AIFS ML Lecture 5: Machine Learning Basics

Suraj Narayanan Sasikumar

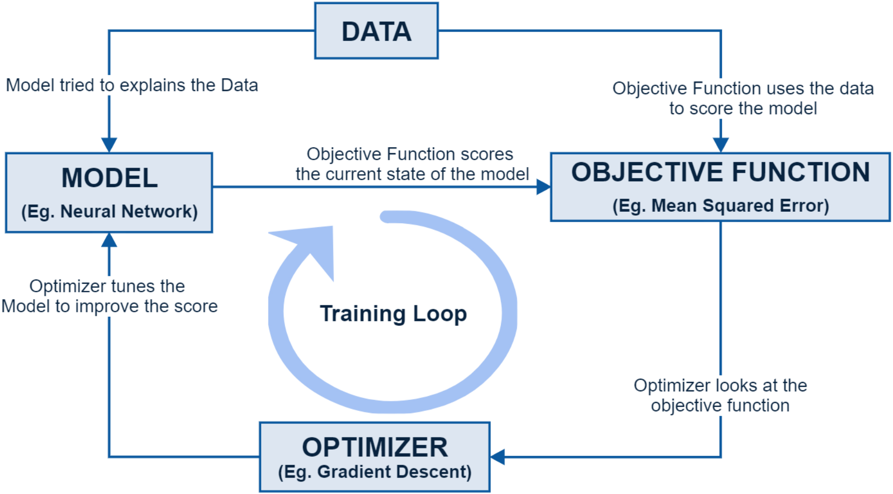
Hessian AI Labs

May 03, 2020

Overview

Table of Contents

Recap - Learning Algorithm



Recap - Linear Regression

- Linear Regression is a machine learning algorithm that finds the best line (or planes/hyper-planes in higher dimensions) that fits the data.
- The goal of linear regression is to find the model that best predicts output label, y, given new input vector, $\mathbf{x}$, using the dataset $\boldsymbol{X}$
- Supervised and Batch Learning style, and uses a parameterized model.
- Components of the Linear Regression algorithm
  - **Data:** $\boldsymbol{X}, \mathbf{y}$
  - **Model:** $\hat{y} = \mathbf{w}^T \mathbf{x}$ (y-intercept included in the dot-product, $x_0 = 1$)
  - **Loss Function:** Minimize Mean Squared Error

$$\text{MSE}(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i)^2$$

  - **Optimizer:** Closed Form Solution obtained by setting gradient of MSE to 0, $\nabla_{\mathbf{w}} \text{MSE}(\mathbf{w}) = 0$

$$\mathbf{w}^* = (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \mathbf{y}$$

Table of Contents

Polynomial Regression

- The goal to find the best polynomial that explains the data, and has good prediction capability.
- Components of the algorithm
  - **Data:** $\boldsymbol{X}, \mathbf{y}$ ($\boldsymbol{X}$ has only one feature, i.e. $(D = 1)$)
  - **Model:** $\hat{y} = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + \ldots + w_M x^M$
  - **Loss Function:** Minimize Mean Squared Error

$$\text{MSE}(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i)^2$$

  - **Optimizer:** Closed Form Solution obtained by setting gradient of MSE to 0, $\nabla_{\mathbf{w}} \text{MSE}(\mathbf{w}) = 0$

$$\mathbf{w}^* = (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \mathbf{y}$$

- When compared to Basic Linear Regression, the only component that has changed is the Model.

Polynomial Regression: Model

- As the name suggests, the model is a polynomial of degree M.

$$\hat{y} = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + \ldots + w_M x^M$$

- Even though the model is now non-linear in the inputs, it is still linear in the parameters.
- Converts a 1-D input, to an M-D feature space

$$\boldsymbol{X} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_N \end{bmatrix}_{N\times 1} \quad \Rightarrow \quad \boldsymbol{X} = \begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 & \ldots & x_1^M \\ 1 & x_2 & x_2^2 & x_2^3 & \ldots & x_2^M \\ 1 & x_3 & x_3^2 & x_3^3 & \ldots & x_3^M \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_N & x_N^2 & x_N^3 & \ldots & x_N^M \end{bmatrix}_{N\times M}$$

- But how do we choose M?

Hyperparameters

- Hyperparameters are a configurable aspect of the learning algorithm that is set before training and whose value has an impact on the performance, speed, and generalization capability of the learning algorithm.
- The choosing of the values of hyperparameter, referred to as hyperparameter tuning, is more of an *art* than a science.
- Hyperparameters can be classified into two categories:
  - **Model Hyperparameters:** There are aspects of the model that can be pre-configured to improve the performance of the model.
  - **Algorithm Hyperparameters:** They are parameters belonging to other aspects of the learning algorithm like, optimizer (learning-rate), loss-function (regularization-parameter) etc.
- Advanced methods like *Bayesian Optimization*, *Neural Architecture Search*, etc. exists for auto-tuning, but their applications are still limited.

Polynomial Regression: Choosing M

- M is a model-hyperparameter that we need to choose.
- The choice of M determines the **capacity** of our model.
- A model that can be adapted (by varying it's adaptable parameters) to a large number of functions is said to have **high capacity**. Similarly, a model that can only be adapted to a small number of function is said to have **low capacity**.
- When M is small, the model has low capacity since the model is not capable enough to explain data that could have been generated by higher-order polynomials. Here we say the model is *Underfitting*.
- If M is too large, then the model becomes so powerful that the model tries to explain the noise in the data as well. This leads to a problem known as *Overfitting*
- The value of M needs to be just right so as the capacity of the model is just right for the complexity of the task at hand.

Synthetic Data: $sin(2\pi x)$

- In order to explain the concepts of underfitting, and overfitting, we'll create a synthetic (generated) dataset.
- The **true** data-generating process is

$$y = sin(2\pi x)$$

,

- We add Gaussian noise $\mathcal{N}(0, 0.3)$ to each datapoint in order to mimic real-world datasets.
- The Taylor series expansion of $sin(x)$ can be written as:

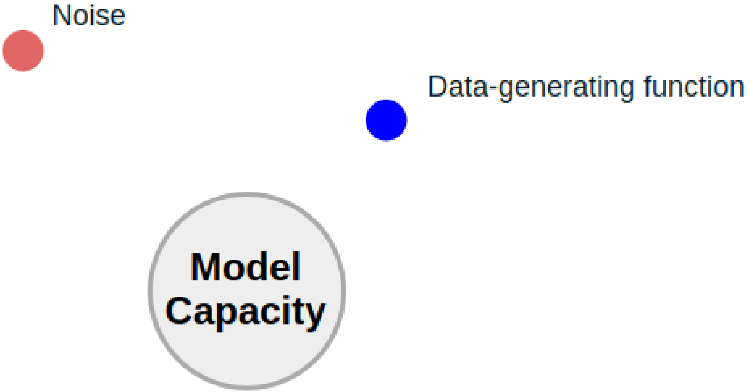
$$sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots$$

Which can be re-written as,

$$sin(x) = 0 + (1)x + (0)x^2 + \left(\frac{-1}{3!}\right)x^3 + (0)x^4 + \left(\frac{1}{5!}\right)x^5 + (0)x^6 + \left(\frac{-1}{7!}\right)x^7 + \dots$$

Underfitting

- Underfitting occurs when the model does not have enough capacity to explain the data. Unable to reduce the training loss sufficiently.
- The data-generating function is not in the class of function that the model can express.



Underfitting

- Suppose we set $M = 1$ in our polynomial regression, the model becomes

$$\hat{y} = w_0 + w_1 x$$

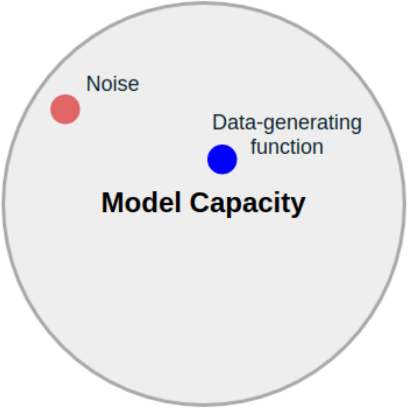
- This cannot effectively model:

$$sin(x) = 0 + (1)x + (0)x^2 + \left(\frac{-1}{3!}\right)x^3 + (0)x^4 + \left(\frac{1}{5!}\right)x^5 + (0)x^6 + \dots$$

- We say that the above model does not have enough capacity to explain the data, ie model the data-generating process.

Overfitting

- Overfitting occurs when the capacity of a model is much more than what is required to model the data-generating process.
- In this situation the excess capacity tries to fit to the noise present in the data, ie the model tries to explain the noise as well.



Overfitting

- Suppose we set $M = 9$ in our polynomial regression, the model becomes,

$$\hat{y} = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + \ldots + w_9 x^9$$

- For a decent approximate of sin(x) locally, only a few higher-order terms are required. It is obvious that this model can model $sin(x)$ pretty well.

$$sin(x) = 0 + (1)x + (0)x^2 + \left(\frac{-1}{3!}\right)x^3 + (0)x^4 + \left(\frac{1}{5!}\right)x^5 + (0)x^6 + \ldots$$

- But, due to its increased capacity, what it can also do is fit to the noise present in the data.
- The model rather than learning a sine function, tries to ensure that the polynomial perfectly fits every data-point. Thus, successfully explaining the noise in the data.

Just Right

- To get the fit just right we have to ensure that the capacity of the model is suitable for the complexity of the task and the amount of data available.