Reconnaissance de caractères

CMI SIC - MASTER SIC - EISTI IE - IEMRIR SoC
Systèmes Intelligents Communiquants
Pierre Andry
Université de Cergy Pontoise
pierre.andry@ensea.fr

1 Séparation de deux classes, règle du perceptron simple

Nous allons programmer en C un réseau de neurones qui fait la différence entre un motif du chiffre '0' et un motif du chiffre '1'. Ces motifs sont représentées par des grilles de 6x8 "pixels", représentés par des caractères (nous restons binaire : le caractère '.' représente le fond, le caratère '*' une zone écrite):

Figure 1: contenu des fichier zero.txt (à gauche) et un.txt (à droite)

chaque motif peut etre écrit sous la forme d'une matrice ou d'un vecteur, selon votre choix.
chaque motif est placée dans un fichier.

important : à chaque motif k est associée à une sortie désirée Y_d^k : $Y_d^0 = 0$ et $Y_d^1 = 1$
Ces valeurs sont fixées arbitrairement, mais elles doivent ensuite rester fixe tout au long du TP. Il est conseillé d'ajouter la valeur de la classe Y_d^k dans chaque fichier, en dernière ligne.

au préalable :

- Créer les deux fichier `zero.txt` et `un.txt`. Pensez à ajouter en fin de fichier la valeur de Y_d^k.
- Creer le tableau des entrées X.
- Créer le tableau des poids W, initialisez aléatoirement et normalisez chaque poids W_{ij}
- Programmer une fonction qui tire aléatoirement une des deux classes et lit le fichier correspondant (`zero.txt` *vs* `un.txt`)
- Fixer les valeurs importantes : $\theta = 0,5$ et $\epsilon = 0,01$

Algorithme

1. Choisir une classe au hasard, c'est à dire un des deux fichier (`zero.txt` *vs* `un.txt`). Mémoriser le Y_d^k correspondant.
2. Propager sur la rétine : $X_j = 0$ si Pix$_j$ ='.' ou bien $X_j = 1$ si Pix$_j$ ='*'
3. Propager pour calculer le potentiel du neurone de sortie : $pot_i = \sum_j W_{ij} X_j - \theta$
4. Calculer la réponse du neurone de sortie : $X_i = f(pot_i)$ avec $f(x) = H_0$
5. Calculer l'erreur : $\delta_i^k = Y_d^k - X_i$, avec k l'indice de la classe qui a été tirée au hasard en entrée à l'étape 1.
6. Apprendre : $W_{ij}(t+1) = W_{ij}(t) + \epsilon \cdot err_i \cdot X_j$
7. Calculer l'erreur totale $Err = |\delta^0| + |\delta^1|$. si $Err > 0$ aller à l'étape 1, sinon fin.

Variante

Une variante efficace consiste à adapter (modifier) aussi θ tout au long de l'apprentissage. Il s'agit du modèle de *neurones à biais*. Dans ce cas on considère une $j+1$ème entrée X_{j+1} reliée à la sortie X_i par un poids supplémentaire W_{ij+1}. W_{ij+1} est initialisé à la valeur $-\theta$ (cette valeur chagera au cours de l'apprentissage), tandis que $X_{j+1} = 1$ en permanence. Dans ce cas, seule l'étape 3 de l'algorithme change et devient : $pot_i = \sum_{j+1} W_{ij} X_j$.

Généralisation

La qualité de notre apprentissage sera révélé par les courbes de génralisation. C'est la faculté du réseau à généraliser la reconnaissance d'une entrée bruitée.

- Créez une fonction qui, en fonction d'un pourcentage de bruit souhaité, inverse la valeur des pixels d'une entrée. Par exemple, un bruit de 50% correspondra à l'inversion de la moitiè des pixels d'un motif, la position des pixels etant tirée aléatoirement.
- Créez une fonction qui compte le nombre de mauvaises réponses (nombre d'erreurs du réseau) pour 50 motifs ayant le même niveau de bruit.
- Constrisez la courbe de généralisation du motif 0 : c'est le taux d'erreur pour des entrées bruitées allant de 0% à 100
- Constrisez et superposez la courbe de généralisation du motif 1 : c'est le taux d'erreur pour des entrées bruitées allant de 0% à 100

2 Séparation de deux classes, règle de Widrow-Hoff

Les résultats de la généralisation (section précédente) mettent en evidence la fin prématurée de l'apprentissage, et l'imposibilité pour la règle du perceptron simple à trouver une position optimale pour l'hyperplan qui sépare les deux classes : les courbes de génrélisation sont fortement dissymétriques, indiquant que l'hyperplan séparateur est bien plus proche du centre d'une classe que de l'autre.
La règle de Widrow Hoff permet d'effectuer un placement optimal de l'hyperplan, en prenant en compte la valeur de l'erreur non binaire. Danc ce cas, on considère, lors de l'apprentissage, la réponse analogique (avant binarisation) du neurone : pot_i
Ainsi, le nouveau calcul de l'erreur (point 5 de l'algorithme précédent) devient :

5. calculez l'erreur : $err_i^k = Y_d^k - pot_i$, avec k l'indice de la classe qui a été tirée au hasard en entrée à l'étape 1.

reprennez l'apprentissage avec le nouveau calcul d'erreur. Re-testez la généralisation et vérifiez que les taux d'erreurs sont symétriques entre les deux classes. Votre hyper plan sera équidistant des deux centres de classes.

3 Séparation de 10 classes, règle de Widrow-Hoff

Appliquez maintenant votre perceptron à la reconnaissance des 10 chiffres :

- Créez les dix fichier, de `zero.txt` à `neuf.txt`.
- Créez un réseau avec 6x8 unités d'entrée (inchangé par rapport aux sections précédentes), et 10 unités de sorties. On considère une sortie pour chaque chiffre reconnu. Par exemple $Y_d^0 = [1,0,0,0,0,0,0,0,0,0]$ (le premier neurone code pour la reconaissance d'un zero); $Y_d^1 = [0,1,0,0,0,0,0,0,0,0]$ (le second neurone reconnait un un);etc. jusque $Y_d^9 = [0,0,0,0,0,0,0,0,0,1]$ (le dernier neurone reconnait un 9).
- Créez le (ou les) tableau(x) de poids W_{ij}, chaque neurone X_i de sortie etant relié à tous les neurones X_j d'entrée. Initialisez aléatoirement et normalisez les poids W_{ij}
- Programmez une fonction qui tire aléatoirement une des dix classes, c'est a dire un des deux fichier (`zero.txt`, `un.txt`, ...,`neuf.txt`.)
- Fixez les valeurs importantes : $\theta = 0,5$ et $\epsilon = 0,01$

4 Séparation de 10 classes, perceptron multi-couches et rétropropagation du gradient appliqué a la base MNIST

Nous allons maintenant traiter la reconnaissance des chiffres manuscrits d'une manière plus réaliste, en utilisant la base de données MNIST.

- Récupérez la base d'entrainement:
  `ttrain-images-idx3-ubyte`,
  `train-labels-idx1-ubyte`
- Récupérez la base de test :
  `t10k-images-idx3-ubyte`,
  `t10k-labels-idx1-ubyte`
- Lisez les spécifications des données (*http://yann.lecun.com/exdb/mnist/*)
- Créez une fonction qui détermine aléatoirement un numéro de motif (entre 0 et 59000) et lit le motif et le label correspondant et affiche. le label est utilisé pour fixer la classe Y_d^k.(comme dans la section suivante).
- Creez un réseau avec une couche d'entrée (28x28 neurones), une couche cachée (100 neurones) et une couche de sortie (10 neurones).
- Creez les tableaux de poids. Initialisez les poids aléatoirement tout en normalisant.

Algorithme

1. Choisir une classe au hasard, c'est à dire un motif parmis les 60000 de la base d'entrainement. Charger le motif et mémoriser le Y_d^k correspondant au label (label de la base d'entrainement).
2. Propager sur la rétine : $X_j = \text{Pix}_j/255$.
3. Calculer la sortie des neurones de la couche cachée: $X_h = f(pot_h)$
   $pot_h = \sum_j W_{hj} X_j$ avec $f(x) = \frac{1}{1+e^{-x}}$, dérivable et continue.
4. Calculer la sortie des neurones de la couche de sortie: $X_i = f(pot_i)$
   $pot_i = \sum_h W_{ih} X_h$ avec $f(x) = \frac{1}{1+e^{-x}}$, dérivable et continue.
5. Pour chaque neurone de la couche de sortie : $\delta_i = f'(pot_i) \cdot Yd_i^k - X_i$ remarque : $f(x)$ est dérivable et continue, on a $f'(x) = f(x) \cdot (1 - f(x))$
6. Pour chaque neurone de la couche cachée : $\delta_h = f'(pot_h) \cdot \sum_i \delta_i \cdot W_{hi}$
   remarque : $f(x)$ est dérivable et continue, on a $f'(x) = f(x) \cdot (1 - f(x))$
7. Apprendre : $W_{ij}(t+1) = W_{ij}(t) + \epsilon \cdot \delta_i \cdot X_j$
8. Calculer le pourcentage d'erreur sur p (par exemple $p = 100$) motifs pris au hasard dans la base d'entrainement présentés $Err = \frac{\sum_p \sum_i |\delta_i|}{p}$. si $Err > seuil$ aller à l'étape 1, sinon fin (on doit normalement pouvoir atteindre un seuil de 96% de bonnes réponses sur la base de test).
9. Testez la performance du réseau sur la base de test