Atanas Delevski
ECE 407 HW5
3/29/2020

Problem 1:

For this problem, we were given the task of using Logistic Regression to train a model to recognize the MNIST database digits.

I decided to use Python for this problem, as I am more familiar with Python than I am with MATLAB, and I also believe that knowing Python is more useful than MATLAB. We were told that if we wanted to use Python for this project, we would have to describe the libraries we chose to use and why we used them. I will do this here:

To begin with, I built a virtual environment in Pycharm, and I installed the following libraries:

1. *JupyterLab* - I used jupyter notebooks for the convenience of having the independently executable cells which are tremendously useful when it comes to training models for machine learning. I don't need to go into detail on this, but basically, since training the model sometimes takes some time, it is useful to have one cell dedicated to fitting the model, that way we don't have to refit the model every time we make a change to the code and want to execute a new line of code. Also, visualising data becomes very easy inside a jupyter notebook.
2. *Numpy* - Using numpy is self-explanatory for anyone that has ever used Python. This is python's main library for dealing with numbers, arrays, matrices, etc. Without this basic library, none of this project would be possible.
3. *Matplotlib* - Similar to numpy, matplotlib is the main Python library for plotting. In order to visualise the data and also be able to make a confusion matrix, we needed matplotlib's pyplot function.
4. *SKlearn* - SKlearn is one of the main Python libraries for machine learning and other data science functions. MATLAB has a built-in Logistic Regression function which the professor told us we can use, but obviously a basic installation of Python does not, so I had to use the SKlearn library for it's Logistic Regression function.
5. *Seaborn* - Seaborn is another data visualisation library for Python that is based on matplotlib. This library was very useful when it came to making a confusion matrix, as it greatly simplified the process.

Steps:

1. Load Data
2. Visualise example data
3. Fit Model
4. Make Predictions
5. Visualise score with Confusion Matrix

For simplicity, I copied all my cells from the jupyter notebook and put it in a regular python script file so that it is easier to turn in.

Problem 2:

ATANAS DELEVSKI
3/29/20

ECE 407
HW #5 Q2

Given: $\left(\begin{bmatrix}1\\3\end{bmatrix}, 1\right), \left(\begin{bmatrix}2\\3\end{bmatrix}, 1\right), \left(\begin{bmatrix}-1\\1\end{bmatrix}, -1\right), \left(\begin{bmatrix}-2\\0.5\end{bmatrix}, -1\right)$

Starting w/ $w = \begin{bmatrix}0\\0\end{bmatrix}$ and $x = \begin{bmatrix}1\\3\end{bmatrix}$, $y = 1$

$w_{new} = w + yx \Rightarrow w_{new} = \begin{bmatrix}1\\3\end{bmatrix}$

then, $x = \begin{bmatrix}2\\3\end{bmatrix}$, $y = 1$

↳ $y(w^T x) = 1 \cdot [1\ 3]\begin{bmatrix}2\\3\end{bmatrix} = 11 > 0$

↳ Do nothing since $11 > 0$ ✓

then, $x = \begin{bmatrix}-1\\1\end{bmatrix}$, $y = -1$

↳ $y(w^T x) = -1 \cdot [1\ 3]\begin{bmatrix}-1\\1\end{bmatrix} = -2 < 0$

↳ Do nothing since $-2 < 0$ ✓

then, $x = \begin{bmatrix}-2\\0.5\end{bmatrix}$, $y = -1$

↳ $y(w^T x) = -1\,[1\ 3]\begin{bmatrix}-2\\0.5\end{bmatrix} = 0.5 > 0$

↳ since 0.5 is > 0, and not less than 0,

we update $w \Rightarrow w + y(x)$

$\Rightarrow w_{new} = \begin{bmatrix}1\\3\end{bmatrix} + \begin{bmatrix}-2\\-0.5\end{bmatrix} = \begin{bmatrix}3\\2.5\end{bmatrix}$

SO, w/ $w = \begin{bmatrix}3\\2.5\end{bmatrix}$, we classify $\begin{bmatrix}1\\-1\end{bmatrix}$

$\Rightarrow y\begin{bmatrix}3 & 2.5\end{bmatrix}\begin{bmatrix}1\\-1\end{bmatrix} \Rightarrow 3 + (-2.5) = 0.5$

$\Rightarrow$ X belongs to positive 1 class