Universität des Saarlandes
Prof. Dr. Dietrich Klakow
Lehrstuhl für Signalverarbeitung
NNTI Winter Term 2023/2024

Exercise Sheet 6

Machine Learning Basics

Deadline: 14.12.2023 23:59

Guidelines: You are expected to work in a group of 2-3 students. While submitting the assignments, please make sure to include the following information for all our teammates in your PDF/python script:

Name:

Student ID (matriculation number):

Email:

Your submissions should be zipped as **Name1_id1_Name2_id2_Name3_id3.zip** when you have multiple files. For assignments where you are submitting a single file, use the **same naming convention** without creating a zip. For any clarification, please reach out to us on the **CMS Forum**.
Note that the above instructions are mandatory. If you are not following them, tutors can decide not to correct your exercise.

Exercise 6.1 - Activation functions (0.5+0.5+0.25+0.25 = 1.5 points)

Show that for $x \in \mathbb{R}$ the following identities hold:

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

b) $\tanh(x) = 2\sigma(2x) - 1$

c) $\frac{d}{dx}\sigma(x) = (1 - \sigma(x)) \cdot \sigma(x) = \sigma(-x) \cdot \sigma(x)$

d) $\frac{d}{dx}\tanh(x) = 4(1 - \sigma(2x)) \cdot \sigma(2x)$

Exercise 6.2 - Introduction to convexity (0.5 + 0.5 + 0.5 = 1.5 points)

Although convex analysis is not a main focus of this lecture, the notion of convexity is an important property in optimization and many traditional machine learning methods, such as the SVM, are designed to yield convex loss functions. In the following, you will see why convexity is such a desirable property and think about how neural networks fit into this picture.
We call a function $f : \mathbb{R}^n \to \mathbb{R}$ convex if for all $\boldsymbol{x}, \boldsymbol{y} \in \text{dom} f$ and any $\alpha \in [0, 1]$

$$f(\alpha\boldsymbol{x} + (1 - \alpha)\boldsymbol{y}) \leq \alpha f(\boldsymbol{x}) + (1 - \alpha)f(\boldsymbol{y}) \qquad \text{(Jensen's inequality)}$$

Intuitively, this tells us that if we pick any two points on the graph of f and connect them by a straight line, the graph of the function between these two points will lie below this line

[2]. You can find a visualization here.
Convex functions have the following property: For all $\boldsymbol{x}, \boldsymbol{y} \in \mathrm{dom} f$:

$$f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \nabla_{\boldsymbol{x}} f(\boldsymbol{x})^T(\boldsymbol{y} - \boldsymbol{x}) \qquad \text{(First-order characterization)}$$
$$\nabla^2_{\boldsymbol{x}} f(\boldsymbol{x}) \succeq 0, \qquad \text{(Second-order characterization)}$$

where we use $\mathbf{A} \succeq 0$ to denote that a matrix $\mathbf{A}$ is positive semi-definite. In one dimension, the first-order characterization means that the graph of the function lies above its tangent at any point (you can find a visualization here) and the second-order characterization means that the second-derivative is always non-negative.

a) Suppose that $f : \mathbb{R}^n \to \mathbb{R}$ is convex and differentiable. Prove that any local minimum $\boldsymbol{x}^\star$ is also a global minimum, i.e. show that $\forall \boldsymbol{y} \in \mathrm{dom} f : f(\boldsymbol{x}^\star) \leq f(\boldsymbol{y})$ holds.

b) Prove that the mean squared error $||\mathbf{X}\boldsymbol{w} - \mathbf{Y}||_2^2$ is a convex function of $\boldsymbol{w}$.

c) Are Neural Networks convex functions in general? Justify your answer.

Exercise 6.3 - General Questions (8×0.25 = 2 points)

Please answer the following questions briefly and justify your answers. Your answers should be no longer than 2-3 sentences.

a) Why are activation functions necessary?

b) What benefit does ReLU have over sigmoid/tanh activation functions?

c) What numerical issues may arise when using the softmax activation function? How could we prevent those?

d) Why is Newtons Method hardly applicable for training Neural Networks?

e) Does gradient descent always converge to the minimum of a convex function?

f) Does weight space symmetry help or hinder the optimization process? Hint: Think about how weight space symmetry affects the loss surface.

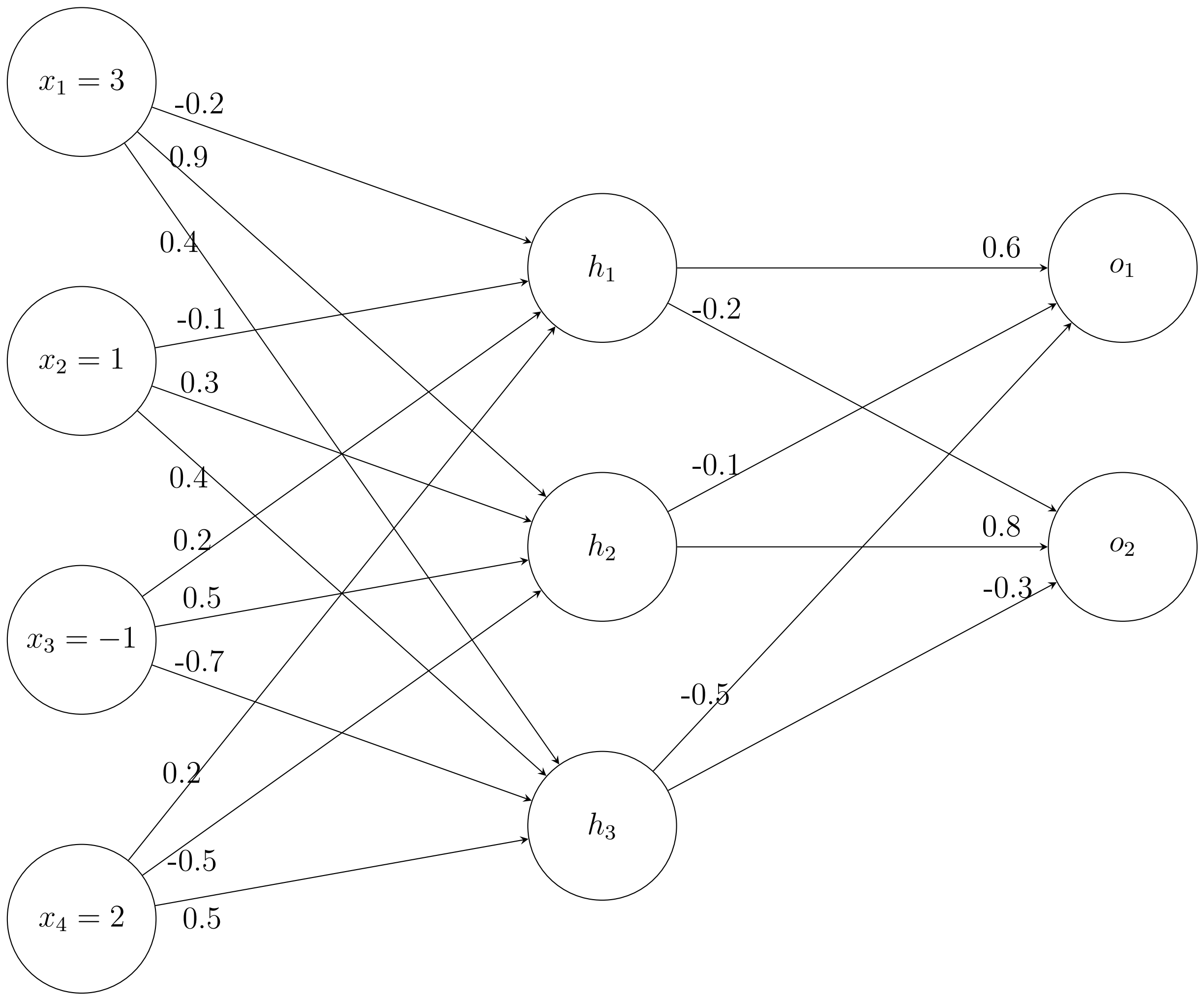
g) Give an example function where Newtons Method converges in one step to the optimum. Does gradient descent require more or less steps?

h) What problem does BatchNorm cause for a batch size of one?

Exercise 6.4 - BatchNorm (1.5 points)

Shortly summarize the main result of the paper [1] on how batch-normalization helps optimization. Keep your answers to a few sentences.

Exercise 6.5 - Forward Pass for a Fully-Connected Neural Network (0.5 points)



Given above is a simple Feed Forward Neural Network (FFNN) with one hidden layer. The input layer consists of four input units, the hidden layer has three hidden units, and the output layer has two output units. The activation functions on the hidden units are ReLUs and we use a Softmax function on the output layer. For the sake of simplicity, we assume that there are no *bias* parameters in the network.

a) Perform one inference step (forward pass) with the given inputs and weights for the given network. State the formulae that you use throughout your computations and show all the intermediate steps, i.e $h_1 = ...$, $o_1 = ...$

$$W^{(1)} = \begin{bmatrix} -0.2 & 0.9 & 0.4 \\ -0.1 & 0.3 & 0.4 \\ 0.2 & 0.5 & -0.7 \\ 0.2 & -0.5 & 0.5 \end{bmatrix} W^{(2)} = \begin{bmatrix} 0.6 & -0.2 \\ -0.1 & 0.8 \\ -0.5 & -0.3 \end{bmatrix}$$

Exercise 6.6 - Clustering Neural Networks (1.5 + 0.5 + 1 points)

In this exercise you gain familiarity with training basic feed forward neural networks in pytorch. You also learn how to run your code on a GPU in the *SIC-cluster*, which will prepare you for upcoming assignments and for the final project. Note that your submission **must** be executable on the cluster to be eligible for grading. That is, your zip file must include all the necessary files to run your code on the cluster. It is currently very tricky to run notebooks on the cluster, so please use python scripts instead.

a) We have prepared a template repo, that allows you to get started on the cluster quickly. You can find it here. Please follow the instructions in the readme to get started.

b) Load the FashionMNIST dataset and split it into train-, validation- and test set. You do not need to write your own dataset or dataloader classes. The `torchvision` package already provides a pre-defined dataset class that works with the default dataloader. You can find the documentation here.

c) Next, train a feed forward neural network to classify the images in the dataset. You can use this pytorch tutorial as a guide for how to do so. Your network should have at least 2 hidden layers and you should use SGD as your optimizer. This optimizer has not yet been covered in the lecture, but pytorch provides an easy-to-use implementation, enabling you to use it anyways. Train the network for 10 epochs and report the accuracy on the train-, validation- and test set.

References

[1] Santurkar, Shibani and Tsipras, Dimitris and Ilyas, Andrew and Madry, Aleksander (2018) *How Does Batch Normalization Help Optimization?*, NeurIPS Vol. 31

[2] Boyd, S., & Vandenberghe, L. (2004). Convex Optimization. Cambridge: Cambridge University Press. doi:10.1017/CBO9780511804441