Assignment-11 : Linear Regression

Harshvardhan Patidar
Department of Artificial Intelligence
Indian Institute of Technology Hyderabad
ai24btech11015@iith.ac.in

Regularization on Multivariable Linear Functions

The issue of overfitting wasn't very troublesome in the case of univariable linear functions, but in the case of multivariable functions, there can be a dimension that appears to be useful to the model, but is in reality irrelevant. This might lead to overfitting of the model on new data. For this reason, using regularization on multivariable linear functions is more common. In regularization, we minimize a cost function which considers both the loss and the complexity of the function, with a hyperparameter λ which serves as a conversion rate between the complexity and loss.

$$\textit{Cost}(h) = \textit{EmpLoss}(h) + \lambda \textit{Complexity}(h)$$

For linear functions complexity can be defined as a function of the weights.

$$\textit{Complexity}(h_{\mathbf{w}}) = L_q(\mathbf{w}) = \sum_i |w_i|^q$$

Where, $q = 1$ minimizes the L_1 loss function and $q = 2$ minimizes the L_2 loss function. The choice of picking q depends upon the specific problem. But the L_1 regularization has an advantage over the other that it often sets many weights to zero, thus producing a sparse model. This reduces the chances of overfitting. This can be visualized clearly from the below graph. The L_1 function is on the left, like a box. While minimizing, the minimum solution usually comes out to be on one of the corner points, which have some dimensions zero. Whereas the L_2 (in the right graph), the optimal solution is likely to occur anywhere on the circle, thus giving no extra benefit of zero weights.

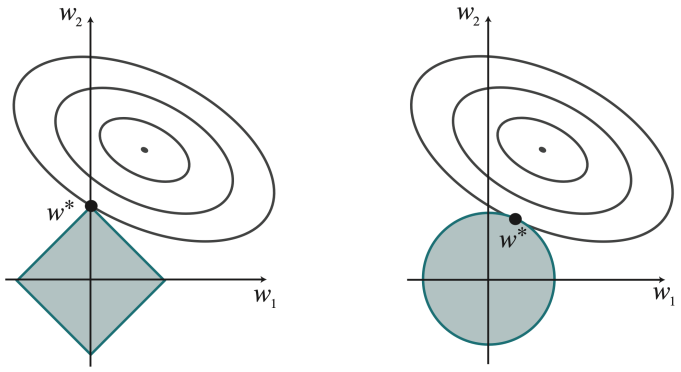


Figure 1: L_1 loss function (left) and the L_2 function (right)

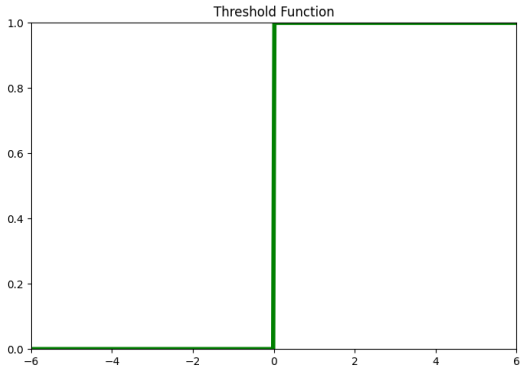
Linear classifiers with a hard threshold

In the case of linear classifiers, we try to classify inputs into two classes. For classification we try to find a decision boundary, which is a line(2D), a plane(3D) or a hyperplane(higher dimensions). Such linear boundaries are called linear separators, and the data which fits into such a type of classification is called linearly separable data. Say we have found a weight vector $\mathbf{w}$, then we can write the classification hypothesis(or function) as

$$h_{\mathbf{w}}(\mathbf{x}) = 1, \text{ if } \mathbf{w} \cdot \mathbf{x} \geq 0 \quad \text{OR} \quad h_{\mathbf{w}}(\mathbf{x}) = 0 \text{ otherwise}$$

To simplify it we can think of h as a result of passing the linear function $\mathbf{w} \cdot \mathbf{x}$ through a Threshold Function.

$$h_{\mathbf{w}}(\mathbf{x}) = \textit{Threshold}(\mathbf{w} \cdot \mathbf{x}) \text{ where } \textit{Threshold}(z) = 1 \text{ if } z \geq 0 \text{ and } 0 \text{ otherwise}$$



For other well defined functions, we could regularize the model by using gradient descent or other methods. But, here since the gradient is zero mostly, or undefined. So that isn't very useful in this case of linear separators. But there is, however, a simpler weight update rule that converges to the linear separator that classifies the data perfectly.

$$w_i \leftarrow w_i + \alpha(y - h_{\mathbf{w}}(\mathbf{x})) \times x_i$$

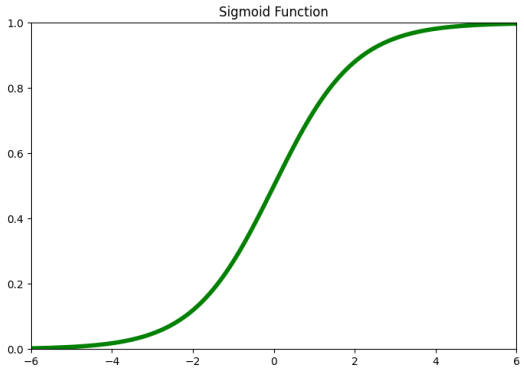
This update rule is very similar to the one for linear regression. It is called the perceptron learning rule. Correct choice of α is also necessary as if α doesnt converge to 0, then the algorithm will not be able to converge to our solution model.

Linear classification with logistic regression

In this method we try to overcome the problem caused by the non-differentiable and non-continuous nature of our hypothesis $h_{\mathbf{w}}(\mathbf{x})$. To achieve this, we can use the logistic function, which is like a softer version of the threshold function. It is also similar to the threshold function in terms of shape.

$$\textit{Logistic}(z) = \frac{1}{1 + e^{-z}}$$

$$h_{\mathbf{w}}(\mathbf{x}) = \textit{Logistic}(\mathbf{w} \cdot \mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w} \cdot \mathbf{x}}}$$



The output of this type of hypothesis can be interpreted as the probability of being labelled to the class which we referred to earlier as 1. It forms a soft boundary between the two classes.

The process of fitting the weights of such a model to find the optimal function is called logistic regression. Here the gradient descent method works straight-forward, because now our hypothesis has a well defined gradient function. In this case, we can use the L_2 loss function. Here, g denotes logistic function and g' is its derivative.

$$\begin{aligned}\frac{\partial}{\partial w_i} Loss(\mathbf{w}) &= \frac{\partial}{\partial w_i}(y - h_{\mathbf{w}}(\mathbf{x}))^2 \\ &= -2(y - h_{\mathbf{w}}(\mathbf{x})) \times g'(\mathbf{w} \cdot \mathbf{x}) \times x_i\end{aligned}$$

The derivative g' of logistic function can be expressed as below

$$g'(\mathbf{w} \cdot \mathbf{x}) = g(\mathbf{w} \cdot \mathbf{x})(1 - g(\mathbf{w} \cdot \mathbf{x})) = h_{\mathbf{w}}(\mathbf{x})(1 - h_{\mathbf{w}}(\mathbf{x}))$$

So, the update rule can be described as taking step in the direction of difference between input and prediction, i.e. $(y - h_{\mathbf{w}}(\mathbf{x}))$, and the magnitude of the step depends upon α and g'. The update rule is :

$$w_i \leftarrow w_i + \alpha(y - h_{\mathbf{w}}(\mathbf{x})) \times h_{\mathbf{w}}(\mathbf{x})(1 - h_{\mathbf{w}}(\mathbf{x})) \times x_i$$

As predicted, the Logistic regression method works better than the threshold function. It either converges to a more predictable function than the threshold one. Or else it converges to an optimal solution much faster than the threshold one. This advantage of logistic regression has made it the most popular classification technique in various fields.