Paarth Lakhani
u0936913

Homework 6:

1. Logistic Regression

a. g(w) = log(1 + exp(-y_i * w^T * x_i))
Derivative of above function wrt the weight vector
$\frac{d}{dw}$ g(w) = $\frac{d}{dw}$ [log(1 + exp(-y_i * w^T * x_i))]

We have to use chain rule of derivation,
= $\frac{1}{1+\exp(-y_i * w^T * x_i)}$ * $\frac{d}{dw}$ [1 + exp(-y_i * w^T * x_i]
= $\frac{1}{1+\exp(-y_i * w^T * x_i)}$ * [exp(-y_i * w^T * x_i) $\frac{d}{dw}$ (-y_i * w^T * x_i)]
= $\frac{\exp(-y_i * w^T * x_i) * (-y_i * x_i)}{1+\exp(-y_i * w^T * x_i)}$

b. If entire dataset is composed of a single example, say (x_i, y_i), then the objective function [J(w)] would be -
J(w) = min (log(1 + exp(-y_i*w^T * x_i) + $\frac{1}{\sigma^2}$ $w^T w$)

c. 'Derive the gradient' is equivalent to taking the derivative
$\frac{d}{dw}$ J(w) = $\frac{d}{dw}$ [min (log(1 + exp(-y_i*w^T * x_i) + $\frac{1}{\sigma^2}$ $w^T w$)]
= $\frac{1}{1+\exp(-y_i * w^T * x_i)}$ $\frac{d}{dw}$ [exp(-y_i*w^T * x_i)] + $\frac{1}{\sigma^2}$ * $\frac{d}{dw}$($w^T w$)
J`(w) = min ($\frac{\exp(-y_i * w^T * x_i) * (-y_i * x_i)}{1+\exp(-y_i * w^T * x_i)}$ + $\frac{1}{\sigma^2}$ $*(2w)$)

d. Pseudo code for stochastic gradient descent using the previous part:
Given a training set S = {(x_i, y_i)}, x ϵ R^d, y ϵ {-1, 1}

```
i.   Initialize $w^0$ = 0 $\epsilon$ $R^d$
ii.  For epoch 1…T:
     1. Pick a random example ($x_i$, $y_i$) from the training set S
     2. Treat ($x_i$, $y_i$) as a full dataset and take the derivative of the SVM
        objective at the current $w^{t-1}$ to be $\nabla J$ ($w^{t-1}$)
     3. Update $w^t$ < ------- $w^{t-1}$ - $\gamma_t \nabla J$ ($w^{t-1}$)
        From the previous question, we know what $\nabla J$ ($w^{t-1}$) is:
            J`($w^{t-1}$) = min( $\frac{\exp(-y_i * w^T * x_i) * (-y_i * x_i)}{1+\exp(-y_i * w^T * x_i)}$ + $\frac{1}{\sigma^2}$ $*(2w)$)
iii. Return final w
```

2. Experiments

2.3

All the algorithms below, the following hold true:

a. Performed 5 cross validation to find out the best hypermeters amongst the ones given in the assignment unless otherwise mentioned.
b. Python programming language is used.
c. I have used the libSVM format data. I am creating numpy ndarrays and storing them as sparse vectors.

1. Support Vector Machines

i. I have referred the class slides to implement stochastic sub-gradient descent for SVM.

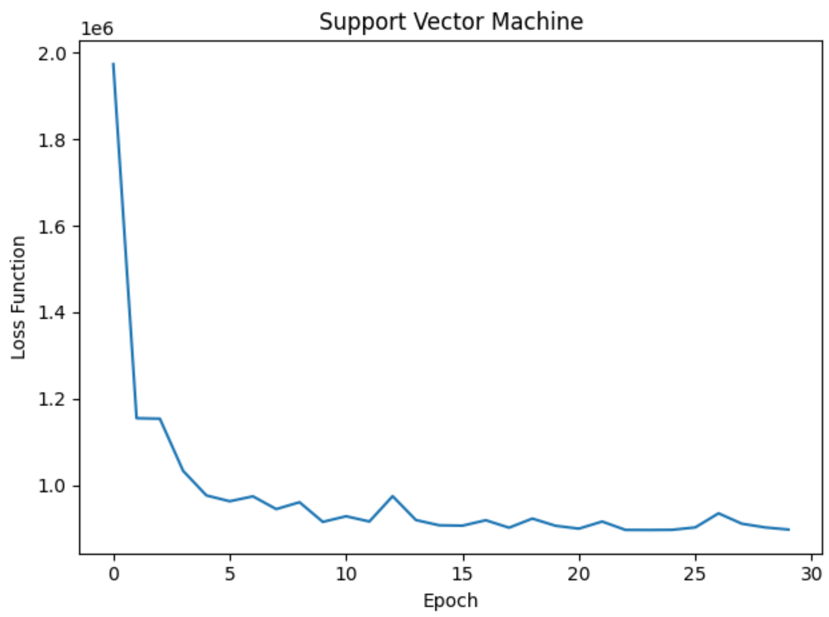
ii. The number of epochs is calculated runtime.

   a. To begin with, I am setting the number of epochs to be a high number.
   b. On every epoch run, I am calculating the loss function. If the absolute difference between current epoch loss function and previous epoch loss function is less than a certain value epsilon, then I would break from the loop. This is done because, at this point, we say that the loss function improvement is so small that there is no use of running for more number of epochs.
   c. The epsilon value is set to **3** which means if the difference between current loss function and previous loss function is less 3 then we assume loss function has been minimized to considerably and we can now end our algorithm to run.

iii. Results table

	Best-hyper parameters	Average cross-validation accuracy	Training accuracy	Test accuracy
SVM				

iv. Plot function of loss function



2. Logistic Regression

i. I have referred the pseudo code written in question 1.

ii. When performing cross validation, for the Tradeoff value of σ^2 = 0.1, it was calculating the value of $\frac{1}{\sigma^2} *(2w)$ as nan. On excluding that value, the algorithm ran as expected.

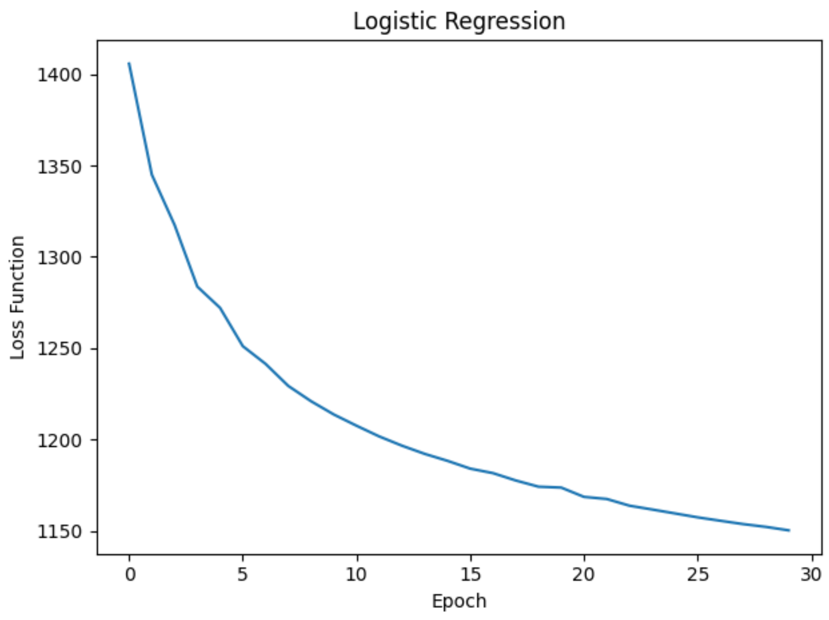
iii. The number of epochs is calculated runtime.

1. To begin with, I am setting the number of epochs to be a high number.
2. On every epoch run, I am calculating the loss function. If the absolute difference between current epoch loss function and previous epoch loss function is less than a certain value epsilon, then I would break from the loop. This is done because, at this point, we say that the loss function improvement is so small that there is no use of running for more number of epochs.
3. The epsilon value is set to **0.01** which means if the difference between current loss function and previous loss function is less 3 then we assume loss function has been minimized to considerably and we can now end our algorithm to run.

iv. Results table

	Best-hyper parameters	Average cross-validation accuracy	Training accuracy	Test accuracy
Logistic regression	Learning rate: 0.01 Loss Trade off: 1000	76.99551569506727	84.26008968609865	79.74910394265234

v. Plot of loss function



3. SVM over trees

i. Results table

	Best-hyper parameters	Average cross-validation accuracy	Training accuracy	Test accuracy
SVM over trees				