Machine Learning

Lab 3

Instructions

This document lists the details of what to do and what to include in the written report. Sample code for both MATLAB and Python is provided and can give hints on how to proceed. Follow the code in your chosen language and the comments for tips. The parts in the sample code that you need to fill in is marked with:

```
# ===================== YOUR CODE HERE ===================== #
```

A written report should be handed in before the end of the course. The report should include the answers to all the asked questions in this document, all the plots that are created, as well as the code used.

Tips: Google Colab or Jupyter notebooks for python or Matlab LiveScript are useful tools for facilitating this.

The report is graded with either PASS or PASS WITH DISTINCTION. Complete all the exercises to receive the higher grade.

Preparation

Some useful resources:

- Notes on NN and backprop

Exercises for Pass (G)

1 Neural Network Backpropagation

Implement the forward and backward pass of the neural network with 1 hidden layer. We initialize the network parameters assuming a neural network with:

- 8 input units
- 5 hidden units
- 10 output units
- 100 samples as batch size

Steps:

1. Perform the feed forward calculations to get the variables $z^{(2)}$, $a^{(2)}$, $z^{(3)}$, and $a^{(3)}$. Use the sigmoid activation function in the hidden layer and the output layer.
2. Calculate the cost function, J. Use the squared error cost function.
3. Calculate the error terms $\delta^{(3)}$ and $\delta^{(2)}$. Remember to include the derivative of the sigmoid function.
4. Use the error terms to calculate the gradients for the weights and biases. Concatenate all the parameters into a flat vector.
5. Use gradient checking to verify that your implementation is correct.

1.1 Report

Show the code of the cost neural network function and report what diff value you got.

1.2 Mathematical background

Forward Pass Calculations

Given input x, weights $W^{(1)}$ and $W^{(2)}$, and biases $b^{(1)}$ and $b^{(2)}$, the forward pass is:

$$\begin{aligned} z^{(2)} &= W^{(1)}x + b^{(1)} \\ a^{(2)} &= \sigma(z^{(2)}) \\ z^{(3)} &= W^{(2)}a^{(2)} + b^{(2)} \\ a^{(3)} &= \sigma(z^{(3)}) \end{aligned}$$

where σ is the sigmoid activation function.

Loss Calculation

For **a single training example**, given the actual label y and the predicted output $a^{(3)}$, the loss (using mean squared error) is:

$$J = \frac{1}{2}\|a^{(3)} - y\|^2 \tag{1}$$

The objective during training is to minimize this loss by adjusting the network parameters (weights and biases).

Weight Update Rule

For each weight w_i:

$$w_i^{new} = w_i^{old} - \alpha \frac{\partial J}{\partial w_i}$$

Where J is the cost function and α is the learning rate.

Gradient Calculation

Since J is a function of $a^{(3)}$ and $a^{(3)}$ is a function of $z^{(3)}$, the chain rule for the last layer gives:

$$\frac{\partial J}{\partial W^{(2)}} = \frac{\partial J}{\partial a^{(3)}} \times \frac{\partial a^{(3)}}{\partial z^{(3)}} \times \frac{\partial z^{(3)}}{\partial W^{(2)}}$$

Starting with the first partial derivative, since the cost function is given by Eq.1, we get:

$$\frac{\partial J}{\partial a^{(3)}} = a^{(3)} - y$$

The second partial derivative, the derivative of the sigmoid function, is:

$$\frac{\partial a^{(3)}}{\partial z^{(3)}} = a^{(3)}(1 - a^{(3)})$$

The third partial derivative is simply:

$$\frac{\partial z^{(3)}}{\partial W^{(2)}} = a^{(2)}$$

For the hidden layer:

$$\frac{\partial J}{\partial W^{(1)}} = \frac{\partial J}{\partial a^{(3)}} \times \frac{\partial a^{(3)}}{\partial z^{(3)}} \times \frac{\partial z^{(3)}}{\partial a^{(2)}} \times \frac{\partial a^{(2)}}{\partial z^{(2)}} \times \frac{\partial z^{(2)}}{\partial W^{(1)}}$$

The derivative of the sigmoid for the hidden layer is:

$$\frac{\partial a^{(2)}}{\partial z^{(2)}} = a^{(2)}(1 - a^{(2)})$$

With the above derivatives at hand, by defining the term $\delta^{(l)}$ as the "error" of layer l, we can simplify the gradient calculations. We now recognize:

$$\begin{aligned} \delta^{(3)} &= (a^{(3)} - y) \times a^{(3)}(1 - a^{(3)}) \\ \delta^{(2)} &= (W^{(2)})^T \delta^{(3)} \times a^{(2)}(1 - a^{(2)}) \end{aligned}$$

Finally, the gradients for the network weights become:

$$\begin{aligned} \frac{\partial J}{\partial W^{(2)}} &= \delta^{(3)}(a^{(2)})^T \\ \frac{\partial J}{\partial W^{(1)}} &= \delta^{(2)} x^T \end{aligned}$$

The gradients for the biases simply become:

$$\begin{aligned} \frac{\partial J}{\partial b^{(2)}} &= \delta^{(3)} \\ \frac{\partial J}{\partial b^{(1)}} &= \delta^{(2)} \end{aligned}$$

Averaging the gradients over m number of training examples and generalizing for layer L give:

$$\begin{aligned} \frac{\partial J}{\partial W^{(L)}} &= \frac{1}{m} \sum_{i=1}^{m} \delta_i^{(L+1)} (a_i^{(L)})^T \\ \frac{\partial J}{\partial b^{(L)}} &= \frac{1}{m} \sum_{i=1}^{m} \delta_i^{(L+1)} \end{aligned}$$

Note that $a^{(1)}$ is the input data x.

2 Neural network for handwritten digit classification

In this part, you will train the neural network from part 1 for the multi-class classification of handwritten digits on the small MNIST dataset. The network will include a hidden layer with a sigmoid activation function. The output layer normally consists of a softmax activation function for multi-class classification, but we will use a sigmoid activation function in the output layer for now.

2.1 Report

What accuracy on the train, val, and test set did you get?

3 Implementing an Auto-encoder

In this task, implement a non-regularized auto-encoder, a neural network that replicates its input as output, creating an internal compressed data representation. Initialize network parameters and define the sigmoid activation function. Include forward propagation for data reconstruction and backpropagation for gradient computation.

3.1 Report

Show the code and the gradient difference value.

4 Reconstructing with Auto-encoder

Now you need to train your autoencoder for the smallMNIST data. After that you need to reconstruct the images with your trained auto encoder (use the provided code to plot the original and reconstructed data). If everything goes well your reconstructed data should be similar with the original input data.

4.1 Report

Select good parameters for the number of hidden units in the auto-encoder and the maximum number of iterations to train so that the reconstructions look like the original input data. Show the plot of the original input and the reconstructions.

Exercises for Pass with Distinction (VG)

5 Bias-Variance Analysis on the Number of Hidden Units in a Neural Network

The objective of this analysis is to determine the optimal number of hidden units in the neural network for the task of handwritten digit classification. The optimal number of hidden units should balance the bias-variance tradeoff to achieve the highest possible accuracy on the validation set. Try 10 different number of hidden units and train the network for each number calculate the individual accuracy and plot all the accuracies in a single plot.

5.1 Report

Show which values you selected and the plot with the overall accuracies.

6 Multi-class classification with Softmax classifier

In this part you will implement the softmax classifier and use it on the smallMNIST data set. We will compare the classification accuracy on the test set using 1) the raw input of the smallMNIST data set, and 2) using the activations of the hidden layer from a pre-trained 1-layered auto-encoder.

Steps:

1. Load the smallMNIST dataset. Split it into train, val, and test sets.

2. Implement the softmax classifier. *Tips: Use a similar approach to how you implemented the neural network. Create a function that returns*

the cost value and the gradients and use gradient checking to verify the implementation.

3. Use the training set to train a softmax classifier on the training set. Calculate the accuracy on the test set.

4. Train an auto-encoder to simply reconstruct the training set. Use the encoder part of the trained auto-encoder (W_1 and b_1 to feed-forward the training and test set to get the activations of the hidden layer.

5. use the hidden layer activations for the training set to to train another softmax classifier and use the trained softmax classifier on the hidden layer activations for the test set to calculate the classification accuracy.

6. Compare the results to see if using the representations from the auto-encoder improved the classification accuracy.

7. Try changing the number of hidden units in the auto-encoder to see how the result changes.

6.1 Mathematical Background

The neural network model for multi-class classification has the following architecture:

- An input layer that receives the feature vector $\mathbf{X}$.
- A hidden layer with a sigmoid activation function:

  $$a^{[1]} = \sigma(\mathbf{W}^{[1]}\mathbf{X} + \mathbf{b}^{[1]})$$

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

- An output layer that provides scores for each class. These scores can be transformed into probabilities using either the sigmoid function for binary-like approaches to multi-class problems or the softmax function for typical multi-class scenarios:

  $$h_\theta^{[2]}(\mathbf{x}) = \text{activation}(\mathbf{W}^{[2]}a^{[1]} + \mathbf{b}^{[2]})$$

where activation can be either σ or softmax, defined as:

$$\text{softmax}(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

for each class j in a total of K classes.

6.2 Report

Report the classification accuracies on the test set using the two different methods and discuss if using an auto-encoder as a pre-processing step is useful or not for this problem.

7 MNIST Classification with pre-existing libraries

In this task, you will develop a neural network using pre-existing libraries to classify handwritten digits. You can use PyTorch for python (use Google Colab if you are unable to install it on your computer) or you can use the Deep Learning Toolbox for MATLAB. You can use the provided small MNIST data set or use the full MNIST dataset. The full MNIST dataset comprises of 60,000 training images and 10,000 test images, each of size 28x28 pixels. You will implement your network architecture, train the model, and evaluate its performance on the test set.

1. Select and define your own neural network structure using the framework of your choice. The network should have an input layer sized for the MNIST data, hidden layers with activation functions of your choice, and an output layer with 10 units corresponding to the 10 digit classes and with a softmax activation function (or use the cross-entropy loss).

2. Train your model using an appropriate number of epochs for training. After training, test your network on the test images and report the classification accuracy.

7.1 Report

Submit your neural network code along with the test accuracy achieved. Provide a brief discussion of your neural network choice and choice of hyper-parameters.