CS6301: R For Data Scientists

LECTURE 14: LEAST SQUARES REGULARIZATION

Problem Setup

Recall the linear regression problem: Suppose we are given a training set $\{(\boldsymbol{X}, y_i)\}$ where the response variable y_i is now numeric and $\boldsymbol{X}$ is a vector of predictors

Our first assumption is that *y* is some function of the predictors, and perhaps a noise component (which we assume is normally distributed with zero mean and constant variance):

$$y = f(\boldsymbol{X}) + \epsilon$$

Our goal is to find an approximation to *f(),* which we will assume has a linear form:

$$f(\boldsymbol{X}) = \beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k, \qquad \boldsymbol{X} = < x_1, \ldots, x_k >$$

Problem Setup

We look at the problem slightly differently: Set up a <u>cost function</u>, then find approximations to the betas that minimize this cost function

For this problem, the cost function is just RSS:

$$RSS = C(\hat{\boldsymbol{\beta}}) = \sum_{i=1}^{n}\left(\hat{\beta}_0 + \sum_{j=1}^{k}\hat{\beta}_j x_{i,j} - y_i\right)^2$$

Finding the $\hat{\beta}_j$ values that minimize this function gives what is known as the <u>Least Mean Squares</u> solution, or ordinary least squares model

A Calculus Detour

If we have a function *G(x,y)*, the domain is the *xy* plane

Can think of the function as being represented as a surface over the plane

The gradient vector is a vector in the *xy* plane defined as:

$$\nabla G(x,y) = \left(\frac{\partial G}{\partial x}, \frac{\partial G}{\partial y}\right)$$

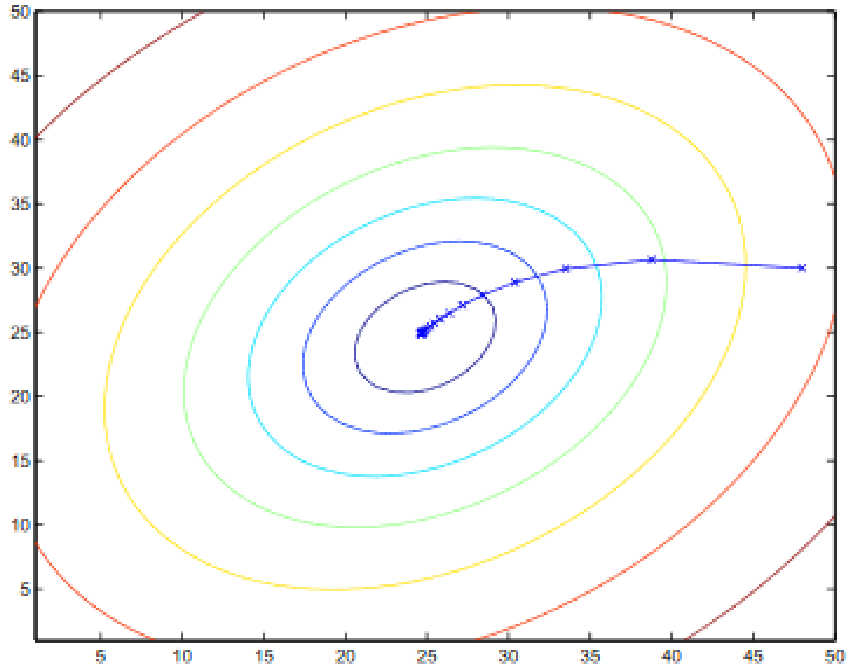
Key factor: At any point *(x, y)* in the domain, the gradient vector points in the direction of fastest increase for *G(x,y).*

Gradient Descent

We can use the gradient of the cost function to try to find values for the betas that will minimize the cost function …

Algorithm: Start with an initial guess for the betas, move in the direction of the <u>negative</u> gradient, update our guess of the betas, repeat until convergence (gradient descent)

Alternative: Set the partial derivatives equal to zero and solve! (These are the *normal equations.*)



Gradient Descent

Let's suppose β_0 is zero for the purposes of this derivation

Define the residual for the *i*th data point to be $e_i = \sum_{j=1}^{k} \hat{\beta}_j x_{i,j} - y_i$

Let's just look at one data point (*i = 1*) for the moment …

Can we use the gradient descent idea to minimize the RSS for this data point?

Note that (for one data point only) $RSS = e_1^2$. So ...

$$\frac{\partial RSS}{\partial \hat{\beta}_j} = 2e_1 x_{1,j}$$

Actually, if we had more data points, we would just get a sum of such terms (for each *j*)

Gradient Descent

Algorithm: We start with an initial (random) value for all the $\hat{\beta}_j$ coefficients …

We then update our coefficients using the gradient descent method

$$\hat{\beta}_j := \hat{\beta}_j - \eta e_1 x_{1,j}, \qquad \mathrm{j} = 1, \ldots, \mathrm{k}$$

Here η is a "learning rate", usually a small number. We can iterate on this equation until convergence

- Note: Each time we update the betas we can calculate a new e_1 value

Gradient Descent

If we have more than one training point, we can use them all (remember *RSS* is just the sum of the residuals squared) …

$$\hat{\beta}_j := \hat{\beta}_j - \sum_{i=1}^{n} \eta e_i x_{i,j}\,, \qquad \mathrm{j} = 1, \ldots, \mathrm{k}$$

This is called <u>batch gradient descent</u> because we look at all of the training examples for each iteration

- Can also do just one training example at a time

Regularization

Regularization puts constraints on betas …

- Essentially will “select” the most important ones

Change the cost function:

$$C(\widehat{\boldsymbol{\beta}}) = \sum_{i=1}^{n} (e_i)^2 + \frac{\lambda}{2} \sum_{j=1}^{k} |\hat{\beta}_j|^q$$

Here λ is a tuning parameter

Notice this limits how large the betas can get

Two popular choices are *q = 1* (Lasso) and *q = 2* (Ridge)

Regularization

Regularization will limit the betas in a situation when there are many variables

By adding the penalty term to the cost function, the betas are forced to be small

- It essentially shrinks the "non-important" features

Lasso vs. Ridge:

- Ridge will keep all variables (betas), but the more important ones will be larger (in magnitude) than the less important ones
- Lasso will set the less important betas to zero and keep the best ones

Alternative formulations

$$\underset{\beta}{\text{minimize}} \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{k} \beta_j x_{ij} \right)^2 \right\} \quad \text{subject to} \quad \sum_{j=1}^{k} |\beta_j| \leq s$$

$$\underset{\beta}{\text{minimize}} \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{k} \beta_j x_{ij} \right)^2 \right\} \quad \text{subject to} \quad \sum_{j=1}^{k} \beta_j^2 \leq s,$$

Ridge vs lasso