October 6, 2017 Lecture 5, Sep. 20, 2017

Linear Regression and Shrinkage

Lecturer: Justin Domke *Scribe: Boya Ren*

1 Summary

Last time we talked about linear regression and minimizing Residual Sum of Squares (RSS). Today we will continue to discuss how to solve a linear regression problem and shrinkage/regularization terms.

Reading

Linear Regression and Shrinkage: "The Elements of Statistical Learning" Sec 3-3.2.1, 3.2.4-3.4.3, 3.9

Cross Validation: "An Introduction to Statistical Learning" Sec 5-5.1.4

2 Linear Regression

Recall from the last lecture, that our objective function for linear regression is to minimize $RSS(\beta) = \sum_{i=1}^{N}(y_i - x_i^T\beta)^2$, y is a length N vector and $\mathbf{X}$ is an $N \times p$ matrix. β is a length p parameter vector that we want to learn. RSS can also be written in the matrix form $RSS(\beta) = (\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta)$. In order to minimize RSS, we need to compute its gradient. First we will give the result as below and then briefly explain how it is obtained.

$$\nabla RSS(\beta) = \begin{bmatrix} \frac{\partial RSS}{\partial \beta_1} \\ \frac{\partial RSS}{\partial \beta_2} \\ \vdots \\ \frac{\partial RSS}{\partial \beta_n} \end{bmatrix} = -2X^T(\mathbf{y} - \mathbf{X}\beta)$$

2.1 Basic Matrix Calculus

A simple but non-elegant approach to do matrix calculus is to use indices. We will first illustrate this method by the following example.

Suppose we have $x \in \mathbb{R}^n$, $y \in \mathbb{R}$ and $A \in \mathbb{R}^{n \times n}$. Function $f(x) = x^TAy$ is a mapping $\mathbb{R}^n \to \mathbb{R}$ and we would like to compute $\nabla f(x)$. Since $f(x)$ can be represented as the sum below,

$$f(x) = \sum_{i=1}^{n}\sum_{j=1}^{n} x_i A_{ij} y_j$$

The k^{th} entry of $\nabla f(x)$ can be computed as

$$\begin{aligned}\frac{\partial f}{\partial x_k} &= \sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij}y_j\frac{\partial x_i}{\partial x_k} \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij}y_j I[i=k] \\ &= \sum_{j=1}^{n} A_{kj}y_j = A_{k\cdot}\mathbf{y}\end{aligned}$$

Here $I[condition]$ is the indicator function that is 1 when the condition is true and 0 when it is not true. $A_{k\cdot}$ is a row vector, standing for the k^{th} row in matrix A. As a result, the derivative of the function can be written as

$$\nabla f(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix} = \begin{bmatrix} A_{1\cdot}y \\ A_{2\cdot}y \\ \vdots \\ A_{3\cdot}y \end{bmatrix} = A\mathbf{y}$$

We can summarize calculating matrix derivatives by indices as the following three steps.

1. "Unroll" all matrix operations into sums.
2. Take calculus by "normal" derivatives.
3. "Re-assemble" the result into the matrix form.

2.2 Solving Linear Regression

Now let's come back to our problem to find the best β that satisfies $\nabla RRS(\beta) = \vec{0}$. So in the first step, we will show how to derive $\nabla f(\beta)$. The target function can be written as the sum below.

$$\begin{aligned}RSS(\beta) &= (\mathbf{y}-\mathbf{X}\beta)^T(\mathbf{y}-\mathbf{X}\beta) \\ &= \sum_{i=1}^{N}(y_i - (\mathbf{X}\beta)_i)^2 \\ &= \sum_{i=i}^{N}\left(y_i - \sum_{j=1}^{p}X_{ij}\beta_j\right)^2\end{aligned}$$

Then by normal calculus, the k^{th} partial derivative can be derived as below.

$$\begin{aligned}\frac{\partial RSS}{\beta_k} &= \sum_{i=1}^{N} 2\left(y_i - \sum_{j=1}^{p}X_{ij}\beta_j\right)\frac{\partial}{\partial\beta_k}\left(y_i - \sum_{j=1}^{p}X_{ij}\beta_j\right) \\ &= -2\sum_{i=1}^{N}\left(y_i - \sum_{j=1}^{p}X_{ij}\beta_j\right)X_{ik} \\ &= -2\mathbf{X}_{\cdot k}^T(\mathbf{y}-\mathbf{X}\beta)\end{aligned}$$

By combining all k into a vector, we get the gradient to be $\nabla f(\beta) = -2\mathbf{X}^T(\mathbf{y} - \mathbf{X}\beta)$.

Finally, we need to find β such that $\nabla RSS(\beta) = 0$, which is equivalent to $\mathbf{X}^T\mathbf{X}\beta = \mathbf{X}^T\mathbf{y}$. As we result, we have $\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$. This is an important result for linear regression, but in practice, usually it is not numerically stable. More often than not, we would used the following procedures.

1) Compute $\mathbf{X}^T\mathbf{X}$ with time complexity $O(Np^2)$.
2) Compute $\mathbf{X}^T\mathbf{y}$ with time complexity $O(Np)$.
3) Solve $(\mathbf{X}^T\mathbf{X})\beta = \mathbf{X}^T\mathbf{y}$ for β with time complexity $O(p^3)$.

There are some numerically stable methods for step **3)** and the total complexity is $O(p^2 \max\{p, N\})$. Consider N is usually much larger than p, the complexity is p^2N.

Some what is the minimum time complexity for doing 2 $N \times N$ matrix multiplication? No one really knows. A naive algorithm takes $O(N^3)$ time, but it can be reduced by some methods such as divide and conquer. It is conjectured that this can be done in $N^{2+\epsilon}$ time with small enough ϵ.

3 Regularization

Regularization has another more straightforward name "shrinkage", which indicates shrinking the expressive ability of the model. Before coming to the concepts, let's look at an example.

Suppose we have some data generated for a polynomial with additional noises, as shown in the left subfigure below. But we don't know what order of polynomial we should use to do the regression task. As seen in the right subfigure, if we use a low order model, as the orange linear line, it produces large error in the data. But we choose a too-high-order model as the red curve, the data is overfit. It seems like somewhere in the middle as the green line is ideal. But how can we adjust our model with no ordering of features?

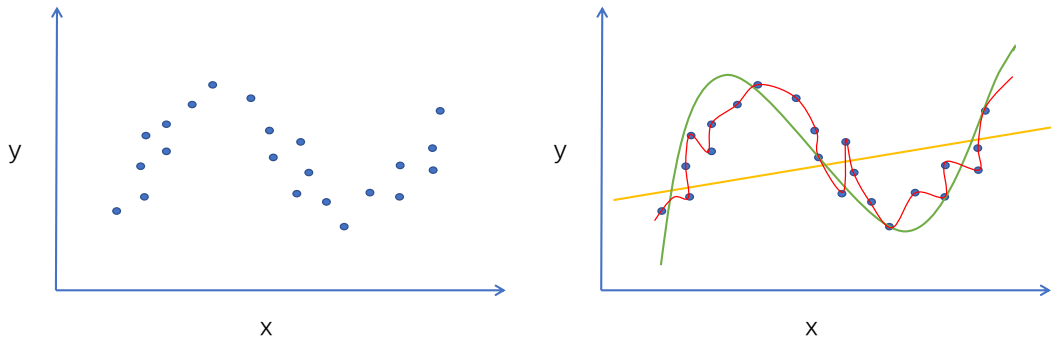

Figure 3.1: Regression with Different Orders of Polynomials

A more formal way to propose the question is to use capacity. Can the given data fit a simple or a complex st of functions to it? As the illustrative example above, low capacity usually means high bias (error between predicted and true output) and low variance (fluctuation of the output), while high capacity usually corresponds to low bias but high variance. So our goal is to adjust the capacity to find the best tradeoff between bias and variance.

There are many ways to change capacity. Let's take linear regression as an example to discuss the intuitions.

Idea 1: Make indices of β zero

- Best subset
- Forward stepwise selection
- Backward stepwise selection

Idea 2: Keep all β but make them smaller

- Ridge regression

Idea 3: A bit of both

- Lasso regression

3.1 Best Subset

The best subset method for linear regression is shown below.

Algorithm 1 Best Subset for Linear Regression

1: **for** $k = 1, 2, ..., p$ **do**
2: Try all subsets of indices of size k
3: Fit training data with each model
4: Remember β with k non-zeros with the lowest training error
return β for each value of k

The pros and cons of best subset is discussed in the following.

Pros:

- Satisfying (because it searches all possible cases)
- Presents sparse solutions

Cons:

- Computationally expensive
- Somewhat "unstable" (totally different βs might be non-zero with different k)