Homework 2
CS 5787 Deep Learning
Spring 2023

Instructor: Prof. Alejandro (Alex) Jaimes

Discussion Section Leader: Jack Morris

TAs: Andrew Bennett, Yair Schiff

Graders: Eylul Ertay, Anastasiia Sorokina, Zhengchun Shang, Jamie Cao

Due: 3/3/23

Problem 1 - Linear Algebra Review #1

Let $\mathbf{A} = \begin{pmatrix} 1 & 2 \\ 5 & -2 \end{pmatrix}$ and $\mathbf{B} = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 0 & 2 \end{pmatrix}$
.

Part 1 (3 points)

Compute $Trace(\mathbf{A})$.

Answer:
$Trace(\mathbf{A}) =?$.

Part 2 (4 points)

Compute $Trace(\mathbf{B}\mathbf{B}^T)$.

Answer:
$Trace(\mathbf{B}\mathbf{B}^T) =?$.

Problem 2 - Linear Algebra Review #2 (4 points)

Let $\mathbf{A} \in \mathbb{R}^{n \times n}$, $\mathbf{D} \in \mathbb{R}^{n \times n}$, $\mathbf{b} \in \mathbb{R}^n$, and $\mathbf{x}^T(\mathbf{A} + \mathbf{D}) = \mathbf{b}^T$. Use the matrix inverse to solve for $\mathbf{x}$ and simplify. Assume that $\det(\mathbf{A} + \mathbf{D}) \neq 0$.

Answer:
$\mathbf{x} =$

Problem 3 - Linear Algebra Review #3

Let matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ and matrix $\mathbf{B} \in \mathbb{R}^{n \times m}$, where $n \neq m$.

Part 1 (1 point)

If it is possible to compute the matrix product $\mathbf{AB}$, give the size of the matrix produced. Otherwise, write, 'Not possible.'

Answer:

Part 2 (1 point)

If it is possible to compute the matrix product $\mathbf{BA}$, give the size of the matrix produced. Otherwise, write, 'Not possible.'

Answer:

Problem 4 - Regression

Tastes in music have gradually changed over the years. Based on this observation, our goal is to predict the year of a song based on its timbre summary features. This dataset is from the 2011 Million Song Challenge dataset: `https://labrosa.ee.columbia.edu/millionsong/`

To establish a baseline, we wish to build a linear model that predicts the year. Given an input $\mathbf{x} \in \mathbb{R}^{90}$, we want to find parameters for a model $\hat{y} = \text{round}\left(f\left(\mathbf{x}\right)\right)$ that predicts the year, where $\hat{y} \in \mathbb{Z}$.

We are going to explore three shallow (linear) neural network models with different activation functions for this task.

To evaluate the model, you must round the output of your linear neural network. You then compute the mean squared error.

Part 1 - Load and Explore the Data (5 points)

Download the music year classification dataset `music-dataset.txt` at `https://archive.ics.uci.edu/ml/datasets/yearpredictionmsd`. Each row is an instance. The first value is the target to be predicted (a year), and the remaining 90 values in a row are all input features. Split the dataset into train and test partitions by treating the first 463,714 examples as the train set and the remaining examples as the test set. The first 12 dimensions are the average timbre and the remaining 78 are the timbre covariance in the song.

Write a function to load the dataset, e.g.,
`trainYears, trainFeat, testYears, testFeat = loadMusicData(fname, addBias)` where `trainYears` has the years for the training data, `trainFeat` has the features, etc. `addBias` appends a '1' to your feature vectors. Each of the returned variables should be NumPy arrays.

Write a function `mse = musicMSE(pred, gt)` where the inputs are the predicted year and the "ground truth" year from the dataset. The function computes the mean squared error (MSE) by rounding `pred` before computing the MSE.

Load the dataset and discuss its properties. What is the range of the variables? How might you normalize them? What years are represented in the dataset?

What will the test mean squared error (MSE) be if your classifier always outputs the most common year in the dataset?

Solution:

Part 2 - Classification? (5 points)

This problem could have been posed as a classification problem by treating each year as a category. What would be the problems with this approach? Support your argument by analyzing a bar chart with the year as the x-axis and the number of examples for that year as the y-axis.

Solution:

Part 3 - Implementing Ridge (Tikhonov) Regression (6 points)

Please read the entire assignment. If you write your code correctly you do not need to implement stochastic gradient descent with mini-batches multiple times. You can implement it once in a function, then you just specify the input features, target values, weight decay factor, the loss type (L2, count, or cross-entropy), form of weight decay (none, L2, or L1), and whether to use momentum or not (momentum is used a later problem). It is up to you, but you can make the assignment much easier by doing this.

Implement stochastic gradient descent with mini-batches to minimize the loss and evaluate the train and test MSE.

Tune the learning rate and weight decay factor. Subsequently, show the train and test loss as a function of epochs, where the number of epochs should be chosen to ensure the train loss is minimized.

This problem can be solved directly using the pseudoinverse. Let $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]$ and $\mathbf{y} = [y_1, y_2, \ldots, y_N]^T$, then the pseudoinverse solution is given by

$$\mathbf{w} = \left(\mathbf{X}\mathbf{X}^T + \alpha\mathbf{I}\right)^{-1}\mathbf{X}\mathbf{y}.$$

Compare both solutions.

Tip: Debug your models by using an initial training set that only has about 100 examples and make sure your train loss is going down.

Tip: If you don't use a constant (bias), things will go very bad. If you don't normalize your features by 'z-score' normalization of your data then things will go very badly. This means you should compute the training mean across feature dimensions and the training standard deviation, then normalize by subtracting the training mean from both the train and test sets, and dividing both sets by the training standard deviation.

Solution:

Part 4 - Implementing L_1 Weight Decay (6 points)

Implement lasso regression using stochastic gradient descent with mini-batches. Tune the learning rate and weight decay factor. Subsequently, show the train and test loss as a function of epochs, where the number of epochs should be chosen to ensure the train loss is minimized.

Solution:

Part 5 - Implementing Count Regression (6 points)

Implement count regression. Minimize the loss using SGD with mini-batches to find an optimal $\mathbf{w}$. Compute the gradient update rule and show it. Plot the train and test loss as a function of epochs. Compute the train and test MSE using the function we created earlier.

Solution:

Part 6 - Model Comparison (10 points)

Plot a histogram of the weights for the ridge, lasso, and count regression models. Discuss how the weights differ.

Discuss and compare the behaviors of the models. Are there certain periods (ranges of years) in which models perform better than others? Where are the largest errors across models. Did regularization help for some models but not others?

Solution:

Problem 5 - Softmax Properties

Part 1 (7 points)

Recall the softmax function, which is the most common activation function used for the output of a neural network trained to do classification. In a vectorized form, it is given by

$$\text{softmax}\left(\mathbf{a}\right) = \frac{\exp\left(\mathbf{a}\right)}{\sum_{j=1}^{K} \exp\left(a_j\right)},$$

where $\mathbf{a} \in \mathbb{R}^K$. The exp function in the numerator is applied element-wise and a_j denotes the j'th element of $\mathbf{a}$.

Show that the softmax function is invariant to constant offsets to its input, i.e.,

$$\text{softmax}\left(\mathbf{a} + c\mathbf{1}\right) = \text{softmax}\left(\mathbf{a}\right),$$

where $c \in \mathbb{R}$ is some constant and $\mathbf{1}$ denotes a column vector of 1's.

Solution:

Part 2 (3 points)

In practice, why is the observation that the softmax function is invariant to constant offsets to its input important when implementing it in a neural network?

Solution:

Problem 6 - Implementing a Softmax Classifier

For this problem, you will use the 2-dimensional Iris dataset. Download `iris-train.txt` and `iris-test.txt` from Canvas. Each row is one data instance. The first column is the label (1, 2 or 3) and the next two columns are features.

Write a function to load the data and the labels, which are returned as NumPy arrays.

Part 1 - Implementation & Evaluation (20 points)

Recall that a softmax classifier is a shallow one-layer neural network of the form:

$$P\left(C = k|\mathbf{x}\right) = \frac{\exp\left(\mathbf{w}_k^T\mathbf{x}\right)}{\sum_{j=1}^{K}\exp\left(\mathbf{w}_j^T\mathbf{x}\right)}$$

where $\mathbf{x}$ is the vector of inputs, K is the total number of categories, and $\mathbf{w}_k$ is the weight vector for category k. The bias has been omitted, but you can incorporate it by appending a '1' to each $\mathbf{x}$.

In this problem you will implement a softmax classifier from scratch. **Do not use a toolbox.** Use the softmax (cross-entropy) loss with L_2 weight decay regularization. Your implementation should use stochastic gradient descent with mini-batches and **momentum** to minimize softmax (cross-entropy) loss of this single layer neural network. To make your implementation fast, do as much as possible using matrix and vector operations. This will allow your code to use your environment's BLAS. Your code should loop over epochs and mini-batches, but do not iterate over individual elements of vectors and matrices. Try

to make your code as fast as possible. I suggest using profiling and timing tools to do this.

Train your classifier on the Iris dataset for 1000 epochs. You should either subtract the mean of the training features from the train and test data or normalize the features to be between -1 and 1 (instead of 0 and 1). Initialize your weights from a Gaussian distribution Hand tune the hyperparameters (i.e., learning rate, mini-batch size, momentum rate, and L_2 weight decay factor) to achieve the best possible training accuracy. During a training epoch, your code should compute the mean per-class accuracy for the training data and the loss. After each epoch, compute the mean per-class accuracy for the testing data and the loss as well. **The test data should not be used for updating the weights.**

After you have tuned the hyperparameters, generate two plots next to each other. The one on the left should show the cross-entropy loss during training for both the train and test sets as a function of the number of training epochs. The plot on the right should show the mean per-class accuracy as a function of the number of training epochs on both the train set and the test set.

What is the best test accuracy your model achieved? What hyperparameters did you use? Would early stopping have helped improve accuracy on the test data?

Solution:

Part 2 - Displaying Decision Boundaries (10 points)

Plot the decision boundaries learned by softmax classifier on the Iris dataset, just like we saw in class. On top of the decision boundaries, generate a scatter plot of the training data. Make sure to label the categories.

Solution:

Problem 7 - Classifying Images (10 points)

Recall the CIFAR-10 dataset from Homework 0. Using the softmax classifier you implemented, train the model on CIFAR-10's training partitions. To do this, you will need to treat each image as a vector. You will need to tweak the hyperparmaters you used earlier.

Plot the training loss as a function of training epochs. Try to minimize the error as much as possible. What were the best hyperparmeters? Output the final test accuracy and a normalized 10×10 confusion matrix computed on the test partition. Make sure to label the columns and rows of the confusion matrix.

Solution:

Linear Models for Regression Code Appendix

Softmax Classifier Code Appendix