Regression

Like classification, regression is a type of supervised learning setup. Thus, its dataset is still $D_n = \{(x^{(1)}, y^{(1)}), ..., (x^{(n)}, y^{(n)})\}$. $x^{(i)} \in \mathbb{R}^d$, but the set $y^{(i)}$ belongs to will change: $y^{(i)} \in \mathbb{R}$. Whereas the y values in classification were discrete categories, the y values in regression are continuous values. For example, one might use regression to predict a company's stock price or the temperature in two days. Neither stock price nor temperature should be predicted using categories like $\{+1, -1\}$; they are continuous values.

The hypothesis for linear regression (finding a trend line) is written as $h(x; \theta, \theta_0) = \theta^T x + \theta_0$ when $\theta \in \mathbb{R}^d$ and $\theta_0 \in \mathbb{R}$.

The standard loss function that is conventionally used for linear regression is called squared loss: $L(g, a) = (g - a)^2$.

The regression setup involving the linear regression hypothesis and squared loss function is so common that it has earned a name: ordinary least squares.

1 Using Optimization to Find a Hypothesis

Quite naturally, upon seeing $h(x; \theta, \theta_0) = \theta^T x + \theta_0$, one should want to use optimization to learn the best hypothesis for the dataset D_n. Thus, an objective function must be written. To highlight how important regularizers are, J will first be written without $\lambda R(\Theta)$: $J(\theta, \theta_0) = \frac{1}{n} \sum_{i=1}^{n} (\theta^T x^{(i)} - \theta_0 - y^{(i)})^2$. The optimal parameters θ_0^* and θ^* are $\arg\min_{\theta, \theta_0} J(\theta, \theta_0)$. The calculus way of finding a minimum will work: find the derivative of the objective function, set it to 0, solve for critical points, and prove that critical points are minima. One can most definitely compute the derivative of J with respect to each entry of θ and whatnot to produce a system of equations that can be solved for optimal parameters, but using the matrix form of the system is more elegant. Say that $W = \begin{bmatrix} x_1^{(1)} & ... & x_d^{(1)} & 1 \\ \vdots & \ddots & & \vdots \\ x_1^{(n)} & ... & x_d^{(n)} & 1 \end{bmatrix}$. Each x value in D_n is turned into a row of W. Each row has a 1 added to its end in order to account for θ_0 (a separator-through-origin in d dimensions can be turned into a

separator-not-through-origin in $d-1$ dimensions). Also, $T = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(d)} \end{bmatrix}$. Now, one can use linear algebra to find the parameter array θ_{array} that minimizes a rewritten objective function $J(\theta_{array})$ instead of finding the best θ vector and θ_0 scalar for the original $J(\theta, \theta_0)$. $J(\theta, \theta_0) = J(\theta_{array}) = (W\theta_{array} - T)^T(W\theta_{array} - T)$. Now, one needs to find the gradient (vector of partial derivatives) of $J(\theta_{array})$, set it to 0, and solve for θ_{array}. $\nabla_{\theta_{array}} J = \frac{2}{n} W^T(W\theta_{array} - T)$. $0 = \frac{2}{n} W^T(W\theta_{array} - T)$, $0 = W^T(W\theta_{array} - T)$, $0 = W^TW\theta_{array} - W^TT$, $W^TW\theta_{array} = W^TT$, $W^TW\theta_{array} = W^TT$, $W^TW\theta_{array} = W^TT$, $W^TW\theta_{array} = W^TT$, $(W^TT)^{-1}W^TW\theta_{array} = (W^TT)^{-1}W^TT$, so $\theta^*_{array} = (W^TT)^{-1}W^TT$. This is an example of the rare closed form solution. One does not need a fancy computer or algorithm to use a closed form solution. It simply describes how to solve for θ^*_{array} using pencil and paper. The problem with $\theta^*_{array} = (W^TT)^{-1}W^TT$ is that (W^TT) might not be invertible such that $(W^TT)^{-1}$ can be computed. The way of overcoming this problem is to add the regularizer $||\theta||^2$ to J to create an objective function called the ridge regression objective function. $J_{ridge}(\theta, \theta_0) = [\frac{1}{n}\sum_{i=1}^{n}(\theta^T x + \theta_0 - y^{(i)})^2] + \lambda||\theta||^2$. The regularizer does not impact θ_0 because offsets do not have to be forced toward 0 to prevent overcomplicated parameters. $\nabla_{\theta_{array}} J_{ridge} = \frac{2}{n} W^T(W\theta_{array} - T) + 2\lambda\theta_{array}$. Solving for 0 yields $\theta^*_{array} = (W^TW + n\lambda I)^{-1}W^TT$. I is the identity matrix (the values along its diagonal are all set to 1 while the rest of the matrix is set to 0). Its diagonal of 1s being multiplied by n and $\lambda > 0$ adds weight to the diagonal (ridge) of W^TT, making it invertible.

Inverting $W^TW + n\lambda I$ takes $O(d^3n)$ time, so for high dimension datasets, gradient descent or stochastic gradient descent will be better for performance than the analytic method described above. For gradient descent or stochastic gradient descent, one must know that $\nabla_{\theta_{array}} J_{ridge} = \frac{2}{n}\sum_{i=1}^{n}(\theta^T_{array}x^{(i)} + \theta_0 - y^{(i)})x^{(i)} + 2\lambda\theta_{array}$ and $\frac{\partial J}{\partial \theta_{array}} = \frac{2}{n}\sum_{i=0}^{n}(\theta^T x^{(i)} + \theta_0 - y^{(i)})$.