|  |
| --- |
| M2 STIM |
| TP- Fusion d’information |
|  |
|  |
| **HAO Chang** |
| **18/02/2016** |

|  |
| --- |
|  |

Sommaire

[Introduction 2](#_Toc445191427)

[I. Méthodes de combinaison de type "Classe" 4](#_Toc445191428)

[1. Méthodes non paramétriques 4](#_Toc445191429)

[1.1 Vote à la pluralité : seuil = 0 4](#_Toc445191430)

[1.2 Vote à la majorité : seuil = 0,5 4](#_Toc445191431)

[2. Méthodes paramétriques : vote pondéré 5](#_Toc445191432)

[II. Méthodes de combinaison de type "Rang" 7](#_Toc445191433)

[1. Méthodes non paramétriques 7](#_Toc445191434)

[1.1 Borda count moyenne 7](#_Toc445191435)

[1.2 Meilleur rang 8](#_Toc445191436)

[2. Méthodes paramétriques 9](#_Toc445191437)

[III. Méthodes de combinaison de type "Mesure" 11](#_Toc445191438)

[1. Méthodes non paramétriques : somme, produit 11](#_Toc445191439)

[2. Méthodes paramétriques 13](#_Toc445191440)

[IV. Comparaison et conclusion 14](#_Toc445191441)

[Partie 2 TP Réalisé 15](#_Toc445191442)

[I. Bagging 15](#_Toc445191443)

[II. RSM (random subspace method) 17](#_Toc445191444)

[III. Adaboost 18](#_Toc445191445)

[Conclusion 19](#_Toc445191446)

# Introduction

Dans le cadre du cours de fusion d’information, nous avons étudié plusieurs méthodes de combinaison de classifieurs. Suivant le niveau d’information apporté par le classifieur, on peut distinguer trois catégories de sorties : classe, rang, mesure.

L’objectif de ce TP est de programmer quelques méthodes simples de combinaison de ces trois types, évaluer leur performance, comparer les méthodes.

***Données de travail:***

On dispose des résultats de 5 classifieurs (cl1, ..., cl5) pour un problème de reconnaissance à 10 classes (notées 0...9). Pour chaque classifieur, on dispose des résultats sur une base d'apprentissage (notée cli.app) et sur une base de test (notée cli.test).   
Chaque fichier contient 10001 lignes (1000 éléments/classe) :

* La première ligne indique le nombre de classe, le nombre maximum de solutions pour chaque élément à reconnaître et le nombre d'éléments à reconnaître par classe ;
* Les lignes suivantes sont les décisions du classifieur pour les éléments à reconnaître. Il y a au maximum 5 solutions proposées composées d'une classe et de la confiance associée entre crochets.

Ce TP est réalisé avec Matlab.

* 1. **Travail préliminaire**

***Objectif :***

Calculer les performances de reconnaissance en Top1 et en Top5 de chacun des classifieurs sur leur base d'apprentissage (les classifieurs ne rejettent pas!). Comparer et commenter.

Les données initiales sont en forme texte. Avant de commencer les travaux pratiques, il est nécessaire d’effectuer un prétraitement pour pouvoir importer les données.

La fonction data\_reader nous permet d’importer les fichiers txt et extraire et enregistrer les données sous la forme *.mat.* Nous avons enregistré les données de la base d’apprentissage et la base de test dans les cells baseApp et baseTest.

***Rappel :***

Pour une entrée donnée, un classifieur peut générer les réponses suivantes :

• Un rejet : pour indiquer que le classifieur n’a pas pu identifier cette entrée.

• Une reconnaissance : dans ce cas, il identifie bien l’entrée, et il lui attribue sa classe appropriée. • Une substitution : le classifieur attribue une autre classe à l’entrée.

La performance d’un classifieur peut être mesurée en calculant les trois taux suivant :

*(1)*

*(2)*

*(3)*

En calculant la réponse de toutes les 10000 données pour chaque classifieur, nous obtenons les performances de reconnaissance en Top1 et en Top5.

**Tableau 1:La performance de reconnaissance en Top1**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Taux\classifieur | 1 | 2 | 3 | 4 | 5 |
| Reco | 60 | 70 | 75 | 80 | 90 |
| Conf | 40 | 30 | 25 | 20 | 10 |
| Rejet | 0 | 0 | 0 | 0 | 0 |

**Tableau 2: La performance de reconnaissance en Top5**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Taux\classifieur | 1 | 2 | 3 | 4 | 5 |
| Reco | 88 | 89 | 89 | 97 | 96 |
| Conf | 12 | 11 | 11 | 3 | 4 |
| Rejet | 0 | 0 | 0 | 0 | 0 |

Notant que le taux de rejet sont nuls mais le nombre de rejet n’est pas zéro.

La performance en top5 est mieux qu’en top1 parce que la bonne solution peut se trouve dans les 5 premières propositions mais pas la première. Parmi les cinq classifieurs, le cinquième (cl5) est mieux que les autres.

# Méthodes de combinaison de type "Classe"

Dans cette partie, on ne prend en compte que la première solution fournie par le classifieur.

Chaque solution fournie par un classifieur est interprétée comme un vote.

## Méthodes non paramétriques

### Vote à la pluralité : seuil = 0

Le principe de l’opérateur est que la plupart du nombre de classifieurs devant répondre la même classe pour que cette classe soit la classe finale sinon rejet.

Résultats obtenus :

* Appliquer sur la base d’apprentissage :
  + - Nombre d’éléments : reco = 9646, conf = 71 rejet = 283
    - le taux de Reco pour la méthode de vote à la pluralité est 0.96460
* Appliquer sur la base de test :
  + - Nombre d’éléments : reco = 9650, conf = 73, rejet = 277
    - le taux de Reco pour la méthode de vote à la pluralité est 0. 965000

### Vote à la majorité : seuil = 0,5

Il s’agit du vote à la majorité. La classe ﬁnale est décidée si plus de la moitié des classiﬁeurs l’ont proposée c’est-à-dire si au moins k classiﬁeurs sont d’accords, k pouvant être déﬁni comme suit :

 (L est le nombre total de classifieurs)

* Appliquer sur la base d’apprentissage :
  + - Nombre d’éléments : reco = 9046, conf = 71, rejet = 883
    - le taux de Reco pour la méthode de vote à la majorité est 0.90460
* Appliquer sur la base de test :
  + - Nombre d’éléments : reco = 9057, conf = 73, rejet = 870
    - le taux de Reco pour la méthode de vote à la majorité est 0. 905700



**Figure 1: résultat de type classe non paramétriques pour la base d'apprentissage**

**(Gauche : majorité ; droite : pluralité)**



**Figure 2: résultat de type classe non paramétriques pour la base de test**

**(Gauche : majorité ; droite : pluralité)**

***Comparer :***

En ajoutant un seuil pour les décisions, le nombre de rejet augment. Par exemple, élément avec les votes [2 1 1 1 1] est rejeté. Donc le taux de reconnaissance est plus grand pour le vote à la pluralité. Mais le nombre d’élément male reconnues reste le même pour les deux opérateurs.

## Méthodes paramétriques : vote pondéré

– Les réponses des classifieurs (votes) sont pondérées par un coefficient wj indiquant leur importance dans la combinaison

– Les coefficients sont la fiabilité des classifieurs (reco/100-rejet)

Sachant qu’on ne prend en compte que la première solution fournie par le classifieur, les poids des classifieurs sur leur base d'apprentissage respective wj sont obtenue par la performance en top1 et ils sont strictement positives et de somme égale à 1：

coeffs\_top1 = Reco\_top1./(100-Ambi\_top1);

wj = coeffs\_top1./sum(coeffs\_top1);

wj = 0.1600 0.1867 0.2000 0.2133 0.2400

Appliquer sur la base de test :

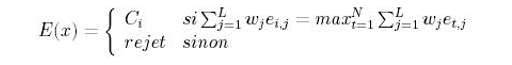
***Algorithme*** : [decision(n),result\_element] = VotePondereeClasse( baseTest,n,wj )

1. Calculer wj
2. Pour chaque élément n, initialiser : reco = 0,rejet = 0,conf = 0.

* extrait les solutions des 5 classifieurs (les 2, 4, 6, 8, 10 colonnes pour la nième ligne)

Ici on n’utilise que la première colonne.

* calcule la réponse (reconnus, rejet, confusion) et parcours toutes les éléments :



Si Ci est la bonne solution, reco = 1, sinon conf = 1.

result\_element = [correct,conf,rejet]

1. Compter tous les éléments et calculer le taux de reconnaissance

Résultat :

Résultat = 9828 ; 162 ; 10

TL = 0.9828

***Conclusion*** :

Comparer les résultats obtenus pour la base de test :

**Tableau 3: comparer les résultats de type classe**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Reconnues | Confusion | Rejet | Fiabilité |
| Vote pluralité | 9650 | 73 | 277 | 0.9925 |
| Vote majorité | 9057 | 73 | 870 | 0.9920 |
| Vote pondéré | 9828 | 162 | 10 | 0.9838 |

TL : Vote pondéré : 0.9828 > Vote pluralité : 0.965 > Vote majorité : 0. 9057

La performance de selon le taux de reconnaissance de combinaison de type « classe » de la méthode paramétrique et mieux que non paramétrique.

# Méthodes de combinaison de type "Rang"

## Méthodes non paramétriques

Programmer les opérateurs "borda count moyenne" et "meilleur rang". Les appliquer à la combinaison des 5 classifieurs sur leur base d'apprentissage et sur leur base de test. Comparer les performances obtenues par les deux opérateurs en Top1 et en Top5.

Les rangs attribués par chaque classifieur sont considérés comme des votes (pondérés) sur chaque classe. Pour un problème à 10 classes, chaque classifieur attribue les poids 10,9,...1. Ici on définit les poids : [1 0.9 0.8 … 0.1]. En réel, pour chaque classifieur, le nombre des solutions proposées maximal est 5 donc on note rij = [ 5 4 3 2 1]/5.

### Borda count moyenne

|  |
| --- |
| ***Algorithme*** : [ vote\_rang, top1, top5 ] = BCmoyenne( base,n );   1. Pour chaque élément n, initialiser : reco = 0,rejet = 0,conf = 0.  * extrait les solutions eji des 5 classifieurs ej * calcule BC(Ci) = somme de rangs pour la classe Ci /nombre de classifieurs dont solution propose * tri décroissant selon le rang total : vote\_rang = [vote, classe] * calcule la décision en top1 et top5 : result\_element = [reco,conf,rejet]  1. Compter tous les éléments et calculer les performances en top1 et en top5. |
| ***Fin.*** |

* Appliquer sur la base d’apprentissage :

**Tableau 4: les résultats de la méthode Borda count moyenne en top1 et top5 pour la base d'apprentissage**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N\_elements | Reconnues | Confusion | Rejet | Fiabilité |
| BCM TOP1 | 6361 | 2703 | 936 | 0. 701787 |
| BCM TOP5 | 8049 | 70 | 1881 | 0.991378 |



**Figure 3: les performances de la méthode Borda count moyenne en top1 et top5 sur la base d’apprentissage**

D’après les résultats obtenus, le nombre de rejet augment en top5 parce qu’il y a plus de probabilité que les points de la position 5 et 6 dans le liste du rang sont égaux mais le taux de reconnaissance et la fiabilité est plus haut.

* Appliquer sur la base de test:

**Tableau 5: les résultats de la méthode Borda count moyenne en top1 et top5 pour la base de test**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N\_elements | Reconnues | Confusion | Rejet | Fiabilité |
| BCM\_TOP1 | 6885 | 2343 | 772 | 0. 746099 |
| BCM\_TOP5 | 8051 | 84 | 1865 | 0. 989674 |



**Figure 4: les performances de la méthode Borda count moyenne en top1 et top5 sur la base de test**

On observe la même caractéristique que la base d’apprentissage.

Lors de la combinaison de cette méthode, la définition des points (la règle des rangs) va influer la performance gravement.

### Meilleur rang

On attribue à chacune des classes le rang le plus élevé *MR(Ci)* (meilleur rang) parmi les rangs proposés par les classifieurs puis tri par ordre décroissant de la liste.

|  |
| --- |
| ***Algorithme*** : [ decisionMR,top1,top5 ] = MeilleurRang( base,n )   1. Pour chaque élément n, initialiser : reco = 0, rejet = 0, conf = 0.  * Extrait les rang (eji ) proposes par les classifieurs * pour chaque classe Ci, calcule MR(Ci) = meilleur rang * Décision finale = {Ci pour MR(Ci)=1} * calcule la réponse de la décision finale en top1 et top5   result\_element = [reco,conf,rejet]   1. Compter tous les éléments et calculer les performances en top1 et en top5. |
| ***Fin.*** |

* Appliquer sur la base d’apprentissage :

**Tableau 6: les résultats de la méthode meilleur rang en top1 et top5 pour la base d'apprentissage**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N\_elements | Reconnues | Confusion | Rejet | Fiabilité |
| MR\_TOP1 | 2316 | 0 | 7684 | 0.944137 |
| MR\_OP5 | 9994 | 6 | 0 | 0.998522 |



**Figure 5: les performances de la méthode meilleur rang en top1 et top5 sur la base d'apprentissage**

* Appliquer sur la base de test :

**Tableau 7: les résultats de la méthode meilleur rang en top1 et top5 sur la base de test**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N\_elements | Reconnues | Confusion | Rejet | Fiabilité |
| MR\_TOP1 | 2679 | 484 | 6837 | 0. 846981 |
| MR\_TOP5 | 9997 | 3 | 0 | 0.9997 |



**Figure 6: les performances de la méthode meilleur rang en top1 et top5 sur la base de test**

Les performances obtenues en top5 (TL = 99.97%) sont beaucoup mieux que celui en top1 (TL = 26.79%). La décision finale n’est pas une seule classe mais toujours un ensemble de classes donc on ne peut pas prendre la décision en top1, le rejet est nombreux. Par contre on peut trouver la bonne solution en prenant compte tous les solutions proposées par le meilleur rang.

On peut imaginer qu’avec un grand nombre de classiﬁeurs cette méthode n’améliore pas la performance.

Comparer avec la méthode Borda count moyenne, en top1 la performance du borda count moyenne est mieux avec un TL 65(+/-3)% et en top5 la méthode du meilleur rang est mieux avec un TL 99.97%.

## Méthodes paramétriques

Programmer l'opérateur "somme pondérée" qui est une généralisation du borda count. Déterminer les poids des classifieurs sur leur base d'apprentissage respective et appliquer l'opérateur à la combinaison des 5 classifieurs sur la base de test. Calculer les performances en Top1 et en Top5. Conclusion.

|  |
| --- |
| ***Algorithme*** :   1. Calculer les poids wj : en utilisant la fiabilité en top5 2. Pour chaque élément n :   [rang\_SP(n,:),decision\_SP, top1,top5]= Rang\_SP( baseTest,n,wjTop5 )  Initialiser : reco = 0, rejet = 0, conf = 0.   * extrait les solutions eji des 5 classifieurs ej * calcule SP(Ci) = * tri décroissant selon le rang SP : decision\_SP = [vote, classe] * calcule la décision en top1 et top5 : result\_element = [reco,conf,rejet]  1. Compter tous les éléments et calculer les performances en top1 et en top5. |
| ***Fin.*** |

***Résultat :***

**Tableau 8: les résultats de la méthode somme pondéré en top1 et top5 pour la base de test**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N\_elements | Reconnues | Confusion | Rejet | fiabilité |
| SP\_TOP1 | 7949 | 1240 | 811 | 0.865056 |
| SP\_TOP5 | 7803 | 19 | 2178 | 0.997571 |



**Figure 7: les performances de la méthode somme pondéré en top1 et top5 sur la base de test pour SP**

En ajoutant les taux de fiabilité des classifieurs, la performance en top1 est améliorée.

La fiabilité en top5 est plus haute qu’en top1.

# Méthodes de combinaison de type "Mesure"

Dans cette partie, on utilise les mesures associées aux classes par des classifieurs. La sortie de type mesure est la plus riche en informations puisque le classifieur dans ce cas associe à chaque classe une mesure de confiance qui peut être, par exemple, une probabilité.

## Méthodes non paramétriques : somme, produit

|  |
| --- |
| ***Algorithme*** : l’opérateur somme & produit   1. Pour chaque élément n :  * Extrait les listes à combiné des classifieurs (mesure : colonne 3 5 7 9 11 ;) * Extrait les mesures (mij ) associées aux classes par les classifieurs * Calculer le somme et le produit des mesures des 5 classifieurs mij de la classe Ci * Tri le liste du somme et du produit par ordre décroissant. * calcule la décision en top1 et top5 : result\_element = [reco,conf,rejet]  1. Compter tous les éléments et calculer les performances en top1 et en top5. |
| ***Fin.*** |
|  |

***Appliquer sur la base d’apprentissage :***

**Tableau 9: les résultats de la méthode de type « mesure » en top1 et top5 pour la base d’apprentissage**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| operateur |  | Reconnues | Confusion | Rejet | Fiabilité |
| somme | Top1 | 9703 | 297 | 0 | 0.9703 |
| Top5 | 9948 | 5 | 47 | 0.9995 |
| produit | Top1 | 6521 | 332 | 3147 | 0.9515 |
| Top5 | 6583 | 270 | 3147 | 0.9606 |



**Figure 8 : les performances des opérateurs de la méthode de type « mesure » en top1 et en top5**

***Appliquer sur la base de test:***

**Tableau 10 : les résultats de la méthode de type « mesure » en top1 et top5 pour la base de test**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| operateur |  | Reconnues | Confusion | Rejet | Fiabilité |
| somme | Top1 | 9709 | 291 | 0 | 0.9709 |
| Top5 | 9952 | 7 | 41 | 0.9992 |
| produit | Top1 | 6626 | 340 | 3034 | 0.9512 |
| Top5 | 6767 | 199 | 3034 | 0.9714 |



**Figure 9 : les performances des opérateurs de la méthode de type « mesure » en top1 et en top5**

## Méthodes paramétriques

|  |
| --- |
| ***Algorithme*** : l’opérateur somme pondéré & produit pondéré appliquent sur la base de test   1. Pour chaque élément n :  * Extrait les listes à combiné des classifieurs (mesure : colonne 3 5 7 9 11 ;) * Extrait les mesures (mij ) associées aux classes par les classifieurs * Calculer :     Tri le liste du somme pondéré et du produit pondéré par ordre décroissant.   * calcule la décision en top1 et top5 : result\_element = [reco,conf,rejet]  1. Compter tous les éléments et calculer les performances en top1 et en top5. |
| ***Fin.*** |

**Tableau 11:les résultats de la méthode de type « mesure » paramétriques en top1 et top5**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| operateur |  | Reconnues | Confusion | Rejet | Fiabilité |
| Somme pondéré | Top1 | 9733 | 267 | 0 | 0. 9988 |
| Top5 | 9988 | 6 | 6 | 0. 9994 |
| Produit pondéré | Top1 | 6630 | 336 | 3034 | 0. 951766 |
| Top5 | 6767 | 199 | 3034 | 0. 971433 |



**Figure 10 : les performances des opérateurs de la méthode de type « mesure » paramétriques en top1 et en top5**

# Comparaison et conclusion

**Figure 11: comparaison les reconnues des méthodes appliquées sur la base de test**

D’après la figure ci-dessous, nous retrouvons que la méthode de type mesure est mieux que la méthode de type rang. L’opérateur de somme pondéré en top5 (99.9%) et le meilleur rang en top5 (99.9%) ont la même performance et sont les meilleurs parmi les méthodes appliquées.

Le taux de reconnaissance des opérateurs de produit (66% en top1 et 67% en top5) n’est pas très satisfaisant mais la fiabilité est acceptable (95% en top1 et 97% en top5).

Les performances des méthodes de combinaison sont très différentes et difficiles d’avoir une conclusion générale. Nous pouvons constater que pour le même type de sortie, la méthode paramétrique est mieux que la méthode non paramétrique. En revanche, la performance en top5 est généralement mieux que celui en top1.

# Partie 2 TP Réalisé

Dans ce partie, nous voulons réaliser les méthodes de construction d’ensembles de classifieurs : bagging, boosting, RSM et ECOC.

Les données que nous avons traitées ici sont les données Pima. Les données sont séparées en deux parties, 2/3 pour la base d’apprentissage et 1/3 pour la base de test.

## Bagging

Le Bagging repose sur la méthode boostrap pour améliorer la prédiction d’une classiﬁeur. On va donc tirer uniformément et avec remise les observations pour créer de nouveaux échantillons, sur chacun desquels on construit un classiﬁeur. On obtient le classiﬁeur ﬁnal par un vote à la majorité parmi les classiﬁeurs intermédiaires. L’idée majeure est donc d’obtenir divers classiﬁeurs, basés sur de nombreux échantillons.

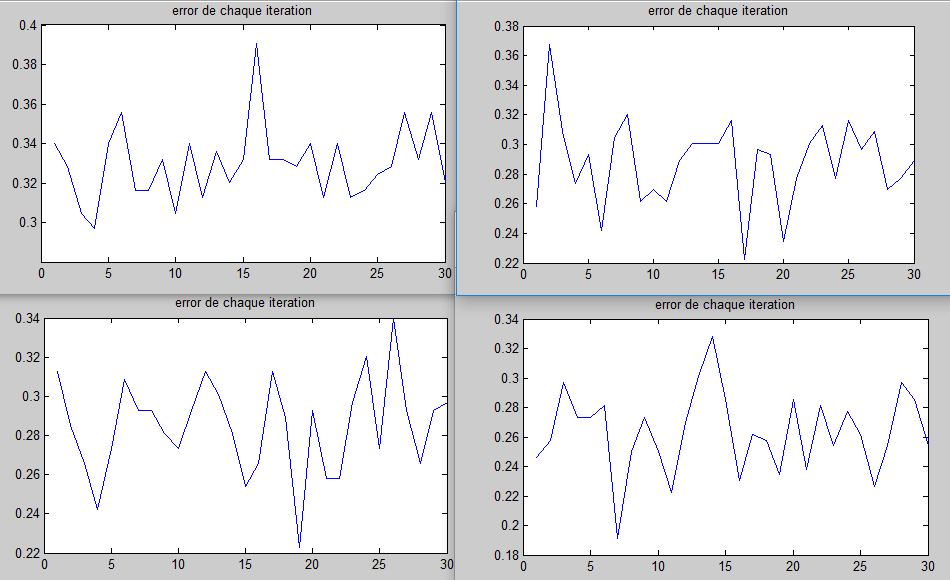
L’algorithme peut alors s’écrire de la manière suivante :

|  |
| --- |
| **Algorithme 1** : Bagging |
| **Entrée:** B= base d’apprentissage  n = taille de B  L = nombre de classifieur    **Début :**  **Pour i = 1 : L, faire**   1. Tirer un échantillon “boostrap” (avec remise dans B) Bbn = {(Xbi ,Ybi ) : 1 ≤ i ≤ n}; 2. Calculer le classiﬁeur Ci sur cet échantillon Bi comme la base d’apprentissage et calculer appliquer sur la base de test. Ici, nous utilisons Kppv comme le classifeur faibel.   Fin  Combiner les L classifieurs comstruit par le vote à majorité afin d’obtenir la décision finale  **Fin** |

D’abord on travaille sur les données PIMA. Ici on utilise le Kppv avec k = 1,5, 7 ,15 à la fois pour la version avec et sans bagging.

Les taux de correct obtenus sont :

|  |  |  |
| --- | --- | --- |
| K | Sans bagging | Avec bagging |
| 1 | 0.675781 | 0.679688 |
| 5 | 0.726563 | 0.722656 |
| 7 | 0.714844 | 0.714844 |
| 15 | 0.769531 | 0.757813 |



**Figure 12: l'erreur de chaque itération avec L = 30, k=1,5, 7 ,15**

En comparant les résultats, nous trouvons que la méthode bagging n’a pas d’influence significatif pour kppv. Sachant que le Kppv est un classifieur stable par rapport à la décision d’arbre, ce résultat remontrer le fait que le bagging a peu d’influence sur les classifieurs stable.

Pour les données SATIMAGE, on prend k = 1, le taux de correction sans bagging k = 1 est 0.8945 et avec bagging le résultat devient 0.685.



**Figure 13: l'erreur de chaque itération avec L = 30, k = 1**

## RSM (random subspace method)

|  |
| --- |
| **Algorithme 2** : RSM |
| **Entrée:** B= base d’apprentissage  M = nombre de l’attribut  L = nombre de classifieur    **Début :**  **Pour i = 1 : L, faire**   1. Choisir les sous-espaces des caractéristiques aléatoirement pour tous les données de la base d’apprentissage et construire la base d’app Bi. 2. Calculer le classiﬁeur Ci sur cet échantillon sous-espace comme la base d’apprentissage et appliquer sur la base de test.   Fin  Combiner les L classifieurs construit par le vote à majorité afin d’obtenir la décision finale  **Fin** |

Pour les données PIMA, on utilise le Kppv avec k =1, le taux de correction obtenu avec la méthode RSM est 0. 656250 et 0.707 pour k = 5.

****

**Figure 14: PIMA : l'erreur de chaque itération avec L = 10, k =5**

D’après les résultats obtenus, nous pouvons constater que la méthode de RSM peut influe sur le classfifieur Kppv. En prenant différents sous-espaces de caractéristique aléatoirement, le résultat de Kppv vas varie, donc dépend des attributs.

## Adaboost

|  |
| --- |
| **Algorithme 3** : Adaboost |
| **Entrée:** B= base d’apprentissage  S = nombre de l’élément  L = nombre de classifieur    **Début :**  Initialiser le poids W1(s) = 1/S, s = 1, 2, …, S  **Pour i = 1 : L, faire**   1. Entrainer un classifieur Ci sur la base d’apprentissage B 2. Calculer le taux d’erreur Ei = 3. . Alpha est la fiabilité du classifier 4. Mis à jour les poids W(i+1) = W(i)\*exp(b\*alpha(i)) b = -1 si bien classé, si non b =1 ;   Fin  Combiner les L classifieurs construit par le vote pondéré avec les alphas afin d’obtenir la décision finale  (T = L)  **Fin** |

Le problème d’utiliser le Kppv ici est que l’ordre des données dans la base d’apprentissage ne change pas donc les résultats classifiés ne change pas et le poids d’erreur d’un élément est cumulé dans chaque itération d’apprentissage. Le résultat obtenu pour k =5 est T\_correc = 0.7266, la performance n’est pas améliorée pour un classifieur Kppv.

Si on utilise cette méthode pour le classifieur threshhold, nous pouvons observer que le taux d’erreur est diminué.



**Figure 15: threshold avec Adaboost**

D’apres les résultst, nous pouvons constater que, dans ce cas, la méthode Adaboost est mieux que la méthode de bagging.

## Conclusion

Dans cette partie, nous avons appliqué trois méthodes de construction d’ensemble de classifieur pour le problème de deux classes. A cause de manque de temps, je n’ai pu appliquer les méthodes pour le cas de multi-classe.

La performance d’amélioration de la méthode bagging dépende de la stabilité du classifieur. Plus le classifieur instable, plus la performance est améliorée. Quand le classifieur est stable, on n’observe pas d’amélioration signifiante.

En générale la méthode de Adaboost est mieux que le bagging parce que il prend en compte les poids pour les donées. Par contre il peut tendance vers les donées qui sont toujours mal classées à cause du poids augmentant. Imaginons qu’il y a quelques éléments anormaux, le résultat de boosting sera non idéal.