Machine Learning Demolished

Linear Regression

The goal of Regression is to predict the value of one or more continuous target variables **t** given the value of a D-dimensional vector x of input variables.

- Training Data: X = $\{\mathbf{x_1}, \mathbf{x_2}, \dots, \mathbf{x_N}\}$
  $\hookrightarrow$ N training examples.

- Response / Target: $\{t_n\}$, $n = 1, 2, \dots, N$
  $\hookrightarrow$ vector **t** (of dimension N)

The D-dimensional vectors $\mathbf{x_i}, i = 1, \dots, N$ are know as variables and they can come from different sources:

- quantitative inputs (e.g. Income, Age, etc.)
- basis function, e.g. $\phi_j = \mathbf{x}^j$ (polynomial regression)
- numeric or "dummy" encodings of qualitative inputs
- interactions between variables, e.g. $\mathbf{x_1} = \mathbf{x_2} \cdot \mathbf{x_3}$

and we assume that these data points are drawn independently from the population distribution.

The simplest form of Linear Regression model is linear functions of the input variables.

$$\text{Simplest Formula: } y(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x_1 + \cdots + w_D x_D$$
$$\hookrightarrow \mathbf{x} = (x_1, \dots, x_D)^T$$

> ⚠️ This formual has the following properties:
>
> 1. linear function of the parameters, $w_1, \dots, w_D$
> 2. linear function of the input variables, $x_1, \dots, x_D$

In order to remove *limitation number 2*, we introduce the **basis functions** so that the simplest formula is *extended* to:

$$\text{Basis Formula: } y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(x) =$$
$$\hookrightarrow \boldsymbol{\phi_j} \text{ : basis functions}$$
$$= \sum_{j=0}^{M-1} w_j \phi_j = \boldsymbol{w}^T \boldsymbol{\phi}(\mathbf{x}) \to \boldsymbol{\phi} = (\phi_0, \dots, \phi_{M-1})$$
$$\hookrightarrow \boldsymbol{w} = (w_o, \dots, w_{M-1})^T$$

> ⚠️ The basis function can be fixed non-linear functions of the input variables $\mathbf{x_i}$
> so that the basis formula follows the properties:
>
> 1. linear function of the parameters, $w_1, \dots, w_D$

Assumption 1: The linear regression formula is a linear function of the parameters $w_1, \cdots, w_D$

To use a model for prediction, we need to derive the weight parameter $\mathbf{w}$. To do this, we have to define a loss function to minimize.

Typical Loss/Error Functions

1. **Sum of Squares:** $E_D(w) = \frac{1}{2} \sum_{n=1}^{N} (t_n - \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x_n}))^2$
2. **Absolute Error:** $E_D(w) = \frac{1}{2} \sum_{n=1}^{N} |t_n - \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x_n})|$

> ⚠️ The most common loss function is the **Sum of Squares**.
> These are some of the reasons why:
>
> 1.
> **Sum of Squares** can be motivated as the **Maximum Likelihood Solution** under an assumed **Gaussian noise n**

> 2. Squared differences have the nice mathematical properties; continuously differentiable which is convenient when try
> 3. Sum of Squares is a **convex function** which mean that the local minimum=global minimum.

In the context of Machine Learning, selecting a Loss function to minimize is more than enough considering that the only interest is to “fit” a line into some data. In other words, minimizing the Sum of Squares is a mathematical minimization problem with no assumptions made for the distribution of the data. To ensure that our $\mathbf{w}$ estimate is unbiased, we need to extend our assumptions about the data.

As a reminder, we made the assumption that a linear function can be adequately approximated by a linear function. In other words, we hope that

$$E(t|x) \approx y(\mathbf{x}, \mathbf{w})$$

holds true and is a reasonable approximation.
We can then safely write:

$$\boldsymbol{t} = y(\mathbf{x}, \mathbf{w}) + \boldsymbol{\epsilon}$$

where,

1. $\mathbf{t} \longrightarrow$ target variable
2. $y(\mathbf{x}, \mathbf{w}) \longrightarrow$ deterministic function
3. $\boldsymbol{\epsilon} \longrightarrow \mathbb{N}(0, \sigma^2)$
   $\hookrightarrow$ residuals (estimation of the error)

Assumption 2:
The residuals are normally distributed with mean=0
Note: This is called "Normality" of the residuals.

Assumption 3:
The residuals have constant variance for every input of the data$\mathbf{x_n}, n = 1, \dots, N$
Note: This is known as "homoscedasticity".

Assumption 4:
The residuals are not correlated with each other, i.e. not auto-correlated.
Auto-correlation takes place when there is a pattern in the rows fo the data (e.g. time-series).

Our goal is to estimate the $\boldsymbol{w}$ parameters that minimize the selected **Loss Function**; in our case the Sum of Squares:

$$E_D(w) = RSS = \frac{1}{2}\sum_{n=1}^{N}(t_n - \mathbf{w}^T\boldsymbol{\phi}(\mathbf{x_n}))^2$$

$\hookrightarrow$ Residual Sum of Squares

In mathematics. to find the minimum of a function, we have to set the derivative of a function to 0. Therefore:

$$\begin{aligned} 0 = \triangledown E_D(\boldsymbol{w}) &= \frac{1}{2}\cdot 2\sum_{n=1}^{N}(\boldsymbol{t}_n - \boldsymbol{w}^T\boldsymbol{\phi}(\mathbf{x_n})(\boldsymbol{t_n} - \boldsymbol{w^T}\boldsymbol{\phi}(\mathbf{x_n}))' \\ &= \sum_{n=1}^{N}(\boldsymbol{t}_n - \boldsymbol{w}^T\boldsymbol{\phi}(\mathbf{x_n}))\boldsymbol{\phi}(\mathbf{x_n})^{\boldsymbol{T}} \\ &= \sum_{n=1}^{N}(\boldsymbol{t}_n\boldsymbol{\phi}(\mathbf{x}_n)^{\boldsymbol{T}} - \boldsymbol{w}^T\boldsymbol{\phi}(\mathbf{x}_n)^{\boldsymbol{T}}) \end{aligned}$$

Converting the equation above into using Matrix notation, we get:

$$\begin{aligned} &0 = \boldsymbol{\Phi^T} - \boldsymbol{\Phi^T}\boldsymbol{\Phi}\boldsymbol{w} \Leftrightarrow \\ &\Leftrightarrow \boldsymbol{\Phi^T}\boldsymbol{\Phi}\boldsymbol{w} = \boldsymbol{\Phi}\boldsymbol{t} \Leftrightarrow \\ &\Leftrightarrow \hat{\boldsymbol{w}} = (\boldsymbol{\Phi^T}\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi^T}\boldsymbol{t} \qquad \longrightarrow \textbf{Normal equations} \end{aligned}$$

where $\mathbf{\Phi}$ is called the *design matrix* $\mathbf{\Phi} = \begin{pmatrix} \phi_o(x_1) & \phi_1(x_1) & \cdots & \phi_d(x_1) \\ \phi_o(x_2) & \phi_1(x_2) & \cdots & \phi_d(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_o(x_N) & \phi_1(x_N) & \cdots & \phi_d(x_N) \end{pmatrix}$

> ⚠ Note:
>
> 1. $\boldsymbol{\phi} = (\phi_0, \dots, \phi_D)^T$
> 2. Φ is a NxD matrix.

Assumption 5:
For the $(\mathbf{\Phi}^T\mathbf{\Phi})^{-1}$ we need to assume that $\mathbf{\Phi}$ is of full rank, i.e. the independent variables are not correlated (e.g. $\phi_1 = 3\phi_3$.
Note: This is know as no multicollinearity.

Model Evaluation

Metric	Syntax	Advantages	Disadvantages
Mean Squared Error (MSE)	$\frac{1}{N}\sum_{n=1}^{N}(t_i - \hat{t}_i)$	Differentiable so can be used as a Loss Function	Not robust to outliers as it penalizes them to the power of 2
Mean Absolute Error (MAE)	$\frac{1}{N}\sum_{n=1}^{N}\lVert t_i - \hat{t}_i\rVert$	More robust to outliers	Not differentiable which needs to the application of optimisers such as Gradient Descent
Root Mean Squared Error (RMSE)	$RMSE = \sqrt{MSE}$	Output is at the same unit as the input (interpretation usefullness)	Not that robust to outliers

- $\textbf{R-Squared}(R^2) \rightarrow$ Not a performance metric
  $\hookrightarrow$ Coefficient of Determinition
  $\hookrightarrow$ Goodness of Fit

- **Description:** R^2 measures how much variance can be explained by your model. R^2 can also be viewed as how much the regression line is better than the mean line.

- **Formula:** $R^2 = 1 - \frac{\text{Unexplained Variance}}{\text{Total Variation}} = 1 - \frac{SS_{reg}}{SS_{mean}} = 1 - \frac{\sum_{i=1}^{N}(t_i - \hat{t}_i)^2}{\sum_{i=1}^{N}(t_i - \overline{t_i})}$
  $\hookrightarrow$ mean of target variable

- **Value's Range:** From 0 (bad model) to 1 (perfect model)

> ⚠
> Note: A problem with the R^2 metric is that sometimes it increases as we add more variables even if the added variables a
> In other words, the model can always map some data to a target variable.

- $\textbf{Adjusted R-Squared}(R^2)$

- **Description:** R^2 - Adjusted overcomes the incorrect increase of the R^2 by adding extra independent variables. In other words, it penalaizes the excess amount of independent variables.

- **Formula:** $R_a^2 = 1 - \{(\frac{n-1}{n-k-1})(1 - R^2)\}$
  $\hookrightarrow$ Adjusted R^2

where $n =$ Number of observations,
$k =$ number of features

- **Value's Range:** From 0 (bad model) to 1 (perfect model)

> ⚠
> Note: As k increases, the denominator decreases which makes the entire value to be subtracted from 1 a large value.

> As a result, the R_a^2 is decreased which means that **the more (irrelevant features, the worse the model.)**