Singapore University of Technology and Design
Information Systems Technology and Design
50.007 Machine Learning

Linear Regression

Linear regression

A linear regression function is simply a linear function of the feature vectors, i.e.,

$$f(x;\theta,\theta_0) = \theta \cdot x + \theta_0 = \sum_{i=1}^{d} \theta_i x_i + \theta_0 \tag{1}$$

Each setting of the parameters θ and θ_0 gives rise to a slightly different regression function. Collectively, different parameter choices $\theta \in \mathcal{R}^d$, $\theta_0 \in \mathcal{R}$, give rise to the set of functions $\mathcal{F}$ that we are entertaining. While this set of functions seems quite simple, just linear, the power of $\mathcal{F}$ is hidden in the feature vectors. Indeed, we can often construct different feature representations for objects. For example, there are many ways to map past values of financial assets into a feature vector x, and this mapping is typically completely under our control. This "freedom" gives us a lot of power, and we will discuss how to exploit it later on. For now, we assume that a proper representation has been found, denoting feature vectors simply as x.

Our learning task is to choose one $f \in \mathcal{F}$, i.e., choose parameters $\hat{\theta}$ and $\hat{\theta}_0$, based on the training set $S_n = \{(x^{(t)}, y^{(t)}), t = 1, \ldots, n\}$, where $y^{(t)} \in \mathcal{R}$ (response). As before, our goal is to find $f(x;\hat{\theta},\hat{\theta}_0)$ that would yield accurate predictions on yet unseen examples. There are several problems to address:

(1) How do we measure error? What is the criterion by which we choose $\hat{\theta}$ and $\hat{\theta}_0$ based on the training set?

(2) What algorithm can we use to optimize the training criterion? How does the algorithm scale with the dimension (feature vectors may be high dimensional) or the size of the training set (the dataset may be large)?

(1) Empirical risk and the least squares criterion

We measure training error in terms of *empirical risk*

$$R_n(\theta) = \frac{1}{n}\sum_{t=1}^{n} \text{Loss}(y^{(t)} - \theta \cdot x^{(t)}) \tag{2}$$

where, for simplicity, we have dropped the parameter θ_0. It will be easy to add it later on in the appropriate place. Note that, unlike in classification, the loss function now depends on the difference between the real valued target value $y^{(t)}$ and the corresponding linear prediction $\theta \cdot x^{(t)}$. There are many possible ways of defining the loss function. We will use here a simple squared error: $\text{Loss}(z) = z^2/2$. The idea is to permit small discrepancies (we expect the responses to include noise) but heavily penalize large deviations (that typically indicate poor parameter choices). As a result, we have

$$R_n(\theta) = \frac{1}{n}\sum_{t=1}^{n} \text{Loss}(y^{(t)} - \theta \cdot x^{(t)}) = \frac{1}{n}\sum_{t=1}^{n} (y^{(t)} - \theta \cdot x^{(t)})^2/2 \tag{3}$$

Our learning goal is not to minimize $R_n(\theta)$; it is just the best we can do (for now). Minimizing $R_n(\theta)$ is a surrogate criterion since we don't have a direct access to the test or generalization error

$$R_{n'}^{test}(\theta) = \frac{1}{n}\sum_{t=n+1}^{n+n'} (y^{(t)} - \theta \cdot x^{(t)})^2/2 \tag{4}$$

Let's briefly consider how $R_n(\theta)$ and $R_{n'}^{test}(\theta)$ are related. If we select $\hat{\theta}$ by minimizing $R_n(\theta)$, our performance will be measured according to $R_{n'}^{test}(\hat{\theta})$. This test error can be large for two different reasons. First, we may have a large *estimation error*. This means that, even if the true relationship between x and y is linear, it is hard for us to estimate it on the basis of a small (and potentially noisy) training set S_n. Our estimated parameters $\hat{\theta}$ will not be entirely correct. The larger the training set is, the smaller the estimation error will be. The second type of error on the test set is *structural error*. This means that we may be estimating a linear mapping from x to y when the true underlying relationship is highly non-linear. Clearly, we cannot do very well in this case, regardless of how large the training set is. In order to reduce structural error, we would have to use a larger set of functions $\mathcal{F}$. But, given a noisy training set S_n, it will be harder to select the correct function from such larger $\mathcal{F}$, and our estimation error will increase. Finding the balance between the estimation and structural errors lies at the heart of learning problems.

When we formulate linear regression problem as a statistical problem, we can talk about the structural error as "bias", while the estimation error corresponds to the "variance" of the estimator $\hat{\theta}(S_n)$. The parameters $\hat{\theta}$ are obtained with the help of training data S_n and thus can be viewed as functions of S_n

(2) Optimizing the least squares criterion

Closed form solution

We can also try to minimize $R_n(\theta)$ directly by setting the gradient to zero. So, formally, we find $\hat{\theta}$ for which $\nabla R_n(\theta)_{\theta=\hat{\theta}} = 0$. More specifically,

$$\begin{aligned}
\nabla R_n(\theta)_{\theta=\hat{\theta}} &= \frac{1}{n}\sum_{t=1}^{n} \nabla_\theta \{\, (y^{(t)} - \theta \cdot x^{(t)})^2/2 \,\}_{|\theta=\hat{\theta}} \\
&= \frac{1}{n}\sum_{t=1}^{n} \{\, -(y^{(t)} - \hat{\theta} \cdot x^{(t)}) x^{(t)} \,\} \\
&= -\frac{1}{n}\sum_{t=1}^{n} y^{(t)} x^{(t)} + \frac{1}{n}\sum_{t=1}^{n} (\hat{\theta} \cdot x^{(t)}) x^{(t)} \\
&= \underbrace{-\frac{1}{n}\sum_{t=1}^{n} y^{(t)} x^{(t)}}_{=b} + \underbrace{\frac{1}{n}\sum_{t=1}^{n} x^{(t)} (x^{(t)})^T}_{=A} \hat{\theta} \\
&= -b + A\hat{\theta} = 0
\end{aligned}$$

where we have used the fact that $\hat{\theta} \cdot x^{(t)}$ is a scalar and can be moved to the right of vector $x^{(t)}$. We have also subsequently rewritten the inner product as $\hat{\theta} \cdot x^{(t)} = (x^{(t)})^T \hat{\theta}$. As a result, the equation for the parameters can be expressed in terms of a $d \times 1$ column vector b and a $d \times d$ matrix A as $A\theta = b$. When the matrix A is invertible, we can solve for the parameters directly: $\hat{\theta} = A^{-1}b$. In order for A to be invertible, the training points $x^{(1)}, \ldots, x^{(n)}$ must *span* $\mathcal{R}^d$. Naturally, this can happen only if $n \geq d$, and is therefore more likely to be the case when the dimension d is small in relation to the size of the training set n. Another consideration is the cost of actually inverting A. Roughly speaking, you will need $\mathcal{O}(d^3)$ operations for this. If $d = 10,000$, this can take a while, making the stochastic gradient updates more attractive.

In solving linear regression problems, the matrix A and vector b are often written in a slightly different way. Specifically, define $X = [x^{(1)}, \ldots, x^{(n)}]^T$. In other words, X^T has each training feature vector as a column; X has them stacked as rows. If we also define $\vec{y} = [y^{(1)}, \ldots, y^{(n)}]^T$ (column vector), then you can easily verify that

$$b = \frac{1}{n} X^T \vec{y}, \;\; A = \frac{1}{n} X^T X$$