Linear Regression

Jack David Carson - February 9, 2025

Linear Regression Definition 1

A linear regression function is merely a linear function of the feature vectors and model parameters, such as

$$f(x; \theta, \theta_0) = \theta^T x + \theta_0 = \sum_{i=1}^{n} \theta_i x_i + \theta_0.$$

for parameter choices $\theta \in \mathcal{R}^d, \theta_0 \in \mathcal{R}$ that give rise to set of functions $f \in \mathcal{F}$.

We usually seek to find the ideal parameters $\hat{\theta}, \hat{\theta_0}$ based on training set S_n. In order to optimize we must first of course specify *optimimality*

1. How do we measure error? I.e. how do we say θ_a is better than θ_b?
2. What algorithm can be used to optimize the training criterion?
3. If the training set is not *much* larger than the parameter dimension d, there may be directions in the parameter space that are free variables unconstrained by the data. How do we set those degrees of freedom and regulate them?

01 Error

If we have a simple loss function $\text{Loss}(z) = z^2/2$, then our SVM is defined by

$$R_n(\theta) = \frac{1}{n} \sum_{i=1}^{n} \text{Loss}\left(y^{(i)} - \theta^\top x^{(i)}\right) = \frac{1}{n} \sum_{i=1}^{n} \left(y^{(i)} - \theta^\top x^{(i)}\right)^2 / 2$$

For entry i we can optimize the criterion by $\nabla_\theta \left(y^{(i)} - \theta^\top x^{(i)}\right)^2 / 2 = -\left(y^{(i)} - \theta^\top x^{(i)}\right) x^{(i)}$ via the chain rule. However, if we want to be more ambitious, we would like a closed form solution to the whole SVM instantly. We can solve for parameter $\hat{\theta}$ by

$$\begin{aligned}
\nabla_\theta R(\theta)_{\theta = \hat{\theta}} &= \frac{1}{n} \sum_{i=1}^{n} \nabla_\theta \left[\left(y^{(i)} - \theta^\top x^{(i)}\right)^2 / 2\right] \\
&= \frac{1}{n} \sum_{i=1}^{n} \left[-\left(y^{(i)} - \hat{\theta}^\top x^{(i)}\right) x^i\right] \\
&= \underbrace{-\frac{1}{n} \sum_{i=1}^{n} y^{(i)} x^{(i)}}_{:= b} + \underbrace{\frac{1}{n} \sum_{i=1}^{n} \left(\hat{\theta}^\top x^{(i)}\right) x^{(i)}}_{:= A} \\
&:= -b + A\hat{\theta} = 0
\end{aligned}$$

Remarkably, we have derived our way down to the most fundemental formula of linear algebra $Ax = b$. For very large examples of feature sizes, even this is still *technically* less efficient than gradient descent. But it's a nice thought. We can actually take this from coordinate form into matricies as

$$b = \frac{1}{n} X^{\mathsf{T}} \boldsymbol{y}, \quad A = \frac{1}{n} X^T X$$

for X^{T} is the matrix of the feature vectors stacked as columns, and $\boldsymbol{y}^{\mathsf{T}} = \left[y^{(1)}, y^{(2)}, ..., y^{(n)}\right]^{\mathsf{T}}$.

02 Regularization

In the event that matrix A is not invertible, the problem is ill-posed without the use of a regularization term. We've seen many times now the formula for loss with regularization. We can express it now in terms of an objective function $J_{n,\lambda}(\theta) = \frac{\lambda}{2}\|\theta\|^2 + R_n$ Using our previous result we can say that

$$\boldsymbol{\nabla}_\theta J_{n,\lambda} = \lambda\theta - \left(y^{(i)} - \theta^{\mathsf{T}} x^{(i)}\right) x^{(i)}$$

where theregularization parameter $\theta \geq 0$ quantifies the trade-off between keeping the parameters small – minimizing the squared norm $\|\theta\|^2/2$–and fitting to the training data-–minimizing the empirical risk $R_{n(\theta)}$. The use of this modified objective is known as **Ridge regression**.

It is reasonable that large values of λ will have a negative impact on the training error, since it is directly impeding the model from fitting the training dataset in the perfectly optimal way, in the hope that it will fit the test dataset better.