Chapter 1

Background

1.1 Machine Learning

Machine learning is the general task of finding patterns given a set of data. The methods by which these tasks are accomplished range from the simple linear regression to more complex neural networks. Machine learning problems fall into primarily two categories: supervised learning and unsepervised learning.

In the case of supervised learning, we are given a set of inputs, $\{x\}_{i=1}^{N} \in X$ and outputs, $\{y\}_{i=1}^{N} \in Y$ to some unknown function, f. These sets are often referred to as "features" and "labels" respectively. The goal is then to determine what that function is. This is usually not feasible due our model for the function not being complex enough, or for being too complex (this is known as overfitting, discussed in section 1.1.2.) The objective is therefore simplified as finding the function, $g \in M$ where M is some set of model functions, and where g minimizes some loss function, $L(g, x, y)$. That is:

$$\arg\min_{g \in M} L(g, x, y) \tag{1.1}$$

One simple case of this is linear regression. In linear regression, M is the set of linear (or in most cases affine) functions mapping X to Y. and the loss function is $L(g, x, y) = \frac{1}{N} \sum_{i=1}^{N} ||g(x_i) - y_i||^2$ It turns out that under these constraints, an

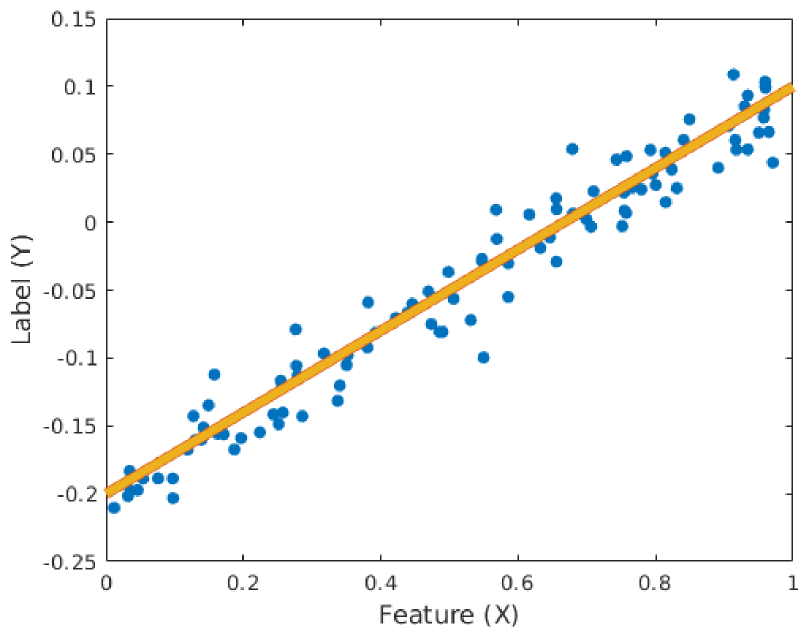

Figure 1.1: Linear regression for a noisy line. A blue dot represents the position of a feature/label pair. The yellow line represents the approximate affine approximation function.

exact solution can be found. An illustration where $X = Y = \mathbb{R}$ is shown in figure 1.1.

In the case of unsupervised learning, we are given only some set of features, $\{x\}_{i=1}^{N}$ and asked to find some pattern in the data. Finding a patterm can consist of finding clusters of data points which are "close" together by some measure, or finding some lower dimensional representation for the data, essentially a problem of lossy compression. Usually algorithms work by minimizing some loss function so that techniques can be borrowed from supervised learning, though these are not necessarily measures of the algorithm's success.

One well known example of unsupervised learning is principal component analysis and its cousin singular value decomposition. Given some features in the form of a matrix X, singular value decomposition can find a matrix, $\hat{X}$ of rank $r < rank(X)$ such that $||X - \hat{X}||_F$ is minimzed. This has the effect of finding a lower dimensional representation of X which contains as much information as possible and therefore tells us which dimensions are "important." One example is shown in figure 1.2 uses singular value decomposition to remove noise from an

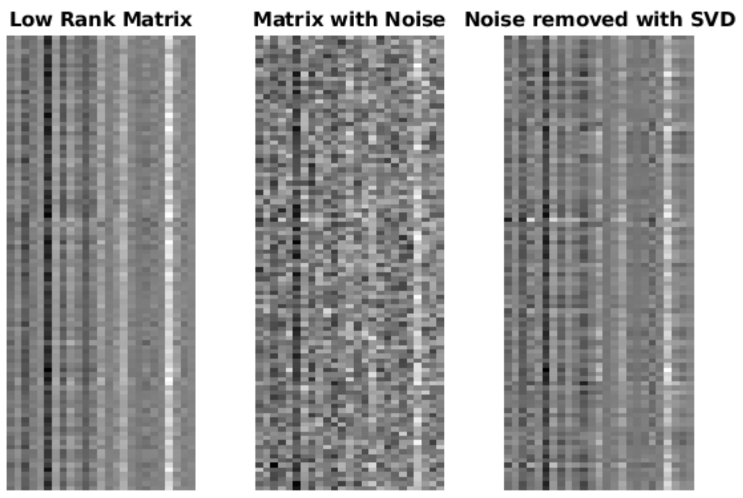

Figure 1.2: Illustration of using SVD to recover an underlying signal. This is an example of unsupervsied learning.

assumed low rank matrix. In figure 1.3, a plot shows the relative contribution of automatically detected signal components.

1.1.1 Hyperparameters

Hyperparameters are parameters of a function which do not change during the "learning process", that is, they are set by the user at the beginning, and the regular parameters of the function are determined with the hyperparameters held constant. In the example of using singular value decomposition to denoise a low rank matrix, the hyperparameter would be the desired rank of the resulting matrix.

We can see from figure 1.4 that this hyperparameter could be calculated, or at least guessed from the number of dominant singular values. However, it may be impossible to know what the underlying rank was as in the case of figure 1.3 if the original matrix does not match our assumptions or if noise dominates the signal. In most modern machine learning models, it is very difficult to efficiently determine optimal hyperparameters, often requiring a brute force search over several combinations. This technique is known as grid search, which searches over n hyperparameters on an n dimensional grid and chooses the hyperparameters that minimize some loss function, not neccesarily the same as the model's loss

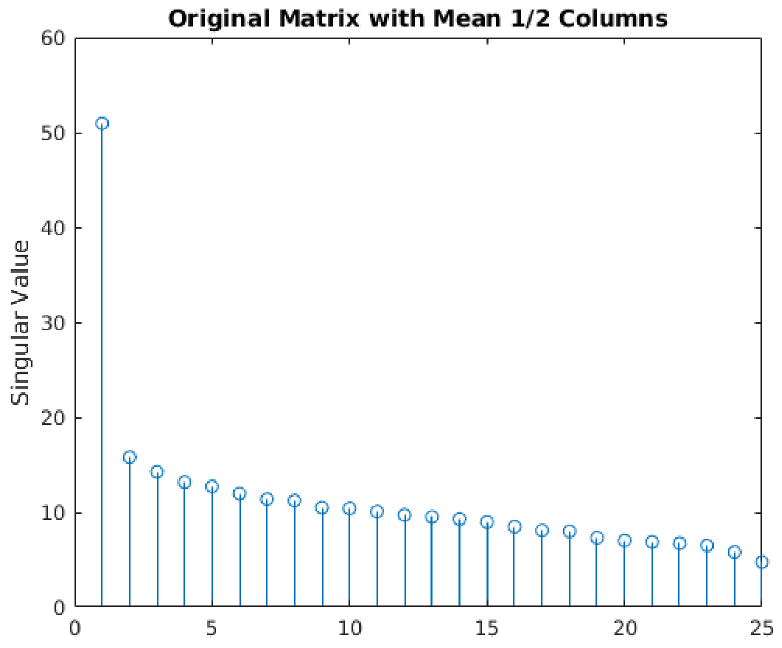

Figure 1.3: Set of 25 singular value for the low rank matrix with noise in figure 1.2. Each column of the original matrix has a mean of about 0.5. Each singular value is a measure of some signal component present in the matrix.

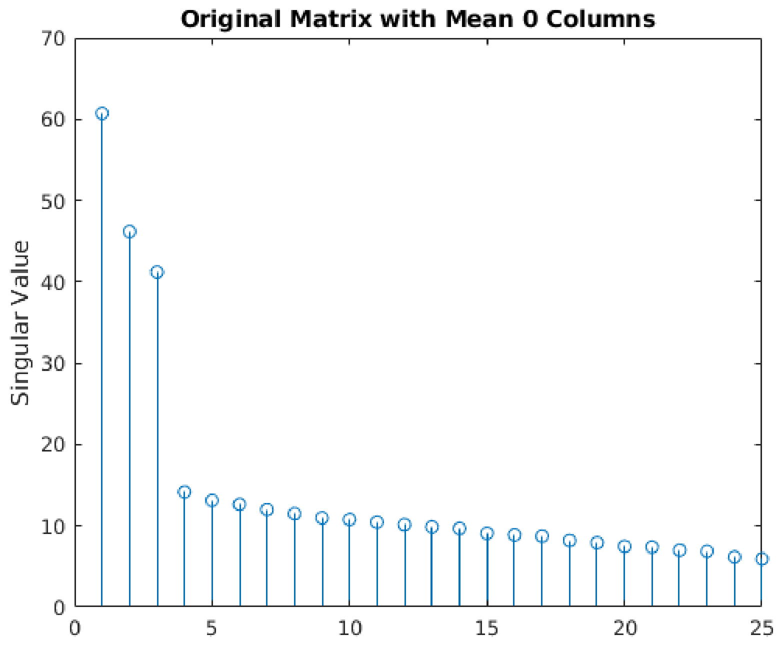

Figure 1.4: Set of 25 singular value for a low rank matrix, each column of the original matrix has mean 0. Here we see three distinct which tells us that the origin rank was very likely three.

function. The process of choosing optimal hyperparameters is known as hyperparameter tuning.

1.1.2 Overfitting

If we allow our model, or set of permissible functions, to be more complex, we can represent more complicated functions. Figure 1.5 shows an example of both a quadratic fit and a 20^{th} order polynomial fit to data points with a more complicated pattern. These two images represent the issue of overfitting: while the underlying curve is in fact quadratic, the higher order polynomial in fact achieves a lower error. It is often true, especially in simpler cases, that increasing the model complexity will reduce the error of the objective. However if our goal is to determine the actual underlying pattern of the data, we may want to choose something simpler.

Regularization is the technique of introducing information to a machine learning problem to reduce overfitting. One of the most common forms of regularization is called Tikhanov regularization. It is also known by the names of ridge regression in statistics and weight decay in machine learning. In linear regression, Tikhanov regularization penalizes a function according to the magnitude squared of each coefficient. By the Riesz representation theorem, for any linear function, $f : \mathbb{R}^m \to \mathbb{R}$, $\exists v \in \mathbb{R}^m$ s.t. $f(x) = \langle x, v \rangle$ for some vector $v \in \mathbb{R}^m$. Suppose v is the vector corresponding to linear function g in this manner, then for linear regression the new loss function is given in equation 1.2

$$L(g, x, y) = ||g(x_i) - y_i||^2 + \lambda ||v||_2^2 \tag{1.2}$$

where λ is a real scalar hyperparameter that may be tuned in the manner discussed in section 1.1.2. While there is no efficient method for computing the optimal value for λ in general, it has a simple value given a certain assumption. If we assume a Gaussian distribution with 0-mean and λ^{-1} variance for the prior distribution of the vector v, then the result of the maximum a posteriori (MAP) estimation is

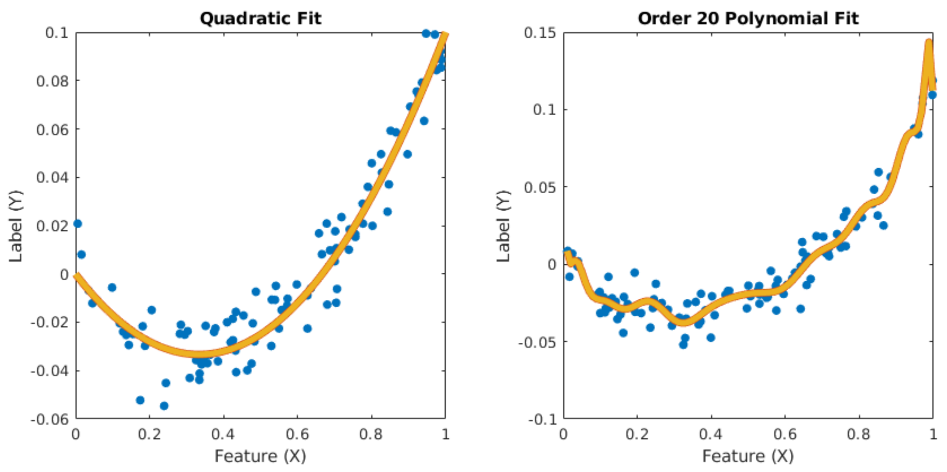

Figure 1.5: The figure on the left is the result of an optimization constrained to second order polynomials while the figure on the right is the result of a 20^{th} order constraint. The second order constraint achieves a result closer to the underlying curve.

the same as the solution to equation 1.2.

1.1.3 Training, Validation, and Testing

Using the loss function over all data points is not a suitable measure of the success of a machine learning algorithm. As we discussed in section 1.1.2, models which fit the underlying function worse may easily achieve a better overall loss. In fact for any finite dataset that could feasible represent a function, it is easy to find a function in the form of a hash table that can exactly represent the mapping of inputs to outputs. Of course, this hash table would not “generalize” to other data points and wouldn’t represent the underlying function in a meaningful way.

To help mitigate this issue, it is standard to separate the data into three parts: training, validation, and testing. Essentially, training is used to tune regular parameters, validation is used to tune hyperparameters, and testing is used only to evaluate the effectiveness of the algorithm.