INF8225 TP2 H18
Date limite : 12h le 18 février 2018

1 Partie I (10 points)

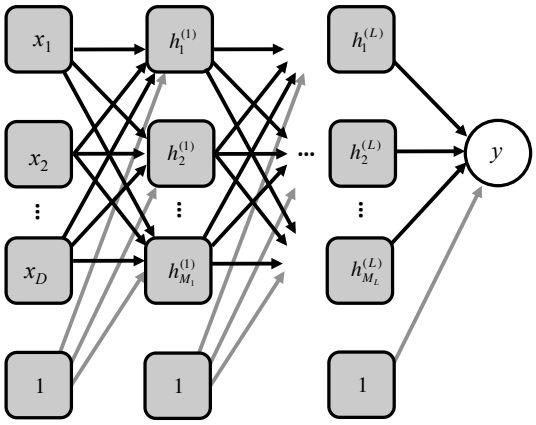

Figure 1: Un réseau de neurones.

Considérons un réseau de neurones avec une seule couche d'entrée avec $D = 100$ unités, L couches cachées, chacun avec 100 unités et une seule neurone de sortie continue. Vous avez $i = 1, .., N$ exemples $y_i \in [0, 1]$, $\mathbf{x}_i \in \mathbb{R}^{100}$ dans un ensemble d'apprentissage. On a une fonction de perte donnée par

$$L = \sum_{i=1}^{N} (y_i - f(\mathbf{x}_i))^2 = \sum_{i=1}^{N} (y_i - f(a^{(L+1)}(\mathbf{h}^{(L)}(\mathbf{a}^{(L)}(\dots \mathbf{h}^{(1)}(\mathbf{a}^{(1)}(\mathbf{x}_i))))))^2 \tag{1}$$

où $\mathbf{x}_i$ est un exemple d'entrée, y_i un scalaire continu avec la cible qui doit être prédite, et avec une fonction d'activation f de la couche finale ayant la forme d'une fonction sigmoïde et chaque fonction d'activation $h_k^{(l)}$ aussi ayant la forme d'une fonction sigmoïde

$$f(a^{(L+1)}(\mathbf{x}_i)) = \frac{1}{1 + \exp(-a^{(L+1)}(\mathbf{h}^{(L)}(\mathbf{x}_i)))}, \quad h_k^{(l)}(\mathbf{a}^{(l)}(\mathbf{x}_i)) = \frac{1}{1 + \exp(-a_k^{(l)}(\mathbf{x}_i))},$$

où $\mathbf{a}^{(l)} = [a_1^{(l)} ... \ a_K^{(l)}]^T$ est le vecteur résultant du calcul de la préactivation habituel $\mathbf{a}^{(l)} = \mathbf{W}^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}$, qui pourrait être simplifié à $\boldsymbol{\theta}^{(l)}\hat{\mathbf{h}}^{(l-1)}$ en utilisant l'astuce de définir $\hat{\mathbf{h}}$ comme $\mathbf{h}$ avec un 1 concaténé à la fin du vecteur.

a) (6 points) Donnez quelque pseudocode incluant des calculs matriciels — vectoriels détaillés pour l'algorithme de rétropropagation pour calculer le gradient pour les paramètres de chaque couche étant donné un exemple d'entraînement.
b) (4 points) Imaginez que vous avez maintenant un jeu de données avec N = 500 000 exemples. Expliquez comment vous utiliserez votre pseudocode pour optimiser ce réseau neuronal dans le contexte d'une expérience d'apprentissage machine correctement effectuée.

Part I – English version (10 points)

Consider a neural network with a single input layer having $D = 100$ units, L hidden layers, each having 100 units and a single continuous valued output neuron. You have $i = 1, .., N$ examples $y_i \in [0, 1]$, $\mathbf{x}_i \in \mathbb{R}^{100}$ in a training set. The loss function is given by

$$L = \sum_{i=1}^{N} (y_i - f(\mathbf{x}_i))^2 = \sum_{i=1}^{N} (y_i - f(a^{(L+1)}(\mathbf{h}^{(L)}(\mathbf{a}^{(L)}(\dots \mathbf{h}^{(1)}(\mathbf{a}^{(1)}(\mathbf{x}_i))))))^2 \tag{2}$$

where $\mathbf{x}_i$ is an input example, y_i is a continuous scalar value which is the prediction target. The output activation function f has the form of a sigmoid and each hidden layer activation function $h_k^{(l)}$ also has the form of a sigmoid function, such that

$$f(a^{(L+1)}(\mathbf{x}_i)) = \frac{1}{1 + \exp(-a^{(L+1)}(\mathbf{h}^{(L)}(\mathbf{x}_i)))}, \quad h_k^{(l)}(\mathbf{a}^{(l)}(\mathbf{x}_i)) = \frac{1}{1 + \exp(-a_k^{(l)}(\mathbf{x}_i))},$$

where $\mathbf{a}^{(l)} = [a_1^{(l)} ... \ a_K^{(l)}]^T$ is the vector resulting from the calculation of the usual preactivation function $\mathbf{a}^{(l)} = \mathbf{W}^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}$, which could be simplified to $\boldsymbol{\theta}^{(l)}\hat{\mathbf{h}}^{(l-1)}$ using the trick of defining $\hat{\mathbf{h}}$ as $\mathbf{h}$ with a 1 concatenated at the end.

a) (6 points) Give pseudocode including detailed *matrix-vector calculations* for the backpropatation algorithm to calculate the gradient for the parameters of each layer given a training example.

b) (4 points) Imagine that you now have a dataset with N=500,000 examples. Explain how you would use your pseudocode to optimize this neural network in the context of a properly performed machine learning experiment.

Partie II (20 points)

L'objectif de la partie II est de se familiariser avec la librairie Pytorch. Vous allez expérimenter avec les données **Fashion Mnist**. Les données sont des images similaires à MNIST et consiste en 10 catégories de vêtements. Vous pouvez faire l'exercice en groupe de 2.

Exécutez des expériences avec différentes architectures. Utilisez un ensemble de validation pour sélectionner la meilleure architecture et utiliser l'ensemble seulement à la fin. Comparez vos résultats sous la courbe d'apprentissage en termes de log vraisemblance. Vous pouvez vous référer au code disponible ici: `https://github.com/AlexPiche/INF8225`