IFT 6390A Fundamentals of Machine Learning
Guillaume Rabusseau

Devoir 3 - Partie Théorique

- Ce devoir doit être déposé sur Gradescope et peut-être être fait en équipes de 3 étudiants. Vous pouvez discuter avec des étudiants d'autres groupes mais les réponses soumises par le groupe doivent être originales.
- La partie théorique doit être envoyée au format pdf. Il est recommandé de l'écrire en LaTeX, en clônant le répertoire (`Menu -> Copy Project`) et en travaillant directement sur ce fichier. Toutefois, toute solution **lisible** au format pdf sera acceptée. Les solutions difficiles à lire pourront être pénalisées, même si elles sont justes
- Seulement un étudiant doit soumettre les solutions et vous devez ajouter votre membre d'équipe sur la page de soumission de gradescope

1. **Dérivées et relations entre fonctions usuelles** [20 points]

   On définit:

   - La fonction "**sigmoïde**": $x \mapsto \sigma(x) = \frac{1}{1+\exp(-x)}$.
   - La fonction "**tangente hyperbolique**": $x \mapsto \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$.
   - La fonction "**softplus**": $x \mapsto \text{softplus}(x) = \ln(1 + \exp(x))$
   - La fonction "**signe**" $x \mapsto \text{sign}(x)$, qui retourne +1 si son argument est strictement positif, -1 s'il est strictement négatif, et 0 si l'argument est 0.
   - La fonction "**Heaviside**" $x \mapsto \text{H}(x)$, qui retourne +1 si son argument est strictement positif, 0 s'il est strictement négatif, et $\frac{1}{2}$ si l'argument est 0.
   - La fonction "**indicatrice**" $x \mapsto \mathbf{1}_S(x)$, qui retourne 1 si $x \in S$ (ou x respecte la condition S), et sinon retourne 0.
   - La fonction "**rectificatrice**" qui ne garde que la partie positive de l'argument: $x \mapsto \text{rect}(x)$ retourne x si $x \geq 0$ et 0 sinon. Elle est nommée RELU généralement. $\text{rect}(x) = \text{RELU}(x) = [x]_+ = \max(0, x) = \mathbf{1}_{\{x>0\}}(x)$
   - La fonction "**diagonale**" : $\mathbf{x} \in \mathbb{R}^n \mapsto diag(\mathbf{x}) \in \mathbb{R}^{n \times n}$ tel que $diag(\mathbf{x})_{ij} = \mathbf{x}_i$ si $i = j$ et $diag(\mathbf{x})_{ij} = 0$ si $i \neq j$. Ici, $\mathbb{R}^{n \times n}$ est l'ensemble des matrices carrées de taille n.
   - La fonction "**softmax**" : $\mathbf{x} \in \mathbb{R}^n \mapsto S(\mathbf{x}) \in \mathbb{R}^n$ tel que $S(\mathbf{x})_i = \frac{e^{\mathbf{x}_i}}{\sum_j e^{\mathbf{x}_j}}$.

   Notez que dans ce devoir, nous utiliserons parfois le symbole de la dérivée partielle pour différentier par rapport à un vecteur, $\mathbf{x}$. Dans ces cas-là, on utilisera pour dénoter le gradient: $\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = \nabla f(\mathbf{x})$.

   (a) [1 points] Ecrivez la fonction signe(x), en utilisant seulement des fonctions indicatrices.

   (b) [1 points] Montrez que la dérivée de l'unité linéaire rectifiée (ReLU) $g(x) = \max\{0, x\}$, **lorsqu'elle existe** est égale à la fonction échelon de Heaviside.

(c) [2 points] Donnez deux définitions alternatives de $g(x)$ en utilisant $H(x)$.

(d) [1 points] Montrer que $H(x)$ peut être bien approchée par la fonction $a(x) = \frac{1}{1+e^{-kx}}$ asymptotiquement (c'est-à-dire aux grands k), où k est un paramètre.

(e) [2 points] Ecrivez la dérivée de la fonction sigmoïde, σ', en utilisant seulement la fonction σ. **Trouvez** la matrice jacobienne de la fonction logistique (Utilizez Kronecker delta $\delta_{i=j}$)

(f) [1 points] Montrez que $\ln \sigma(x) = -\text{softplus}(-x)$.

(g) [2 points] Montrez que $\text{softplus}(x) - \text{softplus}(-x) = x$ et déduire la dérivée de $\text{softplus}(x)$ (vous devez utiliser la fonction sigmoid)

(h) [1 points] Montrez que la fonction softmax est invariante aux translations, c'est-à-dire : $S(\mathbf{x} + c) = S(\mathbf{x})$, où c est une constante.

(i) [1 points] Il est évident que le softmax n'est pas invariant sous la multiplication. Quel est l'effet de la multiplication du vecteur par un scalaire plus grand que 1 ? Plus petit que 1 ? En considérant la sortie du softmax comme la valeur de la prédiction du réseau, quel est l'effet de cette multiplication.

(j) [1 points] Montrez que les dérivées partielles de la fonction softmax sont données par: $\frac{\partial S(\mathbf{x})_i}{\partial \mathbf{x}_j} = S(\mathbf{x})_i(\delta_{i=j} - S(\mathbf{x})_j)$.

(k) [1 points] Exprimez la matrice jacobienne de la fonction softmax $\frac{\partial S(\mathbf{x})}{\partial \mathbf{x}}$, en utilisant la notation matricielle/vectorielle. Vous devez utiliser la fonction *diag* .

On rappelle que $\frac{\partial S(\mathbf{x})}{\partial \mathbf{x}}$ est une matrice de taille $n \times n$, et pour tout $i, j \in \{1, \ldots, n\}$, l'entrée (i, j) de la matrice est $\left(\frac{\partial S(\mathbf{x})}{\partial \mathbf{x}}\right)_{i,j} = \frac{\partial S(\mathbf{x})_i}{\partial \mathbf{x}_j}$.

(l) [3 points] Soient $\mathbf{x}$ une fonction du vecteur $\mathbf{u}$. Montrez que le gradient $\nabla_u \log S(\mathbf{x}(\mathbf{u}))$ est égale à:

$$\nabla_u \mathbf{x}(\mathbf{u})_i - \mathbb{E}_j[\nabla_u \mathbf{x}(\mathbf{u})_j]$$

ou j est un indexe aléatoire suivant une distribution catégorique avec probabilité $S(\mathbf{x}(\mathbf{u}))_j$

(m) [3 points] Soient $\mathbf{x}$ and $\mathbf{y}$ deux vecteurs à K dimensions en relation par $\mathbf{y} = S(\mathbf{x})$. Utilisez le résultat obtenu à la question précédente pour dériver le gradient de *cross-entropy loss* (c.à.d. le *negative log likelihood*) par rapport à l'entrée du softmax, $\nabla_u L(\mathbf{x}, \mathbf{c})$ ou $\mathbf{c}$ est un vecteur one-hot correspondant à l'étiquette de la classe (c.à.d. un vecteur de 0 partout à l'exception de la position associée à la bonne classe qui est représentée par un 1)

$$L(\mathbf{x}, \mathbf{c}) = \sum_{i=1}^{K} -\mathbf{c}_i \log \mathbf{y}_i$$

2. **Calcul de gradients pour l'optimisation des paramètres d'un réseau de neurones pour la classification multiclasse** [35 points]

Soit $D_n = \{(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(n)}, y^{(n)})\}$ un jeu de données avec $x^{(i)} \in \mathbb{R}^d$ et $y^{(i)} \in \{1, \ldots, m\}$ indiquant une étiquette parmi m classes. **Pour les vecteurs et les matrices dans les équations qui vont suivre, les vecteurs sont par défaut considérés comme des vecteurs colonnes.**

On considère un réseau de neurone de type perceptron multicouche (MLP) avec une seule couche cachée (donc 3 couches en tout si on compte la couche d'entrée et la couche de sortie).

La couche cachée est constituée de d_h neurones complètement connectés à la couche d'entrée. Nous allons considérer pour la couche cachée une non-linéarité ϕ, définie comme suit:

$$\phi(x) = softplus(x).$$

La couche de sortie est constituée de m neurones, complètement connectés à la couche cachée. Ils ont une non-linéarité de type **softmax**. La sortie du $j^{\text{ème}}$ neurone de la couche de sortie donnera un score pour la classe j interprété comme la probabilité que l'entrée x soit de cette classe j.

Il vous est fortement conseillé de dessiner le réseau de neurones au fur et à mesure afin que vous puissiez mieux suivre les étapes (mais pas besoin de nous fournir un dessin!)

(a) [2 points] Soit $\mathbf{W}^{(1)}$ lq matrice $d_h \times d$ de poids et soit $\mathbf{b}^{(1)}$ le vecteur de biais caractérisant des connexions synaptiques allant de la couche d'entrée à la couche cachée. Indiquez la dimension de $\mathbf{b}^{(1)}$. Donnez la formule de calcul du vecteur de pré-activations (i.e. avant non-linéarité) des neurones de la couche cachée $\mathbf{h}^a$ à partir d'une observation d'entrée $\mathbf{x}$, d'abord sous la forme d'une expression de calcul matriciel ($\mathbf{h}^a = \ldots$), puis détaillez le calcul d'un éléments $\mathbf{h}^a_j = \ldots$. Exprimez le vecteur des sorties des neurones de la couche cachée $\mathbf{h}^s$ en fonction de $\mathbf{h}^a$.

(b) [2 points] Soit $\mathbf{W}^{(2)}$ la matrice de poids et soit $\mathbf{b}^{(2)}$ le vecteur de biais caractérisant les connexions synaptiques allant de la couche cachée à la couche de sortie. Indiquez les dimensions de $\mathbf{W}^{(2)}$ et $\mathbf{b}^{(2)}$. Donnez la formule de calcul du vecteur d'activations des neurones de la couche de sortie $\mathbf{o}^a$ à partir de leurs entrées $\mathbf{h}^s$ sous la forme d'une expression de calcul matriciel, puis détaillez le calcul de $\mathbf{o}^a_k$.

(c) [1 points] La sortie des neurones de sortie est donnée par

$$\mathbf{o}^s = \text{softmax}(\mathbf{o}^a)$$

Précisez l'équation des $\mathbf{o}^s_k$ en utilisant explicitement la formule du softmax (formule avec des exp). **Démontrez** que les $\mathbf{o}^s_k$ sont positifs et somment à 1. Pourquoi est-ce important?

(d) [1 points] Nous supposons pour pour le moment que notre problème est une **classification binaire**

Montrez que l'utilisation de l'activation softmax pour la couche de sortie est équivalente à utiliser la fonction sigmoid.

(e) [2 points] Nous choisissons comme fonction de coût: l'erreur quadratique moyenne *(MSE)*. Donnez l'expression de $L_{MSE}(\sigma(\mathbf{o}^a), y)$ et calculer:

$$\frac{\partial L_{MSE}(\sigma(\mathbf{o}^a), y)}{\partial \mathbf{o}^a}$$

(f) [2 points] Nous choisissons maintenant comme fonction de coût: l'entropie croisée (Cross-entropy loss) *(CE)*. Donnez l'expression de $L_{CE}(\sigma(\mathbf{o}^a), y)$ et calculer:

$$\frac{\partial L_{CE}(\sigma(\mathbf{o}^a), y)}{\partial \mathbf{o}^a}$$

(g) [2 points] En se basant sur vos résultats des deux dernières questions, montrez que l'entropie croisée est la fonction de coût la plus appropriée pour ce problème de classification binaire.

(h) [2 points] Pour les questions suivantes, le problème est une **classification multi-classes**.

Le réseau de neurones calcule donc, pour un vecteur d'entrée $\mathbf{x}$, un vecteur de scores (probabilités) $\mathbf{o}^s(\mathbf{x})$. La probabilité, calculée par le réseau de neurones, qu'une observation $\mathbf{x}$ soit de la classe y est donc donnée par la $y^{ième} sortie$ $\mathbf{o}^s_y(\mathbf{x})$. Ceci suggère d'utiliser la fonction de perte:

$$L(\mathbf{x}, y) = \text{-log } \mathbf{o}^s_y(\mathbf{x})$$

Précisez l'équation de L directement en fonction du vecteur $\mathbf{o}^a$. Il suffit pour cela d'y substituer convenablement l'équation exprimée au point précédent.

(i) [2 points] L'entraînement du réseau de neurones va consister à trouver les paramètres du réseau qui minimisent le risque empirique $\hat{R}$ correspondant à cette fonction de perte. Formulez $\hat{R}$. Indiquez précisément de quoi est constitué l'ensemble θ des paramètres du réseau. Indiquez à combien de paramètres scalaires $n_\theta celacorrespond$. Formulez le problème d'optimisation qui correspond à l'entraînement du réseau permettant de trouver une valeur optimale des paramètres.

(j) [1 points] Pour trouver la solution à ce problème d'optimisation, on va utiliser une technique de descente de gradient. Exprimez la technique de descente de gradient (full-batch) pour ce problème.

(k) [1 points] Nous proposons pour cette question d'ajouter un terme de régularisation $L2$:

$$\hat{R}_{reg}(\theta) = \hat{R}(\theta) + \lambda \|\theta\|_2^2$$

Réécrire la technique de descente de gradient (après simplification) pour ce problème.

(l) [3 points] Notez que le calcul du vecteur de gradient du risque empirique $\hat{R}$ par rapport à l'ensemble des paramètres θ peut s'exprimer comme

$$\begin{pmatrix} \frac{\partial \hat{R}}{\partial \theta_1} \\ \vdots \\ \frac{\partial \hat{R}}{\partial \theta_{n_\theta}} \end{pmatrix} = \frac{1}{n} \sum_{i=1}^{n} \begin{pmatrix} \frac{\partial L(\mathbf{x}_i, y_i)}{\partial \theta_1} \\ \vdots \\ \frac{\partial L(\mathbf{x}_i, y_i)}{\partial \theta_{n_\theta}} \end{pmatrix}$$

Il suffit donc de savoir calculer le gradient du coût L encouru pour un exemple $(\mathbf{x}, y)$ par rapport aux paramètres, que l'on définit comme:

$$\frac{\partial L}{\partial \theta} = \begin{pmatrix} \frac{\partial L}{\partial \theta_1} \\ \vdots \\ \frac{\partial L}{\partial \theta_{n_\theta}} \end{pmatrix} = \begin{pmatrix} \frac{\partial L(\mathbf{x}, y)}{\partial \theta_1} \\ \vdots \\ \frac{\partial L(\mathbf{x}, y)}{\partial \theta_{n_\theta}} \end{pmatrix}$$

Pour cela on va appliquer la technique de **rétropropagation du gradient**, en partant du coût L, et en remontant de proche en proche vers la sortie $\mathbf{o}$ puis vers la couche cachée $\mathbf{h}$ et enfin vers l'entrée $\mathbf{x}$.

Démontrez que

$$\frac{\partial L}{\partial \mathbf{o}^a} = \mathbf{o}^s - onehot_m(y)$$

Indication: Partez de l'expression de L en fonction de $\mathbf{o}^a$ que vous avez précisée plus haut. Commencez par calculer $\frac{\partial L}{\partial \mathbf{o}^a_k}$ pour $k \neq y$ (en employant au début l'expression de la dérivée du logarithme). Procédez de manière similaire pour $\frac{\partial L}{\partial \mathbf{o}^a_y}$.

IMPORTANT: Dorénavant quand on demande de "calculer" des gradients ou dérivées partielles, il s'agit simplement d'exprimer leur calcul en fonction d'éléments déjà calculés aux questions précédentes (ne substituez pas les expressions de dérivées partielles déjà calculées lors des questions d'avant)!

(m) [3 points] Calculez les gradients par rapport aux paramètres $\mathbf{W}^{(2)}$ et $\mathbf{b}^{(2)}$ de la couche de sortie. Comme L ne dépend des $\mathbf{W}^{(2)}_{kj}$ et $\mathbf{b}^{(2)}_k$ qu'au travers de $\mathbf{o}^a_k$ la règle de dérivation en chaîne nous donne:

$$\frac{\partial L}{\partial \mathbf{W}^{(2)}_{kj}} = \frac{\partial L}{\partial \mathbf{o}^a_k} \frac{\partial \mathbf{o}^a_k}{\partial \mathbf{W}^{(2)}_{kj}}$$

et

$$\frac{\partial L}{\partial \mathbf{b}^{(2)}_k} = \frac{\partial L}{\partial \mathbf{o}^a_k} \frac{\partial \mathbf{o}^a_k}{\partial \mathbf{b}^{(2)}_k}$$

(n) [2 points] Exprimez le calcul du gradient de la question précédente sous forme d'une expression matricielle, en définissant la dimension de chacune des matrices ou vecteurs manipulés. (Pour le gradient par rapport à $\mathbf{W}^{(2)}$, on veut arranger les dérivées partielles dans une matrice qui a la même forme que $\mathbf{W}^{(2)}$, aet c'est ce que l'on appelle le gradient)

Précisez les dimensions.

Assurez-vous de bien comprendre pourquoi ces expressions matricielles sont équivalentes aux expressions de la question précédente.

(o) [2 points] Calculez les dérivées partielles du coût L par rapport aux sorties des neurones de la couche cachée. Comme L dépend d'un neurone caché $\mathbf{h}^s_j$ au travers des activations de tous les neurones de sortie $\mathbf{o}^a$ reliés à ce neurone caché, la règle de dérivation en chaîne nous donne:

$$\frac{\partial L}{\partial \mathbf{h}^s_j} = \sum_{k=1}^{m} \frac{\partial L}{\partial \mathbf{o}^a_k} \frac{\partial \mathbf{o}^a_k}{\partial \mathbf{h}^s_j}$$

(p) [2 points]Exprimez le calcul de la question précédente sous forme d'une expression matricielle, en définissant la dimension de chacune des matrices ou vecteurs manipulées.

Précisez les dimensions...

Assurez-vous de bien comprendre pourquoi cette expression matricielle est équivalente aux calculs de la question précédente.

(q) [3 points] Calculez les dérivées partielles par rapport aux activations des neurones de la couche cachée. Comme L ne dépend de l'activation $\mathbf{h}^a_j$ d'un neurone de la couche cachée qu'au travers de la sortie $\mathbf{h}^s_j$ de ce neurone, la règle de dérivation en chaîne donne:

$$\frac{\partial L}{\partial \mathbf{h}^a_j} = \frac{\partial L}{\partial \mathbf{h}^s_j} \frac{\partial \mathbf{h}^s_j}{\partial \mathbf{h}^a_j}$$

Notez que $\mathbf{h}^s_j = \phi(\mathbf{h}^a_j)$: la fonction ϕ s'applique élément par élément. Comme étape intermédiaire, commencez par exprimer la dérivée de la fonction $\frac{\partial \phi(z)}{\partial z} = \phi'(z) = \ldots$.

(r) [2 points] Exprimez le calcul de la question précédente sous forme d'une expression matricielle, en définissant la dimension de chacune des matrices ou vecteurs manipulés.

3. **Réseau de neurones à convolution** [10 points]

Un réseau de neurones convolutifs (ConvNet / CNN) est un modèle de réseau de neurones qui peut prendre une image d'entrée, attribuer des poids et des biais apprenables à divers aspects / objets de l'image et être capable de reconnaître différentes caractéristiques de l'entrée. La couche convolutive est la pierre angulaire d'un CNN. Les paramètres de la couche sont constitués d'un ensemble de filtres (ou noyaux) apprenables. Après avoir passé une image à travers une couche convolutionnelle, elle devient abstraite dans une carte d'entités, avec la forme (nombre d'images) × (hauteur de la transformation) × (largeur de la transformation) × (canaux de la transformation). Les convolutions peuvent être représentées comme une multiplication de matrice clairsemée, voici les concepts dont vous avez besoin pour cette question:

- **Noyau (Kernel)**: Il est généralement connu comme une carte de caractéristiques ou un filtre convolutif défini par une largeur et une hauteur.
- **Stride** (s_i): Distance entre deux positions consécutives du noyau le long de l'axe i.
- **Zero-padding** (p_i): Nombre de zéros concaténés au début et à la fin d'un axe) le long de l'axe i.
- **Dilation** (d_i): Il est utilisé dans les convolutions dilatées qui «gonflent» le noyau, en insérant des espaces entre les éléments du noyau le long de l'axe i.

On considère un réseau de neurones à convolution. On suppose que l'entrée (*input*) est une image en couleurs de taille 128×128 dans la représentation Rouge Vert Bleu (*RGB*). La première couche convolue 32 noyaux 8×8 avec l'entrée, en utilisant un pas (*stride*) de 2, et une marge (*padding*) nulle de 3. La deuxième couche sous-échantillonne (*downsampling*) la sortie (*output*) de la première couche avec un *max-pool* 2×2 sans chevauchement (*no overlapping*). La troisième couche convolue 64 noyaux 3×3 avec un pas de 1, et une marge de 1 de chaque côté.

(a) [3 points] Quelle est la dimension (scalaire) de la sortie à la dernière couche?

(b) [2 points] Sans compter les biais, combien de paramètres sont requis pour la dernière couche?

(c) [3 points] Calculez la convolution complète (*full* convolution), valide (*valid* convolution), et similaire (*same* convolution), avec retournement de noyau (*kernel flipping*) pour les matrices unidimensionnelles suivantes: $\left[1, 1, 4, 4, 4, 1, 1\right] * \left[1/4, 1/4, 1/4, 1/4\right]$

(d) [2 points] Commentez le résultat de l'opération de convolution précédente (en indiquant ce que cette opération de convolution nous permet de faire)?

Réponses Exercice 1

(a)

$$x \mapsto \text{sign}(x) = \begin{cases} 1 & \text{si } x > 0 \\ 0 & \text{si } x = 0 \\ -1 & \text{si } x < 0 \end{cases}$$

et

$$x \mapsto \mathbf{1}_S(x) = \begin{cases} 1 & \text{si x vérfie S} \\ 0 & \text{sinon} \end{cases}$$

On a donc :

$$\text{sign}(x) = \mathbf{1}_{x>0}(x) - \mathbf{1}_{x<0}(x)$$

b)

$g(x) = max\{0, x\}$ soit $\frac{\partial g}{\partial x} = \begin{cases} 0 & \text{si } x < 0 \\ 1 & \text{si } x > 0 \end{cases}$

En 0, la dérivée n'est pas définie car elle n'est pas la même à "gauche" de 0 ($\frac{\partial g}{\partial x} \mapsto 0$ quand $x \mapsto 0^-$) et à "droite" de 0 ($\frac{\partial g}{\partial x} \mapsto 1$ quand $x \mapsto 0^+$).
On retombe bien sur la fonction Heaviside, $\forall x \in \mathbb{R}^*$.

c) Nous pouvons donc donner deux définitions de la fonction $g(x)$. $\forall x \in \mathbb{R}, H(x) = \begin{cases} \frac{\partial g}{\partial x}(x) & \forall x \in \mathbb{R}^* \\ \frac{1}{2} & \text{si } x = 0 \end{cases}$

d'où $\forall x \in \mathbb{R}, g(x) = \begin{cases} \int H(x)dx & \forall x \in \mathbb{R}^* \\ 0 & \text{si } x = 0 \end{cases}$

Une autre défintion serait :

$$g(x) = xH(x)$$

(d) Montrons que la fonction H peut être approchée par $a(x) = \frac{1}{1+\exp -kx}$ asymptotiquement (aux grands k), où k est un paramètre.

On a:

$$a(0) = \frac{1}{2}$$

Supposons que nous fassions tendre $k \mapsto +\infty$:
Si $x < 0$: $\lim_{k\to+\infty} a(x) = 0$ car $\lim_{k\to+\infty} \exp -kx = +\infty$

Si $x > 0$: $\lim_{k\to+\infty} a(x) = 1$ car $\lim_{k\to+\infty} \exp -kx = 0$
On retombe bien sur la fonction Heaviside.

e) Calculons $\sigma'(x)$
On a:

$$
\begin{aligned}
\sigma'(x) &= \frac{d}{dx}(\frac{1}{1+\exp -x}) \\
&= \frac{d}{dx}[(1+\exp -x)^{-1}] \\
&= -1 \cdot (1+\exp -x)^{-2} \cdot (-\exp -x) \\
&= \frac{\exp -x}{(1+\exp -x)^2} \\
&= \frac{1}{1+\exp -x} \cdot \frac{\exp -x}{1+\exp -x} \\
&= \frac{1}{1+\exp -x} \cdot \frac{\exp -x+1-1}{1+\exp -x} \\
&= \sigma(x)[\frac{\exp -x+1}{1+\exp -x} + \frac{-1}{1+\exp -x}] \\
&= \sigma(x)(1-\sigma(x))
\end{aligned}
$$

Supposons $x \in \mathbb{R}^n$ et calculons la matrice Jacobienne de $\sigma(x)$.
$\forall i, j \in 1, ..., n,$

$$
(J(\sigma(x)))_{i,j} = \frac{\partial(\sigma(x))_i}{\partial x_j} = \begin{cases} \sigma(x_i)(1-\sigma(x_i)) & \text{si } i = j \text{ (d'après notre calcul précédent)} \\ 0 & \text{sinon} \end{cases} = \delta_{i=j}\sigma(x_i)(1-\sigma(x_i))
$$

La matrice Jacobienne de σ est donc une matrice diagonale qui $\forall i = 1, ...n, (J(\sigma(x)))_{i,i} = \sigma(x_i)(1-\sigma(x_i))$.

f) Montrons que $ln\sigma(x) = -softplus(-x)$.
On a :

$$
\begin{aligned}
ln\sigma(x) &= ln(\frac{1}{1+\exp -x}) \\
&= ln(1) - ln(1+\exp -x) \\
&= -ln(1+\exp -x) \\
&= -softplus(-x) \qquad \text{par définition de la fonction softplus}
\end{aligned}
$$

g) Montrons que $softplus(x) - softplus(-x) = x$ et calculons la dérivée de la fonction $softplus$.

On a:
D'après la question précédente, $ln\sigma(x) = -softplus(-x)$.

De plus ,

$$\begin{aligned} -ln\sigma(-x) &= -ln(\frac{1}{1+\exp -x}) \\ &= -ln(1) + ln(1+\exp -x) \\ &= ln(1+\exp -x) \\ &= softplus(x) \qquad \text{par définition de la fonction softplus} \end{aligned}$$

d'où

$$\begin{aligned} softplus(x) - softplus(-x) &= -ln\sigma(-x) + ln\sigma(x) \\ &= ln(\frac{\sigma(x)}{\sigma(-x)}) \\ &= ln(\frac{1+\exp x}{1+\exp -x}) \\ &= ln(\frac{\exp x(1+\exp -x)}{1+\exp -x}) \\ &= ln(\exp x) \\ &= x \end{aligned}$$

Ainsi $softplus(x) = x + softplus(-x) = x - ln(\sigma(x))$.

d'où $\frac{d}{dx}softplus(x) = \frac{dx}{dx} - \frac{d}{dx}(ln(\sigma(x))) = 1 - \frac{d}{dx}(ln(\sigma(x)))$

Or par composée de fonctions, $\frac{d}{dx}(ln(\sigma(x))) = \frac{dln(\sigma(x))}{d\sigma(x)}\frac{d\sigma(x)}{dx} = \frac{1}{\sigma(x)}\sigma(x)(1-\sigma(x)) = 1 - \sigma(x)$.

On a donc finalement, $\frac{d}{dx}softplus(x) = 1 - 1 + \sigma(x) = \sigma(x)$.

h) Montrons que la fonction Softmax est invariante par translation:

$$\begin{aligned} S(x+c)_i &= \frac{\exp{(x_i + c)}}{\sum_j \exp{(x_j + c)}} \\ &= \frac{\exp x_i \exp c}{\exp c \sum_j \exp x_j} \\ &= \frac{\exp x_i}{\sum_j \exp x_j} \\ &= S(x)_i \end{aligned}$$

i) Si nous multiplions par un scalaire plus grand que 1, cela aura pour impact de "rapprocher" nos valeurs vers les extrêmes (c'est à dire 0 ou 1). A contrario, si nous multiplions par un scalaire plus petit que 1, d'éloigner nos valeurs des extrema..

j) Montrons que les dérivées partielles de la fonction Softmax sont données par $\frac{\partial S(\mathbf{x})_i}{\partial \mathbf{x}_j} = S(\mathbf{x})_i(\delta_{i=j} - S(\mathbf{x})_j)$.
Si $i = j$:

$$\begin{aligned}\frac{\partial S(\mathbf{x})_i}{\partial \mathbf{x}_j} &= \frac{\exp x_i \sum_k \exp x_k - (\exp x_i)^2}{(\sum_k \exp x_k)^2} \\ &= \frac{\exp x_i}{\sum_k \exp x_k} - \frac{(\exp x_i)^2}{(\sum_k \exp x_k)^2} \\ &= S(x)_i - (S(x)_j)^2 \\ &= S(x)_i(1 - S(x)_j)\end{aligned}$$

Si $i \neq j$:

$$\begin{aligned}\frac{\partial S(\mathbf{x})_i}{\partial \mathbf{x}_j} &= \exp x_i \frac{-\exp x_j}{(\sum_k \exp x_k)^2} \\ &= \frac{\exp x_i}{\sum_k \exp x_k} \frac{-\exp x_j}{\sum_k \exp x_k} \\ &= -S(x)_i S(x)_j\end{aligned}$$

Soit plus généralement, $\forall i, j \in \{1, ..., n\}$, $\frac{\partial S(\mathbf{x})_i}{\partial \mathbf{x}_j} = S(\mathbf{x})_i(\delta_{i=j}) - S(\mathbf{x})_j$

k) Exprimons la matrice jacobienne de la fonction Softmax $\frac{\partial S(\mathbf{x})}{\partial \mathbf{x}}$.
On a d'après la question précédente: $\forall i, j \in \{1, ..., n\}$, $\frac{\partial S(\mathbf{x})_i}{\partial \mathbf{x}_j} = S(\mathbf{x})_i(\delta_{i=j}) - S(\mathbf{x})_j$.

D'où

$$J(S(x)) = \begin{bmatrix} S_1 - (S_1)^2 & -S_1S_2 & \cdots & -S_1S_n \\ -S_1S_2 & S_2 - (S_2)^2 & \cdots & -S_2S_n \\ \vdots & & \ddots & \vdots \\ -S_1S_n & \cdots & \cdots & S_n - (S_n)^2 \end{bmatrix}$$

Or,

$$S(x)(S(x))^T = \begin{bmatrix} S_1^2 & S_1S_2 & \cdots & S_1S_n \\ S_1S_2 & S_2^2 & \cdots & S_2S_n \\ \vdots & & \ddots & \vdots \\ S_1S_n & \cdots & \cdots & S_n^2 \end{bmatrix}$$

D'où, $J(S(x)) = diag(S(x)) - S(x)(S(x))^T$

l) Soient $\mathbf{x}$ une fonction du vecteur $\mathbf{u}$. Montrons que $\nabla_u \log S(\mathbf{x}(\mathbf{u}))_i = \nabla_u \mathbf{x}(\mathbf{u})_i - \mathbb{E}_j[\nabla_u \mathbf{x}(\mathbf{u})_j]$.

On a:

$$\begin{aligned}\log S(\mathbf{x}(\mathbf{u}))_i &= \log \frac{\exp(x(u))_i}{\sum_j \exp(x(u))_j} \\ &= \log \frac{\exp(x(u))_i}{\sum_j \exp(x(u))_j} \\ &= x(u)_i - \log \sum_j \exp(x(u))_j\end{aligned}$$

On a donc :

$$\nabla_u \log S(\mathbf{x}(\mathbf{u}))_i = \nabla_u x(u)_i - \nabla_u \log \sum_j \exp(x(u))_j$$

Or,

$$\frac{\partial \log \sum_k \exp(x(u))_k}{\partial u} = \frac{\partial \log \sum_k \exp(x(u))_k}{\partial \sum_k \exp(x(u))_k} \frac{\partial \sum_k \exp(x(u))_j}{\partial \mathbf{x}(\mathbf{u})} \frac{\mathbf{x}(\mathbf{u})}{\partial \mathbf{u}}$$

On pose $\mathbf{x}(\mathbf{u}) \in \mathbb{R}^n$, et soit $j \in \{1, ..., n\}$
et

$$\begin{aligned}\frac{\mathbf{x}(\mathbf{u})_j}{\partial \mathbf{u}} &= (\nabla_u \mathbf{x}(\mathbf{u})_j)^T \\ \frac{\partial \sum_k \exp(x(u))_k}{\partial \mathbf{x}(\mathbf{u})_j} &= \exp \mathbf{x}(\mathbf{u})_j \\ \frac{\partial \log \sum_j \exp(x(u))_j}{\partial \sum_j \exp(x(u))_j} &= \frac{1}{\sum_j \exp(x(u))_j}\end{aligned}$$

On a donc:

$$\begin{aligned}\frac{\partial \log \sum_k \exp(x(u))_k}{\partial \sum_k \exp(x(u))_k} \frac{\partial \sum_k \exp(x(u))_k}{\partial \mathbf{x}(\mathbf{u})_j} \frac{\mathbf{x}(\mathbf{u})_j}{\partial \mathbf{u}} &= \frac{1}{\sum_j \exp(x(u))_j} \exp \mathbf{x}(\mathbf{u})_j (\nabla_u \mathbf{x}(\mathbf{u})_j)^T \\ &= S(\mathbf{x}(\mathbf{u}))_j (\nabla_u \mathbf{x}(\mathbf{u})_j)^T\end{aligned}$$

Soit de manière plus générale,

$$
\begin{aligned}
\frac{\partial \log \sum_j \exp\left(x(u)\right)_j}{\partial u} &= \frac{\partial \log \sum_j \exp\left(x(u)\right)_j}{\partial \sum_j \exp\left(x(u)\right)_j} \frac{\partial \sum_j \exp\left(x(u)\right)_j}{\partial \mathbf{x}(\mathbf{u})} \frac{\mathbf{x}(\mathbf{u})}{\partial \mathbf{u}} \\
&= \begin{bmatrix} -- & S(\mathbf{x}(\mathbf{u}))^T & -- \\ & \vdots & \\ -- & S(\mathbf{x}(\mathbf{u}))^T & -- \end{bmatrix} \begin{bmatrix} -- & (\nabla_u \mathbf{x}(\mathbf{u})_1)^T & -- \\ & \vdots & \\ -- & (\nabla_u \mathbf{x}(\mathbf{u})_n)^T & -- \end{bmatrix} \\
&= \begin{bmatrix} -- & \sum_{j=1}^n S(\mathbf{x}(\mathbf{u}))_j (\nabla_u \mathbf{x}(\mathbf{u})_j)^T & -- \\ & \vdots & \\ -- & (\sum_{j=1}^n S(\mathbf{x}(\mathbf{u}))_j (\nabla_u \mathbf{x}(\mathbf{u})_j)^T & -- \end{bmatrix} \\
&= \mathbb{E}_j[\nabla_u \mathbf{x}(\mathbf{u})_j]
\end{aligned}
$$

Nous retrouvons donc finalement , $\nabla_u \log S(\mathbf{x}(\mathbf{u}))_i = \nabla_u \mathbf{x}(\mathbf{u})_i - \mathbb{E}_j[\nabla_u \mathbf{x}(\mathbf{u})_j]$.

m) On a $y = S(x)$ et $L(\mathbf{x}, \mathbf{c})) = \sum_{i=1}^K -\mathbf{c}_i \log \mathbf{y}_i$. Calculons $\nabla_u L(\mathbf{x}, \mathbf{c}))$.
On a:

$$
\begin{aligned}
\nabla_u L(\mathbf{x}, \mathbf{c})) &= \nabla_u \sum_{i=1}^K -\mathbf{c}_i \log \mathbf{y}_i \\
&= \sum_{i=1}^K -\mathbf{c}_i \nabla_u \log S(\mathbf{x}(\mathbf{u}))_i \\
&= \sum_{i=1}^K -\mathbf{c}_i (\nabla_u \mathbf{x}(\mathbf{u})_i - \mathbb{E}_j[\nabla_u \mathbf{x}(\mathbf{u})_j]) \\
&= \sum_{i=1}^K -\mathbf{c}_i \nabla_u \mathbf{x}(\mathbf{u})_i + \mathbb{E}_j[\nabla_u \mathbf{x}(\mathbf{u})_j] \sum_{i=1}^K \mathbf{c}_i
\end{aligned}
$$

Réponses Exercice 2

(a) La dimension de $\mathbf{b}^{(1)}$ est d_h, $\mathbf{b}^{(1)} \in \mathbb{R}^{d_h}$.
Nous avons la formule suivante de préactivation :

$$h^a = \mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}$$

soit

$$h_j^a = \sum_{i=1}^d \mathbf{W}_{ji}^{(1)} \mathbf{x}_i + \mathbf{b}_j^{(1)}$$

et nous avons après activation:

$h^s = softplus(h^a) = \phi(h^a)$

(b) Nous avons pour la dernière couche $\mathbf{b}^{(2)} \in \mathbb{R}^m$ et $\mathbf{W}^{(2)} \in \mathbb{R}^{m \times d_h}$.
Nous avons alors :

$$o^a = \mathbf{W}^{(2)} h^s + \mathbf{b}^{(2)}$$

soit,

$$o_k^a = \sum_{i=1}^{m} \mathbf{W}_{ki}^{(2)} h_i^s + \mathbf{b}_k^{(2)}$$

(c) La sortie des neurones est $o^s = softmax(o^a)$.
Nous avons alors

$$o_k^s = \frac{\exp o_k^a}{\sum_{j=1}^m \exp o_j^a} > 0$$

car par propriété de la fonction exponentielle, elle est toujours strictement positive.

De plus,

$$\begin{aligned} \sum_{k=1}^{m} o_k^s &= \sum_{k=1}^{m} \frac{\exp o_k^a}{\sum_{j=1}^m \exp o_j^a} \\ &= \frac{\sum_{k=1}^m \exp o_k^a}{\sum_{j=1}^m \exp o_j^a} \\ &= 1 \end{aligned}$$

Nous souhaitons que chacun des m neurones indiquent une probabilité d'appartenir à la classe correspondante, c'est pour cela que nous souhaitons que les sorties soient positives (par propriété une probabilité est comprise entre 0 et 1) et la somme de toutes les probabilités d'un évènement, par propriété d'une distribution, doit être égale à 1.

(d) Montrons que la fonction Softmax et Sigmoid sont équivalente dans un problème de classification binaire.

Supposons que nous utilisons la fonction Softmax, on a pour la première sortie de o^s

$$
\begin{aligned}
o_1^s &= \frac{\exp o_1^a}{\exp o_1^a + \exp o_2^a} \\
&= \frac{1}{1 + \exp o_2^a - o_1^a} \\
&= \frac{1}{1 + \exp -(o_1^a - o_2^a)}
\end{aligned}
$$

On pose $o_1^a - o_2^a = x$ on obtient alors $\frac{1}{1+\exp -(x)} = sigmoid(x)$.

Nous avons pour la deuxième sortie donnée par Softmax:

$$
\begin{aligned}
o_2^s &= \frac{\exp o_2^a}{\exp o_1^a + \exp o_2^a} \\
&= \frac{\exp o_2^a - o_1^a}{1 + \exp o_2^a - o_1^a} \\
&= \frac{\exp -x}{1 + \exp -x} \\
&= 1 - \frac{1}{1 + \exp -(x)} \\
&= 1 - o_1^s
\end{aligned}
$$

De plus on a bien $o_1^s + o_2^s = 1$. Cela montre bien que les deux fonctions sont équivalentes.
(e) Soit la fonction de coût $L_{MSE}(\sigma(o^a), y) = \frac{1}{m}\sum_{i=1}^{m}(\sigma(o^a)_i - y_i)^2 = \frac{1}{m}||\sigma(o^a) - y||_2^2$.
Calculons $\frac{\partial L_{MSE}(\sigma(o^a),y)}{\partial o^a} \in \mathbb{R}^{1\times m}$.
Or,

$$
\frac{\partial L_{MSE}(\sigma(o^a), y)}{\partial o^a} = \frac{\partial L_{MSE}(\sigma(o^a), y)}{\partial \sigma(o^a)} \frac{\partial \sigma(o^a)}{\partial o^a}
$$

On a:

$$
\begin{aligned}
\frac{\partial L_{MSE}(\sigma(o^a), y)}{\partial \sigma(o^a)_i} &= \frac{\partial}{\partial \sigma(o^a)_i}(\frac{1}{m}\sum_{k=1}^{m}(\sigma(o^a)_k - y_k)^2) \\
&= \frac{2}{m}(\sigma(o^a)_i - y_i)
\end{aligned}
$$

d'où $\frac{\partial L_{MSE}(\sigma(o^a),y)}{\partial \sigma(o^a)} = \frac{2}{m}(\sigma(o^a) - y)^T \in \mathbb{R}^{1\times m}$
Et d'après les questions précédentes nous savons que $\frac{\partial \sigma(o^a)}{\partial o^a} = diag(\sigma(o^a)) - \sigma(o^a)(\sigma(o^a))^T$.

On a donc dans un cas de classification binaire $\frac{\partial L_{MSE}(\sigma(o^a),y)}{\partial o^a} = (\sigma(o^a) - y)^T(diag(\sigma(o^a)) - \sigma(o^a)(\sigma(o^a))^T)$.

(f) Soit la fonction de coût $L_{CE}(\sigma(o^a), y) = -\sum_{k=1}^{m} y_k \log(\sigma(o^a)_k)$
Calculons $\frac{\partial L_{CE}(\sigma(o^a),y)}{\partial o^a} \in \mathbb{R}^{1\times m}$.

Or,

$$\frac{\partial L_{CE}(\sigma(o^a), y)}{\partial o^a} = \frac{\partial L_{CE}(\sigma(o^a), y)}{\partial \sigma(o^a)} \frac{\partial \sigma(o^a)}{\partial o^a}$$

On a:

$$\frac{\partial L_{CE}(\sigma(o^a), y)}{\partial \sigma(o^a)_i} = \frac{-y_i}{\sigma(o^a)_i}$$

Et d'après les questions précédentes nous savons que $\frac{\partial \sigma(o^a)}{\partial o^a} = diag(\sigma(o^a)) - \sigma(o^a)(\sigma(o^a))^T$.

On a donc dans un cas de classification binaire $\frac{\partial L_{CE}(\sigma(o^a),y)}{\partial o^a} = \begin{bmatrix} \frac{-y_1}{\sigma(o^a)_1} & \frac{-y_2}{\sigma(o^a)_2} \end{bmatrix} (diag(\sigma(o^a)) - \sigma(o^a)(\sigma(o^a))^T)$.

(g) En observant les résultats obtenus, nous pouvons voir que la fonction de cross entropy est ici plus appropriée à notre problème. En effet, le vecteur $(\sigma(o^a) - y)^T$ issu de la dérivée partielle de L_{MSE} peut possèder des valeurs nulles entrainant une dérivée partielle nulle (si nos labels sont 0 et 1 par exemple)ce qui entrainerait une disparition des gradients. Contrairement au vecteur $\begin{bmatrix} \frac{-y_1}{\sigma(o^a)_1} & \frac{-y_2}{\sigma(o^a)_2} \end{bmatrix}$ issu de la dérivée partielle de L_{CE} qui ne risque jamais d'être égal au vecteur nul.

(h) Nous considérons à présent un problème de classification multiclasse. Nous considérons la fonction de perte $L(\mathbf{x}, y) =$ -log $\mathbf{o}^s_y(\mathbf{x})$. Exprimons cette fonction en fonction de $\mathbf{o}^a$.

$$\begin{aligned} L(\mathbf{x}, y) &= \text{-log } \mathbf{o}^s_y(\mathbf{x}) \\ &= -\log S(o^a)_y \\ &= -\log \frac{\exp o^a_y}{\sum_{j=1}^m \exp o^a_j} \\ &= \log \sum_{j=1}^m \exp o^a_j - o^a_y \end{aligned}$$

(i) On pose θ les paramètres du modèle . On a alors comme risque empirique, $\hat{R}(o^s_\theta, D_n) = \frac{1}{n} \sum_{i=1}^n L(x^{(i)}, y^{(i)}) = -\frac{1}{n} \sum_{i=1}^n$ -log $\mathbf{o}^s_i(\mathbf{x^{(i)}})$.

On a alors $\theta = \{W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)}\}$ et donc $|\theta| = n_\theta = d_h \times d + d_h + m \times d_h + m$.

Le problème d'optimisation est donc le suivant:

On cherche $\hat{\theta} = argmin_\theta(\hat{R}(o^s_\theta, D_n))$.

(j) Exprimons la technique de descente de gradient pour ce problème où l'on considère toujours θ comme étant les paramètres du modèle.

On a alors :

$$\theta^{t+1} = \theta^t - \eta \nabla_\theta \hat{R}(o_\theta^s, D_n)$$

avec $t \in$ pour désigner nos epochs, et $\eta \in \mathbb{R}$ le taux d'apprentissage

(k) Nous ajoutons un terme de régularisation à notre risque :

$$\hat{R}_{reg}(\theta) = \hat{R}(\theta) + \lambda \|\theta\|_2^2$$

Nous pouvons réécrire la desente de gradient de la façon suivante:

$$\begin{aligned}
\theta^{t+1} &= \theta^t - \eta \nabla_\theta \hat{R}_{reg}(\theta^t) \\
&= \theta^t - \eta \nabla_\theta (\hat{R}(\theta^t) + \lambda \|\theta^t\|_2^2) \\
&= \theta^t - \eta \nabla_\theta \hat{R}(\theta^t) - 2\eta\lambda\theta^t
\end{aligned}$$

(l)
Démontrons que

$$\frac{\partial L}{\partial \mathbf{o}^a} = \mathbf{o}^s - onehot_m(y)$$

On sait d'après les questions précédentes que :

$$L(\mathbf{x}, y) = \log \sum_{j=1}^{m} \exp o_j^a - o_y^a$$

On a donc:
si $k \neq y$ où $y \in \{1, ..., m\}$:

$$\begin{aligned}
\frac{\partial L}{\partial \mathbf{o}_k^a} &= \frac{\exp o_k^a}{\log \sum_{j=1}^m \exp o_j^a} \\
&= o_k^s - 0
\end{aligned}$$

si $k = y$ où $y \in \{1, ..., m\}$:

$$\begin{aligned}
\frac{\partial L}{\partial \mathbf{o}_y^a} &= \frac{\exp o_k^a}{\log \sum_{j=1}^m \exp o_j^a} - 1 \\
&= o_k^s - 1
\end{aligned}$$

d'où $\frac{\partial L}{\partial \mathbf{o}^a} = \mathbf{o}^s - onehot_m(y) \in \mathbb{R}^m$

(m) Calculons les gradients par rapport aux paramètres $\mathbf{W}^{(2)}$ et $\mathbf{b}^{(2)}$. On a:

$$\frac{\partial L}{\partial \mathbf{W}^{(2)}_{kj}} = \frac{\partial L}{\partial \mathbf{o}^a_k} \frac{\partial \mathbf{o}^a_k}{\partial \mathbf{W}^{(2)}_{kj}}$$

et

$$\frac{\partial L}{\partial \mathbf{b}^{(2)}_k} = \frac{\partial L}{\partial \mathbf{o}^a_k} \frac{\partial \mathbf{o}^a_k}{\partial \mathbf{b}^{(2)}_k}$$

Nous avons déjà calculé $\frac{\partial L}{\partial \mathbf{o}^a_k}$ à la question précédente. Calculons dans un premier temps $\frac{\partial \mathbf{o}^a_k}{\partial \mathbf{W}^{(2)}_{kj}}$:

On sait que $o^a_k = \sum_{i=1}^m \mathbf{W}^{(2)}_{ki} h^s_i + \mathbf{b}^{(2)}_k$

On a donc : $\frac{\partial \mathbf{o}^a_k}{\partial \mathbf{W}^{(2)}_{kj}} = h^s_j$, d'où $\frac{\partial \mathbf{o}^a_k}{\partial \mathbf{W}^{(2)}_{k,:}} = h^s$ et donc $\frac{\partial \mathbf{o}^a_k}{\partial \mathbf{W}^{(2)}} = \begin{bmatrix} | & \cdots & | \\ h^s & \cdots & h^s \\ | & \cdots & | \end{bmatrix} \in \mathbb{R}^{m \times d_h}$

Calculons à présent $\frac{\partial \mathbf{o}^a_k}{\partial \mathbf{b}^{(2)}_k}$:

on a : $\frac{\partial \mathbf{o}^a_k}{\partial \mathbf{b}^{(2)}_k} = 1$ d'où $\frac{\partial \mathbf{o}^a}{\partial \mathbf{b}^{(2)}} = \begin{bmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{bmatrix} \in \mathbb{R}^{m \times m}$

(n) Nous pouvons exprimer le calcul précédent sous forme matricielle. Nous avons alors:

$$\frac{\partial L}{\partial \mathbf{b}^{(2)}} = \frac{\partial \mathbf{o}^a}{\partial \mathbf{b}^{(2)}} \frac{\partial L}{\partial \mathbf{o}^a}$$

où $\frac{\partial L}{\partial \mathbf{b}^{(2)}} \in \mathbb{R}^m$, $\frac{\partial L}{\partial \mathbf{o}^a} \in \mathbb{R}^m$, $\frac{\partial \mathbf{o}^a}{\partial \mathbf{b}^{(2)}} \in \mathbb{R}^{m \times m}$

on a alors $\frac{\partial L}{\partial \mathbf{b}^{(2)}} = \begin{bmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{bmatrix} (\mathbf{o}^s - onehot_m(y))$

$$\frac{\partial L}{\partial \mathbf{W}^{(2)}} = \frac{\partial \mathbf{o}^a}{\partial \mathbf{W}^{(2)}} \frac{\partial L}{\partial \mathbf{o}^a}$$

où $\frac{\partial L}{\partial \mathbf{W}^{(2)}} \in \mathbb{R}^{m \times d_h}$, $\frac{\partial L}{\partial \mathbf{o}^a} \in \mathbb{R}^m$, $\frac{\partial \mathbf{o}^a}{\partial \mathbf{W}^{(2)}} \in \mathbb{R}^{m \times d_h \times m}$

on a alors $\frac{\partial L}{\partial \mathbf{b}^{(2)}} = \begin{bmatrix} \frac{\partial \mathbf{o}^a_1}{\partial \mathbf{W}^{(2)}} & \cdots & \frac{\partial \mathbf{o}^a_m}{\partial \mathbf{W}^{(2)}} \end{bmatrix} (\mathbf{o}^s - onehot_m(y))$

où $\forall k \in \{1, ..., m\}$, $\frac{\partial \mathbf{o}^a_k}{\partial \mathbf{W}^{(2)}} = \begin{bmatrix} | & \cdots & | \\ h^s & \cdots & h^s \\ | & \cdots & | \end{bmatrix} \in \mathbb{R}^{m \times d_h}$

(o) Calculons les gradients par rapport aux paramètres $h^{(s)}$. On a:

$$\frac{\partial L}{\partial \mathbf{h}_j^s} = \sum_{k=1}^{m} \frac{\partial L}{\partial \mathbf{o}_k^a} \frac{\partial \mathbf{o}_k^a}{\partial \mathbf{h}_j^s}$$

Nous connaissons déjà $\frac{\partial L}{\partial \mathbf{o}_k^a}$ des questions précédentes. Calculons donc $\frac{\partial \mathbf{o}_k^a}{\partial \mathbf{h}_j^s}$

On sait que $o_k^a = \sum_{i=1}^{m} \mathbf{W}_{ki}^{(2)} h_i^s + \mathbf{b}_k^{(2)}$
donc $\frac{\partial \mathbf{o}_k^a}{\partial \mathbf{h}_j^s} = \mathbf{W}_{kj}^{(2)}$

d'où

$$\frac{\partial L}{\partial \mathbf{h}_j^s} = \sum_{k=1}^{m} \frac{\partial L}{\partial \mathbf{o}_k^a} \frac{\partial \mathbf{o}_k^a}{\partial \mathbf{h}_j^s} = \sum_{k=1}^{m} \frac{\partial L}{\partial \mathbf{o}_k^a} \mathbf{W}_{kj}^{(2)}$$

De plus $\frac{\partial \mathbf{o}^a}{\partial \mathbf{h}^s} = \mathbf{W}^{(2)}$.

(p) Nous pouvons exprimer le calcul précédent sous forme matricielle. Nous avons alors:

$$\frac{\partial L}{\partial \mathbf{h}^{(s)}} = \frac{\partial \mathbf{o}^a}{\partial \mathbf{h}^{(s)}} \frac{\partial L}{\partial \mathbf{o}^a}$$

où $\frac{\partial L}{\partial \mathbf{h}^{(s)}} \in \mathbb{R}^{d_h}$, $\frac{\partial L}{\partial \mathbf{o}^a} \in \mathbb{R}^m$, $\frac{\partial \mathbf{o}^a}{\partial \mathbf{h}^{(s)}} \in \mathbb{R}^{d_h \times m}$

on a alors $\frac{\partial L}{\partial \mathbf{h}^{(s)}} = \mathbf{W}^{(2)}(\mathbf{o}^s - onehot_m(y))$

(q) Calculons les gradients par rapport aux paramètres $h^{(a)}$ de la couche cachée. On a:

$$\frac{\partial L}{\partial \mathbf{h}_j^a} = \frac{\partial L}{\partial \mathbf{h}_j^s} \frac{\partial \mathbf{h}_j^s}{\partial \mathbf{h}_j^a}$$

Nous connaissons déjà $\frac{\partial L}{\partial \mathbf{h}_j^s}$ de la question précédente. Calculons donc $\frac{\partial \mathbf{h}_j^s}{\partial \mathbf{h}_j^a}$

De manière plus générale, nous avons en posant $\phi(z)$ la fonction softplus :

$$
\begin{aligned}
\frac{\partial \phi(z)}{\partial z} &= \frac{\partial}{\partial z}\log(1+\exp z) \\
&= \frac{\partial \log u}{\partial u}\frac{\partial u}{\partial z} && \text{par composition de fonction et en posant } u = 1+\exp z \\
&= \frac{1}{u}\exp z \\
&= \frac{\exp z}{1+\exp z} \\
&= \frac{1}{1+\exp -z} \\
&= sigmoid(z)
\end{aligned}
$$

d'où appliqué à notre cas en posant $z = h_j^a$, on a $\frac{\partial \mathbf{h}_j^s}{\partial \mathbf{h}_j^a} = sigmoid(\mathbf{h}_j^a)$, ainsi $\frac{\partial \mathbf{h}^s}{\partial \mathbf{h}^a} = diag(sigmoid(\mathbf{h}^a))$.

(r) Nous pouvons exprimer le calcul précédent sous forme matricielle. Nous avons alors:

$$\frac{\partial L}{\partial \mathbf{h}^{(a)}} = \frac{\partial \mathbf{h}^s}{\partial \mathbf{h}^{(a)}}\frac{\partial L}{\partial \mathbf{h}^s}$$

où $\frac{\partial L}{\partial \mathbf{h}^{(a)}} \in \mathbb{R}^{d_h}$, $\frac{\partial L}{\partial \mathbf{h}^s} \in \mathbb{R}^{d_h}$, $\frac{\partial \mathbf{h}^s}{\partial \mathbf{h}^{(a)}} \in \mathbb{R}^{d_h \times d_h}$

D'après la question précédente, $\frac{\partial L}{\partial \mathbf{h}^{(s)}} = \mathbf{W}^{(2)}(\mathbf{o}^s - onehot_m(y)) \in \mathbb{R}^{d_h}$

on a alors $\frac{\partial L}{\partial \mathbf{h}^{(a)}} = diag(sigmoid(\mathbf{h}^a))\mathbf{W}^{(2)}(\mathbf{o}^s - onehot_m(y))$

Réponses Exercice 3

(a) Determinons la dimension de la sortie à la dernière couche.

Après la première couche de convolution notre sortie possède 32 channels dont nous pouvons calculer la dimension:
On rappelle la formule:

$$dim_{sortie} = \lfloor \frac{i+2p-k}{s} \rfloor + 1$$

avec i la dimension d'entrée, p le padding, k la dimension du noyau et s le stride.
On a donc dans notre cas:

$$sortie_1 = \lfloor \frac{128+6-8}{2} \rfloor + 1 = 64$$

Nous avons donc en sortie de la couche 1 une dimension de $64 \times 64 \times 32$

Après la deuxième couche de pooling notre sortie possède toujours 32 channels dont nous pouvons calculer la dimension:

$$sortie_2 = \lfloor \frac{64-2}{2} \rfloor + 1 = 32$$

Nous avons donc en sortie de la couche 2 une dimension de $32 \times 32 \times 32$

Enfin, après la troisème couche de convolution notre sortie possède 64 channels dont nous pouvons calculer la dimension:

$$sortie_3 = \lfloor \frac{32+2-3}{1} \rfloor + 1 = 32$$

Nous avons donc en sortie de la dernière une dimension de $32 \times 32 \times 64$.

(b) Le nombre de paramètres requis pour la dernière couche sans compter les biais est $3 \times 3 \times 64 = 576$.

(c) Toutes ces convolutions sont réalisées à l'aide d'un kernel flipping:

Full Convolution de $\Big[1,1,4,4,4,1,1\Big] * \Big[1/4,1/4,1/4\Big] = \Big[1/4,1/2,3/2,9/4,3,9/4,3/2,1/2,1/4\Big]$

Same Convolution de $\Big[1,1,4,4,4,1,1\Big] * \Big[1/4,1/4,1/4\Big] = \Big[1/2,3/2,9/4,3,9/4,3/2,1/2\Big]$

Valid Convolution de $\Big[1,1,4,4,4,1,1\Big] * \Big[1/4,1/4,1/4\Big] = \Big[3/2,9/4,3,9/4,3/2\Big]$

(d) Nous constatons au vu des résultats que l'opération de convolution permet de lisser nos valeurs (un genre de "smooth data") via le filtre. En effet nos valeurs, proches entre elles possèdent désormais de plus petits écarts de valeurs, nous pouvons considérer ce phénomène comme étant un flou.