IFT 725 : Devoir 3
Travail individuel

Remise : 11 novembre 2013, 9h00 (**au plus tard**)

Dans ce devoir, vous devez implémenter en Python une machine de Boltzmann restreinte et un autoencodeur (avec option de débruitage), utilisé pour le pré-entraînement d'un réseau de neurones.

L'implémentation de ces modèles doit être contenue dans les classes `RBM` et `Autoencoder`, qui héritent de la classe `Learner` de la librairie MLPython. La définition de ces classes doivent être mises dans des fichiers nommés `rbm.py` et `autoencoder.py` respectivement. Ces classes doivent toutes les deux supporter l'utilisation des hyper-paramètres suivants :

- `lr` : le taux d'apprentissage de la descente de gradient stochastique (`float`)
- `hidden_size` : la taille de la couche cachée (`int`)
- `seed` : la souche du générateur de nombres aléatoires utilisé par le modèle (`int`)
- `n_epochs` : nombre d'itérations d'entraînement à effectuer (`int`)

La classe `RBM` doit également supporter l'hyper-paramètre suivant :

- `CDk` : nombre d'itérations d'échantillonnage de Gibbs utilisé par la *contrastive divergence* (`int`)

La classe `Autoencoder` doit quand à elle supporter l'hyper-paramètre suivant :

- `noise_prob` : probabilité de bruitage, i.e. de fixer une entrée à 0 (`float`)

Un squelette des classes `RBM` et `Autoencoder` sont fournies dans les fichiers `rbm.py` et `autoencoder.py` disponibles sur le site web du cours. Vous n'avez qu'à implémenter la méthode `train` dans ces squelettes. **Il est important d'utiliser la librairie Numpy dans votre implémentation, afin qu'elle soit efficace**.

Afin de vérifier si vos implémentations fonctionnent bien, les scripts `run_show_filters_rbm.py` et `run_show_filters_autoencoder.py` peuvent être utilisés afin de comparer les filtres (i.e. les connections des neurones cachés) appris avec ceux obtenus par une implémentation correcte (voir les fichiers `rbm_filters.pdf` et `autoencoder_filters.pdf`, disponibles sur le site web du cours).

Les scripts `run_stacked_rbms_nnet.py` et `run_stacked_autoencoders_nnet.py` sont également mis à votre disposition afin d'utiliser vos implémentations de la machine de Boltzmann restreinte et de l'autoencodeur (respectivement) pour pré-entraîner un réseau de neurones de classification, sur le jeu de données *OCR Letters* (le même que pour le devoir 1).

Le script `run_stacked_rbms_nnet.py` nécessite comme arguments les hyper-paramètres suivants :

```
Usage: python run_stacked_rbms_nnet.py lr dc sizes pretrain_lr pretrain_n_epochs pretrain_CDk seed

Ex.: python run_stacked_rbms_nnet.py 0.01 0 [200,100] 0.01 10 1 1234
```

Le script `run_stacked_autoencoders_nnet.py` nécessite les hyper-paramètres suivants :

```
Usage: python run_stacked_autoencoders_nnet.py lr dc sizes pretrain_lr pretrain_n_epochs
       pretrain_noise_prob seed

Ex.: python run_stacked_autoencoders_nnet.py 0.01 0 [200,100] 0.01 10 0.1 1234
```

Les deux scripts donneront les erreurs moyennes d'entraînement et de validation après chaque époque. De plus, ils enregistreront dans des fichiers nommés `results_stacked_rbms_nnet_ocr_letters.txt` et `results_stacked_autoencoders_nnet_ocr_letters.txt` (respectivement) le résultat final de l'apprentissage (les erreurs moyennes d'entraînement, de validation et de test). Chaque nouvelle exécution des scripts ajoutera une ligne à ces fichiers de résultats. Le nombre d'itérations de pré-entraînement (`pretrain_n_epochs`) doit être spécifié. Le raffinement du réseau de neurones utilise l'arrêt prématuré basé sur l'erreur de classification sur l'ensemble de validation pour déterminer quand arrêter (horizon de 5 époques).

Une fois votre implémentation complète, vous devez générer des résultats sur le jeu de données *OCR Letters* pour la classification séquentielle, permettant d'évaluer la performance de votre implémentation. Spécifiquement, on vous demande :

- de rapporter les taux d'erreur de classification sur l'ensemble d'entraînement et de validation pour **au moins 15 choix différents des hyper-paramètres** (ne faites pas des expériences avec seulement la RBM ou l'autoencodeur, essayez les deux au moins une fois) ;
- d'illustrer **la progression du taux d'erreur de classification en entraînement et en validation**, pour une configuration des hyper-paramètres de votre choix ;
- d'illustrer également **la progression de la log-vraisemblance négative moyenne en entraînement et en validation**, pour une configuration des hyper-paramètres de votre choix ;
- de rapporter le taux d'erreur de test **seulement pour le choix d'hyper-paramètres ayant la meilleure performance de validation** ;
- de spécifier également un **intervalle de confiance à 95%** de l'erreur de test.

Ces résultats doivent être rapportés dans un rapport suivant le format d'un article scientifique de la conférence NIPS [1]. Le document doit être en format PDF et doit avoir été généré à l'aide du langage LaTeX [2].

Le rapport doit contenir les sections suivantes :

- **Introduction** : donne un résumé de l'objectif du devoir et du contenu du rapport ($^1/_2$ page).
- **Description de l'approche** : décrit mathématiquement la méthode (au plus 4 pages), incluant
  - une description de la machine de Boltzmann restreinte :
    - quelle est la fonction d'énergie et la définition de $p(\mathbf{x}, \mathbf{h})$?
    - quelle forme prend $p(\mathbf{x})$?
    - quelles forment prennent les distributions conditionnelles $p(\mathbf{h}|\mathbf{x})$ et $p(\mathbf{x}|\mathbf{h})$?
    - quel est l'objectif qu'on souhaite optimisé pour l'entraînement et comment la *contrastive divergence* approxime les gradients de cet objectif ?
    - pourquoi est-ce qu'on ne peut pas calculer exactement les gradients de l'objectif qu'on aimerait optimiser ?
  - une description de l'autoencodeur (avec/sans débruitage)
    - quelle forme prend la propagation avant ?
    - quelle est l'objectif optimisé lors de l'entraînement ?
    - quelle forme prend la rétropropagation des gradients ?
    - quelle est la différence entre un autoencodeur de base et un autoencodeur de débruitage ?
    - quelle est l'avantage de l'autoencodeur de débruitage sur l'autoencodeur de base ?
  - une description de la procédure de pré-entraînement d'un réseau de neurones à l'aide d'une machine de Boltzmann restreinte ou d'un autoencodeur ;
  - une explication de la raison pour laquelle le pré-entraînement peut améliorer la performance de généralisation d'un réseau de neurones.

1. Voir `http://nips.cc/PaperInformation/StyleFiles` pour obtenir les fichiers permettant de suivre ce format. Vous devrez ajouter la ligne `\nipsfinalcopy` après la commande `\author` pour que l'information soit inscrite.
2. Voir `http://www.maths.tcd.ie/~dwilkins/LaTeXPrimer/GSWLaTeX.pdf` pour une introduction et `http://www.andy-roberts.net/writing/latex/pdfs` pour savoir comment générer un document PDF à partir d'un fichier LaTeX

- **Expériences** : présentation des résultats de vos expériences, c'est-à-dire :
  - un **tableau** avec les erreurs d'entraînement et de validation pour au moins 15 choix d'hyper-paramètres ;
  - un **graphique** illustrant la progression du taux d'erreur de classification en entraînement et en validation, pour une configuration des hyper-paramètres de votre choix
  - un **graphique** illustrant la progression de la log-vraisemblance négative moyenne en entraînement et en validation, pour une configuration des hyper-paramètres de votre choix
  - le résultat sur l'ensemble de test **seulement** pour le choix d'hyper-paramètres ayant la meilleure performance sur l'ensemble de validation ;
  - l'intervalle de confiance de 95% pour le taux d'erreur de classification de test.

La répartition des points suivra la barème suivant :

- **10 points** pour la justesse de l'implémentation, ainsi que sa qualité (e.g. bonne utilisation de Numpy) ;
- **4 points** pour la qualité et justesse des sections **Introduction** et **Description de l'approche** du rapport ;
- **4 points** pour l'inclusion de tous les résultats attendus dans la section **Expériences** et pour leur validité.

Le rapport peut être remis en anglais ou en français. Tout le travail de ce devoir doit être fait individuellement. Aucun code, texte du rapport ou résultats ne doivent être partagés ou transmis entre les étudiants. Par contre, les étudiants sont encouragés à discuter certains éléments de solution du devoir de vive voix.

Veuillez soumettre votre code et votre rapport à l'aide de l'outil **turnin** :

```
turnin -c ift725 -p devoir_3 rbm.py autoencoder.py rapport.pdf
```

Bon travail !