The Influence of Polynomial Order in Logistic Regression on Decision Boundary

The Influence of Polynomial Order in Logistic Regression on Decision Boundary

Xing Wan

Leshan Vocational and Technical College, Leshan, Sichuan, 614000, China

krantson@163.com

Abstract. In machine learning problems, polynomial logistic regression algorithms are often used to classify data. Compared to linear regression, polynomial regression can not only deal with linear problems, but also deal with nonlinear problems. In the polynomial logistic regression algorithm, the polynomial order has a certain influence on the classification effect. This paper studies the influence of the polynomial order on the binary decision boundary in binary classification problem. By choosing different parameter values, an approximate optimal solution can be found.

1. Introduction

Classification problems are fundamental problems in machine learning, and evaluator used for classification are often referred to as classifiers. Classifier uses training data for training or learning, called classifier construction. The trained classifier is used to predict whether a sample belongs to a category. Logistic regression is often used to solve the binary classification problem, that is, there are only two types of sample labels, which are called positive and negative examples. Usually the positive example is represented by 1, and the negative example is represented by 0. Since linear regression is used to obtain a straight line, the output is a continuous value with a large range, so it is not suitable for solving the classification problem, polynomial logistic regression is more suitable[2][3].

2. Hypothetical function

For the two-class problem, assume that the positive case is 1 and the negative case is 0. The hypothesis function of the classification problem must satisfy its predicted value between 0 and 1, and the sigmod function[2] satisfies this property well. The sigmod function, also known as the S-type function or the Logistic function, is formulated as:

$$\mathrm{g(z)} = \frac{1}{1+e^{-z}} \tag{1}$$

Among them, it is often written as exp(-z). When the S-type function is a good approximation to the step function. When the input is greater than zero, the output approaches 1; when the input is less than zero, the output approaches 0. When the input is 0, the output is exactly 0.5. The S-type function simulates the step function very well, the difference is that it is continuous and smooth, strictly monotonically increasing, and has following property:

$$g(z)' = g(z)(1 - g(z)) \tag{2}$$

Logistic regression uses s-type functions, giving function g(z) with linear regression expression θ^Tx transforming and squeezing the value of the function between 0 and 1. For input binary input variables, the logistic regression model is:

$$h(x;\theta) = g(\theta^T x) = g(w_0 + w_1x_1 + w_2x_2) \tag{3}$$

Decision boundaries can help understand the effects of the results of the logistic regression hypothesis function:

- When $\theta^{Tx} \geq 0$, the forecast sample is a positive example
- When $\theta^{Tx} < 0$, the forecast sample is a negative example

Due to$w_0 + w_1x_1 + w_2x_2$, it is a straight line that divides the data into two part. The point above the line becomes a positive sample, and the point below the line becomes a negative sample. For the iris dataset, the irises are divided into Setosa and Versicolor according to petal length and petal width.

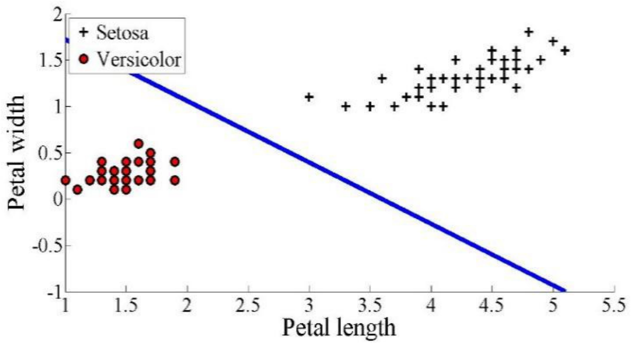

Figure 1. Classification of irises

For more complex data distributions, you can use more complex models with polynomial regression as shown below:

$$h(x;\theta) = g(\theta^T x) = g(w_0 + w_1x_1 + w_2x_2 + w_3x_1^2 + w_4x_2^2 + w_5x_1x_2 + \cdots) \tag{4}$$

using this model, the decision boundary becomes a curve, and the shape is more complicated.

The shape of the decision boundary is determined by the data. If it is one-dimensional data, the decision boundary degenerates into a point; if it is two-dimensional data, the decision boundary is a line or curve; if it is three-dimensional data, the decision boundary is a plane or a surface; if it is higher-dimensional data, the decision boundary is a hyperplane.

Logistic regression algorithm is to find a decision boundary by learning, which can separate different types of data, and has certain generalization ability.

3. Logistic regression algorithm

3.1. Linear logistic regression

The key problem of logistic regression algorithm is to find $\boldsymbol{\theta}$, so the logistic regression algorithm revolves around optimizing the parameters $\boldsymbol{\theta}$. First, it needs to define a cost function $J(\boldsymbol{\theta})$, there is a goal of the parameter $\boldsymbol{\theta}$ optimization, and finding the lowest cost function $\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} J(\boldsymbol{\theta})$. In the logistic regression, the misclassified cost is expressed using a negative log-likelihood cost function, defined as:

$$\mathrm{cost}(h(\mathbf{x};\boldsymbol{\theta}),\mathbf{y}) = \begin{cases} -\log(h(\mathbf{x};\boldsymbol{\theta})) \\ -\log(1 - h(\mathbf{x};\boldsymbol{\theta}) \end{cases} \tag{5}$$

among them, log represents the natural logarithm, $\mathrm{cost}(h(x;\theta))$ has a feature that if the real label is 1 and the hypothesis function $h(x;\theta)$ is also equal to 1, then the cost function is 0. the cost increases when $h(x;\theta)$ decreases. When the true label is 0, if the assumed function is also 0, the cost become 0, otherwise the cost increases as the $\mathrm{h}(\mathrm{x};\theta)$ increases. There are only 2 values, so the formula can be written as:

$$\mathrm{cost}(\mathrm{h}(\mathbf{x};\boldsymbol{\theta}),\mathbf{y}) = -\mathrm{y}\log\big(\mathrm{h}(\mathbf{x};\boldsymbol{\theta})\big) - (1-\mathrm{y})\log(1-\mathrm{h}(\mathbf{x};\boldsymbol{\theta})) \tag{6}$$

Cost function $\mathrm{J}(\theta)$ is N. The mean of the costs of the samples, expressed as follows:

$$\mathrm{J}(\boldsymbol{\theta}) = -\frac{1}{N}\left[\sum_{j=1}^{N} y^{(j)}\log\big(h\big(\boldsymbol{x}^{(j)};\boldsymbol{\theta}\big) + (1-y^{(j)})\big)\right] \tag{7}$$

If using regularization, limiting the range of values of the parameters is necessary. With punishing the parameters with larger values, function $\mathrm{J}(\theta)$ can be expressed as:

$$\mathrm{J}(\boldsymbol{\theta}) = -\frac{1}{N}\left[\sum_{\boldsymbol{j}=1}^{N} y^{(j)}\log\Big(\boldsymbol{h}\big(\boldsymbol{x}^{(j)};\boldsymbol{\theta}\big) + \big(1-\mathrm{y}^{(\mathrm{j})}\big)\Big)\right] + \frac{1}{\lambda}\sum_{\boldsymbol{i}=\boldsymbol{1}}^{D} w_i^2 \tag{8}$$

Logistic regression cannot directly obtain optimal solution of a parameter set using normal equations like linear regression. It can only be optimized using the gradient descent algorithm. The derivatives are as follows:

$$\frac{\partial}{\partial\theta_i}J(\boldsymbol{\theta}) = \frac{1}{N}\sum_{j=1}^{N}\big(h\big(\boldsymbol{x}^{(j)};\boldsymbol{\theta}\big) - y^j\big)x_i^{(j)} \tag{9}$$

substituting the update formula $\theta_\mathrm{i} = \theta_i - \alpha\frac{\partial}{\partial\theta_i}J(\theta)$ with $\frac{\partial}{\partial\theta_i}J(\boldsymbol{\theta})$:

$$\theta_i = \theta_i - \alpha\,\frac{1}{N}\sum_{j=1}^{N}\big(h\big(\boldsymbol{x}^{(j)};\boldsymbol{\theta}\big) - y^j\big)x_i^{(j)} \tag{10}$$

On the surface, the logistic regression gradient descent algorithm is the same as the linear regression gradient descent algorithm. Because the former is a linear output and the latter is a nonlinear output, both usually require feature scaling to speed up the convergence.

3.2. Polynomial logistic regression

If the decision boundary is very complex and you can't separate the different categories with a single line, it is necessary to perform a polynomial transformation on the original data, adding the higher order items, and then using the regularized logistic regression method.

$$\mathrm{h}(\mathbf{x};\boldsymbol{\theta}) = \mathrm{g}(\mathrm{w}_0 + \mathrm{w}_1\mathrm{x}_1 + \mathrm{w}_2\mathrm{x}_2 + \mathrm{w}_3\mathrm{x}_1^2 + \mathrm{w}_4\mathrm{x}_2^2 + \mathrm{w}_5\mathrm{x}_1\mathrm{x}_2 + \cdots) \tag{11}$$

Adding a regularization expression to the original cost function to get a new cost function:

$$\mathrm{J}(\boldsymbol{\theta}) = -\frac{1}{N}\left[\sum_{\boldsymbol{j}=1}^{\boldsymbol{N}} y^{(j)}\log\big(h\big(\boldsymbol{x}^{(j)};\boldsymbol{\theta}\big) + \big(1-\mathrm{y}^{(\mathrm{j})}\big)\log(1-h(\boldsymbol{x}^{(j)};\boldsymbol{\theta}))\right] + \frac{\lambda}{2\mathrm{N}}\sum_{i=1}^{D}\theta_i^2 \tag{12}$$

Finding the derivative of cost function $J(\boldsymbol{\theta})$, getting the following results:

$$\frac{\partial}{\partial\theta_i}J(\boldsymbol{\theta}) = \frac{1}{N}\sum_{1}^{N}\big(h\big(x^j;\boldsymbol{\theta}\big) - y^j\big)x_i^j + \frac{\lambda}{N}\theta_i \tag{13}$$

Bringing the results to the update formula $\boldsymbol{\theta} - \alpha\frac{\partial}{\partial\theta_i}J(\boldsymbol{\theta})$, the following results are obtained:

$$\theta_i = \theta_i - \alpha\,\frac{1}{N}\sum_{\boldsymbol{j}=\boldsymbol{1}}^{\boldsymbol{N}}\big(h\big(\boldsymbol{x}^{(j)};\boldsymbol{\theta}\big) - y^{(i)}\big)\boldsymbol{x}_{\boldsymbol{i}}^{(\boldsymbol{j})} + \frac{\lambda}{N}\theta_i \tag{14}$$

The update algorithm is as follows:

- Set a number of iterations
- Set a initial $\boldsymbol{\theta}$
- Set a initial λ
- Compute cost function
- Update every θ_i
- If completing iteration, optimal $\boldsymbol{\theta}$ is obtained.

When $\boldsymbol{\theta}$ is obtained, adjusting parameter value of λ can find a another optimal λ. Try a few more times, an approximate optimal value can be obtained.

4. The influence of polynomial order on decision boundary

In polynomial logistic regression, the polynomial order has a certain influence on the regression performance. If the decision boundary is more complicated, a higher order polynomial should be used, but the polynomial frequency is too high and the over-fitting phenomenon will occur. Therefore, there must be a good balance between overfitting and training error. Boundary curve by high-order polynomial logistic regression is obtained as follows:

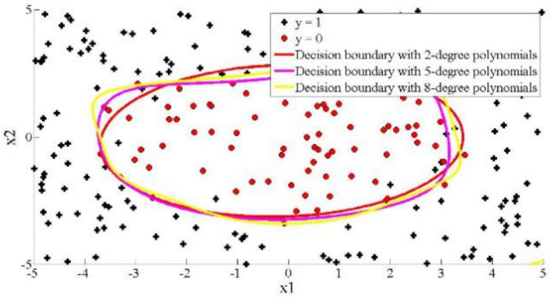

Figure 2. Effect of polynomial order on decision boundary

For the more complex data distribution, you can use a more complex model like the polynomial regression in linear regression. The following is shown in Fig.3: When the highest order of the polynomial is 8, it appears Over-fitting phenomenon; When the highest order of the polynomial order is 2, the under-fitting phenomenon occurs, and when the highest order of the polynomial is 5, the fitting effect is better.

5. Conclusion

Based on the results and discussions presented above, the conclusions are obtained as below:

(1) Polynomial logistic regression is usually than linear logistic regression

(2) Polynomial logistic regression can better classify data

(3) Polynomial logistic regression is more suitable for nonlinear problems

(4) The order of the polynomial is too high to easily produce over-fitting, too low to easily produce under-fitting, and the trade-off between training error and over-fitting must be made.

Acknowledgments

This work was financially supported by the Applied Research Project of Education Department of Sichuan Province (18ZB0280).

References

[1] Van der Geer, J., Hanraads, J.A.J., Lupton, R.A. (2010) The art of writing a scientific article. J. Sci. Commun, 163: 51–59.
[2] Goodfellow, I., Bengio, Y., Courville, A. (2017) Deep Learning. MIT Press, London.
[3] Bishop, C. M. (2006) Pattern Recognition and Machine Learning. Spring, NewYork, NY.
[4] Carbonell, J.G. (1990) Machine Learning: Paradigms and Methods. MIT Press.
[5] Dietterich, T.G. and Bakiri, G. (1995) Solving multiclass learning problems via error-correcting ouput codes. Journal of Artificial Intelligence Research, 2:263-286.
[6] Podani, J. (1994) Multivariate Data Analysis in Ecology and Systematics. SPB Publishing, The Hague.