Lecture Notes on Linear Regression

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

1 Linear Regression Problem

In regression problem, we aim at predicting a continuous target value given an input feature vector. We assume a n-dimensional feature vector is denoted by $x \in \mathbb{R}^n$, while $y \in \mathbb{R}$ is the output variable. In linear regression models, the hypothesis function is defined by

$$h_\theta(x) = \theta_n x_n + \theta_{n-1} x_{n-1} + \cdots + \theta_1 x_1 + \theta_0$$

Geometrically, when $n = 1$, $h_\theta(x)$ is actually a line in a 2D plane, while $h_\theta(x)$ represents a plane in a 3D space when $n = 2$. Generally, when $n \geq 3$, $h_\theta(x)$ defines a so-called "*hyperplane*" in a higher dimensional space. Suppose

$$\theta = \begin{bmatrix} \theta_n \\ \theta_{n-1} \\ \vdots \\ \theta_1 \\ \theta_0 \end{bmatrix}, \quad \text{and } x = \begin{bmatrix} x_n \\ x_{n-1} \\ \vdots \\ x_1 \\ 1 \end{bmatrix}$$

the hypothesis function $h_\theta(x)$ can be re-written as

$$h_\theta(x) = \theta^T x \tag{1}$$

where $\theta \in \mathbb{R}^{n+1}$ is a parameter vector.

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

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

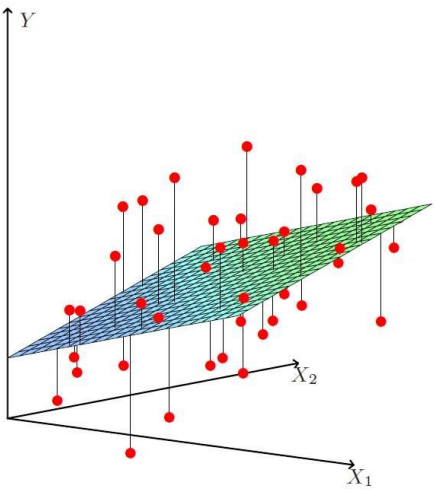

Figure 1: 3D linear regression.

Specifically, we aim at minimizing $J(\theta)$ over θ. We give an illustration in Fig. 1 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 the distance from the (read) training data to the hyperplane is denoted by $|\theta^T x^{(i)} - y^{(i)}|$.

2 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

$$\nabla J(\theta) = [\frac{\partial J}{\partial \theta_0}, \frac{\partial J}{\partial \theta_1}, \cdots, \frac{\partial J}{\partial \theta_n}]^T \tag{2}$$

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

$$\theta \leftarrow \theta - \alpha \nabla J(\theta) \tag{3}$$

where α is a step size. In more details,

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial J(\theta)}{\partial \theta_j} \tag{4}$$

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

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{\partial}{\partial \theta_j} \frac{1}{2} \sum_{i=1}^{m} (\theta^T x^{(i)} - y^{(i)})^2 = \sum_{i=1}^{m} (\theta^T x^{(i)} - y^{(i)}) x_j^{(i)} \tag{5}$$

We summarize the GD method in Algorithm 1. The algorithm usually starts

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

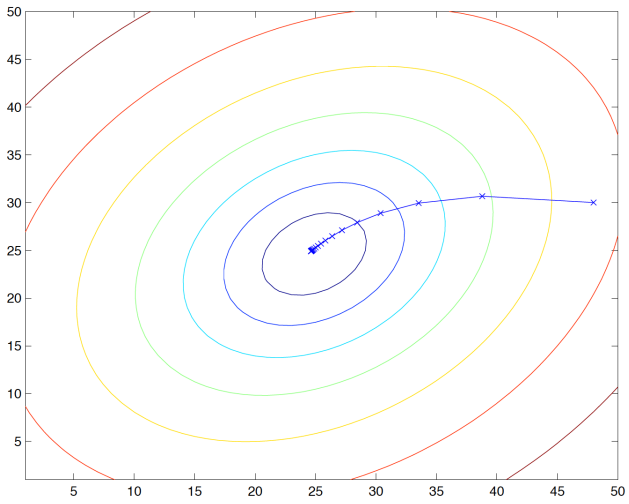

Figure 2: The convergence of GD algorithm.

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 gradient has its magnitude less than or equal to a predefined threshold (say ε), i.e.

$$\|\nabla f(x)\|_2 \leq \varepsilon$$

where $\|\cdot\|_2$ is ℓ_2 norm, such that the values of the objective function differ very slightly in successive iterations. Another convergence criterion is to set a fixed value for the maximum number of iterations, and the algorithm is terminated after the number of the iterations exceeds the threshold. We illustrate how the algorithm converges iteratively in Fig. 2. The colored contours represent the objective function, and 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.

3 Stochastic Gradient Descent

According to Eq. 5, 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.

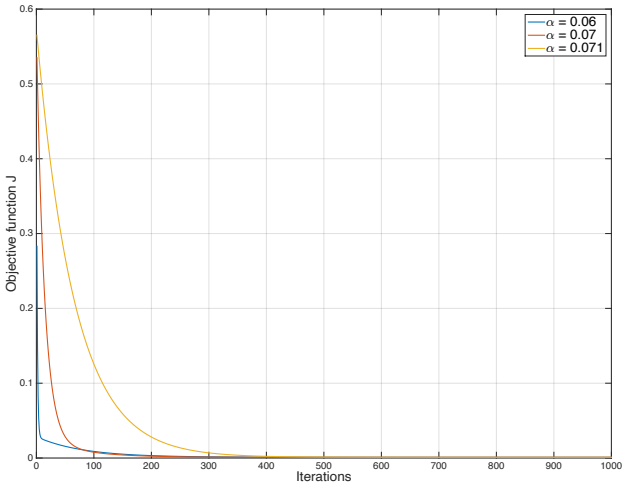

Figure 3: The convergence of GD algorithm under different step sizes.

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

$$\nabla J(\theta; x^{(i)}, y^{(i)}) = (\theta^T x^{(i)} - y^{(i)}) x^{(i)} \tag{6}$$

and the update rule is

$$\theta_j \leftarrow \theta_j - \alpha(\theta^T x^{(i)} - y^{(i)}) x_j^{(i)} \tag{7}$$

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 monotonically in 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

$$\frac{1}{k}\sum_{i=1}^{k}\nabla J(\theta; x^{(i)}, y^{(i)}) \tag{8}$$

4 A Closed-Form Solution to Linear Regression

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

$$X = \begin{bmatrix} (x^{(1)})^T \\ \vdots \\ (x^{(m)})^T \end{bmatrix} \qquad Y = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix} \tag{9}$$

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

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}) = \frac{1}{2}(X\theta - Y)^T(X\theta - Y) \tag{10}$$

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

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

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

5 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" $\theta^T x$ with noise ε. Therefore, we have

$$y = x + \varepsilon$$

We assume ε 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

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

Hence, the conditional probability density function of y given x is defined by

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

It is shown that the distribution of y given x is parameterized by θ, i.e.,

$$y \mid x; \theta \sim \mathcal{N}(\theta^T x, \sigma^2)$$

Considering the training data $\{(x^{(i)}, y^{(i)})\}_{i=1,\cdots,m}$ are sampled independently, we define the corresponding likelihood function

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

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

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

Now, we conclude that, the rationality of adopting the least square in the linear model comes from the fact that the training data are sampled with Gaussian noise.