INSTITUT DE LA FRANCOPHONIE POUR L’INFORMATIQUE (IFI)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



Apprentissage Automatique

**Travaux Pratiques 1 et 2**

**Groupe : 11**

**Participants :**

* **BIAKOTA BOMBIA HERBERT CEPHAS**
* **MILORME PIERRE RUBENS**

**Plan du travail**

**Introduction**

**TP1**

1. **Présentation de la Méthode KNN**

**1.1 Vif du sujet**

* **1.2 Knn manuel**
* **1.3 Implémentation du knn en C++**
* **1.4 Expliquer pourquoi l’ensemble d’arbres de décision améliore la prédiction d’un seul ?**
* **1.5 Démonstration de théorème**

1. **Arbre de décision**

* **2.1 Exercice manuel sur les arbre de décision**
* **2.2 Expliquer pourquoi l’ensemble d’arbre de décision améliore la prédiction d’un seul**
* **2.3 Démontrer que le baggin de k plus proche voisins n’améliore pas le comportement d’un**

**TP2 : Réseaux de neurones et SVM**

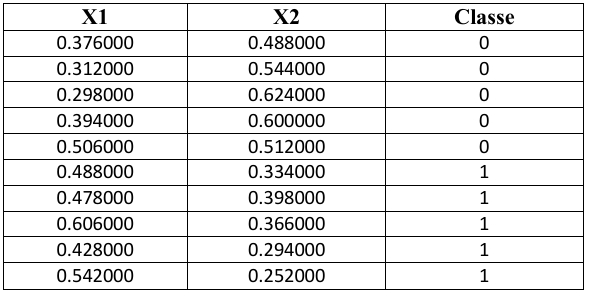
* **1.1 Apprentissage d’un modelé de perceptron simple**
* **1.2 Conception de l'architecture du perceptron multi-couches pour classer un ensemble de données**
* **1.3 Implémentation du perceptron multicouche**

**Introduction**

Apprentissage supervisé (supervised learning en anglais) est une technique où l’on cherche à produire automatiquement des règles à partir d’une base de données d’apprentissage contenant des exemples (en général des cas déjà traités et validés). Il en ressort de cet apprentissage différents types de méthodes parmi lesquelles nous avons : KNN et DECISION TREE qui feront l’objet de tp

1. **Présentation de la méthode KNN (K plus proches voisins)**
   1. **Vif du sujet**

Soit la base d’individus:



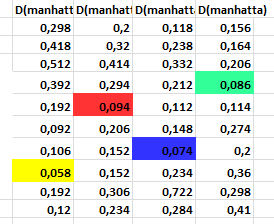
* 1. **KNN manuel**

Le résultat de l’utilisation de *1*NN, *3*NN afin de classifer les individus de la figure2

|  |  |  |  |
| --- | --- | --- | --- |
| X1 | X2 | Classe avec 1NN | Classe avec 3NN |
| 0,55 | 0,364 | 1 | 1 |
| 0,558 | 0,47 | 0 | 1 |
| 0,456 | 0,45 | 1 | 0 |
| 0,45 | 0,57 | 0 | 0 |

Pour y parvenir, nous calculons la distance Euclidienne entre chaque nouvel individu et tous les individus de la base d'apprentissage. Pour l'individu (X1=0,55; X2=0,364), le calcul de distance nous a donné le résultat ci-dessous.

Distance manathan



Distance euclidienn

|  |  |  |  |
| --- | --- | --- | --- |
| **D(euclidienne)** | **D(euclidienne)** | **D(euclidienne)** | **D(euclidienne)** |
| **0,213663287** | **0,182887944** | **0,088566359** | **0,11045361** |
| **0,298402413** | **0,256889081** | **0,171965113** | **0,140427917** |
| **0,362082863** | **0,302185374** | **0,235031913** | **0,16130716** |
| **0,282899275** | **0,209274939** | **0,162308349** | **0,063529521** |
| **0,154402073** | **0,0668431** | **0,079649231** | **0,080622577** |
| **0,068876701** | **0,15295751** | **0,120332872** | **0,239039746** |
| **0,079624117** | **0,107628992** | **0,056462377** | **0,174264167** |
| **0,056035703** | **0,114542569** | **0,171918585** | **0,256811215** |
| **0,140655608** | **0,21880585** | **0,158492902** | **0,276875423** |
| **0,112285351** | **0,218586367** | **0,215870331** | **0,331040783** |

Pour y parvenir, nous calculons la distance Euclidienne entre chaque nouvel individu et tous les individus de la base d'apprentissage. Pour l'individu (X1=0,55; X2=0,364), le calcul de distance nous a donné le résultat ci-dessous.

* 1. **Implémentation du KNN en C++**

**Fonctionnement du programme**

Notre programme permet de classifier les individus d'une base de test a l'aide de l'algorithme des k plus proches voisins. Pour ce faire, il reçoit en entrée une base d'apprentissage, une base de test et un entier positif k indiquant le nombre de voisins a considérer. Après exécution de l'algorithme KNN, il restitue en sortie une matrice de confusion et un taux global de bon classement qui renseigne sur la performance du classement effectue.

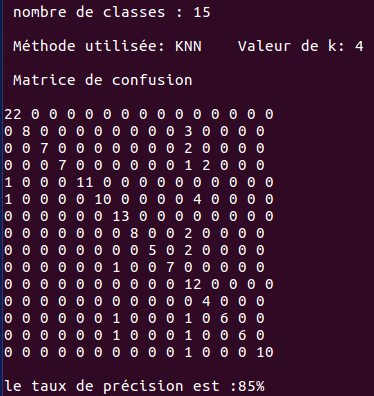
La commande a exécuter pour lancer notre programme est de la forme:

**./apprentissage\_knn « fichier\_base\_apprentissage » « fichier\_base\_test » k**.

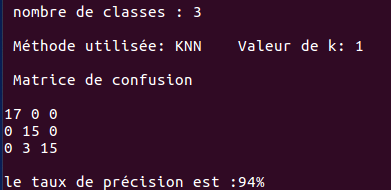
Lorsque cette commande est exécutée, le programme lit l'ensemble des données d'apprentissage et des données de test puis les stocke dans des tableaux. Par la suite, la similarité de chaque individus de l'ensemble de test est évaluée par le calcul de la distance (Euclidienne ou Manhattan) entre ce dernier et tous les individus de l'ensemble d'entrainement. Les distances obtenues sont triées par ordre croissant et la classe majoritaire des k premiers individus est assignée au nouvel arrivant.

Par exemple la commande: **./apprentissage\_knn data/fp/fp.trn data/fp/fp.tst 2**exécute l'algorithme d'apprentissage du 2NN sur la base d'apprentissage fp/fp.trn, puis fait le test sur l'ensemble de données fp/fp.tst. En sortie, nous obtenons la matrice de confusion ainsi que le taux de bon classement tel que présente ci-dessous (cf. fig 1).

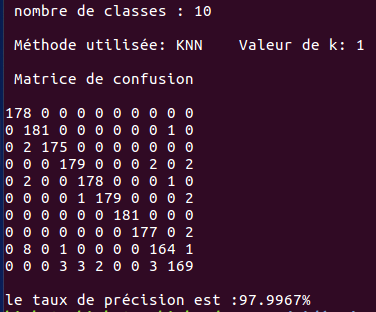
**Ensembles de données expérimentés :** Iris, Optics, Letter,



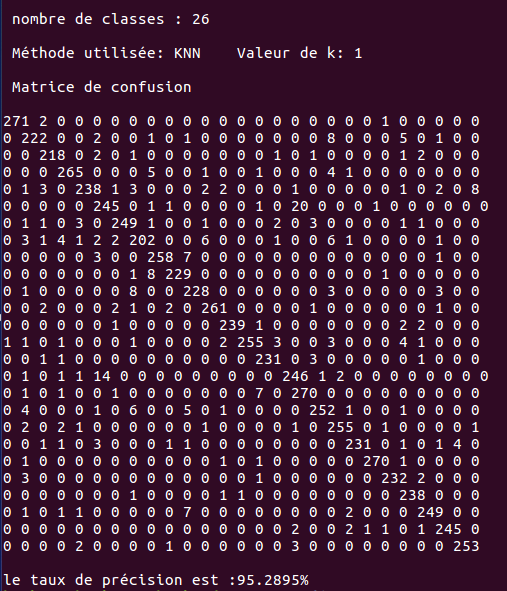
**Figure 1** KNN (k=4) sur l'ensemble de données fp



**Figure 2** KNN sur l'ensemble de donnees optics



**Figure 3** KNN sur l'ensemble de données optics



**Figure 4** KNN sur l'ensemble de données letter

**Expérimentations :**

Les bases utilisées dans le cadre de l'expérimentation sont *iris, letter, optics et fp*. L'evaluation de notre programme se base sur le taux global de bon classement obtenu pour chacune de ces bases.

Un premier jeu de test nous permet de constater que le taux global de bon classement est meilleur lorsque la distance Euclidienne est utilisée contrairement a la distance de Manhattan. Aussi la valeur de k influence la qualité du modèle obtenu. Notamment avec k=5, nous obtenons généralement les meilleurs taux pour les bases classées et cela quel que soit la distance choisie. Ci-dessous, des tableaux récapitulant les taux globaux de bon classement obtenus en fonction de la base, du type de distance et de la valeur de k. Dans ces tableaux, nous présentons uniquement les résultats pour les bases iris, letter et optics. Nous procèderons différemment pour la base fp.

|  |  |  |  |
| --- | --- | --- | --- |
| Dataset | K | Distance | Taux de bon placement global |
| **iris** | **1** | **Euclidienne** | **94%** |
| iris | 2 | Euclidienne | 88% |
| iris | 3 | Euclidienne | 92% |
| iris | 4 | Euclidienne | 88% |
| iris | 5 | Euclidienne | 94% |

**Tableau 1** Expérimentations de l'algorithme du KNN sur la base ***iris***

|  |  |  |  |
| --- | --- | --- | --- |
| Dataset | K | Distance | Taux de bon placement global |
| **optics** | **1** | **Euclidienne** | **97%** |
| optics | 2 | Euclidienne | 97% |
| optics | 3 | Euclidienne | 97% |
| optics | 4 | Euclidienne | 97% |
| optics | 5 | Euclidienne | 97% |

**Tableau 2** Expérimentations de l'algorithme du KNN sur la base

|  |  |  |  |
| --- | --- | --- | --- |
| Dataset | K | Distance | Taux de bon placement global |
| **Letter** | **1** | **Euclidienne** | **95%** |
| lettre | 2 | Euclidienne | 93% |
| letter | 3 | Euclidienne | 94% |
| letter | 4 | Euclidienne | 94% |
| letter | 5 | Euclidienne | 94% |

**Tableau 3** Expérimentations de l'algorithme du KNN sur la base ***iris***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Dataset | K | Distance | Taux de bon placement global | Distance | Taux de bon placement global |
| Letter | 5 | Manathan | 94% | Euclidienne | 94% |
| iris | 5 | Manathan | 92% | Euclidienne | 94% |
| optics | 5 | Manathan | 96% | Euclidienne | 97% |

**Tableau 4** Synthèse des expérimentations de l'algorithme du KNN sur les bases iris, letter et optics

**1.4 Expliquer pourquoi l’ensemble d’arbres de décision améliore la prédiction d’un seul ?**

En construisant une foret d'arbres de décision a partir d'un ensemble de données on assure un meilleur apprentissage car chacun des arbres apprend différemment le même ensemble et lors de la classification on choisit la classe majoritairement prédite par les arbres ce qui assure d'une meilleure classification que si on ne considérait qu'un seul arbre. Le principe est que l'erreur de classification réalisée par un groupe d'arbre est inferieure a celle réalisée par un seul arbre. En effet en termes statistiques si on a les arbres décarrèles cela permet de réduire la variance des prévisions.

**Démonstration du théorème:**

Pour un suffisamment grand ensemble d'apprentissage de taille m, le taux d’erreur de

1NN est inferieure a deux fois le taux d'erreur minimal obtenu par la classification bayésienne.

Soir R le taux d'erreur de 1NN et R\* celui de bayes.

On sait que :

R = E [2η(X) (1 − η(X))]

= 2E [ min { η(X), 1 − η(X)} max { η(X), 1 − η(X) }]

en posant min { η(X), 1 − η(X)} = r\*(X) et E(r\*(X)) =R\* on obtient les developpements ci apres :

R= 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 ]

= 2E [r∗ (X) + R∗2 − 2R∗ r∗ (x) − var (r∗ (X)) ]

≤ 2E [r∗ (X) + R∗2 − 2R∗ r∗ (x) ]

= 2 (E [r∗ (X)] + E [R∗2] − 2E [R∗ r∗ (x)] )

= 2 (R∗ + R∗2 − 2R∗2 )

= 2 R∗( 1 − R∗)

On obtient donc R = 2R∗ (1 − R∗) d'ou le taux d’erreur de 1NN est inferieure a deux fois le

taux d'erreur minimal obtenu par la classification bayésienne.

**2 Arbre de décision**

**2.1 Exercice manuel sur les arbres de décision**

Notre ensemble d'apprentissage est constitué de données météorologiques construites dans le but de savoir si le temps qu'il fait est adapté ou non à la pratique du Golf. Il contient 5 variables et 14 individus tel qu'il suit:



**Tableau 1** Données météorologiques qui seront apprises avec un arbre de décision

Construire l’arbre de décision à partir de cet ensemble d’apprentissage au-dessus qui permet de prédire la pratique du Golf. La fonction de split se base sur l’entropie de Shannon

* + **Formule: H(x) = H2(X)=-**

Partitionnement se décompose un noeud *D* (*m* individus) en *k* sous-noeuds *D1, …, Dk* (*m1, …, mk* individus), l’information est définie par: **info(part./D) = (*m*1 /*m*)\*info(*D*1)+...+ (*mk* /*m*)\*info(*Dk*)**

Résultat de l’application

**1-Choix de la racine de l'arbre**

Info(temperature)=0,692

Info(Humidity)=0,704

***info(outlook)=0,693***

info(Windy)=-0,8921589283

info minimale =>Outlook

**2- Partitionner « outlook=sunny »**

info(temp)urature=0,55

***info(humidity)=0***

info(windy)=0,950

info minimale =>humidity

**3- Partitionner « outlook=overcast »**

**4- Partitionner « outlook=rain »**

info(temperature)=0,8

info(humidity)=0,649

**info(windy)=0**

info minimale =>wind

Les règles de décisions extraites :

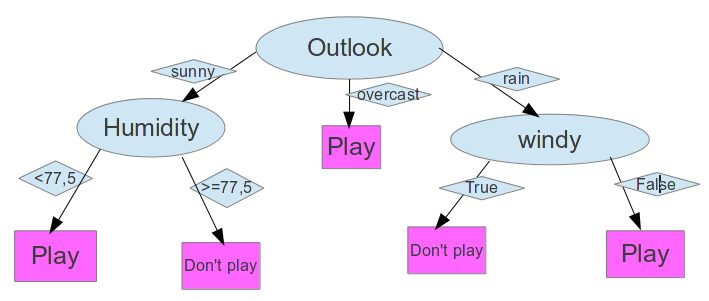
**\*Si(outlook=sunny) Et (Humidity<77,5) Alors Play**

**\*Si(outlook=sunny) Et (Humidity>=77,5) Alors Don't play**

**\*Si(outlook=overcast) Alors Play**

**\*Si(outlook=rain) Et (windy=true)Alors Don't play**

**\*Si(outlook=rain) Et (windy=false)Alors play**



**Classifier les individus suivants**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Outlook | Temperature | Humidity | Windy | Class |
| overcast | 63 | 70 | false | Play |
| rain | 75 | 90 | true | Don't Play |
| sunny | 70 | 73 | true | Play |

**Expliquer pourquoi l’ensemble d’arbres de décision améliore la prédiction d’un seul**

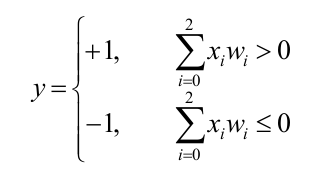
L’amélioration de la prédiction par un ensemble de décision est dû à le fait que l’apprentissage des donné est fait sur diffèrent arbre afin de réduire la complexité de la prédiction et de faire le choix de la classe de manière optimal en se basant sur la partie majoritaire qui est prédite par chaque arbre

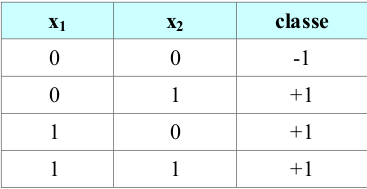
**1.5 Démontrer que le bagging de k plus proches voisins n’améliore pas le comportement d’un seul Bagging** signifie Bootstrap aggregating. Son but est de faire un compromis entre le biais et lavariance pour réduire l'erreur (Biais2+variance). Il est adapté pour les algorithmes d'apprentissage instables: arbre de décision, réseau de neurones, … Mais, ne convient pas aux algorithmes d'apprentissage stables: KNN, …

En effet la qualité du modèle obtenu avec le KNN n'est pas trop altérée par un petit changement dans l'ensemble d'apprentissage. De ces analyses, nous concluons qu’en procédant par un bagging du KNN va plutôt détériorer la stabilité du KNN

**TP2: Réseaux de neurones et SVM**

* 1. **Apprentissage d'un modèle de perceptron simple**
* Fonction d'activation



* Poids initiaux: W0= -0,5; w1=0,4; w2=0,5;
* Pas d'apprentissage=0,2

**Tableau 2.1:** Ensemble à apprendre par le perception simple

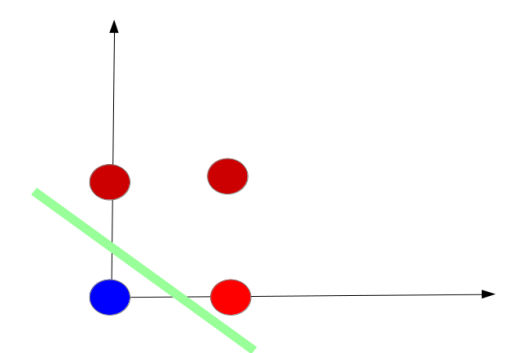
Le tableau ci-dessous récapitule les étapes d'apprentissage de cet ensemble de données avec le perception simple.

L'ensemble de données appris par le perceptron en **2 itérations** est séparable par la droite

**(D): -0,1 + 0,4x1 +0,9x2=0** contenant les points:**P(0, 1/9) et Q(1/4, 0).** Le schéma ci-dessous permet de visualiser la classification de l'ensemble de données par le perceptron.

L'ensemble de données appris par le perceptron en **2 itérations** est séparable par la droite

**(D): -0,1 + 0,4x1 +0,9x2=0** contenant les points:**P(0, 1/9) et Q(1/4, 0).** Le schéma ci-dessous permet de visualiser la classification de l'ensemble de données par le perceptron.



**Figure 1** Visualisation 2D des individus et de la droite obtenue par l'algorithme du perceptron

Nous remarquons que la droite obtenue sépare correctement l'ensemble de données en deux groupes bien distincts.

**Cas d'apprentissage par SVM**

En observant la figure 2.4 on remarque que les données sont linéairement séparables et qu'il y a trois supports de vecteurs à savoir : s1(0,0) s2(0,1) et s3(1,0). Nous noterons Φ() la fonction de mapping qui est aussi la fonction identité et « s'i » le support de vecteur i augmenté de la valeur

Trois supports de vecteurs :

* Le positif et le négatif passe par le points A(0 ;1) et B(1 ;0)
* Le support négatif passe par le point(0 ; 0) et est donc parallèle

On notera le système suivant :

Positif1 : w1(0)+w2(1)+w3=1

Positif2 : w1(0)+w2(0)+w3=-1

Negatif : w1(0)+w2(0)+w3=-1

W2+W3=1 (1)

W1+W3=1(2)

W3=1(3)

Resolution

(2) et (3) donnent : w1+ w3 = 1

W3=-1 (x-1)

Au final

W1=2

W2=2

W3=-1

D’après formule de D : w1X+w2Y+ w3=0

Donc D :2x+2y-1=0

A(0; ½) et B(1/2;0)

Le resultat graphique



 :

* 1. **Implémentation de l'algorithme du perceptron simple en C++**

**Fonctionnement du programme**

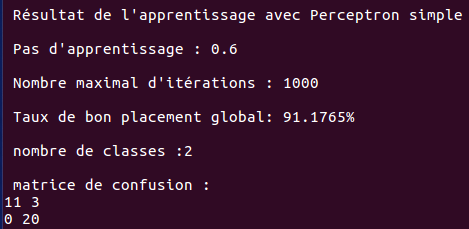
Le programme que nous avons mis en place nous permet de classifier les individus qui sont dans une base de test à l'aide de méthode ou de l'algorithme du perceptron simple. Ainsi, notre programme reçoit en paramètre une base d'apprentissage, une base de test, le pas d'apprentissage η et le nombre maximal d'itérations « max\_iter ». Ensuite l'algorithme du perceptron simple restitue en sortie une matrice de confusion et un taux global de bon classement qui renseignent sur la performance du classement effectué.

**Execution du programme :**

Pour exécuter notre programme nous devons saisir le syntaxe suivante : **./****perceptron\_simple « fichier\_base\_apprentissage » « fichier\_base\_test » η max\_iter**.

Après avoir exécuté cette commande, notre programme va lit l'ensemble des données d'apprentissage et des données de test puis va les stocker dans des tableaux. Ensuite, notre programme va essayer de trouver les bonnes valeurs des poids devant multiplier chaque entrée de la base d'apprentissage afin d'en prédire exactement la classe. Une fois les bons poids obtenus ils sont utilisés pour prédire les classes des entrées de la base de test. A partir de ces résultats la matrice de confusion et le taux de bon classement sont établit.

Par exemple de synthaxe: **./perceptron\_simple data/leukemia/ALLAML.trn data/leukemia/ALLAML.tst 0.6 1000.** Cette syntaxe exécute l'algorithme d'apprentissage du



perceptron simple sur la base d'apprentissage leukemia/ALLAAML.trn, puis fait le test sur l'ensemble de données leukemia/ALLAML.tst. Le résultat en sortie, nous donne une matrice de confusion ainsi qu’un taux de bon classement comme dans la figure si dessus. (fig 3).

**Figure3** résultat de l’exécution du perceptron simple sur l'ensemble de données leukemia

Pour exécuter notre programme sur les bases ovarian et spam, il est nécessaire d'effectuer un hold-out. Ce que nous réalisons au moyen des instructions ci-après:



**Figure4** Instructions nécessaires au hold-out de la base ovarian

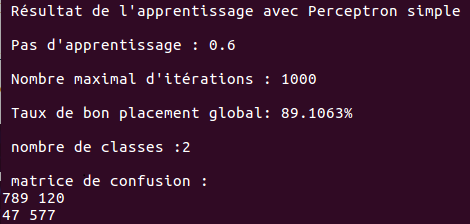


**Figure5** Instructions nécessaires au hold-out de la base spam

à chaque apprentissage des bases ovarian et spam, le résultat obtenu est potentiellement différent avec des paramètres identiques en entrée. Ci-dessous, nous vous présentons quelques résultats obtenus avec ces bases:



**Figure6:** Apprentissage par le Perceptron simple de la base ovarian



Nous avons Indiqué par ailleurs que dans le cas des bases « spam » et « leukemia » il est obligatoire d'effectuer une normalisation des données afin obtenir de bons résultats (taux de précision).

**Expérimentations :**

L’évaluation de l’algorithme d’apprentissage que nous avons implémentée est basée sur le taux global de bon classement et matrice de confusion établie. Pour ce cela, nous avons élaboré un tableau récapitulant les différents taux globaux de bon classement obtenus par la base leukemia avec différents valeur du pas d’apprentissage et du nombre d’itération maximal

|  |  |  |  |
| --- | --- | --- | --- |
| Dataset | Max Iteration | Pas d’apprentissage | Taux global de Bon classement |
| Leukemia | 100 | 0.1 |  |
| Leukemia | 100 | 0.4 |  |
| Leukemia | 100 | 0.6 | **91.1765%** |
| Leukemia | 1000 | 0.1 |  |
| Leukemia | 1000 | 0.4 |  |
| Leukemia | 1000 | 0.6 | **91.1765%** |
| Leukemia | 10000 | 0.1 |  |
| Leukemia | 10000 | 0.4 |  |
| Leukemia | 10000 | 0.6 | **91.1765%** |

**Tableau 3:** Variation du taux global de bon classement en fonction du nombre maximal d'itérations et du pas d'apprentissage sur les données de leukemia

Nous remarquons qu'avec un pas d'apprentissage de 0,6 appliqué sur quel que soit le nombre d'itérations, la base leukemia atteint son taux global maximal de bon classement (91,1765%). Notre programme arrive donc à bien apprendre et classer les données de la base leukemia pour ces valeurs de paramètres.

Comme pour le précèdent nous allons procéder de la même manière pour les bases ovarian et spam, tout en faisant la moyenne du taux global de bon placement sur trois exécution avec les mêmes paramètres d'entrées en raison du hold-out effectué sur les données à chaque exécution.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Dataset :ovarion | | | | | |
|  | Pas apprentissage =0.1 |  | Pas apprentissage =0.4 |  | Pas apprentissage =0.6 |
| Itération | Taux global de bon classement | Itération | Taux global de bon classement | Itération | Taux global de bon classement |
| 1 | 97.619 | 1 | 96.428 | 1 | 87.671 |
| 2 | 96.428 | 2 | 98,809 | 2 | 79.973 |
| 3 | 97.619 | 3 | 100 | 3 | 86.236 |
| **Moyenne** | **97.2222** | **Moyenne** | **98.412** | **Moyenne** | **84.627** |

Tableau4 : résultat l’expérimentation sur la base ovarian

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Dataset :spam | | | | | |
|  | Pas apprentissage =0.1 |  | Pas apprentissage =0.4 |  | Pas apprentissage =0.6 |
| Itération | Taux global de bon classement | Itération | Taux global de bon classement | Itération | Taux global de bon classement |
| 1 | 90.619 | 1 | 89.497 | 1 | 91.128 |
| 2 | 78.343 | 2 | 91,454 | 2 | 73.45 |
| 3 | 90.737 | 3 | 92.367 | 3 | 89.367 |
| **Moyenne** | **86.388** | **Moyenne** | **91.106** | **Moyenne** | **84.648** |

**Tableau 5:** Expérimentation avec la base spam

Suite à ces expérimentations, nous obtenons les meilleurs taux sont obtenus sur la base ≪ ovarian ≫ et la base ≪ spam ≫ avec un pas d’apprentissage de 0,4 et 1000 iterations. Nous remarquons également que pour ces deux bases, le programme a très bien appris et classé les données d’apprentissage. Ainsi , Nous concluons donc que notre implémentation du perceptron simple est correcte.

**1.2 Conception de l'architecture du perceptron multi-couches pour classer un ensemble**

**de données**



Suite aux données ci-dessus nous proposant l’architecture suivante

h1

w1

x1

w5

w2

o

w3

h2

w6

x2

w4

**1.3 Implémentation du perceptron multi-couches en C++**

**Fonctionnement du programme**

Notre programme permet de classifier les individus d'une base de test à l'aide de l'algorithme du perceptron multi-couche. Pour ce faire, il reçoit en entrée une base d'apprentissage, une base de test, le pas d'apprentissage η et le nombre maximal d'itérations ≪ max\_iter ≫. Après exécution de l'algorithme du perceptron simple, il restitue en sortie une matrice de confusion et un taux global de bon classement qui renseigne sur la performance du classement effectue.

**Exécution du programme :**

Pour exécuter notre programme nous devons saisir le syntaxe suivante : **./perceptron\_multiple « fichier\_base\_apprentissage » « fichier\_base\_test » η max\_iter**.

Après avoir exécuté cette commande, notre programme va lit l'ensemble des données d'apprentissage et des données de test puis va les stocker dans des tableaux. Ensuite, notre programme va essayer de trouver les bonnes valeurs des poids devant multiplier chaque entrée de la base d'apprentissage afin d'en prédire exactement la classe. Une fois les bons poids obtenus ils sont utilisés pour prédire les classes des entrées de la base de test. A partir de ces résultats la matrice de confusion et le taux de bon classement sont établit.

Par exemple de syntaxe: **./perceptron\_multiple ckeck.trn check.tst 0.6 1000 e**xécute l'algorithme d'apprentissage du perceptron multi-couches sur la base d'apprentissage test.trn, puis fait le test sur l'ensemble de données check.tst.

Le résultat de la commande donnée précédemment en exemple est :



**Figure8 :** Résultat d’Apprentissage par le Perceptron multi-couches sur la base check

Ce traitement nous donne un résultat tout à fait cohérent puisque les bases check.trn et check.tst sont identiques et ne servent qu'à valider notre programme. Notons que le programme que nous avons implémenté correspond à l'architecture présentée plus haut et n'est donc valable que pour des entrées à deux dimensions.

**Conclusion**

De tout ce précède, Les deux TPs qui nous a été demandés de réaliser dont les travaux ont été présentés ci- haut, nous ont permis de comprendre les concepts et mécanismes relatifs aux algorithmes d'apprentissage KNN, perceptron simple et multicouches et d'acquérir les compétences nécessaires a leur implémentation en c++. En dépit de tous les résultats de nos expérimentations, nous avons exposé les différentes implémentations qui sont valides mais aussi que chacun des algorithmes permettaient de bien classer les données.