TP : Discrimination linéaire

Rémi Flamary

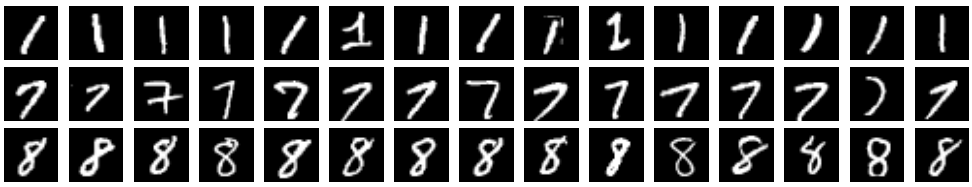
Lors du TP vous aurez besoin des bibliothèque Python `numpy pylab` et `scipy`. Il vous est conseillé de les importer dès le début avec le code suivant :

```
import numpy as np
import pylab as pl
import scipy as sp
```

Vous pourrez ensuite accéder aux fonctions de ces modules à l'aide du point (par exemple `np.zeros(10)` pour la fonction `zeros` de numpy). Dans la suite du TP, pour chaque question la liste des fonctions `numpy/pylab` nécessaires est donné entre parenthèses.

1 Chargement des données et pré-traitement

- Télécharger le fichier "digits.npz".
- Charger ce fichier en mémoire en utilisant la fonction `np.load`. Le fichier contient les matrices suivantes :
  - **x** et **xt** : matrices de données contenant respectivement $n = 3000$ et $nt = 1500$ exemples d'images manuscrites. Chaque ligne de ces matrices correspond à une image stockée sous la forme d'un vecteur transposé.
    Les images sont de la forme suivante :

  - **y** et **yt** : étiquettes des images décrites dans les matrices précédentes. Ce sont des vecteurs qui contiennent la classe $(1, 7, 8)$ de chaque image de **x** et **xt**.
- Utiliser la fonction `np.reshape` pour extraire quelques images de taille 28×28 pour chaque classe. Les visualiser avec la fonction `pl.imshow`.
- Normaliser les données et utiliser les données normalisées dans la suite du TP (`np.mean, np.std`).

2 Discrimination binaire par moindre carré

- On veut créer un problème de classification binaire à partir des trois classes. Vous pourrez par exemple choisir de classifier la classe 8 contre 1 et 7. Stocker les étiquettes binaires $(-1, 1)$ dans les vecteurs `yb` et `ytb` (opérateur `==`) .
- Estimer un classifieur à l'aide de la régression ridge (`np.dot,np.eye,np.linalg.solve`).
- Prédire la classe binaire à partir de la prédiction continue (`np.sign`).
- Calculer le taux de bonne reconnaissance (`np.mean,==`).

— Quel effet a la régularisation sur les performances (apprentissage et test) ?
— Visualiser quelques exemples mal classés sous la forme d'image, conclusions (`np.reshape,pl.imshow`).
— Visualiser le classifieur **w** sous la forme d'une image, interpréter l'image (`np.reshape,pl.imshow`).

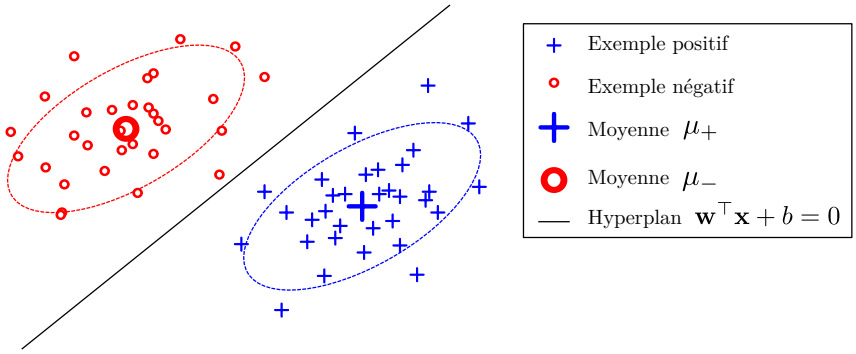
3 Régression logistique

— Coder l'algorithme de descente de gradient pour la régression logistique.
— Calculer le coût le long des itérations et le tracer pour vérifier la décroissance (`pl.plot`).
— Mettre à jour le code pour résoudre la descente de gradient pour la régression logistique **régularisée**.
— Rregarder l'impact des différents paramètre dans la descente de gradient (pas, régularisation, nombre d'itérations)
— Calculer le taux de bonne reconnaissance (`np.mean,==`).
— Quel effet a la régularisation sur les performances (apprentissage et test) ?
— Visualiser quelques exemples mal classés sous la forme d'image, conclusions (`np.reshape,pl.imshow`).
— Visualiser le classifieur **w** sous la forme d'une image, interpréter l'image (`np.reshape,pl.imshow`).

4 Discrimination multiclasse

— Pour effectuer une discrimination multiclasse, une approche commune est de faire ce qui s'appelle du « un contre tous ».
— Pour cela on estime 1 classifieur binaire par classe en prenant tous les exemples des autres classes comme étant négatifs (voir section précédente).
— Les scores de prédiction pour chaque classe sont calculés pour chaque exemple (`np.dot`).
— La prédiction finale consiste à choisir la classe qui a le score le plus important (`np.argmax`).
— Utiliser la méthode « un contre tous » et évaluer les performances sur les données d'apprentissage et de test.
— Discuter les résultats pour la régression ridge et logistique.

5 Bonus : LDA binaire



L'Analyse Linéaire Discriminante (ou Linear Discriminant Analysis en anglais) est une méthode simple de discrimination basée sur une modélisation probabiliste des données. On veut classifier des exemples (vecteurs) $\mathbf{x} \in \mathbb{R}^d$ qui peuvent appartenir à la classe positive $+$ ou à la classe négative $-$ (discrimination binaire). On suppose pour cela que les exemples sont des réalisations de lois normales multidimensionnelles $\mathcal{N}(\mu_+, \Sigma)$ pour la classe positive de probabilité p_+ et $\mathcal{N}(\mu_-, \Sigma)$ pour la classe négative de probabilité p_- telle que $p_+ + p_- = 1$.

En calculant la vraisemblance pour un exemple $\mathbf{x}$ pour chaque classe $\{-1, 1\}$ on se rend compte que la prédiction de la classe peut être faite en prenant le signe d'une fonction linéaire de la forme

$$f(\mathbf{x}) = \mathbf{x}^T \mathbf{w} + b \tag{1}$$

avec $\mathbf{w} \in \mathbb{R}^d$ et b les coefficients du classifieur de valeur

$$\mathbf{w} = \Sigma^{-1}(\mu_+ - \mu_-) \tag{2}$$

$$b = -\mathbf{w}^t(\mu_+ + \mu_-)/2 + \log(p_+) - \log(p_-) \tag{3}$$

Une variante de la LDA visant à promouvoir une meilleure robustesse consiste à remplacer l'inverse de la matrice de covariance Σ^{-1} par l'inverse $(\Sigma + \lambda \mathbf{I})^{-1}$ où λ est un paramètre de régularisation qui assure que la matrice est inversible et $\mathbf{I}$ est la matrice identité. Cette méthode appelée LDA régularisée est préférée lorsque le nombre d'exemples d'apprentissage est limité ou lorsque le nombre de variables est important ($d > n$).

— Estimer les probabilités p_+ et p_- à partir des données d'apprentissage.
— Estimer les moyennes μ_- et μ_+ à partir des données d'apprentissage.
— Centrer les exemples de chaque classe et estimer la matrice de covariance Σ (fonction).
— En déduire les paramètres du classifieur $\mathbf{w}$ et b. Que se passe-t-il si on ne régularise pas ($\lambda = 0$) ?
— Comparer le classifier LDA à celui des moindres carré et de la régression logistique