Deep Learning (83882) - Ex 2

Due: 5.1.2025, 11:59pm

The Mission

In this exercise, you will need to solve a multi-class classification task with a logistic regression classifier and a Neural Network (NN) with one hidden layer. You will need to implement these models from scratch using Numpy only (i.e., using deep learning frameworks like Pytorch or TensorFlow is not allowed). The dataset you will be working on is Fashion-MNIST.

Fashion-MNIST is a dataset of images, consisting of 70,000 examples. Each example is a 28x28 grayscale image, associated with a label from the following 10 classes: T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, and Ankle boot.

In addition to this instruction file, in the exercise folder you will also find the following two files:

- train.csv - a data file containing 56,000 examples that should be used for training your models. Each row in the file is a different example where the first value indicates the class label and the rest of the 784 values are the flatten image (i.e., a feature vector of size 784).
- test.csv - a data file containing 14,000 examples that will be used to evaluate your best model by us. Unlike train.csv, this file contains only the features (i.e., it does not contain label information).

Part 1 - Visualize the Data (10 Pts)

First, you will need to visualize the data using the Matplotlib package. Write code the reads the data from train.csv and plot 4 different examples from each class in a grid of 10x4 (i.e., 10 rows, each contains 4 different examples from the same class). Label each row by the class name. Add the figure to a report that should be submitted along your code.

Part 2 - Logistic Regression Classifier (40 Pts)

Implement a Logistic Regression classifier and train it on the Fashion-MNIST dataset. Note that unlike the recitation, in which we saw a binary logistic regression classifier, here you will need to implement a multi-class logistic regression classifier. More specifically, you will need to write code for (non-exhaustive list):

- Your model.
- Generating predictions.
- Softmax function.
- L2 Regularization term (you are encouraged to try L1 as well).
- Computing the gradients (don't forget here the regularization term as well).
- Updating the parameters with Mini-batch GD.
- Tracking the loss and the accuracy of your classifier on the training and validation sets.

Here, you will also need to do some foundation work that will serve you in the next part as well. For example, you will need to (i) implement code that split the train data to train part and validation part (you may split it however you like) and, (ii) implement code that normalizes the data (a reminder, this should be done based on training part only).

Test different configurations of batch size, learning rate, and regularization coefficient, and write your findings in a report (several lines describing your main insights should suffice). Submit a file named lr_pred.csv with the predictions of your best model only on test.csv. The file should contain only the predictions (no commas, column header, line numbers, etc.), each one in a new line. Make sure that the order of the predictions matches the order of the examples in test.csv. Your grade may also be determined by the quality of the predictions. Add to the report a plot of the train and validation loss and accuracy curves as a function of the number of epochs of your best model.

Tip 1: When working on a multi-class classification task often it is more convenient to represent the labels via one-hot encoding. It will enable you to write a vectorized implementation of the code more easily.

Tip 2: The softmax function is sensitive to underflow and overflow issues. Recall that in Ex1 you showed that $softmax(\mathrm{z}) = softmax(\mathrm{z} + c)$. We will use it here for creating a numerically stable softmax. Let $\mathrm{z} \in \mathbb{R}^{10}$ be the output of your classifier without applying the softmax activation (aka *logits*), instead of applying $softmax(\mathrm{z})$, apply $softmax(\mathrm{z} - max(\mathrm{z}))$, where $max(\mathrm{z})$ is the logit that has the highest value.

Part 3a - Deriving the Gradients of a NN with a Softmax Activation

In the next part you will need to implement a NN with one hidden layer for solving a multi-class classification task. As a preliminary step you should derive the gradients w.r.t the parameters of such models. In this exercise we will consider the following model:

$$\begin{aligned}\hat{\mathrm{y}} &= softmax(W_2^T \mathrm{h} + b_2)\\ \mathrm{h} &= g(W_1^T \mathrm{x} + \mathrm{b}_1)\end{aligned}$$

Where (in our case) $W_1 \in \mathbb{R}^{784\times d}, \mathrm{b}_1 \in \mathbb{R}^d, W_2 \in \mathbb{R}^{d\times 10}, b_2 \in \mathbb{R}^{10}$, d is the hidden layer size and g is an activation function (ReLU, tanh, sigmoid, etc.). We will use the cross-entropy loss function:

$$\begin{aligned}\mathcal{L}(\mathrm{y}, \hat{\mathrm{y}}) &= -\sum_{k=1}^{k=10} \mathbb{1}_{\{y=k\}} log\ p(y = k|\mathrm{x}; W_1, b_1, W_2, b_2)\\ &= -\sum_{k=1}^{k=10} \mathbb{1}_{\{y=k\}} log\ \hat{y}_k\end{aligned}$$

Derive the gradients of $\mathcal{L}(\mathrm{y}, \hat{\mathrm{y}})$ w.r.t the parameters W_1, b_1, W_2, b_2. You do not need to submit anything for this part.

Part 3b - Neural Network with One Hidden Layer (50 Pts)

Now you will need to implement a NN with one hidden layer and train it on the Fashion-MNIST dataset. All the points mentioned in part 2 are relevant here as well. For example, you will need to write the code for generating predictions, computing gradients, tracking the loss function, etc.

Experiment with different hidden layer sizes, activation functions, batch size, learning rate, regularization coefficient, and the effect of applying dropout on the hidden layer. Write your findings in a report (several lines describing your main insights should suffice). Submit a file named NN_pred.csv with the predictions of your best model only on test.csv. The file should contain only the predictions (no commas, column header, line numbers etc.), each one in a new line. Make sure that the order of the predictions matches the order of the examples in test.csv. Your grade may also be determined by the quality of the predictions. Add to the report a plot of the train and validation loss and accuracy curves as a function of the number of epochs of your best model.

Final Remarks

- In this exercise you are allowed to use only the following external packages: Numpy, Pandas - for reading data from files and writing data to files, and Matplotlib for plotting purposes.
- You need to submit: a report, your code, lr_pred.csv, and NN_pred.csv. Add to the report clear instructions how to run your code.
  1. Code should be submitted in .py or .ipynb file format only.
  2. Add to the report your name and id.

3. Pack your submission files with your favorite file archiver (e.g., .rar, .zip).
4. The archive name should be your ID. If the exercise is done in pairs, the name of the file should be in the following format ID1_ID2. Only one member needs to submit the solution.