UNIVERSITÉ DE CERGY - PONTOISE

ÉCOLE DOCTORALE STIC
SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

T H È S E

pour obtenir le titre de

Docteur en Sciences

de l'Université de Cergy - Pontoise

Présentée et soutenue par

Alexis LECHERVY

Apprentissage interactif et multi-classes pour la détection de concepts sémantiques dans des données multimédia

Thèse dirigée par Philippe-Henri GOSSELIN
Co-dirigée par Frédéric PRECIOSO

préparée à ETIS dans l'équipe MIDI

soutenue le 6 décembre 2012

Jury :

Rapporteur	:	Stéphane MARCHAND-MAILLET	-	CUI (UNIGE)
Rapporteur	:	Richard NOCK	-	CEREGMIA (UNIV Antilles-Guyane)
Examinateur	:	Matthieu CORD	-	LIP6 (UPMC)
Examinateur	:	Sébastien LEFÈVRE	-	IRISA (UBS)
Examinateur	:	Véronique SERFATY	-	DGA
Directeur	:	Philippe-Henri GOSSELIN	-	ETIS (ENSEA/U-Cergy)
Co-Directeur	:	Frédéric PRECIOSO	-	I3S (UNS)

Remerciements

Je voudrai, en premier lieu remercier, MM. Stéphane Marchant-Maillet, Richard Nock , Matthieu Cord, Sébastien Lefèvre et Véronique Serfaty qui ont bien voulu me faire l'honneur de participer à mon jury de thèse. Leurs remarques et leurs questions très pertinentes aussi bien lors de la soutenance que pour la lecture du manuscrit, on permis valoriser les travaux réalisés dans ce manuscrit et ont mis en avant de bonnes perspectives d'évolution à ces travaux.

Je remercie Inbar Fijalkow directrice du laboratoire de m'avoir accueilli au sein du laboratoire ETIS et de m'avoir soutenu dans les démarches administratives. Je remercie également Sylvie Philipp-Foliguet et Dan Vodislav directeurs de l'équipe MIDI pour leur soutien.

Je tiens à remercier la DGA qui a financé cette thèse et sans qui ces travaux n'auraient pas été possible.

J'exprime également de vifs et chaleureux remerciements envers mes deux directeurs de thèse Philippe-Henri Gosselin et Frédéric Precioso. Leurs disponibilités, leurs remarques pertinentes, leurs aides, leurs soutiens et la confiance qu'ils ont bien voulu m'accorder a permis de réaliser et d'enrichir les travaux de ce manuscrit. Je suis très reconnaissant de leurs investissements dans cette thèse et ce fut un vrai plaisir de travailler avec eux.

Je remercie avec reconnaissance, les autres membres de l'équipe MIDI pour leurs aides permanentes et efficaces, notamment David Picard, Olivier Khil, Romain Négrel et Michel Jordan.

Je voudrai remercier l'équipe d'enseignement de l'ENSEA, qui m'a permis d'effectuer des enseignements au sein de leur établissement. Je tiens notamment à remercier Michel Leclerc pour son aide constante et ses conseils qui m'ont été très utiles. Je remercie également Thomas Tang et mes autres collègues qui contribué à cette expérience très enrichissante.

Je dis un grand merci à Annick Bertinotti et Anthony Carqueijeiro pour leurs aides dans toutes les démarches administratives, je suis reconnaissant de leur efficacité permanente et de leur disponibilité.

Il me faut absolument remercier l'ensemble des membres du laboratoire et en particulier les doctorants. Sans eux la vie au laboratoire aurait été différente et ils ont tous contribué à rendre ces trois années si agréable. Je vais commencé par David G., JE, Thomas, Sonia, Babar, Guy qui m'ont accueilli et qui m'ont fait apprécier dés le début la vie au laboratoire. Je remercie Rodrigue pour les quelques mois où nous avons partagé notre bureau. J'exprime de profond remerciement à Romain pour avoir été mon co-bureau la quasi totalité du reste de la thèse et pour m'avoir supporté pendant tout ce temps avec une constante bonne humeur, son aide m'a été précieuse et je lui en suis très reconnaissant. Je remercie sincèrement Gaël pour tout le temps qu'il a passé à m'aider et l'énergie qu'il a mis. Merci à Liang pour son soutien en fin de thèse et pour les nombreuses découvertes qu'elle m'a fait faire. J'ai également une pensée très amicale pour Laurent R., Ludo, JC, Leila, Mathilde, Erbao, Alan, Raouia, Yuhui, Fouad, Voisin, Laurent F., Anne et tant d'autres, qui contribuent à rendre ces années inoubliables.

Je remercie l'ensemble de ma famille qui m'accorde un soutien indéfectible dans toutes les actions que j'entreprends et qui a toujours été une source d'encouragement et de réconfort. Je dédie cette thèse à ma famille qui, sans elle, son soutien et les espérances qu'elle a toujours mis en moi, n'aurait jamais pu voir le jour.

Je remercie l'ensemble de mes amis, leur amitié m'a toujours été très précieuses et je leur dit tout simplement merci. De plus, cette thèse fut l'occasion de nouvelles rencontres et de nouvelles amitiés qui j'en suis sûr continueront également bien au de-là de ces trois années.

Enfin le dernier remerciement mais néanmoins l'un des plus important va à la personne qui m'a montré que la vie pouvait être encore plus belle et qui adoucit le pincement au cœur que j'ai eu à clore cette thèse.

Table des matières

Table des figures

Liste des tableaux

Notations

Espaces vectoriels

Scalaire : Les scalaires sont représentés par des lettres en minuscules. Par exemples : i, j, k,...

Vecteur : Les vecteurs sont représentés par des lettres en minuscules. Par exemples : u, v, w,...

Matrice : Les matrices sont représentées par des lettres en majuscules. Par exemples : A, B, C,...

Produit matriciel : Le produit matriciel entre les matrices A et B est noté AB.

Produit de Hadamard : Le produit de Hadamard entre les matrices A et B est noté $A \odot B$.

Transposé : La transposé d'une matrice A est noté $A^\top$.

L'inverse : L'inverse d'une matrice A est noté A^{-1}.

L'inverse de la transposé : L'inverse de la matrice transposé de A est noté $A^{-\top}$.

Trace : La trace d'une matrice A est noté $\mathrm{Tr}(A)$.

Rang : Le rang d'une matrice A est noté $\mathrm{rank}(A)$.

Matrice identité : On note Id_n la matrice identité de taille $n \times n$.

Matrice identiquement à 1 : On note 1_n la matrice carré $n \times n$ composé entièrement de 1.

La matrice de centrage : On note H la matrice de centrage. Par définition on a $H = \mathrm{Id}_n - \frac{1}{n} 1_n$.

Matrice centrée : Les matrices centrées sont représentées surmontées d'une barre horizontale. Par exemple : $\overline{M}$, par définition on a $\overline{M} = HMH$.

Produit scalaire : Le produit scalaire est noté $\langle ., . \rangle$. Le produit scalaire matriciel correspond au produit scalaire de Frobenius : $\langle A, B \rangle = \mathrm{Tr}(AB^\top)$.

Norme : La norme associé au produit scalaire est notée $\|.\|$. La norme associé au matrice est la norme de Frobenuis : $\|A\| = \sqrt{\mathrm{Tr}(AA^\top)}$.

Alignement : L'alignement de deux matrices A, B est noté $\mathcal{A}(A, B)$, par définition on a $\mathcal{A}(A, B) = \dfrac{\langle A, B \rangle}{\|A\|\|B\|}$.

Alignement centré : L'alignement de deux matrices A, B est noté $\mathcal{A}_{\mathbf{H}}(A, B)$. Par définition on a : $\mathcal{A}_{\mathbf{H}}(A, B) = \dfrac{\langle \overline{A}, \overline{B} \rangle}{\|\overline{A}\|\|\overline{B}\|}$.

Matrice définis positive : Les matrices définies positives sont notées $A > 0$.

Matrice semi-définis positive : Les matrices semi-définis positives sont notées $A \geq 0$.

Spectre d'une matrice On note $\mathrm{spec}(A)$, l'ensemble des valeurs propres de la matrice A.

Base d'images

m **ou** n **:** Nombre d'images dans la base.

c **:** Nombre de catégories.

Similarité

k **:** Fonction noyau.

K **:** Matrice associée à la fonction noyau k.

Apprentissage

y_i **:** Annotation de l'image i :

$$y_i \begin{cases} 1 & \text{si } x_i \text{ est positive} \\ -1 & \text{si } x_i \text{ est négative} \\ 0 & \text{si } x_i \text{ n'a pas d'annotation} \end{cases}$$

S **:** Un ensemble d'apprentissage composé d'image x_i et d'annotation y_i.

$\square^\star$ **:** On note avec une étoile tout objet cible. Par exemple une fonction cible $f^\star$, un vecteur cible $v^\star$ ou une matrice noyau cible $K^\star$.

Première partie

Recherche dans les bases d’images

CHAPITRE 1

Introduction

Sommaire

1.1 Présentation du problème

L'arrivée du numérique ces dernières décennies, a permis une multiplication des informations et des données sous format informatisé. La démocratisation des supports d'acquisition (appareils photos numériques, scanners, webcams, téléphones portables...) couplée avec la montée en puissance des capacités de calcul, de stockage et de diffusion numérique, accroît le besoin d'un système de gestion de cette masse d'information.

De grandes quantités d'information sont ainsi organisées sous forme de bases de données numériques et sont utilisées dans de nombreux domaines :

- Bases médicales ;
- Bases d'archives (patrimoine culturel, musées, bibliothèque,...) ;
- Bases d'agences photographiques, bases personnelles ;
- Bases d'images satellites et aériennes
- ...

Ces bases contiennent de plus en plus de données et sont généralement sous-exploitées par manque d'outils performants pour les gérer. Le traitement de ces informations couvre plusieurs champs de recherche :

- Le *Stockage*. La quantité de données étant de plus en importante, un système de stockage matériel et logiciel spécifique devient nécessaire. Il doit à la fois garantir l'intégrité des données ainsi qu'une bonne accessibilité pour leur utilisation.
- Le *Partage*. L'arrivée d'internet et de tous les services de partage ont montré tout l'enjeu de ce domaine. Les données doivent être à la fois sécurisées, accessibles à plusieurs utilisateurs en concurrence et protégées de toute modification indésirable.

– La *Recherche*. La quantité considérable de données ne peut pas être directement traitée par un utilisateur. Les informations utiles pour un utilisateur donné sont noyées parmi un flot d'information non-pertinente pour sa recherche. Pour retrouver une information utile, l'utilisateur a besoin d'utiliser des outils informatiques d'aide à la recherche. La nécessité de développer des techniques d'interrogation et de recherche dans des bases est devenue indispensable pour l'utilisation de ces informations.

Nous nous consacrons dans ce document uniquement à la problématique de recherche d'images dans les bases de données.

1.2 Le fossé numérique/sémantique

Les images sont stockées sous format numérique, où la représentation de l'information se fait sous format binaire. A l'inverse l'utilisateur va chercher une information à l'aide de concepts sémantiques en associant un sens aux données qu'il recherche. On distingue donc les caractéristiques perceptuelles de l'image, des caractéristiques conceptuelles. On peut ainsi définir deux niveaux d'analyse :

– l'analyse *bas-niveau* dite numérique, qui fait référence au contenu signal de l'image
– l'analyse *haut-niveau* dite sémantique, qui fait référence au contenu interprétable de l'image, autrement dit l'ensemble des objets et significations que l'on peut y associer.

Généralement il existe une importante différence entre la description bas-niveau et la description haut-niveau, on appelle cette différence d'interprétation le fossé sémantique/numérique : *"The semantic gap is the lack of coincidence between the information that one can extract from the visual data and the interpretation that the same data have for a user in a given situation."* [Smeulders 2000].

Outre le fossé qui peut exister entre la représentation numérique d'un objet et sa sémantique, une image peut contenir plusieurs significations et notamment une sémantique liée à son contexte d'observation. Par exemple [Santini 2001] fait remarquer qu'un portrait de Modigliani peut aussi bien être classé dans la catégorie "portrait" que dans la catégorie "peinture". Selon le contexte de la recherche les concepts sémantiques qui sont associés à une image peuvent donc aussi varier.

1.3 La recherche d'images

La recherche d'images peut se réaliser selon deux stratégies différentes. Soit en travaillant directement dans un espace sémantique via une recherche textuelle soit par le contenu en exploitant les données brutes des images.

1.3.1 La recherche textuelle

L'approche textuelle consiste à décrire chaque donnée de la base à l'aide d'un mot-clef qui décrit au mieux le contenu visuel de l'image. Chaque donnée de la base est donc directement placée dans un espace sémantique.

Un utilisateur peut par la suite effectuer une requête en formulant sa demande à l'aide d'un ou plusieurs termes. Le système présente alors à l'utilisateur les images qu'il évalue comme étant les plus proches par rapport aux termes proposés. La liaison entre images et requête est ainsi directement effectuée dans un espace sémantique.

Cette approche a plusieurs inconvénients. Tout d'abord elle requiert un important travail de description manuelle des images qui devient rapidement laborieux et coûteux en main d'œuvre. Néanmoins des systèmes d'annotation automatique sont possibles par exemple à l'aide de texte associée à l'image, comme la légende d'une image sur une page internet. Cependant, cette association n'a aucune garantie de fiabilité et peut être tout à fait abusive. Ce type de système est également fortement dépendant des mots clés utilisés, l'utilisateur doit en général devenir expert de la base pour pouvoir définir une requête adaptée à ses besoins.

1.3.2 La recherche par le contenu

L'approche par le contenu (Content-Based Image Retrieval CBIR) utilise un axe différent. Dans ce type d'approche on se base sur le contenu numérique des images pour en déduire la sémantique. Chaque image est représentée par un ensemble de caractéristiques visuelles telles que des couleurs, des textures ou des formes... De même, la requête utilisateur est aussi basée sur le contenu, sous la forme d'images exemples dont le contenu sert à définir le concept recherché.

Généralement la recherche par le contenu repose sur une représentation des images qui nécessite les ingrédients suivant :

- Les *primitives visuelles*. Dans cette première phase, on extrait les parties de l'image qui seront utilisées pour la décrire. On peut par exemple prendre des points, des régions ou des zones d'intérêt. Ces parties sont généralement choisies de manière à extraire l'information discriminante sur le plan psychovisuel.
- Les *caractéristiques/descripteurs visuels*. Au cours de cette phase, on décrit les zones d'intérêts à l'aide de caractéristiques visuelles telles que les couleurs, les textures, les formes...
- Les *signatures*. Les caractéristiques visuelles ne sont pas exploitables en l'état. Elles sont utilisées pour construire une signature qui sera la représentation de l'image ou de la zone traitée.
- La *fonction de similarité*. Cette fonction permet de comparer les signatures et par là même les images sur le plan visuel.

Une fois l'image décrite à l'aide de signatures, il est possible d'effectuer une recherche sur le contenu. Plusieurs types de classification sont possibles en fonction du but visé par l'utilisateur. On peut distinguer trois grandes catégories décrites par [Cox 2000] et reprises par [Smeulders 2000] :

- la recherche *associative*. Dans ce type de recherche, l'utilisateur n'a pas une idée précise de se qu'il recherche. Le système de recherche privilégiera une recherche exploratoire permettant à l'utilisateur d'affiner et de mieux définir sa requête.
- la recherche de *cible*. Pour ce type de recherche, l'utilisateur recherche une image en particulier et toute image qui lui serait très semblable. Ce type de recherche est notamment utilisé pour la détection de copies. Le système de recherche va restreindre le plus possible l'ensemble de base étudié pour retrouver les images possédant les caractéristiques recherchées par l'utilisateur.
- la recherche de *catégorie*. Dans cette thèse nous nous intéresserons plus particulièrement à ce troisième type de recherche. Dans ce cas, l'utilisateur cherche à construire un détecteur d'objet ou de concept sémantique. On peut par exemple construire un classifieur qui recherche toutes les images de voitures ou toutes les photos de vacances ...

1.3.3 La recherche de catégories

La recherche de catégories a pour objectif de scinder la base en plusieurs sous-parties regroupant un sous-ensemble d'images appelé classe ou concept. Ces images possèdent un lien sémantique défini par l'utilisateur ou par un expert.

Plusieurs approches ont été proposées pour effectuer cette catégorisation. Les premières techniques n'utilisent pas de supervision et sont entièrement automatiques. Elles s'appuient uniquement sur les caractéristiques visuelles des images et cherche à former des clusters d'images visuellement proches au sens de la similarité choisie. Les catégories sont définies ainsi automatiquement sans intervention de l'utilisateur hormis le choix des caractéristiques visuelles et de la fonction de similarité. Bien que cette méthode peut permettre de séparer certaines catégories, elle reste le plus souvent insuffisante. En effet le fossé sémantique/numérique ne peut être comblé uniquement avec des informations visuelles. Une image peut appartenir à plusieurs catégories en même temps, avoir plusieurs interprétations possibles en fonction des besoins de l'utilisateur, et être similaire à une image dans un certain contexte et différente dans un autre.

Pour réduire le fossé sémantique/numérique, les méthodes supervisées ont été introduites. Dans ces méthodes, on s'appuie sur un ensemble d'images annotées par l'utilisateur pour en déduire la sémantique recherchée par l'utilisateur. L'idée est de construire un détecteur pour chaque catégorie recherchée par l'utilisateur à partir d'un ensemble d'images appartenant ou non à la catégorie. On cherche à optimiser la fonction de similarité sur la base, pour mieux définir la frontière entre la classe désirée et le reste des données.

Généralement, on sépare la phase de construction du détecteur de sa phase d'exploitation. Néanmoins certaines méthodes combinent les deux, en construisant un détecteur " à la demande" grâce une phase de dialogue entre le système et l'utilisateur, on parle alors de recherche interactive. Dans ce type d'approche, l'utilisateur est amené à donner des précisions sur sa recherche, le plus souvent sous la forme d'exemples à annoter. Ces informations sont exploitées pour améliorer le détecteur au cours d'un processus de bouclage de pertinence (*relevance feedback*). Ainsi pour chaque session de recherche d'un utilisateur, le système construit un détecteur personnalisé, entièrement dédié aux images recherchées.

1.4 L'annotation d'ensemble multimédia

1.4.1 Les annotations

L'annotation de contenu multimédia peut se faire de plusieurs manières. La méthode la plus simple est l'annotation binaire. Pour chaque image, l'annotation spécifie si l'image appartient ou non à la catégorie recherchée. Nous désignerons par image pertinente ou positive, toute image qui appartient à la catégorie recherchée ; à l'inverse nous appellerons image non-pertinente ou négative celles qui ne remplissent pas cette condition.

Dans la littérature plusieurs techniques d'annotation ont été proposées. Ces techniques cherchent à affiner l'annotation, par exemple en autorisant les annotations à prendre des valeurs réelles entre 0 et 1 [Rui 2000]. D'autres chercheurs proposent même de nouvelles interfaces permettant d'autres types d'annotation en disposant par exemple, les images selon un plan en fonction de leurs similarités [Rubner 1999]. Les images les plus proches de la requête sont présentées à l'utilisateur sous la forme d'une mosaïque bidimensionnelle. Cette approche cherche à rendre plus fidèlement le concept de simi-

larité que l'on cherche à établir. Il permet de mieux visualiser la distance entre images dans l'espace de recherche. L'utilisateur entoure l'ensemble d'images qu'il juge pertinent. D'autres chercheurs proposent des systèmes d'interaction encore plus développés [Caenen 2000]. Ce système repose sur l'utilisation de trois fenêtres distinctes permettant à l'utilisateur d'interagir avec le système. La première fenêtre affiche un échantillon aléatoire de la base, la deuxième rappelle les images annotées tandis que la dernière projette les images annotées selon un plan 2D. L'utilisateur peut déplacer les images sur ce plan et rapprocher les images qu'il considère similaires au sens de la catégorie recherchée.

Néanmoins d'un point de vue pratique, l'utilisation de ces méthodes d'annotations plus fines est discutable. En effet, l'utilisateur n'est généralement pas dans la mesure d'établir une telle précision d'annotation. Comment évaluer précisément si une image serait à 60%, à 80% ou à 90% dans une catégorie particulière ? De même pour les méthodes disposant les images sur un plan de similarité. Il est difficile d'établir que deux images sont plus ou moins proches selon la catégorie recherchée. Dans cette thèse nous nous limiterons par conséquent à des systèmes d'annotations binaires qui peuvent éventuellement être étendus à des recherches multi-classes mais où les annotations seront toujours de nature *appartient/n'appartient pas* à la classe donnée.

1.4.2 L'apprentissage actif

L'annotation de contenu multimédia est une tâche fastidieuse qui nécessite un important investissement humain. Des stratégies, dite actives, ont été proposées afin de réduire ce travail d'annotation et impliquer l'algorithme dans cette tâche.

On peut citer par exemple les méthodes de co-apprentissage où les résultats de classification d'un premier classifieur vont permettre d'annoter les images d'entraînement d'un second classifieur. D'autres approches vont tenter de prédire les labels des images à partir de session de recherche antérieure, on parle alors d'approche collaborative. Enfin certaines méthodes établissent un vrai dialogue entre l'utilisateur et le système d'apprentissage via un système d'interaction.

CHAPITRE 2

Contributions et plan de thèse

Nous présentons dans ce chapitre les différents problèmes abordés dans cette thèse et les contributions que nous avons mises en place pour les résoudre.

2.1 Organisation du document

Ce document se développe autour de trois parties.

2.2 Première partie

Dans la première partie dont ce chapitre fait partie, nous présentons le contexte général de cette thèse et nous mettons en avant les différents enjeux auquel on fera face dans ce document. Nous aborderons, par exemple, les problématiques lié au fossé entre la représentation sémantique formulée par un utilisateur et la représentation numérique des bases de données. Nous présenterons les différentes approches possibles de recherche d'images et nous mettrons en avant les différentes techniques d'annotation des bases multimédia.

2.3 Deuxième partie

Cette partie est consacrée à la première contribution de cette thèse. Afin de présenter nos travaux nous réaliserons un état de l'art de la problématique que l'on cherche à résoudre et des méthodes desquelles s'inspire notre approche. Nous étudierons pour cela le protocole d'apprentissage interactif qui permet d'établir un dialogue entre un système d'apprentissage et ses utilisateurs. Ce type d'approche à pour viser de réduire la tâche d'annotation de l'utilisateur pour se concentrer sur des images utiles pour la classification. L'approche que nous présentons dans cette partie, s'inscrit dans les méthodes de Boosting dont un état de l'art est réalisé. Ces méthodes ont pour objectif de définir un hyperplan séparant les données d'apprentissage en fonction des annotations de l'utilisateur dans un nouvel espace de représentation des données. Ces approches sont itératives et ont été initialement conçues pour des recherches hors ligne. L'enjeu de notre approche est de proposer un algorithme de Boosting adapté aux contraintes de la recherche interactive.

2.4 Troisième partie

Cette dernière partie s'articule autour de notre deuxième contribution. Afin de mieux comprendre la méthode que nous proposons, nous avons réalisé un état de l'art sur les fonctions noyaux et notamment

sur les méthodes permettant de créer ces fonctions. Les fonctions noyaux sont des fonctions permettant de calculer un produit scalaire dans un nouvel espace de représentation des données. Elles peuvent notamment être associées à des classifieurs hyperplan afin de réaliser des classifications non linéaires. On s'intéressera plus particulièrement aux méthodes permettant la combinaison linéaire de noyaux en vue de la construction d'une fonction noyau plus performante. Il nous sera alors possible de présenter notre approche. Nous proposons une nouvelle technique de combinaison de fonctions noyaux utilisant les mécanismes du Boosting. Nous verrons au travers d'expériences toute la pertinence du modèle proposé.

2.5 Contributions

2.5.1 Apprentissage interactif par Boosting

L'annotation d'images et plus généralement de données multimédia, est une opération très coûteuse en temps humain. Il est donc important de la réduire le plus possible. Pour cela des recherches utilisant un protocole interactif ont été mises en place. Ces solutions reposent sur une interaction entre le système de recherche et l'utilisateur pour annoter uniquement les images nécessaires à l'apprentissage. Dans ces travaux nous nous sommes intéressés à étendre ce type d'approche aux méthodes de Boosting. Ces approches ont été initialement conçues pour fonctionner avec un nombre d'exemple d'apprentissage très important. L'enjeu de notre approche est donc de trouver des solutions permettant leurs utilisations avec peu d'exemples, intégrés dans un protocole d'interaction avec l'utilisateur.

La solution proposée s'inspire de l'algorithme RankBoost permettant d'établir un classement entre images. Nous avons réalisé des modifications de cet algorithme afin de l'adapter à la recherche interactive. Nous proposons notamment de nouveaux classifieurs faibles qui s'appuient uniquement sur les images annotées positivement et un critère de sélection active des futures images à annoter.

Ces travaux ont fait l'objet des publications suivantes :
[Lechervy 2010a]
[Lechervy 2010b]

2.5.2 Boosting pour la construction de noyaux

Cette partie est consacrée à la proposition d'une nouvelle méthode de création de fonctions noyaux pour l'apprentissage de catégorie dans un contexte multi-classes. Les approches classiques de la littérature pour la combinaison de fonctions noyaux sont très coûteuse en terme de complexité. Nous nous sommes donc attachés à développer une nouvelle approche dont la complexité serait linéaire. Pour cela nous avons créé un nouvel algorithme de Boosting pour la création de fonctions noyaux. Cette méthode construit l'espace où est réalisé le produit scalaire de la fonction noyau. Chaque dimension de cette espace est déterminé itérativement au cours des itérations de la méthode. Leur choix est réalisé afin d'optimiser un critère d'alignement avec une fonction noyau cible issue des labels associés aux images de la base.

Ces travaux ont fait l'objet des publications suivantes :
[Lechervy 2012a]
[Lechervy 2012c]
[Lechervy 2012b]

Deuxième partie

Apprentissage interactif

Comme nous l'avons présenté en introduction, le fossé entre représentation numérique et concept sémantique peut être comblé grâce à l'apprentissage supervisé. Cependant, pour pouvoir tirer parti de cette stratégie, il est nécessaire de disposer d'annotations, dont le coût est particulièrement important. L'une des techniques que l'on peut employer pour limiter le nombre d'annotations est d'impliquer l'utilisateur dans le processus d'apprentissage. Les méthodes ayant pour objectif d'intégrer l'utilisateur dans le processus d'apprentissage sont des méthodes dites interactives que nous aborderons dans cette partie.

Dans ce contexte, nous commencerons par présenter les enjeux et les méthodes classiques de recherche interactive ; nous aborderons ensuite les méthodes de Boosting, la théorie sur laquelle elles reposent et l'usage classique que l'on fait de ces méthodes ; nous pourrons alors présenter notre méthode de recherche interactive inscrite dans le cadre du Boosting, que nous illustrerons par des résultats expérimentaux.

CHAPITRE 3

L'apprentissage interactif

Sommaire

3.1 Principes et objectifs

Nous avons vu en introduction générale que l'un des soucis majeurs lors de la construction d'un classifieur est le fossé sémantique/numérique. Le concept sémantique que cherche à retrouver un utilisateur ne correspond pas nécessairement à une réalité numérique, il peut néanmoins être atteignable à l'aide d'exemples caractéristiques du concept recherché. Ces exemples sont fournis par l'utilisateur sous la forme d'images d'exemples lors d'une phase d'annotation. Cependant cette dernière nécessite une intervention humaine longue et coûteuse. Ainsi pour réduire les efforts humains et mieux définir les besoins de l'utilisateur, des méthodes ont été proposées pour cibler les images utiles à annoter et l'utilisateur peut guider la machine dans son travail d'apprentissage.

Les méthodes de recherche interactive ont été introduites pour répondre à cet objectif. Généralement l'utilisateur propose une première requête initiant le processus d'apprentissage. Le système établit alors un dialogue avec l'utilisateur pour affiner sa recherche et ainsi mieux converger vers les données pertinentes.

Une des stratégies de recherche interactive les plus populaires repose sur le principe de bouclage de pertinence (*relevance feedback*). Le processus d'apprentissage est initialisé à l'aide d'une requête qui permet un premier classement des images de la base selon leur similarité à la requête fournie. L'utilisateur est alors invité à donner son avis sur ce classement. Il peut soit arrêter l'apprentissage, estimant que les images qu'il recherche ont été bien retrouvées, soit continuer en fournissant plus de précisions sur la catégorie en cours de recherche. Généralement ces précisions se font à l'aide de nouvelles annotations. Le système peut alors calculer à nouveau un classement et réitérer le processus d'interrogation de l'utilisateur. L'utilisateur peut fournir de nouvelles annotations autant de fois qu'il le souhaite et à chaque nouvelle mise à jour le système recalcule la pertinence des images de la base.

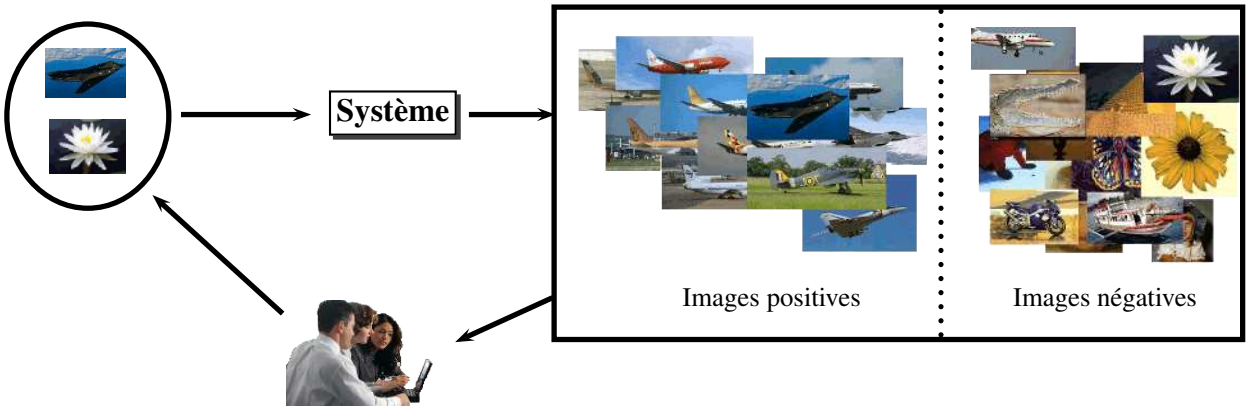


FIGURE 3.1 – Illustration de l'apprentissage interactif avec bouclage de pertinence

Ce mécanisme d'interaction est illustré par la figure (Fig. 3.1) et par les captures d'écran d'une interface graphique d'interaction avec l'utilisateur (Fig. 3.2).

3.2 Les méthodes de bouclages de pertinence

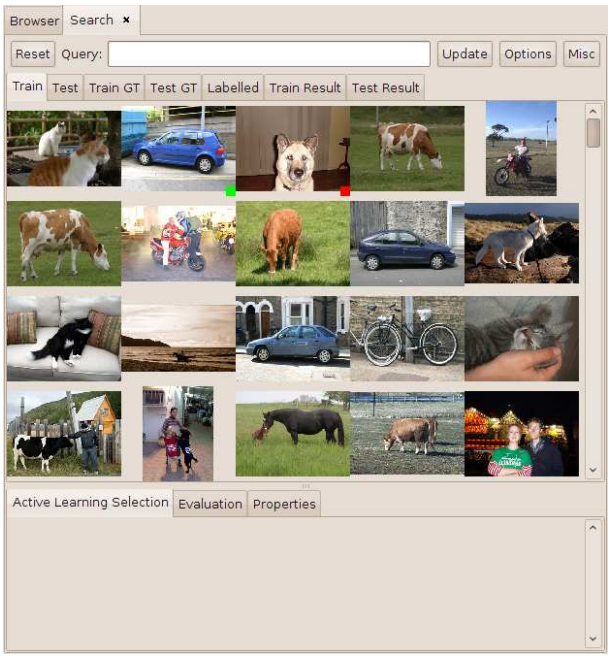
3.2.1 Méthodes issues de la recherche de texte

Les premières méthodes de recherche interactive sont issues de la recherche de texte. Dans ce type de recherche, l'utilisateur propose une *requête* au système. La *requête* est l'objet même de la recherche, il formalise la demande de l'utilisateur. La compréhension de la requête par le système va lui permettre de retrouver les objets attendus par l'utilisateur. Le concept formalisé par l'utilisateur n'étant pas toujours suffisamment précis ou clair pour le système, il fut proposé d'établir un dialogue avec ce dernier pour affiner la requête et ainsi construire une requête plus fine et adaptée au fonctionnement de l'algorithme d'apprentissage.

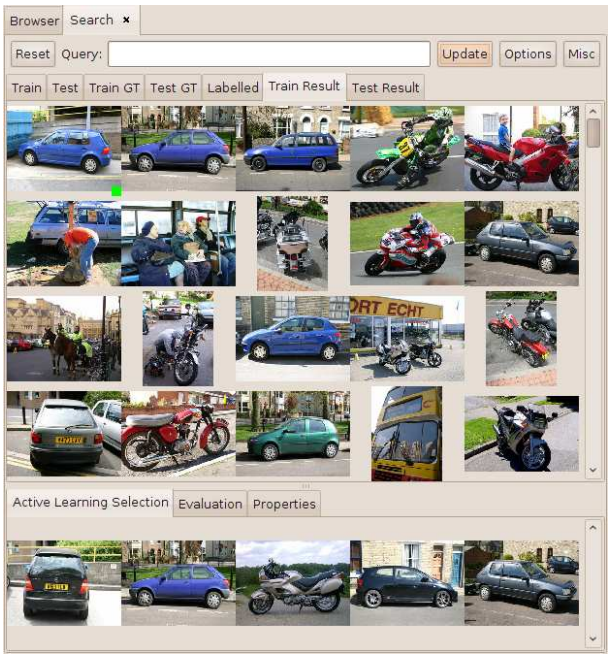
Pour la recherche d'images, la requête initiale peut être par exemple une image semblable au concept que l'on souhaite faire apprendre au système. Le but du système sera de modifier cette requête en fonction des annotations suivantes de l'utilisateur.

La solution la plus simple pour mettre à jour cette requête consiste à calculer la moyenne des signatures de l'ensemble des images annotées comme pertinentes. Cette stratégie est connue sur le nom de *query modification* (QM) et fut introduite pour les images par [Rui 1997].

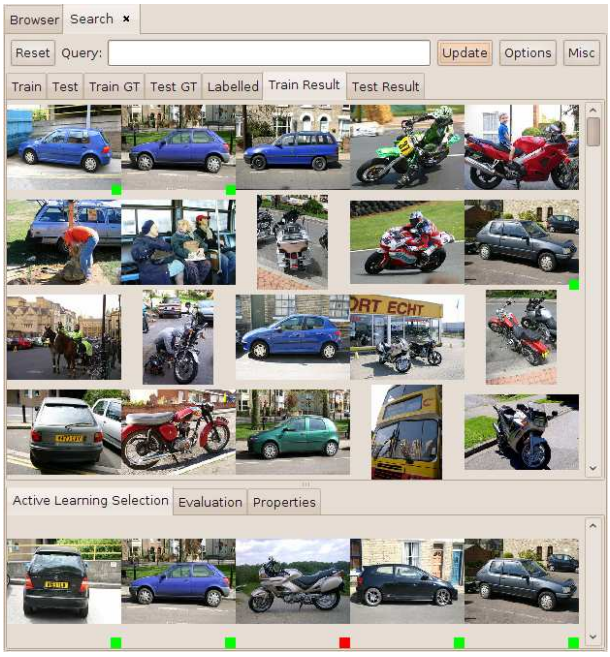
Une autre approche possible est non plus de modifier la requête mais la fonction de similarité comparant les images entre elles. L'idée est de modifier la répartition des exemples afin de mieux concentrer les images pertinentes et d'augmenter l'écart avec les images non-pertinentes. On parle généralement de méthode de type *query reweighting* (QR). On peut par exemple, pondérer les axes des attributs décrivant les images en fonction du rapport entre l'écart-type des images pertinentes et celui de toutes les images [Aksoy 2000]. Ainsi les axes pertinents correspondent aux axes où la variance des images pertinentes est plus faible. D'autres pondérations des axes sont possibles, ainsi [Heinrichs 2000] pondèrent les attributs par rapport au rang moyen calculé sur l'ensemble des images pertinentes et non-pertinentes. L'intuition est qu'un attribut discriminant tend à classer les images pertinentes parmi les premiers résultats tandis que les images non-pertinentes seront plus éloignées dans le classement final.



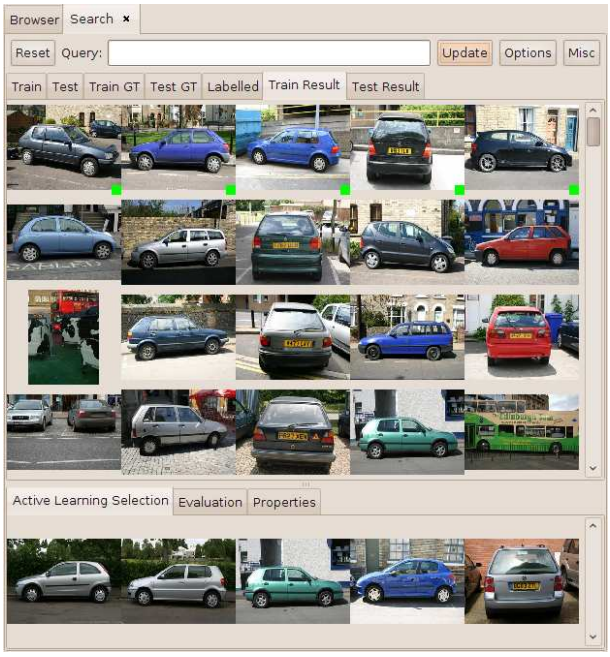
État initial, l'utilisateur lance l'interaction en annotant une image positivement et une négativement.



Le système renvoie un premier classement et des propositions d'images à annoter.



L'utilisateur annote les images proposées par le système et relance le processus.



Après plusieurs itération, le système converge vers la classe recherchée par l'utilisateur.

FIGURE 3.2 – Exemple d'une chaîne d'apprentissage interactif exécuté à l'aide de l'outil développé dans cette thèse.

3.2.2 Méthodes basées optimisation

Les méthodes par optimisation sont une évolution des méthodes précédentes. Dans ces méthodes, on cherche à optimiser la similarité en fonction d'un critère mathématique mesurant généralement l'erreur entre la classification proposée par le système et les annotations fournies par l'utilisateur [Huang 2001]. Ces méthodes, comme les méthodes précédentes, exploitent les annotations pour modifier les paramètres de la fonction de similarité mais le font non pas en fonction d'une heuristique mais suivant l'optimisation d'un critère mathématique.

On peut citer plusieurs applications dans le domaine de la recherche d'images. [Peng 1999] mesurent ainsi la pertinence locale des attributs en fonction d'un critère de réduction de l'erreur de classification bayésienne, [Doulamis 2001] minimisent l'erreur quadratique moyenne sur les images annotées, et [Fournier 2001] rétropropagent l'erreur quadratique entre la similarité réelle et la similarité désirée...

3.2.3 Méthodes probabilistes

Les méthodes probabilistes vont calculer la probabilité qu'une image donnée corresponde ou non à la requête de l'utilisateur. Pour chaque image x, ces méthodes estiment la probabilité $P(x = x_r|H_t)$ que l'image corresponde à la requête x_r connaissant les informations H_t fournies par l'utilisateur au cours de t itérations précédentes. L'utilisateur peut fournir des informations sous la forme de label binaire mais également sous la forme d'annotations relatives du type : "cette image est plus pertinente que celle-ci". Ces approches reposent sur une utilisation de la loi de Bayes qui permet d'estimer la probabilité recherchée en fonction des données des itérations précédentes.

Les premiers travaux utilisant un modèle bayésien de bouclages de pertinence sont les travaux de [Müller 1999] et le système *PicHunter* de [Cox 2000]. L'idée est de prendre en compte les changements d'objectif de l'utilisateur au cours de la recherche. L'information donnée par l'utilisateur via le bouclage de pertinence est une information bruitée voir incohérente d'une itération à l'autre. Les auteurs de ces méthodes ont proposé de pondérer les différents bouclages par un degré de confiance favorisant les informations les plus récentes et la cohérence entre itérations. [Geman 2000] approfondit cette idée et considère le bouclage de pertinence comme un processus aléatoire bruité. La métrique comparant les images n'est plus unique pour tout type de comparaison mais dépend de la cible recherchée et des images que proposent l'utilisateur. Le processus de modélisation de la pertinence repose sur une séquence de métriques indépendantes générées aléatoirement, correspondant à des pondérations différentes sur les attributs.

3.2.4 Méthodes par classification

Jusque-là les méthodes par retour de pertinence se sont intéressées à faire évoluer soit la requête de l'utilisateur, soit la fonction de similarité permettant la comparaison des images entre elles. Cependant il est également possible de faire intervenir le retour de pertinence dans des problèmes de classification binaire.

En recherche d'images diverses solutions ont été proposées sur la base de techniques d'apprentissage statistique. [Vasconcelos 2000] s'intéresse à la classification par critère de Bayes, [Berrani 2003] se consacre aux méthodes par k-Plus-Proches-Voisins, tandis que [Najjar 2003] étudie les méthodes par mélanges de gaussiennes, [Chang 2003, Chapelle 1999, Tong 2001, Saux 2003] les méthodes par SVM et [Collins 2008, Lu 2007, Li 2004] les méthodes de Boosting.

3.3 Les méthodes d'apprentissage actif

Le choix des images à annoter influe directement sur les résultats du classifieur que l'on construit avec. La suite d'exemples choisis impactera directement sur la qualité du résultat final. En effet en fonction de l'ensemble d'apprentissage, certaines parties de la classe seront mieux représentées et plus finement décrites tandis que d'autres peuvent être sous représentées ou même absentes. Ce choix est d'autant plus important dans le contexte de la recherche interactive. L'utilisateur dirige le système à l'aide de ses annotations, il est donc important d'annoter des images pertinentes pour l'évolution des résultats de classification. En effet une suite d'annotations peut conduire à de bien meilleurs résultats qu'une autre. Par exemple si l'utilisateur annote un exemple qui est actuellement bien classé par le système, le gain sera généralement nul, cet exemple est alors non-pertinent.

L'*apprentissage actif* est un concept introduit dans les méthodes d'enseignement pour améliorer la maîtrise des connaissances des élèves. Freinet a écrit en 1964 : "La voie normale de l'acquisition n'est nullement l'observation, l'explication et la démonstration, processus essentiels de l'école, mais le tâtonnement expérimental, démarche naturelle et universelle" [Freinet 1964]. L'approche interactive a pour but de guider l'élève vers la connaissance en le mettant face à des exemples afin qu'il se forge sa propre expérience. Le rôle du professeur est de choisir les exemples qui permettront au mieux à l'élève d'atteindre les objectifs pédagogiques.

Rapporté à notre contexte de recherche interactive, l'apprentissage actif consiste à se demander quels seraient les exemples les plus pertinents à présenter à l'utilisateur afin de faire évoluer le système le plus rapidement possible. Il pose le problème de la sélection des images à faire annoter à l'utilisateur et permet de fixer un cadre formel et d'exprimer mathématiquement le problème de la sélection.

Diverses solutions ont été proposées dans ce cadre. On peut néanmoins en détacher deux grandes familles :

- Les méthodes dites *pessimistes*, ces techniques cherchent à réduire l'ensemble des annotations possibles en minimisant l'espace de possibilités. Ce sont des méthodes exploratoires qui parcourent l'ensemble des possibilités. [Tong 2001] proposent ainsi de diviser en deux l'espace des classifieurs possibles pour chaque nouvelle annotation. Ces méthodes garantissent certaines propriétés sur l'espace des possibilités restantes en fonction du nombre d'itérations réalisées, néanmoins en pratique ces techniques souffrent d'une convergence souvent lente à cause de la taille très importante de l'espace des possibilités à parcourir.
- Les méthodes dites *optimistes*, ces techniques contrairement aux précédentes n'ont pas de garantie de convergence, elles reposent sur un "pari". Ces méthodes proposent d'annoter les images qui une fois annotées de façon correcte permettent d'améliorer le plus les résultats de la classification. Elles sélectionnent les images qui, à condition qu'elles soient associées à la bonne annotation, maximiseront un critère d'optimisation. Si l'utilisateur ne donne pas l'annotation attendue, le classifieur n'évoluera pas de manière significative. [Roy 2001] proposent ainsi de choisir les exemples qui ont le plus de potentiel à généraliser la classification. [Park 2000] proposent de choisir des exemples orthogonaux à l'ensemble des exemples déjà annotés mais proches de la frontière de décision. [Lindenbaum 2004] choisissent les exemples les plus "utiles". [Vijayanarasimhan 2009] utilise une méthode active dépendant de la précision des annotations.

Certaines méthodes actives dépendent directement du type de classifieur. Par exemples, [Park 2000, Tong 2000] proposent des méthodes adaptées aux classifieurs SVM, [Hasenjager 1996] aux méthodes par k-Plus-Proche-Voisins (kPPV) et [Collins 2008, Li 2004, Lu 2007] aux techniques de Boosting.

3.4 Synthèse

Nous avons présenté dans ce chapitre le contexte de la recherche interactive et les différents outils qui sont nécessaires pour mettre en place ce type d'approche. Nous avons vu ainsi que pour établir un dialogue entre le système et ses utilisateurs, une solution efficace est de faire un retour sur la classification et affiner la recherche grâce à l'annotation de nouveaux exemples. Pour cela, le système doit disposer d'une méthode adaptée pour présenter ces nouveaux exemples à l'utilisateur. Cela ce fait généralement au moyen d'une approche de sélection active des prochaines images à annoter.

Les méthodes de recherche interactive ont été adaptées aux différentes techniques de classification. Nous avons voulu dans cette thèse nous consacrer plus particulièrement aux méthodes de type Boosting. Même si ces méthodes peuvent requérir des temps d'apprentissage longs, elles sont très rapides en classification, offrent de bons résultats en pratique et ont de bonnes propriétés théoriques. Le nombre restreint d'images d'apprentissage dans les protocoles interactifs, nous permettra de nous affranchir de la contrainte du temps d'apprentissage.

Nous présentons dans la partie suivante les méthodes de Boosting et certains résultats théoriques qui leur sont associés. Nous pourrons ensuite proposer notre méthode qui fera le lien entre le protocole de recherche interactive et les méthodes de Boosting.

CHAPITRE 4

Introduction au Boosting

Sommaire

4.1 Cadre théorique

4.1.1 Introduction

Lors de la construction d'une nouvelle fonction de classification, on peut constater qu'il est généralement plus facile de définir des règles simples de classification mais peu performantes, que de construire un classifieur complexe permettant de résoudre le plus de cas possibles. Les approches par Boosting s'appuyent sur ce constat et cherchent à établir des règles simples pour les combiner en vus d'obtenir de bonnes performances de classification.

Prenons un exemple, imaginons que l'on cherche à construire une fonction permettant de savoir si un cheval donné va gagner la prochaine course d'un grand prix. Dans un premier temps, on peut étudier toutes les courses précédentes, les chevaux, les conditions climatiques... et chercher à déduire qui sera

le prochain vainqueur. Seul un expert, après des années d'expérience et de travail pourra donner une prédiction performante. Cette stratégie est donc coûteuse et demande beaucoup d'investissement. Une autre stratégie possible consiste à demander à chacun quelle est sa "technique" pour trouver le prochain vainqueur. Bien sûr ces techniques prisent séparément sont bien moins performantes que la technique précédente. Chaque personne rencontrée nous donne ses astuces qui lui permettent de gagner de temps en temps, mieux qu'un simple tirage aléatoire, mais moins bien que les années d'expérience d'un expert.

D'un côté on a donc un expert qu'il est difficile de former et de l'autre on dispose de plein d'avis meilleurs que le hasard, faciles à trouver mais peu performants. On peut alors se demander si on ne pourrait pas exploiter tout ces avis pour construire un classifieur aussi bon, voir meilleur que l'avis de notre expert. On aurait alors réussi à construire un classifieur à moindre coût.

Les méthodes de Boosting s'intéressent à cette problématique : comment combiner un ensemble de règles peu performantes, à peine meilleures que le hasard, pour obtenir une nouvelle règle très performante ? On réduit ainsi le besoin de paramétrage de la méthode d'apprentissage. Les connaissances d'un expert pour obtenir des résultats performants ne sont plus nécessaires.

Les techniques de Boosting ont été pour la première fois utilisées en recherche d'images par [Viola 2001]. Les auteurs proposent d'utiliser différents classifieurs faibles à base d'ondelettes de Haar pour la détection de visage dans des images. Le Boosting a été par la suite utilisé à maintes reprises pour la classification d'images [Tieu 2000, Torralba 2007, Yan 2007, Nock 2012].

4.1.2 L'apprenabilité probablement approximativement correcte, (PAC)-apprenabilité

Nous avons précédemment introduit intuitivement les questions que cherchent à résoudre les méthodes de Boosting. Un cadre plus formel a été proposé pour modéliser cette problématique.

En 1984 [Valiant 1984] introduit le cadre PAC (Probably Approximately Correct) pour l'apprentissage automatique. Ce modèle permet de représenter les algorithmes d'apprentissage à l'aide de trois entités :

- Une population d'exemples Ω. Chaque exemple $\omega \in \Omega$ est décrit à l'aide d'un vecteur x de t attributs : $x =< a_1, ..., a_t >$
- Une fonction d'association à un concept sémantique aussi appelé concept cible. Cette fonction f associe une classe sémantique à un ensemble d'individu de Ω. Elle induit une partition sur Ω.
- Un oracle, annotant un ensemble fini d'exemples de Ω tirés indépendamment selon une distribution inconnue mais figée D. L'oracle permet de créer un ensemble d'apprentissage $S = \{(x_i, y_i)|i \in 0...m\}$ en associant une étiquette à des exemples de Ω.

L'objectif d'une méthode d'apprentissage est de construire un modèle de f à partir d'un échantillon d'apprentissage S fourni par l'oracle. On appellera par la suite $\mathcal{H}$ ce modèle. On dispose également d'une fonction $err(h)$ mesurant l'erreur réelle du modèle par rapport au concept cible.

Dans ce contexte on peut définir deux cadres d'apprentissage :

Définition 1. *Une classe de problèmes F sur une population Ω sera dite PAC-apprenable au sens fort s'il existe une méthode d'apprentissage A telle que pour tout problème f de F, pour toute distribution D sur Ω, pour tout $(\delta, \rho) \in]0; 1/2]^2$, A est capable de fournir en temps polynomial une hypothèse h telle qu'avec une probabilité de $1 - \delta$, l'erreur de h soit inférieure à ρ :*

$$\Pr\left(err(h) < \rho\right) > 1 - \delta.$$

Définition 2. *Une classe de problèmes F sur une population Ω sera dite PAC-apprenable au sens faible s'il existe une méthode d'apprentissage A telle que pour tout problème f de F, pour toute distribution D sur Ω, il existe $\delta \in]0;1]$ et $\gamma \in]0;1/2]$ tel que A est capable de fournir en temps polynomial une hypothèse h telle qu'avec une probabilité $1-\delta$, l'erreur de h soit inférieure à $\frac{1}{2}-\gamma$:*

$$\Pr\left(err(h) < \frac{1}{2} - \gamma\right) > 1 - \delta.$$

Kearns et Valiant [Ehrenfeucht 1988] posent alors le problème du Boosting comme étant : *un problème PAC-apprenable au sens faible est-il également PAC-apprenable au sens fort ?* En d'autres termes, peut-on construire un classifieur fort à l'aide d'hypothèses plus faibles ?

Le Boosting introduit par [Kearns 1988] et rendu possible par [Schapire 1990] est une méthode ayant pour but initial de montrer l'équivalence entre la PAC-apprenabilité forte et la PAC-apprenabilité faible et ainsi répondre par l'affirmatif à la question de Kearns et Valiant.

L'étude du Boosting a évolué ensuite vers des algorithmes à visée pratique et non plus seulement théorique. Le premier algorithme qui popularise le Boosting est AdaBoost [Freund 1999] que nous présenterons par la suite.

4.1.3 L'erreur empirique

L'erreur réelle $err(h)$ du classifieur peut s'exprimer à l'aide d'une fonction de risque. Cette erreur s'écrit alors sous la forme suivante d'une fonction de risque réel $L(h)$:

$$err(h) = L(h) = \int \lambda(h(x), y) \mathrm{d}\Pr(x, y),$$

avec λ une fonction de perte. On peut, par exemple utiliser la fonction 0/1-loss : $\lambda\left(h(x), y\right) = I(yh(x) \leq 0)$. I étant la fonction indicatrice. Cette fonction prend la valeur 1 lorsque la prédiction de h et le label attendu y pour un exemple x sont de signe contraire et 0 sinon.

En pratique cette erreur n'est pas calculable et on utilise à la place le risque empirique $\hat{L}$ obtenu à l'aide de l'ensemble d'apprentissage :

$$\hat{L}(h) = \frac{1}{m} \sum_{n=1}^{m} \lambda(h(x_n), y_n).$$

Dans le cas où la fonction 0/1-loss est utilisée, l'erreur empirique $\hat{L}(h)$ correspond au nombre d'exemples mal classés par la fonction de décision h.

Sur la base de l'erreur empirique, il est possible d'estimer l'erreur réelle, par exemple en s'appuyant sur la dimension de Vapnik-Chervonenkis (dim-VC). Cette dimension est une mesure de la complexité d'un ensemble de fonction de classification. Sa valeur correspond à la taille du plus grand ensemble d'exemples que peut pulvériser [1] l'ensemble des fonctions de classification.

Les travaux de Vapnik et Chervonenkis ont ainsi montré qu'avec une forte probabilité :

$$err(h) \leq e\hat{r}r(h) + \tilde{O}\left(\frac{VCdim(\mathcal{H})}{m}\right),$$

1. générer l'ensemble des 2^m partitions possibles sur les labels par des classifieurs

où $VCdim$ est la dimension de Vapnik, $\hat{err}(h)$ l'erreur empirique, m est le nombre d'exemples utilisés pour l'estimation de l'erreur empirique et $\mathcal{H}$ est l'ensemble des fonctions que l'algorithme d'apprentissage peut produire. Ainsi plus un classifieur est complexe au sens de la dimension de Vapnik, plus la différence entre l'erreur réelle et l'erreur empirique est potentiellement élevée. A l'inverse plus le nombre d'exemples d'apprentissage est important, plus l'erreur empirique représente fidèlement l'erreur réelle.

Le théorème de Glivenko-Cantelli [Vapnik 1995] (page 230 section 7.3) dans les cas où la distribution des exemples d'apprentissage est uniforme sur Ω, assure que l'erreur réelle de la meilleure hypothèse h construite à partir de S tend à être égale à l'hypothèse optimale h^* modélisant f lorsque le nombre d'exemples tend vers l'infini :

$$\forall \delta \leq 1, \varepsilon \geq 0, \exists m, \Pr\left(|err(h_m) - err(h^*)| > \varepsilon\right) < \delta, \tag{4.1}$$

h_m est une hypothèse d'apprentissage construite à partir de m échantillons.

Si le cardinal de $\mathcal{H}$ est fini, on a également :

$$\forall \varepsilon > 0,\ \Pr\left(\max_{h \in \mathcal{H}} |\hat{err}(h_m) - err(h_m)| \geq \varepsilon\right) < 2|\mathcal{H}|e^{-2\varepsilon^2 m}, \tag{4.2}$$

cette équation est une conséquence de l'inégalité de Hoeffdings (cf. Eq. 5.11 page 130 et les équations Eq. 5.23 et Eq. 5.26 page 135 de [Schölkopf 2002]).

L'erreur empirique et l'erreur réelle sont également liées par une formule dépendant du nombre d'exemples utilisés pour l'apprentissage.

Ces formules montrent que pour construire une hypothèse modélisant la fonction recherchée, avec notamment une bonne fiabilité sur l'erreur réelle de classification, il est important d'avoir un grand nombre d'exemples d'apprentissage permettant d'entraîner le classifieur étudié.

4.2 Classification binaire avec Adaboost

AdaBoost (Adaptive Boosting) fut introduit par Yoav Freund et Robert Schapire [Freund 1999]. Cette méthode repose sur le principe de sélection itérative de classifieurs faibles en fonction des exemples d'apprentissage.

Un poids est associé à chaque exemple pour l'erreur que commet le classifieur final à son sujet. Le poids des exemples est utilisé dans le calcul de l'erreur de classification des classifieurs faibles. Plus un exemple est actuellement mal classé, plus son poids est fort et plus son importance dans le calcul de l'erreur des classifieurs faibles sera prépondérante. Les classifieurs faibles sont choisis pour minimiser leur erreur de classification.

Au cours des itérations, l'algorithme va donc se concentrer sur les exemples qu'il a du mal à bien classer et se désintéresser progressivement des exemples qui sont toujours bien classés. Les exemples mal classés prennent de plus en plus d'importance dans le choix des futurs classifieurs faibles sélectionnés.

AdaBoost est l'un des algorithmes de Boosting les plus populaires, beaucoup de méthodes de Boosting s'appuient sur ses propriétés. Nous allons donc l'étudier de façon plus précise et les remarques que l'on pourra faire sur cette méthode sont généralement extensibles aux autres algorithmes de Boosting.

4.2.1 Présentation de l'algorithme

L'algorithme Adaboost est décrit dans l'Algorithme. 1. Il prend en entrée un ensemble d'images annotées et produit un classifieur fort H_T correspondant à une somme pondéré de classifieurs plus faibles h_t :

$$H_T(x) = \sum_{t=1}^{T} \alpha_t h_t(x). \tag{4.3}$$

Pour chaque image d'apprentissage un label 1 ou -1 y est associé et indique ainsi si l'image concernée est ou non dans la classe que l'on souhaite modéliser. Dans la version la plus simple de l'algorithme, un ensemble fixé de classifieurs faibles est également fourni. Les classifieurs faibles sont des fonctions qui à une image associent la valeur 1 ou -1. Cette valeur correspond à la prédiction de la catégorie de l'image testée par le classifieur. La sélection itérative des classifieurs qui permettent de construire le classifieur fort final est effectuée dans cet ensemble. Il est également possible d'apprendre ces classifieurs au cours des itérations en prenant en compte par exemple, la pondération des exemples d'apprentissage.

L'algorithme débute par une initialisation de la pondération des exemples d'apprentissage. Sans à priori, chaque image de la base d'apprentissage possède la même difficulté. En conséquence, chaque image est initialisée avec la même pondération correspondant à la probabilité de tirer cette image au hasard selon un tirage uniforme.

On lance ensuite les itérations de l'algorithme. On commence par calculer l'erreur sur chacun des classifieurs faibles de l'ensemble de sélection. Cette erreur correspond à la somme pondérée des erreurs sur chaque image du classifieur faible h_j :

$$\varepsilon_j = \sum_i D_t(x_i) 1_{h_j(x_i) \neq y_i}. \tag{4.4}$$

Cette erreur prend uniquement en compte le poids des exemples mal identifiés par le classifieur, c'est à dire les images dont le résultat de classification diffère du label qui leur est associé.

Le classifieur h_t minimisant cette erreur est sélectionné, sa pondération est alors calculée par la formule :

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \varepsilon_t}{\varepsilon_t} \right). \tag{4.5}$$

Ce dernier est ajouté au classifieur fort que l'on construit.

Le poids de chacun des exemples est alors mis à jour par :

$$D_{t+1}(x_i) = \frac{D_t(x_i) e^{-\alpha_t y_i h_t(x_i)}}{Z_t}, \tag{4.6}$$

avec Z_t un facteur de normalisation. L'algorithme passe ensuite à l'itération suivante et recommence toutes ces étapes.

Le classifieur fort final correspond à la somme pondérée de chaque classifieur faible sélectionné au cours des itérations de la méthode :

$$H(x) = \text{sign} \left(\sum_{t=0}^{T-1} \alpha_t h_t(x) \right). \tag{4.7}$$

4.2.2 Résultats sur l'erreur empirique

La convergence de l'erreur empirique vers 0 au fil des itérations d'AdaBoost est garantie. On va montrer dans la suite de cette partie qu'AdaBoost minimise l'erreur empirique définie par :

$$\hat{err}(h_t) = \sum_i 1_{h_t(x_i) \neq y_i}. \tag{4.8}$$

Schapire et Singer [Schapire 1990] ont montré que l'erreur empirique est bornée par le produit des coefficients de normalisation $Z_t = \sum_i d_t(x_i) \exp(-\alpha_t y_i h_t(x_i))$ des distributions des exemples :

$$\hat{err}(H_t) = \frac{\sum_i 1_{H_T(x_i) \neq y_i}}{m} \leq \frac{1}{m} \sum_{i=1}^{m} \exp(-y_i f(x_i)) = \prod_{t=1}^{T} Z_t. \tag{4.9}$$

L'objectif d'un algorithme de Boosting est donc de minimiser les coefficients Z_t.

Soit W_t^+ et W_t^- la somme des poids correctement classés et mal classés :

$$W_t^+ = \sum_{(x_i, y_i) \in S | y_i h_t(x_i) = 1} d_t(x_i) = 1 - \hat{err}(h_t), \tag{4.10}$$

$$W_t^- = \sum_{(x_i, y_i) \in S | y_i h_t(x_i) = -1} d_t(x_i) = \hat{err}(h_t). \tag{4.11}$$

Si on exprime Z_t en fonction de ces deux variables, on a :

$$Z_t = \sum d_t(x_i) \exp(-\alpha_t y_i h_t(x_i)) = W_t^- \exp(\alpha_t) + W_t^+ \exp(-\alpha_t). \tag{4.12}$$

Algorithm 1: Algorithme Adaboost

Input: une base d'exemples annotés $S = \{(x_1, y_1), \cdots, (x_m, y_m)\}$.
Input: des labels $y_i \in \{-1, 1\}$, 1 pour les positifs 0 sinon.
Input: des classifieurs faibles $h_1, \cdots, h_N$.
begin
Initialisation de la distribution des exemples $D_0(x_i) = \frac{1}{m}$, $i = 1, \cdots, m$
for $t = 1 \ldots T$ **do**
Trouver le classifieur h_t minimisant le critère d'erreur :

$$\varepsilon_t = \sum_i D_t(x_i) 1_{h_t(x_i) \neq y_i}.$$

Calculer le poids du classifieur h_t correspondant à l'erreur minimale ε_t :

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \varepsilon_t}{\varepsilon_t} \right).$$

Mettre à jour la distribution de chacun des exemples :

$$D_{t+1}(x_i) = \frac{D_t(x_i) e^{-\alpha_t y_i h_t(x_i)}}{Z_t},$$

$$\text{avec } Z_t = \sum_i D_t(x_i) e^{-\alpha_t y_i h_t(x_i)}.$$

Output: la somme pondérée des classifieurs sélectionnés :

$$H(x) = sign \left(\sum_{t=1}^{T} \alpha_t h_t(x) \right).$$

Comme on cherche à minimiser Z_t en fonction de α_t on a :

$$\frac{\partial Z_t}{\partial \alpha_t} = 0 \tag{4.13}$$

$$\Longrightarrow \quad W_t^- \exp(\alpha_t) - W_t^+ \exp(\alpha_t) = 0 \tag{4.14}$$

$$\Longrightarrow \quad \exp(2\alpha_t) = \frac{W_t^+}{W_t^-} \tag{4.15}$$

$$\Longrightarrow \quad \alpha_t = \frac{1}{2} \ln \frac{W_t^+}{W_t^-}. \tag{4.16}$$

D'où la définition de α_t :

$$\alpha_t = \frac{1}{2} \ln \frac{W_t^+}{W_t^-} = \frac{1}{2} \ln \frac{1 - e\hat{r}r(h_t)}{e\hat{r}r(h_t)}.$$

De plus on a :

$$Z_t = 2\sqrt{W_t^- W_t^+} = 2\sqrt{e\hat{r}r(h_t)(1 - e\hat{r}r(h_t))}.$$

Freud et Schapire montrent que l'erreur empirique peut décroître de manière exponentielle avec le nombre d'itérations du Boosting, ce que nous détaillons dans la suite de ce chapitre.

Définition 3. *Soit la variable γ_t définie par :*

$$\gamma_t = W_t^- - W_t^+. \tag{4.17}$$

γ_t mesure la qualité du classifieur par rapport à un classifieur qui effectuerait un tirage aléatoire.

Plus γ_t est proche de 1, plus les réponses du classifieur sont correctes, inversement si γ_t est proche de -1, le classifieur classe incorrectement les données d'apprentissage. A zéro le classifieur correspond à un classifieur effectuant un tirage aléatoire.

On peut alors redéfinir nos équations avec γ_t :

$$W_t^+ = \frac{1 + \gamma_t}{2}, \tag{4.18}$$

$$W_t^- = \frac{1 - \gamma_t}{2}, \tag{4.19}$$

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \gamma_t}{1 + \gamma_t} \right). \tag{4.20}$$

Si on suppose que le classifieur classe mieux que le hasard alors $\gamma_t > 0$ et :

$$Z_t = \sqrt{1 - \gamma_t^2}. \tag{4.21}$$

D'autre part, on a :

$$e\hat{r}r = W_t^- = \frac{1}{2} - \frac{\gamma_t}{2}. \tag{4.22}$$

Ce qui donne :

$$e\hat{r}r = \prod_t Z_t = \prod_t \sqrt{1 - \gamma_t^2} < \exp\left(- \sum_t \gamma_t^2 \right). \tag{4.23}$$

L'erreur empirique du classifieur décroît donc bien de manière exponentielle suivant les itérations du Boosting.

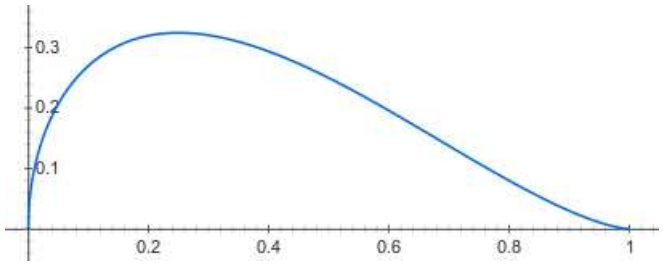


FIGURE 4.1 – Évolution de la majoration $e\hat{r}r(h_t)^{1-0.5}(1-e\hat{r}r(h_t))^{1+0.5}$ en fonction de $e\hat{r}r(h_t)$

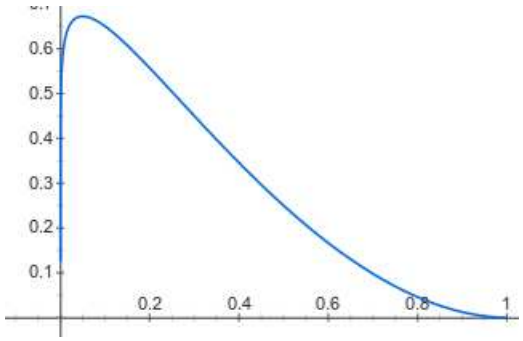


FIGURE 4.2 – Évolution de la majoration $e\hat{r}r(h_t)^{1-0.9}(1-e\hat{r}r(h_t))^{1+0.9}$ en fonction de $e\hat{r}r(h_t)$

4.2.3 Résultats sur l'erreur réelle

La marge d'un exemple x_i traduit la confiance dans la prédiction de f_T. Dans le cas du Boosting, elle est comprise entre $[-1, 1]$ et correspond à la formule :

$$m_f(x_i, y_i) = \frac{y_i f_T(x_i)}{\sum_t \alpha_t} = \frac{y_i \sum_t \alpha_t h_t(x_i)}{\sum_t \alpha_t}. \tag{4.24}$$

Schapire montre que :

$$\forall \theta \ \Pr(y_i f_T(x_i) \leq \theta) \leq 2^T \prod_{t=1}^{T} \sqrt{e\hat{r}r(h_t)^{1-\theta}(1-e\hat{r}r(h_t))^{1+\theta}}. \tag{4.25}$$

Adaboost a donc tendance à réduire le nombre d'exemples ayant une faible marge. Il maximise donc la marge. De plus, cette équation montre l'importance d'avoir des classifieurs suffisamment "faibles" pour obtenir une marge maximale. Cette remarque est illustrée par les figures (Fig. 4.1) et (Fig. 4.2), ces images montrent l'évolution du terme de la majoration pour un classifieur faible en fonction de son erreur sur l'ensemble d'apprentissage.

Nous étudierons dans les sections qui suivront de manière plus détaillée la marge des classifieurs par Boosting et nous verrons comment AdaBoost est un cas particulier de méthode de Boosting plus général reposant sur ce concept de marge.

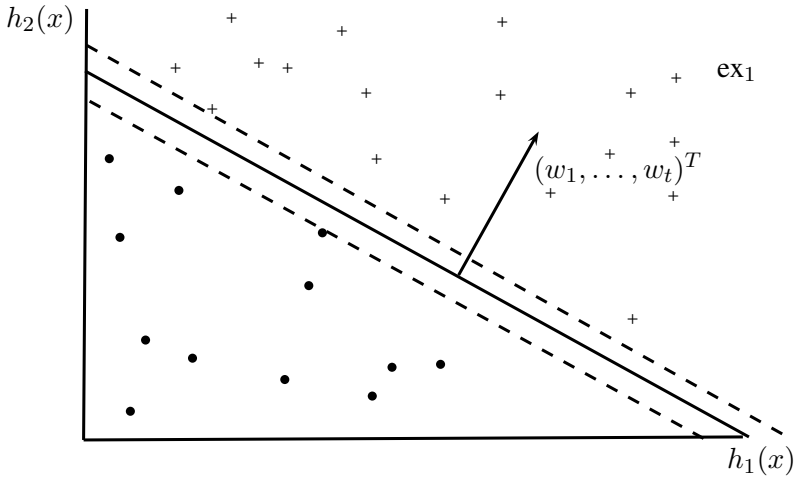


FIGURE 4.3 – Exemple d'hyperplan dans un Boosting à deux classifieurs faibles

4.3 Interprétations théoriques du Boosting pour la classification

Le Boosting repose sur des bases théoriques fortes qui permettent une interprétation de cette famille d'algorithmes selon plusieurs angles. En fonction de l'axe théorique choisi, diverses variantes en découlent. Nous allons dans cette partie présenter divers angles d'approche possibles du Boosting afin de donner un bref aperçu des possibilités théoriques qu'offre ce type de méthode. Nous montrons tout d'abord que le Boosting permet de créer des classifieurs par hyperplan. Nous verrons ensuite les liens possibles avec la théorie des jeux, ainsi qu'avec les problèmes d'optimisation. Nous présenterons le Boosting comme une méthode de descente de gradient, une méthode travaillant avec un espace dual et le lien avec les approches probabilistes. Les méthodes développées dans cette thèse s'appuient sur plusieurs visions du Boosting que nous présentons dans cette partie.

4.3.1 Le Boosting, une distance à un hyperplan

Au cours d'un processus d'apprentissage par Boosting, on construit un classifieur fort composé de T classifieurs faibles. On peut alors définir le vecteur $H(x)$ correspondant aux résultats de classification de chaque classifieur faible sélectionné pour l'exemple x :

$$H(\mathrm{x}) = [h_1(\mathrm{x}), \cdots, h_T(\mathrm{x})]. \tag{4.26}$$

Soit $\phi(\mathrm{x})$ le mapping entre les exemples et l'espace induit par H :

$$\phi(\mathrm{x}) = \mathrm{x} \longmapsto \begin{bmatrix} h_1(\mathrm{x}) \\ h_2(\mathrm{x}) \\ \vdots \end{bmatrix}. \tag{4.27}$$

Cette notion de changement d'espace qu'offre le Boosting est importante et nous nous appuierons dessus pour développer l'approche que nous proposons dans la troisième partie.

La décision du classifieur fort consiste à déterminer le signe de :

$$f(\mathrm{x}) = \sum_{t=1}^{T} w_t h_t(\mathrm{x}) = \mathbf{w}^T H(\mathrm{x}) = \langle \phi(\mathrm{x}), \mathbf{w} \rangle. \tag{4.28}$$

On notera par la suite x pour $\phi(\mathrm{x})$.

On s'intéresse à séparer les positifs des négatifs par un hyperplan A défini par la normale w. Un exemple est donné par la figure (Fig. 4.3) dans le cas où l'on dispose de deux classifieurs faibles. La marge en norme l_p d'un exemple (x_n, y_n) est définie par :

$$\rho_i^p(w) = \frac{y_i \langle x_i, w \rangle}{\|w\|_p}. \tag{4.29}$$

Selon cette définition une marge positive correspond à un exemple bien classé, tandis qu'une marge négative est associée aux exemples mal classés.

La marge à l'hyperplan A est définie comme la distance minimale sur tous les exemples :

$$\rho^p(w) = \min_i |\rho_i^p(w)|. \tag{4.30}$$

Les études théoriques des méthodes d'apprentissage par hyperplan ont montré que la maximisation de la marge permettait une bonne généralisation du classifieur. Dans notre cadre maximiser la marge du classifieur revient donc à résoudre le problème :

$$\max_w \rho^p(w) = \max_w \min_{1 \leq i \leq m} \frac{|y_i \langle x_i, w \rangle|}{\|w\|_p}. \tag{4.31}$$

On peut également exprimer cette formule avec la norme duale de l_p.

Soit l_q la norme duale de l_p ($\frac{1}{p} + \frac{1}{q} = 1$), on a :

$$\frac{|\langle x, w \rangle|}{\|w\|_p} = \|x - A\|_q, \tag{4.32}$$

avec :

$$\|x - A\|_q = \min_{\tilde{x} \in A} \|x - \tilde{x}\|_q. \tag{4.33}$$

Cette équation montre que calculer la projection de x sur le vecteur w normal à l'hyperplan en norme p revient à calculer la distance avec le point le plus proche de x sur l'hyperplan en norme q.

Nous avons montré dans cette partie que le Boosting est une méthode de classification par hyperplan dans l'espace induit par des classifieurs faibles h_t. A ce titre on peut s'intéresser à l'étude du comportement des algorithmes de Boosting en fonction de la marge, ce que nous ferons dans les parties suivantes.

4.3.2 Théorie des jeux

4.3.2.1 Présentation de la théorie des jeux

Le Boosting peut être vu comme la réalisation d'un jeu à deux joueurs et par la même peut s'intégrer dans les études sur la théorie des jeux. Avant de présenter le "jeu du Boosting", nous allons faire quelques rappels sur la théorie des jeux.

La théorie des jeux s'intéresse aux problèmes de stratégie dans le cas où deux joueurs (ou plus) influencent par leur choix, le jeu de leur(s) adversaire(s). Les méthodes qui découlent de cette théorie ont pour objectif de définir une stratégie de jeu optimale pour chacun des joueurs.

Dans notre cas, on se placera dans la branche consacrée aux jeux synchrones à répétition. Ces jeux se caractérisent par la simultanéité des coups de chaque joueur et la répétition des phases de jeu avec une prise en compte des résultats antérieurs.

Dans ce cadre le jeu peut être représenté par une matrice de gain, représentant les gains/pertes des deux joueurs en fonction des actions possibles.

Pour illustrer nos propos on peut prendre pour exemple le jeu pierre-feuille-ciseaux. Pour ce jeu on a une matrice de gain/perte correspondant à la matrice M :

	Pierre	Feuille	Ciseaux
Pierre	1/2	1	0
Feuille	0	1/2	1
Ciseaux	1	0	1/2

Le joueur ligne choisit une ligne i, tandis que le joueur colonne choisit une colonne j. Le but du joueur ligne est de minimiser ses pertes ($M(i,j)$).

Si les deux joueurs jouent au hasard suivant les distributions P pour le joueur ligne et Q pour le joueur colonne alors l'espérance de perte du joueur ligne est de :

$$\sum_{i,j} P(i)M(i,j)Q(j) = P^T MQ. \tag{4.34}$$

On note $M(P,Q)$ ce terme par la suite.

L'objectif du joueur ligne est de trouver une stratégie optimale P^* qui quelque soit la stratégie Q du joueur colonne, minimise ses pertes $M(P^*,Q)$. Tandis que l'objectif du joueur colonne est inverse, il cherche une stratégie Q^* qui maximise ses gains $M(P,Q^*)$ pour toutes les stratégies P du joueur ligne.

Ces observations sont une conséquence directe du théorème de von Neumann :

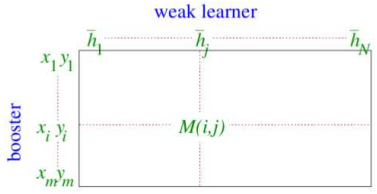
$$\min_P \max_Q M(P,Q) = \max_Q \min_P M(P,Q) = v, \tag{4.35}$$

où v est une constante appelée valeur du jeu.

4.3.2.2 Le jeu du Boosting

Revenons au cas du Boosting, il peut être vu comme un jeu synchrone répété à deux joueurs. Le joueur ligne correspond aux "booster" (l'algorithme de choix des exemples) tandis que le joueur colonne est l'algorithme de choix des classifieurs faibles. Chacun de ces deux joueurs va chercher à optimiser son choix (distributions des exemples pour l'un, classifieurs pour l'autre) pour maximiser leurs gains.

Si h_i sont les classifieurs faibles, on peut définir la matrice M du jeu du Boosting par :



Dans le cas d'AdaBoost cette matrice M peut s'exprimer par :

$$M(i,j) = \begin{cases} 1 & \text{si } y_i = h_j(x_i) \\ 0 & \text{sinon} \end{cases} .$$

On peut créer un algorithme de Boosting, généralisant AdaBoost reprenant le principe de la théorie des jeux :

Algorithm 2: Boosting selon la théorie des jeux

Input: une base d'exemples annotés $S = \{(x_1, y_1), \cdots, (x_m, y_m)\}$.
Input: une matrice de gain M.
Input: une base de classifieurs faibles $H = h_0, \cdots, h_N$.
begin
  for $t = 0 \dots T$ **do**
    Choisir la distribution D_t des exemples (x_i, y_i) selon une distribution P_t sur les lignes de M
    Choisir le classifieur faible h_t selon une distribution Q_t des colonnes de M, ce qui revient à maximiser le gain $M(P_t, Q_t)$

Le Boosting étant maintenant modélisé comme un jeu, la stratégie optimale des deux jeux correspond exactement à la stratégie énoncée par le théorème de von Neumann.

Ce qui nous donne :

$$\gamma^* = \min_d \max_{h \in H} \left(\sum_{n=1}^{m} d_n y_n h(x_n) \right) = \max_w \min_{1 \leq n \leq m} y_n \left(\sum_{t=1}^{T} w_t h_t(x_n) \right) = \rho^*. \tag{4.36}$$

On peut donc reformuler le Boosting comme deux problèmes duaux, soit on cherche à maximiser la marge ρ du classifieur :

$$\rho^* = \max_w \rho, \tag{4.37}$$

avec

$$\rho = \min_{1 \leq n \leq m} y_n \left(\sum_{t=1}^{T} w_t h_t(x_n) \right), \tag{4.38}$$

soit on cherche à minimiser γ dans l'espace dual :

$$\gamma^* = \min_d \gamma, \tag{4.39}$$

où

$$\gamma = \max_{h \in H} \left(\sum_{n=1}^{m} d_n y_n h(x_n) \right). \tag{4.40}$$

Ces deux approches conduisent à deux vues possibles du problème de Boosting et débouchent à des approches plutôt orientées classifieur ou plutôt orientées exemples.

4.3.3 Problème d'optimisation

Nous avons montré dans la partie précédente que le Boosting peut être vu comme la recherche d'une stratégie optimale pour la réalisation d'un jeu.

d_n et w_t sont des densités de probabilité on a donc $\forall i \quad d_i \geq 0$, $\sum_{i=1}^{m} d_i = 1$ et $\forall t \quad w_t \geq 0$, $\sum_{t=1}^{T} w_t = 1$. On peut ainsi reformuler les formules du Boosting issu de la théorie des jeux (4.38) et (4.39) sous la forme d'un problème de programmation linéraire (LP).

On obtient alors les formulations suivantes [Meir 2003] :

$$\max_{\rho,w} \rho$$
$$\text{sous contraintes de } \begin{cases} y_n \langle x_n, w \rangle \geq \rho \quad \forall n \in \{1, 2, ..., m\} \\ w_t \geq 0 \quad \forall t \\ \sum_{t=1}^{T} w_t = 1 \end{cases} \tag{4.41}$$

et

$$\min_{\gamma, 0 \leq d \in \mathbb{R}^N} \gamma$$
$$\text{sous contraintes de } \begin{cases} \sum_{n=1}^{m} d_n y_n h(x_n) \leq \gamma \quad \forall h \in H \\ d_n \geq 0 \quad \forall n \\ \sum_{n=1}^{m} d_n = 1 \end{cases} \tag{4.42}$$

De plus ces équations peuvent être généralisées à une norme quelconque comme pour l'équation (eq. 4.31) :

$$\max_{\rho,w} \rho$$
$$\text{sous contraintes de } \begin{cases} y_n \langle x_n, w \rangle \geq \rho \, \forall n \in \{1, 2, ..., m\} \\ \|w\|_p = 1 \end{cases} \tag{4.43}$$

Une des méthodes standard pour résoudre ce type de problème d'optimisation est l'utilisation d'une fonction barrière exponentielle $\beta \exp(-\frac{z}{\beta})$. Ce qui donne pour notre problème

$$F_\beta(w, \rho) = -\rho + \beta \sum_{n=1}^{m} \exp\left[\frac{1}{\beta}\left(\rho - \frac{y_n f_w(x_n)}{\sum_t w_t}\right)\right].$$

La résolution du problème consiste à présent, à minimiser cette fonction en respectant ρ et w à β fixés, puis à faire tendre β vers 0.

Dans le cas du Boosting et de sa variante AdaBoost$_\rho$, on choisit $\beta = (\sum_t w_t)^{-1}$. On cherche donc à minimiser la fonction :

$$-\rho + \left(\sum_t w_t\right)^{-1} \sum_{n=1}^{m} \exp\left[\rho \sum_t w_t - y_n f_w(w_n)\right]. \tag{4.44}$$

Cette méthode est développée dans la thèse de Gunnar Rätsch [Rätsch 2001a] et conduit à l'algorithme suivant :

Algorithm 3: AdaBoost$_\rho$ [Breiman 1999, Rätsch 2002]

Input: une base d'exemples annotés $S = \{(x_1, y_1), \cdots, (x_m, y_m)\}$.

Input: un nombre T d'itérations et une marge cible ρ.

begin

Initialisation de la distribution des exemples $d_{0,i} = \frac{1}{m}$.

for $t = 0 \dots T$ **do**

Entrainer un classifieur faible h_t suivant $\{S, d_t\}$

Calculer le poids du classifieur sélectionné :

$$w_t = \arg\min_{w \in \mathbb{R}} \sum_{i=1}^{m} d_{t,i} \exp(w(\rho - y_i h_t(x_i))). \tag{4.45}$$

Mettre à jour la distribution des exemples :

$$d_{t+1,i} = \frac{d_{t,i} \exp(-w_t y_i h_t(x_i))}{Z_t}. \tag{4.46}$$

Z_t est un facteur de normalisation

if $\alpha_t \leq 0$ *ou* $\alpha_t = +\infty$ **then**

return

Output: la somme pondérée des classifieurs sélectionnés :

$$H(x) = \sum_{t=0}^{T-1} w_t h_t(x). \tag{4.47}$$

Dans le cas où la marge cible ρ est nulle, on retrouve l'algorithme AdaBoost.

Cette approche s'intéresse donc plus particulièrement à la recherche d'un hyperplan séparateur dans l'espace des classifieurs faibles $\{h_t\}_t$ en cherchant à maximiser la marge de cet hyperplan avec les exemples d'apprentissage.

[Rätsch 2001b] propose de reprendre les problèmes d'optimisations (4.41) et (4.42) en relaxant la contrainte de marge et obtenir par la même une version à marge souple de l'algorithme. On obtient alors les formulations suivantes avec C une constante permettant de régler la relaxation :

$$\max_{\rho, w, \xi} \rho - C \sum_{n=1}^{m} \xi_n$$
$$\text{sous contraintes de } \begin{cases} y_n \langle x_n, w \rangle \geq \rho - \xi_n \quad \forall n \in \{1, 2, ..., m\} \\ w_t \geq 0 \quad \forall t \\ \xi_n \geq 0 \quad \forall n \\ \sum_{t=1}^{T} w_t = 1 \end{cases} \tag{4.48}$$

et dans le dual

$$\min_{\gamma, 0 \leq d \in \mathbb{R}^m} \gamma$$
$$\text{sous contraintes de } \begin{cases} \sum_{n=1}^{N} d_n y_n h(x_n) \leq \gamma \quad \forall h \in H \\ 0 \leq d_n \leq C \quad \forall n \\ \sum_{n=1}^{N} d_n = 1 \end{cases} \tag{4.49}$$

On en déduit donc un nouveau algorithme de Boosting résolvant ces problèmes d'optimisation :

4.3.4 Descente de gradient

On l'a vu dans la partie précédente, le Boosting correspond à une méthode de recherche d'extrema d'une fonction. On va montrer dans cette partie que la méthode utilisée pour atteindre cet extremum correspond à une descente de gradient.

Le Boosting peut donc être vu comme la recherche d'une fonction optimale selon un critère d'erreur sur la marge par une méthode de descente du gradient, cette idée fut proposée la première fois par Breiman [Breiman 1999].

Soit H un ensemble de classifieurs faibles, cet ensemble définit une base de l'espace des fonctions. Nous allons montrer dans cette partie que le Boosting peut être vu comme une méthode de descente de gradient pour une fonction de perte dans l'espace des fonctions.

Soit la fonction f_w combinaison de fonctions de base $\tilde{h}$:

$$f_w \in lin(H) = \left\{ \sum_{\tilde{h} \in H} w_{\tilde{h}} \tilde{h} \quad | \quad w_{\tilde{h}} \in \mathbb{R} \right\}. \tag{4.53}$$

On a pour objectif de trouver le point f_w, dans l'espace engendré par les fonctions de H, qui minimise une fonction de perte G à l'aide d'un ensemble d'apprentissage $S = (x_1, y_1) \cdots (x_m, y_m)$.

On suppose que notre fonction G est dérivable et de forme additive :

$$G(f, S) := \frac{1}{m} \sum_{i=1}^{m} g(f(x_i) y_i). \tag{4.54}$$

On veut résoudre le problème d'optimisation :

$$\min_{f_w \in lin(H)} G(f_w, S) = \min_{w} \left\{ \frac{1}{m} \sum_{i=1}^{m} g(f_w(x_i) y_i) \right\}. \tag{4.55}$$

La méthode employée pour résoudre ce problème est la méthode de Gauss-Southwell [Luenberger 2008] dont la convergence est prouvée par [Luo 1992].

Cette méthode consiste à modifier itérativement les coordonnées des points f_w^t jusqu'à converger vers le point minimisant la fonction. A chaque itération on change la valeur d'un seul axe par $f_w^{t+1} = f_w^t + \alpha_t h_t$. La figure (Fig. 4.4) illustre ce principe pour un cas à deux classifieurs faibles.

La méthode de résolution par descente du gradient propose de diminuer la fonction G dans la direction opposée du vecteur gradient, on cherche donc l'axe h_t qui maximise le produit scalaire $\langle -\nabla G, 1_{h_t} \rangle$,

Algorithm 4: LP-Boost [Rätsch 2001b, Demiriz 2002]

Input: une base d'exemples annotés $S = \{(x_1, y_1), \cdots, (x_m, y_m)\}$.

Input: un nombre T d'itérations.

begin

Initialisation de la distribution des exemples $d_{0,i} = \frac{1}{m}$.

Initialisation de $\gamma_0 = 0$

for $t = 0 \dots T$ **do**

Trouver le classifieur faible h_t qui maximise les performances de classification en fonction du poids de chacun exemple

$$\sum_{i=1}^{m} d_{t,i}\, y_i\, h(x_i). \tag{4.50}$$

if $\sum_{i=1}^{m} d_{t,i}\, y_i\, h_t(x_i) \leq \gamma_t$ **then**

return

else

Sauvegarder le classifieur faible h_t dans la liste des classifieurs sélectionnés.

Mise à jour des poids des exemples et de gamma :

$$(d_{t+1}, \gamma_{t+1}) = \underset{\gamma, 0 \leq d \in \mathbb{R}^N}{\arg\min}\ \gamma$$
$$\text{sous contraintes de } \begin{cases} \sum_{i=1}^{m} d_{t,i}\, y_i\, h_t(x_i) \leq \gamma \\ 0 \leq d_{t,i} \leq C \quad \forall i \\ \sum_{i=1}^{m} d_{t,i} = 1 \end{cases} \tag{4.51}$$

Output: la somme pondérée des classifieurs sélectionnés et des multiplicateurs de Lagrange w_t du dernier problème de programmation linéaire résolue :

$$H(x) = \sum_{t=0}^{T-1} w_t h_t(x). \tag{4.52}$$

où 1_{h_t} est un vecteur de $lin(H)$ qui vaut un pour l'axe h_t et zéro pour les autres axes. Le produit scalaire utilisé sur l'espace $lin(H)$ est celui défini comme étant le résultat de l'équation suivante pour tout $a \in lin(H)$ et $b \in lin(H)$:

$$\langle a, b \rangle = \frac{1}{m} \sum_{i=1}^{m} \sum_{\tilde{h} \in H} a(x_i) b(x_i) \tilde{h}(x_i). \tag{4.56}$$

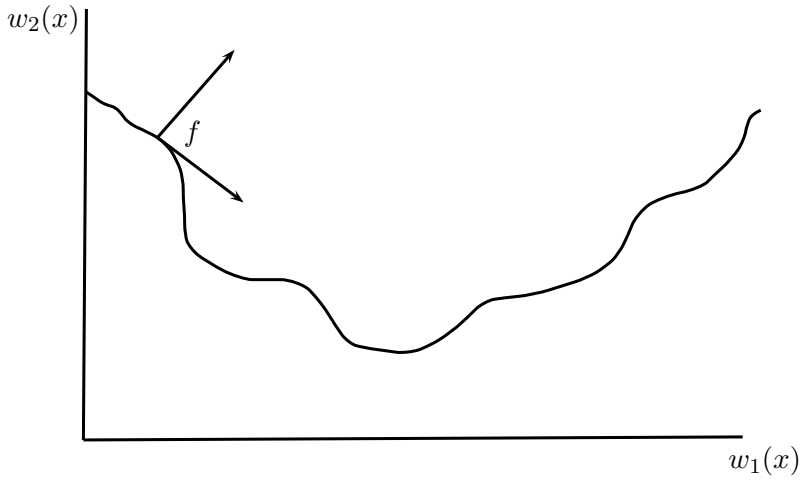


FIGURE 4.4 – Boosting vu comme une descente de gradient dans un exemple à deux classifieurs faibles. On cherche à atteindre les pondérations qui minimisent la fonction f.

Le choix de la direction opposée est motivé par l'approximation du premier ordre :

$$G(f + \varepsilon h) = G(f) + \varepsilon \langle \nabla G, 1_h \rangle. \tag{4.57}$$

On en déduit un premier algorithme :

Algorithm 5: AnyBoost [Mason 2000]

Input: une base d'exemples annotés $S = \{(x_1, y_1), \cdots, (x_m, y_m)\}$.
Input: une fonction de perte additive dérivable $G : \mathbb{R}^m \to \mathbb{R}$

$$\frac{1}{m} G(f, S) = \sum_{i=1}^{m} g(f(x_i) y_i).$$

Input: une base de classifieurs faibles $H = h_0, \cdots, h_N$.
begin
  $f_0 = 0$
  for $t = 0 \dots T$ **do**
    Choisir un classifieur h_t :

$$h_t = \arg\max_{h \in H} -\langle \nabla G, 1_h \rangle.$$

    if $-\langle \nabla G, 1_{h_t} \rangle \leq 0$ **then**
      return return f_t
    Choisir un poids α_t pour le classifieur h_t.
Output: la somme pondérée des classifieurs sélectionnés :

$$f(x) = \sum_{t=0}^{T-1} \alpha_t h_t(x).$$

Une fois l'axe choisi, on met à jour le vecteur :

$$w_{h_t}^{t+1} = w_{h_t}^t + \alpha_t. \tag{4.58}$$

Le gradient peut s'exprimer par :

$$\begin{aligned} \nabla G &= \left(\frac{\partial G(f_w^t, S)}{\partial w_{\tilde{h}}} \right)_{\tilde{h} \in H} && (4.59) \\ &= \left(\frac{1}{m} \sum_{i=1}^{m} g'(y_i f_w^t(x_i)) y_i \right)_{\tilde{h} \in H}. && (4.60) \end{aligned}$$

Le classifieur faible est donc choisi en fonction du critère :

$$\begin{aligned} h_t &= \arg\max_{\tilde{h} \in H} -\langle \nabla G, 1_{\tilde{h}} \rangle && (4.61) \\ &= \arg\max_{\tilde{h}} -\frac{1}{m^2} \sum_{i=1}^{m} g'(y_i f_w^t(x_i)) y_i \tilde{h}(x_i). && (4.62) \end{aligned}$$

On peut définir la pondération des exemples par $d_t(x_i) = \dfrac{-g'(y_i f_w^t(x_i))}{\sum_{i=0}^{N} -g'(y_i f_w^t(x_i))}$.

On cherche donc à chaque étape le classifieur h_t qui maximise $\sum_{i=0}^{m} d_t(x_i) h_t(x_i) y_i$.

On choisit α_t tel que :

$$\alpha_t = \arg\min_{\alpha \in \mathbb{R}} G(f_w^t + \alpha h_t, S). \tag{4.63}$$

On en déduit l'algorithme suivant :

Algorithm 6: MarginBoost [Mason 2000]

Input: une base d'exemples annotés $S = \{(x_1, y_1), \cdots, (x_m, y_m)\}$.

Input: une fonction de perte additive dérivable $G : \mathbb{R}^m \to \mathbb{R}$

$$G(f, S) = \sum_{i=1}^{n} g(f(x_i)y_i).$$

Input: des classifieurs faibles $H = h_0, \cdots, h_N$.

begin

Initialisation de la distribution des exemples, $f_0 = 0$ (on part de l'origine)

for $t = 0 \dots T$ **do**

Choisir un classifieur h_t :

$$h_t = \arg\max_{h \in H} \sum_{i=1}^{m} d_t(x_i)h(x_i)y_i.$$

Calculer le poids α_t du classifieur h_t :

$$\alpha_t = \arg\min_{\alpha} G(f_t + \alpha_t h_t).$$

Mettre à jour la distribution $d_{t+1,i}$ de chacun des exemples :

$$f_{t+1} = f_t + \alpha h_t$$

$$\forall i \ , \ d_{t+1}(x_i) = \frac{-g'(f_{t+1}(x_i)y_i)}{\sum_j -g'(f_{t+1}(x_j)y_j)}.$$

Output: la somme pondérée des classifieurs sélectionnés :

$$f(x) = \sum_{t=0}^{T-1} \alpha_t h_t(x) = \langle A, H(x) \rangle.$$

4.3.5 Analyse dans l'espace dual

Les parties précédentes ont montré comment AdaBoost résout le problème (4.38), en se consacrant à l'étude de la marge. Mais on peut également s'intéresser au problème dual (4.39) et à la stratégie de l'autre joueur dans le jeu du Boosting.

Pour ce problème, plaçons nous dans l'espace des $d_t(x_i)$. Un point d_t dans cet espace correspond à la distribution des exemples. On dispose d'hyperplans modélisant les classifieurs faibles, définis par les vecteurs $(h(x_1)y_1, ..., h(x_m)y_m)^T$.

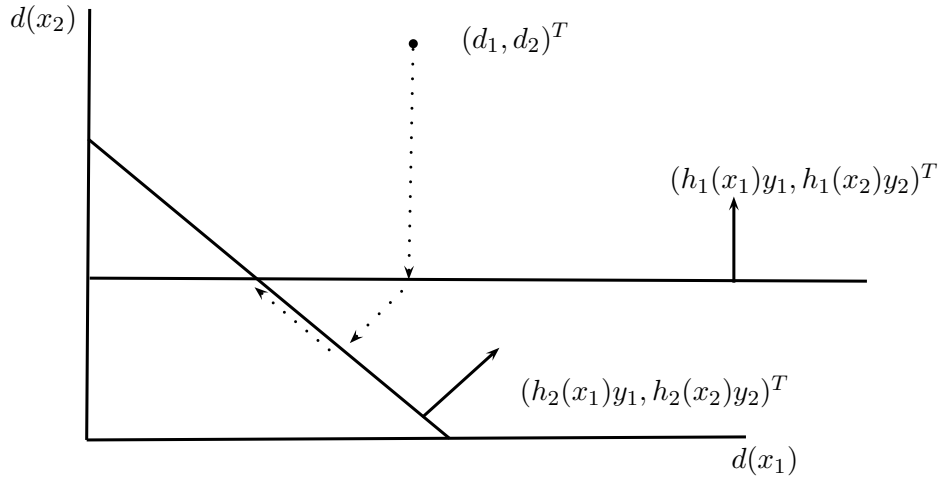


FIGURE 4.5 – Processus du Boosting vue dans le cas dual.

L'objectif du problème (4.39) est de faire converger le point d_t vers un point proche de tous les hyperplans et dans le cas optimal à l'intersection de tous ces hyperplans.

Pour cela on va projeter itérativement le point d_t sur chacun des hyperplans (Fig. 4.5) et on va chercher à minimiser une distance entre l'ancien point et le nouveau. On peut formuler cela à l'aide du problème suivant [Censor 1992] :

$$\begin{aligned} d_{t+1} &= \arg \min_{d \in \mathbb{R}^{m+}} \Delta_{\mathcal{G}}(d, d_t) \\ \text{avec } & \sum_{n=1}^{m} d_t(x_n) y_n h_t(x_n) = 0, \end{aligned} \tag{4.64}$$

où $\Delta_{\mathcal{G}}$ est une divergence.

Dans le cas du Boosting on utilise la divergence de Bregman [Kivinen 1999] :

$$\Delta_{\mathcal{G}}(x, y) := \mathcal{G}(x) - \mathcal{G}(y) - \langle \nabla \mathcal{G}(y), x - y \rangle, \tag{4.65}$$

avec $\mathcal{G}$ le générateur de la divergence $\Delta_{\mathcal{G}}$. $\mathcal{G}$ est une fonction strictement convexe, définie sur un ensemble convexe et différentiable sur son intérieur.

Et l'entropie relative dans le cas particulier d'AdaBoost : $\mathcal{G}(d) = \sum_n d_n \log d_n$.

L'un des problèmes de cette approche est la projection sur un hyperplan à chaque itération. Oza

propose de relaxer cette contrainte et introduit pour cela l'algorithme AveBoost [Oza 2003].

Algorithm 7: AveBoost [Oza 2003]

Input: une base d'exemples annotés $S = \{(x_1, y_1), \cdots, (x_m, y_m)\}$.
Input: Un nombre maximal d'itérations T.
Input: Une méthode d'apprentissage des classifieurs faibles L_b.
Input: des classifieurs faibles $H = h_0, \cdots, h_N$.
begin

Initialisation de la distribution des exemples, $\forall i, d_{1,i} = \frac{1}{m}$.

for $t = 1 \dots T$ **do**

Calculer un classifieur faible :

$$h_t = L_b(S, d_t). \tag{4.66}$$

Calculer l'erreur ε_t du classifieur faible h_t :

$$\varepsilon = \sum_{i:h_t(x_i) \neq y_i} d_{t,i} \tag{4.67}$$

if $\varepsilon \geq \frac{1}{2}$ **then**

$T = t - 1$.

return

On calcul la distribution orthogonale : **for** $i = 1 \dots m$ **do**

Calculer l'incrément sur les distributions des exemples :

$$c_{t,i} = d_{t,i} \begin{cases} \frac{1}{2(1-\varepsilon_t)} & \text{si } h_t(x_i) = y_i \\ \frac{1}{2\varepsilon_t} & \text{sinon} \end{cases}, \tag{4.68}$$

Mis à jour des distributions de chaque exemple :

$$d_{t+1,i} = \frac{t d_{t,i} + c_{t,i}}{t+1}. \tag{4.69}$$

Output: le classifieur final :

$$H(x) = \arg\max_y \sum_{t:h_t(x)=y} \log \frac{1-\varepsilon_t}{\varepsilon_t}. \tag{4.70}$$

4.3.6 Boosting et estimation de densité

Dans cette partie on va interpréter le Boosting sous l'angle de la régression logistique [Friedman 1998].

On cherche à estimer la probabilité d'un label y sachant un exemple x.

Soit $x = (x_1, \cdots, x_m)$ et y des variables aléatoires, comme on l'a vu précédemment, AdaBoost minimise le critère d'erreur $e^{-yf(x)}$. On cherche donc à minimiser :

$$E_{x,y}\left[e^{-yf(x)}\right] = E_x\left[\Pr[y=+1|x]e^{-f(x)} + \Pr[y=-1|x]e^{f(x)}\right]. \tag{4.71}$$

Le minimum de cette fonction, annule sa dérivée, on a donc :

$$\frac{\partial E_{x,y}\left[e^{-yf(x)}\right]}{\partial f(x)} = -\Pr[y=+1|x]e^{-f(x)} + \Pr[y=-1|x]e^{f(x)} = 0. \tag{4.72}$$

On en déduit que :

$$\Pr(y=1|x) = \frac{1}{1+e^{-2f(x)}}, \tag{4.73}$$

ainsi que :

$$f(x) = \frac{1}{2}\log\frac{\Pr(y=1|x)}{\Pr(y=-1|x)}. \tag{4.74}$$

A un facteur $\frac{1}{2}$ près on reconnaît un problème de régression logistique. En effet la régression logistique a pour but de trouver les fonctions h_t telles que :

$$\log\frac{\Pr(y=1|x)}{\Pr(y=-1|x)} = \sum_{t=1}^{T} h_t(x). \tag{4.75}$$

Elle a donc un objectif similaire au Boosting. Les outils de régression logistique conduisent donc à une nouvelle approche de Boosting possible.

La résolution d'un système logistique est généralement réalisée en maximisant la fonction de vraisemblance :

$$\begin{aligned}
L &= \prod_{i=1}^{m} \Pr(y=1|x_i)^{\frac{1+y_i}{2}} \Pr(y=-1|x_i)^{\frac{1-y_i}{2}} && (4.76)\\
&= \prod_{i=1}^{m} \left(\frac{1}{1+e^{-2f(x_i)}}\right)^{\frac{1+y_i}{2}} \left(\frac{1}{1+e^{2f(x_i)}}\right)^{\frac{1-y_i}{2}} && (4.77)\\
&= \prod_{i=1}^{m} \frac{1}{e^{-f(x_i)\frac{1+y_i}{2}}\left(e^{f(x_i)}+e^{-f(x_i)}\right)^{\frac{1+y_i}{2}}} \frac{1}{e^{f(x_i)\frac{1-y_i}{2}}\left(e^{f(x_i)}+e^{-f(x_i)}\right)^{\frac{1-y_i}{2}}} && (4.78)\\
&= \prod_{i=1}^{m} \frac{1}{e^{-f(x_i)y_i}\left(e^{f(x_i)}+e^{-f(x_i)}\right)} && (4.79)\\
&= \prod_{i=1}^{m} \frac{1}{e^{-f(x_i)y_i}\left(e^{f(x_i)y_i}+e^{-f(x_i)y_i}\right)} && (4.80)\\
&= \prod_{i=1}^{m} \frac{1}{1+e^{-2f(x_i)y_i}}. && (4.81)
\end{aligned}$$

En pratique, on prend le log de cette fonction, permettant ainsi de remplacer les produits par des sommes :

$$\log(L) = \sum_{i=1}^{m} -\log(1 + e^{-2f(x_i)y_i}). \tag{4.82}$$

On en déduit donc des algorithmes de Boosting basés sur une minimisation de l'erreur $err(f) = \sum_{i=1}^{m} \log(1 + e^{-2f(x_i)y_i})$ (Algorithme. 8).

Algorithm 8: LogitBoost [Friedman 1998]

Input: une base d'exemples annotés $S = \{(x_1, y_1), \cdots, (x_m, y_m)\}$.
Input: Un nombre maximal d'itérations T.
begin
Initialisation de la probabilité que les exemples est un label positif, $\forall i,\ \ p_0(x_i) = \frac{1}{m}$.
for $t = 1 \dots T$ **do**
Calculer les valeurs attendues du classifieur faible pour chaque exemple :

$$z_{t,i} = \frac{y_i - p_t(x_i)}{p_t(x_i)(1 - p_t(x_i))}. \tag{4.83}$$

Calculer la pondération des exemples :

$$d_{t,i} = p_t(x_i)\left(1 - p_t(x_i)\right). \tag{4.84}$$

Calculer le classifieur faible h_t par régression linéaire pondéré des $z_{t,i}$ pour les exemples x_i en utilisant les poids $d_{t,i}$.
Mettre à jour le classifieur fort :

$$H_{t+1}(x) = H_t(x) + \frac{1}{2}h_t(x). \tag{4.85}$$

Mettre à jour de la probabilité d'avoir un label positif pour chaque exemple :

$$p_{t+1}(x_i) = \frac{e^{H_{t+1}(x_i)}}{e^{H_{t+1}(x_i)} + e^{-H_{t+1}(x_i)}}. \tag{4.86}$$

Output: le classifieur final :

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} h_t(x)\right). \tag{4.87}$$

4.3.7 Conclusion

Nous avons vu, dans ce chapitre, différentes visions possibles du Boosting. Chacune de ces approches permet de créer de nouveaux algorithmes de Boosting répondant à une problématique précise. Ces dif-

férentes approches sont complémentaires et nous nous appuierons sur ces visions pour développer les algorithmes de Boosting de cette thèse.

4.4 Exemples de Boosting pour d'autres paradigmes d'apprentissage

L'algorithme qui a grandement popularisé le Boosting notamment dans les applications liées à des images est la version de Viola et Jones [Viola 2001]. Cette méthode utilise l'algorithme AdaBoost pour détecter des objets dans des images.

Nous allons présenter dans cette partie différentes applications du Boosting dans des contextes différents que ceux évoqués précédemment. Nous verrons par exemple, comment le Boosting peut être utilisé pour classer des images ou comment le Boosting peut s'insérer dans un processus itératif comme le suivi d'objet.

4.4.1 Boosting pour le classement

Le problème du classement s'inscrit dans un contexte différent de la détection d'objet tel que le présentent [Viola 2001]. L'objectif n'est plus d'attribuer un label à un document mais de classer les données étudiées les unes par rapport aux autres. On ne cherche donc pas à répondre à la question "ce document appartient-il à la classe ($H(x) \stackrel{?}{=} 1$)", on cherche seulement à savoir si ce document ressemble plus à la classe recherchée qu'une autre ($H(x_1) \stackrel{?}{>} H(x_2)$). Ce qui revient à transformer la fonction de décision par une fonction de rang :

$$\begin{array}{rccl} h: & I & \rightarrow & [0,1] \\ & \text{zone x} & \mapsto & \text{score de la zone x} \end{array}$$

Le but est de classer un ensemble de document (par exemples des images) les uns par rapport aux autres sans se soucier de la position de la frontière délimitant la classe. Cette approche est très utile pour rechercher un ensemble restreint de documents qui seront placés en tête de classement.

4.4.1.1 RankBoost

RankBoost est une méthode de Boosting introduite par Freund et Schapire [Freund 2003].

C'est un algorithme proche d'AdaBoost, mais adapté au problème de classement de données. Pour cela les classifieurs faibles ne sont plus des fonctions binaires mais ont besoin de renvoyer un score de classification pour chacune des images. On cherchera à sélectionner les classifieurs qui attribuent un meilleur score aux exemples positifs. On va donc s'intéresser à des couples d'exemples, un positif contre un négatif ($x' = (x_p, x_n)$ avec $x_p \in X_p$ et $x_n \in X_n$). L'idée est de choisir un classifieur qui place les données positives devant les données négatives (on cherche à avoir $h(x_p) > h(x_n)$). On peut se ramener à un classifieur à réponse binaire (ce positif est-il mieux classé que ce négatif ?) par :

$$h_2(x') = \begin{cases} 1 & \text{si } h(x_p) - h(x_n) > 0 \\ 0 & \text{sinon} \end{cases}. \tag{4.88}$$

On peut alors appliquer un AdaBoost classique.

Une optimisation de calcul est possible et permet de diminuer la complexité due à l'introduction des couples d'exemples. On peut en effet modifier les formules pour traiter chaque exemple indépendam-

ment, sans se ramener à des couples.

Algorithm 9: RankBoost [Freund 2003]

Input: une base d'exemples annotés $S = \{(x_1, y_1), \cdots, (x_m, y_m)\}$.
Input: des classifieurs faibles $h_0, \cdots, h_N$.
begin
Initialisation de la distribution des exemples $D_0(x_p, x_n) = \frac{1}{pn}$ avec $x_p \in X_p$, $x_n \in X_n$ et $p = |X_p|$, $n = |X_n|$.
for $t = 0 \dots T$ **do**
Trouver le classifieur h_t maximisant le score des classifieurs faible

$$r_{h_t} = \sum_{x_p, x_n} D_t(x_p, x_n)(h_t(x_p) - h_t(x_n)).$$

Calculer le poids du classifieur sélectionné :

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 + r_{h_t}}{1 - r_{h_t}} \right).$$

Mettre à jour la distribution des couples d'exemples :

$$D_{t+1}(x_p, x_n) = \frac{D_t(x_p, x_n) \exp(\alpha_t(h_t(x_n) - h_t(x_p)))}{Z_t}.$$

Z_t est un facteur de normalisation
Output: la somme pondérée des classifieurs sélectionnés :

$$H(x) = \sum_{t=0}^{T-1} \alpha_t h_t(x).$$

Optimisation de l'algorithme :

On peut définir un poids pour chacun des exemples indépendamment des couples positif/négatif :

$$\nu_0(x) = \begin{cases} \frac{1}{p} \text{ si } x \text{ est un exemple positif } (x \in X_p) \\ \frac{1}{n} \text{ si } x \text{ est un exemple négatif} (x \in X_n) \end{cases},$$

$$\nu_{t+1}(x) = \begin{cases} \frac{\nu_t(x) \exp(-\alpha_t h_t(x))}{Z_t^p} \text{ si x est un exemple positif } (x \in X_p) \\ \frac{\nu_t(x) \exp(\alpha_t h_t(x))}{Z_t^n} \text{ si x est un exemple négatif} (x \in X_n) \end{cases},$$

avec Z_t^p et Z_t^n des facteurs de normalisation :

$$Z_t^p = \sum_{x \in X_p} \nu_t(x) \exp(-\alpha_t h_t(x)), \tag{4.89}$$

$$Z_t^n = \sum_{x \in X_n} \nu_t(x) \exp(\alpha_t h_t(x)). \tag{4.90}$$

On remarque que l'on peut calculer D_{t+1} car :

$$D_{t+1}(x_p, x_n) = \frac{D_t(x_p, x_n) \exp(\alpha_t(h_t(x_n) - h_t(x_p)))}{Z_t} \tag{4.91}$$

$$= \frac{\nu_t \exp(-\alpha_t h_t(x_p))}{Z_t^p} \cdot \frac{\nu_t \exp(\alpha_t h_t(x_n))}{Z_t^n} \tag{4.92}$$

$$= \nu_{t+1}(x_p) \cdot \nu_{t+1}(x_n). \tag{4.93}$$

Si on calcule r_{h_t} on a :

$$r_{h_t} = \sum_{x_p, x_n} D(x_p, x_n)(h_t(x_p) - h_t(x_n)) \tag{4.94}$$

$$= \sum_{x_p \in X_p} \sum_{x_n \in X_n} \nu_t(x_p)\nu_t(x_n)(h_t(x_p) - h_t(x_n)) \tag{4.95}$$

$$= \sum_{x_p \in X_p} \left(\sum_{x_n \in X_n} \nu_t(x_n) \right) \nu_t(x_p) h_t(x_p) - \sum_{x_n \in X_n} \left(\sum_{x_p \in X_p} \nu_t(x_p) \right) \nu_t(x_n) h_t(x_n) \tag{4.96}$$

$$= \sum_{x_p \in X_p} \nu_t(x_p) h_t(x_p) - \sum_{x_n \in X_n} \nu_t(x_n) h_t(x_n) \text{ car les } \nu \text{ sont normalisés} \tag{4.97}$$

$$= \sum_{x \in X} s(x) \nu_t(x) h_t(x). \tag{4.98}$$

avec $s(x_i) = y_i = \begin{cases} 1 \text{ si } x_i \in X_p \\ -1 \text{ si } x_i \in X_n \end{cases}$.

Par cette astuce de calcul nous obtenons une complexité linéaire en fonction du nombre d'exemples pour le calcul de r_{h_t}, soit un passage de $O(np)$ à $O(n + p)$. On réduit également la consommation mémoire avec une seule valeur par exemple.

4.4.1.2 AdaRank

AdaRank [Xu 2007] est une méthode concurrente à RankBoost permettant également de classer des exemples entre eux. Contrairement à RankBoost, cette approche ne se base pas sur un ensemble d'exemples annotés mais sur des requêtes correspondant à un classement d'une partie des exemples.

On définit $Y = \{r_1, r_2, \cdots, r_\ell\}$, l'ensemble des rangs possibles pour une donnée, ℓ étant le nombre de rangs attribuables. L'ensemble d'entraînement S est constitué d'un ensemble de requêtes $Q = \{q_1, q_2, \cdots, q_m\}$. Pour chaque requête q_i il est associés une liste de documents $d_i = \{d_{i,1}, d_{i,2}, \cdots, d_{i,n(q_i)}\}$ et un rang pour chacun de ces documents $y_i = \{y_{i,1}, y_{i,2}, \cdots, y_{i,n(q_i)}\}$ avec $y_{i,j} \in Y$.

L'objectif est de construire une fonction f qui pour chaque document associe un score. On définit alors la fonction $\pi(q_i, d_i, f)$ qui réalise le classement des document d_i associé à la requête q_i à l'aide des scores renvoyés par la fonction f. On peut alors mesurer la qualité du classement proposé par la fonction f en comparant ce classement au classement proposé par l'utilisateur y_i. On utilise pour cela une fonction de mesure que l'on appellera de manière générique $E(\pi(q_i, d_i, f), y_i) \in [-1, 1]$ mais qui en pratique peut être une mesure de précision moyenne, de précision à rang n, NDCG, MRR (Mean Reciprocal Rank), WTA (Winners Take All)...

L'algorithme AdaRank est semblable à un algorithme AdaBoost où les exemples ont été remplacées par des requêtes. La fonction de mesure de l'erreur de classification d'AdaBoost est remplacée par la fonction $E(\pi(q_i, d_i, f), y_i)$. Nous donnons les détails de cet algorithme dans l'Algorithme. 10.

Algorithm 10: AdaRank [Xu 2007]

Input: Un ensemble de requêtes $S = \{(q_i, d_i, y_i)\}_{i=1}^{m}$,
Input: T un nombre maximal d'itérations,
Input: E une fonction mesurant la qualité de la fonction de classification.
begin

Initialisation de la distribution de chaque requête $P_1(i) = \frac{1}{m}$.

for $t = 1 \dots T$ **do**

Créer la fonction de classement h_t à l'aide de la distribution P_t et de l'ensemble d'apprentissage S.
Calculer le poids de la fonction de classement :

$$\alpha_t = \frac{1}{2} \ln \frac{\sum_{i=1}^{m} P_t(i)(1 + E(\pi(q_i, d_i, h_t), y_i))}{\sum_{i=1}^{m} P_t(i)(1 - E(\pi(q_i, d_i, h_t), y_i))}.$$

Mise à jour de la fonction de classification finale :

$$f_t(x) = \sum_{k=1}^{t} \alpha_k h_k(x).$$

Mise à jour de la distribution de chaque requête :

$$P_{t+1}(i) = \frac{\exp(-E(\pi(q_i, d_i, f_t), y_i))}{\sum_{j=1}^{m} \exp(-E(\pi(q_i, d_i, f_t), y_i))}.$$

Output: la somme pondérée des classifieurs sélectionnés :

$$f(x) = \sum_{k=1}^{T} \alpha_k h_k(x).$$

4.4.2 Algorithme pour l'apprentissage itératif

Le OnlineBoost est une méthode de Boosting introduite par Oza [Oza 2001] et reprise par Grabner [Grabner 2006]. Ce sont des approches basées exemples travaillant dans le dual comme les méthodes présentées dans la section 4.3.5.

Contrairement aux approches traditionnelles qui nécessitent de parcourir tous les exemples d'apprentissage sans possibilité d'ajout de nouveau, une fois l'approche terminée, l'approche online cherche à optimiser un classifieur fort par des mises à jour successives. Cette méthode offre la possibilité de faire une recherche et une mise à jour en temps réel, permettant notamment le suivi d'objet image par image.

La méthode présentée ici se différencie des méthodes *pseudo-online*, elle cherche à éviter de relancer tout le processus d'apprentissage à chaque ajout d'un nouvel exemple. Elle consiste à mettre à jour les classifieurs et leurs sélections.

Cette approche est utilisée dans le domaine du suivi d'objet car elle permet de prendre en compte des changements et des mises à jour.

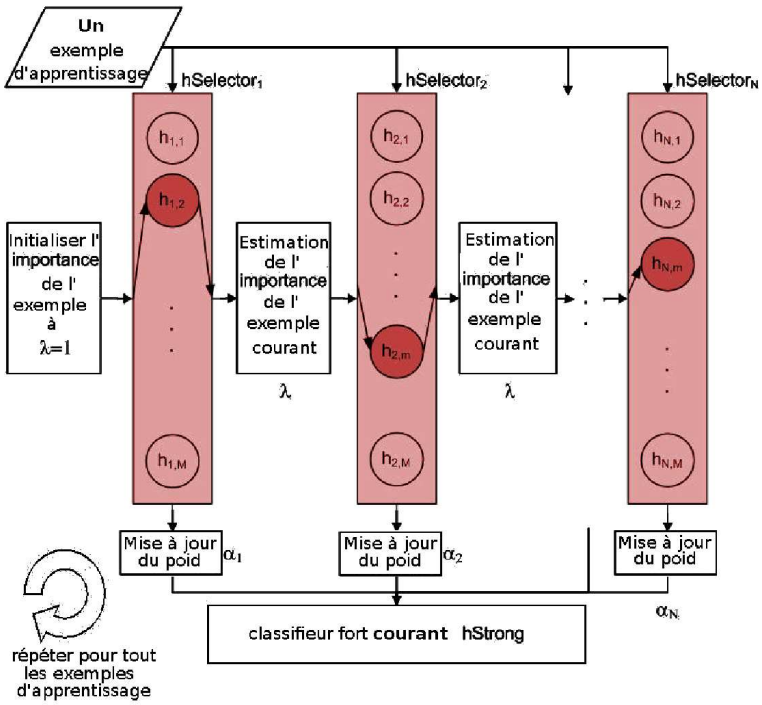


FIGURE 4.6 – OnlineBoost [Grabner 2006]

L'approche de Grabner [Grabner 2006] optimise le processus de mise à jour en introduisant un concept d'ensemble de classifieurs faibles. On cherche, dans cette méthode, à optimiser des ensembles de classifieurs faibles dans lesquels on va effectuer la sélection à chaque nouvel exemple. Ceci permet de limiter le nombre de classifieurs faibles et ainsi diminuer le temps d'apprentissage.

On peut résumer l'algorithme comme suit : à chaque exemple on sélectionne un classifieur faible dans chacun des M ensembles de classifieurs.

Le critère de choix est une approximation de celui utilisé dans AdaBoost. L'idée est de s'en rapprocher de plus en plus à chaque ajout d'un nouvel exemple. On utilise pour cela des variables

$\lambda_{correct}, \lambda_{wrong}$ pour estimer l'erreur du classifieur faible étudié et une variable λ pour prendre en compte la difficulté de l'exemple en cours d'analyse.

4.4.3 Conclusion

Nous avons présenté dans cette section deux utilisations du Boosting dans des contextes différents que celui de la catégorisation.

Nous allons utiliser ces deux approches pour construire une nouvelle méthode de Boosting adaptée au contexte de la recherche interactive. Cette méthode permettra de produire un classement des images en suivant un protocole itératif faisant intervenir l'utilisateur.

Algorithm 11: OnlineBoost [Grabner 2006]

Input: un exemple d'apprentissage annoté x.

Input: des classifieurs faibles $h_0, \cdots, h_{nbH}$ répartis en T ensembles $E_0, \cdots, E_M$.

Input: des $\lambda_{0,m}^{correct}, \lambda_{0,m}^{wrong}$ (initialisés la première fois à 1).

begin

for $t = 1 \dots T$ **do**

forall the *classifieurs de l'ensemble* E_t **do**

Estimation de l'erreur : **if** *le classifieur classe correctement l'exemple* **then**

$$\lambda_{t,m}^{correct} += \lambda.$$

else

$$\lambda_{t,m}^{wrong} += \lambda.$$

$$\varepsilon_{t,m} = \frac{\lambda_{t,m}^{wrong}}{\lambda_{t,m}^{correct} + \lambda_{t,m}^{wrong}}.$$

Trouver le classifieur minimisant l'erreur de classification :

$$\varepsilon_t = \arg\min_m \varepsilon_{t,m}.$$

Choisir

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \varepsilon_t}{\varepsilon_t} \right).$$

Mettre à jour λ :

if *le classifieur minimisant l'erreur classe correctement l'exemple d'apprentissage* **then**

$$\lambda = \lambda \cdot \frac{1}{2(1 - \varepsilon_t)}.$$

else

$$\lambda = \lambda \cdot \frac{1}{2\varepsilon_t}.$$

Output: la somme pondérée des classifieurs sélectionnés :

$$H(x) = sign \left(\sum_{t=1}^{T} \alpha_t h_t(x) \right).$$

CHAPITRE 5

Boosting interactif

Ces travaux ont été publiés dans [Lechervy 2010a] et [Lechervy 2010b].

Sommaire

Nous avons présenté dans le chapitre précédent le cadre théorique des méthodes de Boosting ainsi que certains des algorithmes qui en découlent. Ces méthodes sont généralement peu contraignantes à paramétrer et sont de plus rapide pour évaluer le score de nouveaux exemples. A contrario, comme le montrent les équations (Eq. 4.2) et (Eq. 4.23) pour AdaBoost, il est nécessaire de disposer de nombreux exemples d'apprentissage pour avoir une bonne estimation de l'erreur des classifieurs faibles et il est important de sélectionner beaucoup de ces derniers pour diminuer significativement l'erreur d'apprentissage. Ces méthodes ne sont donc pas, à l'origine, adaptées à la recherche interactive (ce type de recherche est présenté dans le chapitre 3) et ce domaine d'application a été peu exploré. Nous présentons dans les travaux qui suivent une nouvelle méthode de Boosting qui répondra aux contraintes spécifiques de la recherche interactive. L'objectif est d'obtenir des résultats comparables aux performances actuelles, comme celles par SVM, mais en utilisant une approche de Boosting. Par ailleurs cette nouvelle méthode devra être rapide afin de permettre une interaction facile avec un utilisateur humain. Enfin, elle devra se baser sur un ensemble d'apprentissage restreint et évolutif.

Plusieurs travaux préliminaires ont été réalisés par le passé se rapprochant de la recherche interactive. Des méthodes ont par exemple été proposées pour adapter le Boosting dans les cas où le nombre d'exemples d'apprentissage est réduit [Diao 2002, Wolf 2005]. On notera, par exemple, la méthode de [Li 2004] qui propose d'alimenter petit à petit un algorithme de type AdaBoost à l'aide d'exemples proches de la frontière de décision. Ce principe d'annotation de nouvelles images en fonction de leur incertitude est repris dans [Collins 2008]. Les auteurs de cet article proposent également de réduire la famille des classifieurs faibles utilisés à celle engendrée par les annotations. Des méthodes de Boosting ont été également développées pour introduire l'utilisateur dans le processus itératif [Lu 2007].

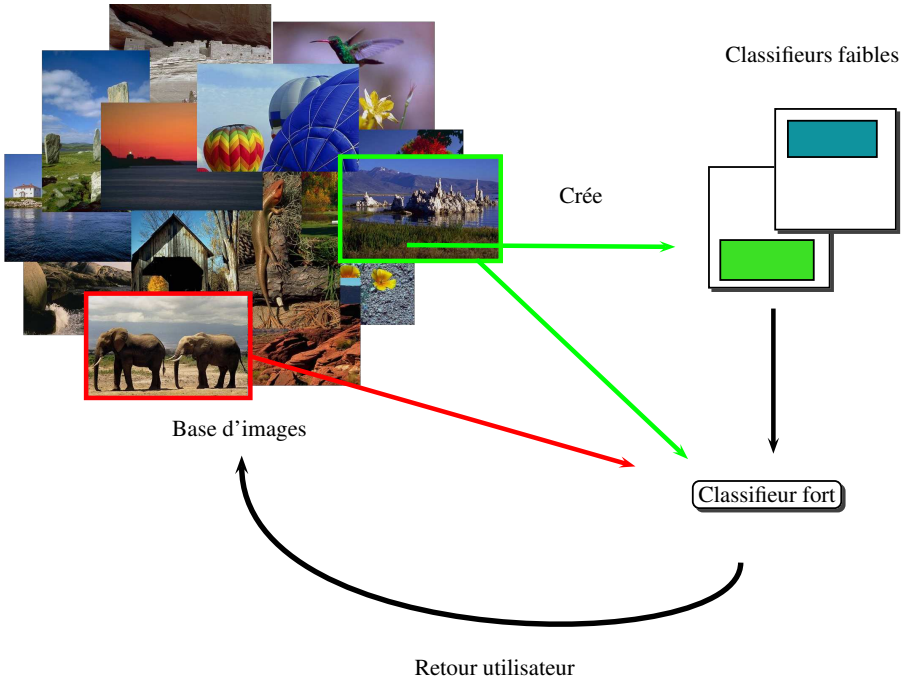


FIGURE 5.1 – Principe de l'algorithme

Une autre famille de méthodes de Boosting a été développée pour le suivi d'objet [Grabner 2006, Thuy 2008]. Ces méthodes ne sont pas entièrement étrangères aux méthodes interactives. Dans les deux cas, le système fonctionne avec un ensemble d'images d'apprentissage restreint et de nouveaux exemples sont introduits progressivement lors des étapes de mise à jour du système. Ces méthodes pour le suivi d'objet s'appuient sur les travaux de [Oza 2001] qui proposent un Boosting en ligne se mettant à jour en fonction des nouvelles annotations introduites dans le système. On peut remarquer dans [Grabner 2006] l'utilisation intéressante de familles dynamiques de classifieurs faibles qui évoluent en fonction du flux d'exemples rencontrés. Cette méthode est présentée dans la section 4.4.2 de ce manuscrit.

5.1 Méthode proposée

La méthode que nous proposons est une méthode de Boosting interactif. Elle repose sur une interaction entre l'utilisateur et le système pour la construction d'un ensemble de classifieurs faibles adaptés à la recherche de l'utilisateur. L'idée que nous introduisons par notre méthode est de construire itérativement l'ensemble des classifieurs faibles à partir des exemples positifs de l'utilisateur. Le protocole interactif alors mis en place dans le cas de notre algorithme est illustré par la figure (Fig. 5.1).

L'algorithme débute avec un ensemble vide W_0 de classifieurs et à chaque ajout d'exemples positifs, nous construisons de nouveaux classifieurs faibles produits à partir des caractéristiques visuelles de l'image annotée positivement par l'utilisateur. Par exemple, si l'image représente un paysage de plage par temps clair, on peut construire un classifieur qui recherche la couleur bleue dans la partie supérieure de l'image pour détecter le ciel bleu, un autre qui recherche le jaune au bas de l'image pour retrouver la plage... Nous aborderons dans la suite plus en détail la manière de construire ces classifieurs faibles.

Algorithm 12: RankBoost interactif après j itérations.

Input: un exemple d'apprentissage $x_i \in X_p \cup X_n$.

Input: un ensemble de classifieurs faibles W_{j-1} initialisés à $\varnothing$.

Initialisation :

begin

Initialisation de la distribution des exemples

$$\nu_0(x_i) = \begin{cases} \dfrac{1}{|X_p|} \text{ si } x_i \text{ est un exemple positif } (x_i \in X_p) \\ \\ \dfrac{1}{|X_n|} \text{ si } x_i \text{ est un exemple négatif} (x_i \in X_n) \end{cases} . \tag{5.1}$$

if $x_i \in X_p$ **then**

$$W_j = W_{j-1} \cup \mathcal{H}_{x_i}. \tag{5.2}$$

else

$$W_j = W_{j-1}. \tag{5.3}$$

Algorithme :

begin

for $t = 0$ *jusqu'à* T **do**

Trouver le classifieur $h_t \in W_j$ maximisant le score des classifieurs faibles :

$$r_t(h) = \sum_{x_p \in X_p} \nu_t(x_p) h(x_p) - \sum_{x_n \in X_n} \nu_t(x_n) h(x_n). \tag{5.4}$$

Calculer le poids du classifieur sélectionné :

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 + r_t}{1 - r_t} \right). \tag{5.5}$$

Mettre à jour la distribution des exemples :

$$\nu_{t+1}(x_i) = \begin{cases} \dfrac{\nu_t(x_i) \mathrm{e}^{-\alpha_t h_t(x_i)}}{\sum_{x_p \in X_p} \nu_t(x_p) \mathrm{e}^{-\alpha_t h_t(x_p)}} \text{ si } x_i \in X_p \\ \\ \dfrac{\nu_t(x_i) \mathrm{e}^{\alpha_t h_t(x_i)}}{\sum_{x_n \in X_n} \nu_t(x_n) \mathrm{e}^{\alpha_t h_t(x_n)}} \text{ si } x_i \in X_n \end{cases} . \tag{5.6}$$

Output: la somme pondérée des classifieurs sélectionnés :

$$H(x) = \sum_{t=0}^{T-1} \alpha_t h_t(x). \tag{5.7}$$

On fait ainsi évoluer l'ensemble W_j des classifieurs faibles à chaque ajout d'exemples positifs selon la règle suivante :

$$\begin{cases} W_0 & = \varnothing \\ W_{j+1} & = W_j \cup \mathcal{H}_{x_i} \end{cases}. \tag{5.8}$$

Avec $\mathcal{H}_{x_i}$ l'ensemble des classifieurs faibles générés par l'image x_i.

$$\mathcal{H}_{x_i} = \{h_{k,x_i}\}_k. \tag{5.9}$$

L'utilisation d'un tel ensemble évolutif de classifieurs faibles permet de gérer le faible nombre d'annotations. On s'assure ainsi que les classifieurs qui seront sélectionnés ont une pertinence vis-à-vis des images à rechercher. Compte tenu du faible nombre d'annotations, si les classifieurs faibles étaient construits aléatoirement, la sélection des classifieurs faibles le serait également. En effet, lors des itérations du Boosting, la sélection des classifieurs faibles n'est pas pertinente faute d'images pour obtenir une bonne estimation. Pour plus de détail, on pourra se référer à l'équation (Eq. 4.2) de la section 4.1.3.

Une fois l'ensemble des classifieurs faibles sélectionnables mis à jour, on lance un algorithme de RankBoost [Freund 2003] sur l'ensemble d'apprentissage formé par les annotations successives X_j de l'utilisateur et l'ensemble évolutif W_j de classifieurs faibles. Pour plus de détail sur l'algorithme RankBoost, on pourra lire la section 4.4.1.1.

L'**Algorithme 12** présente plus en détail une itération de notre algorithme.

Le nombre de classifieurs à étudier et ainsi le temps dédié à l'apprentissage est fortement réduit grâce à l'usage uniquement de classifieurs restreint aux données contenues dans les images d'apprentissage.

Le choix des exemples d'apprentissage influe donc directement sur l'évolution du classifieur fort, puisqu'il conditionne la présence des classifieurs faibles dans l'ensemble des classifieurs sélectionnables. On cherche donc à annoter les exemples les plus pertinents afin de faire évoluer le système le plus rapidement possible. Nous montrons dans la section 5.3 comment, grâce à une stratégie de sélection active,le système sélectionne les images les plus intéressantes pour l'annotation.

5.2 Les Classifieurs faibles

On a introduit précédemment un ensemble de classifieurs faibles $\mathcal{H}_{x_i}$ pour chaque image. Nous proposons dans cette partie deux types de classifieurs faibles possibles. L'ensemble $\mathcal{H}_{x_i}$ peut être découpé selon ces deux types de classifieurs (un spatial et l'un non spatial).

$$\mathcal{H}_{x_i} = \mathcal{H}^1_{x_i} \cup \mathcal{H}^2_{x_i}. \tag{5.10}$$

5.2.1 Extraction de caractéristiques de l'image

Chaque image est découpée en neuf zones z_l selon une grille 3x3. Dans chacune de ces zones on calcule un histogramme d'une propriété (ex : un histogramme de couleur). Nous considérons aussi les régions $\rho_m = \bigcup_l z_l$ correspondant à des combinaisons de zones z_l non nécessairement connexe. Le calcul de l'histogramme de la région correspond alors à la combinaison des histogrammes de chacune des zones la composant. On définit également Z, l'ensemble des zones et R l'ensemble des régions possibles.

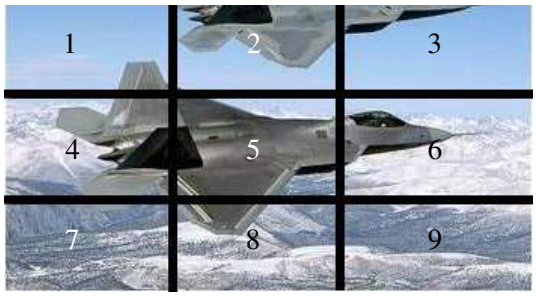


FIGURE 5.2 – Exemples de régions pour un classifieur faible. L'image est découpée en neuf zones et la combinaison des zones 2,5 et 7 forme une région.

La figure (Fig. 5.2) illustre ces notions par un exemple de découpage possible. Chaque case numérotée correspond à une zone, tandis que la combinaison des zones 2,5,7 forme une région ($\rho_{257} = z_2 \cup z_5 \cup z_7$).

5.2.2 Les classifieurs de type 1

$\mathcal{H}^1_{x_i}$ est l'ensemble de tous les classifieurs de type 1 que l'on peut générer avec l'image x_i. Ce classifieur contient une information spatial.

$$\mathcal{H}^1_{x_i} = \{h^1_{k,x_i}\}_k. \tag{5.11}$$

Chaque classifieur de type 1 est composé d'une région ρ_k et d'une image génératrice x_i.
Le classifieur correspond à la fonction de classement :

$$h^1_{k,x_i} : x_j \longmapsto 1 - d(histo_{\rho_k}(x_i), histo_{\rho_k}(x_j)). \tag{5.12}$$

Il effectue la comparaison entre l'histogramme de la région ρ_k dans l'image de référence x_i et dans l'image à tester x_j.
On utilise pour cela une distance du χ^1 définit par :

$$d(g, x) = \frac{1}{M} \sum_{p=0}^{M} \frac{|g_p - x_p|}{g_p + x_p}. \tag{5.13}$$

M correspond à la taille des histogrammes g et x. g_p et x_p correspondent quant à eux, à chaque élément de ces histogrammes.
On peut remarquer que l'image de référence a un score maximal pour tous les classifieurs qu'elle génère.

5.2.3 Les classifieurs de type 2

$\mathcal{H}^2_{x_i}$ est l'ensemble de tous les classifieurs de type 2 que l'on peut générer avec l'image x_i. Ces descripteurs sont non-spatial.

$$\mathcal{H}^2_{x_i} = \{h^2_{k,x_i}\}_k. \tag{5.14}$$

Un classifieur de type 2 est composé uniquement des données d'une zone d'une image de référence. Il correspond à la fonction de classement :

$$h^2_{k,x_i}(x_j) = 1 - \min_{z_{k'} \in Z} d(histo_{z_k}(x_i), histo_{z_{k'}}(x_j)). \tag{5.15}$$

Ces classifieurs comparent chaque zone de l'image de test x_j à une zone donnée z_k de l'image de référence x_i.

La distance d utilisée est la même que précédemment (Eq. 5.13).

5.3 Sélection active des exemples d'apprentissage

L'utilisation d'un protocole interactif est souvent couplé à des méthodes d'apprentissage actif, ces méthodes sont détaillées de manière plus approfondie en section 3.3.

Nous proposons, dans cette section, une méthode active adaptée à l'algorithme que nous venons de présenter. Notre approche s'appuie sur le principe de construction itérative de l'ensemble des classifieurs faibles. L'idée proposée consiste à prendre les exemples d'apprentissage non-annotés et à suggérer les exemples qui, s'ils étaient annotés positivement, apporteraient les classifieurs faibles qui classeraient au mieux les exemples connus du système. C'est à dire les images qui produisent les classifieurs faibles maximisant le critère de sélection r_t de l'algorithme (Eq. 5.4) dans le cas où les exemples ont le même poids ($t = 0$). Notre approche s'intègre dans les approches actives dites *optimistes* (voir section 3.3 pour plus d'information sur ces approches).

On cherche ainsi les images $x_{i^*} \notin X_p \cup X_n$ correspondant à :

$$i^* = \arg\max_i \left(\max_{h \in \mathcal{H}_{x_i}} r_0(h) \right). \tag{5.16}$$

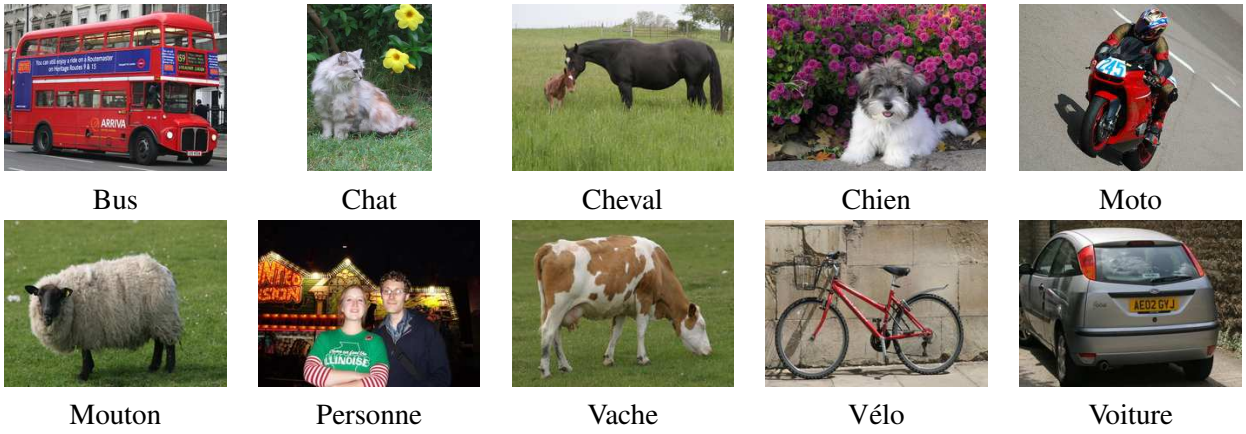


FIGURE 5.3 – Les dix catégories de VOC 2006

	train		val		trainval		test	
	img	**obj**	**img**	**obj**	**img**	**obj**	**img**	**obj**
Bus	93	118	81	117	174	235	180	233
Chat	192	214	194	215	386	429	388	429
Cheval	129	164	118	162	247	326	254	324
Chien	189	211	176	211	365	422	370	423
Moto	118	138	117	137	235	275	234	274
Mouton	119	211	132	210	251	421	238	422
Personne	319	577	347	579	666	1156	675	1153
Vache	102	156	104	157	206	313	197	315
Vélo	127	161	143	162	270	323	268	326
Voiture	271	427	282	427	553	854	544	854
Tous	1277	2377	1341	2377	2618	4754	2686	4753

FIGURE 5.4 – Statistique de répartition des images et des objets dans VOC 2006

5.4 Expériences

5.4.1 Présentation de la base d'images

La base VOC2006 [Everingham 2006] a été mise en place lors de la compétition PASCAL en 2006. Chaque année depuis 2005, une nouvelle base de difficulté croissante est proposée aux participants. La version 2006 contient 5304 images réparties en 10 catégories (Fig. 5.4.1). Les images proviennent principalement des sites web de Microsoft Research Cambridge et Flickr. Les objets recherchés sont principalement centrés et représentent généralement une bonne partie de l'image. La base comporte 9507 annotations, certaines images contiennent des objets appartenant à plusieurs catégories différentes et chaque catégorie est inégalement représentée.

Les images sont réparties selon trois ensembles dissociés : train, val et test (Fig. 5.4.1). Dans nos expériences, nous fusionnerons les ensembles train et val dans un seul ensemble d'apprentissage.

	Vélo	Bus	Voiture	Chat	Vache	Chien	Cheval	Moto	Personne	Mouton	Tous
Boost Lab	29,6	37,4	55,4	25,5	36,5	21,0	14,0	31,2	36,4	44,1	33,1
Boost Qw	37,6	38,1	**68,5**	27,1	24,1	21,2	18,8	40,6	36,9	23,9	33,7
Boost Lab|Qw	**44,3**	**44,9**	67,9	**30,0**	**39,5**	**24,7**	**19,0**	**42,1**	**38,3**	**52,8**	**40,3**
SVM Lab	19	22,1	39,4	22,6	29,2	17,1	12,4	19,6	32,2	44,8	25,8
SVM Qw	20,7	24,3	50,7	19,5	12,4	17,6	13,1	25,6	29,8	24,6	23,8
SVM Lab|Qw	34,1	40	57	25,7	31,8	20,4	15,7	34,6	34,5	46,4	34

FIGURE 5.5 – Précision moyenne en % sur VOC 2006.

	Vélo	Bus	Voiture	Chat	Vache	Chien	Cheval	Moto	Personne	Mouton	Tous
Boost Lab|Qw	71.1	81.3	**87.3**	**71.6**	**76.8**	**64**	**61.3**	74	**61.3**	**81.5**	**73 $\pm$ 8.5**
SVM Lab	62.6	72.4	69	61.5	71.6	55.5	56.2	68.3	58.6	79.2	65.5 $\pm$ 7.4
SVM Qw	64.7	75.9	76.6	57.9	60.1	57.2	57.7	70.7	55.5	70.9	64.7 $\pm$ 7.7
SVM Lab|Qw	**73.9**	**81.6**	80.3	64.8	72.6	60.4	61	**76.1**	60.1	80.5	71.1 $\pm$ 8.3

FIGURE 5.6 – Aire sous la courbe ROC en % sur VOC 2006.

Afin de décrire les images de cette base, nous utiliserons principalement deux types de descripteurs avec plus ou moins de dimensions :

- Des histogrammes de couleurs CIE Lab (Lab) ;
- Des histogrammes d'ondelettes quaternioniques (Qw) [Chan 2004].

5.4.2 Protocol expérimental

Nous utiliserons dans ces expériences un protocole différent du protocole officiel de la campagne VOC Challenge. En effet, contrairement aux applications classiques sur cette base, nous voulons mettre en avant l'avantage de notre méthode dans un contexte de recherche interactive. Nous utiliserons donc un protocole adapté à ce type de recherche qui donne une estimation des performances auxquelles un utilisateur peut s'attendre en démarrant une recherche à partir d'une image quelconque de la catégorie qu'il recherche.

Cette estimation est calculée en simulant un grand nombre de sessions de recherche, chacune étant initialisée à l'aide d'une image prise au hasard dans la catégorie. A chaque bouclage de pertinence, le système annote les images de la base d'entraînement qui ont été sélectionnées par la méthode active, lance la mise à jour de la classification de la base de test, et mesure la qualité du résultat sur cette dernière. Pour chaque session simulée, 5 images ont été annotées par bouclage de pertinence, et 10 boucles ont été effectuées. A l'issue de chacune des sessions simulées, un total de 51 annotations a été fourni au système (Fig. 3.2).

Le critère de comparaison utilisé est la précision moyenne de TrecVid (AP). Elle correspond à l'aire sous la courbe précision/rappel et donne un aperçu de la qualité des premiers résultats. La précision mesure le nombre d'images pertinentes retournées par le système de recherche par rapport au nombre d'images de la base étudiée. Tandis que le rappel mesure le nombre d'images pertinentes retournées par rapport au nombre d'images pertinentes de la base.

5.4.3 Résultats

Nous avons appliqué le protocole présenté précédemment sur la base VOC 2006. Nous avons comparé notre approche à un SVM interactif [Gosselin 2008]. Les expériences réalisées (Fig. 5.7, 5.5 et 5.6) montrent que notre algorithme permet un gain de performance.

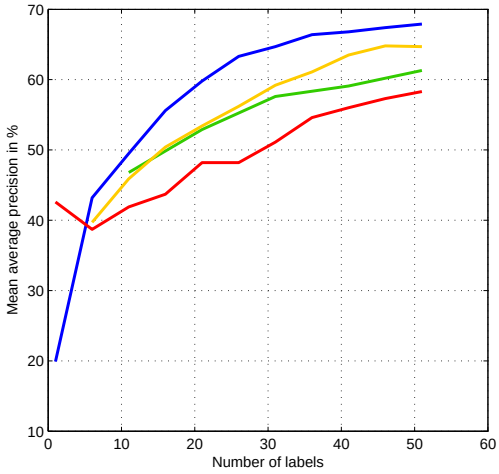
On peut tout d'abord remarquer que la combinaison de différents types d'attributs aussi bien en SVM que pour l'algorithme proposé permet une amélioration des performances (Fig. 5.5, 5.6). Notre méthode permet également l'utilisation de plusieurs descripteurs hétérogènes.

Ces expériences montrent également l'utilité d'une adaptation interactive des algorithmes. Que ce soit pour le SVM ou pour notre méthode, la version interactive fournit de bien meilleurs résultats que la version classique de ces méthodes. La prise en compte de l'interaction entre le système et l'utilisateur est donc bénéfique et permet un gain significatif.

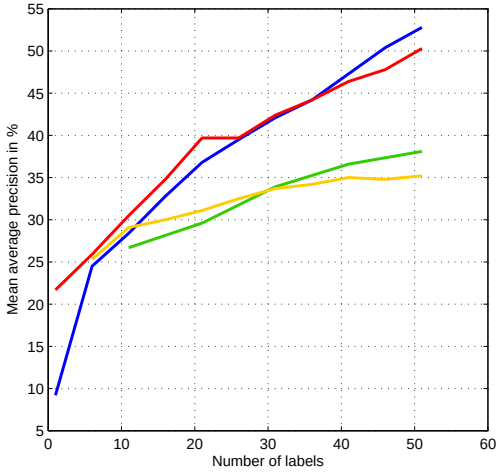
En outre on peut voir à travers ces expériences l'apport de notre méthode par rapport aux méthodes plus classiques basées SVM. Notre approche fournit de meilleurs résultats et propose donc une alternative aux approches par SVM en utilisant un modèle qui, à notre connaissance, est peu utilisé pour faire de l'interaction avec l'utilisateur.

On peut néanmoins constater que les résultats varient en fonction de la nature de la classe recherchée. Les voitures et les moutons ont de bien meilleures performances de classification que les chevaux ou les chiens. Ceci peut être expliqué par la nature des images de la base. La classe *mouton* contient principalement des images ayant une grande quantité de vert et de blanc, les voitures, quant à elles, sont essentiellement composées d'images à fort gradient dans des paysages urbains. A contrario, les chevaux et les chiens sont des classes très variées, situées aussi bien en intérieur qu'à l'extérieur, en villes ou à la campagne... Ce sont des images très différentes qui ont donc peu d'attributs communs. Ces classes sont donc plus difficilement cernables avec peu d'exemples.

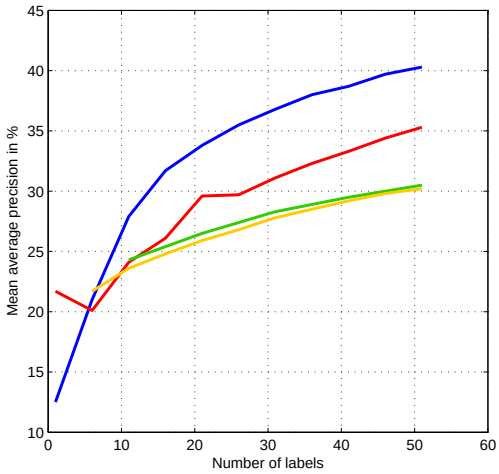
On peut également remarquer que pour très peu d'images le SVM est meilleur. L'hyperplan choisi par le SVM est meilleur que des classifieurs faibles appris sur un ou deux exemples.



Voiture



Mouton



Toutes les catégories

FIGURE 5.7 – Précision moyenne en % sur VOC 2006. La courbe bleue correspond à notre algorithme, la rouge à une méthode de SVM active, la jaune au RankBoost, tandis que la verte correspond aux SVM sans méthode active

5.5 Conclusion

Nous avons présenté dans ce chapitre une nouvelle méthode de Boosting pour la recherche interactive. Pour cela nous avons adapté l'algorithme RankBoost, nous avons également proposé des classifieurs spécifiques pour ce type de recherche et nous avons mis en place une méthode active.

Notre méthode permet de traiter les problèmes liés au contexte d'apprentissage interactif, dont le faible nombre d'exemples. De plus, elle assure une mise à jour très rapide de la classification, quasi imperceptible pour l'utilisateur sur une base de 5 000 images.

Les résultats expérimentaux ont montré la pertinence de notre approche notamment en la comparant avec les méthodes de recherche interactive classiques à base de SVM.

Troisième partie

Boosting pour la construction de noyaux

De nombreuses méthodes ont été développées en classification d'images. Nous avons abordé dans la partie précédente, les méthodes de Boosting. Ces approches calculent un hyperplan séparant les données d'apprentissage dans un nouvel espace, engendré par les différents classifieurs faibles sélectionnés. Chaque classifieur faible peut en effet être vu comme une fonction de changement d'espace.

La littérature des méthodes par changement d'espace pour le calcul d'un hyperplan, mieux adapté à la disposition des exemples d'apprentissage, est riche. L'utilisation de fonctions noyaux à permis, en partie, le succès de ces techniques. Ces fonctions permettent de calculer un produit scalaire dans un nouvel espace de représentation des données. Associées à des techniques comme les SVM, elles permettent ainsi d'apprendre un hyperplan séparateur dans un nouvel espace de représentation des données ; voire de concaténer plusieurs sous-espaces de représentations des données pour des approches de type MKL.

Dans le cas du Boosting, ces sous-espaces de représentation sont engendrés par chaque classifieur faible. Ils sont calculés conjointement à l'hyperplan de manière itérative. Tandis que pour le MKL les sous-espaces sont prédéfinis à l'avance.

Dans un contexte de recherche multi-classes, les premières approches de type MKL imposent de recalculer le nouvel espace de représentation induit pour chacune des classes. Pour contourner ce problème, des approches plus récentes de type MKL permettent de séparer la construction de l'espace de représentation, du calcul de l'hyperplan. Nous proposons dans cette partie une nouvelle méthode basée Boosting qui permet, dans premier temps, de sélectionner les classifieurs faibles les mieux adaptés pour un ensemble de données (i.e les sous-espaces de représentation), le calcul de l'hyperplan séparateur sera effectué dans un second temps. Cette méthode a pour vocation de combiner les avantages des méthodes de MKL et la vitesse d'exécution des méthodes de Boosting.

Nous commencerons par présenter les méthodes par changement d'espace utilisant les fonctions noyaux. Nous illustrons l'application de ces fonctions par des exemples (PCA et SVM). Nous aborderons ensuite les méthodes de combinaisons de noyaux comme les MKL permettant de construire une nouvelle fonction noyau à partir de noyaux déjà existants, pour ensuite présenter la méthode que nous proposons.

CHAPITRE 6

Les noyaux

Sommaire

6.1 Introduction

Les tâches de classification, catégorisation, de recherche de copies... nécessitent une mesure de similarité entre images. L'objectif de ces tâches repose sur une détection d'une forme de régularité ou de regroupement d'images dans l'ensemble permettant de les décrire.

La recherche d'une forme de régularité d'un ensemble de données n'est pas toujours une opération facile ou même possible. Il est parfois plus facile de changer d'espace de représentation et d'effectuer le traitement dans ce nouvel espace plus adapté à l'objectif de la tâche que l'on a assignée au système de recherche.

Pour illustrer notre propos, nous allons introduire les concepts de ce chapitre au travers d'un exemple.

Nous allons chercher à séparer un ensemble de n exemples en deux classes distinctes. Chaque exemple correspond à un point dans un espace à deux dimensions. On cherche donc à tracer une frontière délimitant deux ensembles annotés par une fonction oracle, l'un des ensembles est annoté positivement l'autre négativement, comme sur la figure (Fig. 6.1).

6.1.1 Un produit scalaire

Une manière simple de séparer deux ensembles deux points et de calculer un hyperplan séparateur tel que les points d'un côté de l'hyperplan appartiennent à une catégorie et ceux de l'autre côté appartiennent à l'autre catégorie. Cet hyperplan est décrit par le vecteur qui lui est orthogonal, appelé normale.

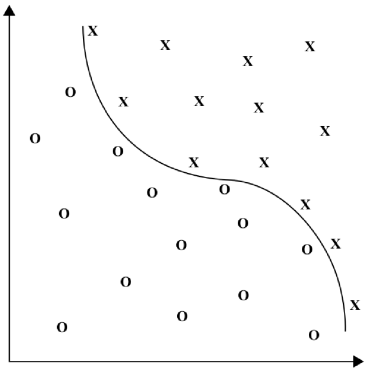


FIGURE 6.1 – Séparation de deux ensembles de points en deux classes distincts

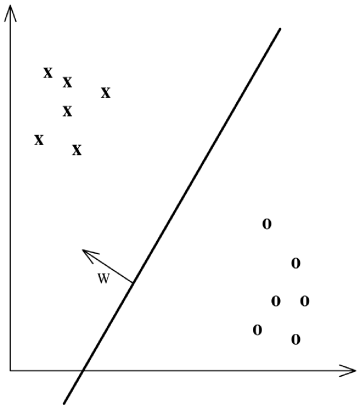


FIGURE 6.2 – Séparation d'un ensemble de point

Définition 4 (Produit scalaire)**.** *Soit $\mathcal{E}$ un espace vectoriel, l'application $k : \mathcal{E} \times \mathcal{E} \longrightarrow \mathbb{R}$ est un produit scalaire dans $\mathcal{E}$, si k est une application symétrique bilinéaire strictement positive.*

Un produit scalaire k, est une fonction qui vérifie :

- *Symétrie :* $\forall (x_1, x_2) \in \mathcal{X}^2, k(x_1, x_2) = k(x_2, x_1)$,
- *Bilinéaire :* $\forall (x_1, x_2, x_3) \in \mathcal{X}^3, k(x_1 + x_3, x_2) = k(x_1, x_2 + x_3)$ *et* $\forall \alpha \in \mathbb{R}, k(\alpha x_1, x_2) = k(x_1, \alpha x_2) = \alpha k(x_1, x_2)$,
- *Définie positive :* $\forall x \in \mathcal{X} \setminus \{0\}, k(x, x) > 0$ *et* $\forall x \in \mathcal{X} k(x, x) = 0 \Longleftrightarrow x = 0$.

Ainsi les exemples seront classés à l'aide d'une fonction de décision de la forme :

$$f : \left| \begin{array}{ccc} \mathcal{X} & \longrightarrow & \mathbb{R} \\ x & \longmapsto & f(x) = \langle w, x \rangle + b \end{array} \right. \tag{6.1}$$

avec w un vecteur normal à l'hyperplan et b un décalage (Fig. 6.2).

Si $f(x) > 0$ l'exemple x sera dans la classe des positifs, sinon il sera considéré comme négatif.

La recherche de cet hyperplan n'est pas toujours possible, les données pouvant être non-linéairement séparables (Fig. 6.1). Un exemple simple de non-séparabilité est de disposer les points selon un damier, il est impossible de trouver un unique hyperplan séparant toute les cases blanches des cases noires (Fig. 6.3).

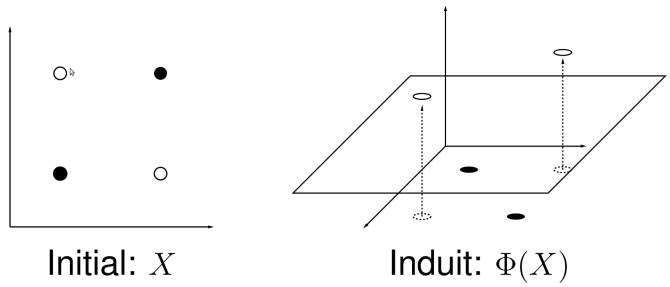


FIGURE 6.3 – Utilisation d'une fonction de changement d'espace pour rendre un problème linéairement séparable.

6.1.2 Changement d'espace de représentation

Pour s'affranchir de ce problème, l'idée est de changer l'espace de représentation des données et de trouver un nouvel espace où nos données deviennent linéairement séparables. On introduit pour cela une fonction ϕ de changement d'espace qui redéfinit la position des points associés à chaque image. On espère alors que dans ce nouvel espace une solution de classification linéaire existe. Cet espace doit être muni d'un produit scalaire afin de pouvoir déterminer l'hyperplan séparateur de nos données.

On définit pour cela les espaces :

Définition 5 (Espace préhilbertien)**.** *Un espace préhilbertien est un espace vectoriel muni d'un produit scalaire.*

Définition 6 (Espace complet)**.** *Un espace complet M est un espace où toute suite de Cauchy de M a une limite dans M. Autrement dit toute suite convergence de l'espace M converge dans M.*

Les noyaux présentés dans cette thèse travaillerons tous dans un espace de Hilbert défini par :

Définition 7 (Espace de Hilbert)**.** *Un espace de Hilbert $\mathcal{H}$ est un espace préhilbertien complet dont la norme $\|\cdot\|$ découle d'un produit scalaire $\langle\cdot,\cdot\rangle$ vérifiant $\|x\| = \sqrt{\langle x, x\rangle}$.*

On transforme donc l'espace des données $\mathcal{X}$ en un nouvel espace hilbertien $\mathcal{H}$ à l'aide de la fonction de changement d'espace $\phi : \mathcal{X} \longrightarrow \mathcal{H}$.

La figure (Fig. 6.3) nous montre un exemple d'application de ce principe sur un espace de départ de la forme d'un damier. L'espace de départ est composé de quatre points disposés selon une fonction XOR, les deux classes ne sont pas linéairement séparables. L'application d'une fonction ϕ de changement d'espace bien choisie permet de rendre le problème séparable par un hyperplan.

La fonction de décision est alors de la forme :

$$f : \left| \begin{array}{ccc} \mathcal{X} & \longrightarrow & \mathbb{R} \\ x & \longmapsto & f(x) = \langle w, \phi(x)\rangle + b \end{array} \right. . \tag{6.2}$$

Avec $b \in \mathbb{R}$ un facteur de translation et $w \in \mathbb{R}^n$ un vecteur normal à l'hyperplan séparateur dans l'espace induit.

On voit au travers de cette écriture que l'étude du produit scalaire dans l'espace induit est importante. On définit en conséquence la fonction noyau k qui calcule le produit scalaire dans l'espace induit à partir de l'espace initial où se trouvent nos données :

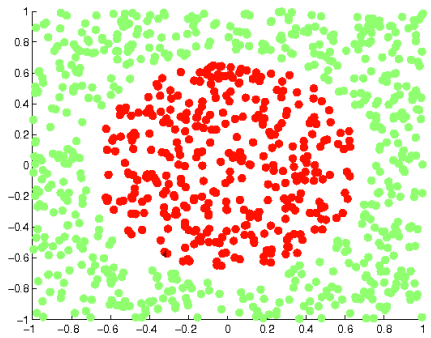


FIGURE 6.4 – Exemple de deux classes non-linéairement séparables dont l'une est placée à l'intérieur d'un disque et l'autre à l'extérieur

Définition 8 (Fonction noyau de Mercer)**.** *Une fonction k définie par*

$$k : \left| \begin{array}{ccc} \mathcal{X} \times \mathcal{X} & \longrightarrow & \mathbb{R} \\ x_1, x_2 & \longmapsto & k(x_1, x_2) = \langle \phi(x_1), \phi(x_2) \rangle \end{array} \right. . \tag{6.3}$$

est appelé noyau de Mercer. Elle calcule le produit scalaire de deux points de l'espace d'entrée $\mathcal{X}$ projetés dans un espace hilbertien $\mathcal{H}$ par une transformation ϕ de changement d'espace.

D'autres définitions équivalentes des fonctions noyaux de Mercer existent.

6.1.3 Une première fonction noyau

Avant de présenter plus en détail les propriétés et les outils dont on dispose pour étudier et utiliser une fonction noyau, nous allons illustrer l'utilisation d'un noyau par l'exemple.

Considérons un espace d'entrée $X \subseteq \mathbb{R}^2$ à deux dimensions comme sur la figure (Fig. 6.4). Les exemples de la première classe sont disposés à l'intérieur d'un disque tandis que la deuxième classe remplit le reste de l'univers. Les deux classes sont clairement non-linéairement séparables dans cet espace.

Soit la fonction de changement d'espace suivante :

$$\phi : \left| \begin{array}{ccc} \mathcal{X} \subseteq \mathbb{R}^2 & \longrightarrow & \mathcal{F} \subseteq \mathbb{R}^3 \\ (x_1, x_2) & \longmapsto & (x_1^2, x_2^2, \sqrt{2} x_1 x_2) \end{array} \right. . \tag{6.4}$$

Cette fonction projette nos données dans un espace d'une dimension supérieure. Dans ce nouvel espace, nos exemples sont maintenant linéairement séparables comme le montre la figure (Fig. 6.5).

Le produit scalaire dans cet espace correspond à la fonction :

$$\begin{aligned} \langle \phi(x), \phi(z) \rangle &= \langle (x_1^2, x_2^2, \sqrt{2} x_1 x_2), (z_1^2, z_2^2, \sqrt{2} z_1 z_2) \rangle & (6.5) \\ &= x_1^2 z_1^2 + x_2^2 z_2^2 + 2 x_1 x_2 z_1 z_2 & (6.6) \\ &= (x_1 z_1 + x_2 z_2)^2 & (6.7) \\ &= \langle x, z \rangle^2 & (6.8) \end{aligned}$$

On remarque alors qu'il n'est pas nécessaire d'expliciter l'espace induit pour calculer le produit scalaire dans cet espace et qu'il est possible de le faire depuis l'espace d'origine. Le calcul d'une fonction

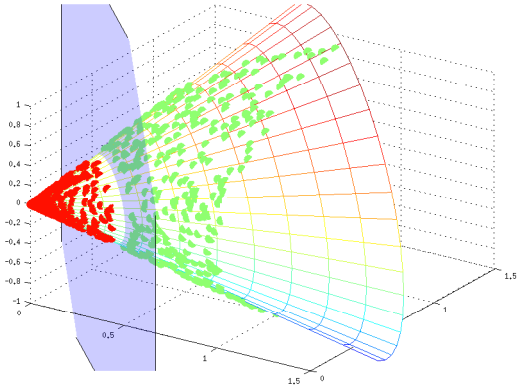


FIGURE 6.5 – Représentation dans un nouvel espace de deux classes initialement non-linéairement séparables

noyau ne requiert donc pas nécessairement l'explicitation de l'espace induit et peut être réalisé dans l'espace initial. C'est un avantage calculatoire certain lorsque l'espace induit est de plus grandes dimensions et même indispensable pour des espaces induits de dimensions infinies.

Si on choisit maintenant la fonction de changement d'espace suivante :

$$\phi : \left| \begin{array}{ccc} \mathcal{X} \subseteq \mathbb{R}^2 & \longrightarrow & \mathcal{F} \subseteq \mathbb{R}^4 \\ (x_1, x_2) & \longmapsto & (x_1^2, x_2^2, x_1x_2, x_2x_1) \end{array} \right. . \tag{6.9}$$

Nos données sont maintenant associées à un espace à quatre dimensions où elles sont également séparables par un hyperplan.

De plus on remarque que l'expression du produit scalaire dans cet espace nous donne :

$$\begin{aligned} \langle \phi(x), \phi(z) \rangle &= \langle (x_1^2, x_2^2, x_1x_2, x_2x_1), (z_1^2, z_2^2, z_1z_2, z_1z_2) \rangle && (6.10) \\ &= x_1^2z_1^2 + x_2^2z_2^2 + x_1x_2z_1z_2 + x_2x_1z_2z_1 && (6.11) \\ &= x_1^2z_1^2 + x_2^2z_2^2 + 2x_1x_2z_1z_2 && (6.12) \\ &= (x_1z_1 + x_2z_2)^2 && (6.13) \\ &= \langle x, z \rangle^2 && (6.14) \end{aligned}$$

On retrouve exactement la même fonction noyau que pour l'espace précédent. On a donc exhibé deux espaces différents qui donnent les mêmes valeurs de produit scalaire en partant de l'espace d'origine. On a ainsi montré que la fonction de changement d'espace n'est pas unique pour une fonction noyau donnée.

Ce noyau peut être généralisé à des espaces de départ de plus grandes dimensions, on peut prendre par exemple le changement d'espace suivant :

$$\phi : \left| \begin{array}{ccc} \mathcal{X} \subseteq \mathbb{R}^n & \longrightarrow & \mathcal{F} \subseteq \mathbb{R}^{n^2} \\ (x_1, \cdots, x_2) & \longmapsto & (x_ix_j)_{i,j=1,\cdots,n} \end{array} \right. . \tag{6.15}$$

On a alors la fonction noyau qui est la suivante :

$$\begin{aligned}
\langle \phi(x), \phi(z) \rangle &= \langle (x_i x_j)_{i,j=1,\cdots,n}, (z_i z_j)_{i,j=1,\cdots,n} \rangle && (6.16)\\
&= \sum_{i,j=1}^{n} x_i x_j z_i z_j && (6.17)\\
&= \sum_{i,j=1}^{n} x_i x_j \sum_{i,j=1}^{n} z_i z_j && (6.18)\\
&= \langle x, z \rangle^2 && (6.19)
\end{aligned}$$

Ce noyau est un noyau polynomial d'ordre 2.

6.2 Propriétés des fonctions noyaux

6.2.1 Fonctions noyaux et matrices de Gram

Pour un ensemble d'images données, il est possible de définir la matrice des similarités entre chacune d'elles. Pour un ensemble de données et pour un noyau fourni, on peut donc associer une matrice des valeurs du noyau pour chaque paire d'exemples. On appelle cette matrice la matrice de Gram associée à la fonction noyau étudiée.

Définition 9 (Matrice de Gram)**.** *Pour toute fonction noyau $k : \mathcal{X} \longrightarrow \mathbb{R}$ et un ensemble d'exemples $x_1, \cdots, x_n \in \mathcal{X}$, on définit la matrice de Gram K comme la matrice $n \times n$ des produits scalaires des exemples entre eux : $K_{i,j} = k(x_i, x_j)$.*

Cette matrice caractérise entièrement la fonction noyau sur l'ensemble étudié.

Si X est la matrice des exemples dans l'espace induit. C'est à dire si chaque ligne de X correspond aux m coordonnées d'un exemple dans l'espace induit $X = (\phi(x)_1 \cdots \phi(x)_m)^\top$, la matrice de Gram est égale à la matrice $G = XX^\top$.

Elle vérifie plusieurs propriétés intéressantes, dont notamment les propriétés suivantes.

Propriété 1. *Toute matrice de Gram K vérifie les propriétés suivantes :*
- *K est symétrique, $\forall (i,j) \in \mathbb{N}^2, K_{i,j} = K_{j,i}$.*
- *K est semi-définie positive, $\forall \lambda \in \mathrm{spec}(K), \lambda \geq 0$.*

Démonstration. La symétrie se montre par :

$$\begin{aligned}
(i,j) \in \mathbb{N}^2, K_{i,j} &= k(x_i, x_j) && (6.20)\\
&= \langle \phi(x_i), \phi(x_j) \rangle && (6.21)\\
&= \langle \phi(x_j), \phi(x_i) \rangle && (6.22)\\
&= k(x_j, x_i) && (6.23)\\
&= K_{j,j}. && (6.24)\\
& && (6.25)
\end{aligned}$$

La semi-définie positivité se montre par :

Soit X la matrice décrivant les exemples d'apprentissage, la matrice de Gram correspond à la matrice $K = XX^\top$. On a donc

$$\begin{aligned} \forall x \in \mathbb{R}^{*n} x^\top K x &= x^\top X X^\top x && (6.26)\\ &= (X^\top x)^\top (X^\top x) && (6.27)\\ &= \|X^\top x\|^2 \geq 0 && (6.28) \end{aligned}$$

La matrice K est donc semi-définie positive. □

Un théorème important lie les matrices semi-définies positives et les fonctions noyaux de Mercer, en effet on a :

Théorème 1. *Une matrice K est une matrice de Gram équivaut à dire que K est une matrice semi-définie positive.*

Démonstration. Nous avons montré précédemment que les matrices de Gram étaient bien semi-définies positives. Réciproquement si K est semi-définie positive, alors il existe une décomposition en vecteurs propres, valeurs propres telles que :

$$K = PDP^\top, \tag{6.29}$$

avec P unitaire et D la matrice diagonale avec les valeurs propres sur la diagonale. Comme toutes les valeurs propres sont positives on peut écrire :

$$K = (P\sqrt{D})(P\sqrt{D})^\top. \tag{6.30}$$

En posant $X = P\sqrt{D}$ on montre que K est bien une matrice de Gram. □

Une matrice semi-définie positive induit donc un produit scalaire sur un certain espace de Hilbert qu'il n'est pas nécessaire d'expliciter. La construction d'une matrice semi-définie positive suffit donc à définir une fonction noyau. Un espace de représentation possible dans lequel est effectué le produit scalaire est donné par les vecteurs propres et valeurs propres de la matrice de Gram. Chaque exemple peut être représenté selon les valeurs de la matrice $X = P\sqrt{D}$ avec P la matrice de vecteurs propres et D la matrice diagonale des valeurs propres.

6.2.2 Les mesures de qualité des fonctions noyaux

La construction d'une fonction noyau performante est une opération compliquée. Plusieurs outils ont été mis au point pour estimer la qualité de la fonction noyau pour les tâches de classification pour lesquelles elle sera utilisée.

6.2.2.1 Cosinus entre paires d'exemples

Dans une tâche de classification on s'attend généralement à ce que deux exemples appartenant à des classes différentes soient le plus dissemblables possible et que deux exemples de la même classe soient similaires. Intuitivement si les images sont représentées par des vecteurs, on souhaite que les images d'une même classe correspondent à des vecteurs colinéaires tandis que des images de classes différentes correspondent à des vecteurs orthogonaux. Une mesure de qualité de notre fonction noyau est donc de

calculer les cosinus entre les vecteurs représentant des images de classes différentes et de chercher à minimiser ces valeurs. Un premier critère de qualité d'une fonction noyau peut donc être la valeur :

$$\sum_{y_i=+1}\sum_{y_j=-1} cos(x_i,x_j) = \sum_{y_i=+1}\sum_{y_j=-1} \frac{k(x_i,x_j)}{\sqrt{k(x_i,x_i)k(x_j,x_j)}}. \tag{6.31}$$

[Meila 2003] propose ainsi le critère suivant :

$$J(\mathbf{K}) = \sum_{y_i=+1}\sum_{y_j=-1} \mathbf{K}(x_i,x_j). \tag{6.32}$$

6.2.2.2 Séparabilité de classe (Class separability)

Afin de bien séparer les images entre elles, on cherche généralement des espaces où les classes sont bien séparées tandis que les exemples d'une même classe sont regroupés autour d'un même centre. Afin de rendre compte de ces propriétés, [Wang 2002] introduit un critère mesurant la séparabilité entre les différentes classes.

Pour cela les auteurs mesurent l'éclatement des exemples par rapport à la moyenne de la classe :

$$tr(\mathbf{S_W}) = \sum_{i=1}^{c}\sum_{j=1}^{n_i} \|\phi(x_j)-\mu_i\|^2 \tag{6.33}$$

$$= \sum_{i=1}^{c} n_i \left[\sum_{i=1}^{c} \mathbf{K}(x_j,x_j) - \frac{Sum(\mathbf{K}_{D_i,D_i})}{n_i}\right], \tag{6.34}$$

n_i est le nombre d'exemples de la classe i, n est le nombre total d'images d'apprentissage, μ_i est le descripteur moyen de la classe i, μ est le descripteur moyen de toutes les images, $\phi(x_j)$ est le descripteur de l'image x_j, $\mathbf{K}(x_i,x_j)$ est la valeur de la fonction noyau pour les images x_i et x_j et $Sum(\mathbf{K}_{D_i,D_j})$ est la somme des valeurs de la fonction noyau sur l'ensemble des images de la classe i comparées au images de la classe j.

Ils calculent également la moyenne des éclatements entre classes :

$$\mathrm{Tr}(\mathbf{S_B}) = \sum_{i=1}^{c} n_i\|\mu_i-\mu\| \tag{6.35}$$

$$= \sum_{i=1}^{c} n_i \left[\frac{Sum(\mathbf{K}_{D_i,D_i})}{n_i^2} - 2\frac{Sum(\mathbf{K}_{D_i,D})}{n_i n} + \frac{Sum(\mathbf{K}_{D,D})}{n^2}\right]. \tag{6.36}$$

Leurs critères est alors le rapport entre la séparation des exemples au sein des classes et la séparation des classes entre elles :

$$J = \frac{\mathrm{Tr}(\mathbf{S}_B)}{\mathrm{Tr}(\mathbf{S}_W)}. \tag{6.37}$$

L'idée est que l'on veux des exemples regroupés autour de leur centre de classe (c'est à dire des petites valeurs pour $\mathbf{S}_W$) et des classes les plus éloignées les unes des autres (c'est à dire de grandes valeurs pour $\mathbf{S}_B$). On cherche donc des noyaux qui maximisent J.

6.2.2.3 Alignement de noyau (Kernel Alignment)

Une autre approche, que nous utiliserons dans cette thèse, est de comparer la fonction noyau que l'on cherche à construire à une fonction cible construite par exemple sur les annotations de l'utilisateur. On s'intéresse alors à des critères comparant ces deux fonctions noyaux.

La fonction noyau cible que l'on cherche à atteindre peut correspondre par exemple, à la fonction comparant les résultats d'un oracle entre les images. C'est à dire la fonction $\mathbf{k}^\star(x_i, x_j) = y_i y_j$, où y_i est le label que la fonction oracle associe à l'image x_i.

Avant de définir un critère de comparaison de fonction noyau, nous allons donner quelques définitions préliminaires.

Définition 10. *Soit $\mathbb{E}_x(\Phi)$ l'espérance de l'application Φ pour la variable x :*

$$\mathbb{E}_x(\Phi) = \int_X \Phi(x) \mathrm{d}P(x). \tag{6.38}$$

Soit $\mathbb{E}_{x,x'}(k)$ l'espérance de l'application k pour les variables x et x' :

$$\mathbb{E}_{x,x'}(k) = \iint_{X\times X} k(x, x') \mathrm{d}P(x) \mathrm{d}P(x'). \tag{6.39}$$

Définition 11. *On définit le noyau centré $\overline{k}$ comme l'application de $X \times X$ dans $\mathbb{R}$ définie pour tout $x, x' \in X$ par*

$$\begin{aligned} \overline{k}(x, x') &= (\Phi(x) - \mathbb{E}_x[\Phi])^\top (\Phi(x') - \mathbb{E}_{x'}[\Phi]) && (6.40)\\ &= k(x, x') - \mathbb{E}_x[k(x, x')] - \mathbb{E}_{x'}[k(x, x')] + \mathbb{E}_{x,x'}[k(x, x')]. && (6.41) \end{aligned}$$

Dans la suite de ce document nous utiliserons systématiquement des fonctions noyaux centrées.

Nous supposons l'existence d'une fonction noyau oracle $k^\star$ donnant la similarité sans erreur de toute image de la base. Cet oracle est l'analogue multi-classe de la fonction donnant les labels pour les classifieurs binaires. On souhaite construire notre noyau k en approchant au mieux cette fonction. Pour cela nous utilisons un critère de comparaison entre fonctions noyaux. Dans cette thèse, le critère choisi correspond à l'alignement centré :

Définition 12. *Soit deux fonctions noyaux k, $k^\star$ définies de $X \times X$ dans $\mathbb{R}$ telles que $0 < \mathbb{E}_{x,x'}[k^2] < +\infty$ et $0 < \mathbb{E}_{x,x'}[k^{\star 2}] < +\infty$. Alors l'alignement centré de ces deux fonctions noyaux est défini par :*

$$A_h(k, k^\star) = \frac{\mathbb{E}_{x,x'}(\overline{kk^\star})}{\sqrt{\mathbb{E}_{x,x'}(\overline{k}^2)\mathbb{E}_{x,x'}(\overline{k^\star}^2)}} \tag{6.42}$$

L'objectif est donc de construire un nouveau k donnant des résultats les plus similaires à ceux fournis par l'oracle $k^\star$ en maximisant le score d'alignement.

Les définitions introduites précédemment ne sont pas calculables en pratique, on préfère pour cela l'approche matricielle. Soit $(x_1, ..., x_n)$ un ensemble de n exemples d'apprentissage chacun décrit par un vecteur $X_i \in \mathbb{R}^p$. Dans cet espace discret, le noyau k peut être représenté à l'aide de la matrice de Gram $\mathbf{K}$ composée des valeurs de la fonction k prises pour chaque exemple x_i, x_j :

$$\mathbf{K}_{i,j} = k(x_i, x_j). \tag{6.43}$$

De manière analogue à l'approche en continue, on peut définir une matrice noyau centrée $\overline{\mathbf{K}}$ telle que :

Définition 13. *On définit le centrage d'une matrice $\overline{\mathbf{K}} \in \mathcal{M}_n$ comme étant la matrice*

$$\overline{\mathbf{K}} = \mathbf{HKH}, \tag{6.44}$$

avec $\mathbf{H} = Id_n - \frac{1}{n}1_n$ la matrice de centrage, Id_n la matrice identité de taille $n \times n$ et 1_n une matrice de taille $n \times n$ de valeur unitaire.

Le centrage de la matrice cible permet par exemple d'être indépendant au déséquilibre d'annotation entre classes.

La fonction oracle peut être discrétisée, on obtient alors une matrice cible $\mathbf{K}^\star$ que l'on souhaite reproduire à l'aide de la matrice $\mathbf{K}$ que l'on construit. Pour cela nous utilisons la version discrète de l'alignement centré que nous chercherons à maximiser.

L'alignement est une méthode de comparaison de fonction noyau similaire à une mesure cosinus. Les matrices de Gram sont vues comme des vecteurs, une fonction assimilable à un cosinus peut alors être calculée. Pour définir cette fonction, il est nécessaire de définir un produit scalaire sur les matrices :

Définition 14 (Produit scalaire de Frobenius)**.** *Soit deux matrices $A, B \in \mathcal{M}_{n\times m}(\mathbb{R})$, on définit le produit scalaire de Frobenius comme étant le réel :*

$$\langle A, B\rangle = \mathrm{Tr}(AB^\top). \tag{6.45}$$

On définit la norme associée comme étant

$$\|A\| = \sqrt{\langle A, A\rangle} = \sqrt{\mathrm{Tr}(AA^\top)}. \tag{6.46}$$

[Cristianini 2001, Cristianini 2006] définissent alors l'alignement, ici dans une version avec centrage, comme la fonction suivante :

Définition 15. *Soit deux matrices noyaux $\mathbf{K}$ et $\mathbf{K}^\star$ de $\mathcal{M}_n(\mathbb{R})$, telles que $\|\mathbf{K}\| \neq 0$ et $\|\mathbf{K}^\star\| \neq 0$. On définit l'alignement centré entre ces deux matrices comme étant le réel :*

$$\mathcal{A}_H(\mathbf{K}, \mathbf{K}^\star) = \frac{\langle \overline{\mathbf{K}}, \overline{\mathbf{K}^\star}\rangle}{\|\overline{\mathbf{K}}\|\|\overline{\mathbf{K}^\star}\|} = \frac{\mathrm{Tr}(\overline{\mathbf{K}}\ \overline{\mathbf{K}^\star}^\top)}{\sqrt{\mathrm{Tr}(\overline{\mathbf{K}}\ \overline{\mathbf{K}}^\top)\,\mathrm{Tr}(\overline{\mathbf{K}^\star}\ \overline{\mathbf{K}^\star}^\top)}} \tag{6.47}$$

Les alignements continus et discrets sont liés par la relation suivante :

Théorème 2. *[Cortes 2010] Les alignements $A_h(k, k^\star)$ et $\mathcal{A}_H(\mathbf{K}, \mathbf{K}^\star)$ satisfont la propriété suivante : Si $\forall x \in X$ tirés indépendamment et identiquement distribué selon une distribution D, $\exists R > 0$ tel que $k(x, x) < R$ et $k^\star(x, x) < R$ alors $\forall \delta > 0$ on a avec une probabilité de $1 - \delta$ l'inégalité suivante*

$$|A_h(k, k^\star) - \mathcal{A}_H(\mathbf{K}, \mathbf{K}^\star)| \leq 18\beta \left(\frac{3}{n} + 4\sqrt{\frac{\log \frac{2}{\delta}}{2n}} \right) \tag{6.48}$$

Avec $\beta = \max\left(\frac{R^4}{\mathbb{E}_{x,x'}(k^2)}, \frac{R^4}{\mathbb{E}_{x,x'}(k^{\star 2})}\right)$.

De plus l'alignement est directement lié à la distance entre les deux matrices grâce à la propriété suivante :

Propriété 2. *Soit deux matrices* $K_1, K_1 \in \mathcal{M}_{n\times m}(\mathbb{R})$*, on a la relation suivante :*

$$\frac{1}{2}\left\|\frac{K_1}{\|K_1\|} - \frac{K_2}{\|K_2\|}\right\|^2 = 1 - \mathcal{A}(K_1, K_2). \tag{6.49}$$

Démonstration.

$$\begin{aligned}
\left\|\frac{K_1}{\|K_1\|} - \frac{K_2}{\|K_2\|}\right\|^2 &= \left\langle \frac{K_1}{\|K_1\|} - \frac{K_2}{\|K_2\|}, \frac{K_1}{\|K_1\|} - \frac{K_2}{\|K_2\|} \right\rangle && (6.50)\\
&= \left\langle \frac{K_1}{\|K_1\|}, \frac{K_1}{\|K_1\|} \right\rangle + \left\langle \frac{K_2}{\|K_2\|}, \frac{K_2}{\|K_2\|} \right\rangle - 2\left\langle \frac{K_1}{\|K_1\|}, \frac{K_2}{\|K_2\|} \right\rangle && (6.51)\\
&= \frac{\|K_1\|^2}{\|K_1\|^2} + \frac{\|K_2\|^2}{\|K_2\|^2} - 2\frac{\langle K_1, K_2\rangle}{\|K_1\|\|K_2\|} && (6.52)\\
&= 2\left(1 - \mathcal{A}(K_1, K_2)\right). && (6.53)
\end{aligned}$$

□

6.2.3 Propriétés permettant de construire de nouvelles fonctions noyaux

La construction d'une fonction noyau est une étape importante permettant par la suite une bonne ou une mauvaise classification des données. Il existe certaines familles de fonctions noyaux classiques que nous verrons dans la section suivante mais il est également possible de construire des fonctions à l'aide de certaines propriétés.

On peut par exemple construire une nouvelle fonction noyau à partir de fonctions noyaux déjà construites.

Propriété 3 (p.75 de [Shawe-Taylor 2004])**.** *Soit* k_1 *et* k_2 *deux fonctions noyaux on a :*

- *La somme de fonctions noyaux est un noyau :* $k_1(x, z) + k_2(x, z)$
- *La multiplication d'une fonction noyau par un réel* α *positif non nul est une fonction noyau :* $\alpha k_1(x, z)$
- *Le produit de deux fonctions noyaux est une fonction noyau :* $k_1(x, z)k_2(x, z)$

Il également possible de construire une nouvelle fonction noyau à partir d'une fonction noyau et d'une fonction de changement d'espace :

Propriété 4 (p.75 de [Shawe-Taylor 2004])**.** *Soit* $k_3 : \mathbb{R}^n \times \mathbb{R}^n \longrightarrow \mathbb{R}$ *une fonction noyau et un changement d'espace* $\phi : X \longrightarrow \mathbb{R}^n$ *alors la fonction suivante est également un noyau :*

$$k_3(\phi(x), \phi(z)).$$

On peut également construire une fonction noyau de rang 1 à l'aide d'une fonction :

Propriété 5 (p.75 de [Shawe-Taylor 2004]). *Soit une fonction à valeurs réelles* $f : X \longrightarrow \mathbb{R}$ *alors la fonction suivante est un noyau :*

$$(x, z) \longmapsto f(x)f(z).$$

Nous rappelons également que d'après le théorème 1, la construction de toute matrice semi-définie positive induit une fonction noyau.

6.3 Exemples de fonctions noyaux classiques

6.3.1 Les noyaux linéaires

Les noyaux les plus simples sont les noyaux linéaires. Ils correspondent à une fonction de changement d'espace identité $\phi : x \longmapsto x$. La fonction noyau correspond alors à un produire scalaire dans l'espace d'origine.

$$k(x_i, x_j) = \langle x_i, x_j \rangle. \tag{6.54}$$

6.3.2 Les noyaux polynomiaux

Nous avons vu dans notre partie introductive, le noyau polynomial d'ordre deux. Ce noyau peut se généraliser aux ordres supérieurs.

Définition 16 (Noyau polynomial)**.**

$$k(x_i, x_j) = (\langle x_i, x_j \rangle + c)^q, \tag{6.55}$$

avec $c \in \mathbb{R}$ *et* $q \in \mathbb{R}^+$.

6.3.3 D'autres noyaux classiques

D'autres noyaux sont classiquement utilisés pour la classification d'images. On peut citer le noyau triangulaire :

Définition 17 (Noyau triangulaire)**.**

$$k(x_i, x_j) = 1 - \frac{\|x_i - x_j\|^2}{\sigma^2}, \tag{6.56}$$

avec $\sigma \in \mathbb{R}^*$.

Ainsi que le noyau Gaussien [Boser 1992, Guyon 1993, Vapnik 1995] qui travaille dans un espace induit de dimensions infinies :

Définition 18 (Noyau Gaussien)**.**

$$k(x_i, x_j) = e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}}, \tag{6.57}$$

avec $\sigma \in \mathbb{R}^*$.

CHAPITRE 7

Exemples de méthodes à noyaux

Sommaire

Nous avons présenté dans le chapitre précédent les fonctions noyaux. Une fonction noyau est un produit scalaire dans un certain espace et peut être vu comme l'application d'une fonction de changement d'espace puis un produit scalaire. Changer de fonction noyau consiste uniquement à remplacer la fonction de changement d'espace et donc à changer l'espace où se fait le produit scalaire. Une telle opération ne nécessite donc pas de modification dans un algorithme entièrement basé sur des produits scalaires. On peut donc établir un raisonnement en utilisant juste un produit scalaire classique et "noyauter" la méthode par la suite en remplaçant tous les produits scalaires par la fonction noyau que l'on souhaite utiliser. On appelle cela *l'astuce du noyau*. Elle fut introduite par [Aizerman 1964].

Nous allons présenter dans ce chapitre trois applications courantes de *l'astuce du noyau*, à savoir l'Analyse en Composantes Principales Noyautée (Kernel PCA en anglais), les Machines à Vecteurs de Support (SVM) et le Multiple Kernel Learning (MKL). Pour plus d'informations sur l'utilisation des fonctions noyaux en apprentissage, on pourra se référer au livre [Schölkopf 2002].

7.1 L'Analyse en Composantes Principales

Un ensemble de données décrivant des images ou tout autre objet statistique comporte généralement des redondances d'information ainsi que des données bruitées. Afin d'améliorer le traitement de ces données, il peut être intéressant de ne garder que les dimensions informatives et de réduire le bruit. L'analyse par composantes principales (Principal Component Analysis, PCA) a été introduite pour étudier la répartition de l'information parmi les variables de description des données. Elle est également utilisée comme étape de pré-traitement permettant de conserver uniquement les informations utiles, de supprimer la redondance et le bruit des données de départ. Cette approche cherche à définir un nouvel espace de représentation des données où chaque dimension est décorrélée et où le bruit a été réduit.

La PCA peut être réalisée par deux approches différentes, soit en utilisant la matrice de covariance, soit en utilisant la matrice de Gram. Nous présenterons dans un premier temps l'approche par matrice de covariance, puis dans un seconde temps l'approche par matrice de Gram. Nous verrons que l'astuce du noyau peut être utilisée dans ce dernier cas pour avoir une analyse non-linéaire.

7.1.1 Introduction à la PCA

L'analyse en composantes principales (PCA) a été introduite par Pearson [Pearson 1901] et développée par la suite par Hotelling [Hotelling 1933]. Cette méthode d'analyse d'un ensemble de données statistiques est utilisée dans de nombreux domaines comme la météorologie, l'économie,... et la classification d'images.

Chaque objet est représenté par un ensemble de n variables aléatoires, l'objectif de la PCA est de réduire ce nombre de variables en exploitant la corrélation éventuelle existante entre les variables initiales. On construit alors une nouvelle description des données à l'aide d'un nombre restreint de variables décorrélées et ordonnées par ordre de valeur informative. Ainsi la première dimension du nouvel espace contient le plus de variation entre chacune des images tandis que la dernière coordonnée contient le moins de variations. Si les variables initiales sont très corrélées, l'information donnée par les différentes variables aléatoires est très redondante, il sera alors nécessaire de garder peu de variables pour prendre en compte la quasi totalité de l'information.

7.1.2 Calcul de la PCA par la matrice de covariance

Nous abordons maintenant la formulation mathématique de ce problème par une approche matricielle. Les images sont décrites grâce à des descripteurs regroupés dans une matrice X. Chaque ligne de cette matrice permet de décrire l'image associée, tandis que chaque colonne est une variable aléatoire. On cherche donc une projection de ces données dans un nouvel espace tel que chaque dimension de cet espace soit décorrélée des autres et que les axes informatifs soient donnés dans l'ordre décroissant. On cherche $\tilde{X}$ tel que la première colonne de cette matrice contienne le plus d'information pour séparer les images tandis que la dernière le moins possible.

On commence par centrer les données. C'est à dire que le nuage de points est recentré sur l'origine. L'opération de centrage consiste à retrancher leur moyenne à chaque colonne, ce qui revient à appliquer la transformation :

$$\overline{X} = \left(\mathrm{Id}_n - \frac{1}{n} \mathbf{1}_n \right) X, \tag{7.1}$$

où $\mathbf{1}_n$ est une matrice de taille $n \times n$ de valeur unitaire.

On s'intéresse ensuite à la matrice de covariance $\Gamma = \overline{X}^\top \overline{X}$. Cette matrice est symétrique et peut se décomposer en $PAP^\top$, avec P la matrice des vecteurs propres et A matrice diagonale avec les valeurs propres sur la diagonale. L'idée est de projeter l'espace des X sur les premiers vecteurs propres :

$$\tilde{X} = XP. \tag{7.2}$$

Ainsi chaque colonne est bien décorrélée et si les vecteurs propres sont classés par ordre de valeur propre décroissante alors les premières colonnes contiennent bien les colonnes avec le plus de variations.

Dans le nouvel espace, les données sont centrées et la matrice de covariance est diagonale correspondant aux valeurs propres de la matrice de covariance initiale :

$$\overline{\tilde{X}} = \tilde{X} \text{ et } \tilde{X}^\top \tilde{X} = A. \tag{7.3}$$

En général on ne conserve pas toutes les valeurs propres, on ne garde que les dimensions les plus informatives et on supprime ainsi une partie du bruit.

La PCA minimise l'erreur quadratique moyenne entre l'espace d'origine et l'espace reconstruit. Cette erreur de reconstruction des données correspond à la somme des valeurs propres qui n'ont pas été sélectionnées pour la reconstruction.

7.1.3 Calcul de la PCA par la matrice de Gram

Nous venons de voir une méthode de calcul de la PCA à l'aide de la matrice de covariance, il est également possible de calculer la PCA à l'aide de la matrice de Gram [Scholkopf 1999].

L'idée est de ne plus utiliser la matrice de covariance mais de passer par la matrice de Gram pour effectuer les calculs :

$$K = \overline{XX}^\top \tag{7.4}$$

K est une matrice symétrique, elle est donc décomposable en

$$K = ULU^\top, \tag{7.5}$$

avec U la matrice des vecteurs propres et L la matrice des valeurs propres de K.

De plus comme U est une matrice orthogonale on peut réécrire la matrice de covariance :

$$\begin{aligned} \Gamma &= \overline{X}^\top \overline{X} && (7.6)\\ &= \overline{X}^\top UU^\top \overline{X} && (7.7)\\ &= \overline{X}^\top UL^{-\frac{1}{2}} LL^{-\frac{1}{2}} U^\top \overline{X}. && (7.8) \end{aligned}$$

Comme la décomposition spectrale est unique, on a :

$$P = \overline{X}^\top UL^{-\frac{1}{2}} \text{ et } A = L. \tag{7.9}$$

On peut donc calculer les nouvelles données ainsi :

$$\begin{aligned} \tilde{X} &= XP && (7.10)\\ &= \overline{XX}^\top UL^{-\frac{1}{2}} && (7.11)\\ &= KUL^{-\frac{1}{2}} && (7.12)\\ &= UL^{\frac{1}{2}}. && (7.13) \end{aligned}$$

Les nouvelles données se déduisent donc directement des vecteurs propres et valeurs propres de la matrice de Gram des données.

7.1.4 Utilisation de noyau pour le calcul de la PCA

L'utilisation de la PCA suppose une relation de dépendance linéaire entre les différentes variables du problème. [Scholkopf 1999] introduit la Kernel PCA (KPCA) pour s'affranchir de cette limitation. Les auteurs de cette méthode, utilisent le calcul de la PCA par matrice de Gram et l'astuce du noyau pour réaliser une PCA non-linéaire.

En utilisant l'astuce du noyau, on peut remplacer la matrice de Gram par n'importe quelle matrice semi-définie positive associée à une fonction noyau. On a alors

$$K_{i,j} = \langle \phi(x_i), \phi(x_j) \rangle.$$

En calculant les valeurs L et vecteurs propres U de la matrice de Gram centrée $\overline{K}$, il est alors possible de faire une PCA non-linéaire $\tilde{X} = UL^{\frac{1}{2}}$.

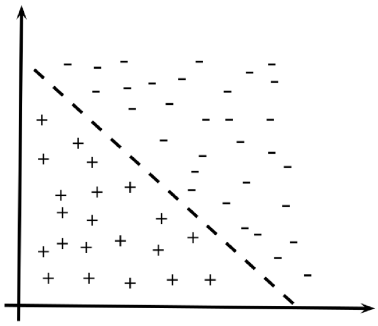


FIGURE 7.1 – Exemples de deux ensembles de points linéairement séparables

7.2 Les séparateurs à vaste marge (SVM)

Nous allons présenter dans cette partie, une méthode populaire de classification, les SVM. Cette méthode a été introduite dans [Boser 1992] comme une méthode de classification par hyperplan et l'astuce du noyau a été employée afin de traiter les cas non-linéaires. Nous commencerons par présenter cette méthode dans le cas linéaire, nous étudierons ensuite une solution pour traiter les cas non-linéaires, et enfin nous verrons comment l'utilisation du noyau offre une solution intéressante.

7.2.1 Les classifieurs par hyperplan

Lors d'une tâche de classification on cherche à définir une frontière de décision entre plusieurs ensembles de points. Dans le cas d'une classification binaire, on souhaite séparer l'espace en deux parties ; l'une comportant les objets d'intérêt et l'autre tous les objets que l'on ne recherche pas. L'approche la plus simple pour séparer un espace en deux sous-espaces est de définir un hyperplan, chaque côté définissant un sous-espace. On parle alors de classifieur par hyperplan, la fonction de décision consiste à regarder de quel côté de l'hyperplan se trouve le point que l'on veut tester.

Pour construire un tel classifieur, il est nécessaire de trouver l'hyperplan séparant au mieux nos données d'apprentissage. Cet hyperplan est défini à l'aide de son vecteur normal w et d'une constante de translation b. On cherche à construire une fonction f de la forme

$$f(x) = \langle w, x \rangle + b, \tag{7.14}$$

telle que les données appartenant à une même classe se trouvent du même côté de l'hyperplan tandis que les données appartenant à l'autre classe se trouvent de l'autre côté. Autrement dit, on cherche à trouver le vecteur w tel que

$$\forall (x_i, y_i) \in S, (\langle w, x_i \rangle + b) y_i > 0. \tag{7.15}$$

Pour illustrer nos propos, nous allons prendre un exemple. Imaginons des points définis sur un plan, nous voudrions séparer ces points en deux ensembles, les points notés + et les points notés -. Plusieurs cas se présentent à nous, tout d'abord le problème peut être linéairement séparable (Fig. 7.1) ou non (Fig. 7.2), il est alors possible ou non de trouver un hyperplan qui sépare nos points en deux ensembles. De plus dans le cas où nos données sont linéairement séparables, plusieurs hyperplans peuvent être validés (Fig. 7.3).

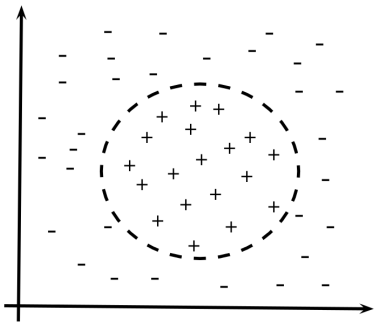


FIGURE 7.2 – Exemples de deux ensembles de points non-linéairement séparables

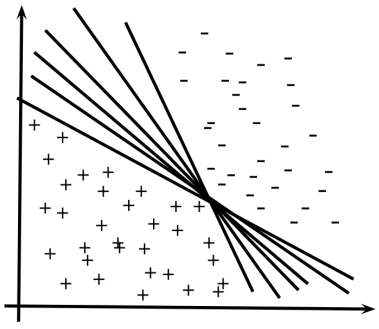


FIGURE 7.3 – Dans le cas où les données sont linéairement séparables, plusieurs hyperplans restent néanmoins possibles

L'approche par SVM propose une solution pour ces différents cas. Cette approche permet de choisir un unique hyperplan qui maximise un critère parmi tous les hyperplans possibles dans le cas linéaire. Elle propose également des solutions pour gérer les cas non-linéairement séparables soit en assouplissant la marge, soit en utilisant des fonctions noyaux permettant de changer d'espace de représentation.

7.2.1.1 Maximisation de la marge

Dans un problème de classification linéairement séparable, il existe une infinité d'hyperplans possibles ayant tous des performances en classification identique sur des données connues mais dont les performances en généralisation à de nouvelles données peuvent différer. [Vapnik 1998] montre que parmi tous ces hyperplans, le meilleur choix est de prendre celui qui maximise la marge ρ_w , c'est à dire celui tel que la plus petite distance entre un point et l'hyperplan soit maximale. Cet hyperplan est unique et possède de bonnes propriétés théoriques, notamment il permet de maximiser les capacités de généralisation du classifieur.

Définition 19 (Marge d'un classifieur par hyperplan)**.** *La marge d'un classifieur par hyperplan correspond au réel :*

$$\rho_w = \min_{i,x}\{\|x - x_i\| \textit{ et } \langle w, x\rangle + b = 0\} \tag{7.16}$$

$$= \min_i f(x_i)y_i. \tag{7.17}$$

La fonction de décision f est invariante par changement d'échelle, on peut donc choisir notre hyperplan tel que la marge soit de 1. C'est à dire tel que les données "supportant" cette marge (cette donnée correspond à celle utilisée pour définir la marge) vérifient $\langle w, x\rangle + b = \pm 1$. Prenons deux points x_p et x_n respectivement sur la marge positive et sur la marge négative (Fig. 7.4), on a :

$$\left\langle \frac{w}{\|w\|}, \frac{x_p - x_n}{\|w\|} \right\rangle = \frac{2}{\|w\|^2}. \tag{7.18}$$

Maximiser la marge revient donc à résoudre le problème d'optimisation suivant :

$$\arg\min_{w\in\mathcal{H}, b\in\mathbb{R}} \quad \frac{1}{2}\|w\|^2 \tag{7.19}$$

$$\text{sous contrainte de} \quad y_i(\langle w, x_i\rangle + b) \geq 1 \; \forall i = 1, \cdots, n. \tag{7.20}$$

Même si des solutions de résolution dans le primal existent [Chapelle 2007], on cherche généralement une solution en passant par la représentation duale. Ce problème d'optimisation peut se résoudre par la méthode classique des multiplicateurs de Lagrange. Le lagrangien est donnée par :

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i((w^\top x_i + b)y_i - 1). \tag{7.21}$$

Il doit être minimisé par rapport à w et b et maximisé selon les α_i.

En considérant les conditions de Kuhn-Tucker du premier ordre, c'est à dire en annulant les dérivées partielles du Lagrangien, on obtient le système d'équation suivant :

$$\begin{cases} w^\star = \sum_{i=1}^n \alpha_i x_i y_i \\ \sum_{i=1}^n \alpha_i y_i = 0 \end{cases} . \tag{7.22}$$

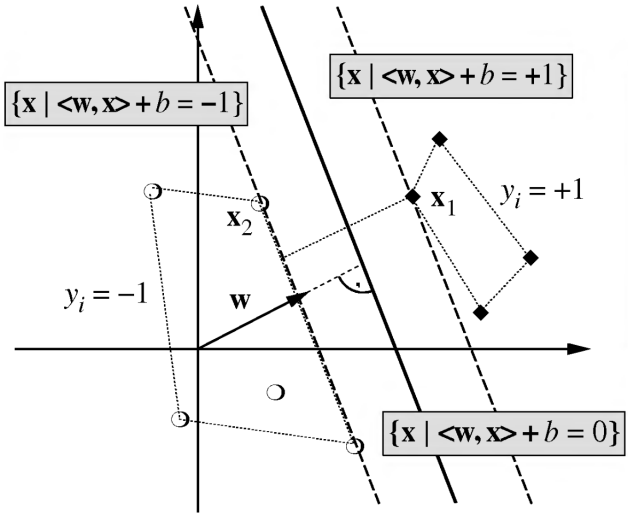


FIGURE 7.4 – Choix de l'hyperplan maximisant la marge

En injectant ces équations dans le problème d'origine, on obtient la formulation duale du problème d'optimisation que l'on cherche à résoudre :

$$\arg\max_{\alpha} L(\alpha) \quad = \quad \arg\max_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \tag{7.23}$$

$$\text{avec} \qquad \sum_i \alpha_i y_i = 0 \tag{7.24}$$

$$\text{et} \qquad \forall i, \alpha_i \geq 0. \tag{7.25}$$

La fonction de classification dans le dual est alors

$$f(x) = \sum_i \alpha_i y_i \langle x, x_i \rangle + b, \tag{7.26}$$

avec $\{x_i$ tel que $\alpha_i \neq 0, \}$ l'ensemble des exemples supports de l'hyperplan.

7.2.1.2 Cas non-linéairement séparable, une première solution : les SVM à marge souple (*Soft-margin SVM*)

Dans le cas linéaire (Fig. 7.5), il est toujours possible de trouver un hyperplan qui sépare nos données en deux ensembles en maximisant la marge. Dans un cas non-linéaire (Fig. 7.6), il n'existe pas d'hyperplan pour séparer nos données. Dans ce cas, une première idée est de relaxer les contraintes sur l'hyperplan et de tolérer des erreurs sur certains points de l'ensemble. Pour cela on définit une variable ξ_i pour chaque élément i qui correspond à la distance d'erreur avec la marge associée à la classe de l'exemple. L'objectif est bien sûr d'avoir les ξ_i les plus faibles possibles. La solution pour atteindre cet objectif est un SVM à marge souple proposé par [Cortes 1995].

Plus formellement on cherche toujours à minimiser la marge (Eq. 7.20) tout en minimisant également

$$\sum_{i=1}^{n} \xi_i^{\sigma}, \tag{7.27}$$

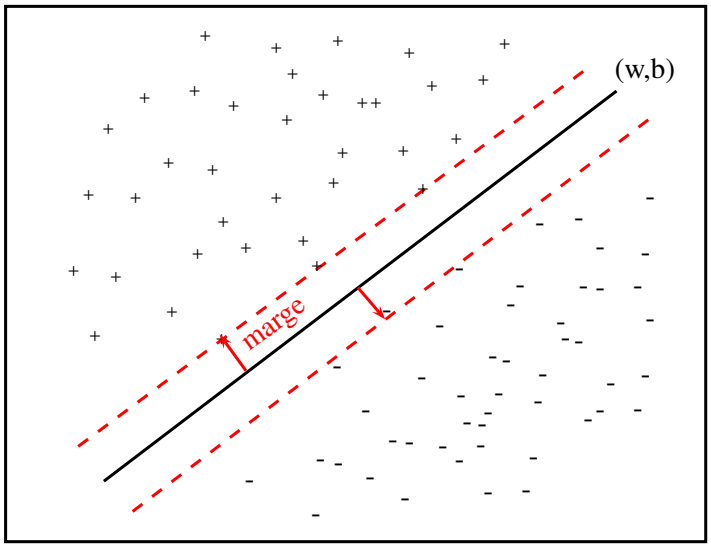


FIGURE 7.5 – Marge dans un cas linéairement séparable

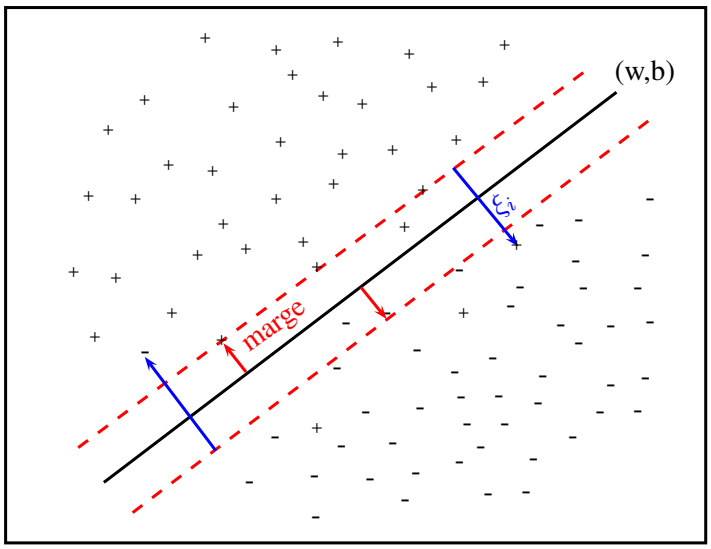


FIGURE 7.6 – Marge souple dans un cas non-linéairement séparable, ξ_i représente une variable de relaxation.

pour $\sigma > 0$ et sous la contrainte d'avoir

$$\forall i, y_i(\langle w, x_i\rangle + b) \geq 1 - \xi_i, \tag{7.28}$$
$$\xi_i \geq 0. \tag{7.29}$$

Pour prendre en compte ces deux problèmes d'optimisation, on peut formuler le problème suivant :

$$\arg\min_{w\in\mathcal{H}, b\in\mathbb{R}} \quad \frac{1}{2}\|w\|^2 + CF\left(\sum_{i=1}^{n}\xi_i^\sigma\right) \tag{7.30}$$
$$\text{sous contrainte de} \quad \forall i = 1, \cdots, n, y_i(\langle w, x_i\rangle + b) \geq 1 - \xi_i \tag{7.31}$$
$$\xi_i \geq 0, \tag{7.32}$$

avec F une fonction monotone convexe et C une constante permettant de régler l'importance de la régularisation dans le problème d'optimisation. Plus C sera grand, plus on autorise des erreurs sur l'hyperplan. L'hyperparamètre C permet un compromis entre la taille de la marge et le nombre d'erreurs tolérées par l'algorithme. On peut remarquer que dans le cas où les données sont linéairement séparables, la solution de ce problème d'optimisation coïncide parfaitement avec le problème des SVM classiques.

Classiquement on prend $\sigma = 1$ et $F = \text{Id}$. Dans ces conditions, le problème d'optimisation peut s'exprimer dans le dual par [Veropoulos 1999] :

$$\arg\max_{\alpha} L(\alpha) = \arg\max_{\alpha} \sum_i \alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \langle x_i, x_j\rangle \tag{7.33}$$
$$\text{avec} \quad \sum \alpha_i y_i = 0, \tag{7.34}$$
$$\text{et} \quad \forall i, 0 \leq \alpha_i \leq C. \tag{7.35}$$

La seul différence avec le problème précédent et que dans ce cas les α_i sont majorés par la constante C. La fonction de classification finale aura une forme similaire.

7.2.1.3 Cas non-linéairement séparable : utilisation d'une fonction noyau

Une autre solution pour traiter des cas non-linéaires est de changer l'espace de représentation des données et de passer dans un nouvel espace où elles seraient linéairement séparables. La forme duale du problème d'optimisation des SVM (Eq. 7.25) et des SVM à marge souple (Eq. 7.35) fait uniquement appel au produit scalaire entre exemples, il est alors possible d'utiliser à l'astuce du noyau et de remplacer ces produits scalaires par une fonction noyau k [Boser 1992]. On obtient donc le problème d'optimisation :

$$\arg\max_{\alpha} L(\alpha) = \arg\max_{\alpha} \sum_i \alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j k(x_i, x_j) \tag{7.36}$$
$$\text{avec} \quad \sum \alpha_i y_i = 0, \tag{7.37}$$
$$\text{et} \quad \forall i, 0 \leq \alpha_i (\leq C). \tag{7.38}$$

La fonction de classification finale sera alors de la forme :

$$f(x) = \sum_i \alpha_i y_i k(x, x_i) + b. \tag{7.39}$$

7.3 La combinaison de noyaux

7.3.1 Les principes et enjeux de la combinaison de noyaux

Le choix du type de fonction noyau utilisée est une opération importante qui impacte directement sur les résultats de la classification. Pour s'affranchir de l'avis d'un expert ou simplifier son travail, on cherche alors à construire une fonction noyau adaptée à notre problème de classification. Une façon simple de construire une nouvelle fonction noyau est de sommer des fonctions noyaux existantes (pour plus de détails voir la section 6.2.3). Cette opération revient à concaténer les espaces induits, par chaque fonction noyau, pour produire un nouvel espace de description où sera réalisé le produit scalaire final. La combinaison de noyaux offre donc la possibilité d'utiliser des fonctions noyaux basées sur différents types de descripteurs visuels, des noyaux de natures différentes (polynomial, gaussien, triangulaire,...) avec des paramétrages différents.

Il se pose alors la question du choix des noyaux à combiner et de leur éventuel poids dans la combinaison. Cette problématique a conduit à un nouveau champ de recherche désigné sous l'acronyme anglais de MKL (Multiple Kernel Learning). Dans cette thèse nous nous intéresserons uniquement à la combinaison linéaire de fonctions noyaux mais d'autres approches sont possibles par exemple avec des combinaisons géométriques [Picard 2012] ou des combinaisons plus générales [Varma 2009].

On souhaite construire un nouveau noyau par combinaison de T autres noyaux, tel que le noyau final soit de la forme :

$$k = \sum_{i=1}^{T} \beta_i k_i, \quad \forall i \quad \beta_i > 0. \tag{7.40}$$

Plusieurs méthodes ont été proposées pour calculer les pondérations β_i. Une solution simple consiste à fixer une pondération commune à toute les noyaux de la forme $\beta_i = \frac{1}{T}$. Cette solution ne prend pas en compte la variation de pertinence des différents noyaux pour trouver la combinaison la plus adaptée à un problème de classification donné. Des solutions basées sur la résolution de problèmes d'optimisation ont été proposées avec succès. Nous verrons dans les parties suivantes plusieurs approches pour le choix de ces pondérations.

7.3.2 Optimisation conjointe avec des SVMs

Les premières approches ont été proposées par [Lanckriet 2004]. Dans ces approches, les auteurs cherchent conjointement la meilleure combinaison de noyaux et l'hyperplan optimal d'un SVM. Ces approches reposent donc sur des formulations proches de celles des SVMs (pour plus de détail on pourra se référer à la section 7.2) dont le problème d'optimisation se formalisait par :

$$\underset{w\in\mathcal{H},b\in\mathbb{R}}{\arg\min} \quad \frac{1}{2}\|w\|^2 + C\sum_i \xi_i \tag{7.41}$$

$$\text{sous contraintes de} \quad y_i \langle w, \phi(x_i)\rangle + b \geq 1 - \xi_i \; \forall i = 1, \cdots, n \tag{7.42}$$

$$\xi_i \geq 0 \tag{7.43}$$

$$\tag{7.44}$$

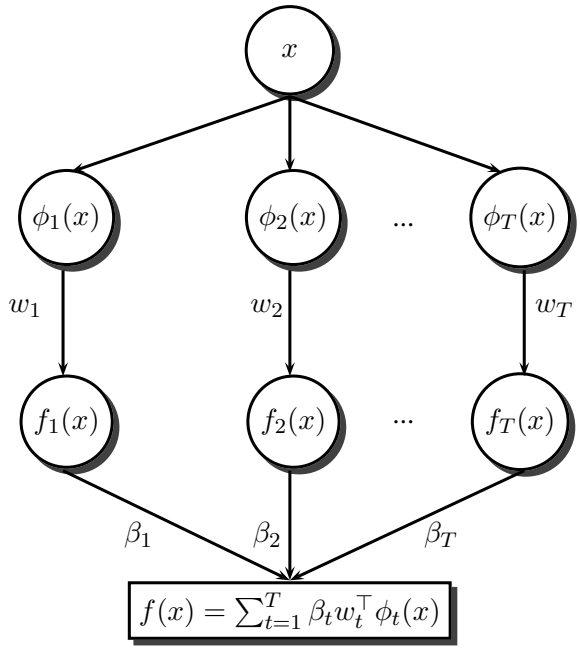


FIGURE 7.7 – Principe du MKL selon l'approche de [Lanckriet 2004].

La somme pondérée de noyaux est équivalente à réaliser le produit scalaire dans un espace où les différents espaces associés à chaque noyau auraient été combinés. Si on désigne par ϕ_t la fonction de changement d'espace associée au noyau k_t, alors sommer les fonctions noyaux k_t revient à réaliser le produit scalaire dans l'espace $(\phi_1, \cdots, \phi_T)$. On peut donc voir le MKL comme la recherche d'hyperplan dans chaque espace engendré par les ϕ_t puis la combinaison du résultat de chacune de ces fonctions pour engendrer un classifieur. Cette formulation correspond à la vision de [Lanckriet 2004] et est illustrée par le schéma (Fig. 7.7). On en déduit alors la formulation sous forme d'un problème d'optimisation quadratique (Quadratic Constraint Quadratic Programming QCQP) :

$$\underset{w_t \in \mathcal{H}, \beta_t \in \mathbb{R}, b \in \mathbb{R}}{\arg\min} \quad \frac{1}{2} \left(\sum_{t=1}^{T} \frac{\|w_t\|}{\sqrt{\beta_t}} \right)^2 + C \sum_i \xi_i \tag{7.45}$$

$$\text{sous contraintes de} \quad y_i \left(\left\langle \sum_{t=1}^{T} w_t, \phi_t(x_i) \right\rangle + b \right) \geq 1 - \xi_i \; \forall i = 1, \cdots, n \tag{7.46}$$

$$\xi_i \geq 0 \tag{7.47}$$

$$\sum_t \beta_t = 1 \tag{7.48}$$

$$\beta_t \geq 0. \tag{7.49}$$

Chaque w_t correspond à un vecteur normal à un hyperplan dans un espace induit par la fonction noyau ϕ_t. Comme le montre la figure (Fig. 7.7), on peut voir le MKL comme la somme pondérée de T projections sur des hyperplans calculés chacune dans un espace différent définis par la fonction de changement d'espace ϕ_t.

Le dual de ce problème fait apparaître les fonctions noyaux :

$$\underset{\gamma\in\mathbb{R},\alpha\in\mathbb{R}^n}{\arg\min} \quad \gamma - \sum_{i=1}^{n}\alpha_i \tag{7.50}$$

$$\text{sous contraintes de} \quad 0 \leq \alpha \leq C, \tag{7.51}$$

$$\sum_{i=1}^{n}\alpha_i y_i = 0 \tag{7.52}$$

$$\frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j k_t(x_i,x_j) \leq \beta_t^2\gamma \quad \forall t, \tag{7.53}$$

Au facteur de normalisation près, le γ optimal représente la plus grande norme au carré parmi les normes de chaque vecteur w_t orthogonal à l'hyperplan qui lui est associé. En effet on a $\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j k_t(x_i,x_j) = \|w_t\|^2$. L'équation dans le dual montre que l'on cherche à maximiser la marge de la plus mauvaise fonction f_t.

La formulation de [Lanckriet 2004] régularise les coefficients de pondération selon les deux normes mixtes (L_2,L_1), ce choix permet d'avoir une solution parcimonieuse en termes de nombre de noyaux sélectionnés.

Néanmoins cette solution n'est pas adaptée à un nombre important de noyaux et d'images. Pour cela [Bach 2004] propose une version plus régulière du problème de [Lanckriet 2004] en utilisant une régularisation de Moreau-Yosida afin de traiter des quantités de données plus importantes.

Plusieurs reformulation du problème ont été proposées afin notamment de pouvoir supporter de plus grandes quantités de noyaux et d'exemples. Ainsi [Sonnenburg 2006] ont formalisé le MKL comme un problème de programmation linéaire semi-infinie (Semi-Infinite Linear Program SILP). Ils utilisent l'algorithme *Column Generation Technique* pour obtenir des résultats plus rapidement. Plus récemment, [Rakotomamonjy 2008] reprend la formulation de [Sonnenburg 2006] mais optimise l'algorithme de résolution en utilisant une descente de gradient. Cette nouvelle méthode est appelée SimpleMKL est permet de surmonter les limites des problèmes SILP (les problèmes SILP utilisent une méthode de minimisation par plan de coupe qui sont connues pour leurs instabilités, ce qui peut conduire à des problèmes de convergence pour certains exemples).

D'autres formulations du problème MKL ont été données, on peut citer pour exemple [Bach 2008] qui utilise le groupe Lasso et [Kloft 2011] qui changent la norme utilisée dans la régularisation.

7.3.3 Choix de la pondération des noyaux séparée de l'étape de classification

Une autre façon d'aborder le problème du MKL a été proposée par [Cortes 2010]. Les auteurs décomposent le MKL en deux étapes. Ils séparent la partie sélection des pondérations des noyaux et le choix de l'hyperplan séparateur selon deux problèmes d'optimisation distincts. L'idée est d'éviter un sur-apprentissage dû à la création d'un espace de description trop spécifique à la tâche de classification que l'on effectue.

Le déroulement de cette approche en deux temps est illustré par la figure (Fig. 7.8).

Lors de la première étape, l'algorithme construit un noyau par combinaison d'autres noyaux. Il cherche à être le plus proche d'un noyau cible issu des annotations de l'utilisateur. Dans le cas d'une classification binaire, si $\mathbf{y}$ est le vecteur associé aux labels des images alors la matrice cible que l'on

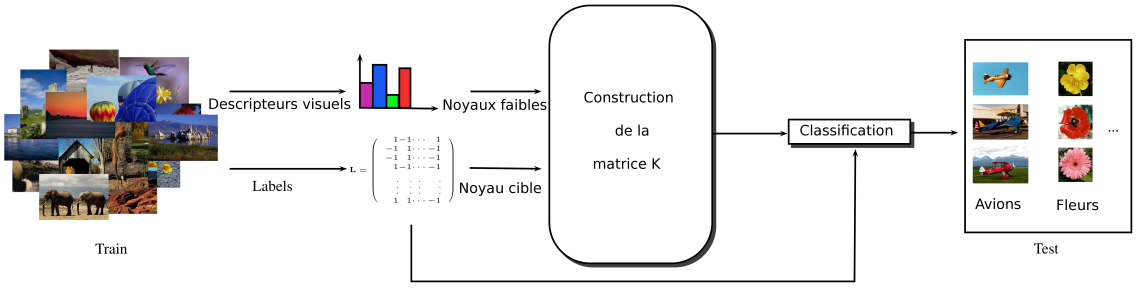


FIGURE 7.8 – MKL en deux temps

cherche à atteindre correspond à la matrice $\mathbf{K}^\top = \mathbf{y}\mathbf{y}^\top$. Le calcul des pondérations se fait par la maximisation d'un critère tel que l'alignement avec le noyau cible (pour plus de détail sur l'alignement on pourra se référer à la section 6.2.2.3).

De manière plus formelle pour un ensemble de noyaux $\{\mathbf{k}_t\}_t$, on cherche les pondérations optimales β_t telles que

$$\arg\max_{\beta} \frac{\langle \sum_{t=1}^{T} \beta_t \mathbf{k}_t, \mathbf{y}\mathbf{y}^\top \rangle}{\| \sum_{t=1}^{T} \beta_t \mathbf{k}_t \|}. \tag{7.54}$$

Ce problème peut être résolu analytiquement à l'aide de la propriété suivante :

Propriété 6 ([Cortes 2010]). *La solution optimale $\beta^\star$ du problème d'optimisation (Eq. 7.54) correspond à :*

$$\beta^\star = \frac{M^{-1}a}{\|M^{-1}a\|}, \tag{7.55}$$

avec M la matrice des produits scalaires entre matrices de Gram :

$$M_{i,j} = \langle \mathbf{K}_i, \mathbf{K}_j \rangle \tag{7.56}$$

et a le vecteur des produits scalaires entre les matrices de Gram et la matrice cible :

$$a_i = \langle \mathbf{K}_i, \mathbf{y}\mathbf{y}^\top \rangle. \tag{7.57}$$

Dans la deuxième étape, un hyperplan est calculé dans l'espace induit par le noyau issu de la première étape en fonction de la classe que l'on souhaite rechercher. Cette étape est identique à tout SVM muni d'un noyau particulier fourni en début d'algorithme, pour plus de détail, on pourra se référer à la section 7.2 sur les SVM.

7.3.4 Combinaison de fonctions noyaux par Boosting

7.3.4.1 L'approche de [Crammer 2002]

L'un des principaux inconvénients des méthodes de MKL classiques est qu'elles nécessitent un ensemble de noyaux fixés avant le lancement de l'algorithme. Certains noyaux peuvent alors se révéler non pertinents et obtenir une pondération nulle. Ces noyaux augmentent le temps de calcul de la méthode pour un apport informatif nul.

Il est alors intéressant de se pencher sur des méthodes permettant de sélectionner itérativement des noyaux parmi un ensemble de noyaux possibles et de calculer la pondération qui doit leur être appliquée. Il est envisageable pour cela d'utiliser des méthodes de Boosting qui répondent bien à des protocoles similaires sur des classifieurs faibles (les méthodes de Boosting font l'objet du chapitre 4).

[Crammer 2002] propose ainsi un algorithme pour la construction d'un noyau à l'aide d'une méthode de Boosting. Cet algorithme correspond à l'Algorithme 13.

Dans cette méthode on remplace le concept habituel d'exemples d'apprentissage par tous les couples d'images possibles. On appelle maintenant un exemple d'apprentissage X_{ij}, un couple de deux images (x_i, x_j). Avec cette astuce, la fonction noyau $k_t(x_i, x_j)$ peut être vue comme une fonction à une seule variable $k_t(X_{ij})$. Dans ces conditions, il est possible d'appliquer un algorithme tel qu'AdaBoost sur les exemples $\{X_{ij}\}_{i,j}$ et les fonctions noyaux $k_t : X_{ij} \longrightarrow k_t(X_{ij})$. Les auteurs de cet algorithme, proposent également une méthode pour construire les fonctions noyaux k_t à chaque itération à l'aide de l'Algorithme 14.

Cette approche permet donc bien d'étudier uniquement les noyaux qui seront effectivement sélectionnés par la méthode mais le coût en complexité de cette approche est quadratique en nombre d'exemples d'apprentissage. Nous verrons dans la méthode que nous proposons comme s'affranchir de cette contrainte et réaliser la construction d'une nouvelle fonction noyau par une méthode de Boosting en complexité temporelle linéaire en nombre d'images d'entraînements.

7.3.4.2 L'approche de [Gehler 2009]

Dans l'approche de [Cortes 2010], le MKL est un processus en deux étapes. La première permet de définir une combinaison linéaire des fonctions noyaux mises à disposition de l'algorithme et la seconde consiste à déterminer l'hyperplan séparant les données dans le nouvel espace induit par cette fonction noyau. [Gehler 2009] proposent d'inverser ces deux étapes. Dans un premier temps, un hyperplan est défini pour chacun des sous-espaces, puis dans une seconde étape, la combinaison des résultats de classification de chaque sous-espace est réalisée.

L'algorithme qui en découle se déroule ainsi : tout d'abord un SVM est appliqué à chacun des noyaux disponibles. Cette étape permet de créer un ensemble de classifieur (SVM). Cet ensemble est alors utilisé afin de définir l'ensemble des classifieurs faibles sélectionnables d'un algorithme de Boosting. Ainsi la combinaison des projections dans chaque sous-espace est réalisée au moyen d'un algorithme de Boosting. Dans l'article [Gehler 2009], le choix de la méthode de Boosting s'est porté sur l'algorithme LP-Boost que nous avons présenté en section 4.3.3.

Cette approche montre bien que les algorithmes de Boosting et les techniques de MKL ont des objectifs semblables et réalisent toutes les deux un calcul d'hyperplan dans un nouvel espace de représentation des données qui est une concaténation de sous-espaces de représentation des données.

7.3.5 Conclusion

Nous avons vu dans cette section plusieurs approches possibles pour la combinaison linéaire de fonctions noyaux. La première approche fut proposée par [Lanckriet 2004] et cherche à résoudre conjointement la sélection des fonctions noyaux et l'hyperplan séparant les données dans l'espace induit. Cette première formulation souffre de plusieurs problèmes notamment si le nombre de noyaux et d'exemples

Algorithm 13: Boosting pour l'apprentissage noyau de [Crammer 2002]

Input: un ensemble d'exemples d'apprentissage annotés $S = \{(x_i, y_i)\}_{i=1}^{n}$.
Input: un ensemble d'exemples non-annotés $\tilde{S} = \{(\tilde{x}_i)\}_{i=1}^{\tilde{n}}$.
begin

$$K \leftarrow 0. \tag{7.58}$$

for $t = 0$ *to* T **do**
Calculer les distributions

$$D_t(i,j) = \begin{cases} \exp(-y_i y_j K(x_i, x_j)) & \text{ExpLoss} \\ \dfrac{1}{1+\exp(-y_i y_j K(x_i, x_j))} & \text{LogLoss} \end{cases}. \tag{7.59}$$

Construction du prochain noyau K_t à partir de $(D_t, S, \tilde{S})$ à l'aide de l'Algorithme. 14
Calcul des valeurs suivantes

$$S_t^+ = \{(i,j) | y_i y_j K(x_i, x_j) > 0\} \quad ; \quad S_t^- = \{(i,j) | y_i y_j K(x_i, x_j) < 0\}; \tag{7.60}$$

$$W_t^+ = \sum_{(i,j)\in S_t^+} D_t(i,j) |K_t(x_i, x_j)| \quad ; \quad W_t^- = \sum_{(i,j)\in S_t^-} D_t(i,j) |K_t(x_i, x_j)|. \tag{7.61}$$

Calcul de la pondération

$$\beta_t = \frac{1}{2} \ln \left(\frac{W_t^+}{W_t^-} \right). \tag{7.62}$$

Mise à jour du noyau

$$K \leftarrow K + \beta_t K_t. \tag{7.63}$$

Output: le noyau $K : \mathcal{X} \times \mathcal{X} \longrightarrow \mathbb{R}$.

sont importants. Pour faire face à ces contraintes des reformulations du problème d'optimisation des MKL ont été proposées [Bach 2004, Sonnenburg 2006, Rakotomamonjy 2008, Bach 2008, Kloft 2011].

En parallèle de ces développements, il fut proposé des méthodes séparant la sélection et le calcul de pondération des fonctions noyaux, de la détermination de l'hyperplan séparateur des données [Cortes 2010]. L'un des principaux soucis de cette approche, que l'on retrouve également dans les autres approches de MKL, est que l'ensemble des fonctions noyaux utilisées pour la combinaison linéaire est fixé à l'avance et ne peut pas évoluer en fonction de la convergence de l'algorithme. Cela conduit généralement à l'utilisation de grands ensembles de fonctions noyaux dont beaucoup ne sont pas utilisées dans le noyau final.

Pour limiter le nombre de fonctions noyaux employées et se concentrer sur des fonctions pertinentes,

Algorithm 14: Construction du noyau K_t

Input: un ensemble d'exemples d'apprentissage annotés $S = \{(x_i, y_i)\}_{i=1}^{n}$.
Input: un ensemble d'exemples non-annotés $\tilde{S} = \{(\tilde{x}_i)\}_{i=1}^{\tilde{n}}$.
Input: la distribution des exemples d'apprentissage annotés $D_t = \{D_t(x_i)\}_{i=1}^{n}$.
begin

Calculer les matrices :

$$A \in \mathbb{R}^{n \times \tilde{n}} \quad , \quad A_{i,r} = x_i^\top \tilde{x}_r; \tag{7.64}$$

$$B \in \mathbb{R}^{n \times n} \quad , \quad B_{i,j} = D_t(x_i, x_j) y_i y_j; \tag{7.65}$$

$$K \in \mathbb{R}^{\tilde{n} \times \tilde{n}} \quad , \quad K_{r,s} = \tilde{x_r}^\top \tilde{x}_s. \tag{7.66}$$

Trouver le vecteur propre $v \in \mathbb{R}^n$ résolvant le problème $A^\top B A v = \lambda K v$ associé à la plus grande valeur propre λ.
Calculer le vecteur

$$w = \frac{\sum_r v_r \tilde{x_r}}{\| \sum_r v_r \tilde{x_r} \|}. \tag{7.67}$$

Output: le noyau $K_t = w w^\top$.

des méthodes de construction de fonctions noyaux par Boosting ont été proposées [Crammer 2002]. L'un des principaux inconvénients de cette dernière approche est le coût quadratique de l'algorithme.

Enfin, [Gehler 2009] montrent que le Boosting à l'aide de classifieurs faibles de type SVM, peut être vu comme une approche similaire à un MKL en deux temps où le calcul de l'hyperplan est réalisé dans chaque sous-espace avant leur combinaison.

Que ce soient les approches par MKL ou l'approche par LP-Boost de [Gehler 2009], ces méthodes utilisent un ensemble de noyaux prédéfinis. Elles cherchent en conséquence, avant tout à réaliser une sélection des noyaux pertinents et faire ainsi le tri parmi les différents noyaux. Elles ne réalisent donc pas une entière conception du noyau final et reposent sur la qualité des noyaux pré-calculés. Afin de réaliser une meilleure conception d'un noyau et de le faire évoluer en fonction des propriétés qu'on recherche, [Crammer 2002] ont proposé une première solution de Boosting à noyau. Cette approche offre la possibilité d'adapter le noyau construit en fonction d'un critère d'optimisation (alignement avec un noyau cible idéal), il permet donc de mieux concevoir le noyau. Le problème de la conception de la fonction noyau est séparé du choix de l'hyperplan et les espaces induits ne sont plus seulement sélectionnés et pondérés, mais sont également construits pour permettre une meilleure classification par la suite. Néanmoins cette approche souffre d'une complexité calculatoire quadratique en nombre d'exemples.

Nous allons présenter dans le chapitre suivant une nouvelle méthode de construction de noyau par Boosting qui s'affranchit du problème de complexité de la méthode de [Crammer 2002] et qui permet néanmoins une vraie conception de la fonction noyau. Nous montrerons que la solution que nous proposons est linéaire en nombre d'images.

CHAPITRE 8

Méthode proposée pour l'apprentissage de noyau

Ces travaux ont fait l'objet des publications [Lechervy 2012a, Lechervy 2012c, Lechervy 2012b].

Sommaire

8.1 Apprentissage d'une fonction noyau

8.1.1 Objectifs et contraintes

L'objectif de la méthode présentée dans ce chapitre est d'établir une nouvelle approche permettant la conception d'une fonction noyau avec des propriétés intéressantes pour la classification, comme une bonne séparabilité des exemples en fonction des différentes classes de la base d'apprentissage. On souhaite réaliser une conception complète de l'espace induit par la fonction noyau et pas seulement une sélection pondérée d'espaces déjà existants.

A cela s'ajoute une contrainte de complexité, nous recherchons une solution donc la complexité serait au pire linéaire en le nombre d'exemples d'apprentissage afin de pouvoir appliquer notre approche à de grandes bases d'images.

La solution proposée doit permettre le calcul de la fonction noyau sur de nouveaux exemples non présents lors de la phase d'apprentissage.

8.1.2 Introduction à la méthode

La méthode que nous proposons dans ce chapitre a pour but de construire une fonction noyau en s'inspirant du cadre du Boosting. Nous utilisons une approche en deux temps, similaire à l'approche de [Cortes 2010] présentée dans la section 7.3.3.

Nous calculons dans un premier temps une fonction noyau commune à toutes les classes, puis nous construisons pour chaque classe l'hyperplan qui sépare les exemples de cette classe, du reste des données. L'intuition sur laquelle repose ce découpage en deux temps est de ne pas biaiser l'espace de représentation des données par la recherche d'une classe particulière et éviter ainsi un sur-apprentissage de cet espace.

Nous construisons notre fonction noyau de manière itérative par l'ajout de dimensions à l'espace induit. Nous cherchons à construire un espace de représentation où les données d'apprentissage sont placées de façon optimale afin de faciliter par la suite la séparation par hyperplan. Pour cela nous allons faire évoluer notre nouvel espace pour que chaque point de l'ensemble d'apprentissage converge vers une position cible dépendant de l'ensemble des classes.

Nous allons présenter dans les parties suivantes, la manière de construire la fonction noyau cible, le problème d'optimisation que l'on cherche à résoudre, et le lien entre la fonction noyau et l'espace sémantique que l'on construit.

8.1.3 Fonctions noyaux cibles et matrices cibles

Notre approche s'appuie sur les annotations d'utilisateurs pour construire une nouvelle fonction noyau qui permettra de classifier un ensemble d'images selon différentes classes. Les annotations des images de l'ensemble d'entraînement sont multi-classes : une image peut appartenir à plusieurs classes en même temps, par exemple l'image d'une course de chevaux peut appartenir à la classe cheval, sport et personne (si on voit le cavalier)...

La fonction noyau que l'on souhaite élaborer s'appuie sur ces annotations pour construire un espace qui a pour objectif de faciliter les futures séparations par hyperplan. Notre méthode cherche à tendre vers une fonction noyau cible qui correspond à un espace où les exemples d'apprentissage auraient été placés de façon optimale pour une séparation par hyperplan.

Pour résumer, les exemples d'apprentissage permettent de construire une fonction noyau où chacun est placé de manière optimale. Notre algorithme cherche alors à construire une fonction noyau qui, à l'aide de données visuelles tend vers cette fonction optimale $k^\star$.

La fonction noyau cible $k^\star$ est donc une fonction issue des labels auxquels on ajoute des propriétés intéressantes pour la séparation par hyperplan des exemples.

En première approche, on peut construire une fonction noyau cible à partir de la fonction oracle nous donnant les labels. Si pour deux exemples donnés la fonction oracle associe une classe commune alors la fonction noyau vaut 1 et -1 sinon. On peut aussi également imaginer une fonction noyau qui compte

$$L = \left(\begin{array}{c|ccccc} & y_1 & y_2 & y_3 & \cdots & y_c \\ \hline x_1 & 1 & -1 & -1 & \cdots & 1 \\ x_2 & -1 & 1 & -1 & \cdots & -1 \\ x_3 & -1 & 1 & -1 & \cdots & -1 \\ x_4 & 1 & -1 & 1 & \cdots & -1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \\ x_n & 1 & 1 & -1 & \cdots & -1 \end{array}\right)$$

FIGURE 8.1 – Exemple de matrice de label pour plusieurs classes

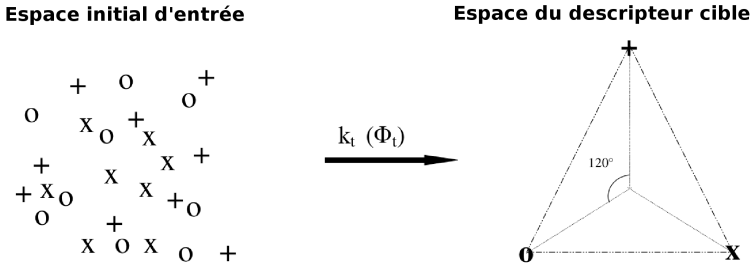


FIGURE 8.2 – Construction d'une fonction cible optimale de trois classes à partir des exemples d'apprentissage selon un simplex [Guermeur 2004]

le nombre de classes communes aux deux images. Ces deux fonctions sont bien des fonctions noyaux, la deuxième peut être représentée matriciellement à l'aide de la matrice $\mathbf{L}\mathbf{L}^\top$ où $\mathbf{L}$ est la matrice des labels par catégorie.

Supposons que les utilisateurs soient à la recherche d'un nombre fini c de catégories, parmi un ensemble connu d'images d'apprentissage n. Chaque image de la base d'apprentissage est annotée pour chaque catégorie positivement ou négativement. L'ensemble de ces annotations peuvent être regroupées dans une matrice $\mathbf{L}$ (un exemple est donné par la figure (Fig. 8.1)) où chaque ligne correspond aux annotations successives d'une même image selon chaque catégorie et chaque colonne aux annotations d'une catégorie donnée. Une valeur de 1 correspond à une annotation positive, une valeur de -1 une annotation négative. On peut également utiliser des valeurs de 0 pour signaler l'absence d'annotation utilisateur.

La fonction noyau $\mathbf{L}\mathbf{L}^\top$ répond bien au premier objectif d'une fonction cible : elle repose sur les annotations. En revanche elle n'a pas nécessairement de bonnes propriétés pour être utilisée dans la recherche d'hyperplans séparateurs de classes. Vapnik [Vapnik 1982] montre que l'éclatement optimal d'un ensemble de points en c catégories avec une marge optimale, est atteint lorsque les points se répartissent autour des c sommets d'un simplex unitaire de dimensions $c-1$ centré sur l'origine (Fig. 8.2). Cette configuration permet à la fois de maximiser la distance inter-classe et de minimiser la distance intra-classe. En effet un simplex unitaire de dimensions $c-1$ permet de placer chaque sommet à égale distance les uns des autres et ainsi maximiser la distance entre chaque sommet. De plus la concentration des exemples d'une même catégorie autour d'un même sommet du simplex permet de minimiser la distance intra-classe.

Pour retrouver ces propriétés, nous proposons dans ce document de considérer la matrice $\mathbf{Q}$ de la dé-

composition QR appliquée à la matrice $\mathbf{L}$. En effet, le calcul de la décomposition de QR peut être obtenu par la méthode de Gram-Schmidt et conduit à la construction d'un repère orthonormé. Les colonnes de la matrice $\mathbf{Q}$ correspondent alors aux coordonnées des vecteurs du nouveau repère tandis que la matrice $\mathbf{R}$ contient les poids de chaque vecteur permettant la reconstruction des vecteurs dans le repère d'origine. Nous nous restreignions aux colonnes de la matrice $\mathbf{Q}$ telles que les éléments associés sur la diagonale de la matrice $\mathbf{R}$ soient non nulles. La matrice $\mathbf{Q}$ ainsi construite est une matrice $n \times c$ de rang plein, elle est orthonormale, chaque colonne et donc chaque classe est indépendante des autres et son vecteur représentant à "égale distance" des autres. Tous d'abord cette matrice remplit bien le premier objectif, elle est construite à partie des annotations de l'utilisateur et définit un simplexe unitaire.

Afin d'obtenir un simplex centré, nous n'effectuons en fait pas la décomposition QR sur la matrice $\mathbf{L}$ directement mais sur la matrice $\mathbf{L}$ centrée par la fonction $\mathbf{H}$, c'est-à-dire la matrice $\mathbf{HL}$. On obtient donc un simplexe unitaire, centré à l'origine grâce au centrage préalable de la matrice $\mathbf{L}$. On considérera donc dans la suite du document, la matrice cible $\mathbf{K}^\star = \mathbf{Q}\mathbf{Q}^\top$.

Cette matrice possède une norme constante, ce qui sera utile dans la suite du document :

Propriété 7. *La norme de la matrice $\mathbf{K}^\star = \mathbf{Q}\mathbf{Q}^\top$ est égale à la racine carré du nombre de catégorie :*

$$\|\mathbf{K}^\star\| = \sqrt{c}. \tag{8.1}$$

Démonstration.

$$\begin{aligned}
\|\mathbf{K}^\star\|^2 &= \|\mathbf{Q}\mathbf{Q}^\top\|^2 \\
&= \mathrm{Tr}(\mathbf{Q}\mathbf{Q}^\top\mathbf{Q}\mathbf{Q}^\top) \\
&= \mathrm{Tr}(\mathbf{Q}\mathbf{Q}^\top) \\
&= \mathrm{Tr}(\mathbf{Q}^\top\mathbf{Q}) \\
&= \mathrm{Tr}(\mathrm{Id}_{c,c}) \\
&= c.
\end{aligned}$$

□

8.1.4 Problème d'optimisation et construction itérative d'un noyau

Afin de faire converger la fonction noyau à construit vers la fonction cible, nous avons besoin d'un critère permettant de calculer la distance que l'on a de notre objectif. Nous pourrons ensuite utiliser des méthodes à base d'optimisation d'une fonction de coût telles que les méthodes de Boosting. Nous avons choisi de mesurer la distance entre la fonction noyau que l'on construit à la fonction cible à l'aide d'un alignement centré (cf. section 6.2.2.3). On cherche, en conséquence, à construire une fonction noyau qui approxime au mieux une fonction noyau $k^\star$ basée sur la fonction oracle au sens du critère d'alignement centré :

$$\arg\max_k A_H(k, k^\star). \tag{8.2}$$

Ne pouvant pas évaluer directement ce critère, on se restreint à un ensemble d'exemples d'apprentissage et on cherche à construire la fonction noyau k telle que sa projection $\mathbf{K}$ sur les exemples d'apprentissage maximise l'alignement centré discret :

$$\arg\max_{k} \mathcal{A}_H(\mathbf{K}, \mathbf{K}^\star). \tag{8.3}$$

Nous proposons dans cette thèse de résoudre ce problème par un ajout incrémental de rectifications à notre matrice : on propose d'ajouter à chaque étape t une fonction noyau k_t de rang un et de la pondérer avec un facteur β_t permettant de se rapprocher de notre fonction cible. La fonction noyau $k_t : X \times X \longrightarrow \mathbb{R}$ est construite à partir d'une fonction $f_t : X \longrightarrow \mathbb{R}$:

$$\forall x_i, x_j \in X \text{ tels que } k_t(x_i, x_j) = f_t(x_i) f_t(x_j). \tag{8.4}$$

Cette fonction est bien une fonction noyau selon la propriété 5.

Notre fonction noyau se construit donc par itérations et on a :

$$\forall i, j \;\; k(x_i, x_j) = \sum_{t=0}^{T} \beta_t f_t(x_i) f_t(x_j) \tag{8.5}$$

Notre problème d'optimisation s'exprime donc ainsi :

$$\{\beta, f_t\} = \arg\max_{\beta>0, f} A_H(k_{t-1} + \beta k, k^\star) \tag{8.6}$$

De manière analogue on peut exprimer ce problème dans le cas discret. Les résultats de la fonction f_t peuvent s'exprimer à l'aide d'un vecteur $\mathbf{f}_t$ des valeurs de la fonction prise en chaque image. On définit ainsi le vecteur $\mathbf{f}_t$ comme :

$$\mathbf{f}_t = (f_t(x_i))_{i \in [\![1,n]\!]}. \tag{8.7}$$

On peut alors définir une matrice $\mathbf{F}_T$ dont chaque ligne correspond à la valeur de chaque fonction f_t pour chaque image $\mathbf{F}_T(\mathbf{x}) = \boldsymbol{\beta}^{\frac{1}{2}} \odot \Big[f_1(\mathbf{x}) \; f_2(\mathbf{x}) \ldots f_T(\mathbf{x}) \Big]$ avec $\odot$ le produit d'Hadamard. On cherche alors à construire la matrice $\mathbf{K}$:

$$\begin{aligned}
(\mathbf{K})_{i,j} &= \sum_{t=0}^{T} \beta_t k_t(\mathbf{x_i}, \mathbf{x_j}) && (8.8)\\
&= \langle \mathbf{F}_{Ti}, \mathbf{F}_{Tj} \rangle && (8.9)\\
\mathbf{K} &= \sum_{t=0}^{T} \beta_t \mathbf{k}_t && (8.10)\\
&= \sum_{t=0}^{T} \beta_t \mathbf{f}_t \mathbf{f}_t^\top && (8.11)\\
&= \mathbf{F}_T \mathbf{F}_T^\top, && (8.12)
\end{aligned}$$

à l'aide du problème d'optimisation suivant :

$$\{\beta, f_t\} = \arg\max_{\beta>0, f} \mathcal{A}_{\mathbf{H}}(\mathbf{K}_{t-1} + \beta \mathbf{f}\mathbf{f}^\top, \mathbf{K}^\star). \tag{8.13}$$

8.1.5 Espace sémantique de sélection

Nous ajoutons une fonction noyau de rang 1 à chaque itération. Cette opération est équivalente à construire un nouvel espace sémantique où est effectué le produit scalaire.

Connaissant l'espace sémantique $\mathbf{F}_t$ il est possible de calculer la matrice de Gram de la fonction noyau grâce à la formule suivante :

$$\mathbf{K}_t = \mathbf{F}_t \mathbf{F}_t^\top. \tag{8.14}$$

On remarque avec cette écriture, que le vecteur $\mathbf{F}_T(\mathbf{x})$ est un nouveau descripteur de l'image $\mathbf{x}$. La matrice noyau $\mathbf{K}_T$ correspond exactement à la matrice de Gram associée au produit scalaire dans l'espace engendré par cette nouvelle représentation des images.

Inversement connaissant les fonctions noyaux de rang 1 qui ont permis de construire notre fonction noyau finale, il est également possible de retrouver l'espace sémantique où est effectué le produit scalaire. Il suffit de calculer la racine carrée des diagonales de chacune des fonctions noyaux et de concaténer les vecteurs ainsi formés pour retrouver $\mathbf{F}_t$:

$$\mathbf{F}_t = \bigcup_{i=1}^{t} \left(diag(\sqrt{\beta_i \mathbf{k}_i}) \right)_i. \tag{8.15}$$

Les matrices $\mathbf{k}_i$ sont les matrices de chacune des fonctions noyaux :

$$\mathbf{k}_t = \begin{pmatrix} k_t(x_1, x_1) & \cdots & k_t(x_1, x_n) \\ \vdots & \ddots & \vdots \\ k_t(x_n, x_1) & \cdots & k_t(x_n, x_n) \end{pmatrix}. \tag{8.16}$$

On peut donc aussi bien travailler sur les espaces sémantiques :

$$\mathbf{F}_t = \left[\mathbf{F}_{t-1} \ \sqrt{\beta_t} \mathbf{f}_t \right], \tag{8.17}$$

que sur les matrices de Gram de la fonction noyau :

$$\mathbf{K}_t = \mathbf{K}_{t-1} + \beta_t \mathbf{k}_t. \tag{8.18}$$

La matrice associée à l'espace sémantique étant plus économique que la matrice associée à la fonction noyau, nous utilisons la première dans tout nos calculs et nous n'expliciterons jamais la matrice de la fonction noyau.

8.1.6 Une approche par Boosting

Afin d'obtenir une solution au problème de l'équation (Eq. 8.13), nous proposons d'utiliser une approche inspirée du Boosting. Cette approche consiste à sélectionner itérativement une fonction f_t (ou de manière équivalente dans ce problème, un noyau k_t) et à définir par la suite sa pondération afin de maximiser notre critère d'alignement centré.

La sélection de f_t se fait parmi un ensemble d'apprenants dits faibles et est effectuée dans une direction (que nous étudierons en détail par la suite) dépendant des performances aux itérations précédentes de la fonction noyau que l'on construit.

Chaque itération est donc décomposée en deux temps. Durant la première phase, nous construisons un ensemble $\mathcal{K}_t$ de noyaux faibles dans lequel la sélection sera effectuée. La notion de faiblesse de l'incrément est important, elle permet en effet de préserver une bonne généralisation et évite, dans les algorithmes de Boosting, le sur-apprentissage (pour plus de détail on pourra se référer aux section 4.2.3 pour la généralisation et 4.2.2 pour le sur-apprentissage).

Nous proposons d'utiliser des noyaux "faibles" de rang 1. Ces noyaux peuvent être construits à l'aide d'une fonction "faible" f_t et on aura $k_t(x,y) = f_t(x)f_t(y)$. L'objectif de la première étape de Boosting est donc de construire un ensemble $\mathcal{F}_t$ de fonctions faibles candidates. Nous montrons par la suite que ces fonctions sont construites dans la direction d'une fonction optimale $f_t^\star$, définissant une direction d'évolution intéressante de la matrice $\mathbf{K}_t$.

Grâce a l'ensemble $\mathcal{F}_t$ nous pouvons passer à la deuxième étape d'un algorithme de Boosting, à savoir le choix de la fonction la plus intéressante et la détermination de sa pondération. Pour chaque fonction cible $f \in \mathcal{F}_t$, nous souhaitons déterminer le poids $\beta(f)^\star$ qui augmenterait le plus l'alignement :

$$\beta(f)^\star = \arg\max_{\beta>0} \mathcal{A}_H(\mathbf{K}_{t-1} + \beta \mathbf{f}\mathbf{f}^\top, \mathbf{K}^\star). \tag{8.19}$$

Pour finir l'itération nous choisissons la fonction qui augmente le plus l'alignement pour sa meilleure pondération, soit le meilleur couple (fonction, pondération) qui maximise l'alignement.

Le problème mis en exergue par l'équation (Eq. 8.19) peut être résolu analytiquement au moyen du théorème suivant :

Théorème 3. *Si* $\mathbf{K}_{t-1} \neq 0$, $\mathbf{k} \neq 0$, $\mathbf{K}^\star \neq 0$, $\mathcal{A}_\mathbf{H}(\mathbf{K}_{t-1}, \mathbf{k}) \neq 1$ *et* $\mathcal{A}_\mathbf{H}(\mathbf{K}_{t-1}, \mathbf{K}^\star) \neq \mathcal{A}_\mathbf{H}(\mathbf{K}_{t-1}, \mathbf{k})\mathcal{A}_\mathbf{H}(\mathbf{K}^\star, \mathbf{k})$, *la solution de l'équation (Eq. 8.19) s'exprime sous la forme suivante :*

$$\beta^\star = \frac{\|\overline{\mathbf{K}}_{t-1}\|^2 \langle \overline{\mathbf{k}}, \overline{\mathbf{K}}^\star \rangle - \langle \overline{\mathbf{K}}_{t-1}, \overline{\mathbf{k}} \rangle \langle \overline{\mathbf{K}}_{t-1}, \overline{\mathbf{K}}^\star \rangle}{\|\overline{\mathbf{k}}\|^2 \langle \overline{\mathbf{K}}_{t-1}, \overline{\mathbf{K}}^\star \rangle - \langle \overline{\mathbf{K}}_{t-1}, \overline{\mathbf{k}} \rangle \langle \overline{\mathbf{k}}, \overline{\mathbf{K}}^\star \rangle}. \tag{8.20}$$

De plus si $\beta^\star \neq 0$ *l'alignement est augmenté :*

$$\mathcal{A}_\mathbf{H}(\mathbf{K}_{t-1} + \beta^\star \mathbf{k}, \mathbf{K}^\star) > \mathcal{A}_\mathbf{H}(\mathbf{K}_{t-1}, \mathbf{K}^\star).$$

Démonstration. On peut remarquer que l'équation (Eq. 8.19) à maximiser peut être réécrite par :

$$\mathcal{A}_\mathbf{H}(\mathbf{K}_{t-1} + \beta \mathbf{k}_t, \mathbf{K}^\star) = \frac{\mathbf{w}^\top \mathbf{b}}{(\mathbf{b}^\top \boldsymbol{\Delta} \mathbf{b})^{\frac{1}{2}} \|\mathbf{K}^\star\|} \tag{8.21}$$

où $\beta = \dfrac{\beta_{\mathbf{k}_t}}{\beta_{\mathbf{K}_{t-1}}}$, $\mathbf{b} = \begin{bmatrix} \beta_{\mathbf{K}_{t-1}} \\ \beta_{\mathbf{k}_t} \end{bmatrix}$, $\mathbf{w} = \begin{bmatrix} \langle \overline{\mathbf{K}}_{t-1}, \overline{\mathbf{K}}^\star \rangle \\ \langle \overline{\mathbf{k}}_t, \overline{\mathbf{K}}^\star \rangle \end{bmatrix}$ et $\boldsymbol{\Delta} = \begin{bmatrix} \|\overline{\mathbf{K}}_{t-1}\|^2 & \langle \overline{\mathbf{K}}_{t-1}, \overline{\mathbf{k}}_t \rangle \\ \langle \overline{\mathbf{k}}_t, \overline{\mathbf{K}}_{t-1} \rangle & \|\overline{\mathbf{k}}_t\|^2 \end{bmatrix}$.

Avec cette écriture, (Eq. 8.19) peut être résolue analytiquement à condition que $\boldsymbol{\Delta}$ soit inversible. En effet :

$$\arg\max_{\|\mathbf{b}\|=1} \frac{\mathbf{w}^\top \mathbf{b}}{(\mathbf{b}^\top \boldsymbol{\Delta} \mathbf{b})^{\frac{1}{2}} \|\mathbf{K}^\star\|} = \arg\max_{\|\mathbf{b}\|=1} \frac{(\mathbf{w}^\top \mathbf{b})^2}{\mathbf{b}^\top \boldsymbol{\Delta} \mathbf{b}} \tag{8.22}$$

$$= \arg\max_{\|\mathbf{b}\|=1} \frac{\mathbf{b}^\top \mathbf{w}\mathbf{w}^\top \mathbf{b}}{\mathbf{b}^\top \boldsymbol{\Delta} \mathbf{b}}. \tag{8.23}$$

On reconnaît un quotient de Rayleigh généralisé.

$\boldsymbol{\Delta} \geq 0$ car $\boldsymbol{\Delta}$ est une matrice de Gram. Il existe donc une décomposition telle que $\boldsymbol{\Delta} = \mathbf{U}\mathbf{D}\mathbf{U}^\top$ avec $\mathbf{U}$ la matrice des vecteurs propres et $\mathbf{D}$ la matrice diagonale des valeurs propres de $\boldsymbol{\Delta}$. Comme $\boldsymbol{\Delta} \geq 0$, les valeurs de $\mathbf{D}$ sont positives ou nulles. On peut donc décomposer $\boldsymbol{\Delta}$ en une matrice $\boldsymbol{\Delta}_{\frac{1}{2}}\boldsymbol{\Delta}_{\frac{1}{2}}^\top$ avec $\boldsymbol{\Delta}_{\frac{1}{2}} = \mathbf{U}\mathbf{D}^{\frac{1}{2}}$. En supposant $\boldsymbol{\Delta}$ inversible, $\boldsymbol{\Delta}_{\frac{1}{2}}$ est aussi inversible et l'équation 8.23 peut se réécrire comme :

$$\underset{\|\boldsymbol{\Delta}_{\frac{1}{2}}^{-1}\mathfrak{b}\|=1}{\arg\max} \frac{\mathfrak{b}^\top \boldsymbol{\Delta}_{\frac{1}{2}}^{-1}\mathbf{w}\mathbf{w}^\top \boldsymbol{\Delta}_{\frac{1}{2}}^{-\top}\mathfrak{b}}{\mathfrak{b}^\top\mathfrak{b}} \text{ avec } \mathfrak{b} = \boldsymbol{\Delta}_{\frac{1}{2}}^\top \mathbf{b}. \tag{8.24}$$

Cette équation est une forme de Rayleigh de type $\arg\max_{\|v\|=1} \frac{v^\top A v}{v^\top v}$. La solution de cette équation correspond au vecteur propre associé à la plus grande valeur propre de A. Or le seul vecteur propre, à un facteur près, de $\boldsymbol{\Delta}_{\frac{1}{2}}^{-1}\mathbf{w}\mathbf{w}^\top\boldsymbol{\Delta}_{\frac{1}{2}}^{-\top}$ est $\boldsymbol{\Delta}_{\frac{1}{2}}^{-1}\mathbf{w}$. En effet soit v un vecteur propre, on a :

$$\boldsymbol{\Delta}_{\frac{1}{2}}^{-1}\mathbf{w}\mathbf{w}^\top\boldsymbol{\Delta}_{\frac{1}{2}}^{-\top} v = \lambda v. \tag{8.25}$$

Or $\mathbf{w}$ et v étant des vecteurs, $\mathbf{w}^\top\boldsymbol{\Delta}_{\frac{1}{2}}^{-\top} v$ est un coefficient, on a donc :

$$\lambda_2 \boldsymbol{\Delta}_{\frac{1}{2}}^{-1}\mathbf{w} = \lambda v \tag{8.26}$$

$$\frac{\lambda_2}{\lambda}\boldsymbol{\Delta}_{\frac{1}{2}}^{-1}\mathbf{w} = v. \tag{8.27}$$

La solution du problème général est donc :

$$\mathbf{b}^\star = \underset{\|\mathbf{b}\|=1}{\arg\max} \frac{\mathbf{w}^\top\mathbf{b}}{(\mathbf{b}^\top\boldsymbol{\Delta}\mathbf{b})^{\frac{1}{2}}\|\mathbf{K}^\star\|} = \frac{\boldsymbol{\Delta}^{-1}\mathbf{w}}{\|\boldsymbol{\Delta}^{-1}\mathbf{w}\|}. \tag{8.28}$$

La solution de (Eq. 8.19) peut s'écrire sous la forme $\beta^\star = \frac{\beta^\star_{\mathbf{k}_t}}{\beta^\star_{\mathbf{K}_{t-1}}}$. □

Dans cette formule, rien n'interdit à $\beta^\star$ d'être négatif auquel cas la fonction qui en découle n'est pas nécessairement un noyau de Mercer. Pour cette raison, les fonctions dont le $\beta^\star$ est négatif ne seront pas sélectionnables. Elles correspondent dans la philosophie Boosting à des fonctions qui pourraient être considérées comme pires que le hasard pour traiter notre problème.

Modulo une normalisation des noyaux utilisés, on peut remarquer que la détermination du bêta optimal repose uniquement sur les alignements entre trois matrices noyaux : la matrice cible, la matrice à l'itération précédente et l'incrément que l'on souhaite ajouter.

Corollaire 1. *Si* $\mathbf{K}_{t-1} \neq 0$, $\mathbf{k} \neq 0$, $\mathbf{K}^\star \neq 0$, $\mathcal{A}_\mathbf{H}(\mathbf{K}_{t-1}, \mathbf{k}) \neq 1$ *et* $\mathcal{A}_\mathbf{H}(\mathbf{K}_{t-1}, \mathbf{K}^\star) \neq \mathcal{A}_\mathbf{H}(\mathbf{K}_{t-1}, \mathbf{k})\mathcal{A}_\mathbf{H}(\mathbf{K}^\star, \mathbf{k})$*, la solution de l'équation (Eq. 8.19) s'exprime sous la forme suivante :*

$$\beta^\star = \frac{\|\overline{\mathbf{K}}_{t-1}\|}{\|\overline{\mathbf{k}}\|}\gamma\left(\frac{1 - \frac{1}{\gamma}\mathcal{A}_H(\mathbf{K}_{t-1}, \mathbf{k})}{1 - \gamma\mathcal{A}_H(\mathbf{K}_{t-1}, \mathbf{k})}\right), \tag{8.29}$$

avec $\gamma = \dfrac{\mathcal{A}_H(\mathbf{k}, \mathbf{K}^\star)}{\mathcal{A}_H(\mathbf{K}_{t-1}, \mathbf{K}^\star)}$.

Démonstration. La formule se prouve grâce au raisonnement suivant :

$$\begin{aligned}
\beta^\star &= \frac{\|\overline{\mathbf{K}}_{t-1}\|^2\langle\overline{\mathbf{k}}, \overline{\mathbf{K}}^\star\rangle - \langle\overline{\mathbf{K}}_{t-1}, \overline{\mathbf{k}}\rangle\langle\overline{\mathbf{K}}_{t-1}, \overline{\mathbf{K}}^\star\rangle}{\|\overline{\mathbf{k}}\|^2\langle\overline{\mathbf{K}}_{t-1}, \overline{\mathbf{K}}^\star\rangle - \langle\overline{\mathbf{K}}_{t-1}, \overline{\mathbf{k}}\rangle\langle\overline{\mathbf{k}}, \overline{\mathbf{K}}^\star\rangle} \\
&= \frac{\|\overline{\mathbf{K}}_{t-1}\|^2\langle\overline{\mathbf{k}}, \overline{\mathbf{K}}^\star\rangle}{\|\overline{\mathbf{k}}\|^2\langle\overline{\mathbf{K}}_{t-1}, \overline{\mathbf{K}}^\star\rangle} \frac{1 - \dfrac{\langle\overline{\mathbf{K}}_{t-1}, \overline{\mathbf{k}}\rangle\langle\overline{\mathbf{K}}_{t-1}, \overline{\mathbf{K}}^\star\rangle}{\|\overline{\mathbf{K}}_{t-1}\|^2\langle\overline{\mathbf{k}}, \overline{\mathbf{K}}^\star\rangle}}{1 - \dfrac{\langle\overline{\mathbf{K}}_{t-1}, \overline{\mathbf{k}}\rangle\langle\overline{\mathbf{k}}, \overline{\mathbf{K}}^\star\rangle}{\|\overline{\mathbf{k}}\|^2\langle\overline{\mathbf{K}}_{t-1}, \overline{\mathbf{K}}^\star\rangle}} \\
&= \frac{\|\overline{\mathbf{K}}_{t-1}\|}{\|\overline{\mathbf{k}}\|} \frac{\|\overline{\mathbf{K}}_{t-1}\|\langle\overline{\mathbf{k}}, \overline{\mathbf{K}}^\star\rangle}{\|\overline{\mathbf{k}}\|\langle\overline{\mathbf{K}}_{t-1}, \overline{\mathbf{K}}^\star\rangle} \frac{1 - \dfrac{\langle\overline{\mathbf{K}}_{t-1}, \overline{\mathbf{K}}^\star\rangle\|\overline{\mathbf{k}}\|}{\langle\overline{\mathbf{k}}, \overline{\mathbf{K}}^\star\rangle\|\overline{\mathbf{K}}_{t-1}\|} \dfrac{\langle\overline{\mathbf{K}}_{t-1}, \overline{\mathbf{k}}\rangle}{\|\overline{\mathbf{K}}_{t-1}\|\|\overline{\mathbf{k}}\|}}{1 - \dfrac{\langle\overline{\mathbf{k}}, \overline{\mathbf{K}}^\star\rangle\|\overline{\mathbf{K}}_{t-1}\|}{\langle\overline{\mathbf{K}}_{t-1}, \overline{\mathbf{K}}^\star\rangle\|\overline{\mathbf{k}}\|} \dfrac{\langle\overline{\mathbf{K}}_{t-1}, \overline{\mathbf{k}}\rangle}{\|\overline{\mathbf{k}}\|\|\overline{\mathbf{K}}_{t-1}\|}} \\
&= \frac{\|\overline{\mathbf{K}}_{t-1}\|}{\|\overline{\mathbf{k}}\|}\gamma\left(\frac{1 - \frac{1}{\gamma}\mathcal{A}_H(\mathbf{K}_{t-1}, \mathbf{k})}{1 - \gamma\mathcal{A}_H(\mathbf{K}_{t-1}, \mathbf{k})}\right).
\end{aligned}$$

Avec $\gamma = \dfrac{\langle\overline{\mathbf{k}}, \overline{\mathbf{K}}^\star\rangle\|\overline{\mathbf{K}}_{t-1}\|}{\langle\overline{\mathbf{K}}_{t-1}, \overline{\mathbf{K}}^\star\rangle\|\overline{\mathbf{k}}\|}$. Or on a :

$$\begin{aligned}
\gamma &= \frac{\langle\overline{\mathbf{k}}, \overline{\mathbf{K}}^\star\rangle\|\overline{\mathbf{K}}_{t-1}\|}{\langle\overline{\mathbf{K}}_{t-1}, \overline{\mathbf{K}}^\star\rangle\|\overline{\mathbf{k}}\|} \\
&= \frac{\|\overline{\mathbf{K}}_{t-1}\|\|\overline{\mathbf{K}}^\star\|}{\langle\overline{\mathbf{K}}_{t-1}, \overline{\mathbf{K}}^\star\rangle} \frac{\langle\overline{\mathbf{k}}, \overline{\mathbf{K}}^\star\rangle}{\|\overline{\mathbf{k}}\|\|\overline{\mathbf{K}}^\star\|} \\
&= \frac{1}{\mathcal{A}_H(\mathbf{K}_{t-1}, \mathbf{K}^\star)}\mathcal{A}_H(\mathbf{k}, \mathbf{K}^\star)
\end{aligned}$$

□

8.2 Choix d'une direction d'évolution de la matrice noyau

8.2.1 Motivations

Dans cette partie nous allons, nous intéresser à trouver une direction de $\mathbf{k}_t$ permettant d'améliorer l'alignement centré entre la matrice que l'on construit et la matrice cible.

On cherche une direction $\mathbf{k}_t$ telle qu'il existe une pondération positive β_t telle que :

$$\mathcal{A}_H(\mathbf{K}_{t-1} + \beta_t\mathbf{k}_t, \mathbf{K}^\star) > \mathcal{A}_H(\mathbf{K}_{t-1}, \mathbf{K}^\star). \tag{8.30}$$

Dans la philosophie du Boosting, les classifieurs sélectionnables doivent être faibles. Afin de retrouver cette idée dans notre méthode nous nous restreignions à des noyaux de rang 1. Outre les avantages calculatoires que nous verrons par la suite, ce choix nous garantit d'avoir des noyaux qui restent faibles.

En effet un noyau de rang 1 n'est pas suffisant pour décrire plus de 2 classes, ce choix est donc effectivement faible dans un cas multi-classes.

Une matrice de rang 1 peut se décomposer de manière unique à l'aide d'une fonction f_t par $\mathbf{k}_t = \mathbf{f}_t\mathbf{f}_t^\top$. On recherche donc une fonction f_t qui nous permettra d'augmenter l'alignement.

Les fonctions f_t sont des fonctions qui permettent de projeter une information visuelle dans un nouvel espace. L'idée est de construire un nouvel espace par une combinaison de sous-espaces, tel que le produit scalaire dans cet espace ait un comportement très semblable à celui qu'utilise la matrice cible. On cherche donc à retrouver la notion de simplex dans l'espace que l'on construit comme pour la matrice cible (cf section. 8.1.3).

La fonction f_t correspond à la projection des exemples sur l'un des sous-espaces et on l'a construit de manière à ajouter l'information qui permet à la matrice de la concaténation des sous-espaces de se rapprocher de la matrice cible.

8.2.2 La matrice des barycentres

L'un des points essentiels de notre approche repose sur le concept de barycentre. Pour chaque classe, il est possible de déterminer la position du barycentre de la classe dans l'espace que l'on construit décrit par $\mathbf{F}_t$. Ce barycentre correspond à la projection des descripteurs sémantiques sur chacune des classes. On peut y associer la matrice suivante :

Définition 20. *On note $\mathbf{G}_t \in \mathcal{M}_{c,t}$ la matrice des barycentres des classes du nouvel espace sémantique, défini par :*

$$\mathbf{G}_t = \mathbf{Q}^\top \mathbf{F}_t. \tag{8.31}$$

Les lignes de cette matrice correspondent aux barycentres de chacune des classes dans l'espace modélisé par les descripteurs sémantiques que l'on construit ($\mathbf{F}_t$).

L'idée de notre méthode est de chercher à placer les barycentres de chacune des classes à égale distance les un des autres pour favoriser la séparabilité de chacune par un hyperplan.

On peut maintenant se demander comment se comportent nos équations en les observant sous ce nouvel angle.

Tous d'abord, si on introduit ces barycentres dans le calcul de l'alignement, on obtient la propriété suivante :

Propriété 8. *Soit une matrice de Gram $\mathbf{K} = \mathbf{F}\mathbf{F}^\top$, l'alignement avec la matrice cible $\mathbf{K}^\star = \mathbf{Q}\mathbf{Q}^\top$ peut s'écrire :*

$$\mathcal{A}_H(\mathbf{K}, \mathbf{K}^\star) = \frac{\|\mathbf{G}\|^2}{\|\overline{\mathbf{K}}\|\|\mathbf{K}^\star\|} \tag{8.32}$$

$$= \frac{1}{\sqrt{c}} \frac{\|\mathbf{G}\|^2}{\|\overline{\mathbf{K}}\|}. \tag{8.33}$$

Démonstration.

$$\begin{aligned}
\mathcal{A}_H(\mathbf{K}, \mathbf{K}^\star) &= \frac{\langle \overline{\mathbf{K}}, \mathbf{K}^\star \rangle}{\|\overline{\mathbf{K}}\| \|\mathbf{K}^\star\|} && (8.34)\\
&= \frac{\langle \overline{\mathbf{F}\mathbf{F}}^\top, \mathbf{Q}\mathbf{Q}^\top \rangle}{\|\overline{\mathbf{K}}\| \|\mathbf{K}^\star\|} && (8.35)\\
&= \frac{\mathrm{Tr}(\mathbf{H}\mathbf{F}\mathbf{F}^\top\mathbf{H}\mathbf{Q}\mathbf{Q}^\top)}{\|\overline{\mathbf{K}}\| \|\mathbf{K}^\star\|} && (8.36)\\
&= \frac{\mathrm{Tr}((\mathbf{Q}^\top\mathbf{F})(\mathbf{Q}^\top\mathbf{F})^\top)}{\|\overline{\mathbf{K}}\| \|\mathbf{K}^\star\|} && (8.37)\\
&= \frac{\mathrm{Tr}(\mathbf{G}\mathbf{G}^T)}{\|\overline{\mathbf{K}}\| \|\mathbf{K}^\star\|} && (8.38)\\
&= \frac{\|\mathbf{G}\|^2}{\|\overline{\mathbf{K}}\| \|\mathbf{K}^\star\|} && (8.39)\\
&= \frac{1}{\sqrt{c}} \frac{\|\mathbf{G}\|^2}{\|\overline{\mathbf{K}}\|}. && (8.40)
\end{aligned}$$

□

On remarque que l'alignement s'exprime uniquement à l'aide de la norme de la fonction noyau $\mathbf{K}$ que l'on construit et de la norme des barycentres $\mathbf{G}$ associés à l'espace sémantique correspondant.

Propriété 9. *A chaque itération le nouveau barycentre s'exprime récursivement à l'aide de la formulation suivante :*

$$\mathbf{G}_t = \begin{bmatrix} \mathbf{G}_{t-1} & \sqrt{\beta}\mathbf{g}_t \end{bmatrix}, \tag{8.41}$$

où g_t est le barycentre des exemples sur la classe fictive décrite par la fonction f_t que l'on introduit.

Démonstration.

$$\begin{aligned}
\mathbf{G}_t &= \mathbf{Q}^\top\mathbf{F}_t \\
&= \mathbf{Q}^\top \begin{bmatrix} \mathbf{F}_{t-1} & \sqrt{\beta}\mathbf{f}_t \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{Q}^\top\mathbf{F}_{t-1} & \sqrt{\beta}\mathbf{Q}^\top\mathbf{f}_t \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{G}_{t-1} & \sqrt{\beta}\mathbf{g}_t \end{bmatrix}.
\end{aligned}$$

□

Dans la suite du document nous allons étudier le choix des fonctions f_t présentés dans la section précédente. Pour cela nous considéreront l'évolution des barycentres de chaque classe dans l'espace engendré par la description sémantique que l'on construit. L'idée est de faire évoluer ces barycentres vers une position optimale similaire à celle des barycentres dans les espaces associables à la matrice cible (c'est à dire dans une direction où les exemples seront plus facilement séparables entre classes par un hyperplan).

8.2.3 Optimiser les barycentres, une condition de convergence

Nous allons voir dans cette partie comment l'utilisation des barycentres permet d'accroître l'alignement. A convergence, c'est à dire quand l'alignement a atteint la valeur maximale de 1, nous avons la propriété suivante :

Propriété 10. *Si l'algorithme est arrivé à convergence, les matrices* $\mathbf{K}$ *et* $\mathbf{K}^\star$ *sont proportionnelles, ce qui est équivalent à* $\mathcal{A}_H(\mathbf{K}, \mathbf{K}^\star) = 1$.

Démonstration. L'équivalence $\exists \lambda > 0 \ \mathbf{K} = \lambda \mathbf{K}^\star \iff \mathcal{A}_H(\mathbf{K}, \mathbf{K}^\star) = 1$ est une conséquence directe du théorème de Cauchy-Schwarz.

En effet le cas d'égalité de ce théorème stipule que : $\exists \lambda > 0$ tel que $a = \lambda b \iff \langle a, b \rangle = \|a\|\|b\|$.

La proportionnalité des matrices $\mathbf{K}$ et $\mathbf{K}^\star$ correspond à l'état de convergence souhaité, on cherche à approximer le plus exactement possible la matrice associée à l'oracle à un facteur de proportionnalité près. □

Intéressons nous maintenant aux propriétés de $\mathbf{G}$ à l'état de convergence.

La matrice cible est orthonormale (ce qui découle du choix de la décomposition QR). La matrice $\mathbf{Q}$ peut être vue également comme une matrice de barycentres. Les c colonnes de $\mathbf{Q}$ représentent la position optimale des barycentres des classes dans un certain espace. Ces barycentres sont orthonormaux dans l'espace cible, on s'attend donc à construire un nouveau descripteur qui préserverait l'orthonormalité des barycentres entre eux dans la nouvelle matrice $\mathbf{K}$.

Ce que confirme la proposition suivante :

Lemme 1. *A convergence, la matrice de Gram des barycentres est proportionnelle à la matrice identité :*

$$\mathcal{A}_H(\mathbf{K}, \mathbf{K}^\star) = 1 \Longrightarrow \exists a > 0, \mathbf{G}\mathbf{G}^\top = a Id_{c,c}. \tag{8.42}$$

Démonstration.

$$\begin{aligned} \mathcal{A}_H(\mathbf{K}, \mathbf{K}^\star) = 1 \quad &\Longrightarrow \quad \exists a > 0, \mathbf{H}\mathbf{F}\mathbf{F}^\top\mathbf{H} = a\mathbf{Q}\mathbf{Q}^\top \text{ (cf. théorème 31)} \\ &\Longrightarrow \quad \exists a > 0, \mathbf{Q}^\top\mathbf{F}\mathbf{F}^\top\mathbf{Q} = a\mathbf{Q}^\top\mathbf{Q}\mathbf{Q}^\top\mathbf{Q} \\ &\Longrightarrow \quad \exists a > 0, \mathbf{G}\mathbf{G}^\top = a\mathrm{Id}_{c,c}. \end{aligned}$$

□

Ce qui veux bien dire que les produits scalaires entre barycentres sont nuls si les barycentres sont distincts, tandis que la norme d'un barycentre est unitaire.

Il en découle une propriété importante sur laquelle nous nous appuierons par la suite :

Propriété 11. *A convergence la matrice de Gram des barycentres est proportionnelle à la matrice identité et le terme de proportionnalité est le rapport de norme* $\dfrac{\|\mathbf{K}^\star\|}{\|\mathbf{K}\|}$.

$$\mathcal{A}_H(\mathbf{K}, \mathbf{K}^\star) = 1 \Longrightarrow Id_{c,c} = \frac{\|\mathbf{K}^\star\|}{\|\mathbf{K}\|}\mathbf{G}\mathbf{G}^\top \tag{8.43}$$

Démonstration.

$$\begin{aligned}
\mathcal{A}_H(\mathbf{K},\mathbf{K}^\star)=1 \Longrightarrow 1 &= \frac{\|\mathbf{G}\|^2}{\|\overline{\mathbf{K}}\|\|\mathbf{K}^\star|} \text{ (cf. propriété 8)}\\
1 &= \frac{\mathrm{Tr}(\mathbf{G}\mathbf{G}^\top)}{\|\overline{\mathbf{K}}\|\|\mathbf{K}^\star\|}\\
1 &= \frac{\mathrm{Tr}(a\mathrm{Id}_{c,c})}{\|\overline{\mathbf{K}}\|\sqrt{\mathrm{Tr}(\mathrm{Id}_{c,c})}} \text{ (cf. propriété 1)}\\
1 &= \frac{a\sqrt{\mathrm{Tr}(\mathrm{Id}_{c,c})}}{\|\overline{\mathbf{K}}\|}\\
\Longrightarrow a &= \frac{\|\overline{\mathbf{K}}\|}{\sqrt{c}} = \frac{\|\overline{\mathbf{K}}\|}{\|\mathbf{K}^\star\|}
\end{aligned}$$

□

Ces propriétés nous montrent, qu'à un facteur de proportionnalité près, la matrice de Gram des barycentres doit tendre vers la matrice identité, ce qui est nécessaire à convergence de l'algorithme. On va donc s'intéresser plus en détail à cette matrice de Gram.

Définition 21. *Soit $\varphi(\mathbf{G}_t)$ la fonction :*

$$\varphi(\mathbf{G}) = \frac{\|\overline{\mathbf{K}^\star}\|}{\|\overline{\mathbf{K}}\|}\arg\min_\lambda \left\{\lambda \in \mathrm{spec}\left(\mathbf{G}\mathbf{G}^\top\right)\right\}. \tag{8.44}$$

De cette définition il découle le théorème important suivant :

Théorème 4. *L'alignement centré est minoré par la fonction $\varphi(\mathbf{G})$, avec égalité à la convergence :*

$$0 \leq \varphi(\mathbf{G}) \leq \mathcal{A}_{\mathbf{H}}(\mathbf{K},\mathbf{K}^\star). \tag{8.45}$$

Démonstration. Soit $\lambda = \arg\min_\lambda \left\{\lambda \in \mathrm{spec}\left(\mathbf{G}_t\mathbf{G}_t^\top\right)\right\}$ on a :

$$\begin{aligned}
& c\lambda && \leq && \textstyle\sum_{\lambda_i\in\mathrm{spec}(\mathbf{G}\mathbf{G}^\top)}\lambda_i\\
\Leftrightarrow\ & \frac{\sqrt{c}}{\|\overline{\mathbf{K}}\|}\lambda && \leq && \frac{\sum_{\lambda_i\in\mathrm{spec}(\mathbf{G}\mathbf{G}^\top)}\lambda_i}{\sqrt{c}\|\overline{\mathbf{K}}\|}\\
\Leftrightarrow\ & \frac{\|\overline{\mathbf{K}}^\star\|}{\|\overline{\mathbf{K}}\|}\lambda && \leq && \frac{\|\mathbf{G}\|^2}{\|\overline{\mathbf{K}}^\star\|\|\overline{\mathbf{K}}\|}\\
\Leftrightarrow\ & \varphi(\mathbf{G}) && \leq && \mathcal{A}_{\mathbf{H}}(\mathbf{K},\mathbf{K}^\star)
\end{aligned}$$

□

On obtient donc une relation directe entre la matrice des barycentres et la maximisation de l'alignement. Notre algorithme utilisera cet axe d'approche et aura pour objectif de faire croître la fonction $\varphi(\mathbf{G})$.

Nous allons donc étudier dans la suite la résolution itérative du problème d'optimisation global suivant :

$$\arg\max_{\mathbf{K}} \varphi(\mathbf{G}) = \arg\max_{\mathbf{K},\text{ s.t. } \|\mathbf{K}\|=1} \left(\arg\min_{\lambda} \left\{ \lambda \in \text{spec}\left(\mathbf{G}\mathbf{G}^\top\right) \right\} \right). \tag{8.46}$$

La stratégie que nous employons dans notre méthode consiste à déplacer une des directions des barycentres vers l'identité. On s'intéresse dans un premier temps à corriger la direction de plus forte erreur, c'est à dire la direction où le barycentre est le plus loin de l'identité. En d'autres termes la direction associée à la plus forte valeur propre de la matrice $\text{Id}_{c,c} - \frac{\|\mathbf{K}^\star\|}{\|\mathbf{K}_{t-1}\|}\mathbf{G}_{t-1}\mathbf{G}_{t-1}^\top$. Cette direction correspond à :

$$\mathbf{g}_t^\star = \sqrt{1-\varphi(\mathbf{G}_{t-1})}\mathbf{v}, \tag{8.47}$$

avec $\mathbf{v}$ le vecteur propre de $\mathbf{G}_{t-1}\mathbf{G}_{t-1}^\top$ associé à la valeur propre $\varphi(\mathbf{G}_{t-1})$.

On cherche alors la fonction $f_t^\star$ qui permettra d'obtenir le barycentre proposé. Pour cela on construit la fonction qui minimise la distance euclidienne avec notre barycentre cible :

$$\mathbf{f}_t^\star = \arg\min_{\mathbf{f}} \|\mathbf{Q}^\top \mathbf{f} - \mathbf{g}_t^\star\|^2. \tag{8.48}$$

Les solutions de ce problème d'optimisation sont données par la propriété suivante :

Propriété 12. *Les solutions du problème* (8.48) *sont de la forme :*

$$\begin{aligned} \mathbf{f_t}^\star &= \mathbf{Q}\mathbf{g}_t^\star + (Id_n - \mathbf{Q}\mathbf{Q}^\top)a & (8.49)\\ &= \mathbf{Q}(\mathbf{g}_t^\star - \mathbf{g}_a) + a, & (8.50) \end{aligned}$$

avec a un vecteur quelconque de $\mathbb{R}^n$ et $\mathbf{g}_a = \mathbf{Q}^\top a$.

Démonstration. Les solutions du problème (8.48) sont de la forme :

$$\mathbf{f}_t^\star = \mathbf{A}^+\mathbf{g}_t^\star + (\text{Id}_n - \mathbf{A}^+\mathbf{A})a, \tag{8.51}$$

avec a un vecteur quelconque de $\mathbb{R}^n$, $\mathbf{A} = \mathbf{Q}^\top$ et $\mathbf{A}^+$ la pseudo-inverse de $\mathbf{A}$:

$$\begin{aligned} \mathbf{A}^+ &= \mathbf{A}^\top(\mathbf{A}\mathbf{A}^\top)^{-1} & (8.52)\\ &= (\mathbf{Q}^\top)^\top(\mathbf{Q}^\top(\mathbf{Q}^\top)^\top)^{-1} & (8.53)\\ &= \mathbf{Q}(\text{Id})^{-1} & (8.54)\\ &= \mathbf{Q}. & (8.55) \end{aligned}$$

□

Parmi toutes les solutions possibles, nous avons choisi celle qui a le coût calculatoire le moins important. Nous prenons donc dans la suite :

$$\mathbf{f_t}^\star = \mathbf{Q}\mathbf{g}_t^\star. \tag{8.56}$$

Il nous reste maintenant à montrer que la solution $\mathbf{f}^\star$ proposée nous permet bien d'augmenter l'alignement. Cette propriété est montrée par le théorème suivant :

Théorème 5. *Si* $\mathcal{A}_{\mathbf{H}}(\mathbf{K}_{t-1}, \mathbf{K}^\star) < 1$ *alors*

$$\beta_t^\star = \frac{\|\overline{\mathbf{K}}_{t-1}\|^2}{(1-\varphi(\mathbf{G}_{t-1}))} \frac{\left(1-\varphi(\mathbf{G}_{t-1})\frac{\|\mathbf{G}_{t-1}\|^2}{\|\overline{\mathbf{K}}_{t-1}\|\|\overline{\mathbf{K}}^\star\|}\right)}{\left(\|\mathbf{G}_{t-1}\|^2-\varphi(\mathbf{G}_{t-1})\frac{\|\overline{\mathbf{K}}_{t-1}\|}{\|\overline{\mathbf{K}}^\star\|}\right)}, \tag{8.57}$$

$\beta_t^\star > 0$ *et* $\mathcal{A}_{\mathbf{H}}(\mathbf{K}_{t-1} + \beta_t^\star \mathbf{f}_t^\star \mathbf{f}_t^{\star\top}, \mathbf{K}^\star) > \mathcal{A}_{\mathbf{H}}(\mathbf{K}_{t-1}, \mathbf{K}^\star)$.

8.2.4 Démonstration du Théorème 5

Avant de démontrer le Théorème 5, nous allons montrer quelques propriétés préliminaires. Nous allons notamment calculer les normes et produits scalaires nécessaires à l'utilisation du corollaire 1.

Tous d'abord la norme de la fonction noyau que l'on ajoute est donnée par la propriété suivante :

Propriété 13. *La norme d'une fonction noyau de la forme* $\mathbf{k} = \mathbf{f}\mathbf{f}^\top$*, avec* $\mathbf{f}$ *un vecteur de* $\mathbb{R}^n$ *est égale à*

$$\|\mathbf{k}\| = \|f\|^2. \tag{8.58}$$

Démonstration.

$$\begin{aligned}
\|\mathbf{k}\|^2 &= \mathrm{Tr}(\mathbf{k}\mathbf{k}^\top) && (8.59)\\
&= \mathrm{Tr}(\mathbf{f}\mathbf{f}^\top\mathbf{f}\mathbf{f}^\top) && (8.60)\\
&= \mathrm{Tr}(\mathbf{f}^\top\mathbf{f}\mathbf{f}^\top\mathbf{f}) && (8.61)\\
&= (\mathbf{f}^\top\mathbf{f})^2 && (8.62)\\
&= \|\mathbf{f}\|^4 && (8.63)
\end{aligned}$$

□

Pour calculer la norme de la fonction $\mathbf{f}_t^\star$, on peut utiliser les propriétés suivantes :

Propriété 14. *La norme de* $\mathbf{f_t}^\star$ *issue de la propriété 8.50 se déduit de la formule suivante :*

$$\begin{aligned}
\|\mathbf{f_t}^\star\|^2 &= \|\mathbf{g}_t^\star\|^2 + \|(Id_n - \mathbf{Q}\mathbf{Q}^\top)a\|^2 && (8.64)\\
&= \|\mathbf{g}_t^\star\|^2 - \|\mathbf{g}_a\|^2 + \|a\|^2, && (8.65)
\end{aligned}$$

avec $\mathbf{g}_a = \mathbf{Q}^\top a$.

Démonstration. La première formule se calcule grâce au raisonnement suivant :

$$\begin{aligned}
\|\mathbf{Q}\mathbf{g}^\star + (\mathrm{Id}_n - \mathbf{Q}\mathbf{Q}^\top)a\|^2 &= \langle \mathbf{Q}\mathbf{g}^\star + (\mathrm{Id}_n - \mathbf{Q}\mathbf{Q}^\top)a, \mathbf{Q}\mathbf{g}^\star + (\mathrm{Id}_n - \mathbf{Q}\mathbf{Q}^\top)a\rangle\\
&= \|\mathbf{Q}\mathbf{g}^\star\|^2 + \|(\mathrm{Id}_n - \mathbf{Q}\mathbf{Q}^\top)a\|^2 + 2\langle \mathbf{Q}\mathbf{g}^\star, (\mathrm{Id}_n - \mathbf{Q}\mathbf{Q}^\top)a\rangle\\
&= \mathrm{Tr}(\mathbf{Q}\mathbf{g}^\star\mathbf{g}^{\star\top}\mathbf{Q}^\top) + \|(\mathrm{Id}_n - \mathbf{Q}\mathbf{Q}^\top)a\|^2 + 2\,\mathrm{Tr}(\mathbf{g}^{\star\top}\mathbf{Q}^\top(\mathrm{Id}_n - \mathbf{Q}\mathbf{Q}^\top)a)\\
&= \mathrm{Tr}(\mathbf{Q}^\top\mathbf{Q}\mathbf{g}^\star\mathbf{g}^{\star\top}) + \|(\mathrm{Id}_n - \mathbf{Q}\mathbf{Q}^\top)a\|^2 + 2\,\mathrm{Tr}((\mathbf{g}^{\star\top}\mathbf{Q}^\top - \mathbf{g}^{\star\top}\mathbf{Q}^\top\mathbf{Q}\mathbf{Q}^\top)a)\\
&= \mathrm{Tr}(\mathbf{g}^\star\mathbf{g}^{\star\top}) + \|(\mathrm{Id}_n - \mathbf{Q}\mathbf{Q}^\top)a\|^2 + 2\,\mathrm{Tr}((\mathbf{g}^{\star\top}\mathbf{Q}^\top - \mathbf{g}^{\star\top}\mathbf{Q}^\top)a)\\
&= \|\mathbf{g}^\star\|^2 + \|(\mathrm{Id}_n - \mathbf{Q}\mathbf{Q}^\top)a\|^2.
\end{aligned}$$

La deuxième est la conséquence de la première formule et des calculs suivants :

$$\begin{aligned}\|(\mathrm{Id}_n - \mathbf{Q}\mathbf{Q}^\top)a\|^2 &= \|a\|^2 + \|\mathbf{Q}\mathbf{Q}^\top a\|^2 - 2\langle a, \mathbf{Q}\mathbf{Q}^\top a\rangle \\ &= \|a\|^2 + \mathrm{Tr}(\mathbf{Q}\mathbf{Q}^\top aa^\top \mathbf{Q}\mathbf{Q}^\top) - 2\,\mathrm{Tr}(aa^\top \mathbf{Q}\mathbf{Q}^\top) \\ &= \|a\|^2 - \mathrm{Tr}(\mathbf{Q}^\top aa^\top \mathbf{Q}) \\ &= \|a\|^2 - \|\mathbf{Q}^\top a\|^2.\end{aligned}$$

□

On en déduit alors la norme de la fonction $\mathbf{f_t}^\star$ que l'on a choisi pour notre algorithme :

Propriété 15. *La solution $\mathbf{f_t}^\star = \mathbf{Q}\mathbf{g}_t^\star$ du problème (8.48) est l'une des solutions de norme minimale. La norme de $\mathbf{f_t}^\star$ est alors la même que la norme de $\mathbf{g}^\star$:*

$$\|\mathbf{f_t}^\star\| = \|\mathbf{g}_t^\star\|. \tag{8.66}$$

Démonstration. C'est une conséquence directe de la propriété précédente prise en $a = 0$. □

En utilisant la définition de la fonction $\mathbf{g}_t^\star$, il est alors possible de donner une expression analytique de sa norme :

Propriété 16. *La norme des barycentres du nouvel axe est égale à :*

$$\|\mathbf{g}_t^\star\|^2 = 1 - \varphi(\mathbf{G}_t). \tag{8.67}$$

Démonstration.

$$\begin{aligned}\|\mathbf{g}_t^\star\|^2 &= \mathrm{Tr}(\mathbf{g}_t^\star {\mathbf{g}_t^\star}^\top) && (8.68)\\ &= \mathrm{Tr}((1-\varphi(\mathbf{G}_t))\mathbf{Q}vv^\top\mathbf{Q}^\top) && (8.69)\\ &= (1-\varphi(\mathbf{G}_t))\,\mathrm{Tr}(vv^\top) && (8.70)\\ &= (1-\varphi(\mathbf{G}_t))\|v\|^2 && (8.71)\\ &= 1-\varphi(\mathbf{G}_t). && (8.72)\end{aligned}$$

□

Pour calculer la valeur du bêta optimal $\beta^\star$, nous avons également besoin de calculer les valeurs des produits scalaires avec le noyau des itérations précédentes et avec le noyau cible. Les valeurs de ces produits scalaires sont données par les deux propriétés suivantes :

Propriété 17. *La valeur du produit scalaire de la nouvelle fonction optimale avec la fonction noyau issue des itérations précédentes de notre processus est donnée par l'équation suivante :*

$$\langle \mathbf{k}_t, \overline{\mathbf{K}_{t-1}}\rangle = (1-\varphi(\mathbf{G}_{t-1}))\frac{\|\overline{\mathbf{K}_{t-1}}\|}{\|\overline{\mathbf{K}^\star}\|}\varphi(\mathbf{G}_{t-1}). \tag{8.73}$$

Démonstration.

$$\begin{aligned}
\langle \mathbf{k}_t, \overline{\mathbf{K}_{t-1}} \rangle &= \langle (1-\varphi(\mathbf{G}_{t-1}))\mathbf{Q}\mathbf{v}\mathbf{v}^\top\mathbf{Q}^\top, \overline{\mathbf{F}_{t-1}\mathbf{F}_{t-1}^\top} \rangle && (8.74)\\
&= (1-\varphi(\mathbf{G}_{t-1}))\,\mathrm{Tr}(\mathbf{v}\mathbf{v}^\top\mathbf{Q}^\top\mathbf{F}_{t-1}\mathbf{F}_{t-1}^\top\mathbf{Q}) && (8.75)\\
&= (1-\varphi(\mathbf{G}_{t-1}))\,\mathrm{Tr}(\mathbf{v}^\top\mathbf{G}_{t-1}\mathbf{G}_{t-1}^\top\mathbf{v}) && (8.76)\\
&= (1-\varphi(\mathbf{G}_{t-1}))\,\mathrm{Tr}\left(\mathbf{v}^\top\left(\frac{\|\overline{\mathbf{K}_{t-1}}\|}{\|\overline{\mathbf{K}^\star}\|}\varphi(\mathbf{G}_{t-1})\right)\mathbf{v}\right) && (8.77)\\
&= (1-\varphi(\mathbf{G}_{t-1}))\frac{\|\overline{\mathbf{K}_{t-1}}\|}{\|\overline{\mathbf{K}^\star}\|}\varphi(\mathbf{G}_{t-1}), && (8.78)
\end{aligned}$$

□

Propriété 18. *La valeur du produit scalaire de la nouvelle fonction optimale avec la fonction noyau cible est donnée par l'équation suivante :*

$$\langle \mathbf{k}_t, \mathbf{K}^\star \rangle = 1 - \varphi(\mathbf{G}_{t-1}). \tag{8.79}$$

Démonstration.

$$\begin{aligned}
\langle \mathbf{k}, \overline{\mathbf{K}^\star} \rangle &= \mathrm{Tr}(\mathbf{f}\mathbf{f}^\top\mathbf{Q}\mathbf{Q}^\top) && (8.80)\\
&= \mathrm{Tr}((\mathbf{Q}^\top\mathbf{f})^\top\mathbf{Q}^\top\mathbf{f}) && (8.81)\\
&= \|\mathbf{Q}^\top\mathbf{f}\|^2 && (8.82)\\
&= \|\mathbf{g}\|^2 && (8.83)\\
&= 1-\varphi(\mathbf{G}_{t-1}). && (8.84)
\end{aligned}$$

□

Des propriétés précédentes on en déduit la pondération optimale de la nouvelle fonction noyau à l'aide du corollaire 1 :

Propriété 19. *La pondération optimale de la fonction noyau issue de la fonction* $\mathbf{f}_t^\star = \mathbf{Q}\mathbf{g}_t^\star$ *est égale à :*

$$\beta_t^\star = \frac{\|\overline{\mathbf{K}}_{t-1}\|^2}{1-\varphi(\mathbf{G}_{t-1})}\frac{1-\varphi(\mathbf{G}_{t-1})\mathcal{A}_H(\mathbf{K}_{t-1},\mathbf{K}^\star)}{\|\mathbf{G}_{t-1}\|^2-\varphi(\mathbf{G}_{t-1})\frac{\|\overline{\mathbf{K}}_{t-1}\|}{\|\overline{\mathbf{K}^\star}\|}}. \tag{8.85}$$

Démonstration.

$$\beta^\star = \frac{\|\overline{\mathbf{K}}_{t-1}\|^2\langle\overline{\mathbf{k}},\overline{\mathbf{K}}^\star\rangle - \langle\overline{\mathbf{K}}_{t-1},\overline{\mathbf{k}}\rangle\langle\overline{\mathbf{K}}_{t-1},\overline{\mathbf{K}}^\star\rangle}{\|\overline{\mathbf{k}}\|^2\langle\overline{\mathbf{K}}_{t-1},\overline{\mathbf{K}}^\star\rangle - \langle\overline{\mathbf{K}}_{t-1},\overline{\mathbf{k}}\rangle\langle\overline{\mathbf{k}},\overline{\mathbf{K}}^\star\rangle} \tag{8.86}$$

$$= \frac{\|\overline{\mathbf{K}}_{t-1}\|^2(1-\varphi(\mathbf{G}_{t-1})) - (1-\varphi(\mathbf{G}_{t-1}))\frac{\|\overline{\mathbf{K}_{t-1}}\|}{\|\overline{\mathbf{K}^\star}\|}\varphi(\mathbf{G}_{t-1})\|\mathbf{G}_{t-1}\|^2}{(1-\varphi(\mathbf{G}_{t-1}))^2\|\mathbf{G}_{t-1}\|^2 - (1-\varphi(\mathbf{G}_{t-1}))(1-\varphi(\mathbf{G}_{t-1}))\frac{\|\overline{\mathbf{K}_{t-1}}\|}{\|\overline{\mathbf{K}^\star}\|}\varphi(\mathbf{G}_{t-1})} \tag{8.87}$$

$$= \frac{\|\overline{\mathbf{K}}_{t-1}\|^2}{1-\varphi(\mathbf{G}_{t-1})}\frac{1-\frac{\varphi(\mathbf{G}_{t-1})\|\mathbf{G}_{t-1}\|^2}{\|\overline{\mathbf{K}^\star}\|\|\overline{\mathbf{K}}_t\|}}{\|\mathbf{G}_{t-1}\|^2 - \frac{\|\overline{\mathbf{K}_{t-1}}\|}{\|\overline{\mathbf{K}^\star}\|}\varphi(\mathbf{G}_{t-1})} \tag{8.88}$$

$$= \frac{\|\overline{\mathbf{K}}_{t-1}\|^2}{1-\varphi(\mathbf{G}_{t-1})}\frac{1-\varphi(\mathbf{G}_{t-1})\mathcal{A}_H(\mathbf{K}_{t-1},\mathbf{K}^\star)}{\|\mathbf{G}_{t-1}\|^2 - \varphi(\mathbf{G}_{t-1})\frac{\|\overline{\mathbf{K}}_{t-1}\|}{\|\overline{\mathbf{K}^\star}\|}}. \tag{8.89}$$

□

Il nous reste plus qu'a vérifier que la valeur de $\beta^\star$ est bien toujours positive. Dans ce cas l'alignement est strictement croissant d'une itération à une autre dans le cas optimal.

Propriété 20. *Pour $\mathbf{f}_t^\star = \mathbf{Q}\mathbf{g}_t^\star$, la pondération associée est strictement positive si l'algorithme n'a pas convergé et si $\mathbf{K}_{t-1} \neq 0$.*

Démonstration. D'après la propriété précédente la pondération optimale s'exprime par :

$$\beta_t^\star = \frac{\|\overline{\mathbf{K}}_{t-1}\|^2}{1-\varphi(\mathbf{G}_{t-1})}\frac{1-\varphi(\mathbf{G}_{t-1})\mathcal{A}_H(\mathbf{K}_{t-1},\mathbf{K}^\star)}{\|\mathbf{G}_{t-1}\|^2 - \varphi(\mathbf{G}_{t-1})\frac{\|\overline{\mathbf{K}}_{t-1}\|}{\|\overline{\mathbf{K}^\star}\|}}. \tag{8.90}$$

Le terme $\frac{\|\overline{\mathbf{K}}_{t-1}\|^2}{1-\varphi(\mathbf{G}_{t-1})}$ est positif car $\|\overline{\mathbf{K}}_{t-1}\|^2$ est le carré d'une norme et $\varphi(\mathbf{G}_{t-1}) < 1$ (cf.Théorème 4).

$\mathcal{A}_H(\mathbf{K}_{t-1},\mathbf{K}^\star) < 1$ et $\varphi(\mathbf{G}_{t-1}) < 1$ donc $1-\varphi(\mathbf{G}_{t-1})\mathcal{A}_H(\mathbf{K}_{t-1},\mathbf{K}^\star) > 0$.

Enfin $\|\mathbf{G}_{t-1}\|^2$ est la somme des valeurs propres de $\mathbf{G}_{t-1}\mathbf{G}_{t-1}^\top$, comme toute les valeurs propres sont positives ou nulles, cette valeur est plus grande strictement à la plus petite valeur propre $\varphi(\mathbf{G}_{t-1})\frac{\|\overline{\mathbf{K}}_{t-1}\|}{\|\overline{\mathbf{K}^\star}\|}$ car $\mathbf{G}_{t-1}\mathbf{G}_{t-1}^\top$ est non nulle. On a donc $\|\mathbf{G}_{t-1}\|^2 - \varphi(\mathbf{G}_{t-1})\frac{\|\overline{\mathbf{K}}_{t-1}\|}{\|\overline{\mathbf{K}^\star}\|} > 0$.

□

8.2.5 Définition d'un apprenant cible

Nous avons défini dans la partie précédente une fonction optimale $\mathbf{f}_t^\star$ que l'on cherche à atteindre à chaque itération t de notre algorithme. Nous construisons les fonctions sélectionnables comme étant des projections des descripteurs visuels sur un hyperplan minimisant la distance à la fonction cible par

Algorithm 15: Initialisation de l'algorithme proposé

Input: Des espaces de descripteurs bas-niveau $\mathcal{X}_s$
Input: Une matrice $\mathbf{Q}$ orthogonale de labels de taille $n \times c$.
begin
  for $t = 1, \cdots, c$ **do**
    Calcul d'une fonction cible $\mathbf{f}_t^\star = \mathbf{Q}(:, t)$.
    $\mathcal{A}_{max} = 0$.
    foreach *descripteurs bas-niveau de l'espace* $\mathcal{X}_s$ **do**
      Calcul d'une fonction faible f_s à partir de l'ensemble d'apprentissage. $\mathbf{X} \subset \mathcal{X}_s$
      if $\mathcal{A}_{\mathbf{H}}(\mathbf{f}_s\mathbf{f}_s^\top, \mathbf{K}^\star) > \mathcal{A}_{max}$ **then**
        $\mathcal{A}_{max} = \mathcal{A}_{\mathbf{H}}(\mathbf{f}_s\mathbf{f}_s^\top, \mathbf{K}^\star)$.
        Mise à jour $\mathbf{F}_t = \left[\mathbf{F}_{t-1}\ \mathbf{f}_s\right]$.

Output: Des fonctions faibles $\left[f_1(\mathbf{x})\ f_2(\mathbf{x}) \dots f_c(\mathbf{x})\right]$.

une méthode de moindres carrés (Least Mean Squares, LMS). On cherche donc pour un ensemble de descripteurs visuels $\mathbf{X}$ la fonction f solution du problème d'optimisation suivant :

$$f = \underset{\mathbf{w} \in \mathbb{R}^d}{\arg\min} \|\mathbf{X}\mathbf{w} - \mathbf{f}^\star\|^2 \ s.t.\ \|\mathbf{w}\|^2 = 1. \tag{8.91}$$

8.3 Algorithme

8.3.1 Initialisation

Au début de l'algorithme, aucune classe n'a encore été apprise, nous initialisons la méthode en construisant au moins une fois une fonction pour chaque classe à apprendre.

C'est pourquoi les c premières fonctions cible $\mathbf{f}_t^\star$ correspondent aux colonnes de la matrice $\mathbf{Q}$, c'est à dire :

$$\forall t \in [1, c], \mathbf{f}_t^\star = \mathbf{Q}(:, t). \tag{8.92}$$

Un ensemble de $\mathcal{F}_t$ de fonctions "faibles" f_t est construit en utilisant les apprenants faibles, comme les LMS présentés en (Eq. 8.91). Ces fonctions peuvent reposer sur différents types de descripteurs bas-niveau (couleurs, textures, points d'intérêt, ...) mais aussi sur plusieurs jeux de paramètres (type des descripteurs, métrique associée, ...). Dans nos expérimentations, ces ensembles peuvent avoir une taille variant de 100 à 1000 fonctions.

La fonction qui sera conservée dans le noyau final correspond à celle qui maximise l'alignement :

$$f_t = \underset{f \in \mathcal{F}_t}{\arg\max}\, \mathcal{A}_{\mathbf{H}}(\mathbf{f}\mathbf{f}^\top, \mathbf{K}^\star). \tag{8.93}$$

Le résultat de l'initialisation est donc un vecteur de c fonctions faibles : $F_c = \left[f_1\ f_2\ \cdots f_c\right]$ avec une pondération de 1 pour chaque fonction.

L'algorithme d'initialisation est donné dans Algorithme. 15.

8.3.2 Itération de la méthode de Boosting

A chaque itération de notre algorithme, on calcule la fonction que l'on cherche à atteindre :

$$\mathbf{f}^\star = \mathbf{Q}\mathbf{g}^\star = \sqrt{1 - \lambda \frac{\sqrt{c}}{\|\overline{\mathbf{K}}_{t-1}\|}}\, \mathbf{Q}\mathbf{v}. \tag{8.94}$$

On utilise ensuite chaque descripteur visuel à notre disposition pour construire un hyperplan qui se rapproche le plus possible de cette fonction cible. Parmi toute ces fonctions, on sélectionne celle qui avec la pondération optimale, augmente le plus l'alignement avec notre matrice cible.

On ajoute alors cette fonction et sa pondération comme une nouvelle colonne à notre descripteur sémantique et on réitère le processus.

Cet algorithme est détaillé dans Algorithme. 16.

8.3.3 Version optimisé de l'algorithme et calcul de complexité

L'algorithme présenté dans Algorithm. 16 peut être optimisé. Nous allons présenter dans cette partie toutes les astuces de calcul permettant de réduire la complexité de l'algorithme. Nous montrons que nous pouvons ramener la complexité de l'algorithme à une complexité de la forme $O(nt)$ avec n le nombre d'images d'apprentissage et t le nombre d'itérations de la méthode.

L'algorithme débute par le calcul de la décomposition en valeurs et vecteurs propres de la matrice de taille $c \times c$ (avec c le nombre de classes) $\mathbf{G}_t\mathbf{G}_t^\top$. Cette opération a une complexité en $O(c^3)$ pour faire la décomposition complète, par contre si on ne considère que les premières valeurs la complexité n'est plus que de $O(c^2)$.

La matrice $\mathbf{G}_t\mathbf{G}_t^\top$ peut être calculée itérativement grâce à la formule suivante :

Propriété 21. *A chaque itération on peut déduire les valeurs de la matrice $\mathbf{G}_t\mathbf{G}_t^\top$ par le processus suivant :*

$$\mathbf{G}_{t+1}\mathbf{G}_{t+1} = \mathbf{G}_t\mathbf{G}_t + \beta_t^\star \mathbf{g}_t\mathbf{g}_t^\top. \tag{8.95}$$

Démonstration.

$$\begin{aligned}
\mathbf{G}_{t+1}\mathbf{G}_{t+1}^\top &= \left(\mathbf{Q}^\top\mathbf{F}_{t+1}\right)\left(\mathbf{Q}^\top\mathbf{F}_{t+1}\right)^\top && (8.96)\\
&= \left(\mathbf{Q}^\top\left[\mathbf{F}_t \;\; \sqrt{\beta_t^\star}\mathbf{f}_t\right]\right)\left(\mathbf{Q}^\top\left[\mathbf{F}_t \;\; \sqrt{\beta_t^\star}\mathbf{f}_t\right]\right)^\top && (8.97)\\
&= \left[\mathbf{Q}^\top\mathbf{F}_t \;\; \sqrt{\beta_t^\star}\mathbf{Q}^\top\mathbf{f}_t\right]\left[\mathbf{Q}^\top\mathbf{F}_t \;\; \sqrt{\beta_t^\star}\mathbf{Q}^\top\mathbf{f}_t\right]^\top && (8.98)\\
&= \left[\mathbf{G}_t \;\; \sqrt{\beta_t^\star}\mathbf{g}_t\right]\left[\mathbf{G}_t \;\; \sqrt{\beta_t^\star}\mathbf{g}_t\right]^\top && (8.99)\\
&= \mathbf{G}_t\mathbf{G}_t^\top + \beta_t^\star\mathbf{g}_t\mathbf{g}_t^\top. && (8.100)
\end{aligned}$$

□

La somme est en $O(c^2)$, le principale calcul restant est donc de déterminer la valeur du produit $\mathbf{g}_t\mathbf{g}_t^\top$. Ce calcul peut être effectué en $O(nc)$ opérations. En effet le calcul du vecteur $\mathbf{g}_t = \mathbf{Q}^\top\mathbf{f}_t$ a un coût de $O(nc)$ car $\mathbf{Q}$ est une matrice $n \times c$ et $\mathbf{f}_t$ un vecteur de taille n (construit à partir de n exemples). L'opération $\mathbf{g}_t\mathbf{g}_t^\top$ a une complexité de $O(c^2)$.

Algorithm 16: Étapes de notre algorithme

Input: Des espaces de descripteurs bas-niveau $\mathcal{X}_s$
Input: Une matrice $\mathbf{Q}$ orthogonale de labels de taille $n \times c$.
Input: Un seuil θ.
Input: Un nombre maximal d'itérations T.
begin
  Initialisation
  Calcul des c premières fonctions avec l'Algorithme 15 $\left[f_1(x)f_2(x)\cdots f_c(x)\right]$.
  for $t = c+1, \ldots, T$ **do**
    Calculer la matrice des barycentres $\mathbf{G}_{t-1} = \mathbf{Q}^\top \mathbf{F}_{t-1}$.
    Calcul de la décomposition en valeurs/vecteurs propres de cette matrice :
    $\mathbf{G}_{t-1}\mathbf{G}_{t-1}^\top = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top$.
    Tri des valeurs propres par ordre ascendant.
    for $(\lambda, \mathbf{v}) \in (\mathbf{\Lambda}, \mathbf{V})$ **do**
      Calcul de $\mathbf{g}^\star = \sqrt{1 - \lambda \frac{\|\overline{\mathbf{K}}^\star\|}{\|\overline{\mathbf{K}}_{t-1}\|}}\mathbf{v}$.
      Calcul de la fonction cible $\mathbf{f}^\star = \mathbf{Q}\mathbf{g}^\star$.
      foreach *descripteurs bas-niveau de l'espace* $\mathcal{X}_s$ **do**
        Calcul d'une fonction faible à partir de l'ensemble d'apprentissage $\mathbf{X} \subset \mathcal{X}_s$:
        $\mathbf{f} = \underset{\mathbf{w}\in\mathcal{X}_s}{\arg\min} ||\mathbf{X}\mathbf{w} - \mathbf{f}^\star||^2 \; s.t. \; ||\mathbf{w}||^2 = 1$.
        Calcul de la pondération optimale associée $\beta^\star(f)$ grâce à l' (Eq. 8.20)
        if $\mathcal{A}_\mathbf{H}(\mathbf{K}_{t-1} + \beta^\star \mathbf{f}\mathbf{f}^\top, \mathbf{K}^\star) > \mathcal{A}_\mathbf{H}(\mathbf{K}_{t-1}, \mathbf{K}^\star) + \theta$ **then**
          Mettre à jour le descripteur sémantique $\mathbf{F}_t = \left[\mathbf{F}_{t-1}\ \sqrt{\beta_t}\mathbf{f}_t\right]$.
          Conserver l'hyperplan et le descripteur visuel associé $\mathbf{X}, \mathbf{w}$.
          Passer à l'étape $t+1$.
      if *aucun noyau n'augmente l'alignement* **then**
        $T = t$.
        return ;

Output: Des fonctions faibles $\left[f_1(\mathbf{x})\ f_2(\mathbf{x}) \ldots f_c(\mathbf{x}) \ldots \beta_t f_t(\mathbf{x}) \ldots \beta_T f_T(\mathbf{x})\right]$.

De manière similaire à $\mathbf{g}_t$, la complexité du calcul de la fonction cible $\mathbf{f}^\star = \mathbf{Q}\mathbf{g}^\star$ est de $O(nc)$.

Le reste de la complexité de l'algorithme repose sur les calculs du vecteur $\mathbf{g}^\star$, de la pondération $\beta^\star$ et du nouvel alignement $\mathcal{A}_H(\mathbf{K}_{t-1} + \beta\mathbf{f}\mathbf{f}^\top, \mathbf{K}^\star)$. Pour cela il est nécessaire de calculer les différents termes suivants : $\|\overline{\mathbf{K}^\star}\|$, $\|\overline{\mathbf{f}\mathbf{f}^\top}\|$, $\langle\overline{\mathbf{f}\mathbf{f}^\top}, \overline{\mathbf{K}}_{t-1}\rangle$, $\langle\overline{\mathbf{f}\mathbf{f}^\top}, \overline{\mathbf{K}^\star}\rangle$, $\langle\overline{\mathbf{K}}_{t-1}, \overline{\mathbf{K}^\star}\rangle$ et $\|\overline{\mathbf{K}}_{t-1}\|$.

Ces différents termes sont calculés grâce aux propriétés suivantes :

La propriété 7 nous donnes la norme de la fonction cible $\|\overline{\mathbf{K}^\star}\| = \sqrt{c}$.

Propriété 22. *La norme de la matrice de Gram de la nouvelle fonction noyau $\|\overline{\mathbf{f}\mathbf{f}^\top}\|$ est calculée en temps linéaire grâce au calcul suivant :*

$$\|\overline{\mathbf{f}\mathbf{f}^\top}\| = \|\mathbf{f} - \textit{moyenne}(\mathbf{f})\|^2. \tag{8.101}$$

Démonstration. On a tout d'abord :

$$\begin{aligned}
\|\overline{\mathbf{f}\mathbf{f}^\top}\|^2 &= \mathrm{Tr}(\mathbf{f}\mathbf{f}^\top\mathbf{f}\mathbf{f}^\top) && (8.102)\\
&= \mathrm{Tr}(\mathbf{f}^\top\mathbf{f}\mathbf{f}^\top\mathbf{f}) && (8.103)\\
&= \mathbf{f}^\top\mathbf{f}\mathbf{f}^\top\mathbf{f} && (8.104)\\
&= (\mathbf{f}^\top\mathbf{f})^2 && (8.105)\\
&= \|\mathbf{H}\mathbf{f}\|^4. && (8.106)
\end{aligned}$$

Puis on a

$$\begin{aligned}
\mathbf{H}\mathbf{f} &= \mathbf{f} - \frac{1}{n}\mathbf{1}\mathbf{1}^\top\mathbf{f} && (8.107)\\
&= \mathbf{f} - \text{moyenne}(\mathbf{f}) && (8.108)\\
&\in \mathbb{R}^n. && (8.109)
\end{aligned}$$

□

Propriété 23. *Les produits scalaires avec le nouveau noyau correspondent aux valeurs suivantes :*

$$\langle\overline{\mathbf{f}\mathbf{f}^\top}, \overline{\mathbf{K}^\star}\rangle = \|\mathbf{f}^\top\mathbf{Q}\|^2. \tag{8.110}$$

$$\langle\overline{\mathbf{f}\mathbf{f}^\top}, \overline{\mathbf{K}}_{t-1}\rangle = \|(\mathbf{H}\mathbf{f})^\top\mathbf{F}_{t-1}\|^2. \tag{8.111}$$

Ils ont respectivement une complexité de $O(nc)$ et de $O(nt)$.

Démonstration. Pour obtenir ces deux résultats on s'appuie sur le développement suivant.

Pour tout vecteur $\mathbf{v} \in \mathbb{R}^n$ et toute matrice $\mathbf{M} \in \mathcal{M}_{n,c}(\mathbb{R})$, le produit scalaire $\langle\mathbf{v}\mathbf{v}^\top, \mathbf{M}\mathbf{M}^\top\rangle$ peut se calculer à l'aide de la norme du vecteur $\mathbf{v}^\top\mathbf{M} \in \mathbb{R}^n$:

$$\begin{aligned}
\langle\mathbf{v}\mathbf{v}^\top, \mathbf{M}\mathbf{M}^\top\rangle &= \mathrm{Tr}(\mathbf{v}\mathbf{v}^\top\mathbf{M}\mathbf{M}^\top) && (8.112)\\
&= \mathrm{Tr}(\mathbf{v}^\top\mathbf{M}\left(\mathbf{v}^\top\mathbf{M}\right)^\top) && (8.113)\\
&= \|\mathbf{v}^\top\mathbf{M}\|^2. && (8.114)
\end{aligned}$$

□

Propriété 24. *Le produit scalaire et la norme liée à la fonction que l'on construit peuvent être calculés itérativement à partir des différents produits scalaires et normes calculés précédemment :*

$$\|\overline{\mathbf{K}}_t\|^2 = \|\overline{\mathbf{K}}_{t-1}\|^2 + \beta_t^{\star 2}\|\overline{\mathbf{f}_t\mathbf{f}_t^\top}\|^2 + 2\beta_t^\star\langle\overline{\mathbf{K}}_{t-1}, \overline{\mathbf{f}_t\mathbf{f}_t^\top}\rangle, \tag{8.115}$$

$$\langle\overline{\mathbf{K}}_t, \overline{\mathbf{K}^\star}\rangle = \langle\overline{\mathbf{K}}_{t-1}, \overline{\mathbf{K}^\star}\rangle + \beta_t^\star\langle\overline{\mathbf{f}_t\mathbf{f}_t^\top}, \overline{\mathbf{K}^\star}\rangle. \tag{8.116}$$

Démonstration.

$$\begin{aligned}
\|\overline{\mathbf{K}}_t\|^2 &= \|\overline{\mathbf{K}}_{t-1} + \beta^\star\|\mathbf{k}_t\|^2 && (8.117)\\
&= \|\overline{\mathbf{K}}_{t-1}\|^2 + +\beta^{\star 2}\|\overline{\mathbf{k}}_t\|^2 + 2\beta^\star\langle\overline{\mathbf{K}}_{t-1}, \overline{\mathbf{k}}_t\rangle && (8.118)\\
&= \|\overline{\mathbf{K}}_{t-1}\|^2 + \|\overline{\mathbf{f}_t\mathbf{f}_t^\top}\|^2 + 2\beta^\star\langle\overline{\mathbf{K}}_{t-1}, \overline{\mathbf{f}_t\mathbf{f}_t^\top}\rangle && (8.119)\\
& && (8.120)
\end{aligned}$$

□

A partir des différentes propriétés que l'on vient de voir, on peut en déduire la complexité de notre algorithme :

Théorème 6. *La complexité de l'Algorithme. 16 pour chaque itération t est en $O(c^2 + nc + nt)$ opérations. En considérant que le nombre de classes est bien inférieur au nombre d'images $c \ll n$ et que le nombre d'itérations est supérieur au nombre de classes $c < t$, la complexité de chaque étape est de l'ordre de $O(nt)$.*

Jusqu'ici nous n'avons pas considéré la complexité des apprenants faibles. C'est un paramètre que l'utilisateur doit fixer. Nous avons dans cette thèse considéré des classifieurs à base de LMS dont la complexité est en $O(nd)$ avec d la dimension des descripteurs visuels. Afin de garder la notion de "faiblesse" des algorithmes de Boosting, le nombre de dimensions des descripteurs visuels est négligeable devant le nombre d'images. En considérant ce type de classifieur, il est possible d'utiliser uniquement le vecteur définissant l'hyperplan et réduire la complexité de $\langle\overline{\mathbf{f}\mathbf{f}^\top}, \overline{\mathbf{K}}_{t-1}\rangle$ à $O(dt)$ opérations. En conséquence, chaque itération de notre algorithme a maintenant un coût linéaire en le nombre d'images d'entraînement, indépendant du nombre d'itérations.

8.4 Conclusion

Nous avons présenté dans ce chapitre une nouvelle méthode de Boosting pour la construction de fonctions noyaux. Cette approche construit un nouvel espace sémantique de description des données où est réalisé le produit scalaire de la fonction noyau finale. La construction de cet espace est effectuée de manière itérative par ajout des dimensions les unes après les autres. Chaque dimension est construite pour optimiser la position des barycentres des classes dans le nouvel espace. La sélection de la meilleure fonction est réalisée en maximisant le critère d'alignement à une fonction idéale et la pondération est déduite d'une formule analytique maximisant ce critère.

Cette approche remplit le cahier des charges posé en section 8.1.1. L'approche proposée n'effectue pas simplement une combinaison de noyaux pré-existants mais construit chaque noyau afin de maximiser l'alignement centré entre le noyau multi-classes idéal et le noyau que l'on construit itérativement. Notre démarche permet de produire une fonction noyau calculable aussi bien pour des images "connues" que pour de nouvelles images non présentes lors de l'entraînement. L'algorithme que nous proposons est linéaire et permet ainsi d'appliquer notre méthode à de grandes bases de données.

CHAPITRE 9

Expériences

Sommaire

9.1 Expériences sur des données de synthèse

9.1.1 Une première expérience

Dans un premier temps nous avons cherché à montrer la convergence des centres des classes vers les sommets d'un simplex, dans le but d'avoir une première mesure qualitative de notre méthode. Pour cela nous avons construit un jeu de données de synthèse dans un espace à 2 dimensions.

On considère trois classes de 200 exemples chacune. Chaque exemple va être décrit par 10 descripteurs à 2 dimensions. On commence par construire les centres de ces classes en tirant uniformément 10 descripteurs dans $[0, 1]^2$ pour chaque centre. Chaque descripteur de chaque exemple est alors tiré aléatoirement selon une Gaussienne centrée sur l'un des centres définis précédemment avec un écart type de 0,5.

Chaque classe est divisée en 100 exemples pour l'entraînement et 100 exemples pour le test.

Afin de visualiser la représentation des exemples (i.e. 10 descripteurs de dimension 2) que construit notre algorithme. On effectue une PCA sur ce dernier et on conserve uniquement les 2 dimensions les plus informatives pour visualisation.

La première expérience (Fig. 9.1) montre que les exemples se placent sur les sommets d'un simplex. Dans un cas 2D, cela correspond au sommet d'un triangle équilatéral. Cette première expérience qualitative montre que sur des exemples simples, notre méthode permet bien de placer les exemples à une position la plus éloignée des autres classes, on tend donc bien, sur ces exemples, vers la position idéale décrite par Vapnik.

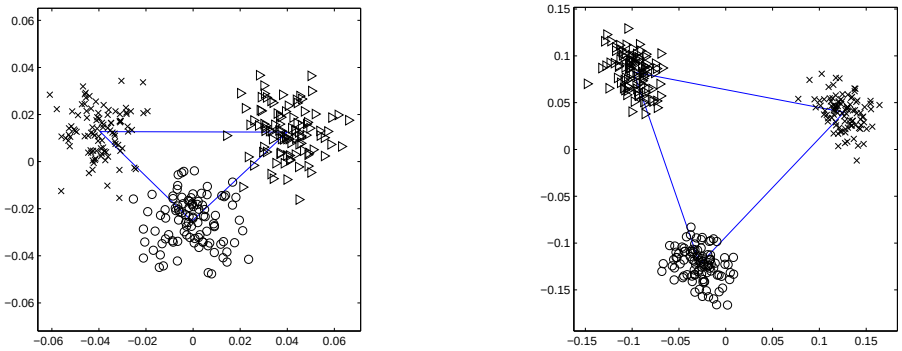


FIGURE 9.1 – Convergence de la méthode vers un 2-simplex régulier pour 3 classes après 2 itérations (initialisation) et 45 itérations.

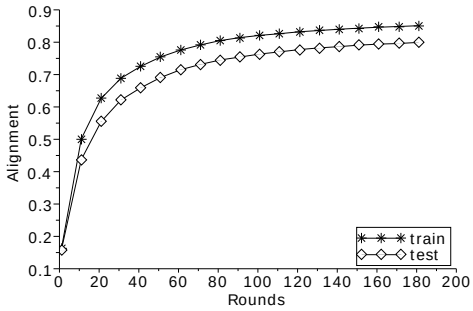


FIGURE 9.2 – Augmentation de l'alignement dans le cas de l'expérience 2 pour 10 classes sur des données de synthèse.

9.1.2 Une seconde expérience

Dans un second temps, nous avons cherché à obtenir des résultats plus quantitatifs sur des données de synthèse. Nous voulions comparer la qualité des descripteurs produits à une autre méthode de l'état de l'art. Pour cette seconde expérience, nous avons utilisé la même approche pour construire nos exemples que celle employée lors de la première expérience. Pour prendre en compte l'aspect multi-classes de notre méthode, nous avons choisi d'utiliser 10 classes d'apprentissage. Un exemple appartient à une et une seule classe. Pour chaque classe nous avons construit 60 exemples, 30 pour l'apprentissage et 30 pour les tests. Nous avons construit les descripteurs de la même façon que précédemment. Chaque exemple comporte 25 descripteurs de 16 dimensions. Le centre de chaque cluster est choisi par un tirage uniforme sur $[0, 1]^{16}$ et les valeurs sont définies par une Gaussienne autour de ces centres avec un écart-type de 0,5.

Cette seconde expérience a pour objectif de confirmer avec des exemples de synthèse, les observations effectuées lors de l'expérience qualitative préliminaire. La figure (Fig. 9.2) montre qu'au cours des itérations de la méthode, l'alignement est toujours croissant. De plus l'alignement augmente aussi bien sur l'ensemble d'apprentissage (ce qui est imposé par notre méthode), que sur l'ensemble de test.

TABLE 9.1 – % d'erreur avec un algorithme de k-Plus-Proche-Voisin sur les descripteurs sémantiques produits par notre méthode avec des données de synthèse

k	1	2	3	4	5	6	7	8	9	10
Kawanabe	20	24	16	13	**8**	**6**	**5**	**5**	**4**	**5**
Notre méthode	**14**	**18**	**11**	**9**	**8**	7	7	6	5	**5**

A partir des descripteurs calculés par notre méthode, il est possible de déduire la matrice de similarité entre images. Cette matrice peut être alors utilisée dans un classifieur par k-Plus-Proches-Voisins (k-nearest neighbor kNN), permettant ainsi une classification facile des différents exemples. Nous comparons alors les résultats de ce kNN avec un kNN appris à partir de la matrice de similarité produite par un algorithme de MKL de la littérature [Kawanabe 2009].

La méthode de [Kawanabe 2009] a une complexité quadratique en nombre d'exemples tandis que notre approche est linéaire. Outre cet avantage calculatoire, les résultats en termes de performance de classification (Tab. 9.1) montrent un gain de notre approche dans le cas où le nombre de voisins de référence est moins important. Cela montre que notre méthode regroupe mieux les exemples d'une classe autour de son barycentre ; ainsi un exemple a généralement un voisin proche appartenant à la même classe que lui.

9.2 VOC 2006

9.2.1 Présentation de la base

La base Visual Object Category (VOC) 2006 est constituée de 5304 images provenant de Microsoft Research Cambridge et du site internet Flickr. Cette base d'images se décompose en trois sous-ensembles d'images : l'ensemble d'apprentissage, de validation et de test.

Chaque image comporte au moins un objet d'une des dix catégories (bus, vélo, chat, voiture, vache, cheval, moto, personne ou mouton), pour un total global de 9507 annotations. Certaines images comportent plusieurs annotations de classes différentes.

Pour plus de détail sur cette base, on peut se reporter à la section 5.4.1.

9.2.2 Résultats expérimentaux

Nous avons utilisé cette base pour montrer le gain offert par notre méthode par rapport aux descripteurs initiaux. Les images ont été décrites à l'aide de descripteurs de textures (histogramme d'ondelette quaternionique [Qw] [Chan 2004]) et de couleurs (histogramme CIE Lab) avec plusieurs tailles d'histogrammes (16, 32, 64 et 128 dimensions). Nous avons effectué des expériences avec ces descripteurs pris séparément ainsi que concaténés tous ensembles (Tab.9.2). La concaténation de tous les descripteurs donnent de meilleures performances que les descripteurs pris séparément. De plus, les descripteurs textures sont plus efficaces que les descripteurs couleurs.

Nous avons par la suite comparé ces performances aux descripteurs produits par notre approche. Les noyaux faibles de chaque itération de notre algorithme sont construit à partir des différents descripteurs


1.Bus


2.Chat


3.Chien


4.Cheval


5.Moto


6.Mouton


7.Personne


8.Vache


9.Vélo


10.Voiture

FIGURE 9.3 – Les 10 catégories de Voc 2006

TABLE 9.2 – Précision moyenne sur VOC2006 en % pour un SVM linéaire et des descripteurs visuels et sémantique

Catégories	Bus	Vélo	Voiture	Chat	Vache	Chien	Cheval	Moto	Personne	Mouton	Tous
lab16	12.4	9.4	24.6	16.0	9.4	11.2	9.0	8.0	27.2	10.9	14.0
lab32	10.7	8.1	45.7	27.1	34.2	15.9	10.0	9.1	27.0	25.6	22.8
lab64	10.0	12.5	47.9	28.5	37.5	19.1	9.9	16.4	31.7	50.3	26.3
lab128	18.7	24.7	46.6	28.8	34.1	20.0	16.0	16.5	33.2	50.0	28.7
qw16	14.0	46.8	55.5	15.1	7.5	14.6	8.3	21.0	25.6	11.2	22.0
qw32	38.5	52.2	60.2	22.2	7.7	15.9	8.9	36.0	36.9	25.6	30.4
qw64	43.0	53.1	63.4	22.0	14.2	18.8	13.2	43.3	37.5	36.2	34.4
qw128	47.9	57.4	65.6	25.8	14.8	20.3	21.6	45.5	33.2	48.6	37.6
Tous	52.1	58.2	72.2	37.4	38.5	27.1	26.7	44.4	39.5	56.1	45.3
Notre méthode	**52.2**	**63.1**	**75.9**	**43.8**	**41.6**	**27.6**	**27.2**	**52.7**	**41.9**	**56.6**	**48.3**

visuels présenter précédemment. Nos nouveaux descripteurs réalisent de meilleures performances sur l'ensemble des catégories, montrant par là même l'intérêt de notre méthode par rapport à une simple concaténation de descripteurs.

9.3 Oxford Flowers 17 et 102

9.3.1 Présentation de la base

Les bases Oxford Flowers 17 et 102 sont des bases d'images de fleurs proposées par [Nilsback 2006]. Elles sont composées respectivement de 17 et 102 catégories de fleurs que l'on peut trouver couramment en Angleterre.

Flowers 17

La base Flowers 17 est composée de 80 images par catégorie. Les fleurs sont à des échelles variables dans l'image et sont soumises à différentes expositions. Certaines classes peuvent être similaires tandis qu'au sein d'une classe certaines images peuvent être très variables.

La base est découpées en trois ensembles : train (40 images par classes), val (20 images par classes) et test (20 images par classes).

Les auteurs proposent également 3 découpages aléatoires prédéfinis. Ces trois versions de découpage permettent de calculer un écart-type sur les résultats d'apprentissage et ainsi donner un aperçu des différences de performances introduites par le choix des images d'apprentissage.

Pour cette base, plusieurs outils sont fournis. En plus des images et des labels, les auteurs proposent des matrices de similarité calculées pour plusieurs descripteurs. Ces matrices correspondent au calcul de la distance χ^2 entre les histogrammes de descripteur de chaque image. Sept descripteurs ont été employés par les auteurs de cette base, des histogrammes :

- de couleurs,
- de formes,
- de texture,
- de HSV,
- de SIFT à l'intérieur de la région d'intérêt,
- de SIFT sur la frontière de la région d'intérêt,
- de gradients orientés.

Flowers 102

La base Flowers 102 est composée de 8189 images. Elle est également découpée en trois ensembles, train (10 images par classe), val (10 images par classe) et test (6129 images avec au moins 20 images par classes). La répartition des images au sein de chaque catégorie dans l'ensemble de test est variable. Les classes ne sont pas équitablement réparties dans la base, le nombre d'images par catégorie varie de 40 images à 250 (Fig. 9.4).

Néanmoins pour cette base, les auteurs proposent un unique découpage aléatoire, aucun calcul d'écart-type n'est alors possible avec les données fournis par les auteurs.

De même que précédemment, les auteurs fournissent des matrices de similarité calculées par distance du χ^2 sur des histogrammes de descripteurs :

- HSV,

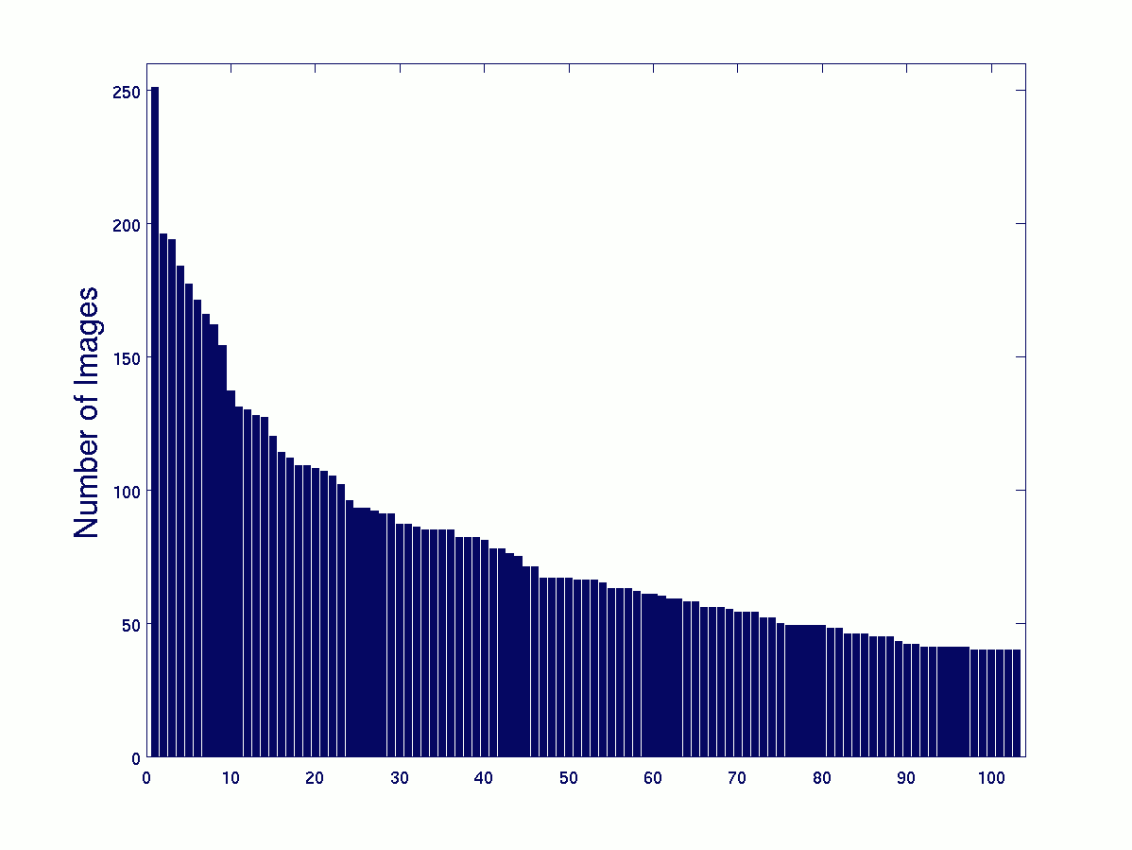


FIGURE 9.4 – Distribution du nombre d'images en fonction des classes dans Flowers 102

– SIFT à l'intérieur de la région d'intérêt,
– SIFT sur la frontière de la région d'intérêt,
– Gradients orientés.

9.3.2 Protocole expérimental

Nous avons réalisé deux types d'expérience sur les bases Flowers. Même si on ne teste pas les mêmes propriétés pour ces deux expériences, nous utilisons un protocole similaire dans les deux cas. Pour réaliser ces expériences nous avons repris le schéma en deux étapes de [Cortes 2010] (pour de plus amples précisions, on pourra se référer à la section. 7.3.3). Dans une première phrase nous créons à l'aide de notre algorithme de nouveaux descripteurs sémantiques qui seront utilisés dans un second temps pour apprendre des SVM gaussien ℓ_2 sur chaque catégorie.

Les données fournies par les auteurs de ces bases sont sous la forme de plusieurs matrices de similarité $\mathbf{D}$. Afin de transformer ces différentes matrices en descripteurs visuels, nous effectuons une PCA sur la matrice $\mathbf{X} = e^{-\mu_f \mathbf{D}}$ où μ_f est l'inverse de la moyenne des distances χ^2 sur tous les exemples d'apprentissage. Nous ne gardons pas toutes les dimensions de la PCA mais seulement les plus informatives.

La première expérience consiste justement à observer les différents résultats de classification en fonction du nombre de dimensions conservées et le nombre d'itérations de la méthode.

Lors du déroulement de l'algorithme nous privilégions en premier les descripteurs de petites dimensions. Si aucune solution n'est trouvée avec ces descripteurs, l'algorithme se met alors à étudier les descripteurs de dimensions supérieures.

Lors de la deuxième expérience, nous comparons notre méthode à d'autres algorithmes de la littérature.

9.3.3 Résultats expérimentaux

Flowers 17

TABLE 9.3 – Taux de classification sur Flowers 17 pour différents nombres d'itérations maximum et différentes dimensions maximum des descripteurs visuels. La première ligne correspond à la dimensiosn max des descripteurs visuels utilisé, tandis que la première colonne correspond au nombre d'itérations maximale atteint par l'algorithme.

	16	32	64	128	256	512	1024
50	78.24 ± 2.70	78.24 ± 2.70	78.24 ± 2.70	78.24 ± 2.70	78.24 ± 2.70	78.24 ± 2.70	78.24 ± 2.70
100	83.73 ± 2.45	83.73 ± 2.45	83.73 ± 2.45	83.73 ± 2.45	83.73 ± 2.45	83.73 ± 2.45	83.73 ± 2.45
200	84.12 ± 1.35	84.8 ± 1.80	85.78 ± 2.67	85.88 ± 2.56	85.88 ± 2.56	85.88 ± 2.56	85.88 ± 2.56
300	-	84.9 ± 1.67	86.27 ± 2.25	85.88 ± 2.33	86.47 ± 2.22	86.47 ± 2.22	86.47 ± 2.22
400	-	-	86.67 ± 2.47	86.27 ± 2.64	87.55 ± 1.80	87.55 ± 1.80	87.55 ± 1.80
500	-	-	-	86.47 ± 2.84	87.94 ± 1.47	**88.33 ± 1.11**	**88.33 ± 1.11**

La première expérience a été mise en place afin d'étudier le comportement de notre algorithme par rapport aux différents paramètres de la méthode. Nous cherchons à connaître comment se déroule le choix des descripteurs visuels en fonction de leurs dimensions pour la construction des fonctions sélectionnées.

TABLE 9.4 – Taux de classification sur Flowers 17

MKL [Awais 2011]	87.2±2.7
NLP-β [Awais 2011]	**87.9**±1.8
NLP-νMC [Awais 2011]	87.8±2.1
NLP-B [Awais 2011]	87.3±2.7
MKL-prod [Gehler 2009]	85.5 ± 1.2
MKL-avg (l_∞) [Gehler 2009]	84.9 ± 1.9
CF (l_∞) / AKM [Gehler 2009]	86.7 ± 2.7
CG-Boost [Gehler 2009]	84.8 ± 2.2
MKL (SILP or Simple) [Gehler 2009, Rätsch 2000]	85.2 ± 1.5
LP-β [Gehler 2009]	85.5 ± 3.0
LP-B [Gehler 2009]	85.4 ± 2.4
MKL-FDA (l_p) [Yan 2010]	86.7 ± **1.2**
Proposé	**88.3** ± **1.1**

Nous voulons également mesurer l'influence du nombre d'itération de la méthode sur les performances de classification finale.

Le premier constat que l'on peut faire est que le nombre d'itérations influe directement sur les résultats de classification. Plus notre approche sélectionne de fonctions, plus les performances sont améliorées. On remarque également que si on dispose uniquement de descripteurs de petites dimensions la méthode ne peut pas effectuer toute les itérations demandées et s'arrête prématurément par manque de fonction améliorant l'alignement. Par exemple pour des dimensions maximales de descripteurs de 16 dimensions, l'algorithme n'a pu atteindre les 300 dimensions.

L'étude des résultats pour 50 et 100 dimensions montre des performances similaires quel que soit le nombre de dimensions des descripteurs visuels. Cela s'explique par le choix exclusif des descripteurs de dimension 16 dans les premières itérations de la méthode. Les descripteurs de plus grande dimension ne sont pas mis à contribution car les descripteurs de taille 16 ne sont pas mis en échec et augmentent à chaque itération l'alignement de manière suffisants. Néanmoins si on augmente suffisamment le nombre d'itérations les descripteurs de taille 16 ne sont plus suffisant et aucun n'augmente suffisamment l'alignement pour poursuivre l'algorithme. Les descripteurs de taille 32 sont alors utilisés par l'algorithme. On remarque ce passage sur la ligne 200, les performances à 16 dimensions commencent à différer des performances à 32 dimensions.

On peut alors se demander pourquoi les performances de 32 et 64 dimensions ne sont pas identiques à la ligne des 200 itérations. Cela s'explique par le mécanisme de traitement des situations d'échecs. Si aucune fonction n'a pu être construite avec la dimension courante, l'algorithme cherche à maximiser l'alignement avec les dimensions supérieures (cette opération correspond au **foreach** de l'Algorithme.16). En cas de nouvel échec l'approche s'intéresse au vecteur propre suivant pour construire la fonction cible (en suivant la boucle sur (Λ, V) de l'Algorithme.16) et recommence le processus décrit précédemment. Donc pour la ligne 200, on observe le déroulement suivant : l'algorithme a commencé par construire les 100 premières fonctions avec les descripteurs de taille 16. A partir d'une certaine itération, ces descripteurs échouent. Si on dispose de descripteur de taille 32, la méthode les utilise pour continuer sa progression. Ces descripteurs échouent alors aussi. Ici deux cas de figure se présentent, soit on dispose de descripteurs

de taille 64 et l'algorithme continue avec ces descripteurs, soit l'algorithme ne dispose que des descripteurs de taille 16 et 32, il s'intéresse donc au vecteur propre suivant pour définir le barycentre cible et recommence le processus de sélection avec les descripteurs de taille 16. Après 200 dimensions tous les descripteurs de taille 16 sont mis en échec pour tous les vecteurs propres, il est alors nécessaire de passer à une dimension plus importante. Cela se comprend bien intuitivement, plus on arrive à une description précise des classes plus il est nécessaire d'être précis dans l'information que l'on apporte pour continuer la progression de la fonction noyau vers la fonction cible.

Dans la deuxième expérience (Tab. 9.4), nous comparons notre approche à d'autres méthodes de la littérature, dont les résultats sont présentés dans [Awais 2011]. Ces expériences montrent que notre méthode améliore les performances sur cette base tout en réduisant l'écart-type entre les différents ensembles d'entraînement.

Flowers 102

TABLE 9.5 – Taux de classification sur Flowers 102

MKL [Awais 2011]	73.4
NLP-β [Awais 2011]	**75.7**
NLP-νMC [Awais 2011]	73.4
NLP-B [Awais 2011]	73.60
MKL-prod [Nilsback 2008]	73.8
MKL-avg [Nilsback 2008]	73.4
MKL [Nilsback 2008]	72.8
Proposé	**77.8**

Sur la base Flowers 102 nous avons uniquement effectué une comparaison avec les méthodes les plus récentes de l'état de l'art issues de [Awais 2011]. Nous observons également un gain en performance sur cette base montrant par là même l'apport de notre approche. Notre approche est à ses débuts, ces expériences sont donc très encourageantes et montrent le potentiel de ce type d'approche.

9.4 Conclusion

Nous avons présenté, dans ce chapitre, différentes expériences montrant la pertinence du modèle que nous proposons.

Dans un premier temps nous avons réalisé des expériences qualitatives sur des données de synthèse. Nous avons pu montrer la convergence, par notre méthode, des barycentres vers les sommets d'un simplex. Cette configuration des barycentres maximise les performances des classifieurs hyperplan que l'on peut utiliser par la suite pour réaliser la catégorisation.

Au cours d'une seconde expérience sur des données de synthèse, nous avons confirmé l'augmentation du critère d'alignement au fil des itérations. L'utilisation d'un algorithme de classification de type kNN a permis de faire une première analyse quantitative des résultats de classification de notre approche.

Nous avons ensuite réalisé des expériences sur des bases d'images standard. Nous avons commencé par étudier la base Pascal VOC 2006. Nous avons comparé les performances de classification en considérant soit des descripteurs visuels classiques, soit les descripteurs sémantiques que produit notre méthode à partir de ces mêmes descripteurs visuels classiques. Ces expériences ont montré tout l'intérêt de notre approche qui classifie mieux avec nos descripteurs sémantiques. Les expériences suivantes ont été réalisées sur les bases Flowers 17 et 102. Nous avons dans un premier temps testé l'influence des paramètres de notre méthode, c'est à dire le "niveau de faiblesse" des noyaux faibles utilisés et l'importance du nombre d'itérations. On a ainsi pu voir que notre approche commence par sélectionner les noyaux les plus faibles et que la précision des fonctions noyaux nécessaires pour poursuivre l'algorithme augmente avec le nombre d'itérations.

Notre approche fut également comparée aux approches les plus récentes sur les bases Flowers 17 et 102. Ces expériences ont mis en avant le gain en performance qu'offre notre approche sur ces bases.

Nous avons présenté dans cette partie une nouvelle méthode pour construire des fonctions noyaux multi-classes pour la catégorisation d'images sur des grandes bases. La fonction noyau construite peut par la suite être utilisée dans une méthode à noyaux comme les SVM. Le protocole alors considéré se déroule en deux temps, dans une première étape un noyau est appris par notre méthode puis dans une seconde étape un classifieur est construit à l'aide de ce noyau. L'idée sous-jacente est de séparer la phase de conception du modèle, de l'étape du choix d'un hyperplan séparant les données.

Notre méthode construit donc un nouveau noyau par combinaison de noyaux faibles selon un processus inspiré des méthodes de Boosting. Cette approche autorise l'utilisation d'un grand nombre de "noyaux faibles" pour un coût calculatoire linéaire en le nombre d'images d'entraînement. Elle n'exprime pas directement la fonction noyau mais permet d'obtenir une fonction de changement d'espace dans lequel effectuer le produit scalaire. Il est alors possible de prendre en compte de nouvelles images, non considérées lors de la phase d'entraînement.

Les différentes expériences réalisées ont permis de montrer la pertinence de notre modèle. Notre méthode obtient de meilleurs résultats que les méthodes concurrentes de l'état de l'art.

CHAPITRE 10

Conclusion générale et perspectives

10.1 Bilan

Ces dernières années ont vu la croissance toujours plus importante de la quantité de données multimédia. L'analyse de ces données, notamment via la recherche de catégories, fait face à un besoin de réduction de la complexité calculatoire. Dans cette thèse nous avons proposé des méthodes qui visent à répondre à cet enjeu, tout en poursuivant le travail d'amélioration des performances de catégorisation.

Nous avons présenté, dans un premier temps, une méthode de recherche interactive. Cette méthode s'appuie sur une adaptation du Boosting à ce type de recherche. Les méthodes de Boosting ont montré leur simplicité de conception. En effet, elles nécessitent peu de paramètres de réglage. Par ailleurs, ces méthodes sont reconnues pour leurs performances en généralisation et leur faible coût calculatoire en phase de test. A contrario ces approches nécessitent un long temps d'apprentissage et un nombre très important d'exemples d'apprentissage. Pour réaliser du Boosting interactif, nous avons proposé une nouvelle méthode de Boosting adaptée au protocole interactif. Cette méthode a pour but de travailler avec peu d'exemples d'apprentissage et un nombre de classifieurs faibles restreint. Afin de remplir cet objectif, nous avons adapté l'algorithme RankBoost à notre contexte. Nous avons, tout d'abord, proposé une nouvelle manière de construire l'ensemble des classifieurs faibles sélectionnables en s'appuyant uniquement sur des classifieurs construits à partir des caractéristiques visuelles des images positives. Puis nous avons proposé des classifieurs faibles reposant sur des images de référence pour répondre au besoin de notre algorithme. Enfin nous proposons un critère actif pour la sélection des prochaines images à annoter. Nous avons ainsi pu adapter le Boosting aux contraintes de la recherche interactive. Cette nouvelle méthode est comparée à une approche SVM interactive et on peut en déduire toute la pertinence de notre approche.

Dans un deuxième temps, nous avons mis en place une technique de Boosting pour l'apprentissage d'espaces. Toujours avec le souci de travailler avec de grandes bases, nous nous sommes attachés à établir un algorithme calculable en temps linéaire en nombre d'images d'entraînement. Nous avons attaché beaucoup d'importance à cette contrainte de complexité et elle fut à l'origine des différents choix réalisés. L'approche proposée construit un nouvel espace sémantique ayant pour but d'améliorer la recherche multi-classes d'un futur classifieur. Notre algorithme repose sur plusieurs points. Tout d'abord, nous proposons une matrice cible basée sur la décomposition QR d'une matrice de labels. Cette matrice possède des propriétés intéressantes. En effet, chacune des classes sont équidistantes entre elles ce qui permet de les séparer plus facilement à l'aide d'un hyperplan. La méthode de construction de l'espace sémantique où est calculé le produit scalaire de notre fonction noyau est réalisée par une méthode de Boosting dimension par dimension. A chaque itération de la méthode, une direction cible est déterminée par l'algorithme en fonction de la position des barycentres de chacune des classes. L'objectif est de faire converger la position des barycentres vers un simplex où chaque classe serait équidistante des autres comme c'est le cas pour la matrice cible. Les classifieurs faibles ensuite utilisés s'appuient sur les descripteurs visuels

des images de la base et cherchent à être les plus proches de la direction cible. L'algorithme choisit alors la fonction ainsi créée qui augmentera le plus l'alignement avec la matrice objectif. Une méthode analytique permet d'en déduire la pondération optimale à lui appliquer pour maximiser l'alignement. En outre, nous proposons un algorithme de combinaison des noyaux dont la complexité est au plus linéaire en fonction du nombre d'images. Les différentes expériences réalisées démontrent la capacité de notre approche à offrir des performances au niveau de l'état de l'art, en particulier lorsqu'on la compare aux meilleures méthodes concurrentes basées sur le MKL. L'un des principaux avantages de notre méthode est qu'elle permet de fabriquer entièrement la fonction noyau finale. Elle ne consiste pas uniquement en une combinaison de fonctions noyaux préexistants mais permet également de construire les fonctions noyaux de la combinaison. Les expériences sur des données de synthèse ont montré que notre approche permet de positionner les exemples dans un nouvel espace mieux adapté à la catégorisation. Chaque centre de classe est équidistant des autres dans le nouvel espace. Les résultats de comparaison avec les descripteurs visuels d'origine sur la base VOC, confirment expérimentalement le gain qu'offre ce nouvel espace de description des images. L'intérêt de construire les fonctions noyaux au cours de l'apprentissage du noyau par rapport à une sélection pondérée de noyaux existants est montré par les expériences comparant notre approche aux méthodes de MKL ou reposant sur des versions LP-Boost à noyau. Sur les bases Flowers 17 et 102, notre approche a obtenu des performances supérieures à celles présentées dans la littérature.

10.2 Perspectives

Dans ce manuscrit nous avons développé des travaux à la fois sur la recherche interactive et sur la construction de nouveaux espaces plus adaptés à la catégorisation d'images. Plusieurs perspectives s'offrent à nous pour poursuivre les idées proposées dans ce document.

- à court terme :
  - Nous avons montré le caractère linéaire de notre approche de construction de fonctions noyaux. Notre méthode devra être testée sur des bases d'images de taille plus importante comme par exemple la base SUN ou les bases du challenge ImageNET.
- à moyen terme :
  - Nos deux approches peuvent être étendues à la recherche sur d'autre modalité par exemple la recherche sur des images satellites ou d'événement dans des vidéos. Ceci peut être rendu possible en combinant notre méthodes à des descripteurs visuels adaptés à ce type de contenu. On peut utiliser, pour les vidéos les descripteurs STIP, MotionSIFT, Tenseurs de mouvement...
  - D'autres classifieurs faibles que les LMS pourraient être utilisés.
  - Il serait intéressant d'étudier l'extension de notre approche de construction de noyau à la recherche collaborative. La matrice cible est actuellement construite à partir de la décomposition QR d'une matrice de labels. On peut envisager d'utiliser d'autres constructions de cette matrice, par exemple à partir de l'ensemble des "logs" des sessions de recherche menées par plusieurs utilisateurs sur un système de recherche interactive.
- à long terme :
  - Les méthodes présentées sont actuellement séquentielles, il serait intéressant d'étudier la faisabilité d'une parallélisation.
  - Nous avons présenté une approche de Boosting interactif ainsi qu'une approche de construction de noyau par Boosting. La fusion des deux idées serait une perspective de recherche très intéressante.

Publications de l'auteur

Journaux internationaux

– A.Lechervy, P-H Gosselin, F. Precioso. *Boosted Kernel for Image Categorization*. Multimedia Tools and Applications (MTAP), 2012.

Conférences internationales

– A.Lechervy, P-H Gosselin, F. Precioso. *Active Boosting for interactive object retrieval*. International Conference on Pattern Recognition (ICPR), Istanbul, Turkey, August 2010.
– A.Lechervy, P-H Gosselin, F. Precioso. *Linear kernel combination using boosting*. European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN), Bruges, Belguim, April 2012.
– A.Lechervy, P-H Gosselin, F. Precioso. *Boosting kernel combination for multi-class image categorization*. IEEE International Conference on Image Processing (ICIP), Orlando, Florida, U.S.A, September 2012.

Conférences nationales

– A.Lechervy, P-H Gosselin, F. Precioso. *Boosting actif pour la recherche interactive d'images*. Reconnaissance des Formes et Intelligence Artificielle (RFIA2010), Caen, France, Jan. 2010.

Bibliographie

[Aizerman 1964] M. A. Aizerman, E. A. Braverman et L. Rozonoer. *Theoretical foundations of the potential function method in pattern recognition learning.* In Automation and Remote Control,, numéro 25, pages 821–837, 1964. (Cité en page 81.)

[Aksoy 2000] S. Aksoy, R.M. Haralick, F.A. Cheikh et M. Gabbouj. *A Weighted Distance Approach to Relevance Feedback*. In IAPR International Conference on Pattern Recognition, volume IV, pages 812–815, Barcelona, Spain, September, 3-8 2000. (Cité en page 16.)

[Awais 2011] M. Awais, F. Yan, K. Mikolajczyk et J. Kittler. *Augmented Kernel Matrix vs Classifier Fusion for Object Recognition*. In Proceedings of the British Machine Vision Conference, pages 60.1–60.11. BMVA Press, 2011. (Cité en pages 130 et 131.)

[Bach 2004] F. Bach, G. R. G. Lanckriet et M. I. Jordan. *Multiple Kernel Learning, Conic Duality, and the SMO Algorithm*. In International Conference on Machine Learning, 2004. (Cité en pages 94 et 97.)

[Bach 2008] F. Bach. *Consistency of the group Lasso and multiple kernel learning*. J. Mach. Learn. Res., vol. 9, pages 1179–1225, 2008. (Cité en pages 94 et 97.)

[Berrani 2003] Sid-Ahmed Berrani, Laurent Amsaleg et Patrick Gros. *Recherche approximative de plus proches voisins : application la reconnaissance d'images par descripteurs locaux*. Technique et Science Informatiques, vol. 22, no. 9, pages 1201–1230, 2003. (Cité en page 18.)

[Boser 1992] Bernhard E. Boser, Isabelle M. Guyon et Vladimir N. Vapnik. *A training algorithm for optimal margin classifiers*. In Proceedings of the fifth annual workshop on Computational learning theory, COLT '92, pages 144–152, New York, NY, USA, 1992. ACM. (Cité en pages 80, 85 et 90.)

[Breiman 1999] Leo Breiman. *Prediction games and arcing algorithms*. Neural Comput., vol. 11, pages 1493–1517, October 1999. (Cité en pages 35 et 36.)

[Caenen 2000] Geert Caenen, Greet Frederix, A. A. M. Kuijk, Eric J. Pauwels et Ben A. M. Schouten. *Show Me What You Mean ! Pariss : A CBIR-Interface that Learns by Example*. In Proceedings of the 4th International Conference on Advances in Visual Information Systems, VISUAL '00, pages 257–268, London, UK, UK, 2000. Springer-Verlag. (Cité en page 7.)

[Censor 1992] Y. Censor et S. A. Zenios. *Proximal minimization algorithm with D-functions*. J. Optim. Theory Appl., vol. 73, pages 451–464, June 1992. (Cité en page 41.)

[Chan 2004] W. L. Chan, H. Choi et R. Baraniuk. *Quaternion wavelets for image analysis and processing*. In International Conference on Image Processing, volume 5, pages 3057–3060, October 2004. (Cité en pages 60 et 125.)

[Chang 2003] E. Chang, B. T. Li, G. Wu et K.S. Goh. *Statistical Learning for Effective Visual Information Retrieval*. In International Conference on Image Processing, pages 609–612, Barcelona, Spain, September 2003. (Cité en page 18.)

[Chapelle 1999] O. Chapelle, P. Haffner et V. Vapnik. *SVMs for Histogram Based Image Classification*. IEEE Transactions on Neural Networks, vol. 10, pages 1055–1064, 1999. (Cité en page 18.)

[Chapelle 2007] Olivier Chapelle. *Training a Support Vector Machine in the Primal*. Neural Computing, vol. 19, no. 5, pages 1155–1178, Mai 2007. (Cité en page 87.)

[Collins 2008] B. Collins, J. Deng, K. Li et L. Fei-Fei. *Towards Scalable Dataset Construction : An Active Learning Approach*. In European Conference on Computer Vision, pages 86–98, 2008. (Cité en pages 18, 19 et 53.)

[Cortes 1995] Corinna Cortes et Vladimir Vapnik. *Support-Vector Networks*. In Machine Learning, pages 273–297, 1995. (Cité en page 88.)

[Cortes 2010] C. Cortes, M. Mohri et A. Rostamizadeh. *Two-Stage Learning Kernel Algorithms*. In ICML, 2010. (Cité en pages 78, 94, 95, 96, 97, 100 et 129.)

[Cox 2000] I.J. Cox, M.L. Miller, T.P. Minka, T.V. Papathomas et P.N. Yianilos. *The Bayesian Image Retrieval System, PicHunter : Theory, Implementation and Psychophysical Experiments*. IEEE Transactions on Image Processing, vol. 9, no. 1, pages 20–37, 2000. (Cité en pages 5 et 18.)

[Crammer 2002] K. Crammer, J. Keshet et Y. Singer. *Kernel design using boosting*. In Advances in Neural Information Processing Systems, pages 537–544. MIT Press, 2002. (Cité en pages 95, 96, 97 et 98.)

[Cristianini 2001] N. Cristianini, J. Shawe-Taylor, A. Elisseff et J. Kandola. *On kernel target alignement*. In Advances in Neural Information Processing Systems, pages 367–373, Vancouver, Canada, December 2001. (Cité en page 78.)

[Cristianini 2006] Nello Cristianini, J Kandola, A Elisseff et John Shawe-Taylor. *On Kernel Target Alignment*. Innovations in Machine Learning : Theory and Application, 2006. (Cité en page 78.)

[Demiriz 2002] Ayhan Demiriz, Kristin P. Bennett et John Shawe-Taylor. *Linear Programming Boosting via Column Generation*. Mach. Learn., vol. 46, no. 1-3, pages 225–254, Mars 2002. (Cité en page 37.)

[Diao 2002] Lili Diao, Keyun Hu, Yuchang Lu et Chunyi Shi. *A Method to Boost Support Vector Machines*. In PAKDD '02 : Proceedings of the 6th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining, pages 463–468, London, UK, 2002. Springer-Verlag. (Cité en page 53.)

[Doulamis 2001] N. Doulamis et A. Doulamis. *A recursive optimal relevance feedback scheme for CBIR*. In International Conference on Image Processing, pages 741–744, Thessaloniki, Greece, October 2001. (Cité en page 18.)

[Ehrenfeucht 1988] A. Ehrenfeucht, David Haussler, Michael Kearns et Leslie Valiant. *A general lower bound on the number of examples needed for learning*. In Proceedings of the first annual workshop on Computational learning theory, COLT '88, pages 139–154, San Francisco, CA, USA, 1988. Morgan Kaufmann Publishers Inc. (Cité en page 23.)

[Everingham 2006] M. Everingham, A. Zisserman, C. K. I. Williams et L. Van Gool. *The PASCAL Visual Object Classes Challenge 2006 Results*. http ://www.pascal-network.org/challenges/VOC/voc2006/results.pdf, 2006. (Cité en page 59.)

[Fournier 2001] J. Fournier, M. Cord et S. Philipp-Foliguet. *Back-propagation algorithm for relevance feedback in image retrieval*. In International Conference on Image Processing, volume 1, pages 686–689, Thessaloniki, Greece, October 2001. (Cité en page 18.)

[Freinet 1964] Célestin Freinet. *Les invariants pédagogiques*, 1964. (Cité en page 19.)

[Freund 1999] Yoav Freund et Robert E. Schapire. *A short introduction to boosting*. J. Japan. Soc. for Artif. Intel., vol. 14, no. 5, pages 771–780, 1999. (Cité en pages 23 et 25.)

[Freund 2003] Y. Freund, R. Yyer, R.E. Schapire et Y. Singer. *An Efficient Boosting Algorithm for Combining Preferences*. Journal on Machine Learning Research, vol. 4, pages 933–969, November 2003. (Cité en pages 46, 47 et 56.)

[Friedman 1998] J. Friedman, T. Hastie et R. Tibshirani. *Additive logistic regression : a statistical view of boosting*. The Annals of Statistics, vol. 38, no. 2, 1998. (Cité en pages 43 et 44.)

[Gehler 2009] P. Gehler et S. Nowozin. *On feature combination for multiclass object classification*. In IEEE International Conference on Computer Vision, pages 221–228, 2009. (Cité en pages 96, 98 et 130.)

[Geman 2000] D. Geman et R. Moquet. *A Stochastic Feedback Model for Image Retrieval*. In Reconnaissance des Formes et Intelligence Artificielle, volume III, pages 173–180, Paris, France, February 2000. (Cité en page 18.)

[Gosselin 2008] P.H. Gosselin, M. Cord et S. Philipp-Foliguet. *Combining visual dictionary, kernel-based similarity and learning strategy for image category retrieval*. Computer Vision and Image Understanding, Special Issue on Similarity Matching in Computer Vision and Multimedia, vol. 110, no. 3, pages 403–417, 2008. (Cité en page 61.)

[Grabner 2006] Helmut Grabner et Horst Bischof. *On-line Boosting and Vision*. In CVPR '06 : Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pages 260–267, Washington, DC, USA, 2006. IEEE Computer Society. (Cité en pages vii, 50, 52 et 54.)

[Guermeur 2004] Yann Guermeur, Alain Lifchitz et Régis Vert. *A Kernel for Protein Secondary Structure Prediction*. In Jean-Philippe Vert Bernhard Schölkopf Koji Tsuda, editeur, Kernel Methods in Computational Biology, Chap. 9 - ISBN 0-262-19509-7, pages 193–206. The MIT Press, Cambridge, Massachussetts, 2004. http ://mitpress.mit.edu/catalog/item/default.asp ?ttype=2&tid=10338&mode=toc. (Cité en pages viii et 101.)

[Guyon 1993] I. Guyon, B. Boser et V. Vapnik. *Automatic Capacity Tuning of Very Large VC-dimension Classifiers*. In Advances in Neural Information Processing Systems, pages 147–155. Morgan Kaufmann, 1993. (Cité en page 80.)

[Hasenjager 1996] M Hasenjager et H Ritter. *Active learning of the generalized high-low game*. In International Conference on Artificial Neural Networks, pages 501–506. Springer-Verlag, 1996. (Cité en page 19.)

[Heinrichs 2000] A. Heinrichs, D. Koubaroulis, B. Levienaise-obadia, P. Rovidia et J.M. Jolion. *Image Indexing and Content-based Search using Pre-attentive Similarities*. In RIAO 2000, pages 1616–1631, 2000. (Cité en page 16.)

[Hotelling 1933] H. Hotelling. *Analysis of a complex of statistical variables into principal components*. J. Educ. Psych., vol. 24, 1933. (Cité en page 82.)

[Huang 2001] Thomas S. Huang et Xiang Sean Zhou. *Image Retrieval with Relevance Feedback : From Heuristic Weight Adjustment to Optimal Learning Methods*. In Proceeding of International Conference on Image Processing, 2001. (Cité en page 18.)

[Kawanabe 2009] M. Kawanabe, S. Nakajima et A. Binder. *A Procedure of Adaptive Kernel Combination with Kernel-Target Alignment for Object Classification*. In ACM International Conference on Image and Video Retrieval, 2009. (Cité en page 125.)

[Kearns 1988] M.J. Kearns. *Thoughts on hypothesis boosting*. ML class project, 1988. (Cité en page 23.)

[Kivinen 1999] Jyrki Kivinen et Manfred K. Warmuth. *Boosting as entropy projection*. In Proceedings of the twelfth annual conference on Computational learning theory, COLT '99, pages 134–144, New York, NY, USA, 1999. ACM. (Cité en page 41.)

[Kloft 2011] M. Kloft, U. Brefeld, S. Sonnenburg et A. Zien. *Lp-norm Multiple Kernel Learning*. Journal of Machine Learning Research, vol. 12, pages 953–997, Mar 2011. (Cité en pages 94 et 97.)

[Lanckriet 2004] G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui et M. I. Jordan. *Learning the Kernel matrix with semi-definite programming*. International Journal on Machine Learning Research, vol. 5, pages 27–72, 2004. (Cité en pages viii, 92, 93, 94 et 96.)

[Lechervy 2010a] A. Lechervy, P.-H Gosselin et F. Precioso. *Boosting actif pour la recherche interactive d'images*. In Reconnaissance des Formes et Intelligence Artificielle (RFIA2010), page 1, Caen, France, Jan. 2010. (Cité en pages 10 et 53.)

[Lechervy 2010b] A. Lechervy, P h. Gosselin et F. Precioso. *Active Boosting for interactive object retrieval*. In International Conference on Pattern Recognition (ICPR), page 1, Istanbul, Turkey, Aug. 2010. (Cité en pages 10 et 53.)

[Lechervy 2012a] A. Lechervy, P h. Gosselin et F. Precioso. *Boosted Kernel for Image Categorization.* In Multimedia Tools and Applications (MTAP),, 2012. (Cité en pages 10 et 99.)

[Lechervy 2012b] A. Lechervy, P h. Gosselin et F. Precioso. *Boosting kernel combination for multi-class image categorization*. In IEEE International Conference on Image Processing (ICIP), page 1, Orlando, Florida, U.S.A, September 2012. (Cité en pages 10 et 99.)

[Lechervy 2012c] A. Lechervy, P h. Gosselin et F. Precioso. *Linear kernel combination using boosting*. In European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN), page 1, Bruges, Belguim, April 2012. (Cité en pages 10 et 99.)

[Li 2004] X. Li, L. Wang et E. Sung. *Improving AdaBoost for Classfication on Small Training Sample Sets with Active Learning*. In Asian Conference on Computer Vision, Jeju, Korea, January 2004. (Cité en pages 18, 19 et 53.)

[Lindenbaum 2004] M Lindenbaum, S Markovitch et D Rusakov. *Selective sampling for nearest neighbor classifiers*. Machine Learning, pages 54(2) :125–152, February 2004. (Cité en page 19.)

[Lu 2007] Y. Lu, Q. Tian et T.S. Huang. *Interactive Boosting for Image Classification*. In International Conference on Multiple Classifier Systems, pages 315–324, 2007. (Cité en pages 18, 19 et 53.)

[Luenberger 2008] David Luenberger et Ye. Linear and Nonlinear Programming. Springer, third édition, 2008. (Cité en page 36.)

[Luo 1992] Z. Q. Luo et P. Tseng. *On the convergence of the coordinate descent method for convex differentiable minimization*. Journal of Optimization Theory and Applications, vol. 72, no. 1, pages 7–35, January 1992. (Cité en page 36.)

[Mason 2000] L Mason, J Baxter, P.L Bartlett et M Frean. *Functional Gradient Techniques for Combining Hypotheses*. In A.J. Smola, P.L. Bartlett, B. Schölkopf et D. Schuurmans, editeurs, Advances in Large Margin Classifiers, pages 221–246, Cambridge, MA, 2000. MIT Press. (Cité en pages 38 et 40.)

[Meila 2003] M. Meila. *Data centering in feature space*. In International Workshop on Artificial Intelligence and Statistics, 2003. (Cité en page 76.)

[Meir 2003] Ron Meir et Gunnar Rätsch. *An introduction to boosting and leveraging*. pages 118–183, New York, NY, USA, 2003. Springer-Verlag New York, Inc. (Cité en page 34.)

[Müller 1999] Wolfgang Müller, David McG. Squire, Henning Müller et Thierry Pun. *Hunting moving targets : an extension to Bayesian methods in multimedia databases*. In Sethuraman Panchanathan, Shih-Fu Chang et C.-C. Jay Kuo, editeurs, Multimedia Storage and Archiving Systems IV (VV02), volume 3846 of *SPIE Proceedings*, Boston, Massachusetts, USA, September 20–22 1999. (SPIE Symposium on Voice, Video and Data Communications). (Cité en page 18.)

[Najjar 2003] N. Najjar, J.P. Cocquerez et C. Ambroise. *Feature Selection for Semi Supervised Learning Applied to Image Retrieval*. In International Conference on Image Processing, volume 2, pages 559–562, Barcelena, Spain, Sept. 2003. (Cité en page 18.)

[Nilsback 2006] M-E. Nilsback et A. Zisserman. *A Visual Vocabulary for Flower Classification*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, volume 2, pages 1447–1454, 2006. (Cité en page 127.)

[Nilsback 2008] M-E. Nilsback et A. Zisserman. *Automated Flower Classification over a Large Number of Classes*. In Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing, Dec 2008. (Cité en page 131.)

[Nock 2012] Richard Nock, Paolo Piro, Frank Nielsen, Wafa Bel Haj Ali et Michel Barlaud. *Boosting k-NN for Categorization of Natural Scenes*. International Journal of Computer Vision, pages 1–21, Juillet 2012. (Cité en page 22.)

[Oza 2001] Nikunj C. Oza. *Online Ensemble Learning*. PhD thesis, The University of California, Berkeley, CA, Sep 2001. (Cité en pages 50 et 54.)

[Oza 2003] Nikunj C. Oza. *Boosting with averaged weight vectors*. In Proceedings of the 4th international conference on Multiple classifier systems, MCS'03, pages 15–24, Berlin, Heidelberg, 2003. Springer-Verlag. (Cité en page 42.)

[Park 2000] J M Park. *Online learning by active sampling using orthogonal decision support vectors*. In IEEE Workshop on Neural Networks for Signal Processing, volume 77, pages 263–283, December 2000. (Cité en page 19.)

[Pearson 1901] K. Pearson. *On lines and planes of closest fit to systems of points in space*. Philosophical Magazine, vol. 2, no. 6, pages 559–572, 1901. (Cité en page 82.)

[Peng 1999] J. Peng, B. Bhanu et S. Qing. *Probabilistic Feature Relevance Learning for Content-Based Image Retrieval*. Computer Vision and Image Understanding, vol. 75, no. 1-2, pages 150–164, July-August 1999. (Cité en page 18.)

[Picard 2012] David Picard, Nicolas Thome, Matthieu Cord et Alain Rakotomamonjy. *Learning geometric combinations of Gaussian kernels with alternating Quasi-Newton algorithm*. In ESANN 2012, pages 79–84, Bruges, Belgique, Avril 2012. (Cité en page 92.)

[Rakotomamonjy 2008] Alain Rakotomamonjy, Francis Bach, Stéphane Canu et Yves Grandvalet. *Y. : SimpleMKL*. Journal of Machine Learning Research 9, 2008. (Cité en pages 94 et 97.)

[Rätsch 2000] Gunnar Rätsch, Bernhard Schölkopf, Alex J. Smola, Sebastian Mika, Takashi Onoda et Klaus-Robert Müller. *Robust Ensemble Learning for Data Mining*. In Proceedings of the 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining, Current Issues and New Applications, PADKK '00, pages 341–344, London, UK, UK, 2000. Springer-Verlag. (Cité en page 130.)

[Roy 2001] N. Roy et A. McCallum. *Toward optimal active learning through sampling estimation of error reduction*. In International Conference on Machine Learning, pages 441–448, 2001. (Cité en page 19.)

[Rubner 1999] Y. Rubner. *Perceptual Metrics for Image Database Navigation*. PhD thesis, Stanford University, May 1999. (Cité en page 6.)

[Rui 1997] Y. Rui, T. Huang, S. Mehrotra et M. Ortega. *A Relevance Feedback Architecture for Content-based Multimedia Information Retrieval Systems*. In IEEE Workshop on Content-Based Access of Image and Video Libraries, pages 92–89, 1997. (Cité en page 16.)

[Rui 2000] Y. Rui et T.S. Huang. *Optimizing Learning In Image Retrieval*. In IEEE International Conference on Computer Vision and Pattern Recognition, volume 1, pages 236–243, Hilton Head, SC, June 2000. (Cité en page 6.)

[Rätsch 2001a] Gunnar Rätsch. *Robust Boosting via Convex Optimization*. PhD thesis, University of Potsdam, Mathematisch-Naturwissenschaftliche Fakultät, Universität Potsdam, 2001. (Cité en page 35.)

[Rätsch 2001b] Gunnar Rätsch, Takashi Onoda et Klaus-R. Müller. *Soft Margins for AdaBoost*. Mach. Learn., vol. 42, pages 287–320, March 2001. (Cité en pages 35 et 37.)

[Rätsch 2002] G Rätsch et M K Warmuth. *Marginal Boosting*. In Proceedings of the Annual Conferences on Computational Learning Theory, 2002. Slides of the Talk. Copyright by Springer. Extended version submitted to JMLR. (Cité en page 35.)

[Santini 2001] S. Santini, A. Gupta et R. Jain. *Emergent semantics through interaction in Image Databases*. IEEE Transactions on Knowledge and Data Engineering, vol. 13, no. 3, pages 337–351, 2001. (Cité en page 4.)

[Saux 2003] B. Le Saux. *Classification non exclusive et personnalisation par apprentissage : Application à la navigation dans les bases d'images*. PhD thesis, INRIA, 2003. (Cité en page 18.)

[Schapire 1990] R. E. Schapire. *The strength of weak learnability*. In Machine Learning, volume 5(2), pages 197–227, 1990. (Cité en pages 23 et 26.)

[Scholkopf 1999] Bernhard Scholkopf, Alexander Smola et Klaus-Robert Müller. *Kernel principal component analysis*. In Advances in Kernel Methods - Support Vector Learning, pages 327–352. MIT Press, 1999. (Cité en pages 83 et 84.)

[Schölkopf 2002] B Schölkopf et A Smola. Learning with kernels. MIT Press, Cambridge, MA, 2002. (Cité en pages 24 et 81.)

[Shawe-Taylor 2004] John Shawe-Taylor et Nello Cristianini. Kernel methods for pattern analysis. Cambridge University Press, New York, NY, USA, 2004. (Cité en pages 79 et 80.)

[Smeulders 2000] Arnold W. M. Smeulders, Senior Member, Marcel Worring, Simone Santini, Amarnath Gupta et Ramesh Jain. *Content-based image retrieval at the end of the early years*. IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 22, pages 1349–1380, 2000. (Cité en pages 4 et 5.)

[Sonnenburg 2006] Gunnar Sonnenburg Sören dand Rätsch, Christin Schäfer et Bernhard Schölkopf. *Large Scale Multiple Kernel Learning*. Journal of Machine Learning Research, vol. 7, pages 1531–1565, dec 2006. (Cité en pages 94 et 97.)

[Thuy 2008] Nguyen Thuy, B. D. Nguyen et Bischof Horst. *Efficient boosting-based active learning for specific object detection problems*. In International Conference on Computer Vision, Image and Signal Processing, 2008. (Cité en page 54.)

[Tieu 2000] Kinh Tieu et Paul Viola. *Boosting Image Retrieval*. In IEEE International Conference on Computer Vision and Pattern Recognition, pages 228–235, 2000. (Cité en page 22.)

[Tong 2000] S. Tong et D. Koller. *Support vector machine active learning with applications to text classification*. In International Conference on Machine Learning, pages 999–1006, San Francisco, 2000. (Cité en page 19.)

[Tong 2001] S. Tong et D. Koller. *Support vector machine active learning with application to text classification*. International Journal on Machine Learning Research, vol. 2, pages 45–66, November 2001. (Cité en pages 18 et 19.)

[Torralba 2007] Antonio Torralba, Kevin P. Murphy et William T. Freeman. *Sharing Visual Features for Multiclass and Multiview Object Detection*. IEEE Trans. Pattern Anal. Mach. Intell., vol. 29, no. 5, pages 854–869, Mai 2007. (Cité en page 22.)

[Valiant 1984] L. G. Valiant. *A theory of the learnable*. Commun. ACM, vol. 27, pages 1134–1142, November 1984. (Cité en page 22.)

[Vapnik 1982] V. Vapnik. Estimation of dependences based on empirical data. Springer-Verlag, 1982. (Cité en page 101.)

[Vapnik 1995] Vladimir N. Vapnik. The nature of statistical learning theory. Springer-Verlag New York, Inc., New York, NY, USA, 1995. (Cité en pages 24 et 80.)

[Vapnik 1998] V. Vapnik. Statistical learning theory. Wiley-Interscience, New York, 1998. (Cité en page 87.)

[Varma 2009] Manik Varma et Bodla Rakesh Babu. *More generality in efficient multiple kernel learning*. In Proceedings of the 26th Annual International Conference on Machine Learning, ICML '09, pages 1065–1072, New York, NY, USA, 2009. ACM. (Cité en page 92.)

[Vasconcelos 2000] N. Vasconcelos. *Bayesian models for visual information retrieval*. PhD thesis, Massachusetts Institute of Technology, 2000. (Cité en page 18.)

[Veropoulos 1999] K. Veropoulos. *Controlling the Sensivity of Support Vector Machines*. In International Joint Conference on Artificial Intelligence (IJCAI), Stockholm, Sweden, 1999. (Cité en page 90.)

[Vijayanarasimhan 2009] S. Vijayanarasimhan et K. Grauman. *What's it going to cost you ? : Predicting effort vs. informativeness for multi-label image annotations*. In IEEE International Conference on Computer Vision and Pattern Recognition, pages 2270–2277, 2009. (Cité en page 19.)

[Viola 2001] P. Viola et M. Jones. *Rapid object detection using a boosted cascade of simple features*. In IEEE International Conference on Computer Vision and Pattern Recognition, pages 511–518, 2001. (Cité en pages 22 et 46.)

[Wang 2002] L. Wang et K. Luk Chan. *Learning kernel parameters by using class separability measure*. In Advances in Neural Information Processing Systems, 2002. (Cité en page 76.)

[Wolf 2005] L. Wolf et I. Martin. *Robust Boosting for Learning from Few Examples*. In IEEE International Conference on Computer Vision and Pattern Recognition, pages I : 359–364, 2005. (Cité en page 53.)

[Xu 2007] Jun Xu et Hang Li. *AdaRank : a boosting algorithm for information retrieval*. In Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR '07, pages 391–398, New York, NY, USA, 2007. ACM. (Cité en pages 48 et 49.)

[Yan 2007] Rong Yan, Jelena Tesic et John R. Smith. *Model-shared subspace boosting for multi-label classification*. In Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '07, pages 834–843, New York, NY, USA, 2007. ACM. (Cité en page 22.)

[Yan 2010] F. Yan, K. Mikolajczyk, M. Barnard, H. Cai et J. Kittler. *Lp Norm Multiple Kernel Fisher Discriminant Analysis for Object and Image Categorisation*. In IEEE International Conference on Computer Vision and Pattern Recognition, 2010. (Cité en page 130.)

ANNEXE A

Propriétés algébriques autour de l'Alignement du noyau

Ce chapitre est consacré à quelques rappels mathématiques sur des propriétés utilisées dans la suite du document. On utilisera dans ce document, sauf mention contraire, les normes et le produit scalaire de Frobenius comme norme et produit scalaire matriciels.

Les propriétés qui sont montrées pour l'alignement seul sont également valables pour l'alignement centré.

Définition 22. *L'alignement centré de deux matrices* $\mathbf{A}, \mathbf{B}$ *de* $\mathcal{M}_n(\mathbb{R})$ *correspond au réel :*

$$\begin{aligned} \mathcal{A}_H(\mathbf{A}, \mathbf{B}) &= \frac{\langle \overline{\mathbf{A}}, \overline{\mathbf{B}} \rangle}{\|\overline{\mathbf{A}}\| \|\overline{\mathbf{B}}\|} \\ &= \frac{\mathrm{Tr}(\overline{\mathbf{A}}\, \overline{\mathbf{B}}^\top)}{\sqrt{\mathrm{Tr}(\overline{\mathbf{A}}\, \overline{\mathbf{A}}^\top)\, \mathrm{Tr}(\overline{\mathbf{B}}\, \overline{\mathbf{B}}^\top)}} \end{aligned}$$

Avec $\mathbf{H} = Id_{n,n} - \frac{1}{n} 1_{n,n}$ *la matrice de centrage.*

Propriété 25. *Soit deux matrices* $\mathbf{A}, \mathbf{B}$ *de* $\mathcal{M}_n(\mathbb{R})$ *on a :*

$$\begin{aligned} \mathrm{Tr}(\overline{\mathbf{A}}\, \overline{\mathbf{B}}^\top) &= \mathrm{Tr}(\overline{\mathbf{A}}\, \mathbf{B}^\top) && \text{(A.1)} \\ &= \mathrm{Tr}(\mathbf{A} \overline{\mathbf{B}}^\top) && \text{(A.2)} \end{aligned}$$

Démonstration.

$$\begin{aligned} \mathrm{Tr}(\overline{\mathbf{A}}\, \overline{\mathbf{B}}^\top) &= \mathrm{Tr}(\mathbf{HAH}(\mathbf{HBH})^\top) && \text{(A.3)} \\ &= \mathrm{Tr}(\mathbf{HAHB}^\top \mathbf{H}) && \text{(A.4)} \\ &= \mathrm{Tr}(\mathbf{HHAHB}^\top) && \text{(A.5)} \\ &= \mathrm{Tr}(\mathbf{HAHB}^\top) && \text{(A.6)} \\ &= \mathrm{Tr}(\overline{\mathbf{A}} \mathbf{B}^\top) && \text{(A.7)} \end{aligned}$$

Idem pour l'autre égalité. □

Lemme 2. *Pour toute matrice* $\mathbf{A} \in \mathcal{M}_{n,m}(\mathbb{R})$ *,*$\mathbf{B} \in \mathcal{M}_{m,n}(\mathbb{R})$*, on a*

$$\mathrm{Tr}(\mathbf{AB}) = \mathrm{Tr}(\mathbf{BA})$$

Démonstration.

$$\mathrm{Tr}(\mathbf{AB}) = \sum_{i,j} (\mathbf{A} \cdot \mathbf{B}^\top)_{i,j} = \sum_{i,j} (\mathbf{B} \cdot \mathbf{A}^\top)_{i,j} = \mathrm{Tr}(\mathbf{BA})$$

□

Propriété 26. *Pour toute matrice* $\mathbf{A} \in \mathcal{M}_{n,m}(\mathbb{R})$,$\mathbf{B} \in \mathcal{M}_{m,n}(\mathbb{R})$*, on a*

$$\mathcal{A}_H(\mathbf{A}, \mathbf{B}) = \mathcal{A}_H(\mathbf{B}, \mathbf{A})$$

Démonstration.

$$\begin{aligned} \mathcal{A}_H(\mathbf{A}, \mathbf{B}) &= \frac{\langle \overline{\mathbf{A}}, \overline{\mathbf{B}} \rangle}{\|\overline{\mathbf{A}}\| \|\overline{\mathbf{B}}\|} \\ &= \frac{\langle \overline{\mathbf{B}}, \overline{\mathbf{A}} \rangle}{\|\overline{\mathbf{A}}\| \|\overline{\mathbf{B}}\|} \\ &= \mathcal{A}_H(\mathbf{B}, \mathbf{A}) \end{aligned}$$

□

Propriété 27. *Pour toute matrice* $\mathbf{A} \in \mathcal{M}_{n,m}(\mathbb{R})$*, on a*

$$\mathcal{A}(\mathbf{A}, \mathbf{A}) = 1$$

Démonstration.

$$\mathcal{A}(\mathbf{A}, \mathbf{A}) = \frac{\mathrm{Tr}(\mathbf{AA}^\top)}{\sqrt{\mathrm{Tr}(\mathbf{AA}^\top)} \sqrt{\mathrm{Tr}(\mathbf{AA}^\top)}}$$

□

Propriété 28. *Pour toute matrice* $\mathbf{A} \in \mathcal{M}_{n,m}(\mathbb{R})$,$\mathbf{B} \in \mathcal{M}_{m,n}(\mathbb{R})$*, on a*

$$\mathcal{A}(\mathbf{A}, -\mathbf{B}) = -\mathcal{A}(\mathbf{A}, \mathbf{B})$$

Démonstration.

$$\begin{aligned} \mathcal{A}(\mathbf{A}, -\mathbf{B}) &= \frac{\mathrm{Tr}(-\mathbf{AB}^\top)}{\|\mathbf{A}\| \|\mathbf{B}\|} \\ &= -\frac{\mathrm{Tr}(\mathbf{AB}^\top)}{\|\mathbf{A}\| \|\mathbf{B}\|} \\ &= -\mathcal{A}(\mathbf{A}, \mathbf{B}) \end{aligned}$$

□

Propriété 29. *Pour toute matrice* $\mathbf{A} \in \mathcal{M}_{n,m}(\mathbb{R})$,$\mathbf{B} \in \mathcal{M}_{m,n}(\mathbb{R})$ *et tout réel* λ *on a*

$$\forall \lambda > 0 \; \mathcal{A}(\lambda \mathbf{A}, \mathbf{B}) = \mathcal{A}(\mathbf{A}, \mathbf{B})$$

Démonstration.

$$\begin{aligned}\mathcal{A}(\lambda\mathbf{A},\mathbf{B}) &= \frac{\mathrm{Tr}(\lambda\mathbf{A}\mathbf{B}^\top)}{\|\lambda\mathbf{A}\|\|\mathbf{B}\|}\\ &= \frac{\lambda\,\mathrm{Tr}(\mathbf{A}\mathbf{B}^\top)}{|\lambda|\|\mathbf{A}\|\|\mathbf{B}\|}\\ &= \mathcal{A}(\mathbf{A},\mathbf{B})\end{aligned}$$

□

Lemme 3. *Pour toute matrice* $\mathbf{A}\in\mathcal{M}_{n,m}(\mathbb{R})$,$\mathbf{B}\in\mathcal{M}_{m,n}(\mathbb{R})$*, on a*

$$\mathcal{A}(\mathbf{A},\mathbf{B}+\mathbf{C}) = \frac{\|\mathbf{B}\|\mathcal{A}(\mathbf{A},\mathbf{B})+\|\mathbf{C}\|\mathcal{A}(\mathbf{A},\mathbf{C})}{\|\mathbf{B}+\mathbf{C}\|}$$

Démonstration.

$$\begin{aligned}\mathcal{A}(\mathbf{A},\mathbf{B}+\mathbf{C}) &= \frac{\mathrm{Tr}(\mathbf{A}(\mathbf{B}+\mathbf{C})^\top)}{\|\mathbf{A}\|\|\mathbf{B}+\mathbf{C}\|}\\ &= \frac{\mathrm{Tr}(\mathbf{A}\mathbf{B}^\top)+\mathrm{Tr}(\mathbf{A}\mathbf{C})^\top}{\|\mathbf{A}\|\|\mathbf{B}+\mathbf{C}\|}\\ &= \frac{\|\mathbf{B}\|\,\mathrm{Tr}(\mathbf{A}\mathbf{B}^\top)}{\|\mathbf{A}\|\|\mathbf{B}\|\|\mathbf{B}+\mathbf{C}\|}+\frac{\|\mathbf{C}\|\,\mathrm{Tr}(\mathbf{A}\mathbf{C})^\top}{\|\mathbf{A}\|\|\mathbf{C}\|\|\mathbf{B}+\mathbf{C}\|}\\ &= \frac{\|\mathbf{B}\|\mathcal{A}(\mathbf{A},\mathbf{B})+\|\mathbf{C}\|\mathcal{A}(\mathbf{A},\mathbf{C})}{\|\mathbf{B}+\mathbf{C}\|}\end{aligned}$$

□

Propriété 30.

$$\forall t\ \alpha_t>0\ \mathcal{A}(\mathbf{A},\sum_t\alpha_t\mathbf{B}_t) = \frac{\sum_t\alpha_t\|\mathbf{B}_t\|\mathcal{A}(\mathbf{A},\mathbf{B}_t)}{\|\sum_t\alpha_t\mathbf{B}_t\|}$$

Démonstration. Montrons par récurrence, supposons la propriété vrai pour un certain T on a :

$$\begin{aligned}\mathcal{A}(\mathbf{A},\sum_{t=1}^{T+1}\alpha_t\mathbf{B}_t) &= \mathcal{A}(\mathbf{A},\sum_{t=1}^{T}\alpha_t\mathbf{B}_t+\alpha_{T+1}\mathbf{B}_{T+1})\\ &= \frac{\|\sum_{t=1}^{T}\alpha_t\mathbf{B}_t\|\mathcal{A}(\mathbf{A},\sum_{t=1}^{T}\alpha_t\mathbf{B}_t)+\|\alpha_{T+1}\mathbf{B}_{T+1}\|\mathcal{A}(\mathbf{A},\alpha_{T+1}\mathbf{B}_{T+1})}{\|\sum_{t=1}^{T+1}\alpha_t\mathbf{B}_t\|}\\ &= \frac{\|\sum_t^T\alpha_t\mathbf{B}_t\|\sum_t^T\alpha_t\|\mathbf{B}_t\|\frac{\mathcal{A}(\mathbf{A},\mathbf{B}_t)}{\|\sum_t^T\alpha_t\mathbf{B}_t\|}+\alpha_{T+1}\|\mathbf{B}_{T+1}\|\mathcal{A}(\mathbf{A},\alpha_{T+1}\mathbf{B}_{T+1})}{\|\sum_t^{T+1}\alpha_t\mathbf{B}_t\|}\\ &= \frac{\sum_t^{T+1}\alpha_t\|\mathbf{B}_t\|\mathcal{A}(\mathbf{A},\mathbf{B}_t)}{\|\sum_t^{T+1}\alpha_t\mathbf{B}_t\|}\end{aligned}$$

La propriété est vrai pour $t=1$:

$$\mathcal{A}(\mathbf{A},\alpha_1\mathbf{B}_1) = \frac{\alpha_1\|\mathbf{B}_1\|\mathcal{A}(\mathbf{A},\mathbf{B}_1)}{\|\alpha_t\mathbf{B}_1\|}$$

□

Lemme 4. *Pour toute matrice* $\mathbf{A} \in \mathcal{M}_{n,m}(\mathbb{R})$,$\mathbf{B} \in \mathcal{M}_{m,n}(\mathbb{R})$*, on a*

$$\mathcal{A}(\mathbf{A},\mathbf{B})^2 = 1 \Longleftrightarrow \exists \lambda \neq 0 \ \mathbf{A} = \lambda \mathbf{B}$$

Démonstration. D'après Cauchy-Schwarz on a :

$$\forall x, y \ \langle x, y \rangle^2 \leq \|x\|^2 \|y\|^2$$

avec égalité si et seulement si x et y sont colinéaires.

La propriété correspond au cas d'égalité. □

Propriété 31. *Pour toute matrice* $\mathbf{A} \in \mathcal{M}_{n,m}(\mathbb{R})$,$\mathbf{B} \in \mathcal{M}_{m,n}(\mathbb{R})$*, on a*

$$\mathcal{A}(\mathbf{A},\mathbf{B}) = 1 \Longleftrightarrow \exists \lambda > 0 \ \mathbf{A} = \lambda \mathbf{B}$$

Démonstration. Si $\lambda > 0$ et $\mathbf{A} = \lambda \mathbf{B}$ on a :

$$\mathcal{A}(\mathbf{A},\mathbf{B}) = \mathcal{A}(\mathbf{A},\lambda\mathbf{B}) = \mathcal{A}(\mathbf{A},\mathbf{A}) = 1$$

Réciproquement si $\mathcal{A}(\mathbf{A},\mathbf{B}) = 1$ alors $\mathcal{A}(\mathbf{A},\mathbf{B})^2 = 1$, d'après la propriété précédente on a :

$$\exists \lambda \neq 0 \ \mathbf{A} = \lambda \mathbf{B}$$

Par l'absurde si $\lambda < 0$ on a :

$$\begin{aligned} \mathcal{A}(\mathbf{A},\mathbf{B}) &= \mathcal{A}(\lambda\mathbf{B},\mathbf{B}) \\ &= \mathcal{A}(-|\lambda|\mathbf{B},\mathbf{B}) \\ &= -\mathcal{A}(\mathbf{B},\mathbf{B}) \\ &= -1 \end{aligned}$$

□

Propriété 32. *Pour toute matrice* $\mathbf{M} \in \mathcal{M}_{n,m}(\mathbb{R})$*,* $\mathbf{M}\mathbf{M}^T$ *est semi-définie positive.*

Démonstration.

$$\begin{aligned} \forall x \in \mathbb{R}^n \qquad & 0 \leq \|\mathbf{M}^\top x\|^2 \\ \Longleftrightarrow \quad & 0 \leq (\mathbf{M}^\top x)^\top (\mathbf{M}^\top x) \\ \Longleftrightarrow \quad & 0 \leq x^\top \mathbf{M}\mathbf{M}^\top x \\ \Longleftrightarrow \quad & \mathbf{M}\mathbf{M}^\top \text{ semi définie positive} \end{aligned}$$

□

Propriété 33. *La trace d'une matrice diagonalisable est la somme de ses valeurs propres.*

Démonstration. $\mathbf{M}$ est diagonalisable $\implies \exists \mathbf{P}\ \mathbf{M} = \mathbf{PDP}^{-}1$

$$\mathrm{Tr}(\mathbf{M}) = \mathrm{Tr}(\mathbf{PDP}^{-1}) = \mathrm{Tr}(\mathbf{P}^{-1}\mathbf{PD}) = \mathrm{Tr}(\mathbf{D}) = \sum_i \lambda_i$$

□

Lemme 5. *La trace d'une matrice* $\mathbf{MM}^\top$ *avec* $\mathbf{M} \in \mathcal{M}_{n,m}(\mathbb{R})$ *est la somme de ses valeurs propres.*

Démonstration. $\mathbf{MM}^\top$ est une matrice réel symétrique donc d'après le théorème spectral, $\mathbf{MM}^\top$ est diagonalisable. □

Apprentissage interactif et multi-classes pour la détection de concepts sémantiques dans des données multimédia

Résumé : Récemment les techniques d'apprentissage automatique ont montré leurs capacités à identifier des catégories d'images à partir de descripteurs extraits de caractéristiques visuelles des images. Face à la croissance du nombre d'images et du nombre de catégories à traiter, plusieurs techniques ont été proposées pour réduire à la fois le coût calculatoire des méthodes et l'investissement humain en terme de supervision. Dans cette thèse nous proposons deux méthodes qui ont pour objectif de traiter un grand nombre d'images et de catégories. Nous proposons, tout d'abord, une solution reposant sur le concept de recherche interactive. Le protocole de recherche interactive propose d'établir un « dialogue » entre le système d'apprentissage et l'utilisateur afin de minimiser l'effort d'annotation. Nous avons voulu proposer dans ces travaux une solution de recherche interactive adaptée aux méthodes de boosting. Dans un second temps, nous nous sommes consacrés plus particulièrement aux méthodes à noyaux dans un contexte d'apprentissage plus classique. Ces méthodes ont montré de très bons résultats mais le choix de la fonction noyau et son réglage reste un enjeu important. Dans ces travaux, nous avons mis en place une nouvelle méthode d'apprentissage de fonction noyau multi-classes pour la classification de grandes bases d'images. Nous avons choisi d'utiliser un modèle inspiré des méthodes de boosting pour créer un noyau fort à partir d'une combinaison de noyaux plus faibles. Nous utilisons la dualité entre fonction noyau et espace induit pour construit un nouvel espace de représentation des données plus adapté à la catégorisation. L'idée de notre méthode est de construire de manière optimale ce nouvel espace de représentation afin qu'il permette l'apprentissage d'un nouveau classifieur plus rapide et de meilleure qualité. Chaque donnée multimédia sera alors représentée dans cette espace sémantique en lieu et place de sa représentation visuelle. Ces travaux se basent sur des fondements mathématiques et font l'objet d'expériences montrant leur intérêt pratique par comparaison avec les méthodes les plus récentes de la littérature.

Mots clés : Classification, Boosting, Kernel, SVM, MKL

Interactive and multi-class Learning to detect semantic concepts in the multimedia data

Abstract : Recent machine learning techniques have demonstrated their capability for identifying image categories using image features. Among these techniques, Support Vector Machines (SVM) present the best results, particularly when they are associated with a kernel function. However, nowadays image categorization task is very challenging owing to the size of benchmark datasets and the number of categories to be classified. In such a context, lot of effort has to be put in the design of the kernel functions and underlying high-level features. In this thesis, we propose a new method to learn a kernel function for image categorization in large image databases. Our learning method is made of two steps : first, a kernel is built and semantic features are deduced ; then each class is learn thanks to a standard SVM. We adopt a Boosting framework to design and combine weak kernel functions targeting an ideal kernel. We propose a new iterative algorithm inspired from Boosting, to create a strong kernel. The weak kernels are learnt thanks to the duality between the kernel space and the semantic feature space. We show that our method builds mapping functions which turn the initial input space to a new feature space where categories are better classified. Furthermore, our algorithm benefits from Boosting process to learn this kernel with a complexity linear with the size of the training set. Experiments are carried out on popular benchmarks and databases to show the properties and behavior of the proposed method. On the PASCAL VOC2006 database, we compare our method to simple early fusion, and on the Oxford Flowers databases we show that our method outperforms the best MKL techniques of the literature.

Keywords : Classification, Boosting, Kernel, SVM, MKL