[17] E Byvatov and G Schneider. Support vector machine applications in bioinformatics. *Applied bioinformatics*, 2002.

[18] Adrien Chan-Hon-Tong, Catherine Achard, and Laurent Lucat. Deeply optimized hough transform : Application to action segmentation. In *Image Analysis and Processing*, 2013.

[19] Adrien Chan-Hon-Tong, Catherine Achard, and Laurent Lucat. Transform'ee de hough sans a priori pour la segmentation. In *Groupe d'Etudes du Traitement du Signal et des Images*, 2013.

[20] Adrien Chan-Hon-Tong, Catherine Achard, and Laurent Lucat. Simultaneous segmentation and classification of human actions in video streams using deeply optimized hough transform. *pattern recognition*, 2014.

[21] Adrien Chan-Hon-Tong, Nicolas Ballas, Catherine Achard, Bertrand Delezoide, Laurent Lucat, Patrick Sayd, and Françoise Prêteux. Skeleton point trajectories for human daily activity recognition. In *International Conference on Computer Vision Theory and Application*, 2013.

[22] C.C. Chang and C.J. Lin. Libsvm : a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2011.

[23] Olivier Chapelle. Training a support vector machine in the primal. *Neural Computation*, 2007.

[24] Jose M. Chaquet, Enrique J. Carmona, and Antonio Fernández Caballero. A survey of video datasets for human action and activity recognition. *Computer Vision and Image Understanding*, 2013.

[25] Lulu Chen, Hong Wei, and James Ferryman. Readingact rgb-d action dataset and human action recognition from local features. *Pattern Recognition Letters*, 2013.

[26] Lulu Chen, Hong Wei, and James Ferryman. A survey of human motion analysis using depth imagery. *Pattern Recognition Letters*, 2013.

[27] C. Cortes and V. Vapnik. Support-vector networks. *Journal of Machine learning*, 1995.

[28] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *Transactions on Information Theory*, 1967.

[29] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition*, 2005.

[30] Sanjoy Dasgupta. *The hardness of k-means clustering.* 2008.

[31] Thomas Dean, Mark A Ruzon, Mark Segal, Jonathon Shlens, Sudheendra Vijayanarasimhan, and Jay Yagnik. Fast, accurate detection of 100,000 object classes on a single machine. In *Computer Vision and Pattern Recognition*, 2013.

[32] Maxime Devanne, Hazem Wannous, Stefano Berretti, Pietro Pala, Mohamed Daoudi, and Alberto Del Bimbo. Space-time pose representation for 3d human action recognition. In *International Conference on Image Analysis and Processing Workshop*, 2013.

[33] Luc Devroye. *A probabilistic theory of pattern recognition.* 1996.

[34] Ahmed Elgammal, David Harwood, and Larry Davis. Non-parametric model for background subtraction. In *Europeen Conference on Computer Vision.* 2000.

[35] Sergio Escalera, Jordi Gonzàlez, Xavier Baró, Miguel Reyes, Oscar Lopes, Isabelle Guyon, Vassilis Athitsos, and Hugo Escalante. Multi-modal gesture recognition challenge 2013 : Dataset and results. In *Proceedings of the 15th ACM on International conference on multimodal interaction*, 2013.

[36] Pedro Felzenszwalb, David McAllester, and Deva Ramanan. A discriminatively trained, multiscale, deformable part model. In *Computer Vision and Pattern Recognition*, 2008.

[37] Martin A Fischler and Robert A Elschlager. The representation and matching of pictorial structures. *Transactions on Computers*, 1973.

[38] Glenn M Fung and Olvi L Mangasarian. A feature selection newton method for support vector machine classification. *Computational optimization and applications*, 2004.

[39] Adrien Gaidon, Zaid Harchaoui, and Cordelia Schmid. Actom sequence models for efficient action detection. In *Computer Vision and Pattern Recognition*, 2011.

[40] Juergen Gall and Victor Lempitsky. Class-specific hough forests for object detection. In *Decision Forests for Computer Vision and Medical Image Analysis*. 2013.

[41] Ross Girshick, Jamie Shotton, Pushmeet Kohli, Antonio Criminisi, and Andrew Fitzgibbon. Efficient regression of general-activity human poses from depth images. In *International Conference on Computer Vision*, 2011.

[42] Hanlin Goh, Nicolas Thome, Matthieu Cord, and Joo-Hwee Lim. Unsupervised and supervised visual codes with restricted boltzmann machines. In *European Conference on Computer Vision*, 2012.

[43] Hanlin Goh, Nicolas Thome, Matthieu Cord, and Joo-Hwee Lim. Top-down regularization of deep belief networks. In *Advances in Neural Information Processing Systems*, 2013.

[44] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, 1988.

[45] He He and Ali Ghodsi. Rare class classification by support vector machine. In *Pattern Recognition*, 2010.

[46] M. Hoai, Z.Z. Lan, and F. De la Torre. Joint segmentation and classification of human actions in video. In *Computer Vision and Pattern Recognition*, 2011.

[47] G Hughes. On the mean accuracy of statistical pattern recognizers. *Information Theory*, 1968.

[48] Don Hush and Clint Scovel. Polynomial-time decomposition algorithms for support vector machines. *Machine Learning*, 2003.

[49] Nazlı Ikizler and Pınar Duygulu. Histogram of oriented rectangles : A new pose descriptor for human action recognition. *Image and Vision Computing*, 2009.

[50] Thorsten Joachims, Thomas Finley, and Chun-Nam John Yu. Cutting-plane training of structural svms. *Machine Learning*, 2009.

[51] G. Johansson. Visual perception of biological motion and a model for its analysis. *Attention, Perception, & Psychophysics*, 1973.

[52] Zia Khan, Tucker Balch, and Frank Dellaert. An mcmc-based particle filter for tracking multiple interacting targets. In *Europeen Conference on Computer Vision*. 2004.

[53] Alexander Klaser, Marcin Marszalek, and Cordelia Schmid. A spatio-temporal descriptor based on 3d-gradients. In *British Machine Vision Conference*, 2008.

[54] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields : Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning*, 2001.

[55] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features : Spatial pyramid matching for recognizing natural scene categories. In *Computer Vision and Pattern Recognition*, 2006.

[56] Bastian Leibe, Ales Leonardis, and Bernt Schiele. Combined object categorization and segmentation with an implicit shape model. In *Workshop on statistical learning in computer vision*, 2004.

[57] Jinyan Li. *Mining emerging patterns to construct accurate and efficient classifiers*. PhD thesis, THE UNIVERSITY OF MELBOURNE, 2001.

[58] Wanqing Li, Zhengyou Zhang, and Zicheng Liu. Action recognition based on a bag of 3d points. In *Computer Vision and Pattern Recognition Workshops*, 2010.

[59] David G Lowe. Object recognition from local scale-invariant features. In *international conference Computer vision*, 1999.

[60] Yui Man Lui and J Ross Beveridge. Tangent bundle for human action recognition. In *Automatic Face and Gesture Recognition workshop*, 2011.

[61] Fengjun Lv, Ramakant Nevatia, and Mun Lee. 3d human action recognition using spatio-temporal motion templates. *Computer Vision in Human-Computer Interaction*, 2005.

[62] Meena Mahajan, Prajakta Nimbhorkar, and Kasturi Varadarajan. The planar k-means problem is np-hard. In *Algorithms and Computation*, 2009.

[63] S. Maji and J. Malik. Object detection using a max-margin hough transform. In *Internationnal Conference on Computer Vision and Pattern Recognition*, 2009.

[64] Giorgos Mountrakis, Jungho Im, and Caesar Ogole. Support vector machines in remote sensing : A review. *ISPRS Journal of Photogrammetry and Remote Sensing*, 2011.

[65] Bingbing Ni, Gang Wang, and Pierre Moulin. Rgbd-hudaact : A color-depth video database for human daily activity recognition. In *International Conference on Computer Vision Workshop*, 2011.

[66] S.M. Oh, J.M. Rehg, T. Balch, and F. Dellaert. Learning and inferring motion patterns using parametric segmental switching linear dynamic systems. *International Journal of Computer Vision*, 2008.

[67] Mikel Olazaran. A sociological study of the official history of the perceptrons controversy. *Social Studies of Science*, 1996.

[68] Emanuel Parzen. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 1962.

[69] John Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 1999.

[70] John Ross Quinlan. *C4. 5 : programs for machine learning*. 1993.

[71] Lawrence Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 1989.

[72] Michalis Raptis, Darko Kirovski, and Hugues Hoppe. Real-time classification of dance gestures from skeleton animation. In *Eurographics Symposium on Computer Animation*, 2011.

[73] Michalis Raptis and Leonid Sigal. Poselet key-framing : A model for human activity recognition. In *Computer Vision and Pattern Recognition*, 2013.

[74] Frank Rosenblatt. The perceptron : a probabilistic model for information storage and organization in the brain. *Psychological review*, 1958.

[75] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *Acoustics, Speech and Signal Processing*, 1978.

[76] Ruslan Salakhutdinov and Geoffrey E Hinton. Deep boltzmann machines. In *International Conference on Artificial Intelligence and Statistics*, 2009.

[77] Christian Schuldt, Ivan Laptev, and Barbara Caputo. Recognizing human actions : A local svm approach. In *International Conference on Pattern Recognition*, 2004.

[78] Wei Shen, Ke Deng, Xiang Bai, Tommer Leyvand, Baining Guo, and Zhuowen Tu. Exemplar-based human action pose correction and tagging. In *Computer Vision and Pattern Recognition*, 2012.

[79] Atsushi Shimada, Kazuaki Kondo, Daisuke Deguchi, Géraldine Morin, and Helman Stern. Kitchen scene context based gesture recognition : A contest in icpr2012. In *Advances in Depth Image Analysis and Applications*. 2013.

[80] Jamie Shotton, Toby Sharp, Alex Kipman, Andrew Fitzgibbon, Mark Finocchio, Andrew Blake, Mat Cook, and Richard Moore. Real-time human pose recognition in parts from single depth images. In *Internationnal Conference on Computer Vision and Pattern Recognition*, 2011.

[81] Josef Sivic and Andrew Zisserman. Video google : A text retrieval approach to object matching in videos. In *International Conference on Computer Vision*, 2003.

[82] Frédéric Suard, Vincent Guigue, Alain Rakotomamonjy, and A Benshrair. Pedestrian detection using stereo-vision and graph kernels. In *Intelligent Vehicles Symposium*, 2005.

[83] J. Sun, X. Wu, S. Yan, L.F. Cheong, T.S. Chua, and J. Li. Hierarchical spatio-temporal context modeling for action recognition. In *Computer Vision and Pattern Recognition*, 2009.

[84] J. Sung, C. Ponce, B. Selman, and A. Saxena. Human activity detection from rgbd images. In *AAAI workshop on Pattern, Activity and Intent Recognition*, 2011.

[85] Moritz Tenorth, Jan Bandouch, and Michael Beetz. The tum kitchen data set of everyday manipulation activities for motion tracking and action recognition. In *International Conference on Computer Vision Workshops*, 2009.

[86] Weida Tong, Huixiao Hong, Hong Fang, Qian Xie, and Roger Perkins. Decision forest : combining the predictions of multiple independent decision tree models. *Journal of Chemical Information and Computer Sciences*, 2003.

[87] Jan C van Gemert, Jan-Mark Geusebroek, Cor J Veenman, and Arnold WM Smeulders. Kernel codebooks for scene categorization. In *European Conference on Computer Vision*. 2008.

[88] Vladimir Vapnik and Olivier Chapelle. Bounds on error expectation for support vector machines. *Neural computation*, 12(9) :2013–2036, 2000.

[89] Vladimir Naumovich Vapnik and Vlamimir Vapnik. *Statistical learning theory*. 1998.

[90] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition*, 2001.

[91] Chunyu Wang, Yizhou Wang, and Alan L Yuille. An approach to pose-based action recognition. In *Computer Vision and Pattern Recognition*, 2013.

[92] Heng Wang, Alexander Kläser, Cordelia Schmid, and Liu Cheng-Lin. Action recognition by dense trajectories. In *Internationnal Conference on Computer Vision and Pattern Recognition*, 2011.

[93] Jiang Wang, Zicheng Liu, Ying Wu, and Junsong Yuan. Mining actionlet ensemble for action recognition with depth cameras. In *Computer Vision and Pattern Recognition*, 2012.

[94] Alan S Willsky, Erik B Sudderth, Michael I Jordan, and Emily B Fox. Nonparametric bayesian learning of switching linear dynamical systems. In *Advances in Neural Information Processing Systems*, 2008.

[95] Paul Wohlhart, Samuel Schulter, Martin Kostinger, Peter Roth, and Horst Bischof. Discriminative hough forests for object detection. In *Conference of British Machine Vision Conference*, 2012.

[96] Christian Wolf, Julien Mille, Eric Lombardi, Oya Celiktutan, Mingyuan Jiu, Moez Baccouche, Emmanuel Dellandréa, Charles-Edmond Bichot, Christophe Garcia, and Bülent Sankur. The LIRIS Human activities dataset and the ICPR 2012 human activities recognition and localization competition. Technical report, LIRIS UMR 5205 CNRS/INSA de Lyon/Université Claude Bernard Lyon 1/Université Lumière Lyon 2/École Centrale de Lyon, 2012.

[97] David H Wolpert. The lack of a priori distinctions between learning algorithms. *Neural computation*, 1996.

[98] Lu Xia, Chia-Chih Chen, and JK Aggarwal. View invariant human action recognition using histograms of 3d joints. In *Computer Vision and Pattern Recognition Workshops*, 2012.

[99] Yuelei Xie, Hong Chang, Zhe Li, Luhong Liang, Xilin Chen, and Debin Zhao. A unified framework for locating and recognizing human actions. In *Computer Vision and Pattern Recognition*, 2011.

[100] Jianchao Yang, Kai Yu, Yihong Gong, and Thomas Huang. Linear spatial pyramid matching using sparse coding for image classification. In *Computer Vision and Pattern Recognition*, 2009.

[101] Angela Yao, Juergen Gall, Gabriele Fanelli, and Luc Van Gool. Does human action recognition benefit from pose estimation? In *British Machine Vision Conference*, 2011.

[102] Angela Yao, Juergen Gall, and Luc Van Gool. A hough transform-based voting framework for action recognition. In *Computer Vision and Pattern Recognition*, 2010.

[103] M. Zanfir, M. Leordeanu, and C. Sminchisescu. The moving pose : An efficient 3d kinematics descriptor for low-latency action recognition and detection. In *International Conference Computer Vision*, 2013.

[104] Yimeng Zhang and Tsuhan Chen. Implicit shape kernel for discriminative learning of the hough transform detector. In *Conference of British Machine Vision Conference*, 2010.

Annexe

Annexe 1 : La classification naïve bayésienne comme résultat d'une optimisation

Contexte et notations : Un algorithme de classification de multi ensembles sur M (un ensemble dans lequel un élément $m \in M$ peut être inclus plusieurs fois) est un algorithme qui, étant donné x un multi ensemble, lui associe une classe dans Y. Parmi ces algorithmes on s'intéresse à ceux qui décident en organisant une élection dans laquelle chaque mot donne une quantité de vote à chacune des classes, et, où la classe ayant le plus de vote est décidée. Une telle méthode est associée à une fonction v de $Y \times M$ vers $\mathbb{R}$ qui quantifie le vote du mot m pour la classe y.

L'apprentissage consiste à choisir une telle fonction v étant donné un ensemble d'apprentissage constitué des multi ensembles $x_1, ..., x_N$ et des classes $y^*(x_1), ..., y^*(x_N)$. Une façon d'apprendre consiste à se donner une fonction dépendante de l'ensemble d'apprentissage et de v et de l'optimiser selon v. Indépendamment, sous des hypothèses d'indépendance sur la présence des mots, il est établi que les votes optimaux sont $P(m|y)$: c'est la *classification naïve bayésienne*.

Si x est un multi ensemble sur M et $m \in M$ on note $x_{\{m\}}$ le nombre d'occurrences du mot m dans x.

Théorème : Introduisons le score donné par les mots de x à une classe y : $S(y, x) = \prod_{m \in x} v_{y,m}$. Alors, la fonction $\prod_{n=1}^{N} S(y^*(x_n), x_n)$ optimisée sous les contraintes $\forall y \in Y, \sum_{m \in M} v_{y,m} = 1$ conduit à la *classification naïve bayésienne*.

Ce théorème appartient aux connaissances générales de l'apprentissage statistique.

Démonstration :

Réduction : L'optimisation de la fonction $\prod_{n=1}^{N} S(y^*(x_n), x_n)$ sous les contraintes $\sum_{m \in M} v_{y,m} = 1$ revient à résoudre $\max_v \prod_{n=1}^{N} \left(\prod_{m \in x_n} v_{y^*(x_n),m} \right)$ sous les contraintes $\forall y \in Y, \sum_{m \in M} v_{y,m} = 1$.

En regroupant les votes identiques dans $\prod_{m \in x_n} v_{y^*(x_n),m}$, on obtient l'objectif : $\max_v \prod_{n=1}^{N} \left(\prod_{m \in M} v_{y^*(x_n),m}^{x_{n,\{m\}}} \right)$.

Ce objectif se factorise lui même en : $\max_{v} \prod_{y\in Y, m\in M} \begin{pmatrix} \sum_{x\in B_y} x_{\{m\}} \\ v_{y,m} \end{pmatrix}$ (et on revient au problème en optimisant cet objectif sous les contraintes $\forall y \in Y$, $\sum_{m\in M} v_{y,m} = 1$).

Ce problème est indépendant sur chaque y et conduit donc à résoudre pour chaque y : $\max_{v_y} \prod_{m\in M} \begin{pmatrix} \sum_{x\in B_y} x_{\{m\}} \\ v_{y,m} \end{pmatrix}$ sous la contraintes $\sum_{m\in M} v_{y,m} = 1$.

Résolution sur la forme réduite : Indépendamment, $\max_{u, \sum_i u_i = 1} \prod_i u_i^{\alpha_i}$ est équivalent à $\max_{u, \sum_i u_i = 1} \sum_i \alpha_i \log(u_i)$ qui est *équivalent* à $\max_u \sum_{i>1} \alpha_i \log(u_i) + \alpha_1 \log\left(1 - \sum_{i>1} u_i\right)$. Notons f la fonction à maximiser. Pour $i > 1$, $\frac{\partial f}{\partial u_i} = \frac{\alpha_i}{u_i} - \frac{\alpha_1}{1-\sum_{i>1} u_i}$. La où le minimum est atteint on a $\frac{\partial f}{\partial u_i} = 0$ pour $i > 1$ c'est à dire $\frac{\alpha_i}{u_i} = \frac{\alpha_1}{1-\sum_{i>1} u_i}$ pour $i > 1$. Il en suit que $\frac{\alpha_i}{u_i}$ est constant puis que pour tous i : $u_i = \frac{\alpha_i}{\sum_i \alpha_i}$.

Conclusion : Ainsi, l'optimisation de $\max_{v_y} \prod_{m\in M} \begin{pmatrix} \sum_{x\in B_y} x_{\{m\}} \\ v_{y,m} \end{pmatrix}$ sous les contraintes $\sum_{m\in M} v_{y,m} = 1$ conduit à $v_{y,m} = \frac{\sum_{x\in B_y} x_{\{m\}}}{\sum_{x\in B_y} |x|}$, c'est à dire $v_{y,m} = P(m|y)$.

Annexe 2 : Un argument soutenant les contraintes de décroissances

Contexte et notations : Soit M un ensemble fini, on note $X = M^+ = \underset{d\in\mathbb{N}}{\cup} M^{d+1}$. La longueur de la suite finie x est notée $|x| : \forall x \in X,\ \exists! d \in \mathbb{N},\ x \in M^{d+1}$ aussi on note $|x| = d+1$. On note Υ l'ensemble des fonctions y^* de X vers $Y = \{+1,-1\}^+$ telle que $\forall x \in X,\ |y^*(x)| = |x|$.

Soit $\triangleright \in \mathbb{N}\backslash\{0\}$, on note Ω l'ensemble des fonctions de $M \times [-\triangleright, \triangleright]$ vers $\mathbb{R}$.

Soit $f \in \Omega$, on note L_f la fonction de Υ tel que $\forall x \in X,\ \forall t \in [1,|x|]$:

$$L_f(x)_t = \text{signe}\left(\sum_{\tau\in[\max(1,t-\triangleright),\min(|x|,t+\triangleright)]} f(x_\tau, t-\tau)\right)$$

On définit alors un nombre qui caractérise les ensembles de fonctions de Υ. Soit $\mathcal{G}$ un tel ensemble de fonctions, alors on note $VCS(\mathcal{G})$:

$$\min\{n \in \mathbb{N},\ \forall x \in X,\ |x| \geq n \Rightarrow (\exists y^* \in \Upsilon,\ \forall L \in \mathcal{G},\ L(x) \neq y^*(x))\}$$

Bien entendu, on sousentend implicitement (mais ce n'est qu'une conjecture) que ce nombre VCS est similaire à la dimension de Vapnik Cherikov dans le contexte présenté.

Démontrer d'une part que le nombre VCS introduit est bien associé à une inégalité de type Vapnik Cherikov dans le contexte présenté, et, d'autre part, que l'ajout des contraintes de décroissance diminue la VCS, formerait une justification mathématique des contraintes de décroissance présentées dans la thèse (comme dans l'équation (5)).

Il n'a pas été possible d'arriver à un tel théorème durant la thèse. Pire, on a démontré que pour M de cardinal 2 et pour $\triangleright = 1$, l'ajout des contraintes de décroissance ne diminue pas la VCS.

Il se trouve que les contraintes de décroissance utilisées dans l'équation (5) forcent une décroissance classe par classe et non globale à toutes les classes. Or, voter au même instant pour toutes les classes permet aux votes de ne pas être globalement décroissants tout en étant décroissants classe par classe. Ainsi, l'existence de *votes doublons* qui du point de vu de l'élection n'apporte aucune information crée une différence entre la décroissance globale et la décroissance classe par classe.

Cela dit, le terme de régularité de l'équation (5) tend à pénaliser les votes doublons mais cette pénalisation ne peut pas être prise en compte par le nombre VCS. Ainsi, on propose dans cette annexe d'introduire des contraintes de décroissance globale qui bien entendu ne sont pas équivalentes (en toute généralité) aux contraintes de décroissance mais dont on conjecture qu'elles ont le même comportement pour des problèmes qui répondent à un besoin.

On note Θ l'ensemble des fonctions $f \in \Omega$ qui vérifient les contraintes suivantes : pour tout mot $m \in M$, $\forall \delta \in [1, \triangleright]$, $|f(m,\delta)| \leq |f(m,\delta-1)|$ et $\forall m$, $\forall \delta \in [-\triangleright, -1]$, $|f(m,\delta)| \leq |f(m,\delta+1)|$. Ces contraintes sont des contraintes de décroissance forte (ou globale) car la décroissance est imposée aux deux classes simultanément (ces contraintes tirent de plus partie de la spécificité du cas binaire). Remarquons que l'apprentissage avec contraintes de décroissance forte est un problème (qu'on conjecture) NP-Complet alors que l'apprentissage avec contrainte de décroissance faible est simplement un programme linéaire (il n'est donc pas possible d'utiliser ces contraintes fortes sur les problèmes étudiés dans la thèse).

On note $\mathcal{H}$ l'ensemble des fonctions L_f pour f dans Ω et $\mathcal{F}$ l'ensemble des fonctions L_f pour f dans Θ.

Il est clair que démontrer que $VCS(\mathcal{F})$ est un $o(VCS(\mathcal{H}))$ soutiendrait la pertinence des contraintes de décroissance (forte et faible). Dans le cadre de la thèse, on démontre dans un cas particulier que $VCS(\mathcal{F}) < (VCS(\mathcal{H}))$.

Théorème Pour M de cardinal 2 i.e. $M = \{\mathcal{U}, \mathcal{V}\}$ et pour $\triangleright = 1$ alors $VCS(\mathcal{F}) \leq 6$ alors que $VCS(\mathcal{H}) = 7$.

À ma connaissance, ce théorème est original.

Démonstration

$VCS(\mathcal{H}) = 7$ **:** Pour démontrer que $VCS(\mathcal{H}) \geq 7$, considérons $x_0 = \mathcal{UUUVVV}$. On va démontrer que $\forall y \in \{+1,-1\}^6$, $\exists f \in \Omega$ tel que $L_f(x_0) = y$. On va même démontrer cette propriété avec $\widetilde{L_f}$ la fonction telle que $L_f = \text{sign}\left(\widetilde{L_f}\right)$. Soit $(f(\mathcal{U},-1), f(\mathcal{U},0), f(\mathcal{U},1), f(\mathcal{V},-1), f(\mathcal{V},0), f(\mathcal{V},1))$ et v sa transposée alors $\widetilde{L_f}(x_0) = Av$ avec A la matrice

$$\begin{pmatrix} 1 & 1 & & & & \\ 1 & 1 & 1 & & & \\ & 1 & 1 & 1 & & \\ & & 1 & 1 & 1 & \\ & & & 1 & 1 & 1 \\ & & & & 1 & 1 \end{pmatrix}$$

Or, A est inversible donc pour tous y, $Av = y$ a comme solution $v = A^{-1}y$. Ainsi, $\forall y$, $\exists f = A^{-1}y \in \Omega$ tel que $L_f(x_0) = y$.

D'autre part, $VCS(\mathcal{H}) \leq 7$ car on ne peut pas avoir 7 points linéairement indépendants dans un espace de dimension 6.

Cela termine cette partie de la démonstration.

$VCS(\mathcal{F}) \leq 6$ **:** Pour démontrer que $VCS(\mathcal{F}) \leq 6$, on va considérer tous les éléments de $\{\mathcal{U},\mathcal{V}\}^6$, et, pour chacun, trouver un y qu'aucun $L \in \mathcal{F}$ ne puisse crée.

Cela dit, il n'est pas toujours évident de démontrer qu'il est impossible de créer un tel y : cela revient à montrer une incompatibilité entre différentes

inégalités. Aussi cette preuve est en grande partie une preuve assistée par ordinateur : on va utiliser la propriété que l'existence d'une solution créant un y en particulier est équivalente à l'existence d'une solution à un programme linéaire avec contraintes binaires. Comme il existe des algorithmes pour résoudre de façon exacte de tels problèmes, il est possible de s'en servir pour prouver l'inexistence de solution. Ainsi, la preuve consiste principalement à *donner ce problème linéaire avec contrainte binaire au programme lpsolve, il répond qu'il n'y a pas de solution, donc cela signifie qu'on ne peut pas créer un tel y avec $f \in \Theta$.*

Programme linéaire avec contrainte binaire Soit $x \in X$ et $y \in Y$ avec $|x| = |y|$ alors l'existence d'une fonction $f \in \Theta$ tel que $L_f(x) = y$ est équivalente à l'existence d'une solution au programme linéaire avec contrainte binaire suivant (il convient de remplacer les `y[t]` et `x[t]` par leurs valeurs en tenant compte de la numérotation qui commence à 0, tous les autres symboles sont des variables/opérateurs du programme) :

```
min: absU0 + absU1 + absU2 + absV0 + absV1 + absV2;
y[0] x[0]1 + y[0] x[1]0 >= 1;
y[1] x[0]2 + y[1] x[1]1 + y[1] x[2]0 >= 1;
y[2] x[1]2 + y[2] x[2]1 + y[2] x[3]0 >= 1;
y[3] x[2]2 + y[3] x[3]1 + y[3] x[4]0 >= 1;
y[4] x[3]2 + y[4] x[4]1 + y[4] x[5]0 >= 1;
y[5] x[4]1 + y[5] x[5]0 >= 1;
absU1-absU0 >= 0;
absU1-absU2 >= 0;
absV1-absV0 >= 0;
absV1-absV2 >= 0;
absU0-U0 >= 0;
U0-absU0 + 1000 alphaU0 >= 0;
absU0+U0 >= 0;
-U0-absU0 + 1000 betaU0 >= 0;
-alphaU0 - betaU0 >= -1;
absU1-U1 >= 0;
U1-absU1 + 1000 alphaU1 >= 0;
absU1+U1 >= 0;
-U1-absU1 + 1000 betaU1 >= 0;
-alphaU1 - betaU1 >= -1;
absU2-U2 >= 0;
U2-absU2 + 1000 alphaU2 >= 0;
absU2+U2 >= 0;
-U2-absU2 + 1000 betaU2 >= 0;
-alphaU2 - betaU2 >= -1;
absV0-V0 >= 0;
V0-absV0 + 1000 alphaV0 >= 0;
absV0+V0 >= 0;
-V0-absV0 + 1000 betaV0 >= 0;
-alphaV0 - betaV0 >= -1;
```

```
absV1-V1 >= 0;
V1-absV1 + 1000 alphaV1 >= 0;
absV1+V1 >= 0;
-V1-absV1 + 1000 betaV1 >= 0;
-alphaV1 - betaV1 >= -1;
absV2-V2 >= 0;
V2-absV2 + 1000 alphaV2 >= 0;
absV2+V2 >= 0;
-V2-absV2 + 1000 betaV2 >= 0;
-alphaV2 - betaV2 >= -1;
free U0,U1,U2,V0,V1,V2;
bin alphaU0, alphaU1, alphaU2, alphaV0, alphaV1, alphaV2,
betaU0, betaU1, betaU2, betaV0, betaV1, betaV2 ;
```

Soit le programme n'a pas de solution, soit $f(\mathcal{U}, -1) =$ `U0` et $f(\mathcal{U}, 0) =$ `U1` et $f(\mathcal{U}, 1) =$ `U2` (de même avec $\mathcal{V}$ et `V`). `absU0` est la valeur absolue de `U0` (il est plus grand que `U0` et `-U0` mais plus petit qu'un seul des deux puisqu'un seul des deux nombres `alphaU0` et `betaU0` peut être non nul).

Il suffit donc de tester pour tous les éléments de X s'il existe un y non réalisable. Il est même possible de réduire le nombre de points à tester.

Réduction Pour des raisons de symétrie entre $\mathcal{U}$ et $\mathcal{V}$, on peut supposer qu'il y a plus de $\mathcal{V}$ que de $\mathcal{U}$. De même, on peut exclure les éléments symétriques l'un de l'autre par rapport au milieu (cela revient à inverser la flèche du temps). Ensuite, si deux suites contiennent le même ensemble de motifs de 3 lettres, alors il suffit de n'en tester qu'une des deux. Enfin, tous les éléments ayant deux motifs de 3 lettres identiques n'ont pas à être testés car il suffit de choisir y qui ne donne pas la même image aux 2 motifs. Finalement, compte tenu des contraintes de décroissance forte, si une suite commence par $\mathcal{VV}$ alors le signe de $f(\mathcal{V}, 0)$ est nécessairement y_1 puisque la valeur absolue de $f(\mathcal{V}, 0)$ est supérieure à celle de $f(\mathcal{V}, -1)$ c'est donc elle qui donne son signe à la somme. De même, le signe de $f(\mathcal{V}, 0)$ est y_6 si la suite finie par $\mathcal{VV}$. Donc, il est inutile de traiter les suites commençant et finissant par $\mathcal{VV}$: il suffit que $y_1 \neq y_6$.

Éléments restants Ainsi, il ne reste à tester que 12 éléments pour lesquels on donne ci dessous un contre exemple infaisable :

$\mathcal{UVUVVV} + - - - + -$	$\mathcal{UVVVUV} + - + - - -$
$\mathcal{VUUVVV} + - + - - -$	$\mathcal{UUUVVV} + - - + - -$
$\mathcal{UUVUVV} + - + - - -$	$\mathcal{UUVVUV} + - + - - -$
$\mathcal{UUVVVU} + - - + - -$	$\mathcal{UVUUVV} + + - - - -$
$\mathcal{UVUVVU} + - - - + -$	$\mathcal{UVVUUV} + + - - - -$
$\mathcal{VUUUVV} + - - + - -$	$\mathcal{VUUVUV} + - + - - -$

Remarques

Décroissance faible Pour cette situation précise, on peut démontrer que la décroissance faible ne réduit pas la dimension VCS : il suffit de tester les 64 y possibles avec le programme linéaire de l'équation (5) avec C grand sur la suite $\mathcal{UUUVVV}$.

Cependant, on observe alors d'importants votes doublons.

Segmentation seule On peut se demander si le contexte de segmentation n'est pas la seule cause de la réduction de la dimension. C'est à dire la propriété que deux points successifs contiennent des éléments en commun (mais positionnés différemment vis à vis du centre) pourrait seule réduire la dimension.

Cependant, ce n'est pas le cas : pour tous les M et $\triangleright$, $VCS(\mathcal{H}) = |M|(2\triangleright+1)$. Il suffit de prendre $m_1...m_1m2...m2...m_{|M|}...m_{|M|}$ chaque mot apparaissant exactement $2\triangleright+1$ fois. Les différentes *fenêtres* de taille $2\triangleright+1$ sont linéairement libres entre elles.

Avec ou sans bords On peut se demander s'il est pertinent dans la définition du nombre VCS de prendre en compte les bords. Cependant, pour le cas simple, on a toujours $VCS(\mathcal{F}) \leq 6$. La réduction conduit à 11 suites, et, on peut associer à chacune d'entre elle un élément y non productible :

$\mathcal{UUUVUVVV} + + + - - +$ $\mathcal{UUUVVVUV} + + - + + -$
$\mathcal{UUVUVVVU} + + + - + -$ $\mathcal{UVUUUVVV} + + - + + -$
$\mathcal{UVUUVVUV} + + - + - +$ $\mathcal{UVUUVVVU} + + + - + -$
$\mathcal{UVVVUUUV} + + - + + -$ $\mathcal{VUUUVUVV} + + + - - +$
$\mathcal{VUUUVVUV} + + - + - +$ $\mathcal{VUUVUVVV} + + + - - +$
$\mathcal{VUUVVVUV} + + + + - -$

Annexe 3 : Contraintes de décroissance et décomposition en fonctions caractéristiques

Contexte et notations : Introduisons χ la notation des fonctions caractéristiques (ici discrètes) i.e. si I est un intervalle (discret) alors $\chi_I(u)$ vaut 1 si $u \in I$, 0 sinon. Soit $I([-\triangleright, \triangleright])$ l'ensemble des intervalles discrets I inclus dans $[-\triangleright, \triangleright]$ contenant 0 i.e.

$$I([-\triangleright, \triangleright]) = \{[\alpha, \beta] \mid -\triangleright \leq \alpha \leq 0 \leq \beta \leq \triangleright\}$$

Soit Ω l'ensemble des combinaisons linéaires positives de fonctions caractéristiques sur $I([-\triangleright, \triangleright])$ formellement $\Omega = \left\{ \sum_{I \in I([-\triangleright, \triangleright])} \omega_I \times \chi_I | \forall I, \omega_I \geq 0 \right\}$

Introduisons de même $\mathcal{F}$, l'ensemble des fonctions discrètes f de $[-\triangleright, \triangleright]$ vers $\mathbb{R}^+$ vérifiant les contraintes de décroissance $\forall u, u', \left\{ \begin{array}{l} 0 \leq u \leq u' \\ u' \leq u \leq 0 \end{array} \right. \Rightarrow f(u') \leq f(u)$. Pour tout $I \in I([-\triangleright, \triangleright])$, on introduit $\mathcal{F}_I$ l'ensemble des fonctions de $\mathcal{F}$ nulles en dehors de I.

Théorème : $\Omega = \mathcal{F}$

Ce théorème appartient aux connaissances générales de la science mais l'application de ce théorème à ce contexte particulier est, à ma connaissance, original.

Démonstration :

Inclusion des fonctions caractéristiques dans $\mathcal{F}$: Pour tout intervalle I dans $([-\triangleright, \triangleright])$, $\forall u, u' \in [-\triangleright, \triangleright]$, si $0 \leq u \leq u'$ il est impossible que $u' \in I$ et $u \notin I$, or dans tous les autres cas $\chi_I(u) \geq \chi_I(u')$. De même si $u' \leq u \leq 0$ il est impossible que $u' \in I$ et $u \notin I$ et donc $\chi_I(u) \geq \chi_I(u')$. Donc $\forall I$, $\chi_I \in \mathcal{F}$.

$\Omega \subset \mathcal{F}$ **:** $\mathcal{F}$ est stable par combinaison linéaire positive pour la simple raison que si $f(u) \geq f(u')$ et $g(u) \geq g(u')$ et $\alpha \geq 0$ alors $f(u) + \alpha g(u) \geq f(u') + \alpha g(u')$. Cette stabilité est directement vraie pour Ω qui par définition est un ensemble maximal de combinaisons linéaires positives. Notamment, Ω est l'ensemble des combinaisons linéaires positives des χ_I contenu dans $\mathcal{F}$. Donc $\Omega \subset \mathcal{F}$.

Plan de la suite de la démonstration : On va démontrer que $\mathcal{F} \subset \Omega$ par *induction* (c'est à dire par un raisonnement de type *récurrence*). On va démontrer d'une part que $\mathcal{F}_{\{0\}} \subset \Omega$ que d'autre part que pour tous $I \in I([-\triangleright, \triangleright])$, si pour tous I' strictement inclus dans I, $\mathcal{F}_{I'} \subset \Omega$ alors $\mathcal{F}_I \subset \Omega$. Donc, *par induction* cela démontre que $\mathcal{F} \subset \Omega$.

Initialisation : $\forall f \in \mathcal{F}_{\{0\}}, f = f(0)\chi_{\{0\}}$ donc $\mathcal{F}_{\{0\}} \subset \Omega$.

Mécanisme de l'induction : Pour tous $I = [\alpha, \beta]$ tel que $\forall I' \in I([\alpha, \beta]), I' \neq I$ on ait $\mathcal{F}_{I'} \subset \Omega$, considérons $f \in \mathcal{F}_I$, et notons $g = \min(f(\alpha), f(\beta))\chi_I$. Alors, $g \in \Omega$ (par définition puisque c'est une fonction caractéristique). De plus, $f - g \geq 0$ car comme $f \in \mathcal{F}_I$, $\forall u \in I$, $f(u) \geq \min(f(\alpha), f(\beta))$. Ensuite, $f - g \in \mathcal{F}$ puisque g est constante sur I et que si $f(u) \geq f(u')$ alors $f(u) - \mu \geq f(u') - \mu$. Enfin, il existe nécessairement $I' \in I([\alpha, \beta]), I' \neq I$ tel que $f - g \in \mathcal{F}_{I'}$ puisque par définition $f - g \in \mathcal{F}_I$ mais que $f - g$ est nulle soit en α soit en β. Donc, pour toutes les fonctions f considérées, il existe g telle que $f = g + (f - g)$ et telle qu'il existe $I' \in I([\alpha, \beta]), I' \neq I$ tel que $(f - g) \in \mathcal{F}_{I'}$ et $g \in \Omega$.

Induction : Pour toutes les fonctions f considérées, $\mathcal{F}_{I'}$ est inclus dans Ω, et, g est aussi incluse dans Ω donc $f = g + (f - g)$ est aussi incluse dans Ω. Donc, pour tout $I = [\alpha, \beta]$ tel que $\forall I' \in I([\alpha, \beta]), I' \neq I$ on ait $\mathcal{F}_{I'} \subset \Omega$, on a $\mathcal{F}_I \subset \Omega$ (puisque c'est vrai pour chacun des éléments individuellement).

Donc, $\mathcal{F} \subset \Omega$.