UNIVERSITE PARIS-SUD

Master Informatique 1ère Année

Année 2015-2016

Rapport de Intro Apprentissage

par

Honglin LI, Yang CHEN, Zhe LYU

Reconnaissance d'images

Enseignant: *Guillaume Wisniewski , Université Paris-Sud*

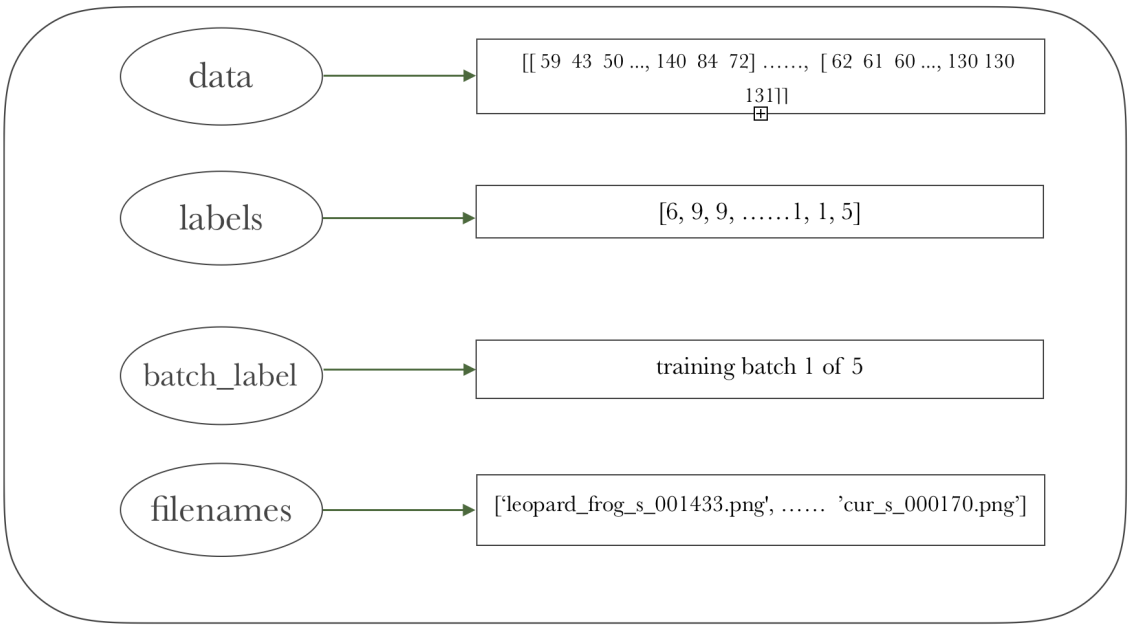
1.Introduction

Réalisation en trinôme d'un projet supervisé dans le cadre de l'UE Introduction à l'apprentissage du semestre 6 de la première année de Master informatique à l'Université de Paris-Sud.

C'est un projet dans la domaine de l'apprentissage statistique. L'objectif de ce projet est la réalisation et la conception d'un système de reconnaissance d'images capable d'identifier automatiquement l'objet représenté sur une image.

1.1 Le base de données Cifar-10

Nous utiliserons la base de données Cifar-10 qui consiste en 60000 32x32 images en couleur dans 10 classes (chien, voiture, avion, …), avec 6000 images par classe. Il y a 50000 images de formation et 10000 images de test. Le jeu de données est divisé en cinq lots de formation et d'un lot de test, chacun avec 10000 images. Le lot de test contient exactement 1 000 images sélectionnées au hasard de chaque classe. Les lots de formation contiennent les images restantes dans un ordre aléatoire, mais certains lots de formation peuvent contenir plus d'images d'une classe à l'autre. Les lots de formation contiennent exactement 5000 images de chaque classe. Le jeu de donnée contient aussi un fichier meta, il décrit les étiquettes entre 0 à 9 corresponde à lequel dans 10 classes, le nombre d'images par lot et le nombre de pixel de chaque image. Nous allons décrire la structure de donnée d'un fichier training ci-dessus :



1.2 Le taux d'erreur de Cifar-10

Une images est en taille 32x32 contient 1024 pixels, il est représenté par le format RGB qui contient trois channels : rouge vert bleu, le type de ces images est bien reconnue par un ordinateur, mais pour un humain c'est très difficile, on ne reconnais que les images visuelle, pas images en pixel. En effet, la tâche est très facile pour un humain, on peut trouver facilement c'est quoi dans l'image, mais pour un ordinateur c'est difficile, parce qu'il ne connais pas les caractéristiques de voiture et chien au début, mais on peut trouver un algorithme pour le résoudre. La base de données Cifar-10 est très populaire dans le domaine d'apprentissage statistiques, il est un des principaux jeux de données utilisées pour évaluer les performances des nouvelles méthodes d'apprentissage développées ces dernières années. Certain chercheur utilise un réseau neuronal convolutif, ils obtiennent des taux d'erreurs très faibles. Ils obtiennent 18% erreur de test sans augmentation des données et moins de 10% avec augmentation des données. Le taux d'erreurs continue à réduire tout le temps.

2.Approche naïve

2.1Méthode kppv

On est en voie de meilleur méthode. Premièrement, nous choisissons une manière naïve pour traiter la tâche, c'est une méthode pour classifier caractéristique d'image, ici la caractéristique est la valeur de chaque pixel, le pixel est consiste à trois chiffres entre 0~255(RGB). On utilise la méthode des k plus proches voisins, cette méthode est une méthode d'apprentissage supervisé, pour cet algorithme l'entrée de donné est K(le nombre de voisin plus proche), un ensemble de données avec label, un ensemble de données de test. L'idée de base d'algorithme est de calculer la distance euclidienne entre les données de test et les données avec label et trouver les k plus proche voisins. Parmi les k plus proche voisins, on choisit le label qui apparaît le plus fréquemment comme le label dans les données de test et voir si c'est un bon résultat.

2.2 La performance

On implémente l'algorithme en Python, on choisit une seule lot training "data_batch_1" et le lot test "test_batch" pour calculer la distance euclidienne entre les deux lots, pour le lot test 'test_batch', il contient exactement 1000 images de chaque classe, donc on peut calculer le taux d'erreur de chaque classe. Chaque test s'exécute pendant 12 minutes, c'est rapide, mais on obtient une faible performance, le taux de réussite est environ 22%, d'abord, on choisit la valeur k(nombre de plus proche voisin) à 4, après on remonte la valeur k jusqu'à 2000, on trouve que quand la valeur k est entre 4 à 100, le taux de réussit remonte un peu avec la croissance de la valeur k, quand la valeur est supérieure à 500, le taux de réussit ne remonte pas. En effet, le taux d'erreur de chaque classe est différente, les classes qui sont bien reconnues sont la classe 'airplane' avec le taux de réussit de moyenne 63.8% et la classe 'ship' avec le taux de réussit de moyenne 33.8%, les classes qui sont systématiquement mal reconnue sont la classe 'dog' avec le taux de réussit de moyenne 6%, la classe 'horse' avec le taux de réussit de moyenne 5.05%, et la classe 'cat' avec le taux d'erreur de moyenne 8.5%.

Pourquoi il y a des différente performances entre les classes? Dans les classes qui sont bien reconnues, la classe 'airplane' et la classe 'ship', la plupart de couleurs des images dans ces classes sont blanc et bleu, ils ont plus beaucoup de pixel 'blanc' et 'bleu' que les autres classes dans dix classes. Mais dans les classes qui sont systématiquement mal reconnue, la classe 'dog', 'horse' et 'cat', il n'y a pas de majorité de la couleur dans ces classes, ses couleur est moyenne distribution par rapport les classes 'airplane' et 'ship', ils ont plus de types de couleur aussi. Donc c'est plus difficile pour classifier les classes.

Utiliser la batch test_batch fait la test sur la batch data_batch_1.(CPU i7-4790)

Paramètre:

K: nombre de voisins kppv

data_batch_1:10000images

test_batch:10000images

Kppv naive	taux de succès total(%)	class 0 airplane	class 1 automobile	class 2 bird	class 3 cat	class 4 deer	class 5 dog	class 6 frog	class 7 horse	class 8 ship	class 09 truck	temps (mins)
k=4	18.82	59	13	31	7	8.1	5.6	10.5	7.8	38.1	8.4	12
k=10	20.27	64	16	36	8	9	6	10	8	37.4	7.4	12
k=20	20.33	68	14	37	8	10	5.4	10.9	7.6	36	7	12
k=100	20.78	73	11	37	7	10	4	13	5	35	13	12
k=200	21.05	73	11.5	38	8	9	4	15	4	33	15	12
k=400	22.02	70	10.7	26.9	7.8	13.6	10.1	8.6	4.4	34	31	12
k =500	21.89	67.8	9.9	26.3	7.5	12.8	10.4	9	4.6	34.8	33.6	12
k=800	22.68	64.3	9.1	25.8	7.7	14.2	12.5	11.2	4.7	34.2	38.1	12
k=1000	22.74	62.7	8.3	33	10	15	3	20	3	31	39	12
k=1500	22.4	54.3	7.5	30.2	10.7	17.2	2.9	22.7	3.2	29.6	42.5	12
k =2000	22.09	46.4	6.7	28.3	11.4	17.6	2.1	26.6	3.3	28.7	44.7	12
moyenne	21.37	63.8	10.7	31.8	8.5	12.4	6	14.3	5.05	33.8	25.4	12

2.3 Avantages et inconvénients

Avantages

C'est facile d'implémenter et ne prend pas de temps pour "apprendre", il n'est pas sensible aux valeurs aberrantes, la précision est grand.

Inconvénients et optimisation

Un des inconvénients est que lorsque l'échantillon est déséquilibre, par exemple la taille de l'échantillon du class est grande et celle des autres classes est très petite, pour le donnée de test, il peut être mis dans la classe qui a le plus grande taille de l'échantillon. Pour résoudre ce problème, on va utiliser le poids(le point plus proche a le poids plus grand). Un autre est que la quantité de calcul dépend de la quantité de l'entrée de données. Si la quantité de l'entrée est grande, donc la quantité de calcul est grande. En général, la quantité de l'entrée de données est grande. Afin de résoudre ce problème, avant de traiter les données on supprime les information non utilisable afin de diminuer la quantité de calcul.

2.4 Méthode k-Means

Pour améliorer la performance, nous implémentons l'algorithme K-Means(K-Moyenne), cette méthode naïve est non supervisé, on le donne les étiquettes pendant l'entrainement. On utilise cet algorithme pour trouver k centroids dans les clusters, dans chaque cluster, je calcule le nombre d'étiquette de chaque classe et sélectionne une étiquette le plus beaucoup dans chaque cluster, j'ajoute cette dans chaque centroid, après on fait le test avec les images de test, on calcule la distance euclidienne entre chaque image et tous les centroids, j'ajoute les centroids et les images dans une liste, on les trie par la distance euclidienne, à la fin on trouve que le centroid de cette image avec la distance plus petite, je retire l'étiquette de cet centroid, je compare cette étiquette avec l'étiquette d'image test, s'ils sont pareil, c'est correct, sinon c'est faute.

2.5 La performance

Nous avons testé cette méthode avec différent paramètres, la performance n'est pas très faible pour une méthode naïve, mais c'est encore loin de notre objectif. on teste un batch entier

contient 10000 images, il peut atteindre un taux de réussit de 30% avec k plus 200 pendant 30 minutes, si le paramètre k plus 500, le taux de réussit ne remonte plus, les taux de réussit de la classe airplane et ship ont une bonne performance, et les classe dog, cat et horse ont une faible performance.

Utiliser la batch test_batch fait la test sur la batch data_batch_1.(CPU i7-4790)
K : nombre de centroids de k-means
data_batch_1:10000images
test_batch:10000images

	Time	success %	TraningSet
k=20	11.72mins	25.77	10000
k=50	18.82mins	27.13	10000
k=60	35.07mins	28.29	10000
k=100	32.76mins	30.18	10000
k=200	37.9mins	30.43	10000
k=500	105mins	30.14	10000
k=1000	200.7mins	30.2	10000

2.5 L'amélioration

Quand nous faisons le test, on trouve qu'il y a quelque centroids sont très loin de cluster, donc les cluster ne pouvons pas le donne les étiquettes, donc on génère une étiquette aléatoire pour le donne.

Pour un taux de réussit de 30%, nous voulons améliorer la méthode pour atteindre une bonne performance, l'idée est pour ajouter l'algorithme de kppv pendant la classification, nous voulons sélectionner la k plus proche centroids d'une image test, et trouver l'étiquette plupart dans ces centroids, nous avons testé cette méthode, mais la performance est très faible, à la fin nous l'abandonnons, nous voulons chercher une méthode plus efficace.

2.6 Avantages et inconvénients

Si le nombre de clusters et la dimension est fixé, la complexité est $O(n^{dk+1} \log n)$,si ce n'est pas fixé, la complexité est Np-hard.

Avantages:

En multipliant les points de départ et les répétitions on peut explorer plusieurs solutions possibles.

inconvénients:

L'inconvénient de cette méthode est qu'elle ne permet pas de détecter des données fruitées ou la présence d'outliers et fournissent des classes convexes, et le nombre de classe doit être fixé au début.

3.Apprentissage de caractéristiques
Classification par kppv

L'approche naïve est d'utiliser une méthode de classification considérant comme caractéristique directement la valeur de chaque pixel. Des valeurs des pixels ne peuvent pas bien représenter les caractéristiques des images. Par exemple, pour la class "chat", des chats en différents couleurs seront classifié comme des classes différents en utilisant l'approche naïve, parce que des valeurs de la plupart des pixels sont différents. Cela n'est pas le cas qu'on espère. En conséquence, on doit utiliser une méthode plus raisonnable. Si on extrait toutes les informations de chaque image, la quantité des données que l'on obtient est très grande. Une telle grande quantité de données ne convient pas au calcul à grande échelle. DE plus, on ne peut pas observer les informations en détail à partir de la tellement grande quantité. Donc, on utilise la méthode la plus couramment utilisée dans le domaine du traitement d'image, l'image est divisé en patch. Dans l'approche proposé, on divise une image en quart patches. Chaque patch est composé de 768 pixels. Ensuite, on met les patches dans un même ensemble. On utilise un algorithme des k-moyennes pour déterminer N représentants.

3.1Explication de l'algorithme K-moyenne

Un premier concept à comprendre est le clustering, qui permet de regrouper les choses similaires dans un même class. Il est différent que la classification. Pour la classification, en général, des données de formation avec label sont nécessaires. L'algorithme de classification peut "apprendre" d'après les données avec label afin d'obtenir la compétence de traiter des

données inconnus. Ce processus est appelé l'apprentissage supervisé. Mais quand on fait le clustering, on n'a pas besoin de savoir auquel class chaque élément appartient. Ce que l'on va faire est de seulement mettre les éléments similaires dans un même class. En conséquence, pour implémenter un algorithme de clustering, la seule chose à faire est de trouver une méthode pour calculer la similarité entre deux éléments. Donc, des données de formation ne sont pas nécessaires pour la clustering. Ce processus est appelé l'apprentissage non supervisé. K-moyenne est un des apprentissage non supervisé pour faire le clustering. L'ensemble de données que l'on a extrait est (X1, X2, …, Xn), et chaque Xi est un vecteur de n dimensions. L'objectif de l'algorithme K-moyenne est que les données brutes sont regroupés en k class:
S = { S1, S2, …, Sk }
avec le nombre de class k (k < n) donné. Pour le modèle mathématique, c'est le calcul de la valeur minimal de l'expression suivante:

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x}_j \in S_i} \|\mathbf{x}_j - \boldsymbol{\mu}_i\|^2$$

Ici, ui représente la moyenne de Si.

Pour programer l'algorithme de K-moyenne, les étapes générale sont:

Choisir k éléments parmi des données au hasard comme les centres des k clusters.

1. Calculer des similarités entre les autre éléments et les k centres, mettre l'élément dans le class avec la plus grande similarité.
2. A partir le résulta obtenu dans l'étape prétendent, recalculer les k centres.
3. Regrouper les données à partir des k nouvelle centres.
4. Répéter l'étape 3-4 jusqu'à le résulta ne change plus.

Supposons que n éléments doivent être groupé en k clusters, ce que l'algorithme k-moyenne va réaliser est de minimiser:

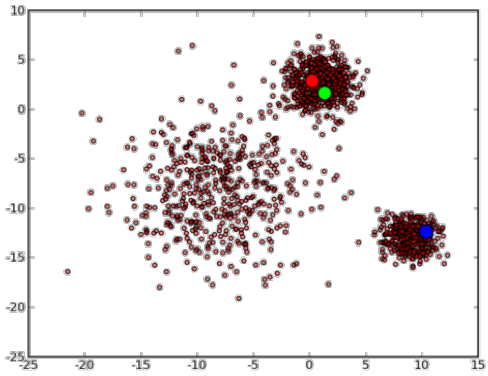
$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|x_n - \mu_k\|^2$$

Dans cette formule, rnk est égale à 1 lors que élément n est mis dans class k, sinon égale à 0. Il est difficile que l'on trouve directement les valeurs propres de rnk et uk pour minimiser J. Cependant, on peut utiliser une approche itérative: d'abord, on fixe μk et choisit la meilleur valeur de rnk. On assure que J est minimal lors que l'élément est mis dans le class le plus proche. Ensuite, on fixe rnk et cherche la meilleur valeur de μk. On calcule la dérivée de J à par rapport à μk et le met égale à zero. On sait que quand J est minimal, μk satisfaire le condition suivante:

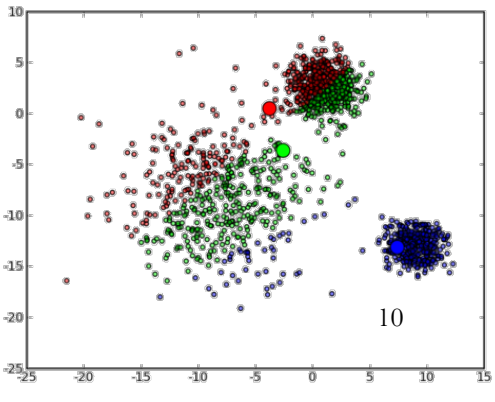
$$\mu_k = \frac{\sum_n r_{nk} x_n}{\sum_n r_{nk}}$$

Étant donné que J devient minimal dans chaque itération, donc J continue de s'amoindrit, ce qui garantit que K-moyenne éventuellement obtenir une valuer minimal. Bien que K-moyenne ne garantit pas toujours trouver la solution global optimal, mais pour ce problème, on peut obtenir un résultat assez bon en utilisant K-moyenne.

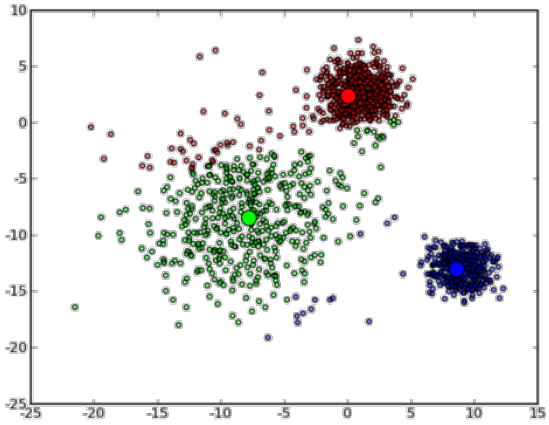
D'abord, trois point sont choisi au hasard, tous les points de données ne sont pas encore regroupés, et par défaut tous sont marqués en rouge :



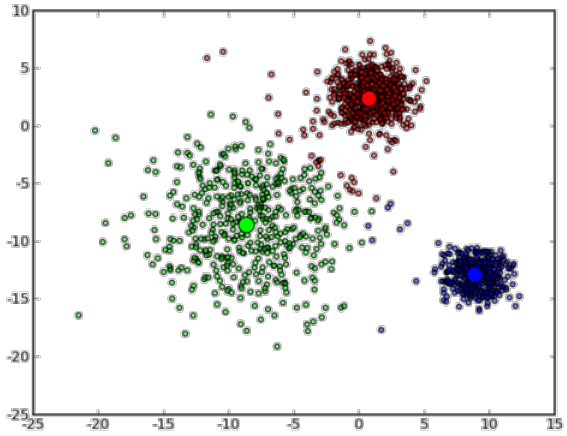
Dans la première itération, on colore les points à partir de la position des centres :



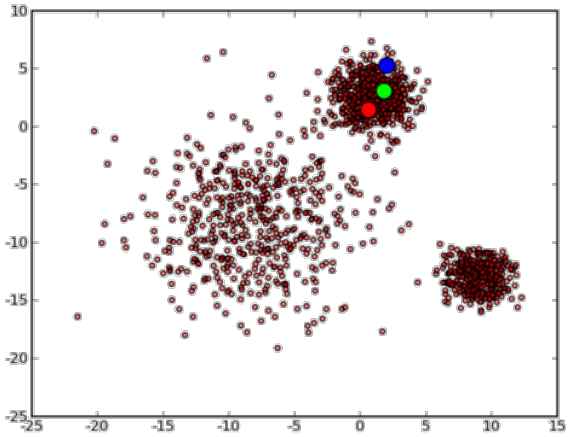
On trouve que le résultat après la première itération n'est pas très bon parce que les points centrals initials sont choisi au hasard. Le résultat après la deuxième itération est comme :



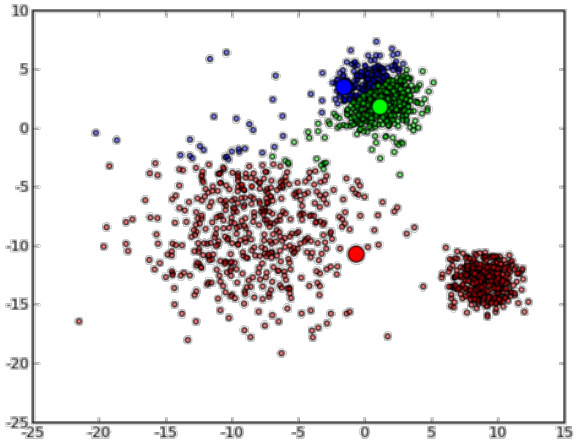
On peut voir que la forme générale est sorti. Puis après deux fois de itérations, le résultat est convergent et finalement :



K-moyenne n'est pas un algorithme parfait, bien qu'il peut converger vers un bon résultat dans nombreux cas. Parfois, in converger vers un mauvais résultat local optimal. Par exemple, l'initialisation des centres est comme ci-dessous:



Finalement, on obtient le résultat :



3.2 Résultat

Paramètre:

K :nombre de voisins de Kppv

N: nombre de centroid de K-means

training_batch:2000images

test_batch:2000images

Table1:
K =4 est fixé

N	taux de réussit(%)
10	18.6
50	22.3
100	20.7
300	22.2
500	20.85

Table 2:
N= 50 est fixé

K	taux de réussit(%)
4	22.3
10	23.7
20	23.85
50	24.95

3.3 Avantages et inconvénients

Avantages:

1. Le principe est simple
2. l'implémentation est facile
3. La performance est bon.

Inconvénients:

1. Impossible de déterminer le nombre de K
2. Sensible aux points loin du group(facilement conduire à l'offset du point central)
3. La complexité de l'algorithme n'est pas facile à contrôler, le nombre d'itérations peut être grande
4. Le résultat est localement optimal(pas globalement optimal)
5. Le résultat est instable (affectée par la séquence d'entrée)

4.Apprentissage de caractéristiques(méthode 2)
Classification par perceptron

Après avoir fini le clustering, on obtient N représentants, décrivant un dictionnaire de patches. Ensuite, on calcule le vecteur de chaque élément à partir du résultat de clustering. On divise l'image en 4 patches et calcule la distance entre chaque patches et les N clusters. Chaque patch est représenté par un vecteur de N caractéristiques binaires; la ie caractéristique est non nulle si et seulement si le centre du ième cluster est le plus proche de patch. Le vecteur de caractéristiques de l'image est obtenu en concaténant les 4 vecteurs

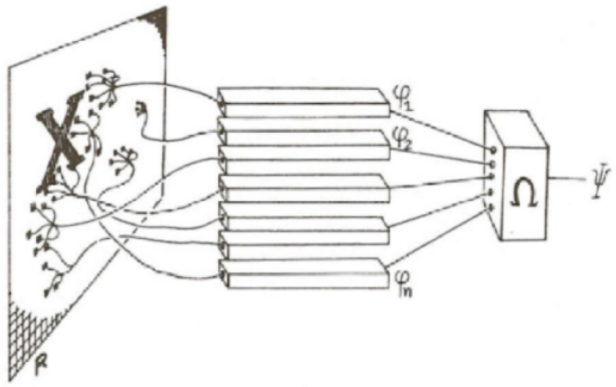
correspondant à chaque patch. Finalement, on utilise l'algorithme K plus proche voisins de calculer le taux de correction.

Après avoir obtenu le dictionnaire de patches: les K représentants, nous avons commencé à trouver une méthode pour classifier une image. Une méthode est d'utiliser le perceptron.

4.1Principe

Perceptron

Le perceptron est le modèle le plus simple des réseaux neurones. Il est un classificateur linéaire. L'algorithme s'inspire du modèle des réseaux neurones.

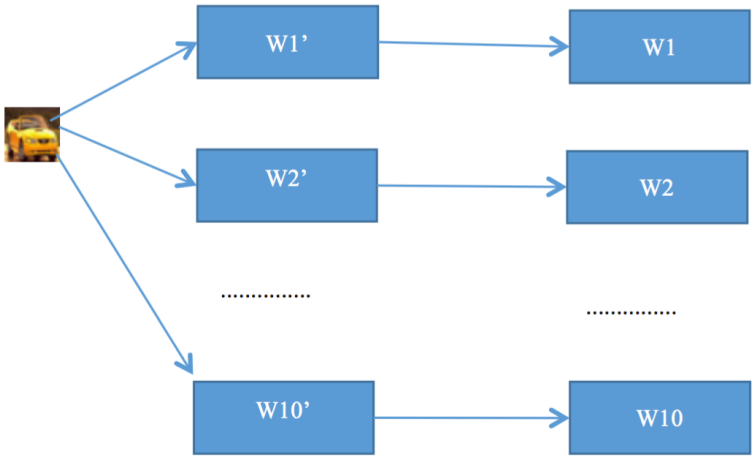


Règle d'apprentissage

Require: a training set x (i) , y (i) i=1 and a learning rate η
1: w ← 0
2: while there are classification errors do
3: for i = 1 to n do
4: y = sign w · x (i)
5: if y = y (i) then
6: w ← w + η · y (i) · x (i)
7: end if
8: end for
9: end while

Adaptation de l'algorithme et l'entraînement

Par rapport à notre sujet, il faut classifier toutes les images en dix classes. Nous avons utilisé le perceptron binaire pour essayer de résoudre ce problème. Nous avons créé un vecteur de caractéristiques pour chaque classe, alors il y a dix perceptron en total. Dans la phase d'entraînement, nous utilisons les observations qui ont déjà été étiquetées pour entraîner tous les dix perceptrons. Finalement, nous pouvons obtenir dix vecteurs de caractéristiques. Le procédure est illustré comme ci-dessous:



Chaque image doit être utilisée dix fois pour entraîner tous les dix vecteurs de caractéristiques afin de réduire la plus grande que possible les jugements erronés.

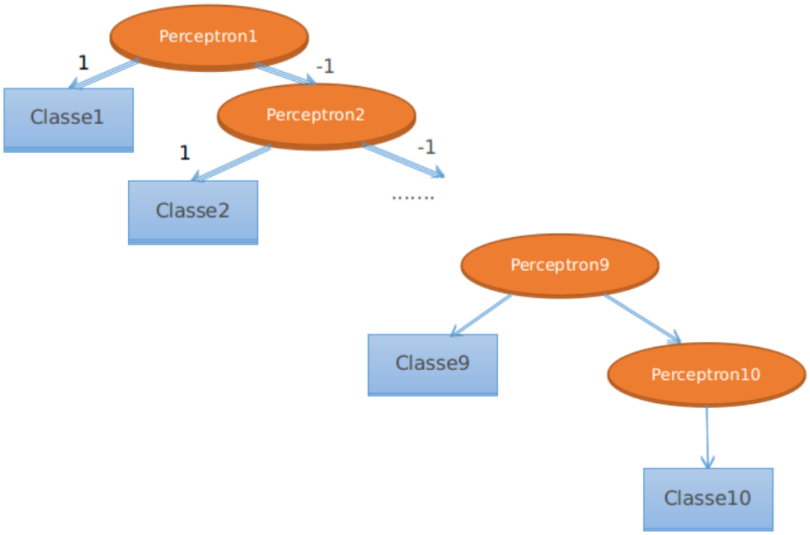
Classification

Méthode 1

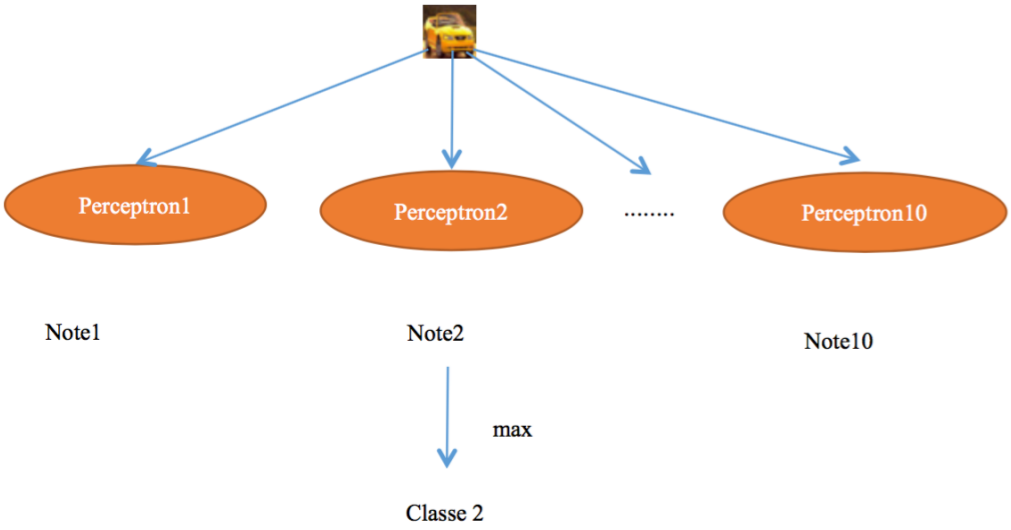
Afin de classifier une image en certain classe, il faut utiliser l'arbre binaire de vecteurs de caractéristiques, la structure est comme ci-dessous:

Nous devons classifier une image en testant à partir du vecteur de caractéristiques de classe 1 jusqu'à de la classe i dont le résultat de classification est positif. Si les résultats des premières 9 classe sont toujours négatifs, nous disons qu'il est de classe 10.

Méthode 2



Pour chaque image, nous calculons le résultat en utilisant le vecteur de caractéristique de chaque classe, ensuite nous l'utilisons comme un note; nous pensons l'image qui s'attribue au cette classe dont le note est le plus grand comme l'étiquette de cette image.



Test et évaluation

Méthode 1

Volume de données de l'apprentissage : 1000 images

Volume de données du test : 1000 images dans test_batch

Le taux de corrects:

classe	K=50	K=100	K=500
1	0.300970873786	0.368932038835	0.252427184466
2	0.202247191011	0.157303370787	0.292134831461
3	0.24	0.04	0.15
4	0.126213592233	0.048543689320	0.135922330097
5	0.155555555556	0.188888888889	0.244444444444
6	0.069767441860	0.139534883721	0.139534883721
7	0.098214285714	0.107142857143	0.133928571429
8	0.117647058824	0.058823529411	0.127450980392
9	0.075471698113	0.235849056604	0.150943396226
10	0.321100917431	0.321100917431	0.201834862385
Correct rate total	0.172	0.168	0.181

Analyse: le règle des données n'est pas très claire, totalement, le taux correct est trop bas. Au début, nous avons pensé c'était à cause du volume des donnée d'entraînement. Parce que le plus les données que les perceptrons apprennent, le plus haut le taux correct sera.
Ensuite, avec cette doute, nous faisons le test ci-dessous:

Volume de données de l'apprentissage : 5000 images
Volume de données du test : 5000 images dans test_batch,
Le taux correct:

classe	K=50	K=100	K=500
1	0.319672131148	0.258196721311	0.381147540984
2	0.134653465347	0.251485148515	0.283168316832
3	0.0546875	0.15234375	0.12890625
4	0.130784708249	0.173038229376	0.130784708249
5	0.278106508876	0.098619329388	0.179487179487
6	0.227459016393	0.112704918033	0.159836065574
7	0.073319755600	0.254582484725	0.177189409369
8	0.070707070707	0.070707070707	0.092929292929
9	0.265873015873	0.12103174603	0.259920634921
10	0.366471734893	0.448343079922	0.391812865497

Analyse: le taux correct est plus haut si nous augmentons le volume d'entraînement. En général, le plus grand la valeur de K, le plus haut le taux correct. Mais le taux correct est toujours trop bas. Donc nous avons amélioré cette méthode de classification.

Méthode 2

Volume de données de l'apprentissage : 1000 images
Volume de données du test : 1000 images dans test_batch
Le taux de corrects:

classe	K = 50	K = 100	K = 500
1	0.300970873786	0.359223300971	0.339805825243
2	0.213483146067	0.314606741573	0.258426966292
3	0.1	0.21	0.3
4	0.155339805825	0.174757281553	0.194174757282
5	0.244444444444	0.122222222222	0.233333333333
6	0.220930232558	0.186046511628	0.0581395348837
7	0.178571428571	0.107142857143	0.169642857143
8	0.147058823529	0.147058823529	0.127450980392
9	0.245283018868	0.301886792453	0.292452830189
10	0.119266055046	0.110091743119	0.110091743119
Taux correct	0.191	0.202	0.209

Volume de données de l'apprentissage : 5000 images
Volume de données du test : 5000 images dans test_batch
Le taux correct:

classe	K = 50	K = 100	K = 500
1	0.477459016393	0.43237704918	0.440573770492
2	0.281188118812	0.28118811881	0.29702970297
3	0.068359375	0.10546875	0.17578125
4	0.221327967807	0.14084507042	0.191146881288
5	0.222879684418	0.24457593688	0.193293885602
6	0.27868852459	0.21926229508	0.188524590164
7	0.083503054989	0.28716904277	0.219959266802
8	0.135353535354	0.11111111111	0.210101010101
9	0.224206349206	0.33531746031	0.323412698413
10	0.109161793372	0.13060428849	0.142300194932
Taux correct	0.2092	0.228	0.2376

Comparaison entre méthode 1 et méthode 2

Dans la méthode 1, l'ordre de classe influence le résultat de classification, suivant cet arbre, les mauvais résultats précédents peuvent influencer les taux corrects des classes arrières. Donc nous avons trouvés la méthode de note. Dans la phase de classification, nous rendons pas 1 et -1 mais plutôt la valeur calculée en utilisant le vecteur de caractéristiques:

Rendre result += observation[i]*vec_params[i]

Donc nous pouvons obtenir dix résultats, nous choisissons le meilleur note qui est le plus grand comme le résultat final de classification.

Optimisation

En faite, le plus grand le volume de données d'entraînement, le plus haut le taux correct sera. Mais à cause de la complexité de KMeans lors de la phase de trouver les clusters, c'est trop coûteux à entraîner tous les batchs donc nous prenons ici que certaines parties des données.

5.Amélioration de l’apprentissage de caractéristique

En effet, la performance de l’apprentissage de caractéristique ne nous satisfait pas, nous voulons implémenter l’algorithme de K-means Triangle pour accélérer la calculation de centroid, et SVM pour classifier les images, mais pour le réaliser en Python, il ne marche toujours pas, à la fin on l’abandonne.

6.Conclusion

Quand nous finissons ce projet, nous avons pris beaucoup d’expérience d’apprentissage caractéristique et apprentissage statistique. Même si nous n’avons pas eu une bonne performance, la méthode de rechercher les algorithme de machine à l’apprentissage est plus important. A présent, ces moments de travail ont beaucoup modifiés nos aptitudes et renforcées notre détermination à trouver des solutions. Nous avons dû améliorer notre méthode et apprendre à travailler dans l'urgence, car bien que nous ayons prévu beaucoup de choses avant de commencer, nous n'avions alors pas assez d'expérience pour voir concrètement la fin du projet.