CS4011-Assignment 1b

S Aakash CS15B060

September 12, 2017

0.1 Question 7

First,Training and Test Data were feature scaled and mean normalized and 2-class logistic regression was applied on training data.Following is the result:[class 0-forest,class1-mountain]

('training_accuracy:', 0.93822393822393824)
('testset_accuracy:', 0.90000000000000002)

	precision	recall	f1-score	support
0.0	0.94	0.85	0.89	20
1.0	0.86	0.95	0.90	20
avg	0.90	0.90	0.90	40

With L1-regularization(Prof Boyd's program),we obtain following:
Classification result:
[positive class count] 21
[negative class count] 19

	precision	recall	f1-score	support
-1.0	1.00	0.95	0.97	20
1.0	0.95	1.00	0.98	20
avg	0.98	0.97	0.97	40

By comparing F-scores,we observe using L1 regularization helps Logistic regression perform slightly better than without regularization(Possibly there would have been slight over-fitting/outlier which is handled by L1-regularisation).

0.2 Question 8

Part A:
Derivation,assuming the output function to be softmax and activation function to be sigmoid.
Assume one hidden layer(layer-1,k-neurons) and output layer as layer-2,p-neurons.Lets analyse with single training example.
$\widehat{y_{2,j}}$=Softmax($a_{2,j}$)=$\frac{e^{a_{2,j}}}{\sum e^{a_{2,i}}}$
CrossEntropy: L=$\sum_1^p$-$y_i$log $\widehat{y_{2,i}}$=-log $\widehat{y_{2,l}}$ (where y_l=1)
Convention: ∇_t=$\frac{\partial L}{\partial t}$,where t is vector[ie$\nabla_t$ represents gradient of L wrt t]
$\frac{\partial L}{\partial a_{2,i}}$=$\frac{\partial L}{\partial \widehat{y_{2,l}}}$. $\frac{\partial \widehat{y_{2,l}}}{\partial a_{2,i}}$
$\frac{\partial L}{\partial \widehat{y_{2,l}}}$=-$\frac{1}{\widehat{y_{2,l}}}$
$\frac{\partial \widehat{y_{2,l}}}{\partial a_{2,i}}$=($\widehat{y_{2,l}}$-$\widehat{y_{2,l}}$.$\widehat{y_{2,l}}$) ,if i=l
= (-$\widehat{y_{2,l}}$.$\widehat{y_{2,i}}$) ,if i≠l

$\frac{\partial L}{\partial a_{2,i}}$=-($\tau$-$\widehat{y_{2,i}}$),where τ=1,if l==i ,0 otherwise
Hence ∇_{a_2} can be computed using above.

Now $\frac{\partial L}{\partial h_{1,i}}$=$\sum_1^p \frac{\partial L}{\partial a_{2,j}} \cdot \frac{\partial a_{2,j}}{\partial h_{1,i}}$
Now,$\frac{\partial a_{2,j}}{\partial h_{1,i}}$=$W_{2,j,i}$
∇_{h_1}=${W_2}^T \nabla_{a_2}$
Now $\frac{\partial L}{\partial a_{1,i}}$=$\frac{\partial L}{\partial h_{1,i}} \cdot \frac{\partial h_{1,i}}{\partial a_{1,i}}$
wkt,$\frac{\partial h_{1,i}}{\partial a_{1,i}}$=sigmoid($a_{1,i}$).(1-sigmoid($a_{1,i}$))[As h_i=sigmoid(a_i)]
∇_{a_1}=∇_{h_1}.sigmoid(a_1).(1-sigmoid(a_1))[Note all are elementwise multiplications]
Now,
$\frac{\partial L}{\partial W_{2,i,j}}$=$\frac{\partial L}{\partial a_{2,i}} \cdot \frac{\partial a_{2,i}}{\partial W_{2,i,j}}$
$\frac{\partial a_{2,i}}{\partial W_{2,i,j}}$=$h_{1,j}$
∇_{W_2}=$\nabla_{a_2} h_1^T$
Similarly,∇_{W_1}=$\nabla_{a_1} X^T$
$\frac{\partial L}{\partial b2_j}$=$\frac{\partial L}{\partial a_{2,i}}$.1
∇_{b2}=∇_{a_2}
Similarly,∇_{b1}=∇_{a_1}
Hence we have obtained all the derivatives of objective function with respect to parameters needed for gradient descent.
Three options are possible!
(i)Stochastic Gradient Descent
(ii)MiniBatch Gradient Descent
(iii)Batch Gradient descent
Two programs have been written Stochastic and Minibatch gradient descent methods respectively.Hence gradient descent update rule can be easily formulated as we know the gradient of L wrt patameters(ie θ=θ_0-α.$\nabla\theta_0$)The Training Set has been shuffled,feature-scale&mean normalised,stored in DS2-q8-train-normalised.csv
The following is the result obtained using stochastic gradient descent(learning rate=0.01,hidden neurons=20)

	precision	recall	f1-score	support
1.0	0.73	0.55	0.63	20
2.0	0.45	0.90	0.60	20
3.0	0.80	0.60	0.69	20
4.0	0.70	0.35	0.47	20
avg	0.67	0.60	0.60	80

The following is the result obtained using minibatch gradient descent(learning rate=0.01,hidden neurons=20,BatchSize=10)

	precision	recall	f1-score	support
1.0	0.60	0.45	0.51	20
2.0	0.58	0.90	0.71	20
3.0	0.70	0.70	0.70	20

4.0	0.64	0.45	0.53	20
avg	0.63	0.62	0.61	80

We can observe both perform equally well(On this training and test set).The optimal values of the number of hidden layers was found to be around 20-30.The number of inner layer neurons were varied from [5-96],in order of 5 and the model was trained,tested using validation set.Both 20 and 30 gave optimal value.Hence the number of hidden layers were fixed to be 20.

Part B:
Let L be the squared loss function,as given in question.
Convention: $\nabla_t = \frac{\partial L}{\partial t}$,where t is vector[ie$\nabla_t$ represents gradient of L wrt t]
$\frac{\partial L}{\partial a_{2,i}} = \sum_1^p \frac{\partial L}{\partial \widehat{y_{2,j}}} \cdot \frac{\partial \widehat{y_{2,j}}}{\partial a_{2,i}}$
$\frac{\partial L}{\partial \widehat{y_{2,j}}} = \widehat{y_{2,j}} - y_{2,j}$
$\frac{\partial \widehat{y_{2,j}}}{\partial a_{2,i}} = (\widehat{y_{2,i}} - \widehat{y_{2,i}}.\widehat{y_{2,i}})$,if i=j
$= (-\widehat{y_{2,j}}.\widehat{y_{2,i}})$,if i≠j

Hence ∇_{a_2} can be computed using above.

Now $\frac{\partial L}{\partial h_{1,i}} = \sum_1^p \frac{\partial L}{\partial a_{2,j}} \cdot \frac{\partial a_{2,j}}{\partial h_{1,i}}$
Now, $\frac{\partial a_{2,j}}{\partial h_{1,i}} = W_{2,j,i}$
$\nabla_{h_1} = {W_2}^T \nabla_{a_2}$ Now $\frac{\partial L}{\partial a_{1,i}} = \frac{\partial L}{\partial h_{1,i}} \cdot \frac{\partial h_{1,i}}{\partial a_{1,i}}$
wkt, $\frac{\partial h_{1,i}}{\partial a_{1,i}}$=sigmoid($a_{1,i}$).(1-sigmoid($a_{1,i}$))
$\nabla_{a_1} = \nabla_{h_1}$.sigmoid($a_1$).(1-sigmoid($a_1$))[Note all are elementwise multiplications]
Now,
$\frac{\partial L}{\partial W_{2,i,j}} = \frac{\partial L}{\partial a_{2,i}} \cdot \frac{\partial a_{2,i}}{\partial W_{2,i,j}}$+gradient due to regularisation term
$\frac{\partial a_{2,i}}{\partial W_{2,i,j}} = h_{1,j}$
$\nabla_{W_2} = \nabla_{a_2} h_1^T + 2\gamma W_2$
Similarly, $\nabla_{W_1} = \nabla_{a_1} X^T + 2\gamma W_1$
$\frac{\partial L}{\partial b2_j} = \frac{\partial L}{\partial a_{2,i}}.1$+gradient due to regularisation term
$\nabla_{b2} = \nabla_{a_2} + 2\gamma b_2$
$\nabla_{b1} = \nabla_{a_1} + 2\gamma b_1$
Hence we have obtained all the derivatives of objective function with respect to parameters needed for gradient descent.Hence gradient descent update rule can be easily formulated as we know the gradient of L wrt patameters(ie $\theta = \theta_0 - \alpha.\nabla\theta_0$)
Two options are possible!
(i)Stochastic Gradient Descent
(ii)MiniBatch Gradient Descent
(iii)Batch Gradient descent

From question 8-part A,it was observed that both (ii) and (iii) performed similar on training data. Now,we will look at variation of F-scores with respect to gamma,taking Stochastic gradient descent(learning rate=0.01,hidden neurons=20)

γ=0.01

	precision	recall	f1-score	support
1.0	0.50	0.35	0.41	20
2.0	0.59	0.85	0.69	20
3.0	0.42	0.25	0.31	20
4.0	0.40	0.50	0.44	20
avg	0.48	0.49	0.47	80

γ=0.1

	precision	recall	f1-score	support
1.0	0.38	0.50	0.43	20
2.0	0.80	0.60	0.69	20
3.0	0.67	0.60	0.63	20
4.0	0.52	0.55	0.54	20
avg	0.59	0.56	0.57	80

γ=1

	precision	recall	f1-score	support
1.0	0.67	0.60	0.63	20
2.0	0.69	0.90	0.78	20
3.0	0.76	0.65	0.70	20
4.0	0.58	0.55	0.56	20
avg	0.68	0.68	0.67	80

γ=10

	precision	recall	f1-score	support
1.0	0.56	0.50	0.53	20
2.0	0.68	0.75	0.71	20
3.0	0.62	0.90	0.73	20
4.0	0.55	0.30	0.39	20
avg	0.60	0.61	0.59	80

γ=100

	precision	recall	f1-score	support
1.0	0.46	0.30	0.36	20
2.0	0.74	0.70	0.72	20
3.0	0.35	0.40	0.37	20
4.0	0.24	0.30	0.27	20
avg	0.45	0.42	0.43	80

Now optimal value of γ is found to be 1.As we increase γ F score first increases,then starts to decrease,Which means γ=1 is optimal.So γ=1 is the "sweet spot" that we discussed in class,and γ<1 are comparitively overfitting and γ>1 are comparitive underfitting.
Also it was observed that as γ increases from 0.01 to 100,the parameter's absolute values decreases..This is as expected as,γ increases means the regularisation term is given more weight,so the parameter's absolute values must decrease.