Machine Learning: Homework #3

Due on April 19, 2018 at 23:59

Professor [redacted]

Name: Zhang ye di **ID:** [redacted]

Problem 1

Perceptron

Part One

Since we update the weight $w(t+1) = w(t) + y(t)x(t)$, then $w(t+1) - w(t) = y(t)x(t)$, and $(y(t) \in R)$ $w^T(t+1) - w^T(t) = y(t)x^T(t) \neq 0$. Then $y(t) \cdot (w^T(t+1) - w^T(t)) \cdot x(t) = y(t) \cdot (y(t)x^T(t)) \cdot x(t) = y^2(t)x^T(t)x(t) > 0$ when $y(t) \neq 0, ||x(t)||_2 \neq 0$.

Part Two

The classify process is: a perceptron fits the data by using a line to separate the +1 from -1 data, and in one iteration the misclassified example is: (x_*, y_*). We use $\hat{y}(t)$ to denote the predict value of y get from $w^T(t)x(t)$. Then $y_*(t)\hat{y}_*(t) = y_*(t)[w^T(t)x_*(t)] < 0$. Then $y_*(t)x_*^T(t)x_*(t)[w^T(t)x_*(t)] < 0$. Moreover, we know $w^T(t+1) - w^T(t) = y_*(t)x_*^T(t)$. Then we get $[w^T(t+1) - w^T(t)]x_*(t)[w^T(t)x_*(t)] < 0$, that is $[\hat{y}_*(t+1) - \hat{y}_*(t)]\ \hat{y}_*(t) < 0$, which means:

- if $\hat{y}_*(t) < 0, \hat{y}_*(t+1) > \hat{y}_*(t)$.
- if $\hat{y}_*(t) > 0, \hat{y}_*(t+1) < \hat{y}_*(t)$.

So the move form $w(t)$ to $w(t+1)$ is a move in the right direction.

Problem 2

Understanding logistic regression

Part One

We can rewrite logistic regression like:

$$w^T x = \log \frac{y}{1-y}$$

For this training data set, $y(i) \in \{0, 1\}$. then logistic regression is linear regression for *log odd ratio*, which means:

$$w^T x = \log \frac{Pr(y=1|x)}{Pr(y=0|x)}$$

So LR model is a meaningful probabilistic model like: $Pr(y=1|x) = \frac{1}{1+e^{-w^T x}}$.

Part Two

For this data set $y \in \{0, 1\}$, we calculate maximum likelihood estimation function like :

$$l(w) = \sum_{i=1}^{m} [y_i \cdot ln(\sigma(w^T x_i)) + (1 - y_i) \cdot ln(1 - \sigma(w^T x_i))]$$

The ideal loss function should guarantee the maximum likelihood estimation, i.e. the ideal loss function should be like (for example):

$$J = -\frac{1}{m} \cdot \sum_{i=1}^{m} [y_i ln\sigma(w^T x_i) + (1 - y_i) ln(1 - \sigma(w^T x_i))]$$

So we can see the MSE is not a good loss function for logistic regression, it cannot maximize the log likelihood estimation function

Part Three

Since in this 3-class task, $y \in \{1, 2, 3\}$, we define:

$$p(y = j|x) = \frac{e^{w_j^T x}}{\sum_{i=1}^{3} e^{w_i^T x}} \qquad for\ j \in \{1, 2, 3\}$$

where: $\sum_{j=1}^{3} p(y = j|x) = 1$.

Then the loss function is like:

$$J(w) = -\sum_{i=1}^{m} \sum_{j=1}^{3} 1\{y_i = j\} \cdot ln(p(y = j|x))$$

i.e.

$$J(w) = -\sum_{i=1}^{m} \sum_{j=1}^{3} 1\{y_i = j\} \cdot ln(\frac{e^{w_j^T x}}{\sum_{k=1}^{3} e^{w_k^T x}})$$

where $1\{\cdot\}$ is a function defined like: $1\{true\ statement\} = 1$, $1\{false\ statement\} = 0$.

Problem 3

Regularization

Part One

Since $Pr\{t; x, \mathbf{w}, \sigma\} \sim N(y(x, \mathbf{w}), \sigma)$, the log likehood is: $Lm\{t; x, \mathbf{w}, \sigma\} = \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma}} exp(-\frac{(t_i - y_i)^2}{2\sigma})$

Then: $lm = \sum_{i=1}^{m} ln(\frac{1}{\sqrt{2\pi\sigma}} exp(-\frac{(t_i - y_i)^2}{2\sigma})) = -\frac{m}{ln(\sqrt{2\pi\sigma})} - \sum_{i=1}^{m} [\frac{(t_i - y_i)^2}{2\sigma}]$

when we maximize lm, we minimize the sum of squares error function: $\sum_{i=1}^{m} (t_i - y_i)^2$.

Part Two

Since $\mathbf{w}$ is distributed as the Guassian distribution, we get:

$Pr(t, \mathbf{w}; x, \sigma, \alpha) = Pr(t; x, \mathbf{w}, \sigma) \cdot Pr(\mathbf{w}; \alpha) = Pr(t; x, \mathbf{w}, \sigma) \cdot Pr(w_0; \alpha) \cdot Pr(w_1; \alpha)$

Then the log likehood function should be like:

$lm\{t, \mathbf{w}; x, \sigma, \alpha\}$

$= \sum_{i=1}^{m} ln(\frac{1}{\sqrt{2\pi\sigma}} \cdot \frac{1}{2\pi\alpha} \cdot exp(-\frac{w_0^2 + w_1^2}{2\alpha} - \frac{(t_i - w_0 - w_1 x_i)^2}{2\sigma}))$

$= -\frac{m}{ln(\sqrt{2\pi\sigma} \cdot 2\pi\alpha)} - \sum_{i=1}^{m} (\frac{w_0^2 + w_1^2}{2\alpha} + \frac{(t_i - w_0 - w_1 x_i)^2}{2\sigma})$

Then maximize the log likehood function is equal to (Note that here the objective function (loss function) has a factor: $\frac{1}{m}$.) :

$$\min_{\mathbf{w}}: \quad \frac{1}{m} \cdot \sum_{i=1}^{m} \left(\frac{w_0^2 + w_1^2}{2\alpha} + \frac{(t_i - w_0 - w_1 x_i)^2}{2\sigma} \right)$$

i.e.

$$\min_{\mathbf{w}}: \quad \frac{w_0^2 + w_1^2}{2\alpha} + \frac{1}{m} \cdot \sum_{i=1}^{m} \left(\frac{(t_i - w_0 - w_1 x_i)^2}{2\sigma} \right)$$

Since $\frac{w_0^2+w_1^2}{2\alpha} = \frac{||\mathbf{w}||_2^2}{2\alpha} = \frac{\mathbf{w}^T\mathbf{w}}{2\alpha}$, then the regularization parameter $\lambda = \frac{1}{2\alpha}$.

Problem 4

Program Logistic regression in matlab

Part One

The norm of gradient corresponding to three algorithms has been shown in Fig1. We set the stop condition is : $||gradient|| < 0.0001$.

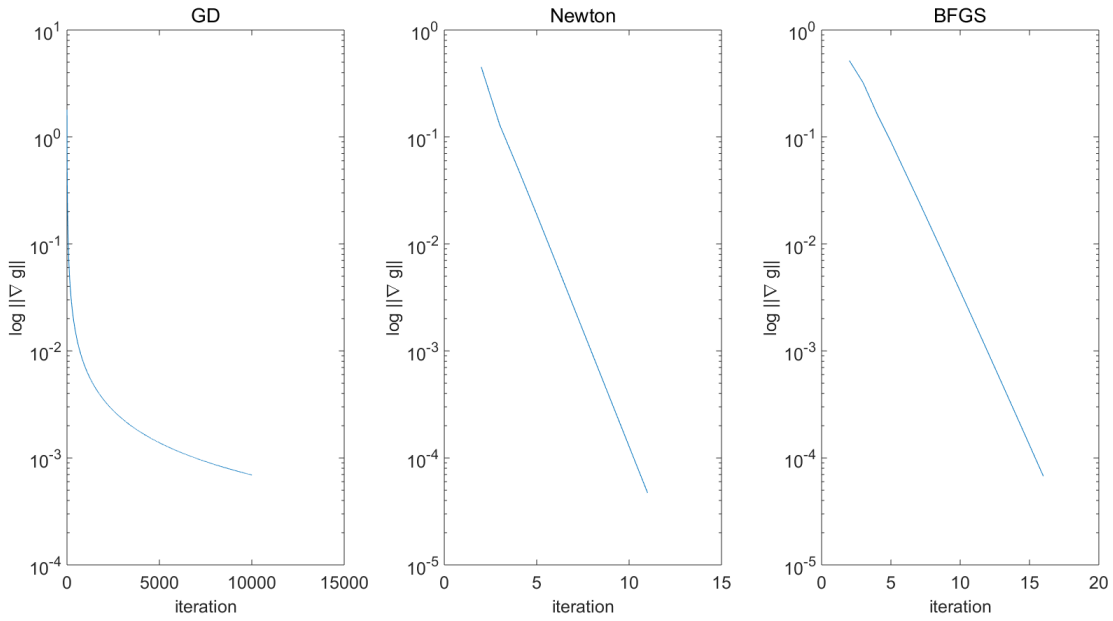

Figure 1: The norm of gradient in three methods

Part Two

The number of misclassified data is 24, and the accuracy of prediction in BFGS is $acc \approx 0.968627$, calculated by $\frac{|wrong\ predict|}{765}$. The norm of gradient is shown in Fig 2.

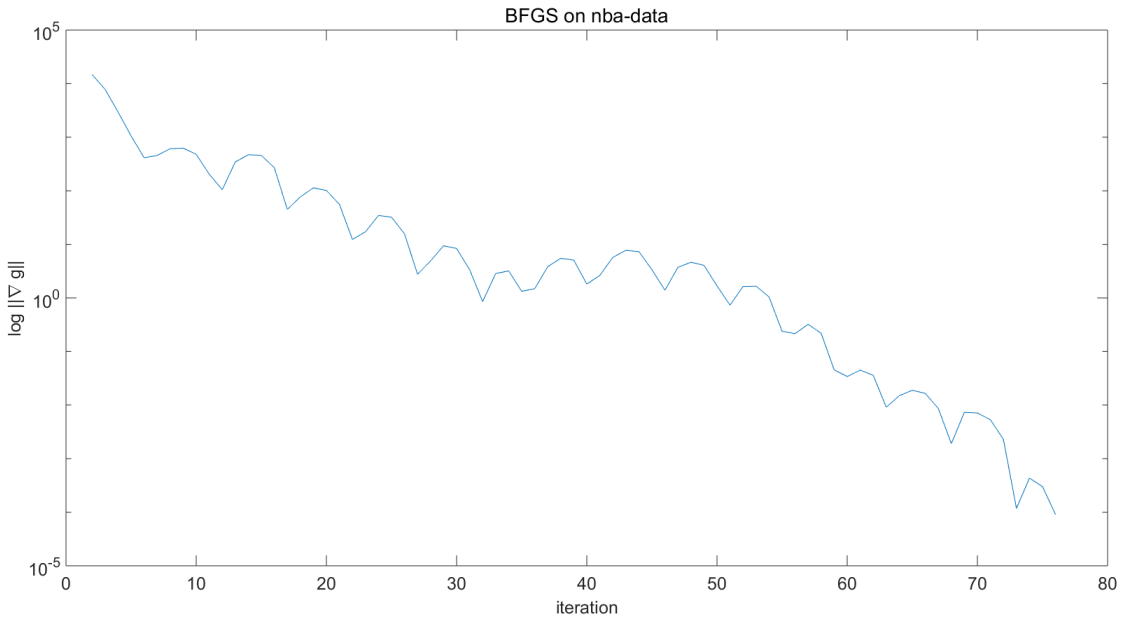

Figure 2: The norm of gradient of BFGS on nba-data

Part Three

The number of misclassified data in modified BFGS is 18, and the accuracy of prediction in BFGS is $acc \approx 0.976471$, calculated by $\frac{|wrong\ predict|}{765}$. The comparison between Modified BFGS and BFGS on nba-data is shown in Fig 3, Fig 4. We can see that the accuracy is improved and the iteration time is also decreased a little.

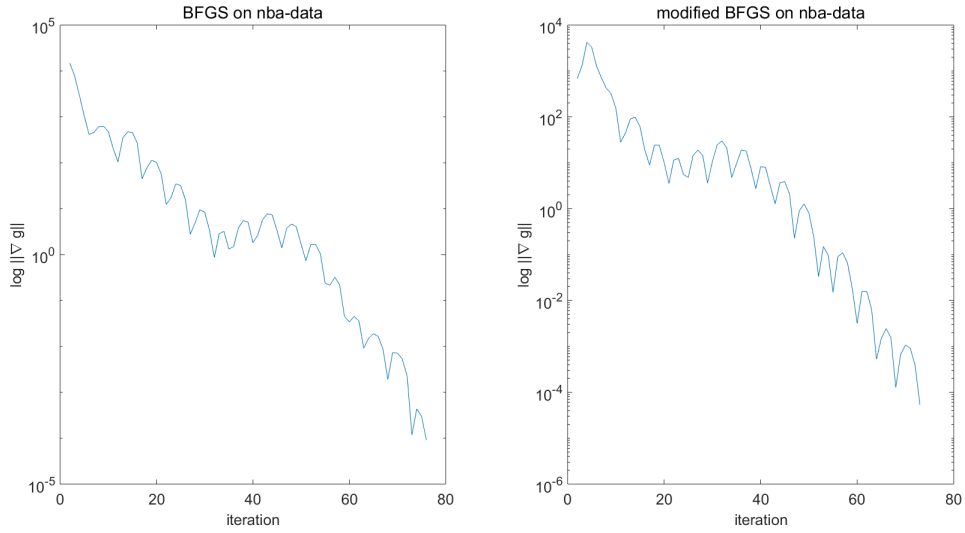

Figure 3: The comparison between Modified BFGS and BFGS on nba-data

Part Three – Analysis

```
命令行窗口
  the number of misclassified data in BFGS is: 24.000000, and pred_accuracy is: 0.968627
fx the number of misclassified data in modified BFGS is: 18.000000, and pred_accuracy is: 0.976471 >>
```

Figure 4: The comparison between Modified BFGS and BFGS on nba-data

In original learning of BFGS, we treat all misclassifications equally, which causes issues in imbalanced classification problems, as there is no extra reward for identifying the minority class over the majority class. So we can penalize misclassifications of the minority class more heavily than we do with misclassifications of the majority class, in hopes that this increases the true positive rate. So in the modified BFGS, we can do it in a similar but simplified way:

```
% -------- modify part in BFGS begin ---------- %
  index_pos = find(h>=0.5);
  index_neg = find(h<0.5);
  h(index_pos) = h(index_pos)+2.0*(h(index_pos)-0.5).*(h(index_pos)-1);
  h(index_neg) = h(index_neg)+2.0*(h(index_neg)-0.5).*(h(index_neg));
% -------- modify part in BFGS end ---------- %
```

Here h is our predicted y. For $h \geq 0.5$, we decrease its probability of reaching 1, and for $h < 0.5$, we increase its probability of reaching 0 (i.e. decrease its probability of reaching 1); Then we can get a more fitting regression function. Note that when we do the decrease and increase, we can change not its judgement, i.e. the new h(index-pos) should still be larger than 0.5, and the new h(index-neg) should still be smaller than 0.5. More details can be got from Fig 5 and Fig 6:

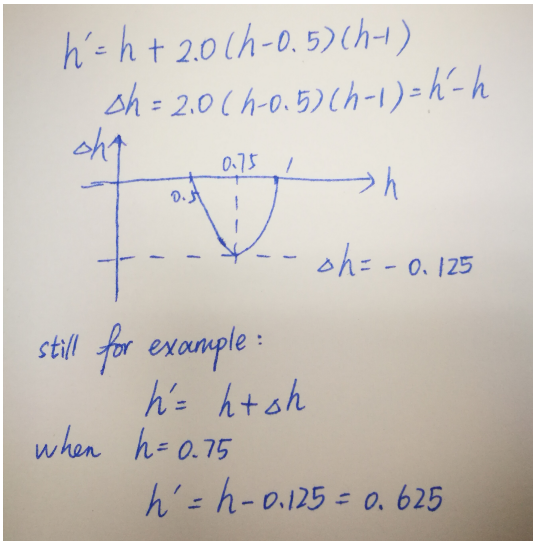

Figure 5: h(index-pos) penalize function

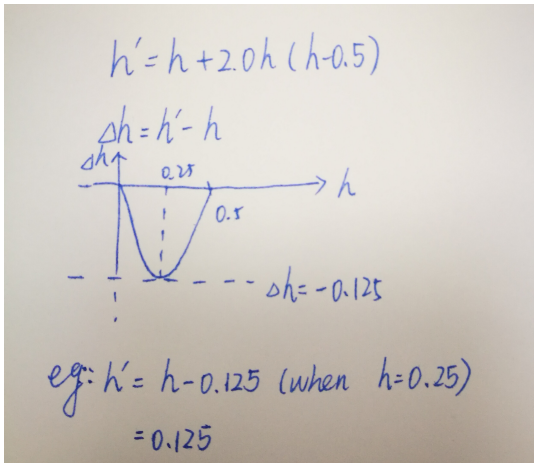

Figure 6: h(index-neg) penalize function