Homework Assignment # 3

Due: Wednesday, November 16, 2016, 11:59 p.m.
Total marks: 100

Submitted By Vinay Vernekar (vrvernek@iu.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.

(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:
The Gaussian naive Bayers is implemented

Observations The column of one is a bias unit with mean 1 and variance zero. This zero variance will cause a problem when we try to calculate the probability of a value from that column (bias column)using the Gaussian formula

$$f(x_j|y=c) = (2\pi\sigma_{j,c})^{\frac{-1}{2}} * e^{-(\frac{(x-\mu_{j,c})^2}{2\sigma_{j,c}^2})}$$

where $\mu_{j,c}$ is the mean for j feature with class "c" and $\sigma_{j,c}^2$ is the variance for j feature with class "c". In my implementation it gives a division by zero error. In the code provided by the Professor the utils function *"calculateprob"* takes care of this issue using check for variance/standard deviation and math.fabs(x-mean) and returns 1 when the result is below a particular threshold. We I commented out this check I got a division by zero error.

When I keep the column of ones, the probability of that column for each value corresponding to its entry is 1 as per the utils formula.Hence it is ideally taking the value or probabilities of all other 8 columns and multiplying it by 1(bias column).

If I keep the Professors code intact, then there is no difference between the answer given by the code with or without the column of ones.

(b) [15 MARKS] Implement logistic regression.

Answer : Implemented logistic regression. The average error of the 23.9132 and a std of +/- 0.149933743242

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

Answer : Implemented Neural Network. The average error of the 26.0786 and a std error of +/-0.437850227142

(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,

The setting used for the above output is 1000-Training samples, 5000 test samples and 100 Runs
Answer:

a) Naive Bayers is a generative model and assumes that all the features are independent. It consists of the priors for each class or labels and I have implemented a linear classifier. It calculates the probability of each feature given the class. Each features probability is calculated based on the mean and the standard deviation of that feature and class. The individual probabilities for each feature for each class is multiplied and finally multiplied with respective prio, then the class has the highest probability(for test data) for that input gets assigned to the data point as its label. The probabilities at the start are big, but subsequent multiplication with other features probability make them very small

b) logistic regression is a discriminative model. It is a part of Generalized model and maps the linear output from the regression to a domain between 0,1 i.e converts them into probabilities using the link function Sigmoid. We use likelihood function and assume that the distribution is Bernoulli. This algorithm works on an iterative approach adjusting the weights which helps improve the accuracy of the classifier. The best line that separates the data points has high likelihood. The algorithm makes use of likelihood derivative like : $\triangle ll(w) = X^T(y-p)$, where y-p gives the error and this error multiplied by the feature. This is then used to update the weight until we fulfill a stopping criteria In the given I have kept it as convergence and count which ever is filled first.

c) Neural Network- Neural networks have layers. Layers are made up of a number of interconnected 'nodes' which contain an 'activation function'. Inputs are presented to the network via the 'input layer', which communicates to one or more 'hidden layers' where the actual processing is done via a system of weighted 'connections'. The hidden layers then link to an 'output layer' where we can see the output.

Each input node is connected with the hidden layer with a weight, these are summarized at each node and an activation function is applied to each entry and return a result of same dimension (X*W). The hidden node and the output layer are also connected with weights. Based on the output of the neural networks compared with the actual label/output the error is calculated, the weights connecting each nodes in the layers are adjusted based on the value of this error(delta) This function is handled by feedback function in my implementation. In the given implementation this will go on for a given number of epochs.

The best result is given when the hidden number of nodes is 32.

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

Table 1: Classifier Avg-error and Std-error

Classifier	Avg-Error	Std-Error +/-
Naive Bayes	25.7784	0.146441966304
Logistic Regression	23.9132	0.149933743242
Neural Network	26.0786	0.437850227142

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: Considering a Bernoulli distribution and taking log we can simplify the given equation and then implement it.

P(y=1—x$_i, w) = 1/2(1+\frac{w^Tx_i}{\sqrt{(1+(w^Tx_i)^2}})$

$P(y=0|x_i,w) = 1-1/2(1+\frac{w^Tx_i}{\sqrt{(1+(w^Tx_i)^2}})$

$l(w) = \prod_{i=1}^n P(y_i=1|x_i,w)^{y_i} * P(y_i=0|x_i,w)^{1-y_i}$

taking log *we get*,

$= \sum_{i=1}^n y_i log(1/2) + y_i log(1+\frac{w^Tx_i}{\sqrt{(1+(w^Tx_i)^2}}) + (1-y_i)log(1/2) + (1-y_i) + log(1-\frac{w^Tx_i}{\sqrt{(1+(w^Tx_i)^2}})$

$= nlog(1/2) + \sum_{i=1}^n y_i log(1+\frac{w^Tx_i}{\sqrt{(1+(w^Tx_i)^2}}) + (1-y_i)log(1-\frac{w^Tx_i}{\sqrt{(1+(w^Tx_i)^2}})$

now taking *derivatives*

$= \sum_{i=1}^n \frac{(2y_i-1)x_{ij}\sqrt{(W^Tx_i)^2+1}-x_{ij}W^Tx_i}{(W^Tx_i)^2+1}$

$= \sum_{i=1}^n \frac{x_{ij}[(2y_i-1)\sqrt{(W^Tx_i)^2+1}-W^Tx_i]}{(W^Tx_i)^2+1}$

This equation is used to calculate the weight in the program.

The above equation is too complicated need to derive hessian taking second derivative we get

d$^2 ll(w)/dw_j dw_k$

$= \sum_{i=1}^n \frac{\frac{d}{dw_k}(x_{ij}[(2y_i-1)\sqrt{(W^Tx_i)^2+1}-w^Tx_i])[(W^Tx_i)^2+1]}{((W^Tx_i)^2+1)^2} - \frac{(x_{ij}[(2y_i-1)\sqrt{(W^Tx_i)^2+1}-W^Tx_i])\frac{d}{dw_k}[(W^Tx_i)^2+1]}{((W^Tx_i)^2+1)^2}$

Now by taking partial derivate on the numerator, for fixed i

=x$_{ij}x_{ik}[\frac{(2y_i-1)W^Tx_i-\sqrt{(W^Tx_i)^2+1}}{\sqrt{(W^Tx_i)^2+1}}]$

first part of the fraction can be solved as

=x$_{ij}x_{ik}[\frac{(2y_i-1)W^Tx_i-\sqrt{(W^Tx_i)^2+1}}{((W^Tx_i)^2+1)^{3/2}}]$

The second part is solved as

=x$_{ij}x_{ik}2W^Tx_i[\frac{(2y_i-1)\sqrt{(W^Tx_i)^2+1}-W^Tx_i}{((W^Tx_i)^2+1)^2}]$

Now combining both we get

$d^2ll(w)/dw_jdw_k =$

$$\sum_{i=1}^n x_{ij}x_{ik}[\frac{(2y_i-1)W^Tx_i-\sqrt{(W^Tx_i)^2+1}}{((W^Tx_i)^2+1)^{3/2}} - \frac{2W^Tx_i[(2y_i-1\sqrt{(W^Tx_i)^2+1})-W^Tx_i]}{((W^Tx_i)^2)+1)^2}]$$

$$= \sum_{i=1}^n x_{ij}x_{ik}[\frac{(2y_i-1)W^Tx_i-\sqrt{(W^Tx_i)^2+1}}{((W^Tx_i)^2+1)^{3/2}} - \frac{2W^Tx_i[(2y_i-1\sqrt{(W^Tx_i)^2+1})-W^Tx_i]}{((W^Tx_i)^2)+1)^2}]$$

Putting all together we get the final update rule as,

$W^{t+1} = W^0 - H^{-1} \cdot \triangledown$, where H is hessian and $\triangledown$ is the gradient

Using the online derivative calculator it can be noted that this equation somewhat gives a constant probability no matter what the input is, hence the classifier gives an error of close to 50It can be noted that the preformance of the algorithm depends upon the data.The algorithm is implemented as Question2 in classalgorithms.py.

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

In logistic regression we get over fitting.To reduce this we can use regularization. In logistic regression we deal with log-likelihood, where we define The likelihood of a set of parameter values, θ, given outcomes x, is equal to the probability of those observed outcomes given those parameter values.

We take the derivative of the log- likelihood which gives us the gradient of log likelihood as ∇ll(W) =X$^T(y-p)$ where y is the actual label and p is estimated posterior probabilities

we then deduct the derivative of the product of the lambda or tuning paramater and the appropriate norm of the weights from the gradient of Log Likelihood

In general we have the gradient of Log-likeihood as $X^T(y-p)$
we have the penalty as lambda * |norm of the weights (W)| lets call this regularization term

putting it all together we total quality as

$X^T(y-p)$ - *derivative* (regularization term), This is a part of the update rule for the weights which helps maximize Likelihood.

For L1 regularization we have part of the update rule as $X^T(y-p)$ - $\lambda * sign(W)$. Where sign basically thershold the weights based on their value e.g if w_i is less than zero it becomes-1, w_i is greater than zero it becomes +1, if w_i is equal to zero it becomes zero.

For L2 regularization we have part of the update rule as $X^T(y-p)$ - $2 * \lambda * (W)$. In most of the literature λ is taken as $\lambda/2$ and hence after derivation the 2 cancels out of the L2 norm (I have used the same implementation assumption) .

Given the above part updates the final update to the present weights is made as follows

$W^{t+1} \leftarrow W^t$+*Stepsize* * *total quality*, where Total quality for L1 is equal to $X^T(y-p)$ - $\lambda * sign(W)$.

The Total quality for L2 is equal to $X^T(y-p)$ - $\lambda * (W)$

And The Total quality for Elastic Net is equal to $X^T(y-p)$ - $\lambda * (W)$ - $\lambda * sign(W)$

(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: The third regularizer used is Elastic Net, Which is a method of linear combination of L1 and L2. The elastic net method overcomes the limitations of the Lasso.

For example, in the "large p, small n" case (high-dimensional data with few examples), the Lasso selects at most n variables before it saturates. Also if there is a group of highly correlated variables, then the Lasso tends to select one variable from a group and ignore the others. To overcome these limitations, the elastic net adds a quadratic part to the penalty which when used alone is ridge regression (known also as Tikhonov regularization).

As above in the L1 and L2 cases we use likelihood (maximize likelihood). Similarly we will have gradient of Log likelihood as $\triangledown$ll(W) =X$^T(y-p)$.

Total Quality=$X^T(y-p)$ - *derivative*(regularization term), here regularization term = L2 +L1

$X^T(y-p)$ - *derivative* $(\lambda \frac{1}{2}|W|_2^2 + \lambda \| W|_1^1)$, This translates to

Total quality=$X^T(y-p)$- $\lambda(W) - \lambda sign(W)$

W^{t+1} $\leftarrow$W^t + *Stepsize* * *total quality*

This process of weight updation goes on till a stopping criteria is meet. Ideally its the change in weights compared to a threshold.

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

Answer:

L2 regularization- It a acts as Gaussian prior. L2 is better in reducing prediction error than L1 when the predictors are highly correlated. L2 tries to distribute the weight among correlated features. L2 pushes the weights close to zero, but not exactly to zero.

L1 regularization- is like a Laplacian prior. L1 penalty supports sparse solutions, its pushes the weight lower and tries to make them zero.

L3 regularization- Elastic Net is a regularization that linearly combines the L1 and L2 penalties of the lasso and ridge methods. The elastic net method overcomes the limitations of the lasso.

For example, in the "large p, small n" case (high-dimensional data with few examples), the lasso selects at most n variables before it saturates. Also if there is a group of highly correlated variables, then the lasso tends to select one variable from a group and ignore the others. To overcome these limitations, the elastic net adds a quadratic part to the penalty which when used alone is ridge regression(quadratic penalty) term makes the loss function strictly convex, and it therefore has a unique minimum. Elastic net works in two steps. First the ridge regression coefficients and then the lasso performs the shrinkage. This regularization may cause double shrinkage

Observations regarding the weights:

a) For Normal vanilla regression it can be observed that the weights are higher in value than compared to other regularized weights.

b) For L2 regularization it can be observed that weights appears to be well distributed and look somewhat same when we observe then at a glance. There also appear to be more +ve weight than negative(checked 5 samples)

Table 2: Avg Error and Std Error

Classifier	Avg-Error	Std-Error +/-
L1-Logistic	23.2274	0.141673870745
L2-Logistic	23.1962	0.139190026169
Alt-Logistic	23.3390	0.140180838805

Table 3: Sum of weights for each regularization

Regularizer	Avg-sum of weights
L1-Logistic	21.33274242332
L2-Logistic	21.63219045462
Alt-Logistic	21.3697319953
No regularization	22.95795230336

c) L1 regularization gives somewhat small values

d) Alt/Elastic net gives weights at prima facie look slightly higher than L1 but less than L2.

It can be noted that ideally that as we increase the lambda or penalty term the weights reduce but the error increases.

In order to check the weight we need a common metric hence I ran Logistic,L1,L2 and Elastic net on the entire Susy dataset and took sum of the weights (may not be correct but its just an approach). I took 5 random sum of weights for common settings like runs 50, penalty =0.02 . the weights include both +ve and -ve values. Buy averaging the the weight for these samples I have prepared Table 3, which can help us gain some intuition about the movement or behavior of weights for regularization of same value i.e 0.02 (where ever applicable)

However there is no significant benefit of applying regularization on the entire dataset

The best setting for L1 is regwt of 0.01

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.

References

https://en.wikipedia.org/wiki/Naive$_Bayes_classifier$
$https://www.youtube.com/watch?v=r1in0YNetG8$
$https://www.youtube.com/results?q=demystifying+neural$
$http://machinelearningmastery.com/blog/$
$https://www.coursera.org/learn/ml-classification/home$
$http://iamtrask.github.io/2015/07/12/basic-python-network/$
$http://www.derivative-calculator.net/$

Note: I tried adjusting the placement of the table 3 but there was not successful apolozies for the formatting

Good luck!