Lecture Notes on Logistic Regression

Feng Li
fli@sdu.edu.cn
Shandong University, China

1 Introduction

We hereby look at classification problems. Compared with regression models where the target values is continuous, we predict only a small number of discrete values in classification models. Given a feature vector x, we aim at categorizing it into one of the discrete classes. The output variable y is so-called *label* for a given input feature vector x. Specifically, for a binary classification problem, we have $y \in \{0, 1\}$, where 0 represents negative class, while 1 denotes positive class.

We first introduce the logistic function before diving into studying logistic regression. The logistic function (or sigmoid function)

$$g(z) = \frac{1}{1 + \exp(-z)} \tag{1}$$

is used to continuously approximate discrete classification. Fig. 1 shows the curve of the logistic function. The logistic function has the following properties

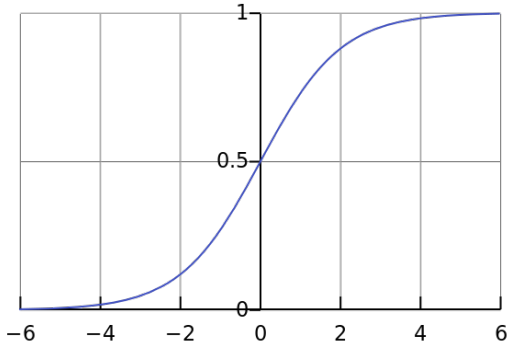

Figure 1: The curve of the logistic function.

which are very useful for our logistic regression modeling.

- Bound: $g(z) \in (0, 1)$
- Symmetric: $1 - g(z) = g(-z)$
- Gradient: $g'(z) = g(z)(1 - g(z))$

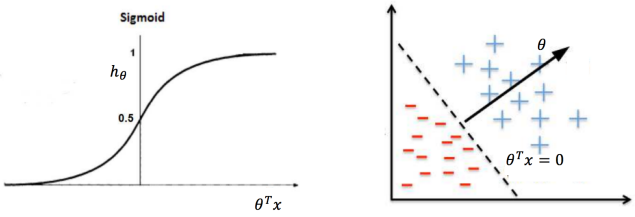

Figure 2: Decision boundary.

The hypothesis function for the logistic regression is defined based on the logistic function

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + \exp(-\theta^T x)} \tag{2}$$

where $\theta^T x$ is a linear combination of the features weighted by θ (similar with linear regression model). Intuitively, the hypothesis function first computes a real-valued "score" ($\theta^T x$) for input x and then "squashes" it between $(0, 1)$ to turn this score into a probability (of x's label being 1). Therefore, we have the following probabilities

$$\Pr(y = 1 \mid x; \theta) = h_\theta(x) = 1/(1 + \exp(-\theta^T x))$$
$$\Pr(y = 0 \mid x; \theta) = 1 - h_\theta(x) = 1/(1 + \exp(\theta^T x))$$

Instead of categorizing x into a certain class, we actually compute the probability that x belongs to some class. Also, we can say, $y = 1$ if $\Pr(y = 1 \mid x; \theta) \geq 0.5$, and $y = 0$ otherwise.

One question is that what is the underlying decision rule in the logistic regression? In another word, what is the decision boundary that can be used to differentiate positive data samples from negative ones. Since both classes are equiprobable, we have $\Pr(y = 1 \mid x; \theta) = \Pr(y = 0 \mid x; \theta)$, which is equivalently to say $1 + \exp(-\theta^T x) = 1 + \exp(\theta^T x)$. Solving the above equation gives $\theta^T x = 0$. It is very interesting to observe that the decision boundary of the logistic regression is nothing but a hyperplane. In particular, $y = 1$ if $\theta^T x \geq 0$; otherwise, $y = 0$.

A example of the decision boundary is given in Fig. 1. In fact, the "score" $\theta^T x$ is also a measure of distance of x from the hyperplane: the score is positive for pos. examples, and negative for neg. examples. In particular, for a given input feature vector, higher positive scores correspond with higher probabilities of label 1, while higher negative scores imply lower probabilities of label 1 (and thus higher probabilities of label 0).

2 Formulating Logistic Regression Models

Assume $\Pr(y = 1 \mid x; \theta) = h_\theta(x)$ and $\Pr(y = 0 \mid x; \theta) = 1 - h_\theta(x)$, then we have

$$\Pr(y \mid x; \theta) = (h_\theta(x))^y (1 - h_\theta(x))^{1-y}$$

Assuming the training examples were generated independently, we define the likelihood of the parameters as

$$\begin{aligned} L(\theta) &= \Pr(Y \mid X; \theta) \\ &= \prod_i \Pr(y^{(i)} \mid x^{(i)}; \theta) \\ &= \prod_i (h_\theta(x^{(i)}))^{y^{(i)}} (1 - h_\theta(x^{(i)}))^{1-y^{(i)}} \end{aligned}$$

To facilitate our computation, we maximize the following log likelihood $\ell(\theta)$ instead of $L(\theta)$

$$\ell(\theta) = \log L(\theta) = \sum_{i=1}^{m} \Big(y^{(i)} \log h(x^{(i)}) + (1 - y^{(i)}) \log(1 - h(x^{(i)}) \Big) \tag{3}$$

One approach is gradient ascent, where we iteratively update θ according to the following rule until convergence condition is respected

$$\theta \leftarrow \theta + \nabla \ell(\theta) \tag{4}$$

For each data example (x, y), since

$$\frac{\partial}{\partial \theta_j} h_\theta(x) = \frac{\exp(-\theta^T x)}{(1 + \exp(-\theta^T x))^2} \cdot x_j$$

a careful derivation leads us to

$$\frac{\partial}{\partial \theta_j} \ell(\theta) = \sum_{i=1}^{m} \Big(y^{(i)} - h_\theta(x^{(i)}) \Big) x_j^{(i)}$$

3 Newton's Method

We hereby talk about a new optimization algorithm, so-called *Newton's method*. An introductive question is that, given a differentiable real-valued function $f : \mathbb{R} \to \mathbb{R}$, how can we find x such that $f(x) = 0$? One approach is to iteratively update x according to the following rule

$$x \leftarrow x - \frac{f(x)}{f'(x)}$$

until a sufficient accurate value is achieved (i.e., $f(x) \approx 0$). Fig. 3 demonstrates the iterative process of the Newton's method. Suppose the value of x in the t-the iteration is $x^{(t)}$, and $y^{(t)} = f(x^{(t)})$. We construct a tangent line at $(x^{(t)}, y^{(t)})$, which intersects the x-axis at $x^{(t+1)}$. Let $y^{(t+1)} = f(x^{(t+1)})$, and we continue to construct a tangent line at $(x^{(t+1)}, y^{(t+1)})$. We then approach the intersection point between f and the x-axis step by step.

To maximize $f(x)$, we have to find the stationary point of $f(x)$ such that $f'(x) = 0$. According to Newton's method, we have the following update

$$x \leftarrow x - \frac{f'(x)}{f''(x)} \tag{5}$$

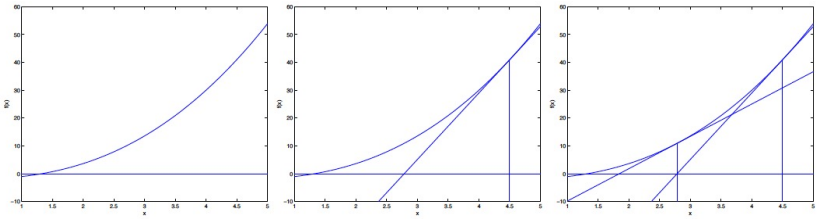

Figure 3: Newton's method.

For $\ell : \mathbb{R}^n \to \mathbb{R}$, we generalization Newton's method to the multidimensional setting

$$\theta \leftarrow \theta - H^{-1} \bigtriangledown_\theta \ell(\theta) \tag{6}$$

where H is the Hessian matrix

$$H_{i,j} = \frac{\partial^2 \ell(\theta)}{\partial \theta_i \partial \theta_j} \tag{7}$$

The Newton's has quadratic convergence rate which is much higher than gradient descent. Therefore, it takes fewer iterations to approach the minimum. However, in the Newton's method, each iteration is more expensive than the one of gradient descent, as it needs to calculating and invert an $n \times n$ Hessian

4 Multiclass Classification

We discuss logistic regression which focuses on binary classification in the previous section. We now look at multiclass classification which aims at classifying instances into one of the more than two classes.

It is apparent that the hypothesis function is parameterized by θ. Since our goal is to make predictions according to the hypothesis function given a new test data, we need to find the optimal value of θ such that the resulting prediction is as accurate as possible. Such a procedure is so-called *training*. The training procedure is performed based on a given set of m training data $\{x^{(i)}, y^{(i)}\}_{i=1,\cdots,m}$. In particular, we are supposed to find a hypothesis function (parameterized by θ) which fits the training data as closely as possible. To measure the error between h_θ and the training data, we define a *cost function* (also called *error function*) $J(\theta) : \mathbb{R}^{n+1} \to \mathbb{R}$ as follows

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

Our linear regression problem can be formulated as

$$\min_\theta J(\theta) = \frac{1}{2} \sum_{i=1}^{m} \left(\theta^T x^{(i)} - y^{(i)} \right)^2$$

We give an illustration in Fig. **??** to explain linear regression in 3D space (i.e., $n = 2$). In the 3D space, the hypothesis function is represented by a *hyperplane*. The red points denote the training data, and it is shown that, $J(\theta)$ is the sum of the differences between the target values of the training data and the hyperplane.

5 Gradient Descent

Gradient Descent (GD) method is a first-order iterative optimization algorithm for finding the minimum of a function. If the multi-variable function $J(\theta)$ is differentiable in a neighborhood of a point θ, then $J(\theta)$ decreases fastest if one goes from θ in the direction of the negative gradient of J at θ. Let

$$\nabla J(\theta) = [\frac{\partial J}{\partial \theta_0}, \frac{\partial J}{\partial \theta_1}, \cdots, \frac{\partial J}{\partial \theta_n}]^T \tag{8}$$

denote the gradient of $J(\theta)$. In each iteration, we update θ according to the following rule:

$$\theta \leftarrow \theta - \alpha \nabla J(\theta) \tag{9}$$

where α is a step size. In more details,

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial J(\theta)}{\partial \theta_j} \tag{10}$$

The update is terminated when convergence is achieved. In our linear regression model, the gradient can be calculated as

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{\partial}{\partial \theta_j} \frac{1}{2} \sum_{i=1}^{m} (\theta^T x^{(i)} - y^{(i)})^2 = \sum_{i=1}^{m} (\theta^T x^{(i)} - y^{(i)}) x_j^{(i)} \tag{11}$$

We summarize the GD method as follows in Algorithm 1. The algorithm

Algorithm 1: Gradient Descent

Given a starting point $\theta \in$ **dom** J
repeat
1. Calculate gradient $\nabla J(\theta)$;
2. Update $\theta \leftarrow \theta - \alpha \nabla J(\theta)$

until convergence criterion is satisfied

usually starts with a randomly initialized θ. In each iteration, we update θ such that the objective function is decreased monotonically. The algorithm is said to be converged when the difference of J in successive iterations is less than (or equal to) a predefined threshold (say ε). Assuming $\theta^{(t)}$ and $\theta^{(t+1)}$ are the values of θ in the t-th iteration and the $(t+1)$-th iteration, respectively, the algorithm is converged when

$$|J(\theta^{(t+1)}) - J(\theta^{(t)})| \leq \varepsilon \tag{12}$$

Another convergence criterion is to set a fixed value for the maximum number of iterations, such that the algorithm is terminated after the number of the iterations exceeds the threshold. We illustrate how the algorithm converges iteratively in Fig. **??**. The colored contours represent the objective function, and the GD algorithm converges into the minimum step-by-step.

The choice of the step size α actually has a very important influence on the convergence of the GD algorithm. We illustrate the convergence processes under different step sizes in Fig. 3.

6 Stochastic Gradient Descent

According to Eq. 11, it is observed that we have to visit all training data in each iteration. Therefore, the induced cost is considerable especially when the training data are of big size.

Stochastic Gradient Descent (SGD), also known as incremental gradient descent, is a stochastic approximation of the gradient descent optimization method. In each iteration, the parameters are updated according to the gradient of the error (i.e., the cost function) with respect to one training sample only. Hence, it entails very limited cost.

We summarize the SGD method in Algorithm 2. In each iteration, we first randomly shuffle the training data, and then choose only one training example to calculate the gradient (i.e., $\nabla J(\theta; x^{(i)}, y^{(i)})$) to update θ. In our linear regression model, $\nabla J(\theta; x^{(i)}, y^{(i)})$ is defined as

$$\nabla J(\theta; x^{(i)}, y^{(i)}) = (\theta^T x^{(i)} - y^{(i)}) x^{(i)} \tag{13}$$

and the update rule is

$$\theta_j \leftarrow \theta_j - \alpha(\theta^T x^{(i)} - y^{(i)}) x_j^{(i)} \tag{14}$$

Algorithm 2: Stochastic Gradient Descent for Linear Regression

1: **Given** a starting point $\theta \in$ **dom** J
2: **repeat**
3: Randomly shuffle the training data;
4: **for** $i = 1, 2, \cdots, m$ **do**
5: $\theta \leftarrow \theta - \alpha \nabla J(\theta; x^{(i)}, y^{(i)})$
6: **end for**
7: **until** convergence criterion is satisfied

Compared with GD where the objective cost function is decreased for each step, SGD does not have such a guarantee. In fact, SGD entails more steps to converge, but each step is cheaper. One variants of SGD is so-called *mini-batch* SGD, where we pick up a small group of training data and do average to accelerate and smoothen the convergence. For example, by randomly choosing k training data, we can calculate the average the gradient

$$\frac{1}{k} \sum_{i=1}^{k} \nabla J(\theta; x^{(i)}, y^{(i)}) \tag{15}$$

7 A Closed-Form Solution to Linear Regression

We first look at the vector form of the linear regression model. Assume

$$X = \begin{bmatrix} (x^{(1)})^T \\ \vdots \\ (x^{(m)})^T \end{bmatrix} \quad Y = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix} \tag{16}$$

Therefore, we have

$$X\theta - Y = \begin{bmatrix} (x^{(1)})^T\theta \\ \vdots \\ x^{(m)})^T\theta \end{bmatrix} - \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix} = \begin{bmatrix} h_\theta(x^{(1)}) - y^{(1)} \\ \vdots \\ h_\theta(x^{(m)}) - y^{(m)} \end{bmatrix}$$

Then, the cost function $J(\theta)$ can be redefined as

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}) = \frac{1}{2}(X\theta - Y)^T(X\theta - Y) \tag{17}$$

To minimize the cost function, we calculate its derivative and let it be zero

$$\begin{aligned} \nabla_\theta J(\theta) &= \nabla_\theta \frac{1}{2}(Y - X\theta)^T(Y - X\theta) \\ &= \frac{1}{2}\nabla_\theta(Y^T - \theta^T X^T)(Y - X\theta) \\ &= \frac{1}{2}\nabla_\theta \mathrm{tr}(Y^TY - Y^TX\theta - \theta^TX^TY + \theta^TX^TX\theta) \\ &= \frac{1}{2}\nabla_\theta \mathrm{tr}(\theta^TX^TX\theta) - X^TY \\ &= \frac{1}{2}(X^TX\theta + X^TX\theta) - X^TY \\ &= X^TX\theta - X^TY \end{aligned}$$

Since $X^TX\theta - X^TY = 0$, we have $\theta = (X^TX)^{-1}X^TY$. Note that the inverse of X^TX does not always exist. In fact, the matrix X^TX is invertible if and only if the columns of X are linearly independent.

8 A Probabilistic Interpretation

An interesting question is why the least square form of the linear regression model is reasonable. We hereby give a probabilistic interpretation. We suppose a target value y are sampled from a "line" θ^Tx with certain noise. Therefore, we have

$$y^{(i)} = x^{(i)} + \varepsilon^{(i)}$$

where $\varepsilon^{(i)}$ denote the noise and is independently and identically distributed (i.i.d.) according to a Gaussian distribution $\mathcal{N}(0, \sigma^2)$. The density of $\varepsilon^{(i)}$ is given by

$$f(\epsilon^{(i)}) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(\epsilon^{(i)})^2}{2\sigma^2}\right)$$

and we equivalently have the following conditional probability

$$\Pr(y^{(i)} \mid x^{(i)}; \theta) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(y^{(i)} - \theta^Tx^{(i)})^2}{2\sigma^2}\right)$$

Therefore, it is shown that the distribution of $y^{(i)}$ given $x^{(i)}$ is parameterized by θ, i.e., $y^{(i)} \mid x^{(i)}; \theta \sim \mathcal{N}(\theta^Tx^{(i)}, \sigma^2)$. Recalled that Y and X are the vector

of the target values and the matrix of the features (see Eq. (16)), and $Y = X\theta$. Considering $\varepsilon^{(i)}$'s are i.i.d., we define the following likelihood function

$$L(\theta) = \prod_i \Pr(y^{(i)} \mid x^{(i)}; \theta) = \prod_i \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right)$$

which denote the probability of the given target values in training data. To calculating the optimal θ such that the resulting linear regression model fits the given training data best, we need to maximize the likelihood $L(\theta)$. To simplify the computation, we use the log likelihood function instead, i.e.,

$$\begin{aligned}
\ell(\theta) &= \log L(\theta) \\
&= \log \prod_i^m \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right) \\
&= \sum_i^m \log \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right) \\
&= m \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{2\sigma^2} \sum_i (y^{(i)} - \theta^T x^{(i)})^2
\end{aligned}$$

Apparently, maximizing $L(\theta)$ is equivalent to minimizing

$$\frac{1}{2} \sum_i (y^{(i)} - \theta^T x^{(i)})^2$$