IFT 3395 Fondements de l'apprentissage machine
Prof. Ioannis Mitliagkas

Devoir 3 - Partie Théorique

- Ce devoir doit être déposé sur Gradescope et peut-être être fait seul ou par équipe 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** [15 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) [2 points] Ecrivez la dérivée de la fonction d'activation ReLU (Unité de Rectification Linéaire) $rect(x) = \max\{0, x\}$, **partout où elle existe** Notez que la dérivée en 0 n'est pas définie, mais votre fonction rect$'$ peut retourner 0 en 0.

(c) [1 points] Ecrivez la dérivée de la fonction sigmoïde, σ', en utilisant seulement la fonction σ.

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

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

(f) [1 points] On définit la norme L_2 d'un vecteur: $\|\mathbf{x}\|_2^2 = \sum_i \mathbf{x}_i^2$. Ecrivez le gradient du carré de la fonction norme L_2, $\frac{\partial \|\mathbf{x}\|_2^2}{\partial \mathbf{x}}$, en forme vectorielle.

(g) [1 points] On définit la norme L_1 d'un vecteur: $\|\mathbf{x}\|_1 = \sum_i |\mathbf{x}_i|$. Ecrivez le gradient de la fonction norme L_1, $\frac{\partial \|\mathbf{x}\|_1}{\partial \mathbf{x}}$, en forme vectorielle.

(h) [1 points] Montrez que la fonction softmax n'est pas invariante aux multiplications scalaires.On définit $S_c(\mathbf{x}) = S(c\mathbf{x})$ où $c \geq 0$.

(i) [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.

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

(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 de la $i^{ème}$ composante de $\nabla_u \log S(\mathbf{x}(\mathbf{u}))$ est égale à:

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

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

2. **Calcul de gradients pour l'optimisation des paramètres d'un réseau de neurones pour la classification multiclasse** [(bonus 4) 28 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 σ (fonction sigmoide), définie comme suit:

$$\sigma(x) = \frac{1}{1 + e^{-x}}.$$

Notez que la dérivée de $\sigma(x)$ est donnée par $\sigma(x)' = \frac{1}{1+e^{-x}} \cdot \left(1 - \frac{1}{1+e^{-x}}\right) = \sigma(x) \cdot (1 - \sigma(x))$
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 $j^{ème}$ 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 $\mathbf{W}^{(1)}$ la 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 donner l'expression pour juste $\mathbf{o}^a_k$.

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

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

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) [2 points] Nous supposons pour le moment que notre problème est une **classification binaire**. Notez que pour un problème de classification binaire, l'utilisation de l'activation softmax pour la couche de sortie est équivalente à utiliser la fonction sigmoïde, vous pouvez donc utiliser $\sigma(x) = \frac{1}{1+exp(-x)}$. Nous choisissons comme fonction de coût: l'erreur quadratique moyenne *(MSE)*. Donnez l'expression de $L_{MSE}(\sigma(\mathbf{o}^a), y)$ et calculer:

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

(e) [(bonus) 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:

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

(f) [(bonus) 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.

(g) [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 $y^{\text{ième}}$ 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.

(h) [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_θ cela correspond. Formulez le problème d'optimisation qui correspond à l'entraînement du réseau permettant de trouver une valeur optimale des paramètres.

(i) [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

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

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)!

(j) [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

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

(k) [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 $\mathbf{W}^{(2)}$, et 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.

(l) [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 $\mathbf{h}_j^s$ 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:

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

(m) [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.

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

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

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

(o) [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** [(bonus 6) 5 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 zéro. La deuxième couche sous-échantillonne (*downsampling*) la sortie (*output*) de la première couche avec un *max-pool* 5×5 sans chevauchement (*no overlapping*). La troisième couche convolue 128 noyaux 4×4 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?

Supposons que l'on nous donne des données $3 \times 32 \times 32$. Donnez une configuration d'une couche d'un réseau neuronal convolutif qui satisfait les hypothèses spécifiées. Répondre avec la taille du noyau (k), le pas (s), la marge (p), et la dilatation (*dilation* d, en utilisant la convention $d = 1$ pour une convolution sans dilatation). Utilisez des fenêtres carrées seulement (par exemple, même valeur de k pour la hauteur et la largeur). La taille de la sortie de la dernière couche est $(28, 4, 4)$.

(c) [(bonus) 2 points] Supposons qu'on n'utilise ni marge ni dilatation.

(d) [(bonus) 2 points] Supposons que $d = 2$, et que $p = 2$.

(e) [(bonus) 2 points] Supposons que $p = 1$, $d = 1$ et que $s = 2$.