Lecture 6: Regression
Modeling Social Data, Spring 2017
Columbia University

Ethan Liu

Feburary 24, 2017

1 Introduction

1.1 Definition

Regression analysis is the process of estimating the relationships among variables. The goal of using regression analysis is to provide a compact summary of available data, forcast future data, and explain the relationships between variables. Ultimately, we would want to strike a balance between being able to describe observed data and predicting future data.

1.2 Framework

We need the following steps to make a good regression analysis on a data set.

1. Make sound decisions about the outcome, predictor, and form of the model.
2. Define a loss function to describe how close the model's prediction is with respect to the observed data.
3. Devise an algorithm to minimize loss.
4. Define metrics to assess model performance and be able to interpret results.

2 Regression Models

2.1 Task

We have inputs k-dimensional vector x_i and scalar y_i for data points $i = 1, \ldots, N$, and would like to find model f such that $\hat{y} = f(x)$ by learning parameters of the model from data.

2.2 Loss Function

We have many different kinds of loss functions. These functions usually outputs some kind of difference between the predicted outcome and the actual outcome. Common loss functions include:

Absolute loss function $L[f] = \frac{1}{N} \sum_1^N |y_i - f(x_i)|$

Squre loss function $L[f] = \frac{1}{N} \sum_1^N (y_i - f(x_i))^2$

2.3 Maximum Likelyhood Interpretation

Assume some family of probablistic model generated the data, then find the model under which the observed data are most likely to occur.
If we assume that $y_i = f(x_i) + \epsilon_i$ where $\epsilon_i \sim N(0, \sigma^2)$, then we can have the following:

$$p(\epsilon_i|f) = p(y_i - f(x_i)|f) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(y_i - f(x_i))^2}$$

Thus we can write the likelyhood for the entire data set as the following:

$$p(D|f) = \prod_i^N p(D_i|f) = \prod_i^N \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(y_i - f(x_i))^2}$$

Taking log probability this becomes:

$$\log p(D|f) = -k_1 N - k_2 \sum_1^N (y_i - f(x_i))^2$$

where k_1 and k_2 are some constants. Since $-k_2 \sum_1^N (y_i - f(x_i))^2 = -k_2 L[f]$ where L is the square loss function, we have shown that minimize the square loss of a data set is equivalent to maximizing the log likelyhood of that data set.

2.4 Model Parameters

Suppose $\hat{y} = f(x) = w \cdot x$ where w is the set of parameters for the model and x is the set of features of the data. We would like to devise a way to find w that minize the loss over the entire data set. Let $L = \frac{1}{N} \sum_1^N (y_i - w \cdot x_i)^2$, we take the derivative of L and set it to 0:

$$\frac{\partial L}{\partial w} = \frac{1}{N} \sum_1^N \frac{\partial}{\partial w} (y_i - w \cdot x_i)^2 = \frac{2}{N} \sum_1^N (y_i - w \cdot x_i) \frac{\partial}{\partial w} (w \cdot x_i) = \frac{2}{N} \sum_1^N (y_i - w \cdot x_i) x_i = 0$$

Then we can write this equation in matrix form as:

$$\frac{\partial L}{\partial w} = X^T (y - Xw) = 0$$

and we finally derive the normal equation:

$$\hat{w} = (X^T X)^{-1} X^T y$$

Computing this closed form solution, however, requires $O(Nk + k^3)$ runtime. This is very computationally expensive if k, the number of dimensions for the input feature, is large, making this method impractical for high dimensional data.

3 Gradient Descent Methods

Luckily we have iterative methods to find the best model parameters. The idea is to start at a random place in the parameter space, then follow the negative gradient to the bottom of the bowl shaped surface. We introduce three versions of this method.

3.1 Gradient Descent

We use

$$w := w - \alpha \frac{\partial L}{\partial w}$$

which is

$$w := w - \alpha \frac{2}{N} X^T (y - Xw)$$

as the update rule at each step. Parameter α is called the learning rate, which is the step size we take at each iteration in the direction of negative gradient. This method take $O(kN)$ to compute the update for w at each step, which is still very expensive if k is large.

3.2 Batch Stochastic Gradient Descent

We could use a randomly selected subset of the data to update the model parameters at each step instead of the whole set of data. So the update rule becomes:

$$w := w - \alpha \frac{2}{n} \sum_{1}^{n} (y_i - w \cdot x_i) x_i$$

where n is a much smaller number than N. This computes the approximate gradient at a cost of $O(kn)$ per step. If n is small enough, this method could make a hugh difference in terms of runtime.

3.3 Stochastic Gradient Descent

We could also use a single randomly selected data point to update parameters in each iteration, using the following update rule:

$$w := w - \alpha (y_i - w \cdot x_i) x_i$$

This method takes $O(k)$ and often can give the model parameters close enough to the optimal ones in much less time than the first gradient decnet method in practice.