Proposal: Machine Learning 1 Project

Mrigank Tiwari, Jacob Maibach

1. **What problem did you select, and why did you select it?**

We have decided to work on the problem of *letter recognition*. We chose the problem since we are interested in image recognition, but would like to start with the basics.
Additionally, the most popular and successful application of backpropagation trained neural networks is in pattern classifications. And because half of our ML1 coursework was focussed on NN, we would like to make the most of this chance to get our hands dirty with this problem.

2. **What database will you use? Is it large enough to train a machine learning algorithm or different algorithms?**

We will use the "Letter Recognition Data Set" from the University of California-Irvine Machine Learning Repository (https://archive.ics.uci.edu/ml/datasets/Letter+Recognition). It contains summaries of 20,000 images of characters. There are 16 variables describing the spatial characteristics of the characters (such as edge counts and x variance). The data set contains 20 different fonts. The researchers distorted images of the original characters in various ways to produce the final images.

This data set should be more than sufficient to train a neural network model, and likely for other models as well. The intentional inclusion of noise in the data (through image distortion) also should make the result more robust, meaning that we would likely need fewer data points to fit a decent model.

3. **What neural network will you use? Will it be a standard form of the network, or will you have to customize it? What algorithms you will be used?**

As per our preliminary discussion and thoughts, we are planning to use a 3 layer Neural network. This comprises 16 dimensional input and 26 output neurons. Additionally, we are considering using 8 neurons in the hidden layer, but we will use the data to determine the optimal number of neurons (cross validation on 20% of the data). The activation function we will use is the sigmoid, which is appropriate since we need a continuous output.

Apart from the neural network, we are also going to attempt to do this classification using one vs all logistic regression for comparison. We will try to incorporate PCA feature selection to try out different models here. We will try to include regularized loss function to improve the model accuracy.
These two machine learning algorithms will allow for a comparative study between logistic regression and neural network for classification problems. And for sure will equip us in practical implementations of algorithms like Backpropagation and Gradient descent.

4. **What software will you use to implement the neural network or different algorithms? Why?**

We have decided to work with Python for this project. The reason we choose Python is because it has vast ML libraries support like scikit-learn and some others. We will most probably be using scikit-learn.

5. **What reference materials will you use to obtain sufficient background on applying the chosen network or algorithm to the specific problem that you selected?**

We will be taking reference of our coursebook “Python Machine Learning”, the research paper “Letter Recognition Using Holland-Style Adaptive Classifiers” (associated with the data set) for reference, and scikit-learn documentation.

6. **How will you judge the performance of the network? What metrics will you use?**

We will use three metrics to judge the performance of the network. The first is simply the mean squared error of prediction (naturally).

The second is a measure of our model’s confidence in its output. Our network outputs values for each letter which describe how likely it is that the input is that letter, and then it classifies via maximizing the likelihood. Ideally, one of these likelihoods is very high and the others are low, but this may not be the case. The confidence measure would be low in the non-ideal case and high in the ideal-case. If a is the actual classification given with likelihood l_a, then the confidence would be

$$c = E[(l_a - l_k)^2]/E[l_k^2]$$

where E denotes the average over k and l_k is the likelihood assigned to the other class k.

Our last metric is a measure of worst comparison. Essentially, we predict that our model will have difficulty distinguishing certain pairs of letters, such as D and O. We want to create a numerical measure of how bad it is at comparing such pairs of letters, and then use the worst as our metric. In particular, we will look at the confusion matrix and take the submatrix corresponding to this pair. The determinant of the submatrix then indicates how much confusion there is: if the model does no better than random at telling the letters apart, the determinant is zero, and if it does perfectly, the determinant is one. Additionally, if the model does worse than random, the determinant is negative.

To apply the metrics, we will use 10-fold cross validation on 80% of our data set. The remaining 20% we will use to choose hyperparameters (particularly, number of neurons in the network).

7. **Provide a rough schedule for completing the project.**

We will finalize our models by April 11th, complete whatever programming and implementation we need by April 18th, and complete the analysis by April 23rd.