

Résultats de recherche sur la classification supervisée

M. Deker Sylvain¹

Mme. Courdy-Bahsoun Clémence²

¹ M1 IGAI Université Paul Sabatier

² M1 IGAI Université Paul Sabatier

sylvain.deker@univ-tlse3.fr

clemence.courdy-bahsoun@univ-tlse3.fr

Résumé

Cet article vise à comparer les avantages et les désavantages des classifications supervisées par bayésienne avec gaussienne et par la méthode des k-plus proches voisins, en s'appuyant sur les résultats obtenus sur un même jeu de données.

Mots Clef

Apprentissage, classification bayésienne, classification k-ppv.

1 Introduction

La classification bayésienne avec gaussienne et l'approche par la méthode des k-ppv sont des classifications supervisées dont l'objectif est de répartir les données par classe avec un taux d'erreur minimisé. Les tests sont menés sur un jeu de données contenant une centaine d'occurrences de la prononciation d'une dizaine de voyelle au format csv.

A l'aide des travaux réalisés précédemment et en adaptant un peu le code nous avons pu implémenter le modèle de Bayes. L'implémentation pour une approche par les k-ppv, le sujet nous imposait l'utilisation de *Scikitlearn*.

Dans les tests menés les données fournies ont été réparties de telle sorte à avoir 80% de données d'apprentissage et 20% de données de tests. Dans les démarches d'apprentissage le choix d'un taux de données d'apprentissage élevé favorise l'obtention de meilleurs résultats, ici une meilleure prédiction de la répartition par classe.

2 Classification bayésienne avec Gaussienne

La classification bayésienne avec Gaussienne est une minimisation de la vraisemblance qui vise à prédire la classe d'appartenance la plus probable d'une donnée. Néanmoins avec des données réelles il est difficile d'obtenir des résultats satisfaisants car les données ne sont pas simples. Ainsi pour améliorer le modèle on tient compte de la probabilité a priori en appliquant la loi de Bayes.

Dans les tps précédant, nous avons déjà implémenté en python l'algorithme de prédiction permettant de classer les

données fournies. Ici nous sommes repartis de ce code qui était basé sur une simplification de la vraisemblance en log-vraisemblance. C'est-à-dire que la vraisemblance qui est une expression exponentielle a été simplifiée par l'application du logarithme népérien afin d'obtenir un calcul un peu plus optimisé. De plus toutes les constantes ne dépendant pas des données ont été supprimées, n'apportant pas une précision indispensable dans la reconnaissance des données.

3 Classification k-NN

La méthode des K plus proches voisins (K-NN) a pour but de classer des points cibles (classe méconnue) en fonction de leurs distances par rapport à des points constituant un échantillon d'apprentissage (c'est-à-dire dont la classe est connue a priori). Le calcul de la distance peut dépendre de l'expérience menée, Scikitlearn se base sur un calcul euclidien de la distance entre les données de l'échantillon, placées dans un repère après un traitement par une Analyse en Composantes Principales, et les centroïdes de la base d'apprentissage. Scikit learn est une bibliothèque libre pour python dont certaines fonctionnalités ont été utilisées pour implémenter la classification par k-NN. En effet toutes les fonctions appelées sur la variable n de la fonction k_NN sont des fonctions appelées pour la classe KNeighborsClassifier, une classe de la bibliothèque permettant d'effectuer les traitements nécessaires à une classification par plus proche voisin.

Algorithm 1 k_NN

Require: train_data, train_labels, test_data, test_labels

Ensure: m : matrice de confusion

p : precision

n = neighbors.KNeighborsClassifier(n_neighbors, weights = 'distance')

n.fit(train_data, train_labels) %apprentissage

y = n.predict(test_data) mconf = matrice-Conf(test_labels, y) %calcul matrice de confusion, fonction rédigée

return (mconf[0], mconf[1])

4 Evaluation par validation croisée

La validation croisée est une méthode d'estimation de la fiabilité d'un modèle fondée sur la technique d'échantillonnage. Les ensembles de données d'apprentissage sont les fichiers de test data2.csv, data3.csv et data12.csv sur lequel on peut entraîner le modèle. Nous appelons "learning ratio" la proportion des données d'un fichier dédiée à l'apprentissage, la partie restante sert d'échantillon de validation. Cette méthode de test s'appelle "testset validation" ou "holdout method".

5 Résultats

5.1 Matrice de confusion

Afin d'analyser et de pouvoir comparer les résultats obtenus par les deux méthodes étudiées ici, on calcule pour chaque échantillon la matrice de confusion, permettant de déduire un taux d'erreur. Cette matrice est construite à l'aide d'une classe présente dans la bibliothèque Scikit-learn et en appliquant les calculs nécessaires pour calculer le taux d'erreur.

5.2 Learning ratio = 0.2

Files test	Bayes		KNN		Evolution from Bayes to KNN	
	Times	Precision	Times	Precision	Time Acceleration	Delta Precision
data2.csv	0.1538571166992	0.8186	0.0031902837783	0.7916	48.22575011047	0.0169
data3.csv	0.1633808888926	0.9789	0.0040844964981	0.9783	40.000252185294	0.0006
data12.csv	0.2776164579391	0.9997	0.0065380048752	1	42.481953338842	0.0003

5.3 Learning ratio = 0.4

Files test	Bayes		KNN		Evolution from Bayes to KNN	
	Times	Precision	Times	Precision	Time Acceleration	Delta Precision
data2.csv	0.270541374588	0.81255	0.0040290403366	0.7911	67.172844450585	0.01545
data3.csv	0.3280623006821	0.9727	0.0050388731232	0.9749	65.083384984512	0.0022
data12.csv	0.5476139545441	0.9995	0.009666534702	1	56.650500019978	0.0005

5.4 Learning ratio = 0.6

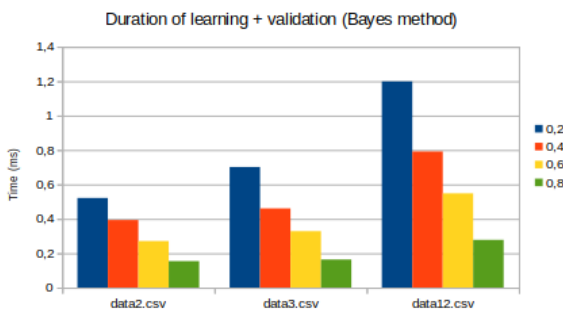
Files test	Bayes		KNN		Evolution from Bayes to KNN	
	Times	Precision	Times	Precision	Time Acceleration	Delta Precision
data2.csv	0.3923126220703	0.8080333333333	0.0050431013107	0.7854333333333	77.79197519797	0.0126
data3.csv	0.460188983154	0.9706666666667	0.00565688959	0.973	78.584675183777	0.0023333333333
data12.csv	0.790948343277	0.9994	0.0108628463745	1	72.812255278951	0.0006

5.5 Learning ratio = 0.8

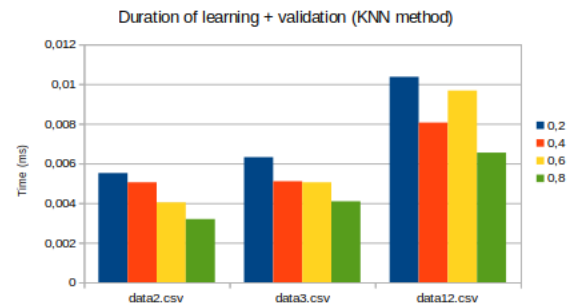
Files test	Bayes		KNN		Evolution from Bayes to KNN	
	Times	Precision	Times	Precision	Time Acceleration	Delta Precision
data2.csv	0.5208987569809	0.79775	0.005506310483	0.7845	94.600324570451	0.01325
data3.csv	0.7010438108444	0.9586	0.0063081979752	0.96305	111.1321828523910	0.00345
data12.csv	1.188479310159	0.9811	0.0103621282114	1	115.65096214433	0.1489

5.6 Analyse temporelle

Bayes.

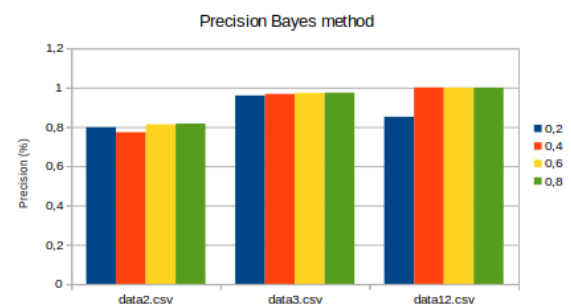


K_NN.

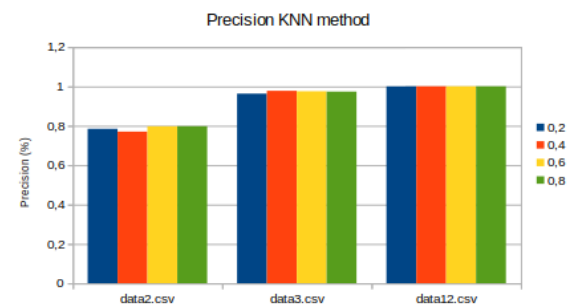


5.7 Analyse précision

Bayes.



K_NN.



6 Conclusion