IFT-6390 Fundamentals of Machine Learning
Professor: Ioannis Mitliagkas

Homework 3 - Theoretical part

Devoir 3 - Partie Théorique

- This homework must be done and submitted to Gradescope and should be done in groups of 3 students. You are welcome to discuss with students outside of your group but the solution submitted by a group must be its own. Note that we will use Gradescope's plagiarism detection feature. All suspected cases of plagiarism will be recorded and shared with university officials for further handling.
  Ce devoir doit être déposé sur Gradescope et doit être fait en équipes de 3. Vous pouvez discuter avec des étudiants d'autres groupes mais les réponses soumises par le groupe doivent être originales. A noter que nous utiliserons l'outil de détection de plagiat de Gradescope. Tous les cas suspectés de plagiat seront enregistrés et transmis à l'Université pour vérification.

- Only one student should submit the homwork and add you should add your group member on the submission page on gradescope
  Seulement une étudiant doit soumettre les solutions et vous devez ajouter votre membre d'équipe sur la page de soumission de gradescope

- You need to submit your solution as a pdf file on Gradescope using the homework titled `(6390: GRAD) Theoretical Homework 3`.
  Vous devez soumettre vos solutions au format pdf sur Gradescope en utilisant le devoir intitulé `(6390: GRAD) Theoretical Homework 3`.

1. **Derivatives and relationships between basic functions - Dérivées et relations entre fonctions usuelles** [13 points]

   We define:

   On définit:

   - The "logistic sigmoid" function - La fonction "sigmoïde" : $x \mapsto \sigma(x) = \frac{1}{1+\exp(-x)}$.

- The "hyperbolic tangent" function - La fonction "tangente hyperbolique" : $x \mapsto \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$.
- The "softplus" function - La fonction "softplus" : $x \mapsto \text{softplus}(x) = \ln(1 + \exp(x))$
- The "sign" function $x \mapsto \text{sign}(x)$ which returns +1 if its argument is positive, -1 if negative and 0 if 0. - 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.
- The "indicator" function: $x \mapsto \mathbf{1}_S(x)$ which returns 1 if $x \in S$ (or x respects condition S), and otherwise returns 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.
- The "rectifier" function which keeps only the positive part of its argument: $x \mapsto \text{rect}(x)$ returns x if $x \geq 0$ and returns 0 if $x < 0$. It is also named RELU (rectified linear unit): - 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. $\text{rect}(x) = \text{RELU}(x) = [x]_+ = \max(0, x) = \mathbf{1}_{\{x>0\}}(x)$
- The "diagonal" function: $\boldsymbol{x} \in \mathbb{R}^n \mapsto diag(\boldsymbol{x}) \in \mathbb{R}^{n \times n}$ such that $diag(\boldsymbol{x})_{ij} = \boldsymbol{x}_i$ if $i = j$ and $diag(\boldsymbol{x})_{ij} = 0$ if $i \neq j$. Here $\mathbb{R}^{n \times n}$ is the set of square matrices of size n. - La fonction "diagonale" : $\boldsymbol{x} \in \mathbb{R}^n \mapsto diag(\boldsymbol{x}) \in \mathbb{R}^{n \times n}$ tel que $diag(\boldsymbol{x})_{ij} = \boldsymbol{x}_i$ si $i = j$ et $diag(\boldsymbol{x})_{ij} = 0$ si $i \neq j$. Ici, $\mathbb{R}^{n \times n}$ est l'ensemble des matrices carrées de taille n.
- The "softmax" function $\boldsymbol{x} \in \mathbb{R}^n \mapsto S(\boldsymbol{x}) \in \mathbb{R}^n$ such that $S(\boldsymbol{x})_i = \frac{e^{\boldsymbol{x}_i}}{\sum_j e^{\boldsymbol{x}_j}}$ - La fonction "softmax" : $\boldsymbol{x} \in \mathbb{R}^n \mapsto S(\boldsymbol{x}) \in \mathbb{R}^n$ tel que $S(\boldsymbol{x})_i = \frac{e^{\boldsymbol{x}_i}}{\sum_j e^{\boldsymbol{x}_j}}$.

Please note that in this homework we will sometimes use the partial derivative symbol to differentiate with respect to a vector, $\mathbf{x}$. In those cases, this notation denotes a gradient: $\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = \nabla f(\mathbf{x})$.

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] Show that $\sigma(x) = \frac{1}{2}\left(\tanh(\frac{1}{2}x) + 1\right)$

Montrez que $\sigma(x) = \frac{1}{2}\left(\tanh(\frac{1}{2}x) + 1\right)$

(b) [1 points] Show that $\ln \sigma(x) = -\text{softplus}(-x)$
Montrez que $\ln \sigma(x) = -\text{softplus}(-x)$

(c) [1 points] Write the derivative of the sigmoid function, σ', using the σ function only
Ecrivez la dérivée de la fonction sigmoïde, σ', en utilisant seulement la fonction σ.

(d) [1 points] Write the derivative of the hyperbolic tangent function, $\tanh'$, using the tanh function only Ecrivez la dérivée de la fonction tangente hyperbolique, $\tanh'$, en utilisant seulement la fonction tanh.

(e) [1 points] Write the sign function using only indicator functions: $\text{sign}(x) = \ldots$ Ecrivez la fonction signe, en utilisant seulement des fonctions indicatrices: $\text{sign}(x) = \ldots$.

(f) [1 points] Write the derivative of the absolute value function ($x \mapsto \text{abs}(x) = |x|$), abs'.
Note: its derivative at 0 is not defined, but your function abs' can return 0 at 0.
Note 2: You have to use the sign function.
Ecrivez la dérivée de la fonction valeur absolue, ($x \mapsto \text{abs}(x) = |x|$), abs'.
Notez que la dérivée en 0 n'est pas définie, mais votre fonction abs' peut retourner 0 en 0. Notez aussi que vous devez utiliser la fonction signe dans votre réponse.

(g) [1 points] Write the derivative of the rectifier function, rect'.
Note: its derivative at 0 is undefined, but your function rect' can return 0 at 0.
Note2: You have to use the indicator function in your answer.
Ecrivez la dérivée de la fonction rectificatrice, rect'.
Notez que la dérivée en 0 n'est pas définie, mais votre fonction rect' peut retourner 0 en 0. Notez aussi que vous devez utiliser la fonction indicatrice dans votre réponse.

(h) [1 points] Let the squared L_2 norm of a vector be: $\|\mathbf{x}\|_2^2 = \sum_i \mathbf{x}_i^2$. Write the the gradient of the square of the L_2 norm function, $\frac{\partial \|\mathbf{x}\|_2^2}{\partial \mathbf{x}}$, in vector form.

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.

(i) [1 points] Let the norm L_1 of a vector be: $\|\mathbf{x}\|_1 = \sum_i |\mathbf{x}_i|$. Write the gradient of the L_1 norm function, $\frac{\partial \|\mathbf{x}\|_1}{\partial \mathbf{x}}$, in vector form.

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.

(j) [1 points] Show that the partial derivatives of the softmax function are given by: $\frac{\partial S(\boldsymbol{x})_i}{\partial \boldsymbol{x}_j} = S(\boldsymbol{x})_i \mathbf{1}_{i=j} - S(\boldsymbol{x})_i S(\boldsymbol{x})_j$.

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

(k) [1 points] Express the Jacobian matrix of the softmax function $\frac{\partial S(\boldsymbol{x})}{\partial \boldsymbol{x}}$ using matrix-vector notation. Use the *diag* function.

Remember that $\frac{\partial S(\boldsymbol{x})}{\partial \boldsymbol{x}}$ is a $n \times n$ matrix, and for all $i, j \in \{1, \ldots, n\}$, the (i, j) entry of the matrix is $\left(\frac{\partial S(\boldsymbol{x})}{\partial \boldsymbol{x}}\right)_{i,j} = \frac{\partial S(\boldsymbol{x})_i}{\partial \boldsymbol{x}_j}$.

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

On rappelle que $\frac{\partial S(\boldsymbol{x})}{\partial \boldsymbol{x}}$ 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 $\left(\frac{\partial S(\boldsymbol{x})}{\partial \boldsymbol{x}}\right)_{i,j} = \frac{\partial S(\boldsymbol{x})_i}{\partial \boldsymbol{x}_j}$.

(l) [2 points] Let $\boldsymbol{y}$ and $\boldsymbol{x}$ be $n-$dimensional vectors related by $\boldsymbol{y} = f(\boldsymbol{x})$, L be a differentiable loss function. According to the chain rule of calculus, $\nabla_{\boldsymbol{x}} L = (\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}})^\top \nabla_{\boldsymbol{y}} L$, which takes up $O(n^2)$ computational time in general (as it requires a matrix-vector multiplication).

Show that if $f(\boldsymbol{x}) = \sigma(\boldsymbol{x})$ or $f(\boldsymbol{x}) = S(\boldsymbol{x})$, the above matrix-vector multiplication can be simplified to a $O(n)$ operation.

Note that here, we used the sigmoid function for a vector input $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \in \mathbb{R}^n \mapsto \sigma(\boldsymbol{x}) = (\sigma(\boldsymbol{x}_1), \ldots, \sigma(\boldsymbol{x}_n))$.

Soient $\boldsymbol{y}$ et $\boldsymbol{x}$ deux vecteurs de taille n reliés par $\boldsymbol{y} = f(\boldsymbol{x})$. Soit L une fonction de coût différentiable. Selon le théorème de dérivation des fonctions composées, $\nabla_{\boldsymbol{x}} L = (\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}})^\top \nabla_{\boldsymbol{y}} L$, ce qui nécessite un temps de calcul en $O(n^2)$ en général (étant donné que ça requière la multiplication d'une matrice par un vecteur).

Montrez que si $f(\boldsymbol{x}) = \sigma(\boldsymbol{x})$ ou $f(\boldsymbol{x}) = S(\boldsymbol{x})$, alors la multiplication matrice/vecteur ci-dessus peut être simplifiée à $O(n)$ opérations.

Notez ici que nous utilisons la fonction sigmoïde pour un argument vectoriel: $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \in \mathbb{R}^n \mapsto \sigma(\boldsymbol{x}) = (\sigma(\boldsymbol{x}_1), \ldots, \sigma(\boldsymbol{x}_n))$.

2. **Gradient computation for parameter optimization in a neural net for multiclass classification - Calcul de gradients pour l'optimisation des paramètres d'un réseau de neurones pour la classification multiclasse** [37 points]

Let $D_n = \{(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(n)}, y^{(n)})\}$ be the dataset with $x^{(i)} \in \mathbb{R}^d$ and $y^{(i)} \in \{1, \ldots, m\}$ indicating the class within m classes. **For vectors and matrices in the following equations, vectors are by default considered to be column vectors.**

Consider a neural net of the type Multilayer perceptron (MLP) with only one hidden layer (meaning 3 layers total if we count the input and output layers). The hidden layer is made of d_h neurons fully connected to the input layer. We shall consider a non linearity of type **rectifier**, called **Leaky RELU** with parameter $\alpha < 1$ (Leaky Rectified Linear Unit) for the hidden layer, defined as follows:

$$LeakyRELU_\alpha(x) = \max(x, \alpha x) = \begin{cases} x & \text{if } x \geq 0 \\ \alpha x & \text{otherwise} \end{cases}$$

The output layer is made of m neurons that are fully connected to the hidden layer. They are equipped with a **softmax** non linearity. The output of the j^{th} neuron of the output layer gives a score for the j-th class which can be interpreted as the probability of x being of class j.

It is highly recommended that you draw the neural net as it helps understanding all the steps.

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 de type **rectifieur**, nommée **Leaky RELU** avec paramètre $\alpha < 1$, définie comme suit:

$$LeakyRELU_\alpha(x) = \max(x, \alpha x) = \begin{cases} x & \text{si } x \geq 0 \\ \alpha x & \text{sinon} \end{cases}$$

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 $j^{\text{è me}}$ 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] Let $\mathbf{W}^{(1)}$ be a $d_h \times d$ matrix of weights and $\mathbf{b}^{(1)}$ the bias vector be the connections between the input layer and the hidden layer. What is the dimension of $\mathbf{b}^{(1)}$? Give the formula of the pre-activation vector (before the non linearity) of the neurons of the hidden layer $\mathbf{h}^a$ given $\mathbf{x}$ as input, first in a matrix form ($\mathbf{h}^a = \ldots$), and then details on how to compute one element $\mathbf{h}^a_j = \ldots$. Write the output vector of the hidden layer $\mathbf{h}^s$ with respect to $\mathbf{h}^a$.

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] Let $\mathbf{W}^{(2)}$ be a weight matrix and $\mathbf{b}^{(2)}$a bias vector be the connections between the hidden layer and the output layer. What are the dimensions of $\mathbf{W}^{(2)}$ and $\mathbf{b}^{(2)}$? Give the formula of the activation function of the neurons of the output layer $\mathbf{o}^a$

with respect to their input $\mathbf{h}^s$ in a matrix form and then write in a detailed form for $\mathbf{o}_k^a$.
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 $\mathbf{o}_k^a$.

(c) [2 points] The output of the neurons at the output layer is given by:

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

Give the precise equation for $\mathbf{o}_k^s$ as a function of $\mathbf{o}_j^a$. **Show** that the $\mathbf{o}_k^s$ are positive and sum to 1. Why is this important?
La sortie des neurones de sortie est donnée par

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

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

(d) [2 points] The neural net computes, for an input vector $\mathbf{x}$, a vector of probability scores $\mathbf{o}^s(\mathbf{x})$. The probability, computed by a neural net, that an observation $\mathbf{x}$ belong to class y is given by the y^{th} output $\mathbf{o}_y^s(\mathbf{x})$. This suggests a loss function such as:

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

Find the equation of L as a function of the vector $\mathbf{o}^a$. It is easily achievable with the correct substitution using the equation of the previous question.
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 $\mathbf{o}_y^s(\mathbf{x})$. Ceci suggère d'utiliser la fonction de perte:

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

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.

(e) [2 points] The training of the neural net will consist of finding parameters that minimize the empirical risk $\hat{R}$ associated with this loss function. What is $\hat{R}$? What is precisely the set θ of parameters of the network? How many scalar parameters n_θ are there? Write down the optimization problem of training the network in order to find the optimal values for these parameters.

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.

(f) [2 points] To find a solution to this optimization problem, we will use gradient descent. What is the (batch) gradient descent equation for this problem?

Pour trouver la solution à ce problème d'optimisation, on va utiliser une technique de descente de gradient. Exprimez sous forme d'un bref pseudo-code la technique de descente de gradient (batch) pour ce problème.

(g) [2 points] We can compute the vector of the gradient of the empirical risk $\hat{R}$ with respect to the parameters set θ this way

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

This hints that we only need to know how to compute the gradient of the loss L with an example$(\mathbf{x}, y)$ with respect to the parameters, defined as followed:

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

We shall use **gradient backpropagation**, starting with loss L and going to the output layer $\mathbf{o}$ then down the hidden layer $\mathbf{h}$ then finally at the input layer $\mathbf{x}$.

Show that

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

Note: Start from the expression of L as a function of $\mathbf{o}^a$ that you previously found. Start by computing $\frac{\partial L}{\partial \mathbf{o}^a_k}$ for $k \neq y$ (using the start of the expression of the logarithm derivative). Do the same thing for $\frac{\partial L}{\partial \mathbf{o}^a_y}$.

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 $\frac{\partial L}{\partial \mathbf{o}_y^a}$.

IMPORTANT: From now on when we ask to "compute" the gradients or partial derivatives, you only need to write them as function of previously computed derivatives (do not substitute the whole expressions already computed in the previous questions)!

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)!

(h) [2 points] Compute the gradients with respect to parameters $\mathbf{W}^{(2)}$ and $\mathbf{b}^{(2)}$ of the output layer. Since L depends on $\mathbf{W}_{kj}^{(2)}$ and $\mathbf{b}_k^{(2)}$only through $\mathbf{o}_k^a$ the result of the chain rule is:

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

and

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

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

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

et

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

(i) [2 points] Write down the gradient of the last question in matrix form and define the dimensions of all matrix or vectors involved. (For the gradient with respect to $\mathbf{W}^{(2)}$, we want to arrange the partial derivatives in a matrix that has the same shape as $\mathbf{W}^{(2)}$, and that's what we call the gradient)

What are the dimensions?

Take time to understand why the above equalities are the same as the equations of the last question.

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.

(j) [2 points] What is the partial derivative of the loss L with respect to the output of the neurons at the hidden layer? Since L depends on $\mathbf{h}_j^s$ only through the activations of the output neurons $\mathbf{o}^a$ the chain rule yields:

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

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:

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

(k) [2 points] Write down the gradient of the last question in matrix form and define the dimensions of all matrix or vectors involved.

What are the dimensions?

Take time to understand why the above equalities are the same as the equations of the last question.

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.

(l) [2 points] What is the partial derivative of the loss L with respect to the activation of the neurons at the hidden layer? Since L depends on the activation $\mathbf{h}_j^a$ only through $\mathbf{h}_j^s$ of this neuron, the chain rule gives:

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

Note $\mathbf{h}_j^s = \text{LeakyRELU}_\alpha(\mathbf{h}_j^a)$: the leaky rectifier function is applied element-wise. Start by writing the derivative of the rectifier function $\frac{\partial \text{LeakyRELU}_\alpha(z)}{\partial z} = \text{LeakyRELU}_\alpha{}'(z) = \ldots$.

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 $\mathbf{h}_j^s = \text{LeakyRELU}_\alpha(\mathbf{h}_j^a)$: la fonction leaky rectifieur s'applique élément par élément. Comme étape intermédiaire, commencez par exprimer la dérivée de la fonction rectifieur $\frac{\partial \text{LeakyRELU}_\alpha(z)}{\partial z} = \text{LeakyRELU}_\alpha{}'(z) = \ldots$.

(m) [2 points] Write down the gradient of the last question in matrix form and define the dimensions of all matrix or vectors involved.

Exprimez le calcul de la question précédente sous forme d'une expression matricielle, en définissant la dimension de chacune des matrices ou vecteurs manipulés.

(n) [2 points] What is the gradient with respect to the parameters $\mathbf{W}^{(1)}$ and $\mathbf{b}^{(1)}$ of the hidden layer?

Hint: same logic as a previous question.

Calculez les gradients par rapport aux éléments des paramètres $\mathbf{W}^{(1)}$ et $\mathbf{b}^{(1)}$ de la couche cachée.

Indication: c'est le même principe que pour une question antérieure.

(o) [2 points] Write down the gradient of the last question in matrix form and define the dimensions of all matrix or vectors involved.

Hint: same logic as a previous question.

Exprimez ce calcul du gradient de la question précédente sous forme d'une expression matricielle, en définissant la dimension de chacune des matrices ou vecteurs manipulées.

Indication: c'est le même principe que pour une question antérieure.

(p) [2 points] What are the partial derivatives of the loss L with respect to $\mathbf{x}$?

Hint: same logic as a previous question.

Calculez les dérivées partielles du coût L par rapport au vecteur d'entrée $\mathbf{x}$.

Indication: c'est le même principe que pour une question antérieure.

(q) [3 points] We will now consider a **regularized** empirical risk : $\tilde{R} = \hat{R} + \mathcal{L}(\theta)$, where θ is the vector of all the parameters in the network and $\mathcal{L}(\theta)$ describes a scalar penalty as a function of the parameters θ. The penalty is given importance according to a prior preferences for the values of θ. The L_2 (quadratic) regularization that penalizes the square norm (norm L_2) of the weights (but not the biases) is more standard, is used in ridge regression and is sometimes called "weight-decay". Here we shall consider a double regularization L_2 and L_1 which is sometimes named "elastic net" and we will use different **hyperparameters** (positive scalars $\lambda_{11}, \lambda_{12}, \lambda_{21}, \lambda_{22}$) to control the effect of the regularization at each layer

$$\begin{aligned}
\mathcal{L}(\theta) &= \mathcal{L}(\mathbf{W}^{(1)}, \mathbf{b}^{(1)}, \mathbf{W}^{(2)}, \mathbf{b}^{(2)}) \\
&= \lambda_{11}\|\mathbf{W}^{(1)}\|_1 + \lambda_{12}\|\mathbf{W}^{(1)}\|_2^2 + \lambda_{21}\|\mathbf{W}^{(2)}\|_1 + \lambda_{22}\|\mathbf{W}^{(2)}\|_2^2 \\
&= \lambda_{11}\left(\sum_{i,j}|\mathbf{W}_{ij}^{(1)}|\right) + \lambda_{12}\left(\sum_{ij}(\mathbf{W}_{ij}^{(1)})^2\right) + \lambda_{21}\left(\sum_{i,j}|\mathbf{W}_{ij}^{(2)}|\right) \\
&\qquad + \lambda_{22}\left(\sum_{ij}(\mathbf{W}_{ij}^{(2)})^2\right)
\end{aligned}$$

We will in fact minimize the regularized risk $\tilde{R}$ instead of $\hat{R}$. How does this change the gradient with respect to the different parameters?

Nous allons maintenant considérer un risque empirique **régularisé**: $\tilde{R} = \hat{R} + \mathcal{L}(\theta)$, où θ est le vecteur de tous les paramètres du réseau et $\mathcal{L}(\theta)$ calcule une pénalité scalaire en fonction des paramètres θ, plus ou moins importante selon une préférence à priori qu'on a sur les valeurs de θ. La régularisaiton L_2 (quadratique) qui pénalise la norme carrée (norme L_2) des poids (mais

pas des biais) est la plus classique, est ce qui est urtilisé dans la "régression de ridge" et est parfous appelée "weight-decay" dans les réseaux de neurones. Ici on va considérer une double régularisation L_2 et L_1 qui est parfois nommée "elastic net" et on utilisera des **hyper-parmètres** différents (scalaires positifs $\lambda_{11}, \lambda_{12}, \lambda_{21}, \lambda_{22}$) pour contrôler la force de la régularisation sur chaque couche

$$\begin{aligned}
\mathcal{L}(\theta) &= \mathcal{L}(\mathbf{W}^{(1)}, \mathbf{b}^{(1)}, \mathbf{W}^{(2)}, \mathbf{b}^{(2)}) \\
&= \lambda_{11}\|\mathbf{W}^{(1)}\|_1 + \lambda_{12}\|\mathbf{W}^{(1)}\|_2^2 + \lambda_{21}\|\mathbf{W}^{(2)}\|_1 + \lambda_{22}\|\mathbf{W}^{(2)}\|_2^2 \\
&= \lambda_{11}\left(\sum_{i,j}|\mathbf{W}_{ij}^{(1)}|\right) + \lambda_{12}\left(\sum_{ij}(\mathbf{W}_{ij}^{(1)})^2\right) + \lambda_{21}\left(\sum_{i,j}|\mathbf{W}_{ij}^{(2)}|\right) \\
&\qquad + \lambda_{22}\left(\sum_{ij}(\mathbf{W}_{ij}^{(2)})^2\right)
\end{aligned}$$

On va donc en fait minimiser le risque régularisé $\tilde{R}$ plutôt que $\hat{R}$. Comment cela change-t-il le gradient par rapport aux différents paramètres?