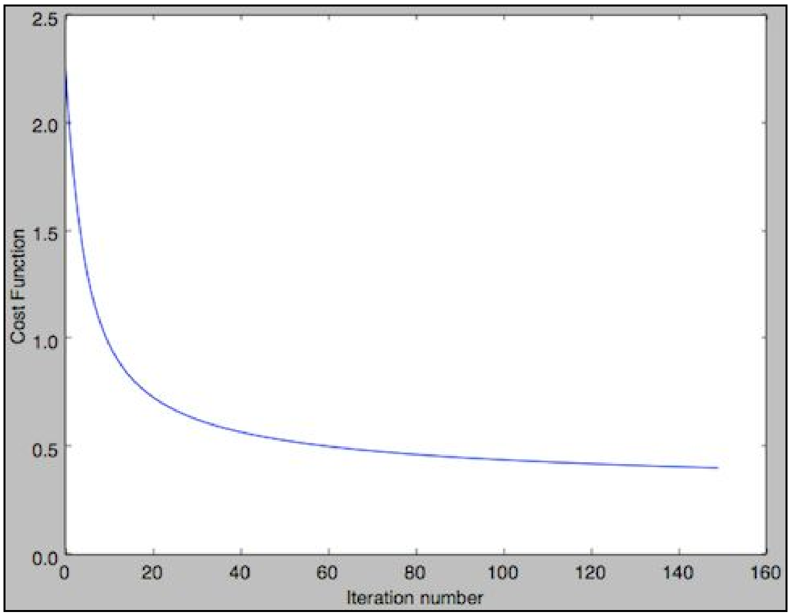
Birkan Uzun
birkanu@mit.edu
No Collaborators

6.036 Project 2: Handwritten Digit Recognition

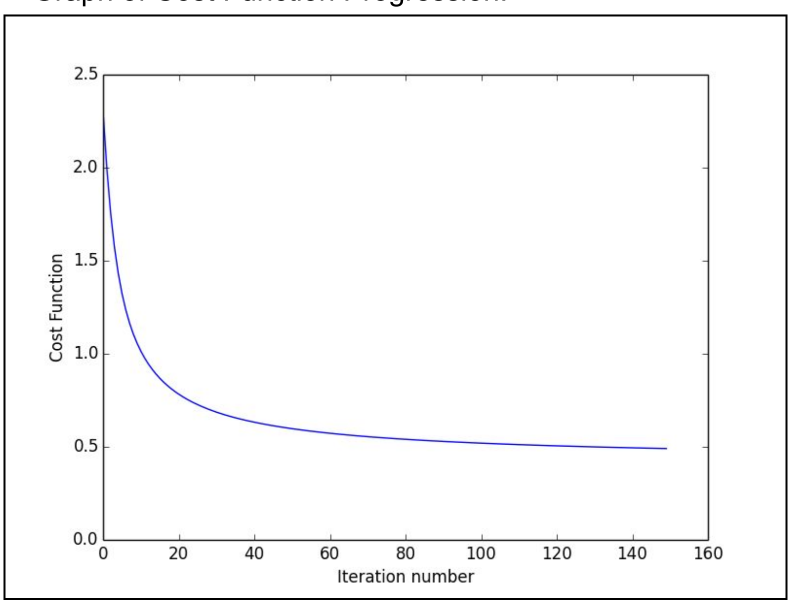
1) Multinomial/Softmax Regression and Gradient Descent

Final Test Error = 0.1005.
Graph of Cost Function Progression:



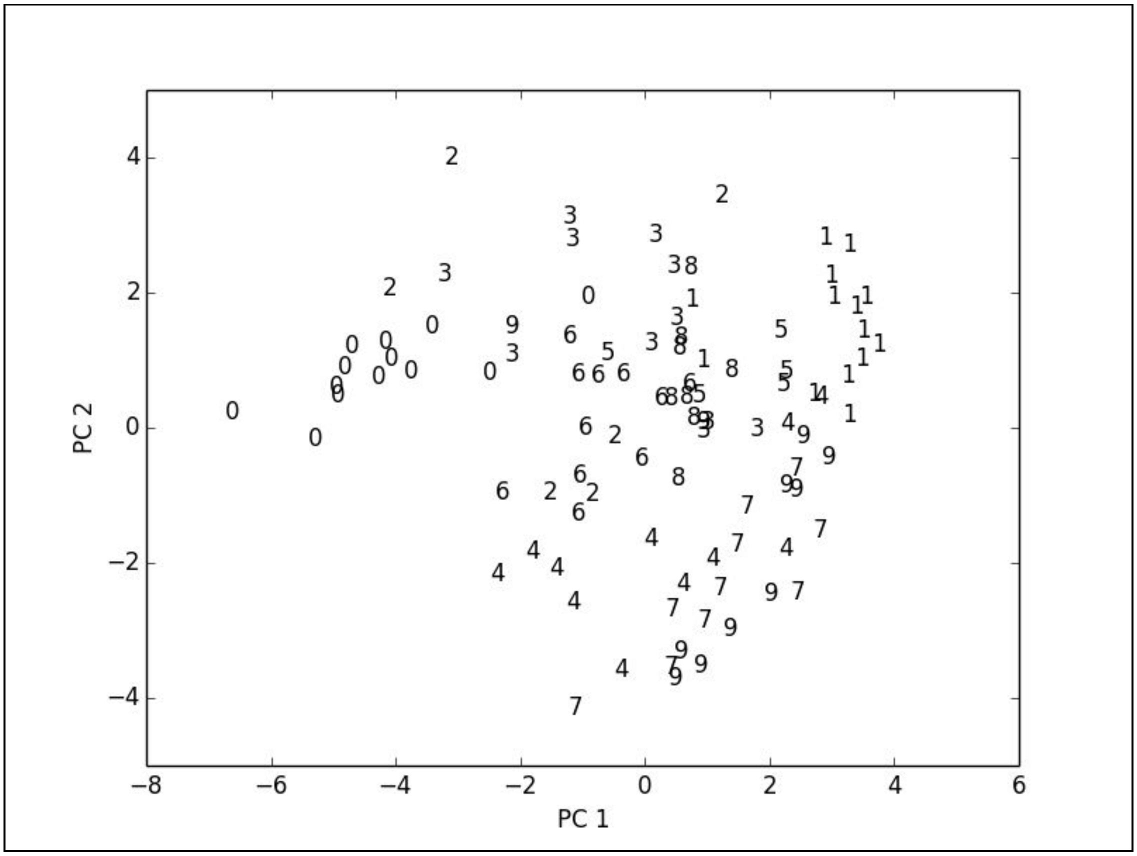
2) Using Manually Crafted Features

2. Test Error = 0.136.
   Graph of Cost Function Progression:



Birkan Uzun
birkanu@mit.edu
No Collaborators

3. Visualization of the first 100 MNIST images, as represented in the space by the first 2 principal components of the training data:



4. Plots of the reconstructions of the first two MNIST images (from their 20-dimensional PCA-representations) alongside the originals:

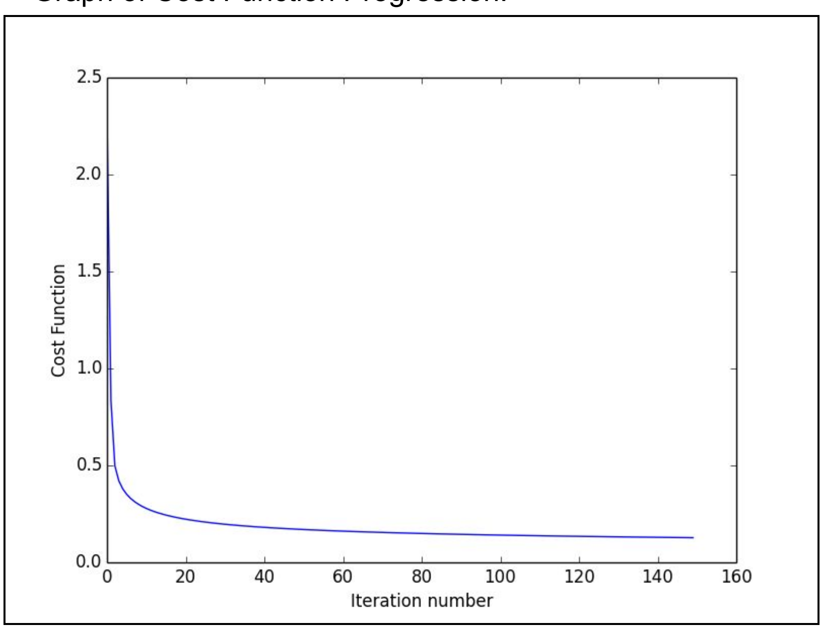
Birkan Uzun
birkanu@mit.edu
No Collaborators

5. $x = [x_1, x_2]$ so that

$$
\begin{aligned}
(1 + x \cdot x')^2 &= (1 + x_1x_1' + x_2x_2')^2 \\
&= 1 + 2(x_1x_1' + x_2x_2') + (x_1x_1')^2 + 2(x_1x_1')(x_2x_2') + (x_2x_2')^2 \\
&= [1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, \sqrt{2}x_1x_2, x_2^2] \cdot [1, \sqrt{2}x_1', \sqrt{2}x_2', x_1'^2, \sqrt{2}x_1'x_2', x_2'^2]
\end{aligned}
$$

So, $\phi(x) = [1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, \sqrt{2}x_1x_2, x_2^2]^T$. The `quadraticFeatures` function is implemented in `features.py`.

6. Test Error = 0.0368.
Graph of Cost Function Progression:



Birkan Uzun
birkanu@mit.edu
No Collaborators

3) Classification Using Deep Neural Networks

1. (b) I got a test accuracy of 98.29%. I increased the number of neurons in the first fully-connected layer to 512, and I increased the learning rate to 0.1. I realized that increasing the learning rate by itself (without changing any other parameters) wasn't enough to get an accuracy above 98%. However, when tweaked together with the number of neurons in the first hidden layer, I observed a better test accuracy.

2. (a) The code for my CNN is in `mnist_nnet_cnn_skeleton.py`. It achieves a training accuracy of 94.50% and a validation accuracy of 98.36%.