COURS APPRENTISSAGE

M2, P-20- IFI 2016

Enseignant: **Thanh-Nghi Do**

Rapport de TP 1 & 2

Étudiants: **Ginel Dorleon**
Gervais Fotsing Sikadie S.

Avril 2017

Dans ce rapport nous allons présenter le travail réalisé dans le cadre de ces deux Tps. Ce rapport est divisé en deux parties. La première partie pour le TP1 consiste en une classification avec la méthode des k plus proches voisins communément abrégés par *knn* les arbres de décision. Dans la deuxième partie, on va présenter les exercices sur les réseaux de neurones et les méthodes SVM.

Pour la première partie qui concerne l'algorithme knn, nous allons d'abord résoudre un problème de classification manuellement et ensuite implémenter un autre en C^{++}. Rappelons que la **méthode des *k* plus proches voisins** est une méthode d'apprentissage supervisé. En abrégé k-NN ou KNN, de l'anglais *k-nearest neighbor*. Dans ce cadre, on dispose d'une base de données d'apprentissage constituée de *N* couples «entrée-sortie». Pour estimer la sortie associée à une nouvelle entrée *x*, la méthode des *k* plus proches voisins consiste à prendre en compte (de façon identique) les *k* échantillons d'apprentissage dont l'entrée est la plus proche de la nouvelle entrée *x*, selon une distance à définir. Dans un problème de classification comme dans notre cas, on retiendra la classe la plus représentée parmi les *k* sorties associées aux *k* entrées les plus proches de la nouvelle entrée *x*.

Pour la deuxième partie, avec les réseaux de neurones et les méthodes svm, nous allons procéder à résolution de certains exercices manuellement puis implémenter l'algorithme perceptron en C/C++. Notre programme va pouvoir prendre en entrée les paramètres suivants:

- Le nom du fichier de la base d'apprentissage

- Le nom du fichier de la base de test

- (le pas d'apprentissage)

- Un entier K représentant le nombre d'itérations.

D'autres questions et démonstrations sont aussi présentes dans ce rapport, les différentes parties sont bien explicites.

KNN Résolu Manuellement

Considérant la base d'individus d'apprentissage de la figure 1 suivante:

X1	X2	Classe
0.376000	0.488000	0
0.312000	0.544000	0
0.298000	0.624000	0
0.394000	0.600000	0
0.506000	0.512000	0
0.488000	0.334000	1
0.478000	0.398000	1
0.606000	0.366000	1
0.428000	0.294000	1
0.542000	0.252000	1

Fig.1

Nous allons l'utiliser avec 1NN et 3NN pour classer les individus du tableau ci-dessous;

X1	X2	Classe
0.550000	0.364000	?
0.558000	0.470000	?
0.456000	0.450000	?
0.450000	0.570000	?

Pour ce faire, nous procédons à la classification en utilisant les deux méthodes suivantes et leur fonction de calcul respective.

Nom	Fonction
distance de Manhattan	$\sum_{i=1}^{n} \lvert x_i - y_i \rvert$
distance euclidienne	$\sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$

- **Distance de Manhattan**

La distance de Manhattan ou encore Taxi-Distance définie telle que entre deux points A et B de distance respectives (X_A, Y_A) et (X_B, Y_B), la distance de Manhattan est alors donnée par:

$$d(A,B) = |X_B - X_A| + |Y_B - Y_A|$$

En utilisant cette formule, nous obtenons le tableau suivant et les différentes classes obtenues suivant la distance de Manhattan:

Index	X1	X2	Classe	Manhattan 1	Manhattan 2	Manhattan 3	Manhattan 4
A	0.376	0.488	0	0.298	0.2	0.118	0.156
B	0.312	0.544	0	0.418	0.32	0.238	0.164
C	0.298	0.624	0	0.512	0.414	0.332	0.206
D	0.394	0.6	0	0.392	0.294	0.212	0.086
E	0.506	0.512	0	0.192	0.094	0.112	0.114
F	0.488	0.334	1	0.092	0.206	0.148	0.274
G	0.478	0.398	1	0.106	0.152	0.074	0.2
H	0.606	0.366	1	0.058	0.152	0.234	0.36
I	0.428	0.294	1	0.192	0.306	0.184	0.298
J	0.542	0.252	1	0.12	0.234	0.284	0.41

	X1	X2	Classe	
			K=1	K=3
1	0.55	0.364	1	1
2	0.558	0.47	0	1
3	0.456	0.45	1	0
4	0.45	0.57	0	0

- **Distance Euclidienne**

La **distance Euclidienne** est une distance géométrique dans un espace multidimensionnel.

Appliquant la formule donnée dan le tableau avant, on trouve les résultats suivants:

Données TEST				
			Classe	
Index	X1	X2	K=1	K=3
1	0,550	0,364	1	1
2	0,558	0,470	0	1
3	0,456	0,450	1	0
4	0,450	0,570	0	0

On constate que la classification des individus dépendamment de la méthode de distance utilisée, diffère d'une méthode à une autre.

Implémentation du KNN en C++

Dans cette partie, nous allons faire une implémentation en C^{++} du KNN. Le programme classifie les individus dans la base de test à l'aide de l'algorithme k plus proches voisins. Nous avons implémenté 3 distances:

1. Distance Euclidienne
2. Distance Manhattan
3. Distance Cosinus

Exécuter/Compiler le programme

Le fichier à exécuter s'appelle « tp1_learning_gervais_ginel». Pour lancer le programme, on doit se placer dans le répertoire principal et faire « **cmake .** » ensuite « **make** »

Méthode de lancement du programme - Fonctionnement

Lancement du programme prenant les paramètres suivants:

- le fichier training
- le fichier test
- le nombre de voisins k
- la distance choisie
- le nombre de lignes du fichiers training
- le nombre d'attributs
- le nombre de lignes du fichier test

on tape:

./tp1_learning_gervais_ginel data/iris/iris.trn data/iris/iris.tst 1 1 100 4 50

tels que:

- le premier paramètre est le nom du fichier d'entraînement
- le deuxième paramètre est le nom du fichier de test
- le troisième paramètre est la valeur de k (k=1, k=2, k=3, k=4, k=5)
- le quatrième paramètre est la valeur de la distance utilisée avec 1 pour la distance euclidienne, 2 pour la distance de Manhattan et 3 la distance cosinus
- Le cinquième paramètre est le nombre de lignes du trainset
- Le sixième paramètre est le nombre d'attributs du trainset
- Le septième paramètre est le nombre de lignes du testset

Lancement du programme pour le hold-out

./tp1_learning_gervais_ginel data/fp/fp.trn data/fp/fp.tst 1 1

En sortie, le programme fournit:

- Le résultat sous la forme d'une matrice de confusion dans un fichier txt.
  - Ce fichier contient le nom de la distance
  - La valeur de K
  - Le taux de bon classement

D'une manière générale, notre 'implémentation de l'algorithme k-nn prend en entrée un ensemble de données d'apprentissage , un ensemble de données de tests et une distance(Euclidienne, Manhattan ou Cosinus) pour calculer la matrice de confusion ainsi que le taux de bon classement.

- Il permet la mise en œuvre du protocole *hold-out*, divisant le jeux de données en deux parties training set et *test-set*. Le *training-set* prend 2/3 des données tandis que le *test-set* prend le 1/3 restant. Le choix des données pour le test et l'entraînement est fait à chaque fois de manière aléatoire, pour faire varier au maximum l'ensemble d'apprentissage et de test.
- D'autre part il permet de prendre en entrée un ensemble d'apprentissage et un ensemble test fixé puis de calculer la matrice de confusion et le taux de bon classement pour un type de distance donnée.

Il faut noter que le taux de bon classement dépendra entre autre de la distance choisie et aussi des ensembles d'apprentissage et de test sélectionnés comme dans *hold-out*. Pour cela on a écrit un script qui permet de lancer le programme successivement sur toutes les données et récupérer après un fichier au format txt contenant la matrice de confusion et le taux de bon classement ainsi que la valeur de k. Ceci nous permettra de mieux générer les fichiers à la fin qui pourrons être interprétés. Nul besoin de changer les paramètres dans le code directement. Le script shell en question est présent dans le répertoire source du projet sous le nom de todo.sh. Les instructions dans ce fichier script sont de la forme :

sudo ./tp1_learning_gervais_ginel data/fp/fp.trn data/fp/fp.tst 1 1

sudo ./tp1_learning_gervais_ginel data/fp/fp.trn data/fp/fp.tst 2 1

sudo ./tp1_learning_gervais_ginel data/fp/fp.trn data/fp/fp.tst 3 1

…….
…….
…….

Après l'exécution de script pour toutes les distances, pour toutes les données du tp et pour toutes les valeurs de k=1...5 on obtient des fichiers au format .txt qui contiennent la matrice de confusion et le taux d'erreur pour chaque valeur de k, chaque distance et chaque jeux de données choisi. Par exemple, en lançant la ligne suivante sur le shell:

sudo ./tp1_learning_gervais_ginel data/fp/fp.trn data/fp/fp.tst 1 1

générera le fichier suivant: resultsfp11.txt où:

- fp représente le nom du fichier
- le premier 1 représente la valeur de k
- le second 1 représente le type de distance utilisée

Ce fichier a le format suivant :

Classification results with KNN

Method used: KNN.

k value: 1

global rate of good classification: 89.375%

confusion Matrix

22 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 9 0 0 0 0 0 0 0 0 1 0 0 0 0

0 0 6 0 0 0 0 0 0 0 0 0 0 2 0

0 0 0 8 0 0 0 0 0 0 1 1 0 1 0

0 0 0 0 18 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 16 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 12 0 0 0 0 0 0 1 1

0 0 0 0 0 0 0 10 0 0 0 0 0 2 0

0 0 0 0 0 0 0 0 3 0 1 1 0 1 0

0 0 0 0 0 0 0 0 0 6 2 0 0 0 0

0 0 0 0 0 0 0 0 0 0 8 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 5 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 5 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 9 0

0 0 0 0 0 0 0 0 0 0 0 0 0 2 6

Résultats et Analyse

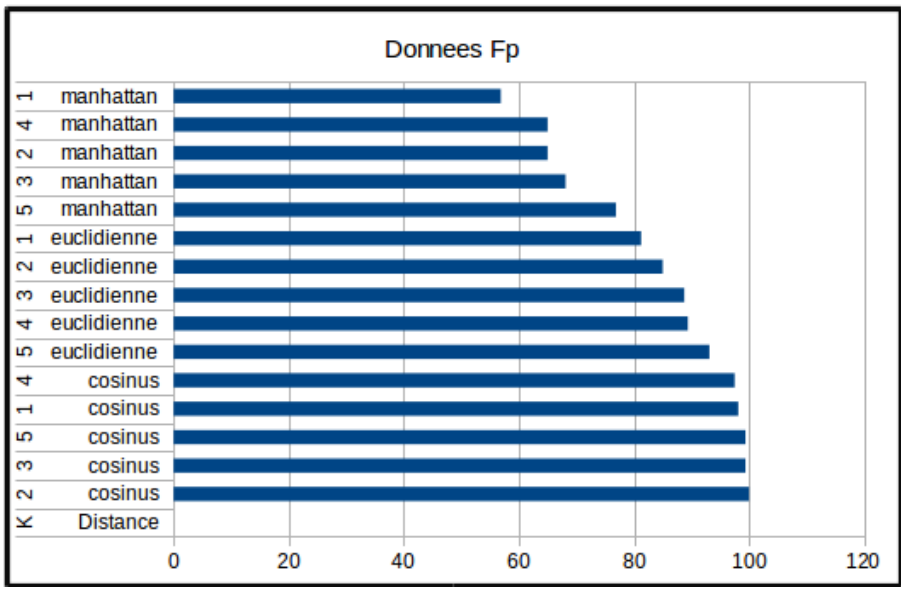
Les bases utilisées dans le cadre de l'expérimentation sont iris, letter, optics et fp. L'évaluation de notre programme se base sur le taux global de bon classement obtenu pour chacune de ces bases et les trois distances implémentées. Ce format de fichier générer directement nous aidera à classer les taux d'apprentissage associé à chaque jeux de données, chaque valeur de k et et chaque distance donnée dans un fichier Excel pour mieux interpréter les résultats afin de choisir la valeur de k et la distance qui donnent le taux d'apprentissage le plus élevé.

Les tableaux et diagrammes suivantes résument les résultats obtenus:

Données Fp:

Meilleure Distance: Cosinus pour k = 2
Meilleur taux d'apprentissage: 100%
Pire: Manhattan avec k=1,et un taux de 56.9%

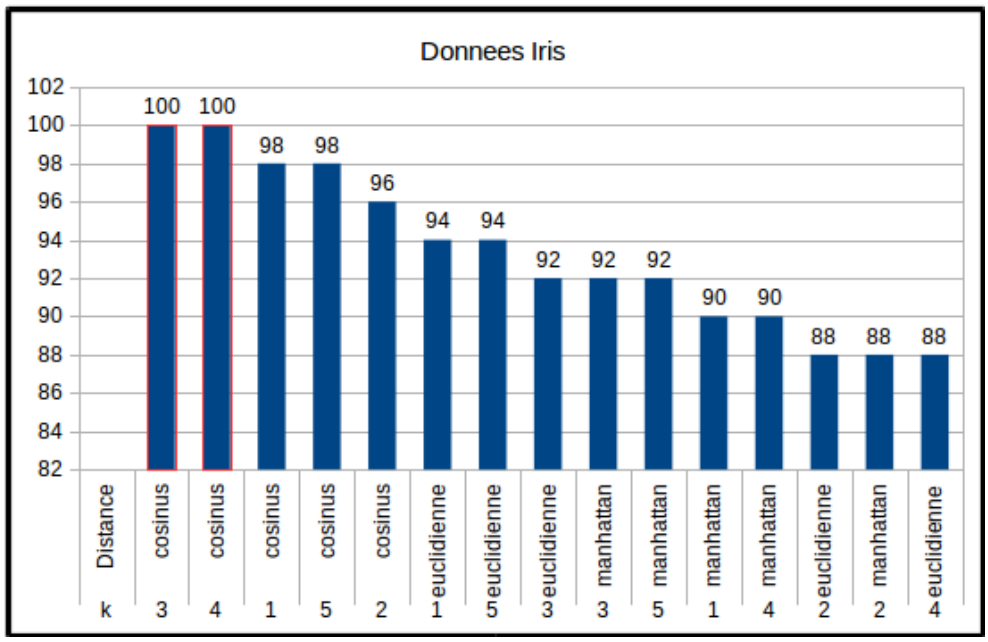
Data Source	K	Distance	Taux d'Apprentissage
Fp	2	cosinus	100,0
Fp	3	cosinus	99,4
Fp	5	cosinus	99,4
Fp	1	cosinus	98,1
Fp	4	cosinus	97,5
Fp	5	euclidienne	93,1
Fp	4	euclidienne	89,4
Fp	3	euclidienne	88,8
Fp	2	euclidienne	85,0
Fp	1	euclidienne	81,3
Fp	5	manhattan	76,9
Fp	3	manhattan	68,1
Fp	2	manhattan	65,0
Fp	4	manhattan	65,0
Fp	1	manhattan	56,9



Données Iris:

Meilleure Distance: Cosinus pour k = 3 et k =4
Meilleur taux d'apprentissage: 100%

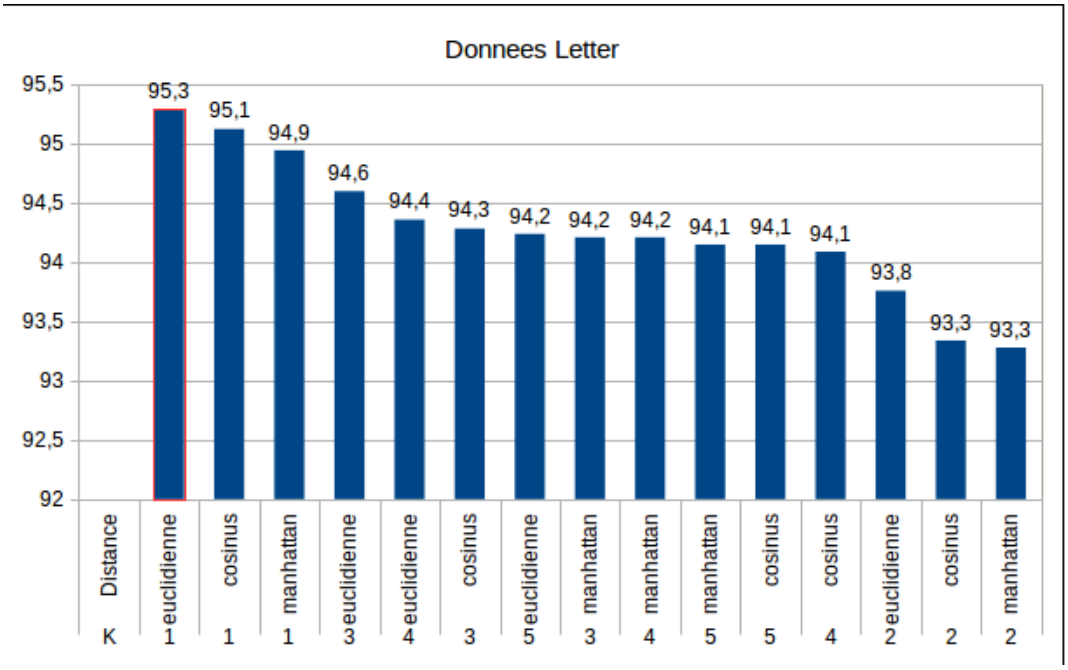
Data Source	k	Distance	Taux d'Apprentissage
Iris	3	cosinus	100
Iris	4	cosinus	100
Iris	1	cosinus	98
Iris	5	cosinus	98
Iris	2	cosinus	96
Iris	1	euclidienne	94
Iris	5	euclidienne	94
Iris	3	euclidienne	92
Iris	3	manhattan	92
Iris	5	manhattan	92
Iris	1	manhattan	90
Iris	4	manhattan	90
Iris	2	euclidienne	88
Iris	2	manhattan	88
Iris	4	euclidienne	88



Données Letter:

Meilleure Distance: Euclidienne pour k = 1
Meilleur taux d'apprentissage: 95,3%

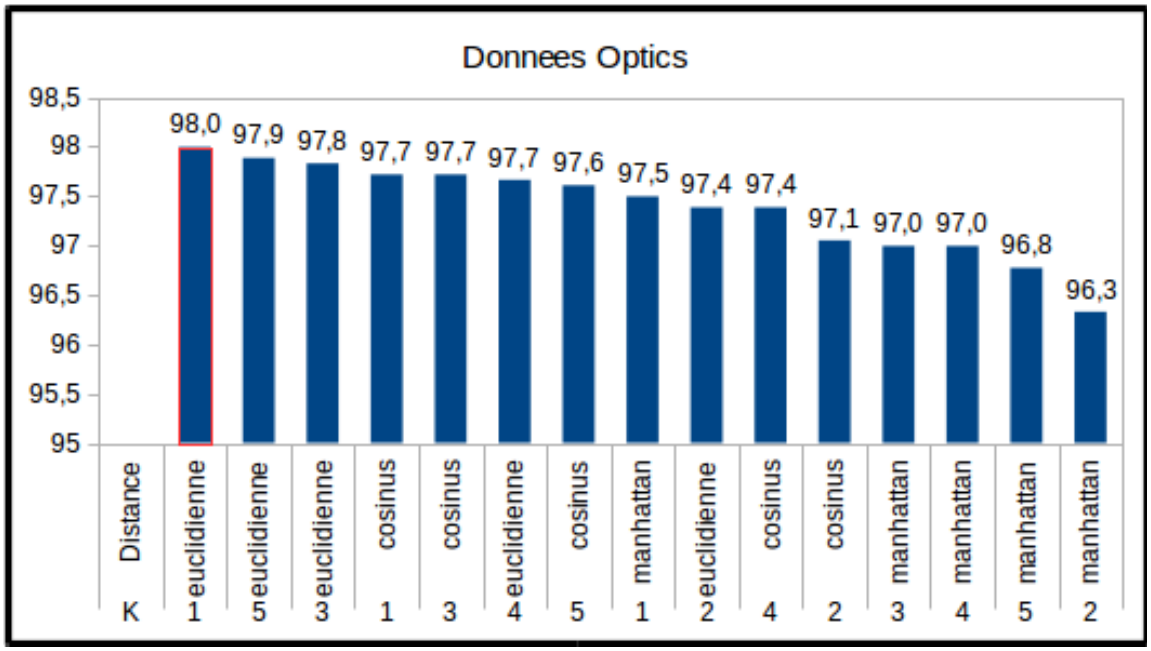
Data Source	K	Distance	Taux d'Apprentissage
Letter	1	euclidienne	95,3
Letter	1	cosinus	95,1
Letter	1	manhattan	94,9
Letter	3	euclidienne	94,6
Letter	4	euclidienne	94,4
Letter	3	cosinus	94,3
Letter	5	euclidienne	94,2
Letter	3	manhattan	94,2
Letter	4	manhattan	94,2
Letter	5	manhattan	94,1
Letter	5	cosinus	94,1
Letter	4	cosinus	94,1
Letter	2	euclidienne	93,8
Letter	2	cosinus	93,3
Letter	2	manhattan	93,3



Données Optics:

Meilleure Distance: Euclidienne pour k = 1
Meilleur taux d'apprentissage: 98 %

Data Source	K	Distance	Taux d'Apprentissage
Optics	1	euclidienne	98,0
Optics	5	euclidienne	97,9
Optics	3	euclidienne	97,8
Optics	1	cosinus	97,7
Optics	3	cosinus	97,7
Optics	4	euclidienne	97,7
Optics	5	cosinus	97,6
Optics	1	manhattan	97,5
Optics	2	euclidienne	97,4
Optics	4	cosinus	97,4
Optics	2	cosinus	97,1
Optics	3	manhattan	97,0
Optics	4	manhattan	97,0
Optics	5	manhattan	96,8
Optics	2	manhattan	96,3



En fait, se basant sur les résultats, on peut conclure que les choix de la meilleure valeur de k et la distance optimale n'obéissent pas à une règle particulière, il faut essayer et choisir la valeur de k et la distance offrant un taux de classement optimal.

Démonstration du Théorème

Soit le théorème suivant:

Démontrons que pour un suffisamment grand ensemble d'apprentissage de taille m, le taux d'erreur de 1NN est inférieur à deux fois le taux d'erreur minimal obtenu par la classification bayésienne. Soit la formule suivante:

$R^* \leq R^{(1)} \leq 2R^* (1 - R^*)$

Où:

$R^{(1)}$: Représente au maximum deux fois le risque de Bayes, ici le taux d'erreur de 1NN
R^* : Le taux d'erreur Bayes

Notez aussi $R^* = 0 \Leftrightarrow R^{(1)} = 0$, et $R^* = 1/2 \Leftrightarrow R^{(1)} = 1/2$.
Preuve:

Appelons $r^*(X)$ la perte (car c'est la perte cela conduit au risque de Bayes). Notons en outre que $E[r^*(X)] = R^*$ par définition de Bayes Risk. Par conséquent, nous avons:

$$R^{(1)} = 2E[r^*(X)(1 - r^*(X))]$$

$$= 2E[r^*(X) - r^{*2}(X)]$$

$$= 2E[r^*(X) - r^{*2}(X) + R^{*2} - R^{*2} + 2R^* r^*(X) - 2R^* r^*(X)]$$

$$= 2E[r^*(X) + R^{*2} - 2R^* r^*(x) - (r^*(X) - R^*)^2] \quad (1)$$

$$= 2E[r^*(X) + R^{*2} - 2R^* r^*(x) - \text{var}(r^*(X))] \quad (2)$$

$$\leq 2E[r^*(X) + R^{*2} - 2R^* r^*(x)] \quad (3)$$

$$= 2(E[r^*(X)] + E[R^{*2}] - 2E[R^* r^*(x)]) \quad (4)$$

$$= 2(R^* + R^{*2} - 2R^{*2}) \quad (5)$$

$\boxed{R^{(1)} = 2R^*(1 - R^*)}$, ce qu'il fallait démontrer.

L'équation (1) découle d'une simple réorganisation des termes, l'équation (2) suit en notant que R*est la valeur attendue de r*(X) et de la Définition de la variance, l'inégalité 3 découle du fait que la variance est non négative, l'équation (4) découle de la linéarité des attentes, l'équation (5) suit à nouveau en notant que R* est la valeur attendue de r*(X).

Arbres de Décision

Soit l'ensemble d'apprentissage correspondant à un ensemble de conditions météorologiques qui permettent (Play pour Positif) ou pas (Don't Play pour Négatif) la pratique du Golf du tableau suivant:

Outlook	Temperature	Humidity	Windy	Class
sunny	85	85	false	Don't Play
sunny	80	90	true	Don't Play
overcast	83	78	false	Play
rain	70	96	false	Play
rain	68	80	false	Play
rain	65	70	true	Don't Play
overcast	64	65	true	Play
sunny	72	95	false	Don't Play
sunny	69	70	false	Play
rain	75	80	false	Play
sunny	75	70	true	Play
overcast	72	90	true	Play
overcast	81	75	false	Play
rain	71	80	true	Don't Play

1 - Construisons l'arbre de décision à partir de cet ensemble d'apprentissage au-dessus qui permet de prédire la pratique du Golf.

- Déterminons la racine de l'arbre

Pour déterminer la racine de l'arbre, nous devons d'abord calculer le gain informationnel des variables *Outlook, Temperature, Humidity, Windy* and *Class*.

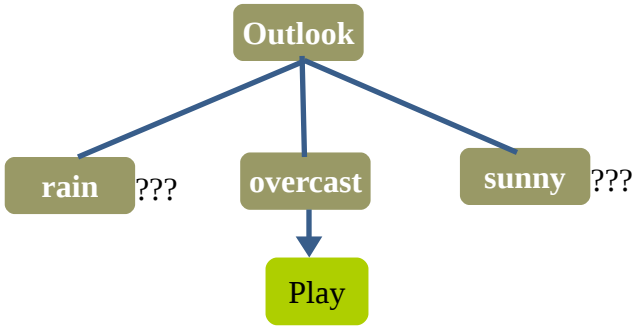
Lors du développement de chaque nœud, choisir l'attribut Ai permettant le gain d'information le plus important avec Gain(Ai) = Io − I(Ai) où:

- ✔ Io correspond à l'entropie de l'ensemble d'exemples correspondant au nœud étudié.
- ✔ I(Ai) correspond à l'entropie 'pondérée' du sous-arbre résultant du développement selon l'attribut Ai

Le tableau suivant présente le gain informationnel calculé pour chacune de nos variables.

Variable	Gain	Racine ?
Outlook	**0,247**	Yes
Temperature	0,1136	No
Humidity	0,236	No
Windy	0,048	No

Considérant le tableau de gain des variables, la variable **Outlook** constituera la racine de l'arbre car elle possède le gain maximal.



La variable Outlook peut prendre trois valeurs possibles:*sunny, overcast, rain*. Afin de déterminer la variable discriminante entre *Windy, Humidity et Temperature*. nous devons donc partitionné chacun des sous-ensembles de Outlook en calculant leur gain respectif .

Le tableau suivant donne les résultat du gain calculé pour chaque valeur.

Outlook: sunny

Variable	Gain
Temperature	0,42
Humidity	**0,971** → Nouvelle racine
Windy	0,0192

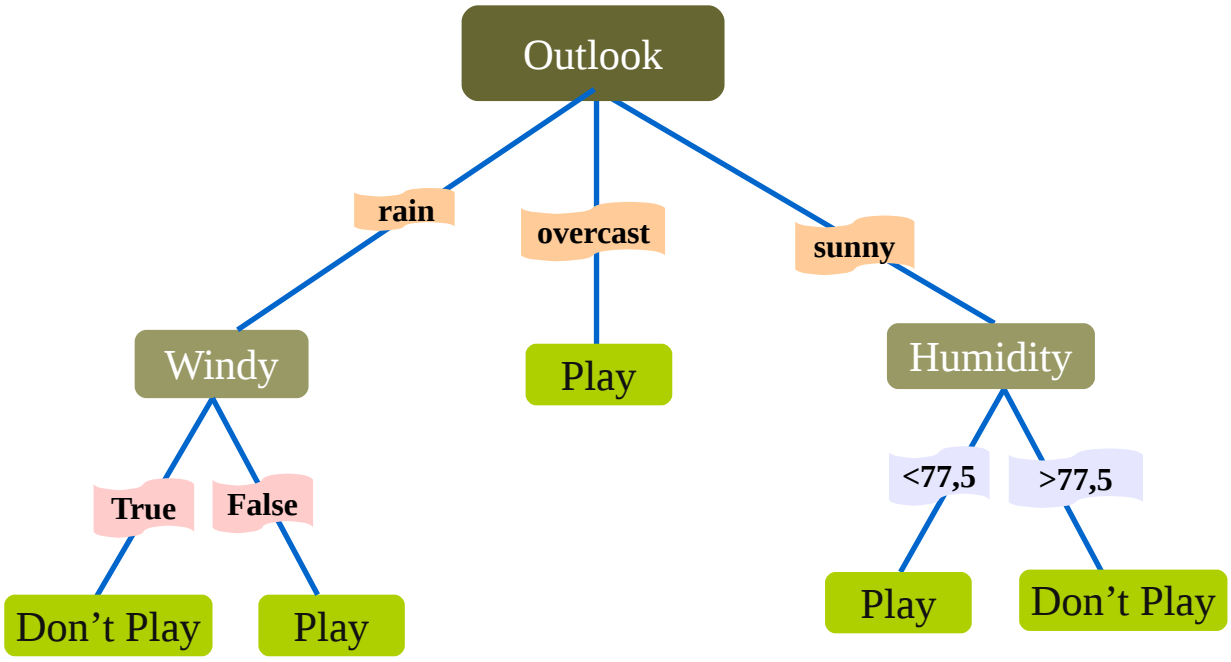
Outlook: overcast: Mélange uniforme, pas besoin de calculer le gain

Outlook: rain

Variable	Gain
Temperature	0,321
Humidity	**0,971** → Nouvelle racine
Windy	0,0...

On constate que la variable avec le gain maximal à considérer est Humidity. Comme le descripteur est continu, il a été nécessaire de définir un seuil dit de discrétisation qui permet de produire le meilleur partitionnement. Dans notre exemple, le seuil qui a été choisi est 77.5 %. Il a permis de produire deux feuilles complètement pures. Ce processus est réitéré sur chaque sommet de l'arbre jusqu'à l'obtention de feuilles pures. Il s'agit bien d'un arbre de partitionnement : un individu ne peut être situé dans deux feuilles différentes de l'arbre.

Ainsi, nous pouvons alors dresser notre arbre de décision.



2 - Déduisons les règles inductives

1. Si (**Outlook** = *rain*) && (**windy** = *true*) Alors ***Don't play***
2. Si (**Outlook** = *rain*) && (**windy** = *false*) Alors ***Play***
3. Si (**Outlook** = *overcast*) Alors ***Play***
4. Si (**Outlook** = *sunny*) && (**Humidity** *<77,5*) Alors ***Play***
5. Si (**Outlook** = *sunny*) && (**Humidity** *>=77,5*) Alors ***Don't play***

3- Classifions les individus suivants:

Outlook	Temperature	Humidity	Windy	Class
overcast	63	70	false	?
rain	73	90	true	?
sunny	70	73	true	?

Ainsi, se basant sur les règles inductives déduites, on a:

Outlook	Temperature	Humidity	Windy	Class	
overcast	63	70	false	Play	***Règle 3’***
rain	73	90	true	Don’t Play	***Règle 1’***
sunny	70	73	true	Play	***Règle 4’***

Expliquer pourquoi l'ensemble d'arbres de décision améliore la prédiction d'un seul ?

Les arbres de décision sont l'une des structures de données majeures de l'apprentissage statistique. Leur fonctionnement repose sur des heuristiques qui, tout en satisfaisant l'intuition, donnent des résultats remarquables en pratique notamment lorsqu'ils sont utilisés en «forêts aléatoires»

Ainsi, pour un ensemble de données choisi on peut obtenir un meilleur taux apprentissage en construisant une forêt d'arbres de décision. Ceci s'explique par le fait que chaque arbre de l'ensemble va apprendre de façon différente le même ensemble. Alors, on choisira la classe majoritairement prédite lors de la classification. Ce qui est beaucoup optimal que si on utilisait un seul arbre. Le taux d'erreur de classification obtenu avec un groupe d'arbre est normalement inférieur à celui obtenu par un seul arbre.

Démontrer que le bagging de k plus proches voisins n'améliore pas le comportement d'un seul.

Le terme *Bagging signifie Bootstrap aggregating. Son but est de faire un compromis entre le biais et la variance pour réduire l'erreur (Biais 2 +variance). Cette méthode est très adaptée pour les algorithmes d'apprentissage souvent instables comme les arbre de décisions, les réseau de neurones, ... cependant, le bagging ne convient pas aux algorithmes d'apprentissage dits stables comme le KNN.*

Ainsi, un modèle obtenu avec le KNN n'est pas totalement altéré par un changement minime dans l'ensemble d'apprentissage. Sur ce, on peut conclure que procéder par un bagging du KNN va plutôt altérer à sa stabilité et ne va pas améliorer son comportement.

TP2

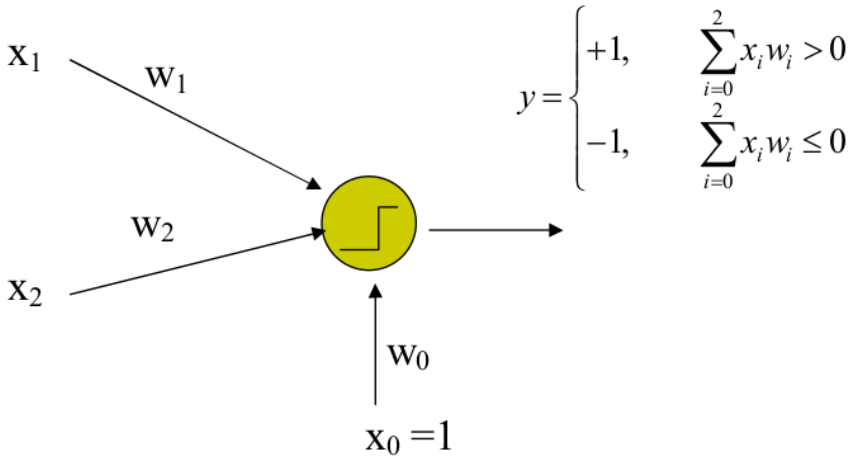
Réseaux de Neurones et SVM

Apprentissage d'un modèle de perceptron simple

Soit la base d'apprentissage suivante:

x_1	x_2	classe
0	0	-1
0	1	+1
1	0	+1
1	1	+1

Soit la fonction d'activation suivante:



avec les poids initiaux tels que:

W0= -0,5; W1=0,4; W2=0,5 , le pas d'apprentissage égal à 0,2

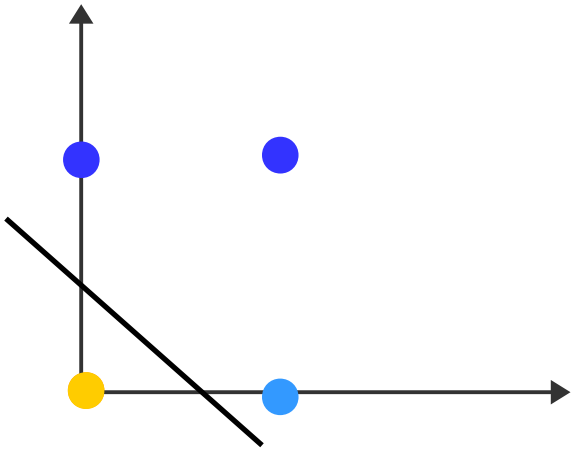
Nous présentons le tableau ci-dessous en faisant une récapitulation des étapes d'apprentissage de cet ensemble de données avec le perceptron simple.

Iteration K	Entrées X			Resultat Attendu	Poids Initial			Resultat Obtenu	Erreur	Poids Final		
K	X0	X1	X2	Y	W0	W1	W2	O	E	W'0	W'1	W'2
1	1	0	0	*-1*	-0,5	0,4	0,5	-1	*0*	0,5	0,4	0,5
	1	0	1	*1*	-0,5	0,4	0,5	-1	*2*	-0,1	0,4	0,9
	1	1	0	*1*	-0,1	0,4	0,9	1	*0*	-0,1	0,4	0,9
	1	1	1	*1*	-0,1	0,4	0,9	1	*0*	-0,1	0,4	0,9
2	1	0	0	*-1*	-0,1	0,4	0,9	-1	*0*	-0,1	0,4	0,9
	1	0	1	*1*	-0,1	0,4	0,9	1	*0*	-0,1	0,4	0,9
	1	1	0	*1*	-0,1	0,4	0,9	1	*0*	-0,1	0,4	0,9
	1	1	1	*1*	-0,1	0,4	0,9	1	*0*	-0,1	0,4	0,9
Fin												

On constate que l'ensemble de données appris par le perceptron en 2 itérations; il est séparable par la droite (D) d'équation *-0,1 + 0,4X1 +0,9X2=0* sur laquelle on dénote les points:P de coordonnées (0, 1/9) et Q(1/4, 0).

b) Visualisation dans l'espace 2D les individus et la droite obtenue

Pour mieux visualiser la classification de l'ensemble de données par le perceptron, nous présentons dans le schéma suivant une visualisation 2D des individus et de la droite obtenue par l'algorithme du perceptron. Cette droite sépare bien les données en 2 groupes bien distinctes.



c) Cas d'apprentissage du SVM à partir de cet ensemble

On remarque que les données de la figure ci-dessus sont linéairement séparables et qu'il y a trois supports de vecteurs: s1(0,0) s2(0,1) et s3(1,0). Soit Φ() la fonction de mapping ou encore la fonction identité et «S'I», le support de vecteur I augmenté de la valeur 1. Nous allons donc utiliser les vecteurs augmentés afin de considérer le biais.

Ainsi, pour S1(0,0) nous aurons S'1(0,0,1).

Alors, évaluer l'apprentissage en utilisant l'algorithme du SVM reviendra donc à déterminer les coefficients α1, α2 et α3 permettant de résoudre les système ci-après:

$$\alpha 1\ \Phi(s1\ ) \cdot \Phi(s1) + \alpha 2\ \Phi(s2) \cdot \Phi(s1\ ) + \alpha 3\ \Phi(s3\ ) \cdot \Phi(s1) = -1$$
$$\alpha 1\ \Phi(s1\ ) \cdot \Phi(s2) + \alpha 2\ \Phi(s2) \cdot \Phi(s2\ ) + \alpha 3\ \Phi(s3\ ) \cdot \Phi(s2) = 1$$
$$\alpha 1\ \Phi(s1\ ) \cdot \Phi(s3) + \alpha 2\ \Phi(s2) \cdot \Phi(s3\ ) + \alpha 3\ \Phi(s3\ ) \cdot \Phi(s3) = 1$$

Ainsi, puisque nous avons Φ() qui est égale à la matrice unitaire I, on peut réécrire notre système d'équation cette nouvelle forme ci-dessous:

$$\alpha 1\ s'1 \cdot s'1 + \alpha 2\ s'2 \cdot s\ '1 + \alpha 3\ s'3 \cdot s'1 = -1$$
$$\alpha 1\ s'1 \cdot s'2 + \alpha 2\ s'2 \cdot s'2 + \alpha 3\ s'3 \cdot s'2 = 1$$
$$\alpha 1\ s'1 \cdot s'3 + \alpha 2\ s'2 \cdot s'3 + \alpha 3\ s'3 \cdot s'3 = 1$$

Maintenant, avec s'1 de coordonnées (0,0,1), s'2 de coordonnées (0,1,1) et s'3 de coordonnées (1,0,1). En faisant la substitution de tous les « s'i » par leur valeurs respectives, nous obtenons:

$\alpha 1 + \alpha 2 + \alpha 3 = -1$

$\alpha 1 + 2\,\alpha 2 + \alpha 3 = 1$

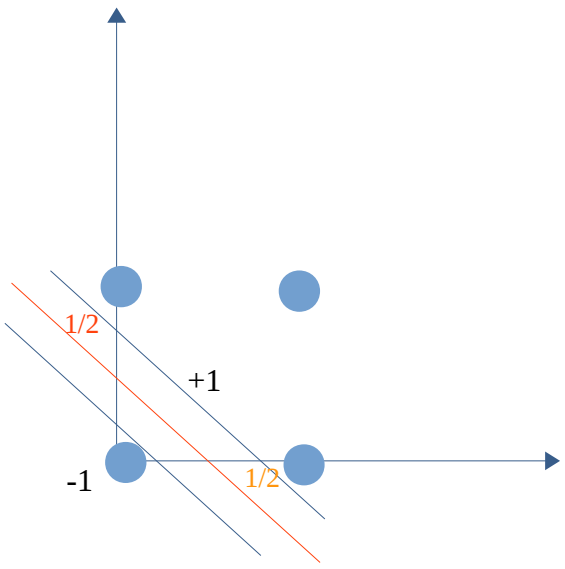
$\alpha 1 + \alpha 2 + 2\alpha 3 = 1$

En résolvant le système, on trouve : $\alpha 1 = -5$; $\alpha 2 = 2$; $\alpha 3 = 2$.

Avec $w' = \Sigma\ \alpha i\ s'i$ on obtient alors w' = -5(0,0,1) + 2(0,1,1) + 2(1,0,1) = (2,2,-1), ce qui permet d'écrire l'équation de la droite optimale séparant nos données.

Y = wx + b

y = 2x1 + 2x2 -1



Implémentation de l'algorithme du perceptron en C++

Nom du programme: perceptron_multi

Exécution:

1. cmake .
2. make
3. ./perceptron_simple *fichier.trn fichier.tst valeur_de_pas k_iteration*

Le perceptron est l'un des réseaux de neurones les plus utilisés pour des problèmes d'approximation, de classification et de prédiction .Dans un premier temps nous allons étudier le perceptron simple. La méthode du perceptron simple permet de classifier des éléments compris dans deux catégories différentes. A condition que ces deux catégories soient linéairement séparables. Par la suite il est également possible de prédire à quelle catégorie appartient un élément.

Dans les lignes suivantes nous allons présenter l'implémentation de l'algorithme de perceptron simple d'abord puis perceptron multicouche après.

Perceptron Simple – Fonctionnement du programme

L'implémentation de l'algorithme du perceptron simple ou encore algorithme du réseau de neurone une couche nous permet de faire le classement des individus d'une base de test. Le nom de notre programme est «*perceptron_simple*». Il reçoit en entrée les 4 paramètres suivants:

- Le premier paramètre est le nom du fichier d'entraînement
- Le deuxième paramètre est le nom du fichier de test
- Le troisième paramètre est la valeur du pas
- Le quatrième paramètre est la valeur K du nombre maximal d'itérations.

Exemple: soit la commande suivante à exécuter,

./perceptron_simple data/leukemia/ALLAML.trn data/leukemia/ALLAML.tst 0 10

Fichier entraînement: *data/leukemia/ALLAML.trn*

Fichier Test: *data/leukemia/ALLAML.tst*

Valeur du pas: 0

K maximal d'itérations: 10

Alors en résumé, précisons que cette implémentation prend en entrée un ensemble de données d'apprentissage , un ensemble de données de tests, un taux d'apprentissage et un nombre maximal k d'itération.

Expérience et résultat

De notre expérience, on note que le taux de bon classement dépendra entre autre du taux d'apprentissage choisi, du nombre maximal d'itération et aussi des ensembles d'apprentissage et de test sélectionné dans le cas du hold-out. Pour cela on a écrit un script qui permet de lancer le programme successivement sur toutes les données et récupérer après un fichier au format.txt contenant la matrice de confusion et le taux de bon classement ainsi que la valeur du taux d'apprentissage t du nombre d'itération maximal optimal. Ceci nous permettra de générer de fichier à la fin qui pourra être interprété. Le script shell en question est présent dans le répertoire source du projet et à la forme suivante:

./perceptron_simple data/leukemia/ALLAML.trn data/leukemia/ALLAML.tst 0.1 10

./perceptron_simple data/leukemia/ALLAML.trn data/leukemia/ALLAML.tst 0.5 10

./perceptron_simple data/ovarian/ovarian.trn data/ovarian/ovarian.tst 0.5 30

./perceptron_simple data/ovarian/ovarian.trn data/ovarian/ovarian.tst 1 30

……...

Soit la commande suivante sur la base d'apprentissage leukemia/ALLAML.trn sur les données de test leukemia/ALLAML.tst.
./perceptron_simple data/leukemia/ALLAML.trn data/leukemia/ALLAML.tst 0.5 15

A l'exécution, on obtient la matrice de confusion suivante:

```
nombre de classes :2
 9 5
 0 20
 le taux de précision est :85.2941%
```

En exécutant le script du fichier todo.sh avec un k itération allant jusqu'à 100. On obtient des fichiers générés d'environ 75MB. Les principales mesures d'évaluation de l'algorithme d'apprentissage ainsi implémenté sont le taux global de bon classement et la matrice de confusion construite. avec différentes valeurs du pas d'apprentissage et du nombre maximal d'itérations. Nous présentons quelques résultats pour toutes les base dans les tableaux ci-dessous pour les k=30 premières itérations, les différents taux globaux de bon classement obtenus. Plus de résultat peuvent être obtenus de façons automatique en exécutant le script présent dans le répertoire principal.

Données: Eukemia

Learning rate	K itté max	Taux d'apprentissage
0.1	10	58.8235
0.1	15	58.8235
0.1	20	58.8235
0.1	25	58.8235
0.1	30	58.8235
0.5	10	85.2941
0.5	15	85.2941
0.5	20	85.2941
0.5	25	85.2941
0.5	30	85.2941
1	10	58.8235
1	15	58.8235
1	20	58.8235
1	25	58.8235
1	30	58.8235

Données: Ovarion

Learning rate	K itté max	taux d'apprentissage
0.1	10	98.8095
0.1	15	98.8095
0.1	20	97.619
0.1	25	98.8095
0.1	30	98.8095
0.5	10	97.619
0.5	15	97.619
0.5	20	97.619
0.5	25	96.4286
0.5	30	96.4286
1	10	100
1	15	100
1	20	98.8095
1	25	100
1	30	97.619

Données Spam

Learning rate	K itté max	Taux d'apprentissage
0.1	10	80.7567
0.1	15	80.3001
0.1	20	0
0.1	25	80.0391
0.1	30	89.2368
0.5	10	89.6282
0.5	15	88.6497
0.5	20	73.8421
0.5	25	88.5845
0.5	30	87.9974
1	10	85.91
1	15	87.4755
1	20	81.6047
1	25	91.4547
1	30	79.1911

On constate, qu'à chaque apprentissage pour les bases eukemia, ovarian et spam le résultat est différent suivant les paramètres d'entrée.

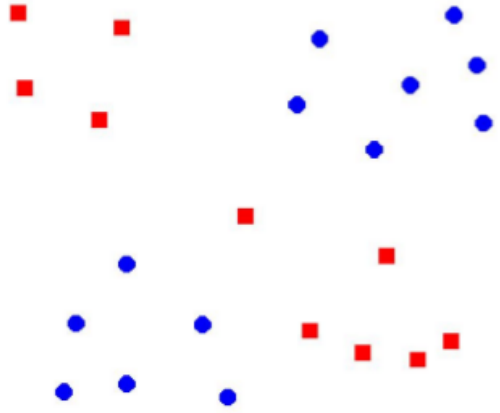
On constate que notre programme d'implémentation de l'algorithme perceptron simple alors fait un bon apprentissage.

Le taux d'apprentissage 1 produit un meilleur résultat dans la 2^{e} et 3^{e} bases pour les k=30 premières itérations.

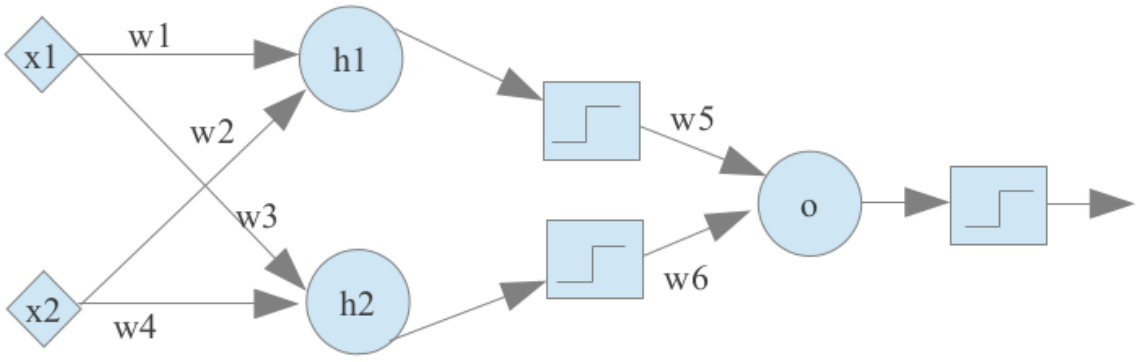
Perceptron Multicouches

3) Soit l'ensemble de données ci-dessous, concevons l'architecture du perceptron multicouches qui permet de classifier l'ensemble de données comme suit:

X_1	X_2	Classe
0.204000	0.834000	0
0.222000	0.730000	0
0.298000	0.822000	0
0.450000	0.842000	0
0.412000	0.732000	0
0.298000	0.640000	0
0.588000	0.298000	0
0.554000	0.398000	0
0.670000	0.466000	0
0.834000	0.426000	0
0.724000	0.368000	0
0.790000	0.262000	0
0.824000	0.338000	0
0.136000	0.260000	1
0.146000	0.374000	1
0.258000	0.422000	1
0.292000	0.282000	1
0.478000	0.568000	1
0.654000	0.776000	1
0.786000	0.758000	1
0.690000	0.628000	1
0.736000	0.786000	1
0.574000	0.742000	1



Alors, nous présentons une architecture en deux dimensions (*x1 et x2*). La première couche est une couche cachée dans laquelle on présente deux neurones intermédiaires (*h1 et h2*). *wi* représente les poids utilisés avec i appartenant à l'intervalle *[1,6]*. Dans notre architecture , nous présentons une sortie unique représentée O.



Implémentation du perceptron multi-couches en C++

Nom du programme: perceptron_multi

Exécution:

4. cmake .
5. make
6. ./perceptron_multi *fichier.trn fichier.tst valeur_de_pas k_iteration*

L'implémentation de l'algorithme multi-couche prend en entrée les paramètres suivants:

- premier paramètre est le nom du fichier d'entraînement

- deuxième paramètre est le nom du fichier de test

- troisième paramètre est la valeur du pas

- quatrième paramètre est la valeur K du nombre maximal d'itérations.

Exemple de commande

./perceptron_multi data/leukemia/ALLAML.trn data/leukemia/ALLAML.tst 0 10

./perceptron_multi data/ovarian/ovarian.trn data/ovarian/ovarian.tst 0 10

Expérience et résultats

L'implémentation de l'algorithme réseau de neurone multi couche prend en entrée un ensemble de données d'apprentissage, un ensemble de données de tests, un taux d'apprentissage et un nombre maximal d'itération.

Il faut noter que le taux de bon classement dépendra entre autre du taux d'apprentissage choisi, du nombre maximal d'itération et et aussi des ensembles d'apprentissage et de test sélectionné dans le cas du hold-out. Pour cela on a écrit un script qui permet de lancer le programme successivement sur toutes les données et récupérer après un fichier au format.txt contenant la matrice de confusion et le taux de bon classement ainsi que la valeur du taux d'apprentissage t du nombre d'itération maximal optimal. Ceci nous permettra de générer des fichiers à la fin qui pourrons être interprétés. Le script shell *todo.sh* en question est présent dans le répertoire source du projet.

La commande suivante

./perceptron_multi data/spam/spam.trn data/spam/spam.tst 0.1 10 permet de faire l'apprentissage sur la base spam/span.trn et le base test spam/spam.tst.

Nous obtenons la matrice de confusion suivante:

```
nombre de classes :2
0 0
0 0
le taux de précision est :-nan%
```

On constate alors que tout est mauvais ici, aucun pourcentage de taux de précision

Dans les tableaux ci-dessous, nous présentons les résultats obtenus pour les k=30 premières itérations.

Données Eukemia

Learning rate	K itté max	Taux d'apprentissage
0.1	10	0
0.1	15	0
0.1	20	0
0.1	25	0
0.1	30	0
0.5	10	0
0.5	15	0
0.5	20	0
0.5	25	0
0.5	30	0
1	10	0
1	15	0
1	20	0
1	25	0
1	30	0

On constate que jusqu' à 30 iterations, le taux d'apprentissage est encore déjà 0 %, il faut alors aller au dessus de 30 pour voir si on peut trouver mieux.

Données Ovarian

Learning rate	K itté max	Taux d'apprentissage
0.1	10	0
0.1	15	0
0.1	20	0
0.1	25	0
0.1	30	0
0.5	10	0
0.5	15	0
0.5	20	73.5294
0.5	25	100
0.5	30	83.3333
1	10	72.4138
1	15	92
1	20	80
1	25	88.4615
1	30	92.3077

Avec un taux d'apprentissage de 0.1 et jusqu'à 30 itérations, le taux d'apprentissage est encore à 0 %

Données Spam

Learning rate	K itté max	Taux d'apprentissage
0.1	10	0
0.1	15	0
0.1	20	0
0.1	25	0
0.1	30	0
0.5	10	0
0.5	15	0
0.5	20	0
0.5	25	0
0.5	30	0
1	10	0
1	15	0
1	20	0
1	25	0
1	30	0

Même constat que pour les données eukemia, on constate que jusqu' à 30 itérations, le taux d'apprentissage est encore déjà 0 %, il faut alors aller au dessus de 30 pour voir si on peut trouver mieux.

Conclusion

Dans ce rapport, nous avons présentés le travail des 2 Tps réalisés dans le cadre du cours d'apprentissage automatique.Ce travail permet de faire le point sur les arbres de décisions, les concepts relatifs aux algorithmes d'apprentissage KNN, perceptron simple et multi-couches. Nous avons aussi procéder à leur implémentation en c++. Les résultats obtenus nos permettent de valider nos approches afin de faire un bon classement des données. Au niveau du KNN, nous avons implémenter trois distances: Manhattan, Euclidienne et Cosinus afin d'obtenir des résultats optimaux pour les différentes bases. Pour la perceptron simple nous avons présenter les différents résultats obtenus et nous mettons déjà disposition un script permettant de lancer automatiquement l'apprentissage et de récupérer les données dans des fichiers au format txt, ce qui permet de gagner du temps. La perceptron multi-couche produit des résultats sans taux d'apprentissage pour les 30 premières itérations donc il faut augmenter le k afin de trouver un meilleur taux.

Références: Cours Apprentissage Automatique,
Dr Thanh-Nghi Do
Université de Can Tho
dtnghi@cit.ctu.edu.vn
M2 - IFI