Alexandre Pais Gomes

Apprentissage Automatique Numérique – Classifieur Bayésien

Master 1 ISI, Université du Maine.

Variantes du classifieur

On obtient des résultats différents selon la taille des données ACP.
D'après mes tests, on obtient le meilleur taux avec 35 et 36 qui ont tout deux 168 erreurs soit 3.36% d'erreurs sur les 5 000 exemples du corpus de développement :

Taille des données d	Nombre d'erreurs	Pourcentage
33	171	3.42%
34	176	3.52%
35	168	3.36%
36	168	3.36%
37	172	3.44%
38	170	3.40%

J'ai choisit d=36 arbitrairement car la courbe me semble moins ascendante après d=36 que descendante avant d=35 d'où mon choix pour d=36 mais rien ne me permet d'affirmer qu'une solution est meilleure que l'autre puisqu'elles ont chacune le même taux d'erreurs.

Pré-requis

Les fichiers de données (appr_cl.ascii, appr.ascii, etc.) doivent se trouver dans un dossier data/ et dans ce dossier data/ il doit exister 3 répertoires : acp/, appr/ et dev/ qui stockeront les données générées dans le script.

Lancement du script

Au début de mon script, j'ai instancié plusieurs variables permettant de lancer (ou non) les différentes phases. Par défaut, toutes les phases sont lancées. Pour n'en lancer que certaines, il faut changer les paramètres initiaux de mon script à la main.

Résultats

Le tableau de confusion résultant de la phase de développement se lit de la façon suivante : confusion(x,y) correspond au nombre de fois où le système a donné une classe x alors que la classe étant y (x étant la ligne, y la colonne). Ainsi par exemple, mon système a pris 11 fois un 9 pour un 5.

0	0	0	0	0	0	3	0	0	1
0	0	0	0	0	0	0	0	1	0
0	1	0	6	4	0	1	9	10	5
2	0	3	0	0	14	0	3	9	2
0	0	0	0	0	0	0	0	0	11
0	0	0	1	0	0	14	1	1	1
0	0	0	0	0	0	0	0	0	0
0	1	0	0	1	0	0	0	0	5
1	13	0	8	2	15	1	5	0	5
0	0	0	0	2	0	0	4	2	0

Performances

Pour éviter la redondance de code, j'ai fusionné les phases (d'apprentissage et de développement) en deux fonctions. Il existe ainsi une fonction par phase soit :

- une fonction apprentissage(appr_cl, appr_acp) qui calcule les probabilités à-priori, les moyennes et covariances de chaque classe
- une fonction developpement(dev_cl, dev_acp, pwi, covariance, ui, d) qui, à partir des données précédemment calculées et les données de développement, calcule le nombre d'erreurs, le pourcentage d'erreur et le tableau de confusion

Cependant, bien que le code soit moins redondant et plus lisible, ici les performances peuvent être questionnées puisque la copie des données passées en paramètres des fonctions est probablement très coûteuse en terme de temps d'exécution.

A noter tout de même que le temps d'exécution du système final n'en sera pas influencé puisque les fonctions en question existent uniquement pour les 2 premières phases (apprentissage et développement) et donc le système final qui charge les données d'apprentissage à partir de fichiers externes et exécute uniquement la phase d'évaluation n'est pas préjudicié.