Deep Learning

Homework #1

Due Friday, January 29 at 10:00 AM

1. Implement and train a logistic regression model from scratch in Python for the MNIST dataset (no PyTorch). The neural network should be trained on the Training Set using stochastic gradient descent. Please submit your code and report the test accuracy on the Test Set.[1]

2. Implement and train a single-layer fully-connected neural network from scratch in Python for the MNIST dataset (no PyTorch). The neural network should be trained on the Training Set using stochastic gradient descent or mini-batch stochastic gradient descent. It should achieve 97-98% accuracy on the Test Set. Please submit your code and report the test accuracy on the Test Set.

3. Prove that the softmax function $F_{\text{softmax}} : \mathbb{R}^K \to \mathbb{R}^K$ produces a probability distribution.

4. Prove that there exists a constant learning rate $\alpha > 0$ such that gradient descent always decreases the objective function $\mathcal{L}(\theta) : \mathbb{R}^d \to \mathbb{R}$ at every iteration if the second derivatives of $\mathcal{L}(\theta)$ are uniformly bounded.

5. Consider a dataset $(x^i, y^i)_{i=1}^N$ where $x \in \mathbb{R}^d$ and $y \in \mathbb{R}$. Recall the least-squares objective function $\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^N (y^i - \theta^\top x^i)^2$ for the linear model $f(x;\theta) = \theta^\top x$. Derive the stochastic gradient descent algorithm for this linear regression model.

6. Let's now consider a nonlinear model $f(x;\theta) = g(\theta^\top x)$ with an objective function $\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^N \left| y^i - g(\theta^\top x^i) \right|$. Derive the stochastic gradient descent algorithm for this model.

7. Provide an example where the global minimum of a neural network is not unique.

[1]For a classification model $f(\cdot;\theta)$, the model prediction given an input x is $\arg\max_k f_k(x;\theta)$. For data samples $(x^n, y^n)_{n=1}^N$, the accuracy is the percent of correct predictions $\frac{1}{N}\sum_{n=1}^N \mathbf{1}_{y^n = \arg\max_k f_k(x^n;\theta)} \times 100\%$.

8. Let us consider the neural network

$$\begin{aligned} Z &= Wx + b^1 \\ H_i &= \sigma(Z_i), \quad i = 0, \dots, d_H - 1, \\ f(x;\theta) &= CH + b^2, \end{aligned} \tag{0.1}$$

where $x \in \mathbb{R}^d$, $W \in \mathbb{R}^{d_H \times d}$, $b^1 \in \mathbb{R}^{d_H}$, $Z \in \mathbb{R}^{d_H}$, $H \in \mathbb{R}^{d_H}$, $C \in \mathbb{R}^{K \times d_H}$, $b^2 \in \mathbb{R}^K$, and the activation function is the clipped ReLU unit

$$\sigma(z) = \min\big(\max(z, 0), t\big), \tag{0.2}$$

where t is a hyperparameter.

The dataset is $(x^n, y^n)_{n=1}^N$ where $(x^n, y^n) \in \mathbb{R}^d \times \mathbb{R}^K$, and the objective function is

$$\begin{aligned} \mathcal{L}(\theta) &= \frac{1}{N} \sum_{n=1}^{N} \rho\big(f(x^n;\theta), y^n\big), \\ \rho(z, y) &= \sum_{k=0}^{K-1} \big|z_k - y_k\big|, \end{aligned} \tag{0.3}$$

where $\theta = \{W, C, b^1, b^2\}$.

Derive the backpropagation algorithm and stochastic gradient algorithm to minimize the objective function (0.3).