Homework Assignment # 3

Submitted by Prerit Anwekar

Total marks: 100

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.

(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.

Answer(a). On including the columns of ones we get NaN as a weights for the columns of ones. After carefully investigating the MLE solution for the Gaussian Naive Bayes it was observed that the standard deviation was zero. And standard deviation being zero we don't get a weight for the ones columns. [For Implementation please check Solution folder.]

(b) [15 MARKS] Implement logistic regression.

Answer(b). [For Implementation please check Solution folder.]

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

Answer(c). [For Implementation please check Solution folder.]

(d) [10 MARKS] Briefly describe the behavior of these classification algorithms you have implemented. You do not need to make claims about statistically significant behavior, but report average error and standard error. You do not need to run on the whole dataset.

Answer(d).

Gaussian Naive Bayes Algorithm:

- A bias column isn't much insightful and it's weight are not defined because of no variance.
- The algorithm is computationally fast and is fairly accurate.
- As this algorithm is largely dependent on counts of class, different sampling from the population can give different results.

Logistic Regression

- By using the sigmoid function, the regression problem is converted to a classification problem.
- The algorithm is fairly simple and can be extended to multi-class classification problem.
- The choice of step size is very crucial in logistic regression or the algorithm might not converge properly.

Neural Networks with single hidden layer

- It's computationally expensive because of we need to compute matrices of weights for each layer.(Highest in fully connected network)
- Choice of number of hidden layer is very crucial.
- Choice of Initial weights is VERY important. A random uniform weights gave better accuracy.

Algorithm	Average Error +/- Standard deviation
Gaussian Naive Bayes	24.2 +/- 3.5527136788e-15
Logistic Regression	25.29 +/- 0.113578166916
Neural Network with single hidden layer	22.62 +/- 0.46432747065

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.

Answer(2). In oder to solve this problem we need to formulate a Maximum likelihood problem and then minimize the cost function.

Since this classifier is a binary classifier we can immediately think of cost as products of Bernoulli's trials. Therefore,

$$\text{Cost(w)} = \prod_{i=1}^{n}\left\{\left(\frac{1}{2}\left(1 + \frac{\mathbf{w}^\top \mathbf{x_i}}{\sqrt{1 + (\mathbf{w}^\top \mathbf{x_i})^2}}\right)\right)^{y_i}\left(1 - \frac{1}{2}\left(1 + \frac{\mathbf{w}^\top \mathbf{x_i}}{\sqrt{1 + (\mathbf{w}^\top \mathbf{x_i})^2}}\right)\right)^{1-y_i}\right\}$$

$$\begin{aligned}
log\ \text{Cost(w)} &= \sum_{i=1}^{n}\left\{\left(log\frac{1}{2} + y_i log\left(1 + \frac{\mathbf{w}^\top \mathbf{x_i}}{\sqrt{1 + (\mathbf{w}^\top \mathbf{x_i})^2}}\right) + (1-y_i)log\left(1 - \frac{1}{2}\left(1 + \frac{\mathbf{w}^\top \mathbf{x_i}}{\sqrt{1 + (\mathbf{w}^\top \mathbf{x_i})^2}}\right)\right)\right\}\\
&= \sum_{i=1}^{n}\left\{\left(log\frac{1}{2} + y_i log\left(1 + \frac{\mathbf{w}^\top \mathbf{x_i}}{\sqrt{1 + (\mathbf{w}^\top \mathbf{x_i})^2}}\right) + (1-y_i)log\left(1 - \frac{1}{2}\left(1 + \frac{\mathbf{w}^\top \mathbf{x_i}}{\sqrt{1 + (\mathbf{w}^\top \mathbf{x_i})^2}}\right)\right)\right)\right\}\\
&= \sum_{i=1}^{n}\left\{\left(log\frac{1}{2} + y_i log(\left(1 + \frac{\mathbf{w}^\top \mathbf{x_i}}{\sqrt{1 + (\mathbf{w}^\top \mathbf{x_i})^2}}\right) + (1-y_i)log\frac{1}{2}\left(2 - \left(1 + \frac{\mathbf{w}^\top \mathbf{x_i}}{\sqrt{1 + (\mathbf{w}^\top \mathbf{x_i})^2}}\right)\right)\right)\right\}\\
&= \sum_{i=1}^{n}\left\{\left(log\frac{1}{2} + y_i log\left(1 + \frac{\mathbf{w}^\top \mathbf{x_i}}{\sqrt{1 + (\mathbf{w}^\top \mathbf{x_i})^2}}\right) + (1-y_i)\left(log\frac{1}{2} + log\left(1 - \frac{\mathbf{w}^\top \mathbf{x_i}}{\sqrt{1 + (\mathbf{w}^\top \mathbf{x_i})^2}}\right)\right)\right)\right\}
\end{aligned}$$

Now, differentiating with respect to w_j,

$$\frac{\partial ll}{\partial w_j} = \sum_{i=1}^{n}\left\{\frac{y_i}{\left(1 + \frac{\mathbf{w}^\top \mathbf{x_i}}{\sqrt{1+(\mathbf{w}^\top \mathbf{x_i})^2}}\right)} - \frac{(1-y_i)}{\left(1 - \frac{\mathbf{w}^\top \mathbf{x_i}}{\sqrt{1+(\mathbf{w}^\top \mathbf{x_i})^2}}\right)}\right\}\frac{\partial}{\partial w_j}\left(\frac{\mathbf{w}^\top \mathbf{x_i}}{\sqrt{1 + (\mathbf{w}^\top \mathbf{x_i})^2}}\right)$$

Now finding the derivative of the second term in the equation above,

$$\frac{\partial}{\partial w_j}\left(\frac{\mathbf{w}^\top \mathbf{x_i}}{\sqrt{1+(\mathbf{w}^\top \mathbf{x_i})^2}}\right) = \frac{\sqrt{1+(\mathbf{w}^\top \mathbf{x_i})^2}x_{ij} - \frac{\mathbf{w}^\top \mathbf{x_i}}{2\sqrt{1+(\mathbf{w}^\top \mathbf{x_i})^2}}2\mathbf{w}^\top \mathbf{x_i}.x_{ij}}{(1+(\mathbf{w}^\top \mathbf{x_i})^2)}$$
$$= \frac{1+(\mathbf{w}^\top \mathbf{x_i})^2 - (\mathbf{w}^\top \mathbf{x_i})^2}{(1+(\mathbf{w}^\top \mathbf{x_i})^2)^{\frac{3}{2}}}x_{ij}$$
$$= \frac{x_{ij}}{(1+(\mathbf{w}^\top \mathbf{x_i})^2)^{\frac{3}{2}}}$$

Now, substituting this value in our likelihood equation we get the complete first order derivative for our weights,

$$\frac{\partial ll}{\partial w_j} = \sum_{i=1}^{n}\left\{\frac{y_i}{\left(1+\frac{\mathbf{w}^\top \mathbf{x_i}}{\sqrt{1+(\mathbf{w}^\top \mathbf{x_i})^2}}\right)} - \frac{(1-y_i)}{\left(1-\frac{\mathbf{w}^\top \mathbf{x_i}}{\sqrt{1+(\mathbf{w}^\top \mathbf{x_i})^2}}\right)}\right\}\frac{x_{ij}}{(1+(\mathbf{w}^\top \mathbf{x_i})^2)^{\frac{3}{2}}}$$

This function can be further be simplified, but in my opinion the above form is self annotating because any one of the term from two in the brackets will be used since $y_i \in \{0,1\}$

Finally we will use the following weight update rule to evaluate the weights on each iteration,

For j = 1,2,3...d, where d represents the total number of features in the dataset. Repeat this step while not converged.

$$w_j = w_j - \alpha\frac{\partial ll}{\partial w_j}$$

Since both Vanilla Logistic Regression and the above given pdf are sigmoid. Theoretically they should have same answers but during my implementation I found out that while using the above pdf its really necessary to tune the learning rate in order for our algorithm to converge to a better solution.

Question 3. [20 MARKS]

In this question, you will add regularization to logistic regression, and check the behavior again on the expanded physics dataset, which has 18 features (called `susy_complete` in the code). Note that using regularization on the base 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 regularizer on $\mathbf{w}$ and an ℓ_1 regularizer on $\mathbf{w}$. Implement both of these.

Answer(a). To add an l_2 regularizer we just need to solve to minimize the following objective function:

$$Cost_{\text{regularizedLogistic}} = Cost_{\text{logistic}} + \lambda\|w\|_2^2$$

To add an l_1 regularizer we just need to solve to minimize the following objective function:

$$Cost_{\text{regularizedLogistic}} = Cost_{\text{logistic}} + \lambda\|w\|_1$$

Now to minimize this objective function we can derive an iterative weight update method by differentiating the cost with respect to each weights. On applying regularization we would get a bit better generalization accuracy.

[For Implementation please check Solution folder.]

(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.

Answer(b). The third regularizer that I chose is Elastic net. This regularizer is a combination of l_1 and l_2 penalty.

$Cost_{regularizedLogistic} = Cost_{logistic} + \lambda\|w\|_2^2 + \lambda\|w\|_1$

As usual, we need to minimize this cost function in order to reach a minima. A naive observer would try to apply an l_2 regularization first and then l_1 regularization. But that method is invalid. We need to solve this equation considering both the regularization terms. Therefore, I have implemented the elastic net regularization problem using two approaches.

First one just finding the gradient of logistic cost + derivative of l2 regularizer + derivative of l1 regularizer using the functions provided by Prof. White and then applying the weight update rule.

Second approach is to solve the elastic net problem I went through following paper "[Hui Zou, Trevor Hastie] Regularization and Variable Selection via the Elastic Net", in which the elastic net weight update algorithm was as follows:

$\hat{w}_i(NaiveElasticNet) = \frac{(|\hat{w}_i(Logistic)| - \frac{\lambda_1}{2})_+}{1+\lambda_2} sgn(\hat{w}_i(Logistic))$

Where $(|\hat{w}_i(Logistic)|)$ is the logistic regression weights found using gradient descent.The x_+ denotes the positve part, which is x if $x > 0$ else 0. The solution of ridge is given by $\frac{(|\hat{w}_i(Logistic)|)}{1+\lambda_2}$ and for lasso is given by :

$(|\hat{w}_i(Logistic)| - \frac{\lambda_1}{2})_+ sgn(\hat{w}_i(Logistic))$

(c) [5 MARKS] In a few sentences, briefly describe the behavior of the regularizers.

Answer(c). On application of L1 and L2 regularization it was observed that the L1 and L2 converge to same minima and both the regularizers give almost same accuracy. The weights learned in case of L1 are smaller than the weights learned by L2. The elastic net regularizer takes into the account of both (l1 and l2) of these regularizations and we get the weight shrinkage by l2 and feature selection ability of lasso. The difference with the vanilla Logistic regression is that all the three regularizers try to shrink the weights and try to remove any correlated features (l1) , This usually helps to improve the test accuracy and avoids overfitting of the model by applying penalties.