Homework #4

Yongjin Shin
20090488
Industrial Management Engineering, POSTECH
dreimer@postech.ac.kr

September 3, 2018

Problem 1. Show detailed derivation of IRLS algorithm.

Solution Suppose the given data is $(X, Y) = \{(x_i, y_i)|i = 1, 2, 3, ..., N\}$. Then if x_i has M number of features, then matrix X has $M \times N$ dimensions. We assumed that y_i is generated from a Bernoulli distribution with a probability p_i which depends on x_i:

$$y_i \sim Bernoulli(p_i) = p_i^{y_i}(1 - p_i)^{1-y_i} \tag{1}$$

As p_i is between 0 and 1, and y_i is either 1 or 0 but not both, so that we can take sigmoid function to represent p_i:

$$p(y_i = 1|x_i) = \sigma(w^T x_i) \tag{2}$$

To find w, which is the weight of each x_i's features so that it has $M \times 1$ dimension, we need to deploy maximum likelihood estimation like follows:

$$\begin{aligned} p(y|X, w) &= \prod_{i=1}^{N} p(y_i = 1|x_i)^{y_i}(1 - p(y_i = 1|x_i))^{1-y_i} \\ &= \prod_{i=1}^{N} \sigma(w^T x_i)^{y_i}(1 - \sigma(w^T x_i))^{1-y_i} \end{aligned} \tag{3}$$

Then, the log likelihood is as follows:

$$\mathcal{L}(y|X, w) = \sum_{i=1}^{N} y_i \log \sigma(w^T x_i) + (1 - y_i) \log (1 - \sigma(w^T x_i)) \tag{4}$$

As far as we know weight vector w to make log-liekhood biggest , the problem can be solved easily. Since it doesn't have closed form of solution, however, we have another way to find a maximum with using Newton's method. By 2ND order of Taylor series, likelihood can be derived as follows:

$$\mathcal{J}_{(2)}(w) = \mathcal{J}(w^k) + [\nabla \mathcal{J}(w^k)]^T (w - w^k) + \frac{1}{2}(w - w^k)^T \nabla^2 \mathcal{J}(w^k)(w - w^k) \tag{5}$$

where w^k is k-th w values. $\nabla \mathcal{J}(w^k)$ and $\nabla^2 \mathcal{J}(w^k)$ are first derivative and second derivative of w_k. Since we want to find minimum, this polynomial needs to be differentiated with respect to w and the result should be 0. Therefore we can get equation with w as follows:

$$\frac{\partial \mathcal{J}_{(2)}(w)}{\partial w} = \nabla \mathcal{J}(w^k) + \nabla^2 \mathcal{J}(w^k)(w - w^k) = 0 \tag{6}$$

$$w = w^k - [\nabla^2 \mathcal{J}(w^k)]^{-1} \nabla \mathcal{J}(w^k) \tag{7}$$

Now we need first derivative(gradient) and second derivative(hessian) to find w with using w_k.

(1)Gradient

From the original log-likelihood, we can get first derivatives as follows: (where $\sigma(x) = \frac{1}{1+e^{-x}}$)

$$y_i \log \sigma(w^T x_i) = -y_i \log(1 + e^{-w^T x_i}) \tag{8}$$

And,

$$\begin{aligned} (1-y_i)\log(1-\sigma(w^T x_i)) &= -w^T x_i - \log(1+e^{-w^T x_i}) - y_i(-w^T x_i - \log(1+e^{-w^T x_i})) \\ &= -w^T x_i - \log(1+e^{-w^T x_i}) + y_i w^T x_i + y_i \log(1+e^{-w^T x_i}) \end{aligned} \tag{9}$$

So that,

$$\begin{aligned} y_i \log \sigma(w^T x_i) + (1-y_i)\log(1-\sigma(w^T x_i)) &= -w^T x_i - \log(1+e^{-w^T x_i}) + y_i w^T x_i \\ &= -\log(e^{w^T x_i}(1+e^{-w^T x_i})) + y_i w^T x_i \\ &= -log(1+e^{w^T x_i}) + y_i w^T x_i \end{aligned} \tag{10}$$

Therefore

$$\begin{aligned} \frac{\partial \mathcal{L}(w)}{\partial w} &= \frac{\partial}{\partial w} \sum_{i=1}^{N} [-log(1+e^{w^T x_i}) + y_i w^T x_i] \\ &= \sum [\frac{-x_i e^{w^T x_i}}{e^{w^T x_i}+1} + y_i x_i] \\ &= \sum [\frac{-x_i}{e^{-w^T x_i}+1} + y_i x_i] \\ &= \sum [\sigma_i(-x_i) + y_i x_i] \\ &= \sum (y_i - \sigma_i) x_i \end{aligned} \tag{11}$$

(2)Hessian

$$\begin{aligned} \frac{\partial \mathcal{L}(w)}{\partial w \partial w^T} &= \frac{\partial}{\partial w} [\sum (y_i - \sigma_i) x_i]^T \\ &= -\sum [\frac{\partial}{\partial w}(\sigma_i x_i)^T] \\ &= -\sum [\sigma_i (1-\sigma_i) x_i x_i^T] \end{aligned} \tag{12}$$

since,

$$\begin{aligned} \frac{\partial \sigma(x)}{\partial x} &= \frac{\partial}{\partial x}(1+e^{-x})^{-1} \\ &= \frac{e^{-x}}{(1+e^{-x})^2} \\ &= \frac{1}{1+e^{-x}} \times \frac{e^{-x}}{1+e^{-x}} \\ &= \sigma_i \times (1-\sigma_i) \end{aligned} \tag{13}$$

As $\mathcal{J}(w) = -\mathcal{L}(w)$, gradient and hessian are:

$$\begin{aligned} \nabla \mathcal{J}(w) &= -\sum (y_i - \sigma_i) x_i \\ \nabla^2 \mathcal{J}(w) &= \sum [\sigma_i (1 - \sigma_i) x_i x_i^T] \end{aligned} \tag{14}$$

Therefore, plug in equation (14) into equation (7) and then represent matrix form as follows:

$$\begin{aligned} w^{k+1} &= w^k - [\nabla^2 \mathcal{J}(w^k)]^{-1} \nabla \mathcal{J}(w^k) \\ &= w^k - [\sum_{i=1}^{N} \sigma_i (1 - \sigma_i) x_i x_i^T]^{-1} [-\sum_{i=1}^{N} (y_i - \sigma_i) x_i] \\ &= w^k + (X S_k X^T)^{-1} X S_k b_K \\ &= (X S_k X^T)^{-1} [(X S_k X^T) w^k + X S_k b_k] \\ &= (X S_k X^T)^{-1} (X S_k) [X^T w^k + b_k] \end{aligned} \tag{15}$$

where

$$\begin{aligned} S &= \begin{pmatrix} \sigma_1(1-\sigma_1) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_N(1-\sigma_N) \end{pmatrix} \\ b &= [\frac{y_1 - \sigma_1}{\sigma_1(1-\sigma_1)}, ..., \frac{y_N - \sigma_N}{\sigma_N(1-\sigma_N)}]^T \end{aligned} \tag{16}$$

Problem 2. Implement your own IRLS algorithm and evaluate the classification accuracy using the tweet dataset.

Solution

1. Method

Algorithm 1: IRLS

input : Training Dataset $\mathcal{D} = \{(x_n, y_n) | n = 1, ..., N\}$
output: w
Initialize w = 0, and $w_0 = \log(\bar{y}/(1-\bar{y}))$
while *convergence* **do**
  for i = *1,2,3, ..., N* **do**
    Compute $\eta_i = w^T x_i + w_0$;
    Compute $\sigma_i = \sigma(\eta_i)$;
    Compute $s_i = \sigma_i(1-\sigma_i)$;
    Compute $z_i = \eta_i + \frac{y_i - \sigma_i}{s_i}$;
  end
  Construct $S = diag(s_1 : s_N)$;
  Update $w = (XSX^T)^{-1}XSz$;
end

IRLS algorithm was used to implement logistic regression classifier. With using achieved w after a few iterations, plug it into $y_i = \sigma(w^T x_i)$ so that we could get the probability of test data set. If $p_i \geq 0.5$, we can consider it as label 1, and the other case is as label 0. After all, we can compare between actual label and predicted label with the model of IRLS. Attached script of implementation in the end of report.

1.1 Dataset

Training data set is composed of over 47,000 tweet data with labels of 1 and 0. Test data set has over 5,800 tweet data.

1.2 Preprocessing

Since data set is a bunch of sentences, each sentence needs to be separated into single word. In the end, separated words consists of word vector used as a feature. Training data set has 65,693 unique words. Among them, there are 155 *stop words*, such as i, my, a, the , you and etc., which are commonly used. Therefore, 65,538 words was used as the bag of words and top K frequently used words were picked. Hence $(K \times N)$ size of training data set was made where K is the number of features and N is that of data.

2. Result

From (a) of figure 1, the highest test accuracy is 73.86% when K is 3000. On the other hand, the highest training accuracy is 79.22% with 6000 K. In (b), the red color plot is of training accuracy, and the green is of test accuracy. The red one is monotonically increasing within the interval, however the green one is increasing until the peak around from 3000 to 4000 and then decreasing.From the test accuracy, the accuracy is 79.7% and 65.6% when label is 1 and 0, respectively.

(b) of figure 2 has slightly less sharper slop around 0 than (a).

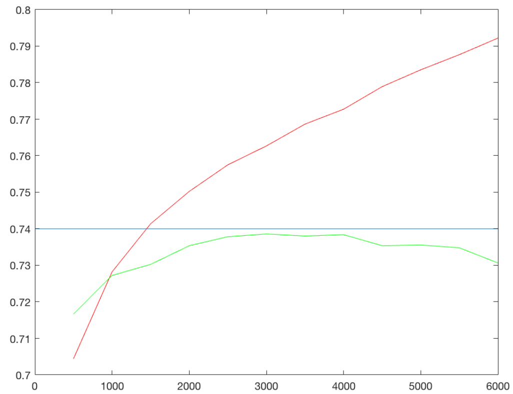
3. Discussion

Since the slope of (b) in figure 2 is less sharper, it is easily assumed that the classification works inside the interval from -5 to 5 is much harder. Therefore low test accuracy can be understandable from figure 2.

As (a) in figure 1 shows, it is quite low training accuracy that the model has. The model cannot learn features enough from the training data, it is hard to say either the result of test accuracy is good or bad. However, it is worth to mention that there must be some reasons to affect that the test accuracy doesn't have non-decreasing graph.First is under-fitting. Since the computational cost is way too expensive, the test with using over 6000 words was not able to be proceeded. Hence the model does not have enough capacity to classify even the training set. Another reason is that there is a calculation for the inverse in the middle of algorithm, however because of the large size of matrix, it comes to be almost singular under

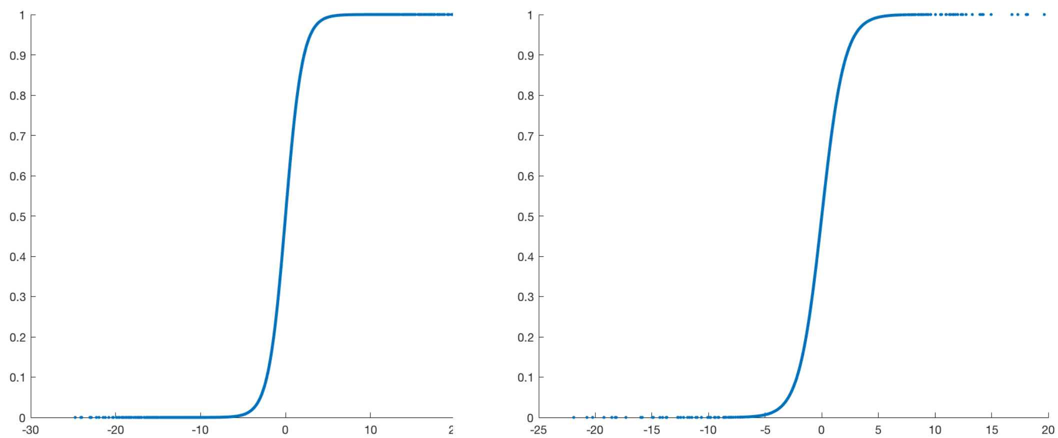
Top K	Training	Test
500	0.7044274	0.7166383
1000	0.7282154	0.7272383
1500	0.7413818	0.7302669
2000	0.7502366	0.7353776
2500	0.7575139	0.7378383
3000	0.762688	**0.7385955**
3500	0.7686613	0.7380276
4000	0.7727416	0.7384062
4500	0.7789252	0.7353776
5000	0.7835314	0.7355669
5500	0.7876748	0.7348098
6000	**0.7922179**	0.7306455

(a) Accuracy Table



(b) Accuracy Plotting

Figure 1: Classification Accuracy



(a) Training Set

(b) Test Set

Figure 2: Probability Distribution

the double precision environment. To prevent this situation, small number(10^{-5}) was added to make it be positive definite. Even though weight has convergence, however, it could possibly have error in the middle of calculation.

References

[1] Kevin Murpy. *Machine Learning: A probabilistic Perspective.*