ROBOT LEARNING

EXCERCISE- NEURAL NETWORKS

SUMMER SEMESTER 2021

<https://fbe-gitlab.hs-weingarten.de/mat-iki/slam-mat/-/tree/master/nn>

**Problem 1**

In this problem you will use a pre-trained Neural Network to predict the label of MNIST dataset images. The dataset contains 5000 training examples of handwritten digits ranging from digit 0-9 (a subset of the MNIST handwritten digit dataset). The weights matrices the the specified pre-trained Neural Network are also provided. This exercise is developed on the work in Stanford Machine Learning Course [CS229](http://cs229.stanford.edu/" \t "/home/sa0102/Documents\\x/_blank) by Andrew Ng. **Note** that the training data is available in Matlab format, but the user could use any other programming language of their choice to solve the below problems (in that case indicies should be taken care of, Bias term is first index).

1. Load the dataset into an Input Matrix (X). Choose a random index from X and print out the image.
2. Implement the Feed-Forward Prediction Algorithm to predict the class of the random index image in above part of this problem. Has the model predicted it correctly ?
3. Compute the accuracy of the pre-trained model.

**Problem 2**

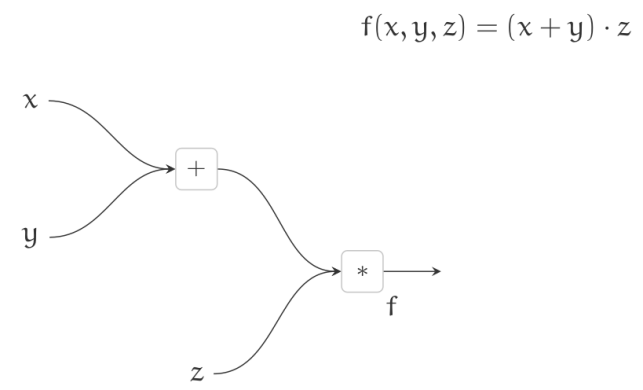
In this problem you will train an Neural Network by Backpropogation method. Same dataset as in Problem 1 is used: The dataset contains 5000 training examples of handwritten digits ranging from digit 0-9.

1. Compute the Loss Function (also called Cost function). As a checkpoint, gradient with the pretrained Theta from Problem 1 should give value of 0.287
2. For intital values of Theta 1 and Theta 2, write to function that intitiates random value Thetas.
3. Compute the Gradient of the Cost function with respect to two Thetas using the theory of backpropogation. Using SciPy, train the find the solution to the Theta arrays.
4. Find the prediction Accuracy with the trained model. This should come about 95 % (mybe be +- 1% difference due to initialization).

**Problem 3**

Just an in previous problems, one needs to compute the Loss function and Gradients in order to optimize the system. With increase in complexity of model, the Gradient formulating and coding process becomes more complicated. It can be made easier with a Python library: [Autograd](https://github.com/HIPS/autograd).

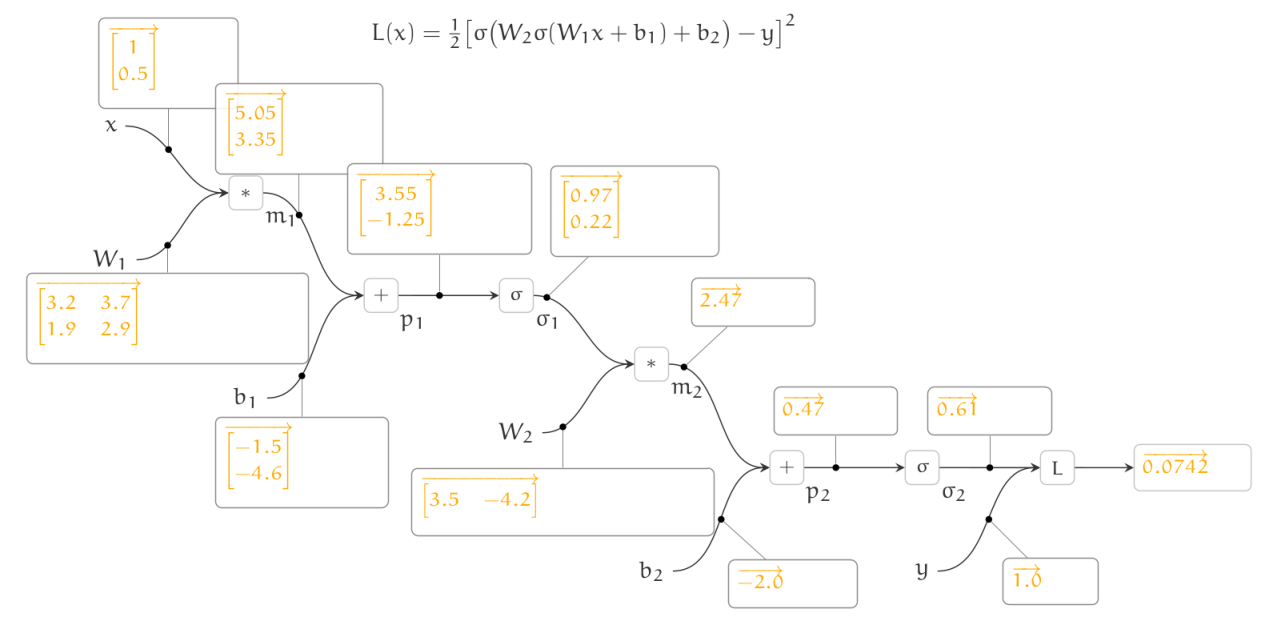
1. Use Autograd to compute gradients for a simple 2 layer Network as shown below. Substitute values as x=2, y=-3 and z=4 to cross check the results obtained in lecture slides.



1. Find the derivates of the Loss function in Problem 2 w.r.t. weight vectors and compare the computed gradientw with the Backpropogation approach.

**Problem 4**

Use a Computation Graph to find the Gradient of each node in below example:



**Problem 5**

Read the following article to understand how can we visualize Neural Networks. We can actually ‚see‘ how one node recognizes the features and uses learned weights to classify objects: <https://distill.pub/2017/feature-visualization/>. The work done is by Olah and Schubert from Google Brain Research Team.

For a quick summary the ppt can be referred: <https://github.com/sa32953/lero>

**Problem 6**

Tinker a NN at Tensorflow. Check out the playground at <http://playground.tensorflow.org.> It is an open source project to better understand and play with configurable simple Neural Network. Have Fun !