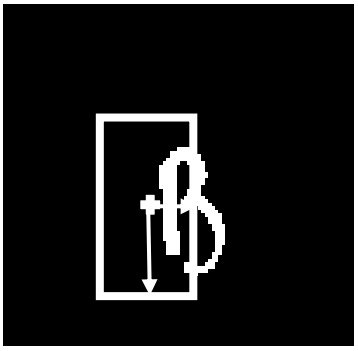
Travaux pratiques Reconnaissance des formes
TP1 – Mise en forme des données et plus proche voisin

On souhaite reconnaître des caractères manuscrits numérisés à l'aide d'un scanner (toute la phase d'extraction des lettres du fond a déjà été effectuée. Les lettres se présentent sous la forme d'images binaires, de taille variable. 250 exemples de chaque lettre sont donnés pour les lettres allant de B à K (image B128.bmp pour le 128ième exemple de la lettre B).

Observation
Visualiser quelques images de la base et identifier les problèmes

Prétraitements (pretraitement.m)
Afin de pouvoir mettre en place un algorithme de reconnaissance des formes, il va falloir 'coder' chaque image. On propose d'utiliser un codage rétinien de taille 12x12 qui consiste à prendre la boite englobante +5 pixels de chaque côté. Un redimensionnement sera ensuite réalisé grâce à la commande imresize pour ramener à un codage rétinien 12x12 (144 pixels).



Programmer ce codage en complétant le fichier pretraitement.m

Le code des images sera rangé dans une matrice de la forme suivante :

144 pixels de la forme ex1	Label ex1
144 pixels de la forme ex2	Label ex2
⋮	⋮

Elle aura donc pour dimension 2500x145

Classification avec les kppv
(fonctions classif et classe_maj à compléter)
La base de données va être divisée en 3 :
- une base de référence composée de150 ex. par lettre (variables *base_ref* et *etiq_ref*)
- une base de validation (50 ex. par lettre) pour mettre au point la méthode (variables *base_ val* et *etiq_ val*)
- une base de test (50 ex. par lettre) pour évaluer les performances de l'algorithme (variables *base_ test* et *etiq_ test*).

Afin de réaliser la classification des exemples de la base de référence, écrire une fonction
[label_classif] = test_ppv(base_ref, label_ref, ex, k);

qui renvoie les labels des k ppv de l'exemple *ex* passé en entrée (label_classif aura ainsi pour dimension 1 x k).
Utiliser ensuite la fonction *classe = classe_maj(label_classif)* qui renvoie la classe majoritaire des voisins ou -1 en cas d'égalité.

En utilisant le 1ppv, construire la matrice de confusion et en déduire le taux de reconnaissance sur la base de validation. Programmer pour cela la fonction *[Conf, TauxRejet,TauxReco] = calc_res(liste_class,liste_vrai)* qui renvoie la matrice de confusion, le taux de rejet et le taux de bonne classification (on passe les labels issus de la classification et les vrais labels).

Optimisation de la méthode

1. Faire varier le k des k-ppv et choisir celui qui amène aux meilleurs résultats. Quel est le taux de reconnaissance sur la base de test ?
2. Afin de diminuer le temps de calcul, modéliser chaque classe uniquement par sa moyenne, et réaliser la classification d'un nouvel exemple par le 1ppv (nearest mean). Déterminer la matrice de confusion et le taux de reconnaissance.
3. Modéliser maintenant chaque classe par sa moyenne et sa matrice de covariance (fonction *mean* et *cov*) et utiliser une distance de Mahalanobis. On écrira une fonction *d=Mahal(x,mu,C)* estimant la distance de Mahalanobis entre le point *x* et la gaussienne (on lui passera le point *x* et les paramètres de la gaussienne et elle renverra la distance).
4. Conclure sur la méthode qui vous parait la plus pertinente pour la classification.