IFT-6390 Fundamentals of Machine Learning
Professor: Ioannis Mitliagkas

Homework 3 - Theoretical part - Correction

Answers (slightly modified) from Rahim Kassanaly, Andre Al-Khoury, David Whipps, David Bieber, Rajkumar Nitarshan

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

$$
\begin{aligned}
\sigma(x) &= \frac{1}{1+\exp(-x)} \\
&= \frac{e^x}{e^x+1} \\
\tanh(x) &= \frac{e^x - e^{-x}}{e^x + e^{-x}} \\
\tanh(\frac{1}{2}x) + 1 &= \frac{e^{\frac{1}{2}x} - e^{-\frac{1}{2}\ x}}{e^{\frac{1}{2}x} + e^{-\frac{1}{2}\ x}} + 1 \\
&= \frac{e^x - 1}{e^x + 1} + 1 \\
&= \frac{e^x - 1}{e^x + 1} + \frac{e^x + 1}{e^x + 1} \\
&= \frac{2e^x}{e^x + 1} \\
\text{So } \sigma(x) &= \frac{1}{2}\left(\tanh(\frac{1}{2}x) + 1\right).
\end{aligned}
$$

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

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

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

$$
\begin{aligned}
\sigma(x) &= \frac{1}{1+\exp(-x)} \\
\frac{d\sigma(x)}{dx} &= -\left(1+\exp(-x)\right)^{-2} \cdot \exp(-x) \cdot -1 \text{ by the chain rule} \\
&= \sigma(x) \cdot \frac{\exp(-x)}{1+\exp(-x)} \\
&= \sigma(x) \cdot \frac{1}{\exp(x)+1} \\
&= \sigma(x) \cdot \sigma(-x)
\end{aligned}
$$

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

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

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

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

(f) [1 points] Write the derivative of the absolute value function ($x \mapsto \text{abs}(x) = |x|$), abs'.

$$
\begin{aligned}
\text{abs}(x) &= \begin{cases} -x & x \leq 0 \\ x & x \geq 0 \end{cases} \\
\text{abs}'(x) &= \begin{cases} -1 & x < 0 \\ 0 & x = 0 \\ 1 & x > 0 \end{cases} \\
&= \text{sign}(x).
\end{aligned}
$$

(g) [1 points] Write the derivative of the rectifier function, rect'.

$$
\begin{aligned}
\text{rect}(x) &= \begin{cases} 0 & x < 0 \\ x & x \geq 0 \end{cases} \\
\text{rect}'(x) &= \mathbf{1}_{x>0}(x)
\end{aligned}
$$

(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.

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

$$\frac{\partial \|\mathbf{x}\|_2^2}{\partial \mathbf{x}} = \begin{bmatrix} 2x_1 \\ 2x_2 \\ \cdots \\ 2x_n \end{bmatrix} = 2\mathbf{x}$$

(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.

$$\frac{\partial \|\mathbf{x}\|_1}{\partial \mathbf{x}_i} = \text{sign}(x_i)$$

$$\frac{\partial \|\mathbf{x}\|_1}{\partial \mathbf{x}} = \begin{bmatrix} \text{sign}(x_1) \\ \text{sign}(x_2) \\ \cdots \\ \text{sign}(x_n) \end{bmatrix}$$

(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.

$$\frac{\partial S(\boldsymbol{x})}{\partial \boldsymbol{x}_j} = \begin{bmatrix} \frac{\partial S(\boldsymbol{x})_1}{\partial \boldsymbol{x}_j} & \frac{\partial S(\boldsymbol{x})_2}{\partial \boldsymbol{x}_j} & \cdots & \frac{\partial S(\boldsymbol{x})_n}{\partial \boldsymbol{x}_j} \end{bmatrix}$$

when $i = j$, we have

$$\frac{\partial S(\boldsymbol{x})_i}{\partial \boldsymbol{x}_j} = \frac{\partial}{\partial \boldsymbol{x}_j} \frac{e^{\boldsymbol{x}_i}}{\sum\limits_k e^{\boldsymbol{x}_k}} = \frac{e^{\boldsymbol{x}_i} \sum\limits_k e^{\boldsymbol{x}_k} - e^{\boldsymbol{x}_i} e^{\boldsymbol{x}_i}}{\left(\sum\limits_k e^{\boldsymbol{x}_k}\right)^2} = S(\boldsymbol{x})_i - S^2(\boldsymbol{x})_i = S(\boldsymbol{x})_i - S(\boldsymbol{x})_i S(\boldsymbol{x})_j$$

otherwise, we have

$$\frac{\partial S(\boldsymbol{x})_i}{\partial \boldsymbol{x}_j} = \frac{\partial}{\partial \boldsymbol{x}_j} \frac{e^{\boldsymbol{x}_i}}{\sum\limits_k e^{\boldsymbol{x}_k}} = \frac{0 - e^{\boldsymbol{x}_i} e^{\boldsymbol{x}_j}}{\left(\sum\limits_k e^{\boldsymbol{x}_k}\right)^2} = \frac{-e^{\boldsymbol{x}_i} e^{\boldsymbol{x}_j}}{\left(\sum\limits_k e^{\boldsymbol{x}_k}\right)^2} = -S(\boldsymbol{x})_i S(\boldsymbol{x})_j$$

The above results can be combined using the indicator function in the following way:

$$\frac{\partial S(\boldsymbol{x})_i}{\partial \boldsymbol{x}_j} = S(\boldsymbol{x})_i \mathbf{1}_{i=j} - S(\boldsymbol{x})_i S(\boldsymbol{x})_j$$

(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. Given that

$$\frac{\partial S(\boldsymbol{x})_i}{\partial \boldsymbol{x}_j} = S(\boldsymbol{x})_i \mathbf{1}_{i=j} - S(\boldsymbol{x})_i S(\boldsymbol{x})_j$$

Then the diagonal elements of $\frac{\partial S(\boldsymbol{x})}{\partial \boldsymbol{x}}$ (i.e. those with $i = j$) are

$$\left(\frac{\partial S(\boldsymbol{x})}{\partial \boldsymbol{x}}\right)_{i=j} = S(\boldsymbol{x})_i - S(\boldsymbol{x})_i S(\boldsymbol{x})_j$$

while the off-diagonal elements are

$$\left(\frac{\partial S(\boldsymbol{x})}{\partial \boldsymbol{x}}\right)_{i \neq j} = -S(\boldsymbol{x})_i S(\boldsymbol{x})_j$$

Based on the above observations we can write

$$\begin{aligned}\frac{\partial S(\boldsymbol{x})}{\partial \boldsymbol{x}} &= diag(S(\boldsymbol{x})) - \begin{bmatrix} S(\boldsymbol{x})_1 \\ S(\boldsymbol{x})_2 \\ \vdots \\ S(\boldsymbol{x})_n \end{bmatrix} \begin{bmatrix} S(\boldsymbol{x})_1 & S(\boldsymbol{x})_2 & \ldots & S(\boldsymbol{x})_n \end{bmatrix} \\ &= diag(S(\boldsymbol{x})) - S(\boldsymbol{x}) S(\boldsymbol{x})^\top\end{aligned}$$

(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.

Consider $\boldsymbol{y} = \sigma(\boldsymbol{x}) = \begin{bmatrix} \sigma(\boldsymbol{x}_1) \\ \sigma(\boldsymbol{x}_2) \\ \vdots \\ \sigma(\boldsymbol{x}_n) \end{bmatrix}$. The Jacobian of $f(x) = \sigma(\boldsymbol{x})$ is diagonal and given by

$$(\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}}) = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_n}{\partial x_1} & \cdots & \frac{\partial y_n}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \sigma(\boldsymbol{x}_1)(1 - \sigma(\boldsymbol{x}_1)) & & \\ & \ddots & \\ & & \sigma(\boldsymbol{x}_n)(1 - \sigma(\boldsymbol{x}_n)) \end{bmatrix}$$

$\nabla_{\boldsymbol{x}} L = (\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}})^\top \nabla_{\boldsymbol{y}} L$ then reduces to

$$\nabla_{\boldsymbol{x}} L = (\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}})^\top \nabla_{\boldsymbol{y}} L = \begin{bmatrix} \sigma(\boldsymbol{x}_1)(1-\sigma(\boldsymbol{x}_1)) & & \\ & \ddots & \\ & & \sigma(\boldsymbol{x}_n)(1-\sigma(\boldsymbol{x}_n)) \end{bmatrix} \begin{bmatrix} (\nabla_{\boldsymbol{y}} L)_1 \\ (\nabla_{\boldsymbol{y}} L)_2 \\ \vdots \\ (\nabla_{\boldsymbol{y}} L)_n \end{bmatrix}$$

$$= \begin{bmatrix} \sigma(\boldsymbol{x}_1)(1-\sigma(\boldsymbol{x}_1)) \left(\nabla_{\boldsymbol{y}} L\right)_1 \\ \sigma(\boldsymbol{x}_2)(1-\sigma(\boldsymbol{x}_2)) \left(\nabla_{\boldsymbol{y}} L\right)_2 \\ \vdots \\ \sigma(\boldsymbol{x}_n)(1-\sigma(\boldsymbol{x}_n)) \left(\nabla_{\boldsymbol{y}} L\right)_n \end{bmatrix}$$

with $O(n)$ time complexity.
As for the case of $\boldsymbol{y} = S(\boldsymbol{x})$, we have

$$\begin{aligned} \nabla_{\boldsymbol{x}} L = (\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}})^\top \nabla_{\boldsymbol{y}} L &= \left(diag(S(\boldsymbol{x})) - S(\boldsymbol{x}) S(\boldsymbol{x})^\top\right)^\top \nabla_{\boldsymbol{y}} L \\ &= \left(diag(S(\boldsymbol{x})) - S(\boldsymbol{x}) S(\boldsymbol{x})^\top\right) \nabla_{\boldsymbol{y}} L \\ &= diag(S(\boldsymbol{x}))\nabla_{\boldsymbol{y}} L - S(\boldsymbol{x}) S(\boldsymbol{x})^\top \nabla_{\boldsymbol{y}} L \\ &= diag(S(\boldsymbol{x}))\nabla_{\boldsymbol{y}} L - S(\boldsymbol{x}) \left(S(\boldsymbol{x})^\top \nabla_{\boldsymbol{y}} L\right) \end{aligned}$$

$$\begin{aligned} &\underset{1\times n}{S(\boldsymbol{x})^\top} \underset{n\times 1}{\nabla_{\boldsymbol{y}} L} \quad \text{is } O(n) \\ &\underset{n\times 1}{S(\boldsymbol{x})} \underset{1\times 1}{\left(S(\boldsymbol{x})^\top \nabla_{\boldsymbol{y}} L\right)} \quad \text{is } O(n) \\ &diag(S(\boldsymbol{x}))\nabla_{\boldsymbol{y}} L \quad \text{is } O(n) \end{aligned}$$

So the overall time complexity is $O(n)$.

2. **Gradient computation for parameter optimization in a neural net for multi-class 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.

(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 $\mathbf{h}^a$.

The dimension of $b^{(1)}$ is d_h.

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

To compute $\mathbf{h}^a_j$, perform the dot product of the j^{th} row of $\mathbf{W}^{(1)}$ with $\boldsymbol{x}$ and add $\mathbf{b}^{(1)}_j$.

$$\mathbf{h}^a_j = \mathbf{W}^{(1)}_{j:}\boldsymbol{x} + \mathbf{b}^{(1)}_j = \mathbf{b}^{(1)}_j + \sum_{i=1}^{d} \mathbf{W}^{(1)}_{ji}\boldsymbol{x}_i$$
$$\mathbf{h}^s = LeakyRELU_\alpha(\mathbf{h}^a)$$

(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 $\mathbf{o}^a_k$.

$\mathbf{W}^{(2)}$ has dimensions (m, d_h) and $\mathbf{b}^{(2)}$ has dimension m.

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

$$\mathbf{o}^a_k = \mathbf{W}^{(2)}_{k:}\mathbf{h}^s + \mathbf{b}^{(2)}_k = \mathbf{b}^{(2)}_k + \sum_{i=1}^{d_h} \mathbf{W}^{(2)}_{ki}\mathbf{h}^s_i$$

(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 $\mathbf{o}^s_k$ 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?

$$\mathbf{o}^s_k = \frac{e^{\mathbf{o}^a_k}}{\sum_{j=1}^{m} e^{\mathbf{o}^a_j}}$$

Since $e^x > 0$, it is evident that $\mathbf{o}_k^s > 0$.

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

It is important that the $\mathbf{o}_k^s$ terms sum to one and are non-negative because they represent a probability distribution over the m output classes.

(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.

$$L(\mathbf{x}, y) = \text{-log} \frac{e^{\mathbf{o}_y^a(x)}}{\sum_{j=1}^{m} e^{\mathbf{o}_j^a(x)}}$$

(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.

$$\hat{R} = \frac{1}{n} \sum_{i=1}^{N} L(x^{(i)}, y^{(i)}) = \frac{1}{n} \sum_{i=1}^{N} \text{-log} \frac{e^{\mathbf{o}_{y^{(i)}}^a(x^{(i)})}}{\sum_{j=1}^{m} e^{\mathbf{o}_j^a(x^{(i)})}}$$

The parameters θ are: $\mathbf{W}^{(1)}$, and $\mathbf{b}^{(1)}$, $\mathbf{W}^{(2)}$, and $\mathbf{b}^{(2)}$.

The total number of scalar parameters is therefore given by
$n_\theta = d \cdot d_h + d_h + m \cdot d_h + m = d_h \cdot (d + 1) + m \cdot (d_h + 1)$.
The optimization problem is:

$$\hat{\theta} = \underset{\theta}{\text{argmin}}\, \hat{R}_\theta = \underset{\theta}{\text{argmin}}\, \frac{1}{n} \sum_{i=1}^{N} L_\theta(x^{(i)}, y^{(i)})$$

(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?

$$\theta^{(t+1)} = \theta^{(t)} - \alpha \nabla \hat{R}_\theta$$

(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

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

$$\begin{aligned} L(\mathbf{x}, y) &= -\log S(\mathbf{o}^a)_y \\ \frac{\partial L}{\partial \mathbf{o}^a_i} &= -\frac{1}{S(\mathbf{o}^a)_y} \cdot (S(\mathbf{o}^a)_i \mathbf{1}_{i=y} - S(\mathbf{o}^a)_i S(\mathbf{o}^a)_y) \text{ using the partial derivative from 1j} \\ &= -\mathbf{1}_{i=y} + S(\mathbf{o}^a)_i \\ \text{So } \frac{\partial L}{\partial \mathbf{o}^a} &= \mathbf{o}^s - \text{onehot}_m(y) \end{aligned}$$

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 $\frac{\partial L}{\partial \mathbf{o}^a_y}$.

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)!

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) [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

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

$$\mathbf{o}_k^a = \mathbf{b}_k^{(2)} + \sum_{i=1}^{d_h} \mathbf{W}_{ki}^{(2)} \mathbf{h}_i^s$$

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

$$\frac{\partial \mathbf{o}_k^a}{\partial \mathbf{b}_k^{(2)}} = 1$$

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

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

(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.

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

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

$\frac{\partial L}{\partial \mathbf{W}^{(2)}}$ has dimension (m, d_h), same as $\mathbf{W}^{(2)}$.
$\frac{\partial L}{\partial \mathbf{o}^a}$ has dimension $(m, 1)$.
$\mathbf{h}^s$ has dimension d_h, or $(d_h, 1)$.
$\frac{\partial L}{\partial \mathbf{b}^{(2)}}$ has dimension m, same as $\mathbf{b}^{(2)}$.
$\frac{\partial L}{\partial \mathbf{o}^a}$ has dimension m, or $(m, 1)$.

(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:

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

$$\mathbf{o}_k^a = \mathbf{b}_k^{(2)} + \sum_{i=1}^{d_h} \mathbf{W}_{ki}^{(2)} \mathbf{h}_i^s$$

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

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

(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.

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

$\frac{\partial L}{\partial \mathbf{h}^s}$ has dimension d_h, or $(d_h, 1)$ same as $\mathbf{h}^s$.
$\frac{\partial L}{\partial \mathbf{o}^a}$ has dimension m, or $(m, 1)$.
$\mathbf{W}^{(2)}$ has dimension (m, 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 $\frac{\partial \,\text{LeakyRELU}_\alpha(z)}{\partial z} = \text{LeakyRELU}_\alpha{}'(z) = \ldots$

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

We note the derivative is not defined at 0, but we permit $\text{LeakyRELU}_\alpha{}'$ to return 1 at 0 anyway.

$$\frac{\partial \mathbf{h}_j^s}{\partial \mathbf{h}_j^a} = \mathbf{1}_{\mathbf{h}_j^a \geq 0} + \alpha \mathbf{1}_{\mathbf{h}_j^a < 0}$$

$$\frac{\partial L}{\partial \mathbf{h}_j^a} = \frac{\partial L}{\partial \mathbf{h}_j^s} \left(\mathbf{1}_{\mathbf{h}_j^a \geq 0} + \alpha \mathbf{1}_{\mathbf{h}_j^a < 0} \right)$$

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

$$\frac{\partial L}{\partial \mathbf{h}^a} = diag\left(\left(\mathbf{1}_{\mathbf{h}^a \geq 0} + \alpha \mathbf{1}_{\mathbf{h}^a < 0}\right)\right) \frac{\partial L}{\partial \mathbf{h}^s}$$

$\frac{\partial L}{\partial \mathbf{h}^s}$ has dimension d_h, or $(d_h, 1)$ same as $\mathbf{h}^s$.
$diag\left(\left(\mathbf{1}_{\mathbf{h}^a \geq 0} + \alpha \mathbf{1}_{\mathbf{h}^a < 0}\right)\right)$ has dimensions (d_h, d_h).
$\frac{\partial L}{\partial \mathbf{h}^a}$ has dimension d_h, or $(d_h, 1)$ same as $\mathbf{h}^a$.

(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.

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

The chain rule gives:

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

$$\frac{\partial \mathbf{h}_k^a}{\partial \mathbf{W}_{kj}^{(1)}} = x_j$$
$$\frac{\partial \mathbf{h}_j^a}{\partial \mathbf{b}_j^{(1)}} = 1$$

So we have:

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

(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.

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

$\frac{\partial L}{\partial \mathbf{h}^a}$ has dimension d_h, or $(d_h, 1)$ same as $\mathbf{h}^a$.
$\frac{\partial L}{\partial \mathbf{W}^{(1)}}$ has dimension (d_h, d), same as $W^{(1)}$.
$\boldsymbol{x}$ has dimension d, or $(d, 1)$.

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

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

The chain rule gives:

$$\frac{\partial L}{\partial \boldsymbol{x}_k} = \sum_j \frac{\partial L}{\partial \mathbf{h}_j^a} \frac{\partial \mathbf{h}_j^a}{\partial \boldsymbol{x}_k}$$

$$\frac{\partial \mathbf{h}_j^a}{\partial \boldsymbol{x}_k} = \mathbf{W}_{jk}^{(1)}$$
$$\frac{\partial L}{\partial \boldsymbol{x}_k} = \sum_j \frac{\partial L}{\partial \mathbf{h}_j^a} \mathbf{W}_{jk}^{(1)}$$

(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?

The gradient with respect to $\mathbf{W}_{ij}^{(1)}$ is increased by $\lambda_{11}\text{sign}(\mathbf{W}_{ij}^{(1)}) + 2\lambda_{12}\mathbf{W}_{ij}^{(1)}$. The gradient with respect to $\mathbf{W}_{ij}^{(2)}$ is increased by $\lambda_{21}\text{sign}(\mathbf{W}_{ij}^{(2)}) + 2\lambda_{22}\mathbf{W}_{ij}^{(2)}$. The gradients with respect to $\mathbf{b}^{(1)}$ and $\mathbf{b}^{(2)}$ are unchanged.

1 Practical Report

1. Undergraduates 5 bonus pts Graduates 5 pts Train a neural network with 2 hidden layers, of size 512 and 256 respectively on the CIFAR-10 dataset, for 50 epochs. Use a

learning rate of 0.003, and a batch size of 100. Use the RELU activation function. For reproducibility purposes, **please set the random seed to 0**.

Include in your report the two following figures:

- A figure containing the evolution of both the training and validation accuracies during training.
  Answer. See Figure 1

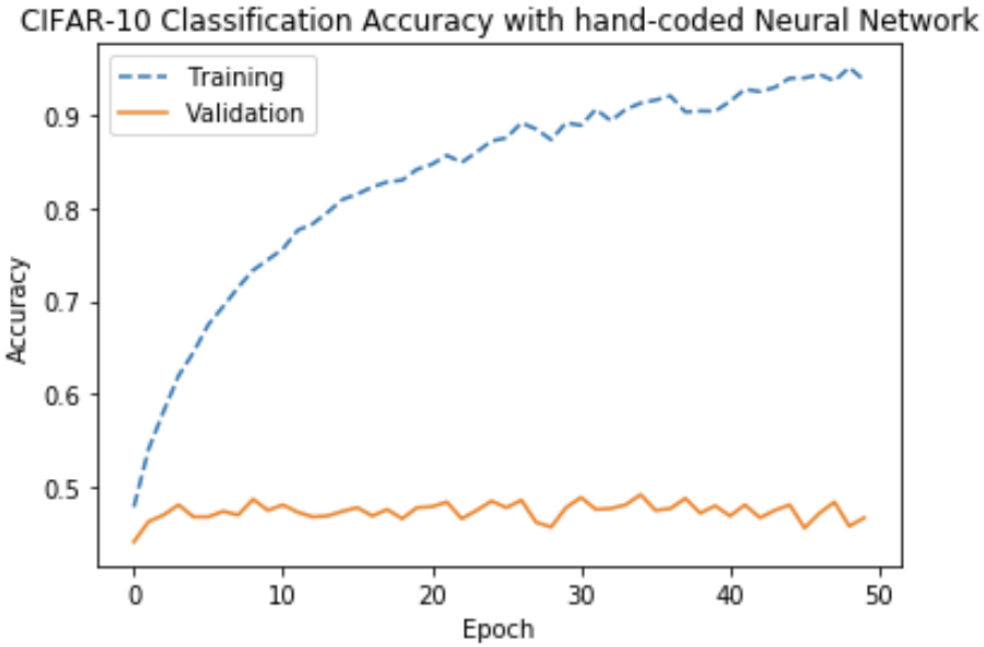

Figure 1: Training and Validation set accuracies on a neural network with two hidden layers of size 512 and 256.

- A figure containing the evolution of both the training and validation losses during training.
  Answer. See Figure 2

2. Undergraduates 5 bonus pts Graduates 5 pts

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

   Answer. The network was trained on the training set, and so is fit specifically for those samples. The validation set represents new, unseen data and so (as expected) the resulting accuracy on that data is lower.

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

   Answer.

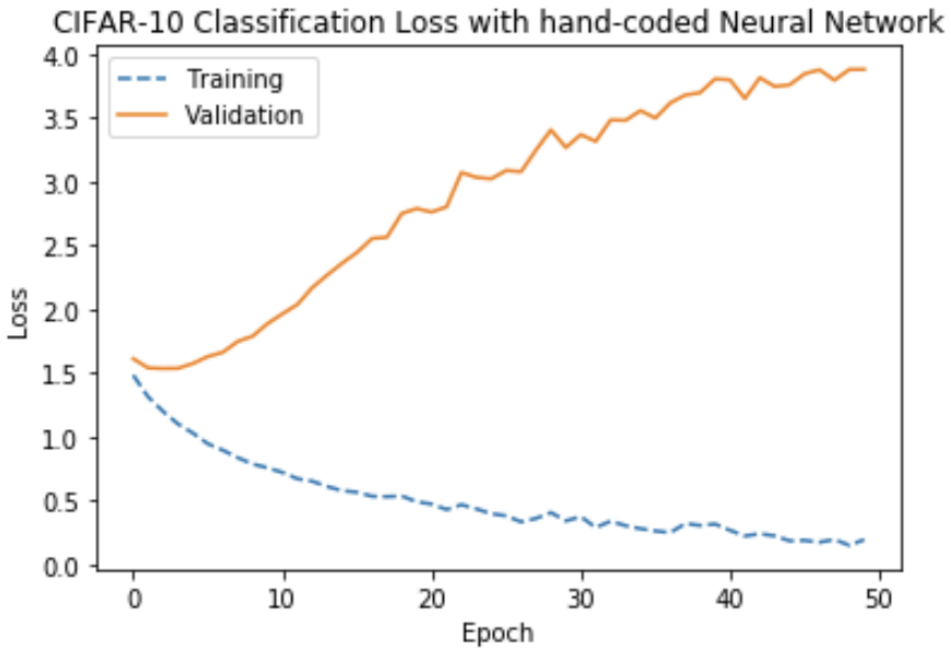

Figure 2: Training and Validation set losses on a neural network with two hidden layers of size 512 and 256.

- Increase the size of the training set. (i.e. Get more data.)
- Increase the complexity of the neural network. (i.e. More neurons.)

(c) How many learnable parameters (scalars) does the previous neural network have ?
Answer. This neural network has 3072 input features, two hidden layers of 512 and 256 neurons respectively, and an output layer of size 10. The total number of learnable parameters is therefore:

$$\begin{aligned}
&\sum_{i=1}^{n_hidden+1} (size(W^{(i)}) + size(b^{(i)})) \\
&= (3072 \text{ x } 512) + 512 \\
&+ (512 \text{ x } 256) + 256 \\
&+ (256 \text{ x } 10) + 10 \\
&\boxed{= 1,707,274}
\end{aligned}$$

3. Undergraduates 5 bonus pts Graduates 5 pts In this question, we are going to compare how the depth and width of the neural network affects the performances of this classification problem. For this purpose, we consider a deep neural network with equal number of neurons amongst the hidden layers starting from the second one. In order to have a fair comparison, we need to ensure that this new neural network has approximately the same number of learnable parameters as the previous one. We consider a neural network with n_hidden hidden layers. The first hidden layer has 512 neurons, and all the remaining ones have 120 neurons each.

- 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.

  Answer. This neural network has 3072 input features, a hidden layers of 512 neurons, an indeterminate number of layers of size 120 and an output layer of size 10. The total number of learnable parameters must be roughly equal to $1,707,274$. Assume a first layer of 120 neurons, and determine how many more are required:

  $$[(3072 \textbf{ x } 512) + 512] + [(512 \textbf{ x } 120) + 120] + [(120 \textbf{ x } 10) + 10] = 1,636,146$$

  Additional required scalar parameters:

  $$(1,707,274 - 1,636,146) = 71,128$$

  Now we determine how many additional layers of size 120 are required to make up the difference:

  $$\frac{71,128}{(120 \textbf{ x } 120) + 120} = \frac{71,128}{14,520} = 4.89 \approx 5$$

  Adding to the one 120 neuron layer already accounted for, we have a total of:

  $$n\_hidden_{120} = 6$$

  $$\boxed{n\_hidden = n\_hidden_{512} + n\_hidden_{120} = 1 + 6 = 7}$$

  And so a total of 7 hidden layers, including the first layer of 512 neurons.

- 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.
  Answer. See Figures 3 and 4

4. Undergraduates 5 bonus pts Graduates 5 pts

- 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?

  Answer. With only a single training run of each network, the standard deviation is zero, and we have no way to put error bars on our calculated accuracies. It is therefore very difficult (and irresponsible) to compare the average performance of each, especially with them converging to values that are quite close.

- 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**

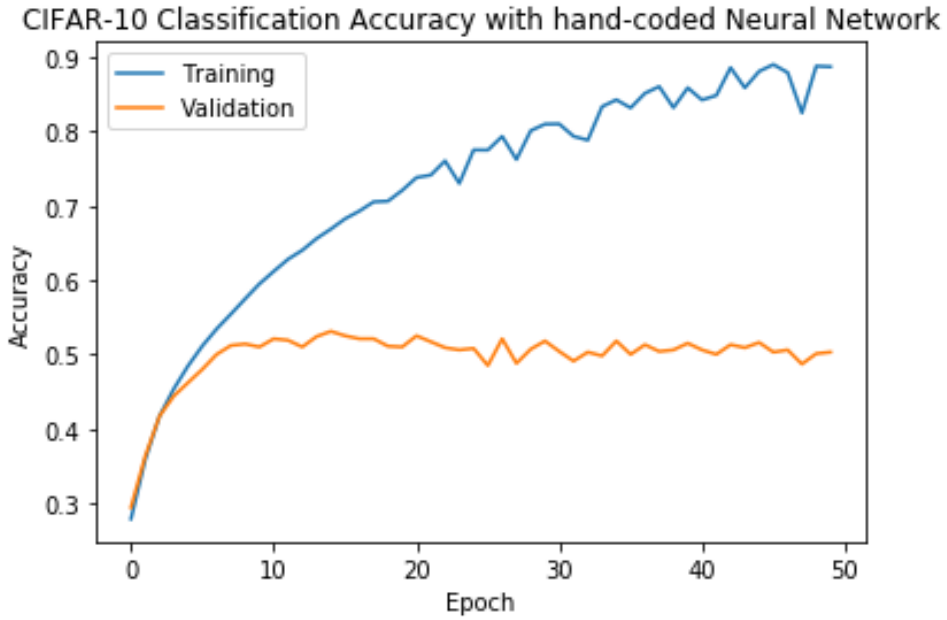

Figure 3: Training and Validation set accuracies on a neural network with 7 hidden layers. One of size 512 and 6 of size 120.

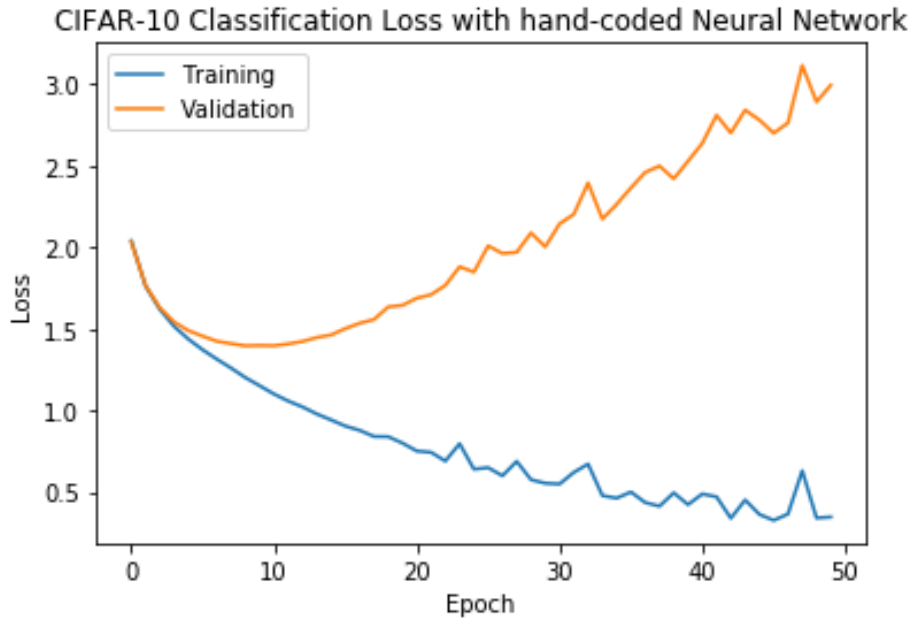

Figure 4: Training and Validation set losses on a neural network with 7 hidden layers. One of size 512 and 6 of size 120.

in your report

Answer. The neural networks were run for 50 epochs, each using seed values of [1, 42, 5000]. The error bars are displayed as 2 times the standard deviation.

See Figure 5

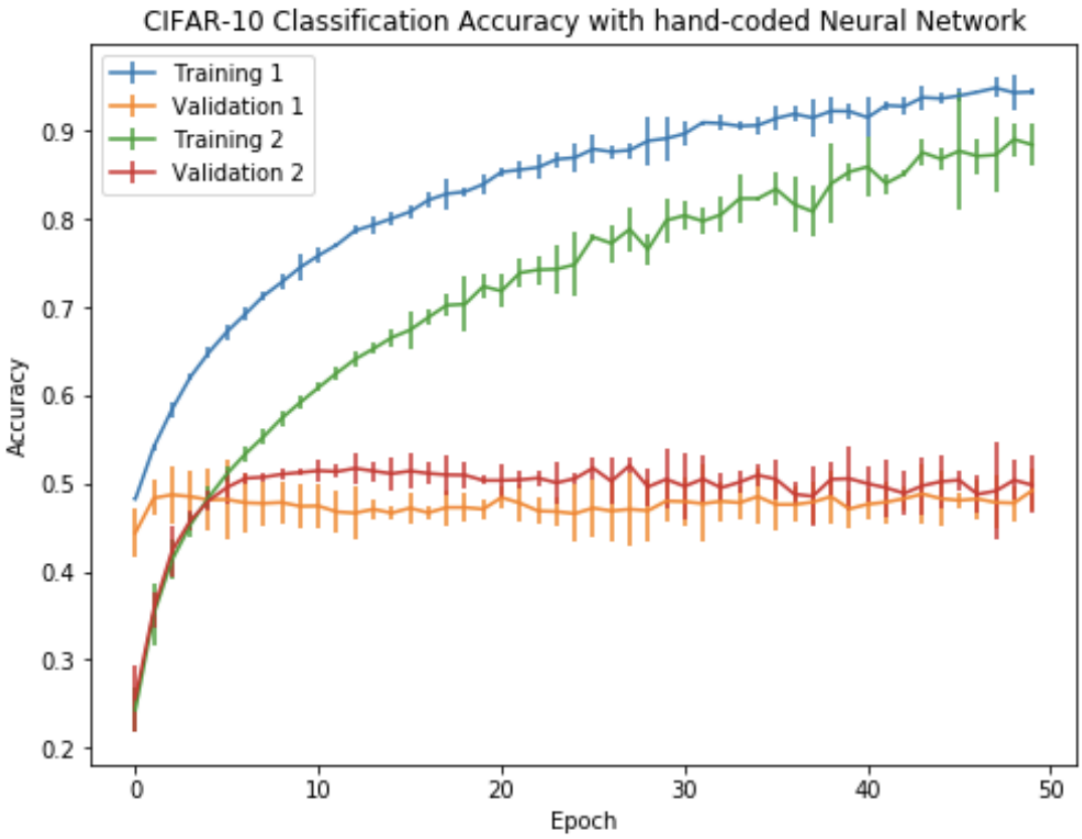

Figure 5: Training and Validation set losses on two neural networks. Set "1" consists of a network with two hidden layers of size 512 and 256. Set "2" consists of a network with 7 hidden layers; one of size 512, 6 of size 120.