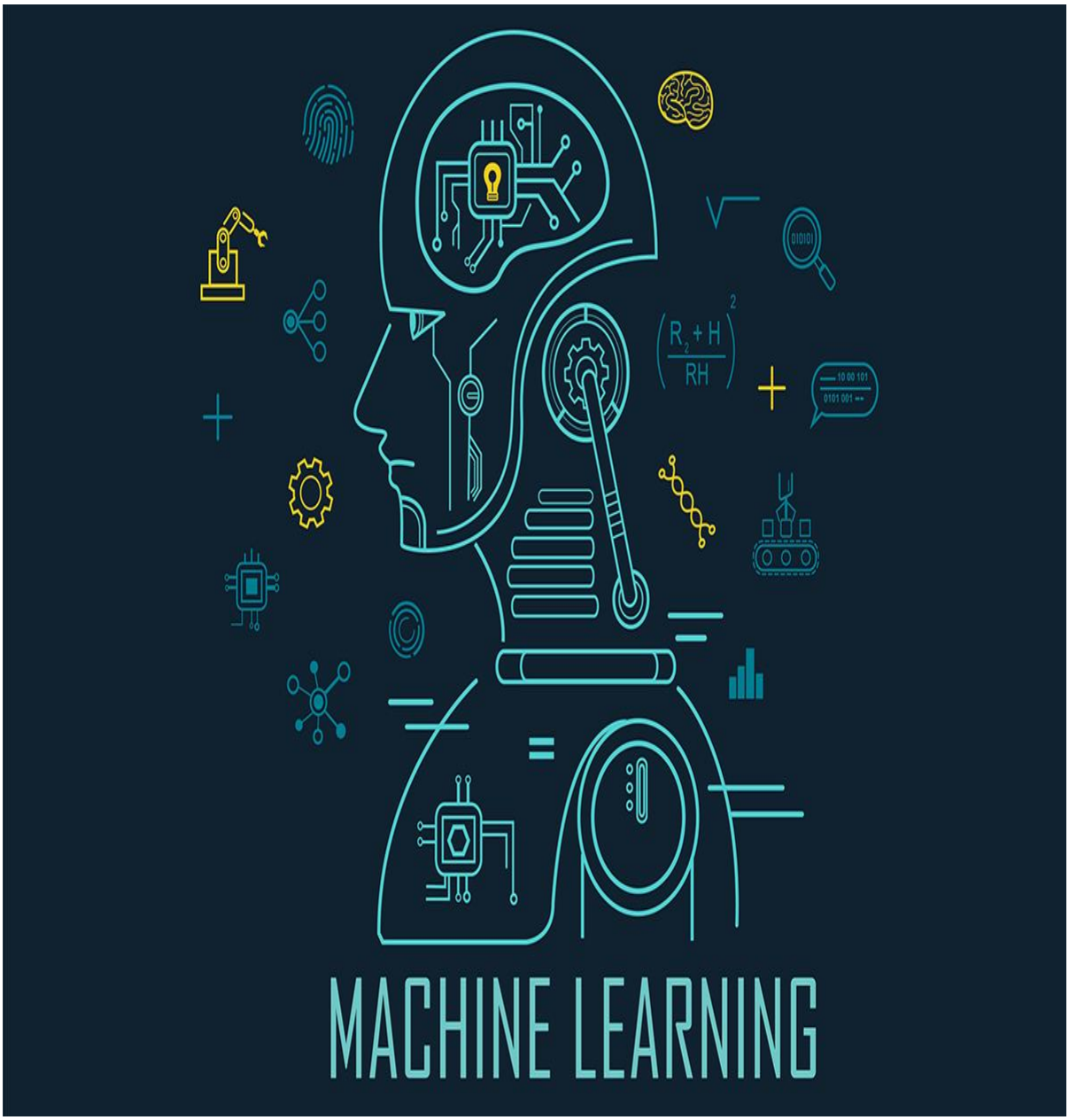


BASIC CONCEPTS COMPILED FROM MY NOTES AND VARIOUS SOURCES AVAILABLE ON INTERNET

- **V V NAGARAJU DUGGIRALA**

TABLE OF CONTENTS

21. Recurrent Neural Networks
    a. RNN and limitations
    b. Differences with CNN
    c. Long Short-Term Memory (LSTM)
    d. Gated Feedback Recurrent Neural Networks (GRU)
22. Restricted Boltzmann Machine (RBM)
23. Deep Belief Networks (DBN)
24. Autoencoders
    a. Types
    b. Applications
    c. Convolutional Autoencoders
    d. Denoising Autoencoders
    e. Deep Autoencoders
    f. Variational Autoencoders (VAE)
    g. Sparse Autoencoders
25. Siamese Neural Networks
26. Generative Adversarial Networks (GAN)
27. Regression Algorithms
    a. Linear Regression
    b. Multiple Linear Regression
    c. Polynomial Regression
    d. Ridge Regression
    e. Lasso Regression
    f. ElasticNet Regression
28. Unsupervised Learning
29. Unsupervised Learning Algorithms
    a. Clustering
    b. K-Means Clustering
    c. Hierarchical Clustering
    d. DBSCAN
    e. Gaussian Mixtures
    f. Spectral Clustering
30. Dimensionality Reduction
    a. Feature Engineering
    b. Multicollinearity
    c. Factor Analysis
    d. Principal Component Analysis (PCA)
    e. Linear Discriminant Analysis (LDA)
    f. Isometric Mapping (IsoMap)
    g. Locally Linear Embedding (LLE)
    h. t-distribution Stochastic Neighborhood Embedding (t-SNE)
31. Semi-supervised Learning
32. Semi-supervised learning algorithms:
    a. Self Training
    b. Generative Models
    c. S3VMs
    d. Graph Based Algorithms
    e. Multiview algorithms

33. Reinforcement Learning
    a. Introduction
    b. Approaches
    c. Markov Decision Process
    d. Q-learning
    e. Application of RL
34. Other topics
    a. Word2Vec
    b. Generalization
    c. Overfitting
    d. Regularization
    e. Bias
    f. Variance
    g. Occam's Razor
    h. Bag of Words (BoW)
    i. Recommender Systems
    j. Ensemble Methods
    k. Natural Language Processing (NLP)

MACHINE LEARNING - INTRODUCTION

DEFINITION:

Machine Learning is a set of methods that can automatically detect patterns in data and then use the uncovered patterns to predict future data or to perform other kinds of decision making under uncertainty (such as planning how to collect more data).

LEARNING PROBLEM:

A computer program is said to **learn** from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

Example: a computer program that learns to play checkers

P in this case is *measured by its ability to win*

T in this case is *playing checkers games*

E in this case is *obtained by playing games against itself*

In general, to have a well-defined learning problem we must identify these three features:

1. The class of tasks (T)
2. The measure of performance to be improved (P)
3. The source of experience (E)

Checkers learning problem:

- Task T – playing checkers
- Performance measure P – percent of games won against opponents
- Training experience E – playing practice games against itself

Handwriting recognition learning problem:

- Task T – recognizing and classifying handwritten words within images
- Performance measure P – percentage of words correctly classified
- Training Experience E – a database of handwritten words with given classifications

Robot driving learning problem:

- Task T – driving on public four lane highways using vision sensors
- Performance measure P – average distance travelled before an error (as judged by human overseer)
- Training experience E – a sequence of images and steering commands recorded while observing a human driver

E-mail classification:

- Task T: Categorize email messages as spam or legitimate.
- Performance P: Percentage of email messages correctly classified.
- Training Experience E: Database of emails, some with human-given labels

MACHINE LEARNING FRAMEWORK:

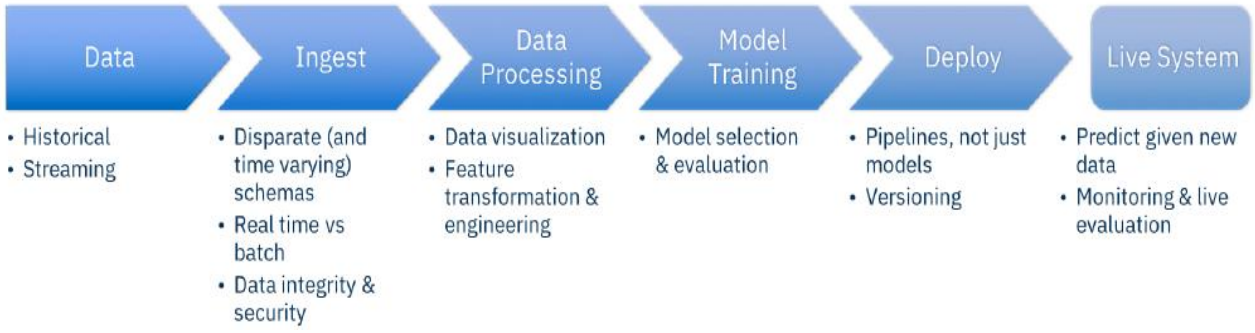
- **y = f(x)**
  - y = output, x = feature representation, f( ) is prediction function
- **Training:** Given a training set, estimate the prediction function f( ) by minimizing the prediction error.
- **Testing:** Apply f ( ) to unknown test sample x and predicted value (output) is y.
- **y = f(w, x)** (for linear model)
  - y = output, x = feature representation, w = weight, f(w,) = prediction function
- **Training:** Given a training set, estimate the prediction function, f( ) by minimizing the prediction error.
- **Testing:** Apply f(w,) to unknown test sample x and predicted value (output) is y
  - Parameter: primary problem is to find the parameters W.

AI/ML Problem – phases:

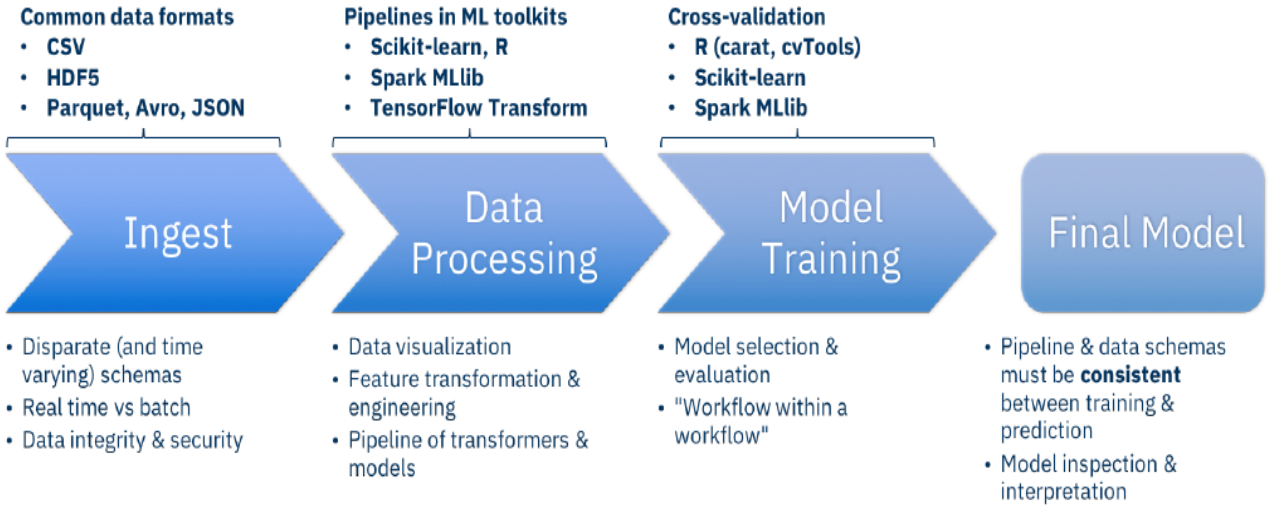
The following are the various steps involved in problem solving using AI/ML:

- Define your task
- Collect Data
- Preprocessing of Data
- Dimensionality Reduction/Feature Selection
- Choose ML Algorithm
- Experimental Design
- Test and Validate
- Run System

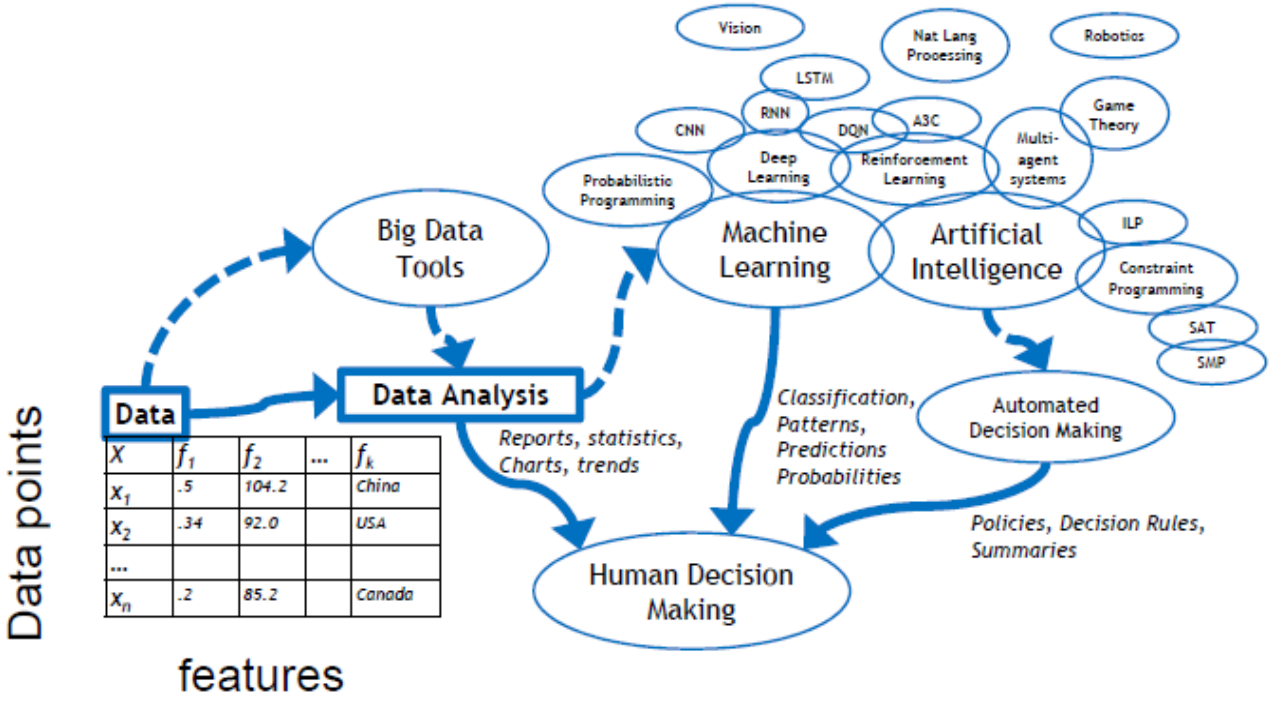
MACHINE LEARNING WORK FLOW:



MACHINE LEARNING TOOLS:



Landscape of Big Data/AI/ML:



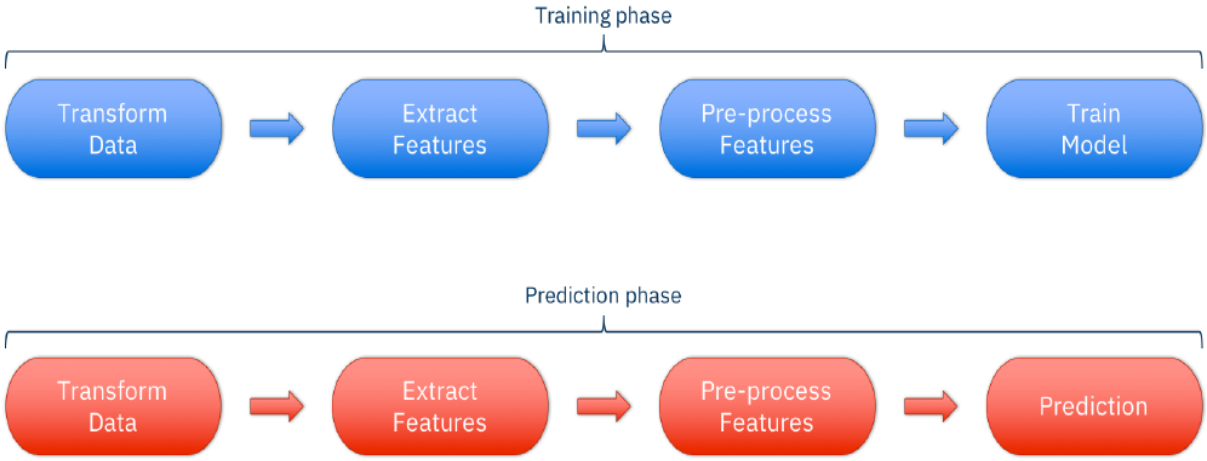
Dataset and subsets:

Training Set:

- Training set is used for learning the parameters of the model.
- Training set has an optimistic bias.
- The training set is usually the biggest one of three sets and used to build the model.

Test Set:

- Test set is used to get a final, unbiased estimate of how well the learning method works.
- We expect that this estimate to be worse than training/validation set.
- Test set is not biased and this distinguishes test set from training set.
- Similar to Training set, Test set is a finite sample and bound to have variance due to sample size, but the test set doesn't have an optimistic or pessimistic bias.
- The test set doesn't effect the outcome of learning process which only uses training set.

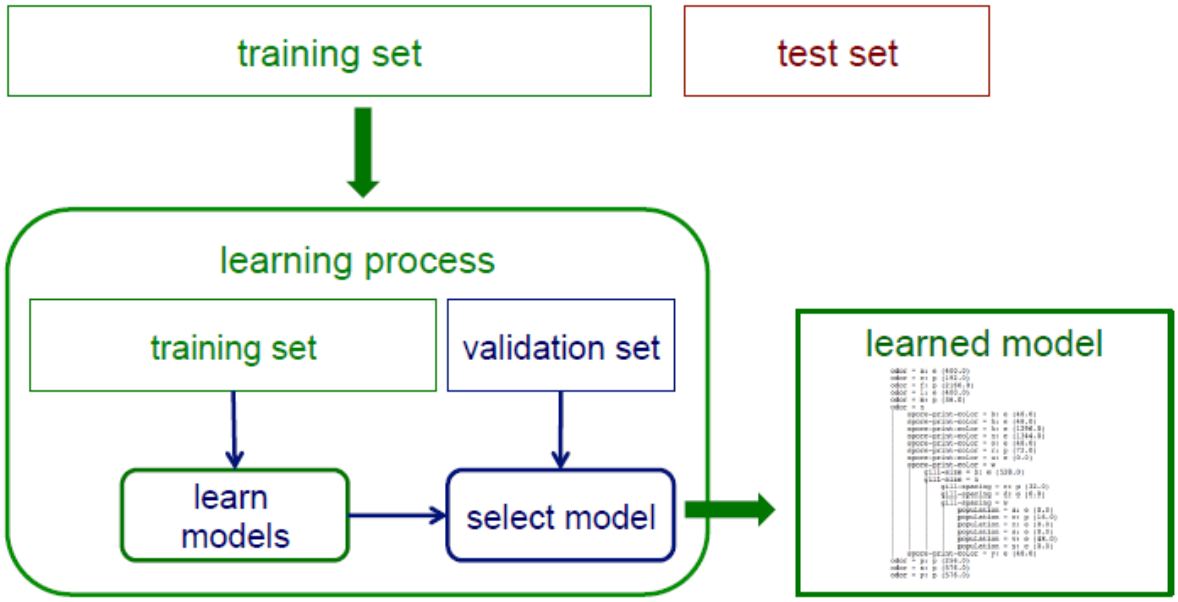


Validation Set:

- Validation set is not used for learning but is for avoiding overfitting.
- The idea of 'Validation Set' is identical to 'Test Set' but there is a difference.
- We remove a subset from data and this subset is not used for training.
- Although validation set is not directly used for training, it will be used in making certain choices in the learning process. As the set affects the learning process, it is no longer a test set.
- The more we use validation set to fine tune the model, the more validation set becomes like a training set.
- The validation and test sets are roughly the same sizes, much smaller than the size of the training set. The learning algorithm cannot use examples from these two subsets to build the model.
- Validation set is used for selecting the model complexity.

In the past, the rule of thumb was to use 70% of the dataset for training, 15% for validation and 15% for testing. However, in the age of big data, datasets often have millions of examples. In such cases, it could be reasonable to keep 95% for training and 2.5%/2.5% for validation/testing.

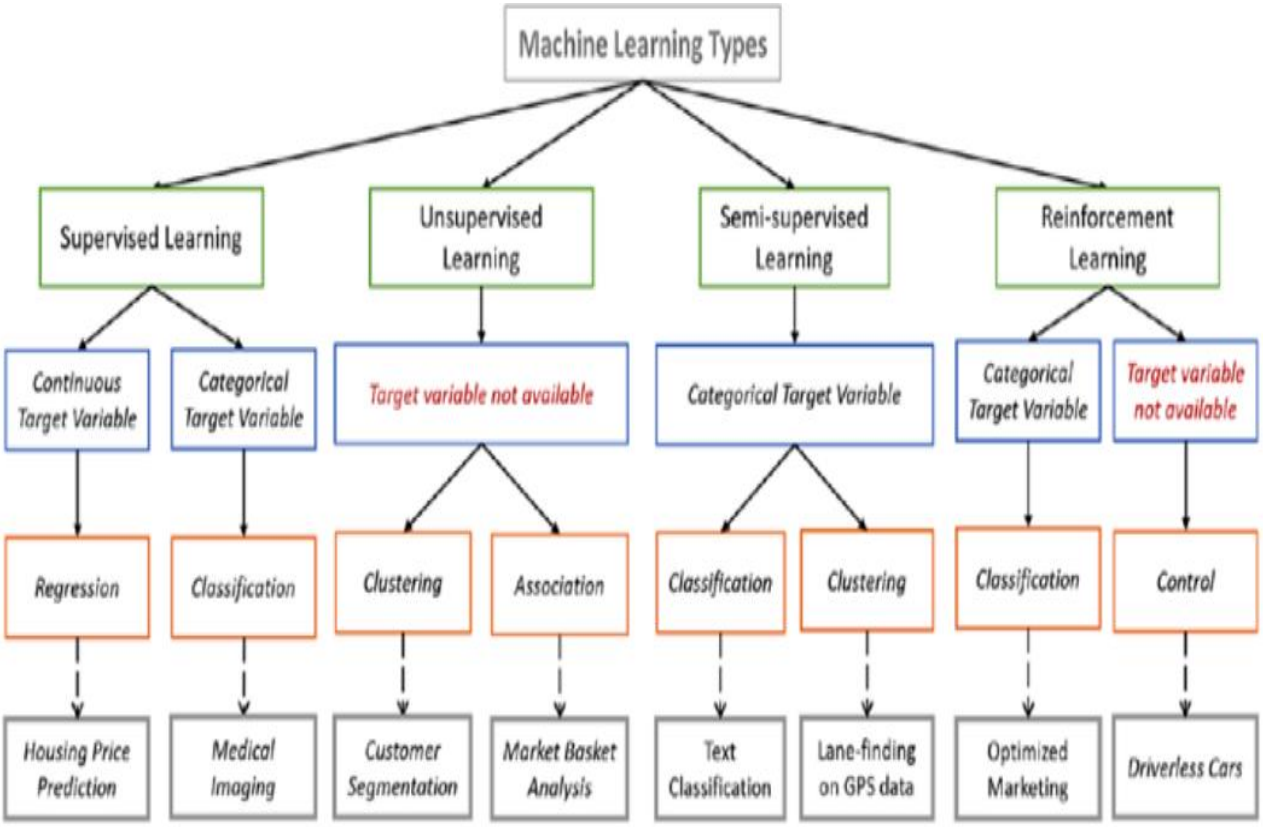
Details of Training set, test set, validation test is shown below:



TYPES OF MACHINE LEARNING

The following are the types of Machine Learning:

- Supervised Learning
- Unsupervised Learning
- Semi-supervised Learning
- Reinforcement Learning



SUPERVISED LEARNING

- Also known as 'Predictive Learning'.
- is provided with training data along with the expected output or rules to categorize this data also known as labels.
- Uses the set of inputs and outputs to predict the output for future unseen inputs
- A type of machine learning used to train models from labeled training data
- Allows to predict output for future or unseen data
- Used in weather forecasting, e-mail filters, Netflix recommendations
- Training data consists of inputs also known as features; and outputs known as labels. For example, for an apple, color, size and weight are features and Red, Medium and 50 gm are labels

Steps in Supervised learning:

1. Train the model – for example provide images of apples along with expected response to the model. Train the algorithm (reverse engineer the relationship between x and y), generated learned algorithm (y = f(x)) and predict for the new instance using y = f(x)
2. Test the model – in this step the model learns from labeled data and generates output of the model based on the images to the model without expected output.

Types of Supervised learning:

- Classification
- Regression

Classification:

- Classification is a problem of automatically assigning a label to an unlabeled example. Spam detection is a famous example of classification.
- In Machine learning, the classification problem is solved by a classification learning algorithm that takes a collection of labeled examples as inputs and produces a model that can take an unlabeled example as input and either directly output a label or output a number that can be used by the analyst to deduce the label.
- the output has predefined labels that have discrete values and the goal is to predict discrete values that belong to a class and evaluate on the basis of accuracy. If the target variable is categorical (classes), then use classification algorithm. It is applied when the output has finite and discrete values.

Regression:

- Regression is just like classification except the response variable is continuous.
- Regression is a problem of predicting a real valued label (often called a target) given an unlabeled example. Estimating house price valuation based on house features, such as area, the number of bedrooms, location and so on is a famous example of regression.
- The regression problem is solved by a regression learning algorithm that takes a collection of labeled examples as inputs and produces a model that can take an unlabeled example as input and output a target.

- the output has a continuous value that is not restricted to defined separate values. Regression predicts a value as close to the actual output value as possible and evaluates it by calculating error value. The error value is inversely proportional to accuracy of model. Regression involves finding the best fit straight line through the plotted points.
- If the target variable is a continuous numeric variable (100-2000) then use regression.

Examples of real-world regression problems:

- Predict tomorrow's stock market price given current market conditions and other possible side information.
- Predict the age of a viewer watching a given video on YouTube.
- Predict the location in 3D space of a robot arm end effector, given control signals (torques) sent to its various motors.
- Predict the amount of prostate specific antigen (PSA) in the body as a function of a number of different clinical measurements.
- Predict the temperature at any location inside a building using weather data, time, door sensors etc.,

Types of Classification Algorithms:

- Logistic Regression
- K-Nearest Neighbors (KNN)
- Decision Trees
- Random Forest
- Naïve Bayes Classifier (NBC)
- Support Vector Machines (SVM)
- Neural Networks

LOGISTIC REGRESSION:

- Logistic Regression is not a regression, but a classification learning algorithm. This is adaptation of linear regression to problems of classification (i.e., yes/no questions, groups, etc.)
- Used to estimate discrete values (binary values like 0,1; yes/no; true/false) based on a given set of independent variables.
- These models are used mostly as a data analysis and inference tool where the goal is to understand the role of the input variables in explaining the outcome.
- We compute parameter estimates as well as their standard errors with logistic regression.

Use cases:

- Loan sanction
- Customer segmentation
- Spam filter
- Exam Result prediction

Applications:

- Weather Forecast
    - The binary dependent variables of sun, cloud, storm and rain are regressed against independent variables that define weather properties.
    - The data is used to conclude whether the weather will be sunny/stormy/cloudy/ rainy.
- Cancer Prediction
    - The outcome will either be malignant or benign.

Advantages:

- Easy to understand.

Disadvantages:

- Sometimes too simple to capture complex relationships between variables.
- Works poorly with correlated features.

Sigmoid Probability:

- The probability in the logistic regression is represented by the Sigmoid function (logistic function or the S-curve).
- The sigmoid function gives an 'S' shaped curve.
- This curve has a finite limit that is Y can only be 0 or 1.
- The probability distribution of output y is restricted to 1 or 0. This is called sigmoid probability.

K-NEAREST NEIGHBORS (KNN) ALGORITHM:

- Is a non-parametric method used for classification and regression.
- The input consists of the k closest training examples in the feature space.
- Output depends on whether k-NN is used for classification or regression.
- No training and learning involved in this algorithm
- Advantage of KNN is easy to generalize multi class classification.
- A simple but powerful approach for making predictions.
- Uses the most similar historical examples to the new data.
- In the absence of prior knowledge, most k-NN classifiers use simple Euclidean distances to measure the dissimilarities between examples represented as vector inputs.
- Euclidean distance metrics, however, do not capitalize on any statistical regularity in the data that might be estimated from a large training set of labeled examples.

Steps in KNN:

1. For each unknown sample find the distance from all labelled samples.
2. Sort these labels by their distance from the unknown sample.
3. Select the 'k' labelled sample nearest to the unknown sample.
4. Identify the majority label.
5. Assign the majority label to the unknown sample.

Advantages of KNN:

- Simple to implement.
- Works well in practice provided it is given a good distance metric and has enough labeled training data.
- Does not require to build a model, make assumptions and tune parameters.
- Can be extended easily with new examples.

Disadvantages of KNN:

- Requires large space to store the entire training data set.
- Slow. Given n examples and d features, the method takes O(n x d) to run.
- Do not work well with high dimensional data and suffers from curse of dimensionality.

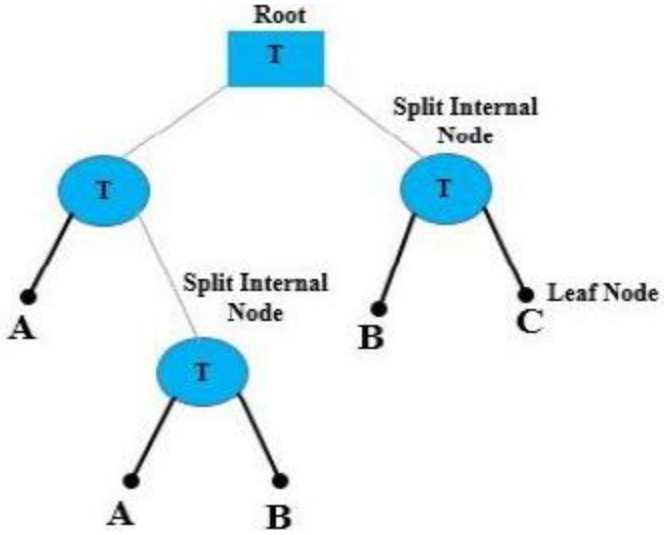
Applications of KNN:

- Information retrieval.
- Handwritten character classification using nearest neighbor in large databases.
- Recommender systems
- Breast cancer diagnosis
- Medical data mining.
- Pattern recognition

DECISION TREE ALGORITHM:

- Is one of the predictive modelling approaches used in statistics, data mining and machine learning.
- Uses a decision tree as a predictive model and maps observations about an item to conclusions about its target value.
- When the target variable can take a finite set of values, the tree models are called classification trees. In these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels.
- Decision trees where the target variable can take continuous values (typically real numbers) are called regression trees.
- Makes sequential, hierarchical decisions about the outcome variable based on the predictor data
- Is a graphical representation of all possible solutions to a decision
- A series of yes/no rules based on the features, forming a tree, to match all possible outcomes of a decision.
- Use predictive models to achieve results
- Is drawn upside down with its root at the top
- Splits into branches based on a condition or internal node
- Doesn't split the end of the branch if it is the decision/leaf
- Predicts continuous values like price of a house
- Referred to as CART (Classification and Regression Tree)
- Represents a single input variable (x) and a split point on that variable
- Goal is to create a model that predicts the value of a target variable based on several input variables.
- Each internal (non-leaf) node is labeled with an input feature.

- The arcs coming from a node labeled with an input feature are labeled with each of the possible values of the target or output feature or the arc leads to a subordinate decision node on a different input feature.
- Each leaf of the tree is labeled with a class or a probability distribution over the classes, signifying that the data set has been classified by the tree into either a specific class or into a probability distribution.
- A tree is built by splitting the source set, constituting the root node of the tree into subsets which constitute the successor children.
- The splitting is based on a set of splitting rules based on classification features.
- Makes sequential, hierarchical decisions about the outcome variable based on the predictor data
- A decision tree is a tree whose internal nodes are tests on input patterns and whose leaf nodes are categories of patterns.
- A decision tree assigns a class number or output to an input pattern by filtering the pattern down through the tests in the tree.
- Each test has mutually exclusive and exhaustive outcomes.
- Are defined by recursively partitioning the input space and defining a local model in each resulting region of input space which can be represented by a tree with one leaf per region.
- A flowchart shaped like a tree where every internal node denotes a check on an attribute
- Each branch represents the outcome of the test and each leaf node represents the class label
- A path from root to leaf represents classification rules
- Follow a natural if-then-else construction
- Are best for categorization problem where attributes are systematically checked
- The goal is to make the optimal choice at the end of each node
- Has applications in medical diagnosis and credit risk analysis
- Decision Tree (a.k.a CART) network will be as shown below:



Terms used in Decision Tree:

- Root Node – the entire population or sample that further gets divided
- Splitting – Division of nodes into two or more sub nodes
- Decision Node – A sub node splits into further sub nodes
- Leaf/Terminal Node – Node that doesn't split
- Pruning – process of removing sub nodes
- Branch/Sub tree – A subsection of the entire tree
- Parent Node – A node which is divided into sub nodes and the sub nodes are the child node
- Makes the decision to split a node based on purity
- Need to reach maximum purity to optimize the model

Steps in Decision Tree:

1. Place the best attribute of the dataset at the root of the tree.
2. Split the training set into subsets.
3. Repeat step 1 and step 2 on each subset until you find leaf nodes in all the branches of the tree.

Decision Tree Formation:

Decision Tree formation depends on two things:

1. Entropy
    a. measures the impurity of a collection of examples
    b. depends on the distribution of the random variable
    c. Measures the amount of information in a random variable
2. Information Gain
    a. Expected reduction in entropy caused by partitioning the examples on an attribute
    b. Higher the information gain, the more effective the attribute in classifying training data

The attribute with the highest information gain is selected as the splitting attribute

Branch with Entropy 0 is a leaf node and branch with entropy greater than 0 needs splitting

The algorithm iterates on the non-leaf branches until all nodes become the leaf node

Advantages of Decision Trees:

- Simple to understand, implement and interpret.
- Able to handle both numerical and categorical data.
- Requires little data preparation.
- Uses white box model.
- Possible to validate a model using statistical tests.
- Scale well with large datasets.
- Mirrors human decision making more closely than other approaches.
- Are insensitive to monotone transformations of the inputs
- Perform automatic variable selection.
- Relatively robust to outliers
- Can modified to handle missing inputs.
- Fast, compact and effective

- Handles categorical variables
- Interpretable as a set of rules
- Can indicate the most useful features
- Empirically valid in many commercial applications
- Handles noisy data

Limitations of Decision Trees:

- Can be very non-robust and unstable. A small change in the training data can result in a large change in the tree due to the hierarchical nature of the tree-growing process causing errors at the top to affect the rest of the tree and consequently the final predictions.
- Decision tree learners can create over complex trees that do not generalize well from the training data, known as 'overfitting'.
- Do not predict very accurately compared to other kinds of model, partly due to the greedy nature of the tree construction algorithm.
- Not suitable for prediction of continuous attribute
- Does not handle non-rectangular regions well
- Computationally expensive to train
- Tends to overfit
- Large decision trees may be hard to understand
- Requires fixed length feature vectors
- Not often used on its own for prediction because it's also often too simple and not powerful enough for complex data.

OVERFITTING in Decision Trees:

- Occurs when the learning algorithm continues to develop hypotheses that reduce training set error at the cost of an increased test set error
- The decision trees will act very well on the training data at the expense of accuracy with the entire distribution
- The sparsity and large margin principles are necessary to prevent overfitting, i.e., to ensure that we do not use all the basis functions.
- It is not always desirable to construct trees that perfectly model the training data due to overfitting.

To avoid overfitting in Decision Trees there are two ways:

1. Allow the tree to grow until it overfits and then post-prune the tree
2. Prevent the tree from growing too deep before it reaches the point where it perfectly classifies the training data

Early Stopping and Pruning:

- Decision Trees are notorious for overfitting
- Early stopping – Do not split beyond a point
- Pruning – Once the full tree is formed remove weakest branches and use validation set to decide when to stop.
- Pruning can be done using a scheme that prunes the branches giving the least increase in the error.
- Pruning of the decision tree is done by replacing a whole subtree by a leaf node

- To determine how far to prune back, we can evaluate the cross-validated error on each such subtree and then pick the tree whose cross validation (CV) error is within 1 standard error of the minimum.
- Both approaches also reduce the depth and improve classification speed
- MAP estimation is a successful way to reduce overfitting
- **min_impurity_split** is the classifier used for early stopping and this is threshold for early stopping in tree growth
- **min_samples_leaf** is the classifier used for pruning. This represents the minimum number of samples required to be at leaf node and this parameter helps limit the growth of the tree

RANDOM FOREST:

- A commonly used class of ensemble methods.
- Is an ensemble of decision trees machine learning classification algorithm
- Gives better prediction and accuracy than a decision tree
- This technique tries to decorrelate the base learners by learning trees based on a randomly chosen subset of input variables as well as a randomly chosen subset of data cases
- Often have very good predictive accuracy and have been widely used in many applications
- Takes advantage of many decision trees with rules created from subsamples of features. Each tree is weaker than a full decision tree but combining them gives better overall performance.
- Each tree in the ensemble is built from a sample drawn with replacement (i.e., a bootstrap sample) from the training set.
- Instead of using all the features, a random subset of features is selected, further randomizing the tree.
- As a result, the bias of the forest increases slightly but due to the averaging of less correlated trees, its variance decreases, resulting in an overall better model.
- Can be used for classification or regression.
- Accuracy and variable importance information is provided with the results.
- Is an ensemble classifier that consists of many decision trees and outputs the class that is the mode of the class's output by individual trees.
- The method combines 'bagging' idea and the random selection of features.
- Random Forests generates its sequence of models by training them on subsets of data. The subsets are drawn at random from the full training set.

Steps in Random Forest Algorithm:

- Grow a forest of many trees.
- Grow each tree on an independent bootstrap sample (sample N cases at random with replacement) from the training data.
- At each node:
    - Select m variables at random out of all M possible variables (independently for each node).
    - Find the best split on the selected m variables.
- Grow the trees to maximum depth (classification)
- Vote/average the trees to get predictions for new data.

Advantages of Random Forest Algorithm:

- A sort of 'wisdom of the crowd'.
- Tends to result in very high-quality models.
- Fast to train.
- No need for pruning trees.
- Accuracy and variable importance generated automatically.
- Overfitting is not a problem.
- Not very sensitive to outliers in training data
- Easy to set parameters.
- Applicable to
- g both regression and classification problems.
- Handles categorical predictors naturally.
- Handles thousands of predictors.
- Computationally simple and quick to fit, even for large problems.
- No formal distributional assumptions (nonparametric).
- Can handle highly non-linear interactions and classification boundaries.
- Automatic variable selection but needs variable importance also.
- Handles missing values using proximities.
- Missing data imputation.
- Feature selection (before using a method that cannot handle high dimensionality).
- Unsupervised learning (cluster analysis).
- Survival analysis without making the proportional hazards assumption.

Disadvantages of Random Forest Algorithm:

- Models can get very large.
- Not easy to understand predictions.
- Regression can't predict beyond range in the training data
- In regression extreme values are often not predicted accurately – underestimate highs and overestimate lows
- Not very easy to interpret if the tree is small.

Applications of Random Forests in remote sensing:

- Classification
    - Land cover classification
    - Cloud/shadow screening
- Regression
    - Continuous fields (percent cover) mapping
    - Biomass mapping

Gradient Boosting Algorithm:

- Uses even weaker decision trees than Random Forest, that are increasingly focused on 'hard' examples.
- Used for regression and classification problems.
- It produces a prediction model in the form of an ensemble of weak prediction models typically decision trees.
- It builds the model in a stage-wise fashion and it generalizes them by allowing optimization of an arbitrary differentiable loss function.
- It develops an ensemble of tree-based models by training each of the trees in the ensemble on different labels and then combining the trees.
- Differs from Bagging and Random Forests in that it can reduce bias in addition to reducing variance.
- With Gradient Boosting, tree depth is only required to the extent that there is a significant interaction between variables.
- The basic difference in principle between bagging and boosting is that boosting constantly monitors its cumulative error and uses that residual for subsequent training. This difference accounts for GB only needing tree depth when there is significant interaction among various attributes in the problem.

Advantages of Gradient Boosting:

- High performance.

Disadvantages of Gradient Boosting:

- A small change in the feature set or training set can create radical changes in the model.
- Not easy to understand predictions.

Naïve Bayes Classifier:

- Is a simple but surprisingly powerful algorithm for predictive modeling
- A classification technique and a Probabilistic Graphic Model
- Uses Bayes theorem for building classifiers
- Based on the assumption that predictors are independent, assumes that the presence of a feature in a class is unrelated to the presence of any other feature
- Model is easy to build and useful for large data sets.
- Bayes theorem equation $P(A|B) = P(B|A)\ P(A)/P(B)$
- Python library used for building Naïve Bayes classifier is scikit learn
- Three types of Naïve Bayes models used (Gaussian, Multinomial and Bernoulli) under scikit learn package
- Generally, requires a small number of training data for classification
- Comprises two types of probabilities
    - The probability of each class
    - The conditional probability of each class based on the value x
- Can be used to make predictions for new data
- Can easily estimate probabilities as bell curve for real-valued data
- Is called naïve because it assumes that each input variable (feature) is independent
- Used for creating classifiers.
- Generally, requires a small number of training data for classification.

Support Vector Machines (SVM):

- Draws hyperplane or set of hyperplanes in a high/infinite dimensional feature space that separates instances into different categories with margins in between as far apart as possible
- Is a supervised learning algorithm and used for both classification and regression problems, but mainly used for classification problems
- A good separation is achieved by the hyperplane that has the longest distance to the nearest training-data point of any class (known as functional margin).
- A larger margin induces lower generalization error of the classifier.
- Main concept is to plot each data item as a point in n-dimensional space with the value of each feature being the value of a coordinate. (n is features that we would have)
- In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high dimensional feature spaces
- To minimize the probability of making errors on new examples, the SVM algorithm, by looking for the largest margin, explicitly tries to draw the decision boundary in such a way that it lies as far as possible from examples of both classes.
- SVM algorithm requires that the positive label has the numeric value of +1 and the negative label has the value of -1
- SVM sees every feature vector as a point in a high-dimensional space

Applications of SVM:

- Image classification
- Hand-written character recognition
- Biological and other sciences.

Advantages of SVM:

- Fast algorithms
- State of the art accuracy
- Power and flexibility from kernels
- Theoretical justification

Potential difficulties with SVM:

- Training time for large datasets
- Large number of support vectors for hard classification problems

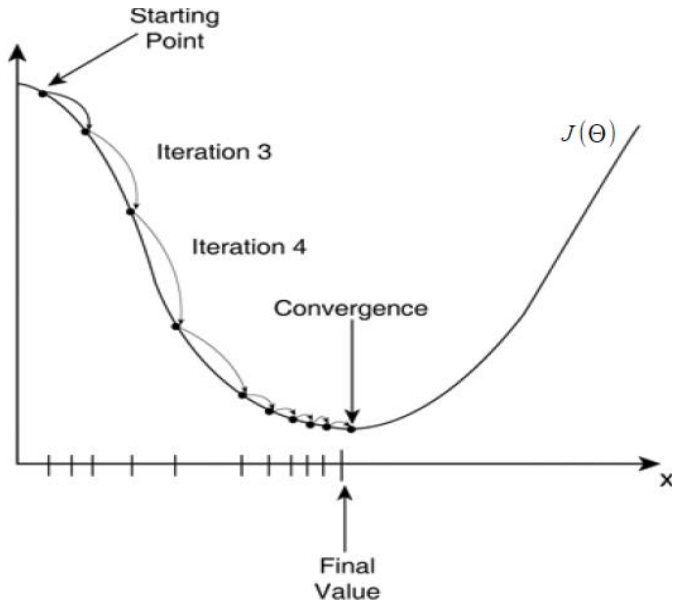
GRADIENT DESCENT ALGORITHM

- Is a Supervised Learning algorithm
- Is a powerful technique to optimize convex and non-convex objective functions
- Used to calculate weights, w.
- Gradient Boosting is based on gradient descent algorithm
- Also known as steepest descent
- Simplest algorithm for unconstrained optimization
- Types:
    - Stochastic Gradient Descent (SGD)
    - Functional Gradient Descent (FGD)
    - Conjugate Gradient Descent (CGD)
    - Projected Gradient Descent (PGD)
    - Proximal Gradient Descent
- Perceptron and Backpropagation are based on Gradient Descent
- Least Mean Squares (LMS) algorithm is a gradient descent algorithm
- Is a first-order iterative optimization algorithm for finding local minimum of a differentiable function
- Used to reduce loss function
- Model stops learning when the gradient (slope) is zero
- Minimum of the function lies in the opposite direction of gradient
- Start with a guess and take a step against gradient
- During training *w* are variable, during testing *x* are variable
- Find value of *w* where *J* value is minimum (*J* = Loss function)
- *f* is linear function, *J* is non-linear function
- Model starts learning quickly, but becomes slower towards end (fast to start, slow to converge).
- Step size too small and very low learning rate, too big step size could oscillate
- Learning rate is critical, start high to make rapid strides and reduce for smoother convergence
- Gradient Descent converges to a local minimum in the training error with respect to the network weights.
- Is a potentially useful method for searching many continuously parameterized hypothesis spaces where the training error is a differentiable function of hypothesis parameters
- Momentum is a heuristic to reduce the effect of zig-zagging and speed up gradient descent algorithm
- Vanishing Gradient problem – the gradient becomes weaker the further we move away from the data
- Gradient Descent is an optimization algorithm for finding the minimum of a function.
- It tweaks its parameters (coefficients) iteratively to minimize a given cost function.
- The model stops learning when the gradient (slope) is zero.
- Learning rate in Gradient Descent must not be too small or too large.

Steps involved in Gradient Descent:

1. Initialize parameter by some value
2. For each iteration calculate the derivative of the cost function and simultaneously update the parameters.

Example of Gradient Descent is shown below:



Epoch:

- In theory, we should sample with replacement, although in practice it is usually better to randomly permute the data and sample without replacement and then to repeat. A single such pass over the entire data set is called an 'Epoch'.

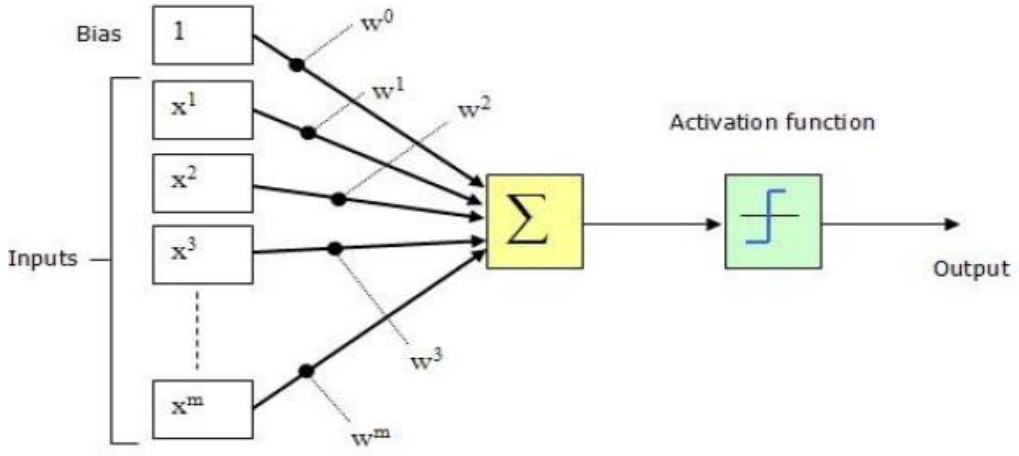
NEURAL NETWORKS

- Learn function using 'neurons'
- Interconnected 'neurons' that pass messages to each other.
- AI with the help of neural networks can analyze the data more deeply
- Are parallel computing devices that are an attempt to make a computer model of brain
- Neural networks are infamous for requiring extensive hyperparameter tuning.

PERCEPTRON:

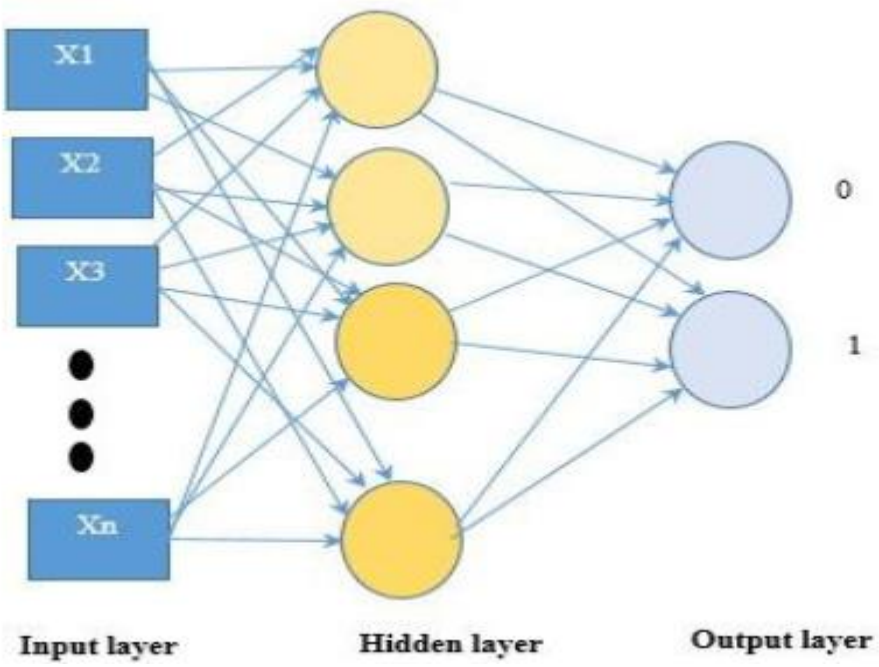
- Is a neural network unit (an artificial neuron) that does certain computations to detect features or business intelligence in the input data.
- Is a machine learning algorithm for supervised learning of binary classifiers
- Building blocks of ANN (mother of all ANNs)
- The perceptron is simply a fancy name for the simple neuron model with the step activation function. It was among the very first formal models of neural computation and because of its fundamental role in the history of neural networks, it wouldn't be unfair to call it the "mother of all artificial neural networks".
- It can be used as a simple classifier in binary classification tasks.
- A method for learning the weights of perceptron from data, called the Perceptron algorithm, was introduced by the psychologist Frank Rosenblatt in 1957. Suffice to say that it is just about as simple as the nearest neighbor classifier. Basic principle is to feed the network training data one example at a time. Each misclassification leads to an update in the weight.
- Follows the 'feed-forward' model, meaning inputs are sent into the neuron, are processed and result in an output.
- A simplest neural network possible, a computational model of a single neuron.

- Consists of one or more inputs, a processor and a single output.
- Perceptron Algorithm:
    - For every input, multiply that input by its weight
    - Sum all the weighted inputs
    - Compute the output of the perceptron based on that sum passed through an activation function.
- Perceptron architecture is shown below:



MULTI LAYER PECEPTRON (MLP):

- A single neuron model and a precursor to larger neural networks
- Investigates how simple models of biological brains can solve difficult computational tasks
- Develop robust algorithms and data structures that can model difficult problems
- Also known as feedforward neural network & is a series of logistic regression models stacked on top of each other with the final layer being either another logistic regression or a linear regression model depending on whether it is a classification or regression problem.
- MLP is a universal approximator (it can model any suitably smooth function given enough hidden units to any desired level of accuracy)
- MLP always requires that the weights form a directed acyclic graph
- MLP uses a supervised learning technique called backpropagation for training
- Its multiple layers and non-linear activation distinguish MLP from a linear perceptron
- It can distinguish data that is not linearly separable
- Consists of multiple layers called layers of hidden nodes stacked in between the layer of input nodes and the layer of output nodes.
- MLP neural network model will be as shown below:

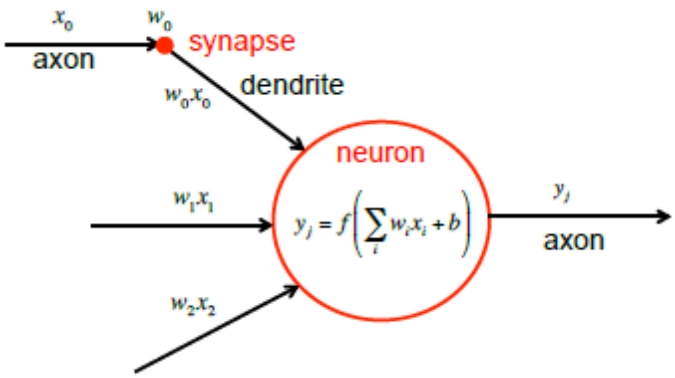


ARTIFICIAL NEURAL NETWORKS (ANN):

- Is a computing system made up of a number of simple, highly interconnected processing elements which process information by their dynamic state response to external inputs.
- A neural network, either biological and artificial, consists of many simple units, neurons, that receive and transmit signals to each other.
- A mathematical function conceived as a model of biological neurons
- Modeled loosely after the human brain
- Designed to recognize patterns
- Interpret sensory data through machine perception, labeling or clustering
- Learn function using 'neurons'
- Also known as connectionist systems
- Learn to perform tasks by considering examples, generally without being programmed with task-specific rules.
- Automatically generate identifying characteristics from the examples that they process.
- Require large amount of data to train

Dendrites, axons, and synapses

- In the biological lingo, we call the wires that provide the input to the neurons dendrites. Sometimes, depending on the incoming signals, the neuron may fire and send a signal out for the other neurons to receive.
- The wire that transmits the outgoing signal is called an axon.
- Each axon may be connected to one or more dendrites at intersections that are called synapses.
- The following figure shows axon, synapse, dendrite, neuron.



Features of Artificial Neuron:

- Cluster and classify the raw input
- Group unlabeled dataset based on the similarities in the inputs
- Classify labeled dataset based on expected results
- Extract features fed to the other algorithms
- The neurons are very simple processors of information, consisting of a cell body and wires that connect the neurons to each other. Most of the time, they do nothing but sit still and watch for signals coming in through the wires.
- Isolated from its fellow neurons, a single neuron is quite unimpressive, and capable of only a very restricted set of behaviors. When connected to each other, however, the system resulting from their concerted action can become extremely complex.
- To find evidence for this, look no further than (to use legal jargon) "Exhibit A": your brain! The behavior of the system is determined by the ways in which the neurons are wired together. Each neuron reacts to the incoming signals in a specific way that can also adapt over time. This adaptation is known to be the key to functions such as memory and learning.

Basic Three Layer Neural Network:

- Input Layer
- Hidden Layer
- Output Layer

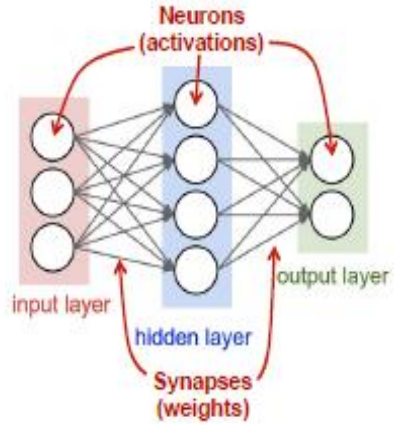
Input layer – vector data, each input collects one feature/dimension of the data and passes it on to the (first) hidden layer. It has many sensors to collect data from the outside world.

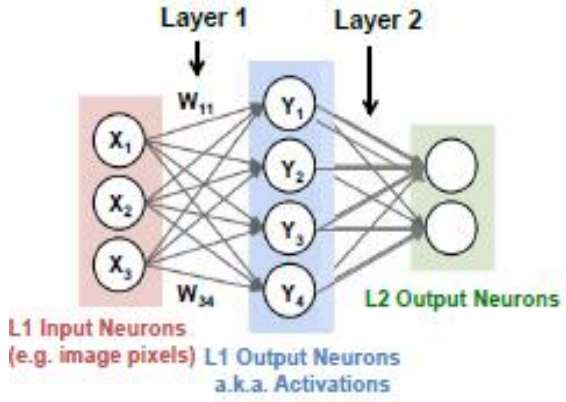
Hidden Layer – Each hidden unit computes a weighted sum of all the units from the input layer (or any previous layer) and passes it through a non-linear activation function. These are hidden between input and output layers. Each additional layer adds further complexity in training the network, but would provide better results in most of the situations. It can be thought of as a classifier or feature detector.

Output Layer – Each output unit computes a weighted sum of all the hidden units and passes it through a (possibly nonlinear) threshold function. This gives the result predicted by the network.

The wider and deeper the network the more complicated the mapping. Cross Validation or hyper parameter search methods are used to determine the number of layers and hidden units in a NN.

A simple neural network example with terminology is given below:





Properties of Neural Networks:

- Universality – given a large enough layer of hidden units (or multiple layers) a neural network can represent any function.
- Representation Learning: classic statistical machine learning is about learning functions to map input data to output. But Neural Networks, and especially Deep Learning are more learning a representation in order to perform classification or some other task.

Key features of Neural Networks:

- In a traditional computer, information is processed in a central processor (CPU) which can only focus on doing one thing at a time. The CPU can retrieve data to be processed from the computer's memory and store the result in the memory. Thus, data storage and processing are handled by two separate components of the computer: the memory and the CPU. In neural networks, the system consists of many neurons, each of which can process information on its own so that instead of having a CPU process each piece of information one after the other, the neurons process vast amounts of information simultaneously.
- The second difference is that data storage (memory) and processing isn't separated like in traditional computers. The neurons both store and process information so that there is no need to retrieve data from the memory for processing. The data can be stored short term in the neurons themselves (they either fire or not at any given time) or for longer term storage, in the connections between the neurons – their so-called weights.
- Because of these two differences, neural networks and traditional computers are suited for somewhat different tasks. Even though it is entirely possible to simulate neural networks in traditional computers, which was the way they were used for a long time, their maximum capacity is achieved only when we use special hardware (computer devices) that can process many pieces of information at the same time. This is called parallel processing. Incidentally, graphics processors (or graphics processing units, GPUs) have this capability and they have become a cost-effective solution for running massive deep learning methods.
- Complex function approximation through composition of functions
- Can learn arbitrary nonlinear decision boundary

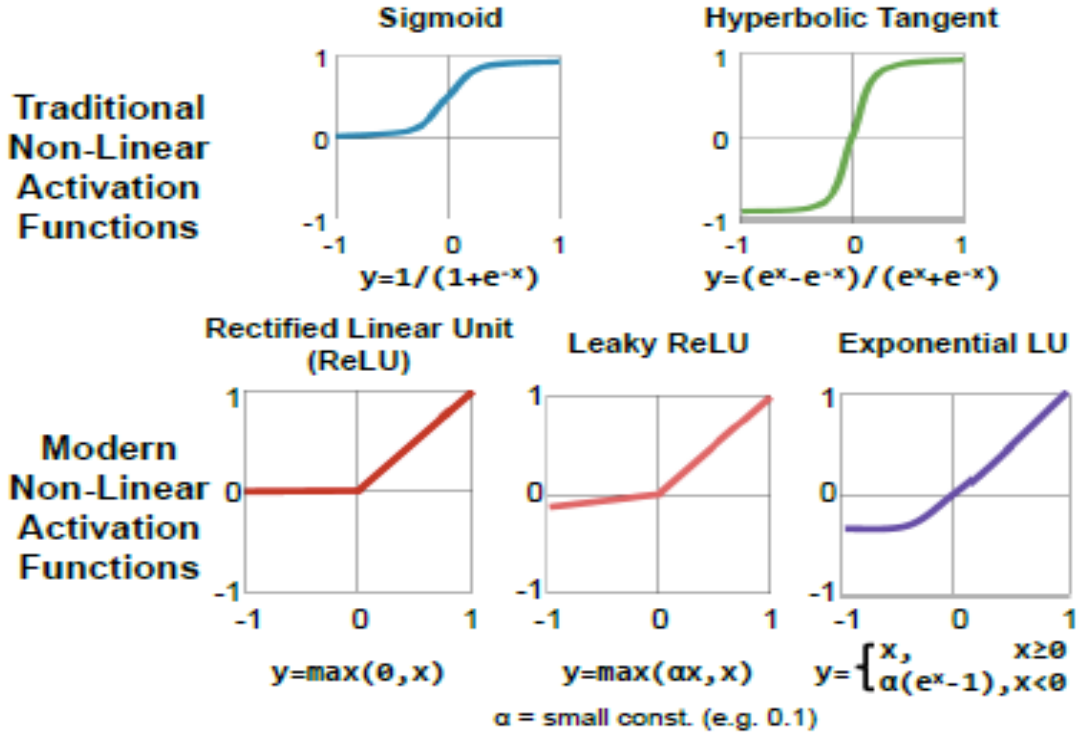
Hidden Layer – adding non-linearity:

- Each hidden unit emits an output that is a nonlinear activation function of its net activation
- This is essential to neural networks power, if it's linear then it all becomes just linear regression. The output is thus threshold through this nonlinear activation function.

Activation Functions:

- Activation Functions used to increase non-linearity of the network without affecting receptive fields of conv layers.
- A proper activation function significantly improves the performance of a CNN for a certain task. The following are the various types of activation functions in CNNs:
    - Sigmoid
    - Tanh
    - ReLU
    - Softmax

The above forms of non-linear activation functions are shown below:



Sigmoid:

- The Sigmoid function is used to represent a probability distribution over a binary variable.
- Sigmoid is now discouraged except for final layer to obtain probabilities.
- Can over-saturate easily and formula for Sigmoid function:

$$\sigma(z) = \frac{1}{1+\exp(-z)}$$

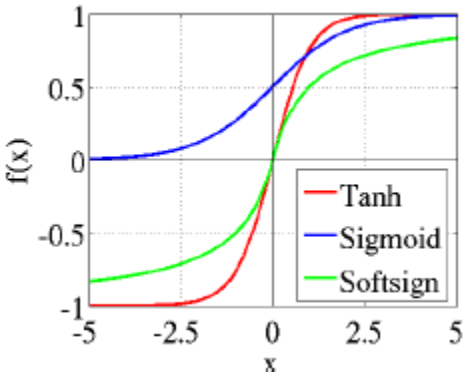
- This activation function saturates to 0 when z becomes very negative and saturates to 1 when z becomes very positive.
- For large absolute values of z, the gradient can become too small to be useful for learning even if the training data abundantly populate these regions.

tanh:

- tanh is most used and often performs best for deep networks.
- Formula for tanh:

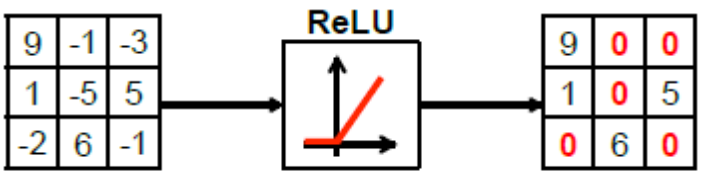
$$f(z) = \tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

- tanh is just a rescaled and shifted sigmoid.
- Sigmoid, tanh and softsign are shown below:



Rectified Linear Unit (ReLU):

- ReLU is the new standard activation function to use.
- One of the most notable non-saturated activation functions.
- Rectified Linear Units (ReLU) – have become standard max(0, net)
    - Strong signals are always easy to distinguish
    - Most values are zero, derivative is mostly zero.
    - They do not saturate as easily as sigmoid.
- ReLU is a non-linear activation function which is used to increase the non-linear properties of the decision function.
- ReLU layers accomplish piece-wise linear tiling, mapping is locally linear.
- ReLU results in faster learning
- Is a piecewise linear function which prunes the negative part to zero and retains the positive part.
- The simple max(.) operations of ReLU allows it to compute much faster than sigmoid or tanh activation functions and it also induces the sparsity in the hidden units and allows the network to easily obtain sparse representations.
- It has been shown that deep networks can be trained efficiently using ReLU even without pre-training.
- ReLU layers do local linear approximation. Number of planes grows exponentially with number of hidden units. Multiple layers yield exponential savings in number of parameters (known as parameter sharing).
- ReLUs are quickly optimized since the derivative is either 0 or a positive constant value through the domain.
- Many works have shown that ReLU works better than sigmoid and tanh activation functions empirically.
- ReLU non-linearity example is shown below:



Drawbacks of ReLU:

- cannot learn via gradient based methods on examples for which the activation is zero.
- It has zero gradient whenever the unit is not active. This may cause units that do not active initially never active as the gradient-based optimization will not adjust their weights.
- It amy slow down the training process due to the constant zero gradients.
- Discontinuity of ReLU at 0 may hurt the performance of backpropagation.

Leaky ReLU:

- To alleviate the problems with ReLU, Leaky ReLU is introduced.
- Compared with ReLU, Leaky ReLU compresses the negative part rather than mapping it to constant zero, which makes it allow for a small, non-zero gradient when the unit is not active.
- Leaky ReLU addresses the vanishing gradient problem

Softmax:

- A special kind of activation layer usually at the end of Fully Connected layer outputs.
- Can be viewed as a fancy normalizer.
- Produce a discrete probability distribution vector
- The Softmax function of z is a generalization of the Sigmoid function that represents a probability distribution over a discrete variable with n possible values.
- Softmax functions are often used as the output units of a classifier.

- Formally the softmax function is given by

$$\mathrm{softmax}(z) = \frac{\exp(z_i)}{\sum_j \exp(z_i)}$$

- Cost functions that do not use a log to undo the exp of the softmax cause a failure to learn when the argument to the exp becomes very negative causing the gradient to vanish.
- Very convenient when combined with cross entropy loss

Types of Neural Networks:

There are countless types of neural networks. The following are some of the most relevant types:

- Feed Forward
- Radial Basis
- Kohonen
- Recurrent
- Modular

Feed Forward Neural Networks

- Used in computer vision and speech recognition when classifying the target classes are complicated.
- Responsive to noisy data.
- Easy to maintain.
- Does not have input at each step
- Has different parameters for each layer

Radial Basis Neural Networks:

- Considers the distance of a point with respect to the center.
- Used for power restoration systems which are notoriously complicated.

Kohonen Neural Networks:

- Recognizes patterns in dat.
- Used in medical analysis to cluster data into different categories.
- Able to classify patients with a diseased glomerular vs a healthy one.

Recurrent Neural Networks:

- Feeds the output of a layer back as input.
- Good for predicting the next word in a body of text but harder to maintain.

Modular Neural Networks:

- Collection of different networks work independently & contribute towards the final output.
- Increases computation speed (through the breakdown of a complicated computational process into simpler computations), but processing time is subject to the number of neurons.

Advantages of Neural Networks:

- Can handle extremely complex tasks.
- No other algorithm comes close in image recognition.

Disadvantages of Neural Networks:

- Very slow to train, because they often have a very complex architecture.
- Almost impossible to understand predictions.
- Diminishing gradient inhibits multiple layers
- Can get stuck in local minimums
- Training time can be extensive

PERFORMANCE METRICS OF SUPERVISED LEARNING ALGORITHMS:

Confusion Matrix:

- A specific table layout that allows visualization of the performance of a supervised learning algorithm.
- Each column of the matrix represents the instances in a predicted class while each row represents the instances in an actual class or vice-versa.
- The name stems from the fact that it makes it easy to see if the system is confusing two classes (i.e. commonly mislabeling one as another).
- For multiple label categorization problems, the confusion matrix can be used as an evaluation metric.
- Confusion Matrix for a multi-class classification demonstrates how the model of classification is confused when it makes projections.

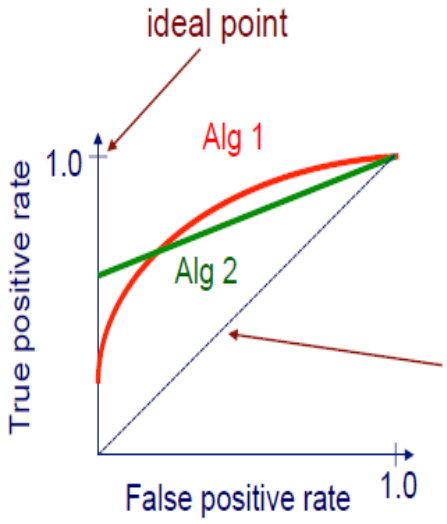
Accuracy:

- Accuracy Matrix provides with a summary of the prediction being made accurately.
    - Accuracy = [True Positive (TP) + True Negative (TN)] / Total
- Accuracy may not be useful measure in cases where
    - There is a large class skew (for example, is 98% accuracy good if 97% of the instances are negative?)
    - There are differential misclassification costs – say, getting a positive wrong, costs more than getting a negative wrong. Consider a medical domain in which a false positive, results in an extraneous test but a false negative, results in a failure to treat a disease.
    - We are most interested in a subset of high-confidence predictions.

ROC Curve:

- Accuracy Matrix includes ROC (Receiver Operating Characteristics) Curve and AUC (Area Under the ROC Curve).
- ROC curve compares the model's TPR (True Positive Rate) and FPR (False Positive Rate) to the ones from a random assignment.
- It is a performance measurement for classification problem at numerous threshold settings.
- ROC is a probability curve and is becoming more popular in ML.
- ROC curve is developed during WW II to statistically model false positive and false negative detections of radar operators
- ROC measure has better statistical foundations than most other measures
- ROC is a standard measure in medicine and biology
- AUC represents degree or measure of separability – the higher the AUC, the better the model is at predicting.
- AUC measures the entire two-dimensional area under the entire ROC curve.
- ROC curve (similar to PR curves, Precision-Recall):
    - allow predictive performance to be assessed at various levels of confidence
    - Assume binary classification tasks
    - Sometimes summarized by calculating area under the curve
- ROC curves are insensitive to changes in class distribution (ROC curve does not change if the proportion of positive and negative instances in the test set varied)

- ROC curves identify optimal classification thresholds for tasks with differential misclassification costs.
- A typical ROC curve is shown below. Different methods can work better in different parts of ROC space. This depends on the cost of FP vs. FN.



expected curve for random guessing

Properties of ROC:

- If ROC Area is:
  - 1.0 – perfect prediction
  - 0.9 – excellent prediction
  - 0.8- Good prediction
  - 0.7 – mediocre prediction
  - 0.6 – poor prediction
  - 0.5 – random prediction
  - <0.5 – something wrong
- Slope is non-increasing and each point on ROC represents different trade-off (cost ratio) between false positives and false negatives
- Slope of line tangent to curve defines the cost ratio
- ROC area represents performance averages over all possible cost ratios
- If two ROC curves do not intersect, one method dominates the other
- If two ROC curves intersect, one method is better for some cost ratios and the other is better for other cost ratios

Algorithm for creating ROC curve:

- Sort test-set predictions according to confidence that each instance is positive
- Step through sorted list from high to low confidence
  - Locate a threshold between instances with opposite classes (keeping instances with the same confidence value on the same side of threshold)
  - Compute TPR, FPR for instances above threshold
  - Output (FPR, TPR) coordinate

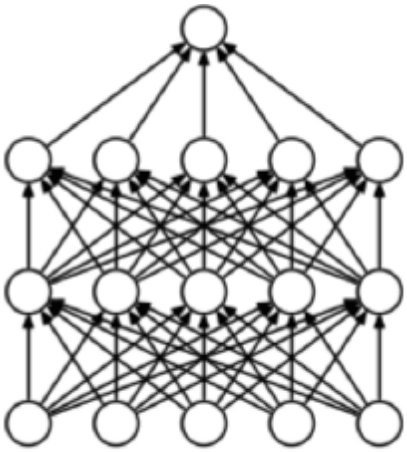
Model Evaluation Metrics:

		Actual Label	
		Positive	Negative
Predicted Label	Positive	**True Positive (TP)**	**False Positive (FP)**
	Negative	**False Negative (FN)**	**True Negative (TN)**

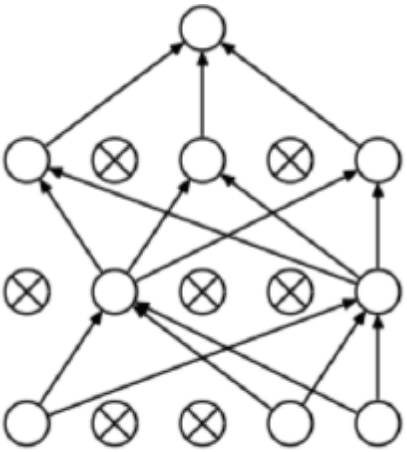
Accuracy	(TP + TN) / (TP + TN + FP + FN)	The percentage of predictions that are correct
Precision	TP / (TP + FP)	The percentage of positive predictions that are correct
Sensitivity (Recall)	TP / (TP + FN)	The percentage of positive cases that were predicted as positive
Specificity	TN / (TN + FP)	The percentage of negative cases that were predicted as negative

DROPOUT:

- Means randomly ignore certain units during training, don't update them via gradient descent, leads to hidden units that specialize
- The key idea is to randomly drop units (along with their connections) from the neural network during training.
- Is a method for regularization of machine learning.
- With probability p, don't include a weight in the gradient updates.
- The key idea of a dropout layer is to randomly disable input units after each iteration of the training.
- A neural network with n units that employs dropout can be seen as a collection of 2^n possible neural networks.
- These possible networks have a smaller number of units but still share weights such that the total number of parameters is unchanged.
- Training a network with dropout can be seen as training a collection of 2^n smaller networks with extensive weight sharing.
- At test time it is easy to approximate the effect of averaging the predictions of the smaller networks by using a single network without dropout that has smaller weights. This significantly reduces overfitting and gives significant improvements over other regularization methods.
- Proven to be very effective in reducing overfitting.
- Dropout can prevent the network from becoming too dependent on any one or any small combination of neurons and can force the network to be accurate even in the absence of certain information.
- Dropout introduces a significant amount of noise in the gradients compared to standard SGD.
- Dropout introduces an extra hyperparameter – the probability of retaining a unit, which controls the intensity of dropout.
- The central idea of dropout is to take a large model that overfits easily and repeatedly sample and train smaller sub-models from it.
- Besides feed forward neural networks, dropout can also be applied to Restricted Boltzmann Machines (RBM).



Standard Neural Net



After applying dropout.

Advantages of Dropout:

- Improves performance of neural networks on supervised learning tasks in vision, speech recognition, document classification and computational biology.
- Reduces overfitting in neural networks by encouraging robustness of weights in the network and gives major improvements over other regularization methods.
- Prevents units from co-adapting too much.

Drawbacks of Dropout:

- It increases training time. A dropout network typically takes 2 to 3 times longer to train than a standard neural network of the same architecture.

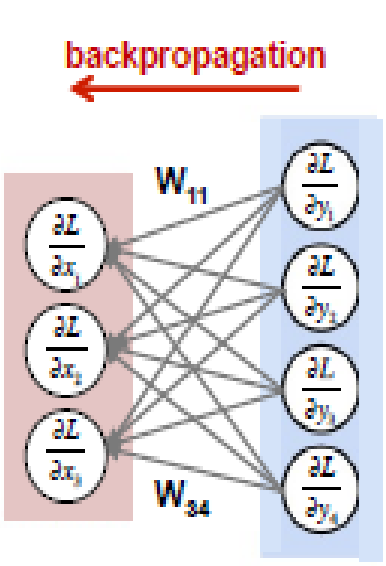
BACKPROPAGATION ALGORITHM

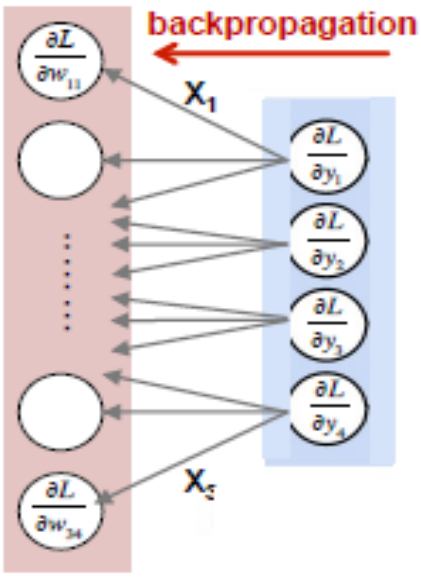
- Is the most common network learning method and has been successfully applied to a variety of learning tasks such as handwriting recognition and robot control
- Considers hypothesis space of all functions that can be represented by assigning weights to the given, fixed network of interconnected units.
- Backpropagation searches the space of possible hypotheses using gradient descent to iteratively reduce the error in the network fit to the training examples.
- One of the most intriguing properties of backpropagation is its ability to invent new features that are not explicit in the input to the network.
- Most common ANN learning algorithm
- Squared error over the entire training set is computed in this algorithm.
- This algorithm computes the gradient vector of the NLL by applying the chain rule of calculus.
- Backpropagation (also known as Chain rule) is the procedure to compute gradients of the loss with respect to parameters in a multi-layer neural network.
- Allows to fit models with hidden layers
- Layer 1 errors can be computed by passing the layer 2 errors back through the W matrix, hence the term 'backpropagation'.
- Propagates backwards the gradients through the network starting at the end, that is why the term 'backpropagation'.

- Given the backpropagation algorithm you can directly run gradient descent using it as a subroutine for computing the gradients.
- Key property is that we can compute the gradients locally: each node only needs to know about its immediate neighbors
- This is supposed to make the algorithm 'neurally plausible', although this interpretation is somewhat controversial.
- Backpropagation requires computing the gradient of the output with respect to the weights (among other things)
- Forward Propagation is the process of computing the output of the network given its input.
- Backpropagation work with any a.e. differentiable transformation in addition to ReLU layer.
- The computational cost of Backpropagation is about twice Forward Propagation as there is a need to compute gradients with respect to input and parameters at every layer.
- FPROP and BPROP are dual of each other.

Process:

- Compute all the gradients as follows:
  - First perform a forward pass to compute (Forward propagation)
  - Then compute the error for the output layer (Compute output units errors)
  - Pass backwards the error through W to compute the error for the hidden layer (Compute hidden units errors)
  - Then compute the overall gradient (Update network weights by gradient descent)
- An example of backpropagation through neural network is shown below. On the left, the backpropagation computes the gradient of loss relative to the filter inputs and on the right, the backpropagation computes the gradient of loss relative to the weights.





Design choices: The following design choices must be made in applying backpropagation in face recognition:

- Input encoding
- Output encoding
- Network graph structure
- Other learning algorithm parameters

Heuristics for training Neural Networks for improving Backpropagation:

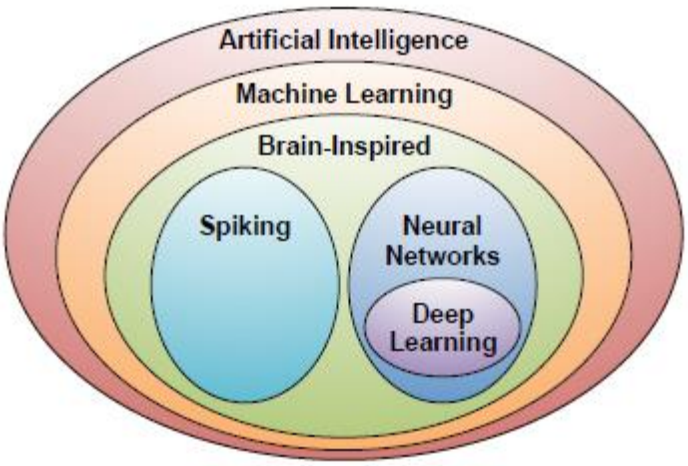
- Less hidden nodes, just enough complexity to work, not too much to overfit.
- Train multiple networks with different sizes and search for the best design
- Validation set – train on training set until error on validation set starts to rise, then evaluate on evaluation set
- Try different activation functions: tanh, ReLU, ELU….
- Dropout – randomly ignore certain units during training, don't update them via gradient descent, leads to hidden units that specialize
- Modify learning rate over time (cooling schedule)
- The backpropagation algorithm iterates through many cycles of two processes. Each cycle is known as an epoch.
- As the network contains no existing knowledge, the starting weights are typically set at random.
- Then the algorithm iterates through the processes, until a stopping criterion is reached.
- Each epoch in the backpropagation algorithm includes:
    - A forward phase in which the neurons are activated in sequence from the input layer to the output layer, applying each neuron's weights and activation function along the way. Upon reaching the final layer, an output signal is produced.
    - A backward phase in which the network's output signal resulting from the forward phase is compared to the true target value in the training data. The difference between the network's output signal and the true value results in an error that is propagated backwards in the network to modify the connection weights between neurons and reduce future errors.
- Over time, the network uses the information sent backward to reduce the total error of the network.
- By applying gradient descent, the algorithm determines how much a weight should be changed even though the relationship between each neuron's inputs & outputs is complex.
- The backpropagation algorithm uses the derivative of each neuron's activation function to identify the gradient in the direction of each of the incoming weights and hence the importance of having a differentiable activation function.
- The gradient suggests how steeply the error will be reduced or increased for a change in the weight. The algorithm will attempt to change the weights that result in the greatest reduction in error by an amount known as the learning rate.
- The greater the learning rate, the faster the algorithm will attempt to descend the gradients.

Drawbacks with back propagation:

- It requires labeled training data. Almost all data is unlabeled.
- The learning time does not scale well. It is very slow in networks with multiple hidden layers.
- It can get stuck in poor local optima.

DEEP LEARNING

- Deep Learning is a subset of ML, which is itself a subset of AI as shown below:



- Deep Learning has been around for a while, but it returned to the headlines in 2016 when Google's AlphaGo program crushed Lee Sedol, one of the highest-ranking Go players in the world.
- With advancements in machine learning algorithms and deep learning chipsets, DL is being more actively implemented.
- Deep Learning is being applied across industries from healthcare to finance to retail and everything in between.
- The global deep learning market is expected to reach USD 10.2 Billion by 2025.
- On a less conceptual and more tactical level, deep learning (DL) is a subset of machine learning (ML) which focuses on learning data representations.
- The focus is on relationships rather than tasks like classical ML algorithms, creates transferable solutions. It's the difference between being able to identify a cat as a whole as opposed to understanding the different concepts defining a cat (like the paws, tail and ears) and the way they are nested.
- This is one of the key reasons why deep learning is more powerful than classical machine learning – it creates transferable solutions. That is, the concepts of paw, tail and ears can be easily reused to understand what a dog is as well.
- Deep Learning algorithms are able to create transferable solutions through neural networks: that is layers of neurons/units.
- Neurons make up neural networks and those in turn allow machines (via deep learning) to 'learn' like humans. To have robust understanding, it's important to understand how those underlying neurons actually work.
- Deep Learning uses several layers of neural networks stacked on top of one another.
- Deep learning refers to certain kinds of machine learning techniques where several "layers" of simple processing units are connected in a network so that the input to the system is passed through each one of them in turn.
- This architecture has been inspired by the processing of visual information in the brain coming through the eyes and captured by the retina.
- This depth allows the network to learn more complex structures without requiring unrealistically large amounts of data.
- Deep Learning deals with finding features automatically from raw data.
- In deep learning the raw data is fed to an algorithm that extracts hierarchical features automatically based on optimizing the performance of the algorithm on the task.
- Deep learning models automatically learn to associate inputs and desired outputs from examples.

- Deep Learning is a subset of machine learning that involves multiple layers of representations that allow a computer to learn and deduce outputs from data.
- Deep Learning is learning hierarchical models. ConvNets are the most successful example.
- DL is a general-purpose framework for representation learning
    - Given an objective
    - Learn representation that is required to achieve objective
    - Directly from raw inputs
    - Using minimal domain knowledge

Steps involved in deep learning:

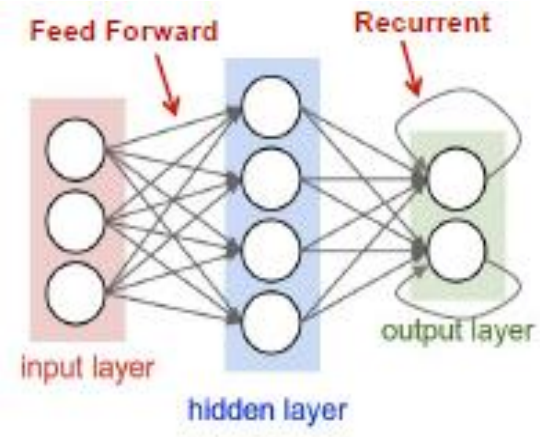
- Pre-processing of input data
- Training the deep learning model
- Inference and deployment of the model

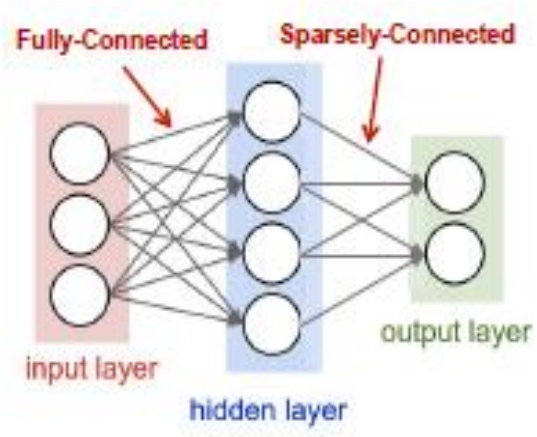
Applications of Deep Learning:

- Manufacturing
- Automotive
- Hospitality
- Health Care
- Banking, Insurance and Finance
- Agriculture
- Entertainment
- IT/Security
- Retail, Supply Chain and Logistics

Drawbacks of fully connected Deep Neural Networks (DNN):

- Large number of parameters
    - Easy to be overfitted
    - Large Memory consumption
- Does not enforce any structure, e.g., local information
    - In many applications, local features are important, e.g., images, language etc.
- While DNNs can deliver outstanding accuracy, it comes at the cost of high computational complexity.





Type of Deep Neural Networks:

- Convolutional Neural Networks (CNN)
- Recurrent Neural Networks (RNN)
- Restricted Boltzmann Machine (RBM)
- Deep Belief Networks (DBN)
- Autoencoders

A Deep Neural Network is typically composed of:

- Linear transformations
- Non-linear activation functions
- A loss function on the output
    - Mean Squared Error (MSE)
    - Log likelihood

Metrics for DNN Models:

- *Accuracy*
    - The accuracy of the model in terms of the top-5 error on datasets such as ImageNet. Also, the type of data augmentation used (e.g., multiple crops, ensemble models) should be reported.
    - The accuracy determines if it can perform the given task.
- *Network architecture*
    - The Network Architecture of the model should be reported, including number of layers, filter sizes, number of filters and number of channels.
- *Number of weights*
    - The number of weights impact the storage requirement of the model and should be reported. If possible, the number of non-zero weights should be reported since this reflects the theoretical minimum storage requirements.
- *Number of Multiply and Accumulates (MACs)*
    - The number of MACs that needs to be performed should be reported as it is somewhat indicative of the number of operations and potential throughput of the given DNN. If possible, the number of non-zero MACs should also be reported since this reflects the theoretical minimum compute requirements.

Metrics for DNN Hardware:

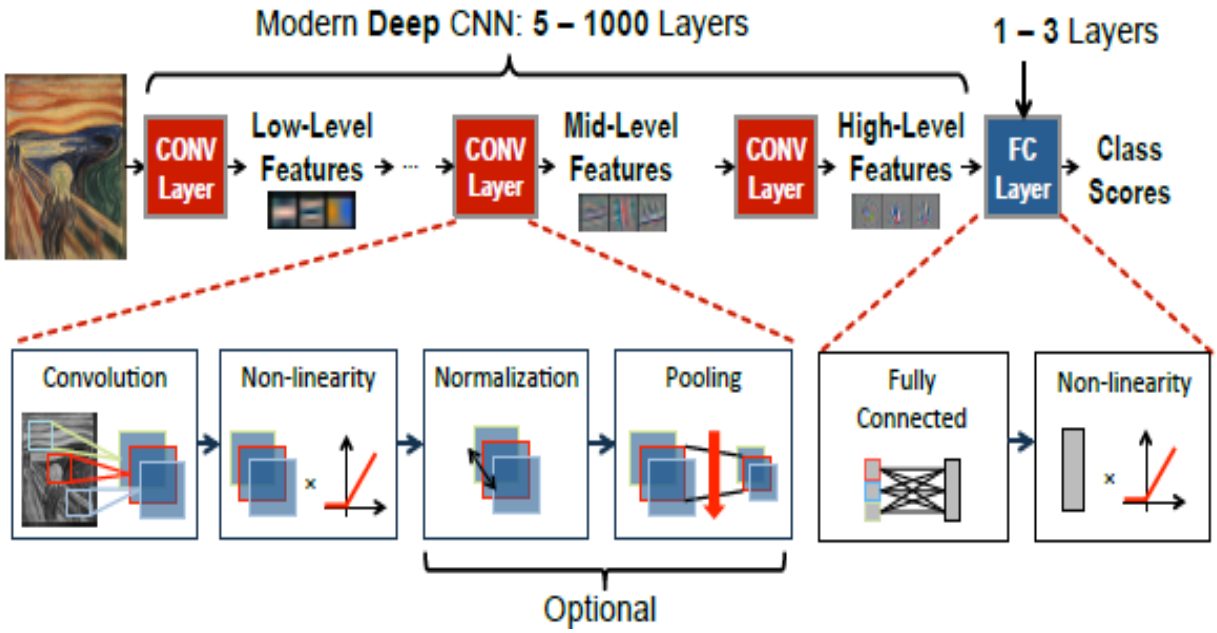
- *Power and energy*
    - The power and energy consumption of the design should be reported for various DNN models; the DNN model specifications should be provided including which layers and bit precision are supported by the hardware during measurement. In addition, the amount of off-chip accesses (e.g., DRAM accesses) should be included since it accounts for a significant portion of the system power; it can be reported in terms of the total amount of data that is read and written off-chip per inference.
    - The energy and power consumption will primarily dictate the form factor of the device where the processing can operate.

- *Latency and throughput*
    - This should be reported in terms of batch size and actual run time for various DNN models, which accounts for mapping and memory bandwidth effects. This provides a more useful and informative metric than peak throughput.
    - The latency and throughput determine if it can run fast enough and in real-time.
- *Cost*
    - The cost of the chip depends on the area efficiency, which accounts for the size and type of memory (e.g., registers or SRAM) and the amount of control logic. It should be reported in terms of the core area in squared millimeters per multiplier along with process technology.
    - The cost, which is primarily dictated by the chip area, determines how much one would pay for this solution.

CONVOLUTIONAL NEURAL NETWORKS (CNN):

- Similar to feed-forward neural networks but dominate computer vision because of their much high accuracy.
- A form of Multi-Layer Perceptron (MLP) which is particularly well suited to one-dimension signals like speech or text, or 2 dimensional signals like images is the 'Convolutional Neural Network' (CNN).
- A network with convolutional layers is called 'Convolutional Neural Network'.
- An MLP in which the hidden units have local receptive fields and in which the weights are tied or shared across the image in order to reduce the number of parameters.
- CNN consists of an input and an output layer, as well as multiple hidden layers.
- The hidden layers of a CNN typically consist of convolutional layers, pooling layers, fully connected layers and normalization layers
- Original goal of CNN is 'how to create good representations of the visual world in a way it could be used to support recognition?'
- CNNs are developed from examining the way our own visual detection cortex works.
- The idea of CNNs was neurobiologically motivated by the findings of locally-sensitive and orientation-selective nerve cells in the visual cortex.
- Inventors of CNN designed a network structure that implicitly extracts relevant features.
- CNNs are a special kind of multilayer neural networks.
- CNN starts with an input image, extracts primitive features and combines them form certain parts of object and finally it pulls together all of various parts of objects together to form object itself.
- A standard neural network applied to images scales quadratically with the size of the input and does not leverage stationarity and this is solved by CNN and use of convolutional layer.
- It is hierarchical way of seeing objects.
    - Layer 1 – very simple features are detected
    - Layer 2 – combined to form more complicated shape of object and so on in subsequent layers.
- CNN is a set of layers with each of them is responsible for detecting a set of feature sets. These features are going to be more abstract as it goes further into examining the next layer.
- CNNs automatically find the features and classify them.
- CNN is a type of neural network empowered with specific hidden layers including the convolution layer, pooling layer and the fully connected layer.

- CNN typical architecture is shown below:



Core idea behind CNN:

- Local connections - Represent how each set of neurons in a cluster is connected to each other, which in turn represents a set of features.
- Layering – represents the hierarchy in features that are learned
- Spatial Invariance – represents the capability of CNNs to learn abstractions invariant of size, contrast, rotation and variation.

Popular CNNs:

- LeNet
- AlexNet
- VGGNet
- ResNet

CNN Architectures:

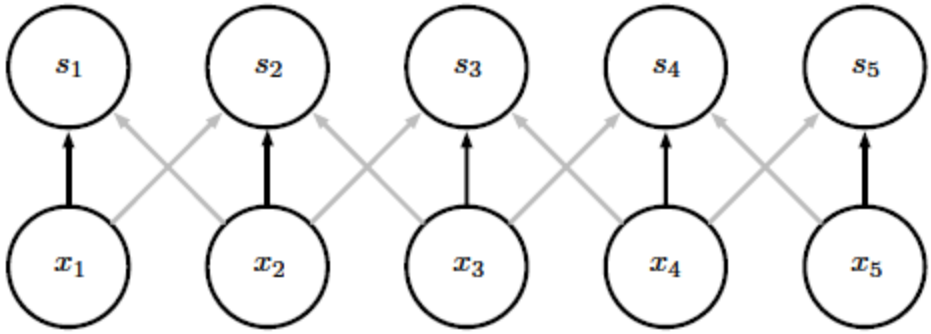
- VGGNet
  - 16 layers
  - Only 3*3 convolutions
  - 138 million parameters
- ResNet
  - 152 layers
  - ResNet50

Key features of CNN:

- Detect and classify objects into categories
- Independence from pose, scale, illumination, conformation and clutter (which is a huge limitation historically of the hand coded algorithms)

Reason for convolutional layer:

- **Parameter sharing** – a feature detector (such as a vertical edge detector) that's useful in one part of the image is probably useful in another part of the image. Convolution shares the same parameters across all spatial locations. Traditional matrix multiplication does not share any parameters. Parameter sharing example is shown below:



- **Sparsity of connections** – in each layer, each output value depends only on small number of inputs
- Main purpose of a convolutional layer is to detect different patterns or features from an input image (for example its edges)
- Convolution function is a simple matrix multiplication.
- Convolution layer connects each hidden unit to a small patch of the input and shares the weight across space.
- Size of output in a convolutional layer and it's computational cost is proportional to the number of filters and depends on the stride.
- If kernels have size KxK, input has size DxD, stride is 1, and there are M input feature maps and N output feature maps then:
    - The input has size M = DxD
    - The output has size N = (D-K+1) x (D-K+1)
    - The kernels have MxNxKxK coefficients (which have to be learned)
    - Computational Cost = M*K*K*N*(D-K+1)*(D-K+1)

Feature Map:

- Is the output of convolution process.
- Usually there are more output feature maps than input feature maps. Convolutional layers can increase the number of hidden units by big factors and are expensive to compute.
- The size of the filters has to match the size/scale of the patterns we want to detect (task dependent).

Stride:

- Governs how many cells the filter is moved in the input to calculate the next cell in the result
- Stride dictates the sliding behavior of the max pooling.
- If stride = 2, the window moves by 2 pixels every time.
- Number of pixels overlap between adjacent filters

Padding – benefits:

- Allows to use a CONV layer without necessarily shrinking the height and width of the volumes. This is important for building deeper networks, since otherwise the height/width would shrink as we go to deeper layers.
- It helps us to keep more of the information at the border of an image. Without padding, very few values at the next layer would be affected by pixels as the edges of an image.

Zero padding:

- Removing edge pixels from filter scan
- Can reduce size of network and deal with edge effects

Weight sharing and local connectivity (convolution):

- Use multiple filters convolve over inputs
- Reduce the number of parameters (less over-fitting)
- Learn local features
- Translation invariance

Pooling (subsampling):

- Make the representations smaller
- Reduce number of parameters and computation
- By applying 'pooling' (e.g., taking max) filter responses at different locations we gain robustness to the exact spatial location of features. This way we can make the detection robust to the exact location of the feature (like eye).
- Pooling is an important concept of CNN.
- It lowers the computational burden by reducing the number of connections between convolutional layers.
- Nearby pixels tend to represent the same thing/class/object. So, pool responses from nearby nodes (e.g., mean, median, min, max).

Weight sharing:

- Apply same weights over the different spatial regions
- One can achieve translation invariance (not perfect though)

Translation invariance:

- When input is changed spatially (translated or shifted), the corresponding output to recognize the object should not be changed
- CNN can produce the same output even though the input image is shifted due to weight sharing

Pooling Layer:

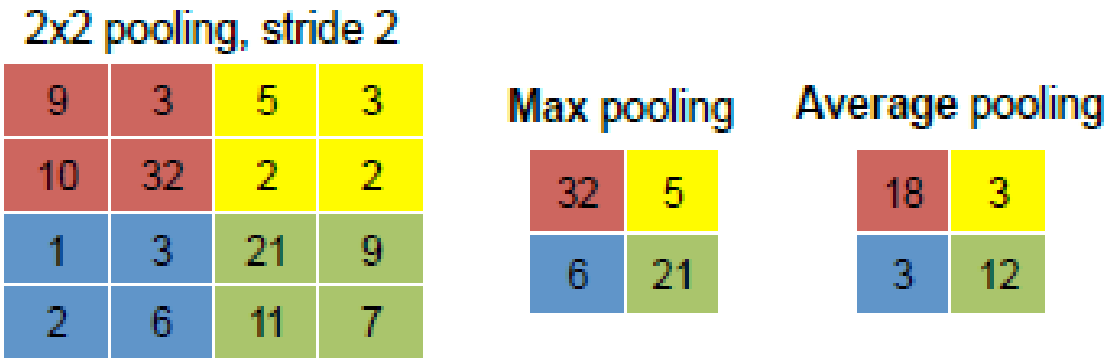
- Makes the representations smaller and more manageable
- Operates over each activation map independently
- Enhance translation invariance (invariance to small transformation)
- Larger receptive fields (see more of input)
- Regularization effect

- Gradually reduces the spatial size of each matrix within the feature map such that the amount of parameters and computation is reduced in the network.
- The size of output in pooling layer depends on the stride between the pools.
- For instance, if pools do not overlap and have size KxK, and the input has size DxD with M input feature maps, then:
  - Output = (D/K) x (D/K)
  - The computational cost is proportional to the size of the input (negligible compared to a convolutional layer)
- Size of the pooling depends on how much 'invariant' or robust to distortions we want the representation to be. It is best to pool slowly (via a few stacks of conv-pooling layers).
- Pooling layer applies non-linear down sampling on activation maps.
- It is common to use zero padding for pooling layers.

Types of Pooling layers:

- Max pooling
- Average pooling
- Stochastic pooling
- ROI pooling
- L2 pooling
- L2 pooling over features

Various forms above pooling are shown below:



Lp Pooling:

- Is a biologically inspired pooling process modelled on complex cells.
- Provides better generalization than max pooling.

Stochastic Pooling:

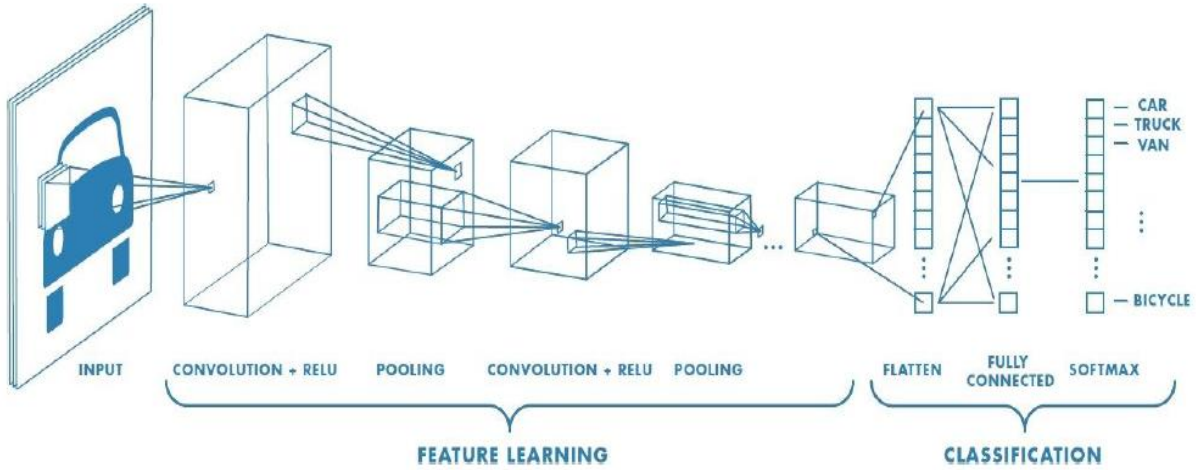
- Is a dropout inspired pooling method.
- Instead of picking the maximum value within each pooling region as max pooling does, stochastic pooling randomly picks the activations according to a multinomial distribution which ensures that the non-maximal activations of feature maps are also possible to be utilized.
- Computes the probabilities for each region by normalizing the activations within the region.
- Compared with max pooling, stochastic pooling can avoid overfitting due to the stochastic component.

Max pooling:

- Most commonly used pooling approach.
- Is on operation that finds maximum values and simplifies the inputs.
- It reduces the parameters within the model
- Turns low level data into higher level information

CNN Architecture:

- The CNN architecture comprises multiple combinations of convolution and pooling layers.
- The reduced image from these layers (convolution + pooling) is then passed through the activation function.
- A simple CNN architecture is shown below:



Factors for choosing the correct CNN architecture:

- Task dependent
- Cross Validation
- The more data: the more layers and the more kernels
    - Look at the number of parameters at each layer
    - Look at the number of flops at each layer
- Computational resources
- Be creative.

How to Optimize CNN:

- SGD (with momentum) usually works very well
- Select learning rate by running on a subset of the data
    - Start with large learning rate and divide by 2 until loss does not diverge
    - Decay learning rate by a factor of ~1000 or more by the end of training
- Use non-linearity
- Initialize parameters so that each feature across layers has similar variance. Avoid units in saturation.

Key techniques for optimizing CNNs:

- Data Augmentation
- Weight Initialization
- Stochastic Gradient Descent
- Batch Normalization
- Shortcut connections

Good practices while working with CNN:

- Check gradients numerically by finite differences
- Visualize features (feature maps need to be uncorrelated) and have high variance
    - If the model training is good – hidden units are sparse across samples and across features
    - If the model training is bad – many hidden units ignore the input and/or exhibit strong correlations.
- Visualize parameters
    - If the model training is good – learned filters exhibit structure and are uncorrelated.
- Measure error on both training and validation set.
- Test on a small subset of the data and check the error tends to 0.

Reasons and what to do if a CNN does not work:

- Training diverges:
    - Learning rate may be too large and decrease learning rate
    - BPROP is buggy and solution is numerical gradient checking
- Parameters collapse / loss is minimized but accuracy is low
    - Check Loss function:
        - Is it appropriate for the task you want to solve?
        - Does it have degenerate solutions? Check 'pull-up' term.
- Network is underperforming
    - Compute flops and nr.params. If too small, make net larger
    - Visualize hidden units/params and fix optimization
- Network is too slow
    - Compute flops and nr. Params. Solution is GPU, distributed framework and make network smaller.

Layers of CNN:

- Based on the connection pattern and operations, we can think of a layer in a CNN as:
    - Convolutional – A layer can have multiple channels
    - Non-linear (often not drawn)
    - Max-pooling
    - Fully connected
    - Soft Max

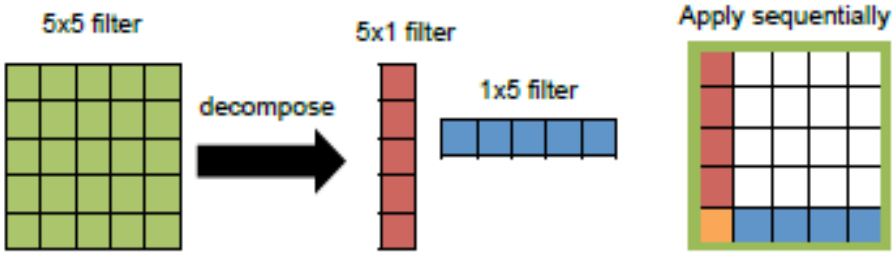
Training in ConvNets:

- All layers are differentiable.
- We can use standard backpropagation.
- Steps in algorithm: Given a small mini batch
    - F-PROP
    - B-PROP
    - PARAMETER UPDATE

Working of CNNs:

- **Learning an Image** - CNN focuses on smaller and specific patterns than the whole image. It's convenient and effective to present a smaller region with fewer parameters, thereby reducing computational complexity.
- **Convolutional Layer –** CNN is a neural network with convolutional layers and other layers. A convolutional layer has several filters that perform the convolution operation.
- **Convolution Operation:**
    - Consider a 6x6 image convolved with 3x3 filter(s) to give an output of size 4x4. Filters can be considered as network parameters to be learned.
    - Shift the filter around the input matrix (commonly known as stride) once a convolved output is achieved. If the stride size is changed, the convolved output will vary (only outputting intense pixels).
    - The convolution operation gets repeated for each filter resulting in a feature map.
    - When RGB image is used as input to CNN, the depth of filter is always equal to the depth of image (3 in case of RGB).
- Decomposing larger filters into smaller filters is shown below

(a) Constructing a 5×5 support from 3×3 filters. Used in VGG-16.



(b) Constructing a 5×5 support from 1×5 and 5×1 filter. Used in GoogleNet/Inception v3 and v4.

Applications of CNN:

- Transfer Learning and Fine Tuning
- Feature Extraction
- Signal and image processing
- Handwritten text/digits recognition
- Natural object classification (photos and videos)
- Segmentation
- Face detection
- Recommender systems
- Speech recognition
- Natural Language Processing
- Object Tracking
- Pose Estimation

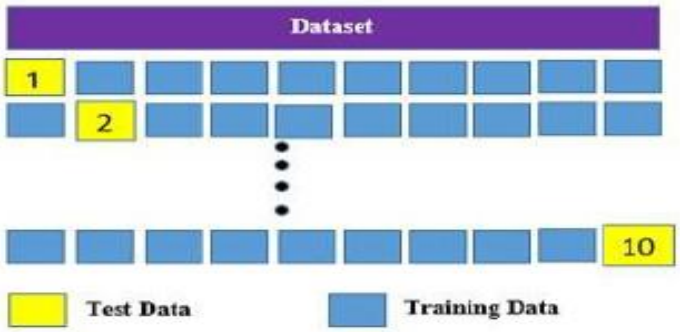
CROSS VALIDATION:

- The term cross-validation is used loosely in literature, where practitioners and researchers sometimes refer to the train/test holdout method as a cross-validation technique.
- It is a crossing over of training and validation stages in successive rounds.
- Main idea behind CV is that each sample in the dataset has the opportunity of being tested.
- The predictive performance of the models is assessed using Cross Validation.
- The motivation to use CV is that when we fit a model, we fit it to a training dataset. Without CV, we just have information on how the models perform to training data.

k-fold Cross Validation:

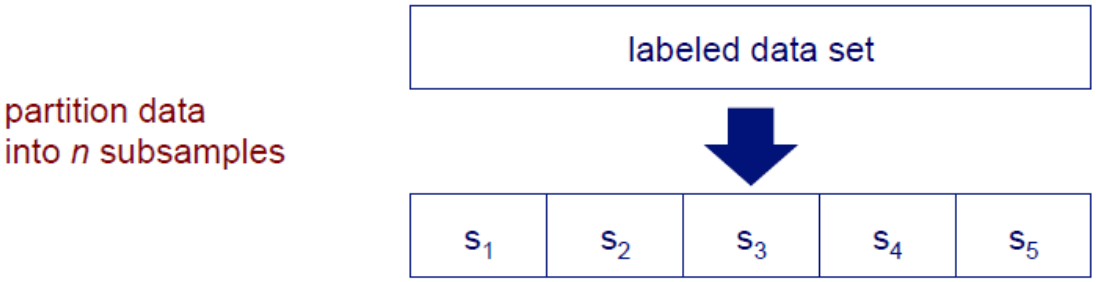
- A method for estimating test error using training data.
- Is a special case of cross-validation where we iterate over a dataset k times.
- In each round, we split the dataset into k parts: one part is used for validation, and the remaining (k-1) parts are merged into a training subset for model evaluation.
- We use a learning algorithm with fixed hyperparameter setting to fit models to the training folds in each iteration when we use the k-fold cross-validation method for model evaluation.
- For example, in 5-fold cross-validation, this procedure will result in five different models fitted; these models were fit to distinctly yet partly overlapping training sets and evaluated on non-overlapping validation sets.
- Eventually, we compute the cross-validation performance as the arithmetic mean over the k performance estimates from the validation sets.
- K-fold cross-validation uses all data for training and testing.
- More commonly used for model selection or algorithm selection rather for model evaluation.
- K-fold CV is less computer intensive than LOOCV.

- A 10-fold CV will be as shown below:



Leave One Out CV (LOOCV):

- Is a good option when working with small sample sizes.
- If we set k = n, that if we set the number of folds as being equal to the number of training instances, we refer to the k-fold cross-validation process as Leave-One-Out Cross-Validation
- In each iteration during LOOCV, we fit a model to n-1 samples of the dataset and evaluate it on the single, remaining data point.
- Although this process is computationally expensive, given that we have n iterations, it can be useful for small datasets, cases where withholding data from the training set would be too wasteful.
- While LOOCV is almost unbiased, one downside of using LOOCV over k-fold cross-validation with k < n is the large variance of the LOOCV estimate.
- LOOCV produces error estimates with high variance given that the testing set at each fold contains only one example.
- The classifier is practically unbiased since each fold trains on almost all the data.
- LOOCV example is shown below:



iteratively leave one subsample out for the test set, train on the rest

iteration	train on	test on
1	s_2 s_3 s_4 s_5	s_1
2	s_1 s_3 s_4 s_5	s_2
3	s_1 s_2 s_4 s_5	s_3
4	s_1 s_2 s_3 s_5	s_4
5	s_1 s_2 s_3 s_4	s_5

Suppose we have 100 instances and we want to estimate accuracy with cross validation:

iteration	train on	test on	correct
1	s_2 s_3 s_4 s_5	s_1	11 / 20
2	s_1 s_3 s_4 s_5	s_2	17 / 20
3	s_1 s_2 s_4 s_5	s_3	16 / 20
4	s_1 s_2 s_3 s_5	s_4	13 / 20
5	s_1 s_2 s_3 s_4	s_5	16 / 20

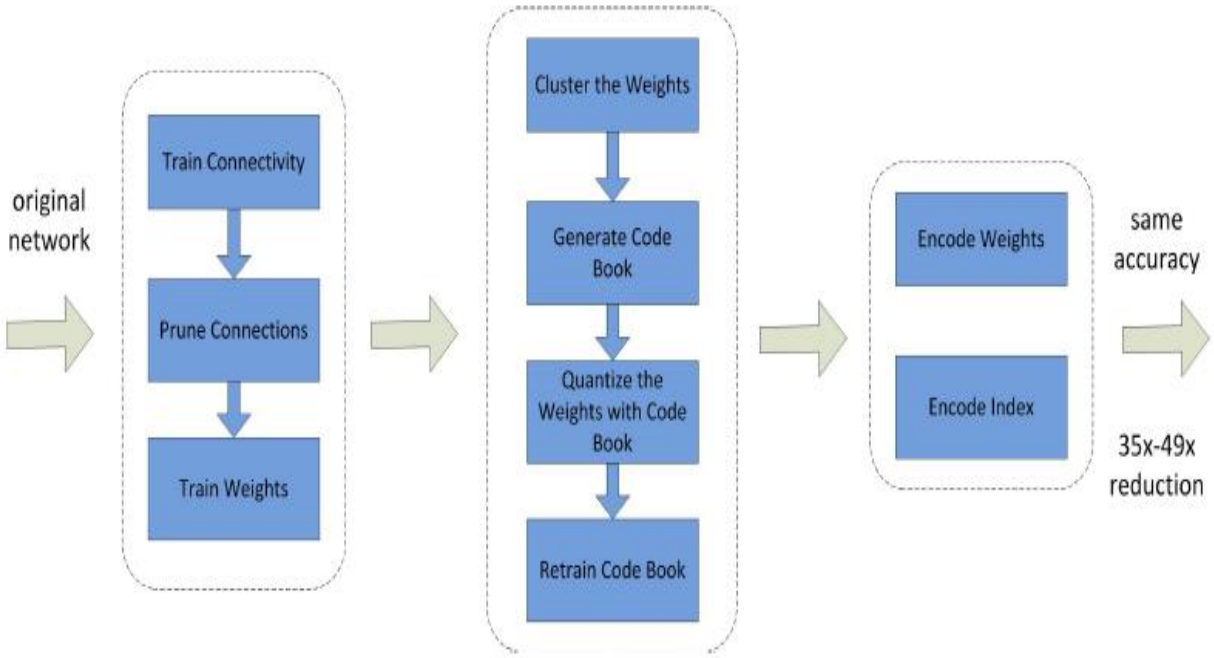
accuracy = 73/100 = 73%

No Free Lunch Theorem:

- Proposed by Wolpert.
- If one algorithm tends to perform better than another on a given class of problems, then the reverse will be true on a different class of problems.
- There is no universally best model.

MODEL COMPRESSION

- Recent deep learning models are becoming more complex. For example, LeNet -5 with 1 M parameters and VGG-16 with 133M parameters.
- The following are the problems with complex deep learning models:
    - Huge storage (memory, disk) requirement
    - Computationally expensive
    - Uses lots of energy
    - Hard to deploy models on small devices (like smart phones)
- The goal of model compression is to make lightweight model that is fast, memory-efficient and energy efficient and especially useful for edge device.
- Key technique to allow using AI everywhere
- Mitigates energy problem.
- There are several ways of model compression – whether training a lightweight model or compressing a trained model. Also applying different techniques.
- Refers to reducing the number of parameters in the convolutional layers and FC layers.
- Approach to reduce the computational cost of CNN.
- The three-stage compression method consisting of pruning, quantization and encoding is shown below. The input is the original model and the output is the compression model.

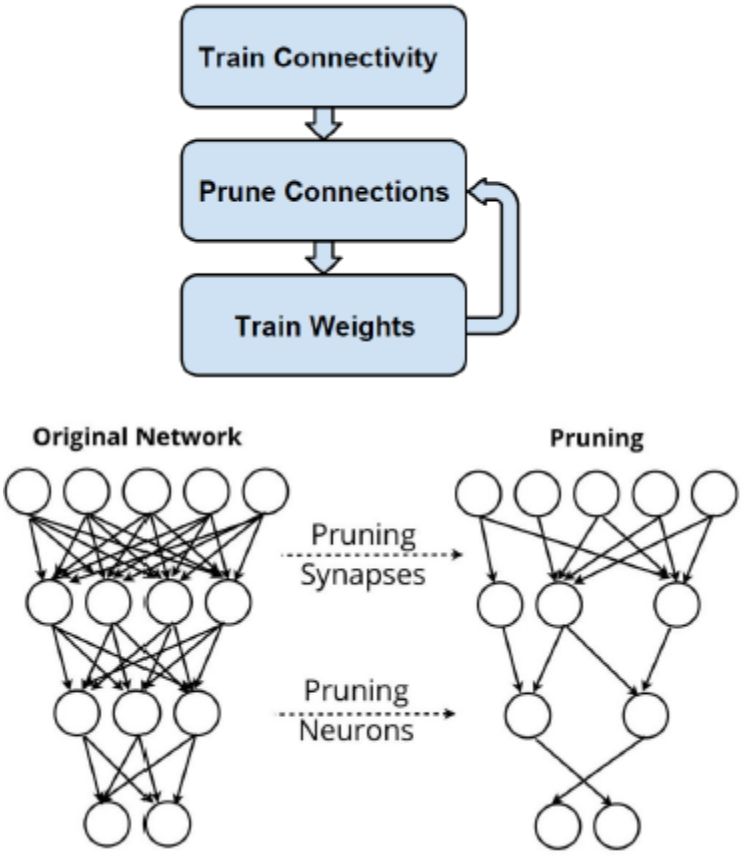


Techniques for Model Compression:

- Pruning
- Weight Sharing
- Quantization
- Low-rank Approximation
- Sparse Regularization
- Distillation
- Hashing

Pruning:

- Reduces the number of parameters and operations in CNNs by permanently dropping less important connections which enables smaller networks to inherit knowledge from the large predecessor networks and maintains comparable of performance.
- Refers to less number of weights
- Pruning works well with quantization and motivated by how real brain learns
- Remove weights which |weight| < threshold and retrain after pruning weights
- Learn effective connections by iterative pruning
- Convolutional layer is more sensitive to pruning compared to fully-connected layer.
- Given a well-trained CNN, prune less important filters together with their connecting feature maps in order to reduce computation cost.
- Minimum weight and smallest activation are the two criteria for pruning CNN.
- Given an RNN, prune weights during initial training in order to gain sparsity of the model.
- Result becomes much better when using both pruning and quantization methods together. Model can be compressed up to 3% of original size.
- Robust to various settings and can achieve good performance.
- Can support both train from scratch and pre-trained model.
- Pruning process is explained as below:

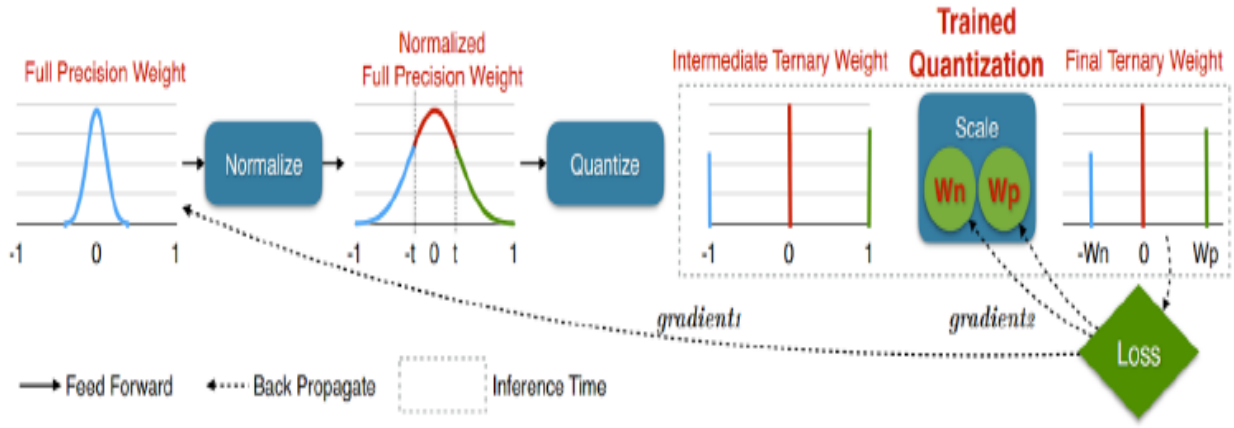


Drawbacks of Pruning:

- Pruning with L1 or L2 regularization requires more iterations to converge than general.
- In addition, all pruning criteria require manual setup of sensitivity for layers which demands fine-tuning of the parameters and could be cumbersome for some applications.

Quantization:

- VQ is a method for compressing densely connected layers to make CNN models smaller.
- Similar to scalar quantization where a large set of numbers is mapped to a smaller set, VQ quantizes groups of numbers together rather than addressing them one at a time.
- Used to significantly reduce the number of dynamic parameters in deep models.
- VQ methods have a clear gain over existing matrix factorization methods.
- Quantization works well on pruned network. For example, unpruned AlexNet has 60 million weights to quantize, while pruned AlexNet has only 6.7 million weights to quantize. Given the same amount of centroids, the latter has less error.
- The main idea is to train the DNN with binary weights during the forward and backward propagations, while retaining precision of the stored weights in which gradients are accumulated in order to regularize all the parameters.
- Network quantization compresses the original network by reducing the number of bits required to represent each weight.
- The following example shows Trained Ternary Quantization (TTQ) in a DNN.

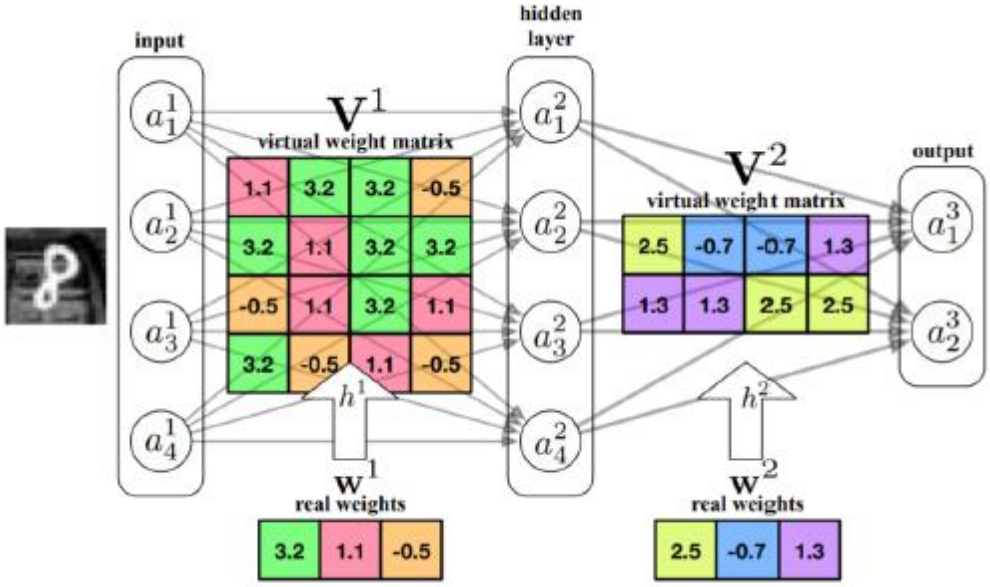


Drawbacks of Quantization:

- The accuracy of the binary nets is significantly lowered when dealing with large CNNs such as GoogleNet.
- Another drawback of such binary nets is that existing binarization schemes are based on simple matrix approximations and ignore the effect of binarization on the accuracy loss.

Hashing:

- Designing a proper hashing technique to accelerate the training of CNNs or save memory space also an interesting problem.
- HashedNets is a recent technique to reduce model sizes by using a hash function to group connection weights into hash buckets and all connections within the same hash bucket share a single parameter value.
- Network shrinks the storage costs of neural networks significantly while mostly preserves the generalization performance in image classification.
- Sparsity will minimize hash collision making feature hashing even more effective.
- HashNets may be used together with pruning to give even better parameter savings.
- Weight sharing is determined by a hash function before the networks see any training data.
- Given a DNN, compress the neural network with weight sharing.
- Hashed Net use a low-cost hash function to randomly group connection weights into hash buckets (weights within the same hash bucket share a same value) in order to reduce the model size.
- An example of Hashed Net is shown below:

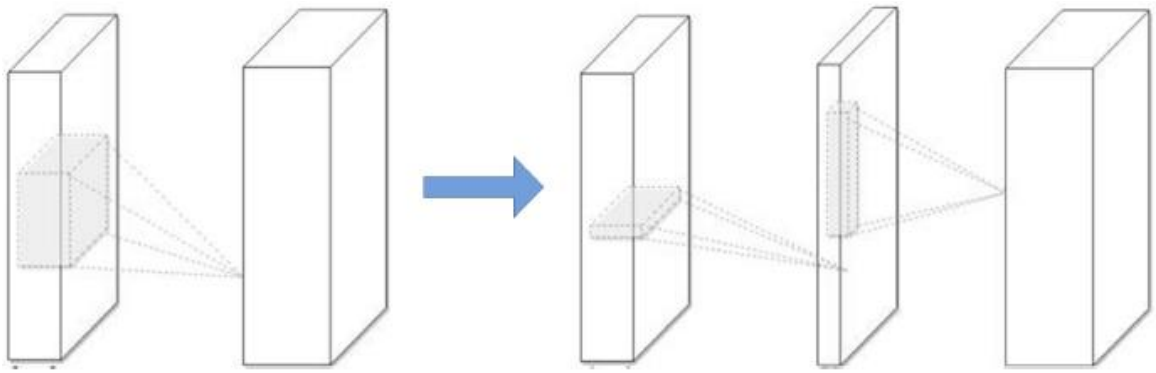


Low-rank Approximation:

- Also known as Low-rank factorization.
- Uses matrix/tensor decomposition to estimate the informative parameters.
- Can be applied to convolutional layer and fully connected layer.
- Standardized pipeline, easy to be implemented.
- Can support both train from scratch and pre-trained model.
- Low-rank approaches are straightforward for model compression and acceleration. The idea complements recent advances in deep learning, such as dropout, rectified units and max out.

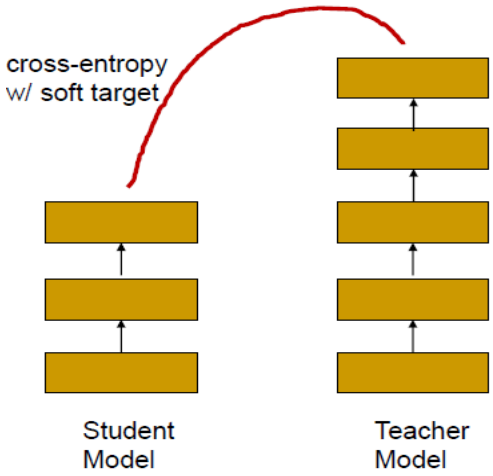
Drawbacks of low-rank approximation:

- The implementation is not that easy since it involves decomposition operation, which is computationally expensive.
- Another issue is that current methods perform low-rank approximation layer by layer and thus cannot perform global parameter compression which is important as different layers hold different information.
- Finally, factorization requires extensive model retraining to achieve convergence when compared to the original model.
- A typical framework for the low rank regularization method is shown below. The left is the original convolutional layer and the right is the low rank constraint convolutional layer with rank k.



Knowledge Distillation:

- Training a compact neural network with distilled knowledge of a large model.
- Can be applied to convolutional layer and fully connected layer.
- Model performances are sensitive to applications and network structure only support train from scratch.
- KD compression framework eased the training of deep networks by following a student-teacher paradigm, in which the student was penalized according to a softened version of the teacher's output.
- KD compresses an ensemble of teacher networks into a student network of similar depth. The student was trained to predict the output and the classification labels.
- Despite its simplicity, KD demonstrates promising results in various image classification tasks.
- KD-based approaches can make deeper models thinner and help significantly reduce the computational cost.
- Based on teacher – student model and shown below:



Drawbacks of KD:

- KD can only be applied to classification tasks with softmax loss function which hinders its usage.
- Another drawback is the model assumptions sometimes are too strict to make the performance competitive with other type of approaches.

LOSS FUNCTION:

- It is important to choose an approximate loss function for a specific task.
- The following are various types of loss functions:
    - Hinge Loss
    - Softmax Loss
    - Contrastive Loss
    - Triplet Loss

Hinge Loss:

- Is usually used to train large margin classifiers such as Support Vector Machine (SVM).

Softmax Loss:

- Is a commonly used loss function which is essentially a combination of multinomial logistic loss and soft max.
- Softmax turns the predictions into non-negative values and normalizes them to get a probability distribution over classes.
- Such probabilistic predictions are used to compute the multinomial logistic loss, i.e., the softmax loss.

Contrastive Loss:

- Is commonly used to train Siamese Network which is a weakly-supervised scheme for learning a similarity measure from pairs of data instances labelled as matching or non-matching.
- A single margin loss function.
- Causes a dramatic drop in retrieval results when fine-tuning the network on all pairs.
- Calculates the similarity between the images.

Triplet Loss:

- Considers three instances per loss function
- The triplet units usually contain an anchor instance as well as a positive instance from the same class and a negative instance.
- Objective of triplet loss is to minimize the distance between the anchor and positive and maximize the distance between the negative and the anchor.
- Triplet loss and its variants have been widely used in various tasks including re-identification, verification and image retrieval.

RECURRENT NEURAL NETWORKS (RNN):

- A great tool for modelling sequential data.
- RNN remembers the analysis done up to a point by maintaining a 'state'
- The 'state' feeds back into the network with each input
- A class of ANN where connections between units form a directed graph along a sequence. This allows it to exhibit dynamic temporal behavior for a time sequence.
- RNNs can use their internal state (memory) to process sequences of inputs. This makes them applicable to tasks such as unsegmented, connected handwriting recognition or speech recognition.
- RNNs remember everything.
- In other neural networks, all the inputs are independent of each other. But in RNN, all the inputs are related to each other.
- RNNs create the networks with loops in them. This loop structure allows the neural network to take the sequence of input.
- RNNs are powerful networks with feedback, with memory and hard to 'fully' understand.

Limitations of RNN:

- Must remember all states at any given time
- Maintaining states is computationally expensive
- Only store states within a time window
- Sensitive to changes in their parameters
- Suffers from Vanishing Gradient problem
- Suffers from Exploding Gradient problem
- Cannot learn long sequences very well
- When dealing with Time Series, RNN tends to forget old information.

Differences between CNN and RNN:

- The main difference between CNN and RNN is the ability to process temporal information or data that comes in sequences such as a sentence for example. CNN and RNN are used for completely different purposes and there are differences in the structures of the neural networks themselves to fit those different use cases.
- CNNs employ filters within convolutional layers to transform data. Whereas RNN reuse activation functions from other data points in the sequence to generate the next output in a series.

LONG SHORT-TERM MEMORY (LSTM):

- Type of RNN to overcome problems with RNN
- LSTM maintains strong gradient over many time steps which means we can train LSTM over long sequences
- By manipulating 3 logic gates recurrent network remembers what it needs in sequence and simply forgets what is not useful
- LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior, not something they struggle to learn.
- LSTMs are essentially the same thing as the RNN, they just have a different way computing the hidden state.

LSTM Architecture:

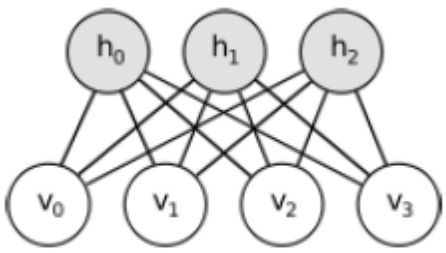
- Consists of a set of recurrently connected subnets, known as memory blocks.
- These blocks can be thought of as a differentiable version of the memory chips in a digital computer.
- Each block contains one or more self-connected memory cells and three multiplicative units that provide continuous analogues of write, read and reset operations for the cells
- 3 logic gates (write, read and forget gates)
- The multiplicative gates allow LSTM memory cells to store and access information over long periods of time, thereby avoiding the vanishing gradient problem.
- The cell state in LSTM is like a conveyor belt. It runs straight down the entire chain, with only some minor linear interaction. It's very easy for information to just flow along it unchanged.
- The LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called gates.
- Gates are a way to optionally let information through. They are composed out of a sigmoid neural net layer and a pointwise multiplication operation.

GATED FEEDBACK RECURRENT NEURAL NETWORKS (GRU):

- Gated Recurrent Unit
- A new type of RNN cell
- Very similar and simpler than LSTM
- It merges the cell state and hidden state
- It combines the forget and input gates into a single 'update gate'
- Computationally more efficient – less parameters, less complex structure
- GRU gaining popularity nowadays
- GRUs don't need cell layer to pass values along like LSTM

Restricted Boltzmann Machine (RBM):

- Boltzmann Machine is a stochastic recurrent neural network with stochastic binary units and undirected edges between units.
- Unfortunately learning for Boltzmann machines is impractical and has a scalability issue and as a result Restricted Boltzmann Machine (RBM) has been introduced.
- RBM has one layer of hidden units and restricts connections between hidden units.
- This allows for more efficient learning algorithm.
- Represented by a bipartite graph, with symmetric, weighted connections.
- One layer has visible nodes and the other hidden variables as shown below:



- We restrict the connectivity to make learning easier.
- In an RBM, the hidden units are conditionally independent given the visible states.

Steps in learning in RBMs:

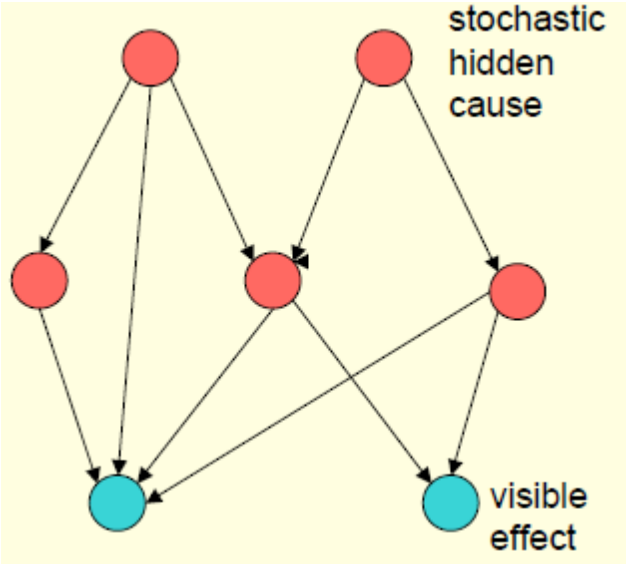
- Start with a training vector on the visible units.
- Update all the hidden units in parallel.
- Update all the visible units in parallel to get a 'reconstruction'.
- Update the hidden units again.

Challenges with RBMs:

- A number of choices to be made:
    - Types of nodes, learning weight, initial values, batch sizes etc.,
    - Care should be taken to avoid over-fitting
    - Lack of ready to go software packages.

Deep Belief Networks (DBN):

- As the name indicates, it is a multi-layer belief network. Each layer is a RBM and they are stacked on each other to construct DBN.
- Consists of two different types of neural networks – Belief Networks and Restricted Boltzmann Machines.
- DBN is unsupervised learning in contrast to perceptron and backpropagation neural networks.
- Multi-layer belief networks.
- Each layer is Restricted Boltzmann Machine and stacked each other to construct DBN.



- DBN is a directed acyclic graph composed of stochastic variables.
- We get to observe some of the variables, and we would like to solve two problems:
    - **The inference problem**: Infer the states of the unobserved variables.
    - **The learning problem**: Adjust the interactions between variables to make the network more likely to generate the observed data.
- Scalability is unique characteristic of DBN.

Training in DBN:

- The first step of training DBN is to learn a layer of features from the visible units, using Contrastive Divergency algorithm.
- The next step is to treat the activations of previously trained features as visible units and learn features of features in a second hidden layer.
- Finally, the whole DBN is trained when the learning for the final hidden layer is achieved.

Learning DBN:

- It is easy to generate an unbiased example at the leaf nodes, so we can see what kinds of data the network believes in.
- It is hard to infer the posterior distribution over all possible configurations of hidden causes.
- It is hard to even get a sample from the posterior.

Applications of DBN:

- Handwritten digit classification
- Learning audio features using 1d/2d convolutional DBNs

AUTOENCODERS

- An autoencoder is a neural network that is trained to copy its input to its output, with the typical purpose of dimension reduction – the process of reducing the number of random variables under consideration.
- An Autoencoder is a kind of unsupervised neural network that is used for dimensionality reduction and feature discovery. Also known as auto-associative learning.
- It features an encoder function to create a hidden layer (or multiple layers) which contains a code to describe the input. Is a neural network which is trained to reproduce its input.
- There is then a decoder which creates a reconstruction of the input from the hidden layer.
- The most intuitive application of autoencoders is data compression. Given a 256 x 256 pixel image for example, a representation of a 28 x 28 pixel may be learned which is easier to handle.
- An autoencoder can then become useful by having a hidden layer smaller than the input layer, forcing it to create a compressed representation of the data in the hidden layer by learning correlations in the data.
- This facilitates classification, visualization, communication and storage of data.
- Autoencoders are a form of **unsupervised learning** meaning that an autoencoder only needs unlabeled data – a set of input data rather than input-output pairs.
- A standard way to train an autoencoder is to ensure that the hidden layer is narrower than the visible layer. This prevents the model from learning the identity function.
- An interesting variation of the MLP whose output is the same as its input.
- The key is to make the hidden layer much smaller than the input and output layers, so the network can't just learn to copy the input to the hidden layer and the hidden layer to the output, in which case we may as well throw the whole thing out. But if the hidden layer is small, something interesting happens: the network is forced to encode the input in fewer bits, so it can be represented in the hidden layer and then decode those bits back to full size.
- It could learn to encode a million-pixel image as just the seven-character word or some short code invented by itself and simultaneously learn to decode the image into another image.

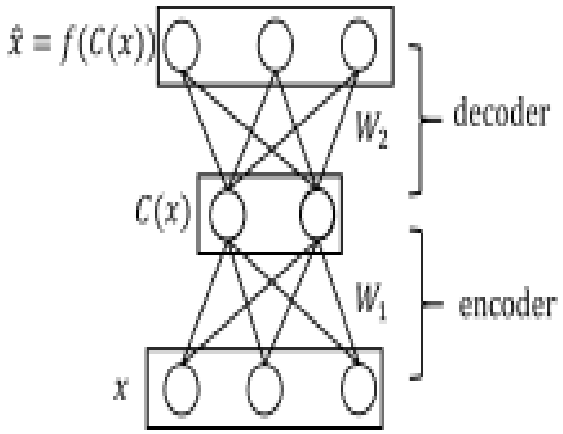
- An autoencoder is like a file compression tool, with two important advantages: it figures out how to compress things on its own and like Hopfield networks it can turn a noisy, distorted image into a nice clean one.
- Autoencoders were very hard to learn in the beginning (1980s) even though they had a single hidden layer. Figuring out how to pack a lot of information into the same few bits is a very difficult problem.
- These were MLPs where the inputs and outputs were clamped together so that the (smaller number of) hidden nodes produced a lower dimensional representation of the inputs.
- Is directional in that the weights run from input to hidden node to output.
- is a feed-forward neural network whose job is to take an input X and predict X.
- To make this non-trivial, we need to add a *bottleneck layer* whose dimension is much smaller than the input.
- Autoencoders have an alternative to manifold learning for conducting nonlinear feature fusion.
- An autoencoder is a type of artificial neural network used to learn efficient data coding in an unsupervised manner.
- Autoencoders analyze nonlinear components unlike PCA and achieves the PCA capacity even without activation functions

Applications of Autoencoders:

- Dimensionality reduction
    - Visualization
    - Feature Extraction
    - High prediction accuracy
    - High speed of prediction
    - Low memory requirements
- Semantic Hashing
- Unsupervised pretraining
- Image denoising
- Anomaly detection
- Feature Learning

Components of an Autoencoder:

- **Encoder –** reduces the input dimensions and compresses the input data into an encoded representation
- **Bottleneck –** contains the compressed representation of the input data in its lowest possible form
- **Decoder –** reconstructs the data from the encoded representation to the original as same as possible
- **Reconstruction Loss –** measures the performance of the decoder and the similarities between the input and the output
- A simple autoencoder is shown below:



Training an Autoencoder:

- To train the model, gradient descent is used
- Use squared loss function
- If input is interpreted as bit vectors or vectors of bit probabilities the cross entropy can be used as a loss function
- A standard way to train an Autoencoder is to ensure that the hidden layer is narrower than the visible layer. This prevents the model from learning the identity function.

Types of Autoencoders:

- There are many varieties of autoencoders:
    - Convolutional autoencoder
    - Denoising autoencoder
    - Deep autoencoder
    - Variational autoencoder
    - Sparse autoencoder

Deep Autoencoders:

- More powerful representations can be learned by using Deep Autoencoders.
- Deep nonlinear autoencoders learn to project the data not onto a subspace, but onto a nonlinear manifold.
- This manifold is the image of the decoder.
- This is a kind of nonlinear dimensionality reduction
- Nonlinear autoencoders can learn more powerful codecs for a given dimensionality, compared with linear autoencoders (PCA).
- Deep autoencoders can learn informative features from raw data. Such features are often used as input to standard supervised learning models.
- Training deep autoencoders with backpropagation does not work well because the gradient signal becomes too small as it passes back through multiple layers and the learning algorithm often gets stuck in poor local minima.
- Deep autoencoders may be trained using layer wise unsupervised pretraining.
- Is usually trained with greedy layer wise pretraining.
- Deep autoencoders also known as 'Stacked Autoencoders'.
- Weights of deep autoencoder should be good initial weights for the classification problem at hand. This is one of the reasons why deep autoencoders work so effectively.

- More power representations can be learned by using Deep Autoencoders.
- Deep Autoencoders can learn informative features from raw data. Such features are often used as input to standard supervised learning methods.

Drawbacks with Deep Autoencoders:

- Training Deep Autoencoder models using back propagation does not work well, because the gradient signal becomes too small as it passes back through multiple layers and the learning algorithm often gets stuck in poor local minima.

Training in Deep Autoencoders:

- One solution to the drawback with Deep Autoencoders as mentioned above, is to greedily train a series of RBMs and to use these to initialize an auto-encoder. The whole system can then be fine-tuned using backprop in the usual fashion.
- This approach works much better than trying to fit the deep autoencoder directly starting with random weights.
- The following are the steps in training a Deep Autoencoder:
    - First, we greedily train some RBMs.
    - Then we construct the Autoencoder by replicating the weights.
    - Finally, we fine-tune the weights using back propagation.

Application of Deep Autoencoders:

- Data Visualization and Feature discovery

<u>Variational Autoencoder (VAE):</u>

- Autoencoders are not a probabilistic model.
- However, there is an autoencoder-like probabilistic model called a 'variational autoencoder'.
- Are generative models that approximate the data distribution P(X) of a high dimensional input X, an image or video.
- Variational approximation of the latent space is achieved using an autoencoder architecture with a probabilistic encoder that produces Gaussian distribution in the latent space and a probabilistic decoder which given a code produces distribution over the input space.
- The motivation behind variational methods is to pick a family of distributions over the latent variables with its own variational parameters and estimate the parameters for this family so that it approaches.
- These are latent-variable models that use a neural network to do approximate inference. The 'recognition network' looks at each datapoint x and outputs an approximate posterior on the latent q (z | x) for that datapoint.
- VAEs are quite popular nowadays. It is a generative model that works well for a dataset with continuous latent variables.
- VAEs are widely used in Natural Language Processing (NLP).
- The following are the pros of Variational autoencoders:
    - Flexible generative model
    - End-to-end gradient training
    - Measurable objective (and lower bound)
    - Fast test-time inference

- The following are the Cons of variational autoencoders:
    - Sub-optimal variational factors
    - Limited approximation to true posterior
    - Can have high-variance gradients
    - Samples tend to have lower quality

Sparse Autoencoder:

- An autoencoder whose training criterion involves a sparsity penalty on the code layer h in addition to the reconstruction error.
- Sparsity is an important concept in Deep Learning and Machine Learning.
- Sparse Autoencoder has the characteristic of getting only a few neurons (say 5%) in any given computation in an encoding layer. This makes it easier for the Sparse Autoencoders to extract features.
- To force sparsity, one introduces sparsity measure and incorporate its minimization in the training time.
- Uses more features where at any given time a significant number of the features will have a 0 value.
- Different bits allowed to represent different inputs; the inputs no longer have to compete to set the same bits. Also, the network has many more parameters so the hyperspace has many more dimensions and you there will be many ways to get out of what would otherwise be local maxima.
- A sparse autoencoder can be implemented as follows:
    - Use more hidden nodes in the encoder
    - Use regularization techniques which encourage sparseness (e.g., a significant portion of nodes have 0 output for any given input)
        - Penalty in the learning function for non-zero nodes
        - Weight decay etc.,

Denoising Autoencoder:

- A stochastic version of the autoencoder.
- In order avoid overfitting in autoencoders, various regularization techniques are applied. One of them is intentionally adding noise, in which case the autoencoder is called as 'Denoising Autoencoder'.
- Stochastically corrupt training instance each time, but sill train autoencoder to decode the uncorrupted instance, forcing it to learn conditional dependencies within the instance
- Better empirical result, handles missing values well
- Unsupervised initialization of layers with an explicit denoising criterion appears to help capture interesting structure in the input distribution.
- This leads to intermediate representations much better suited for subsequent learning tasks such as supervised classification.
- Learns a vector field towards higher probability regions
- Minimizes variational lower bound on a generative model
- Corresponds to regularized score matching on an RBM
- Noise is added to the input layers. For example, we can corrupt some of the inputs, for example by setting them to zero, so the model must learn to predict the missing entries.
- The following are the benefits of denoising autoencoders:
    - Does not learn identity, otherwise it will propagate noise
    - Noise forces autoencoder to learn good model representation of data
        - Density distribution
        - Dependencies between features

SIAMESE NEURAL NETWORKS (SNN)

- Quantifying 'similarity' is an essential component of data analytics.
- Is a special type of neural network architecture. Instead of a model learning to classify its inputs, this neural network learns to differentiate between two inputs. It learns the 'similarity' between them.
- Employ a unique structure to naturally rank similarity between inputs.
- Is a supervised metric based approach.
- Takes in two inputs and outputs a distance metric for the inputs.
- Once a network has been tuned, we can then capitalize on powerful discriminative features to generalize the predictive power of the network not just to new data, but to entirely new classes from unknown distributions.
- Are capable of learning generic image features useful for making predictions about unknown class distributions even when very few examples from these new distributions are available.
- Are easily trained using standard optimization techniques on pairs sampled from the source data
- SNN can be implemented as any kind of neural network, a CNN, an RNN or an MLP.
- Provide a competitive approach that does not rely upon domain-specific knowledge by instead exploiting deep learning techniques.
- First introduced in 1990s to solve signature verification as an image matching problem.
- consists of twin networks which accept distinct inputs but are joined by an energy function at the top. This function computes some metric between the highest-level feature representation on each side. The parameters between the twin networks are tied.
- This strategy has two key properties:
    - It ensures the consistency of its predictions. Weight tying guarantees that two extremely similar images could not possibly be mapped by their respective networks to very different locations in feature space because each network computes the same function.
    - The network is symmetric: if we present two distinct images to the twin networks, the top conjoining layer will compute the same metric as if we were to present the same two images but to the opposite twins.

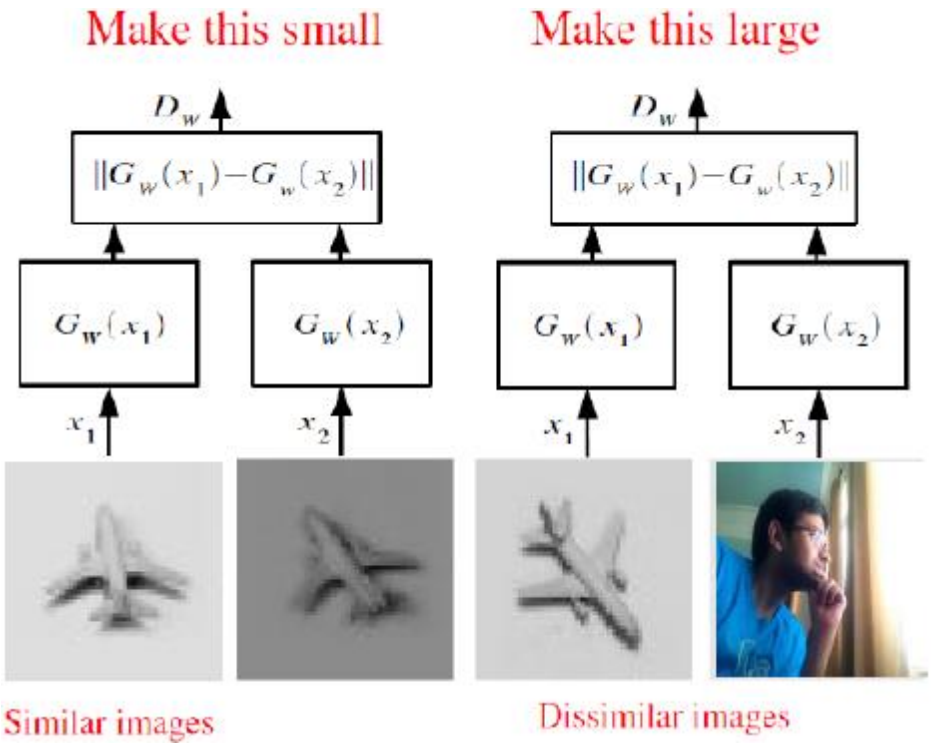
<u>One Shot Classification Model:</u>

- One of the classifications models used for image classification.
- Requires that we have just one training example of each class we want to predict on.
- The model is still trained on several instances, but they only have to be in the similar domain as our training example.

<u>Triplet Loss:</u>

- To train an SNN, we use the triplet loss function.

Siamese CNN – Loss Function:



The final loss is defined as:

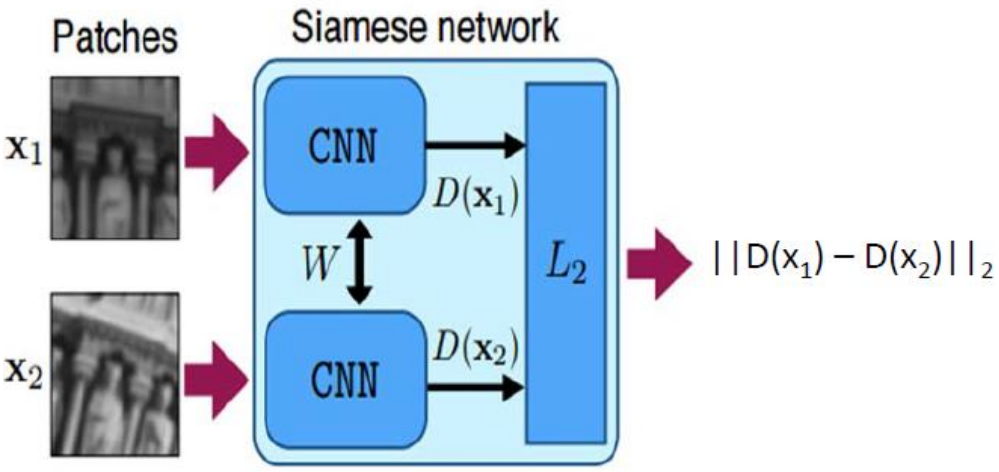
L = ∑ loss of positive pairs + ∑ loss of negative pairs

Contrastive Loss Function:

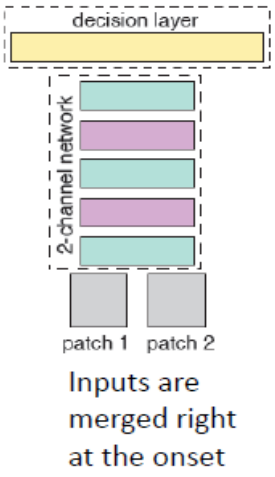
- Calculates the similarity between the images.

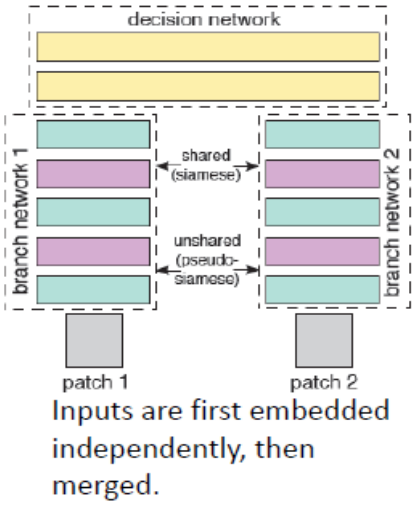
Typical Siamese Neural Network architecture:

- Consists of two identical neural networks (sister networks) each taking one of the two input images.
- The last layers of the two networks are then fed to a 'contrastive loss' function, which calculates the similarity between the images.
- Input: a pair of input signatures
- Output (Target): A label, 0 for similar, 1 else.



- No one 'architecture' fits all.
- Design largely governed by what performs well empirically on the task at hand.





Learning Distance is crucial for:

- Matching
- Retrieval
- Recognition
- Re-identification

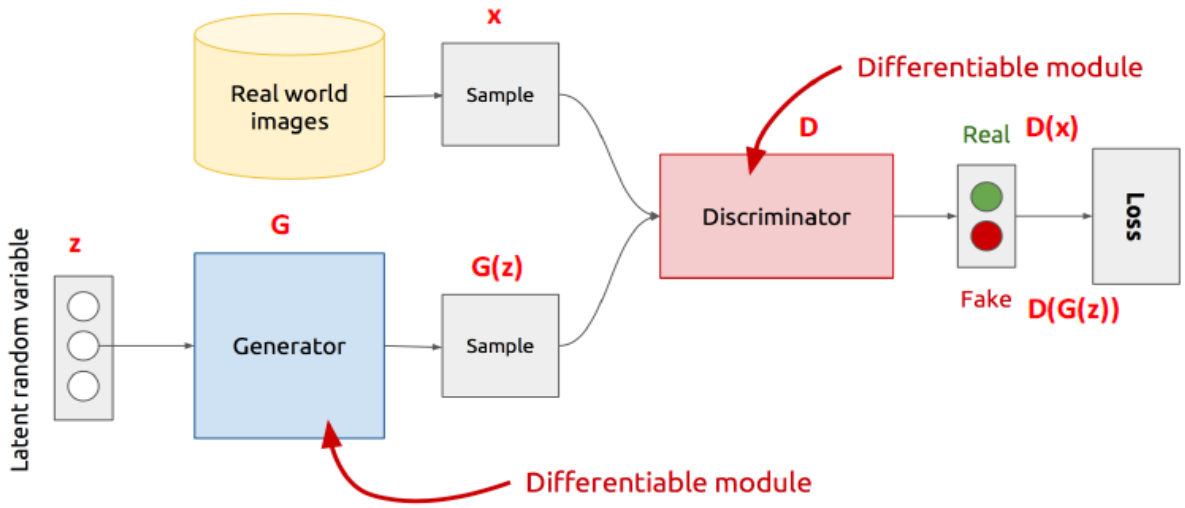
<u>Applications of Siamese Networks:</u>

- Generating invariant and robust descriptors
  - Discriminative Descriptors for local patches
  - Deep Descriptor
- Person re-identification
- Rendering a street from different viewpoints
- Newer nets for Person Re-Id, Viewpoint Invariance and Multimodal Data
- Sentence Matching

GENERATIVE ADVERSARIAL NETWORKS (GAN)

- GANs are a class of neural networks used in unsupervised learning.
- They are implemented as a system of two neural networks contesting with each other in a zero-sum game setting.
- The most popular application of GANs is to learn to generate photographs that look authentic to human observers.
- The first of the two networks take a random input (typically Gaussian noise) and learns to generate an image as a matrix of pixels.
- The second network takes as input two images: one 'real' image from some collection of images as well as the image generated by the first network.
- The second network has to learn to recognize which one of the two images was generated by the first network.
- The first network gets a negative loss if the second network recognizes the 'fake' image.
- The second network on the other hand gets penalized if it fails to recognize which one of the two images is fake.
- Generative – learn a generative model - given training data, generate new samples from the same distribution
- Adversarial – Trained in an adversarial setting
- Networks – Use Deep Neural Networks
- Discriminative models have several key limitations:
    - Can't model P(X), i.e. the probability of seeing a certain image
    - Thus, can't sample from P(X), i.e., can't generate new images
- Generative models in general cope with all of above:
    - They can model
    - They can generate new images
- Use a latent code
- Are unstable to train
- Often regarded as producing the best samples
- Are generative models that use supervised learning to approximate an intractable cost function
- Can simulate many cost functions, including the one used for maximum likelihood
- Are unstable to train and need many tricks to converge. Reaching Nash equilibrium is an important open research question.
- A game between two players:
    - Discriminator D
    - Generator G.
- D tries to discriminate between:
    - A sample from the data distribution
    - A sample from the generator G
- G tries to 'trick' D by generating samples that are hard for D to distinguish from data.
- Deep learning models in general involve a single player, GANs instead involve two or more players.

GAN Architecture:



Advantages of GANs:

- Plenty of existing work on Deep Generative Models
  - Boltzmann Machine
  - Deep Belief Nets
  - Variational Auto Encoders (VAE)
- Sampling (or generation) is straightforward.
- Training doesn't involve Maximum Likelihood estimation.
- Robust to Overfitting since Generator never sees the training data.
- Empirically, GANs are good at capturing the modes of the distribution.

Problems with GANs:

- Probability Distribution is implicit.
  - Not straightforward to compute P(X).
  - Thus, Vanilla GANs are only good for Sampling/Generation.
- Training is hard
  - Non-convergence (we might not converge to Nash Equilibrium at all)
  - Mode-Collapse (Generator fails to output diverse samples)

Applications of conditional GANs (CGAN):

- Image to Image Translation
- Text to Image Synthesis
- Face Aging

Motivation for studying generative models:

- Excellent test of our ability to use high dimensional, complicated probability distributions
- Simulate possible futures for planning or simulated RL
- Missing data
- Semi-supervised learning
- Multi-modal outputs
- Realistic generation tasks

Loss functions in GANs:

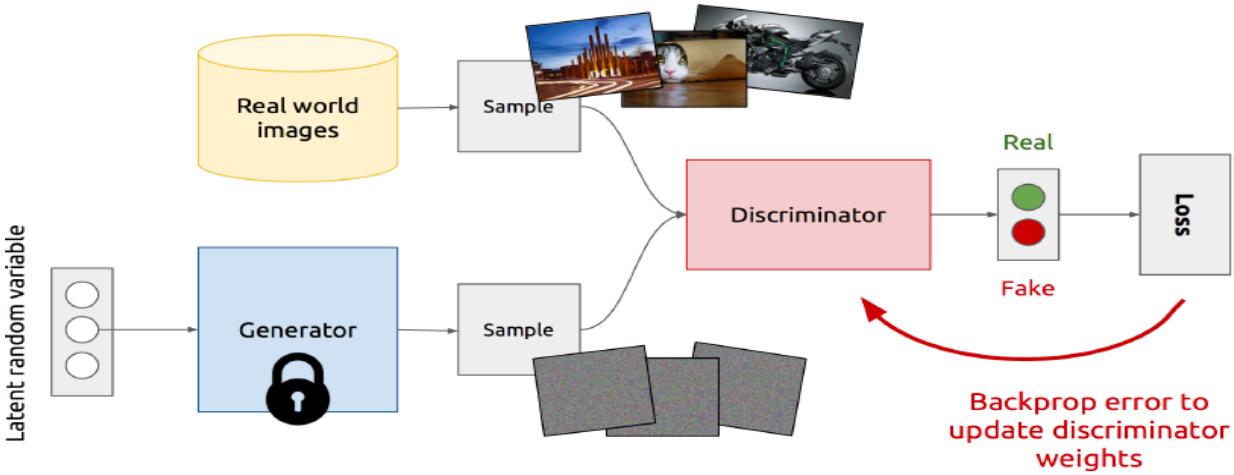
There are two approaches:

- Increase log-likelihood of data (ML)
- Have your network learn its loss function (Adversarial Learning)

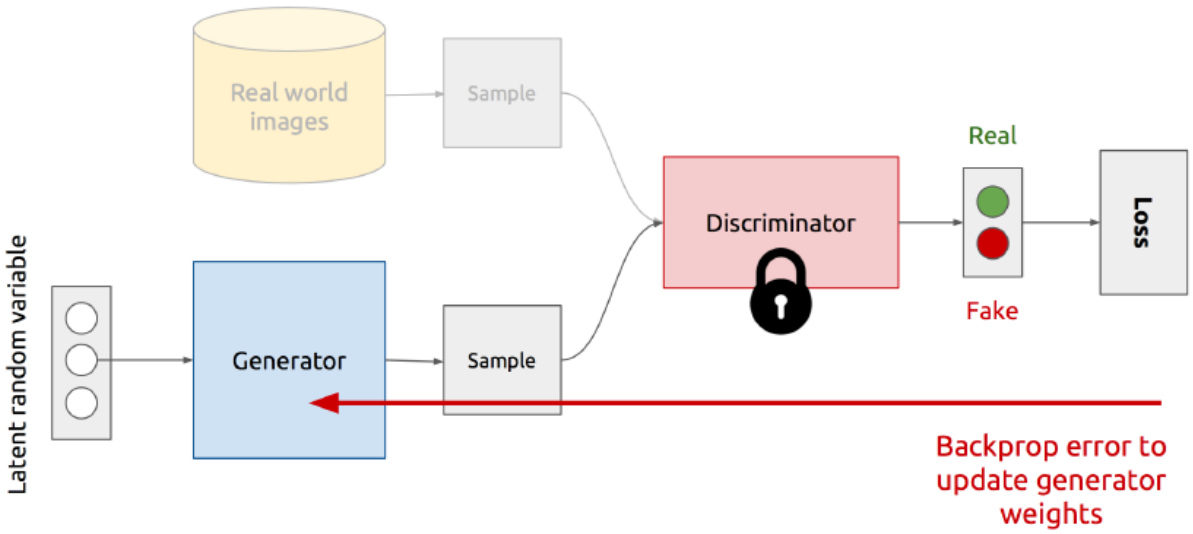
Training Procedures in GANs:

- Use SGD-like algorithm of choice (Adam) on two minibatches simultaneously:
  - A minibatch of training examples
  - A minibatch of generated samples
- Optional: run k steps of one player for every step of the other player.

Training Discriminator:



Training Generator:



Types of Regression Algorithms:

- Linear Regression
- Multiple Linear Regression
- Polynomial Regression
- Ridge Regression
- Lasso Regression
- ElasticNet Regression

Linear Regression:

- Is a popular regression learning algorithm that learns a model which is a linear combination of features of the input example.
- Is a statistical model used to predict the relationship between independent and dependent variables denoted by x and y respectively.
- Can be described as the 'best fit' line through all data points.
- Predictions in linear regression are numerical.

Linear Regression can be examined by below two factors:

- How closely are x and y are related?
  - Linear Regression gives a number between -1 and 1 indicating the strength of correlation between the two variables
  - 0 means no correlation, -1 negatively correlated and 1 is positively correlated
- Prediction
  - When relationship between x and y is known, use this to predict future values of y for a value of x
  - This is done by fitting a regression line and is represented by a linear equation:

Y = a * x + b; where y is called criterion and b is called as predictor.

Advantages of Linear Regression Algorithm:

- Easy to understand
- We can clearly see what the biggest drivers of the model are.

Disadvantages of Linear Regression Algorithm:

- Sometimes too simple to capture complex relationships between variables.
- Works poorly with correlated features.

Multiple Linear Regression:

- It is used to predict the outcome of a response variable through several explanatory variables and to model the relationships between them.
- This can be shown as the following equation:

Y = m1*x1 + m2*x2 + m3*x3 + ……..+ mn*xn + c

Where y = dependent variable

M1, m2, m3 etc., = slopes

C = coefficient

Polynomial Regression:

- The relationship between the dependent variable y and the independent variable x is modeled as an nth degree polynomial in x.

Ridge Regression:

- Is a technique for analyzing multiple regression data that suffer from multicollinearity.
- By adding a degree of bias to the regression estimates, ridge regression reduces the standard errors.
- Works better statistically and also easier to fit numerically.
- Uses a 'soft' weighting of all the dimensions.
- Ridge regression shrinks the regression coefficients by imposing a penalty on their size. The ridge coefficients minimize a penalized residual sum of squares.
- Ridge regression can have better prediction error than linear regression in a variety of scenarios, depending on the choice of lambda.
- The first step in Ridge regression is to standardize the variables (both dependent and independent) by subtracting their means and dividing by their standard deviations.
- All ridge regression calculations are based on standardized variables.

Lasso Regression:

- LASSO – **L**east **A**bsolute **S**hrinkage and **S**election **O**perator.
- Lasso Regression (L1) is like Ridge Regression, but it also performs feature selection.
- It uses shrinkage, where data values are shrunk toward a central point like the mean.
- It tends to exclude variables that are not required from the equation whereas ridge tends to do better when all the variables are present.
- It is well suited for models showing high levels of multicollinearity or when you want to automate certain parts of model selection.

ElasticNet Regression:

- ElasticNet regression combines the strength of Lasso and Ridge Regression.
- It is a convex combination of Ridge and the LASSO.
- If you are not sure whether to use Lasso or Ridge, use ElasticNet.

Accuracy Metrics of Regression models:

- R-square is the most common metric to judge the performance of regression models.
- R-square lies between 0-100%
- The disadvantage with R-squared is that it assumes every independent variable in the model causes variations in the dependent variable. This can be solved using Adjusted R squared, which is adjusted for the number of predictors in the model.

Cost Function and Mean-Squared Error (MSE):

- MSE is used to measure the performance of a model.
- RMSE is the square root of MSE.
- It is the average distance of a data point from the fitted line measured along a vertical line.
- These functions are referred to as the loss function or cost function and the value has to be minimized.

UNSUPERVISED LEARNING

- Also known as 'Descriptive Learning'.
- Learns from an unlabeled dataset
- Uses input data to train the model
- Is expected to find patterns and anomalies
- By unsupervised, it means that we are not trying to predict a variable; instead, we want to discover hidden patterns within our data that will let us to identify groups, or clusters, within that data.

Methods:

- K-Means Clustering
- Hierarchical Clustering
- DBSCAN
- Gaussian mixtures
- Spectral Clustering

Steps in Unsupervised learning:

1. Input unlabeled data – for example provide images of different kinds of fruits without expected output
2. Test the model – in this step the model identifies patterns like shape, color, size and groups the fruits based on these features, attributes or qualities.

Clustering:

- Is the process of grouping data into classes or clusters
- Grouping is done in such a manner that the objectives within the same cluster are very similar to each other, but they are very dissimilar to the objects in some other cluster
- Clustering is a form of 'learning by observations'.
- an unsupervised learning method & does not require a training data set to generate a model
- Can lead to the discovery of previously unknown groups within the data
- Useful for transforming diverse and varied data to much smaller number of groups.
- Results in meaningful and actionable data structures that reduce complexity and provide insight into patterns of relationships.
- Clustering is an unsupervised identification of natural groups in data

Applications of clustering:

- Segmenting customers into groups with similar buying patterns for targeted marketing campaigns.
- Detecting anomalous behavior such as unauthorized network intrusions by identifying patterns of use falling outside the known clusters.
- Simplifying extremely large datasets by grouping features with similar values into a smaller number of homogeneous categories.
- Clustering is often used in marketing in order to group users according to multiple characteristics, such as location, purchasing behavior, age, and gender.
- Also used in scientific research for example to find population clusters within DNA data.

k-Means Clustering:

- Used with unlabeled data (i.e., data without defined categories or groups)
- One of the popular examples of clustering
- Is a way to find clusters or groups in the data
- Is a set of steps that work iteratively to find the groups and label the data
- 'k' is a variable that represents the number of groups
- We need to run the k-Means clustering algorithm for a range of k values and compare the results to find the value of k that best represents the number of clusters in the data
- Algorithm works iteratively to assign each data point to one of k groups based on the features that are provided.
- Data points are clustered based on feature similarity
- The results of the k-means clustering algorithm are the centroids of the k clusters which can be used to label new data
- Uses iterative refinement to produce a result.
- The algorithm inputs are the number of clusters k and the data set
- Data set is a collection of features for each data point.
- Resulting clusters are always convex sets.

The algorithm then iterates between two steps:

1. Data assignment step
2. Centroid update step

In data assignment step, each centroid defines one of the clusters. Each data point is assigned to its nearest centroid, based on the squared Euclidean distance.

In centroid update step, the centroids are computed. This is done by taking the mean of all data points assigned to that centroid's cluster.

Steps in k-Means algorithm:

- Select an initial partition of k clusters (which can be randomly generated or randomly selected from the data set)
- Assign each object to the cluster with the closest center (mean of all data points)
- Compute the new centers of the clusters (mean of all data points)
- Repeat step 2 and 3 until no object changes cluster

Hierarchical Clustering (aka Hierarchical Cluster Analysis (HCA)):

- Is a method which seeks to build a hierarchy of clusters
- A nested tree of partitions is created
- Finds successive clusters using previously established clusters

Dendrogram:

- The most natural representation of HCA is a corresponding tree, called as 'Dendrogram' which shows the samples are grouped.

Strategies for HCA:

- Agglomerative
- Divisive

Agglomerative is a bottom-up approach. Each observation starts in its own cluster and pairs of clusters are merged as one moves up the hierarchy. This strategy starts with each element in a separate cluster and merges them accordingly to a given property. Divisive is a top down approach. All observations start in one cluster and splits are performed recursively as one moves down the hierarchy.

- To decide which clusters should be combined (for agglomerative) or where a cluster should be split (for divisive) a measure of dissimilarity between sets of observations is required.
- This is achieved by use of an appropriate metric (a measure of distance between pairs of observations), and a linkage criterion which specifies the dissimilarity of sets as a function of the pairwise distances of observations in the sets.
- In single linkage hierarchical clustering, the distance between two clusters is defined as the shortest distance between two points in each cluster. In complete linkage hierarchical clustering the distance between two clusters is defined as the longest distance between two points in each cluster.

DBSCAN Algorithm:

- Simple clustering algorithm.

Steps in DBSCAN:

- Select a distance for your radius and you select a point within your dataset – then all other data points within the radius's distance from your initial point are added to the cluster.
- Repeat the process for each new point added to the cluster and repeat until no new points within the radii of the most recently added points.
- Choose another point within the dataset and build another cluster using the same approach.

Advantages of DBSCAN:

- Is intuitive.

Disadvantages of DBSCAN:

- Its effectiveness and output rely heavily on what you choose for a radius.
- Won't react well to certain types of distributions.

Gaussian mixtures:

- Gaussian Mixture Models (GMMs) have been until very recently regarded as the most powerful model for estimating the probabilistic distribution of speech signals associated with each of Hidden Markov Model (HMM) states.
- GMM is a density model where we combine a finite number of Gaussian distributions.

Spectral Clustering:

- Spectral clustering is based on 'connectivity' criteria in Clustering.
- Algorithms that cluster points using eigenvectors of matrices derived from the data.
- Useful in hard non-convex clustering problems.
- Obtain data representation in the low-dimensional space that can be easily clustered.
- Variety of methods that use eigenvectors of unnormalized of normalized Laplacian, differ in how to derive clusters from eigenvectors, k-way vs repeated 2-way.
- Empirically very successful.

- One of the popular clustering algorithms.
- Simple to implement, can be solved efficiently by standard linear algebra software, and very often outperforms traditional clustering algorithms such as k-means algorithm.
- The success of spectral clustering is mainly attributed to the reason that it does not make any assumptions on the form of the clusters.
- As opposed to k-means where the resulting clusters are always convex sets, spectral clustering can solve very general problems like intertwined spirals.
- Can be implemented efficiently even for large data sets as long as we make sure that he similarity graph is sparse.
- Can be considered as a power tool which can produce extremely good results if applied with care.

Clustering Vs. Classification: The following table shows the differences between Clustering and Classification.

Clustering	Classification
• Unsupervised	• Supervised
• Uses unlabeled data	• Uses labeled data
• Organize patterns w.r.t. an optimization criteria	• Requires training phase
• Requires a definition of similarity	• Domain sensitive
• Hard to evaluate	• Easy to evaluate (you know the correct answer)
• Examples: K-means, Fuzzy C-means, Hierarchical Clustering, DBScan	• Examples: Naive Bayes, KNN, SVM, Decision Trees, Random Forests

DIMENSIONALITY REDUCTION

- The process of converting a set of data having large number of dimensions into data with smaller number of dimensions is called Dimensionality Reduction.
- When dealing with high dimensional data, it is often useful to reduce the dimensionality by projecting the data to a lower dimensional subspace which captures the 'essence' of the data. This is called 'Dimensionality Reduction'.
- The motivation behind this technique is that although the data may appear high dimensional, there may only be a small number of degrees of variability, corresponding to latent factors. For example, when modeling the appearance of face images, there may only be a few underlying latent factors which describe most of the variability, such as lighting, pose, identity etc.
- The following are the benefits of dimensionality reduction:
    - Compresses the Data and reduces the storage space required
    - Requires lesser computation time
    - Removes redundant features
    - Reduces the dimensions of data to 2D or 3Dmay allow us to plot and visualize it precisely
    - Potentially reduces the noise
- There are two types of dimensionality reduction:
    - Feature Extraction – finds new features in the data after it has been transformed from a high dimensional space to a low dimensional space
    - Feature Selection – finds the most relevant features to a problem. This is done by obtaining a subset or key features of the original variables.

Data Visualization:

- Two sides of data visualization are:
    - Data Exploration: making sure you understand your data.
    - Data communication: making sure that others understand your insights and/or can use your data easily.

Assumptions of Dimensionality Reduction:

- High-dimensional data often lives in a lower dimensional manifold (sub-space).
- We can represent the data points well by using just their lower dimensional coordinates.
- The lower dimensional data will capture the distribution of (pairwise distances between) points in high dimensions.
- If the manifold is a linear subspace, we can use PCA.

FEATURE ENGINEERING

Feature Selection:

- Is used to select those features that contribute most to the prediction variable that we are interested in
- Benefits of feature selection are:
  - Reduces overfitting by making data less redundant
  - Reduces training time by eliminating the misleading data
  - Improves accuracy by collating fewer data points
- Statistical relationship among variables depend on regression and correlation
- Two techniques used for feature selection are:
  - Regression
  - Factor Analysis
- Until the late 2000s, the broader class of systems that fell into the category 'machine learning' relied heavily on feature engineering.
- Features are transformations of input data resulting in numerical features that facilitate a downstream algorithm such as a classifier to produce correct outcomes on new data.
- Feature engineering aims to take the original data and come up with representations of the same data that can be fed to an algorithm to solve a problem.

REGRESSION:

- Tells the relationship between variables
- Quantifies the relationship by using the set of equations
- Determines how an independent variable predicts the values of the dependent variable

MULTICOLLINEARITY PROBLEM:

- In multicollinear data, interdependence among explanatory variables may lead to an unreliable model.
- We perform Factor Analysis to identify the underlying causes for this behavior.
- When multicollinearity occurs, least squares estimates are unbiased, but their variances are large so they may be far from the true value.
- Multicollinearity is the existence of near-linear relationships among the independent variables.
- One of the first steps in a regression analysis is to determine if multicollinearity is a problem.
- The following are the effects of Multicollinearity:
  - can create inaccurate estimates of the regression coefficients
  - inflate the standard errors of the of the regression coefficients.
  - Deflate the partial t-tests for the regression coefficients,
  - Give false, nonsignificant, p-values and degrade the predictability of the model
- The source of the multicollinearity impacts the analysis, the corrections and interpretation of the linear model.
- The sources of Multicollinearity are:
  - Data collection
  - Physical constraints
  - Over-defined model
  - Model choice or specification
  - Outliers

FACTOR ANALYSIS:

- Reduces many variables into fewer numbers of factors
- Puts the maximum common variance into a common score
- Associates multiple observed variables with a latent variable
- Has the same number of factors and variables where each factor contains a certain amount of the overall variance

Eigenvalue:

- A measure of the variance that a factor explains for the observed variables. A factor with an eigenvalue less than 1 explains less variance than a single observed value

Factor Analysis process:

- Analyzes the underlying causes that can affect the output
- There are two algorithms:
    - Principal Component Analysis (PCA)
    - Linear Discriminant Analysis (LDA)

PRINCIPAL COMPONENT ANALYSIS (PCA) ALGORITHM

Principal Component Analysis (PCA):

- Extracts hidden factors from the dataset
- One of the most popular technique for dimensionality reduction, data compression and data visualization.
- Also used for the identification of simple patterns, latent factor and structures of high-dimensional data.
- Is unsupervised (does not use label information)
- Input to the PCA is a set of samples x1, x2……, xn each in d dimension. The objective is to get a set of new samples z1, z2……,zn each in k dimension, where k < d
- The central piece of PCA is covariance matrix. If data was scalar, mean of standard deviation could have helped to capture the distribution of the data.
- But in this case data is a vector x. To model the distribution of the data, we need a matrix, i.e., covariance matrix.
- Defines data using a smaller number of components, explaining the variance in your data
- PCA reduces the computational complexity
- Determines whether a new data point is part of the group of data points from the training set
- PCA is based on feature extraction
- Refers to a very specific technique for finding a linear transformation (a matrix) that transforms a data set to a lower dimension.
- If the training data set has vectors (data points) of dimension N, the PCA transformation matrix will have dimension M by N and will transform each vector to a lower dimension M by a simple matrix multiply.
- PCA is a variance-maximizing technique that projects data onto a direction that maximizes variance.
- PCA performs linear mapping of the original data to a lower dimensional space such that the variance of the data in the low dimensional representation is maximized.

- PCA is performed by carrying out the eigen-decomposition of the covariance matrix. The result would be a set of eigenvectors and a set of eigenvalues which can then be used to describe the original data.
- Steps involved in PCA:
    - Constructing a co-variance matrix
    - Performing an eigen-decomposition of that matrix to obtain a set of eigenvectors (W).
    - Columns of W are ordered by the size of their corresponding eigenvalues
    - Choose the first n columns of W and use it to describe your data
- Direction of maximum variance:
    - PCA seeks a linear combination of variables in order to extract maximum variance
    - Compute eigenvectors that are the principal components of a dataset and collect them in a projection matrix
    - Each of these eigenvectors is associated with an eigenvalue which is the magnitude
    - Reduce the dataset into a smaller dimensional subspace by dropping the less informative eigenpairs
- PCA finds the best line depending on two criteria:
    - The variation of values should be maximal along this line
    - The error should be minimum if you reconstruct the original two positions of a blue dot from the new position of a red dot
- PCA can be appreciated in the following ways:
    - Minimizing Variance
    - Minimizing Reconstruction Loss
    - Line Fitting that minimize orthogonal distance

Applications of PCA:

- In Biology – PCA is used to interpret gene microarray data, to account for the fact that each measurement is usually the result of many genes which are correlated in their behavior by the fact that they belong to different biological pathways.
- In NLP – a variant of PCA called 'latent semantic analysis' is used for document retrieval in Natural Language Processing.
- In Signal processing – variant of PCA, Independent Component Analysis (ICA) is used to separate signals into their different sources.
- In Computer Graphics – it is common to project motion capture data to a low dimensional space and use it to create animations.

Drawbacks of PCA:

- World is often non-linear.
- Consider Swiss roll dataset, what PCA will give and what we want.

Linear Discriminant Analysis (LDA):

- Reduces dimensions
- Searches the linear combination of variables that best separates two classes
- Reduces the degree of overfitting
- Determines how to classify a new observation out of a group of classes
- The decision boundary between any two classes is a straight line.

LOCALLY LINEAR EMBEDDING (LLE) ALGORITHM

- The idea is to preserve the structure of local neighborhood
- Most of the data is close to the center of the space
- A few points are far from the center to satisfy the unit variance constraint.

Steps in LLE:

- Represent each point as a weighted combination of its neighbors in High Dimensionality. Remember the Wij
- Find a lower dimensional representation that minimize the representation error.

ISOMETRIC MAPPING (ISOMAP) ALGORITHM

- Is a nonlinear dimensionality reduction method and derived from **Iso**metric feature **Map**ping
- Connects each data point in a high dimensional space (a face for example) to all nearby points (very similar faces), computes the shortest distances between all pairs of points along the resulting network and finds the reduced coordinates that best approximate these distances.
- In contrast to PCA, faces coordinates in this space are often quite meaningful: one may represent which direction the face is facing (left profile, three quarters, head on etc.,); another how face looks (very sad, a little sad, neutral, happy, very happy etc.,); and so on.
- From understanding motion in video to detecting emotion in speech, IsoMap has a surprising ability to zero in on the most important dimensions of complex data.
- Maps points on a high-dimensional non-linear manifold to a lower dimensional set of coordinates.
- Is a multi-dimensional scaling (MDS) method that use geodesic to measure distance so it can capture manifold structure.
- Geodesic - In differential geometry, a geodesic is a curve representing in some sense the shortest path between two points in a surface, or more generally in a Riemannian manifold. It is a generalization of the notion of a "straight line" to a more general setting.

IsoMap algorithm:

- Find the neighbors of each point
  - Points within a fixed radius
  - K nearest neighbors
- Construct a graph with those nodes
- Compute neighboring edge weights (distance between neighbors)
- Compute weighted shortest path between all points
- Run MDS using the computed distance above

Limitations with IsoMap:

- The connectivity of each data point in the neighborhood graph is defined as its nearest k Euclidean neighbors in the high-dimensional space. This step is vulnerable to 'short-circuit errors' if k is too large with respect to the manifold structure or if noise in the data moves the points slightly off the manifold.
- Even a single short-circuit error can alter many entries in the geodesic distance matrix which in turn can lead to a drastically different and incorrect low-dimensional embedding.
- Conversely, if k is too small, the neighborhood graph may become too sparse to approximate geodesic accurately.

t-distribution STOCHASTIC NEIGHBORHOOD EMBEDDING (t-SNE) ALGORITHM

- Stands for t distribution Stochastic Neighborhood Embedding.
- Popular dimensionality reduction algorithm.
- Visualizes high-dimensional data by giving each datapoint a location in a two or three-dimensional map.
- Is a variation of Stochastic Neighborhood Embedding (SNE)
- t-SNE is better than existing techniques at creating a single map that reveals structure at many different scales.
- Fairly complex non-linear technique
- Uses an adaptive sense of 'distance'. Translates well between geometry of high and low dimensional space
- Has become a standard tool
- Idea in SNE and t-SNE is simple – instead of distance think about probabilities. Pij as the probability of j in the neighborhood of i.
- For each point we have now a probability vector of size N.
    - SNE uses Gaussian. T-SNE uses another t-distribution with 1 degree of freedom.
- We want these probability vectors to be the same in low dimensional
- Optimize using gradient descent.
- t-SNE is unsupervised.
- Classes are much better separated
- Efficient approximations exist
- Most popular lower dimensional visualization at the moment.

t-SNE converts the higher dimensional data into the lower dimensional data by following steps:

- It measures the similarity between the two data points and it does for every pair. Similar data points will have more value of similarity and the different data point have less value.
- Then it converts that similarity distance to probability (joint probability) according to normal distribution.
- It does the similarity check for every point. Thus it will have the similarity matrix 'S1' for every point. This is all calculation it does for our data points that lie in higher dimensional space.
- Now, t-SNE arranges all of the data points randomly on the required lower dimensional
- And it does all of the same calculation for lower dimensional data points as it does for higher ones – calculating similarity distance but with a major difference it assigns probability according to t-distribution instead of normal distribution and this is because it is called t-SNE, not simple SNE.
- Now we also have the similarity matrix for lower dimensional data points. Let's call it S2.
- Now what t-SNE does is it compares matrix S1 and S2 and tries to make the difference in between matrix S1 and S2 much more smaller by doing some complex mathematics.
- At the end we will have lower dimensional data points which tries to capture even complex relationships at which PCA fails.
- So on a very high level this is how t-SNE works.
- The visualizations produced by t-SNE are significantly better than those produced by the other techniques on almost all of the datasets.
- t-SNE use random walks on neighborhood graphs to allow the implicit structure of all of the data to influence the way in which a subset of the data is displayed.

- t-SNE is better than existing techniques at creating a single map that reveals structure at many different scales.
- t-SNE is capable of retaining local structure of the data while also revealing some important global structure of the data (such as clusters at multiple scales).
- Both the computational and memory complexity of t-SNE are O(n^2)

Drawbacks of t-SNE:

- It is unclear how t-SNE performs on general dimensionality reduction tasks
- The relatively local nature of t-SNE makes it sensitive to the curse of the intrinsic dimensionality of the data
- t-SNE is not guaranteed to converge to a global optimum of its cost function.

SEMI-SUPERVISED LEARNING

- The goal in semi-supervised learning is to use both labeled and unlabeled data to build better learners, rather than using each one alone.
- Semi-supervised learning is applied to the test data.
- SSL algorithms can use unlabeled data to help improve prediction accuracy if data satisfies appropriate assumptions.
- Uses unlabeled data to help solve a supervised task.
- The following are the semi-supervised learning algorithms:
    - Self Training
    - Generative Models
    - S3VMs
    - Graph Based Algorithms
    - Multiview algorithms

Self Training:

Advantages:

- The simplest semi-supervised learning algorithm.
- A wrapper method, applies to existing (complex) classifiers.
- Often used in real tasks like NLP.

Disadvantages:

- Early mistakes could reinforce themselves.
- Cannot say too much in terms of convergence.

Generative Models:

- SSL is fairly easy to do using generative models.

Examples:

- Mixture of Gaussian Distributions (GMM):
    - Image classification
    - Expectation – Maximization (EM) Algorithm
- Mixture of Multinomial distributions (Naïve Bayes)
    - Text categorization
    - EM Algorithm
- Hidden Markov Models (HMM)
    - Speech Recognition
    - Baum-Welch Algorithm

Advantages:

- Clear, well studies probabilistic framework.
- Can be extremely effective, if the model is close to correct.
- HMMs have been very successful in handling variable length sequences as well as modeling the temporal behavior of speech signals using a sequence of states, each of which is associated with a probability distribution of observations.

Disadvantages:

- Often difficult to verify the correctness of the model.
- Model identifiability
- EM local optima
- Unlabeled data may hurt if generative model is wrong.

Semi-Supervised Support Vector Machines (S3VM):

- Maximizes unlabeled data margin.
- SVM objective is convex, where as S3VM objective is non-convex.
- Finding a solution to S3VM is difficult and focus of S3VM research.

Advantages:

- Can be applied wherever SVMs are applicable.
- Clear mathematical framework.

Disadvantages:

- Optimization difficult and can be trapped in bad local optima.
- More modest assumption than generative model or graph-based methods
- Potentially lesser gain.

Graph-based Algorithms:

Advantages:

- Clear mathematical framework.
- Performance is strong if the graph happens to fit the task.
- The inverse of Laplacian can be viewed as kernel matrix.
- Can be extended to directed graphs

Disadvantages:

- Performance is bad if the graph is bad.
- Sensitive to graph structure and edge weights.

Multiview Algorithms:

Co-training idea:

- Co-training refers to two views of an item, for example image and HTML text.
- Train an image classifier and text classifier.
- The two classifiers teach each other.
- Assumes that features can be split into two sets and each sub-feature set is sufficient to train a good classifier.
- Initially two separate classifiers are trained with the labeled data on the two sub-feature sets
- Each classifier then classifies the unlabeled data and teaches the other classifier with the few unlabeled example (and the predicted labels) they feel most confident.
- Each classifier is retrained with the additional training examples given by the other classifier and the process repeats.

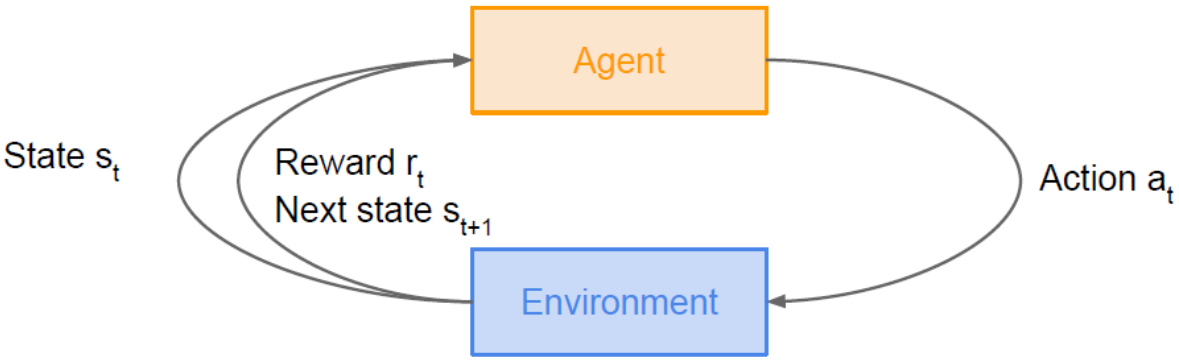
Pros of Co-training algorithm:

- Simple wrapper method and applies almost to all existing classifiers.
- Less sensitive to mistakes than self-training.

Cons of co-training algorithm:

- Natural feature splits may not exist and models using both features should do better.

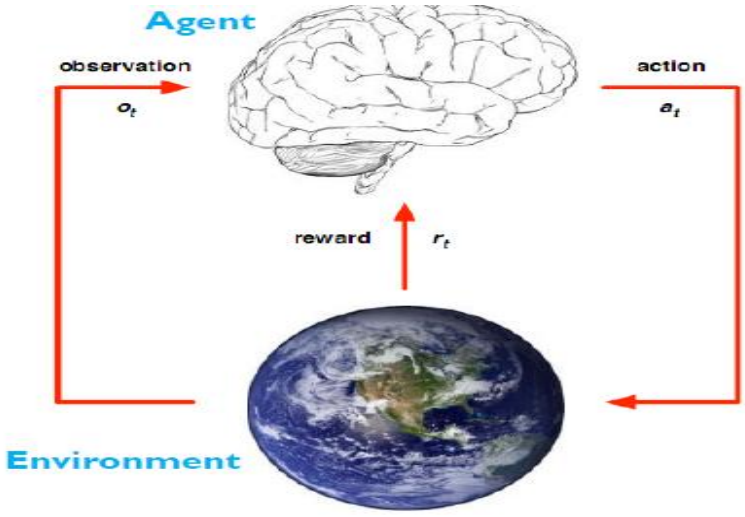
REINFORCEMENT LEARNING



- Reinforcement Learning (RL) solves a very specific kind of problem where the decision making is sequential.
- RL is a class of learning problems in which an agent interacts with an unfamiliar, dynamic and stochastic environment.
- It's called reinforcement learning because the agent gets positive reinforcement for tasks done well and negative reinforcement for tasks done poorly.
- The purpose of RL is to learn the optimal policy based only received rewards.
- RL provides a general-purpose framework for AI
- RL problems can be solved by end-to-end deep learning
- A single agent can now solve many challenging tasks

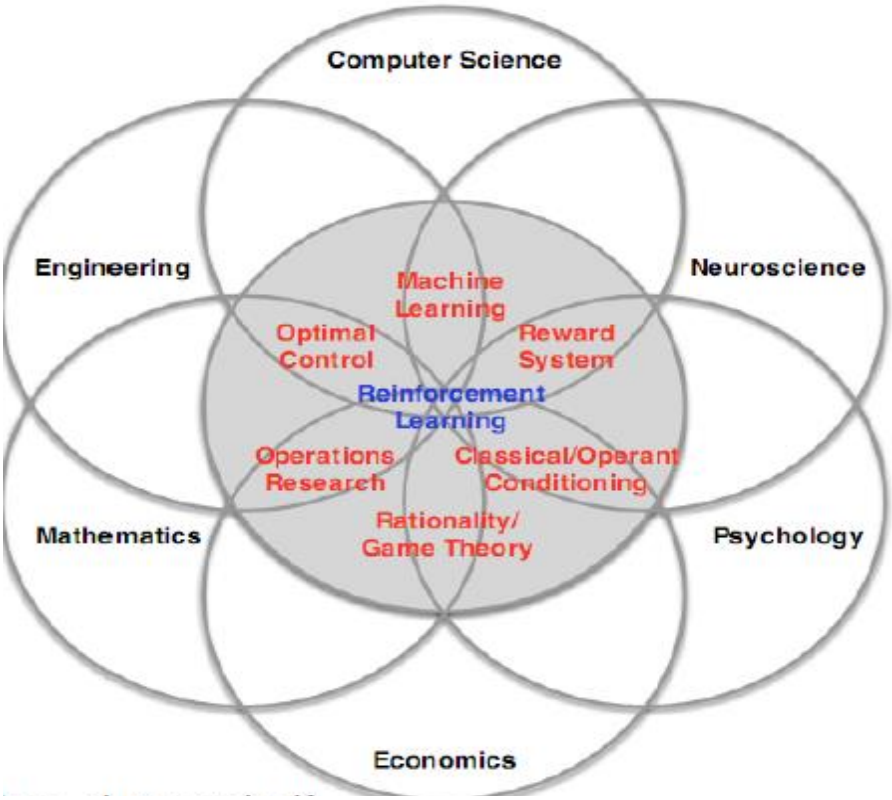
Agent and Environment:

- Environment and Agent are main building blocks of reinforcement learning in AI
- Usually there is an agent acting in an unknown environment.
- At each step t the agent:
  - Executes action
  - Receives observation
  - Receives scalar reward
- The environment:
  - Receives action
  - Emits observations
  - Emits scalar reward



- They (goal, agent and the environment) interact continually. The agent selecting actions and the environment responding to those actions and presenting new situations to the agent.
- The environment also gives rise to rewards, special numerical values that the agent tries to maximize over time.
- These rewards are observed as a function of actions performed on the environment.
- Each action brings a reward and moves the agent to another state of the environment (usually, a s a result of some random process with unknown properties).
- The learner is called the agent and the goal of the agent is to optimize its long term reward.
- Reinforcement learning algorithms such as Q-learning and their neural network based counterparts are used in learning to play video games, robotic navigation and coordination, inventory and supply chain management, optimization of complex electric power systems (power grids) and the learning of financial trading strategies.
- Observes the environment and learns the ideal behavior
- Selects and takes certain actions, and receives rewards in return
- Receives feedback in a loop (indirect feedback after many examples)
- Learns the strategy or policy that maximizes rewards
- Stands in the middle ground between supervised and unsupervised learning
- The algorithm is provided information about whether the answer is correct but not how to improve it
- The reinforcement learner must try out different strategies and see which works best
- The algorithm searches over the state space of possible inputs and outputs in order to maximize a reward
- Less commonly used
- Useful for learning how to act or behave when given occasional reward or punishment signals.
- Used to reinforce or strengthen the network based on critic information
- A network trained under reinforcement learning receives some feedback from the environment
- Feedback is evaluative and not instructive as in the case of supervised learning
- Based on feedback the network performs the adjustments of the weights to obtain better critic information in future
- These algorithms learn from experience and tries to capture the best possible knowledge to make accurate decisions
- Markov Decision Process model is an example of reinforcement machine learning algorithms
- Initially developed for machines to play games
- Differs from the supervised learning in that we need not supply the labelled input/output pairs
- Focus is on finding the balance between exploring the new solutions versus exploiting the learned solutions
- Temporal difference learning is used in reinforcement learning where supervision is delayed reward
- Reward is the type of feedback in reinforcement learning
- Feedback (positive or negative reward) given at the end of a sequence of steps
- No supervised output but delayed reward
- The task of reinforcement learning is to use observed rewards to learn an optimal policy f or the environment.

- Learn to interact with the world from the reinforcement you get
- RL is a general-purpose framework for decision making
    - RL is for an agent with the capacity to act
    - Each action influences the agent's future state
    - Success is measured by a scalar reward signal
    - **Goal**: select actions to maximize future reward
- At each step *t* the agent:
    - Executes action
    - Receives observation
    - Receives scalar reward
- The environment:
    - Receives action
    - Emits observation
    - Emits scalar reward
- An RL agent may include one or more of these components:
    - **Policy**: agent's behavior function and determines which action should be performed in each state.
    - **Value function**: how good is each state and/or action. The value of a state is defined as the sum of the reinforcement received when starting in that state and following some fixed policy to a terminal state. Value of a state is dependent upon the policy. The value function at state 's', is the expected cumulative reward from following the policy from state 's'.
    - **Model**: agent's representation of the environment



Approaches to Reinforcement Learning:

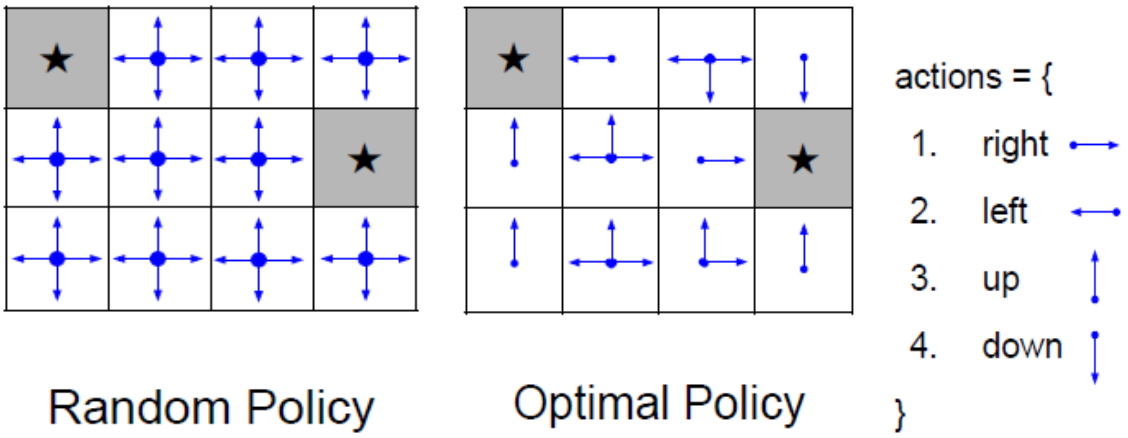
- Value-based RL
  - Estimate the optimal value function
  - This is the maximum value achievable under any policy
- Policy-based RL
  - Search directly the optimal policy
  - This is the policy achieving maximum future rewards
- Model-based RL
  - Build a model of the environment
  - Plan (e.g. by lookahead) using model

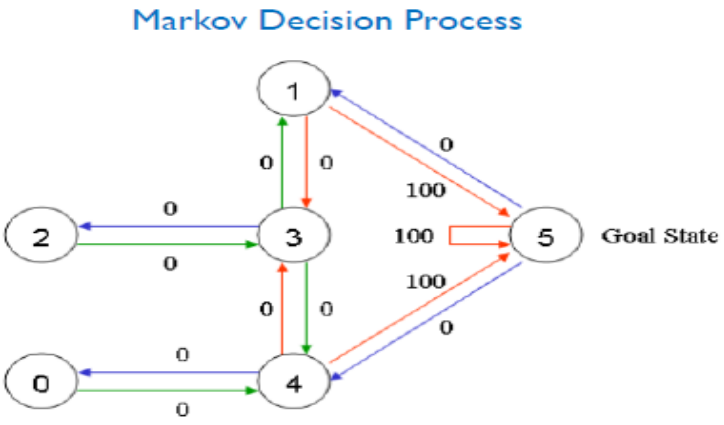
Markov Property:

- At each time step t, agent chooses an action which depends on current state S_t.
- Current state completely characterizes the state of the world.

Markov Decision Process model:

- A sequential decision problem for a fully observable, stochastic environment with a Markovian transition function and additive reward is called a Markov Decision Problem.
- Deals with mathematical formulation of the RL problem.
- A reinforcement learning task which satisfies the Markov property is called MDP.
- **Finite MDP**: MDP with finite state and action spaces.
- The state, the dynamics of the environment and action sets define a particular finite MDP.
- An MDP is defined as a 5-tuple consisting of (state space of the process, action space of the process, probability distribution over next state, probability distribution over rewards, initial state distribution).





Q-Learning:

- Represents a value function by a neural network.
- Optimal Q-values should obey Bellman equation.
- Treat right hand side of Bellman equation as a target.
- Minimize MSE loss by stochastic gradient descent.
- A Q-learning agent learns an action utility function or Q-function giving the expected utility of taking a given action in a given state.
- The Q-function at state 's' and action 'a', is the expected cumulative reward from taking action 'a' in state 's' and then following the policy.
- Does not always work but when it works, usually more sample efficient.
- Challenge is exploration.

APPLICATIONS OF REINFORCEMENT LEARNING:

- Google DeepMind Deep Q-learning for Atari Games
- Deep RL for Computer Games
- Deep RL for Drug design
- RL for traffic
- Traffic Light Control
- RL for Robotics
- Has applications in game playing, robot in a maze
- Inventory Management
- Dynamic Channel Allocation
- Elevating scheduling
- Helicopter control
- Autonomous vehicles

Bayesian Learning:

Pros:

- Principled treatment of uncertainty
- Conceptually simple
- Immune to overfitting
- Facilitates encoding of domain knowledge

Cons:

- Mathematically and computationally complex

WORD2VEC ALGORITHM

- Is a Machine Learning model used to generated Word Embeddings with words which are similar to each other are in close proximity in vector space.
- If we want to train a machine learning model on textual input or to find relations between words, first we need to convert text to vectors. These are called Word Embeddings.
- Word2Vec can automatically capture the relations between words like Paris, Beijing, Tokyo, Delhi, New York are all clustered together in vector space. Similarly Cat, Dog, Rat, Duck are all clustered together in vector space.
- Word2Vec can be used to perform relational operations and also extract and provide the most similar words, how similar are two words, pick the odd word out of group of words if the model is trained with large enough corpus of data.

Representing words:

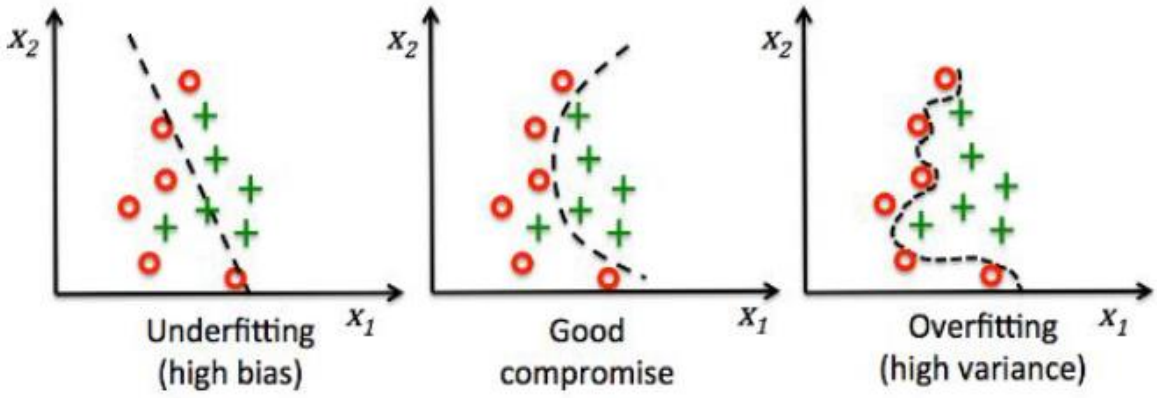
- Words are atomic entities
- One-hot representation does not capture meaning due to:
    - Words with similar or different meanings are equally apart
    - Cosine distance is 0 between any two words

GENERALIZATION

- Refers to the capability of applying learned knowledge to previously unseen data, i.e, to make predictions on novel inputs.
- Without generalization there is no learning, but just memorizing
- If the validation loss decreases as well, the learned patterns seem to generalize
- Data augmentation is key to improve generalization:
    - Random translation
    - Left/right flipping
    - Scaling
- The following are ways to improve Generalization:
    - Weight sharing (greatly reduce the number of parameters)
    - Data Augmentation (e.g., jittering, noise injection etc.,)
    - Dropout
    - Weight delay (L2, L1)
    - Sparsity in the hidden units
    - Multi-task (unsupervised learning)

OVERFITTING

- A model overfits when it simply memorizes the data, e.g., a curve that fit through every training data.
- If the overfitted model is tested on the training data, the model will give 0 training errors.
- To test the model generalization ability, the model should be tested on unseen test cases.
- Underfitting, Good and Overfitting are shown below:



Methods to reduce/prevent overfitting:

- Train with more data
- Reduce/Remove features
- Early stopping
- Regularization
- Ensembling

REGULARIZATION

- Used to reduce/prevent 'Overfitting'.
- Without regularization, the learned function varies extensively depending on the dataset.
- It constrains the learning algorithm to improve out of sample error, especially when noise is present.
- One popular regularization technique is 'weight decay'.

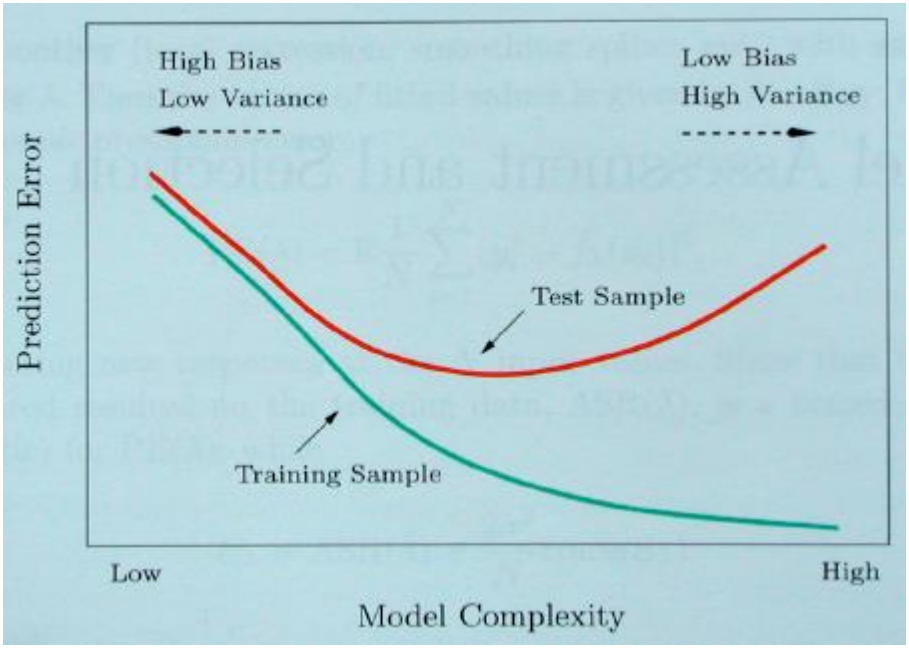
BIAS

- Defined as the average squared difference between predictions and true values.
- It's a measure of how good your model fits the data
- Zero bias would mean that the model captures the true data generating process perfectly. Both training and validation loss would go to zero. That's unrealistic as data is almost always noisy, so some bias inevitable.
- Bias error is useful to quantify how much on average are the predicted values different from the actual value.
- A high bias error means we have a under-performing model. Thus, we aim at low bias.

VARIANCE

- Variance quantifies how the predictions made on same observation are different from each other.
- A high variance model will over-fit on your training population and perform badly on any observation beyond training. Thus, we aim at low variance.

- The tradeoff between Bias/Variance is shown below:



Occam's Razor:

- This is one of the three principles of learning from data; others being Sampling Bias and Data Snooping.
- The principles is: *The simplest model that fits the data is also the most plausible.*
- pick the simplest model that adequately explains the data
- Select the simplest hypothesis (solution) that suits/fits the data
- When Occam's Razor says that simpler is better, it doesn't mean simpler is more elegant. It means simpler has a better chance of being right.
- This principle is about performance, not about aesthetics. If a complex explanation of the data, performs better, we will take it.
- Has been formally proved under different sets of idealized conditions.

BAG OF WORDS

- Is a method to extract features from text documents.
- These features are used for training a Machine Learning algorithm.
- A sentence or a document is considered as a 'Bag' containing words.
- Considers the words and their frequency of occurrence in the sentence or the document disregarding sematic relationship in the sentences.
- Creates a vocabulary of all the unique words occurring in all the documents in the training set.
- The number of occurrence and not the sequence or order of words is what matters in this approach.

TF-IDF:

- One of the approaches to scoring in which the frequency of words is rescaled by how often they appear in all documents so that the scores for frequent words like *the* that are also frequent across all documents are also penalized.

Term Frequency (TF):

- TF is the scoring of the frequency of the word in the current document.
- Higher the frequency of the word, the more relevant it is.
- TF (T) = No. of occurrences of T in Di / Number of words in Di.
- The following is the example for TF calculation:

D_1: Andrew is a tall boy.
D_2: Ram is a good boy. Ratna is also good.

Dictonary

$$\begin{bmatrix} A \\ Also \\ Andrew \\ Boy \\ Good \\ Tall \\ Is \\ Ram \\ Ratna \end{bmatrix}$$

$$TF = \frac{\#\ of\ occurences\ of\ T\ in\ D_i}{\#\ of\ words\ in\ D_i}$$

$$TF\ of\ \text{“}Andrew\text{”}\ in\ D_1 = \frac{1}{5} = 0.2$$
$$TF\ of\ \text{“}good\text{”}\ in\ D_2 = \frac{2}{9} = 0.22$$

Inverse Document Frequency (IDF):

- IDF is a scoring of how rare the word is across the documents.
- Thus, the IDF of a rare item is high, whereas IDF of a frequent term is likely to be low.
- IDF (T) = LOG_e (No. of documents / Number of documents with term T)
- The following is the example of IDF:

D_1: Andrew is a tall boy.
D_2: Ram is a good boy. Ratna is also good.

Dictonary

$$\begin{bmatrix} A \\ Also \\ Andrew \\ Boy \\ Good \\ Tall \\ Is \\ Ram \\ Ratna \end{bmatrix}$$

$$IDF(T) = \log_e\left(\frac{\#\ of\ documents}{\#\ of\ docs\ with\ term\ T}\right)$$
$$IDF('A') = \log_e\left(\frac{2}{2}\right) = 0$$
$$IDE('Andrew') = \log_e\left(\frac{2}{1}\right) = 0.69$$
$$IDF('is') = \log_e\left(\frac{2}{2}\right) = 0$$
$$IDF('good') = \log_e\left(\frac{2}{1}\right) = 0.69$$

- The following example is for calculating TF-IDF for D1 and D2:

D_1 : Andrew is a tall boy.

D_2 : Ram is a good boy. Ratna is also good.

$$\text{Dictionary} \begin{bmatrix} A \\ Also \\ Andrew \\ Boy \\ Good \\ Tall \\ Is \\ Ram \\ Ratna \end{bmatrix} \underset{\text{IDF}}{\begin{bmatrix} 0 \\ 0.69 \\ 0.69 \\ 0 \\ 0.69 \\ 0.69 \\ 0 \\ 0.69 \\ 0.69 \end{bmatrix}} \times \underset{\text{TFD1}}{\begin{bmatrix} 0.2 \\ 0 \\ 0.2 \\ 0.2 \\ 0 \\ 0.2 \\ 0.2 \\ 0 \\ 0 \end{bmatrix}} = \underset{\text{TFIDFD1}}{\begin{bmatrix} 0 \\ 0 \\ 0.14 \\ 0 \\ 0 \\ 0.14 \\ 0 \\ 0 \\ 0 \end{bmatrix}}$$

D_1 : Andrew is a tall boy.

D_2 : Ram is a good boy. Ratna is also good.

$$\text{Dictionary} \begin{bmatrix} A \\ Also \\ Andrew \\ Boy \\ Good \\ Tall \\ Is \\ Ram \\ Ratna \end{bmatrix} \underset{\text{IDF}}{\begin{bmatrix} 0 \\ 0.69 \\ 0.69 \\ 0 \\ 0.69 \\ 0.69 \\ 0 \\ 0.69 \\ 0.69 \end{bmatrix}} \times \underset{\text{TFD2}}{\begin{bmatrix} 0.11 \\ 0.11 \\ 0 \\ 0.11 \\ 0.22 \\ 0 \\ 0.22 \\ 0.11 \\ 0.11 \end{bmatrix}} = \underset{\text{TFIDFD2}}{\begin{bmatrix} 0 \\ 0.08 \\ 0 \\ 0.0 \\ 0.15 \\ 0 \\ 0 \\ 0.08 \\ 0.08 \end{bmatrix}}$$

<u>Advantages of BoW:</u>

- BoW model is very simple to understand and implement.
- Offers lot of flexibility for customization on specific text data.
- Used with great success on prediction problems like language modeling and documentation classification.

<u>Limitations of BoW:</u>

- **Vocabulary:** Has dimensionality issue as the total dimension is the vocabulary size. It can easily overfit the model. The vocabulary requires careful design, most specifically in order to manage the size, which impacts the sparsity of the document representations. The remedy is to use some well-known dimensionality reduction technique to input data.
- **Sparsity:** Sparse representations are harder to model both for computational reasons (space and time complexity) and also for information reasons, where the challenge is for the models to harness so little information in such a large representational space.
- **Meaning:** BoW representation doesn't consider the semantic relation between words. Generally, the neighbor words in a sentence should be useful for predicting the target word. Discarding word order ignores the context, and in turn meaning of words in the document (semantics). Context and meaning can offer a lot to the model, that if modeled could tell the difference between the same words differently arranged (*this is interesting* vs *is this interesting*), synonyms (*old bike* vs *used bike*) and much more.

RECOMMENDER SYSTEMS

- Recommender Systems are software tools and techniques providing suggestions for items to be of use to a user. The suggestions relate to various decision-making processes, such as what items to buy, what music to listen to or what online news to read.
- Recommender system is an information-filtering technique that provides users with recommendations for which they might be interested in.
- The following are the benefits of Recommender Systems to companies:
    - Increase the number of items sold
    - Sell more diverse items
    - Increase the user satisfaction
    - Increase user fidelity
    - Better understand what the user wants
- The following are the benefits of Recommender Systems to users:
    - Find some good items
    - Find all good items
    - Annotation in context
    - Recommend a sequence
    - Recommend a bundle

Recommender Systems – Features:

- Recommends items that are most popular among all the users.
- Divides users into multiple segments based on their preferences and recommends items to them based on the segment they belong to.

Purposes of recommender systems:

- Predictive perspective
- Interaction perspective
- Conversion perspective
- Retrieval perspective
- Recommendation perspective

Collaborative Filtering:

- Collaborative filtering systems make recommendations based on historic preferences of the users.
- It uses item based nearest neighbor or user based nearest neighbor method.
- User based nearest neighbor recommends items by finding users similar to the active user.
- Item based nearest neighbor is easy to scale and can be computed offline and served without constant retraining.
- Item based nearest neighbor can be implemented best through KNN model.
- A common example of CF is predicting which movies people will want to watch based on how they and other people have rated movies which they have already seen. The key idea is that the prediction is not based on features of the movie or user, but merely on a ratings matrix.

Cosine similarity:

- Produces better results in item-to-item filtering
- Ratings are vectors in n-dimensional space
- Similarity is calculated based on the angle between the vectors

Adjusted cosine similarity:

- Takes average user ratings into account
- Transforms the original ratings

Association Rule:

- Association rule mining uses machine learning models to analyze data for patterns or co-occurrence in a database.
- Each transaction is a list of items.
- Association rule finds all rules that correlate the presence of one set of items with that of another set of items.
- It identifies frequent patterns and is most commonly used for market basket analysis.
- Performance measures for association rules include support, confidence and lift.
- Support indicates how frequently the items appear in the data, provides fraction of transactions that contain X and Y.
- Confidence indicates the number of times the if-then statements are found true and indicates how often X and Y occur together, given the number of times X occurs.
- Lift compare the actual confidence with the expected confidence and indicate the strength of a rule over the random co-occurrence of X and Y.

A priori Algorithm:

Steps:

- Create itemsets using the large itemsets of the previous pass.
- This large itemset is joined with itself to generate all itemsets with a size larger by one.
- Each generated itemset with a subset that not large is deleted.
- The remaining itemsets are the candidates.
- A priori algorithm uses frequent itemsets to generate association rules.
- Support value of frequent itemsets is greater than the threshold value.
- The algorithm reduces the number of candidates being considered by only exploring the itemsets whose support count is greater than the minimum support count.

ENSEMBLE METHODS

- Ensemble methods refer to combining many different machine learning models in order to get a more powerful prediction.
- Thus ensemble methods increase the accuracy of the predictions.
- A commonly used class of ensemble methods are forests of randomized trees.
- Ensemble Methods are used in order to:
    - Decrease variance (bagging)
    - Decrease bias (boosting)
    - Improve predictions (stacking)

Bagging:

- Bagging refer to **B**ootstrap **Agg**regation averages a given procedure over many samples to reduce its variance.
- Bagging tests multiple models on the data by sampling and replacing data, i.e, it utilizes bootstrapping.
- This reduces the noise and variance by utilizing multiple samples.
- Each hypothesis has the same weight as all the others. Now, aggregating of the output of various models is done.
- Averaging the prediction over a collection of unstable predictors generated from bootstrap samples (both classification and regression)
- A technique for reducing the variance of an estimated prediction function.
- **Bootstrap:** randomly drawn datasets with replacement from the training data, each sample the same size as the original training set. Sampling the training data with replacement.
- Sample training data (Di), 'k' times – train a classifier Ci on Di.
- Each test sample is classified by a 'k' classifiers.
- Results are averaged to obtain the final decision.
- A technique for reducing the variance of an estimated prediction function.

Boosting:

- Generate a set of weak classifiers
- Combines them using a weighted combination (probabilities)
- Weights proportional to their performance on validation set
- It is an iterative technique which adjusts the weight of an observation based on the last classification.
- If an observation was classified incorrectly, it tries to increase the weight of this observation and vice versa.
- Boosting in general decreases the bias error and builds strong predictive models.
- Weighted vote with a collection of classifiers that were trained sequentially from training sets given priority to instances wrongly classified (classification)
- Weak learners combine to form strong classifiers
- Does not work that well with strong learners
- Boosting combined with trees gives some of the most powerful classifiers

The basic difference in principle between bagging and boosting is that boosting constantly monitors its cumulative error and uses that residual for subsequent training.

AdaBoost:

- is a popular variant of boosting
- Generate classifiers by weighted sampling
- Has no random elements and grows an ensemble of trees by successive re weightings of the training set where current weights depend on the history of the ensemble formation.
- Does not overfit as more trees are added to the ensemble

NATURAL LANGUAGE PROCESSING (NLP)

- NLP is a field of computer science, AI and computational linguistics concerned with the interactions between computers and human (natural) languages.

Language Model:

- Models that predict the probability distribution of language expressions.
- Probabilistic language models based on n-grams recover a surprising amount of information about a language.
- They can perform well on such diverse tasks as language identification, spelling correction, genre classification and named-entity recognition.
- Language models can have millions of features and hence feature selection and pre-processing of data to reduce noise is important.

Neural Network Language Models (NNLMs):

Two types:

- Feed Forward NNLM
- Recurrent NNLM

Limitations:

- Sparsity – Solved
- World Similarity – Solved
- Finite context – Not
- Computational Complexity - Softmax

N-gram model:

- N-gram is a sequence of n words
- A model of the probability distribution of n-letter sequence is called n-gram model.
- N-grams estimate word conditional probabilities via counting.
- Sparse (alleviated by back-off, but not entirely)
- Doesn't exploit word similarity
- Finite Context
- Unigram, bi-gram, trigram are some of the forms of N-grams.

Corpus:

- A body of text is called 'Corpus' from the Latin word for 'body'.

Smoothing:

- The process of adjusting the probability of low-frequency counts is called 'smoothing'.

Text Classification:

- Text classification can be done with Naïve Bayes n-gram models or with any of the classification algorithms.

Applications of NLP:

- Language Modeling (Speech recognition, Machine Translation)
- Word-sense Learning
- Reasoning over Knowledge Bases
- Acoustic Modeling
- Parts-Of-Speech Tagging
- Chunking
- Named Entity Recognition
- Semantic Role Labeling
- Parsing
- Sentiment Analysis
- Paraphrasing
- Question-Answering