Homework Assignment # 3

Due: Wednesday, November 18, 2015, 11:59 p.m.
Total marks: 100

Name:Srijita Das
Email id:sridas@indiana.edu

Question 1. [60 MARKS]

In this question, you will implement several binary classifiers: naive Bayes, logistic regression and a neural network. An initial script in python has been given to you, called `script_classify.py`, and associated python files. You will be running on a physics dataset, with 9 features and 100,000 samples. The features are augmented to have a column of ones. Baseline algorithms, including random predictions and linear regression, are used to serve as sanity checks. We should be able to outperform random predictions, and linear regression for this binary classification dataset. To implement these algorithms, use the updates given in the notes. You can augment from these initial updates, depending on the requirements.

(a) [15 MARKS] Implement naive Bayes, assuming a Gaussian distribution on each of the features. Try including the columns of ones and not including the column of ones in the predictor. What happens? Explain why.

Ans:I implemented naive Bayes assuming a Gaussian distribution on each feature and jotted below are the results. It did almost as good as Linear Regression. The below results are jotted down for a training size of 67000 and Test size of 33000.

Method	Accuracy
Random	49.66
Linear Regression	75.02
Naive Bayes(With 1)	74.34
(Naive Bayes(Without 1)	74.34

The Naive Bayes performs as good as the Linear Regression and of course outperforms Linear regression.When I reduce the training size to 20000 and the test size to 10000, then the accuracy of Naive Bayes is 74.42. My code gives the same result when I try implementing Naive Bayes with or without the columns of ones at the end.

To cross validate the results I ran the Python ready made function for Naive Bayes $GaussianNB()$ from the package *sklearn.naive_bayes* and ran it on a training set of 67000 and a testset of 33000. It gave me an accuracy of 74.03 for both the cases(using ones and not using one's). This accuracy is the same as I got when I implemented Naive Bayes in my code.

From what I notice after implementing the Naive Bayes Classifier is that there is no difference in the accuracy when I implement the Naive Bayes with ones and without ones. Without doing any analysis, from intuition I can say that the bias column should not matter for Naive Bayes because here, we are not fitting any straight line and hence, the concept of intercept is not required. So, the presence or absence of a column of 1 should not matter. Mathematically, the mean of the column of 1 in the dataset for any class label is 1 and the variance is 0 since all the values are 1. In the Naive Bayes implementation, we have put the condition that if the standard deviation is less than 0.0001 and mean is less than 0.001, then the probability for that feature given a class label is returned as 1. If this approximation is not done, then due to the variance of 0, the probability in the Gaussian probability distribution will be ∞ and that would cause a problem. So, if a column of 1 is added to the dataset, the P(feature=1—y=classlabel) is always returned as 1 and since, in the final probability calculation of a data point, this value is multiplied with the probability of all other features given a particular class label, so, it does not change the probability at all. Hence, the accuracy comes out to be the same in the presence or absence of 1 in the dataset of Naive Bayes.

(b) [15 MARKS] Implement logistic regression.

Ans: I implemented Logistic Regression with no regularisation, L1 regularisor and L2 regularisor for the Physics dataset and I was able to outperform Linear Regression by some amount.Below are the results I ran on a training set of 20000 and testset of 10000.

Method	Accuracy
Random	49.81
Linear Regression	75.51
Logistic Regression	76.81
Logistic Regression(L1)($\lambda = 0.1$)	76.39
Logistic Regression(L2)($\lambda = 0.1$)	76.8

The function in Logistic Regression is a nice Convex function and while implementing the Logistic Regression, when I checked for the distance between the previous and the current weight vector, the distance between these two vectors decreases continuously as well as the function value decreases continuously. This makes me believe that this is a nice Convex function and we are descending down this objective function.

The general observation is that Logistic Regression performs a little better than Linear Regression though the difference is not that significant visually. For that we have to do some hypothesis testing which I did not perform for this assignment. Also, one thing to notice is that L1 and L2 regularisation did not have much impact on this dataset. That could be because the number of features is not that large as well as none of the weights in Logistic Regression(no regularisation) gets too large for L2 regularisation to cause a significant effect. Also I changed the hard margin from 0.5 to various values for clasification but the hard margin of 0.5 seems to work the best.
The weight vector for Logistic Regression looks like below

```
[[ 1.96666346]
 [ 0.01885084]
 [ 0.03072323]
 [-0.57153284]
 [-0.00314416]
 [ 0.00578743]
 [ 1.80066063]
 [-0.01742854]
 [-3.06297684]]
```

(c) [20 MARKS] Implement a neural network with a single hidden layer.

Ans: I implemented the Neural network on a trainsize of 20000 and a testset of 10000 with various number of hidden layers and jotted down below are the results:

Method	Accuracy
Random	50.17
Linear Regression	74.38
Neural N/W(64 hidden nodes)	76.5
Neural(N/W(40 hidden nodes)	77.18
Neural(N/W(120 hidden nodes)	78.02

Offcourse, the neural network outperforms the Random and the Linear Regression. When the Neural network is run multiple times on the same number of training and test data, it gives accuracy tat ranges from 76.5 to 79.1. Another observation is that when I increased the number of hidden nodes to 120, the neural network at times performed as good as an accuracy of 79 and it also performed as average as an accuracy of 72.28.I implemented Stochastic Gradient Descent on Neural network and not the batch Gradient descent.

(d) [10 MARKS] You will do a more complete empirical analysis in another assignment. However, briefly describe the behavior of these classification algorithms you have implemented. You do not need to make claims about statistically significant behavior, but report properties you notice, when you run the script multiple times.

Ans: Jotted below is a table containing accuracy of 5 runs of the program for all the algorithms I implemented for this assignment. The below results are run on a training set of 10000 and test set of 5000.

Method	**Run1**	**Run2**	**Run3**	**Run4**	**Run5**
Random	50.02	49.06	49.2	49.58	50.16
Linear Regression	75.0	75.1	74.96	74.98	75.94
Logistic Regression	76.6	76.44	75.82	77.14	77
Naive Bayes	74.06	74.76	73.8	73.74	74.4
Neural Network	77.48	76.62	75.22	74.66	78.1

Below are my observations from multiple runs of the code.

1) The random Classifier as expected predicts around 50% all the time. That is intuitive because the physics dataset is kind of balanced with equal number of positive and negative instances. So, even if the classifier predicts one class label all the time, it would be correct 50% of the time.

2) The Linear Classifier does a good job in acting as a binary classifier and almost all of the time predicts with an accuracy of very close to 75%.

3) The Logistic Regression at least visually does a little better than Linear regression and predicts from a range of 75% to 77% at times. Also, regularisation does not have any significant

effect on the accuracy of Logistic Regression on the Physics dataset.

4) The Naive Bayes also does consistently in all the five runs. It does prediction with an accuracy of around 74.5%.

5) The Neural network performs almost every time better than the Linear Regression. Without any significance testing, it cannot be told if it also performs better than Logistic Regression on an average. It's accuracy ranges from 75% to 78%.

Question 2. [20 MARKS]

Consider a classification problem where $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \{0, 1\}$. Based on your understanding of the maximum likelihood estimation of weights in logistic regression, develop a linear classifier that models the posterior probability of the positive class as

$$P(y = 1|\mathbf{x}, \mathbf{w}) = \frac{1}{2}\left(1 + \frac{\mathbf{w}^\top \mathbf{x}}{\sqrt{1 + (\mathbf{w}^\top \mathbf{x})^2}}\right)$$

Implement the iterative weight update rule and compare the performance on the physics dataset to logistic regression. As before, you do not need to check for statistically significant behavior, but in a few sentences describe what you notice.

Ans: I implemented both stochastic and Batch gradient descent to implement a Linear Classifier.

$$P(y = 1|\mathbf{x}, \mathbf{w}) = \frac{1}{2}\left(1 + \frac{\mathbf{w}^\top \mathbf{x}}{\sqrt{1 + (\mathbf{w}^\top \mathbf{x})^2}}\right)$$
$$P(y = 0|\mathbf{x}, \mathbf{w}) = 1 - P(y = 1|\mathbf{x}, \mathbf{w}) = \frac{1}{2}\left(1 - \frac{\mathbf{w}^\top \mathbf{x}}{\sqrt{1 + (\mathbf{w}^\top \mathbf{x})^2}}\right)$$

From these two equations, we can model the target as a Bernauli distribution as below:

$$p(y|x) = \{ \quad \begin{matrix} \frac{1}{2}\left(1 + \frac{\mathbf{w}^\top \mathbf{x}}{\sqrt{1+(\mathbf{w}^\top \mathbf{x})^2}}\right) & y = 1 \\ \frac{1}{2}\left(1 - \frac{\mathbf{w}^\top \mathbf{x}}{\sqrt{1+(\mathbf{w}^\top \mathbf{x})^2}}\right) & y = 0 \end{matrix}$$

w here is a vector of weight components for all the d dimensions of each input point.

$$p(y|x) = \frac{1}{2}\left(1 + \frac{\mathbf{w}^\top \mathbf{x}}{\sqrt{1 + (\mathbf{w}^\top \mathbf{x})^2}}\right)^y * \frac{1}{2}\left(1 - \frac{\mathbf{w}^\top \mathbf{x}}{\sqrt{1 + (\mathbf{w}^\top \mathbf{x})^2}}\right)^{1-y}$$

For the simplicity of the calculation, let us consider $\mathbf{w}^\top \mathbf{x} = t$
The log likelihood function derivation then becomes as follows:

$$l(w) = \prod_{i=1}^{n} p(y_i|x_i, w)$$

$$\begin{aligned}
w_{ML} &= arg\ max\ w\{l(w)\} \\
&= arg\ max\ w\{\prod_{i=1}^{n} p(y_i|x_i, w)\} \\
&= arg\ max\ w\{\frac{1}{2}\left(1+\frac{t}{\sqrt{1+t^2}}\right)^{y} * \frac{1}{2}\left(1-\frac{t}{\sqrt{1+t^2}}\right)^{1-y}\} \\
ll(w) &= \log(\frac{1}{2}\left(1+\frac{t}{\sqrt{1+t^2}}\right)^{y} * \frac{1}{2}\left(1-\frac{t}{\sqrt{1+t^2}}\right)^{1-y}) \\
&= \sum_{i=1}^{n} y.\log[\frac{1}{2}\left(1+\frac{t}{\sqrt{1+t^2}}\right)] + (1-y)\log[\frac{1}{2}\left(1-\frac{t}{\sqrt{1+t^2}}\right)] \\
&= \sum_{i=1}^{n} y\log\frac{1}{2} + y\log\left(1+\frac{t}{\sqrt{1+t^2}}\right) + (1-y)\log\frac{1}{2} + (1-y)\log\left(1-\frac{t}{\sqrt{1+t^2}}\right)
\end{aligned}$$

Going forward, we can neglect the $y)\log[\frac{1}{2}$ and $(1-y)\log\frac{1}{2}$ term because the derivative of both of these terms will be 0.
Continuing further with the log likelihood equation, we get the below:

$$\begin{aligned}
ll(w) &= \sum_{i=1}^{n} y\log(\frac{\sqrt{1+t^2}+t}{\sqrt{1+t^2}}) + \log(\frac{\sqrt{1+t^2}-t}{\sqrt{1+t^2}}) - y\log(\frac{\sqrt{1+t^2}-t}{\sqrt{1+t^2}}) \\
&= \sum_{i=1}^{n} y\log(\frac{\sqrt{1+t^2}+t}{\sqrt{1+t^2}-t}) + \log(\frac{\sqrt{1+t^2}-t}{\sqrt{1+t^2}}) \\
ll(w) &= \sum_{i=1}^{n} y\log(\frac{1+\frac{t}{\sqrt{1+t^2}}}{1-\frac{t}{\sqrt{1+t^2}}}) + \log(1-\frac{t}{\sqrt{1+t^2}}) \\
&= \sum_{i=1}^{n} y\log(\frac{p(y=1|x,w)}{p(y=0|x,w)}) + \log(p(y=0|x,w)
\end{aligned}$$

Now, we have to calculate the derivative of this likelihood function w.r.t to a single weight w_j and then generalise the gradient expression for all weights contained in the weight vector. From all calculations of the derivative, the expression finally reduces to

$$\frac{\partial}{\partial w_j} ll(w) = \sum_{i=1}^{n} x_{ij}(\frac{p(y_i=1|x,w).p(y_i=0)|x,w).(2y_i-1}{\sqrt{1+(w^Tx_i)^2}})$$

So, now, we can create a vector of probabilities containing $\frac{p(y_i=1|x,w).p(y_i=0|x,w).(2y_i-1}{\sqrt{1+(w^Tx_i)^2}}$ for every data point. Converting this to matrix form, we get the final gradient as:
$\frac{\partial}{\partial w_j} ll(w) = X^T.P_i$ where P_i whose each element is calculated considering the individual data points.
So, the gradient descnet rule now becomes:
$w^{(t+1)} = w^{(t)} + X^T.P^{(t)}$
We add the gradient because we are maximizing the log likelihood function and not minimizing the loss function. We continue with this Gradient descent algorithm until the weights converge.

Jotted below is the result of Linear Classifier as well as the Logistic regression on a training set of 10000 and testset of 5000.

Method	Accuracy
Logistic Regression	78.1
My Classifier	50.18

My Classifier at times does a little better than Random and gives an accuracy of 52%. Another thing I noticed is the step size.The function converges at a step size of 0.00001. If I give step size more than that, the log likelihood function does not converge. This gives me the intuition that the function is a non-convex function with a local maxima right at the starting. So, a small step size is required to converge the function . Also, the reason why it does not give a good solution may be because that it gives the local maxima and there may be some other global maxima of the function somewhere else which it is not being able to reach.
Also, while predicting, I find out the $p(y = 1|x, w)$ and $p(y = 0|x, w)$ for each data point and whichever probability is more, I assign the data points to that particular class label.

Question 3. [20 MARKS]

In this question, you will add regularization to logistic regression, and check the behavior again on a new dataset with 500 features (`https://archive.ics.uci.edu/ml/datasets/Madelon`). Note that using regularization on the physics dataset, which only has 9 features, would not have as strong of an effect; with more features, the regularization choice is likely to have more impact. For all the three settings below, implement the iterative update rule and in a few sentences briefly describe the behavior, versus vanilla logistic regression and versus the other regularizers.

(a) [5 MARKS] Explain how you would add an ℓ_2 regularizar on $\mathbf{w}$ and an ℓ_1 regularisor on $\mathbf{w}$. Implement both of these.
Ans: Before processing the dataset, I normalized the madelon dataset to a suitable scale using R and then ran Logistic Regression,L1 and L2 regularisor on it.
To implement L1 regularisor, I basically took the sign vector of the weights and multiplied the parameter λ to it. The derivative of an absolute function is a piece wise function having different values for greater than 0 values and less than zero values. Hence, a sign vector of weight would account for whether the parameter λ needs to be added to the weighth vector or subtracted from the weight vector.
The update equation for Logistic Regression with L1 regularisor then becomes:
$w^{(t+1)} = w^{(t)} + (X^T P^{(t)}(I - P(t))X)^{-1}(X^T(y - p^{(t)}) - \lambda sign(w^{(t)})$

For implementing L2 regularisor, I took the derivative of $\frac{1}{2}\lambda||w||_2^2$. The final weight update rule for L2 regularisor is as below:
$w^{(t+1)} = w^{(t)} + (X^T P^{(t)}(I - P(t))X + \lambda I)^{-1}(X^T(y - p^{(t)}) - \lambda w^{(t)})$

I implemented the below regularisor on the madelon dataset and as such it did not make a huge difference. For both L1 as well as L2 regularisor, a λ value of 0.1 gives the best result for my code. The results are as jotted down below:

Method	Accuracy
LR	58.83
LR(L2)	59.16
LR(L1)	58.83

For L1- regularisor the function does not converge if I set the threshold to be very low like 0.0001. However, it converges if I set the threshold to 0.01. For the Madelon dataset also, I did not notice a significant change in the accuracy after adding L1 as well as L2 regularisor to it. One of the reasons might be that the number of features is too large whereas the number of data points is only 2000. So, the model might not be well trained with less number of data points for such a large number of features. I tried implementing L1 regularisor with a couple of other parameters but in all cases, it was giving the accuracy as 58.83.I even tried to lower bound the weights and set them to 0 when a particular weight in the weight vector becomes less than 0.0001. However, in this case, my accuracy for L1 regularization dropped to 54%.
For sanity Check I even tried running the inbuilt logistic Regression package in Python with no regularisor and L2 regularisor. In the first case, the accuracy was 58.83% and in the second case (L2), the accuracy for my code was 58.16%.

(b) [10 MARKS] Pick a third regularizer of your choosing, and explain how you would learn with this regularizer in logistic regression (i.e., provide an iterative update rule and/or an algorithm). Implement this regularization.
Ans: I tried implementing a third regularisor of my choice. My choice of regularisor was a mixture of L1 and L2 regularisor.The λ in my case is the probability that I would assign to each of these regularisation.This would lead to a weight update and then the model with predict with these weights. The weight update equation for my regularisor now becomes as below:
$w^{(t+1)} = w^{(t)} + (X^T P^{(t)}(I - P(t))X + (1-\alpha)I)^{-1}(X^T(y - p^{(t)}) - \alpha sign(w^{(t)})(1-\alpha)w^{(t)})$
α in the above expression is the weight of L1 regularisor and (1-α) is the weight of the L2 regularisor. The results for various values of α is as jotted below:

α	Accuracy
0.1	58.5
0.2	58.66
0.3	58.66
0.5	58.0
0.7	58.16
0.9	57.33

From the above values, we see that the choice of α to be 0.2 or 0.3 is a reasonable choice. Another observation seen by me while implementing regularisation is that the distance between the weight vectors increases as I increase the probability of L1.

(c) [5 MARKS] In a few sentences, briefly describe the behavior of the regularizers.
Ans: The behavior of the regularisors was not very prominently visible in either of the dataset. Basically, none of the algorithms worked very well on the madelon dataset. The reason might be because of 500 features in the dataset and only 2000 data points available to learn the model. L1 regularisor is used in a sense for feature reduction because it adjusts the weights and sets the weights of the not so important features as 0. I was expecting L1 regularisor to perform better than L2 regularisor for madelon dataset but L1 regularisor did not have that much of an effect on the madelon dataset. L2 regularisor on the other hand did a little better job and had an accuracy of 59.16.L2 Regularisation normally puts penalties on the large weights and brings down those weights so that the Regression model fits better. However, when I apply regularisation on Physics dataset, then also L1 and L2 regularisation did not have any impact. For the Physics dataset, L1 did not have any impact because there were less number of features and hence, the effect is hard to see.

However, L2 regularisation could have an effect but when I see the final weights after applying L2 regularisation on this dataset, I see that the weights are more or less not absurdly large and hence L2 regularisation did not have an effect on Physics dataset.

Homework policies:

Your assignment will be submitted as a single pdf document and a zip file with code, on canvas. The questions must be typed; for example, in Latex, Microsoft Word, Lyx, etc. or must be written legibly and scanned. Images may be scanned and inserted into the document if it is too complicated to draw them properly. All code (if applicable) should be turned in when you submit your assignment. Use Matlab, Python, R, Java or C. Policy for late submission assignments: Unless there are legitimate circumstances, late assignments will be accepted up to 5 days after the due date and graded using the following rule:

on time: your score 1

1 day late: your score 0.9

2 days late: your score 0.7

3 days late: your score 0.5

4 days late: your score 0.3

5 days late: your score 0.1

For example, this means that if you submit 3 days late and get 80 points for your answers, your total number of points will be $80 \times 0.5 = 40$ points. All assignments are individual, except when collaboration is explicitly allowed. All the sources used for problem solution must be acknowledged, e.g. web sites, books, research papers, personal communication with people, etc. Academic honesty is taken seriously; for detailed information see Indiana University Code of Student Rights, Responsibilities, and Conduct.

Good luck!