Machine Learning and Physics
Winter Semester 2023/2024

Exercise Sheet #3

Due: **13.11.**2023 11h00

Jan Pawlowski, Tilman Plehn
Jonas Spinner

General Regulations.

- Please hand in your solutions in groups of three people. A mix of attendees from different tutorials is fine. We will not correct submissions from single students.
- Your solutions to theoretical exercises can be either handwritten notes (scanned to `pdf`), typeset using LaTeX, or directly in the jupyter notebook using Markdown.
- For the practical exercises, the data and a skeleton for your jupyter notebook are available at `https://github.com/heidelberg-hepml/mlph2023-Exercises`. Always provide the (commented) python code as well as the output, and don't forget to explain/interpret the latter, we do not give points for code that does not run. Please hand in both the notebook (`.ipynb`) and an exported pdf. Combine your the pdfs from theoretical and notebook exercises into a single pdf.
- Submit all your files in the Übungsgruppenverwaltung, only once for your group of three. Please list all names and tutorial numbers to simplify things for us.

1 The logistic sigmoid

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

(b) Show that the tanh function $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$, another function sometimes used as the activation function in neural networks, is simply a scaled and shifted version of the binary logistic sigmoid. Where in your neural network would you use the sigmoid as activation, and where would you rather use tanh?

(1 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 $\mathbf{b}$ such that the activation

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

separates the two classes. (1 pts)

2 Projection Trick

The data in `data2d.npy`, `labels.npy` is a 2D classification problem.

(a) Can this problem be solved with a linear decision boundary in 2D? Apply logistic regression (as implemented in scikit-learn, `sklearn.linear_model.LogisticRegression`), and visualize the decision boundary of the classifier in a scatter plot of the data. Which accuracy do you get? (2 pts)

(b) Come up with a nonlinear transformation to enhance the feature space, such that the classes can be linearly separated. Visualize the points in the new, 3D feature space[1]. Demonstrate that the problem can be solved in the enhanced space using logistic regression. (2 pts)

(c) Describe (in words) how this problem could be solved by an MLP with a single hidden layer. (1 pts)

[1]You could for example use `https://matplotlib.org/stable/gallery/mplot3d/scatter3d.html`.

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 $\mathbf{b} \in [-10, 10] \times [-10, 10]$ and visualize it as an image. Repeat for a larger range; How far do you have to zoom out to capture all the structure? (1 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? (1 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 Learning by hand

Before starting to train neural networks on the next sheet, we take a step back and look into how backpropagation works. The flow of information through a "standard" neural network falls into two phases. First we propagate information forward through the network from the input layer to a scalar output (sometimes referred to as *forward propagation*). The second phase then consists of backpropagating the error we received from the scalar loss/error/cost function that we try to optimize. For this exercise we take the function

$$L(\mathbf{x}) = \left(\sin\frac{x_1}{x_2} + \frac{x_1}{x_2} - \exp(x_2)\right) \cdot \left(\frac{x_1}{x_2} - \exp(x_2)\right),$$

with initialization $\mathbf{x} = (1.5, 0.5)$. We recommend micrograd for an (extensive) intro to backpropagation and autodifferentiation.

(a) Give the computational graph of the function $L(\mathbf{x})$. (1 pts)

(b) Give the forward trace at the given $\mathbf{x}$, i.e. the values of all intermediate elementary expressions. (1 pts)

(c) Give the backward/reverse trace, i.e. the gradients of the final result $L(\mathbf{x})$ with respect to all intermediate elementary expressions. Avoid rounding errors. (2 pts)

(d) Now assume that we want to optimize x_1 such that $L(\mathbf{x})$ becomes minimal. Use the famous `Adam` algorithm with default hyperparameters and learning rate 0.1 to perform two optimization steps. What is your result for x_1? (2 pts)

5 ML4Jets

The ML4Jets conference on applications of Machine Learning in High Energy Physics takes place from 06.11. to 10.11.2023. The program is available at `https://indico.cern.ch/event/1253794/timetable/#all.detailed`. You can join the talks with the following links

- Zoom room for sessions in Auditorium:
  `https://cern.zoom.us/j/64915610406?pwd=WGNNTHIzS2p6cWNKSHJYZXd5TjAzdz09`
  Meeting ID: 64915610406
  Passcode: 84449936
- Zoom room for sessions in SR4:
  `https://cern.zoom.us/j/68767228603?pwd=cHhTbm8rS0Jxbzh6c2VjRHc0UlpDQT09`
  Meeting ID: 68767228603
  Passcode: 26979774
- Zoom room for online discussion session (Thursday evening):
  `https://uni-hamburg.zoom.us/j/69422341424?pwd=L2hJMlcxa1N6V0Z5SkZIV0xJVFU3dz09`
  Meeting ID: 694 2234 1424
  Passcode: 92659282
- Zoom room for colloquium (Tuesday afternoon):
  `https://desy.zoom.us/j/99616528733`
  Meeting ID: 996 1652 8733
  Passcode: 733220

Attend a talk of your choice and summarize what you have learnt. What is the physics problem? Which machine learning tools have been used? What are the remaining challenges? (2 pts)