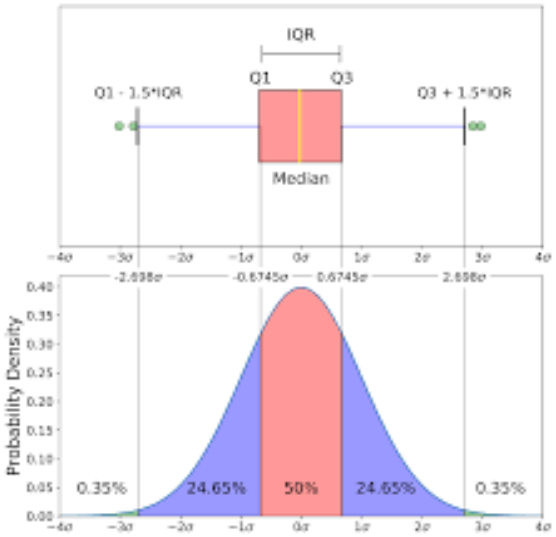
Machine Learning theoretical concepts

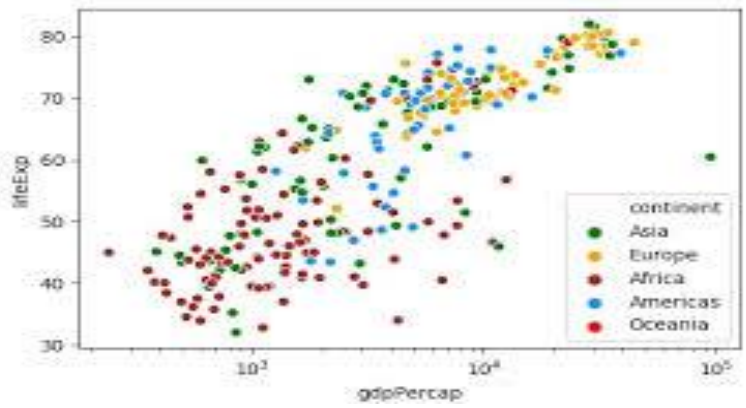
By: Vikram Pal

Data cleaning and reducing bias & variance :

1. Inconsistent column:
    a. pandas.DataFrame.drop
2. Missing data:
    1. Leave as it is, 2. Filling the missing values, 3. Drop them
3. Outliers:
    a. For detecting the outliers, we can use:
        1. Box Plot



2. 2. Scatter plot



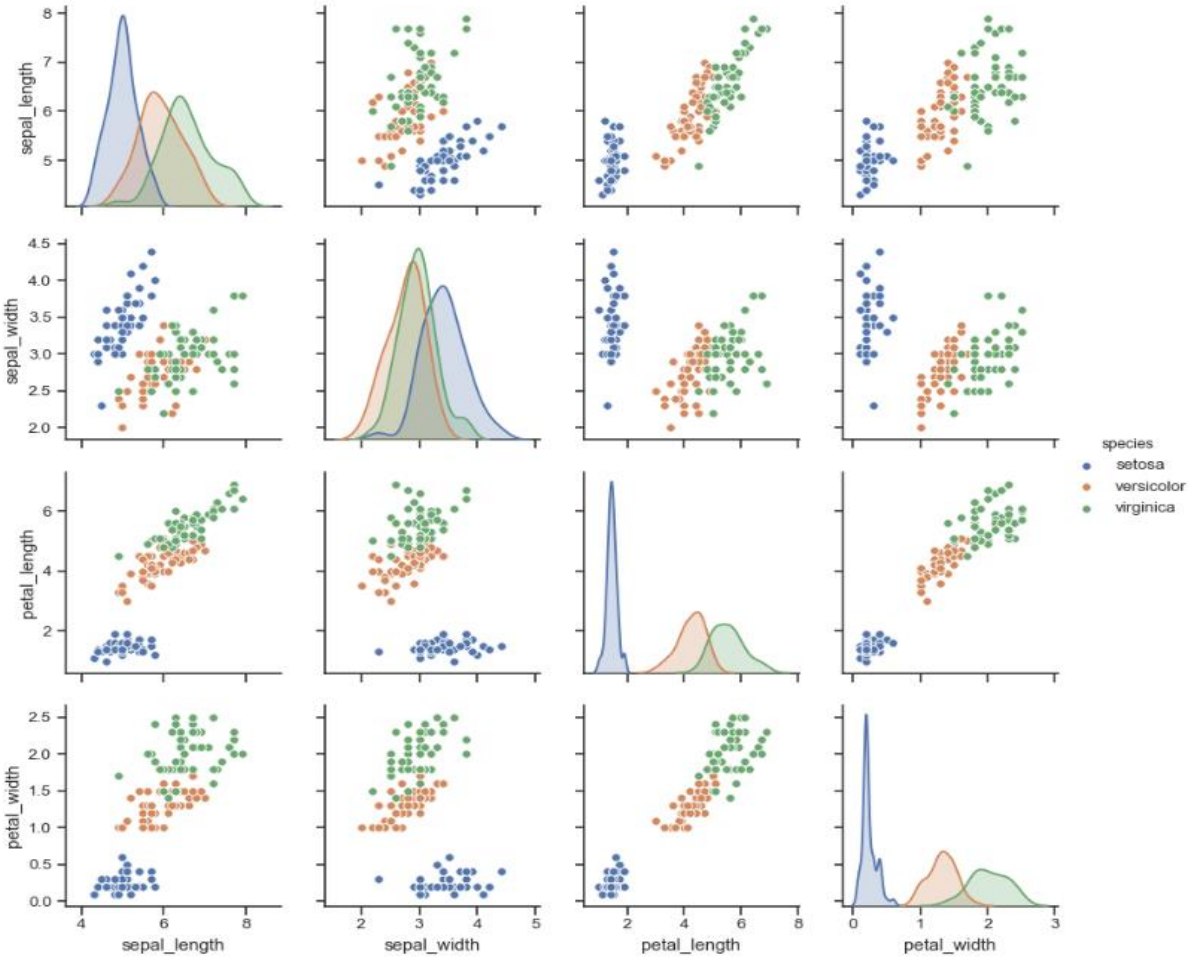
2. Z-score etc.

4. Duplicate rows:
    i. dataset_name.drop_duplicates()
5. Tidy Data set:
    a. pandas.melt.
6. Converting data types:
    a. In Data Frame data can be of many types. As example: 1. Categorical data, 2. Object data, 3. Numeric data, 4. Boolean data
7. String Manipulation

Machine Learning theoretical concepts

By: Vikram Pal

8. Data concatenation
9. Correlation: it is used for checking linear relation b/w two variable.
10. Pair plot: multiple pairwise bivariate distribution



Feature Selection Methods:

1. Univariate Selection: scikit-learn library provides the SelectKBest class
2. Feature importance: it gives you a score for each feature of your data, the higher the score more important or relevant is the feature towards your output variable.
3.Correlation Matrix with Heatmap: Correlation states how the features are related to each other or the target variable.

Must read this:

https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/
https://towardsdatascience.com/the-5-classification-evaluation-metrics-you-must-know-aa97784ff226

t-SNE (t-distributed stochastic neighbor embedding):

. probability distribution:- a. High dimensional object→ similar objects assigned higher probability and dissimilar object assigned lower

Machine Learning theoretical concepts

By: Vikram Pal

. minimize KL divergence

The t-SNE algorithm comprises two main stages.

First, t-SNE constructs a **probability distribution over pairs of high-dimensional objects** in such a way that similar objects are assigned a higher probability while dissimilar points are assigned a lower probability.

Second, t-SNE defines a similar probability distribution over the points in the low-dimensional map, and it minimizes the Kullback–Leibler divergence (KL divergence) between the two distributions with respect to the locations of the points in the map. While the original algorithm uses the Euclidean distance between objects as the base of its similarity metric, this can be changed as appropriate.

KL divergence: is a measure of how a probability distribution differs from another probability distribution.

Variance Inflation Factor:

Variance inflation factor (VIF) is a measure of the amount of **multicollinearity in a set of multiple regression variables.**

VIF>5 means that variables have strong multicollinearity.

Regression Analysis: Femoral Neck versus %Fat S, Weight S, Activity S

Analysis of Variance

Source	DF	Adj SS	Adj MS	F-Value	P-Value
Regression	4	0.55578	0.138946	27.95	0.000
%Fat S	1	0.04786	0.047863	9.63	0.003
Weight S	1	0.30473	0.304728	61.29	0.000
Activity S	1	0.04703	0.047027	9.46	0.003
%Fat S*Weight S	1	0.04175	0.041745	8.40	0.005
Error	87	0.43256	0.004972		
Total	91	0.98834			

Model Summary

S	R-sq	R-sq(adj)	R-sq(pred)
0.0705118	56.23%	54.22%	50.48%

Coefficients

Term	Coef	SE Coef	T-Value	P-Value	VIF
Constant	0.82161	0.00973	84.40	0.000	
%Fat S	-0.00598	0.00193	-3.10	0.003	3.32
Weight S	0.00835	0.00107	7.83	0.000	4.75
Activity S	0.000022	0.000007	3.08	0.003	1.05
%Fat S*Weight S	-0.000214	0.000074	-2.90	0.005	1.99

SMOTE:

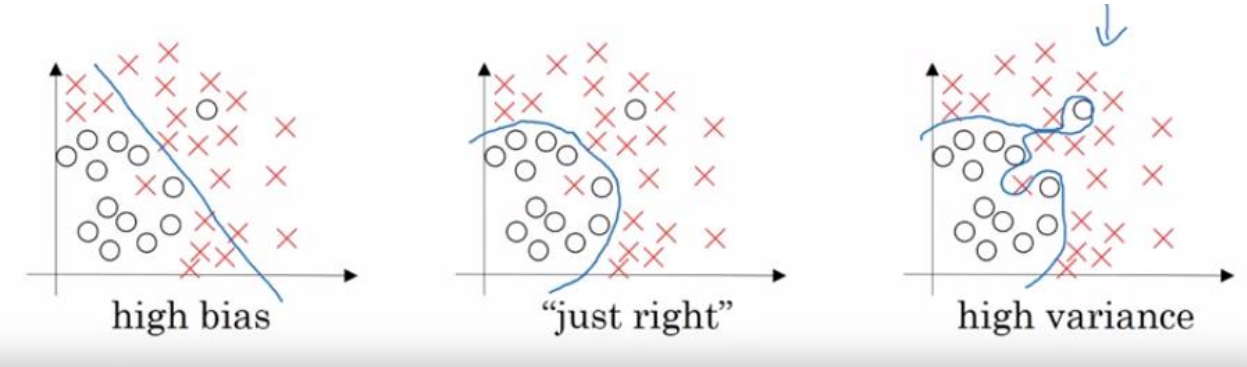
is an **oversampling technique** where the synthetic samples are generated for the minority class. This algorithm helps to overcome **the overfitting problem posed by random oversampling**

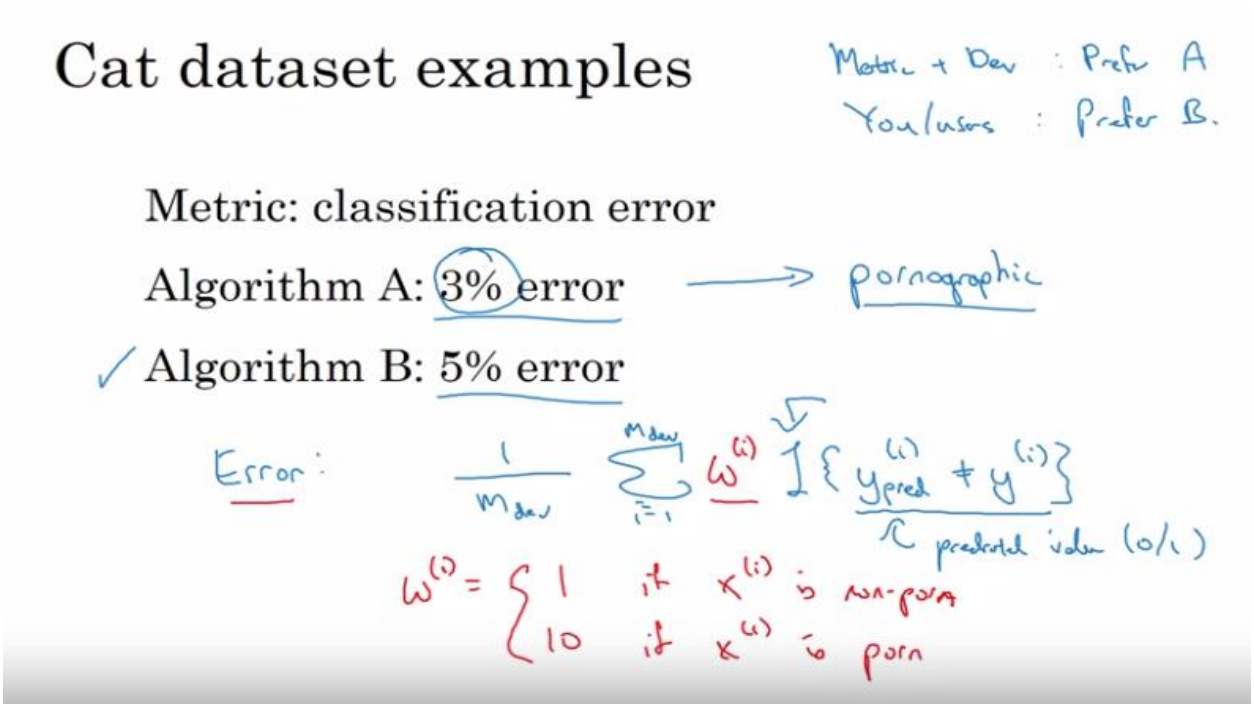
Machine Learning theoretical concepts

By: Vikram Pal

When to Change Dev/Test Sets and Metrics?

https://www.coursera.org/learn/machine-learning-projects/lecture/Ux3wB/when-to-change-dev-test-sets-and-metrics





How do you calculate bias-variance trade-off?

K-fold cross validation + Grid-Search == compare the score across the different tuning options.

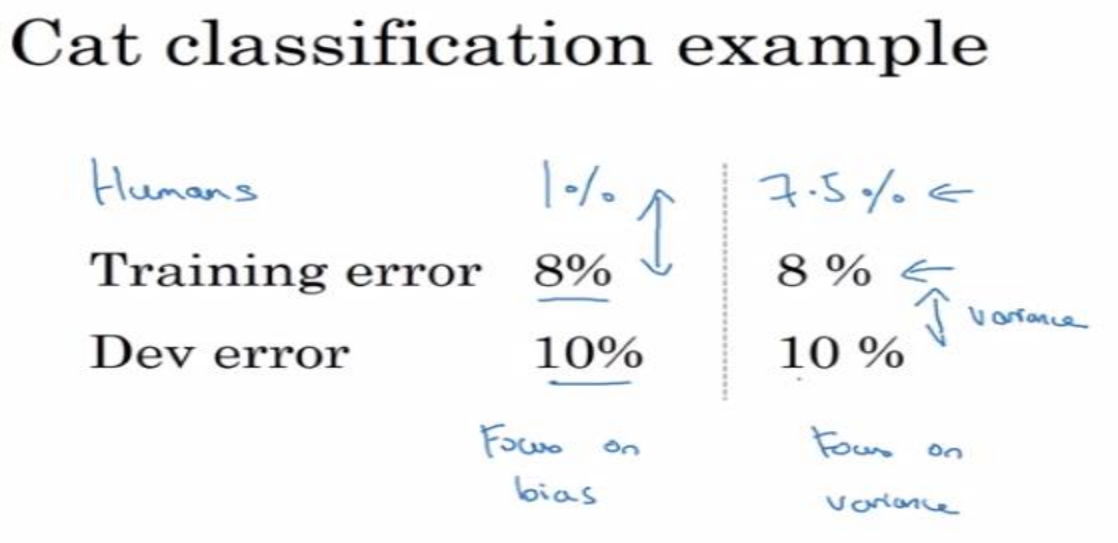
You can measure the bias-variance trade-off using k-fold cross validation and applying Grid-Search on the parameters. This way you can compare the score across the different tuning options that you specified and choose the model that achieve the higher test score.

Avoidable Bias:

https://www.coursera.org/learn/machine-learning-projects/lecture/4IPD6/improving-your-model-performance

Machine Learning theoretical concepts

By: Vikram Pal



To avoid bias:

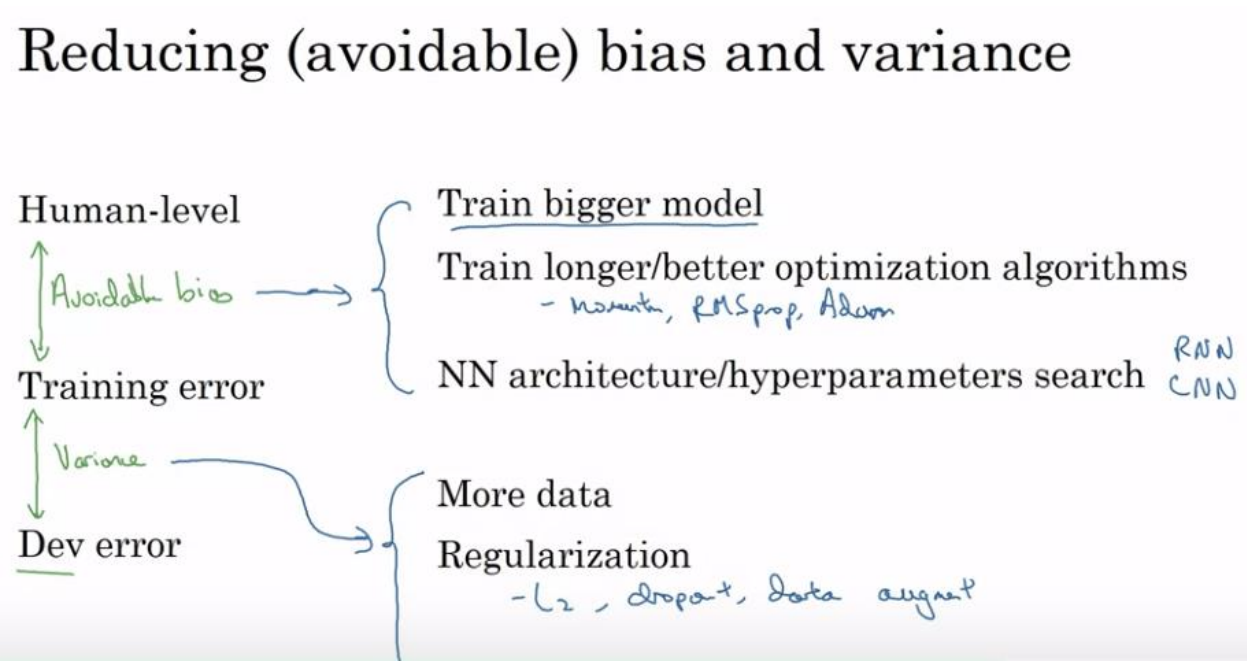
1. Train bigger model

2. Train longer/better optimization algorithms(Momentum, RMSprop and Adam)

3. NN architecture/ Hyperparameters Search (RNN and CNN)

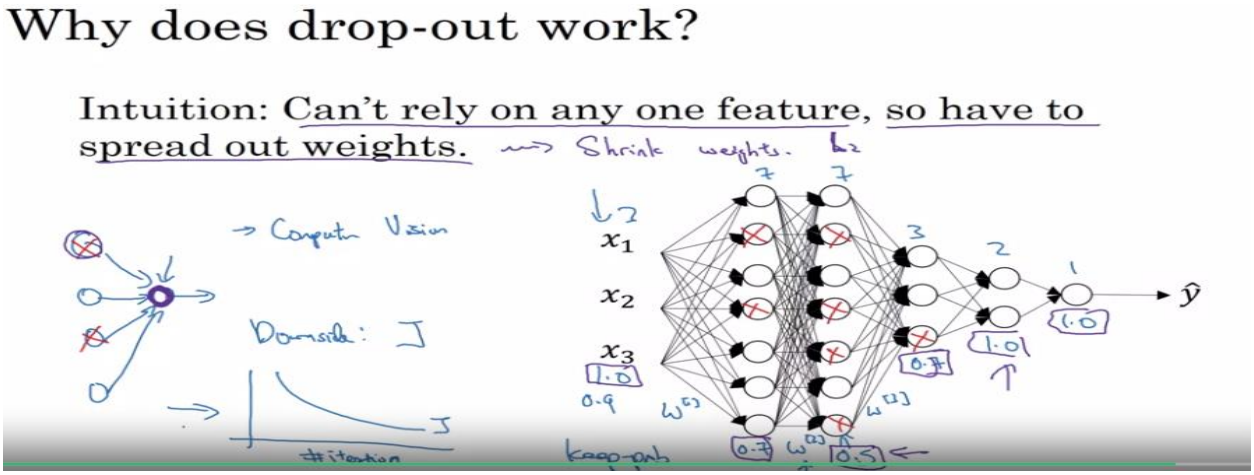
To avoid Variance:

1. More data 2. Regularization (l2) 3. Dropout 4. Data augment



Machine Learning theoretical concepts

By: Vikram Pal



Regression & Classification:

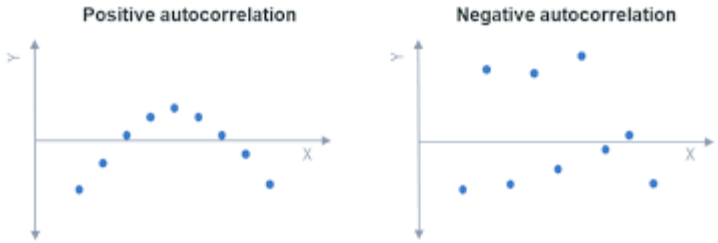
Residual Sum of Squares (RSS):

The residual sum of squares (RSS) is a statistical technique used to measure the **amount of variance in a data set that is not explained by a regression model itself**. Instead, it estimates the variance in the residuals, or error term.

Linear Regression:

The regression has five key assumptions:

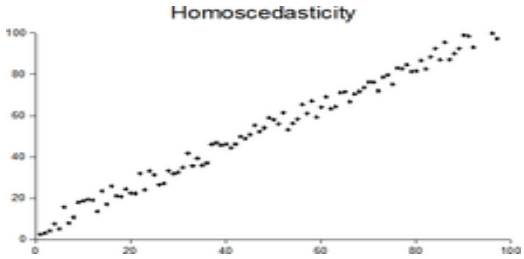
- a.Linear relationship b. Multivariate normality c. No or little multicollinearity(**residuals are normally distributed**) d. No-autocorrelation e. Homoscedasticity (**error term is the same across all values)**
- **Linear relationship**: linear relationship between the features and target
- **Multivariate normality:** Multiple regression assumes that the **residuals are normally distributed**.
- **No or little multicollinearity:** Multicollinearity is a state of **very high inter-correlations** or inter-associations among the independent variables
  ➔ Pair Plot and heatmap for correlation.
- **No autocorrelation**: No autocorrelation refers to a situation in which **no identifiable relationship exists between the values of the error term**



- **Homoscedasticity:** Homoscedasticity describes a situation in which the **error term is the same across all values** of the independent variables**.**

Machine Learning theoretical concepts

By: Vikram Pal



Plot with random data showing homoscedasticity: at each value of *x*, the *y*-value of the dots has about the same variance.

Evaluation Matrices for a regression model:

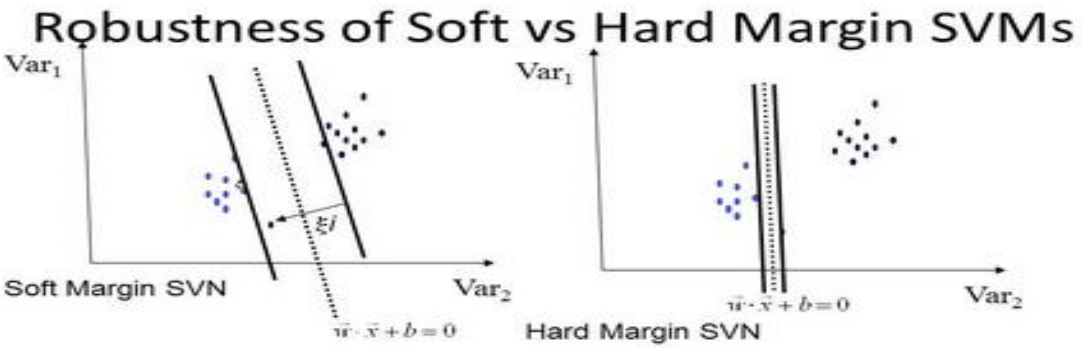
- Mean Absolute Error(MAE)
- Mean Squared Error(MSE)
- Root Mean Squared Error(RMSE)
- R-Squared(Coefficient of Determination)
- Adjusted R-Squared

Which evaluation metric should you prefer to use for a dataset having a lot of outliers in it?

Mean Absolute Error(MAE) is preferred when we have too many outliers present in the dataset because MAE is robust to outliers whereas MSE and RMSE are very susceptible to outliers and this start penalizing the outliers by squaring the error terms, commonly known as residuals.

SVM: (kernel tricks): -

- Gamma: kernel coefficient small gamma gives less complexity and large gamma give more complexity
- C parameter: regularization parameter, it tells us how much you want to penalize the misclassified error
- Trade off: **width of the margin vs no of error** committed by the linear decision boundary.
- **Linearly Separable (hard margin) [underfitting] and Non-linearly Separable** (soft margin) [overfitting] data.
- Number of dimensions are greater than the number of samples



- Soft margin – underfitting
- Hard margin – overfitting

Trade-off: width of the margin vs. no. of training errors committed by the linear decision boundary

Machine Learning theoretical concepts

By: Vikram Pal

To meet the soft margin objective, we need to introduce a slack variable ε>=0 for each sample; it measures how much any instance is allowed to violate the margin.

Kernel: The function used to map a **lower dimensional data into a higher dimensional data**.
Kernel — It specifies the kernel type to be used. There are different kernel options such as linear, radial basis function (RBF), polynomial and sigmoid. Here "rbf" and "poly" are useful for non-linear hyper-plane.

SVM Regression:

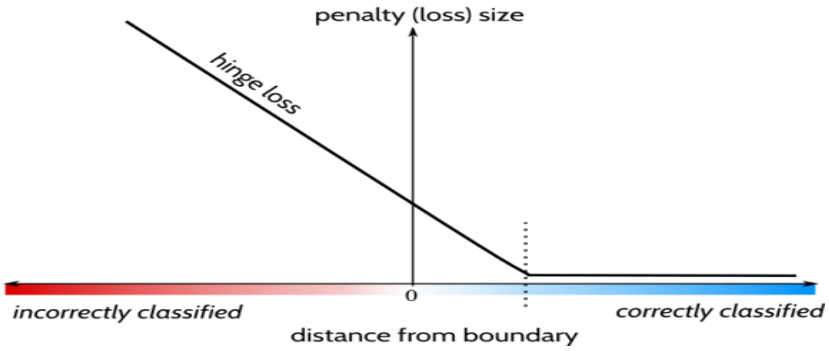
it tries to fit as many instances as possible between the margin while limiting the margin violations.

SVM Classification:

fit the largest possible street between two classes.

Gamma — It is the **kernel coefficient** for the '**rbf', 'poly' and 'sigmoid'**. Small Gamma (less variance) gives less complexity and larger gamma(more variance) gives more complexity.

C — It is the **regularization parameter**. It allowed you to decide how much you want to penalize the misclassified points



Gamma vs C parameter

For a linear kernel, we just need to optimize the c parameter. However, if we want to use an RBF kernel, both c and gamma parameters need to optimize simultaneously. If gamma is large, the effect of c becomes negligible. If gamma is small, c affects the model just like how it affects a linear model. Typical values for c and gamma are as follows. However, specific optimal values may exist depending on the application:

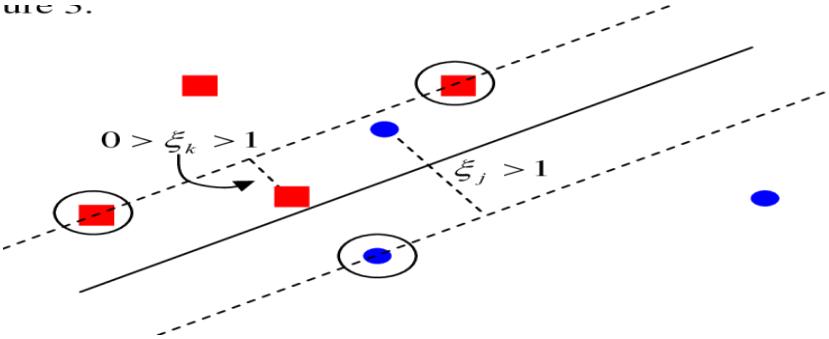
For RBF:
If gamma large, effect of c become negligible
If gamma small, c affects the model just like how it affects a linear model

0.0001 < gamma < 10
0.1 < c < 100

Slack Variable:
To meet the soft margin objective, we need to introduce a slack variable ε>=0 for each sample; it measures how much any particular instance is allowed to violate the margin.

Machine Learning theoretical concepts

By: Vikram Pal



Popular kernel

1. Gaussian Redial basis function

It is one of the most popular kernels used in SVM. It is used when there is no prior knowledge about data.

$$k(\mathbf{x_i}, \mathbf{x_j}) = \exp(-\gamma \|\mathbf{x_i} - \mathbf{x_j}\|^2)$$

Gaussians redial basis function

2.Gaussian function

$$k(\mathbf{x}_i, \mathbf{x}_j) = e^{-\left(\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)}$$

3.Polynomial Kernel Function

It is written as,

$$k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + a)^b$$

4.Linear Kernel

It is just the normal dot product,

$$k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j :$$

Small comparison of all the above kernel,

***time of SVM learning**: linear < poly < rbf*
ability to fit any data: linear < poly < rbf
risk of overfitting: linear < poly < rbf
risk of underfitting: rbf < poly < linear
number of hyperparameters: linear (0) < rbf (2) < poly (3)

Pros:

- It is useful for both **linearly Separable (hard margin) and Non-linearly Separable** (soft margin) data.
- It is effective **in high dimensional spaces**.

- It is effective in cases where several dimensions are greater than the number of samples.
- It uses a **subset of training points** in the decision function (called support vectors), so it is also memory efficient.
- **Outliers** do not impact the SVM function.

Cons:

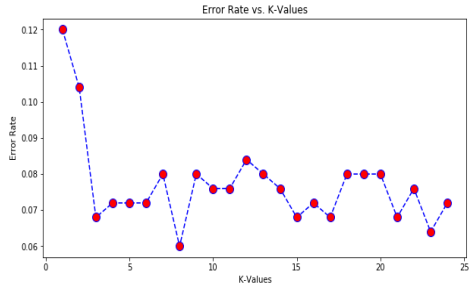
- **Picking the right kernel** and parameters can be computationally intensive.
- It also doesn't perform very well, when the **data set has more noise** i.e., target classes are overlapping
- SVM doesn't directly provide probability estimates, these are calculated using an **expensive five-fold cross-validation.**

KNN:(similar neighbors for the new data point.) classification(count) and Regression(Average) K larger underfitting and K small overfitting

(Hyperparameters: {**n_neighbors*int, default=5},{*weights*{'uniform', 'distance'} or callable, default='uniform'},algorithm*{'auto', {'ball_tree', 'kd_tree', 'brute'}, default='auto'})**

K is a **number used to identify similar neighbors** for the new data point**.**
How to choose the value of K?

Derive a plot between error rate and K denoting values in a defined range. Then choose the K value as having a minimum error rate.



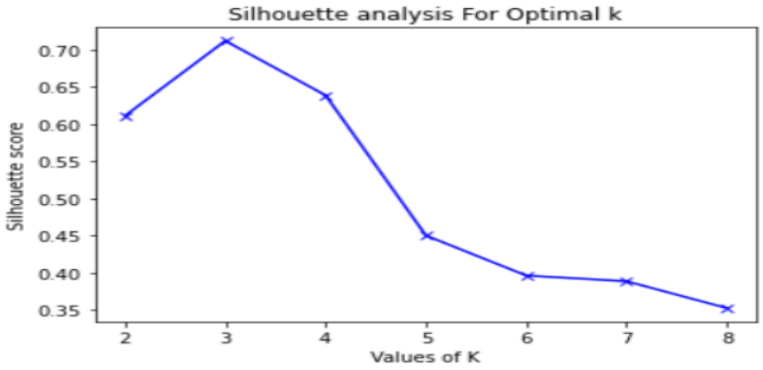
If there too many elbows, then It would be hard to pick a optimal number of K. In that situation, we use Silhouette analysis.
The silhouette coefficient is a measure of how similar a data point is within-cluster (cohesion) compared to other clusters (separation).
The equation for calculating the silhouette coefficient for a particular data point:

$$S(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

a. S(i) is the silhouette coefficient of the data point i.
b. a(i) is the average distance between i and all the other data points in **the cluster to which i belongs.**
c. b(i) is the average distance from i to all clusters to **which i does not belong**.

Machine Learning theoretical concepts

By: Vikram Pal



Line plot between K and Silhouette score

Supervised machine learning algorithm as target variable is known.

Having small K value causes overfitting. Because small k value causes noise to have higher influence on Result.

Having large K value leads to underfitting. Because having k large value, the model won't be able to generalize.

KNN can be used for value imputation in both Categorical and Continuous categories of data. It is the only algorithm that can achieve this.

The KNN algorithm doesn't learn anything from the training data, but rather it just store the training data at the time of training.

- Nonparametric as **it does not make an assumption** about the underlying data distribution pattern.
- **Lazy algorithm as KNN does not have a training step**. All data points will be used only at the time of prediction. With no training step, prediction step is costly. An eager learner algorithm eagerly learns during the training step.
- Used for both Classification and Regression.
- Uses **feature similarity to predict** the cluster that the new point will fall into.

Note: - How to pick K value: **by plotting accuracy rate or F1 score against different values of K**.

For classification(Count), **count the number of data points in each category** among the k neighbors. **New data point will belong to class that has the most neighbors**.

For regression(average), value for the **new data point will be the average of the k neighbors**.

- **Euclidean distance**
- **Manhattan distance**
- **Hamming Distance**
- **Minkowski Distance**

Pros of K Nearest Neighbors

- Simple algorithm and hence easy to interpret the prediction.
- Nonparametric, so makes no assumption about the underlying data pattern.

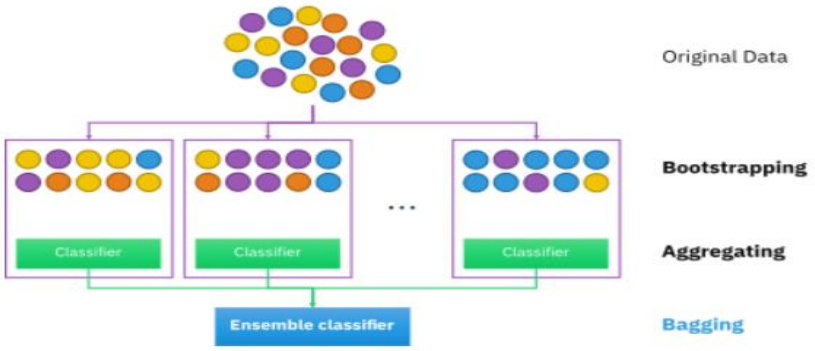
- Used for both classification and Regression.
- Training step is **much faster for nearest neighbor** compared to another machine learning algorithms

Cons of K Nearest Neighbors

- KNN is computationally expensive as it searches the nearest neighbors for the new point at the prediction stage.
- High memory requirement as KNN must store all the data points.
- Prediction stage is very costly.
- **Sensitive to outliers, accuracy is impacted by noise or irrelevant data. Cons of K Nearest Neighbors**

Random Forest:

(Hyperparameters: {**n_estimatorsint, default=100}, {criterion{"gini", "entropy"}, default="gini"},{ max_features{"auto", "sqrt", "log2"}, int or float, default="auto"**})
Random forest works on the Bagging principle.



It operates by constructing a multitude of decision trees at training time and outputting the class that is the **mode** of the **classes (classification)** or **mean prediction (regression)** of the individual trees)

- **Each tree draws a random sample from the original data set when generating its splits, adding a further element of randomness that prevents overfitting.**

Feature and Advantages of Random Forest:

- It is one of the most **accurate learning algorithms** available. For many data sets, it produces a highly accurate classifier.
- It generates an **internal unbiased estimate of the generalization error** as the forest building progresses.
- It has an **effective method for estimating missing data and maintains accuracy when a large proportion of the data are missing.**

Disadvantages of Random Forest:

- Random forests have been observed to overfit for some datasets with noisy classification/regression tasks.
- **For data including categorical variables with different number of levels, random forests are biased in favor of those attributes with more levels**. Therefore, the variable importance scores from random forest are not reliable for this type of data.

Machine Learning theoretical concepts

By: Vikram Pal

Proximities: are calculated for each pair of cases/observations/sample points. If two cases occupy the same terminal node through one tree, their proximity is increased by one. At the end of the run of all trees, the proximities are normalized by dividing by the number of trees. Proximities are used in replacing **missing data, locating outliers, and producing illuminating low-dimensional views of the data**.

Decision Tree:

(Hyperparameters: criterion{"Gini", "entropy"}, splitter{"best", "random"}, max_features int, float or {"auto", "sqrt", "log2"})

Regression: Impurity metric that is suitable for continuous variables, so we define the impurity measure using the **weighted mean squared error** (**MSE**) of the children's nodes instead

Classification: - In classification, entropy is the most common impurity measure or splitting criteria.

What causes overfitting in decision tree?

In decision trees, over-fitting occurs when the tree is designed to perfectly fit all samples in the training data set. Thus, it ends up with branches with **strict rules of sparse data**. Thus, this effects the accuracy when predicting samples that are not part of the training set.

How do you fix overfitting in decision tree?

Pre-pruning that stops growing the tree earlier, before it perfectly classifies the training set.
Post-pruning that allows the tree to perfectly classify the training set, and then post prune the tree.

Advantage:

- Easy to understand and interpret, perfect for visual representation.
- Can work with numerical and categorical features.
- Decision Trees **are not sensitive to noisy data or outliers** since, extreme values or outliers, never cause much reduction in Residual Sum of Squares(RSS), because they are never involved in the split.

Disadvantage:

- main drawback of Decision Tree is that it generally leads to overfitting of the data.
- Information gain is defined as the reduction in entropy due to the selection of a particular attribute. Information gain biases the Decision Tree against considering attributes with **a large number of distinct values which might lead to overfitting**.

Gini Impurity:

$$Gini = 1 - \sum_{i=1}^{n} (p_i)^2$$

Entropy:

$$Entropy(D_1) = -\sum_{i=1}^{m} p_i \log_2 p_i$$

Machine Learning theoretical concepts

By: Vikram Pal

Information Gain:

$$Gain\ Ratio = \frac{Information\ Gain}{SplitInfo} = \frac{Entropy\ (before) - \sum_{j=1}^{K} Entropy(j, after)}{\sum_{j=1}^{K} w_j\ log_2\ w_j}$$

Naïve Bayes:

Bayes Theorem helps us to find the probability of a hypothesis given our prior knowledge.

It can also be trained on small dataset. This algorithm assumes as all the variables in the dataset is "Naive" i.e., not correlated to each other.

Boosting:

https://www.youtube.com/watch?v=kho6oANGu_A

Function	XGBoost	CatBoost	Light GBM
Important parameters which control overfitting	1. **learning_rate or eta** – optimal values lie between 0.01-0.2 2. **max_depth** 3. **min_child_weight:** similar to min_child leaf; default is 1	1. **Learning_rate** 2. **Depth** - value can be any integer up to 16. Recommended - [1 to 10] 3. No such feature like min_child_weight 4. **l2-leaf-reg**: L2 regularization coefficient. Used for leaf value calculation (any positive integer allowed)	1. **learning_rate** 2. **max_depth**: default is 20. Important to note that tree still grows leaf-wise. Hence it is important to tune **num_leaves** (number of leaves in a tree) which should be smaller than 2^(max_depth). It is a very important parameter for LGBM 3. **min_data_in_leaf**: default=20, alias= min_data, min_child_samples
Parameters for categorical values	Not Available	1. **cat_features**: It denotes the index of categorical features 2. **one_hot_max_size**: Use one-hot encoding for all features with number of different values less than or equal to the given parameter value (max – 255)	1. **categorical_feature:** specify the categorical features we want to use for training our model
Parameters for controlling speed	1. **colsample_bytree**: subsample ratio of columns 2. **subsample**: subsample ratio of the training instance 3. **n_estimators**: maximum number of decision trees; high value can lead to overfitting	1. **rsm:** Random subspace method. The percentage of features to use at each split selection 2. No such parameter to subset data 3. **iterations**: maximum number of trees that can be built; high value can lead to overfitting	1. **feature_fraction:** fraction of features to be taken for each iteration 2. **bagging_fraction:** data to be used for each iteration and is generally used to speed up the training and avoid overfitting 3. **num_iterations:** number of boosting iterations to be performed; default=100

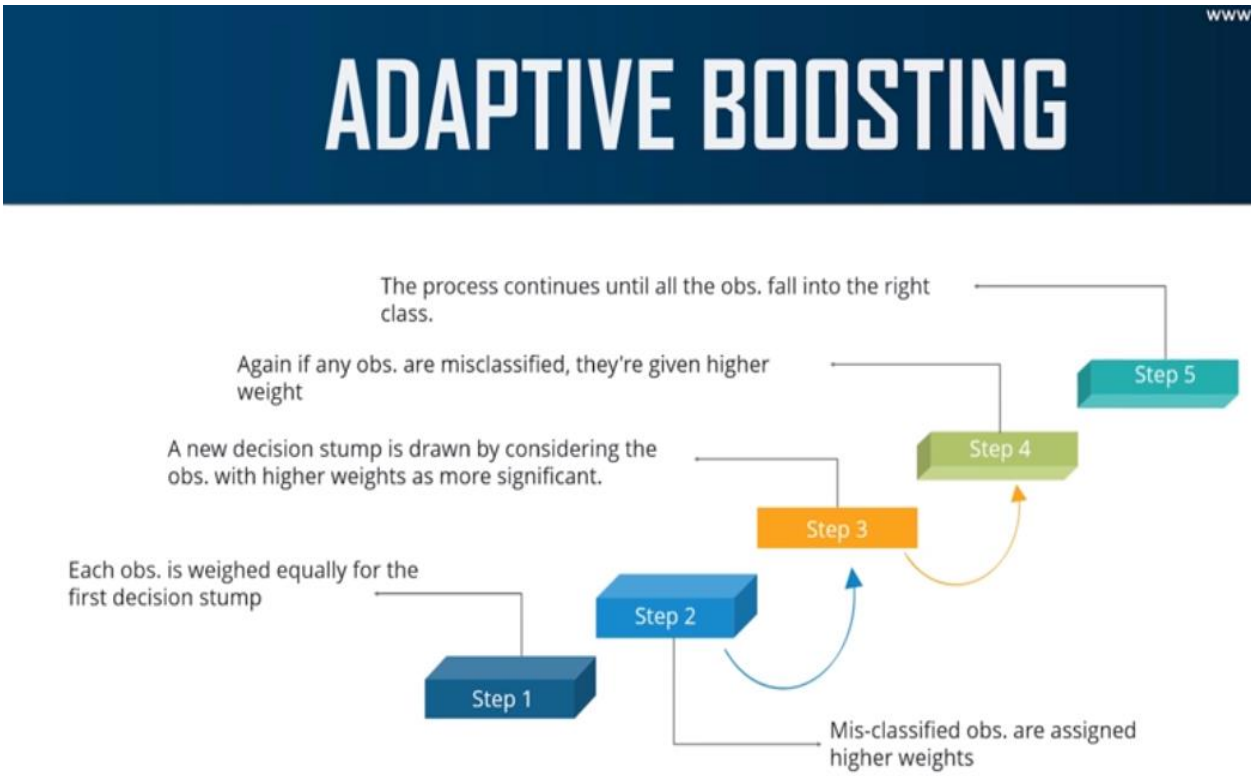
Unlike CatBoost or LGBM, XGBoost cannot handle categorical features by itself, it only accepts numerical values similar to Random Forest. Therefore, one must perform various encodings like label encoding, mean encoding or one-hot encoding before supplying categorical data to XGBoost.

Catboost offers a new technique called **Minimal Variance Sampling** (MVS), **which is a weighted sampling version of Stochastic Gradient Boosting**. ... XGboost is not using any weighted sampling techniques, which makes its splitting process slower compared to GOSS and MVS

Machine Learning theoretical concepts

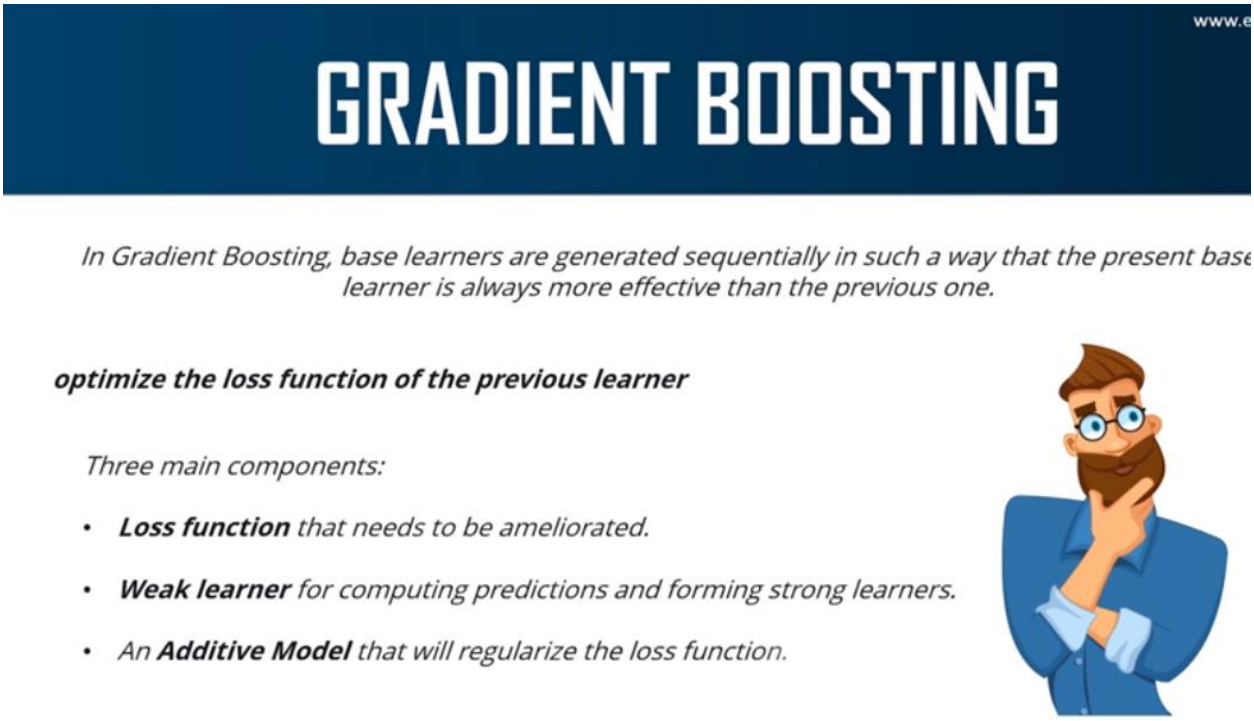
By: Vikram Pal

AdaBoost	GradientBoost
Both AdaBoost and Gradient Boost use a base weak learner and they try to boost the performance of a weak learner by iteratively shifting the focus towards problematic observations that were difficult to predict. At the end, a strong learner is formed by addition (or weighted addition) of the weak learners.	
In AdaBoost, shift is done by up-weighting observations that were misclassified before.	Gradient boost identifies difficult observations by large residuals computed in the previous iterations.
In AdaBoost "shortcomings" are identified by high-weight data points.	In Gradientboost "shortcomings" are identified by gradients.
Exponential loss of AdaBoost gives more weights for those samples fitted worse.	Gradient boost further dissect error components to bring in more explanation.
AdaBoost is considered as a special case of Gradient boost in terms of loss function, in which exponential losses.	Concepts of gradients are more general in nature.



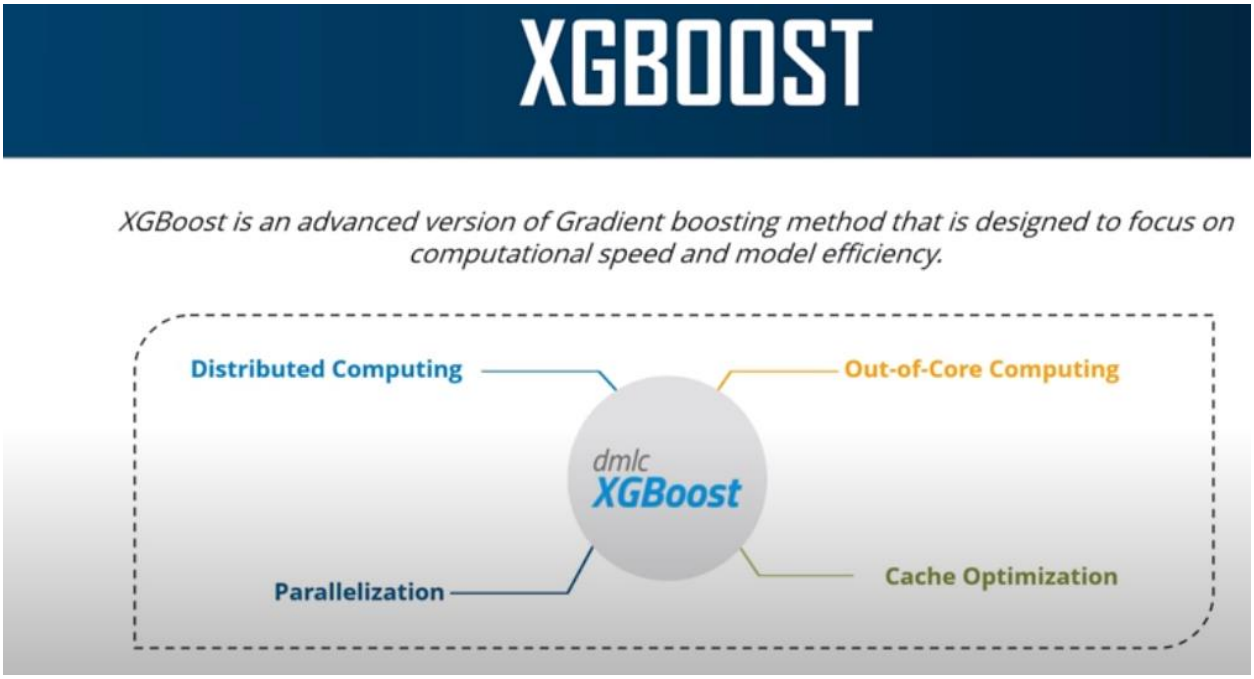
Machine Learning theoretical concepts

By: Vikram Pal



XGboots:

XGBoost is a **decision-tree-based ensemble Machine Learning algorithm** that uses a **gradient boosting framework** (Bagging In Random Forest). when it comes to small-to-medium structured/tabular data, decision tree-based algorithms are considered best-in-class right now.



Note pointed to check: How in boosting algo output is used for next model (learner)

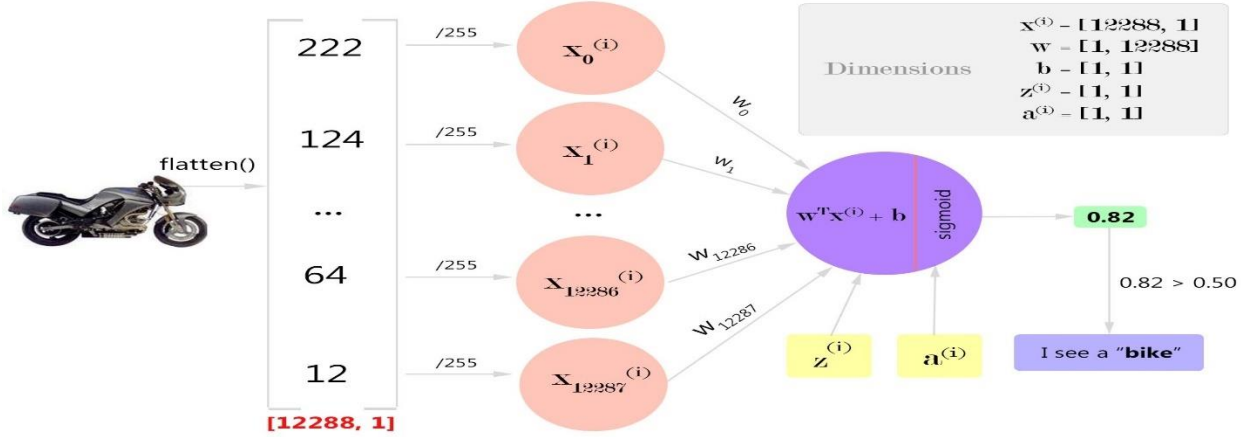
Machine Learning theoretical concepts

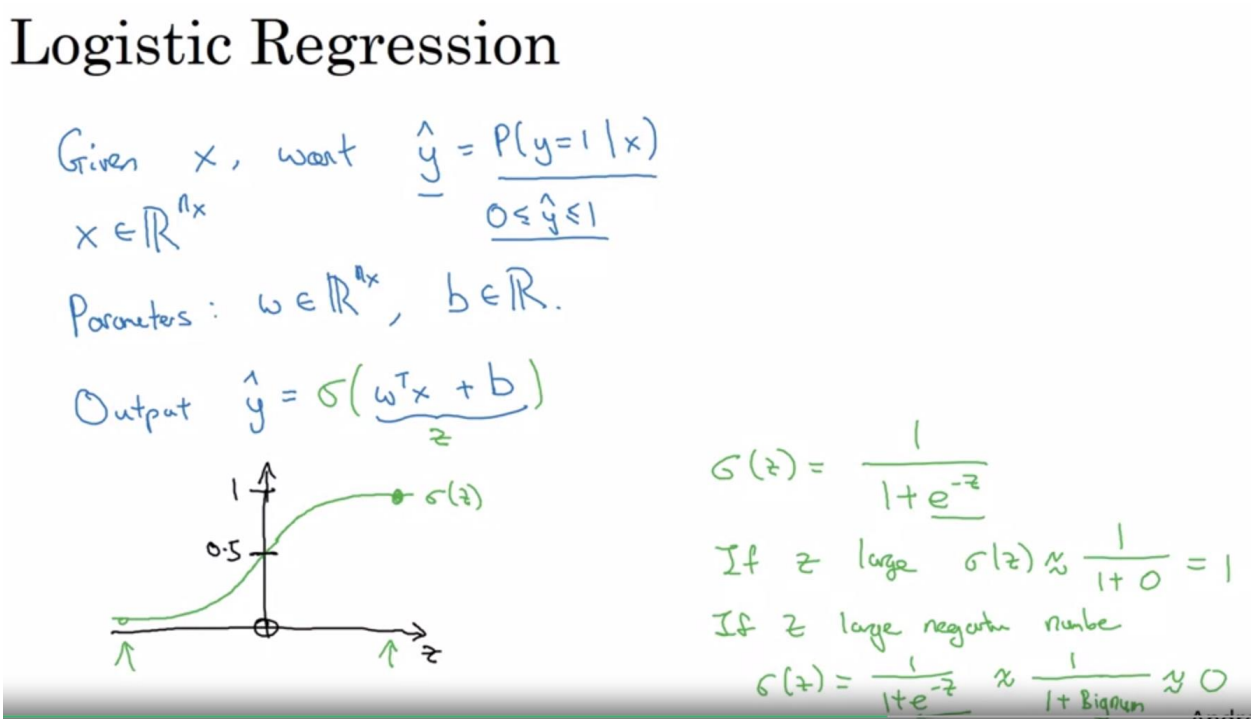
By: Vikram Pal

Logistic Regression:

It's a classification algorithm that is used where the **target variable is of categorical nature**.

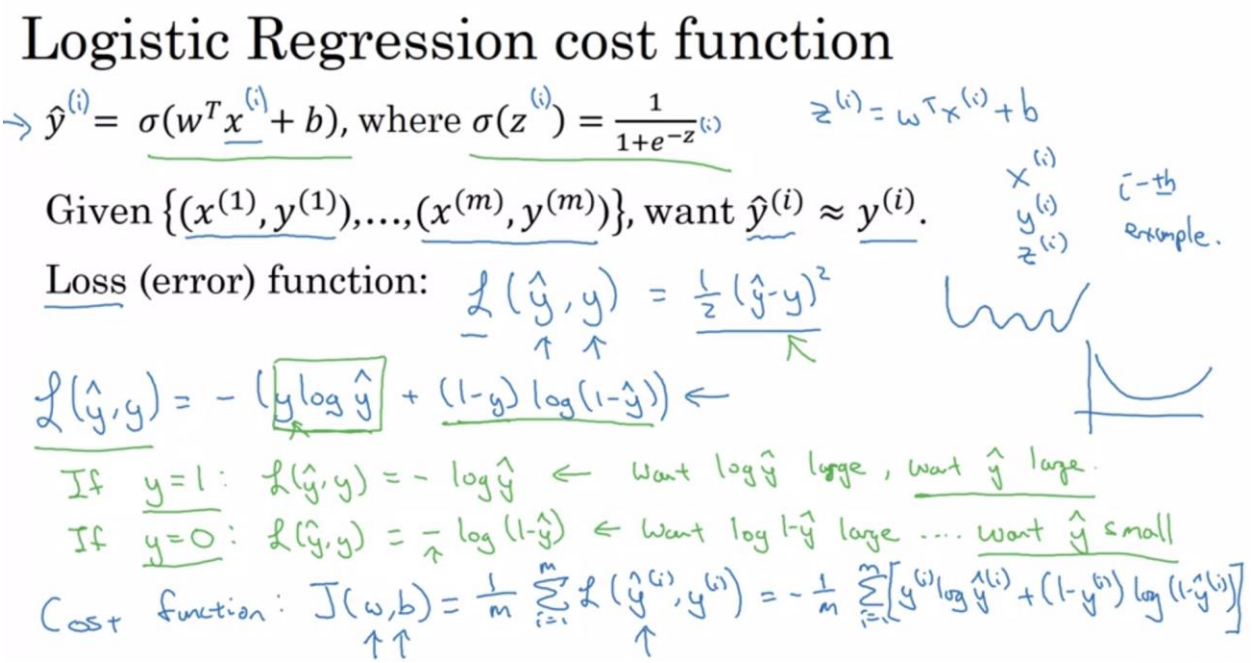
The main objective behind Logistic Regression is to determine the **relationship between features and the probability of a particular outcome.**



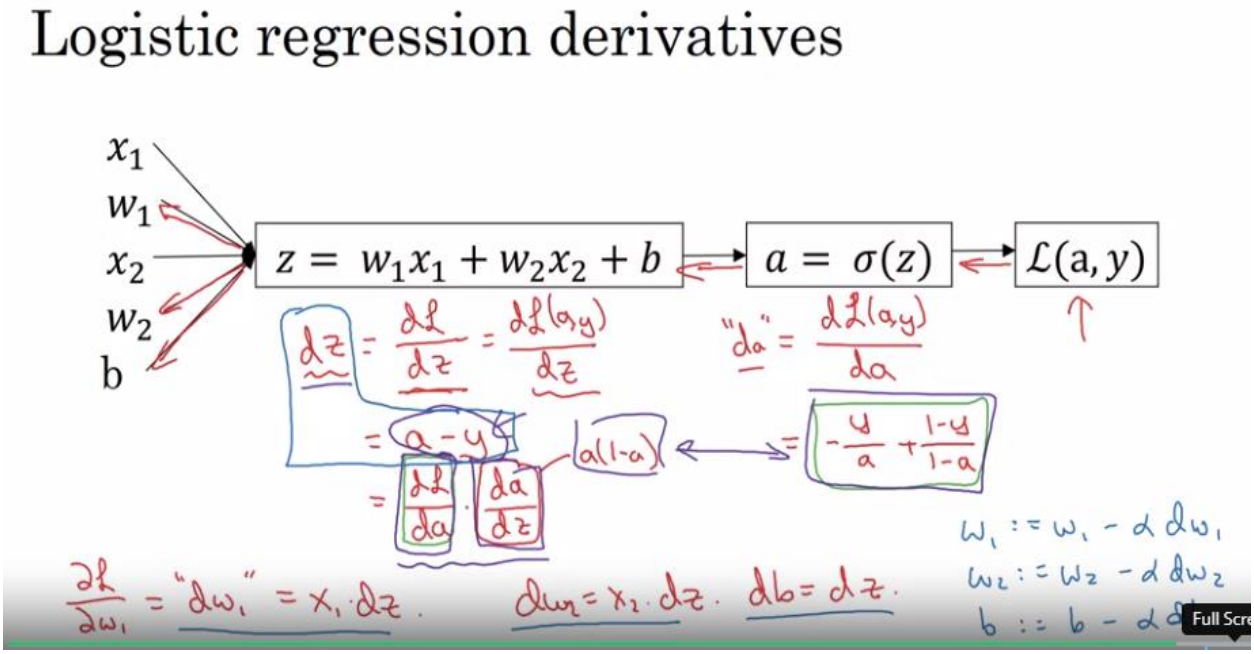


Machine Learning theoretical concepts

By: Vikram Pal



In Logistic Regression, we log loss function. Because it gives us non-convex function.



Some of the assumptions of Logistic Regression are as follows:

1. It assumes that there is minimal or **no multicollinearity** among the independent variables i.e, predictors are not correlated.

2. There should be a **linear relationship between the logit of the outcome and each predictor variable**. The logit function is described as **logit(p) = log(p/(1-p))**, where p is the probability of the target outcome.

3. Sometimes to predict properly, it usually requires a **large sample size**.

4. The Logistic Regression which has **binary classification** i.e, two classes assume that the target variable is binary, and ordered Logistic Regression requires the target variable to be ordered.
For example, Too Little, About Right, Too Much.
5. It assumes there is **no dependency** between the observations.

Multiclass classification using Logistic Regression:

Yes, in order to deal with multiclass classification using Logistic Regression, the most famous method is known as the **one-vs-all approach**. In this approach, a number of models are trained, which is equal to the number of classes. These models work in a specific way.

For Example, the first model classifies the datapoint depending on whether it belongs to class 1 or some other class(not class 1); the second model classifies the datapoint into class 2 or some other class(not class 2) and so-on for all other classes.
So, in this manner, each data point can be checked over all the classes.

Linear Regressions cannot be used in the case of binary classification due to the following reasons:

1. **Distribution of error terms:** It assumes that error terms are normally distributed. But this assumption does not hold true in the case of binary classification.
2. **Model output:** In Linear Regression, the output is continuous(or numeric) while in the case of binary classification, an output of a continuous value does not make sense. For binary classification problems, Linear Regression may predict values that can go beyond the range between 0 and 1
3. **The variance of Residual errors:** Linear Regression assumes that the variance of random errors is constant. This assumption is also not held in the case of Logistic Regression

R2 vs Adjusted R2:

R-squared:

R2 explains the **degree to which your input variables explain the variation of your output / predicted variable**. So, if R-square is 0.8, it means 80% of the variation in the output variable is explained by the input variables. So, in simple terms, higher the R squared, the more variation is explained by your input variables and hence better is your model.

$$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \overline{y})^2}$$

$$\mathrm{R}^2 = 1 - \frac{\text{Unexplained Variation}}{\text{Total Variation}}$$

Adjusted R2:

However, the problem with R-squared is that it will either stay the same or increase with addition of more variables, even if they do not have any relationship with the output variables. This is where “Adjusted R

square" comes to help. **Adjusted R-square penalizes you for adding variables which do not improve your existing model.**

Hence, if you are building **Linear regression on multiple variables**, it is always suggested that you use Adjusted R-squared to judge goodness of model. In case you only have one input variable, R-square and Adjusted R squared would be exactly same.

$$R_a^2 = 1 - \left[\left(\frac{n-1}{n-k-1}\right) \times (1 - R^2)\right]$$

where:
n = number of observations
k = number of independent variables
R_a^2 = adjusted R^2

Lasso(L1) [sparsity]and Ridge(L2)[simplicity]:

They are some of the simple techniques to reduce model complexity and **prevent over-fitting** which may result from simple linear regression.

Lasso Regression(L1):

The cost function for Lasso (**least absolute** shrinkage and selection operator) regression can be written as (L1 regularization)

$$\sum_{i=1}^{M} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{M} \left(y_i - \sum_{j=0}^{p} w_j \times x_{ij} \right)^2 + \lambda \sum_{j=0}^{p} |w_j| \qquad (1.4)$$

It is also called **regularization for sparsity**. As the name suggests, it is used to handle sparse vectors which consist of mostly zeroes. Sparse vectors typically result in very high-dimensional feature vector space. Thus, the model becomes very difficult to handle.

L1 regularization forces the weights of uninformative features to be zero by subtracting a small amount from the weight at each iteration and thus making the weight zero, eventually.

L1 regularization penalizes |weight|.

Ridge Regression(L2):

In ridge regression, the cost function is altered by adding a penalty equivalent to square of the magnitude of the coefficients. (L2 regularization)

$$\sum_{i=1}^{M} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{M} \left(y_i - \sum_{j=0}^{p} w_j \times x_{ij} \right)^2 + \lambda \sum_{j=0}^{p} w_j^2 \qquad (1.3)$$

It is also called **regularization for simplicity**. If we take the model complexity as a function of weights, the **complexity of a feature is proportional to the absolute value of its weight**.

Machine Learning theoretical concepts

By: Vikram Pal

$$y = w_1 x_1 + w_1 x_1 + \ldots + w_n x_n$$

$$L2\ regularization\ terms = w_1^2 + w_2^2 + \ldots + w_n^2$$

L2 regularization forces weights toward zero but it does not make them exactly zero. L2 regularization acts like a force that removes a small percentage of weights at each iteration. Therefore, weights will never be equal to zero.

L2 regularization penalizes $(weight)^2$

There is an additional parameter to tune the L2 regularization term which is called **regularization rate** (lambda). Regularization rate is a scalar and multiplied by L2 regularization term.

Chose right machine learning Algorithm:

- Size of the training data
- Accuracy and interpretability of the output
- Speed or Training time
- Linearity
- Number of features

Size of the training data:

If the training data is smaller or the training data has a fewer number of observation and higher number of features.

Chose algorithm with a high bias/low variance like linear regression, Naive Bayes or linear SVM.

If the training data is larger/sufficient or the training data has a greater number of observations as compared to number of features.

Chose algorithm with a **low bias/high variance** like **KNN, Decision tree or Kernel SVM**.

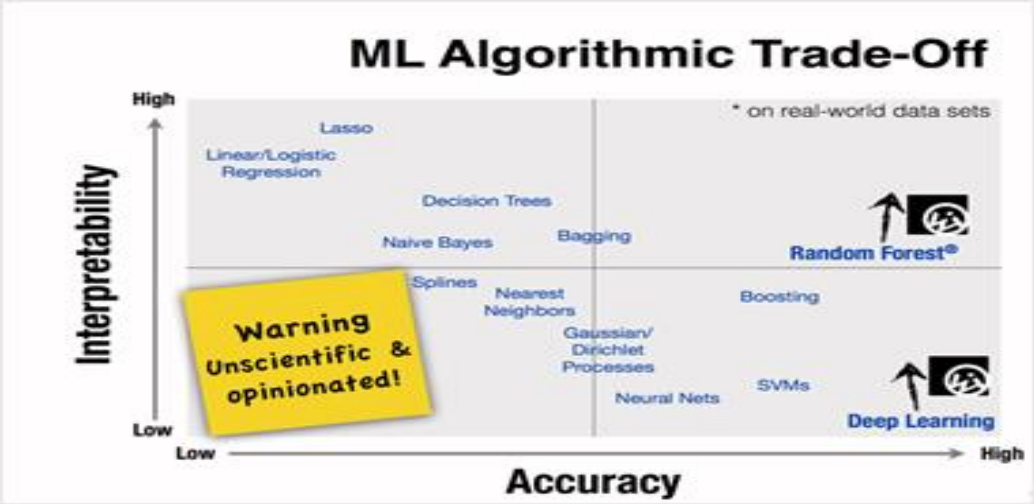
Accuracy and interpretability of the output:

Accuracy means that the model **predicts the response for a given observation, that is close to the true value of that observation**. (High flexibility and low interpretability)

Interpretability means that one can easily understand how the **individual predictor is associated with the response**. (Low flexibility and high interpretability)

Machine Learning theoretical concepts

By: Vikram Pal



Now, to use which algorithm depends on the objective of the **business problem**.

- If inference is the goal, then restrictive models are better as they are much more interpretable.
- Flexible models are better if higher accuracy is the goal.

Speed or Training time

High Accuracy mean higher training time. Also, the algorithms require more time to train on large training data. In real world application, the choice of algorithm is driven by these two factors.
Algorithm like Naïve Bayes, linear Regression and Logistic regression are easy to implement and quick to run.
Algorithm like SVM which involve tuning of parameters, Neural network with convergence time and random forest need a lot of time to train on data.

Linearity:

The many algorithms work on the assumption that the data is separated by a straight line (or it analog in higher dimension). This example includes linear regression, Logistic regression and SVM.

However, not all data is not linear. In that case, **we need some algorithm to work in nonlinear and high dimension data.** For example, random forest, kernel SVM and neural net.

Number of features:

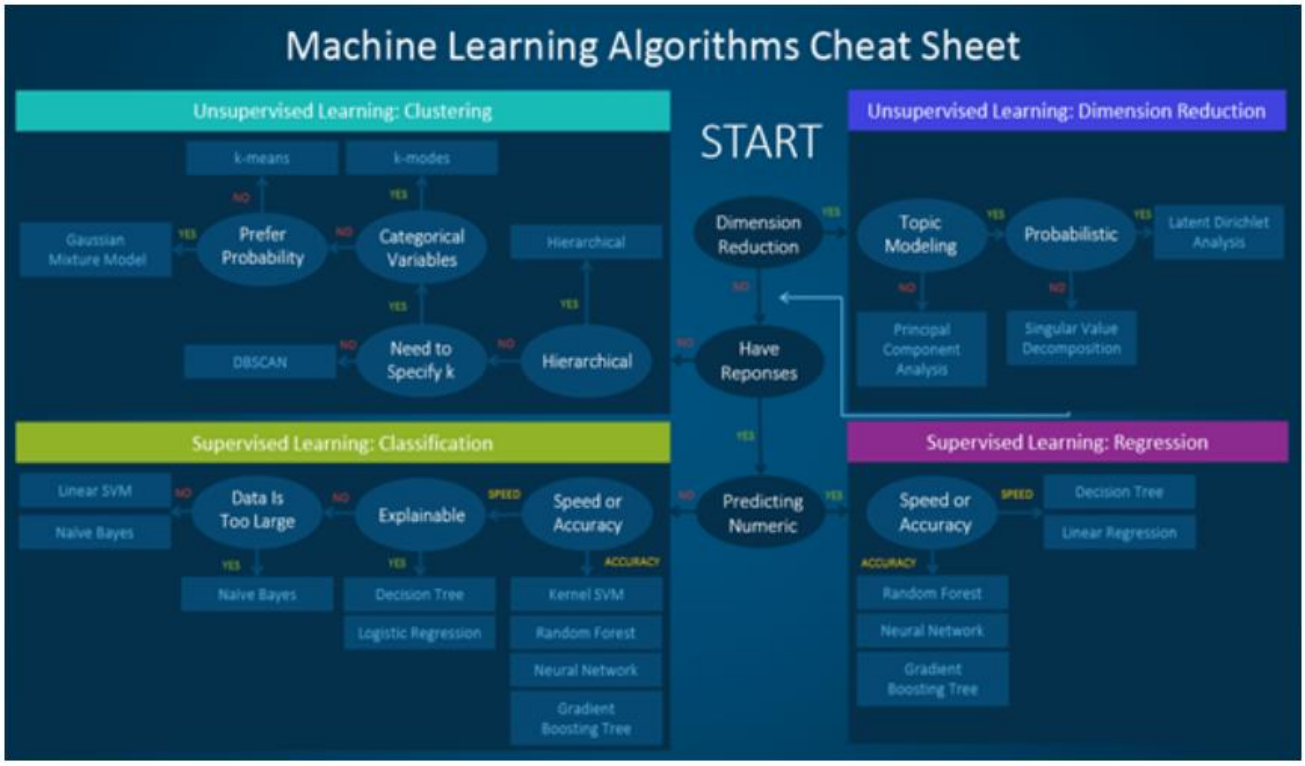
The dataset may have many features that may not all be relevant and significant. For a certain type of data, such as genetics or textual, the number of features can be large compared to the number of data points.

Many features can bog down some learning algorithms, making training time unfeasibly long. SVM is better suited in case of data with large feature space and lesser observations.

Machine Learning theoretical concepts

By: Vikram Pal

*** PCA and feature selection techniques should be used to reduce dimensionality and select important features.



Machine Learning theoretical concepts

By: Vikram Pal

Model Evaluation Metrics:

Precision, Recall, and F1 Score:

		Predicted Class: Positive	Predicted Class: Negative	
Actual Class	Positive	True Positive (TP)	False Negative (FN) **Type II Error**	**Sensitivity** $\frac{TP}{(TP+FN)}$
	Negative	False Positive (FP) **Type I Error**	True Negative (TN)	**Specificity** $\frac{TN}{(TN+FP)}$
		Precision $\frac{TP}{(TP+FP)}$	**Negative Predictive Value** $\frac{TN}{(TN+FN)}$	**Accuracy** $\frac{TP+TN}{(TP+TN+FP+FN)}$

Precision is equal to true positive by total predicted positive. It is a good measure when the false positive cost is high.

For example: In case of email spam detection, an email is identified as spam (false positive) that in actuality not a spam email. By the user might can loose valuable email.

Recall is equal to true positive by total actual positive. It is a good measure when the false negative cost is too high.

For example: case of fraud detection and sick patient detection

Accuracy:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Machine Learning theoretical concepts

By: Vikram Pal

F1 is used when we seek a balance b/w precision and recall.

Difference b/w accuracy and F1 score.

F1 Score is needed when you want to seek a balance between Precision and Recall. Right…so what is the difference between F1 Score and Accuracy then?

We have previously seen that accuracy can be largely contributed by a large number of True Negatives which in most business circumstances, we do not focus on much whereas **False Negative and False Positive usually has business costs** (tangible & intangible) thus F1 Score might be a better measure to use if we need to seek a balance **between Precision and Recall** AND there is an uneven class distribution (large number of Actual Negatives).

$$F1 = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

Micro-Averaged and Weighted Average:

Micro-averaged: all samples equally contribute to the final averaged metric. Macro-averaged: all classes equally contribute to the final averaged metric.

Weighted-averaged: each classes' contribution to the average is weighted by its size.

Type I Error: False positive (rejection of a true null hypothesis)
Type II Error: False negative (non-rejection of a false null hypothesis)

AUC-ROC Curve:

https://www.analyticsvidhya.com/blog/2020/06/auc-roc-curve-machine-learning/

Sensitivity / True Positive Rate / Recall:

Sensitivity tells us what proportion of the positive class got correctly classified.

$$Sensitivity = \frac{TP}{TP + FN}$$

A simple example would be to determine what proportion of the actual sick people were correctly detected by the model.

False Negative Rate:

False Negative Rate (FNR) tells us what proportion of the positive class got incorrectly classified by the classifier.

$$FNR = \frac{FN}{TP + FN}$$

A higher TPR and a lower FNR is desirable since we want to correctly classify the positive class.

Specificity / True Negative Rate:

Specificity tells us what proportion of the negative class got correctly classified.

Machine Learning theoretical concepts

By: Vikram Pal

$$Specificity = \frac{TN}{TN + FP}$$

Taking the same example as in Sensitivity, Specificity would mean determining the proportion of healthy people who were correctly identified by the model.

False Positive Rate:

FPR tells us what proportion of the negative class got incorrectly classified by the classifier.

$$FPR = \frac{FP}{TN + FP} = 1 - Specificity$$

A higher TNR and a lower FPR is desirable since we want to correctly classify the negative class.

Probability of Predictions:

A machine learning classification model can be used:

a. predict the actual class of the data point directly
b. predict its probability of belonging to different classes.

Setting different thresholds for classifying positive class for data points will change the Sensitivity and Specificity of the model.

And one of these thresholds will probably give a better result than the others, depending on whether we are aiming to lower the number of False Negatives or False Positives.

ID	Actual	Prediction Probability	>0.6	>0.7	> 0.8	Metric
1	0	0.98	1	1	1	
2	1	0.67	1	0	0	
3	1	0.58	0	0	0	
4	0	0.78	1	1	0	
5	1	0.85	1	1	1	
6	0	0.86	1	1	1	
7	0	0.79	1	1	0	
8	0	0.89	1	1	1	
9	1	0.82	1	1	1	
10	0	0.86	1	1	1	
			0.75	0.5	0.5	TPR
			1	1	0.66	FPR
			0	0	0.33	TNR
			0.25	0.5	0.5	FNR

What is the AUC-ROC curve?

The Receiver Operator Characteristic (ROC) curve is an evaluation metric for binary classification problems. It is a probability curve that plots the **TPR against FPR at various threshold values and essentially separates the 'signal' from the 'noise'.**

The Area Under the Curve (AUC) is the measure of the **ability of a classifier to distinguish between classes and is used as a summary of the ROC curve**.

a. AUC=1 perfectly distinguish between all the Positive and the Negative class points correctly.
b. AUC=0 the classifier would be predicting all Negatives as Positives, and all Positives as Negatives.
c. 0.5<AUC<1 there is a high chance that the classifier will be able to distinguish the positive class values from the negative class values.
d. AUC=0.5, then the classifier is not able to distinguish between Positive and Negative class points.

Parameters vs Hyperparameters:-

Some examples of model parameters include: (model building)

- The weights in an artificial neural network.
- The support vectors in a support vector machine.
- The coefficients in a linear regression or logistic regression.

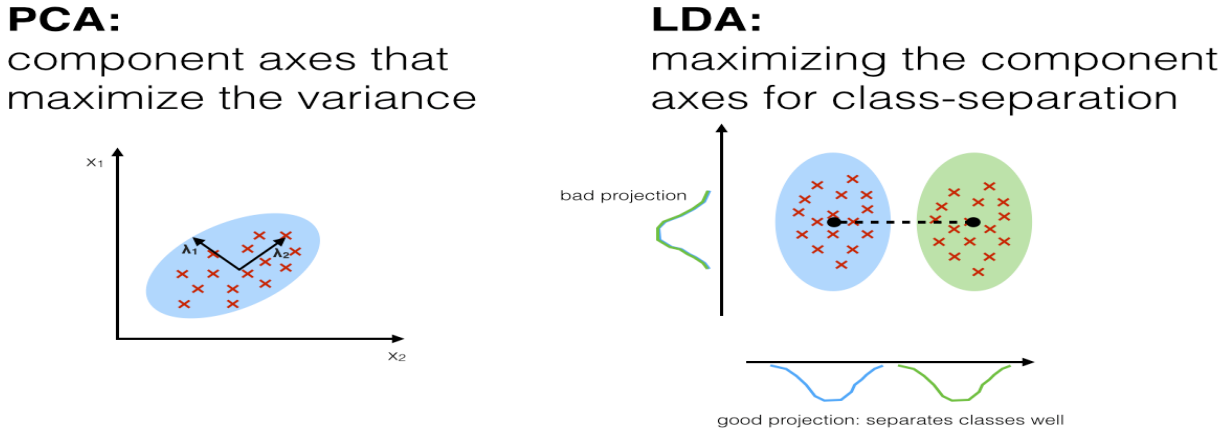
Machine Learning theoretical concepts

By: Vikram Pal

Some examples of model hyperparameters include: (fine tuning of model)

- The learning rate for training a neural network.
- The C and sigma hyperparameters for support vector machines.
- The k in k-nearest neighbors.

Dimensionality Reduction:



Principal Component Analysis (Unsupervised):

It reduces the dimension of a d-dimensional dataset by projecting it onto a (k)-dimensional subspace (where k<d) . There is a certain step to perform PCA:

1. Standardize the dataset.
2. Find the Eigenvalue and Eigenvectors using **covariance matrix** or correlation matrix.
3. Sort the Eigenvectors in descending order.
4. Create a projection matrix w using top K Eigenvectors.
5. Transform the original dataset x using projection matrix to obtain k-dimensional feature subspace Y.

Picking Principal Components Using the Explained Variance:

we want to see how much of the variance in data is explained by each one of these components. It is a convention to use 95% explained variance.

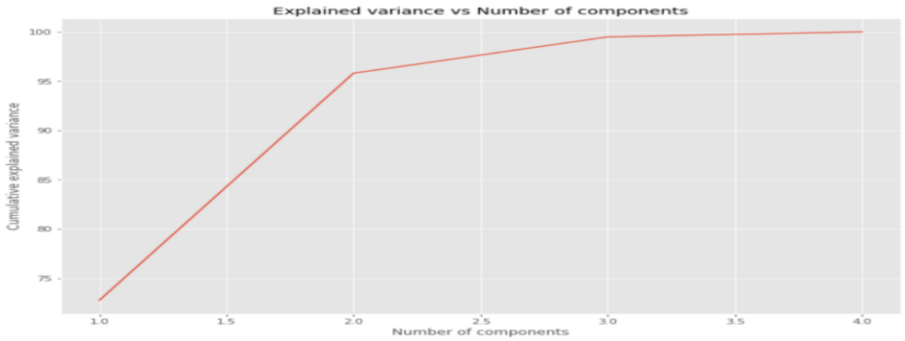
Determining how many components

Some rules to guide in choosing the number of components to keep:

- Keep components with eigenvalues greater than 1, as they add value (because they contain more information than a single variable). This rule tends to keep more components than is ideal
- Visualize the eigenvalues in order from highest to lowest, connecting them with a line. Upon visual inspection, keep all the components whose eigenvalue falls above the point where the slope of the line changes the most drastically, also called the “elbow”
- Including variance cut-offs where we only keep components that explain at least 95% of the variance in the data

Machine Learning theoretical concepts

By: Vikram Pal

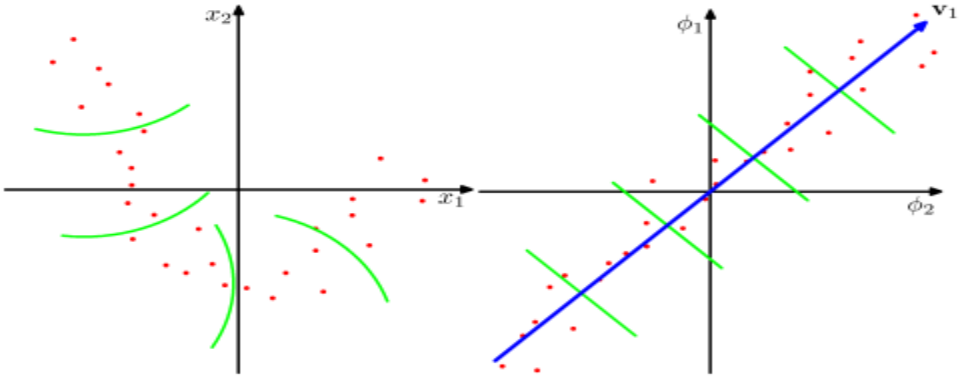


Linear Discriminant Analysis (Supervised): (d-dimensional mean vectors for the different classes from the dataset)

It is used to project a feature space onto a small subspace while maintaining the class discriminatory information.
There is a certain step to perform LDA:

1. Compute the **d-dimensional mean vectors for the different classes from the dataset**.
2. Compute the **scatter matrices** (in b/w classes and within class scatter matrices)
3. Compute Eigenvectors and Eigenvalues from matrices
4. Sort the Eigenvector and choose k eigenvectors with largest eigenvalues to form a d * k dimensional matrix (d dataset dimension)
5. Use this d*k matrix to transform the samples onto the new subspace.

Kernal PCA:



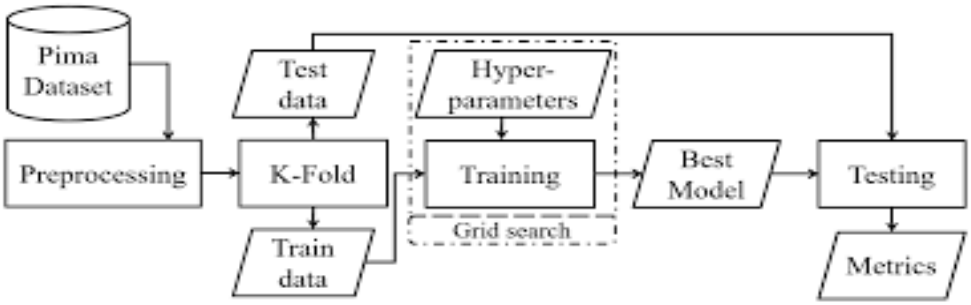
The data points here (on the left) are located mostly along a curve in 2D. PCA cannot reduce the dimensionality from two to one, because the points are not located along a straight line. **But still, the data are "obviously" located around a one-dimensional non-linear curve. So, while PCA fails, there must be another way!** And indeed, kernel PCA can find this non-linear manifold and discover that the data are in fact nearly one-dimensional.
It does so by mapping the data into a higher-dimensional space. This can indeed look like a contradiction, but it is not. The data are mapped into a higher-dimensional space, but then turn out to lie on a lower dimensional subspace of it. So, you increase the dimensionality in order to be able to decrease it.
The essence of the "kernel trick" is that one does not actually need to *explicitly* consider the higher-dimensional space, so this potentially confusing leap in dimensionality is performed entirely undercover. The idea, however, stays the same.

Machine Learning theoretical concepts

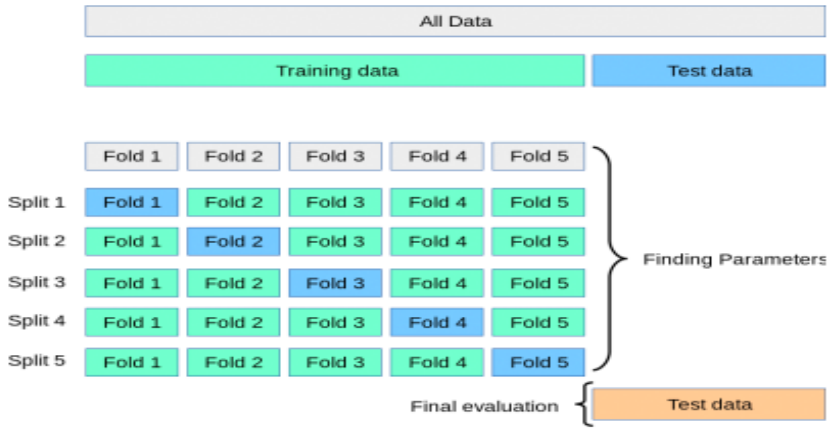
By: Vikram Pal

Note: PCA and LDA both are dimensionality reduction technique. PCA is unsupervised technique where LDA is supervised technique because of the relation to the dependent variable.

Model Selection:



K-Fold cross validation:



By training and testing the model K number of times on different subsets of the same training data we get a more accurate representation of how well our model might perform on data it has not seen before. In a K-fold CV we score the model after every iteration and compute the average of all scores to get a better representation of how the model performs compared to only using one training and validation set.

Grid Search:

One of the most popular approach to **tune machine learning hyperparameters** is called Grid search (Randomised Grid search cross validation)

In Randomised Grid Search Cross-Validation we start by creating a grid of hyperparameters we want to optimise with values that we want to try out for those hyperparameters.

Machine Learning theoretical concepts

By: Vikram Pal

```
# Create the model to be tuned
rf_base = RandomForestRegressor()

# Create the random search Random Forest
rf_random = RandomizedSearchCV(estimator = rf_base, param_distributions = rf_grid,
                               n_iter = 200, cv = 3, verbose = 2, random_state = 42,
                               n_jobs = -1)

# Fit the random search model
rf_random.fit(X_train_temp, y_train_temp)

# View the best parameters from the random search
rf_random.best_params
```

Parametric vs non-parametric model

Parametric models:

To summarise, parametric methods in Machine Learning usually take a model-based approach where we make an assumption with respect to form of the function to be estimated and then we select a suitable model based on this assumption in order to **estimate the set of parameters**.

Some examples of parametric methods in Machine Learning include Linear Discriminant Analysis, Naive Bayes and Perceptron.

Non-Parametric Methods:

On the other hand, non-parametric methods refer to a set of algorithms that do not make any underlying assumptions with respect to the form of the function to be estimated. And since no assumption is being made, such methods are capable of estimating the unknown function f that could be of any form.

Some examples of non-parametric methods in Machine Learning include Support Vector Machines and K-Nearest Neighbours.

Clustering:

K-mean Clustering:

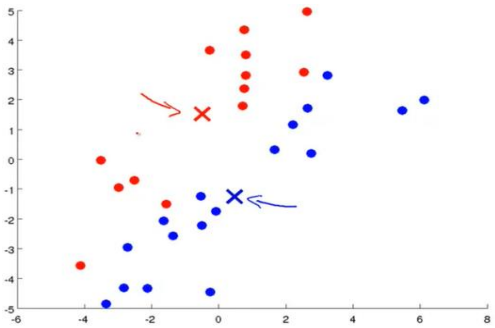


Fig.1 random centroids

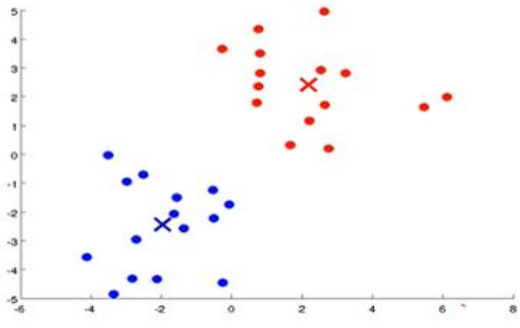


Fig.2 Average of all red and blue datapoints

There is a certain step to perform K-mean:

1. Choose the number K of clusters.
2. Select at random K points as centroids.

3. Assign each data point to the nearest Centroid.

(We must take an average of all the red dots that are assigned to the red cluster centroid and move the red cluster centroid to that average. We need to do the same for the blue cluster centroid.)

4. Computer and place the new centroids of each cluster.
5. Reassign each data point to the new closest centroid. If any reassignment took place, go back to step 4, otherwise go to finish.

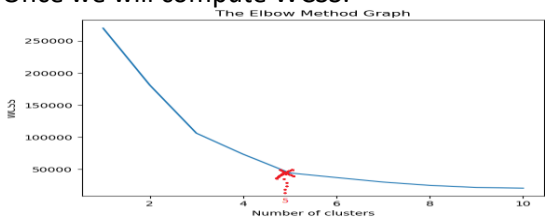
K-Mean initialization trap:

Random initialization trap is a problem that occurs in the K-means algorithm. In random initialization trap when the **centroids of the clusters to be generated are explicitly defined by the User then inconsistency may be created**, and this may sometimes **lead to generating wrong clusters in the dataset**.

Choosing the right number of clusters:

$$\text{WCSS} = \sum_{P_i \text{ in Cluster 1}} \text{distance}(P_i, C_1)^2 + \sum_{P_i \text{ in Cluster 2}} \text{distance}(P_i, C_2)^2 + \sum_{P_i \text{ in Cluster 3}} \text{distance}(P_i, C_3)^2$$

Once we will compute WCSS:



Number of clusters are 5 in the above diagram.

Hierarchical Clustering:

It is two types: **agglomerative(top-bottom approach) and Divisive(bottom-up approach).**

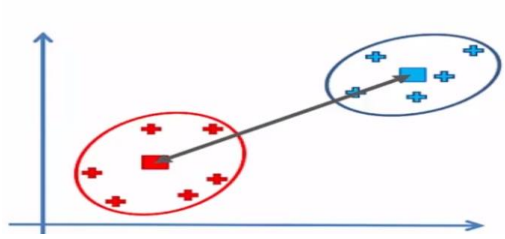
a. Make each data point a single point cluster --- > That form N clusters

b. Take the two closest data points and make them one cluster --- > that forms N-1 clusters.

c. Take the two closest data points and make them one cluster --- > that forms N-2 clusters.

d. Repeat Step 3 until there is only one cluster.

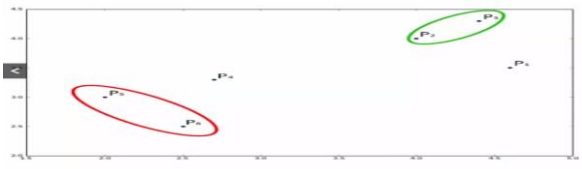
How to measure distance b/w two clusters?

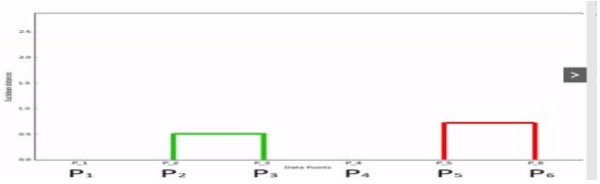


Distance Between Two Clusters:

- Option 1: Closest Points
- Option 2: Furthest Points
- Option 3: Average Distance
- Option 4: Distance Between Centroids

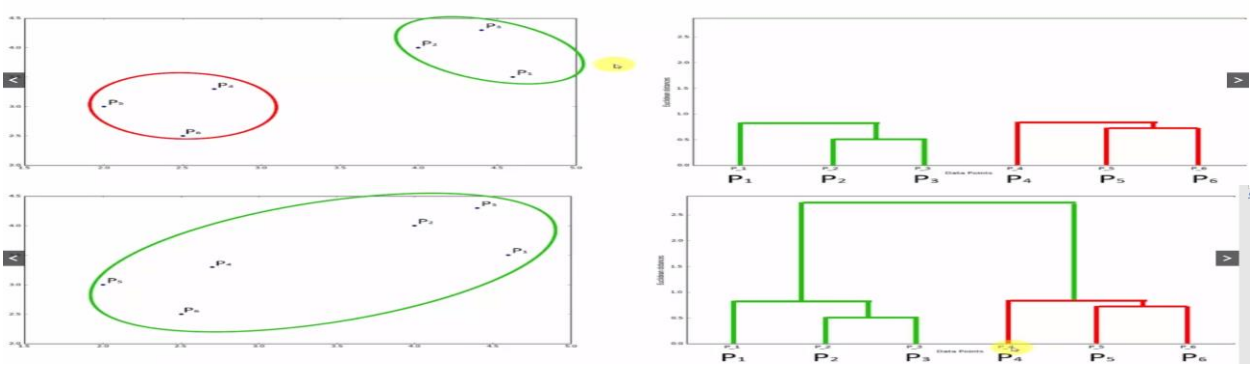
How do Dendrograms work?



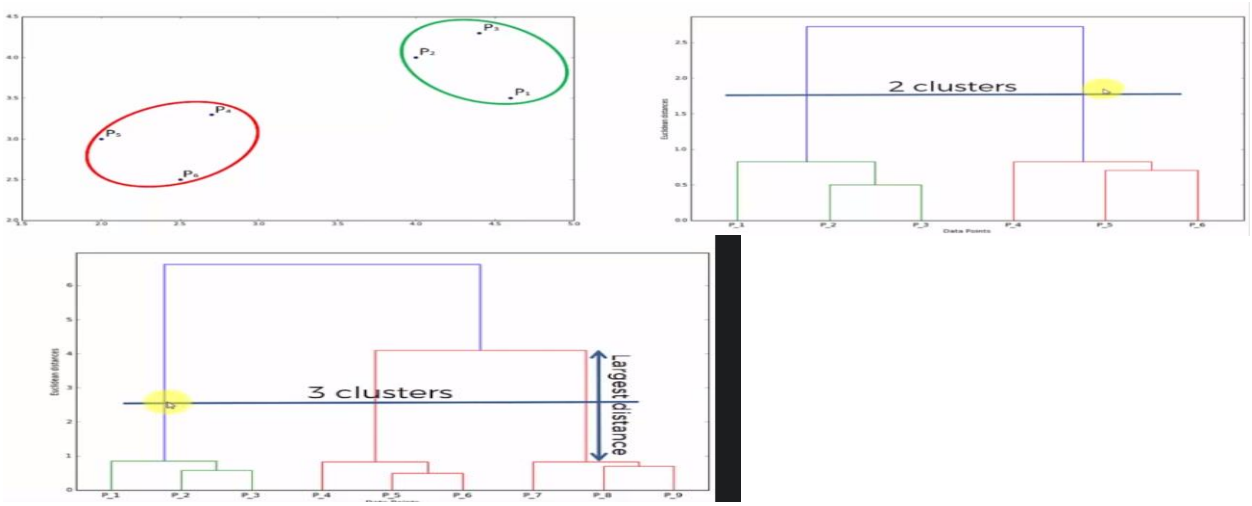


Machine Learning theoretical concepts

By: Vikram Pal



We will setup a threshold of Euclidean Distance that will give number of clusters (Number of lines cut with threshold point or largest distance)



Gradient descent (Which way to go and how big a step to take {Tangent gives us a sense of steepness of the slope}:

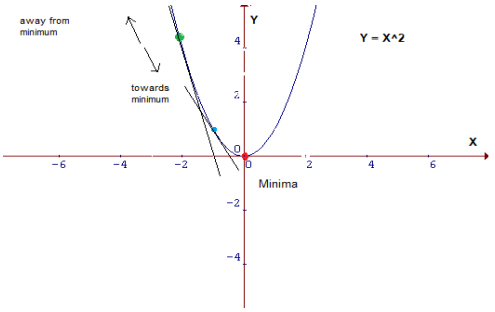
Gradient: a gradient is a partial derivative with respect to its inputs. A gradient measure how much the output of a function changes if you change the inputs a little bit. **You can also think of a gradient as the slope of a function**. Higher the gradient, steeper the slope and the faster a model can learn. If the slope is almost zero, the model stops to learn. A gradient simply measures the change in all weights with regard to the change in error.

It is one of the most popular algorithms to perform optimization and by far the most common way to optimize neural networks.

Essentially, there are two things that you should know to reach the minima, i.e., **which way to go and how big a step to take**. The **tangent gives us a sense of the steepness of the slope**.

Machine Learning theoretical concepts

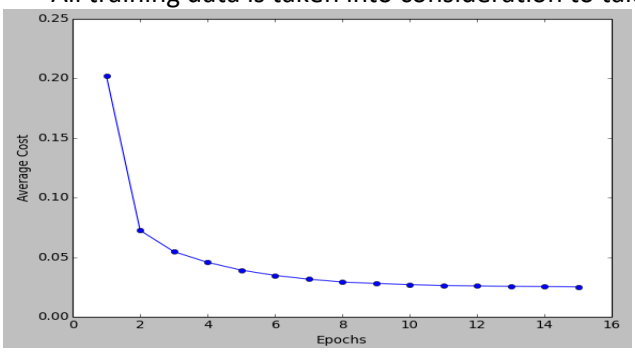
By: Vikram Pal



Type of Gradient Descents:

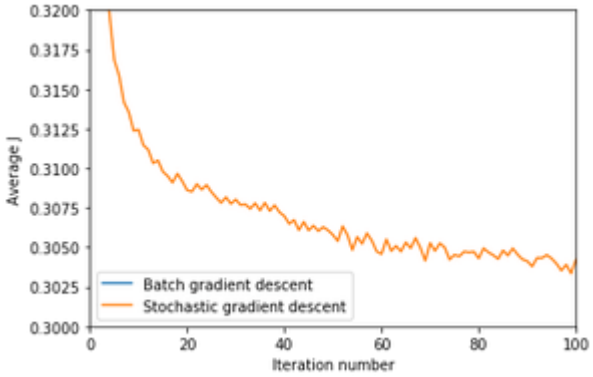
a. Batch Gradient Descent:

All training data is taken into consideration to take a single step.



b. Stochastic Gradient Descent:

we consider **one example at a time** to take a single step.

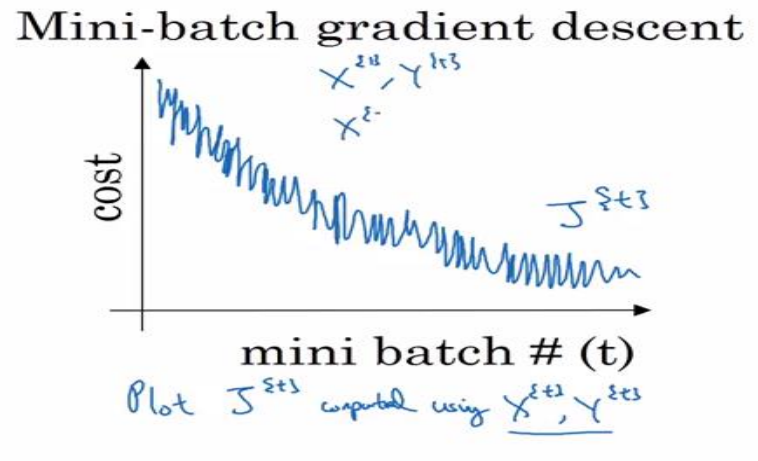


c. Mini Batch GD:

it is mixture of batch and Stochastic gradient descent.

Machine Learning theoretical concepts

By: Vikram Pal



Learning rate:

This size of steps taken to reach the minimum or bottom is called Learning Rate

Source

Different types of setting up a learning rate:

1. Learning Rate Schedules:

a. Constant Learning Rate

b. Time Based Decay

```
lr *= (1. / (1. + self.decay * self.iterations))
```

c. Step Decay:

Step decay schedule drops the learning rate by a factor every few epochs. The mathematical form of step decay is :

lr = lr0 * drop^floor(epoch / epochs_drop)

d. Exponential decay

`lr = lr0 * e^(-kt)`,

where `lr`, `k` are hyperparameters and `t` is the iteration number

Machine Learning theoretical concepts

By: Vikram Pal

2. Adaptive learning rate methods:

a. Momentum*:*

It is based on **exponential weighted average**.

An exponential moving average (EMA) is a type of moving average (MA) that places a greater weight and significance on the most recent data points. The exponential moving average is also referred to as the exponentially weighted moving average.

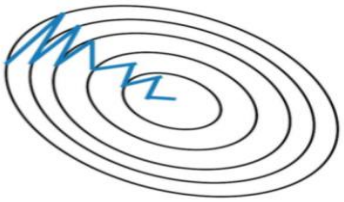
How does Momentum work?

The basic idea is to compute an exponentially weighted average of your gradients, and then use that gradient to update your weights instead

For instance, if we have a momentum of 0.90, we will take 90% of the previous direction plus 10% of the new direction and adjust weights accordingly -- multiplying that direction vector by the learning rate.

Momentum is a technique to **prevent sensitive movement**. When the gradient gets computed every iteration, it can have totally different direction and the steps make a zigzag path, which makes training very slow. Something like this.

Stochastic Gradient Descent **withhout** Momentum



Stochastic Gradient Descent **with** Momentum

On iteration t:

Compute dW, db on the current mini-batch

$$v_{dW} = \beta v_{dW} + (1-\beta)dW$$

$$v_{db} = \beta v_{db} + (1-\beta)db$$

$$W = W - \alpha v_{dW}, \quad b = b - \alpha v_{db}$$

Hyperparameters: α, β $\beta = 0.9$

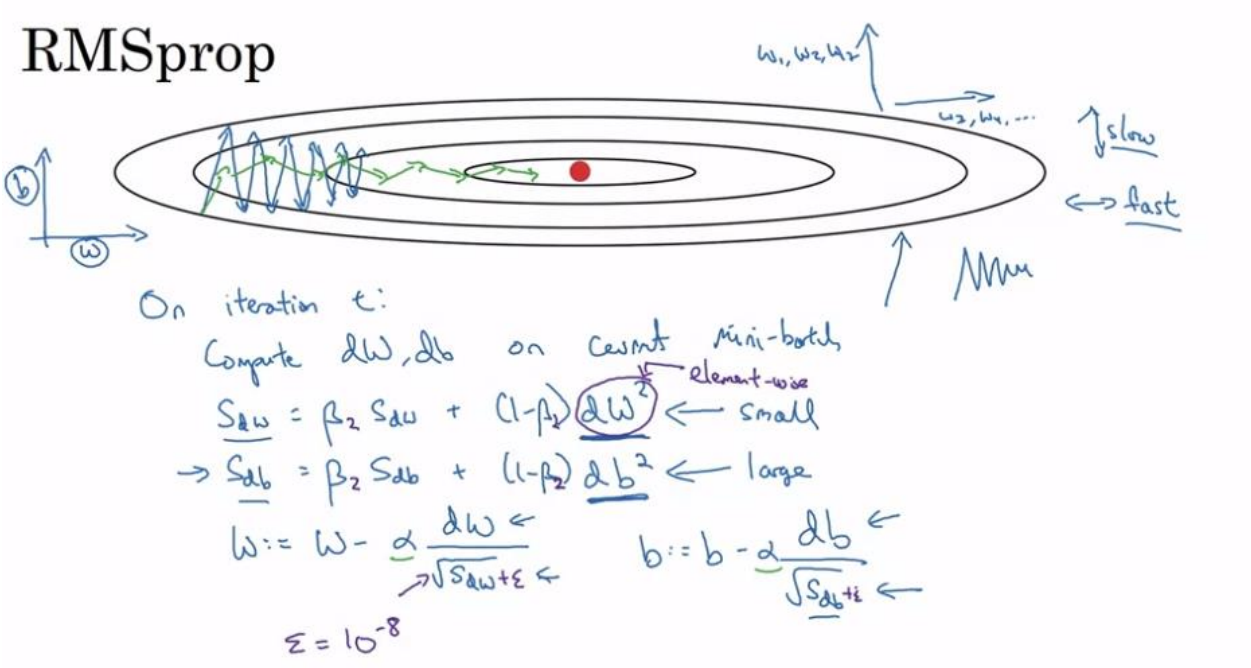
b. RmsProp:

RMSprop is a gradient based optimization technique used in training neural networks. Gradients of very complex functions like neural networks have a tendency to either vanish or explode as the data propagates through the function (*refer to vanishing gradients problem). Rmsprop was developed as a stochastic technique for mini-batch learning.

RMSprop deals with the above issue by using a moving **average of squared gradients to normalize the gradient**. This normalization balances the step size (momentum), **decreasing the step for large gradients to avoid exploding, and increasing the step for small gradients to avoid vanishing.**

Machine Learning theoretical concepts

By: Vikram Pal

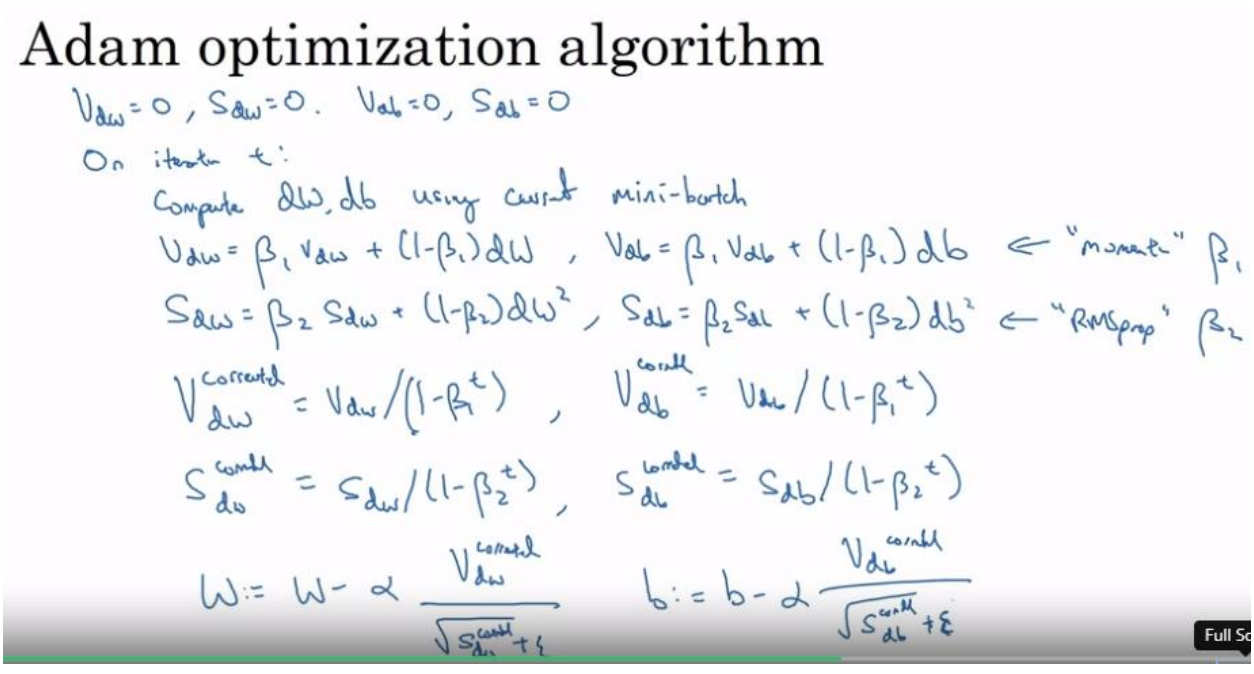


c. Adam:

It can be considered as combination of RMSProp and Stochastic Gradient Descent.

→It uses **Squared Gradient to scale the learning rate** like RMSprop.

→It takes advantage of **momentum by using moving average of the gradient** instead of gradient itself.



Note:

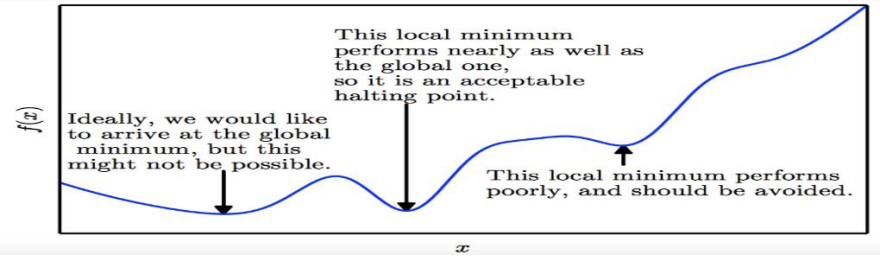
During the batch gradient descent, we look at the error of all the training examples at once while in the SGD we look at each error at a time.

Challenges with Optimization:

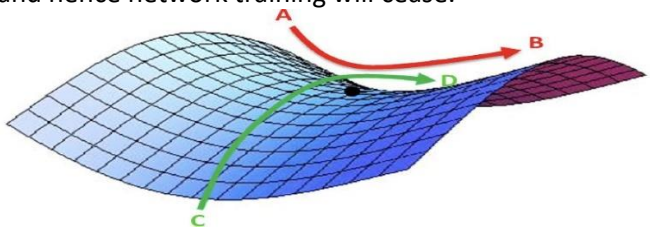
Non-convex optimization involves a function which has multiple optima, only one of which is the global optima. Depending on the loss surface, it can be exceedingly difficult to locate the global optima.
There are certain problems associated with this:

1. What is reasonable learning rate to use?
2. How do we avoid getting stuck in local optima?
3. What if the loss surface morphology changes?

Local Optima: Previously, local minima were viewed as a major problem in neural network training. Nowadays, researchers have found that when using sufficiently large neural networks, most local minima incur a low cost, and thus it is not particularly important to find the true global minimum — **a local minimum with reasonably low error is acceptable**.



Saddle Points: Recent studies indicate that in high dimensions, saddle points are more likely than local minima. Saddle points are also more problematic than local minima because close to a saddle point the gradient can be exceedingly small. Thus, gradient descent will result in negligible updates to the network and hence network training will cease.



Parameter Initialization:

What should be the scale of this initialization? If we choose large values for the weights, this can lead to exploding gradients. On the other hand, small values for weights can lead to vanishing gradients.

Xavier Initialization: We need to initialize the weights in such a way that the **variance remains the same for both the input** and the output**.**

$$W_{ij} \sim N\left(0, \frac{1}{m}\right)$$

The value m is sometimes called the fan-in: the number of incoming neurons (input units in the weight tensor).

He Normal Initialization:

Machine Learning theoretical concepts

By: Vikram Pal

He normal initialization is essentially the same as Xavier initialization, except that the **variance is multiplied by a factor of two**.

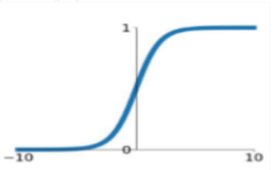
In this method, the **weights are initialized keeping in mind the size of the previous layer** which helps in attaining **a global minimum of the cost function** faster and more efficiently. The weights are still random but differ in range depending on the size of the previous layer of neurons. This provides a controlled initialization hence the faster and more efficient gradient descent.

$$W_{ij} \sim N\left(0, \frac{2}{m}\right)$$

Nonlinear Activation function

Sigmoid or logistic Activation Function:

Function	Equation	Range	Derivative
Sigmoid (Logistic)	$f(x) = \frac{1}{1+e^{-x}}$	0,1	$f'(x) = f(x)(1-f(x))$

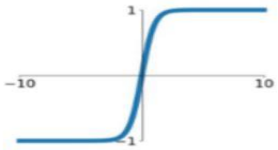


Cons:

- **Derivative of sigmoid function suffers "Vanishing gradient and Exploding gradient problem".**
- Sigmoid function in **not "zero-centric".** This makes the gradient updates go too far in different directions. 0 < output < 1, and it makes optimization harder.
- Slow convergence- as its computationally heavy.

Tanh Activation Function:

Function	Equation	Range	Derivative
Tanh (Hyperbolic tangent)	$f(x) = \frac{2}{1+e^{-2x}} - 1$	-1,1	$f'(x) = 1 - f(x)^2$

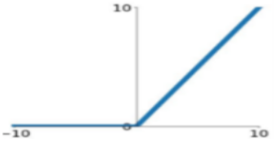


Cons:

- **Derivative of Tanh function suffers "Vanishing gradient and Exploding gradient problem".**
- Slow convergence- as its computationally heavy.

ReLu Activation Function (ReLU-Rectified Linear units):

Function	Equation	Range	Derivative
ReLu (Rectified Linear Unit)	$f(x) = \begin{cases} 0 \ for \ x < 0 \\ x \ for \ x \geq 0 \end{cases}$	$0, +\infty$	$f'(x) = \begin{cases} 0 \ for \ x < 0 \\ 1 \ for \ x \geq 0 \end{cases}$



Problem of Dying neuron/Dead neuron : As the ReLu derivative f'(x) is not 0 for the positive values of the neuron (f'(x)=1 for x ≥ 0),

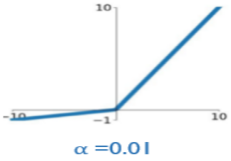
Machine Learning theoretical concepts

By: Vikram Pal

ReLu does not saturate (exploid) and **no dead neurons** (Vanishing neuron)are reported. Saturation and vanishing gradient only occur for negative values that, given to ReLu, are turned into 0- This is called the problem of **dying neuron**."

Leaky ReLu Activation Function:

Function	Equation	Range	Derivative
Leaky ReLu	$f(x) = \begin{cases} \alpha x & for\ x < 0 \\ x & for\ x \geq 0 \end{cases}$	$-\infty, +\infty$	$f'(x) = \begin{cases} \alpha & for\ x < 0 \\ 1 & for\ x \geq 0 \end{cases}$



α =0.01

Leaky ReLU is defined to address problem of dying neuron/dead neuron.

Computer Vision:

Evaluation Matrics in computer Vision:

Peak signal-to-noise ratio (PSNR):

it is the ratio between the maximum possible power of an image and the power of corrupting noise that affects the quality of its representation. To estimate the PSNR of an image, it is necessary to compare that image to an ideal clean image with the maximum possible power.

PSNR is defined as follows:

$$PSNR = 10log_{10}(\frac{(L-1)^2}{MSE}) = 20log_{10}(\frac{L-1}{RMSE})$$

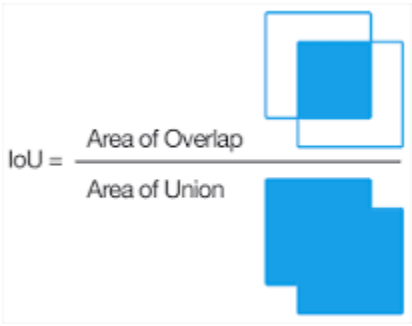
Structural Similarity Index metric:

SSIM is used as a metric to measure the similarity between two given images.
The Structural Similarity Index (SSIM) metric extracts 3 key features from an image:

- Luminance (averaging over all the pixel values)
- Contrast (standard deviation (square root of variance) of all the pixel values.)
- Structure

It varies b/w -1 and +1. +1 means two images are similar and -1 means two images are totally different.

IOU (intersection over union):



Machine Learning theoretical concepts

By: Vikram Pal

Intersection over Union is an evaluation metric used to measure the accuracy of an object detector on a particular dataset. Any algorithm that provides predicted bounding boxes as output can be evaluated using IoU.

Inception Score:

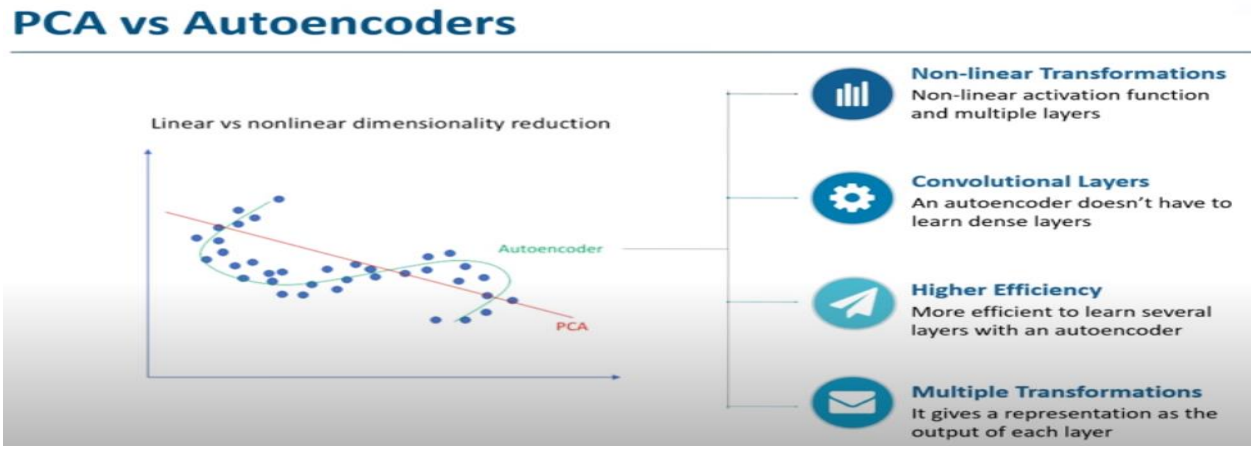
The Inception Score, or IS for short, is an objective metric for evaluating the quality of generated images, specifically synthetic images output by generative adversarial network models. They developed the inception score as an attempt to remove the subjective human evaluation of images.

Frechet Inception Distance:

FID is a **metric for evaluating the quality of generated images** and specifically developed to evaluate the performance of generative adversarial networks

Autoencoder:

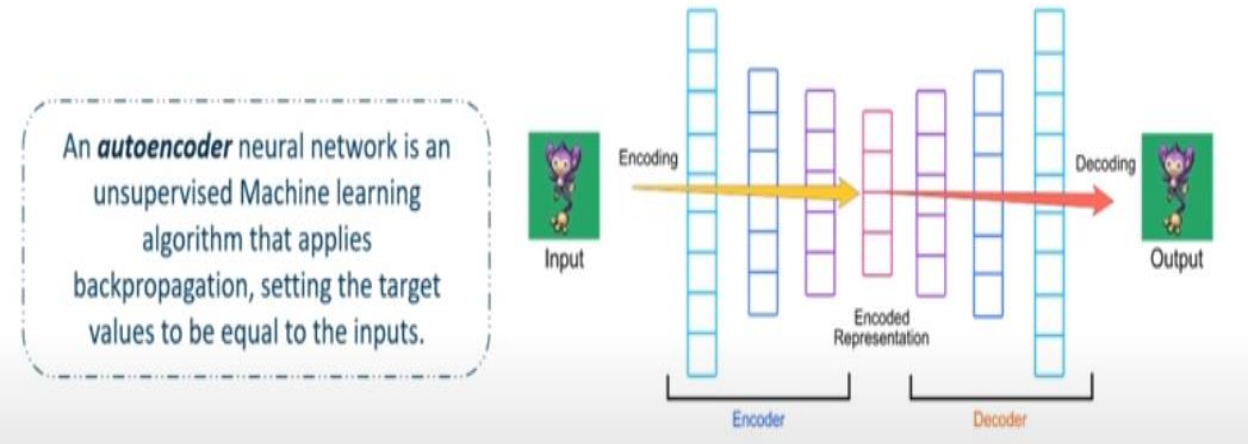
https://www.youtube.com/watch?v=nTt_ajul8NY
https://blog.keras.io/building-autoencoders-in-keras.html

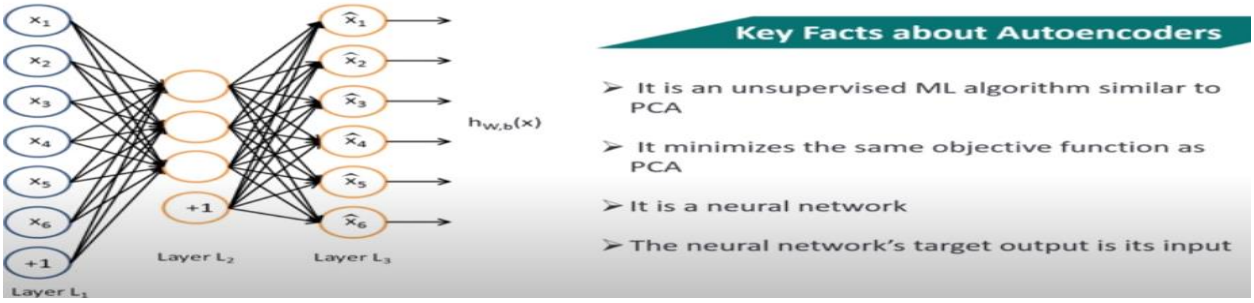


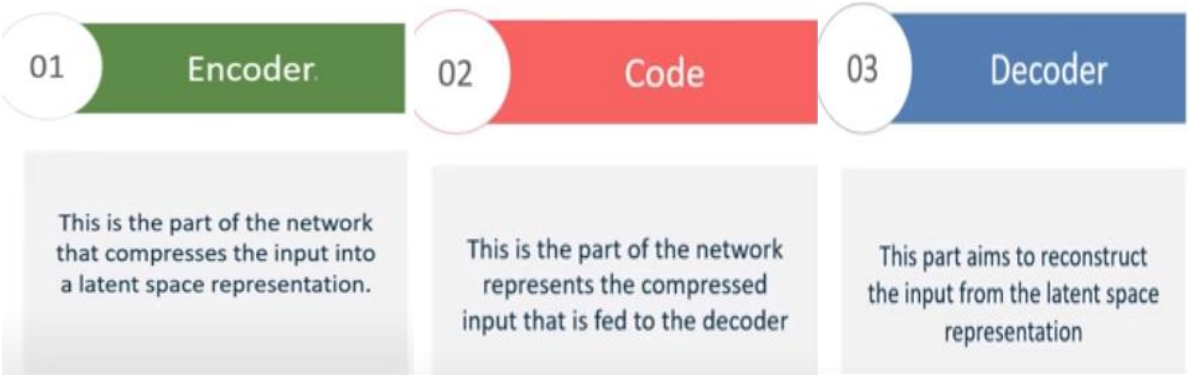
The biggest advantage of Autoencoder is that we can apply fine tuning. That mean we can use already trained learning from some place to new model. By that means we don't need to train model again and again. Two main applications of autoencoder are: 1. Denoising 2. Dimensionality Reduction.

Machine Learning theoretical concepts

By: Vikram Pal

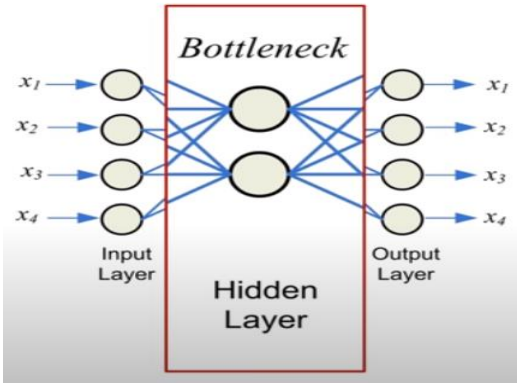






Machine Learning theoretical concepts

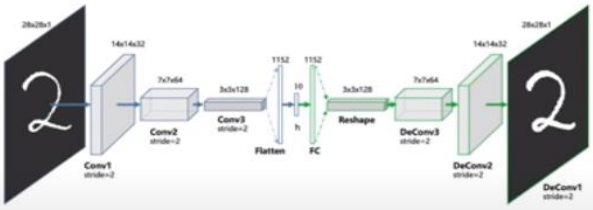
By: Vikram Pal



Bottleneck approach is an approach to for deciding which aspects of observed data are relevant information and what aspects can be thrown away

- Compactness of representation, measured as the compressibility
- Representation retains about some behaviourally relevant variables

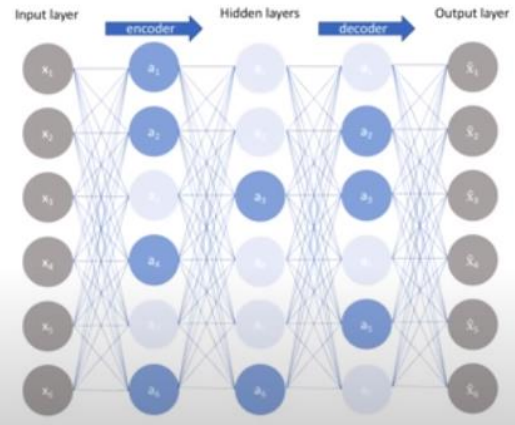
Convolution Autoencoders



Convolution Autoencoders use the convolution operator to learn to encode the input in a set of simple signals and then try to reconstruct the input from them.

$$f(t) * g(t) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau$$

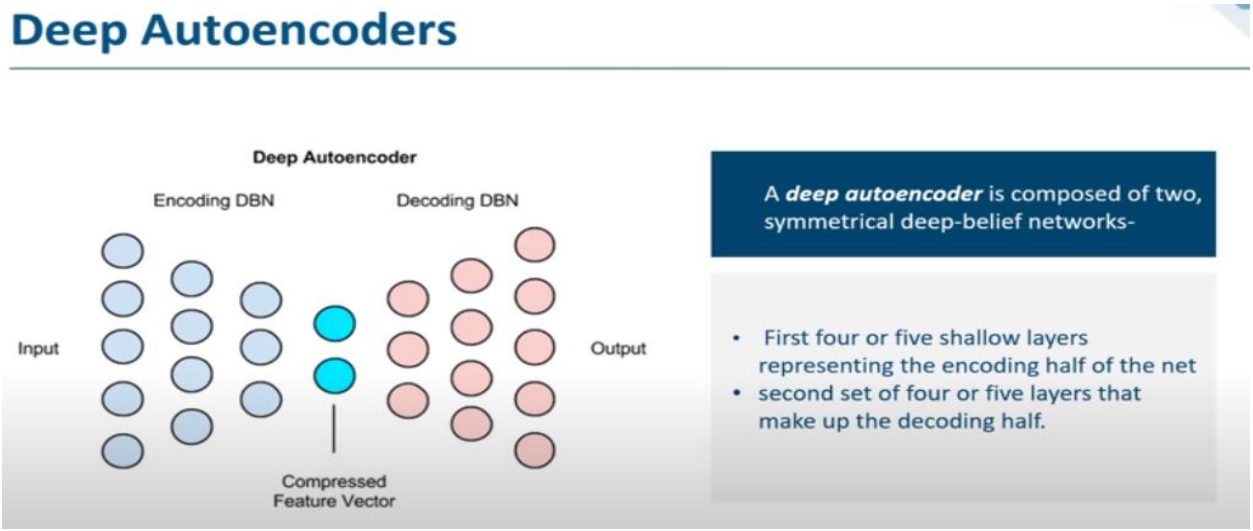
Sparse Autoencoders

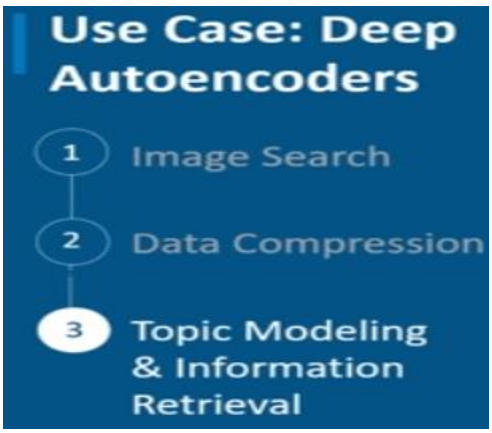


Sparse autoencoders offer us an alternative method for introducing an information bottleneck without requiring a reduction in the number of nodes at our hidden layers

Machine Learning theoretical concepts

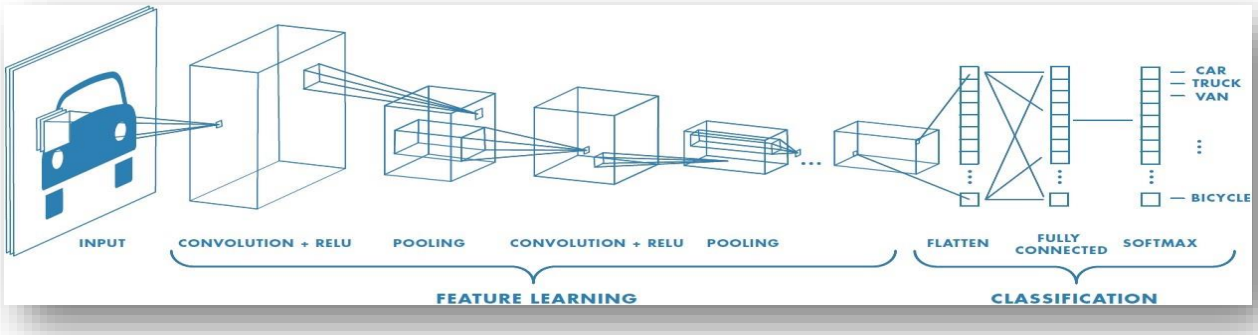
By: Vikram Pal





Convolution Neural Network:

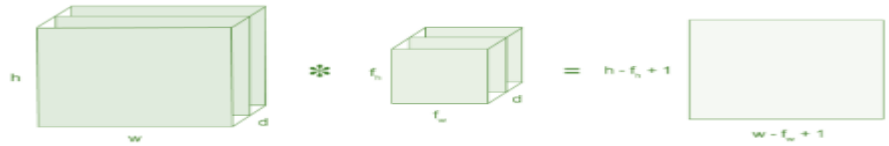
CNNs are fully connected feed forward neural networks. CNNs are very effective in reducing the number of parameters without losing on the quality of models. Images have high dimensionality



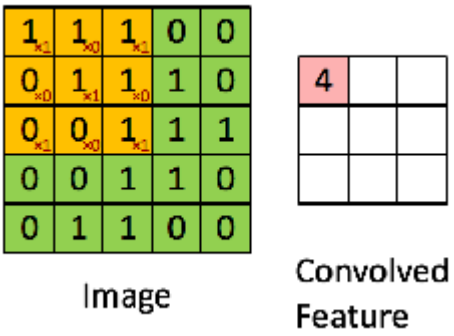
Machine Learning theoretical concepts

By: Vikram Pal

- An image matrix (volume) of dimension **(h x w x d)**
- A filter **(f_h x f_w x d)**
- Outputs a volume dimension **(h - f_h + 1) x (w - f_w + 1) x 1**



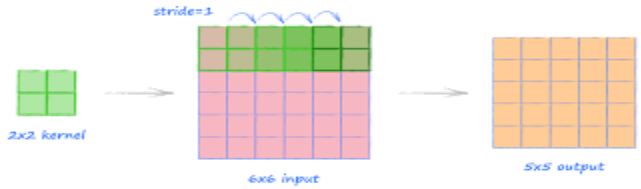
Then the convolution of 5 x 5 image matrix multiplies with 3 x 3 filter matrix which is called “Feature Map” as output shown in below



Convolution of an image with different filters can perform operations such as edge detection, blur and sharpen by applying filters.

Strides:

Stride is the number of pixels shifts over the input matrix.



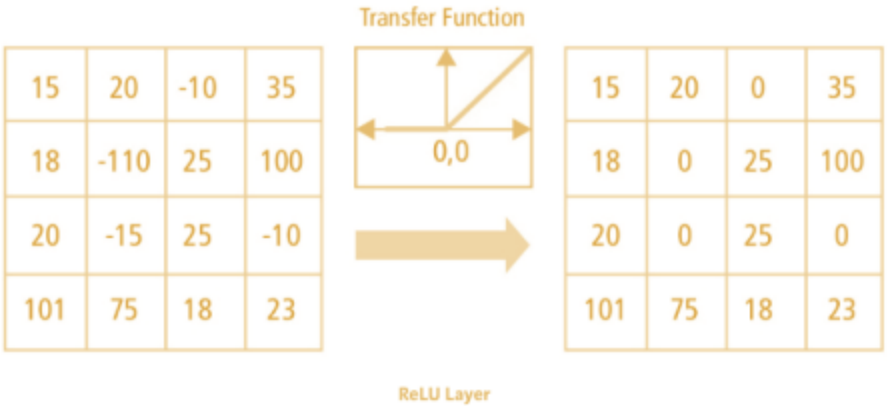
Non-Linearity (ReLU):

ReLU stands for Rectified Linear Unit for a non-linear operation. The output is ƒ(x) = max(0,x).

Why ReLU is important : ReLU’s purpose is to introduce non-linearity in our ConvNet. Since, the real-world data would want our ConvNet to learn would be non-negative linear values.

Machine Learning theoretical concepts

By: Vikram Pal

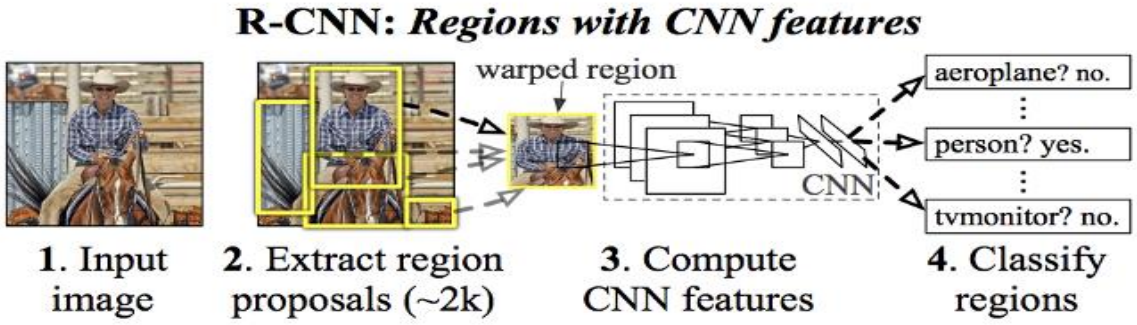


Pooling Layer: It reduce the number of parameter when the image is too large.
a. Max Pooling
b. Average Pooling
c. Sum Pooling

Fully Connected Layer:
The layer we call as FC layer, we flattened our matrix into vector and feed it into a fully connected layer like a neural network.

R-CNN:

To bypass the problem of selecting a huge number of regions , we use selective search to extract just 2000 regions from the image, those are region proposals. We generate using the **selective search algorithms**
Selective Search:
1. Generate initial sub-segmentation, we generate many candidate regions
2. Use greedy algorithm to recursively combine similar regions into larger ones
3. Use the generated regions to produce the final candidate region proposals



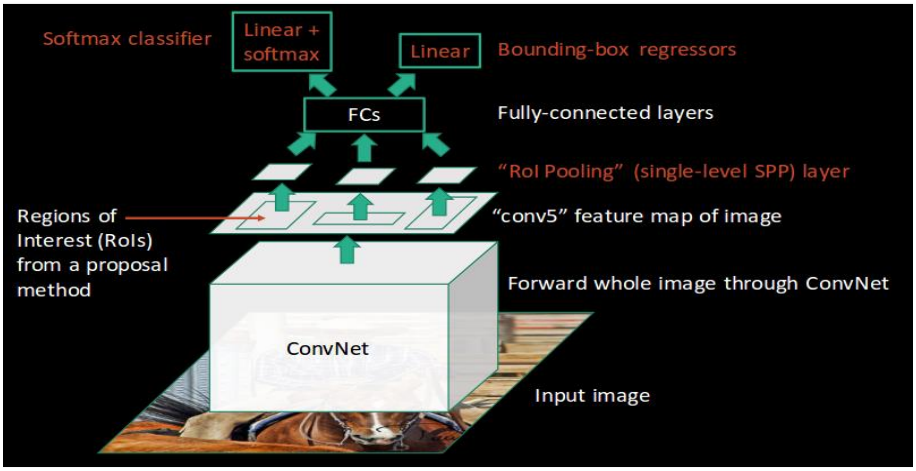
The CNN acts as a feature extractor and the output dense layer consists of the features extracted from the image and the extracted features are fed into an **SVM to classify the presence of the object** within that candidate region proposal

Fast R-CNN:

It is similar to R-CNN. But instead of generating regions, we use CNN to generate a convolutional feature map.

Machine Learning theoretical concepts

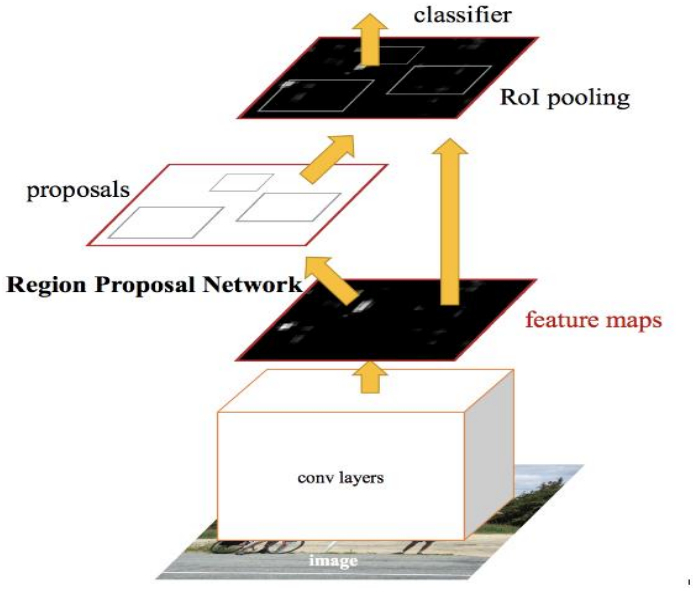
By: Vikram Pal



From the convolutional feature map, we identify the region of proposals and warp them into squares and by using a RoI(Region of interest) pooling layer we reshape them into a fixed size so that it can be fed into a fully connected layer.

From the ROI feature vector, we use **a Softmax layer to predict** the class of the proposed region and also the offset values for the bounding box.

Faster R-CNN:



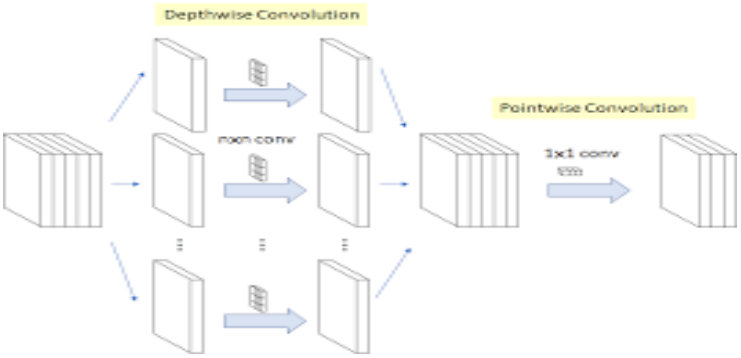
Instead of using selective search algorithm on the feature map to identify the region proposals, a **separate network is used to predict the region proposals**. The predicted region proposals are then reshaped using a RoI pooling layer which is then used to classify the image within the proposed region and predict the offset values for the bounding boxes.

All of the previous object detection algorithms use regions to localize the object within the image. The network does not look at the complete image. Instead, parts of the image which have high probabilities of containing the object.

Machine Learning theoretical concepts

By: Vikram Pal

MobileNet:



Depth-wise convolution

In this convolution, we apply a **2-d depth filter at each depth level of input tensor**. Let's understand this through an example. Suppose our input tensor is 3* 8 *8 (input_channels*width* height). Filter is 3*3*3. In a standard convolution we would directly convolve in depth dimension as well (fig 1).

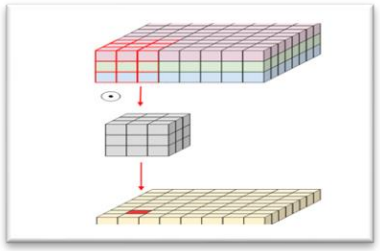


Fig 1. Normal convolution

In depth-wise convolution, we use each filter channel only at one input channel. In the example, we have 3 channel filter and 3 channel images. What we do is — break the filter and image into three different channels and then convolve the corresponding image with corresponding channel and then stack them back (Fig 2)

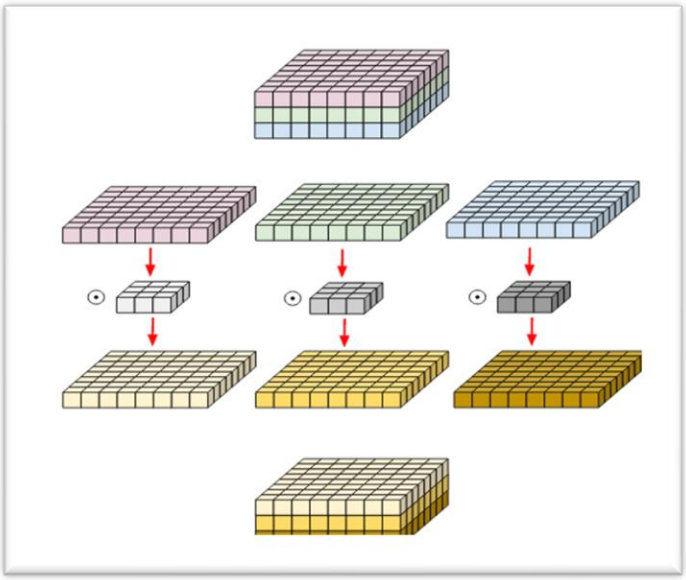
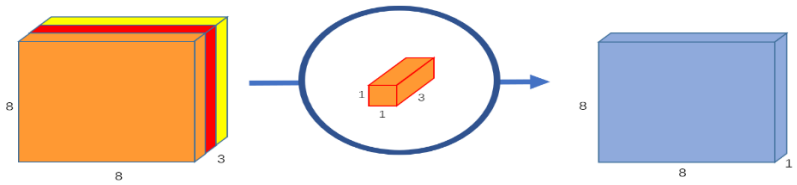


Fig 2. Depth-wise convolution.

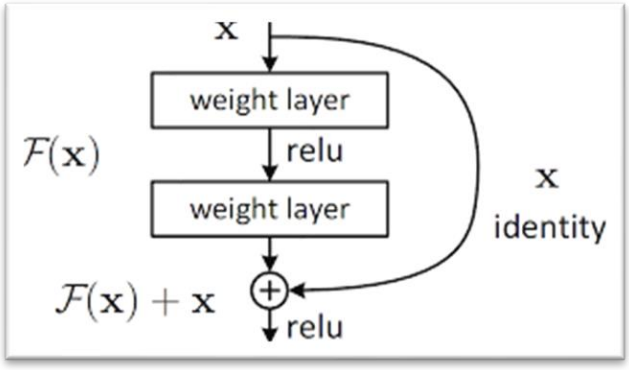
Filters and image have been broken into three different channels and then convolved separately and stacked thereafter

Pointwise Convolution:

Pointwise Convolution is a type of convolution that uses a 1x1 kernel: a kernel that iterates through every single point. This kernel has a depth of however many channels the input image has. It can be used in conjunction with depthwise convolutions to produce an efficient class of convolutions known as depthwise-separable convolutions.



Residual Network:



YOLO:

In YOLO a single convolutional network predicts the **bounding boxes and the class probabilities** for these boxes

Machine Learning theoretical concepts

By: Vikram Pal

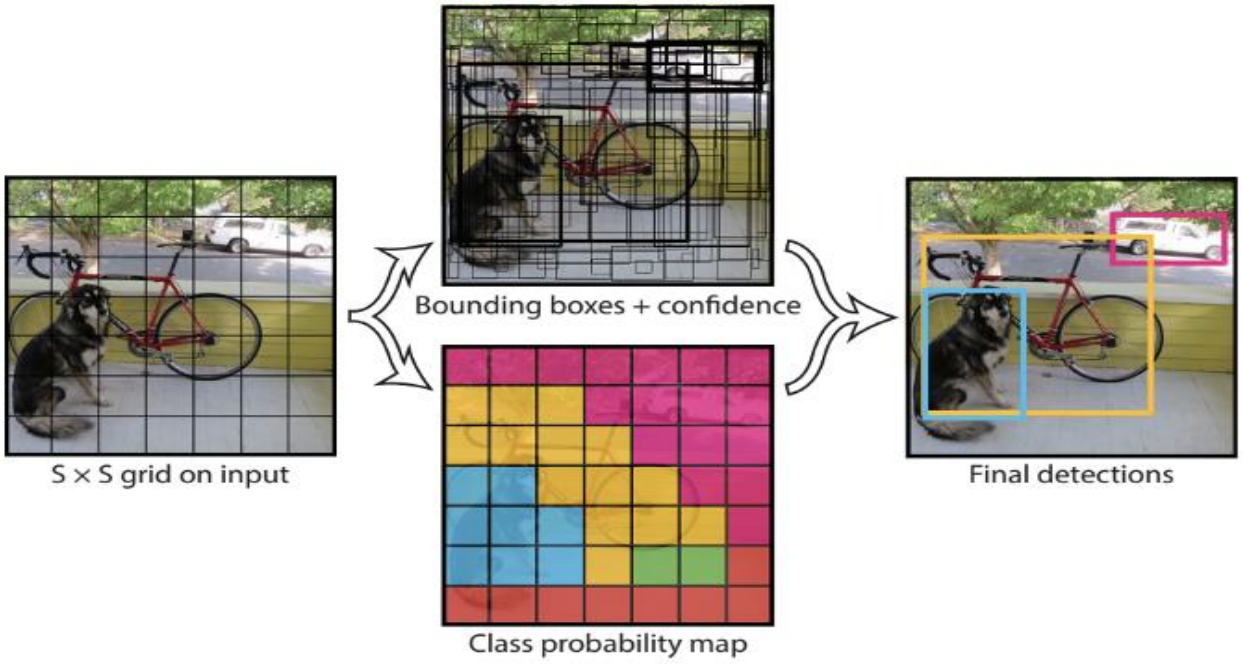
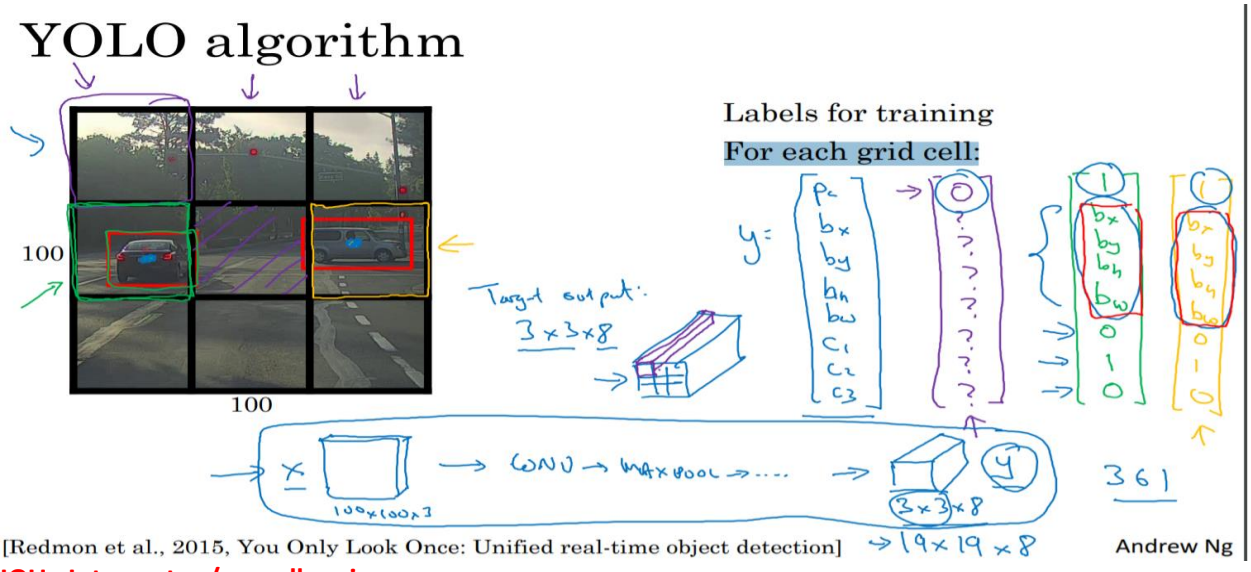


Image split (S*S) grid → each grid m bounding boxes → Each bounding box (network o/p class probability and offset values) → bounding box having probability above a threshold value is selected

How YOLO works is that we take an i**mage and split it into an SxS grid,** within each of the grid we take m bounding boxes. For each of the bounding box, the network outputs a class probability and **offset values for the bounding box**. **The bounding boxes having the class probability above a threshold value is selected and used to locate the object within the image**.

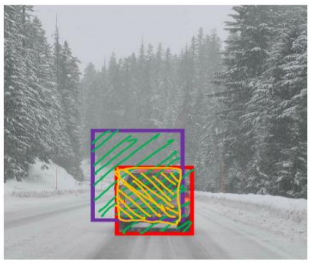


IOU= Intersecton/overall region

Machine Learning theoretical concepts

By: Vikram Pal

Evaluating object localization



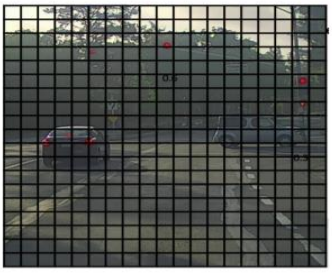
Intersection over Union (IoU)

$= \frac{\text{size of}}{\text{size of}}$

"Correct" if IoU $\geq$ 0.5 ←

0.6 ←

Non-max suppression algorithm



19×19

Each output prediction is: p_c, b_x, b_y, b_h, b_w, c_1, c_2, c_3

Discard all boxes with $p_c \leq 0.6$

While there are any remaining boxes:

- Pick the box with the largest p_c Output that as a prediction.
- Discard any remaining box with IoU $\geq$ 0.5 with the box output in the previous step

How can I improve my Yolo accuracy?

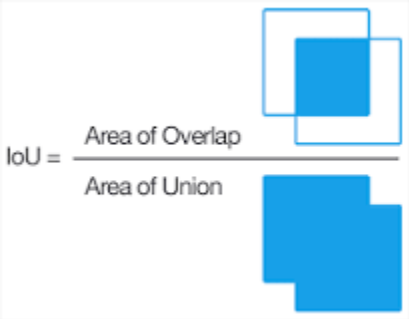
Different Training Heuristics for Object Detection

- Image mix-up with geometry preserved alignment.
- Using cosine learning rate scheduler.
- Synchronized batch normalization.
- Data augmentation.
- Label smoothing.

Machine Learning theoretical concepts

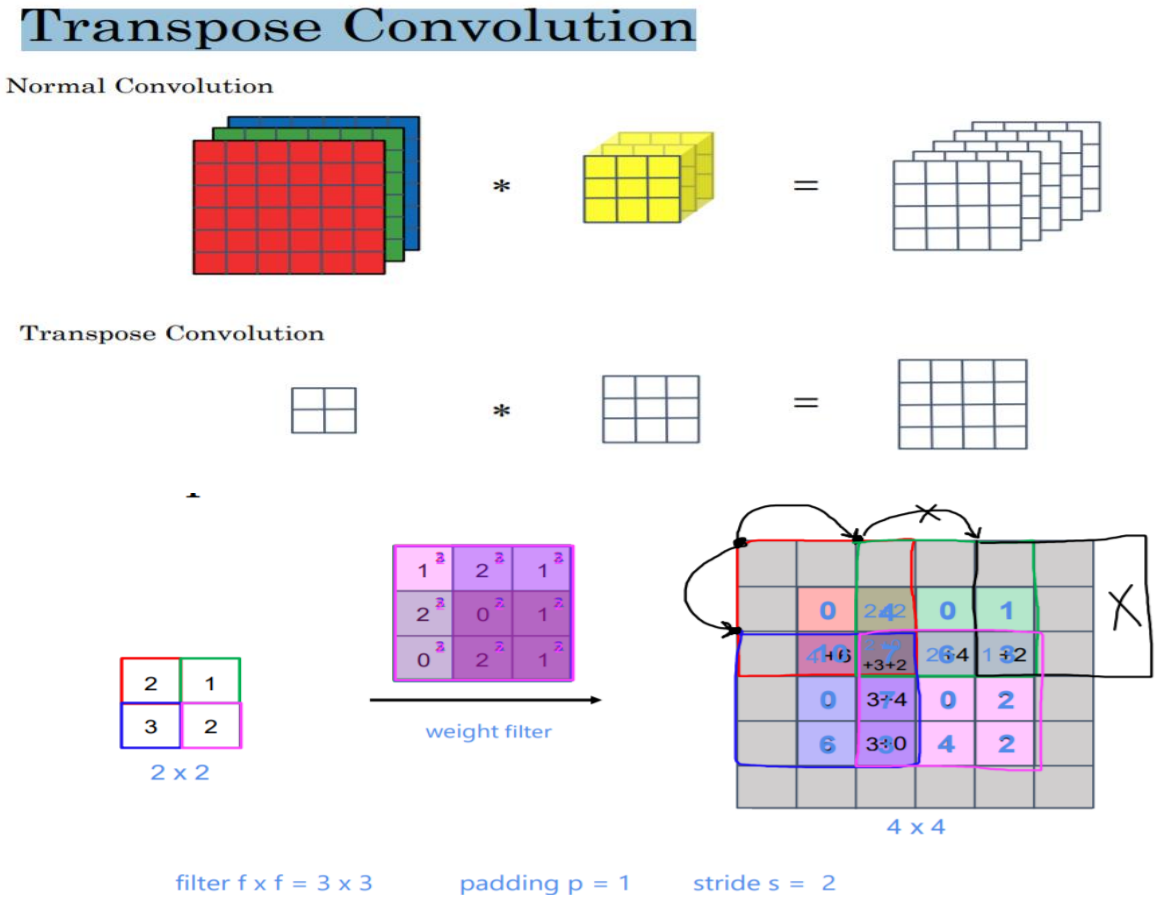
By: Vikram Pal

What is IoU in Yolo?



Intersection over Union is an **evaluation metric** used to measure the accuracy of an object detector on a particular dataset. Any algorithm that provides predicted bounding boxes as output can be evaluated using IoU.

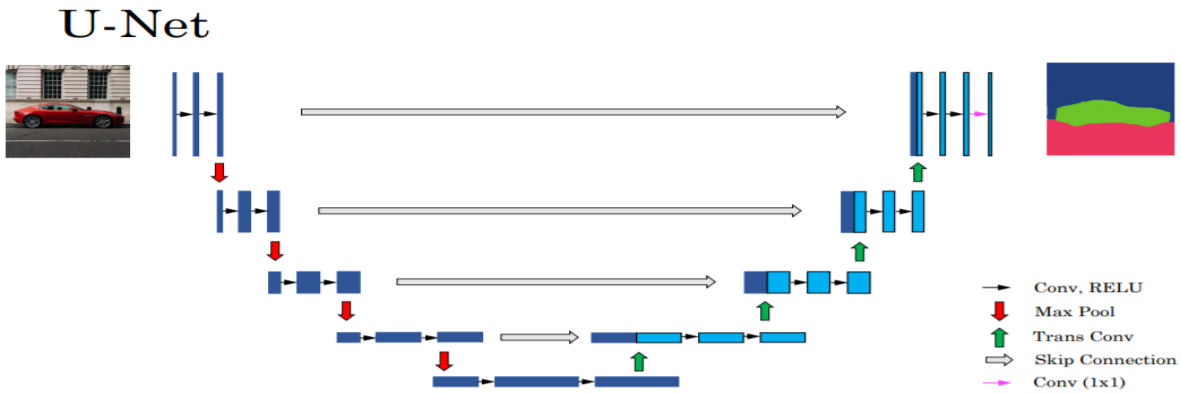
Transpose Convolution:



Machine Learning theoretical concepts

By: Vikram Pal

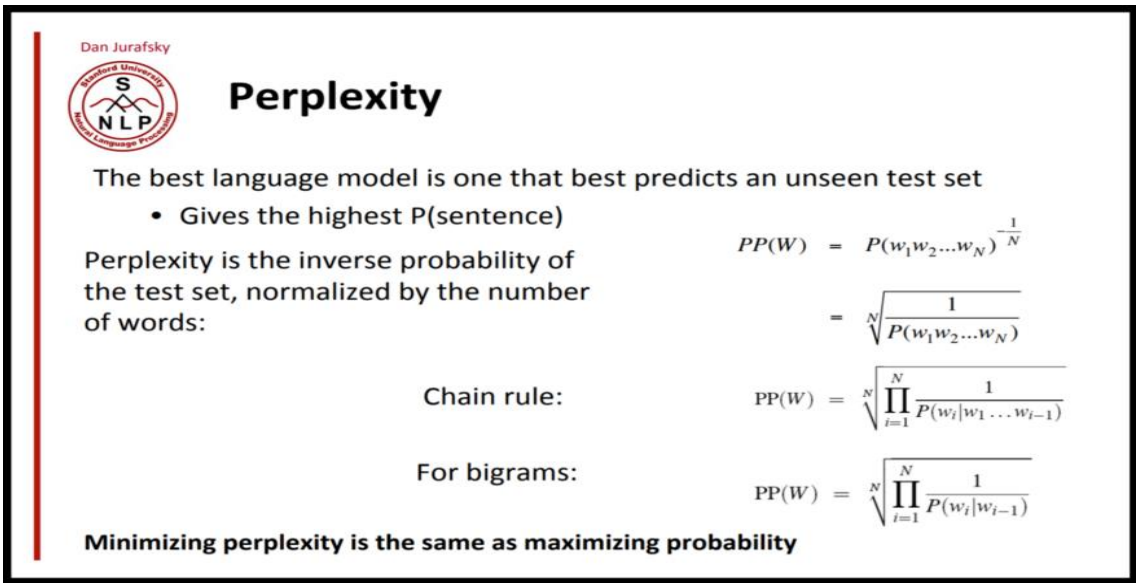
U-Net:



NLP basic

NLP model evaluation Matrics:

Preplexity:



BLEU (Bi-Lingual Evaluation Understudy):

BLEU compares the machine's translation — what's known as the candidate translation — with existing human-generated translations, known as the reference translations.

TFIDF:

TF= No of repeated words in sentence/No of words in Sentence

IDF= log(No of Sentence/No of Sentences containing words)

Term Frequency: The number of times a **term occurs in a document** is called its *term frequency*.

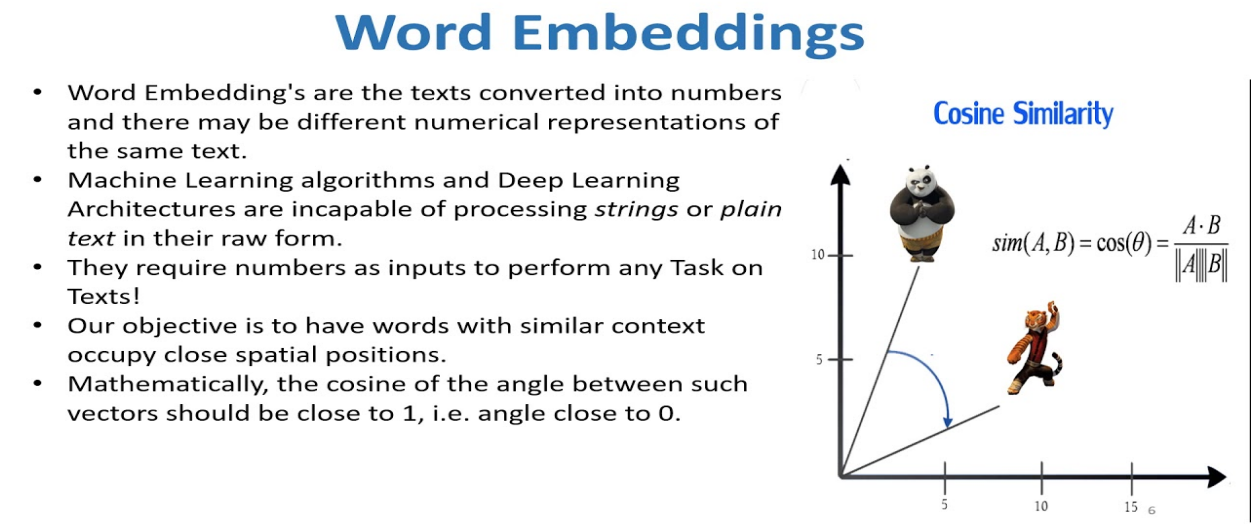
Machine Learning theoretical concepts

By: Vikram Pal

Document Frequency: The only difference is that TF is frequency counter for a term t in document d, where as DF is the **count of occurrences of term** t in the document set N.

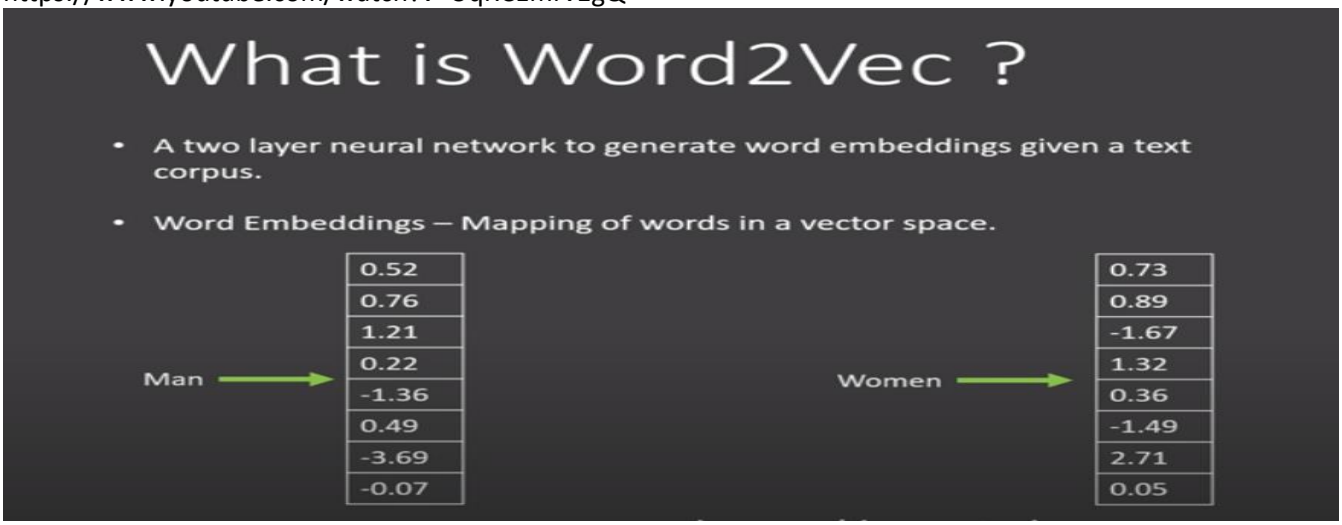
Inverse Document Frequency (IDF):
While computing TF, all terms are considered equally important. However, it is known that certain terms, such as "is", "of", and "that", may appear a lot of times but have little importance. Thus, we need to weigh down the frequent terms while scale up the rare ones, by computing IDF, an inverse document frequency factor is incorporated which diminishes the weight of terms that occur very frequently in the document set and increases the weight of terms that occur rarely.

Word Embedding:



Word2Vec:

https://www.youtube.com/watch?v=UqRCEmrv1gQ



Machine Learning theoretical concepts

By: Vikram Pal

- Preserves relationship between words.
- Deals with addition of new words in the vocabulary.
- Better results in lots of deep learning applications.

Working of word2Vec

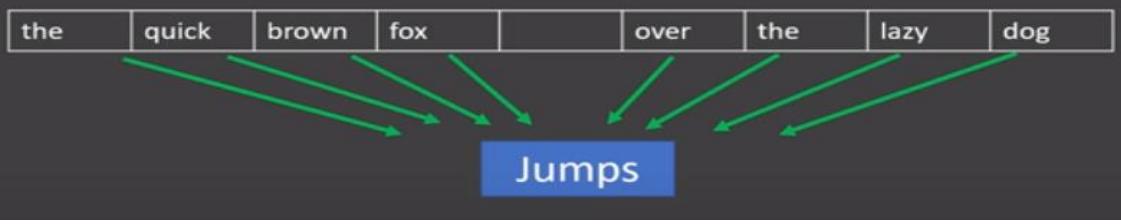
- The word2vec objective function causes the words that occur in similar contexts to have similar embeddings.

Example: The **kid** said he would grow up to be superman.
The **child** said he would grow up to be superman.

The words kid and child will have similar word vectors due to a similar context.

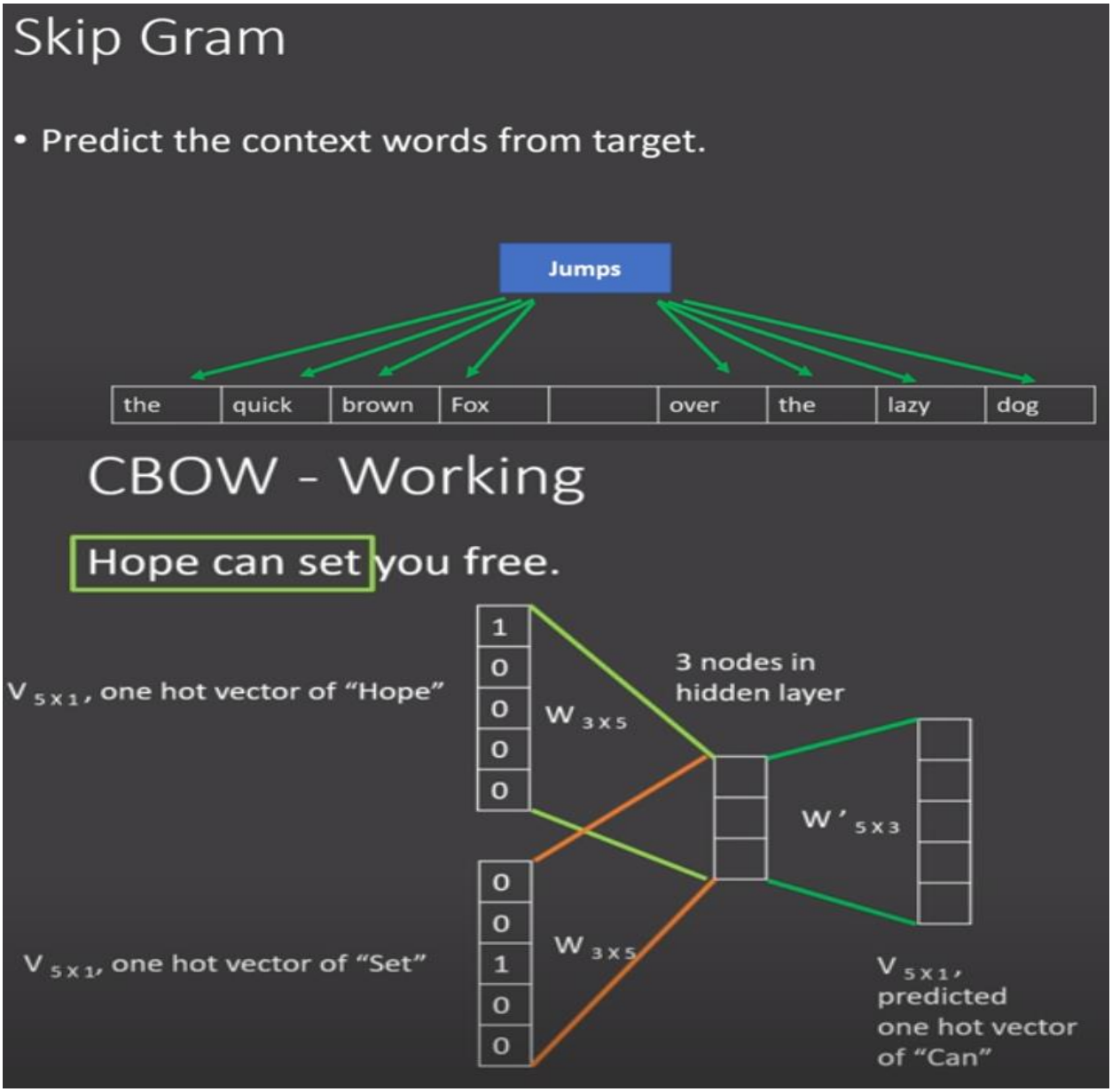
CBOW

- Predict the target word from the context.



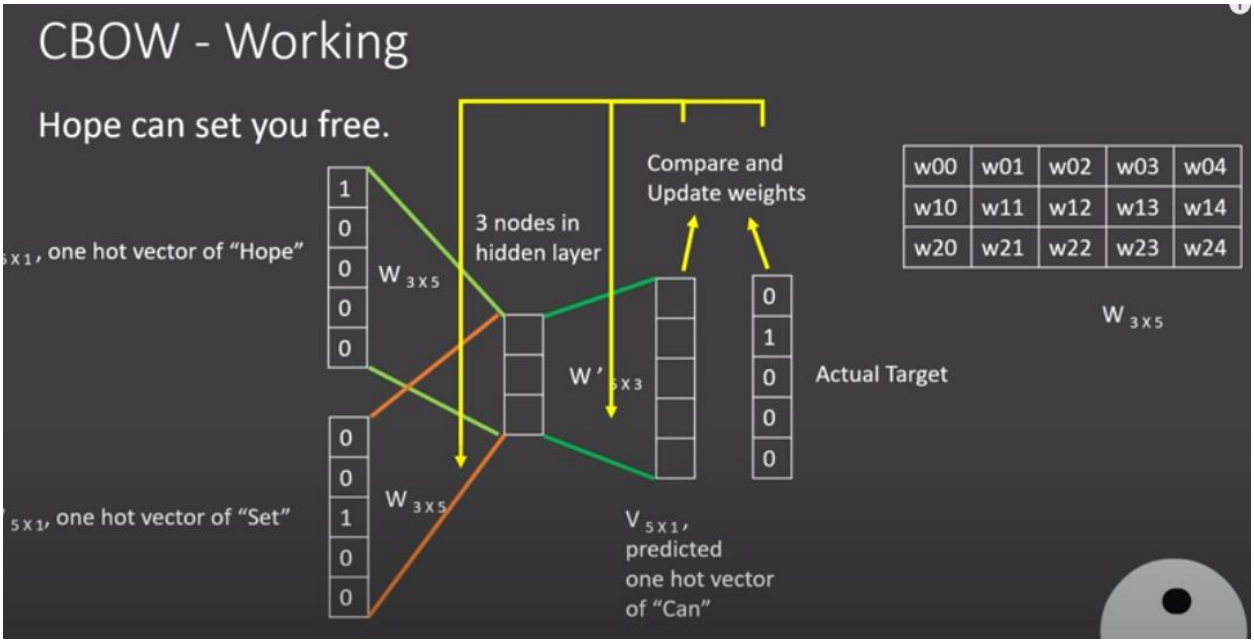
Machine Learning theoretical concepts

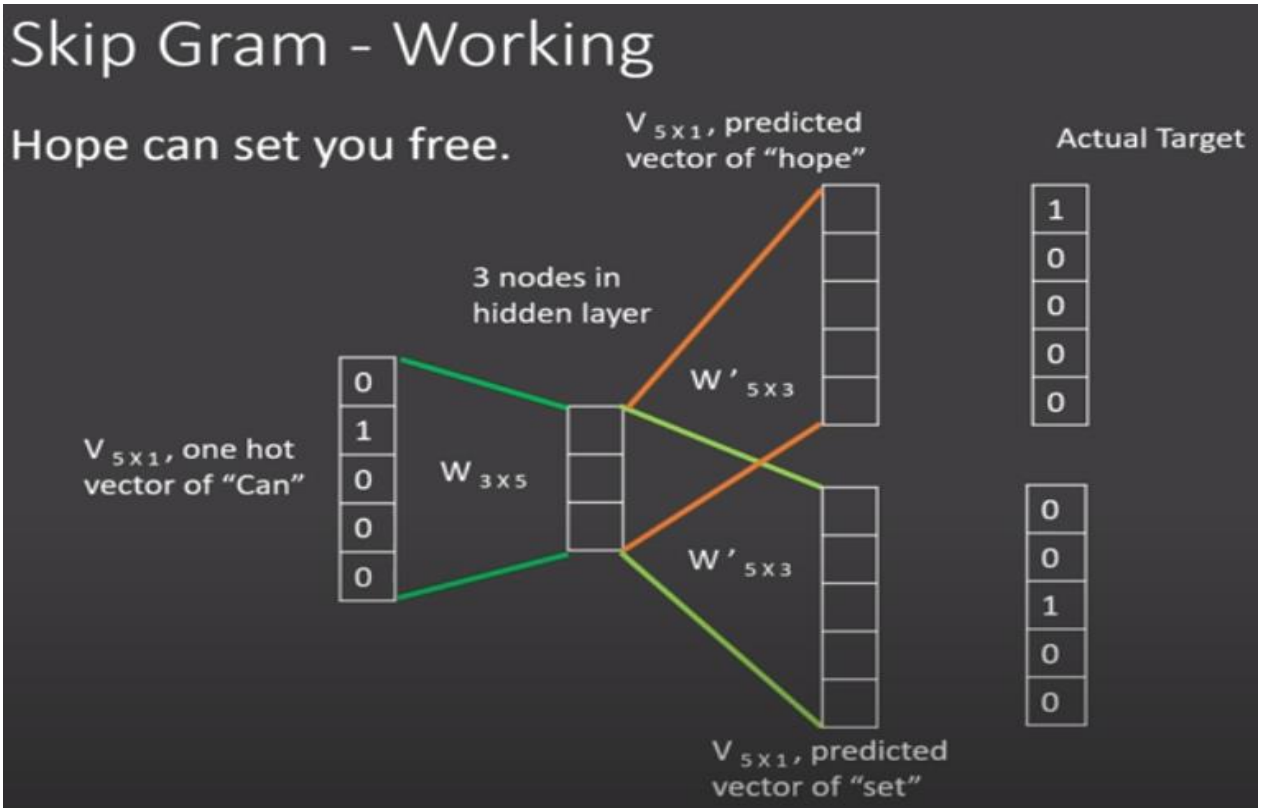
By: Vikram Pal



Machine Learning theoretical concepts

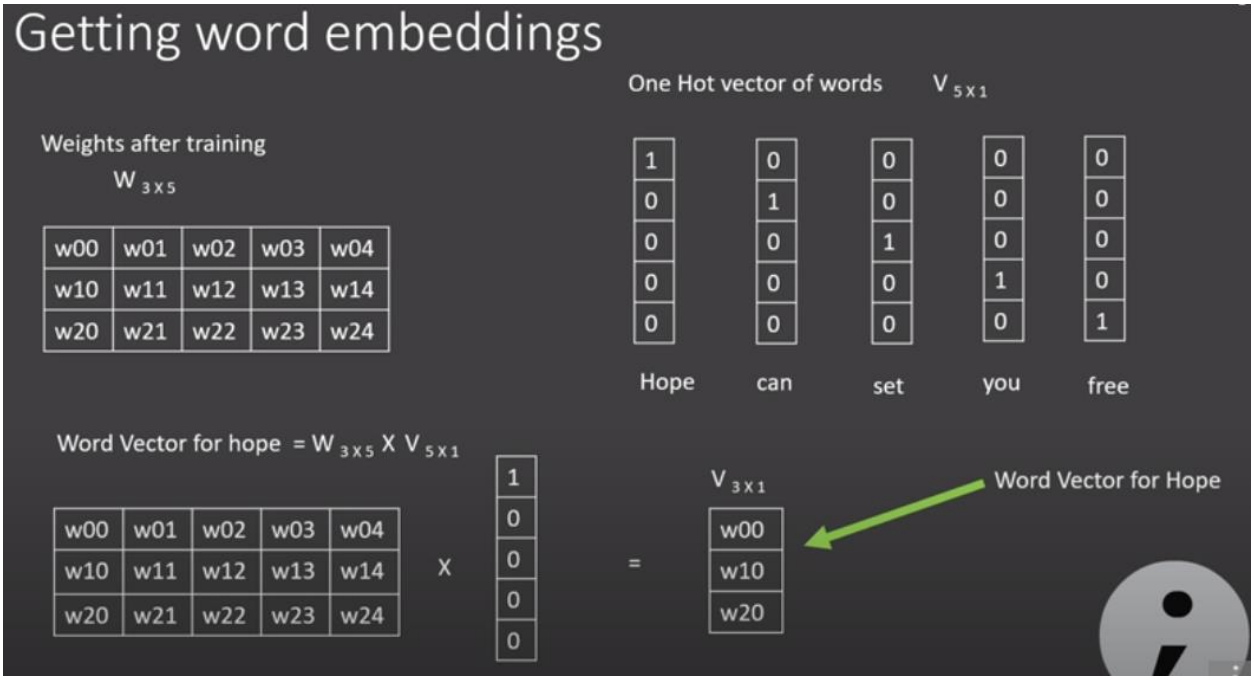
By: Vikram Pal

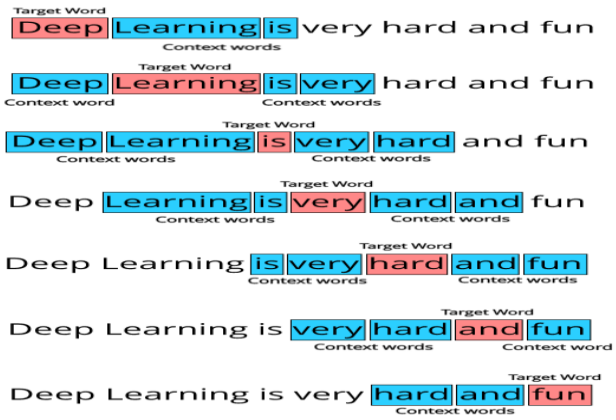




Machine Learning theoretical concepts

By: Vikram Pal





CBOW: the only difference is that we try to predict the target word given the context words.

Glove

Glove is based on **matrix factorization technique** on word context matrix. It first constructs a large matrix of (words x context) **co-occurrence information** ie. for each word, you count how frequently we see those word in some context in a large corpus.

In order to understand how GloVe works, we need to understand 2 main methods which GloVe was built on -

Global matrix factorization:

In NLP, global matrix factorization is the process of using **matrix factorization form linear algebra to reduce large term frequency matrices**. These **matrices usually represent the occurrences or the absence of words in the document.**

Local context window:

Machine Learning theoretical concepts

By: Vikram Pal

Local context window methods are CBOW and Skip-Gram

Topic modeling:

Note*: Word→document→ Corpus.

Topic modeling: it is a type of statistical modeling for discovering the abstract "topics" that occur in a collection of documents. It builds a topic per document model and words per topic model, modeled as Dirichlet distributions.

LSA: Latent Semantic Analysis: raw count in (Documents, terms) replace with tf-idf score ➔ (documents, topic) * (topic*term)
Decomposing Documents and terms matrix into a separate document-topic matrix and a topic-term matrix.
LSA models typically replace raw counts in the document-term matrix with a tf-idf score.
This dimensionality reduction can be performed **using** truncated SVD. SVD, or singular value decomposition, is a technique in linear algebra that factorizes any matrix M into the product of 3 separate matrices: M=U*S*V, where S is a diagonal matrix of the singular values of M.

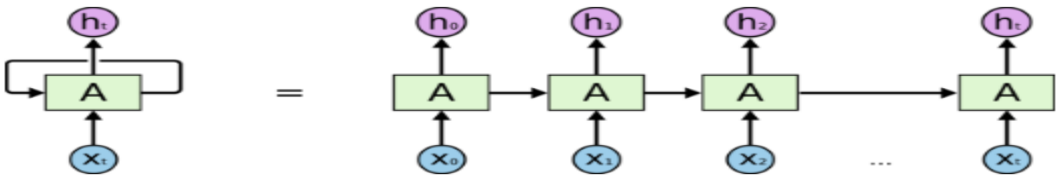
pLSA, or Probabilistic Latent Semantic Analysis: It uses a ***probabilistic method instead of SVD*** to tackle the problem. The core idea is to find a probabilistic model with latent topics that can generate the data we observe in our document-term matrix.

LDA stands for Latent Dirichlet Allocation: LDA is a Bayesian version of pLSA. It uses **Dirichlet priors** for the document-topic and word-topic distributions, **lending itself to better generalization.**

Recurrent Neural Network (RNN):

It is a generalization of **feedforward neural network that has an internal memory**. RNN is recurrent in nature as it performs the **same function for every input of data while the output of the current input depends on the past one computation.**

In other neural networks, **all the inputs are independent of each other. But in RNN, all the inputs are related to each other.**



An unrolled recurrent neural network.

Gradient issues in RNN:
While training an RNN algorithm, sometimes gradient can become too small or too large. So, the training of an RNN algorithm becomes very difficult in this situation. Due to this, following issues occur:

1. Poor Performance
2. Low Accuracy
3. Long Training Period

Machine Learning theoretical concepts

By: Vikram Pal

Exploding Gradient: When we assign high importance to the weights, exploding gradient issue occurs. In this case, values of a gradient become too large, and slope tends to grow exponentially. This can be solved using following methods:

1. **Identity Initialization**
2. **Truncated Back-propagation**
3. **Gradient Clipping:** With gradient clipping, pre-determined gradient threshold be introduced, and then gradients norms that exceed this threshold are scaled down to match the norm. This prevents any gradient to have norm greater than the threshold and thus the gradients are clipped. There is an introduced bias in the resulting values from the gradient, but gradient clipping can keep things stable.

Vanishing Gradient: This issue occurs when the values of a gradient are too small, and the model stops learning or takes way too long because of that. This can be solved using following methods:

1. **Weight Initialization**
2. **Choosing the right Activation Function**
3. **LSTM (Long Short-Term Memory) Best way to solve the vanishing gradient issue is the use of LSTM (Long Short-Term Memory).**

The formula for the current state is

$$h_t = f(h_{t-1}, x_t)$$

Applying Activation Function:

$$h_t = \tanh (W_{hh}h_{t-1} + W_{xh}x_t)$$

W is weight, h is the single hidden vector, Whh is the weight at previous hidden state, Whx is the weight at current input state, tanh is the Activation function, that implements a non-linearity that squashes the activations to the range[-1.1]
Output:

$$y_t = W_{hy}h_t$$

Yt is the output state. Why is the weight at the output state.

LSTM:

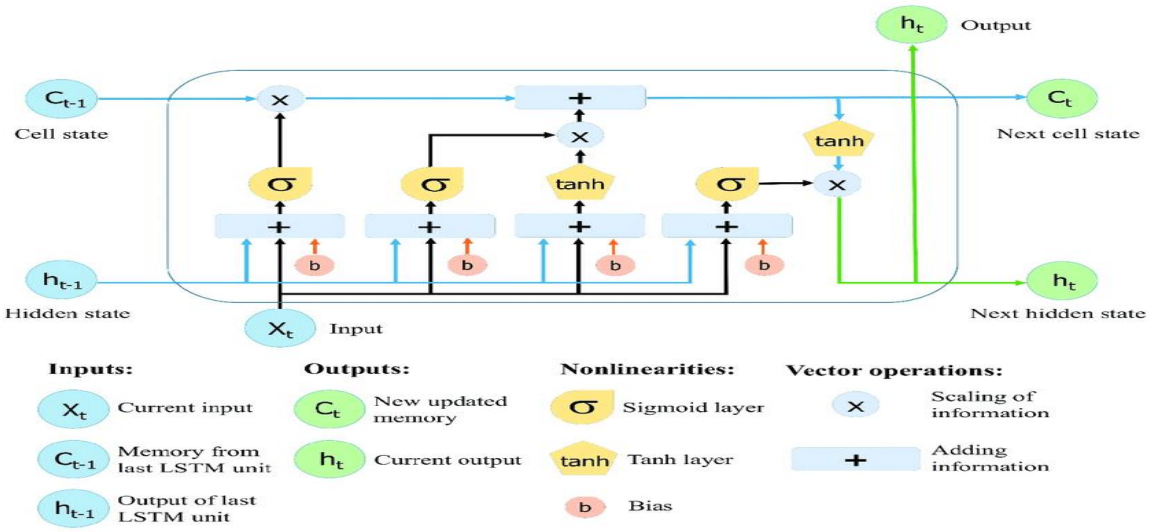
Long Short-Term Memory (LSTM) networks are a modified version of recurrent neural networks, which makes it easier to remember past data in memory.

The vanishing gradient problem of RNN is resolved here. LSTM is well-suited to classify, process and predict time series given time lags of unknown duration.

It trains the model by using back-propagation. In an LSTM network, three gates are present:

Machine Learning theoretical concepts

By: Vikram Pal



Input Gate:

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Forget Gate:

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Output Gate:

$$o_t = \sigma\left(W_o \; [\, h_{t-1}, x_t] \; + \; b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

BPTT(Backpropagation through time):

Backpropagation Through Time, or BPTT, is the application of the **Backpropagation training algorithm to recurrent neural network applied to sequence data like a time series.**

We can summarize the algorithm as follows:

1. Present a sequence of timesteps of input and output pairs to the network.
2. Unroll the network then calculate and accumulate errors across each timestep.
3. Roll-up the network and update weights.

Repeat.

BPTT can be computationally expensive as the number of timesteps increases.

Truncated BPTT:

We can summarize the algorithm as follows:

1. Present a sequence of k1 timesteps of input and output pairs to the network.
2. Unroll the network then calculate and accumulate errors across k2 timesteps.
3. Roll-up the network and update weights.

Repeat

The TBPTT algorithm requires the consideration of two parameters:

1. **k1**: The number of forward-pass timesteps between updates. Generally, this influences how slow or fast training will be, given how often weight updates are performed.

2. **k2**: The number of timesteps to which to apply BPTT. Generally, it should be large enough to capture the **temporal structure in the problem for the network to learn**. Too large a value results in vanishing gradients.

Return Sequence=true:- It makes possible to access the hidden state output for each input time step.
Return State=True:-

1. The LSTM hidden state output for the last time step.
2. The LSTM hidden state output for the last time step (again).
3. The LSTM cell state for the last time step.

Return States=True and Sequences=True:
We can access both the sequence of hidden state and the cell states at the same time.
This can be done by configuring the LSTM layer to both return sequences and return states.
lstm1, state_h, state_c = LSTM(1, return_sequences=True, return_state=True)

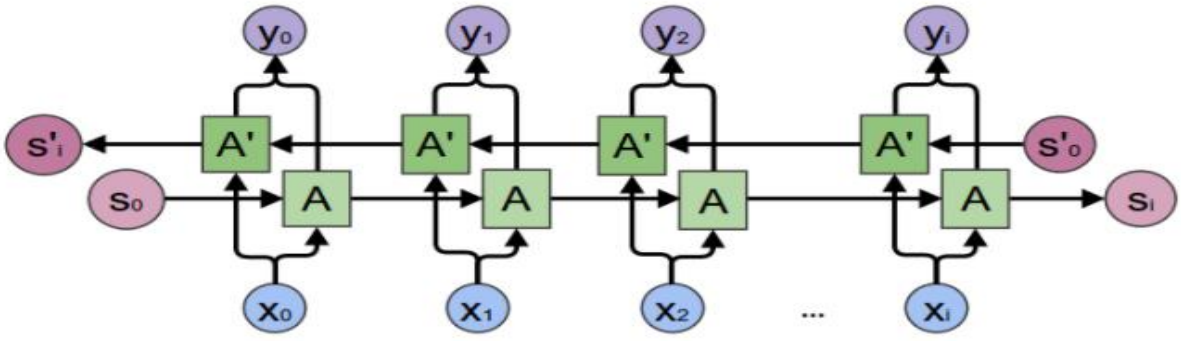
Note:

- That return sequences return the hidden state output for each input time step.
- That return state returns the hidden state output and cell state for the last input time step.
- That return sequences and return state can be used at the same time.

Bidirectional RNN:

Bidirectional recurrent neural networks(RNN) are really just putting two independent RNNs together. The **input sequence is fed in normal time order for one network, and in reverse time order for another.** The outputs of the two networks are usually concatenated at each time step, though there are other options, e.g. summation.

This structure allows the networks to have both backward and forward information about the sequence at every time step. The concept seems easy enough.

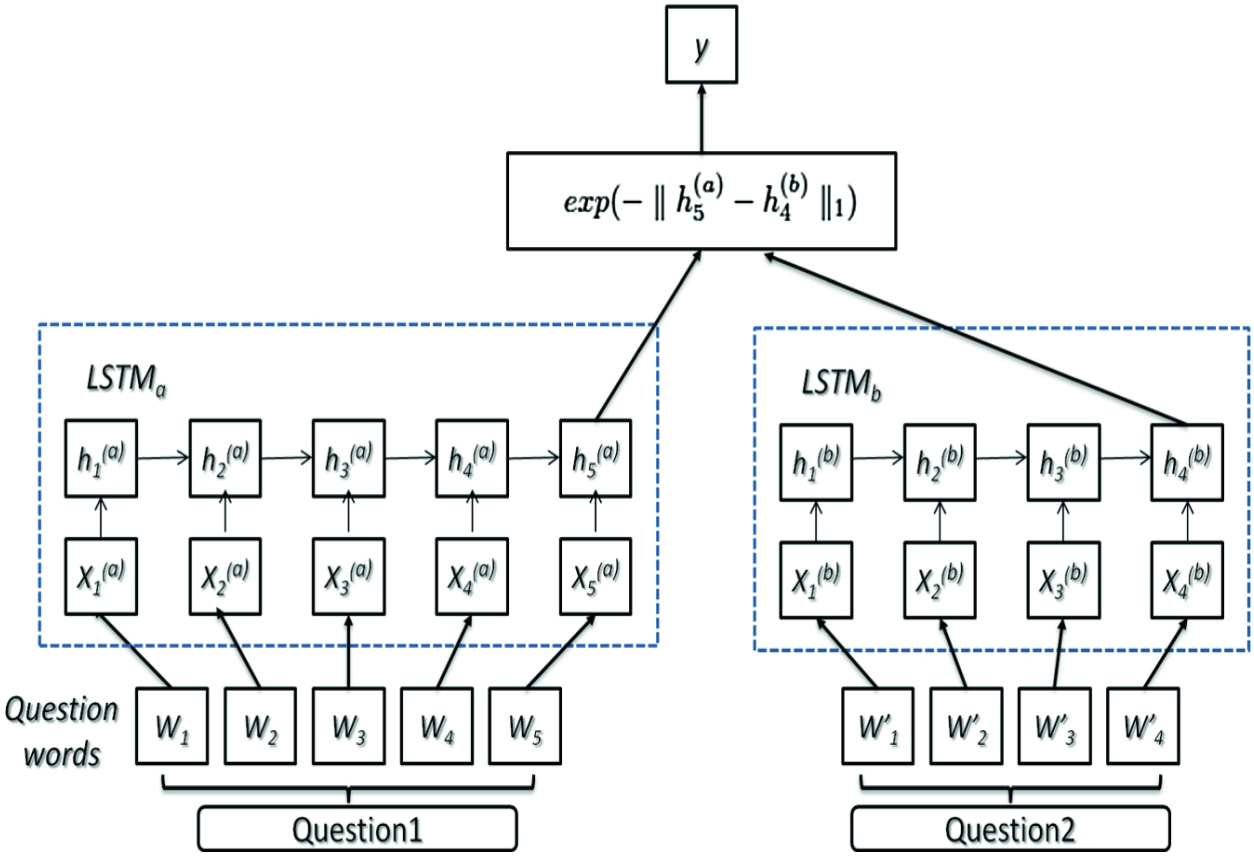


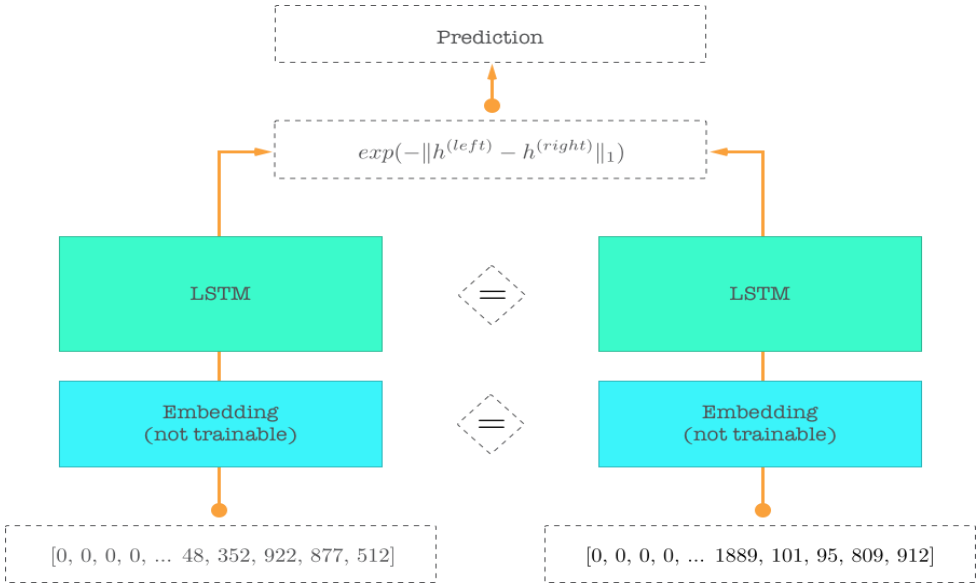
Siamese Network:

Siamese networks are networks that have two or more identical sub-networks in them.
The weights and embedding are shared b/w both sides.

Machine Learning theoretical concepts

By: Vikram Pal



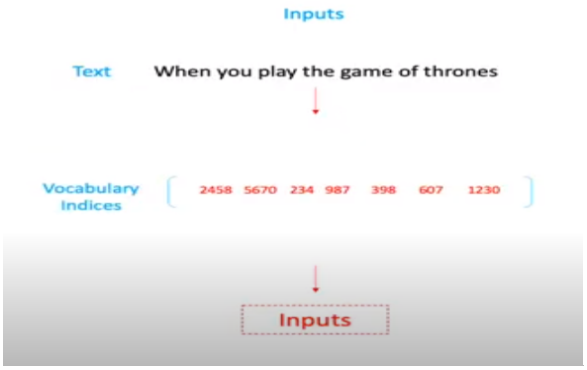


Machine Learning theoretical concepts

By: Vikram Pal

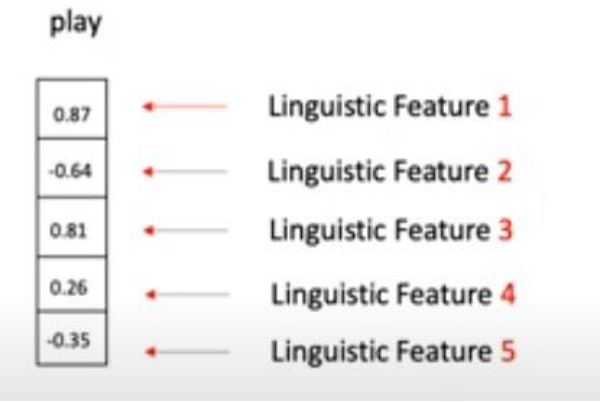
Transformer:

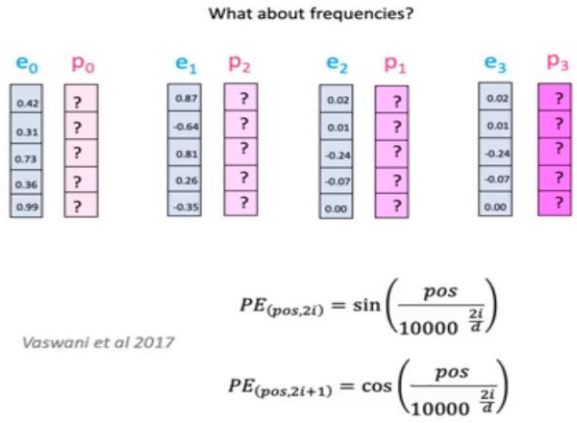
Input and inputs embedding:



We don't feed text input directly to transformer. We use indices of input. In input embedding, the above vector initialized at random number.

Position Embedding:

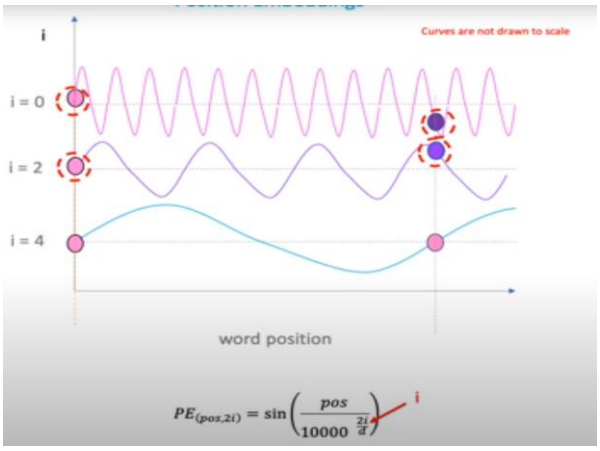




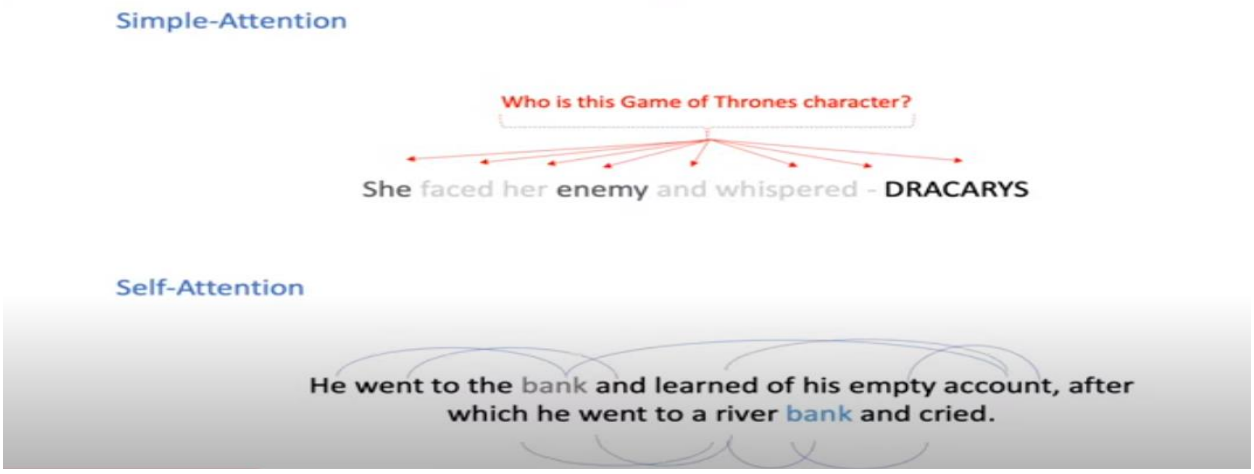
Each number in the vector represent some linguistic features. We use wave frequency to capture position information.

Machine Learning theoretical concepts

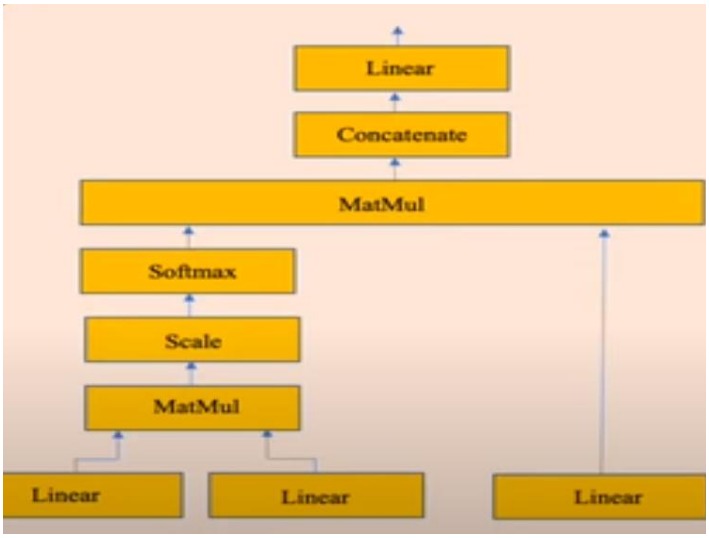
By: Vikram Pal

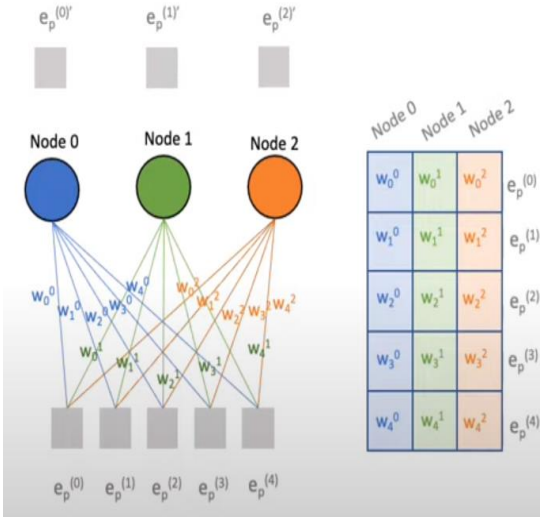


Simple Attention vs self Attention:
The simple attention pays attention based on external query and the self attention pays attention with themselves.



Multiheaded Attention:





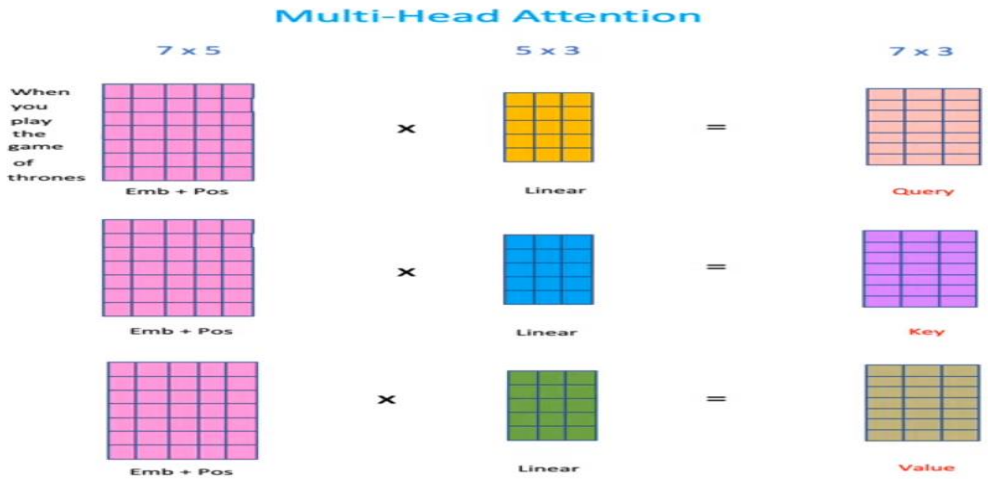
Machine Learning theoretical concepts

By: Vikram Pal

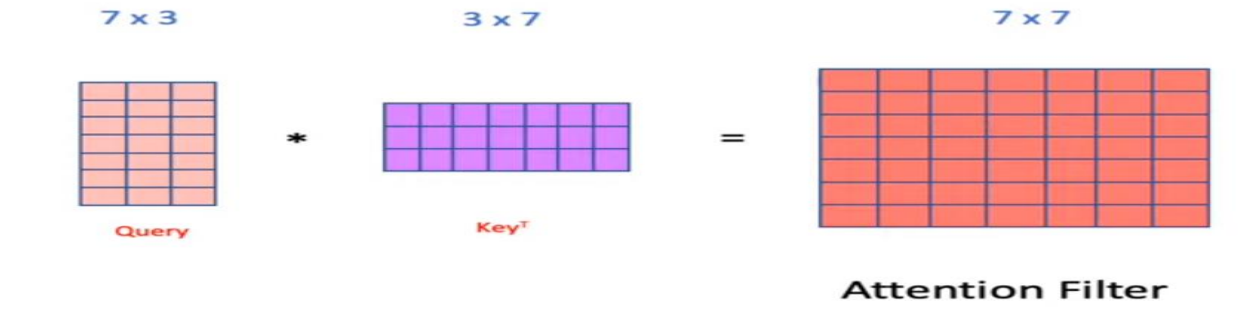
Linear Layer: It has two functions

1. Mapping inputs onto the outputs.
2. Changing matrix/vector dimensions.

Linear layer weights are feed as matrix to model.



Attention filters: It is a dot product of query and key.



7 x 7

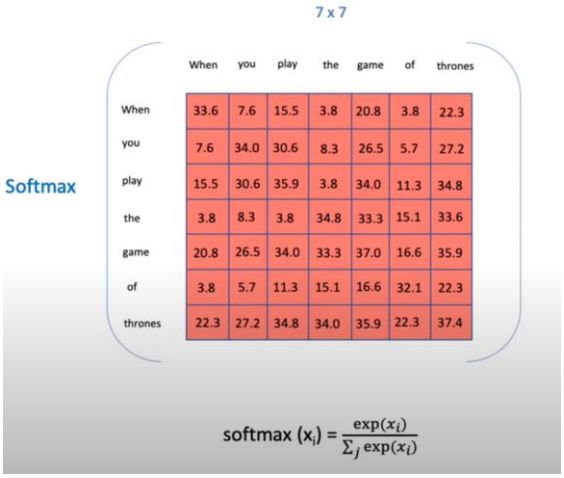
	When	you	play	the	game	of	thrones
When	89	20	41	10	55	10	59
you	20	90	81	22	70	15	72
play	41	81	95	10	90	30	92
the	10	22	10	92	88	40	89
game	55	70	90	88	98	44	87
of	10	15	30	40	44	85	59
thrones	59	72	92	90	95	59	99

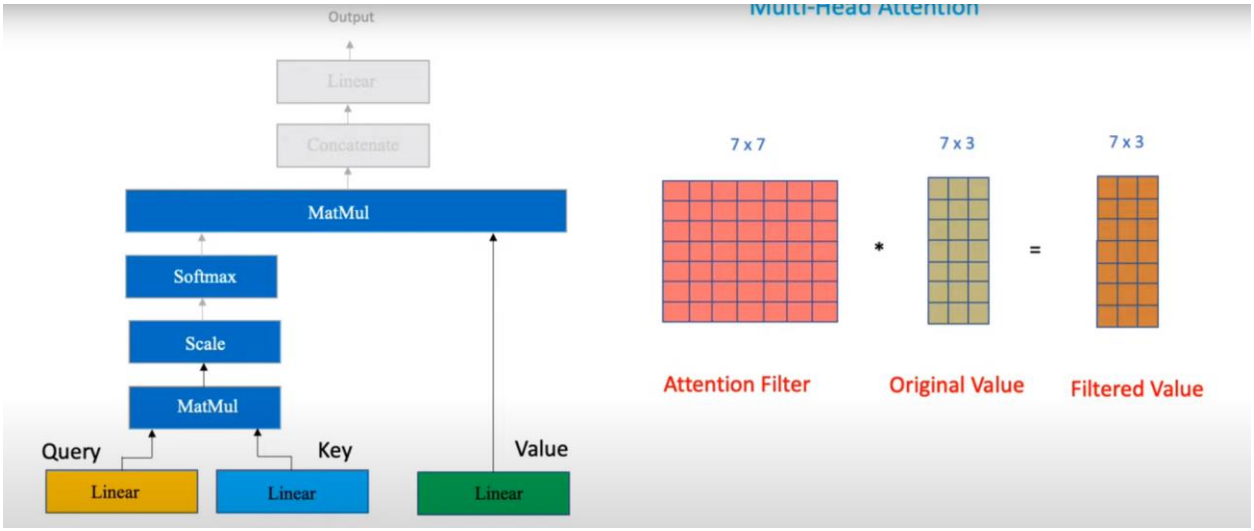
$$\frac{}{\sqrt{d_k}}$$

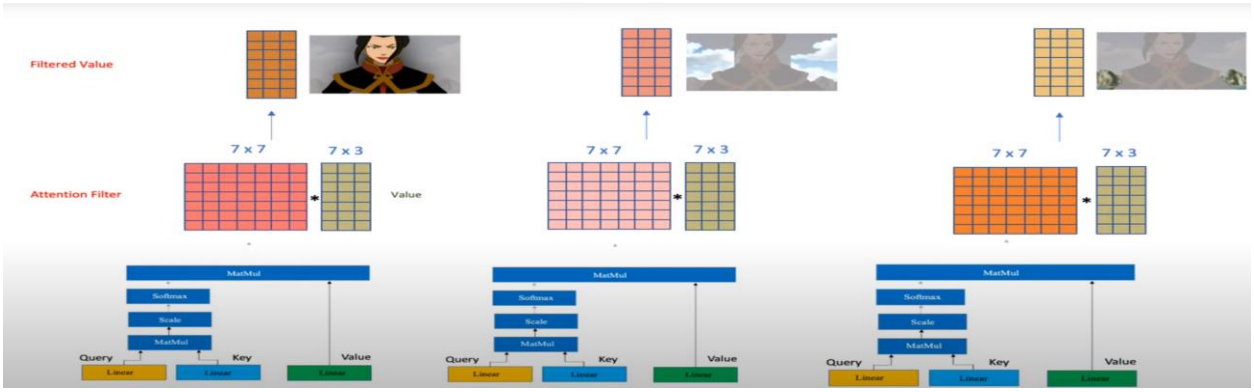
dk=7 in our case.

Machine Learning theoretical concepts

By: Vikram Pal





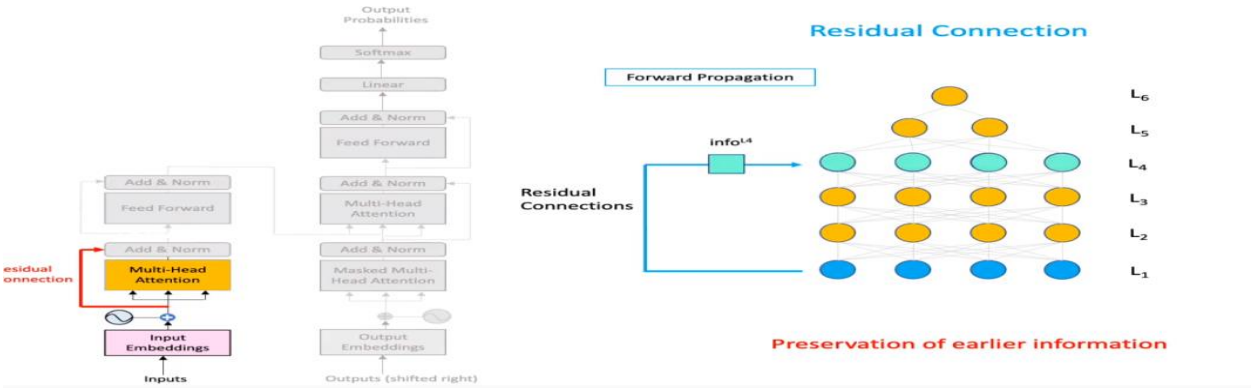


Residual connections: It has mainly two tasks

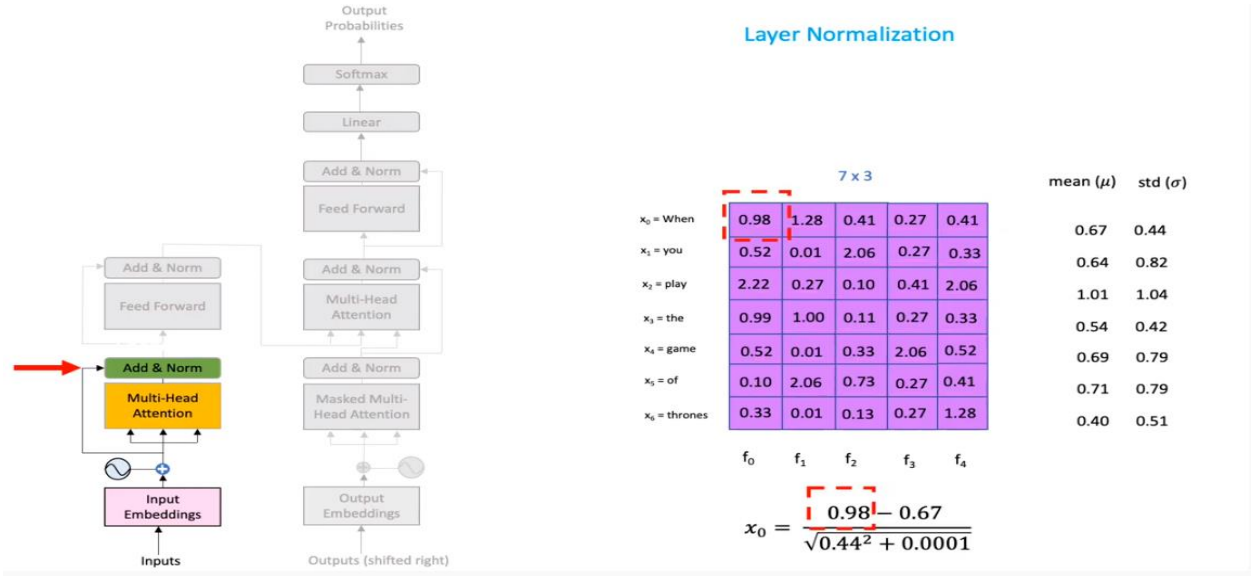
1. **Knowledge Preservation.**
2. **Vanishing Gradient Problem.**

Machine Learning theoretical concepts

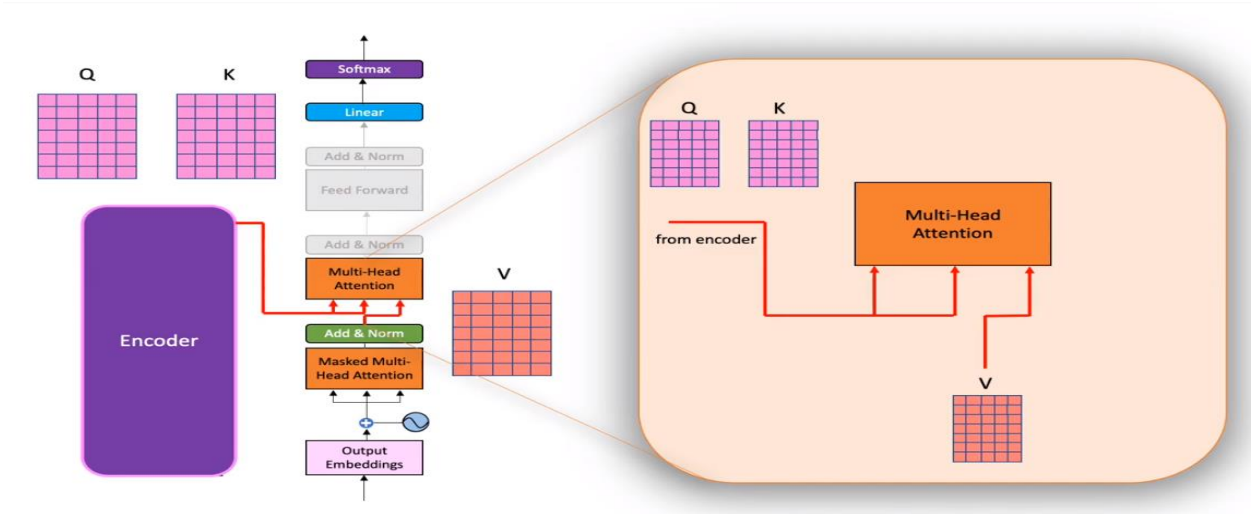
By: Vikram Pal



In Normalization, we computer z-score of matrix.



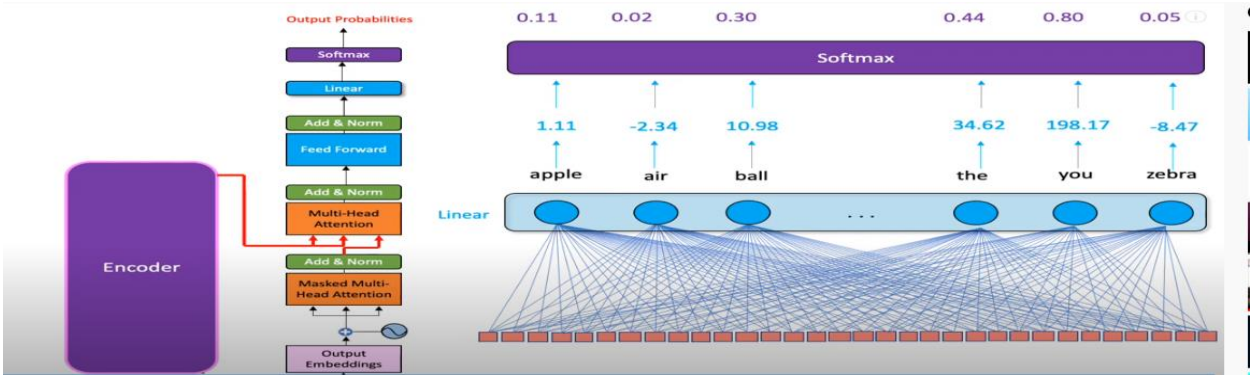
Decoder:



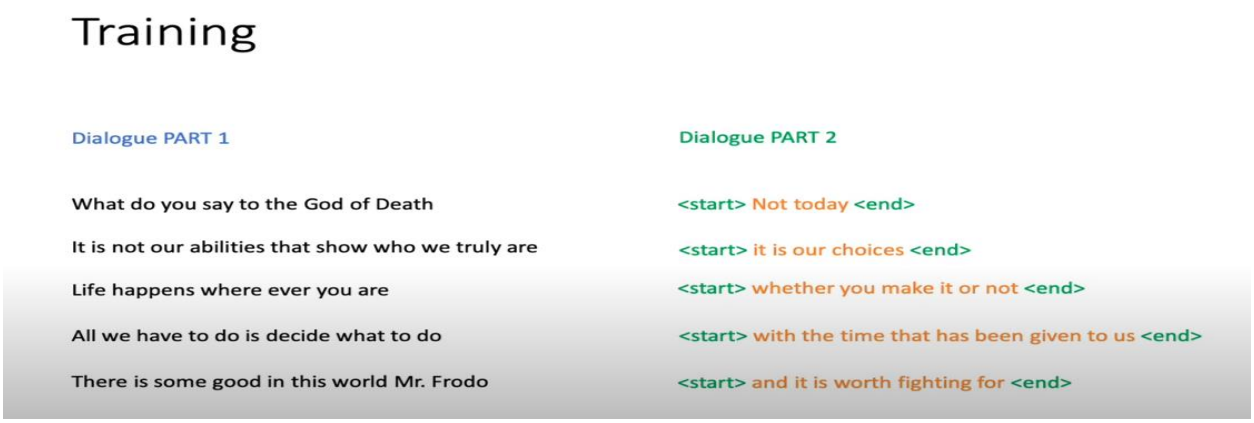
Machine Learning theoretical concepts

By: Vikram Pal

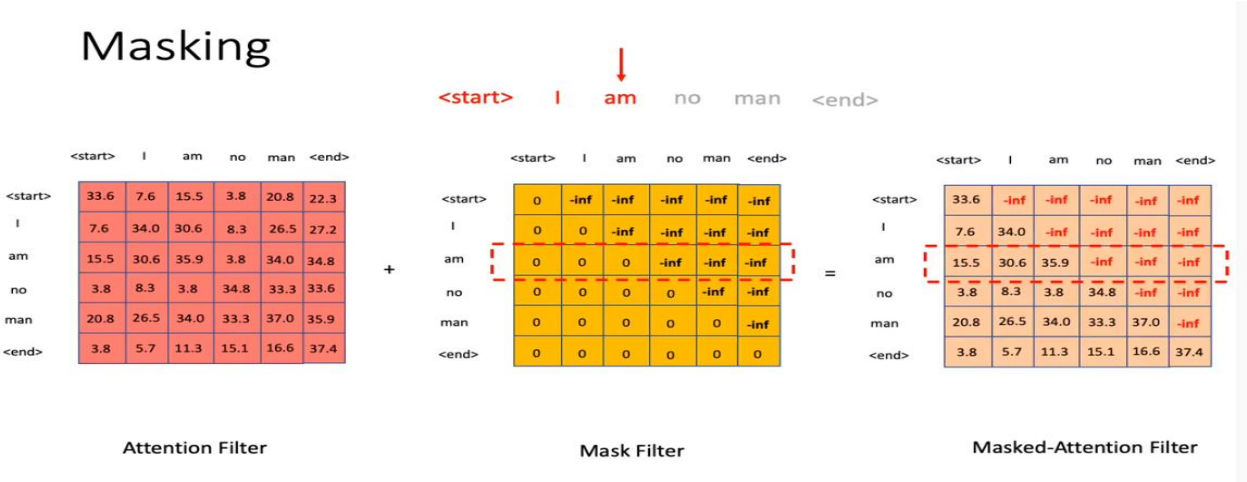
Decoder linear layer



Training Data example:

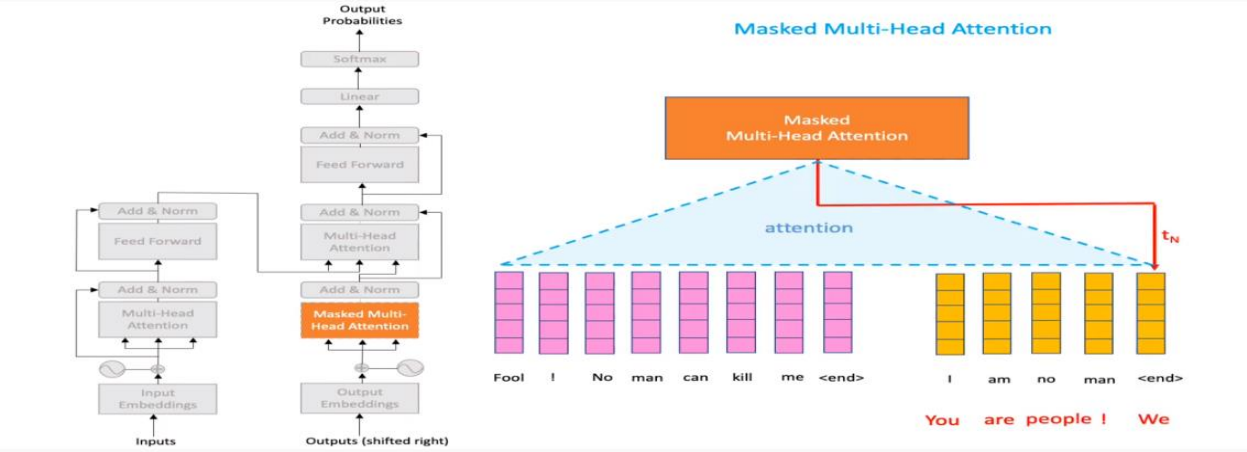


Masking:



Machine Learning theoretical concepts

By: Vikram Pal



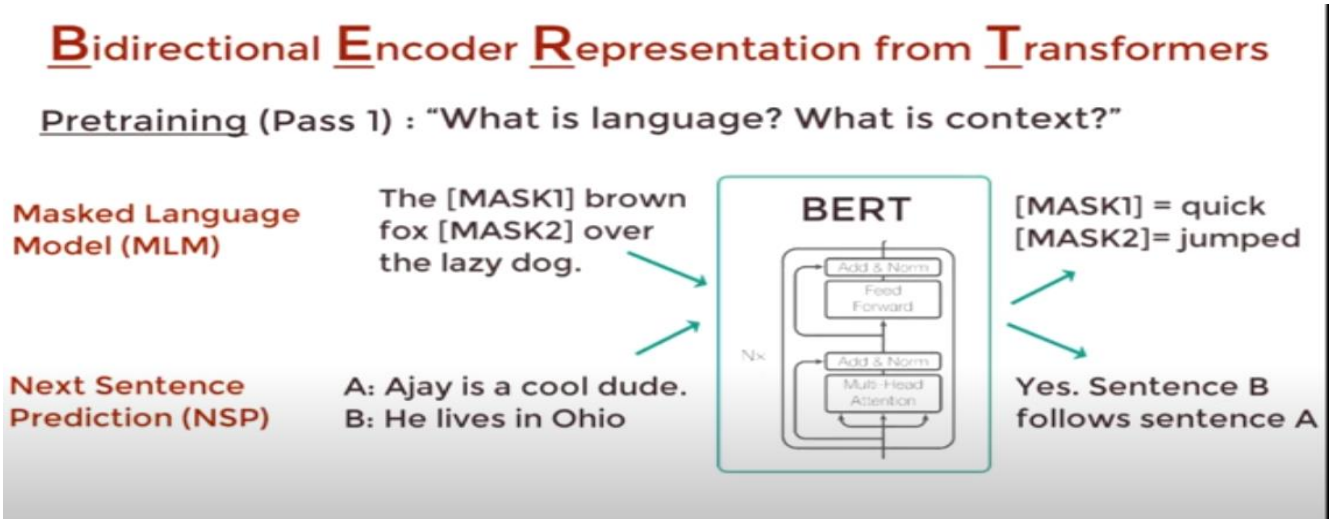
BERT: (Bidirectional Encoder Representation from Transformers):

(Token embedding+ Segment Embedding+ Position Embedding) input → (Binary C value and a bunch of word vectors) → fully connect layer (same number of neurons as the number of words) -→ Softmax activation → Vector to a distribution(**actual label for this distribution will be one hot encoded vector for the actual word.**)

Output: The output is a binary value C and a bunch of word vectors.
Two key things here:

1. **All word vectors are in same size**
2. **All word vectors are generated simultaneously.**

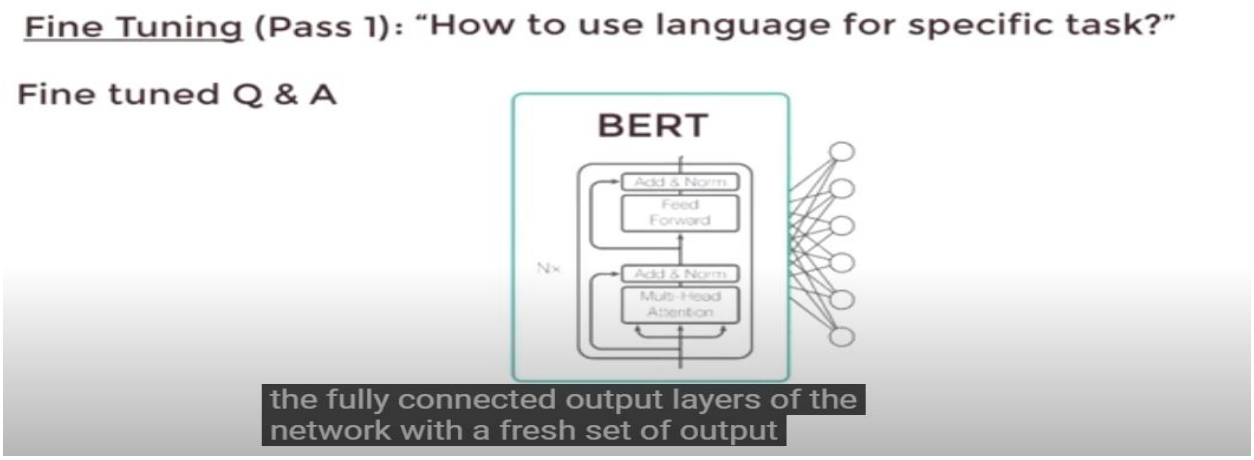
Then we will compare these with actual words one hot encoded and train the network using Cross entropy loss.

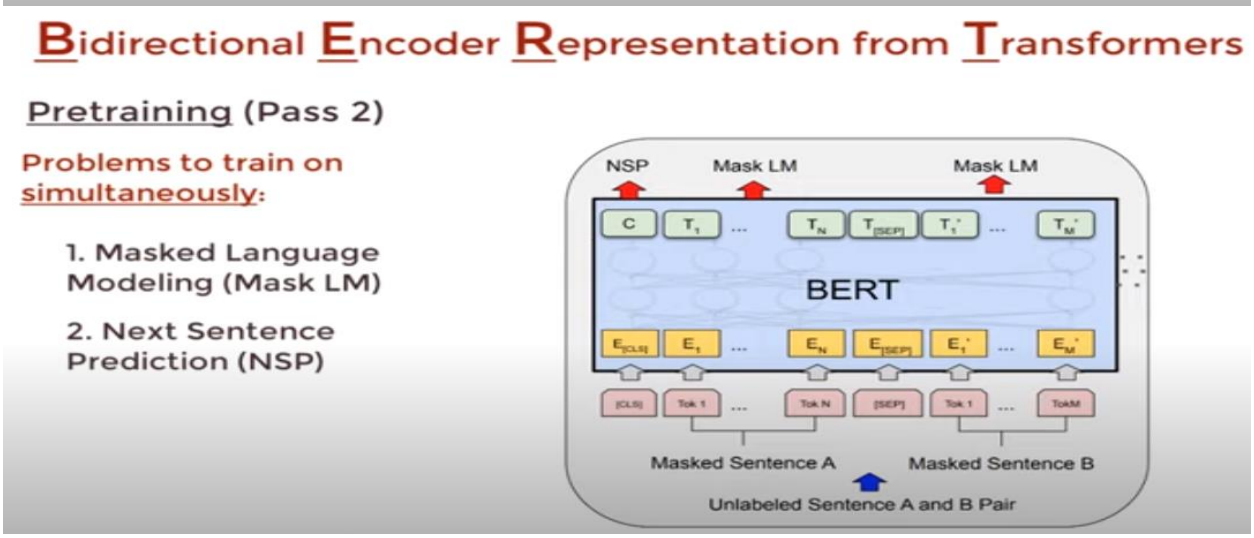


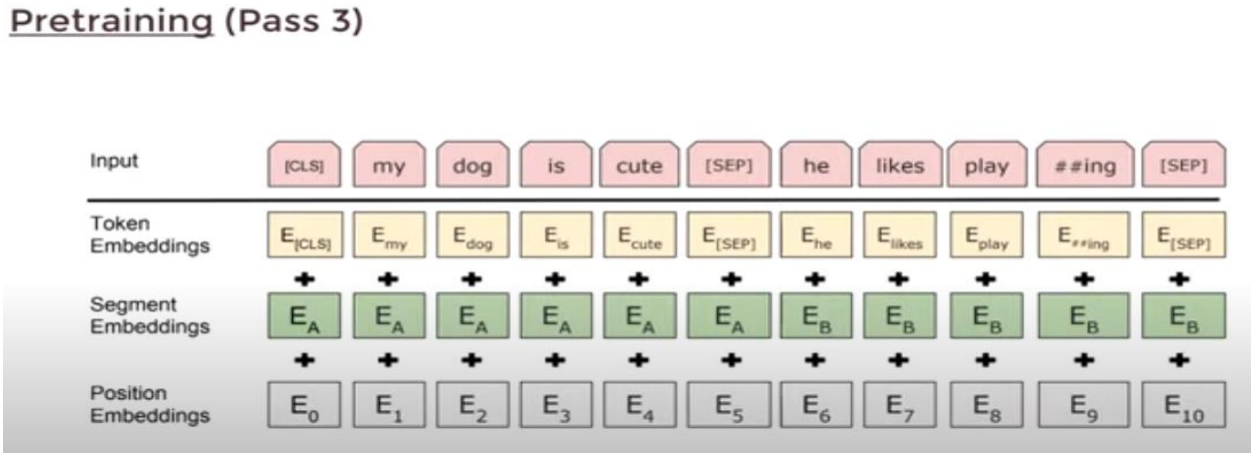
In fine tuning, we replace **fully connected layer wit a fresh set of layers**.

Machine Learning theoretical concepts

By: Vikram Pal







Input to BERT:

- Token embedding: it is pretrained word embedding. The original paper uses Word Pieces – 30 k Vocabulary.
- Segment Embedding: It is sentence numbers that encoded into vector.
- Position Embedding: It is the position of word within that sentence that is encoded into a vector.

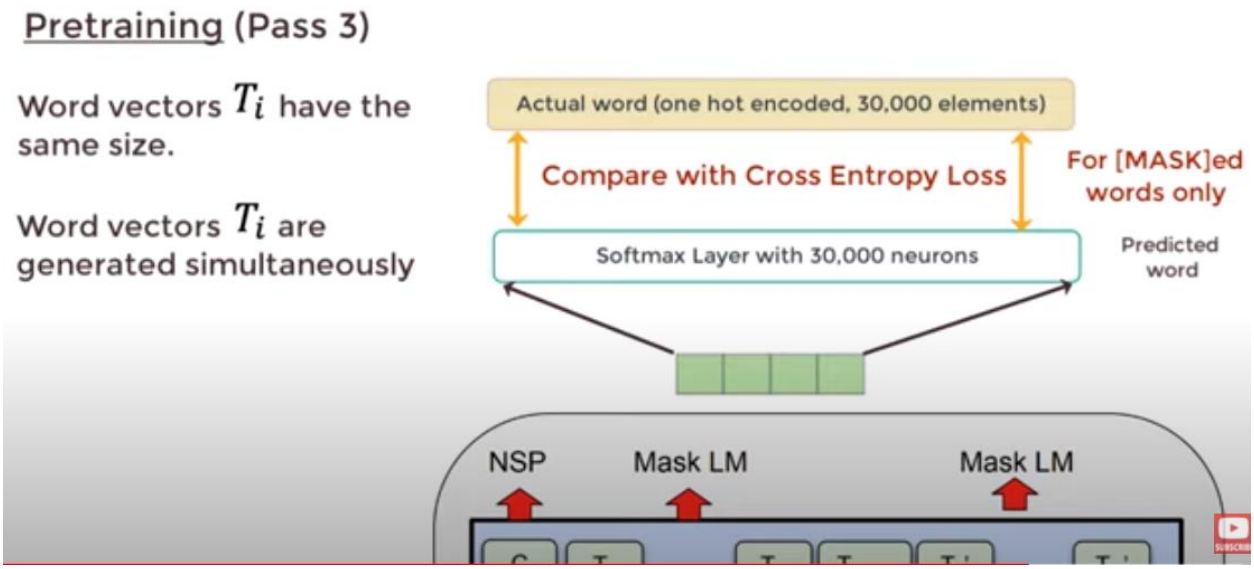
By combining these words embedding, we will get the input word embedding for the Bert.
Segment and positional word embedding are required for temporal ordering because all input vectors are fed in simultaneously into BERT. The language models need this ordering preserved.

Output: The output is a binary value C and a bunch of word vectors.
Two key things here:

3. **All word vectors are in same size**
4. **All word vectors are generated simultaneously.**

Then we will compare these with actual words one hot encoded and train the network using Cross entropy loss.

We will pass generated vectors to a **fully connected layer with the same number of neurons as the number of words**. Then we will apply a Softmax activation function by that we will convert all word vectors to a distribution. The **actual label for this distribution will be one hot encoded vector for the actual word.**
Then we will compare these with actual words one hot encoded and train the network using Cross entropy loss.



Time Series Analysis

Time Series Plot:

A) Seasonality
B) Trend
C) Cyclical
D) Noise

Machine Learning theoretical concepts

By: Vikram Pal

Time Series Models:

A) Naive approach: last period's actuals are used as this period's forecast
B) Exponential smoothing: older data is given progressively-less relative importance whereas newer data is given progressively greater importance.
C) Moving Average: moving-average (MA) model is a common approach for modeling univariate time series (**time series that consists of single (scalar) observations recorded sequentially over equal time increments**)

Model used for forecasting:

- ARIMA/SARIMA
- Linear Regression
- XGBoost
- K-Nearest Neighbors Regression
- Random Forest
- Long Short-Term Memory (LSTM)

In the retail field, the most applicable time series models are the following:

1. **ARIMA** (auto-regressive integrated moving average) models aim to describe the **autocorrelations** in the time series data. When planning short-term forecasts, ARIMA can make accurate predictions. By providing forecasted values for user-specified periods**, it clearly shows results for demand, sales, planning, and production**.

2. **SARIMA** (Seasonal Autoregressive Integrated Moving Average) models are the extension of the ARIMA model that supports uni-variate time series data involving backshifts of **the seasonal period**.

3. **Exponential Smoothing** models generate forecasts by using weighted averages of past observations to predict new values. The essence of these models is in combining **Error, Trend, and Seasonal components into a smooth calculation**.

4. **Long Short-Term Memory (LSTM):** LSTM is a type of recurrent neural network that is particularly useful for making predictions with sequential data. For this purpose, we will use a very simple LSTM. For additional accuracy, seasonal features and additional model complexity can be added

Traditional time series forecasting techniques:

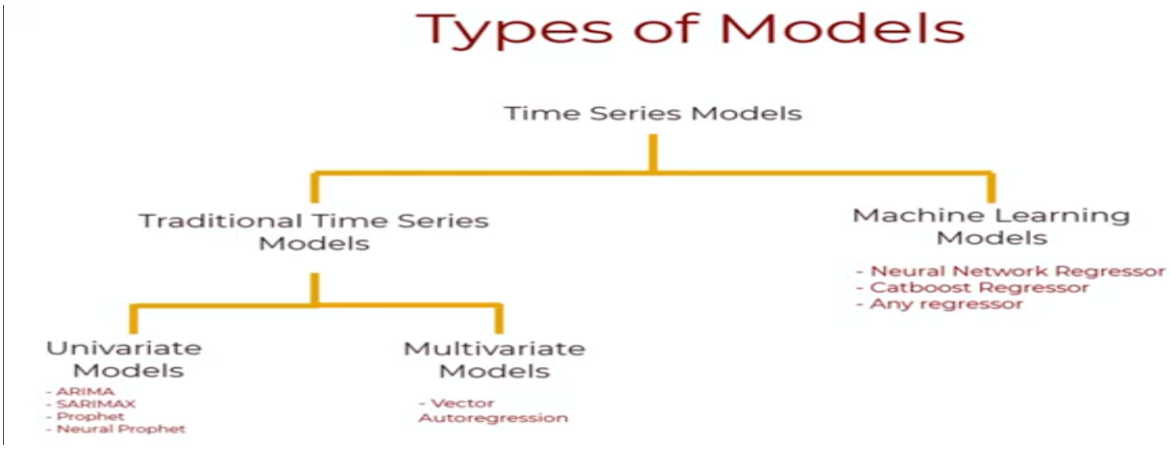
1. ARIMA
2. Prophet (missing data better)
3. Neural Prophet (neural network)
   Disadvantage: We can only add exogenous variable
   (i.e., features we know ahead of time)

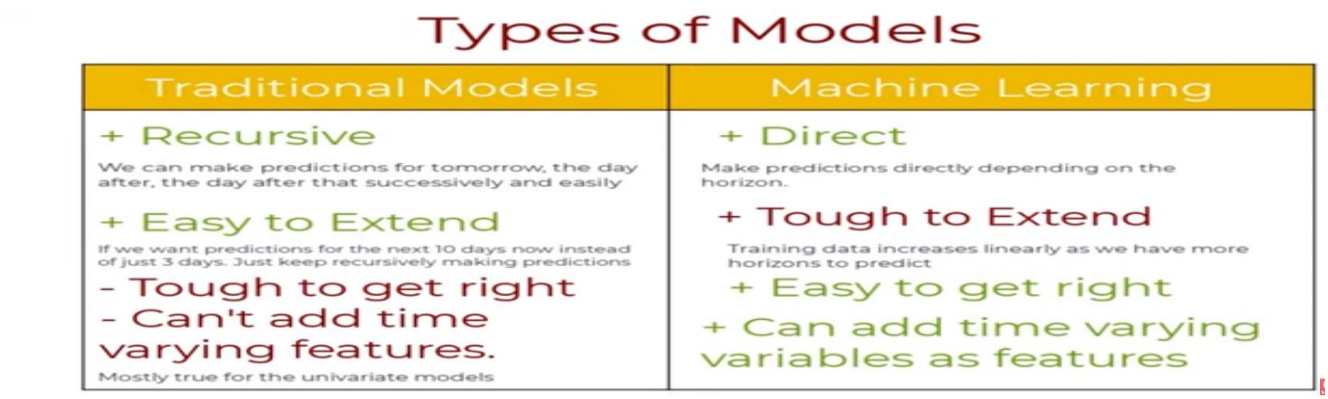
   ARIMA and Prophet are univariate models
4. Vector Autoregression (VAR)
   Multivariate models

Machine Learning theoretical concepts

By: Vikram Pal

For multivariate models, we need more data





Cross validation techniques are better suited for time series data:

Forward Chaining Cross Validation: Forward-chaining cross-validation, also called rolling-origin cross-validation, is **similar to k-fold** but suited to sequential data such as time series. There is no random shuffling of data to begin but a test set may be set aside.

Stationary Process is a stochastic process whose unconditional joint probability distribution does not change when shifted in time.

Statistics

What Is a Null Hypothesis?

A null hypothesis is a type of hypothesis used in statistics that proposes that there is **no difference between certain characteristics of a population** (or data-generating process).

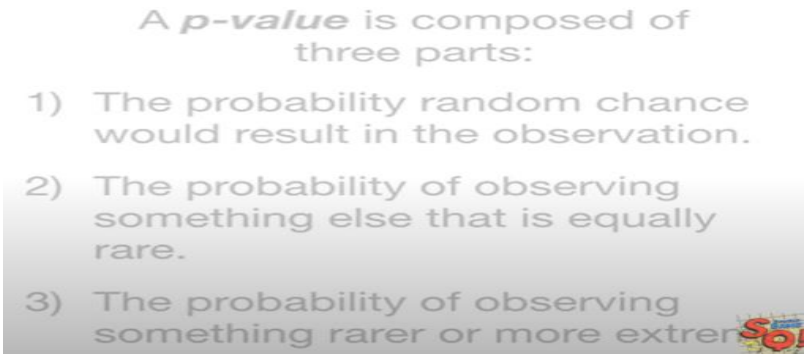
Hypothesis Testing for Investments:

As an example, related to financial markets, assume Alice sees that her investment strategy produces higher average returns than simply buying and holding a stock. The null hypothesis states that there is no difference between the two average returns.

Machine Learning theoretical concepts

By: Vikram Pal

What is the P-value?
A p-value, or probability value, is a number describing how likely it is that **your data would have occurred by random chance** (i.e., that the null hypothesis is true).



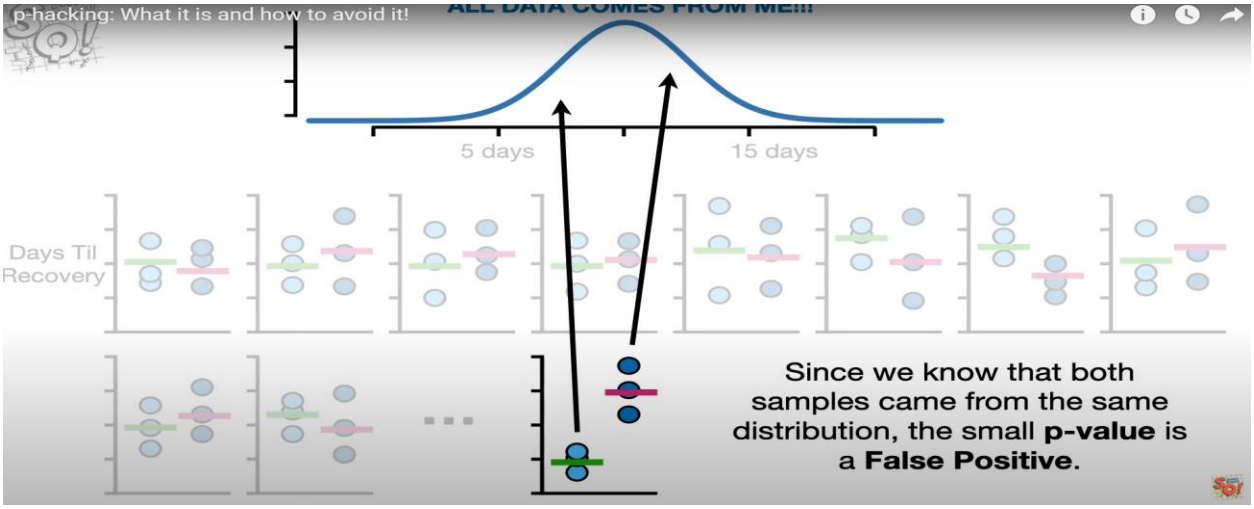
P-Value of two heads:
One coin tossed three times:
HHH, HHT, HTH, THH,TTH, TTT,THT,HTT

1. probability random change: 3/8
2. Probability of equal rare: 3/8
3. Probability of rarer or extreme rarer: 2/8

If the P value is smaller, then it tells us that some other distribution would do a better job explaining the data.

P-hacking:

it refers to the misuse and abuse of analysis techniques and results in being fooled by false positives.



Covariance:

is a measure of the **relationship between two random variables**. The metric evaluates how much – to what extent – the variables change together.

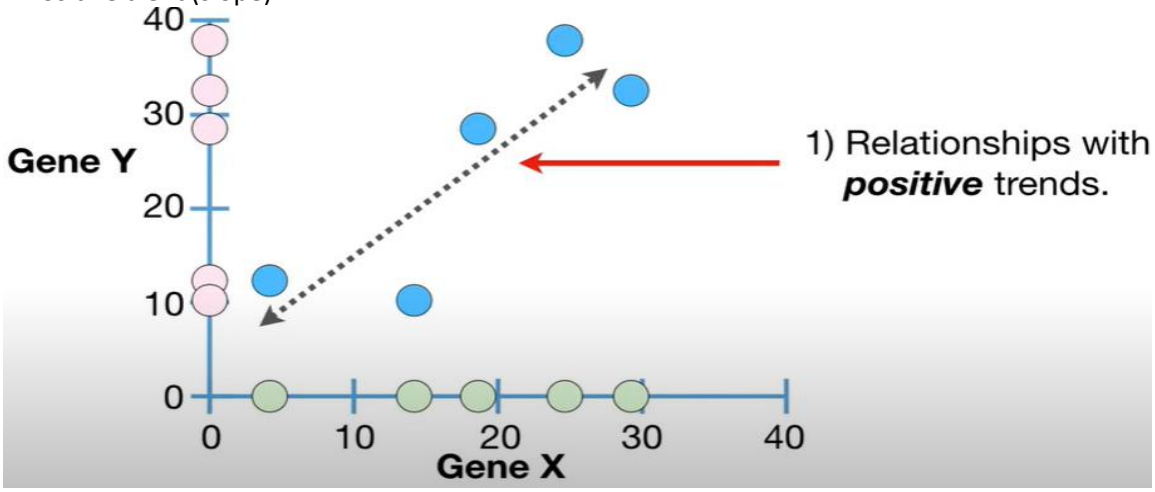
Machine Learning theoretical concepts

By: Vikram Pal

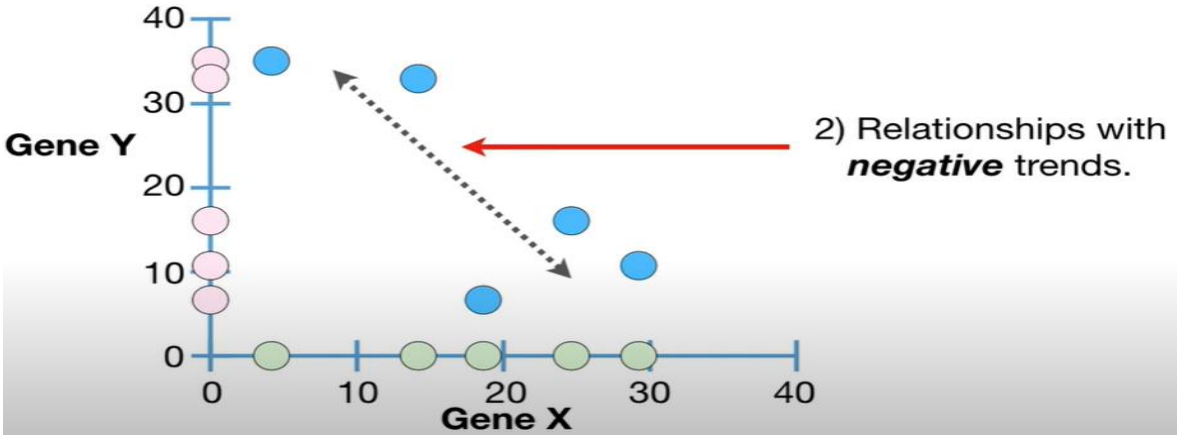
$$\text{Cov}(X, Y) = \frac{\sum(X_i - \bar{X})(Y_j - \bar{Y})}{n}$$

Covariance can classify three types of relationships.

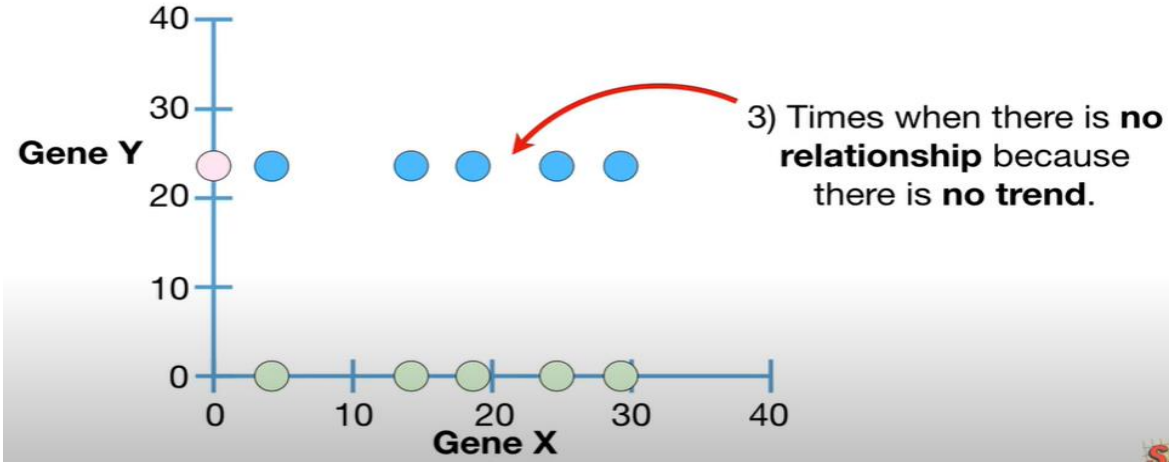
1.Positive trend(slope):



2.Negative Trend:



3.No relationship:



Machine Learning theoretical concepts

By: Vikram Pal

Correlation:

The correlation coefficient is a statistical **measure of the strength of the relationship between the relative movements of two variables.** The values range between -1.0 and 1.0.

$$\rho_{xy} = \frac{\mathrm{Cov}(x,y)}{\sigma_x \sigma_y}$$

where:

$\rho_{xy} = \text{Pearson product-moment correlation coefficient}$

$\mathrm{Cov}(x,y) = \text{covariance of variables } x \text{ and } y$

$\sigma_x = \text{standard deviation of } x$

$\sigma_y = \text{standard deviation of } y$

Causation:

indicates that **one event is the result of the occurrence of the other event**, i.e., there is a causal relationship between the two events.

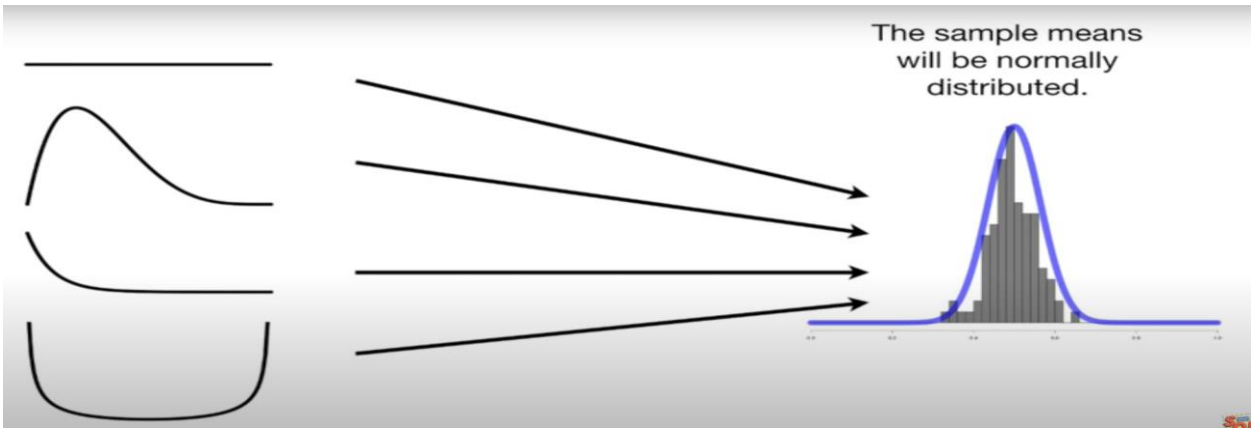
The use of a **controlled study is the most effective** way of establishing causality between variables. In a controlled study, the sample or population is split in two, with both groups being comparable in almost every way. The two groups then receive different treatments, and the outcomes of each group are assessed.

For example, in medical research, one group may receive a placebo while the other group is given a new type of medication. If the two groups have noticeably different outcomes, the different experiences may have caused the different outcomes.

e.g.: **(age and experience)**

Central limit theorem:

The Central Limit Theorem states that the sampling distribution of the sample means approaches a normal distribution as the sample size gets larger — no matter what the shape of the population distribution.



Machine Learning theoretical concepts

By: Vikram Pal

Note:The Standard deviation of means is called Standard error.

Interquartile range

QR = Q3 – Q1
Where, Q3 is the third quartile (75 percentile)
Where, Q1 is the first quartile (25 percentile)

How to find outliers?
Widely used – Any data point that lies outside the 1.5 * IQR
Lower bound = Q1 – (1.5 * IQR)
Upper bound = Q3 + (1.5 * IQR)

Hypothesis testing:

https://towardsdatascience.com/hypothesis-testing-in-machine-learning-using-python-a0dc89e169ce

What is hypothesis testing ?
It is a statistical decision-making technique. In which we assume about population.

why do we use it ?
A hypothesis test evaluates two **mutually exclusive statements about a population to determine which statement is best supported by the sample data.** When we say that a finding is statistically significant, it's thanks to a hypothesis test.

what is basis of hypothesis ?
The basic of hypothesis is normalisation and standard normalisation. all our hypothesis revolves around basic of these 2 terms

Normal Distribution (3 M are equal) –
A variable is said to be normally distributed or have a normal distribution if its distribution has the shape of a normal curve — a special bell-shaped curve.
which has all of the following properties: 1. The mean, median, and mode are equal.

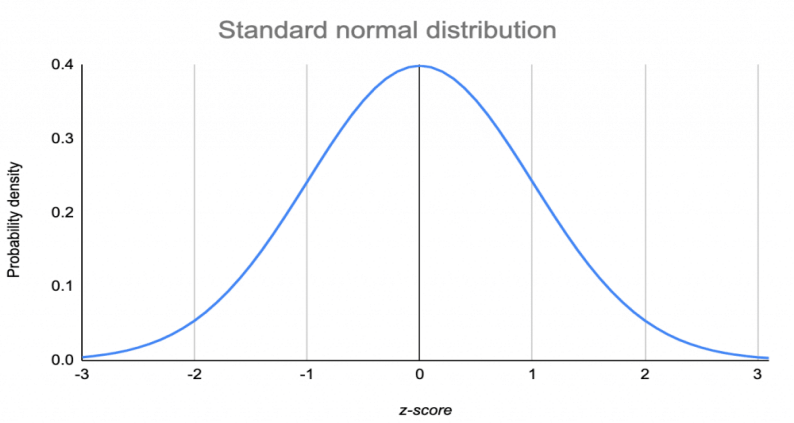
Standardised Normal Distribution (mean 0 and SD 1) :
A standard normal distribution is a normal distribution with **mean 0 and standard deviation 1**

Standard Normal Distribution

Machine Learning theoretical concepts

By: Vikram Pal

$$x_{new} = \frac{x - \mu}{\sigma}$$



Which is important parameter of hypothesis testing ?
Null hypothesis :-
The null hypothesis is a general statement or default position that there is **no relationship between two measured phenomena, or no association among groups**

Example : a company production is = 50 unit/per day etc.

Alternative hypothesis :-
The alternative hypothesis is the hypothesis used in hypothesis testing that is contrary to the null hypothesis. **It is usually taken to be that the observations are the result of a real effect** (with some amount of chance variation superposed)

Example : a company production is !=50 unit/per day etc.

Level of significance (0.05): Refers to the degree of significance in which we accept or reject the null hypothesis. 100% accuracy is not possible for accepting or rejecting a hypothesis, so we therefore select a level of significance that is usually 5%.

Type I error: When we reject the null hypothesis, although that hypothesis was true.

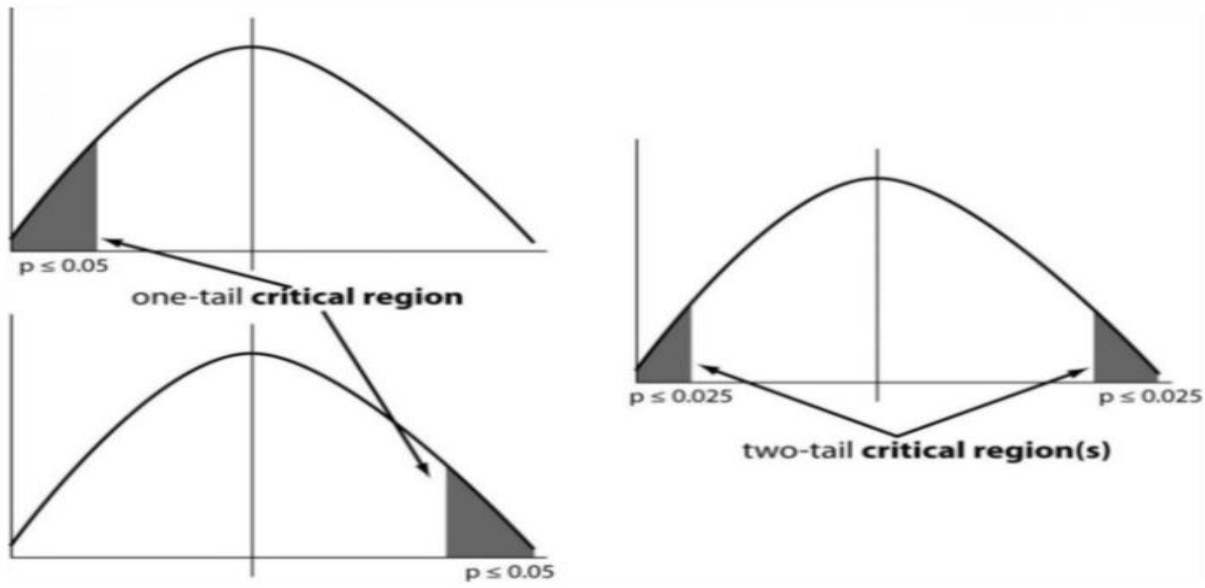
Type II errors: When we accept the null hypothesis, but it is false.

One tailed test (>=) :- A test of a statistical hypothesis , where the region of rejection is on only one side of the sampling distribution , is called a one-tailed test.
Example :- **a college has ≥ 4000 student** or data science ≤ 80% org adopted.

Two-tailed test (not equal to) :- A two-tailed test is a statistical test in which the critical area of a distribution is two-sided and tests whether a sample is greater than or less than a certain range of values.
Example : a college != 4000 student or data science != 80% org adopted

Machine Learning theoretical concepts

By: Vikram Pal



one and two-tailed images

P-value :- The P value, or calculated probability, is the probability of finding the observed, or more extreme, results when the null hypothesis (H 0) of a study question is true — the definition of 'extreme' depends on how the hypothesis is being tested.

Widely used hypothesis testing type :-

T-test:

It is also used for **comparing two population mean when SD is unknown.**

A t-test is a type of inferential statistic which is used to determine if there is a **significant difference between the means of two groups which** may be related in certain features.

T-test has 2 types : 1. one sampled t-test 2. two-sampled t-test.

One sample t-test : The One Sample T Test determines whether the sample mean is statistically different from a known or hypothesised population mean.

Two sampled T-test :- The Independent Samples t Test or 2-sample t-test compares the means of two **independent groups in order to determine whether there is statistical evidence that the associated population means are significantly different.**

Z-test:

It is used for **comparing two population mean when SD is known**. You would use a Z test if:

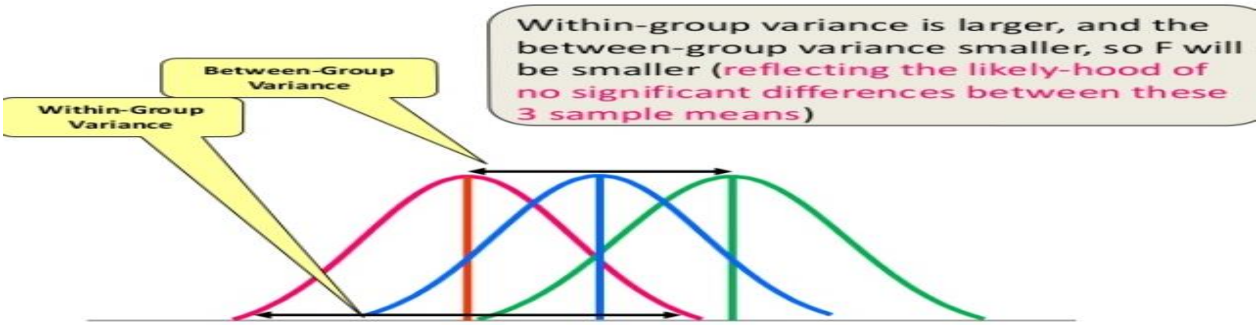
- Your sample size is greater than 30. Otherwise, use a t test.
- Data points should be independent from each other. In other words, one data point isn't related or doesn't affect another data point.
- Your data should be normally distributed. However, for large sample sizes (over 30) this doesn't always matter.
- Your data should be randomly selected from a population, where each item has an equal chance of being selected.
- Sample sizes should be equal if possible.

Machine Learning theoretical concepts

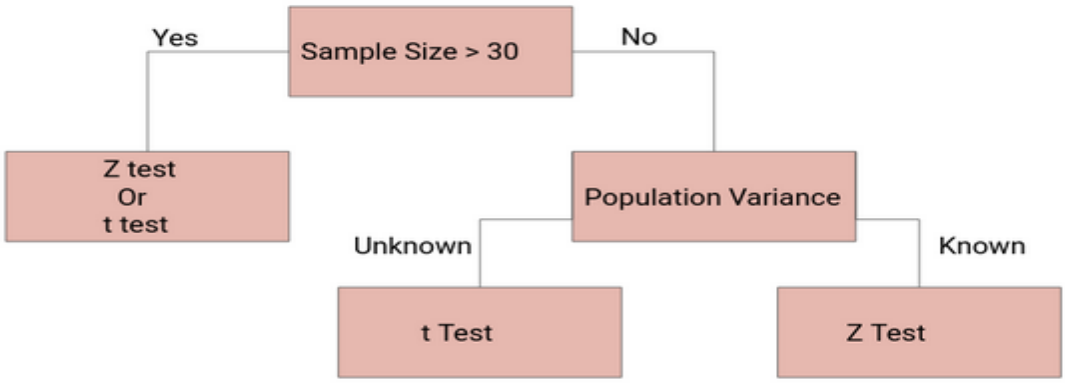
By: Vikram Pal

Two-sample Z test- In two sample z-test , like t-test here we are checking **two independent data groups and deciding whether sample mean of two group is equal or not**.
H0 : mean of two group is 0
H1 : mean of two group is not 0
Example : we are checking in blood data after blood and before blood data.

ANOVA: It is used for comparing more than two groups. The analysis of variance or ANOVA is a statistical inference test that lets you compare multiple groups at the same time.
F = Between group variability / Within group variability



Number of Independent Variables	ANOVA Used	Example
One	One-way ANOVA	Independent: *-fertilizer type* Dependent: -fruit produced
Two or more	Two-way ANOVA	Independent: *-fertilizer type* *-frequency of watering* Dependent: -fruit produced



Machine Learning theoretical concepts

By: Vikram Pal

Chi-square Test:

Chi Square test :**difference between the expected frequencies and the observed frequencies in one or more categories** of a contingency table

Contingency Table:

		Sport Preference			
		Archery	Boxing	Cycling	
Gender	Female	35	15	50	100
	Male	10	30	60	100
		45	45	110	200

Assume that we want to test if there is a statistically significant difference in Genders(M, F) population between Smokers and Non-Smokers.

Example: a scientist wants to know if education level and marital status are related for all people in some country.

Sampling Techniques:

Sampling is a method that allows us to get information about the population based on the statistics from a subset of the population (sample), without having to investigate every individual.

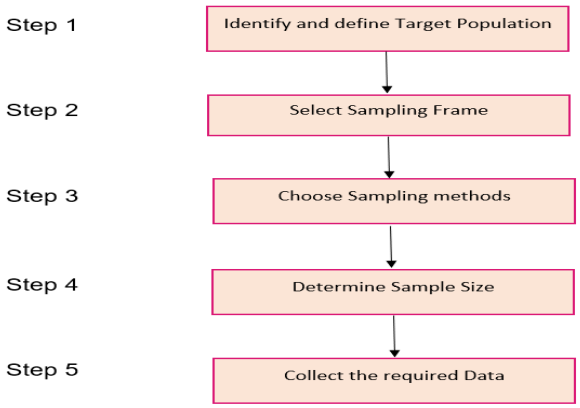
Let's say we go to a basketball court and take the average height of all the professional basketball players as our sample. This will not be considered a good sample because generally, a basketball player is taller than an average male and it will give us a bad estimate of the average male's height.

Here's a potential solution – find random people in random situations where our sample would not be skewed based on heights.

Steps involved in Sampling:

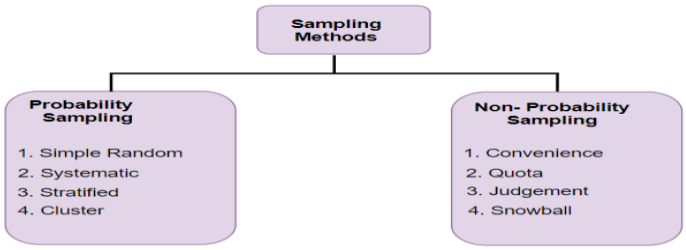
Machine Learning theoretical concepts

By: Vikram Pal



Different Types of Sampling Techniques

- **Probability Sampling**: In probability sampling, every element of the population has an equal chance of being selected. Probability sampling gives us the best chance to create a sample that is truly representative of the population
- **Non-Probability Sampling**: In non-probability sampling, all elements do not have an equal chance of being selected. Consequently, there is a significant risk of ending up with a non-representative sample which does not produce generalizable results

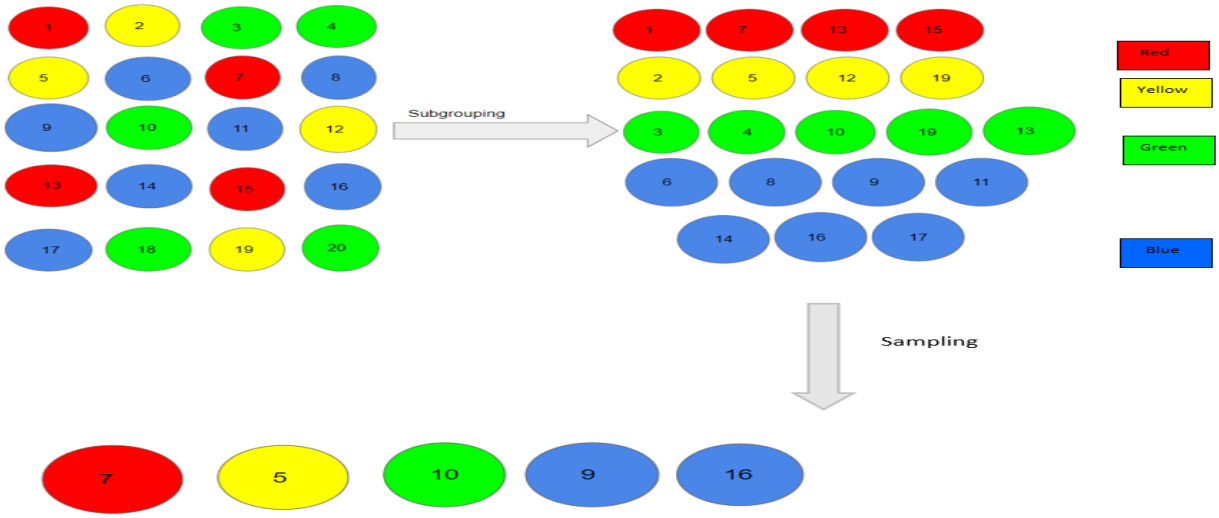


1. **Simple Random:** This is a type of sampling technique you must have come across at some point. Here, every individual is chosen entirely by chance and each member of the population has an equal chance of being selected.

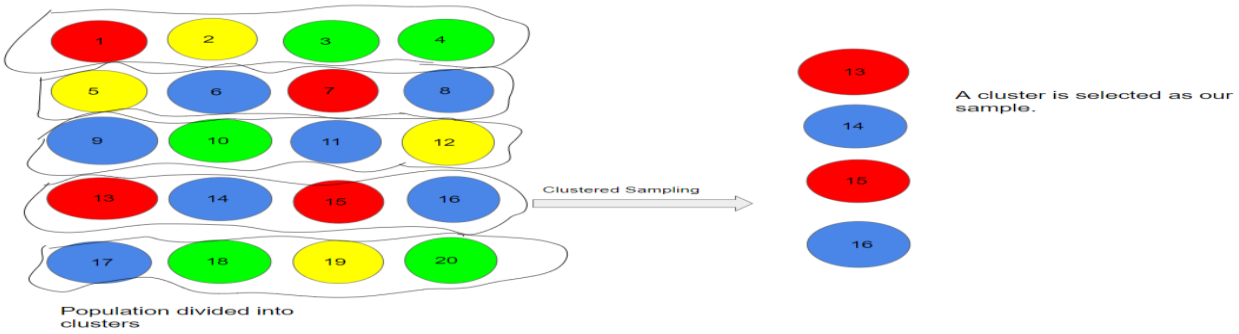
2. Stratified Sampling: In this type of sampling, we divide the population into subgroups (called strata) based on different traits like gender, category, etc. And then we select the sample(s) from these subgroups:

Machine Learning theoretical concepts

By: Vikram Pal



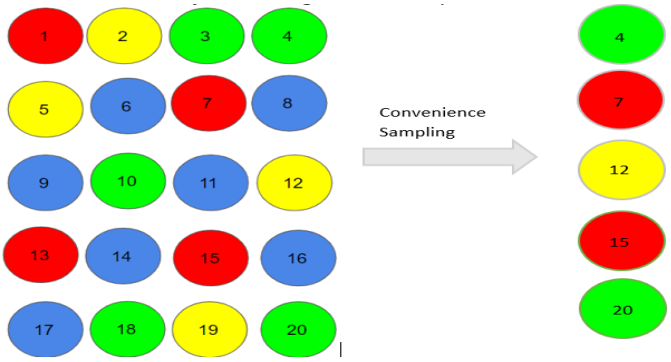
Cluster Sampling: In a clustered sample, we use the subgroups of the population as the sampling unit rather than individuals. The population is divided into subgroups, known as clusters, and a whole cluster is randomly selected to be included in the study:



Types of Non-Probability Sampling:

1. Convenience Sampling: This is perhaps the easiest method of sampling because individuals are selected based on their availability and willingness to take part.

Here, let's say individuals numbered 4, 7, 12, 15 and 20 want to be part of our sample, and hence, we will include them in the sample

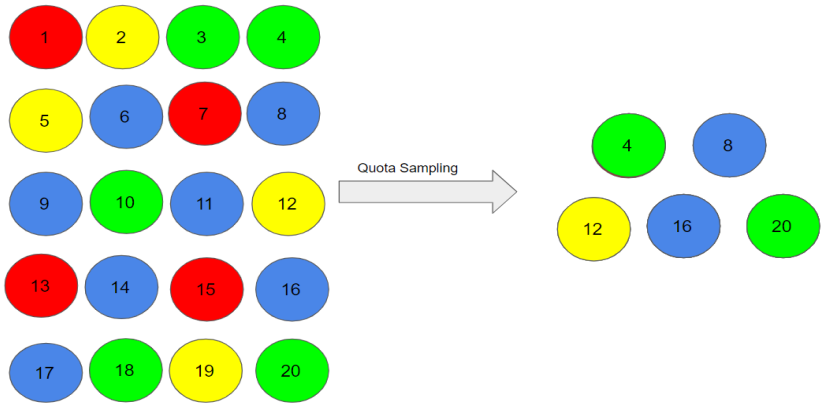


Convenience sampling is prone to significant bias, because the sample may not be the representation of the specific characteristics such as religion or, say the gender, of the population.

Machine Learning theoretical concepts

By: Vikram Pal

2. Quota Sampling: In this type of sampling, we choose items based on predetermined characteristics of the population. Consider that we have to select individuals having a number in multiples of four for our sample:



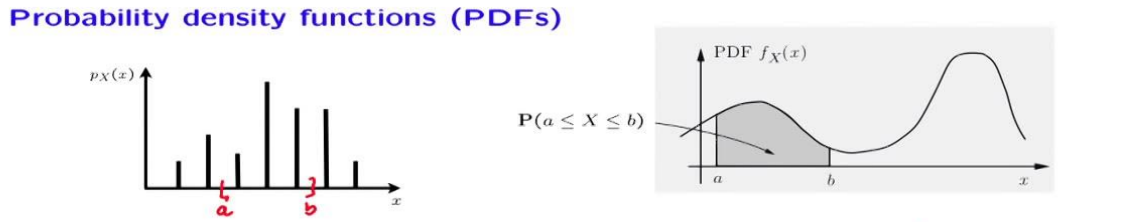
3. Judgment Sampling

It is also known as selective sampling. It depends on the judgment of the experts when choosing whom to ask to participate.

4. Snowball Sampling:

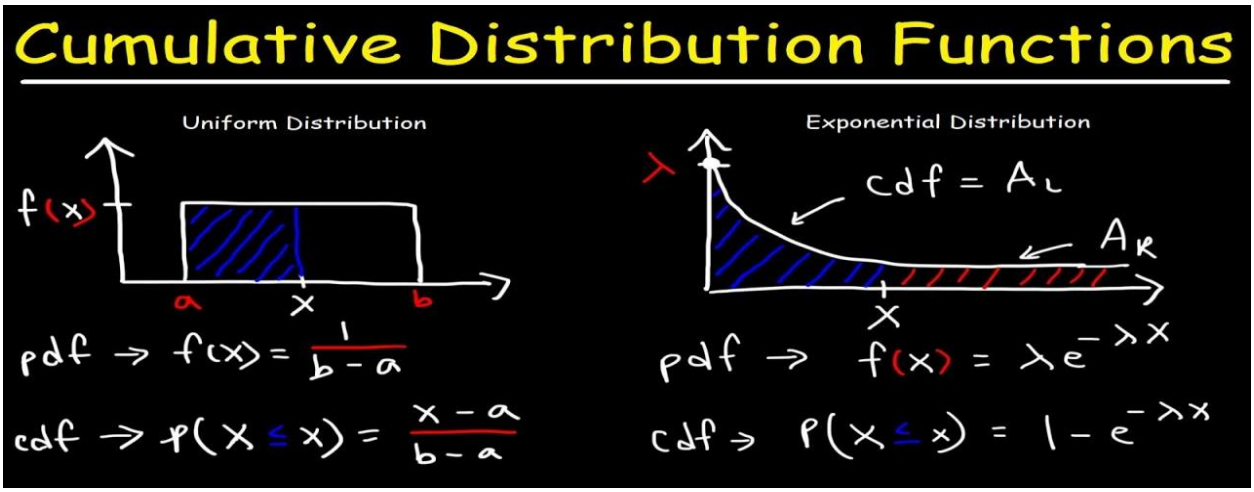
I quite like this sampling technique. Existing people are asked to nominate further people known to them so that the sample increases in size like a rolling snowball. This method of sampling is effective when a sampling frame is difficult to identify.

Probability Density Function:

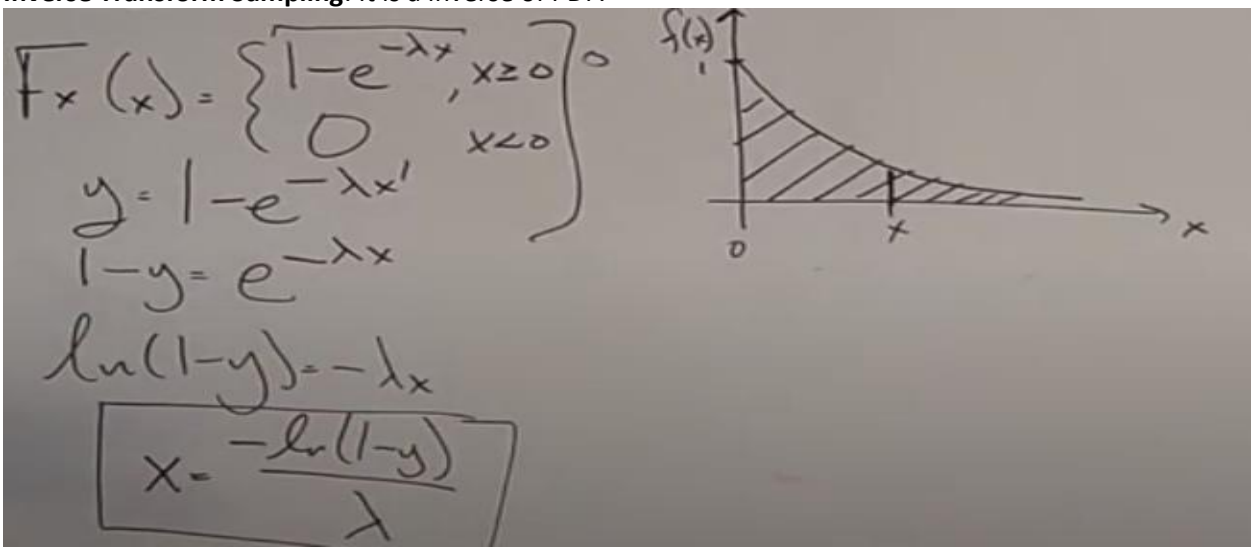


Machine Learning theoretical concepts

By: Vikram Pal



Inverse Transform Sampling: It is a inverse of PDF.



Monte Carlo sampling methods that are able to **draw independent samples from the distribution.**
Markov Chain Monte Carlo: method **draw samples where the next sample is dependent** on the existing sample, called a Markov Chain.

Machine Learning theoretical concepts

By: Vikram Pal

Probability:

Conditional Probability:

Conditional Probability Formula

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}$$

Probability of A given B

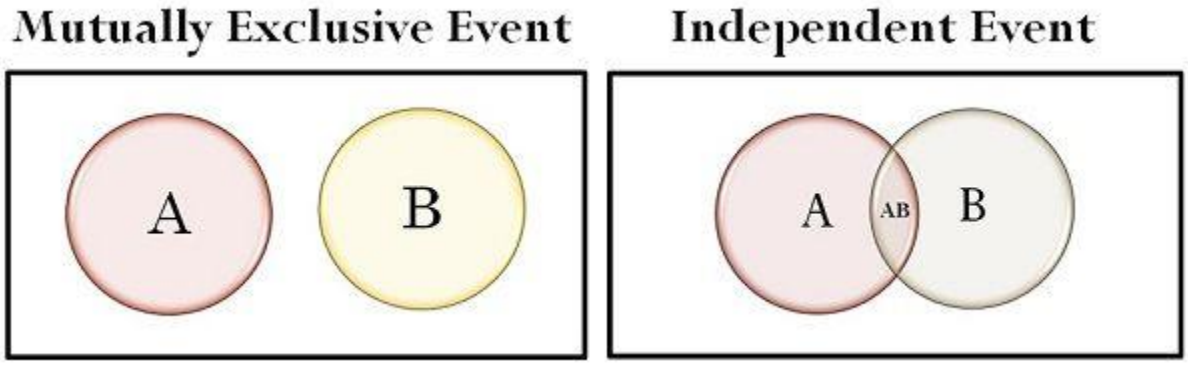
Probability of A and B

Probability of B

Intendent Probability:

If one event already occurred, then that wouldn't affect the occurrence of other events. '

Mutually exclusive: events are things that can't happen at the same time. For example, you can't run backwards and forwards at the same time.



Bayes Theorem:

$$P(B_J \mid A) = \frac{P(A \mid B_J)\,P(B_J)}{\sum_{i=1}^{n} P(A \mid B_i)\,P(B_i)}$$

Calcworkshop.com

Machine Learning theoretical concepts

By: Vikram Pal

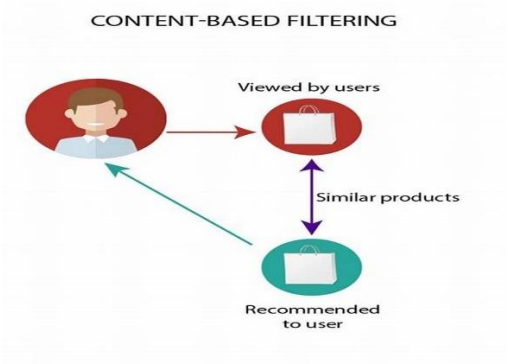
Recommendation System:

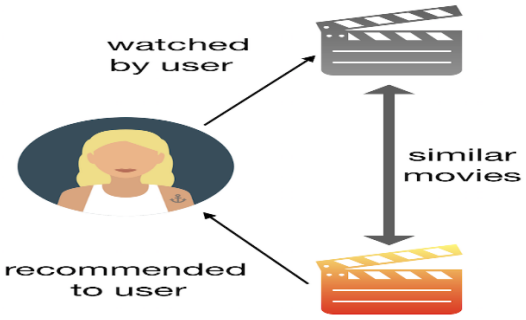
Popularity based recommendation:

Popularity based recommendation system works with the trend. It basically uses the items which are in trend right now.
E.g.: new user to OTT platform and suggesting them a movie without any idea of user preferences.

Content based recommendation:

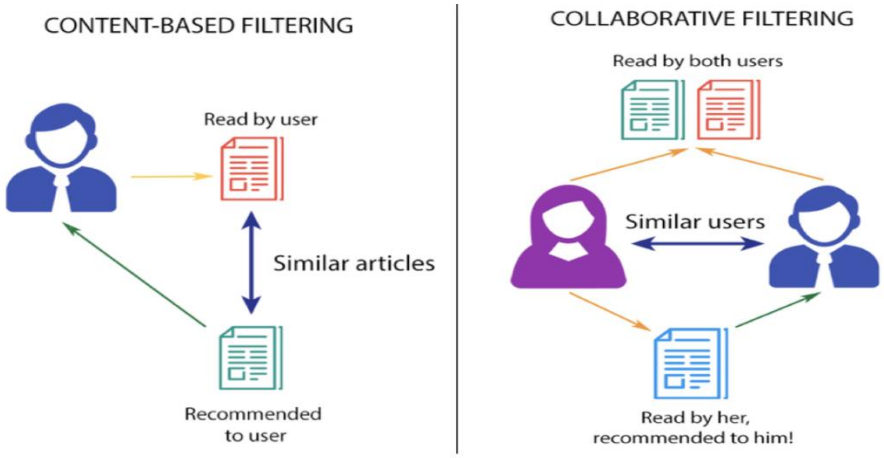
Content based means recommending **content to you based on its similarities with other content** without regarding the preferences of other users.





Searching a book on Amazon website for example “Long Live” and Amazon suggesting book with having title “Long Live”

Collaborative filtering:



Collaborative is the inverse, it does not concern itself with non-abstract features of an item, but rather **uses the wisdom of the crowd to generate recommendations.**

Collaborative filtering discovers patterns in user-item preferences and leverages these patterns to generate recommendations.

Within the collaborative filtering ecosystem, there are number of options:

Machine Learning theoretical concepts

By: Vikram Pal

Singular value decomposition (SVD): is one of the first such techniques, which (as the name suggests) decomposes the user-item preference matrix into three elements: **the user-feature eigenvectors, the feature-item eigenvectors, and a diagonal matrix of eigenvalues.**

SVD → (user, item) --- > (user,feature) (feature, items) and a diagonal matrix of eigenvalues.

Gradient descent-based matrix factorization: The core idea here is to create **parameters and iteratively update them to minimize some cost function**.

1 Cost Function and Gradient

1.1 Mean Squared Error

$$MSE(U_1I_1) = (U_1I_1 - [(U_1F_1 * F_1I_1) + (U_1F_2 * F_2I_1)])^2 \quad (1)$$

1.2 Gradient with respect to U_1F_1

$$\frac{\partial U_1I_1}{\partial U_1F_1} = -2 * (U_1I_1 - [(U_1F_1 * F_1I_1) + (U_1F_2 * F_2I_1)]) * (F_1I_1) \quad (2)$$

1. User-user based (Mean user rating)
2. Item-item based (Mean Item rating)

User-User collaborative filtering

User/Movie	x1	x2	x3	x4	x5	Mean User Rating
A	4	1	–	4	–	3
B	–	4	–	2	3	3
C	–	1	–	4	4	3

$r_{AC} = [(1-3)*(1-3) + (4-3)*(4-3)]/[((1-3)^2 + (4-3)^2)^{1/2} * ((1-3)^2 + (4-3)^2)^{1/2}] = 1$

$r_{BC} = [(4-3)*(1-3) + (2-3)*(4-3) + (3-3)*(4-3)]/[((4-3)^2 + (2-3)^2 + (3-3)^2)^{1/2} * ((1-3)^2+(4-3)^2 + (4-3)^2)^{1/2}] = -0.866$

Machine Learning theoretical concepts

By: Vikram Pal

Item-Item collaborative filtering

User/Movie	x1	x2	x3	x4	x5
A	4	1	2	4	4
B	2	4	4	2	1
C	–	1	–	3	4
Mean Item Rating	3	2	3	3	3

$$C_{14} = [(4-3)*(4-3) + (2-3)*(2-3)] / [((4-3)^2 + (2-3)^2)^{1/2} * ((4-3)^2 + (2-3)^2)^{1/2}] = 1$$

$$C_{15} = [(4-3)*(4-3) + (2-3)*(1-3)] / [((4-3)^2 + (2-3)^2)^{1/2} * ((4-3)^2 + (1-3)^2)^{1/2}] = 0.94$$

Challenges Faced by Recommendation Systems:

Any predictive model or recommendation systems with no exception rely heavily on data. They make reliable recommendations based on the facts that they have. It's only natural that the finest recommender systems come from organizations with large volumes of data, such as Google, Amazon, Netflix, or Spotify. To detect commonalities and suggest items, good recommender systems evaluate item data and client behavioral data. Machine learning thrives on data; the more data the system has, the better the results will be.

Data is constantly changing, as are user preferences, and your business is constantly changing. That's a lot of new information. Will your algorithm be able to keep up with the changes? Of course, real-time recommendations based on the most recent data are possible, but they are also more difficult to maintain. Batch processing, on the other hand, is easier to manage but does not reflect recent data changes.

The recommender system should continue to improve as time goes on. Machine learning techniques assist the system in "learning" the patterns, but the system still requires instruction to give appropriate results. You must improve it and ensure that whatever adjustments you make continue to move you closer to your business goal.

Evaluation Metrics for Recommendation System:

https://analyticsindiamag.com/how-to-measure-the-success-of-a-recommendation-system/

Predictive Accuracy Metrics:

Predictive accuracy or rating prediction measures address the subject of how near a recommender's estimated ratings are to genuine user ratings. This sort of measure is widely used for evaluating non-binary ratings.

Machine Learning theoretical concepts

By: Vikram Pal

Classification Accuracy Metrics:

Classification accuracy measures attempt to evaluate a recommendation algorithm's successful decision-making capacity (SDMC). They are useful for user tasks such as identifying nice products since they assess the number of right and wrong classifications as relevant or irrelevant things generated by the recommender system.

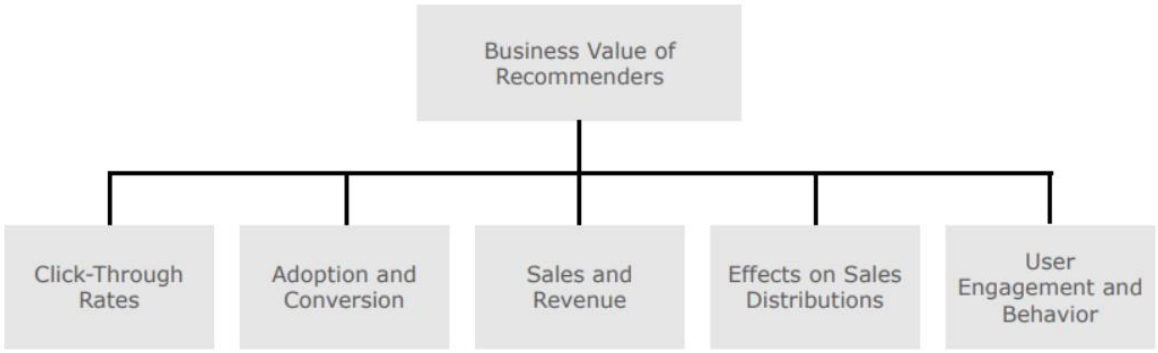
The exact rating or ranking of objects is ignored by SDMC measures, which simply quantify correct or erroneous classification. This type of measure is particularly well suited to e-commerce systems that attempt to persuade users to take certain actions, such as purchasing products or services.

Rank Accuracy Metrics:

In statistics, a rank accuracy or ranking prediction metric assesses a recommender's ability to estimate the correct order of items based on the user's preferences, which is known as rank correlation measurement. As a result, if the user is given a long, sorted list of goods that are recommended to him, this type of measure is most appropriate.

The relative ordering of preference values is used in a rank prediction metric, which is independent of the exact values assessed by a recommender. A recommender that consistently overestimates item ratings to be lower than genuine user preferences, for example, might still get a perfect score as long as the ranking is correct.

Business Value of Recommender:



Click-Through Rates

The click-through rate (CTR) is a metric that measures how many people click on the recommendations. The basic notion is that if more people click on the recommended things, the recommendations are more relevant to them.

For Example: In news recommendations, the CTR is a widely used metric.

Adoption and Conversion

Click-through rates are often not the final success measure to pursue in recommendation scenarios, unlike online business models dependent on adverts. While the CTR can measure user attention or interest, it can't tell you whether users liked the recommended news article they clicked on or if they bought something based on a recommendation.

Machine Learning theoretical concepts

By: Vikram Pal

For Example: YouTube employs the idea of "long CTRs," in which a user's clicks on suggestions are only tallied if they view a particular percentage of a video. Similarly, Netflix utilizes a metric called "take-rate" to determine how many times a video or movie was watched after being recommended.

Sales and Revenue:
In many cases, the adoption and conversion measures outlined in the previous section are more telling of a recommender's prospective business value than CTR measures alone. When customers choose more than one item from a list of suggestions, this is a good indicator that a new algorithm was successful in identifying later purchases or views. stuff that the user is interested in.

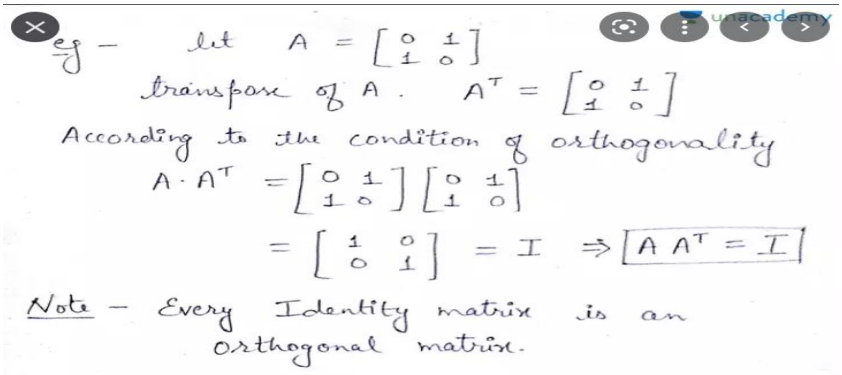
Measurement	Remarks
Click-Through Rates	Easy to measure and established, but often not the ultimate goal.
Adoption and Conversion	Easy to measure, but often requires a domain- and application specific definition. Requires interpretation and does not always translate directly into business value.
Sales and Revenue	Most informative measure, but cannot always be determined directly.
Effects on Sales Distribution	A very direct measurement; requires a thorough understanding of the effects of the shifts in sales distributions.
User Engagement and Behavior	Often, a correspondence between user engagement and customer retention is assumed; still, it remains an approximation.

Apart from all of this, we can also leverage our standard ML evaluation metrics to evaluate ratings and predictions that are as follows.

- Precision
- Recall
- F1-measure
- False-positive rate
- Mean average precision
- Mean absolute error
- The area under the ROC curve (AUC)

Matrix Operations:

Orthogonal Matrix:



Machine Learning theoretical concepts

By: Vikram Pal

Orthonormal Matrix:

$$Q^TQ = \begin{bmatrix} q_1^T \\ q_2^T \\ q_3^T \\ \vdots \\ q_n^T \end{bmatrix} [q_1\ q_2\ q_3\ \cdots\ q_n] = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix} = I$$

Diagonal Matrix:

Diagonal Matrices

A square matrix is called a diagonal matrix if nondiagonal entries are all zero. The main diagonal can be constants or zeros. A diagonal matrix must fit the following form:

$$D = \begin{bmatrix} d_{11} & 0 & 0 & \cdots & 0 \\ 0 & d_{22} & 0 & \cdots & 0 \\ 0 & 0 & d_{33} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & d_{nn} \end{bmatrix}$$

- *If any of the diagonals of a diagonal matrix are zero, the matrix is singular, meaning it is not invertible or does not have an inverse matrix.*

Determinant of a matrix:

Finding the Determinant of a Three-By-Three Matrix

$$A = \begin{bmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{bmatrix}$$

$$\det(A) = a_1 \begin{vmatrix} b_2 & b_3 \\ c_2 & c_3 \end{vmatrix} - a_2 \begin{vmatrix} b_1 & b_3 \\ c_1 & c_3 \end{vmatrix} + a_3 \begin{vmatrix} b_1 & b_2 \\ c_1 & c_2 \end{vmatrix}$$

$$= a_1(b_2c_3 - b_3c_2) - a_2(b_1c_3 - b_3c_1) + a_3(b_1c_2 - b_2c_1)$$

Egin Values of a matrix:

$$\det(A - \lambda I) = 0$$

$$(A - \lambda I)x = 0$$

$$A = \begin{bmatrix} 1 & 4 \\ 3 & 2 \end{bmatrix}$$

$$\det \begin{bmatrix} a & b \\ c & d \end{bmatrix} = ad - bc$$

$$\det(A - \lambda I) = 0$$

$$\det\left(\begin{bmatrix} 1 & 4 \\ 3 & 2 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right) = 0$$

$$\det\left(\begin{bmatrix} 1-\lambda & 4 \\ 3 & 2-\lambda \end{bmatrix}\right) = 0$$

$$(1-\lambda)(2-\lambda) - 12 = 0$$

$$\lambda^2 - 3\lambda - 10 = 0$$

$$(\lambda - 5)(\lambda + 2) = 0$$

$$\lambda = 5, -2$$

Lambda is eginvalues.

Egin Vectors:

Machine Learning theoretical concepts

By: Vikram Pal

By plugging in the discovered eigenvalues into our originally derived equation, we can find the eigenvectors.

$$\lambda = 5, -2$$

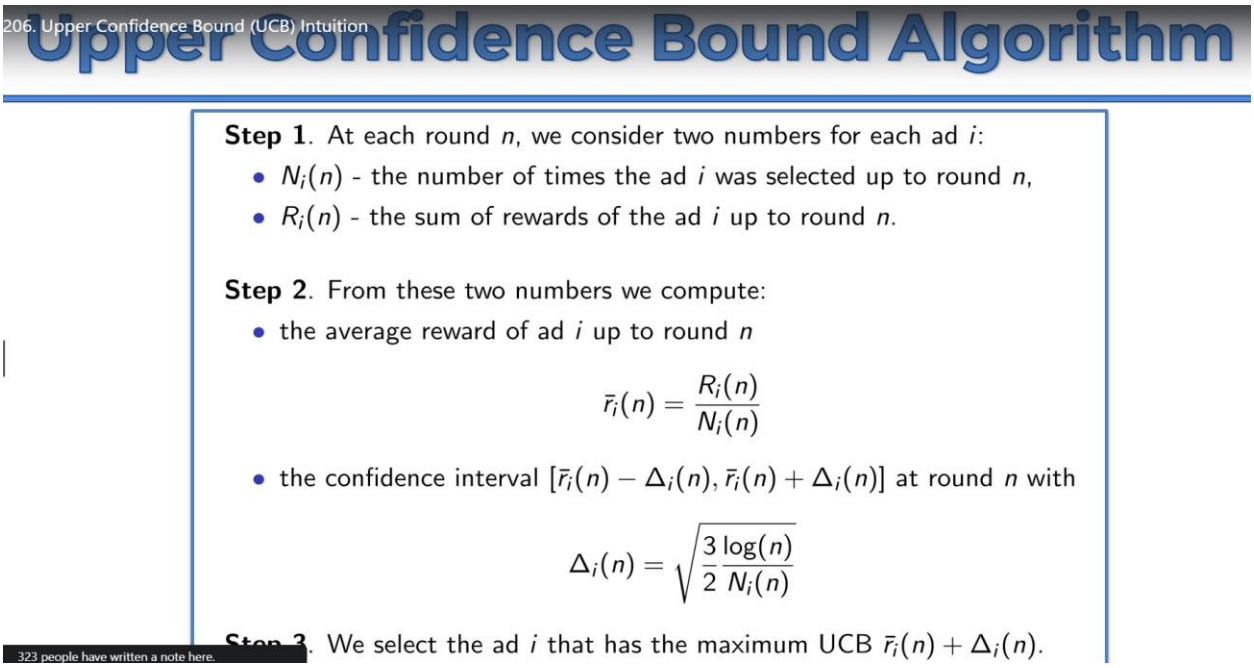
$$A = \begin{bmatrix} 1 & 4 \\ 3 & 2 \end{bmatrix} \qquad \begin{bmatrix} 1-\lambda & 4 \\ 3 & 2-\lambda \end{bmatrix} x = 0 \qquad \begin{bmatrix} 1-\lambda & 4 \\ 3 & 2-\lambda \end{bmatrix} x = 0$$

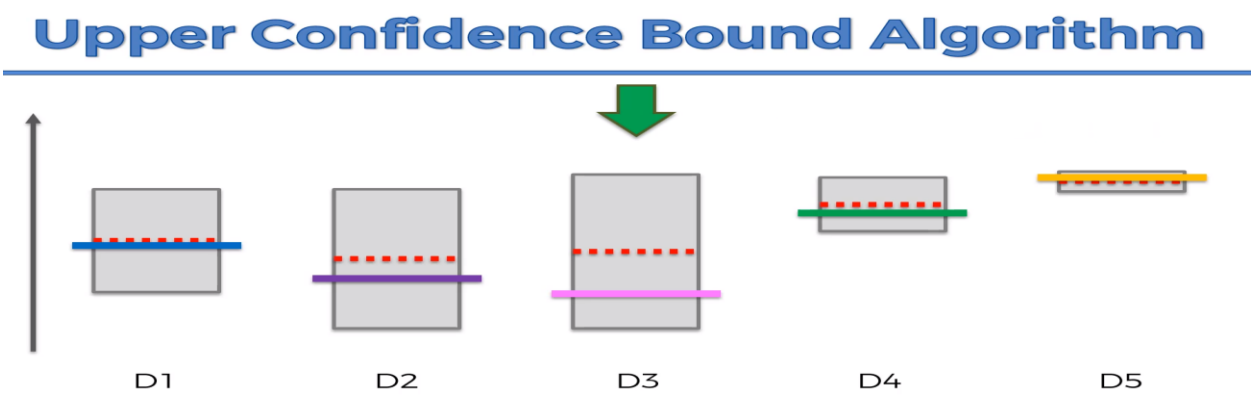
$$(A - \lambda I)x = 0 \qquad \begin{bmatrix} -4 & 4 \\ 3 & -3 \end{bmatrix} x = 0 \qquad \begin{bmatrix} 3 & 4 \\ 3 & 4 \end{bmatrix} x = 0$$

$$x = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \qquad x = \begin{bmatrix} -4 \\ 3 \end{bmatrix}$$

Reinforcement Learning:

Upper Confidence Bound:





Machine Learning theoretical concepts

By: Vikram Pal

Thomson Sampling Algorithms:

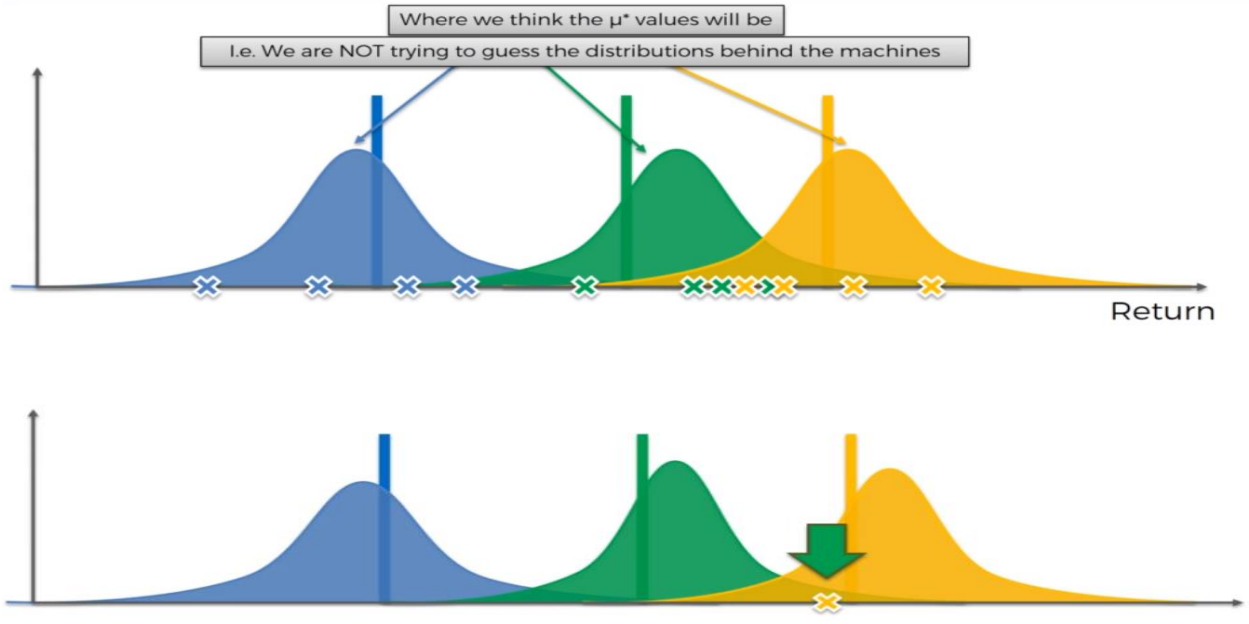
Bayesian Inference

- Ad i gets rewards $\mathbf{y}$ from Bernoulli distribution $p(\mathbf{y}|\theta_i) \sim \mathcal{B}(\theta_i)$.
- θ_i is unknown but we set its uncertainty by assuming it has a uniform distribution $p(\theta_i) \sim \mathcal{U}([0,1])$, which is the prior distribution.
- Bayes Rule: we approach θ_i by the posterior distribution

$$\underbrace{p(\theta_i|\mathbf{y})}_{\text{posterior distribution}} = \frac{p(\mathbf{y}|\theta_i)p(\theta_i)}{\int p(\mathbf{y}|\theta_i)p(\theta_i)d\theta_i} \propto \underbrace{p(\mathbf{y}|\theta_i)}_{\text{likelihood function}} \times \underbrace{p(\theta_i)}_{\text{prior distribution}}$$

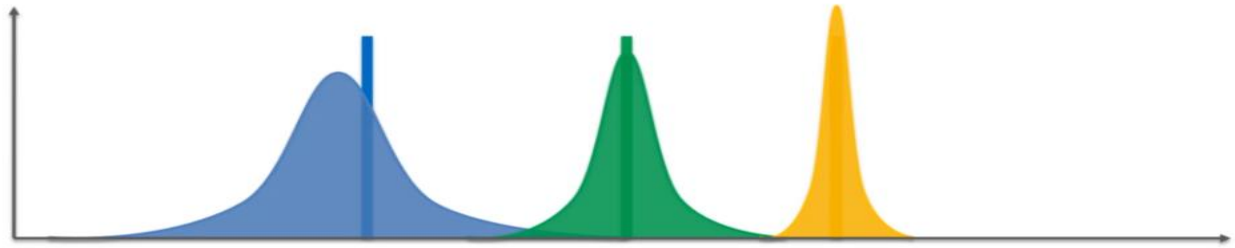
- We get $p(\theta_i|\mathbf{y}) \sim \beta(\text{number of successes} + 1, \text{number of failures} + 1)$
- At each round n we take a random draw $\theta_i(n)$ from this posterior distribution $p(\theta_i|\mathbf{y})$, for each ad i.
- At each round n we select the ad i that has the highest $\theta_i(n)$.

Thompson Sampling Algorithm

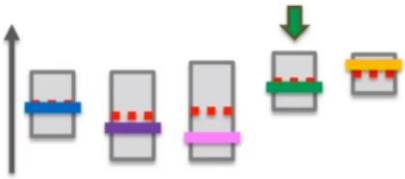


Machine Learning theoretical concepts

By: Vikram Pal



UCB



- Deterministic
- Requires update at every round

Thompson Sampling

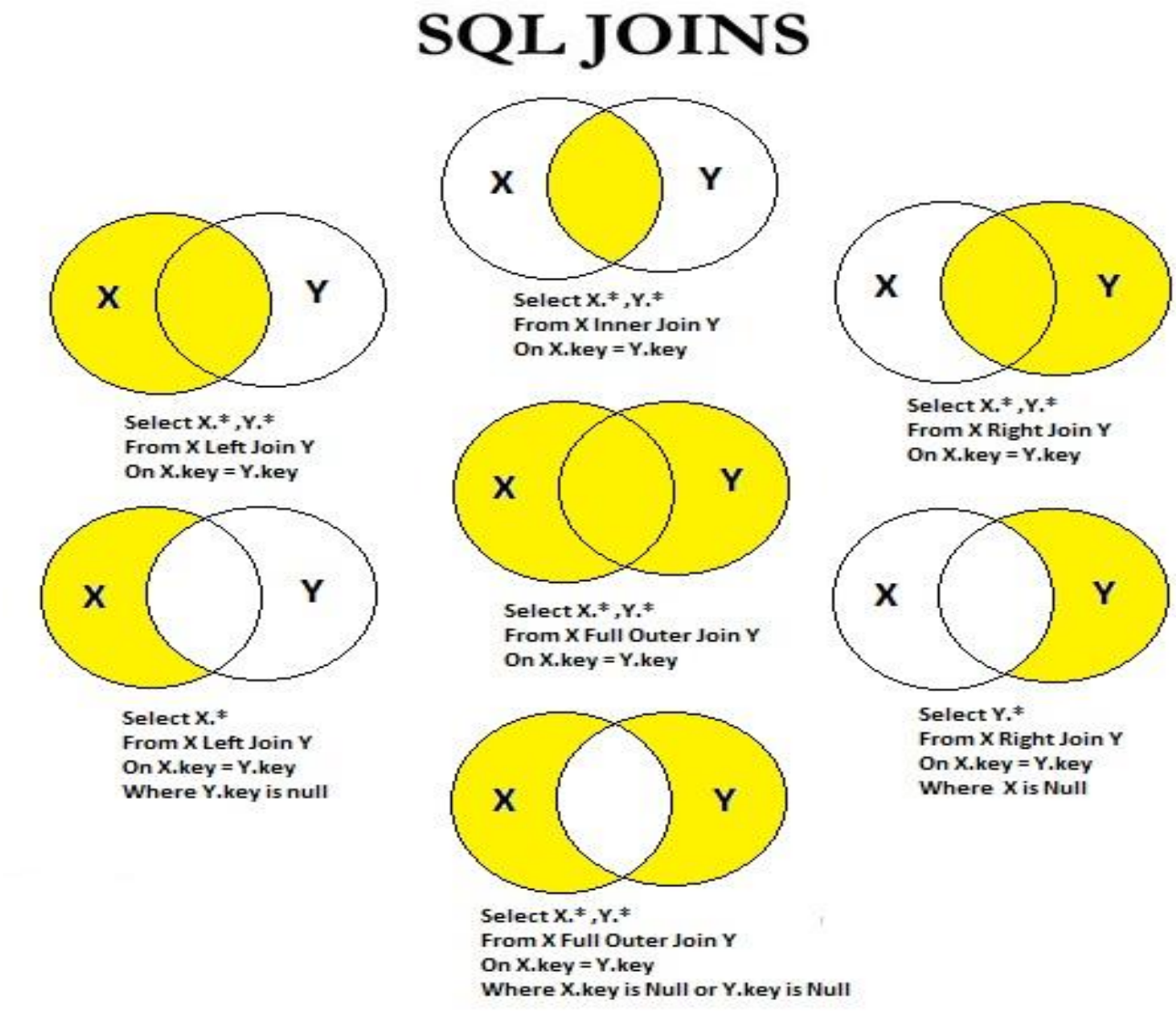
- Probabilistic
- Can accommodate delayed feedback
- Better empirical evidence

Machine Learning theoretical concepts

By: Vikram Pal

SQL & Pandas

Joins:



Pandas Left Only and right only:

```
df2=pd.merge(df_stream, df_transaction,on="customer_id",how="outer",indicator=True)
```

left only:

```
df[df["_merge"]=="left_only"]
```

right only:

```
df[df["_merge"]=="right_only"]
```

Machine Learning theoretical concepts

By: Vikram Pal

Self Join:

```
SELECT A.CustomerName AS CustomerName1,
B.CustomerName AS CustomerName2, A.City
FROM Customers A, Customers B
WHERE A.CustomerID <> B.CustomerID
AND A.City = B.City
ORDER BY A.City;
```

Having:

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country
HAVING COUNT(CustomerID) > 5;
```

Between and Not Between:

```
SELECT * FROM Products
WHERE Price BETWEEN 10 AND 20;
```

```
ANY:
SELECT ProductName
FROM Products
WHERE ProductID = ANY
  (SELECT ProductID
  FROM OrderDetails
  WHERE Quantity = 10);
```

Notes:

Multiple tables with summations:

https://www.codesadda.com/hackerrank/mysql/HackerRank%20MySQL%20-%20Interviews/