GSI-ACAD (2020-2021)
Machine Learning
TP 2 : modèles linéaires pour la classification

Remarques préliminaires :
— Ce TP est noté.
— Vos pouvez travailler en monôme ou en binôme.
— Votre travail est à rendre, au plus tard le 6/11.
— Ce qu'il faut rendre : tous vos codes sources + compte rendu, par mail : `moncef.hidane@insa-cvl.fr`.

1. **Données simulées**

   (a) Commencez par récupérer le fichier `sim_data_set.mat` sur Célène. La commande `load sim_data_set.mat` permet alors de charger une matrice, nommée `R`, dans le workspace. Les deux premières colonnes correspondent respectivement aux abscisses et ordonnées des points d'apprentissage ; la troisième à la classe de chaque point.

   (b) Dans une premier temps, pour vous entraîner, construisez les matrices `X1`, `X2`, `X3`, `X4` qui correspondent aux abscisses et ordonnées de points dans une même classe. Visualisez à l'aide de 4 appels à la fonction `scatter` (il faut un `hold on` après le premier appel à `scatter`). Avez-vous une idée de comment ont été générés ces points ?

   (c) On travaillera dans toute la suite de ce sujet à partir de 2 matrices `X` et `T` construites à partir de `R` : `X` contient les points d'entrée et `T` la classe correspondante. Écrire la fonction

   ```
   ToneofK = classes2oneofK(T)
   ```

   qui permet de passer d'un vecteur de labels dans $\{1, \ldots, K\}$ vers la représentation 1 parmi K vue en cours. La valeur de K sera trouvée dans le corps de la fonction. Suggestion : on pourra utiliser des indices linéaires obtenus à l'aide de la fonction `sub2ind`.

   (d) Écrire la fonction

   ```
   [W,ersub] = fitclog(X,T,optimStruct)
   ```

   qui ajuste les paramètres d'une régression logistique multiclasse. `ersub` désigne l'erreur de resubstitution, c'est-à-dire le proportion de mauvaise classification dans la base d'apprentissage. Vous utiliserez un algorithme de descente de gradient dont les paramètres seront dans la structure `optimStruct`. Rappel : on est incapable ici de donner un borne supérieure sur le pas de descente. Vous pouvez tester les valeurs suivantes : $\eta \in \{0.01, 0.03, 0.1, 0.3, 1\}$. Pensez à visualiser l'évolution des itérations comme au TP précédent. Suggestion : programmez les fonctions softmax et sigmoïde logistique pour pouvoir les appeler dans fitclogreg.

   (e) Écrire la fonction

   ```
   [C] = predclog(X,W)
   ```

   qui calcule les prédictions produites par une régression logistique multiclasse de paramètres W. La variable C ne sera pas codée en schéma 1 parmi K mais aura des valeurs dans $\{1, \ldots, K\}$.

   (f) Vous allez dans cette question tester le résultat pour une classification binaire. En prendra les classes 1 et 2. Voici le début du fichier `demo.m`

```
clear

% Load and prepare dataset
load sim_data_set.mat;
R = R(R(:,3)==1|R(:,3)==2,:); % binary classification
X = R(:,1:2);
T = R(:,3);
ToneofK = classes2oneofK(T);

% Logistic reg fit
[W,~] = fitclog(X,ToneofK);
```

— Comment retrouver l'hyperplan séparateur à partir du codage 1 parmi K ?
— Générez une figure avec les points (fonction `scatter`) et l'hyperplan séparateur. Suggestion : pour le séparateur, on pourra utiliser `fcontour` avec l'attribut `'LevelList'` égal à `0`.

(g) Ajoutez à l'ensemble d'apprentissage 10 points de la classe 1 autour du point de coordonnées $(0,5)$. Ajustez à nouveau les paramètres et visualisez le résultat.

(h) Vous allez à présent tester le résultat pour une classification multiclasse avec $K = 3$ (on prendra les classes 1, 2 et 3).
— Comment obtenir l'équation de l'hyperplan séparant deux classes.
— En utilisant `fcontour` afficher les 3 séparateurs (ainsi que les points d'apprentissage). Êtes-vous satisfaits du résultats ? :)

(i) La stratégie que vous allez adopter pour visualiser les régions de décisions associées à W consiste à construire une grille dans $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$. Vous évaluerez la prédiction en chaque point de la grille et associerez une couleur à chaque classe. La fonction à programmer est la suivante :

```
plotdecr(W,minx,maxx,miny,maxy,nbp,tit)
```

où `nbp` désigne le nombre de points dans chaque dimension et `tit` est un titre optionnel. Suggestions : lors de l'appel, l'intervalle $[x_{\min}, x_{\max}]$ peut être calculé à partir des données d'apprentissage. De même pour $[y_{\min}, y_{\max}]$. Vous pouvez utiliser `linespace` puis `meshgrid` pour construire la grille. Pour la figure, faire une boucle avec un appel à `scatter` à chaque itération (mettez un `hold on` juste après `figure`).

(j) Ajustez les paramètres d'un classification multiclasse avec $K = 4$ et visualisez les régions de décision obtenues.

(k) Essayez d'inclure de nouveaux points appartenant à une classe mais se trouvant loin des points d'apprentissage. Ajustez à nouveau les paramètres et visualisez les régions de décision.

(l) Modifier la fonction `predclog` de sorte à pouvoir prendre des différences d'importances entre les erreurs de classification :

```
[C] = predclog(X,W,L)
```

$L_{i,j}$ désignera, comme en cours, le coût associé à l'association d'un point de la classe j à la classe i.

(m) Visualisez la prédiction précédente en prenant différents L avec différents niveaux d'asymétrie.

(n) Écrire la fonction

```
[W,ersub] = fitcreglog(X,T,alpha,optimStruct)
```

qui correspond à une version régularisée du la régression logistique.

(o) Produire une figure qui compare les 2 approches : régression logistique, régression logistique régularisée.

2. **Données réelles**
   Récupérer et charger le fichier `fisheriris.mat`. L'objectif ici est de prédire l'espèce des Iris. Le choix se fait parmi : `setosa`, `versicolor`, et `virginica`. La première colonne de `meas` contient la longueur des pétales et la deuxième leur largeur. Ce sont ces deux caractéristiques que vous utiliserez pour vos prédictions. Testez les algorithmes que vous avez développés en vous inspirant du protocole du TP1. Le critère d'évaluation sera le taux de bonne classification (moyen sur 10 expériences).