Machine Learning

Compte-rendu TP2

Écrit par :
Romain Charpentier
Morgane Thielemann

Date : 23/11/2018

Note : Attention. Dans ce rapport, nous n'affichons qu'un seul schéma par jeu de tests. En effet, les schémas étaient quasiment identiques en faisant variant les données. La différence ne se voit donc que par le nombre d'erreurs.

1. Explications des algorithmes de décision

a. Algorithme KNN

Pour décider de la classe d'un point, on va le comparer à k voisins proche de l'ensemble d'apprentissage(test). Pour cela, on remplit une première matrice de distance appelé *norm* qui indique la distance entre l'individu de *x* et l'individu de *test.* Ainsi *norm(i,j)* représente la distance entre l'individu *x(i)* et l'individu *test(j)*.

On trie ensuite la matrice obtenu par colonne pour obtenir les k plus proches voisins de *x(i).* On récupère ainsi un tableau *index* qui indique le réarrangement des données précédentes. Plus clairement, si index(i,j) vaut k, alors cela signifie que la valeur *sortNorm(i,j)* se trouvait auparavant à la position (i,k) dans *norm.*

On va donc utiliser cet index pour recréer un tableau de classe qui ne contient que les classes des k premiers individus. Ensuite, on va remplir classTest pour compter le nombre de fois où chaque classe est représentée dans les k plus proches voisins.

On récupère ensuite le maximum pour chaque individu afin d'en déterminer la classe qu'on récupère dans *clas*.

b. Algorithme Bayes

L'algorithme de Bayes se déroule en 2 étapes, on prépare d'abord l'algorithme à travers une phase d'entraînement pour ensuite pouvoir le lancer dans une 2ème étape.

Pour l'entraînement, on dispose d'un jeu de données dont on connaît la classification. Ainsi pour chaque classe, on peut calculer la moyenne (donc le centre) de cette dernière. On calcule également la probabilité à priori de chaque classe. Il faut donc diviser le nombre d'éléments de la classe par le nombre d'éléments totals pour l'obtenir. Pour finir, on prépare une hypermatrice de taille 2*2*C (C étant le nombre de classes). Pour chaque classe, il y a donc une matrice 2*2 correspondant aux variances et covariance de la classe (s'il n'y a que 2 mesures, et c'est le cas dans nos exemples).

Ensuite pendant l'algorithme, on prend nos données de l'entraînement pour appliquer la formule à chaque point que l'on étudie et pour chaque classe. On choisit la classe dont la valeur résultante de la formule est la plus forte pour ce point.

2. Premiers tests

Dans un premier temps, voici les résultats de l'algorithme Knn pour les diverses valeurs de k. Ensuite les résultats de l'algorithme naïf de Bayes.

Paramètre	**Nombre d'erreurs Knn**
k=1	18
k=3	13
k=5	10
k=7	8
k=13	6
k=15	8

Erreurs Bayes : 8.

Quand on regarde ces résultats, on constate que Bayes est plus efficace dans la plupart des cas. Cependant pour k=13, Knn est plus efficace. En y regardant de plus près, on peut constater que le jeu de données est fait de sorte que la dispersion des valeurs rend l'algorithme plus efficace pour k=13 car les données de la première classe sont très dispersées.

3. Entraînement des données sans expert : utiliser l'algorithme Kmean

Résultats KNN :

Paramètre	Nombre d'erreurs
k=1	20
k=3	18
k=5	15
k=7	15
k=13	16
k=15	17

Résultat Bayes : nombre d'erreurs valant **14**

On peut observer que Knn a atteint sa limite, nous n'arrivons pas à avoir moins de 15 erreurs pour cet exemple, même en agrandissant k. Cependant avec Bayes, nous avons 14 et il n'y a pas besoin de choisir une valeur pour k. Donc l'algorithme de Bayes a le meilleur résultat pour le second jeu de données.

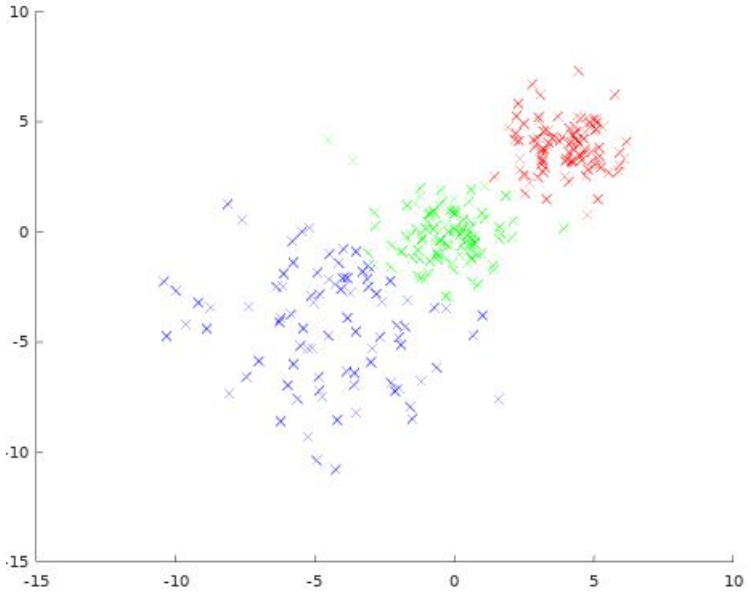


Schéma de répartition des points

4. L’influence de la taille de la base de d’apprentissage

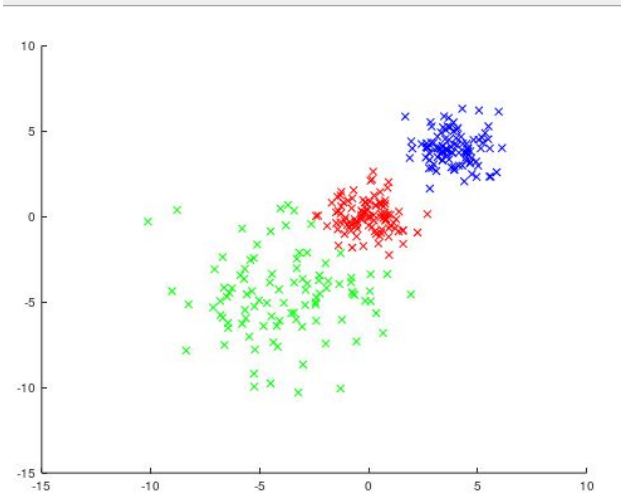
a. Petite base de donnée

Paramètre	**Nombre d’erreurs Knn**
k=1	11
k=3	9
k=5	7
k=7	8
k=13	11
k=15	11

Résultat Bayes : nombre d’erreurs : 5

On observe avec ces résultats que Bayes est constamment plus efficace que Knn. Cela peut s’expliquer par le fait que Knn a facilement tendance à regarder des voisins qui ne sont pas significatifs parce qu’il y a peu de données. (En effet, si on a 5 données en 3 classes, si on regarde sur 2 données, le résultat est peut-être juste mais il le sera sûrement moins si on regarde les 5 plus proches voisins).

Bayes, lui, ne subit pas cette influence car il est plus influencé par la dispersion des valeurs au sein d’une même classe que par le nombre d'éléments dans cette classe.



Graphique des données regroupées avec l’algorithme de Bayes

b. Grande base de donnée

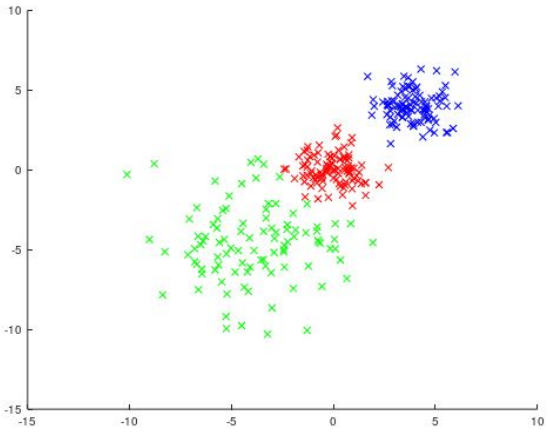
Résultats :

Paramètre	Nombre d'erreurs Knn
k=1	6
k=3	5
k=5	4
k=7	3
k=13	5
k=15	4

Résultats Bayes : 4

On constate une fois de plus que Bayes obtient de meilleurs résultats que Knn pour la plupart des k mais une fois de plus Knn est meilleur pour k=7.

Cependant, en comparant avec les résultats précédents, on peut en déduire que sur un grand jeu de données, utiliser Knn ou Bayes ne fait pas une si grande différence (au pire 2 erreurs de différences pour les données testées). Au contraire, sur un ensemble plus petit, il devient très intéressant d'utiliser Bayes car il fait beaucoup moins d'erreurs (au pire 6 erreurs de différences, au mieux 4).



Graphique des données regroupées avec l'algorithme de Bayes

5. Distribution non gaussienne : distribution inconnue

Résultats KNN :

Paramètre	**Nombre d'erreurs**
k=1	5
k=3	5
k=5	5
k=7	3
k=13	9
k=15	6

Résultat Bayes : nombre d'erreurs valant **6**

Pour ce dernier exemple, la répartition est non-gaussienne. L'algorithme de Bayes utilisant une approche gaussienne pour son algorithme de décision est donc désavantagé.
On observe donc que Knn a de meilleurs résultats pour presque toutes valeurs de k. On arrive même jusqu'à seulement 3 erreurs (sur 300 donc seulement 1%). Cependant, il faut faire attention car pour k=13, le nombre d'erreurs est plus important que pour Bayes.
Pour k=17 et k=20, on obtient respectivement 7 et 6 erreurs, donc cette hausse pour k=13 est dû à de la malchance dans l'algorithme (point situé au bord de la classe).

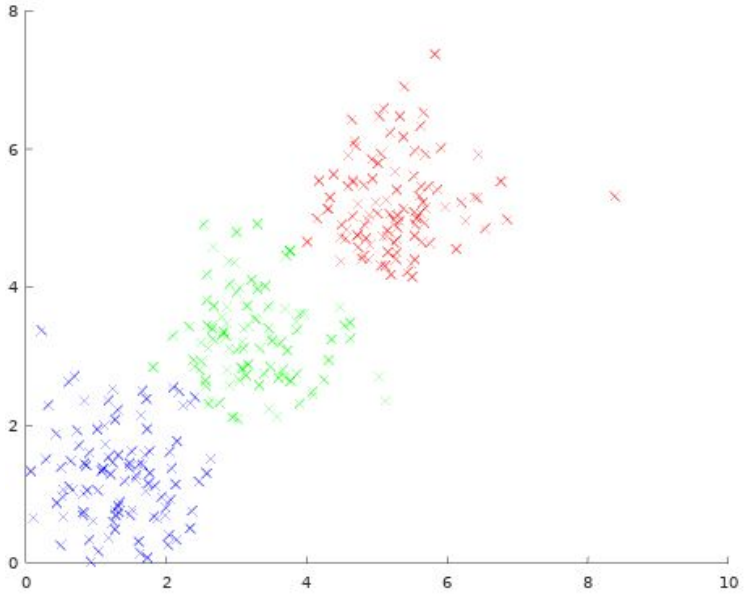


Schéma de répartition des points