TP - Restricted Boltzmann Machines

Yohan Petetin (Télécom SudParis - IP Paris)

08/01/2021

1 Données

Récupérer les images de la base de données Binary AlphaDigits à partir de l'adresse suivante : `http://www.cs.nyu.edu/~roweis/data.html`. Il s'agit d'un fichier .mat, lisible en Matlab ou Python.

2 Construction d'un RBM et test sur Binary AlphaDigits

Un RBM pourra être représenté par un objet contenant un champ W (matrice de poids reliant les variables visibles aux variables cachées), un champ a (biais des unités d'entrée) et un champ b (biais des unités de sortie).

On complètera au fur et à mesure un script permettant d'apprendre les caractères de la base Binary AlphaDigits de votre choix via un RBM et de générer des caractères similaires à ceux appris. La construction de ce script nécessite les fonctions suivantes:

Écrire une fonction `lire_alpha_digit` permettant de récupérer les données sous forme matricielle (en ligne les données, en colonne les pixels) et qui prend en argument les caractères que l'on souhaite "apprendre" (ex: mettre en argument le vecteur $[10, 11, 12]$ pour extraire les A, B et C).

Écrire une fonction `init_RBM` permettant de construire et d'initialiser les poids et les biais d'un RBM. Cette fonction retournera une structure RBM avec des poids et biais initialisés. On initialisera les biais à 0 tandis que les poids seront initialisés aléatoirement suivant une loi normale centrée, de variance égale à 0.01.

Écrire une fonction `entree_sortie_RBM` qui prend en argument une structure RBM et des données d'entrée et qui retourne la valeur des unités de sortie calculées à partir de la fonction sigmoïde.

Écrire une fonction `sortie_entree_RBM` qui prend en argument un RBM, des données de sortie et qui retourne la valeur des unités d'entrée à partir de la fonction sigmoïde.

Écrire une fonction `train_RBM` permettant d'apprendre de manière non supervisée un RBM par l'algorithme Contrastive-Divergence-1. Cette fonction retournera un structure RBM et prendra en argument une structure RBM, le nombre d'itérations de la descente de gradient, le learning rate, la taille du mini-batch, des données d'entrées... À la fin de chaque itération du gradient, on affichera l'erreur quadratique entre les données d'entrées et les données reconstruites à partir de l'unité cachée afin de mesurer le pouvoir de reconstruction du RBM.

Écrire une fonction `generer_image_RBM` permettant de générer des échantillons suivant un RBM. Cette fonction retournera et affichera les images générées et prendra en argument une structure de type RBM, le nombre d'itérations à utiliser dans l'échantillonneur de Gibbs et le nombre d'images à générer.

3 Étude

Analyser les images générées et l'erreur de reconstruction en fonction des hyperparamètres (nombre d'unités cachées,...) et des caractères considérés dans la base de données. Essayer le modèle sur des bases de données alternatives (MNIST,...), en pensant à binariser les données (0 et 1). Comparer les images générées avec celles produites par des architectures que vous auriez éventuellement déjà utilisés par le passé (ex : GAN, VAE,...).