Introduction
For this project, we are given a dataset of handwritten digits and drawings, and must create a Multilayer Perceptron to classify. The datasets given are:

mnist_all: Dataset of handwritten digits, with 10 matrices for testing and 10 matrices for training, which correspond to 10 digits.
mnist_sample: Initial calibration for the mnist
AI_quick_draw: Dataset of hand drawn images

General Dataset Information
The datasets consist of an n x 784 matric, where each row is one image of 28 x 28, flattened into a 784 length vector. We must reshape these vectors to view the image.

MNIST Dataset Information
Each pixel of the image is represented by an integer between 0 and 255, where 0 is black, 255 is white and the inbetween is greyscale. We will use the training set to train the NN, and the test set to estimate the hyper-parameters of the network (regularization constant lambda, number of hidden units).

Neural Network Information
The NN will consist of 3 layers:

- Layer 1 comprises of (d + 1) units, each represents a feature of the image, with the +1 for bias.
- Layer 2 is the hidden layer of the NN. We must choose an appropriate amount of hidden units for this layer.
- Layer 3 is the output layer. The value of the n unit in the output layer represents the probability of a certain hand-written image belongs to digit n. Since we have 10 possible digits,there are 10 units in the output layer. In this document, we denote k as the number of output units in output layer.

There will be 2 weight matrices, one for the weights from the input layer to the hidden layer, and one for the hidden layer to the output layer.

Neural Network Learning
Given the parameters of the NN (the weight matrices) and a feature vector x, we want to find the probability that this feature vector belongs to a particular digit. The way we will compute the value of the output units will be the following: Take the sum of each corresponding weight of a hidden unit dotted(multiplied) with the corresponding feature. We will use an activation function (sigmoid) of the following: $1 \div (exp(-a_j)$ Where a is the linear combination of input data. We then apply the sigmoid function to the outputs to get the value one vs all binary classification.

Backpropagation

The error function we will use is the *negative log-likelihood error function* which can be summarized to the following: $J(W^1, W^2) = 1/n \, * \sum_{i=1}^{n} J_i(W^1, W^2)$ where

$J_i(W^1, W^2) = -\sum_{l=1}^{k}(y_{il} ln(o_{il}) + (1 - y_{il}) ln(1 - o_{il}))$ Where o_{il} is the lth output value for the ith data examples

To learn model parameters we will initialize the weights to some random numbers and compute the output value, then compute the error in prediction, transmit the error backwards and update the weights accordingly.

We will use gradient descent to update each weight with the following rule:

$w^{new} = w^{old} - \gamma \nabla J(w^{old})$ where γ is the learning rate

Regularization

To avoid overfitting (the model picking up random fluctuations in the training data and learning these fluctuations as concepts by the model) we must rewrite the objective function with a regularization constant.

$$J''(W^1,\ W^2) = J(W^1, W^2) + \lambda/2n((\sum_{j=1}^{m} \sum_{p=1}^{d+1} (W_{ij}{}^{(1)})^2 + \sum_{l=1}^{k} \sum_{j=1}^{m+1} (W_{lk}{}^{(2)})^2))$$