CSCI 1051 Problem Set 1

January 12, 2025

Submission Instructions

Please upload your solutions by **5pm Friday January 10, 2025.**

- You are encouraged to discuss ideas and work with your classmates. However, you **must acknowledge** your collaborators at the top of each solution on which you collaborated with others and you **must write** your solutions independently.
- Your solutions to theory questions must be written legibly, or typeset in LaTeX or markdown. If you would like to use LaTeX, you can import the source of this document here to Overleaf.
- I recommend that you write your solutions to coding questions in a Jupyter notebook using Google Colab.
- You should submit your solutions as a **single PDF** via the assignment on Gradescope.

Problem 1: Linear Regression

Consider a d-dimensional multivariate linear regression problem with n samples. Each sample i consists of a data vector $\mathbf{x}^{(i)} \in \mathbb{R}^d$ and a label $y^{(i)} \in \mathbb{R}$. From these samples, let $\mathbf{X} \in \mathbb{R}^{d \times n}$ be the data matrix where the ith row consists of the ith data vector. Similarly, let $\mathbf{y} \in \mathbb{R}^n$ be the target vector where the ith entry consists of the ith label.

When we solve the multivariate linear regression problem, we find the coefficients

$$\mathbf{w}^* = \arg\min_{\mathbf{w} \in \mathbb{R}^d} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2. \tag{1}$$

Part A: Computing the Optimal Solution

Using the strategy described in class, show that

$$\mathbf{w}^* = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}. \tag{2}$$

Part B: The Optimal Solution on Real Data

Load a real regression dataset and compute the optimal coefficients $\mathbf{w}^*$.

I recommend:

- coding in a Colab notebook,
- using the scikit-learn regression datasets such as the diabetes dataset, and
- adapting the example given in the documentation at the above link.

Part C: scikit-learn on Real Data

On the same regression dataset from Part B, use a linear regression solver to compute the optimal coefficients.

I (still) recommend:

- coding in a Colab notebook,
- using the scikit-learn linear regression solver,
- adapting the example given in the documentation at the above link.

Sanity Check: Ensure that both methods of solving the same regression problem result in the same coefficients.

Problem 2: Logistic Regression

Consider a d-dimensional multivariate logistic regression problem with a single labelled pair. The data vector is $\mathbf{x} \in \mathbb{R}^d$ and the label is $y \in \{0, 1\}$.

Consider a logistic regression model with weights $\mathbf{w} \in \mathbb{R}^d$. Recall that the sigmoid function is given by $\sigma(z) = \frac{1}{1+e^{-z}}$. Then the model is $\sigma(\mathbf{w} \cdot \mathbf{x})$. The cross entropy loss is given by

$$\mathcal{L}(\mathbf{w}) = -\left[y \log(\sigma(\mathbf{w} \cdot \mathbf{x})) + (1 - y) \log(1 - \sigma(\mathbf{w} \cdot \mathbf{x}))\right] \tag{3}$$

Part A: Computing the Gradient

Show that

$$\nabla_w \mathcal{L}(\mathbf{w}) = (\sigma(\mathbf{w} \cdot \mathbf{x}) - y)\mathbf{x}. \tag{4}$$

Hint: First, show that $\frac{\partial}{\partial z}\sigma(z) = \sigma(z)(1 - \sigma(z))$. Next, apply the chain rule.

Part B: Optimization?

Why can't we (easily) compute the exact solution in closed form? What should we do instead?

Part C: scikit-learn on Real Data

Load a real classification dataset and compute the optimal coefficients for logistic regression.

I (still) recommend:

- coding in a Colab notebook,
- using the scikit-learn logistic regression solver,
- adapting the example given in the documentation at the above link.

Problem 3: Neural Networks

In this problem, we will gain familiarity building and training neural networks in PyTorch. Please use the neural network demo as a starting point.

Hint: I suggest making modular changes to the demo. That is, implement the changes for one part and make sure everything still runs before moving on.

Part A: Different Data

In the demo, we used the MNIST dataset. Load a `torchvision` dataset of your choice from this list.

Part B: Different Architecture

In the demo, we used a single layer neural network with no activation. Modify the `Network` class to add two new layers with ReLU activations between them.

Part C: Evaluation During Training

We only evaluated our model on the data that we used to train it. But then how do we know if it is overfitting to the training data or actually learning? Load a testing set and plot the loss on the training data and on the testing data by epoch.

How does the training loss compare to the test loss?

Problem 4: Optimizers

In this problem, we will compare the performance of several optimizers.

Part A: Small Dataset

In order to test optimizers, we'll want a dataset that we can quickly train on. You can find a new dataset or subsample from the existing dataset you selected for the last assignment.

Part B: Training Function

Write a function that takes `num_epochs` and an optimizer class e.g., `torch.optim.SGD`. Your function should initialize a model (new random weights) and train it with an instance of the optimizer class. During training, record the *test* loss for each epoch and return it from the function call.

Part C: Optimizer Comparison

Run your function on at least three different optimizers selected from `torch.optim`. In order to remove the effect of the randomness in the training process, train with each optimizer at least 5 times and average the resulting test losses by epoch. Finally, plot the *average* test loss by epoch for each optimizer you selected.