Supervised Machine Learning: Regression and Classification, Arjunan K

Detailed pdf plus handwritten notes of Machine Learning Specialization by Andrew Ng in collaboration between DeepLearning.AI and Stanford Online. In this you can view the first course notes of the specialization, Supervised Machine Learning: Regression and Classification.

Intro to Machine Learning

Machine Learning is the Ability of computers to learn without being explicitly programmed. There are different types of Machine Learning Algorithms.

1. Supervised Learning
2. Unsupervised Learning
3. Recommender Systems
4. Reinforcement Learning

Supervised Learning

Machines are trained using well "labelled" training data, and on basis of that data, machines predict the output. The labelled data means some input data is already tagged with the correct output. It find a mapping function to map the input variable(x) with the output variable(y). Some use cases are given below,

- Spam Filtering
- Speech Recognition
- Text Translation
- Online Advertising
- Self-Driving Car
- Visual Inspection (Identifying defect in products)

Types of Supervised Learning

1. Regression
2. Classification

Unsupervised Learning

Models are not supervised using labelled training dataset. Instead, models itself find the hidden patterns and insights from the given data. It learns from un-labelled data to predict the output.

Types of Unsupervised Learning

1. Clustering (Group similar data)
2. Anomaly Detection (Finds unusual data points)
3. Dimensionality Reduction (Compress data to fewer numbers)

REGRESSION

It’s used as a method for predictive modelling in machine learning in which an algorithm is used to predict continuous outcomes. Commonly used regression is

Linear Regression

1. Simple Linear Regression - (one dependent and one independent variable)
2. Multiple linear regression - (one dependent and multiple independent variable)

CLASSIFICATION

In Classification, a program learns from the given dataset or observations and then classifies new observation into a number of classes or groups. Such as, Yes or No, 0 or 1, Spam or Not Spam, cat or dog etc. Classes can be called as targets/labels or categories. Commonly used classification is

Logistic Regression

In this note we will be focusing on the math behind the Linear and Logistic Regression Models.

SIMPLE LINEAR REGRESSION

- It is fitting a straight line to your data.
- Model the relation between independent (Input X) and dependent (Output Y) by fitting a linear equation to observed data.

What is Cost Function?

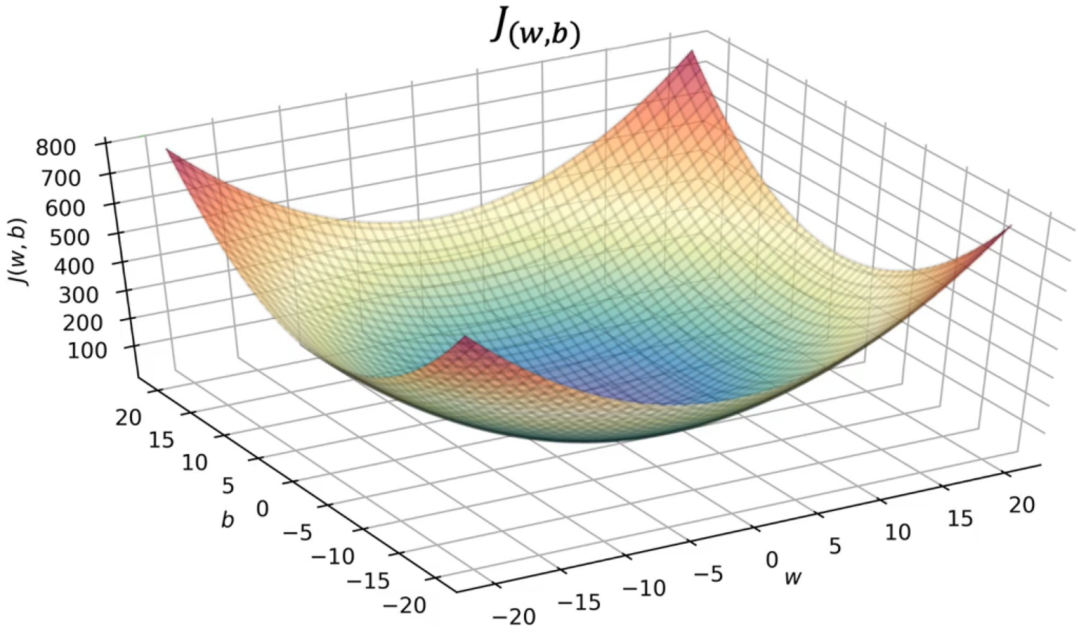
A cost function is an important parameter that determines how well a machine learning model performs for a given dataset. It calculates the difference between the expected value and predicted value and represents it as a single real number. It is the average of loss function (Difference between predicted and actual value).

Our aim is to minimize the cost function, which is achieved using **Gradient Descent**.

Types of cost function.

1. Mean Squared Error (MSE) for Linear Regression
2. Log Loss for Logistic Regression

Cost Function for Linear Regression - MSE (Convex)



Gradient Descent

Gradient descent is an optimization algorithm which is commonly-used to train machine learning models and neural networks. Training data helps these models learn over time, and the cost function within gradient descent specifically acts as a barometer, gauging its accuracy with each iteration of parameter updates. Until the function is close to or equal to zero, the model will continue to adjust its parameters to yield the smallest possible error.

Normal Equation (Alternative for Gradient Descent)

- Only for Linear Regression
- Solve w and b without iteration.
- But not a generalized one for other learning algorithm.
- when number of features is large > 1000, it is slow.
- most libraries use this under the hood.
- but gradient is better

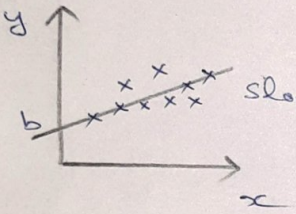
Simple Linear Regression.

$$f_{w,b}(x) = wx + b$$

or

$$\hat{Y}_i = wx_i + b$$

$\hat{Y}_i$ = estimated Y_i



Cost function – MSE

$$J(w,b) = \frac{1}{2m} \sum_{i=1}^{m} (\hat{Y}_i - Y_i)^2$$

m = number of Samples.

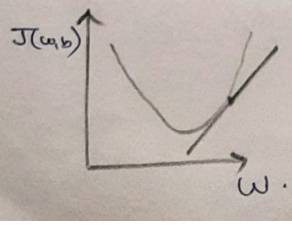
Gradient Desent

Minimize $J(w,b)$

$$w = w - \alpha \frac{d}{dw} J(w,b)$$

$$b = b - \alpha \frac{d}{db} J(w,b)$$

α is learning rate.



$\frac{d}{dw} J(w,b)$ is derivative. ie Slope of tangent.

$$\frac{d}{dw} J(w,b) = \frac{d}{dw} \frac{1}{2m} \sum_{i=1}^{m} (\hat{Y}_i - Y_i)^2$$

$$= \frac{d}{dw} \frac{1}{2m} \sum_{i=1}^{m} (wx_i + b - Y_i)^2$$

$$= \frac{1}{2m} \times \sum_{i=1}^{m} (wx_i + b - Y_i) \times 2x_i$$

$$= \frac{1}{m} \sum_{i=1}^{m} (wx_i + b - Y_i) x_i$$

$$= \frac{1}{m} \sum_{i=1}^{m} (\hat{Y}_i - Y_i) x_i$$

$$\frac{d}{db} J(w,b) = \frac{d}{db} \frac{1}{2m} \sum_{i=1}^{m} (\hat{Y}_i - Y_i)^2$$

$$= \frac{d}{db} \frac{1}{2m} \sum_{i=1}^{m} (wx_i + b - Y_i)^2$$

$$= \frac{1}{2m} \times \sum_{i=1}^{m} (wx_i + b - Y_i) \times 2$$

$$= \frac{1}{m} \sum_{i=1}^{m} (\hat{Y}_i - Y_i)$$

Since $\hat{Y}_i$ is $f(x_i)$

$$w = w - \alpha \frac{d}{dw} J(w,b)$$

$$w = w - \alpha \frac{1}{m} \sum_{i=1}^{m} (f(x_i) - Y_i) x_i$$

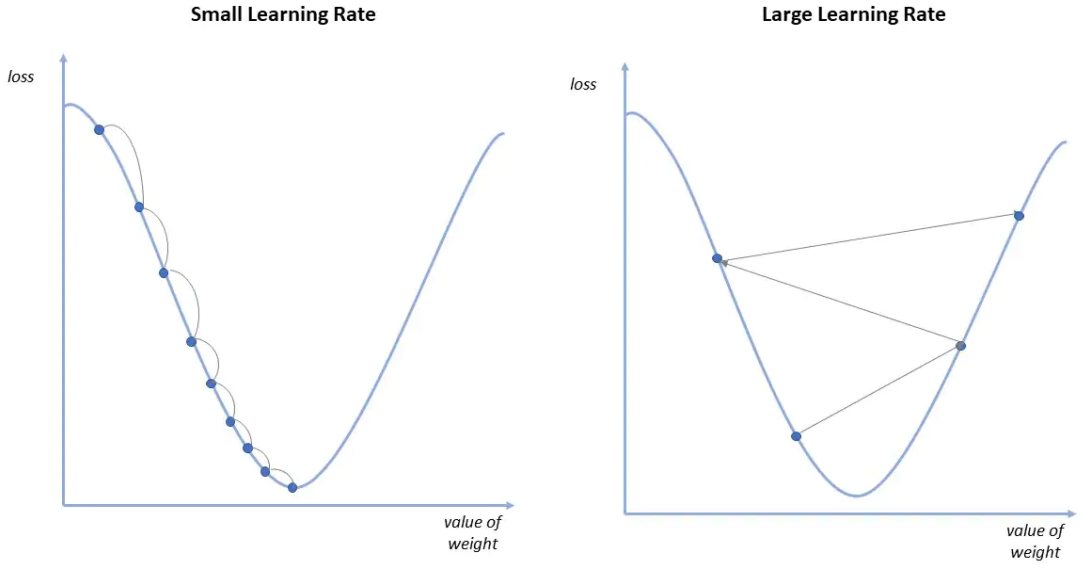
$$b = b - \alpha \frac{d}{db} J(w,b)$$

$$b = b - \alpha \frac{1}{m} \sum_{i=1}^{m} (f(x_i) - Y_i)$$

Learning Rate

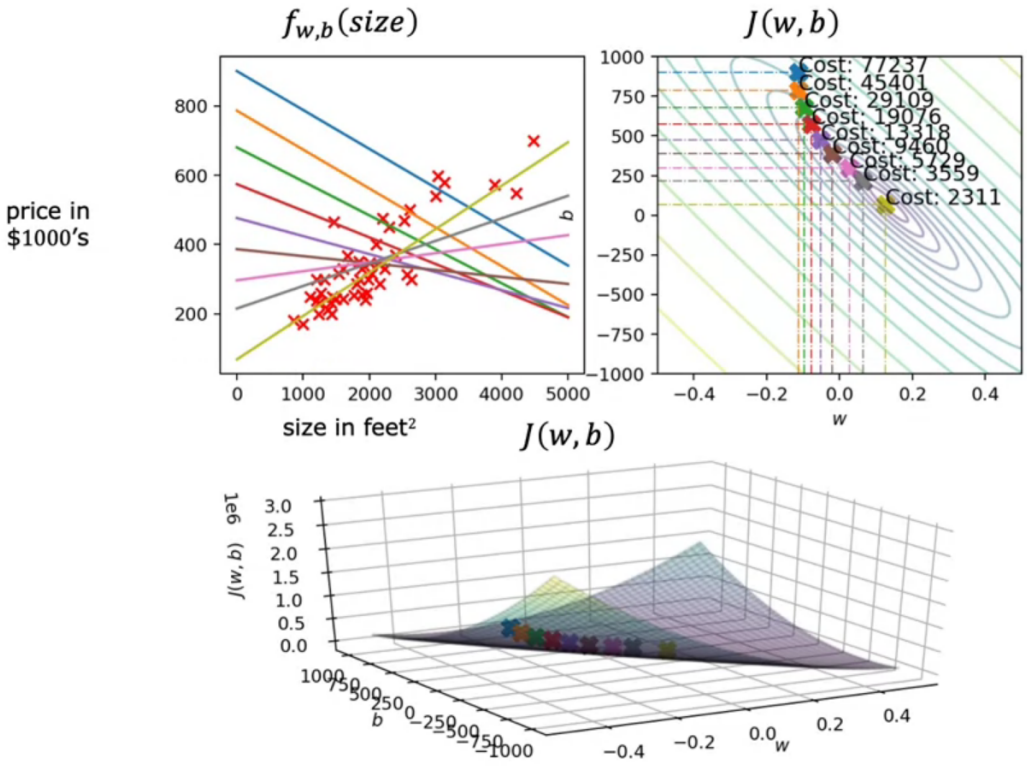
The learning rate is a hyperparameter that controls how much to change the model in response to the estimated error each time the model weights are updated.

- Small learning rate may result slow gradient descent.
- Large learning rate may result in divergence of descent (Fail to converge to minimum).
- If the cost function reaches local minimum, slope become zero. So w = w, not going to descent after that.
- Fixed learning rate has no problem, since derivative part decrease as we decent.
- **Alpha** value can be 0.001 and increased 3X times based on requirement. If Cost function is increasing decrease **Alpha** and Vice Versa.



1. Batch gradient descent (Used in this course)

Each step of gradient descent uses all training data. This process referred to as a training epoch.



2. Stochastic gradient descent

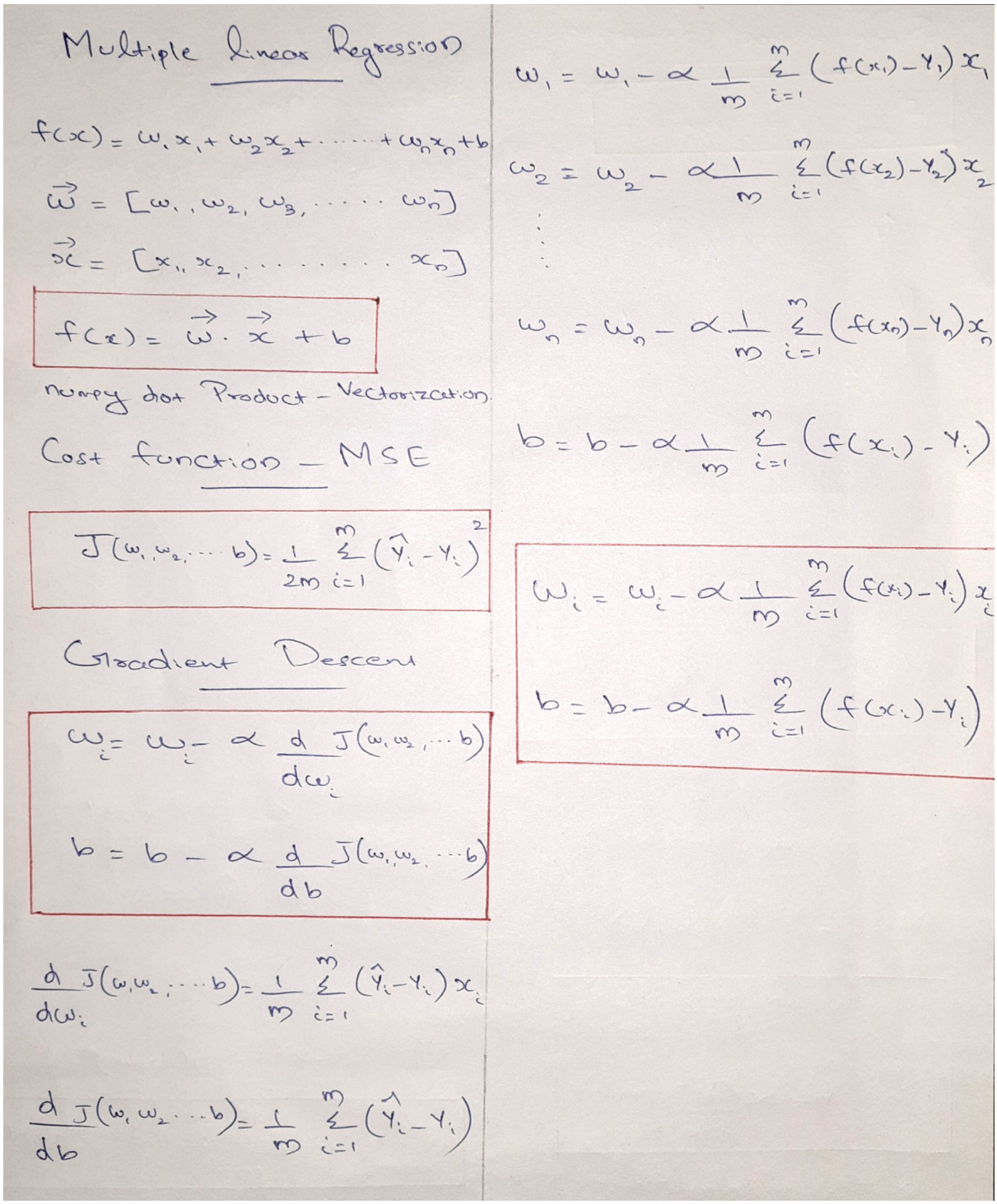
Each step of gradient descent uses a subset of training data. It runs a training epoch for each example within the dataset and it updates each training example's parameters one at a time.

3. Mini-batch gradient descent

Mini-batch gradient descent combines concepts from both batch gradient descent and stochastic gradient descent. It splits the training dataset into small batch sizes and performs updates on each of those batches. This approach strikes a balance between the computational efficiency of batch gradient descent and the speed of stochastic gradient descent.

Multiple Linear Regression

- Here we predict one dependent variable from multiple independent variable.



WHAT IS VECTORIZATION?

Vectorization is used to speed up the code without using loop. Using such a function can help in minimizing the running time of code efficiently. Various operations are being performed over vector such as **dot product of vectors** which is also known as **scalar product.**

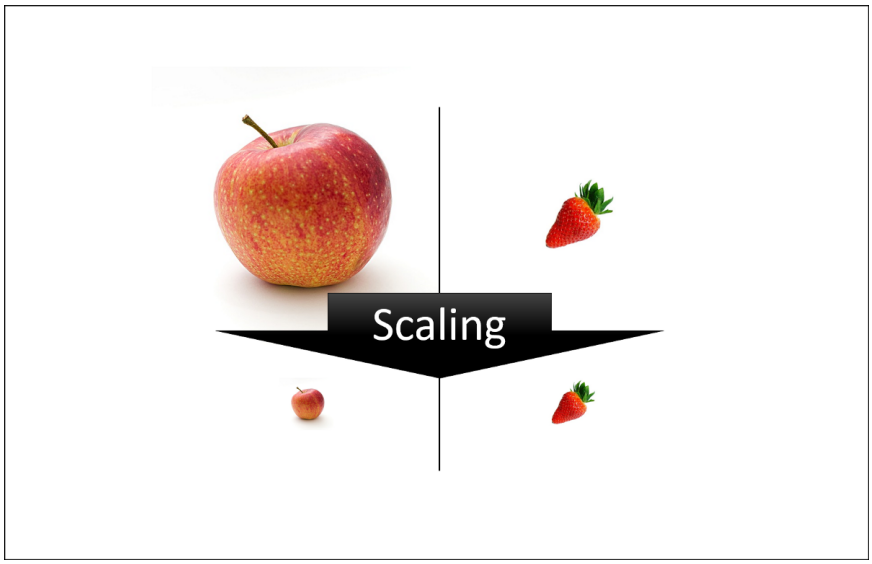
It uses principle of parallel running, which is also easy to scale.

$$a \cdot b = \underset{(1 \times n)}{\begin{bmatrix} a_1 & a_2 & a_3 & a_4 & a_5 \end{bmatrix}} \underset{(n \times 1)}{\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix}} = \left\{ a_1b_1 + a_2b_2 + a_3b_3 + a_4b_4 + a_5b_5 \right\}$$

Feature Scaling

Feature Scaling is a technique to standardize the independent features present in the data in a fixed range.

Example, if we have weight of a person in a dataset with values in the range 15kg to 100kg, then feature scaling transforms all the values to the range 0 to 1 where 0 represents lowest weight and 1 represents highest weight instead of representing the weights in kgs.



Types of Feature Scaling:

1. Standardization

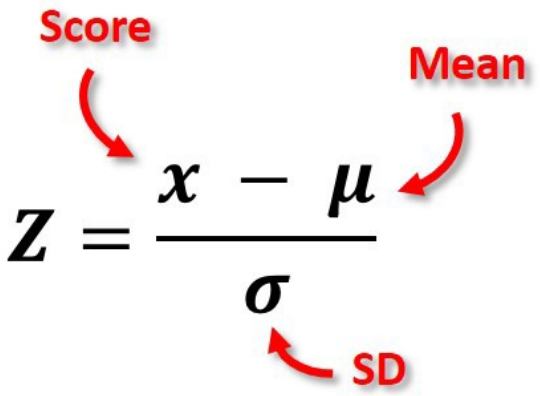
- Standard Scaler - (Z Score Normalization)

2. Normalization

- Min Max Scaling
- Max Absolute Scaling
- Mean Normalization
- Robust Scaling

Standardization (Standard Scaler)

Standardization is a scaling technique where the values are centered around the mean with a unit standard deviation. This means that the mean of the attribute becomes zero and the resultant distribution has a unit standard deviation.



Normalization

The goal of normalization is to change the values of numeric columns in the dataset to use a common scale, without distorting differences in the ranges of values or losing information.

1. Min Max Scaling

The minimum value of that feature transformed into 0, the maximum value transformed into 1, and every other value gets transformed into a decimal between 0 and 1.

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

2. Max Absolute Scaling

maximal value of each feature in the training set will be 1. It does not shift/center the data, and thus does not destroy any sparsity.

$$x_{scaled} = \frac{x}{max(|x|)}$$

3. Mean Normalization

It is very similar to Min Max Scaling, just that we use mean to normalize the data. Removes the mean from the data and scales it into max and min values.

$$x' = \frac{x - \mu}{\max(x) - \min(x)}$$

4. Robust Scaling

This Scaler removes the median and scales the data according to the quantile range (defaults to IQR: Interquartile Range). The IQR is the range between the 1st quartile (25th quantile) and the 3rd quartile (75th quantile).

$$\frac{x_i - Q_1(x)}{Q_3(x) - Q_1(x)}$$

The Big Question – Normalize or Standardize?

- Normalization is good to use when you know that the distribution of your data does not follow a Gaussian distribution. This can be useful in algorithms that do not assume any distribution of the data like K-Nearest Neighbors and Neural Networks.
- Standardization, on the other hand, can be helpful in cases where the data follows a Gaussian distribution. However, this does not have to be necessarily true. Also, unlike normalization, standardization does not have a bounding range. So, even if you have outliers in your data, they will not be affected by standardization.

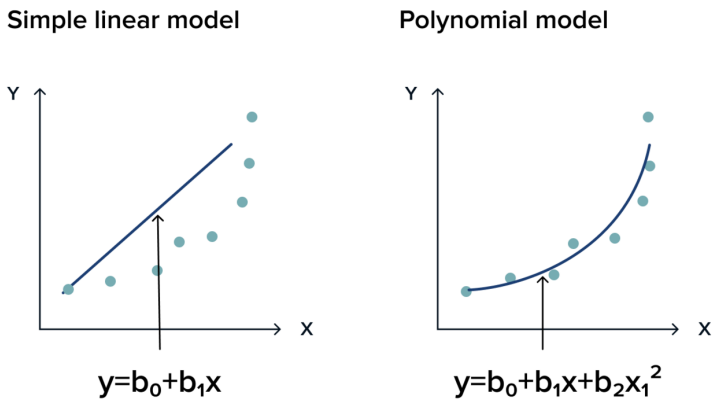
What is Feature Engineering?

Feature Engineering is the process of extracting and organizing the important features from raw data in such a way that it fits the purpose of the machine learning model. It can be thought of as the art of selecting the important features and transforming them into refined and meaningful features that suit the needs of the model.

Eg: Creating a feature Area from length and breadth features in data.

Why Polynomial Regression?

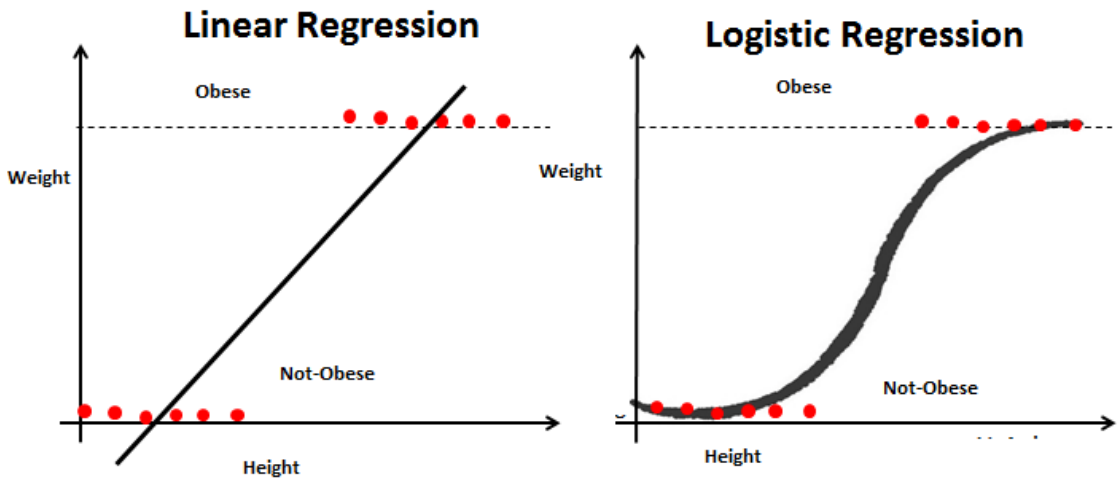
Suppose if we have non-linear data then Linear regression will not capable to draw a best-fit line and It fails in such conditions. consider the below diagram which has a non-linear relationship and you can see the Linear regression results on it, which does not perform well means which do not comes close to reality. Hence, we introduce polynomial regression to overcome this problem, which helps identify the curvilinear relationship between independent and dependent variables.



LOGISTIC REGRESSION

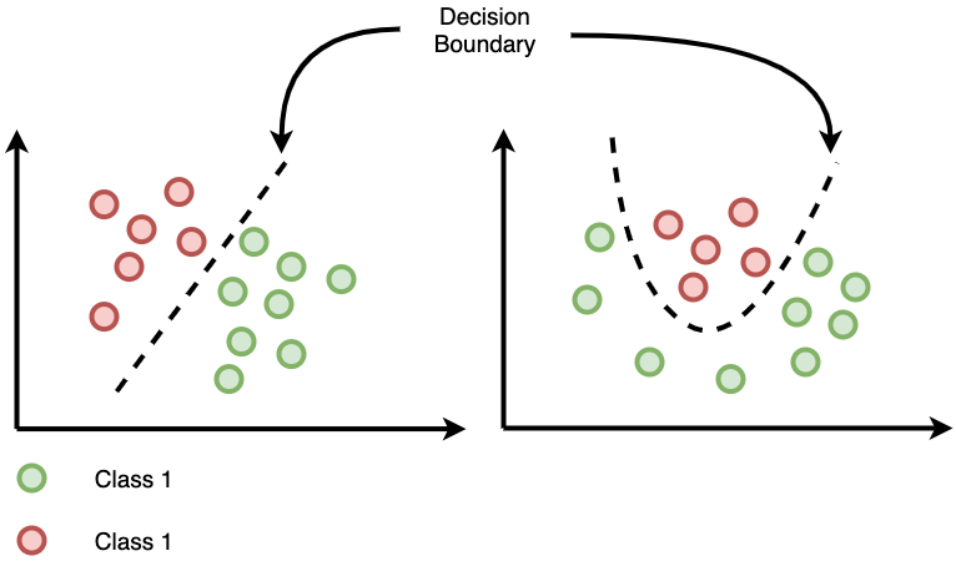
- Logistic regression is a predictive analysis of Probabilities for classification problems.
- It is used when our dependent variable has only 2 outputs.
- Eg: A person will survive this accident or not, The student will pass this exam or not.

Here we replace linear function with Logistic/Sigmoid Function



Decision Boundary – Logistic Regression

- The line or margin that separates the classes.
- Classification algorithms are all about finding the decision boundaries.
- It need not be straight line always.



What is Log Loss?

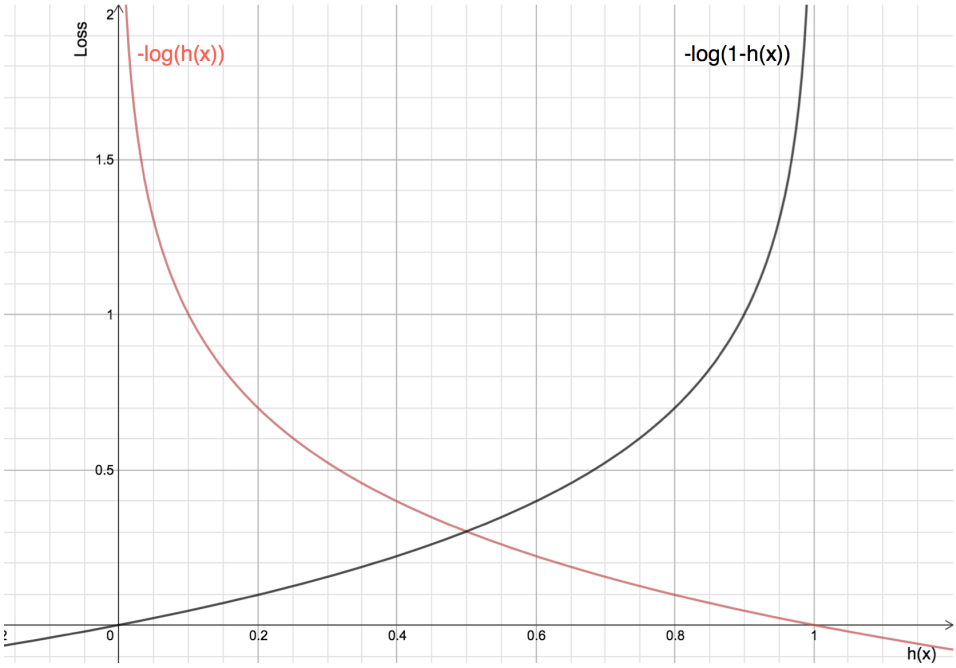
Log-loss is indicative of how close the prediction probability is to the corresponding actual/true value (0 or 1 in case of binary classification).

- Lower log loss value means better prediction
- Higher log loss means worse prediction

Equation of Log Loss Cost Function

$$LogLoss = -\frac{1}{n}\sum_{i=0}^{n}[y_i log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)]$$

- For Y = 0, Log Loss graph show low loss for y = 0 and high loss for y = 1
- For Y = 1, Log Loss graph show high loss for y = 0 and low loss for y = 1



Learning Curve, Vectorization, Feature Scaling all works same for Logistic Regression just like Linear Regression.

Logistic Regression

Sigmoid / logistic function.

$$g(z) = \frac{1}{1+e^{-z}} \qquad 0 < g(z) < 1$$

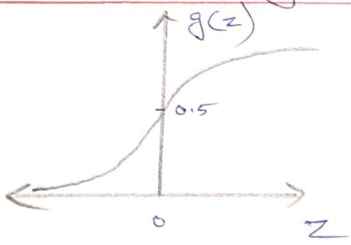
$$f(x) = g(\vec{w} \cdot \vec{x} + b)$$

$$f(x) = \frac{1}{1+e^{-(\vec{w} \cdot \vec{x} + b)}}$$

Decision boundary

for $g(z) = \frac{1}{1+e^{-z}}$

Decision boundary at $z = 0$



$g(z) \geq 0.5$

$z \geq 0$

$\vec{w} \cdot \vec{x} + b \geq 0$

$\Rightarrow \hat{y} = 1$

when $g(z) < 0.5$ then $\hat{y} = 0$

$\hat{y}$ estimate is decided based on $f(x)$

$f(x) \geq$ decision boundary

$f(x) <$ decision boundary.

Cost function

MSE will not be convex for logistic regression.

$$L(f(x), y_i) = -\log f(x) \quad y_i = 1$$
$$-\log(1 - f(x)) \quad y_i = 0$$

$$J(w,b) = \frac{1}{m} \sum_{i=1}^{m} L(f(x), y_i)$$

$$J(w,b) = -\frac{1}{m} \sum_{i=1}^{m} \left(\log f(x) \times y_i \right) + \left(\log[1 - f(x)] \times (1 - y_i) \right)$$

Gradient Descent

$$w = w - \alpha \frac{d}{dw} J(w,b)$$

$$b = b - \alpha \frac{d}{db} J(w,b)$$

$$w_i = w_i - \alpha \frac{1}{m} \sum_{i=1}^{m} (f(x_i) - y_i) x_i$$

$$b = b - \alpha \frac{1}{m} \sum_{i=1}^{m} (f(x_i) - y_i)$$

Where

$$f(x_i) = \frac{1}{1+e^{-(\vec{w} \cdot \vec{x} + b)}}$$

Overfitting and Underfitting in Machine Learning

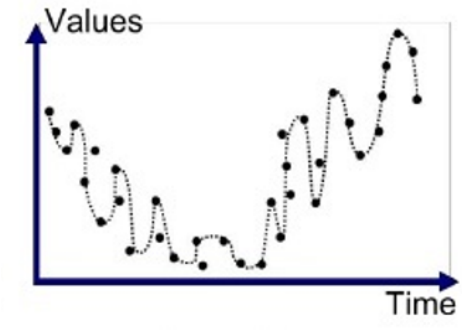
Overfitting and Underfitting are the two main problems that occur in machine learning and degrade the performance of the machine learning models.

Before understanding the overfitting and underfitting, let's understand some basic term that will help to understand this topic well:

- **Signal:** It refers to the true underlying pattern of the data that helps the machine learning model to learn from the data.
- **Noise:** Noise is unnecessary and irrelevant data that reduces the performance of the model.
- **Bias:** It is the difference between the predicted values and the actual values. It is a prediction error in the model due to oversimplifying the machine learning algorithms.
- **Variance:** If the machine learning model performs well with the training dataset, but does not perform well with the test dataset, then variance occurs.

Overfitting

- Overfitting occurs when our machine learning model tries to cover all the data points.
- Model starts caching noise and inaccurate values present in the dataset, and all these factors reduce the efficiency and accuracy of the model.
- The overfitted model has **low bias** and **high variance.**
- The chances of overfitting increase as we provide more training to our model.



Overfitted

It may look efficient, but in reality, it is not so. Because the goal of the regression model to find the best fit line, but here we have not got any best fit, so, it will generate the prediction errors.

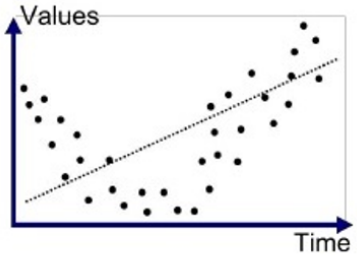
How to avoid the Overfitting:

Both overfitting and underfitting cause the degraded performance of the machine learning model. But the main cause is overfitting, so there are some ways by which we can reduce the occurrence of overfitting in our model.

- **Cross-Validation**
- **Training with more data**
- **Ensembling (T**echnique that combines several base models to produce one optimal model)
- **Removing features**
- **Early stopping the training**
- **Regularization (Reduce Size of Parameters)**

Underfitting

- In the case of underfitting, the model is not able to learn enough from the training data, and hence it reduces the accuracy and produces unreliable predictions.
- Underfitted model has **high bias** and **low variance**.



Underfitted

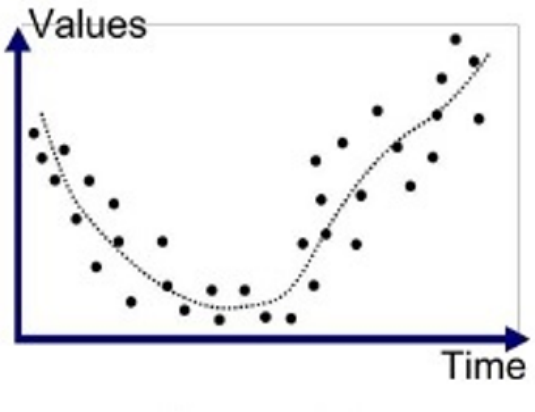
How to avoid Underfitting:

- By increasing the training time of the model.
- By increasing the number of features.

Goodness of Fit

The model with a good fit is between the underfitted and overfitted model, and ideally, it makes predictions with 0 errors, but in practice, it is difficult to achieve it.

As when we train our model for a time, the errors in the training data go down, and the same happens with test data. But if we train the model for a long duration, then the performance of the model may decrease due to the overfitting, as the model also learn the noise present in the dataset. The errors in the test dataset start increasing, so the point, just before the raising of errors, is the good point, and we can stop here for achieving a good model.



Good Fit/Robust

REGULARIZATION

We mainly regularizes or reduces the coefficient of features toward zero. In simple words, "In regularization technique, we reduce the magnitude of the features by keeping the same number of features."

Regularization

$$J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}(f(x_i)-Y_i)^2 + \frac{\lambda}{2m}\sum_{j=1}^{n} w_j^2$$

λ = Regularization Parameter.

$\frac{\lambda}{2m}\sum_{j=1}^{n} w_j^2$ is Regularization term.

$\lambda = 10^{10}$ large the $w_i \approx 0$

Then it cause Underfitting

$\lambda \approx 0$ Small then Overfit.

Regularization Linear Regression

We want to minimize $J(w,b)$

$$J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}(f(x_i)-Y_i)^2 + \frac{\lambda}{2m}\sum_{j=1}^{n} w_j^2$$

$$w_j = w_j - \alpha \frac{d}{dw} J(\vec{w},b)$$

$$b = b - \alpha \frac{d}{db} J(\vec{w},b)$$

$$\frac{d}{dw_j} J(\vec{w},b) = \frac{1}{m}\sum_{i=1}^{m}(f(x_i)-Y_i)x_j + \frac{\lambda}{m} w_j$$

$$\frac{d}{db} J(\vec{w},b) = \frac{1}{m}\sum_{i=1}^{m}(f(x_i)-Y_i)$$

$$w_j = w_j - \alpha\left[\frac{1}{m}\sum_{i=1}^{m}(f(x_i)-Y_i)x_j + \frac{\lambda}{m} w_j\right]$$

$$w_j = w_j\left(1-\alpha\frac{\lambda}{m}\right) - \alpha\frac{1}{m}\sum_{i=1}^{m}(f(x_i)-Y_i)x_j$$

$m=5 \quad \alpha=0.01 \quad \lambda=1$

$\frac{\alpha\lambda}{m} = 0.01\times\frac{1}{50} = 0.0002$

$1-0.002 = 0.9998$

So w_j is decreasing at each step.

Regularized Logistic Regression

$$J(w,b) = -\frac{1}{m}\sum_{i=1}^{m}\left[Y_i \log(f(x_i)) + (1-Y_i)\log(1-f(x_i))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n} w_j^2$$

$$w_j = w_j - \alpha \frac{d}{dw} J(\vec{w},b)$$

$$b = b - \alpha \frac{d}{db} J(\vec{w},b)$$

$\frac{d}{dw} J(\vec{w},b)$ and $\frac{d}{db} J(\vec{w},b)$ and w_j and b equation is Same.

But $f(x) = \frac{1}{1+e^{-(\vec{w}\cdot\vec{x}+b)}}$

No Change in b, Because in both cases we are not regularizing b.

Types of Regularization Techniques

There are two main types of regularization techniques: L1(Lasso) and L2(Ridge) regularization

1) Lasso Regularization (L1 Regularization)

In L1 you add information to model equation to be the absolute sum of theta vector (θ) multiply by the regularization parameter (λ) which could be any large number over size of data (m), where (n) is the number of features.

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} Cost(h_\theta(x^{(i)}), y^{(i)}) + \frac{\lambda}{m}\sum_{j=1}^{n} |\theta_j|$$

2) Ridge Regularization (L2 Regularization)

In L2, you add the information to model equation to be the sum of vector (θ) squared multiplied by the regularization parameter (λ) which can be any big number over size of data (m), which (n) is a number of features.

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} Cost(h_\theta(x^{(i)}), y^{(i)}) + \frac{\lambda}{2m}\sum_{j=1}^{n} \theta_j^2$$

THANK YOU