HHU WS1920 Generative Models (MSC) — Tuesday, 15. October 2019
Prof. Dr. Markus Kollmann — DUE 11:59 am Tuesday, 22. October 2019

Exercise set #1

Please provide your documented solution (along with comments/markdown) in an Jupyter notebook. Use Python3 and for the optimization and machine learning tasks always Pytorch. You can submit via sciebo. Submit everything in one zip file with your full name plus exercise number for the filename as an indicator, e.g. max_mustermann_1.zip

1. **Likelihood**
   First, run the following code. It will generate a dataset of samples from a categorical distribution where $x \in \{1, ..., 100\}$. Take the first 80% of the samples as a training set and the remaining 20% as a test set.

```
import numpy as np
def sample_data():
    count = 10000
    rand = np.random.RandomState(0)
    a = 0.3 + 0.1 * rand.randn(count)
    b = 0.8 + 0.05 * rand.randn(count)
    mask = rand.rand(count) < 0.5
    samples = np.clip(a * mask + b * (1 - mask), 0.0, 1.0)
    return np.digitize(samples, np.linspace(0.0, 1.0, 100))
```

   Let $\theta = (\theta_1, ..., \theta_{100}) \in \mathbb{R}^{100}$, and define the model

$$p_\theta(x) = \frac{e^{\theta x}}{\sum_{x'} e^{\theta x'}}$$

   Fit p_θ with maximum likelihood via stochastic gradient descent on the training set, using θ initialized to zero. Use Pytorch with the built in version of stochastic gradient descent, and optimize your hyperparameters on a validation set of your choice. Provide these deliverables:

   (a) Over the course of training, record the average negative log likelihood of the training data (per minibatch) and validation data (for your entire validation set). Plot both on the same graph, the x-axis should be training steps and the y-axis should be negative log likelihood; feel free to compute and report the validation performance less frequently. Report the test set performance of your final model. Be sure to report all negative log likelihoods in bits.

   (b) Plot the model probabilities in a bar graph, with $\{1, ..., 100\}$ on the x-axis and a real number in [0,1] on the y-axis. Next, draw 1000 samples from your model, and plot their empirical frequencies on a new bar graph with the same axes. How do both plots compare visually to the data distribution?

50 points

2. **Two-dimensional data**

In this problem, you will work with bivariate data of the form $x = (x_1, x_2)$, where $x_1, x_2 \in \{0, 1, ..., 199\}$. In the file called `distribution.npy`, you are provided with a 2-dimensional array of floating point numbers representing the joint distribution of x: element (i, j) of this array is the joint probability $p_{data}(x_1 = i, x_2 = j)$. Sample a dataset of 100,000 points from this distribution. Treat the first 80% as a training set and the remaining 20% as a test set. Implement and train the following model for this data:

(a) $p_\theta(x) = p_\theta(x_1)p_\theta(x_2|x_1)$, where $p_\theta(x_1)$ is a distribution represented in the same way as in part 1, and $p_\theta(x_2|x_1)$ is a multilayer perceptron (MLP) that takes x_1 as input and produces a distribution over x_2. (You have some freedom in designing the architecture of this MLP. For example, it can read the x_1 input either as a real number or as a one-hot vector. Experiment with such designs and pick what works best on validation data.)

(b) Over the course of training, record the average negative log likelihood of the training data (per minibatch) and the validation data (for the entire validation set). Plot both on the same graph (the x-axis should be training steps, and the y-axis should be negative log likelihood and report the test set performance of your final model. Report all negative log likelihoods as bits per dimension (i.e. bits divided by 2).

(c) Draw samples and plot them in a 2D histogram with 200 bins on each side. (Consider using the `hist2d` function in `matplotlib`.)

50 points