CS 4011 : Principles of Machine Learning
Programming Assignment #1 : Part-b

Jahnvi Patel
CS15B046

September 12, 2017

7 Logistic Regression

Binary Logistic Regression or 2-class Logistic Regression is a linear classification model that models the odds of a given data point being in a particular class (0/1) as a linear function in the input variable. Logistic Regression models are fit by maximizing the conditional likelihood of $G = G_i$ given X.
Part-1 of the question implements logistic regression for 2 classes with L-2 penalty.

(i) Data: 40 test instances and 518 training instances on dataset with 96 features.

(ii) Model: Two-class logistic regression classifier with L-2 penalty.

(iii) Parameters: The learnt coefficients $\{\beta_0, \beta_i\}$ for the logit function.

(iv) Objective function: To maximize the conditional log likelihood of G given X.

The per-class classification report for the above model when run on test data is as follows:

Class Label	Precision	Recall	FMeasure
Mountains	0.93	0.70	0.80
Forest	0.76	0.95	0.84

Part-2 of the question utilizes an implementation of the interior-point method for L1-regularized logistic regression. The regularisation parameter is used to control the number of features selected from the dataset. Since the objective function for L1-regularized logisic regression is a convex, non-differentiable problem, the code uses Primal-Dual interior point method for solving the convex optimization problem. We set the value of regularization constant to be an appropriate value (0.01 in this case).
The per-class classification report for the above model when run on test data is as follows:

Class Label	Precision	Recall	FMeasure
Mountains	1.00	0.95	0.97
Forest	0.95	1.00	0.98

8 Backpropagation

Artificial neural networks can be used for classification and an ANN for classification has been implemented in the following way:

1. Activation function: Sigmoid/ Logistic function
2. No. of hidden layers = 1
3. No. of neurons in the hidden layer = 50
4. Output activation function: Softmax
5. Loss function: Cross Entropy
6. Update Method: Stochastic Gradient Descent

It has to be ensured that the data input is well-shuffled so that the mini-batches do not end up with correlated data. Also, data is normalized before-hand to avoid any overflows in the softmax function. The derivation of gradients for the above neural network is as follows:

(i) $\mathcal{L} = -\sum_j t_j \log y_j \implies \frac{\partial \mathcal{L}}{\partial a_L} = -\sum_j t_j \frac{\partial \log y_j}{\partial a_{Lk}}$
$y_j = \frac{1}{\sum_i e^{a_{Li}}} e^{a_{Lj}} \implies \log y_j = a_{Lj} - \log \sum_i e^{a_{Li}}$
$\frac{\partial \log y_j}{\partial a_{Lk}} = \delta_{jk} - \frac{1}{\sum_i e^{a_{Li}}} \frac{\partial \sum_i e^{a_{Li}}}{\partial a_{Lk}}$
$\frac{\partial \sum_i e^{a_{Li}}}{\partial a_{Lk}} = \sum_i e^{a_{Li}} \delta_{ik} = e^{a_{Lk}}$
$\frac{\partial \log y_j}{\partial a_{Lk}} = \delta_{jk} - y_k$
$\frac{\partial y_j}{\partial a_{Lk}} = y_j(\delta_{jk} - y_k)$
$\frac{\partial \mathcal{L}}{\partial a_{Lk}} = \sum_j t_j(y_k - \delta_{jk}) = y_k \left(\sum_j t_j\right) - t_k$
$\implies \frac{\partial \mathcal{L}}{\partial a_{Lk}} = y_k - t_k$ (As probabilities sum up to 1)
$\frac{\partial \mathcal{L}}{\partial a_L} = Y - T$

(ii) Next, we derive:
$\frac{\partial \mathcal{L}}{\partial h_{ij}} = \sum_{m=0}^{k-1} \frac{\partial \mathcal{L}}{\partial a_{i+1,m}} \frac{\partial a_{i+1,m}}{\partial h_{ij}}$
$= \sum_{m=0}^{k-1} \frac{\partial \mathcal{L}}{\partial a_{i+1,m}} W_{i+1,m,j} = (W_{i+1,m,j})^T \nabla_{a_{i+1}} \mathcal{L}$
$\nabla_{h_i} \mathcal{L} = (W_{i+1})^T \nabla_{a_{i+1}} \mathcal{L}$

(iii) Next,
$\frac{\partial \mathcal{L}}{\partial a_{ij}} = \frac{\partial \mathcal{L}}{\partial h_{ij}} \frac{\partial h_{ij}}{\partial a_{ij}}$
$= \nabla_{h_i} \mathcal{L} \cdot [.., g'(a_{ik}), ..]$

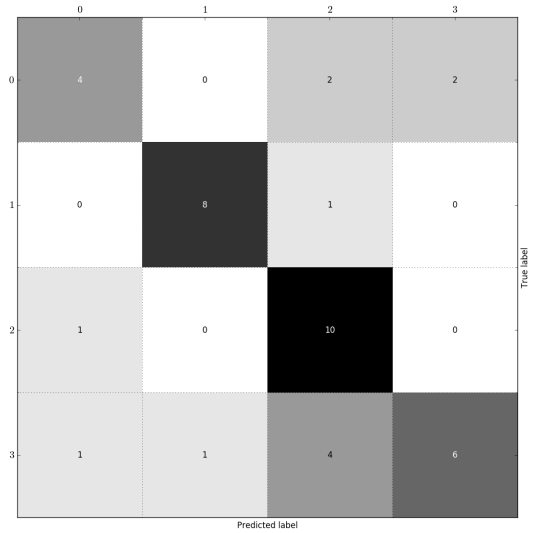
(iv) $\frac{\partial \mathcal{L}}{\partial W_{kij}} = \frac{\partial \mathcal{L}}{\partial a_{ki}} \frac{\partial a_{ki}}{\partial W_{kij}} = \frac{\partial \mathcal{L}}{\partial a_{ki}} h_{k-1,j}$
$\nabla_{W_k} \mathcal{L} = \nabla_{a_k} \mathcal{L} {h_{k-1}}^T$

(v) $\frac{\partial \mathcal{L}}{\partial b_{ki}} = \frac{\partial \mathcal{L}}{\partial a_{ki}} \frac{\partial a_{ki}}{\partial b_{ki}} = \frac{\partial \mathcal{L}}{\partial a_{ki}}$
$\nabla_{b_k} \mathcal{L} = \nabla_{a_k} \mathcal{L}$

The per-class classification report for the above model when run on test data is as follows:

Class Label	Precision	Recall	FMeasure
Mountains	0.67	0.50	0.57
Forest	0.89	0.89	0.89
Insidecity	0.59	0.91	0.71
Coast	0.75	0.50	0.60

The accuracy on the test set is 0.70.
The confusion matrix showing the relation between true label and predicted label is as follows:



The per-class classification report for the above model when run on for only two classes:

Class Label	Precision	Recall	FMeasure
Mountains	0.88	1.00	0.93
Forest	1.00	0.92	0.96

On comparison, it is observed that the ANN-based classification provides better results than normal 2-class Logistic Regression, but worse than interior-point method for L1-regularized logistic regression.
The regularized loss function for ANN ensures that weights do not grow beyond a limit. The regularization parameter γ is used to control the model complexity. For the modified loss function,

(i) $\mathcal{L} = -\sum_j (y_j - t_j)^2 \implies \frac{\partial \mathcal{L}}{\partial a_L} = -\sum_j 2(y_j - t_j)\frac{\partial y_j}{\partial a_{Lk}}$
$y_j = \frac{1}{\sum_i e^{a_{Li}}} e^{a_{Lj}} \implies \log y_j = a_{Lj} - \log \sum_i e^{a_{Li}}$
$\frac{\partial \log y_j}{\partial a_{Lk}} = \delta_{jk} - \frac{1}{\sum_i e^{a_{Li}}} \frac{\partial \sum_i e^{a_{Li}}}{\partial a_{Lk}}$
$\frac{\partial \sum_i e^{a_{Li}}}{\partial a_{Lk}} = \sum_i e^{a_{Li}} \delta_{ik} = e^{a_{Lk}}$
$\frac{\partial \log y_j}{\partial a_{Lk}} = \delta_{jk} - y_k$
$\frac{\partial y_j}{\partial a_{Lk}} = y_j(\delta_{jk} - y_k)$
$\frac{\partial \mathcal{L}}{\partial a_{Lk}} = \sum_j 2(y_j - t_j) t_j (\delta_{jk} - y_k)$

(ii) Next, we derive:
$\frac{\partial \mathcal{L}}{\partial h_{ij}} = \sum_{m=0}^{k-1} \frac{\partial \mathcal{L}}{\partial a_{i+1,m}} \frac{\partial a_{i+1,m}}{\partial h_{ij}}$
$= \sum_{m=0}^{k-1} \frac{\partial \mathcal{L}}{\partial a_{i+1,m}} W_{i+1,m,j} = (W_{i+1,m,j})^T \nabla_{a_{i+1}} \mathcal{L}$
$\nabla_{h_i} \mathcal{L} = (W_{i+1})^T \nabla_{a_{i+1}} \mathcal{L}$

(iii) Next,
$\frac{\partial \mathcal{L}}{\partial a_{ij}} = \frac{\partial \mathcal{L}}{\partial h_{ij}} \frac{\partial h_{ij}}{\partial a_{ij}}$
$= \nabla_{h_i} \mathcal{L} \cdot [.., g'(a_{ik}), ..]$

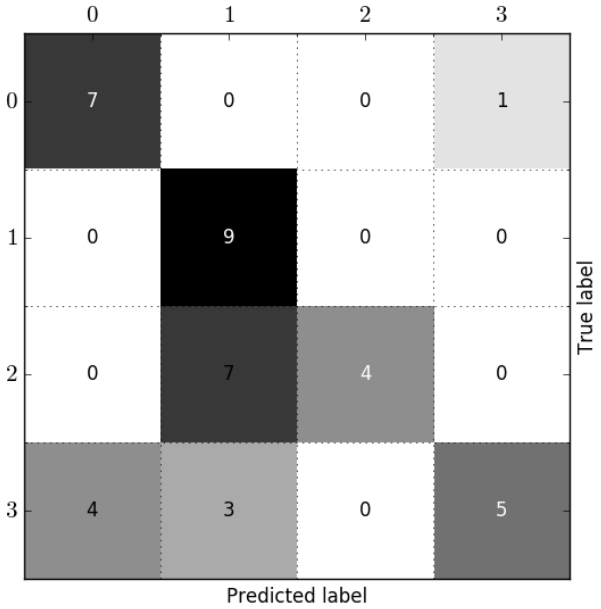
(iv) $\frac{\partial \mathcal{L}}{\partial W_{kij}} = \frac{\partial \mathcal{L}}{\partial a_{ki}} \frac{\partial a_{ki}}{\partial W_{kij}} + 2\gamma W_{kij}$
$= \frac{\partial \mathcal{L}}{\partial a_{ki}} h_{k-1,j} + 2\gamma W_{kij}$
$\nabla_{W_k} \mathcal{L} = \nabla_{a_k} \mathcal{L} {h_{k-1}}^T + 2\gamma W_k$

(v) $\frac{\partial \mathcal{L}}{\partial b_{ki}} = \frac{\partial \mathcal{L}}{\partial a_{ki}} \frac{\partial a_{ki}}{\partial b_{ki}} = \frac{\partial \mathcal{L}}{\partial a_{ki}}$
$\nabla_{b_k} \mathcal{L} = \nabla_{a_k} \mathcal{L}$

The per-class classification report for the above model when run on test data is as follows:

Class Label	Precision	Recall	FMeasure
Mountains	0.64	0.88	0.74
Forest	0.47	1.00	0.64
Insidecity	1.00	0.36	0.53
Coast	0.83	0.42	0.56

The accuracy on the test set is 0.62. The confusion matrix plotted for the obtained output is as follows:



Trend of accuracy wrt γ:
For $\gamma = 0.0001$, we observe an accuracy% of 65. As γ increases initially, the

model complexity decreases and the accuracy increases. This is evident from the observed value for $\gamma = 0.001$, for which an accuracy of 70% is obtained. On further increasing γ, the model would start under-fitting and the accuracy drops drastically to 40% for $\gamma \geq 0.01$. Once γ becomes sufficiently large, the overflow in soft-max prevents us from making meaningful observations. Inference from weights:
It is observed that with increase in value of γ, the weights start moving closer and closer to zero. Therefore, by varying the value of γ, we can alter the model complexity.
It is clear that L-2 regularization is not the best possible alternative for reducing model complexity. An alternative solution has been implemented in the code that considers the validation error for last 3 epochs and stops training, if the 3 consecutive validation errors do not reduce.

9 References:

MachineLearningMastery blog
Petereolants blog