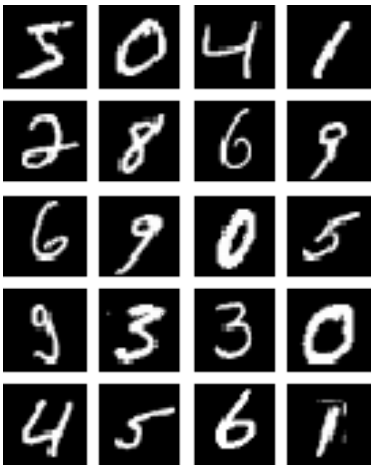
Classification binaire d'image: le perceptron sur la base MNIST

Alexandre Allauzen

2014



L'objectif de ce TP est d'expérimenter l'algorithme d'apprentissage du perceptron pour la classification d'image. La tâche considérée est la reconnaissance de chiffres manuscrits. Voir cette page pour les informations pratiques sur la base d'image et le code python nécessaire:

`http://perso.limsi.fr/Individu/allauzen/webpages/pmwiki.php?n=Cours.Main#toc7`

Cette base d'image contient 50 000 images d'apprentissage et 10 000 pour l'évaluation. Chaque image représente un chiffre et le but est de prédire quelle chiffre elle représente (de 0 à 9). C'est une tâche *multiclasse* que nous allons dans un premier temps considérée comme une tâche de classification binaire. Pour cela, prévoir une variable qui correspond au chiffre à reconnaître. Le rôle du classifieur est alors de dire si une image correspond ou non à ce chiffre.

1 Analyse du corpus

1. Regarder comment sont stockées les informations (les images d'une part et les étiquettes d'autre part.

2. Quelle est la dimension d'une image ?

3. Vérifiez le nombre d'image disponible pour l'apprentissage et le test.

4. Quelles sont les valeurs des pixels ?

2 Perceptron: premier pas

Pour mettre en place le perceptron, nous allons dans un premier temps utiliser les 100 premières images. Ce nombre 100 doit être une variable. Pour cela choisir un nombre de votre choix, par exemple 5. Le classifieur binaire doit dans ce cas répondre oui ou non selon que l'image représente un 5 ou non.

5. Mettre en place une manière de parcourir les 100 premières images d'apprentissage dans un ordre aléatoire.

6. Coder une fonction python qui effectue un pas d'apprentissage du perceptron: prendre une image, évaluer la réponse du classifieur, puis effectuer une mise à jour des paramètres si nécessaire. Cette fonction doit renvoyer un booléen indiquant s'il y a eu mise à jour ou non.

7. Coder une fonction python qui effectue une époque d'apprentissage et qui réutilise la précédente. Les données d'apprentissage sont parcourues dans un ordre aléatoire. Cette fonction doit également compter le nombre de mise à jour effectuée lors de l'époque.

8. Coder une fonction python, qui prend en paramètre un jeu de test et qui compte le nombre d'erreur effectuée.

3 Perceptron: assemblage et courbe d'apprentissage

Nous avons maintenant tout ce qui faut pour faire nos premières expériences.

9. Tracer la courbe d'apprentissage pour 100, 1000, 5000 et 10000 image d'apprentissage. La courbe d'apprentissage prend en abscisse le nombre d'époque et en ordonnée le nombre de mise à jour.

10. Tracer la même courbe sur les données de test: après chaque époque évaluer le classifieur à l'état courant sur les données de test. On a donc en abscisse le nombre d'époque et en ordonnée le nombre d'erreurs effectuées.

11. Comparer les résultats selon les chiffres traités: par exemple en prenant 1 et 7 qui donnent lieu à des images plus difficile à reconnaitre.