Regressions And Stuff

1 Function Basis

Consider a basic prediction problem where given an input, x, we wish to predict a value y (e.g. predict income given GPA, or predict crime rate given temperature). We train on a sample of labeled data (a set of (x, y) pairs) to find a function $f(x)$ that will give the expected y for new inputs x.

One approach to this problem is to model f as the linear combination of *basis functions*. In other words, we manually choose functions $\phi_1(x), \ldots, \phi_n(x)$ and algorithmically pick coefficients $c_1, \ldots, c_n$, which we put together as

$$bmf(x) = c_1\phi_1(x) + \cdots + c_n\phi_n(x).$$

We can rewrite this with

$$\begin{aligned} \boldsymbol{c} &= \begin{bmatrix} c_1 & \cdots & c_n \end{bmatrix} \\ \boldsymbol{\phi}(x) &= \begin{bmatrix} \phi_1(x) & \cdots & \phi_n(x) \end{bmatrix} \end{aligned}$$

so that

$$f(x) = \boldsymbol{c} \cdot \boldsymbol{\phi}(x). \tag{1}$$

The choice of the basis functions is somewhat arbitrary and often depends on the problem domain. Common choices are monomials ($\phi_i(x) = x^i$) and sinusoidals ($\phi_i(x) = \sin(i\pi x)$).

As for the algorithm to pick the coefficient vector $\boldsymbol{c}$, we discuss three methods: *linear regression*, *ridge regression*, and *LASSO* (least absolute shrinkage and selection operator). We devote one section to exploring each method in the next three sections.

Before moving on, we develop common notation to set up the algorithmic problem. Suppose we are given the training data as column vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ of length m and have chosen n basis functions given by $\boldsymbol{\phi}$. We let $X = \boldsymbol{\phi}(\boldsymbol{x})$ be the m by n matrix where the i-th row is $\boldsymbol{\phi}(x_i)$ and x_i is the i-th element of $\boldsymbol{x}$. Thus, we can roughly frame our algorithm's goal as finding a column vector $\boldsymbol{c}$ of size n such that

$$\boldsymbol{y} \approx X\boldsymbol{c}, \tag{2}$$

where X and $\boldsymbol{y}$ are taken as inputs.

2 Linear Regression

Linear regression is a regression method that simply aims to minimize the square of the magnitude of the difference of the two sides of (2):

$$\left|\boldsymbol{y} - X\boldsymbol{c}\right|^2. \tag{3}$$

The expression in (3) is known as the *sum of squares error*, or SSE, as it is the sum of the squares of the differences in predicted versus real y values.

One way to find the optimal $\boldsymbol{c}$ is to expand the expression and set its partial derivative with respect to $\boldsymbol{c}$ equal to zero. Doing so yields

$$\boldsymbol{c} = \left(X^T X\right)^{-1} X^T \boldsymbol{y}. \tag{4}$$

We refer to (4) as the closed form for $\boldsymbol{c}$.

Another way to find $\boldsymbol{c}$ is to perform gradient descent with SSE as the objective function of $\boldsymbol{c}$.

We will illustrate and contrast these methods with an example, and then show the effects of choosing a different function basis. Our example is generated by taking $\boldsymbol{x}$ as numbers from 0 to 1 at intervals of 0.1 and

$$\boldsymbol{y} = \cos(\pi \boldsymbol{x}) + \cos(2\pi \boldsymbol{x}) + \text{noise}.$$

2.1 Polynomials with Closed Form

We use $\boldsymbol{\phi}$ of size $n+1$ whose components are monomials of order 0 through n as our funciton basis. Using this to compute X and evaluate the closed form, we can obtain $\boldsymbol{c}$. We view the results as a plot of the training data points, the generating curve (without noise), and the predictor $f(x) = \boldsymbol{c} \cdot \boldsymbol{\phi}(x)$. This is shown in Figure 2.1 for several values of n.

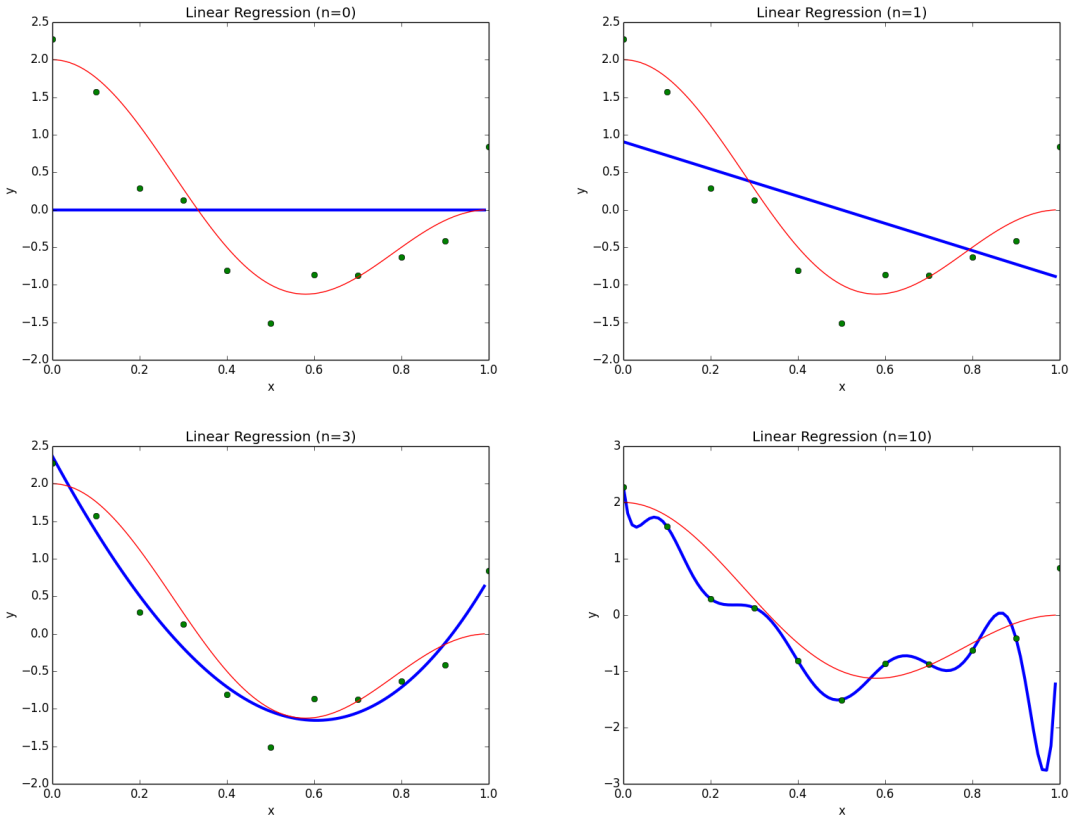

Fig. 2.1: Training data (green), generating curve (red), and predicted curve (blue) for various n.

We can see that increasing the dimension of the function basis improves the training accuracy of our predictor.

2.2 Polynomials with Gradient Descent

Use batch gradient descent on the SSE function on some values of M. Describe your experience with initial guesses, step sizes and convergence thresholds. Compare with using SGD on the same data. Explain your results in terms of the properties of the function being optimized and the properties of the algorithms.

2.3 Cosines with Closed Form and Gradient Descent

We now try using the function basis consisting of $\cos(\pi x), \cos(2\pi x), \ldots, \cos(n\pi x)$. In this case the closed form and both versions of gradient descent all give the same coefficient vectors after converging. We plot in Figure 2.3 comparisons for a couple values of n.

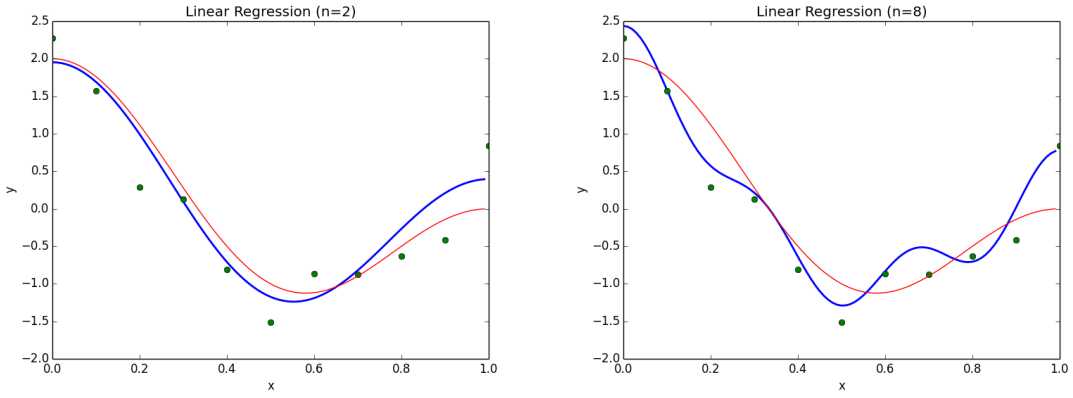

Fig. 2.2: Training data (green), generating curve (red), and predicted curve (blue) for various n.

A notable observation is that even when the same basis functions were used as in the generating function ($\cos(\pi x)$ and $\cos(2\pi x)$ when $n = 2$), we obtained a slightly different predicted curve. This is due to the added noise.

Also, when we increase the dimension of the basis ($n = 8$) the predicted curve becomes more complex. Although it is more accurate for the training data, it is more sensitive to the noise and so wouldn't generalize well to new test data. This behavior is called *overfitting*, which results when the coefficients become too large or when too many features (basis functions) are used. We will see below that these issues can be addressed using ridge regression and LASSO, respectively.

3 Ridge Regression

Ridge regression is similar to linear regression in that its choice of $\boldsymbol{c}$ aims to minimize SSE, but in addition also aims to minimize the size of the coefficients themselves. Tangibly, this is achieved by choosing $\boldsymbol{c}$ that gives the minimal value of

$$|\boldsymbol{y} - X\boldsymbol{c}|^2 + \lambda |\boldsymbol{c}|^2 . \tag{5}$$

We can tune the sensitivity to coefficient sizes by adjusting the parameter λ. At the extreme, $\lambda = 0$ will have no consideration of coefficient size and will be the same as linear regression.

Again, there are two ways to arrive at the desired $\boldsymbol{c}$ from here. First, we can expand (5) and set the partial derivative with respect to $\boldsymbol{c}$ equal to zero, which gives

$$\boldsymbol{c} = \left(X^T X + \lambda I\right)^{-1} X^T \boldsymbol{y}.$$

Second, we can perform gradient descent and converge to the optimal value of $\boldsymbol{c}$ numerically.

In general, the advantage of the second term in the objective function is that resulting coefficients are smaller, thus empirically less sensitive to noise and more accurate on unseen data.

4 LASSO