Machine Learning Problem Set 2, Fall 2018

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

Sep 29th, 2018

1. Note: The answer to this problem is with reference to the website/material listed below:

   - Image Classification Algorithm - Amazon SageMaker
   - Classify your own images using Amazon SageMaker — AWS Machine Learning Blog
   - The Top 5 Uses of Image Recognition - Imagga Blog
   - Generate Hotel Listing Through Aggregate Image From Travel Websites — Imagga Technologies Ltd.

   One of the real-world applications of the supervised machine learning method is the image classification. Image classification is a quite meaningful task, especially for the businesses/companies that possess huge visual databases. Those companies would like to use the image classification methods to help themselves make sense and organize the visual data in a more efficient way, because if the images are left uncategorized, the visual data will be less valuable to them.

   Supervised machine learning provides an effective solution to the image classification problem. For example, Amazon has given an supervised learning algorithm named 'The Amazon SageMaker image classification algorithm' to help users deal with the image classification task. Users could use this algorithm to train their labeled data sets, and after training, users are able to take images as input and then they could know which category each image should be classified into.

   For instance, Tavisca, which is a travel technology company, has made use of the image classification method. Tavisca's hotel content system has an unclassified image dataset of more than 25 million images from hotels all over the world and wants to classify all these images across pre-defined categories. Based on the supervised machine learning technology, Tavisca now has a powerful image classifier which processes and categorizes excessive amounts of photos per day with a high precision rate.

   To sum up, I'd like to show how the task above is formulated as a machine learning problem:

   - Task/Objective: Classifying the image into several pre-defined categories
   - Training Data: images stored in the visual databases and each image has a label of which category it should be classified into, and $\mathbf{x}$ defined as the images/image features, y defined as the several predefined categories(i.e. $1, 2, \cdots, n$)
   - Machine Learning Methods: a proper model, such as a convolutional neural network; the learning algorithm, such as The Amazon SageMaker image classification algorithm

2. Note: the solution is with reference to the lecture slides and the reference book *Pattern Classification* written by Richard O. Duda et al. , and Lagrange multiplier - Wikipedia.

Following the settings in the lecture, assume that we can category the sample set $X = \{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$ into two classes(X_1 and X_2). We also denote the projection $X \to Y : y_i = \mathbf{w}^T\mathbf{x}_i, i = 1, \cdots, N$.

In X space, we define class mean $\mathbf{m}_i = \frac{1}{N_i}\sum_{\mathbf{x}_j \in X_i} \mathbf{x}_j, i = 1, 2$, within-class scatter matrix $\mathbf{S}_i = \sum_{\mathbf{x}_j \in X_i}(\mathbf{x}_j - \mathbf{m}_i)(\mathbf{x}_j - \mathbf{m}_i)^T, i = 1, 2$, total within-class scatter matrix $\mathbf{S}_w = \mathbf{S}_1 + \mathbf{S}_2$, and between-class scatter matrix $\mathbf{S}_B = (\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^T$.

In Y space, we define class mean $\tilde{m_i} = \frac{1}{N_i}\sum_{y_j \in Y_i} y_j, i = 1, 2$, within-class scatter $\tilde{S_i} = \sum_{y_j \in Y_i}(y_j - \tilde{m_i})^2$ *, total within-class scatter $\tilde{S_w} = \tilde{S_1} + \tilde{S_2}$, and between-class scatter $\tilde{S_b} = (\tilde{m_1} - \tilde{m_2})^2$.

Since Fisher would like to maximum the between-class scatter and to minimum the within-class scatter, he gives the Fisher's Criterion

$$\max \quad J_F(\mathbf{w}) = \frac{(\tilde{m_1} - \tilde{m_2})^2}{\tilde{S_1} + \tilde{S_2}} \qquad s.t. \quad y_i = \mathbf{w}^T\mathbf{x}_i.$$

From straightforward calculations,

$$\begin{aligned}
(\tilde{m_1} - \tilde{m_2})^2 &= \big(\frac{1}{N_1}\sum_{y_j \in Y_1} y_j - \frac{1}{N_2}\sum_{y_j \in Y_2} y_j\big)^2 \\
&= \big(\frac{1}{N_1}\sum_{\mathbf{x}_j \in X_1} \mathbf{w}^T\mathbf{x}_j - \frac{1}{N_2}\sum_{\mathbf{x}_j \in X_2} \mathbf{w}^T\mathbf{x}_j\big)^2 \\
&= (\mathbf{w}^T\mathbf{m}_1 - \mathbf{w}^T\mathbf{m}_2)^2 \\
&= (\mathbf{w}^T(\mathbf{m}_1 - \mathbf{m}_2))^2 \\
&= \mathbf{w}^T(\mathbf{m}_1 - \mathbf{m}_2)\mathbf{w}^T(\mathbf{m}_1 - \mathbf{m}_2) \\
&= \mathbf{w}^T(\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^T\mathbf{w} \\
&= \mathbf{w}^T\mathbf{S}_b\mathbf{w}.
\end{aligned}$$

Similarly,

$$\begin{aligned}
\tilde{S_1} + \tilde{S_2} &= \sum_{y_j \in Y_1}(y_j - \tilde{m_1})^2 + \sum_{y_j \in Y_2}(y_j - \tilde{m_2})^2 \\
&= \sum_{\mathbf{x}_j \in X_1}(\mathbf{w}^T\mathbf{x}_j - \frac{1}{N_1}\sum_{\mathbf{x}_k \in X_1}\mathbf{w}^T\mathbf{x}_k)^2 + \sum_{\mathbf{x}_j \in X_2}(\mathbf{w}^T\mathbf{x}_j - \frac{1}{N_2}\sum_{\mathbf{x}_k \in X_2}\mathbf{w}^T\mathbf{x}_k)^2 \\
&= \sum_{\mathbf{x}_j \in X_1}(\mathbf{w}^T\mathbf{x}_j - \mathbf{w}^T\mathbf{m}_1)^2 + \sum_{\mathbf{x}_j \in X_2}(\mathbf{w}^T\mathbf{x}_j - \mathbf{w}^T\mathbf{m}_2)^2 \\
&= \sum_{\mathbf{x}_j \in X_1}(\mathbf{w}^T(\mathbf{x}_j - \mathbf{m}_1))^2 + \sum_{\mathbf{x}_j \in X_2}(\mathbf{w}^T(\mathbf{x}_j - \mathbf{m}_2))^2 \\
&= \sum_{\mathbf{x}_j \in X_1}\mathbf{w}^T(\mathbf{x}_j - \mathbf{m}_1)(\mathbf{x}_j - \mathbf{m}_1)^T\mathbf{w} + \sum_{\mathbf{x}_j \in X_2}\mathbf{w}^T(\mathbf{x}_j - \mathbf{m}_2)(\mathbf{x}_j - \mathbf{m}_2)^T\mathbf{w} \\
&= \mathbf{w}^T\mathbf{S}_1\mathbf{w} + \mathbf{w}^T\mathbf{S}_2\mathbf{w} \\
&= \mathbf{w}^T\mathbf{S}_w\mathbf{w}.
\end{aligned}$$

Therefore, we can rewrite the FIsher's criterion as

$$\max \quad J_F(\mathbf{w}) = \frac{\mathbf{w}^T\mathbf{S}_b\mathbf{w}}{\mathbf{w}^T\mathbf{S}_w\mathbf{w}}.$$

*The within-class scatter is slightly different from the formula given in the lecture slide. Since y_j and $\tilde{m_i}$ are both scalars not vectors, I think maybe we don't need to write $\tilde{S_i}$ as $\sum_{y_j \in Y_i}(y_i - \tilde{m_i})(y_i - \tilde{m_i})^T$.

Since we note that the norm of $\mathbf{w}$ doesn't affect the quotient $J_F(\mathbf{w})$, we can just fix the denominator $\mathbf{w}^T\mathbf{S}_w\mathbf{w} = c > 0$ and try to maximum $\mathbf{w}^T\mathbf{S}_b\mathbf{w}$, i.e., we want to solve the following problem

$$\max \quad \mathbf{w}^T\mathbf{S}_b\mathbf{w} \qquad s.t. \quad \mathbf{w}^T\mathbf{S}_w\mathbf{w} = c.$$

The method of Lagrange multipliers can be applied to solve this problem. It is easy to construct the Lagrange function

$$L(\mathbf{w}, \lambda) = \mathbf{w}^T\mathbf{S}_b\mathbf{w} - \lambda(\mathbf{w}^T\mathbf{S}_w\mathbf{w} - c).$$

The necessary condition for solving this problem is

$$\frac{\partial L}{\partial \mathbf{w}} = 2\mathbf{S}_b\mathbf{w} - 2\lambda\mathbf{S}_w\mathbf{w} = 0.$$

Hence, we have

$$\mathbf{S}_b\mathbf{w} = \lambda\mathbf{S}_w\mathbf{w}$$

If we assume that $\mathbf{S}_w$ is nonsingular, then we have

$$\mathbf{S}_w^{-1}\mathbf{S}_b\mathbf{w} = \lambda\mathbf{w},$$

and it is clear that $\mathbf{w}$ is just the eigenvector of matrix $\mathbf{S}_w^{-1}\mathbf{S}_b$.

Since we only care about the direction of $\mathbf{w}$, we don't need to calculate the eigenvalue or the eigenvector seriously. If we substitute $\mathbf{S}_b$ into the equation above and denote $R = (\mathbf{m}_1 - \mathbf{m}_2)^T\mathbf{w}$, we have

$$\lambda\mathbf{w} = \mathbf{S}_w^{-1}(\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^T\mathbf{w} \doteq \mathbf{S}_w^{-1}(\mathbf{m}_1 - \mathbf{m}_2)R.$$

Note that R is only a real number and we only care about the direction of optimal $\mathbf{w}$. Based on the equation above, we can just know the direction of the optimal $\mathbf{w}$, and we can choose this $\mathbf{w}$ to be our optimal weight vector,

$$\mathbf{w} = \mathbf{S}_w^{-1}(\mathbf{m}_1 - \mathbf{m}_2).$$

Based on the references, below is a brief description about the principle of Lagrange multipliers.

The method of Lagrange multipliers is mainly used to find the local maximum or minimum of a function subject to one or more equality constraints. Assume that $f(x_1, x_2, \cdots, x_n)$ is the function we try to optimize, and there are M constraints satisfying $g_i(\mathbf{x}) = 0, i = 1, \cdots, M$, then we can introduce the Lagrangian

$$L(x_1, \cdots, x_n, \lambda_1, \cdots, \lambda_M) = f(x_1, \cdots, x_n) - \sum_{k=1}^{M} \lambda_k g_k(x_1, x_2, \cdots, x_n),$$

and then we can solve the equation below to gain the possible optimal point

$$\nabla_{x_1, x_2, \cdots, x_n, \lambda_1, \lambda_2, \cdots, \lambda_M} L(x_1, \cdots, x_n, \lambda_1, \cdots, \lambda_M) = \mathbf{0}.$$

We can note that as for the equations above, there are $n + M$ unknowns and exactly $n + M$ different equations. We also need to note that the equations are only the necessary conditions for the optimization problem, we may also need to check the Hessian Matrix to further evaluate the necessary condition of the optimization problem.

3. (Optional) Note: The answer is with reference to the websites below:

- ADALINE - Wikipedia
- Machine Learning FAQ
- Microsoft PowerPoint - lecture5-adaline.pptx

Both the perceptron algorithm and ADALINE are the oldest and the simplest methods to classify the samples. And ADALINE is usually regarded as an improvement of the perceptron.

Generally speaking, the two algorithms deal with the same kind of problems: they are both mainly used to deal with the binary classification problems, they both give a linear decision boundary and can be implemented sample by sample iteratively, and they are both instances of Error Correction Learning.

However, there are also some differences between these two algorithms. Let's have a look at the iterative formula given by these two algorithms. The sample by sample version iterative formula given by perception is

$$\alpha(k+1) = \alpha(k) + \rho_k \mathbf{y}^k,$$

but the iterative formula given by ADALINE is

$$\alpha(k+1) = \alpha(k) + \rho_k (b_k - \alpha(k)^T \mathbf{y}^k)\mathbf{y}^k.$$

The formula above means that ADALINE uses the continuous predicted value to help itself to update the augmented weight vector. For each misclassified point, the update will be weighted according to the difference between the target value(b_k) and a continuous predicted value($\alpha(k)^T \mathbf{y}^k$).

From the formulas above, we can also know that compared to the perceptron, the ADALINE allow abritrary real values for the output data. In addition, the ADALINE always converges to the minimum squared error, but the perceptron only converges under the linear separable cases.

4. It's not a good idea to take the test.

   We assume that a positive test result usually means the person carries the disease, and a negative result usually indicates the person doesn't carry the disease. We denote event A to be the test shows a positive result, and event B to be the person actually carries the disease.

   Since the sensitivity is 100% and the specificity is 99.99%, then we have $P(A|B) = 100\%, P(A|\bar{B}) = 1 - 99.99\% = 0.01\%$. We also know that only one in a million people carry the disease, then we know the prior probability $P(B) = \frac{1}{10^6}$.

   From the Bayes' theorm, we can calculate

$$\begin{aligned} P(B|A) = \frac{P(AB)}{P(A)} &= \frac{P(B)P(A|B)}{P(B)P(A|B) + P(\bar{B})P(A|\bar{B})} \\ &= \frac{\frac{1}{10^6}100\%}{\frac{1}{10^6}100\% + (1 - \frac{1}{10^6})0.01\%} \\ &\approx 0.99\%. \end{aligned}$$

   The result indicates that even if the test shows a positive result, the possibility that this person really carries the disease is only 0.99%, which will be quite misleading. Therefore, this test should not be taken.