Exercise Sheet #7

Due: 18.12.2023 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 Log-sum-exp and softmax

The log-sum-exp and softmax functions are defined on vectors $\boldsymbol{\sigma} \in \mathbb{R}^k$ as

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

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 softmax yield identical results? Show in general whether the softmax is invariant under (i) constant offset and (ii) rescaling of its input. (1 pts)

(b) Make a 2D contour plot of $\text{lse}((\sigma_1,\sigma_2)^T;\lambda)$ 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. (1 pts)

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

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

(1 pts)

(d) Prove that the derivative of the lse is the softmax. (1 pts)

(e) Prove that

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

for all $\boldsymbol{\sigma} \in \mathbb{R}^n$. (1 pts)

2 Top Tagging with Point Clouds

Instead of interpreting jets as images and using image vision techniques as we did on the last sheet, state-of-the-art approaches to top tagging work with jets as point clouds. This representation captures the properties of a jet more naturally. As of today, the leading architectures are ParticleNet[1] and the Particle Transformer[2]. They are based on Dynamic Graph Convolutional Neural Networks (DGCNN) and the Transformer architecture, respectively.

The goal of this exercise is to study one of the two architectures in detail. You can use the tutorials 7 (DGCNN) or 8 (Transformer) from https://github.com/heidelberg-hepml/ml-tutorials as a guideline.

(a) Pick one of the two point cloud architectures mentioned above and explain it in detail. How is the point cloud processed by the neural network? Which parts of the architecture feature learnable parameters? (2 pts)

(b) Implement a top tagger based on the architecture picked in part (a). Train it on a single batch to test your implementation. (2 pts)

(c) Train your top tagger on the full training dataset. Evaluate the performance in the same way as done on the last sheet, and compare the results with the CNN top tagger. (2 pts)

3 CNNs for Galaxy Classification

The Galaxy10 SDSS[3] dataset consists of 21785 colored (green, red and near-infrared band) images of galaxies with a resolution of 69 by 69 pixels, taken from the Sloan Digital Sky Survey (SDSS). They have been annotated by human volunteers: Each image is assigned to one of nine classes which describe the morphology of the depicted galaxy. In this task, you will train Convolutional Neural Networks (CNNs) to classify the images into those classes.

(a) Implement a small CNN for the classification task: It should consist of three consecutive blocks:

1. A convolutional layer with kernel size 5 and 8 output features, ReLU activation and max-pooling,
2. A convolutional layer with kernel size 5 and 16 output features, ReLU activation and max-pooling,
3. A flattening layer that reshapes the channel, width and height axes into a single axis, followed by an MLP with two hidden layers of size 64 and 32. (1 pts)

(b) Train the model and compute the accuracies on both training and test dataset. Create and plot a confusion matrix of the trained model on the validation set. (2 pts)

(c) Repeat the above with a more powerful architecture: Modify a Resnet34 (as implemented in torchvision) to the correct number of output channels. What is a Resnet? Note: For this, training on a GPU using google-colab[4] should result in a significant speedup compared to most CPUs. (2 pts)

(d) In contrast to natural images, which oftentimes have a preferred orientation, the galaxies in this dataset are oriented arbitrarily. Hence, it would be desirable to adapt the model such that its prediction is invariant with respect to at least some rotations, e.g. that rotating the input image by a multiple of 90° does not change the outputs. How could this be achieved? Can you also come up with a method that would work for a larger set of rotations? (1 pts)

[1] https://arxiv.org/pdf/1902.08570.pdf
[2] https://arxiv.org/pdf/2202.03772.pdf
[3] https://astronn.readthedocs.io/en/latest/galaxy10sdss.html
[4] https://colab.research.google.com/