CSCI 1051 Homework 1

January 11, 2023

Submission Instructions

Please upload your solutions by **5pm Friday January 13, 2023.** Remember you have 24 hours no-questions-asked *combined* lateness across all assignments.

- 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 typeset in LaTeX or markdown. I strongly recommend uploading the source LaTeX (found here) to Overleaf for editing.
- Your solutions to coding questions must be written in a Jupyter notebook. I strongly suggest working with colab as we do in the demos.
- You should submit your solutions as a **single PDF** via the assignment on Canvas.

Problem 1 (from January 5)

Consider a linear regression problem as described in class. We have n labelled data points $(\mathbf{x}^{(i)}, y^{(i)})$ where $i \in [n]$, $\mathbf{x}^{(i)} \in \mathbb{R}^d$, and $y \in \mathbb{R}$. Recall our goal is to find a function $f : \mathbb{R}^d \to \mathbb{R}$ so that $f(\mathbf{x}^{(i)}) \approx y^{(i)}$. We choose to model $f(\mathbf{x}) = \langle \mathbf{x}, \mathbf{w} \rangle$ where we have weights $\mathbf{w} \in \mathbb{R}^d$.

We also choose the following loss function

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{n}(y^{(i)} - \langle \mathbf{x}^{(i)}, \mathbf{w} \rangle)^2 = \frac{1}{2}\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2$$

where vector $\mathbf{y} \in \mathbb{R}^n$ contains the labels and matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$. We build $\mathbf{X}$ so that its ith row is ${\mathbf{x}^{(i)}}^\top$.

In class, I waved my hands and said that the gradient of the loss function is

$$\nabla \mathcal{L}(\mathbf{w}) = -\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\mathbf{w}). \tag{1}$$

Your assignment is to show why this is true.

Part 1

Prove that

$$\frac{\partial \mathcal{L}(\mathbf{w})}{\partial w_j} = -(\mathbf{X}^\top)^j(\mathbf{y} - \mathbf{X}\mathbf{w})$$

where $(\mathbf{X}^\top)^j$ denotes the jth row of $\mathbf{X}^\top$ (and the jth column of $\mathbf{X}$).

Part 2

Argue why Part 1 implies Equation 1.

Problem 2 (from January 9)

In class, we saw several activation functions. Consider the sigmoid function defined by

$$\sigma(z) = \frac{1}{1 + e^{-z}}.$$

Part 1

Show that

$$\sigma'(z) = \frac{d\sigma(z)}{dz} = \sigma(z)(1 - \sigma(z)).$$

Part 2

In the demo, we implemented several operations for our `Value` class including ReLU. Your assignment is to modify the notebook from the demo so the `Value` class supports sigmoid. Then, modify the `Net` architecture at the end of the notebook to use sigmoid instead of ReLU. How does the prediction quality for the moons dataset change if we use sigmoid instead of ReLU?

Problem 3 (from January 10)

In the first demo, we classified FashionMNIST images with a logistic regression architecture. Now, we'll use a neural network to improve the performance.

Repeat the same experiment in the demo but now use a (dense) neural network with three hidden layers. The first layer should have 256 neurons, the second should have 128, and the third should have 64, all with ReLU activations. Remember to then use one last layer so the output is 10-dimensional (for the 10 classes). Display train- and test-loss curves. You may have to tweak the total number of training epochs to get reasonable performance.

Finally, sample any three images from the validation set and visualize the predicted class probabilities for each sample. Comment on what you observe from the plots.

Problem 4 (from January 11)

In this problem, we'll apply several 3×3 filters to your favorite (small) image. A lot of deep learning—especially work on images— requires putting together functions from different libraries and working with large tensors. I want you to practice these skills in this problem so I'm purposefully asking you to build the code from scratch (with the help of the internet, of course).

Part 1

The first part of the problem is to write down the 3×3 weights for the following filters: a *blurring* filter (this averages pixels), a *horizontal sharpening* filter (changes in the horizontal direction of an image become more pronounced), and a *diagonal sharpening* filter (changes from bottom-left to top-right become more pronounced).

Part 2

Now, I want you to write code to do the following

- import your favorite small picture from the internet (no more than 256×256 pixels),
- convert the image into a tensor so we can apply a filter,
- pick your favorite filter from part 1 and apply it to your image,
- your filter will likely change the range of your pixels so you'll need to bring them back to the original range before plotting,
- plot your filtered image and comment on how it differs from the original image.

Hint: If you are using a color image, then you need to apply the convolution to each channel (red, green, and blue) separately.

Problem 5 (from January 12)

In this problem, we'll investigate how adding a regularization term impacts the weights. Assume we have put all our weights into a long vector $\mathbf{w}$ and our loss (without the regularization term) is $\mathcal{L}(\mathbf{w})$. Now our regularized loss (with the regularization term) is $\mathcal{L}(\mathbf{w}) + \lambda\|\mathbf{w}\|_2^2$.

Then

1. Write down the gradient descent update rule for this loss.
2. Rewrite the update so that it's clear how the weights are "shrunk" or "decayed" by a multiplicative factor.
3. What mathematical inequality should λ satisfy so that the weights remain stable (rather than swapping sign at every step)?