Exercises for Foundations of Data Science

University of Zurich[UZH]

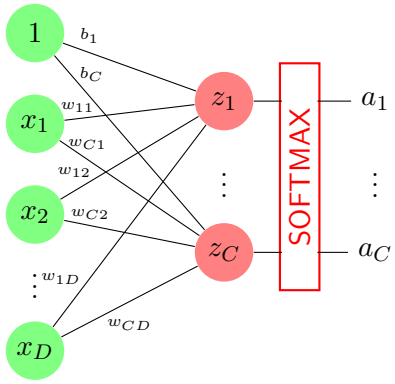
Prof. Dan Olteanu,
Dr. Ahmet Kara, Dr. Nils Vortmeier,
Haozhe Zhang

DaST Data•(Systems+Theory)

Fall 2020/2021 | Sheet 6 | 20.11.2020

- The solutions will be discussed on Friday 04.12.2020, 14:00-15:45 on Zoom.
- Videos with solutions will be posted on OLAT after the exercise session.

Exercise 6.1 [Backpropagation for Multiclass Logistic Regression]
We can view multiclass logistic regression as a simple neural network with no hidden layer. Let us suppose that the inputs are D dimensional and that there are C classes. Then the input layer has $D+1$ units, simply the inputs $x_1, \ldots, x_D$ and a constant 1 for the bias terms. The output layer has C units; there is full connection between the layers, i.e. every input unit is connected to every output unit. Finally, there is a non-linear activation function on the output layer, the softmax function. Pictorially, we may represent the neural network as follows:



Dropping superscripts to denote layers, as there are only two layers, the input and the output layer, and using the notation in the lectures, we have:

$$\mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b} \tag{1}$$

$$\mathbf{a} = \text{softmax}(\mathbf{z}) \tag{2}$$

Above $\mathbf{W}$ is a $C \times D$ matrix, and $\mathbf{b}$ is a column vector in C dimensions. Recall that the softmax function is defined as:

$$\text{softmax}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{l=1}^{C} e^{z_l}} \tag{3}$$

For a point $(\mathbf{x}, y)$ in the training data, y is an element of the set $\{1, \ldots, C\}$. The model parameters to be learnt are $\mathbf{W}$ and $\mathbf{b}$. We will use the cross entropy loss function, so the contribution of the point $(\mathbf{x}, y)$ to the objective function is given by:

$$\ell(\mathbf{W}, \mathbf{b}; \mathbf{x}, y) = -\log a_y \tag{4}$$

Write the expressions for $\frac{\partial \ell}{\partial \mathbf{a}}$, $\frac{\partial \ell}{\partial \mathbf{z}}$, $\frac{\partial \ell}{\partial \mathbf{W}}$ and $\frac{\partial \ell}{\partial \mathbf{b}}$ for a single training point $(\mathbf{x}, y)$. You should use the backpropagation equations to obtain these derivatives.

Exercise 6.2 [Output Encoding of Neural Nets]
In this exercise, we will consider a neural network with fully connected layers to classify handwritten digits.
The inputs are vectors of length 784 (obtained from 28×28 grey pixel images). Rather than outputting a class, the network will output a vector $\hat{\mathbf{y}} \in \mathbb{R}^{10}$. The target will be represented as a one-hot encoding, e.g. if the label is 3, we will use the vector $[0,0,1,0,0,0,0,0,0,0]^\mathsf{T}$ (by convention the digit 0 will be the last component in the encoding, not the first).
The data fed into the neural network is $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_N, \mathbf{y}_N)$, where $\mathbf{x}_i \in \mathbb{R}^{784}$ and $\mathbf{y}_i \in \{0,1\}^{10}$ is the one-hot encoding of the digit. The output of the neural network is also a vector $\hat{\mathbf{y}} \in \mathbb{R}^{10}$.
Suppose that in addition to the input layer and the output layer there is one hidden layer, and that we use the squared error as a loss function. The gradient of this loss function with respect to the parameters can be computed using the backpropagation algorithm as we have seen in the lectures.
Answer the following questions. You do not have to implement the network, qualitative answers according to your intuition are sufficient. Of course, you may nevertheless implement the network to back up your answers.

a) To reduce the size of the network and increase efficiency, your colleague suggests using a binary encoding for the outputs instead of one-hot encoding; so 0 is encoded as 0000, 1 as 0001, and so on. In this case, the output layer only has four neurons instead of ten. Perhaps you could also reduce the number of neurons in the middle layer. What do you think about this suggestion?

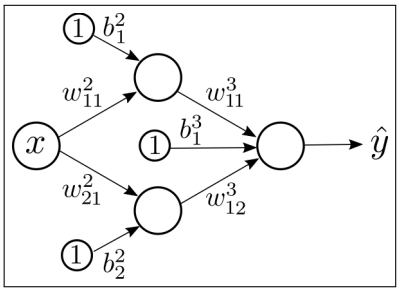
b) Show that if you have an already trained neural network that has high accuracy with the one-hot encoding, you can design a network that uses binary encoding and achieves roughly the same error.

c) What do you think would happen if you tried to train the neural network you suggested in the previous part directly (rather than adapting the network manually)?

Exercise 6.3 [Neural Networks as Universal Function Approximators]
As remarked in the lectures, neural networks are universal function approximators. In this exercise, we will discuss some steps for showing this result and what it means to be a universal function approximator a bit further.
Subsequently, we will exclusively consider neural networks with

- a single input,
- a single hidden layer with an arbitrary number of neurons with ReLU activation functions, and
- an output layer consisting of a single neuron with the identity activation function.

Consequently, the network implements a function $h\colon \mathbb{R} \to \mathbb{R}$. Whenever you are asked below to provide a neural network, your network is required to have those properties. The following picture shows an example of such a network with two neurons in the hidden layer:



a) Write down the model output $\hat{y}$ as a function of x by application of the forward equations.

b) Set $w_{11}^2 = w_{21}^2 = w_{11}^3 = 1$, $w_{12}^3 = -1$, $b_1^2 = -3$, $b_2^2 = -4$, and $b_1^3 = 0$. Compute $\hat{y}$ for the input $x = 3$ as well as for the input $x = 4$. State and sketch the function $h\colon \mathbb{R} \to \mathbb{R}$ implemented by the network.

c) Given constants $c < d$, provide a neural network which implements a function $h_{c,d}\colon \mathbb{R} \to \mathbb{R}$ such that for all $x \in \mathbb{R}$

$$h_{c,d}(x) = \begin{cases} 0 & \text{if } x < c \\ \frac{1}{d-c} \cdot x - \frac{c}{d-c} & \text{if } c \leq x \leq d \\ 1 & \text{if } x > d. \end{cases}$$

d) Let $f\colon \mathbb{R} \to \mathbb{R}$ be any function such that on the interval $[0, 1]$, f is continuous and bounded, i.e., there is some constant $c \geq 0$ such that $|f(x)| \leq c$ for all $x \in [0, 1]$. Argue that the class of neural networks considered in this exercise can approximate f arbitrarily close on the interval $[0, 1]$. More formally, given f and $\varepsilon > 0$, we can provide a neural network implementing some $h\colon \mathbb{R} \to \mathbb{R}$ such that $|f(x) - h(x)| < \varepsilon$ for all $x \in [0, 1]$.

Hint: Can you provide a neural network that implements the function $-h_{c,d}$? How can these functions be combined to obtain other interesting functions (note that the respective parameter values c, d may be chosen differently for different instantiations of the functions $h_{c,d}$ and $-h_{c,d}$)? Can you combine them to build a "pixelated" version of f, in the sense that the graph of this function only consists of (almost) vertical and horizontal lines?

Exercise 6.4 [Convolutional Neural Nets]

a) Suppose you have a simple convolutional neural net whose inputs are coloured images of size 64×64, where the information for each pixel is given by intensities between 0 and 255 for each colour channel. Now, a convolution filter of dimension $3 \times 3 \times 3$ is applied to the input with stride $(2, 2)$ (so, the filter slides over the input with step size 2 in every direction) and no zero-padding. What is the dimension of the output of this filter?

b) Consider the following two matrices:

$$I = \begin{pmatrix} 1 & 0 & 5 & 0 & 2 & 0 \\ 0 & 1 & 0 & 0 & 0 & 9 \\ 0 & 7 & 0 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 2 & 0 & 2 & 0 & 6 & 0 \end{pmatrix} \qquad R = \begin{pmatrix} 3 & 8 & 5 & 2 & 20 \\ 15 & 22 & 6 & 9 & 9 \\ 10 & 7 & 3 & 3 & 2 \\ 7 & 4 & 6 & 12 & 19 \end{pmatrix}$$

The matrix R is the result of applying some 2×2 convolution filter F to the input matrix I, with stride $(1, 1)$ and no zero-padding. Can you recover F?

c) In this part of the exercise, we will design convolutional layers to detect vertical boundaries in input images. Consider for example an input image from the MNIST data set as presented in Figure 1, which is a 28×28 matrix of numbers from 0 to 255. In the visualisation of this exercise, 0 is mapped to white, and 255 is mapped to black.

i) In a first convolutional layer, we want to apply two filters which detect vertical dark/light and light/dark boundaries in the input image, respectively. You may assume that the rectifier activation function with bias 0 is applied to the result of the convolutions to obtain the final results of the filters. Give two 3×3 filters that detect this kind of boundaries. Their respective result should look similar to the images given in Figure 2.

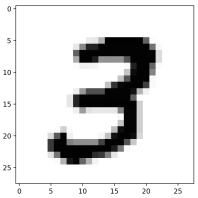
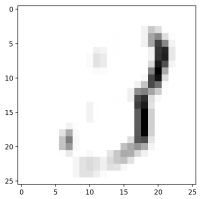
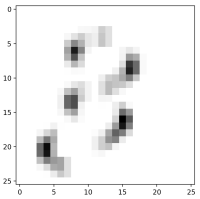


Figure 1: The original input image



(a) vertical dark/light boundaries



(b) vertical light/dark boundaries

Figure 2: Possible results of the filters.

ii) In a second convolutional layer, we want to combine the results of the first layer to produce an image that represents all the detected vertical boundaries. The result could for example look like the picture in Figure 3. Propose a filter that achieves such a result.

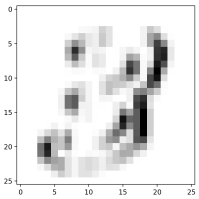


Figure 3: Vertical dark/light and light/dark boundaries of the input image.

Exercise 6.5 [k-Means Clustering]

a) Consider the points $p_1, \ldots, p_9$ with coordinates (x, y) as given below.

	x	y
p_1	2	3
p_2	3	1
p_3	4	2
p_4	14	5
p_5	12	4
p_6	13	7
p_7	6	5
p_8	9	4
p_9	6	4

Compute the result of Lloyd's Algorithm for $k = 2$, starting with the initial cluster centers p_2 and p_8.

Is there another choice of initial cluster centers that leads to a different result? If so, which is considered better with respect to the k-means objective?

b) The quality of a clustering obtained by Lloyd's Algorithm heavily depends on the initial choice of cluster centers. Show that the value of the clustering obtained by this algorithm can be arbitrarily worse than the optimal clustering, if the initial cluster centers are chosen unfavourably. More precisely, given any $k \geq 3$ and $m > 1$, give a set of points in the Euclidean space and a subset of k of these points as initial cluster centers, determine the clustering obtained by Lloyd's Algorithm and its value v_A, and give an optimal clustering and its value v_O, such that $v_A \geq m \cdot v_O$. Here, the value of a clustering $(C_1, \ldots, C_k)$ with means $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k$ is the value of of the k-means objective $\sum_{j=1}^{k} \sum_{i \in C_j} ||\mathbf{x}_i - \boldsymbol{\mu}_j||_2^2$.

c) Show that, regardless of the initialisation, Lloyd's Algorithm terminates after at most k^N iterations, where N is the number of input points and k is the number of clusters.