ECE 6254, Spring 2024

Homework # 6

Due Tuesday, April 23, at 11:59pm EST.

Problems:

1. **Neural networks outputs as probabilities.** Suppose we are applying a neural network to a classification problem with K classes. One approach to predicting a class label, as well getting some notion of "confidence" is to map an input to a discrete probability distribution over all K classes. One way to do this is to have the neural network will map an input data point $\boldsymbol{x} \in \mathbb{R}^D$ to a vector $\boldsymbol{z} \in \mathbb{R}^K$, then pass $\boldsymbol{z}$ through the *softmax function* $\sigma : \mathbb{R}^K \to \mathbb{R}^K$. We will denote output vector of the softmax as $\boldsymbol{p} \in \mathbb{R}^K$, with i-th entry given as

$$\boldsymbol{p}_i = \frac{e^{\boldsymbol{z}_i}}{\sum_{k=1}^K e^{\boldsymbol{z}_k}} \tag{1}$$

In this problem, we explore connections between the softmax function and projection onto the $K-1$ dimensional simplex, defined as:

$$\Delta_{K-1} = \left\{ \boldsymbol{p} \in \mathbb{R}^K : \sum_{k=1}^K \boldsymbol{p}_k = 1 \text{ and } \boldsymbol{p}_k \geq 0 \ \forall k = 1, \dots K \right\} \tag{2}$$

We will show that

$$\begin{aligned} \boldsymbol{p} = \sigma(\boldsymbol{z}) = &\operatorname*{arg\,min}_{\boldsymbol{x} \in \mathbb{R}^K} - \langle \boldsymbol{z}, \boldsymbol{x} \rangle - H(\boldsymbol{x}) \\ s.t. \qquad &\sum_{k=1}^K \boldsymbol{x}_k = 1 \\ &0 \leq \boldsymbol{x}_k \leq 1 \qquad \forall k = 1, \dots, K \end{aligned}$$

where H is the entropy of $\boldsymbol{x}$, defined as

$$H(\boldsymbol{x}) = -\sum_{k=1}^K \boldsymbol{x}_k \log(\boldsymbol{x}_k)$$

For this problem, you will need a slightly more general version of the Lagrangian/KKT conditions that the one in class in order to account for the equality constraint. This can be found here: `https://en.wikipedia.org/wiki/KarushKuhnTucker_conditions`.

(a) What is the Lagrangian for the above projection?

(b) Write down the KKT conditions for the above problem.

(c) Using the KKT conditions, show that $\boldsymbol{p}_i = \frac{e^{\boldsymbol{z}_i}}{\sum_{k=1}^K e^{\boldsymbol{z}_k}}$.

2. **Classification and regression with neural networks** In this problem, we'll investigate the performance of neural networks[1] as both classifiers and regressors. In particular, we will use a very basic type of neural network called the multi-layer perceptron (MLP) for both settings. To get started, download `neural_net.py`.

   (a) The provided code loads the same digits dataset used in the previous problem, but with all 10 classes. Train a MLP classifier with the provided training set. Set `max_iter=1000`. Feel free to adjust other parameters such as `hidden_layer_sizes` (both size of the layers and number of layers) or `activation`. Report the test accuracy and the configuration of parameters that achieved this accuracy.

   (b) The provided code generates noisy samples of the function $\sin(9x) + x$. Train a MLP regressor with the provided training set. Set `max_iter=1000`. Feel free to adjust other parameters such as `hidden_layer_sizes` (both size of the layers and number of layers) or `activation`. Report the mean squared error on the test set and the configuration of parameters that achieved this error. Include a plot of the true function and the neural network output.

[1] In practice, neural networks are not implemented with scikit-learn, but rather with dedicated frameworks such as PyTorch (`https://pytorch.org`). Such frameworks often have a bit of a learning curve, so we use the scikit-learn implementation for this problem.