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

Himani Arora

February 24, 2017

1 Regression Task

We have a set of predictors (also called inputs or features) $\{x_i\}_{i=1}^N$ with their corresponding outcomes $\{y_i\}_{i=1}^N$ where each x_i is a k dimensional vector. We want to learn a function f from the data such that

$$\hat{y} = f(x) \tag{1}$$

where $\hat{y}$ is our prediction.
To evaluate how well f predicts, we define a loss function.

2 Loss Function

The squared loss $L[f]$ is given by

$$L_i[f] = (y_i - \hat{y}_i)^2 \tag{2}$$

where $\hat{y}_i = f(x_i)$ Then the total loss over all the N data vectors is

$$L[f] = \frac{1}{N}\sum_{i=1}^{N} L_i[f] = \frac{1}{N}\sum_{i=1}^{N}(y_i - f(x_i))^2 \tag{3}$$

It may seem a bit abstract why we are using this particular loss function and where it came from. To study the motivation behind this, we define the Maximum Likelihood.

3 Maximum Likelihood Estimation

There are two steps to define the MLE:

- Assume some family of probabilistic models generated from the data.
- Find the model under which the observed data is most likely.

This can be mathematically framed as

$$f^* = \arg\max_x p(D|f) \tag{4}$$

A common assumption is

$$y_i = f(x_i) + \epsilon_i \tag{5}$$

where, $\epsilon_i = N(0, \sigma^2)$ is normally distributed additive noise. Thus,

$$P(\epsilon_i|f) = P(y_i - f(x_i)|f) = \frac{1}{\sqrt{2\pi\sigma^2}}exp(-\frac{1}{2\sigma^2}(y_i - f(x_i))^2) \tag{6}$$

The likelihood is then given by

$$P(D|f) = \prod_{i=1}^{N} P(\epsilon_i|f) = (2\pi\sigma^2)^{-N/2} exp(-\frac{1}{2\sigma^2}\sum_{i=1}^{N}(y_i - f(x_i))^2) \tag{7}$$

Noticing that the exponents form a sum, we can instead take the log likelhood to simplify the expression. Moreover, since the log function is a monotonically increasing function, this will not affect the optimal solution.

$$\log P(D|f) = -\frac{N}{2}\log 2\pi\sigma^2 - \frac{1}{2\sigma^2}\sum_{i=1}^{N}(y_i - f(x_i))^2 \tag{8}$$

A closer inspection tells us that the first term is independent of f and the second term is nothing but a scaled version of the loss function $L[f]$ as defined earlier. Therefore, we can conclude that minimizing the squared loss is equivalent to maximizing the (log) likelihood assuming additive Gaussian noise.

4 Finding f(x)

Note on matrix notation:

- X is a Nxk matrix where each column X_i is the i^{th} feature of all data points $x_1 \cdots x_N$ and the first column X_1 corresponds to the intercept and is equal to 1
- w is a kx1 column matrix where the first element corresponds to the bias.
- y is a Nx1 column matrix where y_i is the predictor for the i^{th} data point x_i

To find f, we assume a linear model in the weight vector w

$$\hat{y} = f(x) = Xw \tag{9}$$

For a single data point, this can be written as

$$\hat{y_i} = f(x_i) = x_i^T w = w_1 x_{i1} + w_2 x_{i2} + \cdots + w_k x_{ik} \tag{10}$$

Now the task of finding f boils down to finding k numbers $w_1 \cdots w_k$. This is known as a parametric model.

$$w^* = \arg\min_w \frac{1}{N}\sum_{i=1}^{N}(y_i - wx_i)^2 \tag{11}$$

This can be done in any one of the following ways each of which has its own limitations and use-cases.

4.1 Brute Force

This is the most naive method which is generally not feasible in many practical problems. In this, all possible values of w are tried and the one at which the loss function is minimum is chosen to be the optimal w^*. As the dimension of the weight vector increases, the number of possible choices of w also increases exponentially, and this can become a near-impossible task.

4.2 Normal Equations

In this we try to minimize the loss function by equating its gradient w.r.t w to zero.

$$\frac{\partial L}{\partial w} = \frac{1}{N}\sum \frac{\partial}{\partial w}(y_i - wx_i)^2 = 0 \tag{12}$$

Solving this, we get

$$\frac{\partial L}{\partial w} = -\frac{2}{N}\sum_{i=1}^{N}(y_i - wx_i)x_i = 0 \tag{13}$$

In matrix notation, this can be written as

$$X^T(y - Xw) = 0 \tag{14}$$

This gives,

$$\hat{w} = (X^T X)^{-1} X^T y \tag{15}$$

This is called Normal Equation.

4.2.1 Cost

Analyzing the cost in terms of running time we get:

- $X^T y = O(kN)$
- $X^T X = O(k^2 N)$
- $(X^T X)^{-1} = O(k^3)$

Thus, the total cost is $O(k^2 N + k^3)$. This is good if k is small (of the order 100). However, when k is very large this will take too much time to compute. Thus, this method is not suitable for very wide data ($k >> N$).

4.3 Batch Gradient Descent

Instead of directly computing w^* which corresponds to the w at the minima of the loss function, we can take an iterative approach. This is based on the idea that if we start from some (random) point and follow the loss surface, we will eventually get to the bottom of it. Since gradient gives the direction of steepest ascent, we follow the direction of the negative of gradient which gives the direction of steepest descent. Thus, the updation step to calculate the new weight vector becomes:

$$w = w - \eta\frac{\partial L}{\partial w} \tag{16}$$

where, w is the current weight vector, $\frac{\partial L}{\partial w}$ is the gradient evaluated at the current weight vector and η is the step size which controls how big the step is.
Note on step size:

- Larger η results in bigger steps and thus, care must be taken to not overshoot the minima.
- Smaller η results in smaller steps and too small η can take really long to reach the minima.

Substituting the gradient in equation(16), we get:

$$w = w + \eta\frac{2}{N}X^T(y - Xw) \tag{17}$$

where, $X^T(y - Xw)$ is a vector of residuals (errors) that takes the error on each example, multiplies it with the k^{th} feature and averages it over all the samples.

4.3.1 Cost

Analyzing the cost per iteration in terms of running time we get:

- $Xw = O(kN)$
- $y - Xw = O(N)$
- $X^T(y - Xw) = O(kN)$

Thus, the total cost per iteration is $O(kN)$. This is good (and better than the previous methods) as long as we can converge in a few steps.

4.4 Stochastic Gradient Descent

While batch gradient descent follows the gradient, stochastic gradient descent follows an approximation of the gradient. This can be understood by examining that the gradient as given by

$$\frac{\partial L}{\partial w} = -\frac{2}{N}\sum_{i=1} N(y_i - wx_i)x_i \tag{18}$$

is an average of N observations and therefore, we can just take a subset of the N observations to get a stochastic estimate of the gradient.
Thus, stochastic gradient descent take a random sample set of n observations where $n \leq N$ to estimate the gradient.

$$\frac{\partial L}{\partial w} = -\frac{2}{n}\sum_{i=1} n(y_i - wx_i)x_i \tag{19}$$

Although this is not always exactly equal to the actual gradient, however, it isn't too far off and on an average we usually move in the right direction and eventually reach the minima. In practice, $n = 1$ that is, estimating the gradient using just a single example also works surprisingly well. The step size η, is generally made smaller to guarantee convergence.

4.4.1 Cost

Analyzing the cost per iteration in terms of running time we get:

- $Xw = O(kn)$
- $y - Xw = O(n)$
- $X^T(y - Xw) = O(kn)$

Thus, the total cost per iteration is $O(kn)$. However, we need more iterations now as compared to batch gradient descent.
If we have a lot of samples, stochastic gradient descent works much better, however, with fewer examples it is not a good choice.

4.5 Second Order Derivative methods

In some methods such as the Newton's method, second order derivative of the loss function is calculated to estimate the curvature of the surface. This is because the curvature of the surface can give useful information for tuning the step size. For example, when the surface is flat, we can afford to take big jumps, on the other hand, when the surface is steep we should take smaller steps. However, calculating the second order derivative is computationally expensive and is done only if enough resources are available and if the gradient descent either takes too long or does not converge at all.

5 Additional notes on R Example

- In heavy tail distribution, log-scale is a better option as it can offer more insights into the data and its distribution.
- When the data is heavily skewed, median is more indicative than mean.
- Geometric mean (in log space) and median are very close in terms of numerical value. For this reason, many times the median is estimated by calculating the geometric mean which offers the advantage of being easier to compute and can also be calculated in a streaming setting. It is given by $10^{Mean(\log_{10} x)}$

- Data can be transformed by applying a function $\phi(x)$ without changing anything else.

$$f(x) = wx \Rightarrow f(x) = w\phi(x) \tag{20}$$

Depending on $\phi(x)$, $f(x)$ can also be non-linear in x. The squared loss is then given by

$$L[f] = \frac{1}{N}\sum_{i=1}^{N}(y_i - w\phi(x_i))^2 \tag{21}$$

which is still quadratic in w and hence, the problem remains the same.