Project II: Neural Networks - Exercises

1 Exercise

Below, you find a snippet of a code where h_1 refers to the first hidden layer and y refers to the output of a Neural Network. Answer the following questions respective to the code:

(a) How are the weights 'w1' and 'w2' initialized?

(b) What is the dimension of 'w1'?

(c) Which is the underlying activation function of h_1? (Name it and write it down mathematically)

```
In [ ]: weights = {
            'w1': tf.Variable(tf.truncated_normal([20, 15], stddev=  1)),
            'w2': tf.Variable(tf.truncated_normal([15,  1], stddev=0.1)),
        }


        biases = {
            'b1': tf.Variable(tf.constant(0.1, shape = [15])),
            'b2': tf.Variable(tf.constant(0.1, shape = [ 1])),
        }


        h_1 = tf.add(tf.matmul(x, weights['w1']), biases['b1'])
        y   = tf.math.softmax(tf.matmul(h_1, weights['w2']) + biases['b2'])
```

2 Exercise

Consider you have an input of volume 56x56x3 and you construct a Convolutional Layer using two filter of size 4x4xd with stride 2.

(a) Which value does d have to be?

(b) What is the volume of the output i.e. of the Convolutional Layer?

(c) When you flatten your output afterwards, what would be the resulting dimension?

3 Exercise

(a) Sketch the architecture of a four-layer Neural Network, where you consider $x = (x_1, \dots, x_D)$ as the input and y as the output. All layer shall consist of three Neurons, while the second hidden layer consists of only two Neurons and the output of a single Neuron.

(b) Consider now that you implement a dropout rule to increase generalisation. Sketch how the architecture *could* look like during a single iteration.

4 Exercise

Consider you have the following Loss in $\mathcal{L}_1$:

$$L(w) = \frac{1}{2}\Big[|y_1(w) - y_1^*| + |y_2(w) - y_2^*|\Big] .$$

Derive $\frac{\partial}{\partial w}L(w)$.

5 Exercise

The sigmoid activation function is given by $\sigma(x) = \frac{1}{1+\mathrm{e}^{-x}}$. Show that

$$\frac{\partial}{\partial x}\sigma(x) = \sigma(x)(1 - \sigma(x)) .$$

6 Exercise

Show that the tanh function is a re-scaled sigmoid function with both horizontal and vertical stretching, as well as vertical translation: $\tanh(x) = 2 \cdot \sigma(-2x) - 1$.

$$\textit{Remember:} \quad \begin{aligned} \sigma(x) \quad &= \frac{1}{1+\mathrm{e}^{-x}} \\ \tanh(x) = \quad &= \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)} = \frac{\exp(2x) - 1}{\exp(2x) + 1} \end{aligned} .$$

7 Exercise

Show that the derivative of the tanh activation function is at most 1, irrespective of the value of its argument. At what value of its argument does the tanh activation takes on its maximum value?

$$\textit{Remember:}\ \tanh(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)} = \frac{\exp(2x) - 1}{\exp(2x) + 1} .$$

8 Exercise

Show the following properties of the sigmoid and tanh activation functions:

(a) $\sigma(-x) = 1 - \sigma(x)$

(b) $\tanh(-x) = -\tanh(x)$

$$\textit{Remember:} \quad \begin{aligned} \sigma(x) \quad &= \frac{1}{1+\mathrm{e}^{-x}} \\ \tanh(x) = \quad &= \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)} = \frac{\exp(2x) - 1}{\exp(2x) + 1} \end{aligned} .$$

9 Exercise

A Neuron with four inputs has the weight vector $w = \begin{pmatrix} 1 & -2 & 3 & 4 \end{pmatrix}'$ and a bias $b = 0$. The activation function is the identity. What is the output of the Neuron, given an input $x = \begin{pmatrix} 4 & 8 & 5 & 6 \end{pmatrix}$?

10 Exercise

Consider an output is given by $y = \begin{pmatrix} 3 & 7 & 2 & 6 & 1 \end{pmatrix}$. Applying the Softmax activation function, what would be the result?

Note: You are allowed to give exact results as well as approximate results.

11 Exercise

A Perceptron has two inputs with weights $w_1 = -\frac{1}{5}$ and $w_2 = \frac{1}{2}$, no bias and a threshold $\theta = \frac{1}{3}$. For a given training example $x = \begin{pmatrix} 0 & 1 \end{pmatrix}$, the desired output is 0 (zero).

(a) Show that the Perceptron does not give the correct answer.

(b) Implement a bias b to receive the correct answer.

12 Exercise

A Neural Network shall consist of a single neuron with the identity as the given activation function. Your training data is given by $x_{[1]} = \begin{pmatrix} -1 & 2 \end{pmatrix}$ and $x_{[2]} = \begin{pmatrix} 2 & -3 \end{pmatrix}$ with correct outputs $y^*_{[1]} = 3$ and $y^*_{[2]} = -5$, respectively.
You aim to optimize your Neural Network using the squared $\mathcal{L}_2$ loss.

(a) Calculate the optimal weights for your Neural Network.

Note: You can use that the loss has only one extreme value, which is a minimum.

(b) You further want to implement the following regularization term:

$$R(w) = \frac{1}{2}\|w\|_2^2 \, .$$

Calculate the optimal weights for your Neural Network.

Note: You can use that the loss has only one extreme value, which is a minimum.

13 Exercise

Consider the function f given as

$$f : \mathbb{R}^2 \to \mathbb{R}, \; \begin{pmatrix} x \\ y \end{pmatrix} \mapsto \frac{1}{2}x^2 + x \cdot \cos y \, .$$

(a) Derive the extreme values of f.

(b) Conduct two steps with the Gradient Descent Algorithm starting at $\begin{pmatrix} x_0 \\ y_0 \end{pmatrix} = \begin{pmatrix} 1 \\ \pi \end{pmatrix}$ with learning rate $\eta = \pi$.

14 Exercise

Tired of using the Gradient Descent Algorithm to find extreme values, you read about the Newton Method in your favourite maths book. With the Newton Method, you can find the roots of a function f, i.e. all x such that $f(x) = 0$. The steps for the Newton Method are calculated by

$$x_{t+1} = x_t - \frac{f(x_t)}{f'(x_t)} ,$$

where f' is the first derivative of f.

(a) Write a Pseudocode, where you conduct the Newton Method for 1000 iterations.

(b) Write a Pseudocode, where you conduct the Newton Method and implement any *meaningful* stopping rule.

(c) How can you use the Newton Method to find the extreme values of a function f?

(d) Conduct one step using the Newton Algorithm to find the extreme values of $f(x) = 2x^3 - 10x^2 - 56x + 5$ starting at $x_0 = 0$.