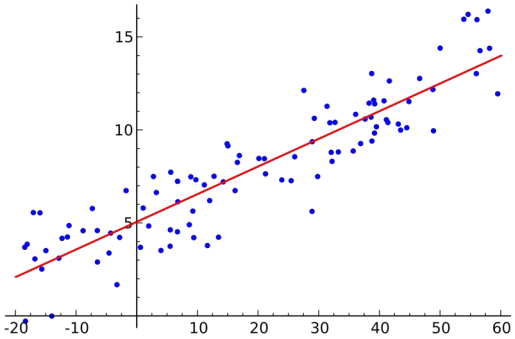
Linear Regression

The art of regression is the exact opposite of plotting a graph given its equation. As you guessed rightly, in regression we tend to get the best fit curve equation given the data points.



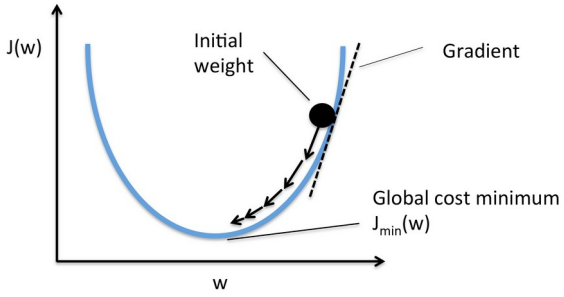
In linear regression, we tend to fit a simple line or a hyperplane (if we have more than one feature, or more formally more than one dimension) for the data points. As this technique falls into supervised learning, we tend to penalize the model according the how off the value predicted by it is from the actual value and use that correct the model.

The error function which we declare to penalise the model is called squared error and is defined as $\sum_{1}^{N} \frac{1}{2}(Y_i - Y_{pi})^2$ here Y is the ground truth (or actual value to be predicted) and Y_p is the value our model predicts. One may get a very obvious question as why the error has been defined so and not as a simple difference of Y and Y_p. The answer to that is quite simple. Consider a case where two points lie symmetrically above and below our predicted line. According to a simple difference, the total error constituted by both the points is zero, while it is actually not so, and hence we need a need a more robust definition of the error function which does not involve a sign but considers only the magnitude. The best way to represent just the magnitude is to use Euclidean Distance, and that is what we have declared. Note that, in our case, we tend to consider the total error constituted by all N-points.

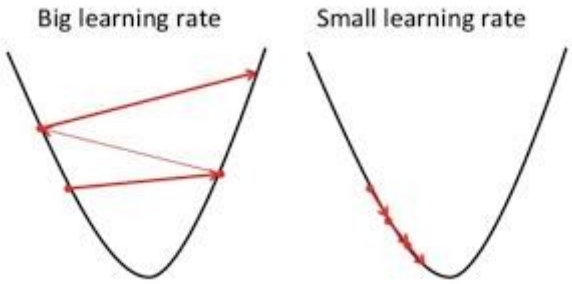
The parametric form of our prediction is as follows: $Y_{pi} = \sum_{0}^{D} W_i X_i$ (assuming that we have D features D_1 to D_{10}). We can easily see that this represents a hyperplane equation. Here we consider each data point to be a (D+1)-tuple and the Ws are called the weights or parameters. The W_0 term is called the bias term and the associated feature X_0 is hard-wired to be 1. So, in a more informal way, $Y_{pi} = W_0 + \sum_{1}^{D} W_i X_i$ Now, we attempt to learn the optimal weights so that our model can give a proper prediction, i.e. to have as minimum total error as possible. As we tend to learn the weights, we can start off with random weights initially and adjust them later. Note that, here the variables are Ws and not Xs.

Now, minimizing the error is a typical optimization problem. We tend to minimize the function $\sum_{1}^{N} \frac{1}{2}\left(Y_i - \sum_{0}^{D} W_i X_i\right)^2$ with respect to all Ws, i.e. find the local minima of this function. The best way is to differentiate it wrt each W_i and set that value to 0. This kind of a error function can be

proved to convex, and hence a single global/local minima exists. The vector of these derivatives actually constitutes the gradient of the error function at that point. Moving along the negative of this gradient will ensure that we reach the minimum [see the picture below. Here the cost function is represented as J(w)]. Once we reach the global minima, we can notice that the gradients become zero and we automatically stop there, or in a better way, get stuck there. Hence, once we reach the global minima, the error gets saturated and can this can be used a termination condition. Since we are moving down gradient every single time, this algorithm is called *Gradient Descent*.



The step-size used for moving down gradient is called learning rate. It is very important to choose a proper value for this hyperparameter. A really small step or learning rate can cause a very slow descent and a very large learning rate can cause the model to bounce rapidly and exit the valley. In a more mathematical sense, gradient descent is simply $W = W - \alpha dW$, where α is the learning rate and dW is the gradient vector.



I leave the intricacies of the mathematics for you to solve. A crazy statement about Regression is that we believe that the output is somehow hidden in the input itself, i.e. as a linear combination of the inputs, and it works! Nature is weird, isn't it?

Examples of a Non-Convex and Convex functions.