Introduction aux réseaux de neurones – exercices

Question 1. Considérons un problème de régression où l'ensemble d'apprentissage $S \in \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ contient des couples $(\mathbf{x}_i, y_i) \in \mathbb{R}^d \times \mathbb{R}$. Nous exprimons ainsi la fonction objectif à minimiser pour résoudre les moindres carrés régularisés (aussi nommé la *Régression de Ridge*) :

$$F(\mathbf{w}, b) = \frac{1}{n} \sum_{i=1}^{n} (f_{\mathbf{w},b}(\mathbf{x}_i) - y_i)^2 + \frac{\rho}{2} \|\mathbf{w}\|^2 ,$$

où le prédicteur linéaire $f_{\mathbf{w},b}(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$ est caractérisé par un vecteur de poids $\mathbf{w} \in \mathbb{R}^d$ et un biais $b \in \mathbb{R}$.

(a) Pour $k \in \{1, \ldots, d\}$, calculer $\dfrac{\partial F(\mathbf{w}, b)}{\partial w_k}$, c'est-à-dire le gradient de la fonction objectif selon le $k$ième paramètre du vecteur de poids.

(b) Déduire de la réponse précédente l'expression du gradient $\nabla_{\mathbf{w}} F(\mathbf{w}, b)$.

(c) Calculez $\dfrac{\partial F(\mathbf{w}, b)}{\partial b}$, c'est-à-dire le gradient du biais.

Question 2. Considérons un problème de classification binaire où l'ensemble d'apprentissage $S \in \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ contient des couples $(\mathbf{x}_i, y_i) \in \mathbb{R}^d \times \{0, 1\}$. Nous exprimons ainsi la fonction objectif à minimiser pour résoudre la régression logistique :

$$F(\mathbf{w}, b) = \frac{1}{n} \sum_{i=1}^{n} F_i(\mathbf{w}, b) ,$$

avec

$$\begin{aligned} F_i(\mathbf{w}, b) &= -y_i(\mathbf{w} \cdot \mathbf{x}_i + b) + \log(1 + e^{\mathbf{w} \cdot \mathbf{x}_i + b}) + \frac{\rho}{2} \|\mathbf{w}\|^2 \\ &= L_{\text{nlv}}(\sigma(\mathbf{w} \cdot \mathbf{x}_i + b), y_i) + \frac{\rho}{2} \|\mathbf{w}\|^2 \end{aligned}$$

(a) Calculer $\dfrac{\partial L_{\text{nlv}}(\hat{y}, y)}{\partial \hat{y}}$, où $L_{\text{nlv}}(\hat{y}, y) = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y})$.

(b) Calculer $\dfrac{\partial \sigma(a)}{\partial a}$ où $\sigma(a) = \dfrac{1}{1 + e^{-a}}$.

(c) Pour $k \in \{1, \ldots, d\}$, calculer $\dfrac{\partial(\mathbf{w} \cdot \mathbf{x} + b)}{\partial w_k}$ où $\mathbf{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_d \end{bmatrix}$, $\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix}$.

(d) Pour $k \in \{1, \ldots, d\}$, calculer $\dfrac{\partial \|\mathbf{w}\|^2}{\partial w_k}$.

Rappel : $\|\mathbf{w}\|^2 = \mathbf{w} \cdot \mathbf{w}$

(e) À partir des réponses aux questions précédentes, calculer $\dfrac{\partial F_i(\mathbf{w}, b)}{\partial w_k}$.

Rappel de la règle de la dérivation en chaîne : $\frac{\partial f(h(x))}{\partial x} = \left[\frac{\partial f(a)}{\partial a}\right]_{a=h(x)} \frac{\partial h(x)}{\partial x}$.

(f) Déduire de la réponse précédente l'expression du gradient $\nabla_{\mathbf{w}} F_i(\mathbf{w}, b)$.

(g) Calculez $\dfrac{\partial F_i(\mathbf{w}, b)}{\partial b}$, c'est-à-dire le gradient du biais.

Question 3. Imaginons que nous optimisons la descente en gradient en initialisant tous les paramètres à zéro.

(a) Est-ce que cela peut poser problème dans le cas de la régression logistique ? Rappelons que la régression logistique peut être exprimée comme un réseau n'ayant aucune couche cachée.

(b) Est-ce que cela peut poser problème dans le cas d'un réseau de neurones à une couche cachée ?

(c) Dans le cas d'un réseau de neurones à une couche cachée, est-ce qu'utiliser le «dropout» sur la couche cachée peut résoudre un éventuel problème dû à l'initialisation des paramètres.

Question 4. Les fonctions d'activations utilisées pour les couches cachées d'un réseau de neurones sont rarement des fonctions linéaires $f(a) = a$. Illustrez la raison à l'aide d'un exemple.

Question 5. Considérons une architecture de réseau de neurones dédiée à un problème de classification à C classes. On représente chaque classe par un entier $y \in \{1, \ldots, C\}$, et la couche de sortie du réseau possède C neurones. Pour un exemple d'apprentissage $(\mathbf{x}, y)$, on converti y sous la forme d'un vecteur «one-hot» $\mathbf{y} \in \mathbb{R}^C$, possédant la valeur 1 à l'index correspondant à y, et les valeurs 0 autrement :

$$y = 1 \mapsto \mathbf{y} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \qquad y = 2 \mapsto \mathbf{y} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Les valeurs de couche de sortie $\hat{\mathbf{y}} \in \mathbb{R}^C$ sont obtenues en appliquant la fonction d'activation «softmax» aux valeurs $\mathbf{a} \in \mathbb{R}^C$ propagées par les couches précédentes :

$$\hat{\mathbf{y}} = \begin{bmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_C \end{bmatrix} \quad \text{avec } \hat{y}_i = \text{softmax}(a_i) = \frac{e^{a_i}}{\sum_{j=1}^{C} e^{a_j}}$$

On interprète le vecteur $\mathbf{y}$ comme une distribution de probabilité sur les classes. Lors de l'optimisation du réseau, on minimise la perte du négatif log-vraisemblance :

$$L_{\text{nlv}}(\hat{\mathbf{y}}, y) = -\ln(\hat{y}_y),$$

où $\hat{y}_y$ corresponds à la sortie d'index y du réseau (la probabilité associée à la classe de l'exemple).

Cet exercice consiste à calculer $\nabla_{\mathbf{a}} L_{\text{nlv}}(\hat{y}_y, y)$, c'est-à-dire le gradient qui modifiera les poids associés à la couche de sortie. Pour ce faire, procédons en trois étapes.

(a) Calculer la dérivée partielle associée au neurone de sortie correspondant à la bonne classe ($y = i$) :

$$\frac{\partial}{\partial a_y} L_{\text{nlv}}(\hat{y}_y, y) = \frac{\partial}{\partial a_y} \ln \left[\frac{1}{\text{softmax}(a_y)} \right].$$

(b) Calculer la dérivée partielle associée au neurone de sortie correspondant à la bonne classe ($y \neq i$) :

$$\frac{\partial}{\partial a_i} L_{\text{nlv}}(\hat{y}_y, y) = \frac{\partial}{\partial a_i} \ln \left[\frac{1}{\text{softmax}(a_y)} \right], \text{ avec } a_i \neq a_y \,.$$

(c) À partir des réponses (a) et (b), déduire le gradient $\nabla_{\mathbf{a}} L_{\text{nlv}}(\hat{y}_y, y)$.