TP3: Diagnostic du sur-apprentissage

Gaétan Marceau Caron

18 décembre 2015

1 Introduction

Lors du tp2, vous avez implémenté l'algorithme de backpropagation qui permet de calculer efficacement le gradient de la fonction de coût par rapport aux paramètres du modèle. Ainsi, vous pouvez maintenant optimiser ces paramètres à l'aide de l'algorithme de descente de gradient. Ce problème d'optimisation s'exprime en général comme la minimisation d'une fonction de coût :

$$\min_{\theta} \quad \mathbb{E}_{d\sim\mathcal{D}}\left[l_d(\theta)\right] \tag{1}$$

où $\mathcal{D}$ est une distribution de probabilités sur les données, $\mathbb{E}_{d\sim\mathcal{D}}$ est l'espérance associée à cette distribution et $l_d(\theta)$ est une fonction qui évalue l'erreur du modèle paramétré par θ pour l'exemple d. Autrement dit, nous voulons minimiser l'erreur espérée sur les exemples générés par $\mathcal{D}$ (modèle génératif). À titre d'exemple, dans le TP2, nous avions $d = (x, t)$ où x était une image et t une étiquette, et la fonction de coût était l'opposée de la log-vraisemblance $l_d(\theta) = -\ln p_\theta(y = t|x)$ où $p_\theta(y|x)$ est la distribution modélisée par le réseau de neurones associé à θ sachant x.

Cependant, nous ne connaissons pas explicitement $\mathcal{D}$ et travaillons plutôt avec un nombre fini d'exemples $\mathcal{D}' = \{d_1, \ldots, d_N\}$ que nous appelons la base d'apprentissage. Nous utilisons donc cette base d'apprentissage afin d'obtenir une approximation de l'équation (1) :

$$\min_{\theta} \quad \frac{1}{N}\sum_i \left[l_{d_i}(\theta)\right] \text{ où } d_i \sim \mathcal{D} \tag{2}$$

Malheureusement, pour n'importe quel $\mathcal{D}'$ tiré aléatoirement, cette approximation sera biaisée par le nombre fini d'exemples N et nous observerons le phénomène de *sur-apprentissage*. Plus le modèle est complexe et N petit, plus l'approximation sera biaisée par le *dilemne bias-variance* (cf. Bishop, 2006, p. 46 [1]).

1. `http://tinyurl.com/m62yc3c`

2 Description

Le but du TP3 est de tester différentes configurations de réseaux de neurones, à l'aide du code développé lors du TP2, afin de minimiser l'erreur de validation sur le base d'apprentissage MNIST. La première étape consiste à comprendre les sorties du programme `miniNN.py` et de les visualiser avec `gnuplot nll.gnu` et `gnuplot class_err.gnu` (la sortie du programme `python miniNN.py` doit être stockée dans le fichier `nn.dat`). Si l'erreur d'entraînement stagne à moins de 50% et que le test des différences finies n'échoue pas, il vous faudra revérifier l'initialisation, le step-size de la descente de gradient et la mise à jour des paramètres. Lorsque votre réseau atteindra 97% de précision sur l'ensemble d'entraînement, il vous faudra implémenter la fonction d'activation *Rectified Linear Unit* (ReLU) définie comme :

$$a = \max(0, z) \tag{3}$$

En résumé, voici les étapes du TP3 :

1. Vérifier que le code du TP2 fonctionne bien (réseau avec 97% de précision sur l'entraînement)
2. Implémenter la fonction ReLU en vous basant sur l'implémentation de la fonction sigmoïde
3. Montrer à l'aide d'une figure la différence entre l'utilisation de la fonction sigmoïde et ReLU sur différents réseaux, e.g., 128-128, 128-64-32-16, 256-128-64-32-16, 512-256-128-64-32-16
4. Montrer à l'aide d'une figure le phénomène de sur-apprentissage sur les expériences précédentes ou sur un réseau 800-800

Le rapport du TP3 doit contenir les figures ainsi qu'une brève analyse des phénomènes observés. Vous pouvez aussi tenter de répondre aux questions ouvertes que vous avez éventuellement notées dans le rapport du TP2. Dans tous les cas, le rapport du TP3 doit inclure le rapport du TP2.

3 Livrable

Date du livrable : avant le 8 janvier 2016
Format du livrable : un fichier compressé nommé *DL_tp3_prénom_nom.zip* contenant le code et le résumé
Dépôt : à l'adresse `gaetan.marceau-caron@inria.fr` avec comme objet du message *DL_tp3_prénom_nom*.
Description :
Le livrable associé au TP3 doit contenir le code de MiniNN complété et accompagné d'un résumé de trois à quatre pages incluant le résumé du TP2. Le code doit s'exécuter avec la commande `python miniNN.py` et afficher l'évolution de l'apprentissage (sortie par défaut du programme). Le résumé doit être succinct et se focaliser uniquement sur les points essentiels reliés à l'entraînement des

réseaux de neurones. Ce document doit décrire les difficultés que vous avez rencontrées et, dans le cas échéant, les solutions utilisées pour les résoudre. Vous pouvez aussi y décrire vos questions ouvertes et proposer une expérience sur MNIST afin d'y répondre.

Références

BISHOP, Christopher M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA : Springer-Verlag New York, Inc. ISBN : 0387310738.