ANNÉE 2014

Université de Nationale de Hanoi - Agence universitaire de la Francophonie
Institut de la Francophonie pour l'Informatique

Algorithme parallèle de Descente de gradient stochastique multi-classes pour la classification d'images

---

Réalisé par
NGUYEN Quoc Khai

Sous la supervision de
DO Thanh Nghi
PHAM Nguyen Khang
Professeurs de l'Université de Cantho

HO Tuong Vinh
Professeurs de l'Institut de la Francophonie pour l'Informatique

---

Stage du Master 2 réalisée à la laboratoire de traitement intelligent des informations de la faculté des technologies de l'information et de la communication, Université de Cantho

Université de Nationale de Hanoi - Agence universitaire de la Francophonie

Institut de la Francophonie pour l'Informatique

Stage du Master 2 réalisée à la laboratoire de traitement intelligent des informations de la faculté des technologies de l'information et de la communication, Université de Cantho.

Réalisé par
NGUYEN Quoc Khai

Sous la supervision de
DO Thanh Nghi
PHAM Nguyen Khang
Professeurs de l'Université de Cantho

HO Tuong Vinh
Professeurs de l'Institut de la Francophonie pour l'Informatique

Table des matières

1 Introduction générale **3**
1.1 Introduction 3
1.2 Description par chapitre 3

2 Extraction des caractéristiques **5**
2.1 Introduction 5
2.2 Description locale des images 6
2.3 Méthode SIFT (Scale-invariant feature transform) 7
2.3.1 Introduction 7
2.3.2 Détection d'extrema dans l'espace des échelles 8
2.3.3 Localisation précise de points d'intérêt 10
2.3.4 Assignation d'orientation 10
2.3.5 Descripteur de point d'intérêt 11
2.4 Méthode BoW (Bag of word) 12

3 Apprentissage automatique **14**
3.1 Introduction 14
3.2 Méthode SVM (Support Vector Machine) 14
3.3 Méthode SVM avec SGD (Stochastic gradient descent) 17
3.4 Méthode MC-SGD (Multi Class - Stochastic gradient descent) 17

4 Implémentation **19**
4.1 Introduction 19
4.2 Représentation par des descripteurs et méthode sac de mots . 19
4.3 Apprentissage automatique 20
4.3.1 Descente de gradient stochastique (SGD) 20
4.3.2 Descente de gradient stochastique pour multi-classe (MC-SGD) 21
4.3.3 Parallélisation de MC-SGD 23

5 Résultat obtenue **25**
5.1 Introduction 25
5.2 Méthode SGD-SVM 25
5.3 Méthode MC-SGD 27
5.4 Classification d'images avec MC-SGD 29

6 Conclusion et perspectives **31**

References **32**

Table des figures

2.1 Différence de Gausienne [15] 9
2.2 [15] Le maxima et le minima des images de différence de gaussienne sont détectés en comparant une pixel (marqué X) à ses 26 voisins dans les régions de 3x3 aux échelles actuels et adjacents (marqué avec des cercles). 10
2.3 Illustration de la construction de l'histogramme des orientations 11
2.4 Construction d'un descripteur SIFT 12
2.5 Model de BOW 13

3.1 Classification linéaire 15
3.2 L'hyperplan optimal 15

4.1 Problème de multi-classes 22
4.2 Problème de multi-classes 22

5.1 Comparaison de la vitesse entre LIBSVM et SGD binaire . . . 26

Remerciement

A lot of people helped me.

Résumé

Dans ce projet, nous avons étudié les problèmes concernant la classification d'images et développé un algorithme parallèle multi-classes basé sur la descente de gradient stochastique. Dans un premier temps, nous avons étudié la représentation des images par descripteurs locaux SIFT (Scalable Invariant Feature Transform) [11]. D'abord, on extrait des SIFT à partir des images. À fin de cette étape, nous obtenons un ensemble des SIFT pour chaque image. L'étape suivante consiste à construire un vocabulaire visuel en appliquant un algorithme de clustering (e.g. k-means) sur un ensemble des SIFT. Un cluster correspond à un mot visuel. Enfin, une image se représente par un histogramme des mots visuels. Cette approche s'inspire au modèle sac-de-mots largement utilisé dans l'analyse des données textuelles. Dans un second temps, nous nous concentrons sur le problème d'apprentissage automatique basé sur la descente de gradient stochastique [3]. Pour améliorer la vitesse de l'algorithme sur des machines de multi-cœurs, nous avons aussi parallélisé cet algorithme en utilisant la libraire OpenMP.

Chapitre 1

Introduction générale

1.1 Introduction

La classification ou la catégorie des images sont importantes pour accéder à l'information visuelle au niveau d'objets, qui consiste à étiqueter automatiquement des images en catégories prédéfinies. Ces méthodes sont largement utilisées tels que : la reconnaissance des scènes naturelles, la reconnaissance des chiffres sur des chèques, la reconnaissance des codes postaux pour la classification automatique des courriers, la reconnaissance des visages pour l'authentification, etc.

Dans ce domaine, le résultat de classification n'est pas très exact. Surtout, la vitesse d'apprendre des méthode actuelle est base. Fait face de ce problème, on a besoin d'améliorer la vitesse des méthodes ou développer une autre méthode qui est moins compliqué et qui donne le résultat de classification acceptable. La méthode Descente de Gradient de Stochastique (SGD) est une bon choix pour ce problème. Donc, ce stage fait le point sur la méthode SGD, MC-SGD et sa version parallèle.

1.2 Description par chapitre

Avant de parler de notre travail, dans premier temps, nous allons présenter la théorie de base des méthodes utilisées. Tout d'abord, nous allons présenter la méthode SIFT et la méthode Sac de Mots dans le chapitre 2. Ensuite, nous allons présenter l'étape d'apprentissage automatique qui se compose la méthode SVM standard et une version de SVM avec SGD dans le chapitre 3. Dans ce chapitre, nous parlerons aussi des façons pour résoudre le problème de multi-classes avec un classificateur de 2 classes. Dans le second temps,

nous présenterons notre implémentation dans le chapitre 4. Pour le chapitre 5, nous présenterons le résultat obtenue et l'analyserons en détaillé. A fin de ce rapport, nous terminerons avec la conclusion et perspective dans le chapitre 6.

Chapitre 2

Extraction des caractéristiques visuelles

2.1 Introduction

Définition 1. ***Caractéristique visuelle***
Une caractéristique d'une image est définie comme une abstraction des informations visuelles de l'image qui sont sélectionnée pour des tâches de calcul reliées à une certaine application (par ex : classification d'images, recherche d'images).

Les caractéristiques sont extraites soit globalement sur une image entière, soit sur une petite groupe de pixel (une région) d'une image. Le résultat d'une étape d'extraction de caractéristiques (globales ou locales) est appelé *descripteur de caractéristiques.*

Définition 2. ***Descripteur de caractéristiques***
Nous appelons la description mathématique d'une image ou une région locale de l'image après une étape d'extraction de caractéristiques sont ***descripteur de caractéristiques***

Les descripteurs se présentent normalement sous forme d'une vecteur dans un espace vectoriel, $\mathbb{R}^D$, appelé *l'espace de caractéristiques.*

Dans le cas d'une extraction globale, on récupère une seule descripteur par image tandis qu'une description locale permet d'obtenir d'un ensemble de descripteur locaux pour une image.

Jusqu'à maintenant, les cherches se basent plusieurs types de caractéristiques pour la classification ou la reconnaissance d'images. Nous pouvons lister

quelques types de caractéristiques les plus couramment utilisées pour calculer des descripteur : *la couleur, la texture, la forme, les points d'intérêt et les relations spatiales* qui sont décrit dans [6].

2.2 Description locale des images

Afin d'obtenir des descripteur locaux à partir une image, on commence par extraire des régions. La façon la plus simple est d'utiliser une *partition* qui découpe l'image en rectangles ou en cercles de même taille. Une telle partition simple ne génère pas de région perceptuelle-ment significatives mais c'est une manière simple d'obtenir des caractéristiques globales de l'image avec une résolution plus fine. Dans la lecture, nous trouvons que les deux approches les plus utilisées pour localiser les région d'intérêt dans l'image : l'une fournit des régions qui se chevauchent (détection des points d'intérêt) et l'autre segmente l'image en région sans intersection (segmentation d'image)[6]. La première approche est efficace pour la classification d'images, donc, dans cette section nous décrivons la première approche.

Détection des points d'intérêt
Les points d'intérêt sont traditionnellement utilisés pour la *stéréo vision* mais sont utilisés aussi dans la classification d'images. Ils sont déterminés de manière telle qu'un point trouvé dans une image sera aussi trouvé dans une autre image qui diffère légèrement de la première. La signification de tels points spéciaux est due à leur représentation compacte des régions importantes de l'image qui conduit à une indexation efficace, et à leur pouvoir discriminant surtout dans la recherche d'objets.

Un des premiers travaux sur ce sujet [7] utilise un détecteur de Harris [8] pour localiser des points d'intérêt invariants à la rotation. Dans [9], les auteurs montrent que les descripteurs ne peuvent pas être invariants au changement d'échelle si les points d'intérêt extraits ne sont pas invariants eux même au changement d'échelle. Par conséquent, plusieurs détecteurs ont été proposé pour obtenir l'invariance au changement d'échelle des points d'intérêt [10, 11, 12, 15]. La sélection automatique de l'échelle est effectuer en choisissant les *extrema* d'une fonction de l'échelle (par ex. laplacien normalisé, différence de gaussiennes).

Caractérisation des points d'intérêt
Après avoir détecté des points d'intérêt, pour les utiliser, il faut caractériser

la région autour de ces points. La caractérisation d'un point d'intérêt est calculée, à une échelle choisie, sur la région autour de ce point. Différents descripteurs ont été proposés dans la littérature : *Shape context* [14], *Scale Invariant Feature Transform (SIFT)* [15], *PCA-SIFT* [16], *Gradient Location and Orientation Histogram (GLOH)* [17]. Parmi des descripteurs listés, le descripteur SIFT est le plus utilisé [6]. Dans ce travail, nous concentrerons au descripteur SIFT, donc, nous décrivons ce descripteur et cette méthode dans la partie ci-dessous.

2.3 Méthode SIFT (Scale-invariant feature transform)

2.3.1 Introduction

Dans la lecture, nous trouvons qu'on traduit en français "transformation de caractéristiques visuelles invariante à l'échelle". SIFT est une méthode utilisé dans le domaine de la vision par ordinateur pour détecter et identifier les éléments similaires entre différentes images numériques (éléments de paysages, objets, personnes, etc.). Cette méthode a été développé en 1999 par le chercheur David Lowe [11].

L'étape fondamentale de la méthode SIFT consiste à calculer les *descripteurs SIFT* des images à étudier. Il s'agit d'informations numériques dérivées de l'analyse locale d'une image et qui caractérisent le contenu visuel de cette image de la façon la plus indépendante possible de l'échelle, du cadrage, de l'angle d'observation et de l'exposition (luminosité) [11].

La méthode proposée par Lowe comprend deux parties [15] :

1. un algorithme de détection de caractéristiques et de calcul de descripteurs
2. un algorithme de mise en correspondance proprement dit

De ces deux aspects, le premier est celui qui a le plus assuré la popularité de la méthode [18]. La deuxième permet d'utiliser le résultat de la première partie pour l'usage de la méthode. Dans notre travail, nous utilisons la méthode SIFT comme un résultat de base. C'est à dire, nous n'utilisons que la première partie de la méthode.

La première partie, partie de détection de caractéristiques et de calculer des descripteurs comprend 4 étapes principales [11, 15] :

1. Détection d'extrema dans l'espace des échelles.
2. Localisation précise de points d'intérêt
3. Assignation d'orientation
4. Descripteur de point d'intérêt

Nous décrivons tout de suite pas à pas ces étapes.

2.3.2 Détection d'extrema dans l'espace des échelles

La détection s'effectue dans un espace discret (espace des échelles ou *scale space* en anglais) qui comporte trois dimensions : les coordonnées cartésiennes x et y et le facteur d'échelle σ. Le gradient de facteur d'échelle σ (noté L) est le résultat de la convolution d'une image I par un filtre gaussien G de paramètre d'échelle σ, soit [15] :

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{2.1}$$

Et

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2}$$

Cette convolution a pour effet de lisser l'image originale I de telle sorte que les détails trop petits, c'est-à-dire de rayon inférieur à [1], sont estompés. Par conséquent, la détection des objets de dimension approximativement égale à σ se fait en étudiant l'image appelée différences de gaussiennes (en anglais difference of gaussians, DoG) définie comme suit :

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma) \tag{2.2}$$

Ou

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y)$$

où k est un paramètre fixe de l'algorithme qui dépend de la finesse de la discrétisation de l'espace des échelles voulue [15].

[1] Ici comme dans la littérature scientifique en général, le facteur d'échelle – paramètre du filtre gaussien σ – est assimilé à une distance en pixels sur l'image, que l'on pourrait appeler rayon associé r. En fait, ils sont proportionnels ($r = \alpha\sigma$), avec un facteur α qui varie généralement entre 3 et 4 selon les auteurs. Il est tout simplement lié au nombre de coefficients au-delà duquel les valeurs de la gaussienne deviennent négligeables.

Pour chaque octave de l'espace des échelle, l'image initiale répétée est fait la convolution avec gaussiennes pour produire l'ensemble des images de l'espace des échelle (image ci-dessous, à gauche). Images gaussiennes adjacentes sont soustraites pour produire les images de différence de gaussienne sur la droite. Après chaque octave, l'image Gaussian est sous-échantillonnée par un facteur de 2, et le processus est répété pour toutes les échelle.

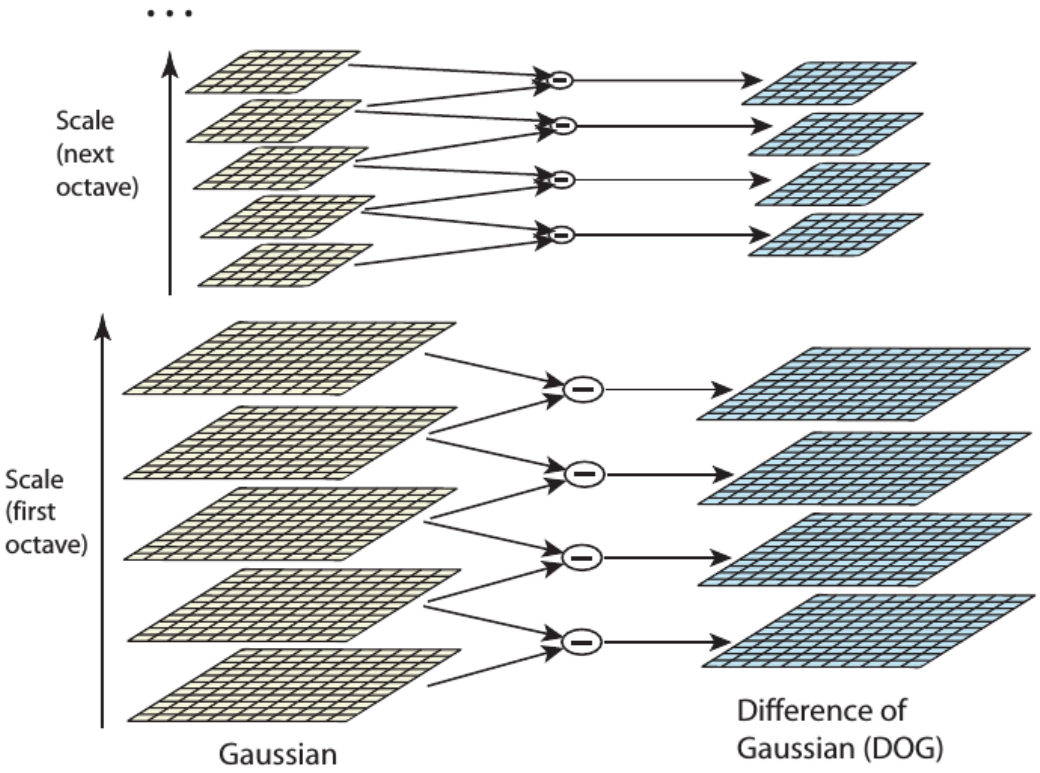


FIGURE 2.1 – Différence de Gausienne [15]

Un point d'intérêt candidat (x, y, σ) est défini comme un point où un extremum du DoG est atteint par rapport à ses voisins immédiats, c'est-à-dire sur l'ensemble contenant 26 autres points défini par :

$$\{D(x + \delta_x, y + \delta_y, s\sigma), \delta_x \in \{-1, 0, 1\}, \delta_y \in \{-1, 0, 1\}, s \in \{k^{-1}, 1, k\}\}$$

On peut voir l'image ci-dessous pour être facile à comprendre

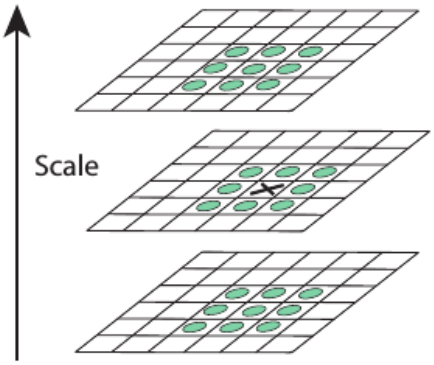


FIGURE 2.2 – [15] Le maxima et le minima des images de différence de gaussienne sont détectés en comparant une pixel (marqué X) à ses 26 voisins dans les régions de 3x3 aux échelles actuels et adjacents (marqué avec des cercles).

2.3.3 Localisation précise de points d'intérêt

L'étape de détection d'extremums produit en général un grand nombre de points-clés candidats, dont certains sont instables. De plus, leur localisation, en particulier aux échelles les plus grandes (autrement dit dans les octaves supérieures de la pyramide où la résolution est plus faible) reste approximative. De ce fait, des traitements supplémentaires sont appliqués, pour un objectif double : d'une part, reconverger la position des points pour améliorer la précision sur x, y et σ, d'autre part, éliminer les points de faible contraste ou situés sur des arêtes de contour à faible courbure et donc susceptibles de "glisser" facilement.

2.3.4 Assignation d'orientation

L'étape d'assignation d'orientation consiste à attribuer à chaque point-clé une ou plusieurs orientations déterminées localement sur l'image à partir de la direction des gradients dans un voisinage autour du point. Dans la mesure où les descripteurs sont calculés relativement à ces orientations, cette étape est essentielle pour garantir l'invariance de ceux-ci à la rotation : les mêmes descripteurs doivent pouvoir être obtenus à partir d'une même image, quelle qu'en soit l'orientation[15].

Pour un point-clé donné (x_0, y_0, σ_0), le calcul s'effectue sur $L(x, y, \sigma_0)$, à savoir le gradient de la pyramide dont le paramètre est le plus proche du facteur d'échelle du point. De cette façon, le calcul est également invariant à l'échelle. À chaque position dans un voisinage du point-clé, on estime le

gradient par différences finies symétriques, puis son amplitude (c.-à-d. sa norme) $m(x, y)$, et son orientation $\theta(x, y)$ [15] :

$$m\left(x,y\right)=\sqrt{\left(L\left(x+1,y\right)-L\left(x-1,y\right)\right)^{2}+\left(L\left(x,y+1\right)-L\left(x,y-1\right)\right)^{2}} \tag{2.3}$$

$$\theta\left(x,y\right)=\tan^{-1}\left(\frac{L\left(x,y+1\right)-L\left(x,y-1\right)}{L\left(x+1,y\right)-L\left(x-1,y\right)}\right)\forall\left(x,y\right)\ \text{dans un voisinage de}\ \left(x_0,y_0\right) \tag{2.4}$$

Un histogramme des orientations sur le voisinage est réalisé avec 36 intervalles, couvrant chacun 10 degrés d'angle. L'histogramme est doublement pondéré : d'une part, par une fenêtre circulaire gaussienne de paramètre égal à 1,5 fois le facteur d'échelle du point-clé σ_0, d'autre part, par l'amplitude de chaque point. Les pics dans cet histogramme correspondent aux orientations dominantes. Toutes les orientations dominantes permettant d'atteindre au moins 80% de la valeur maximale sont prises en considération, ce qui provoque si nécessaire la création de points-clés supplémentaires ne différant que par leur orientation principale[15].

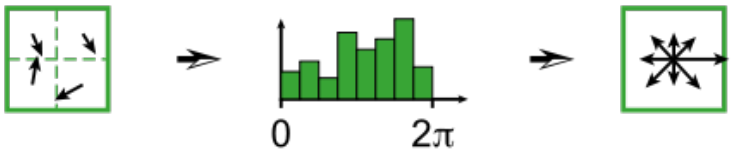


FIGURE 2.3 – Illustration de la construction de l'histogramme des orientations

À l'issue de cette étape, un point-clé est donc défini par quatre paramètres (x, y, σ, θ). Il est à noter qu'il est parfaitement possible qu'il y ait sur une même image plusieurs points-clés qui ne différent que par un seul de ces quatre paramètres (le facteur d'échelle ou l'orientation, par exemple).

2.3.5 Descripteur de point d'intérêt

Une fois les points-clés, associés à des facteurs d'échelles et à des orientations, détectés et leur invariance aux changements d'échelles et aux rotations assurée, arrive l'étape de calcul des vecteurs descripteurs, traduisant numériquement chacun de ces points-clés. À cette occasion, des traitements supplémentaires vont permettre d'assurer un surcroît de pouvoir discriminant en rendant les descripteurs invariants à d'autres transformations telles

que la luminosité, le changement de point de vue 3D, etc. Cette étape est réalisée sur l'image lissée avec le paramètre de facteur d'échelle le plus proche de celui du point-clé considéré[15].

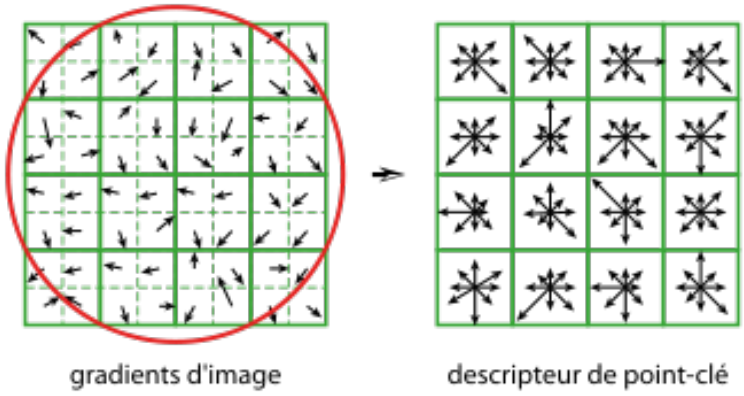


FIGURE 2.4 – Construction d'un descripteur SIFT

Autour de ce point, on commence par modifier le système de coordonnées local pour garantir l'invariance à la rotation, en utilisant une rotation d'angle égal à l'orientation du point-clé, mais de sens opposé. On considère ensuite, toujours autour du point-clé, une région de $16x16$ pixels, subdivisée en $4x4$ zones de $4x4$ pixels chacune. Sur chaque zone est calculé un histogramme des orientations comportant 8 intervalles. En chaque point de la zone, l'orientation et l'amplitude du gradient sont calculés comme précédemment. L'orientation détermine l'intervalle à incrémenter dans l'histogramme, ce qui se fait avec une double pondération par l'amplitude et par une fenêtre gaussienne centrée sur le point clé, de paramètre égal à 1,5 fois le facteur d'échelle du point-clé[15].

Ensuite, les 16 histogrammes à 8 intervalles chacun sont concaténés et normalisés. Dans le but de diminuer la sensibilité du descripteur aux changements de luminosité, les valeurs sont plafonnées à 0,2 et l'histogramme est de nouveau normalisé, pour finalement fournir le descripteur SIFT du point-clé, de dimension 128.

2.4 Méthode BoW (Bag of word)

La représentation par sac de mots *(ou bag of words en anglais)* est une description de document (texte, image, ...) très utilisée en recherche d'information. Spécialement, dans la classification d'images, cette méthode est

largement utilisée après l'étape d'extraction des descripteurs. Dans DoW, les méthodes de cluster sont utilisées pour grouper des descripteurs. Actuellement, la méthode K-moyenne (Kmeans)[19] est beaucoup utilisée pour grouper des des descripteurs SIFT aux clusters, chaque cluster est considère comme un mot visuel, l'ensemble de mots visuels jouent le rôle d'un dictionnaire de mots visuels. Ensuite, BoW met chaque descripteur de chaque image au cluster qui est le plus proche(se base sur la distance entre chaque descripteur et sa future cluster). Suite, chaque image est décrite comme un histogramme des mots. Cette méthode est décrite comme l'image ci-dessous.

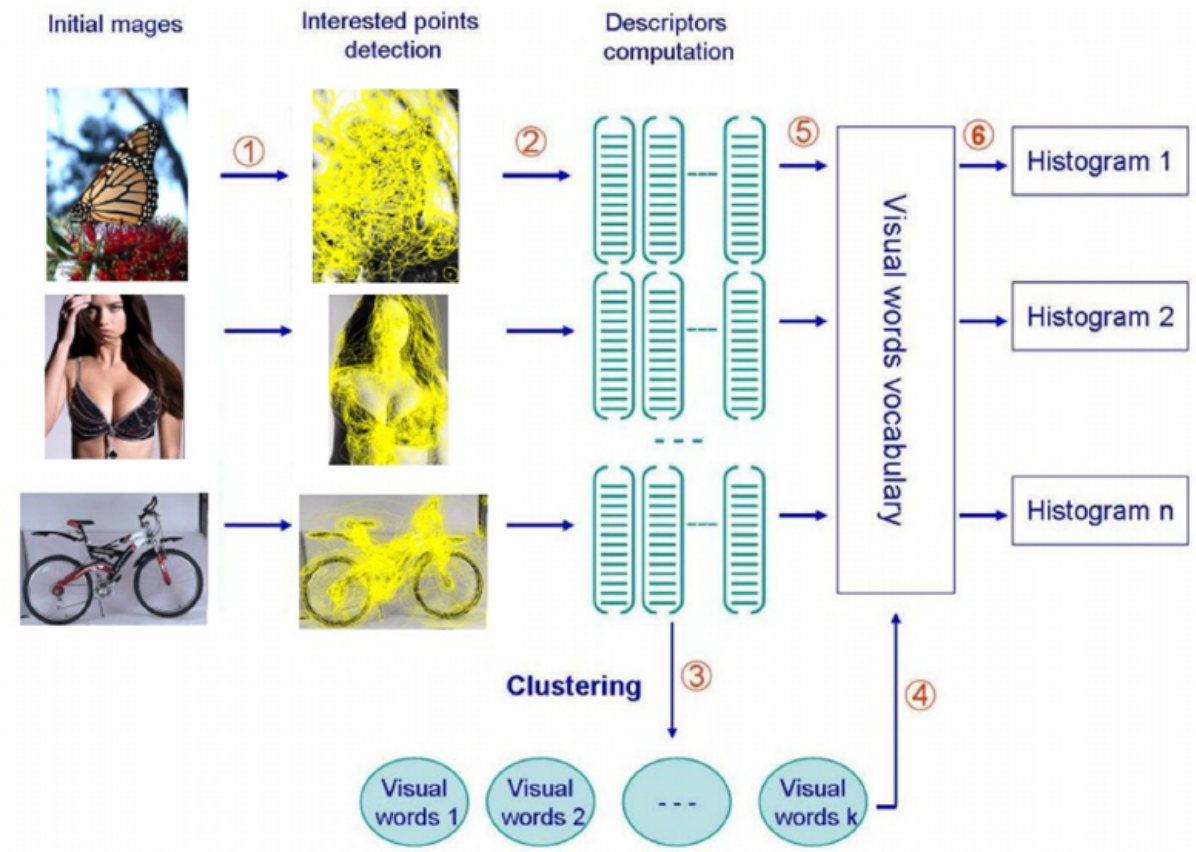


FIGURE 2.5 – Model de BOW

Chapitre 3

Apprentissage automatique

3.1 Introduction

L'apprentissage automatique (machine learning) est un domaine de l'intelligence artificielle, qui permet aux machines d'apprendre à partir des données. L'apprentissage automatique est la discipline scientifique concernée par le développement, l'analyse et l'implémentation de méthodes automatisables qui permettent à une machine d'évoluer autonome-ment grâce à un processus d'apprentissage, et ainsi de remplir des tâches qu'il est difficile ou impossible de remplir par des moyens algorithmiques plus classiques. Le problème ici est que la donnée d'observation est petite, elle ne peut pas comprendre tout l'ensemble de données d'entrées (tous les cas) qui est trop grande. Un programme d'apprentissage automatique doit généraliser des données limite afin de données des réactions intelligentes sur des nouveaux exemples.

3.2 Méthode SVM (Support Vector Machine)

Les machines à vecteurs de support (Support Vector Machine, SVM) sont un ensemble de techniques d'apprentissage supervisé destinées à résoudre des problèmes de classification et de régression. Dans cette section, nous ne parlons que SVM pour la classification concernant notre projet dans la pratique(classification).

La méthode SVM cherche l'hyperplan optimal qui permet de séparer les points (les données) en deux parties en respectant le principe où les points d'une même classe doit être dans la même partie (classification de 2 classes). Pour les problèmes de classification multi-classes, on a des méthodes pour

qu'il devient la problème de 2 classes que nous parlerons dans la partie MC-SGD.

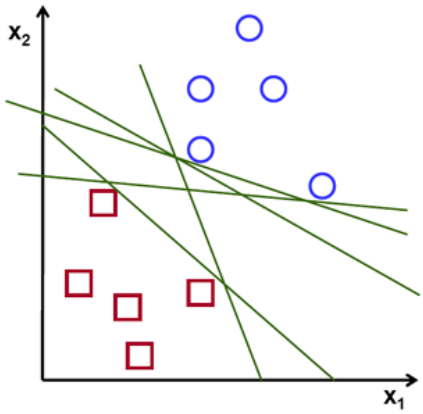


FIGURE 3.1 – Classification linéaire

Le cas simple est le cas d'une fonction de classification linéaire comme dans l'image 3.1. On a m exemples d'entrées $x_1, x_2, ..., x_m$ dans l'espace de N dimensions. Chaque exemple x_i a un label $y_i : y_1, y_2, ..., y_m$ ($y_i \in \{-1, 1\}$). Plusieurs hyperplans peuvent séparer ces 2 classes, mais SVM cherche l'hyperplan optimal qui est défini comme l'hyperplan qui maximise la marge entre les échantillons et l'hyperplan séparateur. La marge est la distance entre l'hyperplan et les échantillons les plus proches. Ces derniers sont appelés *vecteurs supports*. Dans l'espace de N dimensions, l'hyperplan est défini par le vecteur $w = [w_1, w_2, ..., w_n]$ et b. SVM cherche l'hyperplan (w, b) pour classifier les données comme l'image 3.2 :

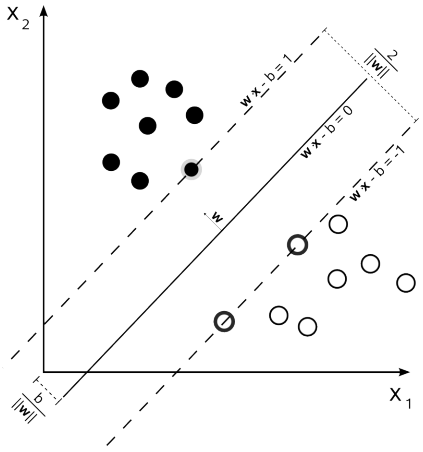


FIGURE 3.2 – L'hyperplan optimal

Après avoir trouvé w et b, l'hyperplan optimal est défini : $x^T.w - b = 0$ et 2 supports hyperplans $x^T.w - b = 1$ et $x^T.w - b = -1$. Par rapport à 2 supports hyperplans parallèles (voir l'image 3.2), la classification est réalisée grâce aux 3.1 et 3.2.

$$x_i.w - b \geq +1, \Rightarrow y_i = 1 \tag{3.1}$$

$$x_i.w - b \leq -1, \Rightarrow y_i = -1 \tag{3.2}$$

La classification peut être expliqué : l'hyperplan supporte la classe ***(+1)*** est l'hyperplan que les points qui a le label ***(+1)*** au dessus de l'hyperplan. Comme ça, l'hyperplan supporte la classe ***(-1)*** est l'hyperplan que les points qui a le label ***(-1)*** au dessous de l'hyperplan. Par rapport aux 3.1 et 3.2, on a le formule 3.3

$$y_i.(x_i.w - b) \geq 1 \tag{3.3}$$

Dans le cas où l'algorithme ne peut pas trouver (w, b) satisfait le problème (les données sont inséparable), nous devons accepter des erreurs z_i. Chaque exemple i est dans son vrais hyperplan, son erreur $z_i = 0$, sinon, son erreur z_i est défini par la distance entre cet exemple et son hyperplan. Le formule 3.3 devient 3.4 :

$$y_i.(x_i.w - b) + z_i \geq 1 \tag{3.4}$$

Nous trouvons que la classification est facile si l'on a trouvé w et b. La difficulté de la méthode SVM est que comment trouver w et b. Ce tache est réalisé après avoir trouvé la solution du programme de quadratique 3.5.

$$\begin{aligned} \min \quad & \Psi(w, b, z) = \frac{1}{2}||w||^2 + c.\sum_{i=1}^{m} z_i \\ & \text{s.t} \quad y_i.(x_i.w - b) + z_i \geq 1 \\ & \text{and} \quad z_i \geq 0 \end{aligned} \tag{3.5}$$

Le problème d'optimisation quadratique 3.5 est un des problèmes d'optimisation qui est normalement recherché dans le domaine de mathématique d'optimisation. Pour l'implémentation ce problème, [20] et [21] ont la complexité de $O(N^2)$ (N est le nombre d'exemples). Ce sont des programmes les plus utilisés actuel.

3.3 Méthode SVM avec SGD (Stochastic gradient descent)

SVM standard est efficace mais la complicité est grande ($O(N^2)$), donc, c'est intraitable pour les larges données. La méthode SVM avec SGD (Stochastic Gradient Descent), ou pour plus sample, on appelle la méthode SGD est créées pour résoudre ce problème. SGD ne cherche pas la solution pour résoudre le problème du programme de quadratique. Cette méthode ignore b dans le formule 3.5. Le contraint $y_i.(x_i.w - b) + z_i \geq 1$ peut être remplacé par :

$$z_i \geq 1 - y_i.(x_i.w) \tag{3.6}$$

A partir du contraint 3.6 et $z_i \geq 0$, on peut écrire la function de perte :

$$z_i = max\{0, 1 - y_i.(x_i.w)\} \tag{3.7}$$

Donc, le programme de quadratique 3.5 peut remplacer par le formule 3.9 :

$$\min \quad \Psi(w, x, y) = \frac{1}{2}||w||^2 + \frac{1}{m}\sum_{i=1}^{m} max\{0, 1 - y_i.(x_i.w)\} \tag{3.8}$$

Cette méthode est proposée dans [3]. Dans cette méthode, w est mise à jour en T étapes avec une vitesse d'apprentissage η. A chaque étape t, SGD utilise un exemple (x_i, y_i) aléatoire pour calculer le sous-gradient et met à jour w_{t+1}.

$$w_{t+1} = w_t - \nabla_w \Psi(w_t, x_i, y_i) \tag{3.9}$$

Cette algorithme est linéaire avec le nombre d'exemple. C'est la raison pour laquelle il est beaucoup plus vite que SVM standard.

3.4 Méthode MC-SGD (Multi Class - Stochastic gradient descent)

Plupart d'algorithme de SVM ne traite que le problème de 2 classes (binaire). Il existe plusieurs extensions d'une classification binaire afin de traiter le problème de multi-class classification. Actuellement, on a deux façon

pour traiter ce problème. L'une considère directement à résoudre le problème de multi-classes [24, 25]. L'autre divise d'abord le problème en plusieurs problèmes de 2 classes et chaque problème de 2 classes est traité par la classification binaire comme one-versus-one [22] et one-versus-all [23].

En pratique, one-vs-one et one-vs-all sont beaucoup utilisés par leur simple. one-vs-one construit *k(k-1)/2* (k est le nombre de classes) classificateurs, c'est à dire, il utilise tous les paires de classes différents pour l'apprentissage. La prédiction est réalisée par voter et la majorité classificateur va être choisi et la classe correspondant est la classe de prédiction. One-vs-all construit k classificateurs, à chaque classificateur c, il divise la classes c contre tous les restes. La classe de prédiction est la classe ayant la distance la plus courte entre son classificateur et l'exemple d'entrée.

Chaque méthode a l'avantage et l'inconvénient. One-vs-one est balancé mais il construit beaucoup de classificateurs. Par contre, one-vs-all construit moins classificateur mais il n'est pas balancé.

Chapitre 4

Implémentation

4.1 Introduction

Dans le chapitre précédent, nous avons généralement présenté des méthodes concernant notre travail. Après avoir étudié des différentes méthodes, dans chaque étape, nous avons choisi une méthode qui peut mieux adapter à notre travail. Durant notre recherche, il existe déjà des codages qui a la performance efficace, donc, nous avons réutilisé ces codages. Au contrait, pour les étapes qui n'existent pas encore des codages, nous avons les implémenté nous-même ou nous avons modifié des codages qui n'ont bien adapté pas à notre travail.

4.2 Représentation par des descripteurs et méthode sac de mots

La description des images est très importante dans la classification d'images. Cette étape est beaucoup influencée au résultat final. Dans le domaine de recherche d'images et de classification d'images, le descripteur SIFT [15] est un caractéristique important pour la représentation des images. Cette méthode est de plus en plus populaire. A partir de l'idée de la classification de textes, dans la recherche en 2007 [2], les auteurs proposent un système de classification d'images utilisant le descripteur SIFT et le modèle sac de mot visuel (Bag of Visual Word). Cette méthode peut diviser en 3 étapes :

1. trouver des descripteurs locales des images (des caractères)
2. construire un dictionnaire a partir des caractères
3. représenter des images par des histogrammes

Dans ces étapes, nous avons réutilisé des codages existant en modifiant des entrées et des sorties pour adapter aux étapes suivantes. Précisément, dans l'étape *1*, nous avons réutilisé le codage de D.Lowe [11] pour créer des vecteurs de caractéristiques (descripteurs). Ensuite, l'étape *2* sert à construire un dictionnaire. Dans cette étape, la méthode k-moyenne [26] est appliquée afin de créer des clusters a partir des vecteurs SIFT créés dans l'étape *1*. L'ensemble de clusters est considéré comme un dictionnaire. Enfin, dans l'étape *3*, une image est représentée par un histogramme. Dans cette étape, d'abord, on met chaque vecteur SIFT dans chaque image à un cluster le plus proche de ce vecteur (sa base sur la distance entre ce vecteur au centre des clusters). Ensuite, on compte le fréquence des mots d'une image existe dans le dictionnaire créé dans l'étape *2* pour représenter l'image comme un histogramme de fréquence. Pour l'étape *3*, nous avons implémenté en C/C++.

4.3 Apprentissage automatique

Avec notre processus, après la méthode sac de mot visuel, nous appliquons un algorithme d'apprentissage pour classifier des images vers leurs classes. Nous avons développé la méthode MC-SGD et pour vérifier la performance de notre méthode, nous avons fait la comparaison avec l'application LIBSVM [21], l'implémentation de SVM le plus utilisée actuellement.

4.3.1 Descente de gradient stochastique (SGD)

Pour cette méthode, il existe plusieurs d'implémentations différentes. Dans notre travail, nous avons réutilisé l'implémentation *Pegasos* [3] ci-dessous :

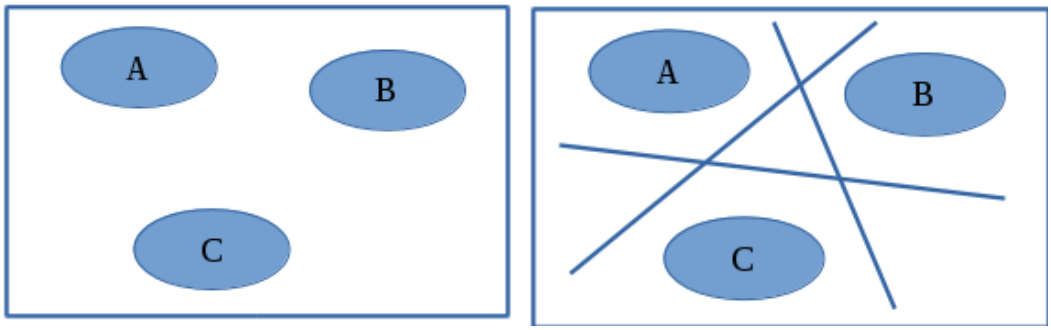
Algorithm 1 L'algorithm d'apprentissage SGD-SVM binaire

1: **procedure** TRAINBINAIRESGDSVM(D, y, λ, T, n)
2: Initialiser $w_1 = 0$
3: **for** `t = 0; t < T; t++` **do**
4: $\eta = \frac{1}{\lambda}$
▷ Boucle pour choisir ex exemples chaque cycle
5: **for** `ex = 0; ex < n; ex++` **do**
6: `choisir un exemple aléatoire dans D`
7: **if** $y_{it}.(w_t, x_{it}) < 1$ **then**
8: $w_{t+1} = (1 - \eta_t.\lambda).w_t + \eta_t.y_{it}.x_{it}$
9: **else**
10: $w_{t+1} = (1 - \eta_t.\lambda).w_t$
11: Return w_t

4.3.2 Descente de gradient stochastique pour multi-classe (MC-SGD)

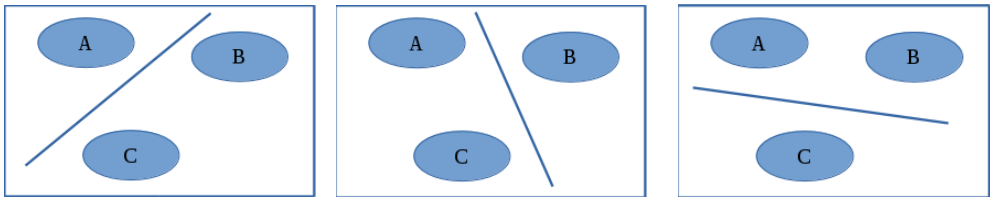
Comme les autres algorithme de SVM, SGD est implémenté pour résoudre le problème de classification de 2 classes. L'implémentation *Pegasos* [3] est efficace. A partir de cette implémentation, nous les avons modifié et développé pour résoudre le problème de multi-classes (k classes, $k \geq 3$).

Pour le problème de classification de multi-classes, comme nous avons expliqué dans la partie précédente, nous avons utilisé la méthode *one-vs-all*. Cette méthode ne construit que k classificateur (k est le nombre de classes). Par exemple, quand le problème a trois classes comme l'image 4.1a. *One-vs-all* va construire trois classificateurs comme l'image 4.1b.



(a) Problème (b) one-vs-all

FIGURE 4.1 – Problème de multi-classes



(a) classe A (b) class B (c) class C

FIGURE 4.2 – Problème de multi-classes

L'image 4.1b est comme un résumé de one-vs-all. Pour plus claire, nous vous listons les trois classificateurs de ces trois classes comme l'image 4.2

Algorithm 2 L'algorithm d'apprentissage SGD-SVM pour multi-classes

procedure TRAINMCSGDSVM(D, y, k, λ, T, n)
▷ D est les données d'apprentissage
▷ y est les labels des données
▷ k est le nombre de classes dans D
▷ λ est le constant possible
▷ T est le nombre de cycles
▷ n est le nombre d'exemple par cycle

    Initialiser k modèles w
    for `c = 0; c < k; c++` **do**
        ▷ Préparation des données pour one-vs-all
        $y_i = +1$ `si l'exemple de classe c,` $y_i = -1$ `sinon`
        $w_c = trainBinaireSGDSVM(D, y, \lambda, T, n)$ $(c - vs - all)$
    Return w

4.3.3 Parallélisation de MC-SGD

Nous trouvons que l'algorithme 2 linaire, il apprend les modèles l'un après l'autre sur un seul cœur du processeur d'ordinateur. Donc, les autres cœurs ne font rien et l'algorithme est lent. A notre époque, nous pouvons utiliser tous les cœurs possibles pour améliorer la vitesse de notre algorithme en utilisant la programmation parallèle.

Pour le problème de multi-classes (k classes), l'algorithme SGD-SVM apprend indépendant chaque classificateur. Donc, c'est possible d'apprendre chaque classificateur sur chaque cœur différent. Dans notre implémentation, nous avons parallélisé la boucle sur l'apprentissage des classificateur. Supposons que nous avons p processeurs et k classificateurs. Donc, chaque processeur apprend $n = \frac{k}{p}$ ou $n = \frac{k}{p} + 1$ classificateurs.

Algorithm 3 L'algorithm d'apprentissage SGD-SVM parallèle pour multi-classes

procedure TRAINMCSGDSVMPARALLEL(D, y, k, λ, T, n)
    Initialiser k modèles w

Pocesseur 1 :
    $y_i = +1$ `si l'exemple de classe c (c = 1, classe 1.p + 1, classe 2.p + 1, ...) et` $y_i = -1$ `sinon`
    $w_c = trainBinaireSGDSVM(D, y, \lambda, T, n)$ $(c - vs - all)$

Pocesseur 2 :
    $y_i = +1$ `si l'exemple de classe c (c = 2, classe 1.p + 2, classe 2.p + 2, ...) et` $y_i = -1$ `sinon`
    $w_c = trainBinaireSGDSVM(D, y, \lambda, T, n)$ $(c - vs - all)$
    .
    .
    .

Pocesseur p :
    $y_i = +1$ `si l'exemple de classe c (c = p, classe 1.p + p, classe 2.p + p, ...) et` $y_i = -1$ `sinon`
    $w_c = trainBinaireSGDSVM(D, y, \lambda, T, n)$ $(c - vs - all)$
Return w

Avec la structure comme l'algorithme 3, à chaque boucle, p processeurs apprennent p classificateurs en parallèle. Alors, la vitesse améliore presque p fois si l'on compare avec la version linaire de cet algorithme, l'algorithme 2.

Chapitre 5

Résultat obtenue

5.1 Introduction

Dans le chapitre précédent, nous avons généralement présenté des méthodes concernant notre travail avec leur implémentation détaillées. Dans ce chapitre, nous allons présenter notre résultat en comparant avec la méthode SVM standard, l'implémentation LIBSVM [21].

5.2 Méthode SGD-SVM

Pour cette étape, nous voulons comparer SGD-SVM binaire avec LIBSVM pour voir si quelle méthode est plus efficace. Nous utilisons ci-dessous la base de données binaire (2 classes) de LIBSVM [5], Adult avec la taille d'exemple d'entrées différente.

Dans cette expérimentation, nous avons utilisé la fonction linaire de SVM car elle est plus efficace que les autre fonctions (seulement avec cette base de données). Pour la méthode SGD, nous avons utilisé $T = 10000$ cycles et chaque cycle nous choisissons 10 exemples. Avec cette méthode, le résultat change un peu (moins de 10%) chaque case de test car elle choisit aléatoire des exemples dans chaque cycle, donc, nous avons testé 10 fois chaque exemple et fait la moyenne pour comparer avec la méthode SVM.

Données	#Exemple	LIBSVM(%)	SGD(%)	LIBSVM(s)	SGD(s)	$\frac{SVM(s)}{SGD(s)}$
a1a	1,605	83.82	84.30	0.438	0.044	10
a2a	2,265	84.27	84.48	0.826	0.045	18
a3a	3,185	84.33	84.31	6.990	0.050	139
a4a	4,781	84.44	84.33	3.162	0.043	73
a5a	6,414	84.39	84.33	5.766	0.048	120
a6a	11,220	84.72	84.34	20.846	0.049	425
a7a	16,100	84.83	84.45	42.392	0.056	757
a8a	22,696	85.16	84.95	91.500	0.054	1,694
a9a	32,561	84.97	84.64	299.648	0.064	4,682

TABLE 5.1 – Comparaison entre LIBSVM et SGD-SVM

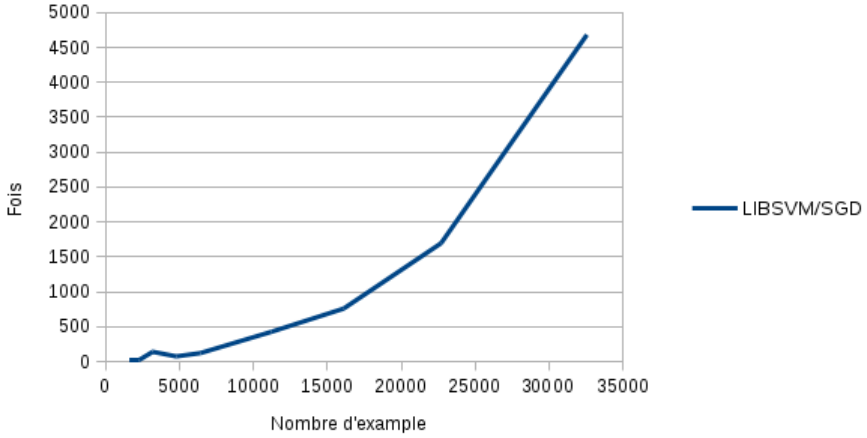


FIGURE 5.1 – Comparaison de la vitesse entre LIBSVM et SGD binaire

Par rapport au table 5.1, nous trouvons que le taux de classification de SGD-SVM et SVM est presque pareil tandis que le temps d'apprendre est différent. La vitesse de SGD-SVM est très vite que la vitesse de SVM standard. Cet avantage de SGD-SVM est plus claire quand le nombre d'exemples d'apprentissage augmente. Cette chiffre est démontrée dans la colonne de table 5.1 ou aussi dans le figure 5.1.

Les données que nous avons utilisé ont 123 caractéristiques. Nous trouvons que quand le nombre d'exemple est de 1,605, la méthode SGD est plus vitesse que la méthode SVM 10 fois, mais quand le nombre d'exemple de 32,561, SGD

plus vitesse que SVM 4,682 fois. Ce chiffre confirme que la méthode SGD est beaucoup plus vitesse que SVM, surtout pour les grandes données. Donc, la méthode SGD s'adapte mieux au problème de classification d'image dont la base de données est très grande. En raison de son avantage, nous voulons la développer pour qu'elle puisse résoudre le problème de classification de multi-classes pour utiliser dans la domaine de classification d'images.

5.3 Méthode MC-SGD

Avant d'utiliser pour le problème de classification d'images, nous testons cette méthode avec les données multi-classes de LIBSVM [4]. Tout d'abord, on teste avec la base de données *Protein* de 3 classes

Données	LIBSVM(%)	SGD(%)	LIBSVM(s)	LIN(s)	PARAL(s)	$\frac{SVM(s)}{PARAL(s)}$
Protein	68.23	68.66	511	0.35	0.2	2555

TABLE 5.2 – Comparaison entre LIBSVM et MC-SGD linaire et MC-SGD parallèle

En voyant le table 5.2, nous trouvons que la méthode MC-SGD parallèle plus vite que la méthode SVM 2555 fois. Nous trouvons aussi que la version parallèle n'est pas beaucoup plus vite que sa version linaire. La raison est que le temps d'apprendre d'un classificateur est très petit et cette base de données ne comprend que 3 classes. Donc, La version parallèle économie seulement 0.15 seconds.

A fin de vous montrer la vitesse de la version parallèle de MC-SGD, nous avons testé avec la base de données de 10 classes

Données	LIBSVM(%)	SGD(%)	LIBSVM(s)	LIN(s)	PARAL(s)	$\frac{SVM(s)}{PARAL(s)}$
mnist	86.92	86.46	2810	0.86	0.72	3902.8
fr	84.08	88.58	327	3.60	1.64	199.4
3d	76.54	76.8	144	1.62	0.90	160

TABLE 5.3 – Comparaison entre LIBSVM et MC-SGD linaire et MC-SGD parallèle

En générale, le résultat de classification de MC-SGD ne meilleure pas que SVM. Par contre, le temps d'apprendre est beaucoup plus vite que SVM pour le problème de classification de multi-classes. Le table 5.3 est une preuve.

Cette expérimentation est déroulée dans l'ordinateur de 8 cœurs, la version parallèle n'est pas beaucoup plus vite que la version linaire car j'ai utilisé l'option d'optimiser le temps de toute les deux versions et le résultat de classification est le même. Donc, dès maintenant, les autres comparaisons entre SVM et MC-SGD seront réalisées avec la version parallèle.

5.4 Classification d'images avec MC-SGD

Comme nous avons parlé dans le processus, notre processus pour la classification d'images comprend 3 étapes principales : extraction des descripteurs avec le descripteur SIFT, construit la dictionnaire avec la méthode K-MOYENNE et l'apprentissage automatique pour la classification avec la méthode MC-SGD. Dans la partie précédente nous avons présenté le résultat de la méthode MC-SGD(l'étape 3, l'étape d'apprentissage automatique) avec la base de données existantes. Maintenant nous appliquons la méthode Sac de Mots pour préparer les entrées pour la méthode MC-SGD avec les bases d'images de classification pour voir si la méthode MC-SGD s'adapte pour le problème de classification d'images.

Données	LIBSVM(%)	SGD(%)	LIBSVM(s)	MC-SGD(s)	$\frac{SVM(s)}{MC-SGD(s)}$
Scènes 8 catégories	58.04	57.74	15	0.18	83.3
Caltech 101	61.52	67.12	2873	106.95	26.9
Caltech 101	61.52	57.12	2873	74	38.8
Caltech 7 3D	91.52	88.3	113.4	0.81	140

TABLE 5.4 – Comparaison entre LIBSVM et MC-SGD parallèle pour la classification d'images

Le tableau 5.4 montre que le pourcentage de classification de SVM est presque égale que MC-SGD. Par contre, SVM est plus lent que MC-SGD environs 20 fois, ça dépend la base de test. Ces bases d'images ne sont pas très grandes, la première ne contient que 8 catégories, la deuxième de 101 classes et la troisième de 7 classes. Nous rappelons que LIBSVM utilise one-vs-one et MC-SGD utilise one-vs-all. Donc, si la base d'image augmente plus de catégories ou classes, la méthode MC-SGD sera plus vitesse si on compare avec la méthode SVM.

Chapitre 6

Conclusion et perspectives

Se base sur la méthode SVM, une méthode très populaire dans le domaine d'apprentissage automatique, les auteurs ont développé Pegasos [3] qui utilise SGD binaire. En générale, la méthode dans Pegasos n'est pas meilleure que la méthode SVM sur la classification car on peut comprendre SGD est une version simple de SVM où l'on ne doit pas résoudre le problème de programme de quadratique. Se base sur SGD binaire, nous avons développé notre algorithme pour que SGD s'adapte bien au problème de multi-classes. Pour le problème de classification de multi-classes avec one-vs-all, la version dans Pegasos trouve le problème de balance. Pour cette raison, nous avons modifié la méthode dans Pegasos. Dans cette recherche, nous ne faisons le point sur le résultat de classification, mais sur la vitesse de classification sur les bases d'images réelles. Donc, SGD s'adapte bien.

References

[1] S.Shwartz, Y.Singer and N.Srebro *Pegasos : Primal esti- mated sub-gradient solver for svm.*, Proceedings of the Twenty-Fourth International Conference Machine Learning, pp. 807-814. ACM (2007)

[2] A.Bosch, A.Zisserman and X.Munoz *Scene classification via pLSA.*, Proceedings of the European Conference on Computer Vision, pp. 517–530 (2006).

[3] Shalev-Shwartz, S.Singer, Y.Srebro, *Primal esti-mated sub-gradient solver for svm.*, Proceedings of the Twenty-Fourth International Conference Machine Learning. pp. 807–814, ACM (2007).

[4] *LIBSVM Data : Classification (Multi-class)*, `<http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass.html`

[5] *LIBSVM Data : Classification (Binary Class)*, `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html`

[6] PHAM Nguyen Khang, *Analyse factorielle des correspondances pour l'indexation et la recherche d'information dans une grande base de données d'image*, thèse IRISA, 2009.

[7] C.Schmid and R.Mohr, *Local Greyvalue Invariants for Image Retrieval*, IEEE Transactions on Pattern Analysis and Machine Intelligence,530–534, 19, 5 (1997).

[8] Chris Harris and Mike Stephens, *A combined corner and edge de detector*, in Alvey Vision Conf., 1988, pp. 147 - 151.

[9] Y.Dufournaud, C.Schmid and R.Horaud, *Matching Images with Different Resolutions*, in "International Conference on Computer Vision and Pattern Recognition (CVPR '00) 1 (2000) 612–618"

[10] Tony Lindeberg, *Feature Detection with Automatic Scale Selection,* Technical report ISRN KTH/NA/P–96/18–SE, May 1996, Revised August 1998. Int. J. of Computer Vision, vol 30, number 2, 1998. (In press).

[11] David G. Lowe, *Object Recognition from Local Scale-Invariant Features,* In proceeding of the 7th International conference of computer vision, pages 1150 - 1157, Corfou, Grèce, 1999.

[12] K.Mikolajczyk and C.Schmid, *Indexing based on scale invariant interest points*, in "Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on (Volume :1)", pages 525 - 531, 2001.

[13] K.Mikolajczyk, Detection of local features invariant to affine transformations, Ph.D. thesis, Institut National Polytechnique de Grenoble, France, 2002.

[14] S.Belongie, J.Malik and J.Puzicha, *Shape matching and object recognition using shape contexts*, in "Pattern Analysis and Machine Intelligence, IEEE Transactions on (Volume :24 , Issue : 4)", pages 509 - 522, 2002.

[15] David G. Lowe, *Distinctive Image Features from Scale-Invariant Keypoints*, International journal of computer vision, 60(2), 91 - 110, 2004.

[16] Yan Ke and Rahul Sukthankar, *PCA-SIFT : A More Distinctive Representation for Local Image Descriptors*, in "Computer Vision and Pattern Recognition", 2004.

[17] K.Mikolajczyk and C. Schmid, *A performance evaluation of local descriptors*, in "Pattern Analysis and Machine Intelligence, IEEE Transactions on (Volume :27 , Issue : 10)", 2005.

[18] Marco Treiber, *An introduction to object recognition : selected algorithms for a wide variety of application*, ISBN 9781849962346, pages p. 147, 2010.

[19] J.MacQueen, *Some methods for classification and analysis of multivariate observations. Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, Berkeley, University of California Press Vol.1, pp. 281-297 (1967).

[20] J.Platt, *Sequential Minimal Optimization : A Fast Algorithm for Training Support Vector Machines*, Microsoft Research Technical Report MSR-TR-98-14 (1998).

[21] C.Chang and C.Lin, *LIBSVM – a library for support vector machines*, http ://www.csie.ntu.edu.tw/ cjlin/libsvm (2001).

[22] V.Vapnik, *The Nature of Statistical Learning Theory.*, Springer, New York (1995).

[23] U.Krebel, *Pairwise classification and support vector machines*, Support Vector Learning, Advances in Kernel Methods. pp.255–268. (1999).

[24] J.Weston, C.Watkins*Support vector machines for multi-class pattern recognition.*, Proceedings of the Seventh European Symposium on Artificial Neural Networks. pp. 219–224. (1999)

[25] J.Weston, C.Watkins*Svm multiclasses.*, théorie et applications (2007)

[26] J.MacQueen *Some methods for classification and analysis of multivariate observations.*, Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, University of California Press Vol.1, pp. 281-297 (1967).