Master 2 IAAA
Cours de Deep Learning
TD 5 - 2019-2020

T. Artières et QARMA

Septembre 2019

I. Normalisation des données d'entrée

On s'intéresse à la normalisation des données d'entrée (de dimension d) d'un réseau de neurones totalement connecté. On considère des neurones qui réalisent un produit scalaire de l'entrée et de leur vecteur de poids suivi d'une activation de type sigmoide.

On suppose les données centrées réduites, c'est à dire que chaque composante des entrées a une moyenne nulle et un écart type 1 sur la base d'apprentissage.

On initialise les poids de la première couche cachée avec une loi gaussienne centrée d'écart type σ.

(1) En considérant les composantes d'un vecteur en entrée comme des variables aléatoires normales centrées réduites calculez la moyenne et la variance de l'activation d'un neurone de la première couche cachée.

En déduire une bonne valeur de σ.

(2) L'intialisation dite *Glorot Normal* implémentée dans *Keras* utilise une distribution normal avec $\sigma = \frac{2}{\sqrt{d+N_h}}$ où N_h est le nombre de neurones sur la couche cachée. Commentez cette initialisation.

(3) L'intialisation dite *Glorot Uniform* implémentée dans *Keras* utilise une distribution uniforme dans l'intervalle $[-\frac{6}{\sqrt{d+N_h}}, \frac{6}{\sqrt{d+N_h}}]$ où N_h est le nombre de neurones sur la couche cachée. Commentez cette initialisation.

II. Batch Normalization

Une couche de batch normalisation prend en entrée la sortie de la couche précédente et la normalise par batch puis la transforme par une fonction affine. Elle est en général suivie d'une fonction d'activation de type Relu. On note $x_i, i = 1..m$ les valeurs d'activation d'un neurone de la couche précédente sur un batch d etaille m. Les x_i sont centrés réduits et transformés en $\hat{x}_i$ à partie de statistqiues calsulées sur un batch B suivant:

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

$$\mu_B = 1/m \sum_i x_i$$

$$\sigma_B = 1/m \sum_i (x_i - \mu_B)^2$$

La sortie de la couche de Batch norm est:

$$y_i = \gamma \hat{x}_i + \beta$$

.

Pour utiliser la bacth normalisation il faut pouvoir apprendre les paramètres de la batch normalisation et propager le gradient à travers cette couche.

(1) On suppose que l'on sait calculer le gradient du critère d'erreur en sortie du réseau, noté l comme *loss*, par rapport aux sorties de la batch normalisation layer $\frac{\partial l}{\partial y_i}$. Calculer le gradient de l par rapport aux paramètres de la batch normalisation γ et β.

(2) Calculer le gradient de l par rapport aux entrées de la batch norm layer $\frac{\partial l}{\partial x_i}$.

III. **Gradient et Hessien**

L'optimisation par descente de gradient consiste à se déplacer à chaque itération dans la direction opposée à celle donnée par le gradient.

On s'intéresse à la minimisation d'un critère quadratique $C(w)$ dont le Hessien est noté $\mathcal{H} = [\frac{\partial^2 \mathcal{H}}{\partial w_i \partial w_j}]_{i,j}$. On note w^* le minimiseur global de $C(w)$.

(1) Montrez que le gradient ne pointe pas nécéssairement vers l'optimum, même dans le cas d'un critère d'erreur quadratique dans le vecteur de paramètres w.

(2) On s'intéresse à déterminer comment la valeur optimale du pas de gradient dépend du conditionnement du hessien. Pour commencer exprimez $C(w)$ en fonction de $C(w^*)$ et du Hessien.

(3) Le Hessien d'une forme quadratique est une matrice PSD dont les valeurs propres sont ≥ 0. On note $(u_i)_i$ les vecteurs propres de $\mathcal{H}$ associés aux valeur propres λ_i. En exprimant w^0 dans la base de ces vecteurs propres déterminer le vecteur w^t à la ième itération de la descente de gradient utilisant un pas de gradient ϵ.

(4) En déduire la valeur optimale et la valeur maximale du pas de gradient pour obtenir la convergence. En supposant le pas de gradient optimal choisi, de quoi dépend la vitesse de convergence ?