IFT-6390 Fundamentals of Machine Learning
Professor: Ioannis Mitliagkas

Homework 3 - Theoretical part

Students: Maria Fernanda Robles
Laura Sanchez Fernandez
Gustavo Alonso Patino

1. **Derivatives and relationships between basic functions** [13 points]

   We define:

   - The "logistic sigmoid" function : $x \mapsto \sigma(x) = \frac{1}{1+\exp(-x)}$.
   - The "hyperbolic tangent" function: $x \mapsto \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$.
   - The "softplus" function $x \mapsto \text{softplus}(x) = \ln(1 + \exp(x))$
   - The "sign" function $x \mapsto \text{sign}(x)$ which returns +1 if its argument is positive, -1 if negative and 0 if 0. $x \mapsto \text{sign}(x)$
   - 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.
   - 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) $\text{rect}(x) = \text{RELU}(x) = [x]_+ = \max(0, x) = \mathbf{1}_{\{x>0\}}(x)$
   - 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.
   - The "softmax" function $\boldsymbol{x} \in \mathbb{R}^n \mapsto S(\boldsymbol{x}) \in \mathbb{R}^n$ such that $S(\boldsymbol{x})_i = \frac{e^{\boldsymbol{x}_i}}{\sum_j e^{\boldsymbol{x}_j}}$

   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: $\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = \nabla f(\mathbf{x})$.

(a) [1 points] Show that $\sigma(x) = \frac{1}{2}\left(\tanh(\frac{1}{2}x) + 1\right)$
Answer: According to the definition

$$\frac{1}{2}\tanh\left(\frac{1}{2}x + 1\right) = \frac{1}{2}\left(\frac{e^{x/2} - e^{-x/2}}{e^{x/2} + e^{-x/2}} + 1\right) = \frac{e^{x/2}}{e^{x/2} + e^{-x/2}} = \frac{1}{1 + e^{-x}}$$

so we conclude that

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

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

$$\ln \sigma(x) = -\ln(1 + e^{-x}) = -\text{softplus}(-x)$$

(c) [1 points] Write the derivative of the sigmoid function, σ', using the σ function only
Answer:

$$\frac{d\sigma}{dx} = \frac{e^{-x}}{(1 + e^{-x})^2} = \frac{1}{1 + e^{-x}}\left[1 - \frac{1}{1 + e^{-x}}\right] = \sigma(1 - \sigma)$$

(d) [1 points] Write the derivative of the hyperbolic tangent function, $\tanh'$, using the tanh function only
Answer:

$$\frac{d\tanh(x)}{dx} = \frac{(e^x + e^{-x})^2 - (e^x - e^{-x})^2}{(e^x + e^{-x})^2} = 1 - \frac{(e^x - e^{-x})^2}{(e^x + e^{-x})^2} = 1 - \tanh(x)^2$$

(e) [1 points] Write the sign function using only indicator functions: $\text{sign}(x) = \ldots$
Answer:

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

(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.
Answer:

$$\frac{d|x|}{dx} = \text{sign}(x)$$

(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.
Answer:

$$\frac{d\ \mathrm{rect}(x)}{dx} = \mathbf{1}_{x>0}$$

(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.
Answer: Considering that $\|x\|_2^2 = \mathbf{x} \cdot \mathbf{x}$ so

$$\frac{\partial \|x\|_2^2}{\partial \mathbf{x}} = 2\mathbf{x}$$

(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.
Answer:

$$\frac{\partial \|x\|_1}{\partial \mathbf{x}} = \mathrm{sign}(\mathbf{x})$$

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

$$\frac{\partial S(\mathbf{x})_i}{\partial x_j} = \begin{cases} -\frac{e^{x_i}}{\sum_j e^{x_j}} \frac{e^{x_j}}{\sum_j e^{x_j}} = -S(\mathbf{x})_i S(\mathbf{x})_j, \text{ if } i \neq j, \\ \frac{e^{x_i}}{\sum_j e^{x_j}} - \frac{e^{x_i}}{\sum_j e^{x_j}} \frac{e^{x_j}}{\sum_j e^{x_j}} = S(\mathbf{x})_i - S(\mathbf{x})_i S(\mathbf{x})_j, \text{ if } i = j \end{cases}$$

Using the definition of the indicator function we conclude that $\frac{\partial S(\boldsymbol{x})_i}{\partial \boldsymbol{x}_j} = S(\boldsymbol{x})_i \mathbf{1}_{i=j} - S(\boldsymbol{x})_i S(\boldsymbol{x})_j$.

(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 $\left(\frac{\partial S(\boldsymbol{x})}{\partial \boldsymbol{x}}\right)_{i,j} = \frac{\partial S(\boldsymbol{x})_i}{\partial \boldsymbol{x}_j}$.
Answer: By components the equation of the last item can be written as $\frac{\partial S(\mathbf{x})_i}{\partial x_j} = \mathrm{diag}[S(\mathbf{x})]_{ij} - S(\mathbf{x})_i S(\mathbf{x})_j$ which in vector-matrix form is expressed as

$$\frac{\partial S(\boldsymbol{x})}{\partial \boldsymbol{x}} = \mathrm{diag}(S(\boldsymbol{x})) - S(\boldsymbol{x}) \cdot S(\boldsymbol{x})^T$$

(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 $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \in \mathbb{R}^n \mapsto \sigma(\boldsymbol{x}) = (\sigma(\boldsymbol{x}_1), \ldots, \sigma(\boldsymbol{x}_n))$.
Answer: From the last item we know that

$$\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}} = \text{diag}(S(\boldsymbol{x})) - S(\boldsymbol{x}) \cdot S(\boldsymbol{x})^T$$

when $\boldsymbol{y} = S(\boldsymbol{x})$, so the product $(\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}})^\top \nabla_{\boldsymbol{y}} L$ is actually a vector-vector multiplication with computational time $O(n)$

2. **Gradient computation for parameter optimization in a neural net for multiclass classification -** [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.

(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 $\mathbf{h}^a$.
Answer:

- $\mathbf{b}^{(1)} \in \mathbb{R}^{d_h}$
- $\mathbf{h}^a = \mathbf{b}^{(1)} + \mathbf{W}^{(1)} \cdot \mathbf{x}$
- $\mathbf{h}^a_j = \mathbf{b}^{(1)}_j + \mathbf{W}^{(1)}_{j,:} \cdot \mathbf{x} = \mathbf{b}^{(1)}_j + \sum_{i=1}^{d} \mathbf{W}^{(1)}_{j,i} \cdot \mathbf{x}_i$
- $\mathbf{h}^s = LeakyRELU_\alpha(\mathbf{h}^a)$

(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 $\mathbf{o}^a_k$.
Answer:

- $\mathbf{W}^{(2)} \in \mathbb{R}^{m \times d_h}$
- $\mathbf{b}^{(2)} \in \mathbb{R}^{m}$
- $\mathbf{o}^a = \mathbf{b}^{(2)} + \mathbf{W}^{(2)} \cdot \mathbf{h}^s$
- $\mathbf{o}^a_k = \mathbf{b}^{(2)}_k + \mathbf{W}^{(2)}_{k,:} \cdot \mathbf{h}^s = \mathbf{b}^{(2)}_k + \sum_{i=1}^{d_h} \mathbf{W}^{(2)}_{k,i} \cdot \mathbf{h}^s_i$

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

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

Give the precise equation for $\mathbf{o}^s_k$ as a function of $\mathbf{o}^a_j$. **Show** that the $\mathbf{o}^s_k$ are positive and sum to 1. Why is this important?
Answer:

- $\mathbf{o}^s = \text{softmax}(\mathbf{o}^a) = \frac{\exp(\mathbf{o}^a)}{\sum_{j=1}^{m} \exp(\mathbf{o}_j^a)}$

  $\mathbf{o}_k^s = \text{softmax}(\mathbf{o}_k^a) = \frac{\exp(\mathbf{o}_k^a)}{\sum_{j=1}^{m} \exp(\mathbf{o}_j^a)}$

  $\sum_{k=1}^{m} \mathbf{o}_k^s = \sum_{k=1}^{m} \frac{\exp(\mathbf{o}_k^a)}{\sum_{j=1}^{m} \exp(\mathbf{o}_j^a)} = \frac{1}{\sum_{j=1}^{m} \exp(\mathbf{o}_j^a)} \sum_{k=1}^{m} \exp(\mathbf{o}_k^a) = 1$

  Since $\mathbf{o}_k^s$ is positive for all k (by definition of the exponential), and they sum to one, this allows us to interpret the output of the network as the vector of the probabilities of x belonging to each class.
  In addition to that, we can interpret W_{ij} as the nature of the interaction between pairs of variables, and b_i indicates the inclination of a given x_i to take a value of 1.

(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.
Answer:

$$\begin{aligned} L(\mathbf{x}, y) &= -\log \mathbf{o}_y^s(\mathbf{x}) \\ &= -\log \left(\frac{\exp(\mathbf{o}_y^a(\mathbf{x}))}{\sum_{j=1}^{m} \exp(\mathbf{o}_j^a(\mathbf{x}))} \right) \\ &= -\mathbf{o}_y^a(\mathbf{x}) + \log \left(\sum_{j=1}^{m} \exp(\mathbf{o}_j^a(\mathbf{x})) \right) \end{aligned}$$

(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.
Answer:

- $\hat{R} = -\frac{1}{n}\sum_{i=1}^{n} \log \mathbf{o}^s_{y^{(i)}}(\mathbf{x}^{(i)})$
  $= -\frac{1}{n}\sum_{i=1}^{n} \left[\mathbf{o}^a_{y^{(i)}}(\mathbf{x}^{(i)}) - \log\left(\sum_{j=1}^{m} \exp\left(\mathbf{o}^a_j(\mathbf{x}^{(i)})\right)\right)\right]$
  $= -\frac{1}{n}\sum_{i=1}^{n} \left[\mathbf{b}^{(2)}_{y^{(i)}} + \mathbf{W}^{(2)}_{y^{(i)},:}\mathbf{h}^s(\mathbf{x}^{(i)}) - \log\left(\sum_{j=1}^{m} \exp\left(\mathbf{b}^{(2)}_{y^{(i)}} + \mathbf{W}^{(2)}_{y^{(i)},:}\mathbf{h}^s(\mathbf{x}^{(i)})\right)\right)\right]$
  $= -\frac{1}{n}\sum_{i=1}^{n} \left[\mathbf{b}^{(2)}_{y^{(i)}} + \mathbf{W}^{(2)}_{y^{(i)},:}\text{LeakyRELU}(\mathbf{b}^{(1)}_{y^{(i)}} + \mathbf{W}^{(1)}_{y^{(i)},:} \cdot \mathbf{x}^{(i)})\right]$
  $+\frac{1}{n}\sum_{i=1}^{n} \left[\log\left(\sum_{y^{(i)}=1}^{m} \exp\left(\mathbf{b}^{(2)}_{y^{(i)}} + \mathbf{W}^{(2)}_{y^{(i)},:}\text{LeakyRELU}(\mathbf{b}^{(1)}_{y^{(i)}} + \mathbf{W}^{(1)}_{y^{(i)},:} \cdot \mathbf{x}^{(i)})\right)\right)\right]$
- $\theta = \{\mathbf{W}^{(1)} \in \mathbb{R}^{d_h \times d}, \mathbf{b}^{(1)} \in \mathbb{R}^{d_h}, \mathbf{W}^{(2)} \in \mathbb{R}^{m \times d_h}, \mathbf{b}^{(2)} \in \mathbb{R}^{m}\}$
- $n_\theta = d_h(d+1) + m(d_h+1)$
- $\theta^* = \underset{\mathbf{W}^{(1)} \in \mathbb{R}^{d_h \times d}, \mathbf{b}^{(1)} \in \mathbb{R}^{d_h}, \mathbf{W}^{(2)} \in \mathbb{R}^{m \times d_h}, \mathbf{b}^{(2)} \in \mathbb{R}^{m}}{\text{argmin}} \hat{R}$

(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?
Answer:

$$\theta_{t+1} = \theta_t - \eta\left(-\frac{1}{n}\sum_{i=1}^{batch_size} \frac{\partial}{\partial\theta} \log \mathbf{o}^s_y(\mathbf{x})\right)$$

(g) [3 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 $\frac{\partial L}{\partial \mathbf{o}^a_y}$.

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)!

Answer:

$$\begin{aligned}
&\frac{\partial L}{\mathbf{o}^a_k} = \frac{\partial}{\mathbf{o}^a_k}\left[-\mathbf{o}^a_y(\mathbf{x}) + \log\left(\sum_{j=1}^m \exp\left(\mathbf{o}^a_j(\mathbf{x})\right)\right)\right] \\
&= -\frac{\partial}{\partial \mathbf{o}^a_k}(\mathbf{o}^a_y(\mathbf{x})) + \frac{1}{\sum_{j=1}^m \exp\left(\mathbf{o}^a_j(\mathbf{x})\right)} \sum_{j=1}^m \exp\left(\mathbf{o}^a_j(\mathbf{x})\right) \cdot \frac{\partial}{\partial \mathbf{o}^a_k}\mathbf{o}^a_j(\mathbf{x}) \\
&= -\frac{\partial}{\partial \mathbf{o}^a_k}(\mathbf{o}^a_y(\mathbf{x})) + \frac{1}{\sum_{j=1}^m \exp\left(\mathbf{o}^a_j(\mathbf{x})\right)} \exp\left(\mathbf{o}^a_k(\mathbf{x})\right) \\
&= -\frac{\partial}{\partial \mathbf{o}^a_k}(\mathbf{o}^a_y(\mathbf{x})) + \mathbf{o}^s_k(\mathbf{x})
\end{aligned}$$

Then we have two cases:

i) if $k \neq y$: $\frac{\partial L}{\mathbf{o}^a_k} = -0 + \mathbf{o}^s_k(\mathbf{x}) = \mathbf{o}^s_k(\mathbf{x})$

ii) if $k = y$: $\frac{\partial L}{\mathbf{o}^a_k} = -1 + \mathbf{o}^s_y(\mathbf{x}) = \mathbf{o}^s_y(\mathbf{x}) - 1$

Then by definition of onehot, adding to previous calculus, we can get that:

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

(h) [3 points] Compute the gradients with respect to parameters $\mathbf{W}^{(2)}$ and $\mathbf{b}^{(2)}$ of the output layer. Since L depends on $\mathbf{W}^{(2)}_{kj}$ and $\mathbf{b}^{(2)}_k$only through $\mathbf{o}^a_k$ the result of the chain rule is:

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

and

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

Answer:

$$\frac{\partial L}{\partial \mathbf{W}_{kj}^{(2)}} = \frac{\partial L}{\partial \mathbf{o}_k^a} \frac{\partial \mathbf{o}_k^a}{\partial \mathbf{W}_{kj}^{(2)}} = \frac{\partial L}{\partial \mathbf{o}_k^a} \frac{\partial (W_k^{(2)} \mathbf{h}^s + \mathbf{b}_k^{(2)})}{\partial \mathbf{W}_{kj}^{(2)}} = \frac{\partial L}{\partial \mathbf{o}_k^a} \mathbf{h}_j^s$$

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

(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.

Answer:

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

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

Where:

$$\frac{\partial L}{\partial \mathbf{w}^{(2)}} \in \mathbb{R}^{m \times d_h}, \frac{\partial L}{\partial \mathbf{b}^{(2)}} \in \mathbb{R}^m, \frac{\partial L}{\partial \mathbf{o}^a} \in \mathbb{R}^m \ \ and \ \ \mathbf{h}^s \in \mathbb{R}^{d_h}$$

.

(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:

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

Answer:

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

(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.
Answer:

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

Where:

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

(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 $\frac{\partial \text{LeakyRELU}_\alpha(z)}{\partial z} = \text{LeakyRELU}_\alpha{}'(z) = \ldots$.
Answer:

$$\frac{\partial \text{LeakyRELU}_\alpha(z)}{\partial z} = \text{LeakyRELU}_\alpha{}'(z) = \begin{cases} 1 & \text{if } \mathbf{z} \geq 0 \\ \alpha & \text{otherwise} \end{cases}$$

$$\frac{\partial L}{\partial \mathbf{h}_j^a} = \frac{\partial L}{\partial \mathbf{h}_j^s} \frac{\partial \mathbf{h}_j^s}{\partial \mathbf{h}_j^a} = \frac{\partial L}{\partial \mathbf{h}_j^s} \frac{\text{LeakyRELU}_\alpha(z)}{\partial \mathbf{h}_j^a} = \begin{cases} \frac{\partial L}{\partial \mathbf{h}_j^s} & \text{if } \mathbf{h}_j^a \geq 0 \\ \frac{\partial L}{\partial \mathbf{h}_j^s} \cdot \alpha & \text{otherwise} \end{cases}$$

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

$$\frac{\partial L}{\partial \mathbf{h}^a} = \begin{cases} \frac{\partial L}{\partial \mathbf{h}^s} * 1 & \text{if } \mathbf{h}^a \geq 0 \\ \frac{\partial L}{\partial \mathbf{h}^s} * \alpha & \text{otherwise} \end{cases}$$

Where * is a term to term multiplication,or also called element wise product,and $\frac{\partial L}{\partial \mathbf{h}^s} \in \mathbb{R}^{d_h}$, $[1, 1, ...1, \alpha, ...\alpha] \in \mathbb{R}^{d_h}$, this means, dimensions d_h in both cases, then are vectors of dimension d_h.

(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.
Answer:

$$\frac{\partial L}{\partial \mathbf{W}^{(1)}_{kj}} = \frac{\partial L}{\partial \mathbf{h}^a_k} \frac{\partial \mathbf{h}^a_k}{\partial \mathbf{W}^{(1)}_{kj}} = \frac{\partial L}{\partial \mathbf{h}^a_k} \frac{\partial (W^{(1)}_k \mathbf{x} + \mathbf{b}^{(1)}_k)}{\partial \mathbf{W}^{(1)}_{kj}} = \frac{\partial L}{\partial \mathbf{h}^a_k} \mathbf{x}_j$$

$$\frac{\partial L}{\partial \mathbf{b}^{(1)}_k} = \frac{\partial L}{\partial \mathbf{h}^a_k} \frac{\partial \mathbf{h}^a_k}{\partial \mathbf{b}^{(1)}_k} = \frac{\partial L}{\partial \mathbf{h}^a_k} \frac{\partial (W^{(1)}_k \mathbf{x} + \mathbf{b}^{(1)}_k)}{\partial \mathbf{b}^{(1)}_k} = \frac{\partial L}{\partial \mathbf{h}^a_k}$$

(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.
Answer:

$$\frac{\partial L}{\partial \mathbf{W}^{(1)}} = \frac{\partial L}{\partial \mathbf{h}^a} \otimes \mathbf{x} = \frac{\partial L}{\partial \mathbf{h}^a} (\mathbf{x})^T$$

$$\frac{\partial L}{\partial \mathbf{b}^{(1)}} = \frac{\partial L}{\partial \mathbf{h}^a}$$

Where $\frac{\partial L}{\partial \mathbf{W}^{(1)}} \in \mathbb{R}^{d_h \times d_h}$, $\frac{\partial L}{\partial \mathbf{b}^{(1)}} \in \mathbb{R}^{d_h}$, $\frac{\partial L}{\partial \mathbf{h}^a} \in \mathbb{R}^{d_h}$ *and* $\mathbf{x} \in \mathbb{R}^{d_h}$

(p) [2 points] What are the partial derivatives of the loss L with respect to $\mathbf{x}$?
Hint: same logic as a previous question.
Answer:

$$\frac{\partial L}{\partial \mathbf{x}_i} = \sum_{j=1}^{d_h} \frac{\partial L}{\partial \mathbf{h}^a_j} \frac{\partial \mathbf{h}^a_j}{\partial \mathbf{x}_i} = \sum_{j=1}^{d_h} \frac{\partial L}{\partial \mathbf{h}^a_j} \frac{\partial (W^{(1)}_k \mathbf{x}_i + \mathbf{b}^{(1)}_j)}{\partial \mathbf{x}_i} = \sum_{j=1}^{d_h} \frac{\partial L}{\partial \mathbf{h}^a_j} \mathbf{W}^{(1)}_{ji}$$

$$\frac{\partial L}{\partial \mathbf{x}} = (\mathbf{W}^{(1)})^T \frac{\partial L}{\partial \mathbf{h}^a}$$

(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

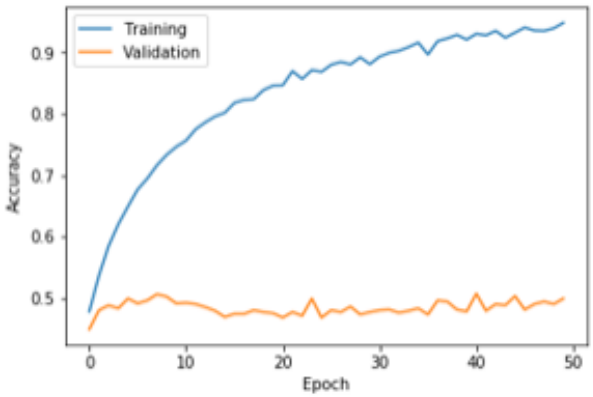
$$\begin{aligned}
\mathcal{L}(\theta) &= \mathcal{L}(\mathbf{W}^{(1)}, \mathbf{b}^{(1)}, \mathbf{W}^{(2)}, \mathbf{b}^{(2)}) \\
&= \lambda_{11}\|\mathbf{W}^{(1)}\|_1 + \lambda_{12}\|\mathbf{W}^{(1)}\|_2^2 + \lambda_{21}\|\mathbf{W}^{(2)}\|_1 + \lambda_{22}\|\mathbf{W}^{(2)}\|_2^2 \\
&= \lambda_{11}\left(\sum_{i,j}|\mathbf{W}_{ij}^{(1)}|\right)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

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?

Answer:

From 1.h) and 1.i) results, we can conclude that :

- for $W^{(1)}$ the gradient will be "increased" by $\lambda_{11} \cdot \text{sign}(\mathbf{W}^{(1)}) + 2\lambda_{12} \cdot \mathbf{W}^{(1)}$.
- for $W^{(2)}$ the gradient will be "increased" by $\lambda_{21} \cdot \text{sign}(\mathbf{W}^{(2)}) + 2\lambda_{22} \cdot \mathbf{W}^{(2)}$

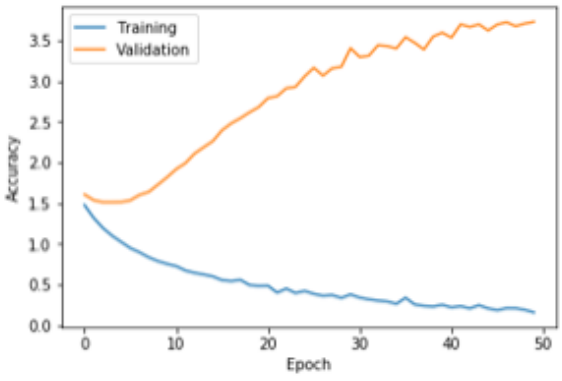


1 Trying your implementation on the CIFAR-10 dataset

1.1

Include in your report the two following figures:

- A figure containing the evolution of both the training and validation accuracies during training
- A figure containing the evolution of both the training and validation losses during training.



1.2

1. Explain in no more than two sentences why the performances on the validation set are not as good as on the training set.

   The model overfitted. The Neural Network is fitting the data very accuratelly, not being prepare anymore fot the unknown data.

2. How could the performance gap be lowered? Make two propositions in the form of short bullet points.

   Apply early stopping method.

   Apply Dropout method.

3. How many learnable parameters (scalars) does the previous neural network have ?

   Knowing that the number of weights in a layer, equals the number of inputs times the number of outputs, and the number of biases equals the number of outputs, we find:

$$n_W 1 = (3072)(512) = 1,572,864 \tag{1}$$

$$n_b 1 = 512 \tag{2}$$

$$n_W 2 = (512)(256) = 131,072 \tag{3}$$

$$n_b 2 = 256 \tag{4}$$

$$n_W 3 = (256)(10) = 2,560 \tag{5}$$

$$n_b 3 = 10 \tag{6}$$

   Being the total of parameters the sum of all weights and biases.

$$Tot_{par} = 1,707,264 \tag{7}$$

1.3

- Find *n_hidden* such that the new network has a number of parameters that is as close as possible to the number of parameters of the initial neural network. Include this number, along with your reasoning, in the report.

We define the following function:

$$1,707,264 \simeq (3075)(512)+512+(512)(120)+120+x(120)(120)+x(120)+120(10)+10 \tag{8}$$

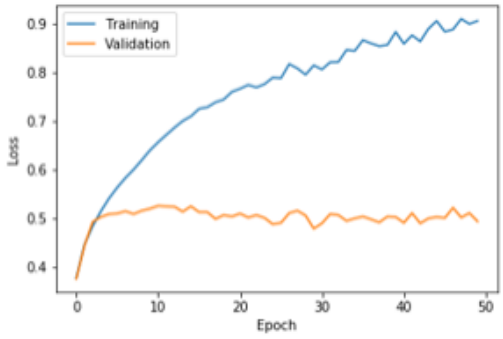
Where, we use the concept of the number of weights in a layer, equals the number of inputs times the number of outputs, and the number of biases equals the number of outputs. The value x is the number of hidden layers with 120 neurons. If we isolate x, we obtain the number of layers with 120 neurons that will give the model aproximatelly 1,707,264 parameters to learn.

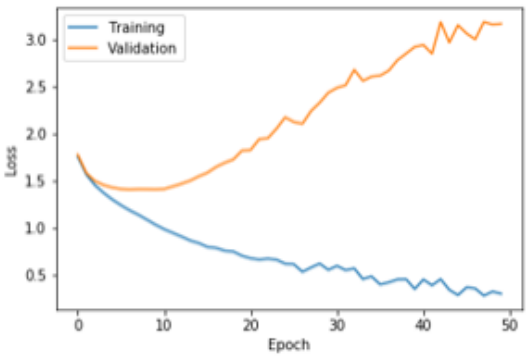
With algebraical tools, this x was valued in:

$$x \simeq 4.91 \tag{9}$$

Then, we choose to have 4 layers with 120 neurons, having as hidden layers: (512, 120, 120, 120, 120, 120).

- Train the new network with the same parameters as in the first question (same learning rate, batch size, activation function, and same seed). Include in your report two figures containing the training/validation accuracies and losses, similar to the first question.





1.4

- Why isn't it sufficient to visually compare the figures of the previous question to the figures of the first question to decide which neural network performs best ?

  Because we are not considering the precision or the variance of the models.

- Train both neural networks for 50 epochs, using 3 different seeds of your choice (you need to report them), with the same parameters as in the first question (you thus have to train 6 networks). Include in your report one figure containing the average training and validation accuracies of both neural networks during training, along with error bars corresponding to the standard deviations you obtain in the 3 different runs multiplied by a factor of your choice **that you need to specify in your report**

  Seeds used: 1, 2, 3.

  Factor of error: 2.

Figure 1: This plot shows the resulting plot with (512, 120, 120, 120, 120, 120) hidden layers.

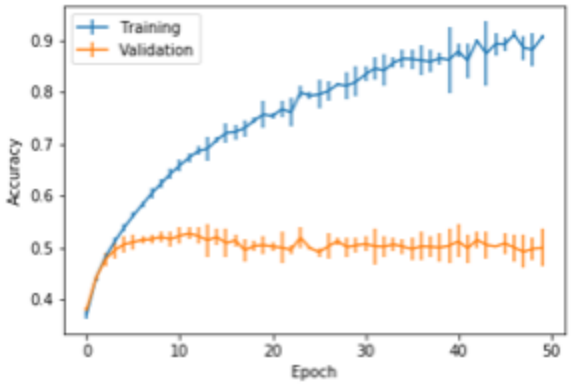


Figure 2: This plot shows the resulting plot with (512, 256) hidden layers.