Machine Learning Homework 1, Fall 2018

Jianqiao Hao, 2018211670
School of Economics and Management
hjq18@mails.tsinghua.edu.cn

Sep 24th, 2018

1. The linear decision boundary in d-dimensional space is a hyperplane in $\mathbb{R}^d$, whose equation is
$$g(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + w_0 = 0,$$
where $\mathbf{x} = (x_1, x_2, \cdots, x_d)^T, \mathbf{w} = (w_1, w_2, \cdots, w_d)^T$.

   (a) According to the linear algebra, the norm vector of the decision boundary above is $\mathbf{w} = (w_1, w_2, \cdots, w_d)^T$, and the unit norm vector is $\frac{\mathbf{w}}{\|\mathbf{w}\|}$, where $\|\mathbf{w}\|$ is the L_2 norm(which is also named the Euclidean norm) of $\mathbf{w}$.

   (b) I'd like to derive the distance from any given point $\mathbf{y} = (y_1, y_2, \cdots, y_d)^T$ to the decision boundary by using the method of Lagrange multiplers. [*]
   Let $\mathbf{x} = (x_1, x_2, \cdots, x_d)^T$ be an arbitrary point on the decision boundary. Then the distance from $\mathbf{y}$ to the decision boundary should be the minimum value of $\|\mathbf{y} - \mathbf{x}\|$. In other words, we want to find $\mathbf{x}$ such that
$$\min \|\mathbf{y} - \mathbf{x}\|^2 = (\mathbf{y} - \mathbf{x})^T(\mathbf{y} - \mathbf{x}) = \sum_{i=1}^{d}(y_i - x_i)^2, \quad s.t. \quad w_0 + \mathbf{w} \cdot \mathbf{x} = w_0 + \sum_{i=1}^{d} w_i x_i = 0.$$
   Then the Lagrangian is
$$L = \sum_{i=1}^{d}(y_i - x_i)^2 + \lambda(w_0 + \sum_{i=1}^{d} w_i x_i).$$
   It's clear that for any $i = 1, 2, \cdots, d$, we have
$$\frac{\partial L}{\partial x_i} = \lambda w_i - 2(y_i - x_i) = 0.$$
   Hence,
$$y_i - x_i = \frac{1}{2}\lambda w_i. \tag{0.1}$$
$$x_i = y_i - \frac{1}{2}\lambda w_i \tag{0.2}$$
   From (0.1), we have
$$\min \|\mathbf{y} - \mathbf{x}\|^2 = \frac{1}{4}\lambda^2 \sum_{i=1}^{d} w_i^2,$$

[*]The method is with reference to https://math.stackexchange.com/questions/2639662/find-the-distance-between-a-point-and-a-hyperplane-in-mathbbrn.

which means the distance from $\mathbf{y}$ to the decision boundary equals

$$distance = \min \|\mathbf{y} - \mathbf{x}\| = \frac{1}{2}|\lambda|\|\mathbf{w}\|. \tag{0.3}$$

Also note that

$$\frac{\partial L}{\partial \lambda} = w_0 + \mathbf{w} \cdot \mathbf{x} = w_0 + \sum_{i=1}^{d} w_i x_i = 0, \tag{0.4}$$

Substitute (0.2) into (0.4), we have

$$w_0 + \mathbf{w} \cdot \mathbf{y} - \frac{1}{2}\lambda \sum_{i=1}^{d} w_i^2 = 0,$$

then

$$\lambda = \frac{w_0 + \mathbf{w} \cdot \mathbf{y}}{\frac{1}{2}\|\mathbf{w}\|^2}. \tag{0.5}$$

Substitute (0.5) into (0.3), we have the distance from $\mathbf{y}$ to the decision boundary is

$$distance = \min \|\mathbf{y} - \mathbf{x}\| = \frac{|w_0 + \mathbf{w} \cdot \mathbf{y}|}{\|\mathbf{w}\|}.$$

(c) From (b), let $\mathbf{y}$ be the original point, i.e. $\mathbf{y} = (0, 0, \cdots, 0)^T$, then it is clear that the distance from the original point to the decision boundary equals

$$distance = \frac{|w_0|}{\|\mathbf{w}\|}.$$

The figure below shows the quantities in 3-dimensional space.

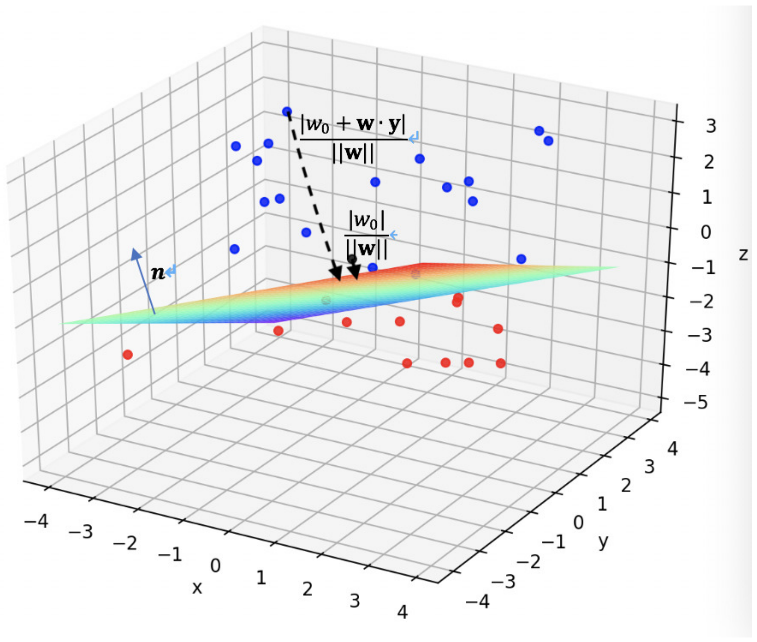

Figure 1: Basic quantities in 3-dimensional space

2. The proof is given below.[†]

   Denote $\mathbf{a}$ as the augmented weighted vector, and $\mathbf{y}$ as the normalized augmented feature vector. Without loss of generality, we set the learning rate η to be 1. Then the iterating formula given by fixed-increment perceptron algorithm is

   $$\mathbf{a}(k+1) = \mathbf{a}(k) + \mathbf{y}^k, \tag{0.6}$$

   where $k = 1, 2, \cdots$, and $\mathbf{y}^k$ refers to the k-th misclassified data in sample set $\{\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_n, \mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_n, \cdots\}$.
   Then I go on to prove the convergence of the algorithm by examining the Euclidean norm between each $\mathbf{a}(k)$ and the solution vector, which can separate each sample correctly.
   Denote $\hat{\mathbf{a}}$ as an arbitrary solution vector, then it is clear that for any positive real number α, $\hat{\mathbf{a}}^* = \alpha\hat{\mathbf{a}}$ is also a solution vector. From (0.6), we have

   $$\mathbf{a}(k+1) - \alpha\hat{\mathbf{a}} = \mathbf{a}(k) - \alpha\hat{\mathbf{a}} + \mathbf{y}^k,$$

   Therefore,

   $$\|\mathbf{a}(k+1) - \alpha\hat{\mathbf{a}}\|^2 = \|\mathbf{a}(k) - \alpha\hat{\mathbf{a}}\|^2 + 2\mathbf{a}(k)^T\mathbf{y}^k - 2\alpha\hat{\mathbf{a}}^T\mathbf{y}^k + \|\mathbf{y}^k\|^2.$$

   Since $\mathbf{y}^k$ is misclassified, then $\mathbf{a}(k)^T\mathbf{y}^k \le 0$. Hence,

   $$\|\mathbf{a}(k+1) - \alpha\hat{\mathbf{a}}\|^2 \le \|\mathbf{a}(k) - \alpha\hat{\mathbf{a}}\|^2 - 2\alpha\hat{\mathbf{a}}^T\mathbf{y}^k + \|\mathbf{y}^k\|^2. \tag{0.7}$$

   Denote $\beta^2 = \max_i \|\mathbf{y}_i\|^2, \gamma = \min_i \hat{\mathbf{a}}^T\mathbf{y}_i > 0$, then from (0.7),

   $$\|\mathbf{a}(k+1) - \alpha\hat{\mathbf{a}}\|^2 \le \|\mathbf{a}(k) - \alpha\hat{\mathbf{a}}\|^2 - 2\alpha\gamma + \beta^2.$$

   If we choose $\alpha = \frac{\beta^2}{\gamma}$, then we have

   $$\|\mathbf{a}(k+1) - \hat{\mathbf{a}}^*\|^2 \le \|\mathbf{a}(k) - \hat{\mathbf{a}}^*\|^2 - \beta^2. \tag{0.8}$$

   From (0.8), it it easy to know after k times updates,

   $$\|\mathbf{a}(k+1) - \hat{\mathbf{a}}^*\|^2 \le \|\mathbf{a}(1) - \hat{\mathbf{a}}^*\|^2 - k\beta^2. \tag{0.9}$$

   From (0.8) and (0.9), we can know that each time the algorithm finds a misclassified point, the square of the Euclidean distance between $\mathbf{a}(k)$ and the solution vector can be reduced by β^2. Since the LHS(left-hand-side) of (0.8) and (0.9) must be non-negative, then the distance between $\mathbf{a}(k)$ and the solution vector can finally converge to zero, which means that $\mathbf{a}(k)$ can finally converge to the solution vector.
   In addition, also considering that the LHS of (0.8) and (0.9) must be non-negative, the times of updates k_0 must satisfy that

   $$k_0 \le \frac{\|\mathbf{a}(1) - \hat{\mathbf{a}}^*\|^2}{\beta^2},$$

   which means, no matter what $\mathbf{a}(1)$ chooses, the times of updates which the algorithm takes to converge to the solution vector must be finite and cannot be more than $\frac{\|\mathbf{a}(1)-\hat{\mathbf{a}}^*\|^2}{\beta^2}$.

3. From straightforward calculations, we have

   (a) $\theta(s) = \frac{1}{1+e^{-s}} = \frac{1}{1+\frac{1}{e^s}} = \frac{e^s}{e^s+1}$,

   (b) $1 - \theta(s) = 1 - \frac{e^s}{e^s+1} = \frac{1}{e^s+1} = \theta(-s)$,

[†]The proof is with reference to the textbook *Pattern Classification* written by Richard O. Duda, Peter E. Hart and David G. Stork.

(c) $\theta'(s) = \frac{e^s\cdot(e^s+1)-e^s\cdot e^s}{(e^s+1)^2} = \frac{e^s}{(e^s+1)^2} = \frac{e^s}{e^s+1}\cdot\frac{1}{e^s+1} = \theta(s)(1-\theta(s))$.

Then I'm going to derive the relationship between $f(s) = \tanh s$ and the $\theta(s)$. From straightforward calculations,

$$f(s)+1 = \frac{e^s - e^{-s}}{e^s + e^{-s}} + 1 = \frac{e^{2s}-1}{e^{2s}+1} + 1 = \frac{2e^{2s}}{e^{2s}+1} = 2\theta(2s),$$

therefore,

$$f(s) = 2\theta(2s) - 1.$$

So the derivative of $f(s)$ is

$$f'(s) = 4\theta'(2s) = 4\theta(2s)(1-\theta(2s)) = \frac{4e^{2s}}{(e^{2s}+1)^2} = \frac{4}{e^{2s}+e^{-2s}+2} = 1-[f(s)]^2.$$

4. The solution is given below.[‡]

Following the settings in the lecture, we would like to use $h(\mathbf{x}) = \theta(\mathbf{w}^T\mathbf{x})$ to estimate $f(\mathbf{x}) = P(y|\mathbf{x})$, where $\mathbf{x}\in\mathbb{R}^{N+1}$ is the augmented feature vector, $\mathbf{w}\in\mathbb{R}^{N+1}$ is the augmented weight vector, $y\in\{-1,1\}$, and $\theta(s)$ is the sigmoid function.

Consider the probability that we can gain the data $\{(\mathbf{x}_1,y_1),(\mathbf{x}_2,y_2),\cdots,(\mathbf{x}_N,y_N)\}$

$$p(\mathbf{x}_1)p(y_1|\mathbf{x}_1)\cdot p(\mathbf{x}_2)p(y_2|\mathbf{x}_2)\cdot\cdots\cdot p(\mathbf{x}_N)p(y_N|\mathbf{x}_N),$$

if we use h to estimate f, from Problem 3, we have

$$p(y_i|\mathbf{x}_i) = \begin{cases} h(\mathbf{x}_i) & \text{if } y_i = 1 \\ 1-h(\mathbf{x}_i) = h(-\mathbf{x}_i) & \text{if } y_i = -1 \end{cases}$$

then we can rewrite the probability above as

$$p(\mathbf{x}_1)h(y_1\mathbf{x}_1)\cdot p(\mathbf{x}_2)h(y_2\mathbf{x}_2)\cdot\cdots\cdot p(\mathbf{x}_N)h(y_N\mathbf{x}_N),$$

and we want to maximize this probability. Note that no matter which h we choose, $p(\mathbf{x}_1),p(\mathbf{x}_2),\cdots,p(\mathbf{x}_N)$ are still the same. Therefore we only need to maximize

$$\prod_{j=1}^N h(y_j\mathbf{x}_j) = \prod_{j=1}^N \theta(y_j\mathbf{w}^T\mathbf{x}_j).$$

Since it may be easier for us to deal with the sum, we logarithm the probability above and take the log likelihood function. We may want to use the gradient descent algorithm, therefore we also take the negative sign of the probability above. According to the lecture given by Hsuan-Tien Lin, in order to make the function seem to be an error measure, we also multiply a factor $\frac{1}{N}$ and it won't affect our optimization problem. Therefore, we obtain our likelihood function

$$\begin{aligned}\min \quad E(\mathbf{w}) &= -\frac{1}{N}\ln\big(\prod_{j=1}^N\theta(y_j\mathbf{w}^T\mathbf{x}_j)\big)\\ &= \frac{1}{N}\sum_{j=1}^N\ln\big(\frac{1}{\theta(y_j\mathbf{w}^T\mathbf{x}_j)}\big)\\ &= \frac{1}{N}\sum_{j=1}^N\ln\big(1+e^{-y_j\mathbf{w}^T\mathbf{x}_j}\big).\end{aligned}$$

[‡]The solution is with reference to the Machine Learning Foundations course material given by Hsuan-Tien Lin.

Then we can calculate the gradient of $E(\mathbf{w})$

$$\begin{aligned}\nabla E(\mathbf{w}) &= \frac{1}{N}\sum_{j=1}^{N}\frac{1}{1+e^{-y_j\mathbf{w}^T\mathbf{x}_j}}e^{-y_j\mathbf{w}^T\mathbf{x}_j}(-y_j\mathbf{x}_j)\\ &= -\frac{1}{N}\sum_{j=1}^{N}\frac{e^{-y_j\mathbf{w}^T\mathbf{x}_j}}{1+e^{-y_j\mathbf{w}^T\mathbf{x}_j}}y_j\mathbf{x}_j\\ &= -\frac{1}{N}\sum_{j=1}^{N}\frac{1}{1+e^{y_j\mathbf{w}^T\mathbf{x}_j}}y_j\mathbf{x}_j\\ &= -\frac{1}{N}\sum_{j=1}^{N}\theta(-y_j\mathbf{w}^T\mathbf{x}_j)y_j\mathbf{x}_j.\end{aligned}$$

Therefore, we can give the basic gradient descent optimization algorithm of logistic regression

- Step1: Initialize $\mathbf{w}(0)$ randomly,
- Step2: Use the formula $\mathbf{w}(k+1) = \mathbf{w}(k) - \eta\nabla E$ to update the $\mathbf{w}$, where $\nabla E = -\frac{1}{N}\sum_{j=1}^{N}\theta(-y_j\mathbf{w}(k)^T\mathbf{x}_j)y_j\mathbf{x}_j$ and η is a positive real number representing the learning rate, until meeting the stopping criterion,
- Step3: Return the final $\mathbf{w}$ as the result.