REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEURE ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE IBN KHALDOUN - TIARET

MEMOIRE

Présenté à :

FACULTÉ MATHEMATIQUES ET INFORMATIQUE
DÉPARTEMENT D'INFORMATIQUE

Pour l'obtention du diplôme de :

MASTER

Spécialité :[*Génie Informatique*]

Par :

[KEBAILI Hocine]

Sur le thème

Construction d'un Modèle à base d'Apprentissage Profond pour la Reconnaissance de Panneaux de Signalisation

---

Soutenu publiquement juillet2019 à Tiaret devant le jury composé de :

MrMEZOUG Karim	**MAA**	Université Ibn-KhaldounTiaret	Président
Mr CHENINE Abdelkader	**MAA**	Université Ibn-KhaldounTiaret	Encadreur
Mr BAGHDADI Mohamed	**MAA**	Université Ibn-KhaldounTiaret	Examinateur

Remerciements

Tout d'abord, je tiens à remercier avant et après tout ALLAH, *mon créateur de m'avoir donné la force, la santé, la volonté et le courage afin d'accomplir ce travail.*

J'adresse un remerciement tout particulier à mon Encadrant, MR. Chenin abdelkader, d'avoir eu confiance en moi. Je le remercie profondément pour son encouragement continu, disponibilité et ses critiques constructives qui m'ont permis d'améliore considérablement ce mémoire.

Je remercie les membres du Jury pour l'intérêt qu'ils portent à ce travail et d'avoir accepté d'évaluer ce mémoire.

Introduction Général :

La conduite assistée par ordinateur a pour objectif d'aider le conducteur à prendre les Bonnes décisions dans des circonstances difficiles (mauvaise vision, fatigue) pour Mieux pouvoir contrôler son véhicule et augmenter sa sécurité et la sécurité des autres Conducteurs, piétons etc. prenant part à la circulation sur les routes.

La vision artificielle constitue une approche prometteuse pour aborder le problème Susmentionné. Différentes installations d'assistance à la conduite pourraient être développées en se basant sur cette approche (système de freinage automatique pour ne pas dépasser la limitation de vitesse autorisée sur la route, système d'alarme qui émet un signal acoustique dès que le conducteur franchi une ligne continue sans le signaler etc.).

Le but de ce travail est de développer un système de reconnaissance automatique des signaux routiers algériens, connaissant les caractéristiques et les normes, telles que les informations sur les couleurs utilisées, les formes des signaux ou le lieu de stationnement des signaux routiers.

Chapitre 01

Chapitre 01

Détection et reconnaissance des panneaux de signalisation

Introduction

Supposons que vous conduisez à une vitesse de 80 km / h sur une route à sens unique et que vous êtes sur le point de rejoindre une nouvelle route. Même s'il y avait un panneau « Danger : route à double sens » dans la jonction, vous n'avez pas vu le panneau et vous continuez de rouler dans la voie opposée à la nouvelle route. Il s'agit d'une situation dangereuse pouvant entraîner un accident mortel, car le conducteur présume qu'il conduit toujours dans une route à double voie. Ce n'était qu'un exemple simple dans lequel le fait de ne pas détecter un panneau de signalisation pouvait avoir des conséquences irréversibles.

Ce danger devient encore plus grave avec des conducteurs inexpérimentés et des conducteurs âgés, en particulier sur des routes inconnues.

Acausedesonimportanteamplificationcesdernièresannées,letransportroutierconfrontedes problèmes qui ne cessent d'augmenter chaque année, qui se présentent dans les accidents routiers fréquents (Tableau 1et 2 : statistique d'Algérie 2017).[1]

Tableau1:Etatcomparatifdesaccidentscorporelsdelacirculationroutièreenregistrésdurant les périodes cités ci-dessous en zonesurbaines.

Désignation	du 19 au 25/02/2019	du 26/02 au 04/03/2019
Nombre d'accidents	309	293
Nombre de Blessés	373	379
Nombre de Décès	18	15

Tableau2 : Les principales causes des accidents corporels des périodes cités.

la cause	le nombre
le facteur humain	281
le véhicule	07
la route et l'environnement	05
Total	293

L'exemple ci-dessus est l'un des scénarios pouvant survenir en cas d'échec d'identification des panneaux de signalisation

En outre, les voitures autonomes seront couramment utilisées dans un proche avenir. Ils doivent également respecter le code de la route afin de ne pas mettre en danger les autres usagers de la route.

De même, les voitures intelligentes essaient d'assister les conducteurs humains et rendent la conduite plus sûre et confortable. Le système ADAS (Advanced Driver Assistant System) est un composant essentiel de ces voitures. L'une des tâches principales de ce module est de reconnaître les panneaux de signalisation. Cela aide un conducteur humain à connaître tous les panneaux de signalisation et à offrir une expérience de conduite plus sûre.

1. Défis

Un module de reconnaissance des panneaux de signalisation se compose de deux étapes principales : la détection et la classification. Ceci est illustré à la Fig. 1.1. L'étape de détection numérise l'image de la scène de manière multi-échelle et recherche l'emplacement des panneaux de signalisation sur l'image. Dans cette étape, le système ne distingue généralement pas un panneau de signalisation d'un autre. Au lieu de cela, il décide si une région inclut ou non un panneau de signalisation, quel que soit son type. La sortie de l'étape de détection est un ensemble de régions dans l'image contenant des panneaux de signalisation. Comme le montre la figure, le module de détection peut commettre des erreurs et générer quelques faux panneaux de signalisation. En d'autres termes, il pourrait y avoir quelques régions dans la sortie du module de détection sans aucun panneau de signalisation. Ces sorties ont été marquées à l'aide d'un rectangle rouge (en pointillé) dans la figure.

Ensuite, le module de classification analyse chaque région séparément et détermine le type de chaque panneau de signalisation. Par exemple, il y a un panneau «Interdiction de tourner à gauche », un panneau «Rond-point» et un panneau «Céder le passage» sur la figure. Il y a également trois panneaux «passage pour piétons». De plus, même s'il n'ya pas de panneau de signalisation dans les régions fausses-positives, le module de classification les étiquette dans l'une des classes de panneaux de signalisation. Dans cet exemple, les régions fausses-positives ont été classées en tant que panneaux « sens interdit à tout véhicule ».

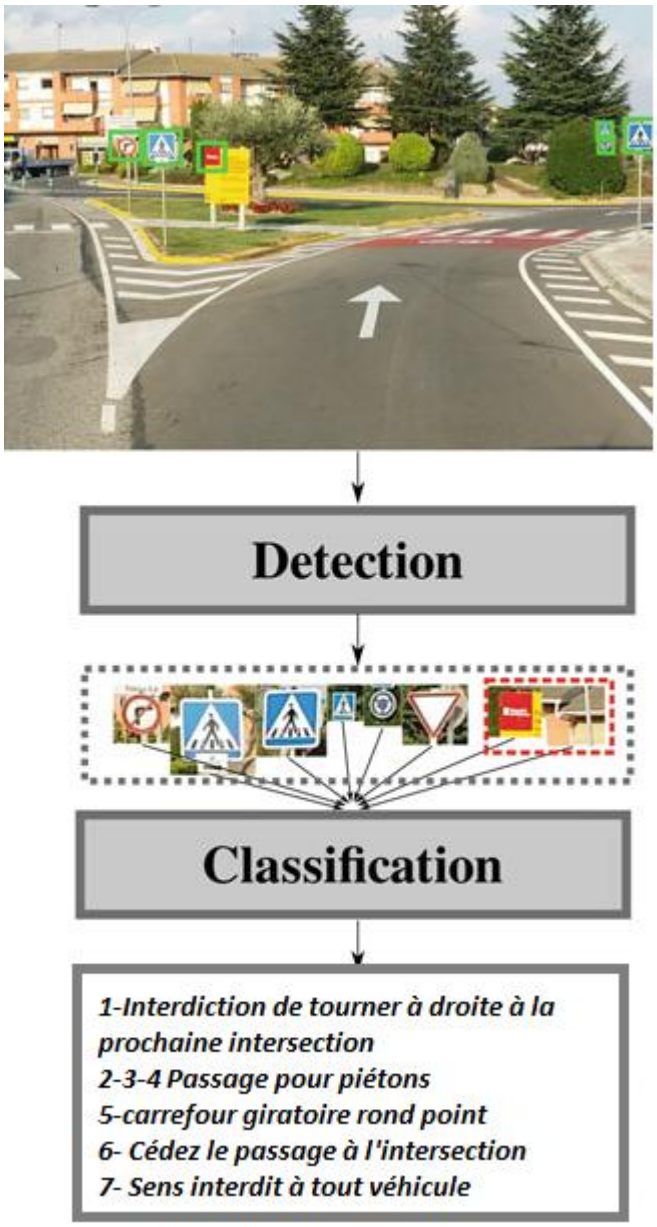


Fig. 1.1 La détection et la classification des panneaux de signalisation

Le traitement des régions fausses-positives générées par le module de détection est l'un des principaux défis à relever pour développer un système pratique de reconnaissance des panneaux de signalisation. Par exemple, une voiture autonome peut freiner brusquement dans l'exemple hypothétique ci-dessus car elle a détecté un panneau d'interdiction d'entrée. Par conséquent, l'un des défis pratiques du développement d'un module de détection est d'avoir zéro région fausses-positive. En outre, il doit détecter tous les panneaux de signalisation dans l'image. Techniquement, son taux de vrais-positifs doit être de 100%. Satisfaire à ces deux critères n'est pas trivial dans les applications pratiques.

La conception de panneaux de signalisation a deux objectifs principaux. Premièrement, ils doivent pouvoir être facilement distingués du reste des objets en scène, et deuxièmement, leur signification doit être facilement perceptible et indépendante du langage parlé. À cette

fin, les panneaux de signalisation sont conçus avec une forme géométrique simple telle qu'un triangle, un cercle, un rectangle ou un polygone. Pour être facilement détectables du reste des objets, les panneaux de signalisation sont peints avec des couleurs de base telles que le rouge, le bleu, le jaune, le noir et le blanc. Enfin, la signification des panneaux de signalisation est principalement réalisée par des pictogrammes au centre des panneaux de signalisation. Il convient de noter que certains signes dépendent fortement d'informations textuelles. Cependant, nous pouvons toujours considérer les textes des panneaux de signalisation comme des pictogrammes.

Bien que la classification des panneaux de signalisation soit une tâche facile pour un être humain, le développement d'un algorithme à cette fin pose certains défis. Certains de ces défis sont illustrés à la Fig. 1.2. Premièrement, l'image des panneaux de signalisation peut être capturée de différentes perspectives. Cela pourrait déformer de manière non linéaire la forme des panneaux de signalisation.

Deuxièmement, les conditions météorologiques peuvent considérablement affecter l'apparence des panneaux de signalisation. Un exemple est illustré à la Fig. 1.2, où le signe «Arrêt et stationnement interdits» est recouvert de neige.

Troisièmement, les panneaux de signalisation sont altérés au fil du temps et certains artefacts peuvent apparaître sur les panneaux, ce qui pourrait avoir une incidence négative sur leur classification.

Quatrièmement, les panneaux de signalisation peuvent être partiellement obstrués par un autre panneau ou des objets.

Cinquièmement, la zone du pictogramme peut être manipulée par des humains, ce qui peut dans certains cas modifier la forme du pictogramme. Un autre défi important est la variation de l'éclairement provoquée par les conditions météorologiques ou les changements de lumière du jour.

Le dernier et plus important problème illustré dans cette figure concerne les différences de pictogrammes du même panneau de signalisation d'un pays à l'autre. Plus en particulier, nous observons que le panneau «danger : passage à vélo» présente des différences importantes entre deux pays.

En ce qui concerne la Convention de Vienne sur les panneaux de signalisation routière, nous pouvons trouver environ 230 panneaux de signalisation illustrés. Ici, les signes textuels et les variations de signes picturaux sont comptés. Par exemple, le panneau de limitation de vitesse peut comporter 24 variations, dont 12 pour indiquer les limites de vitesse et 12 pour la fin de vitesse. De même, les panneaux de signalisation tels que la vitesse recommandée, la vitesse minimale, la distance minimale avec la voiture avant, etc. peuvent présenter plusieurs variantes. Par conséquent, la reconnaissance des panneaux de signalisation est un gros problème de classification multi-classes. Cela rend le problème encore plus difficile.

Fig 1.2 Quelques problèmes posé pour classer les panneaux de signalisation

Fig 1.3 Simple Différences entre deux panneaux de signalisation

Notez que certains des panneaux tels que les panneaux «carrefour» et «route secondaire» ne diffèrent que par des détails très fins. Ceci est illustré à la Fig. 1.3 où les deux signes ne diffèrent que par une petite partie du pictogramme. En regardant la Fig. 1.1, nous voyons des signes qui ne se trouvent qu'à 30 m de la caméra occupent une très petite région de l'image. Parfois, ces régions peuvent être aussi petites que 20 × 20 pixels. Pour cette raison, l'identification des détails fins devient très difficile sur ces panneaux.

En résumé, la classification des panneaux de signalisation est un cas spécifique de classification d'objet où les objets sont plus rigides et bidimensionnels. En outre, leurs parties

discriminantes sont bien définies. Cependant, l'élaboration d'un système pratique de détection et de classification des panneaux de signalisation pose de nombreux problèmes.

2. Travaux antérieurs

2.1 Correspondance des modèles

On peut soutenir que la méthode la plus simple pour reconnaître des objets est la correspondance de modèles. Dans cette méthode, un ensemble de modèles est stocké sur le système. Avec une nouvelle image, le modèle est associé à chaque emplacement de l'image et un score est calculé pour chaque emplacement.

Le score peut être calculé à l'aide de la corrélation croisée, de la somme des différences au carré, de la corrélation croisée normalisée ou de la somme normalisée des différences au carré. [02] ont stocké un ensemble de panneaux de signalisation en tant que modèles. Ensuite, l'approche ci-dessus a été utilisée afin de classer l'image d'entrée. Notez que l'approche par correspondance de modèle peut être utilisée à la fois pour la détection et la classification.

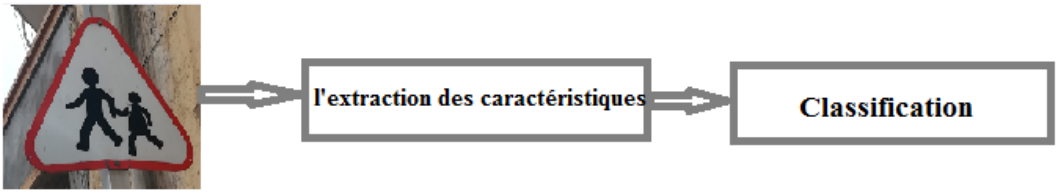
En pratique, cette approche pose de nombreux problèmes. Tout d'abord, elle est sensible à la perspective, à l'éclairage et à la déformation. Deuxièmement, elle n'est pas capable de traiter les signes de mauvaise qualité. Troisièmement, il faudra peut-être un grand ensemble de données de modèles pour couvrir différents types d'échantillons pour chaque panneau de signalisation. Pour cette raison, la sélection des modèles appropriés est une tâche fastidieuse.

D'une part, l'appariement de modèles consiste à comparer les intensités brutes de pixels entre le modèle et l'image source. D'autre part, les intensités de pixels dépendent grandement de la perspective, de l'éclairage et de la déformation. En conséquence, une légère modification de l'éclairage peut affecter de manière significative le score correspondant. Pour résoudre ce problème, nous appliquons généralement des algorithmes sur l'image afin d'en extraire des informations plus utiles. En d'autres termes, dans le cas d'images en niveaux de gris, un algorithme d'extraction de caractéristiques accepte une image W × H et transforme le vecteur dimensionnel W × H en un vecteur D-dimensionnel dans lequel le vecteur D-dimensionnel contient davantage d'informations utiles sur l'image et il est plus tolérant vis-à-vis des changements de perspective, de l'éclairage et de la déformation. Sur la base de cette idée, [03] ont extrait les caractéristiques de forme du modèle et de l'image source et les ont comparées à la place des valeurs d'intensité de pixels brutes. Dans ce travail, la correspondance des caractéristiques était réalisée à l'aide de la fonction de distance euclidienne. Ceci est équivalent à la fonction somme des différences carrées. Le problème principal de cette fonction de correspondance était que chaque caractéristique était également importante. Pour faire face à ce problème [04] ont appris une mesure de similarité pour faire correspondre le signe de la requête avec des modèles.

2.2 Caractéristiques artisanales

La procédure de mise en correspondance de modèles peut être décomposée en deux étapes. Dans la première étape, un modèle et un patch d'image sont représentés à l'aide de vecteurs plus informatifs appelés vecteurs de caractéristiques. Dans la deuxième étape, les vecteurs de caractéristiques sont comparés afin de trouver la classe du patch d'image. Cette approche est illustrée à la Fig. 1.4. Traditionnellement, la seconde étape est réalisée à l'aide de

techniques d'apprentissage automatique. Nous allons expliquer les bases de cette étape dans Sect. 2.1. Cependant, en gros, l'extraction d'un vecteur de caractéristiques d'une image peut



être effectuée à l'aide de méthodes artisanales ou automatiques.

Fig 1.4 Approche traditionnelle pour la classification d'objets

Les méthodes artisanales sont généralement conçues par un expert humain. Ils peuvent appliquer des séries de transformations et de calculs afin de créer un vecteur de caractéristiques. Par exemple[05] ,ont généré une image binaire en fonction de la couleur du panneau de signalisation. Ensuite, des caractéristiques invariantes au moment ont été extraites de l'image binaire pour former le vecteur de caractéristiques. Cette méthode peut être très sensible au bruit, car une image nette et sa version dégradée peuvent avoir deux images binaires différentes. Par conséquent, les moments des images binaires peuvent varier considérablement. [06] ont transformé l'image en espace colorimétrique HSI et ont calculé l'histogramme des composantes teinte et saturation. Bien que ce vecteur de caractéristiques puisse distinguer une catégorie générale de panneaux de signalisation (par exemple, des panneaux obligatoires par rapport à des panneaux de danger), ils peuvent également mal agir sur la modélisation des panneaux de signalisation de la même catégorie. Cela est dû au fait que les panneaux de signalisation de la même catégorie ont la même couleur et la même forme. Par exemple, tous les signes de danger sont un triangle avec une marge rouge. Par conséquent, la seule différence serait le pictogramme des signes. Comme tous les pictogrammes sont en noir, ils tomberont dans la même classe sur cet histogramme. En conséquence, théoriquement, cette classe sera la principale source d'information pour la classification des signes de la même catégorie.

Dans une autre méthode, ont classé les panneaux de signalisation en utilisant uniquement le pictogramme de chaque panneau. À cette fin, ils segmentent d'abord le pictogramme à partir de l'image du panneau de signalisation. Bien que la région du pictogramme soit binaire, la segmentation précise d'un pictogramme n'est pas une tâche aisée, car les méthodes de seuillage automatique telles que Otsu peuvent échouer compte tenu de la variation de l'éclairage et du bruit inattendu dans les applications du monde réel. Pour cette raison, ont entraîné SVM avec une entrée de 31 × 31 blocs de pixels dans une version en niveaux de gris du pictogramme. Dans une approche plus compliquée[07], ont proposé un framework corrigeant les erreurs pour la classification de 31 panneaux de signalisation et ont comparé leur méthode à différentes approches.

[08] ont utilisé un algorithme d'extraction de caractéristiques plus sophistiqué appelé histogramme de gradient orienté (HOG). De manière générale, la première étape de l'extraction de la fonctionnalité HOG consiste à calculer les gradients de l'image dans les directions x et y. Puis l'image est divisée en régions appelées cellules qui ne se chevauchent pas. Un histogramme est calculé pour chaque cellule. Les cases de l'histogramme indiquent

l'orientation du vecteur gradient. La valeur de chaque groupe est calculée en accumulant les amplitudes de gradient des pixels dans chaque cellule. Ensuite, des blocs sont formés en utilisant des cellules voisines. Les blocs peuvent se chevaucher. L'histogramme d'un bloc est obtenu en concaténant des histogrammes des cellules à l'intérieur du bloc. Enfin, l'histogramme de chaque bloc est normalisé et le vecteur de caractéristiques final est obtenu en concaténant l'histogramme de tous les blocs.

Cette méthode est formulée en utilisant la taille de chaque cellule, la taille de chaque bloc, le nombre de segments dans les histogrammes de la cellule et le type de normalisation. Ces paramètres sont appelés hyper-paramètres. En fonction de la valeur de ces paramètres, nous pouvons obtenir différents vecteurs de caractéristiques de différentes longueurs sur la même image. HOG est connu pour être un puissant algorithme d'extraction de caractéristiques conçu à la main. Cependant, les objets peuvent ne pas être séparables linéairement dans l'espace des fonctions. Pour cette raison) ont formé une forêt aléatoire et un SVM à la classification des panneaux de signalisation à l'aide de fonctions HOG.

La différence entre ces travaux réside principalement dans leur modèle de classification (par exemple, SVM, Cascade SVM, Extreme Learning Machine, plus proche voisin et LDA).

Cependant, contrairement aux autres travaux, [09] ont utilisé un modèle de classification à deux niveaux. Au premier niveau, l'image est classée dans l'une des superclasses. Chaque classe contient plusieurs panneaux de signalisation de forme / couleur similaire. Ensuite, la perspective de l'image d'entrée est ajustée en fonction de sa superclasse et un autre modèle de classification est appliqué à l'image ajustée. Le problème principal de cette méthode est la sensibilité de la classification finale à la procédure d'ajustement.

ontpropose une procédure plus complexe d'extraction de caractéristiques. Plus précisément, les premières fonctionnalités extraites de HOG avec plusieurs configurations d'hyper-paramètres. En outre, ils ont extrait davantage de vecteurs de caractéristiques à l'aide de différentes méthodes. Enfin, ils ont concaténé tous ces vecteurs et construit le dernier vecteur de caractéristiques. Néanmoins, cette méthode pose quelques problèmes. Leur vecteur de caractéristiques est un vecteur de 9 000 dimensions construit en appliquant cinq méthodes différentes. Ce vecteur de grande dimension est ensuite projeté dans un espace de dimension inférieure à l'aide d'une matrice de transformation.

2.3 Apprentissage des descripteurs

L’expertmis au point une méthode d’extraction des caractéristiques artisanale qui applique une série de transformations et de calculs afin d’extraire le vecteur final. Le choix de ces étapes dépend entièrement de l'expert. Un problème avec les caractéristiques artisanales est leur pouvoir de représentation limité. Cela entraîne le chevauchement de certaines classes d'objets avec d'autres classes, ce qui nuit aux performances de la classification. Deux approches communes pour résoudre partiellement ce problème consistent à développer un algorithme d'extraction de nouveauxdescripteurs et à combiner diverses méthodes. Le problème avec ces approches est que la conception d'une nouvelle méthode d'extraction de caractéristiques conçue à la main n'est pas triviale et que combiner différentes méthodes peut ne pas séparer les classes qui se chevauchent.

L'idée de base de l'apprentissage des descripteurs est de les apprendre à partir de données. Pour être plus précis, étant donné un ensemble de données de panneaux de signalisation, nous voulons apprendre un mappage :

$$M{:}R^d \rightarrow R^n$$

qui prend en entréedes vecteurs de d dimension (d = W × H) avec W : la largeur de l'image et H sa hauteur, et renvoie un vecteur à n dimensions.

Ici, l'entrée est une image aplatie obtenue en plaçant les lignes d'image les unes à côté des autres et en créant un tableau unidimensionnel. Le mappage M peut être toute fonction arbitraire linéaire ou non linéaire. Dans le scénario le plus simple, M peut être une fonction linéaire telle que:

$$\mathrm{M}\,(\mathrm{x}) = W^{+}\,(x^{T} - \bar{x}) \quad (1.1)$$

Où W $\in \mathbb{R}^{d\times n}$est une matrice de pondération, $x \in \mathbb{R}^{d}$ est l'image aplatie et $\bar{x} \in \mathbb{R}^{d}$ est l'image moyenne aplatie. De plus, $W^{+} = (W^{T}\mathrm{w})^{-1}W^{T}$désigne le pseudoinverse de MoorePenrose de W. Étant donné la matrice W, nous pouvons mapper chaque image dans un espace à n dimensions en utilisant cette transformation linéaire. Maintenant, la question est de savoir comment trouver les valeurs de W?

Pour obtenir W, nous devons concevoir un objectif et essayer de nous rapprocher le plus possible de l'objectif en modifiant les valeurs de W. Par exemple, supposons que notre objectif est de projeter x dans un espace à cinq dimensions où la projection est effectuée arbitrairement.

Il est clair que tout W $\in \mathbb{R}^{3\times d}$ servira notre objectif. En notant *M (x)* avec *z*, notre objectif pourrait être de projeter des données sur un espace n ≤ d tout en maximisant la variance de *z*.

Le W trouvé à l'aide de cette fonction objectif est appelé analyse en composantes principales. a expliqué que pour trouver *W* qui maximise cette fonction objective, nous devons calculer la matrice de covariance des données et trouver les vecteurs propres et les valeurs propres de la matrice de covariance. Ensuite, les vecteurs propres sont triés en fonction de leurs valeurs propres par ordre décroissant et les *n* premiers vecteurs propres sont choisis pour former *W*.

Maintenant, étant donné n'importe quelle image W × H, on la branche dans (1.1) pour calculer *z*. Ensuite, le vecteur dimensionnel z est utilisé comme vecteur de caractéristique. ont précédemment utilisé cette méthode pour modéliser les visages humains. ont également projeté l'image dans l'espace composant principal et ont trouvé une classe d'image en calculant la distance euclidienne de l'image projetée avec les images de la base de données.

Si nous multiplions les deux côtés avec W et que nous réarrangeons (1.1), nous obtiendrons

$$x^{T} = \mathrm{Wz} + \bar{x}^{T}. \quad (1.2)$$

Supposons que $\bar{x}^{T} = 0$. Techniquement, nous disons que nos données sont centrées sur zéro. Selon cette équation, nous pouvons reconstruire x en utilisant W et son mappage z.

Chaque colonne de W est un vecteur de dimension d qui peut être vu comme un modèle appris à partir de données. Avec cette intuition, la première ligne de W montre l'ensemble des valeurs du premier pixel de notre dictionnaire de modèles. De même, la n^{ième} ligne de W est un ensemble de valeurs de n^{ième} pixel dans les modèles.

Par conséquent, le vecteur z montre comment combiner linéairement ces modèles afin de reconstruire l'image d'origine. Lorsque la valeur de n augmente, l'erreur de reconstruction diminue.

La valeur de W dépend directement des données que nous avons utilisées lors de la phase d'apprentissage. En d'autres termes, en utilisant les données d'apprentissage, un système apprend à extraire des fonctionnalités.

Cependant, nous ne prenons pas en compte la classe d'objets en recherchant *W*. En général, les méthodes qui ne considèrent pas la classe d'objet sont appelées méthodes non *supervisées*.

Une limite de l'analyse en composantes principales est que n ≤ d. En outre, z est un vecteur réel susceptible d'être peu dense. Nous pouvons simplifier (1.2) en omettant le second terme.

Maintenant, notre objectif est de trouver W et z en minimisant la reconstruction contrainte. Erreur:

$$E = \sum_{i=1}^{N} \|x_i^T - W z_i\|_2^2 \quad \text{s.t.} \quad \|z\|_1 \quad (1.3)$$

Où μ est une valeur définie par l'utilisateur et N le nombre d'images d'apprentissage. W et z_i ont également la même signification que celle mentionnée ci-dessus. La L_1 contrainte dans l'équation ci-dessus obligez_i à être rare. Un vecteur est appelé fragment lorsque la plupart de ses éléments sont nuls. Minimiser la fonction objective ci-dessus nécessite une optimisation alternative de W et z_i. Cette méthode s'appelle le codage fragmenté. Les lecteurs intéressés peuvent trouver plus de détails dans [10]. Il convient de noter qu'il existe d'autres formulations pour la fonction objective et la contrainte.

L'approche de codage clairsemé présente deux avantages par rapport à l'analyse en composantes principales. Tout d'abord, le nombre de colonnes dans W (c'est-à-dire n) n'est pas limité à d. Deuxièmement, z_i est un vect0eur clairsemé. Le codage clairsemé a également été utilisé pour coder des images de panneaux de signalisation.

[11] ont codé chaque panneau de signalisation à l'aide de l'algorithme MatchingPursuit. Au cours des tests, l'image d'entrée est projetée sur différents ensembles de bases de filtres afin de trouver la meilleure correspondance. ontpropose une approche d'intégration graphique pour la classification des panneaux de signalisation. Ils ont préservé la représentation clairsemée dans l'espace d'origine en utilisant la norme L1,2. ont construit le dictionnaire en appliquant k signifie le regroupement sur les données d'apprentissage. Ensuite, chaque donnée est codée à l'aide d'une nouvelle entrée de codage similaire à l'approche de codage linéaire local .

[12] ont proposé une méthode basée sur les attributs visuels et le réseau bayésien. Dans cette méthode, chaque panneau de signalisation est décrit en termes d'attributs visuels. Afin de détecter les attributs visuels, l'image d'entrée est divisée en plusieurs régions et chaque région

est codée à l'aide d'un procédé de codage net maigre élastique. Enfin, les attributs sont détectés à l'aide d'un classifieur de forêt aléatoire. Les attributs détectés sont ensuite affinés à l'aide d'un réseau bayésien. La figure 1.5 illustre un dictionnaire appris par 43 classes de panneaux de signalisation.

Il existe d'autres techniques d'apprentissage de fonctions non supervisées. Parmi eux, les auto-encodeurs, les réseaux de croyances profonds et l'analyse de composants indépendants ont été largement étudiés et utilisés dans la communauté de la vision par ordinateur. L'un des inconvénients majeurs des méthodes d'apprentissage avec fonctions non supervisées est qu'elles ne prennent pas en compte la classe d'objets au cours du processus d'apprentissage. Des résultats plus précis ont été obtenus à l'aide de méthodes d'apprentissage supervisées. Comme nous le verrons au Chap. 3, les réseaux de neurones convolutifs (ConvNet) ont montré un grand succès dans la classification et la détection d'objets.

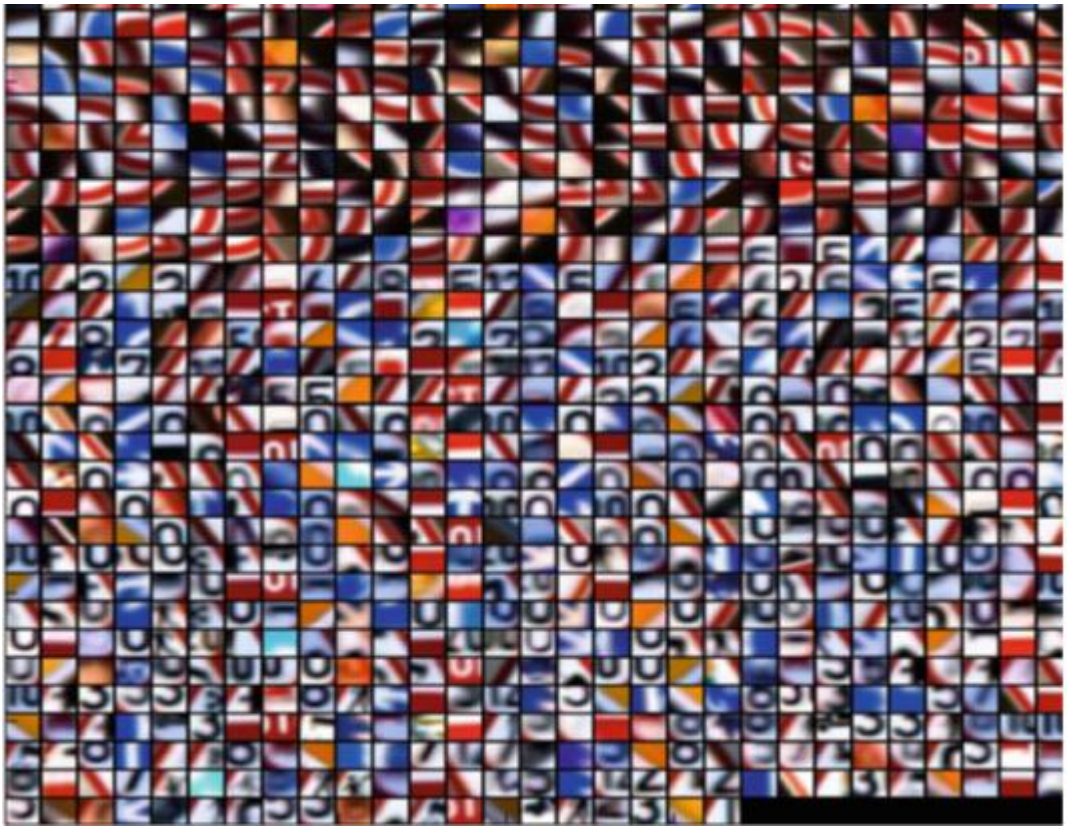


Fig. 1.5 Dictionnaire des classes de panneaux de signalisation

2.4ConvNets

ConvNetsont été utilisés pour la première fois dans le domaine de la classification des panneaux de signalisation lors de la compétition allemande GTSRB (GermanTraffic Sign Recognition Benchmark) où l'ensemble des ConvNets ont dépassé les performances humaines et ont remporté le concours en classant correctement 99,46% des images testées. De plus, le ConvNet s'est classé à la deuxième place avec une différence considérable par rapport à la troisième place attribuée à une méthode basée sur l'approche de classification traditionnelle. La précision dela classification du finaliste et de la troisième place était respectivement de 98,97 et 97,88%.

[13] construit un ensemble de 25 ConvNets comprenant chacun 1 543 443 paramètres. créent un réseau unique défini par 1 437 791 paramètres. En outre, alors que ConvNet, le gagnant, utilise la fonction d'activation hyperbolique, le vice-conviste ConvNet utilise la sigmoïde rectifiée comme fonction d'activation. C'est une pratique courante dans ConvNets de faire une prédiction en calculant le score moyen des versions légèrement transformées de l'image de requête.

Cependant, ne mentionnent pas clairement comment ils font une prédiction. En particulier, il n'est pas clair que le second ConvNet classe uniquement l'image d'entrée ou différentes versions de l'entrée et fusionne les scores pour obtenir le résultat final.

Quoi qu'il en soit, les deux méthodes souffrent du nombre élevé d'opérations arithmétiques. Pour être plus précis, ils utilisent des fonctions d'activation sophistiquées. Pour atténuer ces problèmes, ont propose une nouvelle architecture comprenant 1 162 284 paramètres et utilisant l'unité linéairement rectifiée (ReLU) .

En outre, il existe une couche de normalisation de réponse locale après chaque couche d'activation. Ils ont construit un ensemble de 20 ConvNets et ont classé 99,65% des images de test correctement. Bien que le nombre de paramètres soit réduit en utilisant cette architecture par rapport aux deux réseaux, l'ensemble est construit à l'aide de 20 ConvNets, ce qui n'est pas toujours efficace du point de vue des calculs dans les applications réelles. Ça vaut la peine mentionner qu'une couche ReLU et une couche de normalisation de réponse locale nécessite à peu près le même nombre d'opérations arithmétiques qu'une couche hyperbolique uniqueLe point commun à propos de tous les ConvNets ci-dessus est qu'ils ne conviennent que pour le module de classification et qu'ils ne peuvent pas être utilisés directement dans la tâche de détection.

Cela est dû au fait que l'application de ces réseaux ConvNets sur des images haute résolution n'est pas faisable par calcul. D'autre part, la précision du module de classification dépend également du module de détection. En d'autres termes, tous les résultats faux-positifs produits par le module de détection seront entrés dans le module de classification et classés comme l'un des panneaux de signalisation. Idéalement, le taux de faux-positifs du module de détection doit être égal à zéro et son taux de vrais positifs à 1. L'atteinte de cet objectif nécessite généralement des modèles de représentation d'image et de classification plus complexes.

Les ConvNets proposés pour la classification des panneaux de signalisation peuvent être expliqués de trois points de vue, à savoir l'évolutivité, la stabilité et le temps d'exécution. Du point de vue de la généralisation, aucun des quatre ConvNets n'a évalué la performance d'autres jeux de données. Il est crucial d'étudier le fonctionnement des réseaux lorsque les signes changent légèrement d'un pays à l'autre. Plus important encore, la puissance de transfert du réseau doit être estimée en ajustant la même architecture sur un nouvel ensemble de données avec un nombre différent de classes. De cette manière, nous pouvons estimer l'évolutivité des réseaux. Du point de vue de la stabilité, il est essentiel de déterminer le degré de tolérance du réseau au bruit et à l'occlusion. Cela peut être fait en générant quelques images bruitées et en les récupérant sur le réseau. Cependant, cette approche ne trouve pas l'image de bruit minimale qui est mal classée par le réseau. Enfin, l'efficacité du ConvNet en termes d'exécution doit être examinée. Cela est dû au fait que le ConvNet doit consommer le moins de cycles de processeur possible pour permettre aux autres fonctions d'ADAS de fonctionner en temps réel.

Conclusion

Dans ce chapitre, nous avons formulé le problème de la reconnaissance des panneaux de signalisation en deux étapes, à savoir la détection et la classification. L'étape de détection est chargée de localiser les régions d'image contenant des panneaux de signalisation et l'étape de classification est chargée de rechercher la classe de panneaux de signalisation. Les travaux connexes dans le domaine de la détection et de la classification des panneaux de signalisation sont également examinés. Nous avons mentionné plusieurs méthodes basées sur des fonctionnalités fabriquées à la main, puis nous avons introduit l'idée qui sous-tend

l'apprentissage des caractéristiques. Ensuite, nous avons expliqué certains des travaux basés sur les réseaux de neurones convolutionnels

Chapitre 02

Chapitre 02

Classification d'objets à partir d'images

Introduction

Les problèmes d'apprentissage automatique peuvent être classés en deux catégories : apprentissage supervisé, apprentissage non supervisé et apprentissage par renforcement. En apprentissage supervisé, nous avons un ensemble de vecteurs de caractéristiques et leurs valeurs cibles correspondantes. L'apprentissage supervisé vise à apprendre un modèle permettant de prédire avec précision les cibles à partir de nouveaux vecteurs de caractéristiques. En d'autres termes, l'ordinateur doit apprendre un mappage de vecteurs de caractéristiques vers leurs valeurs cibles.

Les vecteurs caractéristiques peuvent être appelés variables indépendantes et les valeurs cibles, variables dépendantes. L'apprentissage se fait à l'aide d'une fonction objectif qui dépend directement des valeurs cibles. Par exemple, la classification des panneaux de signalisation est un problème d'apprentissage supervisé. Dans un environnement non supervisé, nous n'avons qu'un ensemble de vecteurs de caractéristiques sans valeur cible. L'apprentissage non supervisé a pour objectif principal d'apprendre la structure des données. Ici, comme les valeurs cibles n'existent pas, il n'existe pas de moyen spécifique d'évaluer les modèles appris. Par exemple, supposons que nous ayons un jeu de données de 10 000 enregistrements dans lequel chaque donnée est un vecteur composé de [âge du conducteur, sexe du conducteur, niveau d'instruction du conducteur, expérience de conduite, type de voiture, modèle de voiture, constructeur de la voiture, point d'accident GPS, température, humidité, conditions météorologiques, lumière du jour, heure, jour de la semaine, type de route].

L'objectif pourrait être de diviser cet ensemble de données en 20 catégories. Ensuite, nous pouvons analyser les catégories pour voir combien d'enregistrements appartiennent à chaque catégorie et ce qui est commun à ces enregistrements. En utilisant cette information, nous pourrions peut-être dire dans quelles conditions les accidents de voiture se produisent plus fréquemment. Comme nous pouvons le voir dans cet exemple, il n'existe pas de moyen clair de déterminer dans quelle catégorie les enregistrements sont classés.

L'apprentissage par renforcement se produit généralement dans des environnements dynamiques où une série d'actions amène le système à obtenir une récompense ou une punition. Par exemple, considérons un système qui apprend à conduire une voiture. Le système se met en marche et quelques secondes plus tard, il heurte un obstacle. Une série d'actions a amené le système à heurter l'obstacle. Néanmoins, aucune information ne peut nous dire quelle était l'efficacité des actions effectuées par les systèmes à un moment donné. Au lieu de cela, le système est puni car il heurte un obstacle. Maintenant, le système doit déterminer quelles actions ne sont pas correctes et agir en conséquence.

1. Formulation

0L'apprentissage supervisé se décompose principalement en classification et régression. La principale différence entre eux est le type de valeurs cibles. Alors que les valeurs cibles d'un problème de régression sont des nombres réels / discrets, les valeurs cibles d'un problème

de classification sont des nombres catégoriels appelés étiquettes. Pour être plus précis, supposons que $\mathcal{F}_r : \mathbb{R}^d \rightarrow \mathbb{R}$ est un modèle de régression qui renvoie un nombre réel. De plus, supposons que nous avons la paire (x_r, y_r) incluant un vecteur d'entrée d dimensionnel x_r et un nombre réel y_r. Idéalement, $\mathcal{F}_r$ (x_r) doit être égal à (y_r). En d'autres termes, nous pouvons évaluer la précision de la prédiction en calculant simplement $|\ \mathcal{F}_r\ (x_r)$ - $y_i|$. En revanche, supposons le modèle de classification

$$\mathcal{F}_c : \mathbb{R}^d \rightarrow \{limitation\ de\ vitesse, danger, interdiction, obligation\} \quad (2.1)$$

Qui renvoie un numéro catégorique / une étiquette. Étant donné la paire (x_c, danger), $\mathcal{F}_c$ (x_c) doit être idéalement égal au danger. Cependant, il pourrait retourner obligation à tort. Il n'est pas possible de simplement soustraire la sortie de $\mathcal{F}_c$ avec l'étiquette réelle pour déterminer l'écart du modèle par rapport à la sortie réelle. La raison en est qu'il n'y a pas de définition spécifique de la distance entre les étiquettes. Par exemple, nous ne pouvons pas dire quelle est la distance entre «danger» et «interdiction» ou «danger» et «obligation». En d'autres termes, l'espace d'étiquette n'est pas un ensemble ordonné. Les problèmes de détection et de reconnaissance des panneaux de signalisation sont formulés à l'aide d'un modèle de classification.

Supposons un ensemble de paires $\mathcal{H}$ = $\{(x_0, y_0),. . . , (x_n, y_n)\}$ où $x_i \in \mathbb{R}^2$ est un vecteur d'entrée bidimensionnel et yi {0, 1} est son étiquette. Malgré le fait que 0 et 1 soient des nombres, nous les traitons comme des étiquettes catégoriques. Par conséquent, il n'est pas possible de calculer leur distance. La valeur cible y_i dans cet exemple ne peut prendre qu'une des deux valeurs. Ce type de problème de classification dans lequel la valeur cible ne peut prendre que deux valeurs est appelé problème de classification binaire. De plus, comme les vecteurs d'entrée sont bidimensionnels, nous pouvons facilement les tracer. La figure 2.1 illustre le diagramme de dispersion d'un échantillon $\mathcal{H}$.

Les carrés bleus indiquent les points appartenant à une classe et les cercles roses représentent les points appartenant à l'autre classe. Nous observons que les deux classes se chevauchent à l'intérieur du polygone vert. De plus, les vecteurs indiqués par les flèches vertes sont susceptibles d'être des données bruitées. Plus important encore, ces deux classes ne sont pas séparables linéairement. En d'autres termes, il n'est pas possible de séparer parfaitement ces deux classes l'une de l'autre en traçant une ligne sur le plan.

Supposons qu'on nous donne un $x_q \in \mathbb{R}^2$ et qu'on nous demande de dire à quelle classe x_q appartient. Ce point est représenté par une flèche noire sur la figure. Notez que nous ne connaissons pas la valeur cible de x_q. Pour répondre à cette question, nous devons d'abord apprendre un modèle de $\mathcal{H}$ capable de discriminer les deux classes. Il existe de nombreuses façons d'atteindre cet objectif dans la littérature. Cependant, nous ne sommes intéressés que par une technique particulière appelée modèles linéaires. Avant d'expliquer cette technique, nous mentionnons une méthode appelée k-voisin le plus proche.

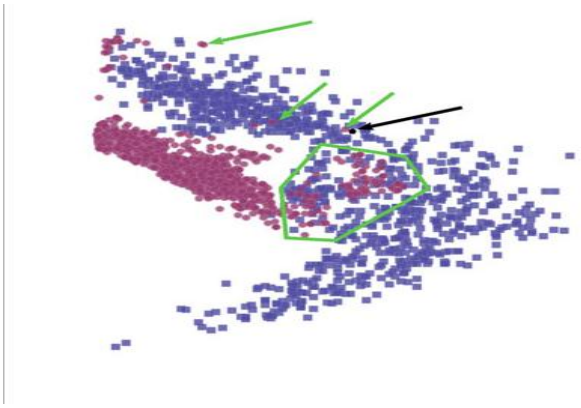


Fig.2.1. Un ensemble de données de vecteurs bidimensionnelsreprésentant deux classes desobjets

1.1 K-voisin le plus proche

D'un certain point de vue, les modèles d'apprentissage automatique peuvent être classés en modèles paramétriques et non paramétriques. En gros, les modèles paramétriques ont certains paramètres qui sont directement appris à partir des données. En revanche, les modèles non paramétriques n'ont aucun paramètre à apprendre des données. K plus proche voisin (KNN) est une méthode non paramétrique qui peut être utilisée dans les problèmes de régression et de classification.

Avec l'ensemble d'apprentissage $\varkappa$, KNN stocke tous ces échantillons en mémoire. Ensuite, étant donné le vecteur de requête x_q., il trouve les K échantillons les plus proches de $\varkappa$ à x_q. Désignant les K voisins les plus proches de x_q avec N_K $(x_q; \varkappa)$,la classe de est déterminée par:

$$F(x_q) = \arg\max_{v \epsilon \{0,1\}} \sum_{p \epsilon N_K(x_q)} \delta(v, (f_p)) \quad (2.2)$$

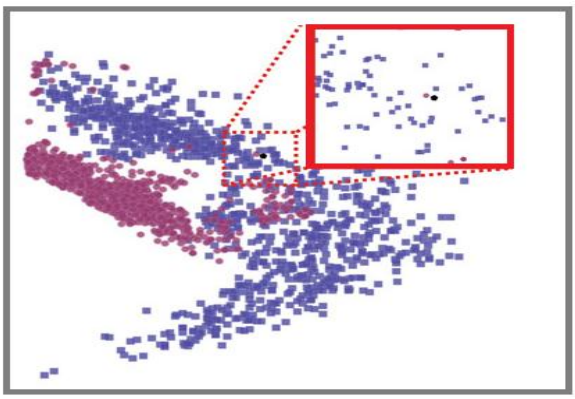


Fig. 2.2. K-NN cherche les points K plus proches points dans l'ensemble d'entraînement aupoint requête.

$$\delta(a, b) = \begin{cases} 1 & a = b \\ 0 & a \neq b \end{cases} \quad (2.3)$$

Où f (p) renvoie l'étiquette de l'échantillon d'apprentissage p $\in \varkappa$. Chacun des K voisins les plus proches vote pour $x_q$en fonction de leur libellé. Ensuite, l'équation ci-dessus compte les votes et renvoie la majorité des votes en tant que classe de x_q. (voirla figure 2.2.). En supposant que K = 1, KNN recherche le point le plus proche de $x_q$dans l'ensemble d'apprentissage (indiqué par un polygone noir sur la figure). Selon la figure, le cercle rouge est le point le plus proche. Parce que K = 1, il n'y a plus de point à voter. En conséquence, l'algorithme classe $x_q$en rouge.

En définissant K = 2, l'algorithme recherche les deux points les plus proches qui sont dans ce cas un cercle rouge et un carré bleu. Ensuite, l'algorithme compte les votes pour chaque étiquette. Les votes sont égaux dans cet exemple. Par conséquent, la méthode n'est pas sûre de sa décision. Pour cette raison, en pratique, nous avons attribué à K un nombre impair de sorte qu'un des libellés ait toujours la majorité des voix. Si nous fixons K = 3, il y aura deux votes pour la classe bleue et un vote pour la classe rouge. En conséquence, $x_q$sera classé comme bleu.

Nous avons classé chaque point du plan en utilisant différentes valeurs de K et $\varkappa$. La figure 2.3 illustre le résultat. La ligne pleine noire sur les graphiques montre la frontière entre deux régions avec des étiquettes de classe différentes. Cette frontière est appelée limite de décision. Lorsque K = 1, il y a toujours une région autour des points bruyants, où ils sont classés dans la classe rouge. Cependant, en réglant K = 3, ces régions bruitées disparaissent et font partie de la classe correcte. Lorsque la valeur de K augmente, la limite de décision devient plus lisse et les petites régions disparaissent.

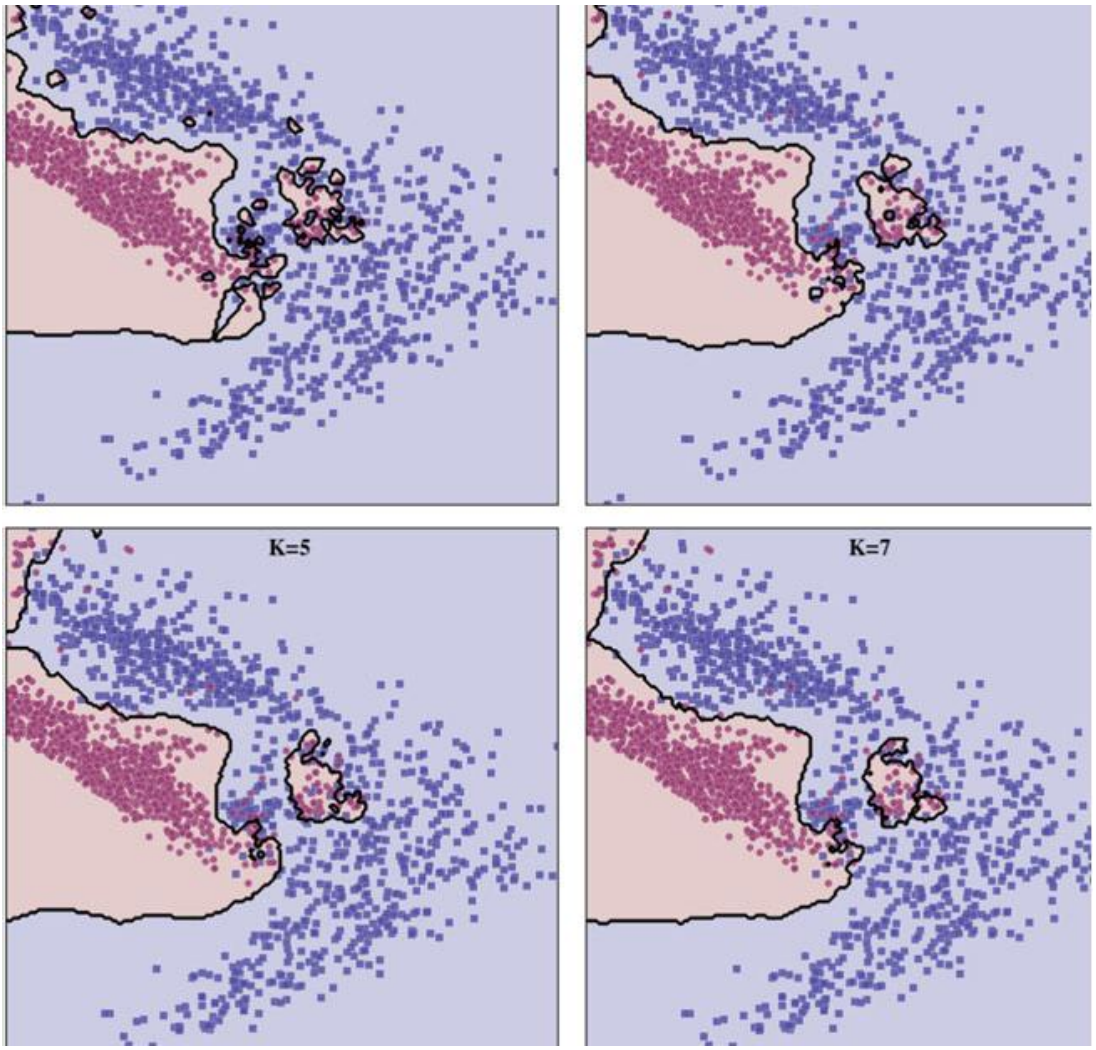


Fig.2.3.K plus proche voisin appliqué sur chaque point du plan pour différentes valeurs de K

Le KNN d'origine ne prend pas en compte les distances de ses voisins lorsqu'il compte les votes. Dans certains cas, nous pouvons vouloir pondérer les votes en fonction de la distance qui les sépare des voisins. Cela peut être fait en ajoutant un terme de poids à (2.2):

$$F(x_q) = \arg\max_{v\epsilon\{0,1\}} \sum_{p\epsilon N_K(x_q)} w_i \delta(v,(f_p)) \quad (2.4)$$

$$w_i = \frac{1}{d\,(x_{q\,,p})} \quad (2.5)$$

Dans l'équation ci-dessus, d (.) Renvoie la distance entre deux vecteurs. Selon cette formulation, le poids de chaque voisin est égal à l'inverse de sa distance àx_q. Par conséquent, les voisins les plus proches ont des poids plus élevés. KNN peut facilement être étendu à des jeux de données comportant plus de deux étiquettes sans aucune modification. Cependant, cette méthode pose deux problèmes importants. Premièrement, pour trouver la classe d'un vecteur de requête, il est nécessaire de calculer séparément la distance par rapport à tous les

échantillons du jeu d'apprentissage. Si nous ne concevons pas une solution telle que la partition de l'espace d'entrée, cela peut prendre beaucoup de temps et de mémoire lorsque nous traitons de grands ensembles de données. Deuxièmement, il souffre d'un phénomène appelé malédiction de la dimensionnalité. En termes simples, la distance euclidienne devient très similaire dans les espaces de grande dimension. En conséquence, si l'entrée de KNN est un vecteur de grande dimension, la différence entre le vecteur le plus proche et le plus éloigné peut être très similaire. Pour cette raison, il est possible que la classification des vecteurs de requête soit incorrecte.

Pour résoudre ces problèmes, nous essayons de trouver une fonction discriminante afin de modéliser directement la limite de décision. En d'autres termes, une fonction discriminante modélise la limite de décision à l'aide d'échantillons d'apprentissage dans $\varkappa$. Une fonction discriminante pourrait être une fonction non linéaire. Cependant, l'un des moyens les plus simples de modéliser les limites de décision consiste à utiliser des classificateurs linéaires.

2. Classificateur linéaire

Supposons un problème de classification binaire dans lequel les étiquettes du vecteur d'entrée d-dimensionnelle $x \in \mathbb{R}^d$ ne peuvent être que 1 ou -1. Par exemple, la détection de panneaux de signalisation dans une image peut être formulée comme un problème de classification binaire. Pour être plus spécifique, étant donné un patch d'image, la détection vise à déterminer si l'image représente un panneau de signalisation ou non. Dans ce cas, les images des panneaux de signalisation et d'autres objets peuvent être indiquées à l'aide des étiquettes 1 et -1, respectivement. Notant le $i^{\text{ème}}$élément de x avecx_i, il peut être classé en calculant la relation linéaire suivante :

$$f\ (\text{x}) = w_1 x_1 + \cdots + w_i x_i + \cdots + w_d x_d + \text{b} \quad (2.6)$$

Où w_i est un paramètre (qu'on peut apprendre) associé à $x_i$et b est un autre paramètre appelé biais. L'équation ci-dessus représente un hyperplan dans un espace euclidien à d dimensions. L'ensemble des poids $\{\forall i = 1 \ldots dw_i\}$ détermine l'orientation de l'hyperplan et b indique la distance de l'hyperplan à l'origine. Nous pouvons aussi écrire l'équation ci-dessus en termes de multiplications matricielles :

$$f(x) = wx^T + \text{b} \quad (2,7)$$

Où w = [w_1, ...,w_d]. De même, il est possible d'augmenter w avec b et d'afficher tous les paramètres de l'équation ci-dessus dans un seul vecteur$w_{w|\text{b}}$= [b,w_1 ...,w_d]. Avec cette formulation, nous pouvons également augmenter x avec 1 pour obtenir $x_{x|1}$= [1,x_1 ...,x_d]. et écrire l'équation ci-dessus en utilisant la multiplication de matrice suivante:

$$f(x) = w_{w|\text{b}} X^T_{x|1} \quad . (2.8)$$

Enfin, x est classé en appliquant la fonction de signe sur $f\ (x)$ comme suit :

$$\text{F(x)} = \begin{cases} 1 & \text{f(x)>0} \\ \text{Indéterminé} & \text{f(x)= 0} \\ -1 & \text{f(x)<0} \end{cases} \quad (2.9)$$

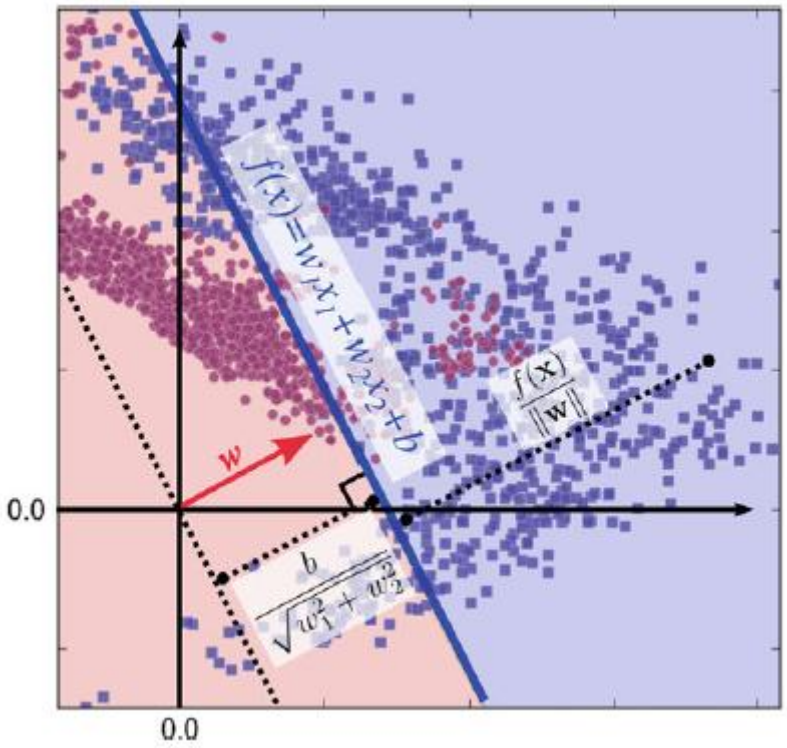


Fig.2.4 Géométrie des modèles linéaire

En d'autres termes, x est classé comme 1 si f (x) est positif et il est classé comme -1 lorsque f (x) est négatif. Le cas particulier se produit lorsque f (x) = 0 dans lequel x n'appartient à aucune de ces deux classes. Bien qu'il puisse ne jamais arriver en pratique d'avoir un x tel que f (x) soit exactement égal à zéro, cela explique un concept théorique important appelé limite de décision. Clairement, f (x) est zéro quand x est exactement sur l'hyperplan. Considérant le fait que w et x sont tous deux des vecteurs dimensionnels d + 1, (2.8) indique le produit scalaire des deux vecteurs. De plus, nous savons en algèbre linéaire que le produit scalaire de deux vecteurs orthogonaux est 0. Par conséquent, le vecteur w est orthogonal à chaque point de l'hyperplan.

Cela peut être étudié sous un autre angle. Ceci est illustré à l'aide d'un exemple bidimensionnel à la Fig. 2.4. Si nous réécrivons (2.6) sous forme d'interception de pente, nous obtiendrons:

$$x_2 = -\frac{w1}{w2} x_1 - \frac{b}{w2} \qquad (2.10)$$

Où la pente de la ligne est égale à m = $-\frac{w1}{w2}$. De plus, une ligne est perpendiculaire à la ligne ci-dessus si sa pente est égale à $m' = -\frac{1}{m} = -\frac{w2}{w1}$

Il en résulte que le vecteur de pondération w = [w_1,w_2] est perpendiculaire à chaque point de la ligne ci-dessus, car sa pente est égale à $\frac{w_2}{w_1}$. Regardons de plus près la géométrie du

modèle linéaire. La distance du point x' = [x'_1, x'_2] à partir du modèle linéaire peut être trouvé en projetant x - x' sur w qui est donné par :

$$r = \frac{|f(x)|}{||w||} \qquad (2.11)$$

Ici, w fait référence au vecteur de poids sans le b. De plus, la distance signée peut être obtenue en supprimant l'opérateur abs (valeur absolue) du numérateur :

$$r_{signed} = \frac{f(x)}{||w||} . \quad (2.12)$$

Lorsque x est sur la ligne (c'est-à-dire un hyperplan dans un espace à N dimensions), alors f (x) = 0. La distance à partir de la limite de décision sera donc égale à zéro. Ensemble de tous les points {x | x$\in\mathbb{R}^d$et f (x = 0)} représente la limite entre les régions étiquetées -1 et 1. Cette limite est appelée limite de décision. Cependant, si x n'est pas sur la limite de décision, sa distance sera une valeur différente de zéro. De plus, le signe de la distance dépend de la région dans laquelle se trouve le point. Intuitivement, le modèle est plus confiant quant à sa classification lorsqu'un point est loin de la limite de décision. Au contraire, à mesure qu'on se rapproche de la limite de décision, la confiance du modèle diminue. C'est la raison pour laquelle nous appelons parfois $f(x)$ le score de classification de x.

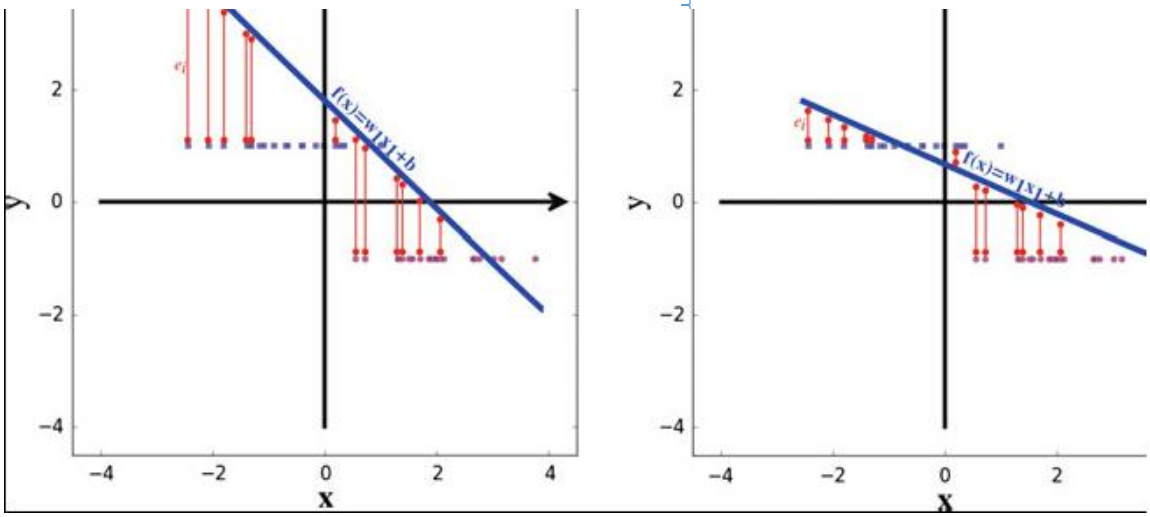
2.1 Formation d'un classificateur linéaire

Selon (2.9), la sortie d'un classifieur linéaire pourrait être 1 ou -1. Cela signifie que les étiquettes des données d'apprentissage doivent également être membres de l'ensemble {−1, 1}. Supposons que nous recevions l'ensemble d'apprentissage $\varkappa$ = {(x_0, y_0),... , (x_n, y_n)} tel que $x\in\mathbb{R}^d$ est un vecteur à deux dimensions et $y_i\in$ {−1, 1} représentant l'étiquette de l'échantillon. Pour former un classificateur linéaire, nous devons définir une fonction objective. Pour tout poidsw_t, la fonction objective utilise $\varkappa$ pour indiquer la précision de f (x) =$w_t x^T$lors de la classification des échantillons dans $\varkappa$. La fonction objective peut également être appelée fonction d'erreur ou fonction de perte. Sans la fonction de perte, il n'est pas anodin d'évaluer la qualité d'un modèle.

Notre objectif principal dans la formation d'un modèle de classification est de minimiser le nombre d'échantillons mal classés. Nous pouvons formuler cet objectif en utilisant l'équation suivante :

$$\mathcal{L}_{0/1}(w) = \sum_{i=1}^{n} H_{0/1(wx^T, y_i)} \qquad (2.13)$$

1 (wx^T , y_i) (2.14)



$H_{0/1(wx^T ,y_i)} =$ 0 autrement

Fig 2.5. L'intuition derrière la fonction de perte au carré est de minimiser la différence au carré entre la réponse réelle et la valeur prédite. Les tracés gauche et droit montrent deux lignes avec différents w1 et b. La ligne du tracé de droite est mieux ajustée que celle du tracé de gauche car son erreur de prédiction est plus faible au total.

La fonction de perte ci-dessus est appelée fonction de perte 0/1. Un échantillon est classé correctement lorsque les signes wx^T et y_i sont identiques. Si x n'est pas correctement classé par le modèle, les signes de ces deux termes ne seront pas identiques. Cela signifie que l'un de ces deux termes sera négatif et l'autre sera positif. Par conséquent, leur multiplication sera négative. Nous voyons que $H_{0/1}(.)$ Renvoie 1 lorsque l'échantillon est classé incorrectement. Sur la base de cette explication, la fonction de perte ci-dessus compte le nombre d'échantillons mal classés. Si tous les échantillons de $\varkappa$ sont classés correctement, la fonction de perte ci-dessus sera égale à zéro. Sinon, il sera supérieur à zéro. La fonction de perte ci-dessus pose deux problèmes qui la rendent impraticable. Premièrement, la fonction de perte 0/1 est non convexe. Deuxièmement, il est difficile d'optimiser cette fonction à l'aide de méthodes d'optimisation basées sur le gradient, car la fonction n'est pas continue au point 0 et son gradient est nul ailleurs.

Au lieu de compter le nombre d'échantillons mal classés, nous pouvons formuler le problème de classification comme un problème de régression et utiliser la fonction de perte au carré.

Ceci peut être mieux décrit à l'aide d'un vecteur d'entrée unidimensionnel x ϵ $\mathbb{R}$ sur la figure 2.5.

Sur cette figure, les cercles et les carrés illustrent les échantillons avec les étiquettes -1 et 1, respectivement. Puisque x est unidimensionnel (échelle), le modèle linéaire sera $f\ (x)$ = w_1x_1+ b avec seulement deux paramètres pouvant être entraînés. Ce modèle peut être tracé en utilisant une ligne dans un espace à deux dimensions. Supposons la ligne montrée dans cette figure. Étant donné x, la sortie de la fonction est un nombre réel. Dans le cas des cercles, le modèle devrait idéalement renvoyer -1. De même, il doit renvoyer 1 pour tous les carrés de cette figure. Néanmoins, parce que f(x) est un modèle linéaire $f\ (x_1) \neq f\ (x_2)$ si $x_1 \neq x_2$.

Cela signifie qu'il est impossible que notre modèle retourne 1 pour chaque carré de cette figure. En revanche, il retournera une valeur unique pour chaque point de cette figure.

Pour cette raison, il existe une erreur entre la sortie réelle d'un point (cercle ou carré) et la valeur prédite par le modèle. Ces erreurs sont illustrées par des lignes continues rouges dans cette figure. L'erreur d'estimation pour $x_i$peut être formulée comme e_i= (f (x_i) - y_i), où $y_i \in \{-1, 1\}$ est la sortie réelle de $x_i$telle que définie précédemment dans cette section. En utilisant cette formulation, nous pouvons définir la fonction de perte au carré comme suit :

$$\mathcal{L}sq(\mathrm{w}) = \sum_{i=1}^{n} \sqrt{(e_i)^2} == \sum_{i=1}^{n} \sqrt{(wx_i^T - y_i\ )^2} \quad (2.15)$$

Dans cette équationx $\epsilon\ \mathbb{R}^d$est un vecteur d-dimensionnel et $y_i \in \{-1, 1\}$ est son étiquette réelle. Cette fonction de perte traite les étiquettes comme un nombre réel plutôt que comme des valeurs catégoriques. Cela permet d'estimer l'erreur de prédiction en soustrayant les valeurs prédites des valeurs réelles. Notez sur la Fig. 2.5 qu'en peut être une valeur négative ou positive. Afin de calculer la magnitude dee_i, calculons d'abord le carré de e_i et appliquons la racine carrée afin de calculer la valeur absolue de e_i. Il convient de noter que nous pourrions définir la fonction de perte comme

$$\sum_{i=1}^{n} |\ wx_i^T - y_i\ | \text{ au lieu de } \sum_{i=1}^{n} \sqrt{(wx_i^T - y_i\ )^2}.$$

Cependant, comme nous le verrons bientôt, la deuxième formulation a une propriété souhaitable lorsque nous utilisons une optimisation basée sur un gradient pour minimiser la fonction de perte ci-dessus.

Nous pouvons encore simplifier (2.15). Si nous enlevons l'opérateur de somme en (2.15), cela ressemblera à :

$$\mathcal{L}sq(\mathrm{w}) = \sqrt{\left(wx_1^T - y_1\ \right)^2} + \cdots + \sqrt{(wx_n^T - y_n\ )^2}\ . \quad (2.16)$$

Tenant compte du fait que la racine carrée est une fonction monotone croissante et qu'elle est appliquée à chaque terme individuellement, le fait de supprimer cet opérateur de l'équation ci-dessus ne modifie pas le minimum de $\mathcal{L}$ (w). En appliquant ceci à l'équation ci-dessus, nous obtiendrons :

$$\mathcal{L}sq(\mathrm{w}) = \sum_{i=1}^{n} (wx_i^T - y_i\ )^2 \quad . (2.17)$$

Notre objectif est de minimiser l'erreur de prédiction. En d'autres termes :

$$\mathrm{w} = \min \mathcal{L}(w') \quad (2.18)$$

$$w' \in \mathbb{R}^{d+1}$$

Ceci est réalisable en minimisant $\mathcal{L}sq$ par rapport à w $\in\mathbb{R}^{d+1}$. Afin de minimiser la fonction de perte ci-dessus, nous pouvons utiliser une méthode d'optimisation itérative basée sur un gradient, telle que la méthode de descente de gradient. En partant du vecteur initial $w_{sol} \in\mathbb{R}^{d+1}$, cette méthode modifie itérativement w_{sol} proportionnellement au gradient

$\nabla\mathcal{L}= [\frac{\delta\mathcal{L}}{\delta w_0},\frac{\delta\mathcal{L}}{\delta w_1}\ldots,\frac{\delta\mathcal{L}}{\delta w_d}]$

Ici, nous avons montré l'interception utilisant w_0 au lieu de b. Par conséquent, nous devons calculer la dérivée partielle de la fonction de perte pour chacun des paramètres dans w comme suit :

$$\frac{\delta\mathcal{L}}{\delta w_i} = 2\sum_{i=1}^{n} x_i\,(w_x^T - y_i)\forall_i = 1 \ldots d$$

$$\frac{\delta\mathcal{L}}{\delta w_0} = 2\sum_{i=1}^{n}(w_x^T - y_i) \quad (2.19)$$

L'un des problèmes de l'équation ci-dessus est que $\mathcal{L}sq$ peut constituer une valeur importante s'il existe de nombreux échantillons d'apprentissage dans $\varkappa$. Pour cette raison, nous aurons peut-être besoin d'utiliser un taux d'apprentissage très faible dans la méthode de descente par gradient. Pour résoudre ce problème, nous pouvons calculer l'erreur quadratique moyenne en divisant $\mathcal{L}sq$ par le nombre total d'échantillons d'apprentissage. De plus, nous pouvons éliminer 2 de la dérivée partielle en multipliant $\mathcal{L}sq$ par 1/2. La fonction de perte au carré finale peut être définie comme suit :

$$\mathcal{L}sq(\mathrm{w}) = \frac{1}{2n}\sum_{i=1}^{n} x_i\left(wx_i^T - y_i\right)^2 \quad (2.20)$$

Avec ses dérivées partielles égales à :

$$\frac{\delta\mathcal{L}}{\delta w_i} = \frac{1}{n}\sum_{i=1}^{n} x_i\left(wx^T - y_i\right)\forall_i = 1 \ldots d \quad (2.21)$$

$$\frac{\delta\mathcal{L}}{\delta w_0} = \frac{1}{n}\sum_{i=1}^{n}\left(wx^T - y_i\right)$$

Notez que l'emplacement de minimum de (2.17) est identique à (2.20). Cette dernière fonction est simplement multipliée par une valeur constante. Cependant, l'ajustement du taux d'apprentissage est plus facile lorsque nous utilisons (2.20) le w optimal. Une propriété importante de la fonction de perte au carré avec les modèles linéaires est qu'il s'agit d'une fonction convexe. Cela signifie que la méthode de descente par gradient convergera toujours au minimum global, quel que soit le point initial. Il est à noter que cette propriété ne tient pas si le modèle de classification est une fonction non linéaire de ses paramètres. Nous avons

minimisé la fonction de perte de carré sur l'ensemble de données illustré à la Fig. 2.1. La figure 2.6 montre l'état de la descente du gradient dans quatre itérations différentes.

L'arrière-plan des graphiques montre l'étiquette de chaque région en fonction du signe du score de classification calculé pour chaque point du plan. Le modèle initial est très imprécis car la plupart des vecteurs sont classés en rouge. Cependant, il devient plus précis après 400 itérations. Enfin, il converge à l'Itération 2000. Comme vous pouvez le constater, la quantité de changement dans les premières itérations est supérieure à celle des dernières itérations. En regardant les dérivées partielles, on se rend compte que le changement d'un paramètre est directement lié à l'erreur de prédiction. Comme l'erreur de prédiction est élevée dans les premières itérations, les paramètres du modèle changent considérablement. Au fur et à mesure que l'erreur diminue, les paramètres changent également légèrement. L'intuition qui se cache derrière la fonction de moindre perte quadratique peut être étudiée sous un autre angle.

Supposons que les deux lignes hypothétiques parallèles au modèle linéaire illustré à la Fig. 2.7. La distance réelle de ces lignes par rapport au modèle linéaire est égale à 1. Dans le cas d'une région négative, la distance signée de la ligne hypothétique est −1. D'autre part, nous avons appris de notre discussion précédente que la distance normalisée des échantillons x à partir de la limite de décision est égale à$\frac{f(x)}{||w||}$ où, iciw désigne le vecteur de paramètre avant d'augmenter. Si on considère la projection de x sur w et utilise le fait que**wx**= $||w||||x||$ cos (θ), nous verrons que la distance non normalisée de l'échantillon x du modèle linéaire est égale à f (x). Sur cette base, la perte des moindres carrés tente de minimiser la somme de la distance non normalisée des échantillons à partir de leur ligne hypothétique réelle.

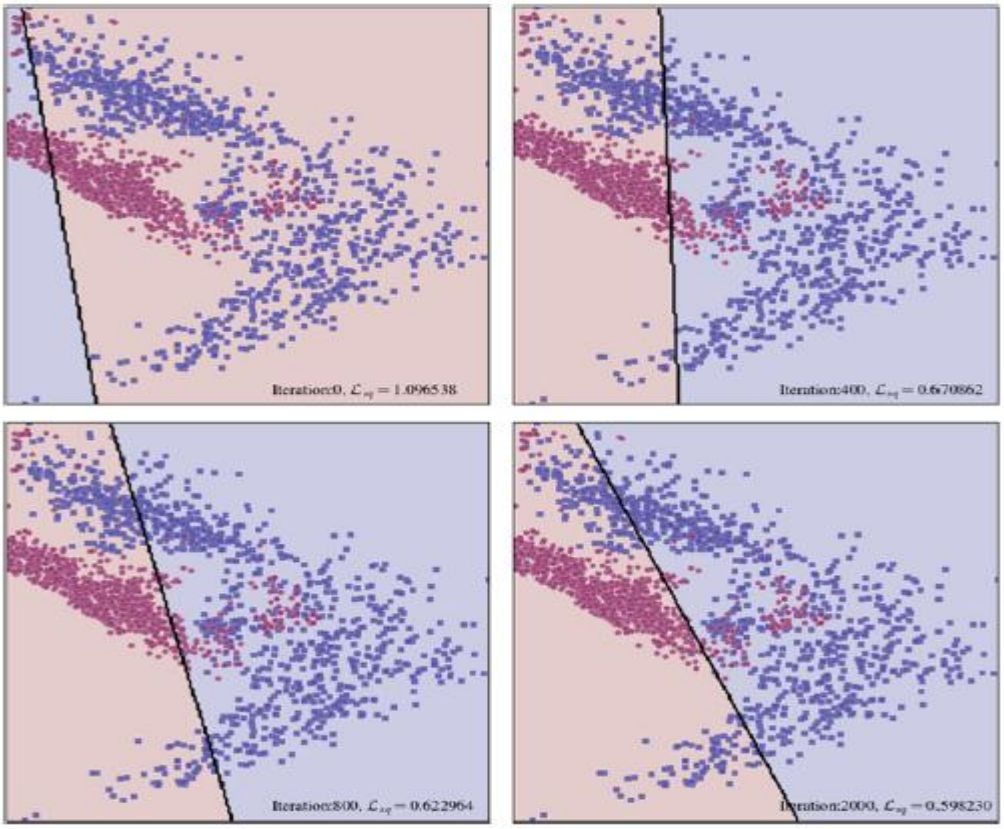


Fig.2.6Le statut du dégradé est divisé en quatre itérations différentes. Le vecteur de paramètre w change grandement dans les premières itérations. Cependant, comme il se rapproche du minimum de la fonction de perte au carré, ça change légèrement.

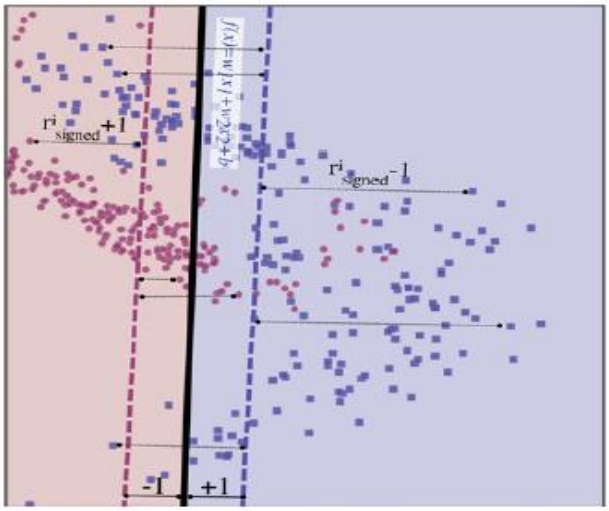


Fig.2.7 L'intuition géométrique derrière la fonction de moindre carré est de minimiser la somme de non normalisée distances entre les échantillons d'apprentissage xi et leur ligne hypothétique correspondante

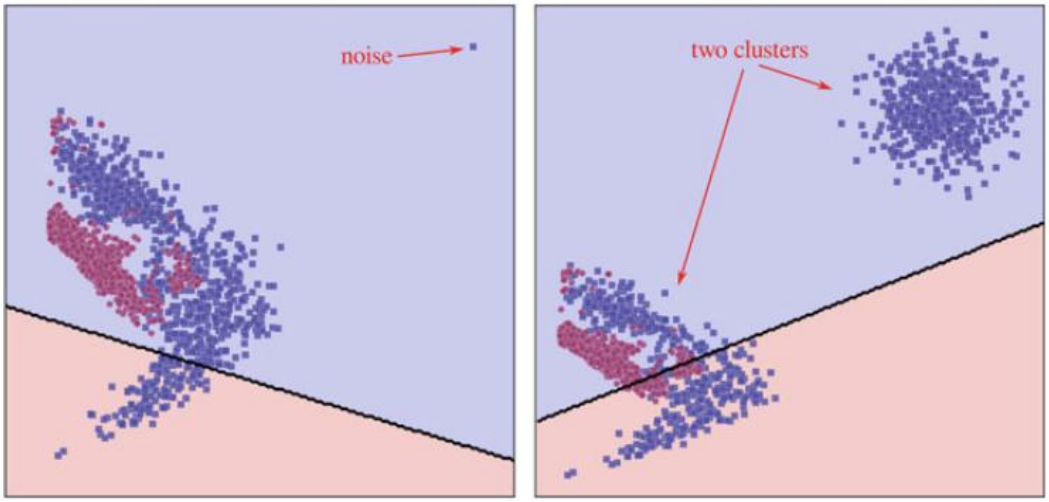


Fig.2.8 La fonction de perte de place peut ne pas correspondre aux données d'entraînementsi des échantillons bruyants se trouvent dans ensemble de données

En pratique, il est également probable que des échantillons propres forment deux ou plusieurs grappes distinctes dans l'espace à trois dimensions. Semblable au scénario des échantillons bruyants, la perte quadratique tente également de minimiser l'erreur de prédiction des échantillons de la grappe lointaine. Comme on peut le voir sur la figure, le modèle linéaire peut ne pas être ajusté avec précision sur les données si des échantillons propres forment deux ou plusieurs grappes distinctes.

Ce problème est dû au fait que la perte au carré ne prend pas en compte l'étiquette de la prédiction. Au lieu de cela, il considère le score de classification et calcule l'erreur de prédiction. Par exemple, supposons les paires d'entraînement :

$$\{(\mathbf{x}a,\ 1),\ (\mathbf{x}b,\ 1),\ (\mathbf{x}c,-1),\ (\mathbf{x}d,-1)\} \quad (2.22)$$

Aussi, supposons deux configurations différentes $w_1$et w_2 pour les paramètres du modèle linéaire avec les réponses suivantes sur le jeu d'apprentissage :

$f_{w1}(x_a)$ = 10	$f_{w2}(x_a)$ = 5
$f_{w1}(x_b)$ = 1	$f_{w2}(x_b)$ = 2
$f_{w1}(x_c)$ = -0.5	$f_{w2}(x_c)$ = 0.2
$f_{w1}(x_d)$ = -1.1	$f_{w2}(x_d)$ = -0.5
$\mathcal{L}sq(w_1)$ = 10.15	$\mathcal{L}sq(w_2)$ = 2.33

(2.23)

En termes de perte au carré, w_2 est meilleur que w_1. Mais si nous comptons le nombre d'échantillons mal classés, nous constatons que $w_1$est la meilleure configuration. En ce qui concerne les problèmes de classification, nous cherchons principalement à réduire le nombre d'échantillons mal classés. En conséquence, $w_1$est favorable à $w_2$dans ce réglage. Afin de résoudre ce problème de la fonction de perte au carré, nous pouvons définir la fonction de perte suivante pour estimer la perte 0/1:

$$\mathcal{L}sg(w_1) = \sum_{i=1}^{n} 1 - sign\big(f(x_i)\big)y_i \quad (2.24)$$

Si f (x) prédit correctement, son signe sera identique au signe de y_i dans lequel leur multiplication sera égale à +1. Ainsi, le résultat de 1 - $signe\ (f\ (xi))y_i$ sera zéro. Au contraire, si f (x) prédit de manière incorrecte, son signe sera différent de y_i. Donc, leur multiplication sera égale à -1. Cela étant le cas, le résultat de 1 - $signe\ (f\ (xi))y_i$sera égal à 2. Pour cette raison, w_{sg} renvoie le double du nombre d'échantillons mal classés.

La fonction de perte ci-dessus a un aspect intuitif et n'est pas sensible aux échantillons lointains. Toutefois, il est difficile de trouver le minimum de cette fonction de perte à l'aide de méthodes d'optimisation basées sur les gradients. La raison est à cause de la fonction de signe. Une solution pour résoudre ce problème consiste à approximer la fonction de signe en utilisant une fonction différentiable. Heureusement, *tanh* (tangente hyperbolique) est capable d'approcher avec précision la fonction de signe.

Plus précisément,*tanh* (kx) ≈ signe (x) lorsque k>> 1.Ceci est illustré à la Fig.2.9

À mesure que k augmente, la fonction *tanh* sera en mesure d'approcher la fonction signe plus précisément.

En remplaçant la fonction de signe par *tanh* dans (2.24), nous obtiendrons :

$$\mathcal{L}sg(w) = \sum_{i=1}^{n} 1 - tanh\big(kf(x_i)\big)y_i. \quad (2.25)$$

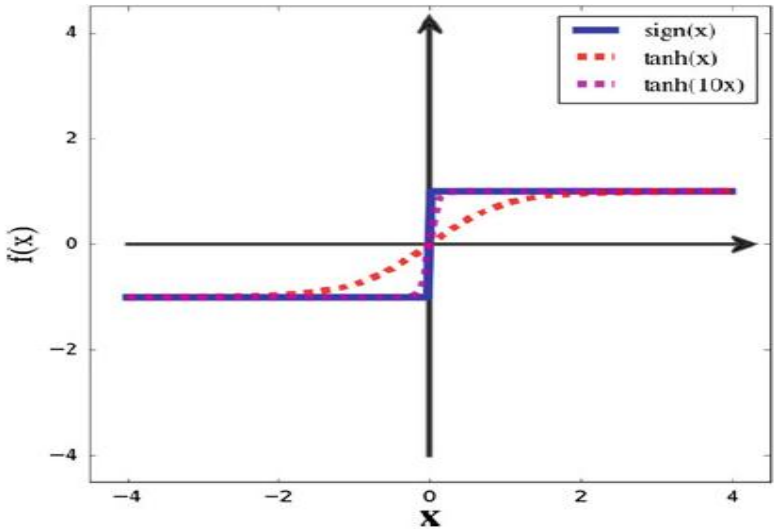


Fig.2.9La fonction de signe peut être avec précision approximée à l'aide de tanh (kx) quand k>> 1

Semblable à la fonction de perte au carré, la fonction de perte de signe peut être minimisée à l'aide de la méthode de descente par gradient. À cette fin, nous devons calculer les dérivées partielles de la fonction de perte de signe par rapport à ses paramètres:

$$\frac{\delta \mathcal{L}sg(w)}{w_i} = - \mathrm{K}x_i y(1 - tanh^2(kf(x))) \tag{2.26}$$

$$\frac{\delta \mathcal{L}sg(w)}{w_0} = - \mathrm{K}y(1 - tanh^2(kf(x)))$$

Si nous entraînons un modèle linéaire en utilisant la fonction de perte de signe et la méthode de descente par gradient sur les jeux de données illustrés aux Fig. 2.1 et 2.8 nous obtiendrons les résultats illustrés à la Fig. 2.10. Selon les résultats, la fonction de perte de signe est capable de traiter des grappes séparées d'échantillons et des valeurs aberrantes par opposition à la fonction de perte au carré.

Bien que la perte de signe utilisant l'approximation *tanh* fasse un assez bon travail sur notre jeu de données exemple, elle a un problème qui ralentit l'optimisation. Afin d'expliquer ce problème, nous devrions étudier la dérivée de la fonction de *tanh*. Nous savons par calcul que $\frac{\delta \tanh(x)}{\delta x} = 1 - \tanh^2(x)$La figure 2.11 montre son graphique. Nous pouvons voir que le dérivé de *tanh* sature comme | x | augmente. En outre, il sature plus rapidement si nous fixons k à un nombre positif supérieur à 1. D'autre part, nous savons d'après (2.26) que le gradient de la fonction de perte de signe dépend directement de la fonction dérivée de *tanh.*

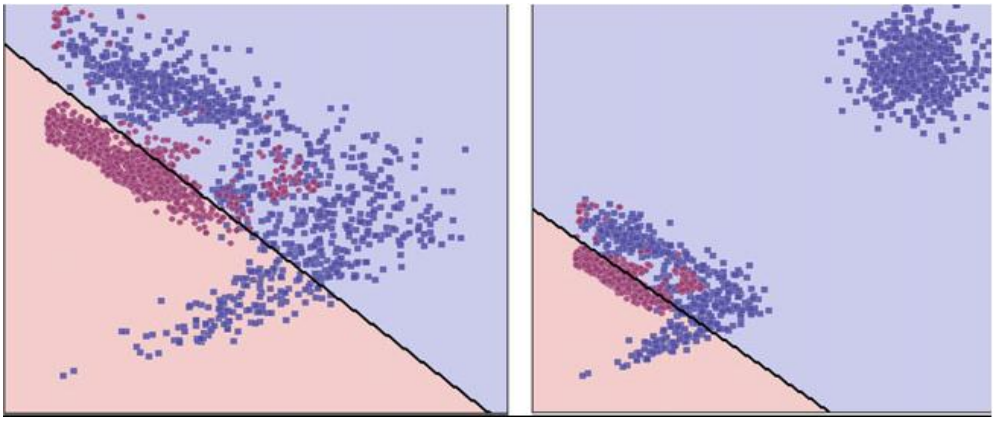


Fig.2.10 La fonction de perte de signe est capable de traiter les problèmes de jeux de données bruyants et de clusters séparés mentionné précédemment

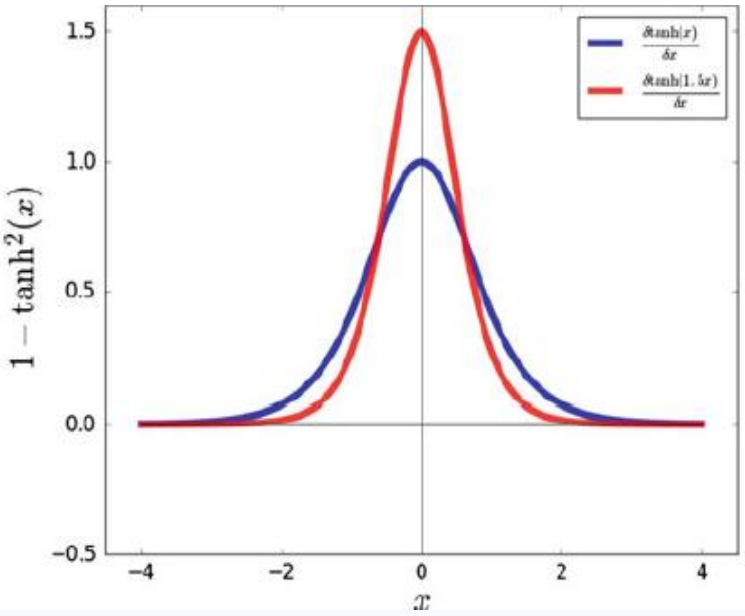


Fig.2.11 Dérivé de La fonction tanh (kx) sature comme | x | augmente. Également rapport de croissance de saturation rapidement quand k> 1

Cela signifie que si la dérivée d'un échantillon tombe dans la région saturée, sa magnitude est proche de zéro. En conséquence, les paramètres changent très légèrement. Ce phénomène, appelé problème de gradients saturés, ralentit la vitesse de convergence de la méthode de descente de gradient. Comme nous le verrons dans les chapitres suivants, dans les modèles complexes tels que les réseaux de neurones avec des millions de paramètres, le modèle peut ne pas être en mesure d'ajuster les paramètres des couches initiales car les gradients saturés sont propagés des dernières couches aux premières couches.

2.2 Perte de charnière

Plus tôt dans ce chapitre, nous avons expliqué que la distance normalisée de l'échantillon x à partir de la limite de décision est égale à $\frac{f(x)}{||w||}$. De même, la marge de x est obtenue en calculant (wx^T) y où y est l'étiquette correspondante de x. La marge nous dit à quel point la classification de l'échantillon est correcte. Supposons que l'étiquette de x_a soit −1. Si wx_a^T a est négatif, sa multiplication avec y = −1 sera positive, ce qui montrera que l'échantillon est correctement classé avec une confiance analogue à | wx^T |. De même, si wx^T a est positif, sa multiplication avec y = −1 sera négative, ce qui montrera que l'échantillon est classé incorrectement avec une magnitude égale à | wx_a^T|.

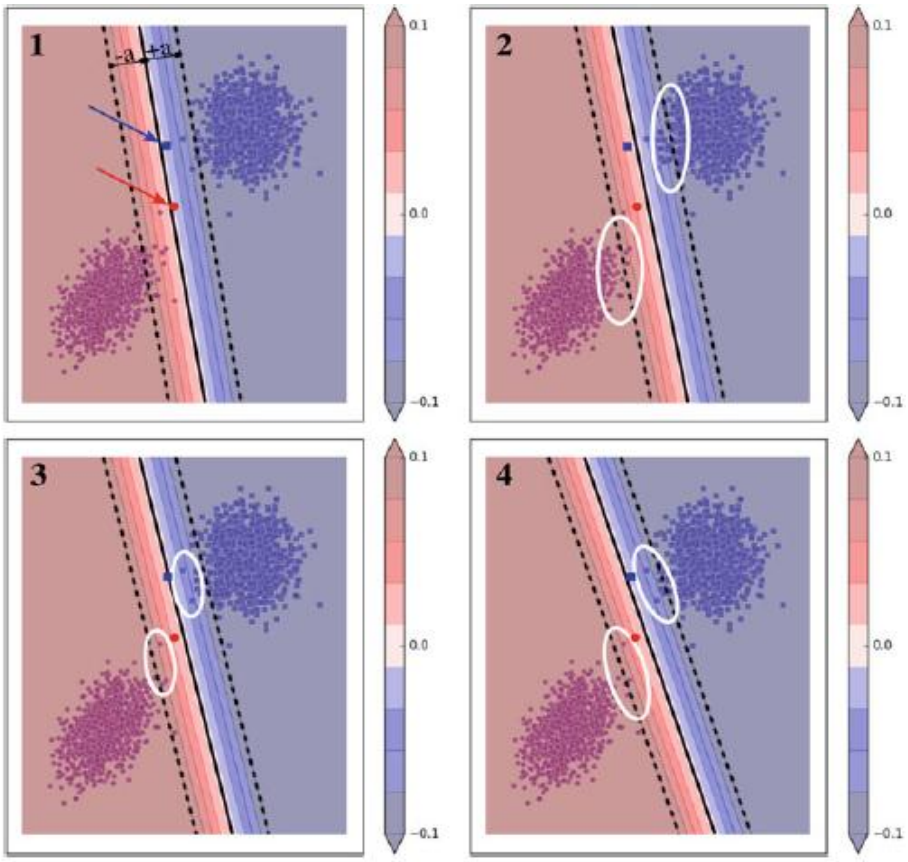


Fig.2.12 La perte de charnière augmente la marge des échantillons pendant que l'on essaie de réduire la classification Erreur. Reportez-vous au texte pour plus de détails

L'idée de base de la perte de charnière n'est pas seulement de former un classificateur, mais également d'augmenter la marge d'échantillons. C'est une propriété importante qui peut augmenter la tolérance du classifieur aux échantillons bruyants. Ceci est illustré à la Fig. 2.12 sur un ensemble de données synthétiques parfaitement séparables à l'aide d'une ligne. La ligne continue montre la décision.

Les limites et les lignes pointillées illustrent les limites de la région critique centrée sur la limite de décision de ce modèle. Cela signifie que la marge des échantillons dans cette région est inférieure à | a |. En revanche, la marge des échantillons en dehors de cette région est élevée, ce qui implique que le modèle est plus confiant dans la classification des échantillons en dehors de cette région. En outre, la barre de couleur à côté de chaque parcelle décrit la marge correspondant à chaque couleur sur les parcelles.

Dans la première parcelle, sont indiqués deux échantillons d'essai qui ne sont pas utilisés pendant la phase d'entraînement. L'un d'eux appartient aux cercles et l'autre aux carrés.

Bien que la ligne ajustée sur les échantillons d'apprentissage permette de distinguer parfaitement les échantillons d'apprentissage, elle classera incorrectement l'échantillon rouge à tester. En comparant le modèle de la deuxième parcelle avec la première, nous observons que moins de cercles se trouvent dans la région critique, mais que le nombre de carrés augmente dans cette région. Dans la troisième parcelle, la marge globale des échantillons est

meilleure si on compare les échantillons marqués avec des ellipses blanches sur ces parcelles. Enfin, la meilleure marge globale se trouve dans la quatrième parcelle où les échantillons de test sont également correctement classés. Il est important de maximiser la marge car cela peut augmenter la tolérance du modèle au bruit. Les échantillons de test de la Fig. 2.12 peuvent être des échantillons bruyants. Toutefois, si la marge du modèle est grande, il est probable que ces échantillons sont classés correctement.

Néanmoins, il est toujours possible que nous concevions un scénario de test dans lequel la première parcelle pourrait être plus précise que la quatrième. Mais, à mesure que le nombre d'échantillons d'apprentissage augmente, un classifieur avec une marge maximale est susceptible d'être plus stable. Maintenant, la question est de savoir comment nous pouvons forcer le modèle par une fonction de perte à augmenter simultanément sa précision et sa marge. La fonction de perte de charnière atteint ces objectifs en utilisant la relation suivante:

$$\mathcal{L}_{\mathrm{hnge}}(\mathrm{w}) = \frac{1}{\mathrm{n}}\sum_{\mathrm{i}=1}^{\mathrm{n}} \max\left(0, \mathrm{a} - \mathrm{wx_i^T y_i}\right) \qquad (2{,}27)$$

Où yi ∈ {−1, 1} est l'étiquette de l'échantillon d'apprentissage x_i. Si les signes de wx_i et de yi sont égaux, le terme à l'intérieur de l'opérateur de somme retournera 0 puisque la valeur du deuxième paramètre de la fonction max sera négative. En revanche, si leur signe est différent, ce terme sera égal à$a - wx_i^T y_i$augmentant la valeur de la perte.

De plus, si$wx_i^T y_i$<a, cela implique que x est dans la région critique du modèle et augmente la valeur de la perte. En minimisant la fonction de perte ci-dessus, nous obtiendrons un modèle avec une marge maximale et une grande précision en même temps. Le terme à l'intérieur de l'opérateur sum peut être écrit de la manière suivante:

$$\max\left(0, \mathrm{a} - \mathrm{wx_i^T y_i}\right) = \begin{cases} \mathrm{a} - \mathrm{wx_i^T y_i} & \mathrm{wx_i^T y_i} < a \\ 0 & \mathrm{wx_i^T y_i} \geq \mathrm{a} \end{cases} \qquad (2{,}28)$$

En utilisant cette formulation et en notant $\max(0, a - wx_i^T y_i)$ avec H, nous pouvons calculer les dérivées partielles de $\mathcal{L}_{hinge}(w)$ par rapport à w:

$$\frac{\delta H}{\delta w_i} = \begin{cases} -x_i y_i & \boldsymbol{wx_i^T y_i} < a \\ 0 & \boldsymbol{wx_i^T y_i} \geq \boldsymbol{a} \end{cases} \qquad (2.29)$$

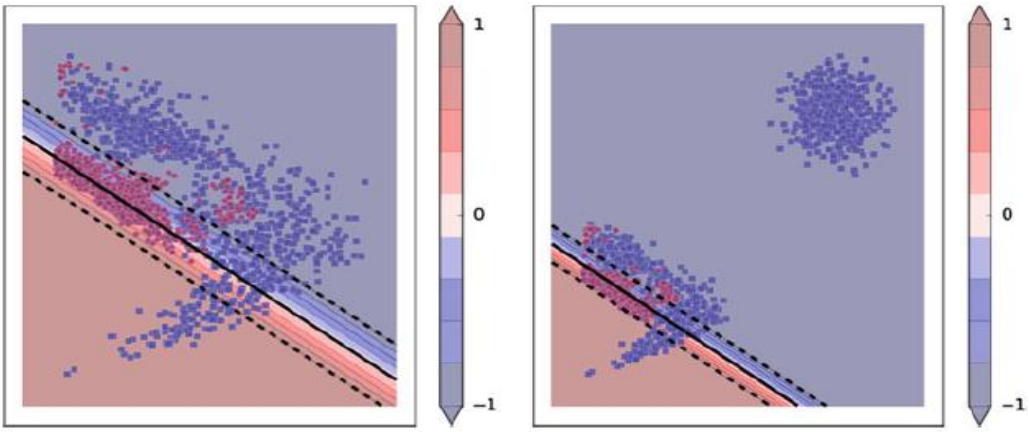


Fig.2.13Formation d'un classifieur linéaire à l'aide de la fonction de perte de charnière sur deux jeux de données différents

2.3 Régression logistique

Aucun des modèles linéaires mentionnés précédemment ne permet de calculer la probabilité que des échantillons x appartiennent à la classe y = 1. Formellement, étant donné un problème de classification binaire, nous pourrions être intéressés par le calcul de p (y = 1 | x). Cela implique que p (y = −1 | x) = 1 - p (y = 1 | x). Par conséquent, l'échantillon x appartient à la classe 1 si p (y = 1 | x)> 0,5.

Sinon, il appartient à la classe -1. Dans le cas où p (y = 1 | x) = 0,5, l'échantillon se situe exactement sur la limite de décision et n'appartient à aucune de ces deux classes.

L'idée de base de la régression logistique est d'apprendre p (y = 1 | x) à l'aide d'un modèle linéaire.

À cette fin, la régression logistique transforme le score d'un échantillon en probabilité en le faisant passer par une fonction sigmoïde. Formellement, la régression logistique calcule la probabilité postérieure comme suit:

$$p\,(y = 1 \mid x; w) = \sigma\,(wx^{T}) = \frac{1}{1+e^{-wx^{T}}}.(2{,}30)$$

$$p\,(y = 1 \mid x; w) = \sigma\,(wx^{T}) = \frac{1}{1+e^{-wx^{T}}}.\ (2{,}31)$$

Dans cette équation, σ: $\mathbb{R} \rightarrow [0, 1]$ est la fonction logistique sigmoïde. Comme le montre la figure 2.14, la fonction a une forme en S et sature comme | x | augmente. En d'autres termes, la dérivée de la fonction s'approche de zéro comme | x | augmente.

Comme la plage de la fonction sigmoïde est [0, 1], elle répond aux exigences d'une fonction de mesure de probabilité. Notons que (2.31) modélise directement la probabilité a posteriori, ce qui signifie qu'en utilisant des techniques appropriées que nous expliquerons plusloin, il est possible de modéliser la vraisemblance et a priorités classes. Compte tenu du fait que (2.31) renvoie la probabilité d'un échantillon,fonction de perte doit également être

construite à partir deprobabilité de l'ensemble de la formation donnée un w spécifique. Officiellement, étant donné un ensemble de données de n échantillons d'entraînement, notre objectif est de maximiser leur probabilité conjointe définie comme suit :

$$\mathcal{L}_{logistique}(w) = p(x_1 \cap x_1 \dots \dots \dots \cap x_n) = p\left(\bigcap_{i=1}^{n} x_i\right) \quad (2.32)$$

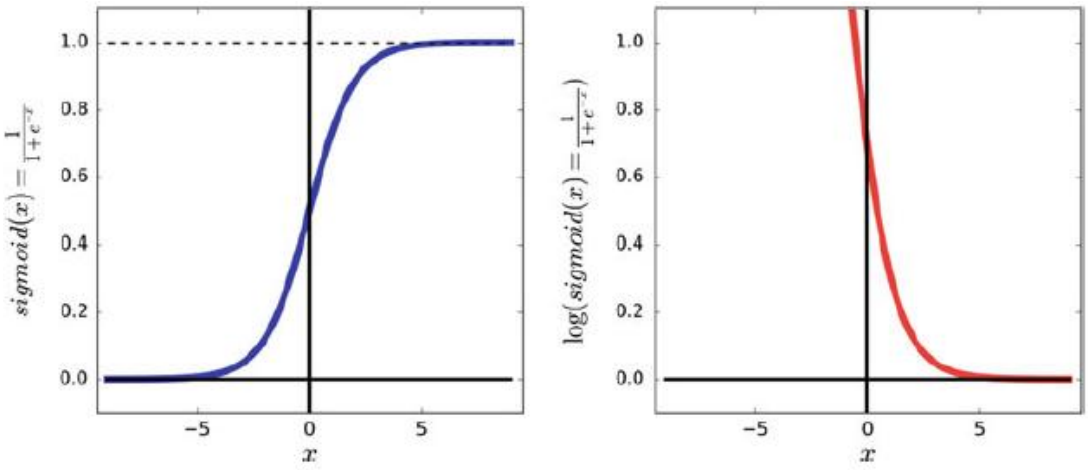


Fig.2.14 Tracé de la fonction sigmoïde (à gauche) et logarithme de la fonction sigmoïde (à droite). Le domaine de la fonction sigmoïde est constitué de nombres réels et sa plage est [0, 1]

La modélisation de la probabilité conjointe ci-dessus n'est pas triviale. Cependant, il est possible de décomposer cette probabilité en composantes plus petites. Pour être plus précis, la probabilité de xi ne dépend pas de la probabilité de xj. Pour cette raison et en tenant compte du fait que p (A, B) = p (A) p (B) si A et B sont des événements indépendants, nous pouvons décomposer la probabilité conjointe ci-dessus en produit de probabilités:

Logistique (w) = n i = 1 p (yi | xi) (2,33)

où p (xi) est calculé en utilisant:

p (yi | xi) = p (y = 1 | x; w) yi == 1 1 - p (y = 1 | x; w) yi == −1 (2,34)

Représentant la classe négative avec 0 plutôt que −1, l'équation ci-dessus peut êtreécrit comme:

p (xi) = p (y = 1 | x; w) yi (1 - p (y = 1 | x; w)) 1 − yi. (2.35)

Cette équation, appelée distribution de Bernoulli, est utilisée pour modéliser des variables aléatoires avec deux résultats

2.4 Classification multiclasse

Dans la section précédente, nous avons mentionné quelques techniques pour former un classifieur linéaire sur des problèmes de classification binaire. Rappelez-vous de la section précédente que dans un problème de classification binaire, notre objectif est de classer l'entrée $x \in R^d$ dans l'une des deux classes.

Un problème de classification multi-classe est un concept plus généralisé dans lequel x est classé dans plus de deux classes. Par exemple, supposons que nous voulions classer 10 différents panneaux de limitation de vitesse allant de 30 à 120 km / h. Dans ce cas, x représente l'image d'un panneau de limitation de vitesse. Ensuite, notre objectif est de trouver le modèle $f: R^d \rightarrow y$où y = {0, 1, ..., 9}. Le modèle $f(x)$ accepte un vecteur réel de dimension d etrenvoie un entier catégorique compris entre 0 et 9. Il est à noter que y n'est pas un ensemble ordonné. Cela peut être n'importe quel jeu avec 10 symboles différents. Cependant, dans un souci de simplicité, nous utilisons habituellement des nombres entiers pour afficher les classes.

2.5 Un contre un

Un classifieur multi-classe peut être construit en utilisant un groupe de classificateurs binaires. Par exemple, supposons le problème de classification à 4 classes illustré à la Fig. 2.20 où y = {0, 1, 2, 3}. Une technique permettant de construire un classifieur multi-classe à l'aide d'un groupe de classifiants binaires est appelée un contre un (OVO) (*one-versus-one*). Étant donné le jeu de données$\mathcal{H}$ = $\{(x_0, y_0), \ldots, (x_n, y_n)\}$ où $x_i \in : R^d$ et $y_i \in \{0, 1, 2, 3\}$, nous sélectionnons d'abord les échantillons de$\mathcal{H}$ avec l'étiquette 0 ou 1. Formellement, nous créons le jeu de données suivant:

$$x_{0|1} = \{x_i | x_i \in X \wedge y_i \in \{0,1\}\} \quad (2.36)$$

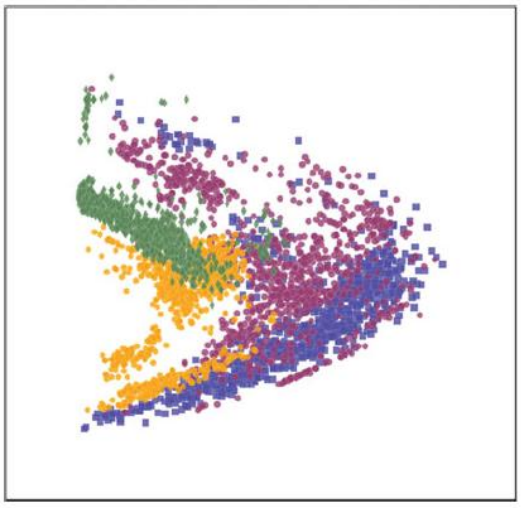


Fig.2.15 Un échantillon de données y compris quatre différents Des classes. Chaque classe est montrée en utilisant une couleur unique et forme

est un classificateur binaire est monté sur $X_{0|1}$. De même, $X_{0|2}$, $X_{0|3}$, $X_{1\,|\,2}$, $X_{1\,|\,3}$ et $X_{2\,|\,3}$ sont créés et des classificateurs binaires distincts sont installés sur chacun d'entre eux. De cette manière, il y aura six classificateurs binaires. Afin de classer la nouvelle entrée x_q dans l'une des quatre classes, elle est d'abord classée à l'aide de chacun de ces 6 classificateurs. Nous savons que chaque classificateur donnera un nombre entier compris entre 0 et 3. Puisqu'il y a six classificateurs, l'un des nombres entiers sera répété plus que d'autres. La classe de x_q est égale au nombre avec l'occurrence la plus élevée. D'un autre point de vue, nous pouvons considérer le résultat de chaque classificateur binaire comme un vote. Ensuite, la classe gagnante est celle avecmajorité des voix. Cette méthode de classification est appelée vote à la majorité. La figure 2.16 montre six classificateurs binaires formés sur six paires de classes mentionnées ci-dessus. En outre, il illustre comment les points du plan sont classés dans l'une des quatre classes utilisant cette technique.

Cet exemple peut facilement être étendu à un problème de classification multiclass avec N classes. Plus précisément, toutes les paires de classes $Xa \mid b$ sont générées pour tout a = 1 ... N - 1 et b = a + 1 ... N. Ensuite, un modèle binaire $f\ a \mid b$ est ajusté sur le jeu de données correspondant. De cette façon, $\frac{N\,(N-1)}{2}$classificateurs binaires seront formés. Enfin, un échantillon invisible x_q est classé en calculant la majorité des votes produits par tous les classificateurs binaires.

L'un des problèmes évidents entre une technique et une technique est que le nombre de classificateurs binaires augmente de façon quadratique avec le nombre de classes d'un ensemble de données. Cela signifie qu'en utilisant cette technique, nous devons former des classificateurs binaires 31125 pour un problème de classification de 250 classes tel que la classification des panneaux de signalisation. Cela rend l'approche une contre une impraticable pour les grandes valeurs de N. De plus, des résultats parfois ambigus peuvent être générés par une technique contre une. Cela peut arriver quand il y a deux classes ou plus avec la majorité des voix. Par exemple :

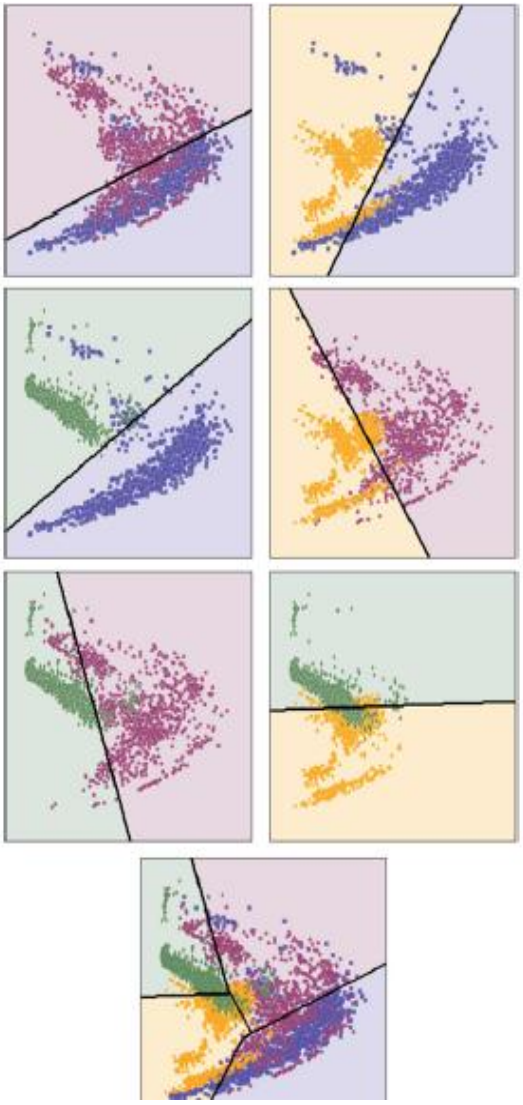


Fig.2.16 Formation de six classificateurs sur le problème de classification en quatre classes.

2.6 Un contre les autres

Une autre approche populaire pour construire un classifieur multi-classe à l'aide d'un groupe de classificateurs binaires est appelée un contre reste (OVR) (one versus rest). Cela peut aussi s'appeler approche un contre tous. Contrairement à l'approche un contre un où $\frac{N(N-1)}{2}$classificateurs binaires sont créés pour un problème de classification de classe N, l'approche un contre repos entraîne uniquement N classificateurs binaires à faire des

prédictions. La principale différence entre ces deux approches réside dans la manière dont elles créent les jeux de données binaires.

Dans la technique (one versus rest), un jeu de données binaire pour la classe a est créé comme suit:

$$X_{a|reste} = \{(x_i, 1)|x_i \in X \wedge y_i = a\} \cup \{(x_i, -1)|x_i \in X \wedge y_i \neq a\} \quad (2.37)$$

Littéralement, $X_{a|reste}$ est composé de tous les échantillons de X. La seule différence est l'étiquette des échantillons. Pour créer $X_{a|reste}$, nous sélectionnons tous les échantillons de X avec l'étiquette a et nous les ajoutons à $X_{a|reste}$ après avoir changé leur étiquette en 1. Ensuite, l'étiquette de tous les échantillons restants dans X est remplacée par -1 et ils sont ajoutés. à$X_{a|reste}$. Pour un problème de classification de classe N, $X_{a|reste}$ est généré pour tous a = 1 ... N. Enfin, un classificateur binaire $f_{a|rest}(x)$ est formé sur chaque $X_{a \mid reste}$ en utilisant la méthode que nous avons précédemment mentionnée dans ce chapitre. . Un échantillon non vux_q est classé par calcul:

$$\hat{y}_q = \arg max_{a=1.....N} f_{a|reste}(x_q) \quad (2.38)$$

En d'autres termes, le score de tous les classificateurs est calculé. Le classificateur avec le score maximum indique la classe de l'échantillon x_q.

En comparant les résultats d'un à un et d'un à tous, nous constatons qu'ils ne sont pas identiques. L'un des avantages de l'approche un contre repos sur une approche est que le nombre de classificateurs binaires augmente de façon linéaire avec le nombre de classes.

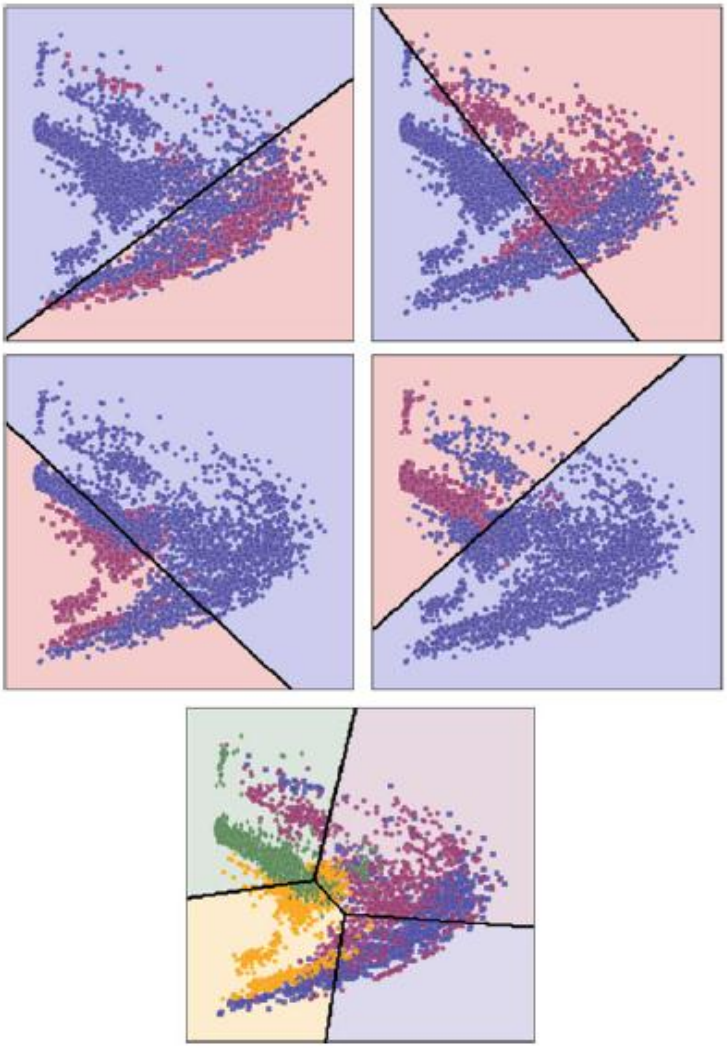


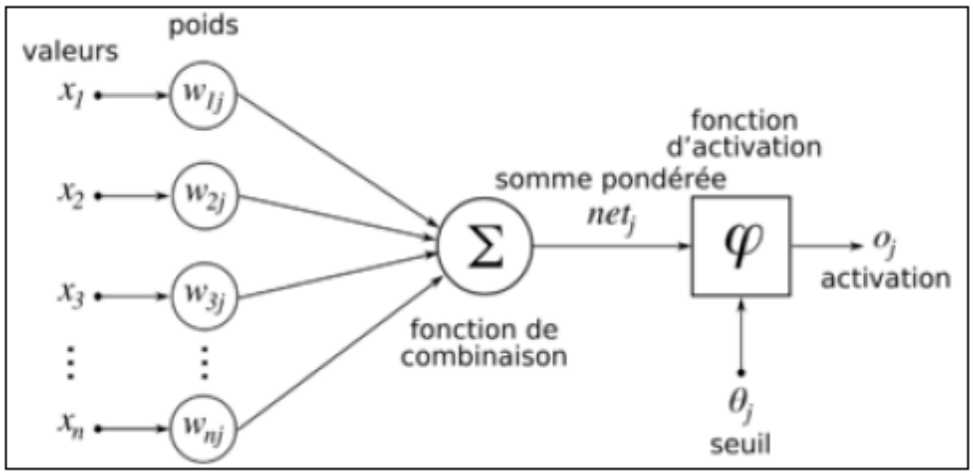
Fig. 2.17 L'approche un contre les autres crée un jeu de données binaire en modifiant l'étiquette de la classe d'intérêt à 1 et le libellé des autres classes à −1.

3. Réseaux de neurones artificiels

Nous allons 0commencer par définir l'unité de base appelée « Perceptron » qui compose unréseau de neurones.

3.1 Perceptron

Les perceptrons ont été développées dans les années 50 par le scientifique Frank Rosenblattinspirés par las travaux de Warren McCulloch et Walter [14]. Un perceptronreprésente une unité de traitement qui reçoit des données en entrée, sous la forme d'un
vecteur, et produit une sortie réelle [15]. Cette sortie est une fonction des entrées et des poids de connexions comme illustrée dans la figure ci-dessous



.

Fig.2.18 La structure interne d'un perceptron [15]

Formellement, un perceptron est défini comme suit [16] :

$(\{x_1, x_2, \dots, _j\}) = f(X) = g(\sum_j w_j x_j)$

où $\{x_1, x_2, \dots, _j\}$ représentent les vecteurs d'entrée, $\{w_1, w_2, \dots, w_j\}$ les poids de connexion, $f(X)$ est la sortie du perceptron et $g(\,.\,)$ une fonction d'activationqui est définie, dans le cas d'un perceptron, comme suit :

$$(\sum_j w_j x_j) = \begin{cases} 1 \quad \text{si } \sum_j w_j x_j >= \text{seuil} \\ 0 \text{ si } \sum_j w_j x_j < \text{seuil} \end{cases}$$

Pour simplifier la façon dont nous décrivons le perceptron, nous pourrons apporter deux changements : le premier changement consiste à écrire $\Sigma_j w_j x_j$sous la forme d'un produit scalaire de deux matrices $W.\,X = \Sigma$.

Le second changement consiste à déplacer le seuil de l'autre côté de l'inégalité, et de le remplacer par ce que l'on appelle le biais du perceptron $b = -$.
Ainsi, un perceptron peut être réécrit comme suit:

$$f(X) = g(W.\,X + b)$$
$$g(W.\,X + b) = \begin{cases} 1 \quad si W.\,X + b \geq 0 \\ 0 \quad si W.\,X + b < 0 \end{cases}$$

Le perceptron comme présenté ci-dessus présente plusieurs limitations. L'une des limitationsmajeures est que la sortie du perceptron est discrète. Elle est soit 0 ou 1. En effet, ce perceptron ne fournit pas une sortie continue probabiliste et cela le rend beaucoup moins flexible. Pour pallier ce problème, la fonction d'activation définie auparavant sera remplacée par la fonction sigmoïde qui est définie par :

$\sigma(z) = \frac{1}{1+e^{-z}}$

Cette nouvelle sortie sera une valeur continue comprise entre 0 et 1.
La nouvelle formulation mathématique est la suivante :

$$(y = 1|X) = \sigma(W.\ X + b) = (X)0$$

$$(y = 0|X) = 1 - (W.\ X + b) = 1 - (X)$$

Avec σétant la fonction sigmoïde.
Ce nouveau neurone peut être interprété comme un classificateur binaire qui estime $(y = 1|X)$ (c'est-à-dire la probabilité de x appartenant à la classe 1). Il est capable d'approximerquelques fonctions telles que les fonctions booléennes suivantes : AND, OR, NAND, NOR mais pas le XOR.
Pour pouvoir approximer le XOR et le reste de fonctions non linéaire, on construit un ensemble de perceptrons appelé communément perceptron multicouche (multilayer perceptron MLP en anglais) ou simplement un réseau de neurones .

3.2 Perceptron multicouche ou réseau de neurones

Dans cette partie, nous allons introduire un réseau de neurones qui permet, théoriquement, d'approximer n'importe quelle fonction non linéaire.

3.2.1 Définition :

Un réseau de neurones est un graphe valué orienté, composé d'une succession de couches dont chacune prend ses entrées sur les sorties de la précédente. Chaque couche est composée de (i)unités ou neurones, prenant leurs entrées sur les $N_{(i-1)}$ neurones de la couche précédente.

Chaque entrée peut être représentée par un vecteur de descripteurs/caractéristiques (ou featuresen anglais) à valeurs quantitatives ou qualitatives.

Les caractéristiques extraites de la première couche sont des caractéristiques de bas niveauPour obtenir des représentations de niveau supérieur du vecteur d'entrée, plusieurs couches peuvent être ajoutées. A chaque couche, les représentations (c'est-à-dire les caractéristiques) de la couche précédente sont combinées avec un ensemble de poids pour coder une représentation plus abstraite. A condition que le réseau soit suffisamment profond, les représentations de la couche cachée finale sont considérées comme des caractéristiques de haut niveau extraites de l'entrée.

Dans un réseau de neurones, la première couche est appelée couche d'entrée, la dernière couche est appelée couche de sortie et toutes les autres couches intermédiaires sont appelées couches cachées. Les neurones dans une couche cachée sont appelés unités cachées.
La Figure 2.19 illustre une représentation graphique d'un réseau de neurones multicouche

.

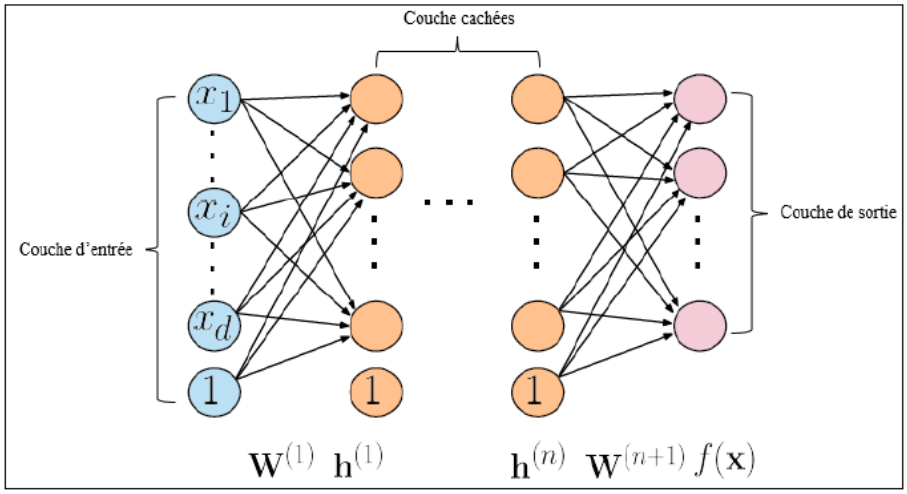


Fig. 2.19Architecture d'un réseau de neurones.

3.3Apprentissage dans les réseaux de neurones

Les réseaux de neurones sont des modèles paramétriques. Comme vu précédemment, ce type de modèle apprend via l'algorithme de descente de gradient.

Le problème qui se pose maintenant c'est comment calculer ces gradients efficacement. Pour ce faire, l'algorithme de **rétropropagation**est utilisé. Il permet de calculer les gradients du réseau en démarrant de la dernière couche puis en les propageant vers les couches inférieures, d'où le nomrétropropagation.

3.4L'algorithme de rétropropagation

L'algorithme de rétropropagation est le suivant [16] :
- Initialiser tous les paramètres du réseau avec des valeurs aléatoire entre 0 et 1.
- Pour chaque itération :

- À partir de la couche d'entrée, faire une **passe en avant** (ou forwardpass en anglais) à travers le réseau, en calculant les activations des neurones à chaque couche.

- Calculer le gradient de la fonction perte par rapport aux activations de la couche de sortie.

- Calculer le gradient de la fonction perte par rapport à l'entrée linéaire des neurones dans la couche supérieure.

- Calculer le gradient de la fonction perte par rapport aux paramètres de la couche actuelle.

- Calculer le gradient de la fonction perte par rapport aux activations des neurones dans la couche actuelle.

- Faire une **passe en arrière** (ou backwardpass en anglais) pour propager les gradients calculés et mettre à jours les paramètres du réseau.

3.5 Autres fonctions d'activation

Nous distinguons deux types de fonctions d'activation :

3.5.1Les fonctions d'activation sur la couche de sortie

Sur la couche de sortie, comme nous nous intéressons aux problèmes de classification ou de segmentation, nous aimerions bien avoir en sortie des valeurs continues ayant la propriété d'une loi de probabilité, i.e. la somme des valeurs de chaque neurone de sortie est égale à 1, et ainsi nous pourrons définir la sortie du $n_{ième}$ neurone comme étant la probabilité d'appartenance à la $n_{ième}$ classe.

Pour une classification binaire, la fonction sigmoïde comme définie auparavant remplit la Condition d'une loi de probabilité et donc sur la couche de sortie il y'aura un seul neurone avec sigmoïde comme fonction d'activation.

Pour une classification multi classes, la couche de sortie contient autant de neurones qu'il y a de classes et sigmoïde ne permet pas d'avoir une loi de probabilité.

Une autre fonction notée **Softmax**est utilisée, mathématiquement elle s'écrit comme suit (I.Goodfellow et al., 2016) :

$$(y = i|x) = Softmax(xi\ ) = \frac{e^{xi}}{\sum_{j=1}^{c} e^{xi}} pour i = 1, \dots, C$$

oùx_iÉtant la sortie pré-activation du $i_{éme}$neurone et Cest le nombre total de classes.

4. Réseau de neurones convolutifs

4.1 Mécanisme global de fonctionnement

Les réseaux de neurones convolutifs (CNN) ont été inventés dans le but de résoudre le problème **d'invariance à la translation.** En effet, de nombreuses caractéristiques dans une image sont invariantes par la translation. Par exemple, si l'on veut détecter une voiture dans une image, onpourrait essayer de la détecter par ses parties. Mais cela s'avère difficile puisqu'il y'a beaucoup de positions où les roues pourraientêtre.

Le principe de fonctionnement de ce type de réseau est simple : ça consiste en l'extraction automatique de différentes caractéristiques à partir de l'image d'entrée en appliquant des opérations dites **opérations de convolutions** à l'aide des filtres dont les valeurs sont des paramètres que le réseau apprend via l'algorithme de la décente de. La façon dont les CNN sont construits leur permet de [17] :

• Capturer des caractéristiques de bas niveau qui sont invariantes par la translation sur les couches inférieures du réseau et de capturer sur les couches supérieures, descaractéristiques de haut niveau qui sont eux même des combinaisons d'autrescaractéristiques de bas niveau. Cette hiérarchie de caractéristiques capturées sur les couches supérieures du réseau lui permet de capturer plus facilement les parties importantes des données.

• Prendre en considération le fait que les pixels dans une image sont ordonnés.
Une architecture de réseau de neurones convolutifs est formée typiquement par un empilementde trois types de couches de traitement, à savoir :

1 Couche de convolution avec une fonction d'activation non linéaire.
2 Couche de « pooling ».
3 Couche de normalisation.

4.2 Différentes couches composant le CNN

4.2.1 Couche de convolution

Comme dans les réseaux de neurones standard, plusieurs couches convolutives peuvent êtreempilées les unes sur les autres pour former une hiérarchie de caractéristiques, Chaque couchepeut être vue comme une extraction de caractéristiques à partir de la couche qui la précède dans la hiérarchie.

La couche de convolution prend en entrée plusieurs carte de caractéristiques (featuremap en anglais) et produit $n$featuremaps en sortie ou $n$est le nombre de filtres qu'on lui a appliqué. Chaque featuremap est une matrice de deux dimensions $m \in Rl*h$ou $l$représente la largeur et $h$représente la hauteur de cette matrice.

Dans le cas d'une première couche de convolution, l'entrée de cette couche correspond aux différents canaux de l'image en entrée. Ces canaux peuvent être par exemple, dans le cas d'une image RGBles 3 couleurs.

Les cartes de caractéristiques sont calculées moyennant une opération dite opération de **convolution**.

Opération de convolution : Chaque featuremap$O_s$est associé avec un filtre (ou kernel), $O_s$est calculé comme suit :

$O_s = b_s + \sum_r w_{sr} * x_r$
où : Xrest le r^{ième}canal d'entrée,
W_{sr}Est le sous filtre pour ce canal, $b_s$est le biais,
$*$ Est l'opération de convolution.

En d'autres termes, l'opération effectuée sur chaque featuremap est la somme de l'application de Rdifférents filtres de convolution bidimensionnels de taille N × N (un par canal) plus un biais qui est ajouté à chaque pixel comme l'illustre l'exemple montré par la Figure **2.20**L'opération de convolution de l'image Xet le filtre West calculé comme suit :

$$Cij = (W*X)ij = \sum\sum \mathrm{Xi + m, j + nW - m, -n}$$

Dans l'équation ci-dessus, la région de la matrice Xutilisée pour le calcul de Cijest appelée **champ réceptif local** (local receptivefield en anglais) (**Fig.2.20**) pour Cij, et donc Cijest seulement connectée à son champ réceptif, plutôt qu'à l'image entière comme c'était le cas dans les réseaux de neurones standards, ce qui permet de réduire considérablement le nombre

de paramètres du modèle. Ce champ réceptif est ensuite translaté spatialement avec un pas (stride en anglais) Safin de couvrir toute l'image.

Un autre concept crucial qui définit les CNNs est le **partage des paramètres**. Traditionnellement dans le cas d'un MLP, chaque neurone d'une couche cachée est connectéàtous les autres neurones de sa couche précédente. Avec le partage des paramètres chaque
neurone de la carte de caractéristiques est connecté seulement à une région de l'entrée dite champ réceptif. Intuitivement, la raison du partage des paramètres est que chaque filtre peut être considéré comme un détecteur de caractéristiques qui essaie d'identifier cette caractéristique particulière peu importe sa position spatiale dans l'image.

Les hyperparamètres de la couche de convolution sont :
• Le nombre total de filtres $n$ou la profondeur de la couche.
• Le couple (K_l, K_h) qui définitla taille de filtre à appliquer.
• La fonction d'activation.
• Le stride : définit le pas avec lequel le filtre est déplacé pour couvrir toute l'image. Dans les deux exemples montrés dans la Figure 2.19 et la Figure **2.20** lestride est 1.
• Le padding : la taille de la carte de caractéristique diminue en sortie de chaque couche de convolution, c'est alors pour préserver la taille initiale qu'a été introduit lehyperparamètre « Padding » qui a pour but de rajouter des bordures de '0' à l'image ensortie afin dereconstruire la taille initiale comme l'illustre la Figure **2.20**

La sortie d'une couche de convolution aurait une taille de $O \times O$
Avec : $O = \frac{\mathrm{I+2P-K}}{s}+1$et $I \times I$étant la taille de la carte caractéristique en entrée.
Il est important de noter que la couche de convolution est toujours suivie d'une fonctiond'activation permettant ainsi d'introduire la non-linéarité dans le modèle

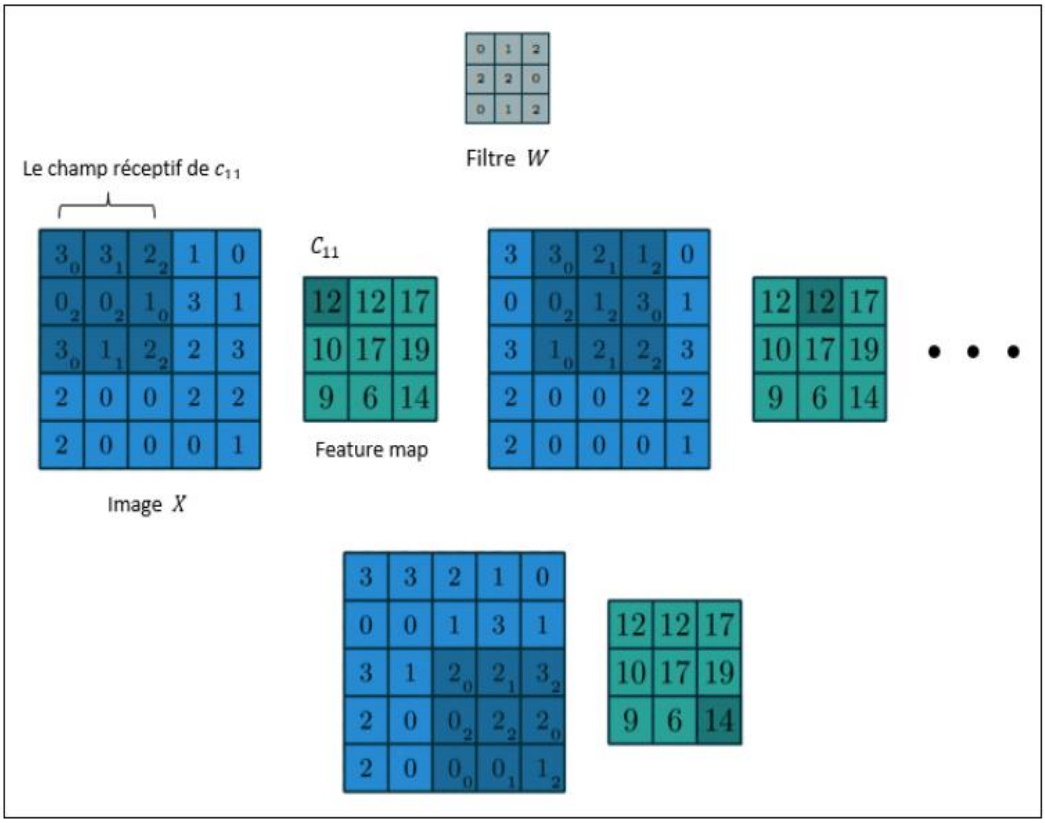


Fig.2.20 Exemple d'une opération de convolution d'une image de taille 5×5 avec un filtre 3×3, Stride=1 et Padding=0.

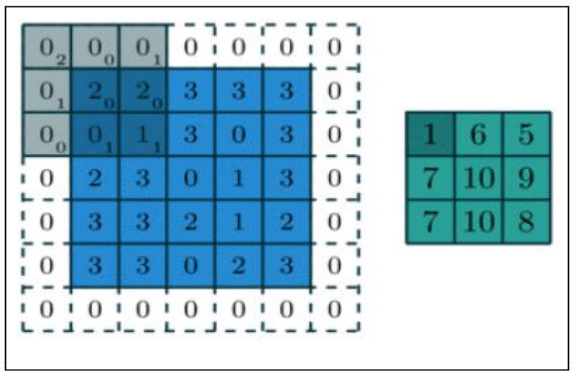


Fig.2.21 Exemple d'une opération de convolution d'une imagede taille 5×5avec un filtre 3×3, Stride=1 et Padding=1.

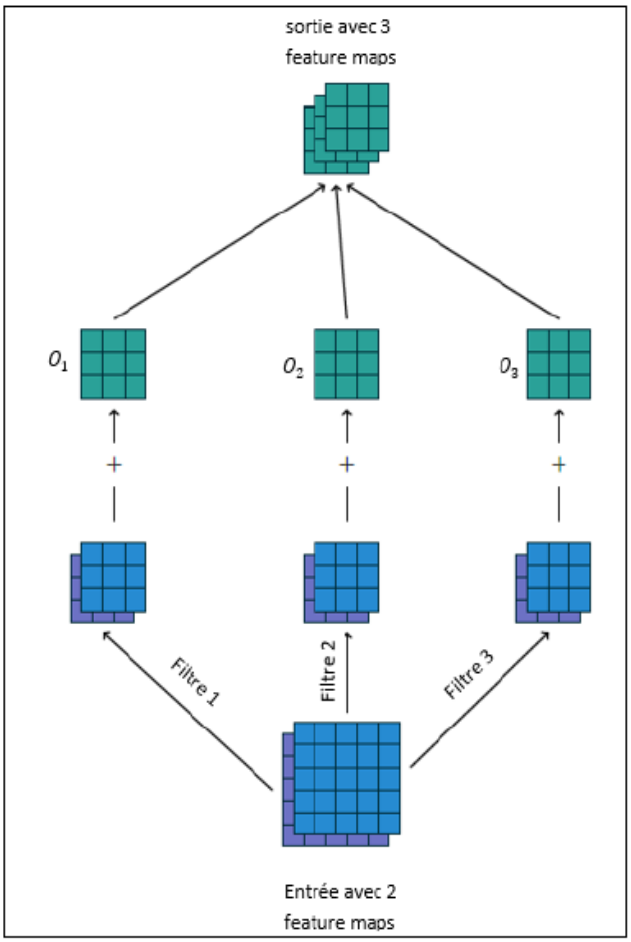


Fig.2.22 Dans cet exemple 3 filtres différents sont appliqués à une entrée avec 2 featuremapsproduisant ainsi une sortie avec 3 featuremaps. Pour le premier filtre, la première featuremap de l'entrée est convoluée avec le sous filtre $W11$ et la deuxième featuremap est convoluée avec le sous filtre $W12$,les résultats de ces deux opérations sont additionnés ensemble avec le biais pour former la première featuremap de sortie.de la même manière, les deux autres featuremaps de sortie sont form

4.3 Exemple d'architectures CNN

La forme la plus commune d'une architecture de réseau de neurones convolutifs empile quelques couches de convolution, les suit avec des couches de max pooling, et répète ce schéma jusqu'à ce que l'entrée soit réduite dans un espace d'une taille suffisamment petite. À un moment, il est fréquent de placer des couches entièrement connectées[3] (fullyconnected FC). La dernière couche entièrement connectée est reliée vers la sortie. Voici quelques architectures connues de réseau de neurones convolutifs qui suivent ce modèle :

- **LeNet**: Ce fut la première architecture utilisée pour reconnaitre les caractères manuscrits, Elle a été développée par Yann LeCun en 1998 (voir**Fig.2.23**). Entre 2000 et 2012 durant ces années les CNNs étaient en incubation, alors que de plus en plus de données et de puissance de calcul soient disponible, les tâches que les réseaux de neurones convolutifs pourraient aborder deviennent de plus en plus intéressantes.

- **AlexNet**: Alex Krizhevsky et son équipe a publiéAlexNet, qui était une version plus profonde et plus large du LeNet et a remporté avec une large marge le concours de reconnaissance visuelle à grande échelle ImageNet (ILSVRC) en 2012. Il s'agissait d'une percée significative par rapport aux approches précédentes et l'applicationrépandue actuelle des CNN pourrait être attribuée à ce travail (voir Figure 2.24).

- **ZFNet**: est l'architecture gagnante du concours ILSVRC 2013, elle a été conçue par Matthew Zeiler et Rob Fergus et il y'avait pas un grand changement par rapport à l'architecture précédente ils ont juste modifié les hyperparamètres de l'architecture.

- **GoogLeNet**: Le gagnant ILSVRC 2014 était un réseau convolutif dela part de Szegedy et son équipe de Google. Sa contribution principale a été le développement d'un « inception module » qui a considérablement réduit le nombre de paramètres dans le réseau (4 million, comparé à AlexNet avec 60 million).

- **VGGNet**: Sa principale contribution était de montrer que laprofondeur du réseau (nombre de couches) est un composant primordial pour obtenir de bonnes performances (voir Figure 2.24)

- **ResNets**: Connue sous le nom « réseaux résiduels » en anglais Residual Networks, Cette architecture a été développée et conçue par Kaiming He et son équipe et elle a été le lauréat de lILSVRC 2015.faisant preuve d'une grande stabilité et de bonnesperformances dans plusieurs problèmes, ResNets sont devenus actuellement le choix par défaut pour lequel les praticiens optent.

- **DenseNet**: Publié en Août 2016 par Gao Huang et son équipe, Il a été démontré que le DenseNet permet d'obtenir des améliorationssignificatives par rapport aux architectures antérieures sur les tâches sur lesquelles il a été testé.

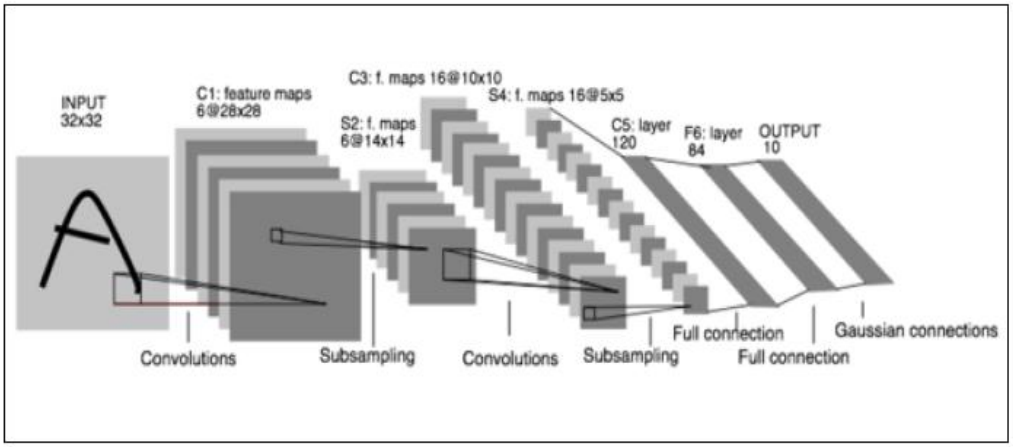


Fig.2.23 LeNet est composée de 2 couches de convolution et 2 couches de max pooling et unecouche FC avec une sortie softmax.

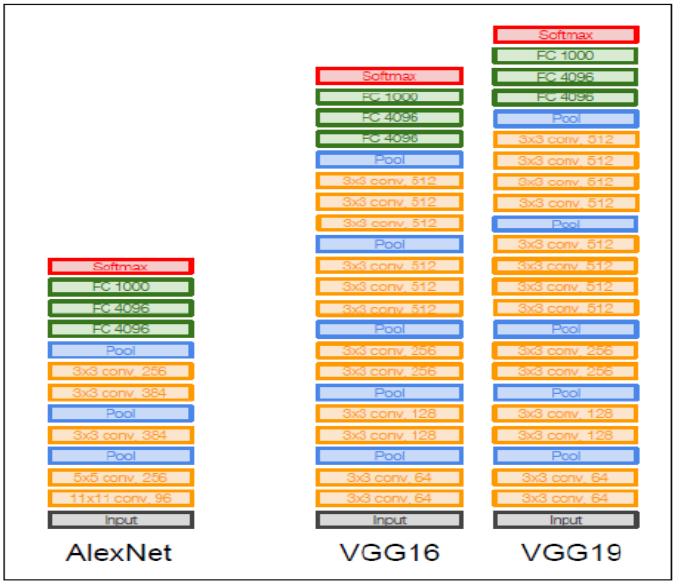


Fig.2.24Max pooling est appliqué avec un filtre 2×2 et Stride=2 et les convolutions sont avec un Stride=1

Conclusion

Dans ce chapitre, nous avons d'abord expliqué ce que sont les problèmes de classification et une frontière de décision. Ensuite, nous avons montré comment modéliser une limite de décision à l'aide de modèles linéaires. Afin de mieux comprendre l'intuition qui se cache derrière un modèle linéaire, ils ont également été étudiés sous l'angle géométrique. Un modèle linéaire doit être formé sur un jeu de données d'apprentissage. À cette fin, il doit exister un moyen d'évaluer la qualité d'un modèle linéaire dans la classification des échantillons d'apprentissage. À cette fin, nous avons expliqué en détail quelques fonctions de perte, notamment perte 0/1, perte au carré, perte logistique. Ensuite, nous avons passé en revue les méthodes permettant d'étendre les modèles binaires aux modèles multi-classes, notamment one-versus-one et one-to-rest. Il est possible de généraliser un modèle linéaire binaire directement dans un modèle multi-classe. Cela nécessite des fonctions de perte pouvant être appliquées à un jeu de données multi-classe.

Le gros problème des modèles linéaires est qu'ils ne fonctionnent pas correctement sur des jeux de données dans lesquels les classes ne sont pas séparables linéairement. Une meilleure solution est d'apprendreune fonction de transformation de caractéristique directement à partir des données d'apprentissage et sur l'entraînement d'un classifieur linéaire.

Pour reconnaitre des imagesle réseau de neurones à convolutions est crucial afin de développer des architectures fiables. Dans ce chapitre, nous avons expliqué comment les opérations de convolution sont dérivées de couches entièrement connectées. À cette fin, le mécanisme de partage du poids des réseaux de neurones de convolution a été discuté. Le bloc de construction suivant dans le réseau de neurones à convolution est la couche de mise en commun, en a aussi donner des exemples pour les architectures CNN

Chapitre 03

Chapitre 03

Démarche et Implémentation

Itroduction

Dans ce projet, j'ai utilisé ce que j'ai appris sur les réseaux de neurones profonds et les réseaux de neurones convolutionnels pour classer les panneaux de signalisation allemands à l'aide du cadre TensorFlow.

1.Anaconda (Python distribution)

Anaconda est une distribution libre et open source des langages de programmation Python et R appliqué au développement d'applications dédiées à la science des données et à l'apprentissage automatique (traitement de données à grande échelle, analyse prédictive, calcul scientifique), qui vise à simplifier la gestion des paquets et de déploiement. Les versions de paquetages sont gérées par le système de gestion de paquetsconda. La distribution Anaconda est utilisée par plus de 6 millions d'utilisateurs et comprend plus de 250 paquets populaires en science des données adaptés pour Windows, Linux etos .[18]

2. Python

Python est un langage de programmation de haut niveau interprété (il n'ya pas d'étape de compilation) et orienté objetavec une sémantique dynamique. Il est trèssollicitépar une large communautéde développeurs et de programmeurs. Python est un langage simple, facile à apprendre et permet une bonne réduction du cout de la maintenance des codes. Les bibliothèques (packages) python encouragent la modularité et la réutilisabilité des codes. Python et ses bibliothèques sont disponibles (en source ou en binaires) sans charges pour la majorité des plateformeset peuvent être redistribués gratuitement.

3.TensorFlow

TensorFlow est un framework de programmation pour le calcul numérique qui a été rendu Open Source par Google en Novembre 2015. Depuis son release, TensorFlow n'a cessé de gagner en popularité, pour devenir très rapidement l'un des frameworks les plus utilisés pour le Deep Learning et donc les réseaux de neurones. Son nom est notamment inspiré du fait que les opérations courantes sur des réseaux de neurones sont principalement faites via des tables de données multi-dimensionnelles, appelées Tenseurs (Tensor). Un Tensorà deux dimensions est l'équivalent d'une matrice. Aujourd'hui, les principaux produits de Google sont basés sur TensorFlow: Gmail, Google Photos, Reconnaissance de voix.[19]

4.Jupyter

Est un projet *open source* sans but lucratif dont la mission est de servir le calcul scientifique et la science des données interactives. Initié en 2014 dans le cadre du IPython Project, la portée de Project Jupyter s'étend à plusieurs autres langages de programmation. L'application web Jupyter Notebook rend possibles la création et le partage de documents contenant aussi bien du code, des équations et des visualisations que du texte.

Jupyter Notebook fonctionne sur un nœud de calcul ou sur un nœud frontal (non recommandé). Dans le cas du nœud frontal, diverses limites sont imposées tant pour l'utilisateur que pour les processus, et les applications sont parfois terminées quand elles utilisent trop de temps CPU ou de mémoire. Dans le cas du nœud de calcul, la tâche est soumise avec la spécification du nombre de CPUs ou de GPUs à utiliser, la quantité de mémoire et le temps d'exécution. Les directives qui suivent concernent la soumission d'une tâche Jupyter Notebook. [20]

5.Matplotlib

est une bibliothèque du langage de programmation Python destinée à tracer et visualiser des données sous formes de graphiques. Elle peut être combinée avec les bibliothèques python de calcul scientifique NumPy et SciPy6. Matplotlib est distribuée librement et gratuitement sous une licence de style BSD4. Sa version stable actuelle (la 2.0.1 en 2017) est compatible avec la version 3 de Python.

Plusieurs points rendent cette bibliothèque intéressante :

- Export possible en de nombreux formats matriciels (PNG, JPEG...) et vectoriels (PDF, SVG...)
- Documentation en ligne en quantité, nombreux exemples disponibles sur internet
- Forte communauté très active
- Interface pylab : reproduit fidèlement la syntaxe MATLAB
- Bibliothèque haut niveau : idéale pour le calcul interactif

6.random

random est un module Python regroupant plusieurs fonctions permettant de travailler avec des valeurs aléatoires.
La distribution des nombres aléatoires est réalisée par le générateur de nombres pseudo-aléatoires « Mersenne Twister », l'un des générateurs les plus testés et utilisés dans le monde informatique.

6.1 Module random

Le module comprend plusieurs fonctions travaillant chacune avec un type défini de variables.
Ces fonctions peuvent être séparées en trois groupes :

Celles qui travaillent avec des nombres entiers
Celles qui travaillent avec des nombres réels
Celles qui travaillent avec des séquences (par exemple des listes).

7.Reconnaissance des panneaux de signalisation avec Tensorflow

Dans ce travail nous avons créé un modèle d'apprentissage approfondi pour la reconnaissance des panneaux de signalisation. L'objectif est de créer un modèle capable de détecter et de classer les panneaux de signalisation dans un flux vidéo provenant d'une voiture en mouvement.

7.1Premier objectif: Classification des panneaux de signalisation

Nous avonscommencer par un objectif simple: la classification. Étant donné l'image d'un panneau de signalisation, notre modèle devrait pouvoir indiquer son type (panneau d'arrêt, limite de vitesse, panneau d' obligation, etc.). Nous travaillerons avec des images correctement recadrées, de sorte que le panneau de signalisation occupe la majeure partie de l'image.

Pour ce projet, nous avons utilisé Python 3.6, Tensorflow1.13 , Numpy, Sci-kit Image et Matplotlib. Tous les outils standards de l'apprentissage automatique qui sont défini au début de ce chapitre.

```
import os
import random
import skimage.data
import skimage.transform
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
# tf.saved_model.simple_save
# Allow image embeding in notebook
%matplotlib inline
```

Première étape, importons les bibliothèques nécessaires et mettons-les en place.

Fig.3.1 les bibliothèques nécessaires dans notre programme

8.Jeu de données de formation

Nous utilisons le jeu de données de signalisation BelgiumTSC_Training et BelgiumTSC_Testing.

Notre jeu de données contient : 62Etiquettes uniques et le Total des images égale 4575

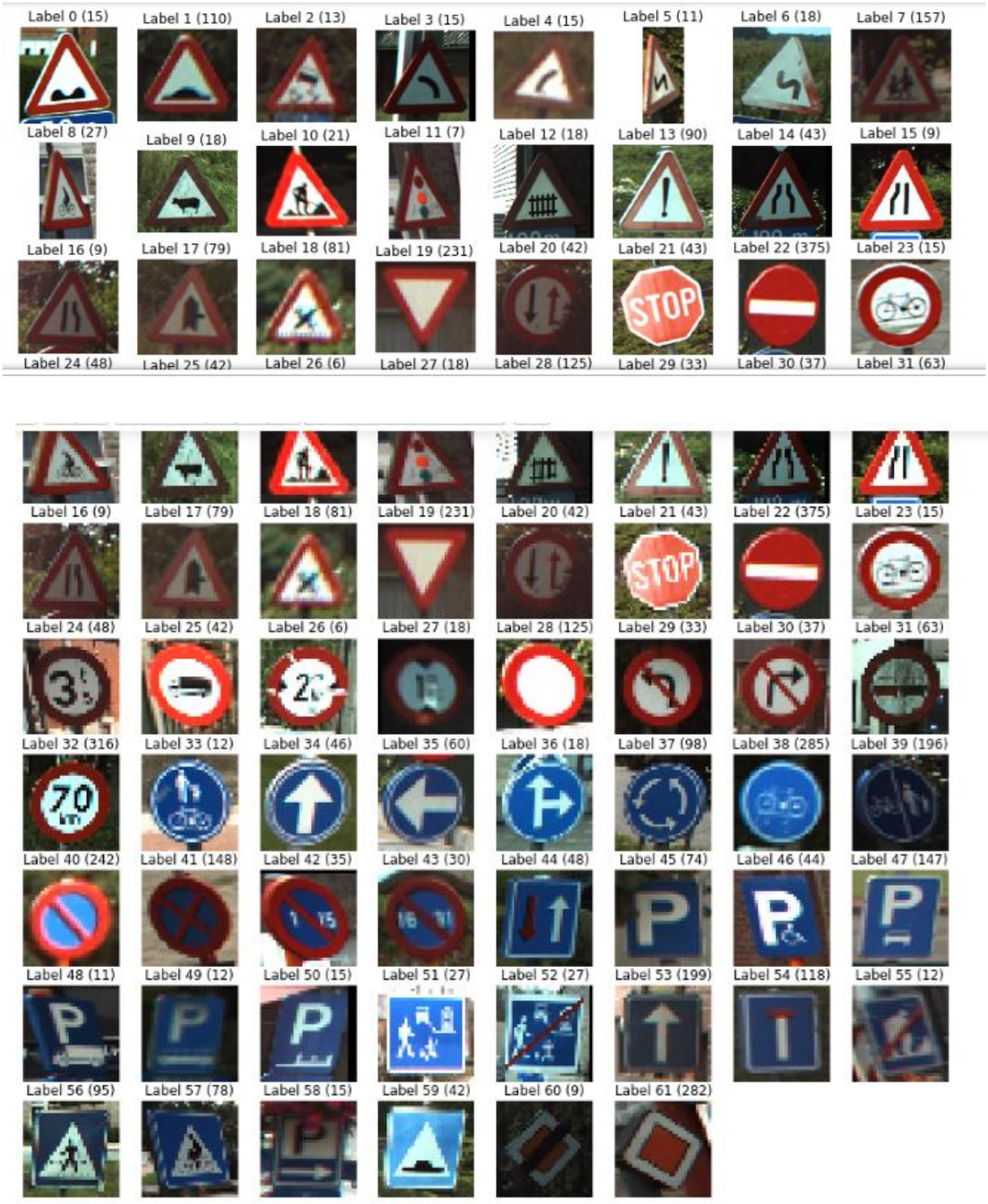


Fig.3.2 la liste des images des panneaux de signalisation utilisée

Les panneaux de signalisation occupent la majeure partie de chaque image, ce qui facilitera notre travail: nous n'avons pas à chercher le panneau dans l'image. Et nous avons une variété d'angles et de conditions d'éclairage, ce qui aidera notre modèle à généraliser.

Cependant, bien que les images soient carrées, elles ne sont pas toutes de la même taille. Notre réseau de neurones simple prend une entrée de taille fixe, nous avons donc un peu de prétraitement à faire.

Il semble que notre ensemble de données considère tous les signaux de limitation de vitesse comme appartenant à la même classe, quel que soit le nombre de chiffres qu'ils contiennent.

Fig 3.3les panneaux de signalisation de limitation de vitesse

La plupart des réseaux de neurones attendent une entrée de taille fixe, et notre réseau ne fait pas exception. Mais comme nous l'avons vu ci-dessus, nos images ne sont pas toutes de la même taille. Une approche courante consiste à rogner et cadrer les images à un rapport sélectionné, mais nous devons ensuite nous assurer de ne pas couper certaines parties des panneaux de signalisation au cours du processus. Cela semble nécessiter un travail manuel! Faisons plutôt une solution plus simple ,nous redimensionnons simplement les images à une taille fixe et ignorons les distorsions causées par les différents rapports de format. Une personne peut facilement reconnaître un panneau de signalisation, même s'il est comprimé ou étiré un peu. Nous espérons donc que notre modèle le pourra également.

Et pendant que nous y sommes, réduisons les images. Plus les données d'entrée sont volumineuses, plus le modèle est volumineux et plus l'apprentissage est lent. Dans les premiers stades de développement, nous souhaitons un apprentissage rapide afin d'éviter les longues attentes entre les itérations tout en modifiant rapidement le code.

Les images 32x32 ne sont pas aussi nettes mais toujours reconnaissables. Notez que l'affichage ci-dessus montre les images plus grandes que leur taille réelle car la bibliothèque matplotlib tente de les adapter à la taille de la grille. Imprimons les tailles de quelques images pour vérifier si nous avons bien compris.

9.Utilisation du modèle

L'objet de session contient les valeurs de toutes les variables de notre modèle (c'est-à-dire les poids)

10.Evaluation

C'est intéressant de visualiser les résultats, mais nous avons besoin d'un moyen plus précis de mesurer la précision de notre modèle. En outre, il est important de le tester sur des images qu'il n'a pas vues. Et c'est là que les données de validation entrent en jeu.

11.perspective

Comme perspective nous avons collecté un ensemble d'images et nous souhaitons développer deux modules : un pour la segmentation de l'image et l'autre pour la détection de la plaque, le résultat de ce dernier module sera l'entrée pour notre travail actuel.

Conclusion générale

La détection d'objet dans les images et plus particulièrement la détection de panneaux est un problème traité par la vision artificielle. Parmi ses domaines d'application, on peut citer la vidéo surveillance, la sécurité routière, les systèmes d'aide à la détection.

Dans notre travail, nous nous sommes intéressés à la détection de panneaux de signalisation routière circulaire. Une étude bibliographique a été réalisée afin décerner les éventuelles méthodes de détection adopté dans ce cas. Un intérêt particulier a été donné à l'algorithme de détection de symétrie radiale. Celui –ci, est dédié à détecter des formes circulaire en se basant sur la mesure de symétrie.

Notre travail a été d'implémenter ce détecteur et de l'intégrer dans un système de détection de panneau de signalisation routière circulaire.

Nous avons testé notre programme sur des images réelles et les résultats obtenus sont encourageants. D'autres formes de panneaux régulières triangulaires, carrés auraient pu être prise en considération, mais par défaut de temps, nous n'avons pas pu le faire. Ceci feral'objet de future perspective.

References

[01] www.DGSN.dz

[02] Piccioli G, De Micheli E, Parodi P, Campani M (1996) Robustmethod for road sign detection and recognition. Image Vis Comput 14(3):209–223

[03] Gao XW, Podladchikova L, Shaposhnikov D, Hong K, Shevtsova N (2006) Recognition of
traffic signs based on their colour and shape features extracted using human vision models. J Visual Commun Image Represent 17(4):675–685

[04] Ruta A, Li Y, Liu X (2010) Robust class similarity measure for traffic sign recognition. IEEE Trans IntellTranspSyst 11(4):846–855

[05] Paclík P, Novovi˘cová J, Pudil P, Somol P (2000) Road sign classification usingLaplace kernel classifier. Pattern Recognit Lett 21(13–14):1165–1173

[06] Maldonado Bascón S, Acevedo Rodríguez J, Lafuente Arroyo S, FernndezCaballeroA, López- Ferreras F (2010) An optimization on pictogram identification for the road-sign recognition task using SVMs. Comput Vis Image Underst 114(3):373–383.

[07] Baró X, Escalera S, Vitrià J, Pujol O, Radeva P (2009) Traffic sign recognition using evolutionaryadaboost detection and forest-ECOC classification. IEEE Trans
Syst 10(1):113–126.

[08] Zaklouta F, Stanciulescu B (2012) Real-time traffic-sign recognition using tree. IEEE Trans IntellTranspSyst 13(4):1507–1514.

[09] HuangGB, MaoKZ, SiewCK,HuangDS(2013)Ahierarchical method for traffic sign classification with support vector machines. In: The 2013 international joint conference neural networks (IJCNN). IEEE, pp 1–6.

[10] Mairal J, Bach F, Ponce J (2014) Sparse modeling for image and vision processing. Trends Comput Graph Vis 8(2–3):85–283

[11] Hsu SH, Huang CL (2001) Road sign detection and recognition using matching pursuit method.vImage Vis Comput 19(3):119–129

[12] Liu H, Liu Y, Sun F (2014) Traffic sign recognition using group sparse coding. InfSci 266:75–89.

[13] CiresanD,Meier U, Masci J, Schmidhuber J (2012b) Multi-column deep neural network for
traffic sign classification. Neural Netw 32:333–338.

[14] Nielsen, M. (2015). Deep learning.

[15] Goodfellow, I. J., Warde-Farley, D., Mirza, M., Courville, A., &Bengio, Y. (2013). Maxout networks.

[16] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1988). Learning representations by backpropagating errors.

[17] Havaei, S. M. (2017). machine learning methods for brain tumor segmentation. (Phd), Université de sherbrooke.

[18] MacOS[https://fr.wikipedia.org/wiki/Anaconda_(Python_distribution)]

[19] Classification des images avec les réseaux de neurones convolutionnels

[20] https://docs.computecanada.ca/wiki/Jupyter/fr

Table de matière

Liste des tableaux

Listes des figures