Machine Learning and Physics
Winter Semester 2022/2023

Exercise Sheet #7

Due: 12.12.22 11h00

Roman Remme

General Regulations.

- Please hand in your solutions in groups of three people. A mix of attendees from Monday and Tuesday tutorials is fine.
- Your solutions to theoretical exercises can be either handwritten notes (scanned), or typeset using LaTeX.
- For the practical exercises, the data and a skeleton for your jupyter notebook are available at `https://github.com/hci-unihd/mlph_sheet07`. Always provide the (commented) code as well as the output, and don't forget to explain/interpret the latter. Please hand in both the notebook (`.ipynb`), as well as an exported pdf.
- Submit all your files in the Übungsgruppenverwaltung, only once for your group of three.

1 The logistic sigmoid

(a) Compute and simplify the derivative of the binary logistic sigmoid. (1 pt)

(b) Show that the tanh function $\tanh(x) = \frac{e^x - e^{-1}}{e^x + e^{-x}}$, another function sometimes used as the activation function in neural networks, is simply a scaled and shifted version of the binary logistic sigmoid. (2 pts)

(c) Consider binary classification: Given two points of class 1, one at $(1, 1)$ and one at $(2, 2)$ as well as two points of class 2, at $(1, 2)$ and $(2, 3)$, find a weight vector $\mathbf{w}$ and bias b such that the activation

$$a = \sigma(\mathbf{w}^T \mathbf{x} + b)$$

separates the two classes. (2 pts)

2 Log-sum-exp and soft(arg)max

The log-sum-exp and soft(arg)max[1] functions are defined on vectors $\boldsymbol{\sigma} \in \mathbb{R}^k$ as

$$\operatorname{lse}(\boldsymbol{\sigma}; \lambda) = \frac{1}{\lambda} \log \left(\sum_{j=0}^{K} \exp(\lambda \sigma_j)) \right) \quad \text{and} \quad \operatorname{soft(arg)max}(\boldsymbol{\sigma}; \lambda)_k = \frac{\exp(\lambda \sigma_k)}{\sum_{j=0}^{K} \exp(\lambda \sigma_j))} \, ,$$

with a scalar parameter $\lambda \in \mathbb{R}^+$ and $k = 1, .., K$.

(a) On which subset of the vectors $\boldsymbol{\sigma}^1 = (1, 2, 3)^T$, $\boldsymbol{\sigma}^2 = (11, 12, 13)^T$, $\boldsymbol{\sigma}^3 = (10, 20, 30)^T$ does the soft(arg)max yield identical results? Show in general whether the soft(arg)max is invariant under (i) constant offset and (ii) rescaling of its input. (2 pts)

(b) Make a 2D contour plot of $\operatorname{lse}((\sigma_1, \sigma_2)^T; \lambda)$ for $\lambda \in \{1, 10, 100\}$ and both σ_1 and σ_2 in the range $[-1, 1]$. Compare this to a contour plot of $\max(\sigma_1, \sigma_2)$ over the same range. (2 pts)

[1] in the literature know as just "softmax".

(c) Plot the two components of the soft(arg)max, over the same range and the same choices for λ as in (c), but as images instead of contour plots. Compare this to corresponding plots of the two components of the arg max over the 2D vectors, represented as a 2D one-hot vectors instead of indices:

$$\text{onehot}(\arg\max_i \sigma_i) = \begin{cases} (1,0)^T & \text{if } \sigma_1 < \sigma_2, \\ (0,1)^T & \text{else.} \end{cases}$$

(2 pts)

(d) Prove that the derivative of the lse is the soft(arg)max. (1 pt)

(e) **Bonus** Prove that

$$\lim_{\lambda \to \infty} \text{lse}(\boldsymbol{\sigma}; \lambda) = \max(\boldsymbol{\sigma}),$$

for all $\boldsymbol{\sigma} \in \mathbb{R}^n$. (2 pts)

3 Linear regions of MLPs

In this exercise you will build and investigate two regression Multi Layer Perceptrons (MLPs) using the pytorch (https://pytorch.org/) deep learning library. The input is two dimensional and each hidden layer consists of a linear transformation (`torch.nn.Linear`) followed by a ReLU activation fuction ($\text{ReLU}(x) = max(x, 0)$, `torch.nn.ReLU`)). The final layer is a linear transformation without activation function and should produce one scalar output. Formally, for H hidden layers, we have

$$\mathbf{a}_0 = \mathbf{x}\,, \quad \mathbf{a}_{i+1} = \text{ReLU}(\mathbf{W}_i \mathbf{a}_i + \mathbf{b}_i) \quad \text{for} \quad i \in \{0, .., H-1\}\,, \quad \mathbf{y} = \mathbf{W}_H \mathbf{a}_H + \mathbf{b}_H\,. \tag{1}$$

Here, $\mathbf{x} \in \mathbb{R}^2$ is the input, $\mathbf{y} \in \mathbb{R}^1$ the output, $\mathbf{a}_i$ are the activations and $\mathbf{W}_i, \mathbf{b}_i$ the weight matrices and bias vectors of the linear layers (the parameters of the model).

(a) Implement a shallow model with a single hidden layer with 20 neurons. For a tutorial on how to do this with pytorch, see for example https://pytorch.org/tutorials/recipes/recipes/defining_a_neural_network.html. How many parameters does the model have? (2 pts)

(b) pytorch automatically takes care of the random initialization of the model. Compute the output for a dense grid of at least 500 by 500 points in the range $\mathbf{x} \in [-10, 10] \times [-10, 10]$ and visualize it as an image. Repeat for a larger range; How far do you have to zoom out to capture all the structure? (2 pts)

(c) Use `numpy.gradient` to (approximately) compute the spatial gradient of the network output (as an image, from part (b)) and visualize both of its components as images using 'prism' as the colormap. What do you observe? (2 pts)

(d) Implement a deeper model with four hidden layers with 5 neurons each, and repeat part (b). Interpret your results and compare to the shallow model. (2 pts)

4 Bonus: Number of linear regions

What is the maximum number of linear regions of an MLP with a two dimensional input and one hidden layer with n neurons and ReLU activations? Hint: Consider the construction in the lecture, representing each hidden neuron as a line in the input space. (5 pts)