2 Linear Regression

In regression, our goal is to learn a mapping from one real-valued space to another. After learning, we can use the regression model to predict an output given some new test input. *Linear regression* is the simplest form of regression. It is easy to understand, often quite effective, and very efficient to learn and use.

2.1 The 1D Case

We will start by considering linear regression with one dimensional inputs and outputs. Our goal is to learn a mapping $y = f(x)$, where x and y are both real-valued scalars (i.e., $x \in \mathbb{R}, y \in \mathbb{R}$). We will take f to be a linear function,

$$y = wx + b\ , \tag{2.1}$$

where w is a *weight* and b is a *bias*. These two scalars are the parameters of the model, which we would like to learn from training data. In particular, we wish to estimate w and b from the N training pairs $\{(x_i, y_i)\}_{i=1}^N$. Then, once we have values for w and b, we can compute (or predict) outputs y for some new inputs x.

Given two training data points (i.e., N=2), we can exactly solve for the unknown slope w and offset b. (How would you formulate this solution?) Unfortunately, this approach is extremely sensitive to noise in the training data measurements, so you cannot usually trust the resulting model. Instead, we can find much better models when the two parameters are estimated from larger data sets. When $N > 2$ we will not be able to find unique parameter values for which $y_i = wx_i + b$ for all i, since we have many more constraints than parameters. That is, given noisy data it is unlikely that 3 or more points will lie exactly on some line. Rather the best we can hope for is to find a line that is as close to the training points as possible. To this end, we want to minimize the errors between the data points and our hypothetical linear model, i.e., $y_i - (wx_i + b)$.

The most common way to estimate the parameters of a line in a problem like this is by *least-squares regression*. In least-squares regression the quality of the fit between the model and the training data is specified in terms of the squared error (a.k.a. the squared loss). In more detail, for the ith training point, the model gives a prediction for y, namely $wx_i + b$. So let's define the error for the ith training point, given the model line with parameters w and b, to be the (vertical) distance from the line to the training point:

$$e_i \ = \ y_i - (wx_i + b)\ . \tag{2.2}$$

The sum of the squared errors over all training points is then used to measure the quality of the fit. It is often called the *objective function*, *energy function* or *empirical loss*:

$$E(w, b) \ = \ \sum_{i=1}^{N} e_i^2 \ = \ \sum_{i=1}^{N} (y_i - (wx_i + b))^2\ . \tag{2.3}$$

Importantly, by squareing the errors we obtain a function that is non-negative, and zero if and only if all the errors are zero. Further, as shown below, it yields a very straightforward computational solution.

Finding the line that minimizes the squared error is equivalent to solving for the w and b that minimize $E(w,b)$. This can be done by setting the derivatives of E with respect to these parameters to zero and then solving; i.e.,

$$\frac{\partial E}{\partial b} \;=\; -2\sum_i (y_i - (wx_i + b)) \;=\; 0\;. \tag{2.4}$$

Solving for b gives us the estimate

$$b^* \;=\; \frac{\sum_i y_i}{N} - w\frac{\sum_i x_i}{N} \;=\; \hat{y} - w\hat{x}\;, \tag{2.5}$$

where we define $\hat{x}$ and $\hat{y}$ as the averages of the x's and y's, respectively. This equation for b^* still depends on w, but we can nevertheless substitute this estimate for b in the original energy function, and then solve for w. The substitution produces the following energy,

$$E(w,b) \;=\; \sum_i ((y_i - \hat{y}) - w(x_i - \hat{x}))^2\;. \tag{2.6}$$

Then,

$$\frac{\partial E}{\partial w} \;=\; -2\sum_i ((y_i - \hat{y}) - w(x_i - \hat{x}))(x_i - \hat{x})\;, \tag{2.7}$$

and solving $\frac{\partial E}{\partial w} = 0$ yields our estimate:

$$w^* \;=\; \frac{\sum_i (y_i - \hat{y})(x_i - \hat{x})}{\sum_i (x_i - \hat{x})^2}\;. \tag{2.8}$$

The values w^* and b^* are the least-squares estimates for the parameters of the linear regression.

2.2 Multi-Dimensional Inputs

Now, suppose we wish to learn a mapping from D-dimensional inputs to scalar outputs: $\mathbf{x} \in \mathbb{R}^D$, $y \in \mathbb{R}$. To this end we will learn a vector of weights $\mathbf{w}$, so the mapping has the form:[1]

$$f(\mathbf{x}) \;=\; \mathbf{w}^T\mathbf{x} + b \;=\; \sum_{j=1}^{D} w_j x_j + b \quad . \tag{2.9}$$

For convenience, we can fold the bias b into the weight vector, if we augment the input vector $\mathbf{x}$ with an additional 1; i.e., if we define

$$\tilde{\mathbf{w}} = \begin{bmatrix} w_1 \\ \vdots \\ w_D \\ b \end{bmatrix}, \quad \tilde{\mathbf{x}} = \begin{bmatrix} x_1 \\ \vdots \\ x_D \\ 1 \end{bmatrix} \tag{2.10}$$

[1]Above we used subscripts to index the training set, while here we are using the subscript to index the elements of the input and weight vectors. In what follows the context should make it clear what the index denotes.

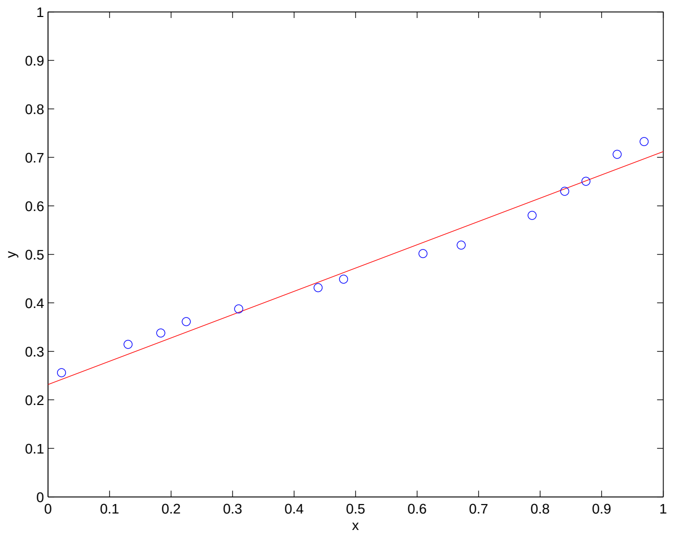

Figure 2.1: An example of linear regression: The red line is fit to the blue data points by minimizing the sum of squared errors. The error of a data point is defined to be the vertical distance from the point to the model line.

then the mapping can be written:

$$f(\mathbf{x}) \;=\; \tilde{\mathbf{w}}^T \tilde{\mathbf{x}} \,. \tag{2.11}$$

Given N training input-output pairs, the least-squares objective function is

$$E(\tilde{\mathbf{w}}) \;=\; \sum_{i=1}^{N} (y_i - \tilde{\mathbf{w}}^T \tilde{\mathbf{x}}_i)^2 \,. \tag{2.12}$$

If we stack the outputs in a vector and the inputs in a matrix, then we can also write this as

$$E(\tilde{\mathbf{w}}) \;=\; ||\mathbf{y} - \tilde{\mathbf{X}}\tilde{\mathbf{w}}||^2 \,, \tag{2.13}$$

where

$$\mathbf{y} \;=\; \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}, \quad \tilde{\mathbf{X}} \;=\; \begin{bmatrix} \mathbf{x}_1^T & 1 \\ \vdots & \\ \mathbf{x}_N^T & 1 \end{bmatrix} \tag{2.14}$$

and $|| \cdot ||$ is the usual Euclidean norm, i.e., $||\mathbf{v}||^2 = \sum_i v_i^2$. (You should verify for yourself that Equations (2.12) and (2.13) are equivalent).

Equation (2.13) is known as a linear least-squares problem, and can be solved by methods from linear algebra. We can rewrite the objective function as:

$$\begin{aligned} E(\mathbf{w}) &= (\mathbf{y} - \tilde{\mathbf{X}}\tilde{\mathbf{w}})^T (\mathbf{y} - \tilde{\mathbf{X}}\tilde{\mathbf{w}}) \\ &= \tilde{\mathbf{w}}^T \tilde{\mathbf{X}}^T \tilde{\mathbf{X}} \tilde{\mathbf{w}} - 2\mathbf{y}^T \tilde{\mathbf{X}} \tilde{\mathbf{w}} + \mathbf{y}^T \mathbf{y} \end{aligned} \tag{2.15}$$

We can optimize this by setting all values of $\partial E/\partial w_i = 0$ and solving the resulting system of equations.[2] With some work, the solution can be shown to be

$$\mathbf{w}^* \;=\; (\tilde{\mathbf{X}}^T\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^T\mathbf{y} \tag{2.16}$$

(You may wish to verify for yourself that this reduces to the solution for the 1D case in Section 2.1; however, this takes quite a lot of linear algebra and a little cleverness). The matrix $\tilde{\mathbf{X}}^+ \equiv (\tilde{\mathbf{X}}^T\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^T$ is called the *pseudoinverse* of $\tilde{\mathbf{X}}$, and so the solution can also be written:

$$\tilde{\mathbf{w}}^* \;=\; \tilde{\mathbf{X}}^+\mathbf{y} \tag{2.17}$$

In MATLAB, one can directly solve the system of equations using the slash operator:

$$\tilde{\mathbf{w}}^* = \tilde{\mathbf{X}}\backslash\mathbf{y} \tag{2.18}$$

There are some subtle differences between these two ways of solving the system of equations. We will not concern ourselves with these here except to say that we recommend using the slash operator rather than the pseudoinverse.

2.3 Multi-Dimensional Outputs

In the most general case, both the inputs and outputs may be multi-dimensional. For instance, one might want to predict expected crop yields of several different crops, as a function of different growing conditions, such as soil concentrations, temperatures and precipitation. With D-dimensional inputs, and K-dimensional outputs $\mathbf{y} \in \mathbb{R}^K$, a linear mapping from input to output can be written as

$$\mathbf{y} \;=\; \tilde{\mathbf{W}}^T\tilde{\mathbf{x}} \tag{2.19}$$

where $\tilde{\mathbf{W}} \in \mathbb{R}^{(D+1)\times K}$. It is convenient to express $\tilde{\mathbf{W}}$ in terms of its column vectors, i.e.,

$$\tilde{\mathbf{W}} \;=\; [\tilde{\mathbf{w}}_1 \,\ldots\, \tilde{\mathbf{w}}_K] \;\equiv\; \begin{bmatrix} \mathbf{w}_1 & \ldots & \mathbf{w}_K \\ b_1 & \ldots & b_K \end{bmatrix} . \tag{2.20}$$

In this way, we then express the mapping from the input $\tilde{\mathbf{x}}$ to the j_{th} element of $\mathbf{y}$ as $y_j = \tilde{\mathbf{w}}_j^T\mathbf{x}$. Now, given N training samples, denoted $\{\tilde{\mathbf{x}}_i, \mathbf{y}_i\}_{i=1}^N$ a natural energy function to minimize, to estimate $\tilde{\mathbf{W}}$, is just the squared residual error over all training samples and all output dimensions, i.e.,

$$E(\tilde{\mathbf{W}}) \;=\; \sum_{i=1}^N\sum_{j=1}^K (y_{i,j} - \tilde{\mathbf{w}}_j^T\tilde{\mathbf{x}}_i)^2 . \tag{2.21}$$

There are several ways to conveniently *vectorize* this energy function. One is to express E solely as a sum over output dimensions. That is, let $\mathbf{y}'_j$ be the N-dimensional vector comprising the j^{th} component of each output training vector, i.e., $\mathbf{y}'_j = [y_{1,j}, y_{2,j}, ..., y_{N,j}]^T$. Then we can write

$$E(\tilde{\mathbf{W}}) \;=\; \sum_{j=1}^K ||\mathbf{y}'_j - \tilde{\mathbf{X}}\tilde{\mathbf{w}}_j||^2 \tag{2.22}$$

[2] We will cover this in more detail later in Chapter **??**. In the meantime, if this is unclear, start by reviewing your linear algebra and vector calculus.

where $\tilde{\mathbf{X}}^T = [\tilde{\mathbf{x}}_1 \, \tilde{\mathbf{x}}_2 \, \ldots \, \tilde{\mathbf{x}}_N]$. With a little thought you can see that this really amounts to K distinct estimation problems, the solutions for which are given by $\tilde{\mathbf{w}}_j^* = \tilde{\mathbf{X}}^+ \mathbf{y}'_j$.

Another common convention is to stack up everything into a matrix equation, i.e.,

$$E(\tilde{\mathbf{W}}) = ||\mathbf{Y} - \tilde{\mathbf{X}}\tilde{\mathbf{W}}||_F^2 \tag{2.23}$$

where $\mathbf{Y} = [\mathbf{y}'_1 \, \ldots \mathbf{y}'_K]$, and $||\cdot||_F$ denotes the Frobenius norm: $||\mathbf{Y}||_F^2 = \sum_{i,j} \mathbf{Y}_{i,j}^2$. The Frobenius norm can also be expressed in terms of the matrix trace, i.e., $||\mathbf{Y}||_F^2 = \text{tr}(\mathbf{Y}^T\mathbf{Y})$. You should verify that Equations (2.22) and (2.23) are equivalent representations of the energy function in Equation (2.21). Finally, the solution, now simultaneously for all K regressors, is again provided by the pseudoinverse:

$$\tilde{\mathbf{W}}^* = \tilde{\mathbf{X}}^+ \mathbf{Y} \tag{2.24}$$