SANJANA AGARWAL.
sa14593

ML ASSIGNMENT 3.

1.(a) NAIVE BAYES:

For implementation, used the following parameters:

numruns = 3.

trainsize = 80000

testsize = 20000

After three runs:

Run 1 for 'usecolumnones': False

Error : 27.145

Accuracy : 72.855

'usecolumnones' : True

Error : 27.145

Accuracy : 72.855

Run 2 :	'usecolumnones': False	'usecolumnones': True
Error :	26.735	26.735
Accuracy :	73.265	73.265

Run 3 :	'usecolumnones': False	'usecolumnones': True
Error :	26.535	26.535
Accuracy :	73.465	73.465

∴ Average error for Naive Bayes when 'usecolumnones' is False is : 26.805 +- 0.287 where +-0.287 is the standard error.

Average error for Naive Bayes when 'usecolumnones' is True is : 26.805 +- 0.287. where +-0.287 is the standard error.

Thus, I observed no change in the error & accuracy terms, when we add an

extra column of ones.

There should not be a change in accuracy because the extra column of ones should have no impact on Naive Bayes.

μ (mean) for the extra column will be 1 & Standard deviation will be 0.

Thus, it will have no impact on increasing/decreasing the accuracy. Since the standard deviation is 0, it gives rise to a 'divide by 0' error which has been handled.

1.(b)

	RUN 1	RUN 2	RUN 3
Error:	25.465	26.045	25.83
Accuracy:	74.535	73.955	74.17

Trainsize : 80000
Testsize : 20000
numruns : 3
alpha (Learning rate) : 0.001
epochs : 9

Average error : 25.78 +- 0.271, (+-0.271 - Standard error)

I chose to use Stochastic Gradient Descent instead of Batch Gradient Descent, because Batch GD takes longer to run. Thus, we get better results in lesser number of epochs.

Weight updation formula:

self.weight - = $\alpha \left(\sum_{i=1}^{n} (h_\theta(x) - y) x_{ij} \right)$

Scanned by CamScanner

1.(c.) For implementation:

numruns = 3, stepsize = 0.01, epochs = 10, ni = 9, nh = 4

trainsize = 8000

testsize = 2000

After these runs:

	Run 1	Run 2	Run 3
~~Error~~ :	~~23.9~~	~~25.8~~	~~26.0~~
~~Accuracy~~:	~~76.1~~	~~74.2~~	~~74.0~~

	Run1	Run 2	Run 3
Error	23.9	22.5	23.65
Accuracy	76.1	77.5	76.35

Scanned by CamScanner

1.(d) I have reported the errors in the respective algorithms.

Behavior:

(1.) Neural networks provides with the best performance, having the highest accuracy

(2) Logistic regression outperformed Naive Bayes.

(3.) Linear regression & Naive Bayes had a similar performance

(4.) Our algorithms outperformed the Random classifier, as was required.

(5.) Logistic Regression & Neural network perform better with a larger dataset.

(6.) Also, the order in which accuracy increases, variance increases.

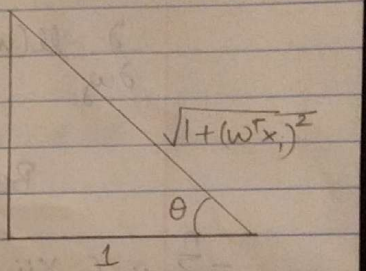
Scanned by CamScanner

2. $$P(y=1|x_i, w) = \frac{1}{2}\left(1 + \frac{w^T x_i}{\sqrt{1+(w^T x_i)^2}}\right)$$

$$\therefore P(y=0|x_i, w) = 1 - \frac{1}{2}\left(1 + \frac{w^T x_i}{\sqrt{1+(w^T x_i)^2}}\right)$$

Let's assume,

$$\frac{w^T x_i}{\sqrt{1+(w^T x_i)^2}} = \sin\theta$$



$$L(w) = \prod_{i=1}^{n}\left[\frac{1}{2}(1+\sin\theta)\right]^{y_i}\left(1-\frac{1}{2}(1+\sin\theta)\right)^{1-y_i}$$

$$= \prod_{i=1}^{n}\left[\left(\frac{1}{2}(1+\sin\theta)\right)^{y_i}\left(\frac{1}{2}(1-\sin\theta)\right)^{1-y_i}\right]$$

Taking log on both sides,

$$ll(w) = \sum_{i=1}^{n}\left[y_i \log\left(\frac{1+\sin\theta}{2}\right) + (1-y_i)\log\left(\frac{1-\sin\theta}{2}\right)\right]$$

Now, $\tan\theta = \frac{\text{opposite side}}{\text{adjacent side}}$

$$\therefore \tan\theta = w^T x_i$$

Taking the partial derivative, wrt w_j

$$\frac{d}{dw_j}\tan\theta = \frac{\partial}{\partial w_j} w^T x_i$$

$$\sec^2\theta \frac{\partial\theta}{\partial w_j} = x_{ij}$$

Scanned by CamScanner

$$\therefore \frac{\partial \theta}{\partial w_j} = x_{ij} \cos^2\theta \quad \text{---} \quad (1.)$$

Using (1.) & taking partial derivative of $ll(w)$, wrt w_j, we get,

$$\frac{\partial}{\partial w_j} ll(w) = \sum_{i=1}^{n} y_i \left[\left(\frac{\cos\theta/2 \cdot d\theta/dw_j}{(\frac{1+\sin\theta}{2})} \right) + (1-y_i) \left(\frac{-\cos\theta/2 \cdot d\theta/dw_j}{(\frac{1-\sin\theta}{2})} \right) \right]$$

But, $d\theta/dw_j = x_{ij} \cos^2\theta$

$$= \sum_{i=1}^{n} y_i \left(\frac{x_{ij} \cdot \cos\theta \cdot \cos^2\theta}{(1+\sin\theta)} \right) + (1-y_i) \left(\frac{x_{ij} * \cos^2\theta * (-\cos\theta)}{(1-\sin\theta)} \right)$$

We know, $\frac{\cos^2\theta}{1+\sin\theta} = 1 - \sin\theta$

& $\frac{\cos^2\theta}{1-\sin\theta} = 1 + \sin\theta$

Thus,

$$= \sum_{i=1}^{n} y_i \cos\theta \cdot x_{ij} (1-\sin\theta) - (1-y_i) \cos\theta \cdot x_{ij} \cdot (1+\sin\theta)$$

$$= \sum_{i=1}^{n} y_i \cos\theta - y_i \cos\theta \cdot \sin\theta - \cos\theta - \cos\theta \cdot \sin\theta + y_i \cos\theta + y_i \cos\theta \cdot \sin\theta$$

$$= \sum_{i=1}^{n} x_{ij} \cdot \cos\theta \left[2y_i - (1+\sin\theta) \right]$$

Substituting for values of $\cos\theta$ & $\sin\theta$,

Scanned by CamScanner

we get,

$$\frac{\partial}{\partial w_j} ll(w) = \sum_{i=1}^{n} x_{ij} \cdot \frac{1}{\sqrt{1+(w^T x_i)^2}} \left[2y_i - \left(1 + \frac{(w^T x_i)}{\sqrt{1+(w^T x_i)^2}}\right)\right]$$

Thus, to find the weight updation rule,

$w^{(t+1)} = w^{(t)} + \Delta w^{(t)}$, we need to find $\Delta w^{(t)}$

$$\nabla ll(w) = f_j^T \cdot g \left[2(y - p) \right]$$

where $f_j^T = x_{ij}$, $g = \frac{1}{\sqrt{1+(w^T x_i)^2}}$

$\therefore$

For stochastic gradient optimization,

$$\Delta w^{(t)} = \eta X^T (y - p^{(t)}) \cdot \frac{1}{\sqrt{1+(w^T x)^2}}$$

$$\therefore w^{(t+1)} = w^{(t)} + \eta X^T (y - p^{(t)}) \cdot \frac{1}{\sqrt{1+(w^T x)^2}}$$

This weight updation was performed in the code and the results were pretty impressive.

The results have been tabulated below.

Scanned by CamScanner

	RUN 1	RUN 2	RUN 3
Error :	21.5	22.7	22.66
Accuracy :	78.5	77.3	77.34

∴ Average error : 22.287 +- 0.623
The value of α = 0.1
testsize = 80000
trainsize = 20000.
numruns = 3.
∴ We observe that the results are very good with this function.
As we increase the number of epochs, we get better results & if we decrease the value of α, we get better results.

Scanned by CamScanner

3.(a) I have applied L2 and L1 regularizer on Logistic Regression.

For L2 regularizer, I have written,

$$R(w) = \lambda \|w_i\|_2^2$$

where lambda is a constant.
w_i is the weight for the i^{th} sample.

So the updation rule I used for weights is,

$$w^{(t+1)} = w^{(t)} - \alpha(\text{loss\_function} - R(w))$$

where $\text{loss\_function} = \sum_{i=1}^{n} (h_\theta(x) - y) x_{ij}$

$$R(w) = \sum_{i=1}^{n} \lambda w_i$$

α = learning parameter.

For L1 regularizer, I have written,

$$R(w) = \lambda \, \text{sign}(w_i) \quad (\lambda |w_i|)$$

where λ is a constant.
w_i is the weight for the i^{th} sample.

So the updation rule, I used for weights is,

$$w^{(t+1)} = w^{(t)} - \alpha(\text{loss\_function} - R(w))$$

where $\text{loss\_function} = \sum_{i=1}^{n} (h_\theta(x) - y) x_{ij}$

Scanned by CamScanner

$$R(w) = \sum_{i=1}^{n} \lambda \, \text{sign}(w_i)$$

which is the L1 regularized loss penalty & derivative of $|w|$.

I used a similar approach for L1 & L2 regularizer. (except the regularization function).

1) I used random weight initialization.
2) I used stochastic gradient descent
3) Tried various values of λ & found on several sources that good values are like $\lambda = 0.1, 0.01$.
4) I updated the weights as per the iterative function stated above.
5) The results in comparison to logistic regression have been tabulated below:

Logistic (no regularization)			Logistic (with L1)			Logistic (with L2)		
RUN 1	RUN 2	RUN 3	RUN 1	RUN 2	RUN 3	RUN 1	RUN 2	RUN 3
25.47	25.71	25.24	26.38	27.115	26.27	26.22	24.995	25.825
74.53	74.29	74.76	73.62	72.885	73.73	73.78	75.005	74.175

Average error for Logistic (no regression): 25.473 +- 0.217 where +- 0.217 is the standard error
Average error for Logistic (L1): 26.588 +- 0.424 where +- 0.424 is the standard error.
Average error for Logistic (L2): 25.68 +- 0.578 where +- 0.578 is the standard error

Scanned by CamScanner

3.(b.) The third regularizer that I chose is the Elastic Net regularizer.
It basically functions as:

$$R(w) = \lambda_1 \|w\|_1 + \lambda_2 \|w\|^2$$

$$= \lambda_1 \|L_1\|_1 + \lambda_2 \|L_2\|^2$$

where L1 & L2 are the L1 & L2 regularizers respectively.
Its performance is a little poorer than individual L1, L2.

The values I achieved are.

	Run 1	Run 2	Run 3
Error	28.995	28.24	27.995
Accuracy	71.005	71.76	72.045

The average error is: 28.397 +- 0.496
where +- 0.496 is the standard error.

The parameters are:

$\alpha = 0.001$
$\lambda_1 = 0.01$
$\lambda_2 = 0.01$
testsize = 80000
trainsize = 20000
epochs = 9

Scanned by CamScanner

3. (c.) The parameters I used are:
$\lambda = 0.01$, $\alpha = 0.001$
trainsize = 80000, testsize = 20000
epochs = 9

The average & standard errors have been written in the above observations.
We see that L2 performs better than L1.
and both L1 & L2 outperform Elastic Net.

Theoretically, L2 gives better performance than L1 because L2 is better for minimizing the prediction error. L1 while reduces overfitting, it performs poorly in prediction compared to L2.
As expected, L1 regularization did produce sparse weight Matrix.
The speed of performance was better for L1 than L2.
We also observe that L1 & L2 have lower variance.

Elastic Net should perform better, but it does not. It gives poor accuracy & high variance.
Thus, all the regression algorithms give better results with increase in epoch & training size & with decrease in step size.

Scanned by CamScanner