IFT 725 : Devoir 1
Travail individuel

Remise : 21 septembre 2012, 17h00 (**au plus tard**)

Dans ce devoir, vous devez implémenter en Python un réseau de neurones multi-couche pour la classification multi-classe.

L'implémentation du réseau de neurones doit être contenue dans une classe `NeuralNetwork` qui hérite de la classe `Learner` de la librairie MLPython. La définition de la classe doit être mise dans un fichier nommé `nnet.py`. Votre implémentation doit supporter l'utilisation des hyper-paramètres suivants :

- `lr` : le taux d'apprentissage de la descente de gradient stochastique (`float`)
- `dc` : la constante de décroissement du taux d'apprentissage
- `sizes` : la liste du nombre de neurones de chaque couche cachée, de la première à la dernière (`list` de `int`)
- `L2` : le taux de régularisation L2 sur les matrices de poids du réseau (`float`)
- `L1` : le taux de régularisation L1 sur les matrices de poids du réseau (`float`)
- `seed` : la souche du générateur de nombre aléatoire pour l'initialisation des paramètres (`int`)
- `tanh` : booléen indiquant si la tangente hyperbolique doit être utilisée comme fonction d'activation des neurones cachés (`True`) plutôt que la fonction sigmoïde (`False`)
- `n_epochs` : nombre d'itérations d'entraînement à effectuer (`int`)

Un squelette de la classe `NeuralNetwork` est fourni dans le fichier `nnet.py` disponible sur le site web du cours. Le squelette spécifie également la signature de toutes les méthodes que vous devez implémenter. **Il est important d'utiliser la librairie Numpy dans votre implémentation, afin qu'elle soit efficace**.

La méthode `verify_gradients` est déjà implémentée. Elle compare le calcul des gradients par la rétropropagation avec une approximation par différence finie. Il est important de l'utiliser afin de vous assurer que votre implémentation de la propagation avant et la rétropropagation sont correctes. Le script `run_verify_gradients.py` vous est fourni afin de procéder à cette vérification, pour différentes configurations des hyper-paramètres. Les différences rapportées entre votre implémentation et l'approximation par différences finies devraient être plus petite que 10^{-10}.

De plus, un script `run_nnet.py` est également mis à votre disposition afin d'entraîner votre réseau de neurones selon la méthode de l'arrêt prématuré (*early stopping*), sur le jeu de données *OCR Letters*. Le script nécessite comme arguments la valeur de chaque hyper-paramètre, comme suit :

```
Usage: python run_nnet.py lr dc sizes L2 L1 seed tanh

Ex.: python run_nnet.py 0.1 0 [20,10] 0 0 1234 False
```

Le script donnera les erreurs moyennes d'entraînement et de validation après chaque époque. De plus, il enregistrera dans un fichier nommé `results_nnet_ocr_letters.txt` le résultat final de l'apprentissage, c'est-à-dire les erreurs moyennes d'entraînement, de validation et de test, et ce pour le nombre d'époques minimisant l'erreur de validation. Les erreurs qui doivent être calculées par votre implémentation sont l'erreur de classification et la log-vraisemblance négative régularisée. Chaque nouvelle exécution du script ajoutera une ligne au fichier `results_nnet_ocr_letters.txt` avec le résultat associé. L'arrêt prématuré utilise l'erreur

de classification sur l'ensemble de validation pour déterminer quand arrêter et utilise un horizon (*"look ahead"*) de 5 époques.

Pour que le script puisse fonctionner correctement, vous devrez préalablement télécharger le jeu de données *OCR Letters* à l'aide de la procédure automatique de MLPython. Pour cela, vous devez définir la variable d'environnement `MLPYTHON_DATASET_REPO` et utiliser le script `mlpython_helper` pour procéder au téléchargement, comme suit :

```
mlpython_helper -datasets -download ocr_letters
```

Vous trouverez plus d'information dans le tutoriel de MLPython, qui traite entre autre de la manipulation des jeux de données (classe `MLProblem`) [1].

Une fois votre implémentation complète, vous devez générer des résultats sur le jeu de données *OCR Letters* 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** ;
- 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 régularisée 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 [2]. Le document doit être en format PDF et doit avoir été généré à l'aide du langage LaTeX [3].

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 propagation avant ;
  - une description de la rétropogation des gradients ;
  - une description de l'objectif optimisé par l'apprentissage ;
  - une description de l'algorithme de la descente de gradient stochastique dans le contexte d'un réseau de neurones ;
  - une description de tous les hyper-paramètres.
- **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 régularisée 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.

1. `http://www.dmi.usherb.ca/~larocheh/mlpython/tutorial.html#processed-datasets-mlproblems`
2. Voir `http://nips.cc/PaperInformation/StyleFiles` pour obtenir les fichiers permettant de suivre ce format
3. 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

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 implémentation et votre rapport à l'aide de l'outil **turnin** :

```
turnin -c ift725 -p devoir_1 nnet.py rapport.pdf
```

Bon travail !