Deep Learning

Linear Regression

Tiago Vieira

Institute of Computing
Universidade Federal de Alagoas

Summary

Introduction

Let's start with a very simple example: Linear Regression.

- ML algorithm: Performance (P) improvement with respect to a given Task (T) via Experience (E).

Introduction

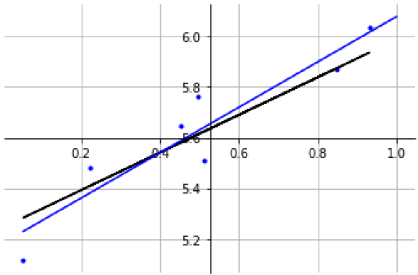

Figure: Example of linear regression. The black solid line represents $f(x)$ corresponding to the real phenomenon generating the data. Blue dots represent the samples obtained after the insertion of the noise. The blue solid line represents the model obtained by linear regression.

Introduction

Linear regression:
Input:

$$\mathbf{x} \in R^n. \tag{1}$$

Output is a linear function of the input:

$$y \in R. \tag{2}$$

That is:

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in \mathcal{R}^n \mapsto y \in \mathcal{R} \tag{3}$$

- Parameters are values that control the behavior of the system.
- w_i are coefficients that are multiplied by feature x_i before adding the contributions of all attributes.
- $[w_i]$ is a set of *weights* determining how much each attribute contributes to the prediction $\hat{y}$.
- $w_i > 0: \uparrow x_i \quad \uparrow \hat{y}$
- $w_i < 0: \uparrow x_i \quad \downarrow \hat{y}$
- $w_i = 0:$ x_i has no influence on $\hat{y}$
- $w_i \gg 0:$ x_i has a large influence on $\hat{y}$

Problem statement:

- Task T: Predict y from a given input $\mathbf{x}$ by computing $\hat{y} = \mathbf{w}^T\mathbf{x}$.

Suppose:

- Design matrix containing m examples:

$$X^{test} = \left[- - \mathbf{x}^{(i)} - - \right]_{i=1}^{m} \tag{4}$$

- Vector of regression targets:

$$\mathbf{y}^{(test)}. \tag{5}$$

Performance (P) is given by the Mean Squared Error (MSE) computed on the model's predictions on test examples $\mathbf{y}^{(test)}$:

$$MSE_{test} = \frac{1}{m} \sum_{i=1}^{m} \left(\hat{y}^{(test)} - y^{(test)} \right)_i^2 \tag{6}$$

- One can see that the error is zero for $y = \hat{y}$.

Also:

$$MSE_{test} = \frac{1}{m} ||\hat{y}^{(test)} - y||_i^2 \tag{7}$$

so that the error increases whenever the euclidean distance between predictions and ground-truth targets increases.

ML algorithm's goal:

- Improve weights **w** in order to reduce the error $MSE_{(test)}$.
- But the algorithm can "learn" (gain experience) through observations in training dataset $(\mathbf{x}^{(i)}, y^{(i)})_{i=1}^{m}$.

How do we do that?

- Minimizing the mean squared error using the gradient and making it equal to zero:

$$\nabla_{\mathbf{w}} MSE_{(train)} = 0 \tag{8}$$

$$\nabla_{\mathbf{w}} \frac{1}{m} ||\hat{y}^{(train)} - y^{(train)}||_2^2 = 0$$

$$\frac{1}{m} \nabla_{\mathbf{w}} ||X_{(m,n)}^{(train)} \mathbf{w}_{(n,1)} - \mathbf{y}_{(m,1)}^{(train)}||_2^2$$

Solving for **w** gives:

$$\boxed{\mathbf{w} = \left(X^T X\right)^{-1} X^T \mathbf{y}} \tag{9}$$

- Known as the *normal equations*.
- Simple learning algorithm.

Thank you!

tvieira@ic.ufal.br