Math 345 - Notes
Linear Regression
October 17, 2018

Linear Regression

We try to estimate the *unknown* target function $f : \mathcal{X} \to \mathcal{Y}$ that takes *input* from the set $\mathcal{X}$, and maps it to *output*, an element of $\mathcal{Y}$. Let $\mathcal{D}$ denote the data set used for training our model. Every pair $(input, output)$ in $\mathcal{D}$ must satisfy $f(input) = output$.

Suppose we established that *output* depends on *input* almost linearly, either by doing exploratory data analysis, computing parameters such as correlation, or having additional subject knowledge that expects linear relationship. If the *input* is one-dimensional, we use simple regression, otherwise, we use multiple regression. However, the algorithm is the same for both cases, so we will use the generalized setup for multiple regression. Assume the data set $\mathcal{D}$ has N points with d-dimensional input vectors $[x_1, x_2, \cdots, x_d]^T$. We are then searching for a function

$$h(x_1, x_2, \cdots, x_d) = w_0 + w_1 x_1 + \cdots + w_d x_d \approx y = f(x_1, x_2, \ldots, x_d).$$

For each input vector, add a 0^{th} coordinate, and set it equal to 1, as we did in the PLA algorithm, so the *input* is $(d+1)$-dimensional: $\mathbf{x} = [1, x_1, x_2, \cdots, x_d]^T$. Now the N data points are given by $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$. We want to find a coordinate vector $\mathbf{w}_{\text{lin}} = [w_0, w_1, \ldots, w_d]^T$ so that

$$y \approx h(\mathbf{x}) = \mathbf{w}_{\text{lin}}^T \mathbf{x}$$

Model error

This model is probabilistic in nature, that is, each pair $(\mathbf{x}, y)$ occurs with joint probability $P(\mathbf{x}, y)$, a probability with unknown distribution. The (least squares) error resulting from approximation of the target function with a hyperplane given by coordinate vector $\mathbf{w}$, is given by the expectation

$$E_{out}(\mathbf{w}) = \mathbb{E}\left[(\mathbf{w}^T \mathbf{x} - y)^2\right].$$

Without knowing the probability distribution P, we cannot compute the expectation and hence the *out* error. However, we can compute the error resulting from the approximation of sample data points form $\mathcal{D}$: for all $1 \le k \le N$, it approximates

$$y_k \approx \mathbf{w}^T \mathbf{x}_k,$$

so we define the *in* error as the average error of square distances from data points to the approximating hyperplane:

$$E_{in}(\mathbf{w}) = \frac{1}{N} \sum_{k=1}^{N} (\mathbf{w}^T \mathbf{x}_k - y_k)^2.$$

The goal is to minimize this error. We use multi-variable calculus to find the minimum of a multi-variable function and get

$$\mathbf{w}_{\text{lin}} = \underset{\mathbf{w}\in\mathbb{R}^{d+1}}{\operatorname{argmin}} E_{in}(\mathbf{w}),$$

that is, $\mathbf{w}_{\text{lin}}$ is the argument that minimizes the function E_{in}.

Let us set up some notation first: construct the $N\times(d+1)$-dimensional (input) matrix $X = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \cdots \\ \mathbf{x}_N^T \end{bmatrix}$, and output vector $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_N \end{bmatrix}$. We express the error function using this notation:

$$\begin{array}{rcll} E_{in}(\mathbf{w}) & = & \frac{1}{N}\sum_{k=1}^{N}(\mathbf{w}^T\mathbf{x}_k - y_k)^2 & \text{(by definition of } E_{in}(\mathbf{w})) \\ & = & \frac{1}{N}\left\|X\mathbf{w}-\mathbf{y}\right\|^2 & (\text{since } \|\mathbf{v}-\mathbf{u}\|^2 = \sum(v_k-u_k)^2) \\ & = & \frac{1}{N}\left(X\mathbf{w}-\mathbf{y}\right)^T\left(X\mathbf{w}-\mathbf{y}\right) & (\text{since } \|\mathbf{v}\|^2 = \mathbf{v}^T\mathbf{v}) \\ & = & \frac{1}{N}\left(\mathbf{w}^TX^T-\mathbf{y}^T\right)\left(X\mathbf{w}-\mathbf{y}\right) & (\text{from } (\mathbf{vu})^T = \mathbf{u}^T\mathbf{v}^T) \\ & = & \frac{1}{N}\left(\mathbf{w}^TX^TX\mathbf{w}-\mathbf{w}^TX^T\mathbf{y}-\mathbf{y}^TX\mathbf{w}+\mathbf{y}^T\mathbf{y}\right) & (\text{matrix multiplication}) \\ & = & \frac{1}{N}\left(\mathbf{w}^TX^TX\mathbf{w}-2\mathbf{y}^TX\mathbf{w}+\mathbf{y}^T\mathbf{y}\right) & (\text{using } \mathbf{v}^T\mathbf{u} = \mathbf{u}^T\mathbf{v} = \mathbf{v}\cdot\mathbf{u}) \end{array}$$

As in one-variable calculus, to find the minimum or maximum, we set the derivative equal to zero. But since here we have $(d+1)$ variables, we set all *partial* derivatives equal to zero.

The idea behind partial derivatives is simple. When taking the partial derivative of a function with respect to a variable, you treat all other variables as constants. Here is and example. Suppose you want to take partial derivatives of $f(x,y) = x^2y + \cos(x+y)$. To take derivative of f with respect to x, treat y as a constant, and similarly for derivative with respect to y we treat x as a constant:

$$f_x = \frac{\partial f}{\partial x} = 2xy + \cos(x+y)(1+0) \qquad f_y = \frac{\partial f}{\partial y} = x^2 + \cos(x+y)(0+1).$$

We collect all partial derivatives into a row vector called the gradient. For our example,

$$\nabla f = \left[2xy + \cos(x+y)\,,\; x^2 + \cos(x+y)\right]$$

The gradient vector has many important applications, but we will not go into details in this class. Since all partial derivatives have to be zero at the minimum, the gradient has to be the zero vector. A few properties we will use, analogous to properties of one-dimensional derivatives are

- If c is a constant scalar, $\nabla_{\mathbf{w}}(c) = \mathbf{0}$
- If $\mathbf{b}$ is a constant vector, $\nabla_{\mathbf{w}}(\mathbf{b}^T\mathbf{w}) = \mathbf{b}^T$

- For a matrix A, using the product rule, $\nabla_{\mathbf{w}}(\mathbf{w}^T A\mathbf{w}) = A\mathbf{w} + A^T\mathbf{w} = (A + A^T)\mathbf{w}$

Thus,

$$\begin{aligned}
\nabla_{\mathbf{w}} E_{in}(\mathbf{w}) &= \mathbf{0} \\
\nabla_{\mathbf{w}} \left[\frac{1}{N} \left(\mathbf{w}^T X^T X\mathbf{w} - 2\mathbf{y}^T X\mathbf{w} - \mathbf{y}^T\mathbf{y} \right) \right] &= \mathbf{0} \\
\nabla_{\mathbf{w}} \left[\left(\mathbf{w}^T (X^T X)\mathbf{w} - 2(X^T\mathbf{y})^T\mathbf{w} - \mathbf{y}^T\mathbf{y} \right) \right] &= \mathbf{0} \\
\left[X^T X + (X^T X)^T \right] \mathbf{w} - 2X^T\mathbf{y} + \mathbf{0} &= \mathbf{0} \\
2X^T X\mathbf{w} - 2X^T\mathbf{y} &= \mathbf{0} \\
X^T X\mathbf{w} &= X^T\mathbf{y}
\end{aligned}$$

Note that $X^T X$ is a square matrix. If it is also invertible, we can solve for $\mathbf{w}$:

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

If $X^T X$ is not invertible, one may still be able to find a solution for $\mathbf{w}$ in $X^T X\mathbf{w} = X^T\mathbf{y}$, but the solution may not be unique. Note that for our applications, since the number of data points in the training data set is much larger than the number of dimensions for the input ($N >> d$), the column vectors in X will be linearly independent, and thus $X^T X$ will be invertible.
Thus, we arrived at a relationship between the training data and the coordinate vector $\mathbf{w}_{\text{lin}}$ that minimizes the least squares error. We summarize the algorithm below.

Linear Regression Algorithm:

Step 1. Construct the $N \times (d+1)$-dimensional matrix X with rows $\mathbf{x}_k^T$, and the vector $\mathbf{y} = [y_1, \ldots, y_N]^T$

Step 2. Compute the matrix $A = (X^T X)^{-1} X^T$

Step 3. Find $\mathbf{w}_{\text{lin}} = A\mathbf{y}$.

Remark: $X\mathbf{w}_{\text{lin}}$ only approximates $\mathbf{y}$ due to sampling error.

Example 1: Consider the following selling data from sample of 10 Corvette cars, aged 1-6 years, available from the *Kelley Blue Book*. Here x denotes the age of the car and y denotes the selling price, in hundreds of dollars:

x	6	6	6	2	2	5	4	5	1	4
y	270	260	275	405	364	295	335	308	405	305

(a) Draw a scatter plot to determine if linear regression should be used.

(b) Run through the Linear Regression Algorithm to find $\vec{\mathbf{w}}_{\text{lin}}$

(c) Price a 3 year old Corvette, using your result in (b).