Algorithm Selection

How to pick ML Algorithms

Anatomy of ML Algorithms

We will discuss how machine learning algorithms operate in general.

Goal is to explain a few essential concepts on the flow and basic building blocks.

The Basic Flow ML Algorithms

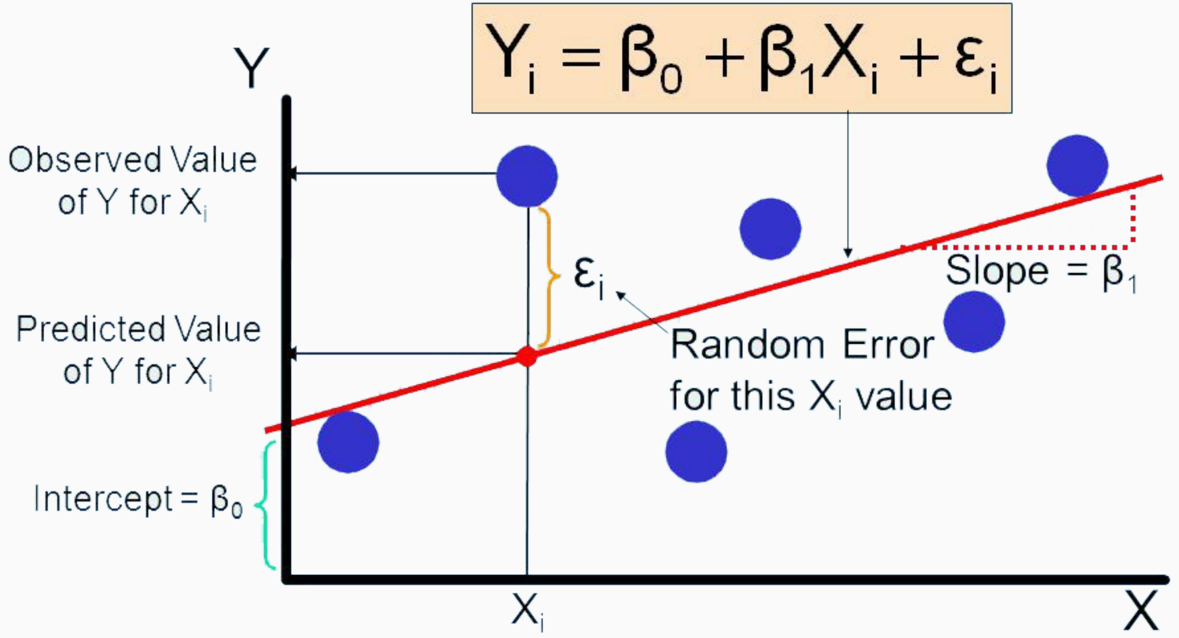
- Predict or Inference
    - hypothesis function

      $h\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \cdots + \theta n x_n$
- Error or Loss
    - Cost function MSE
- Train or Learn
    - Gradient Descent function

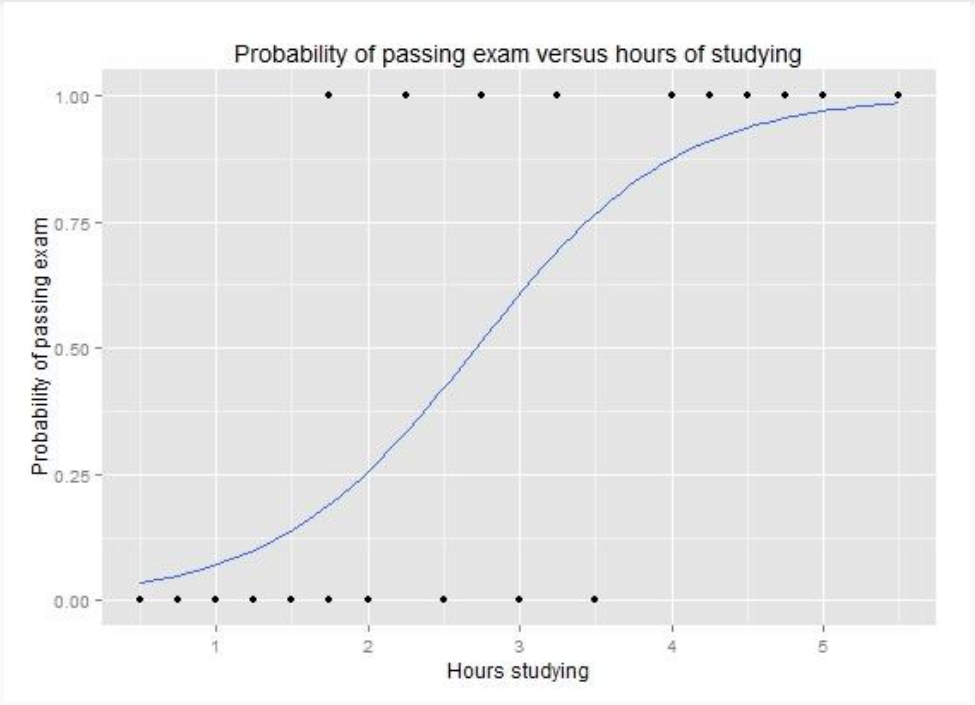
Building Blocks of Many ML Algorithms

- Linear Regression
- Logistic Regression

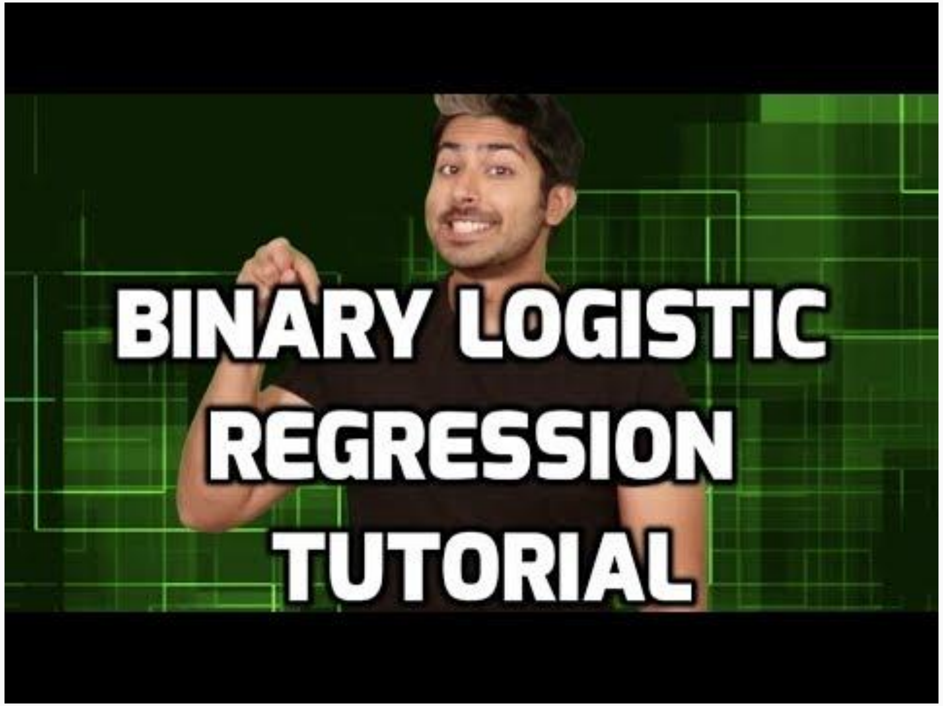
Linear Regression


$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i$
Y
Observed Value of Y for X_i
Predicted Value of Y for X_i
Intercept = β_0
ε_i
Random Error for this X_i value
Slope = β_1
X_i
X

Logistic Regression



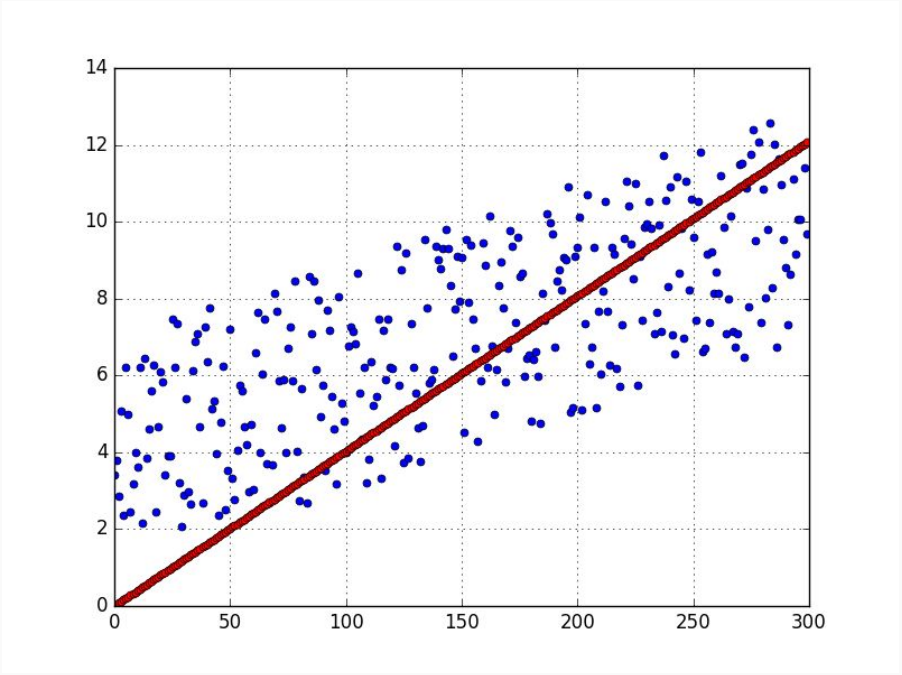
Logistic Regression



Why Linear Regression is Flawed

Simple linear regression models fit a "straight line". In practice, they rarely perform well.

Their main advantage is that they are easy to interpret and understand.



Why Linear Regression is Flawed

In this regard, simple linear regression suffers from two major flaws:

1. **It's prone to overfit with many input features.**
2. **It cannot easily express non-linear relationships.**

Let's take a look at how we can address the first flaw.

Regularization in Machine Learning

Extreme example of overfitting:

- Let's say you have 100 observations with 100 features in your training dataset.
- If you fit a linear regression model with all of those 100 features, you can perfectly "memorize" the training set.
- Each coefficient would simply memorize one observation. This model would have perfect accuracy on the training data, but perform poorly on unseen data.
- It hasn’t learned the true underlying patterns; it has only memorized the noise in the training data.

Regularization in Machine Learning

Regularization is a technique used to prevent overfitting by artificially penalizing model coefficients.

- It can discourage large coefficients (by dampening them).
- It can also remove features entirely (by setting their coefficients to 0).
- The "strength" of the penalty is tunable.

Regularized Regression Algos

There are 3 common types of regularized linear regression algorithms

1. Lasso Regression
2. Ridge Regression
3. Elastic-Net

Lasso Regression

Lasso, or LASSO, stands for:

- Least
- Absolute
- Shrinkage
- Selection Operator

Lasso Regression

- Lasso regression penalizes the absolute size of coefficients.
- Practically, this leads to coefficients that can be exactly 0.
- Thus, Lasso offers automatic feature selection because it can completely remove some features.
- Remember, the "strength" of the penalty should be tuned.
- A stronger penalty leads to more coefficients pushed to zero.

Ridge Regression

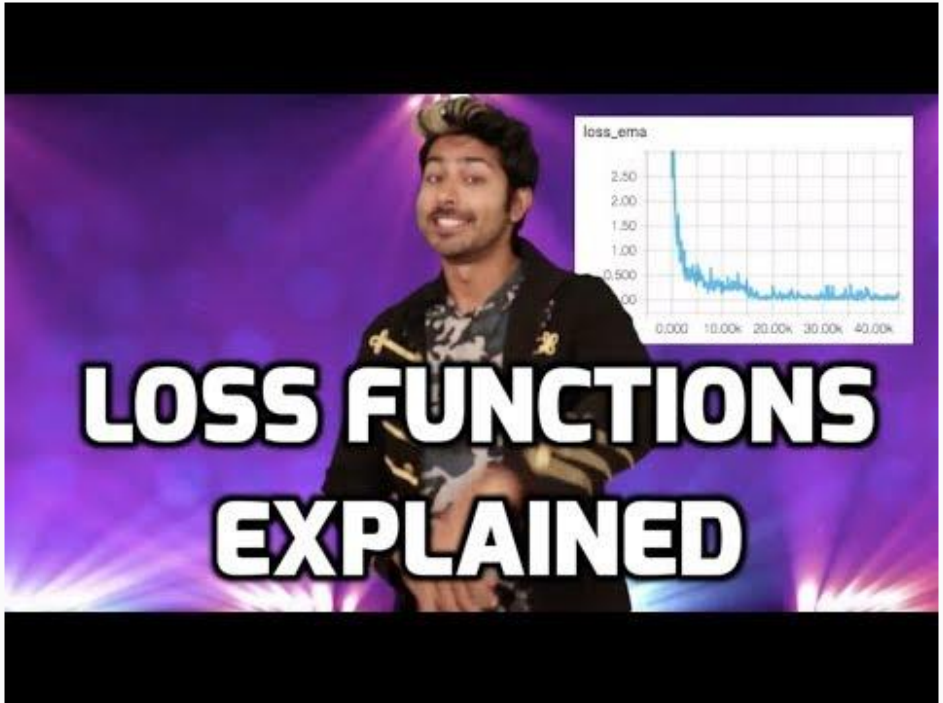
- Ridge regression penalizes the squared size of coefficients.
- Practically, this leads to smaller coefficients, but it doesn't force them to 0.
- In other words, Ridge offers feature shrinkage.
- Again, the "strength" of the penalty should be tuned.
- A stronger penalty leads to coefficients pushed closer to zero.

Elastic-Net

Elastic-Net is a compromise between Lasso and Ridge.

- Elastic-Net penalizes a mix of both absolute and squared size.
- The ratio of the two penalty types should be tuned.
- The overall strength should also be tuned.

Loss Functions



KNN - K Nearest Neighbors

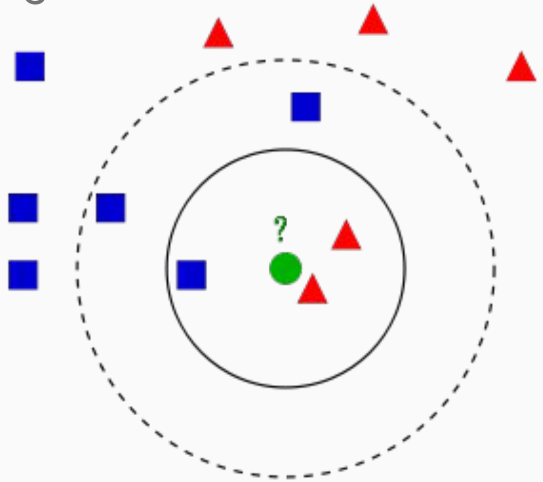
KNN is an ML algorithm that classifies new data points, by a measure of distance, to its K nearest neighbors — in the existing data.

🟦 Class A

🔺 Class B

🟢 New Data Point

- —-- K = 3
- - - - K = 5



KNN Considerations

Choosing K: A problem all of its own, best to try different sizes and compare results.

Rule of Thumb: k < sqrt(n) , n = total # of samples

Weights: Necessary to attach “weights” to more important features, could be an iterative process.

KNN Strengths

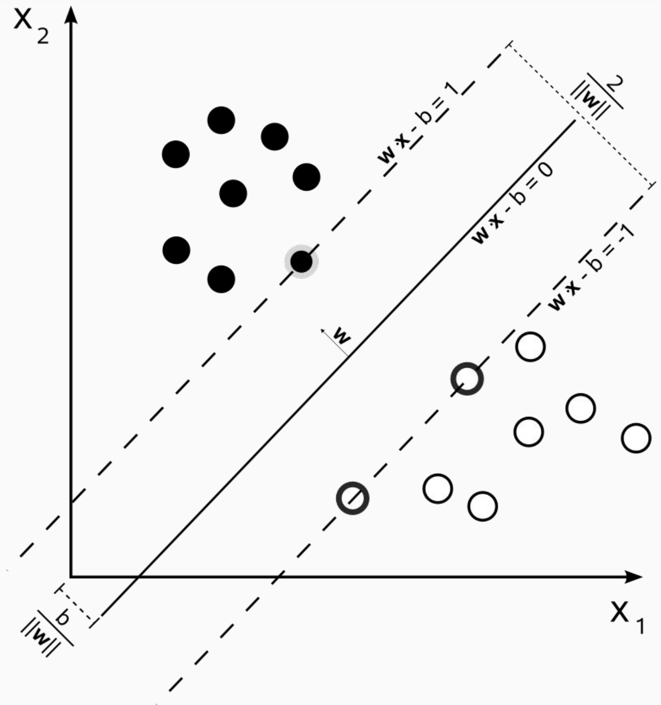
- Simple and intuitive
- Can be applied to any type of distribution
- Great classifier if Sample size is large.

KNN Weaknesses

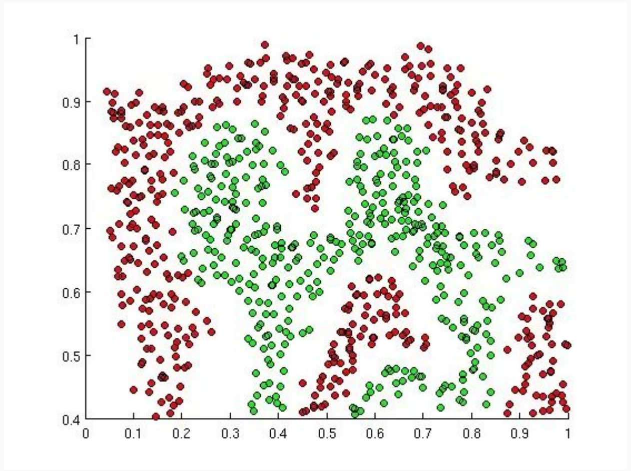
- Requires a large sample size
- Slow to classify.
    - Need to calculate and compare distance from new example to all other examples in the sample space.
    - Then, compare example to the k nearest points
- Choosing size of K can be tricky

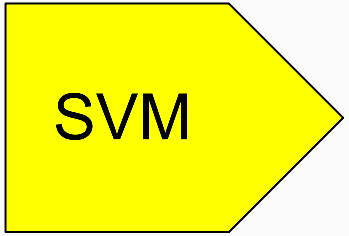
Support Vector Machines

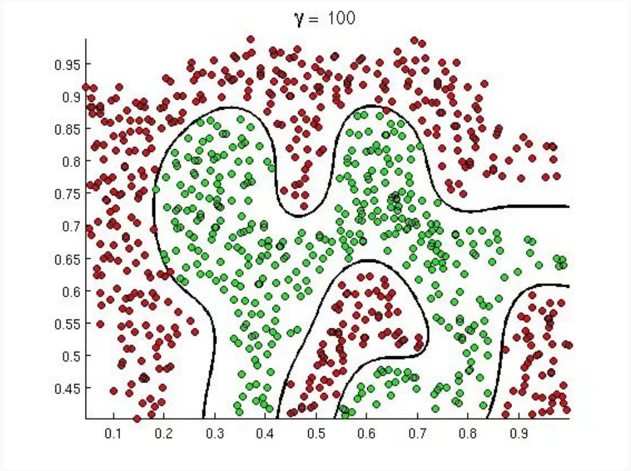
- ML algorithm that maps a training set onto a feature space and divides points into classes with a “hyperplane” **w**.
- Support vectors *maximize* the margin around the hyperplane
  - Require EXTENSIVE tuning of a “kernel” function.
- SVMs classify new points based on what side of the dividing hyperplane they land on.



SVMs in Action







SVM Strengths

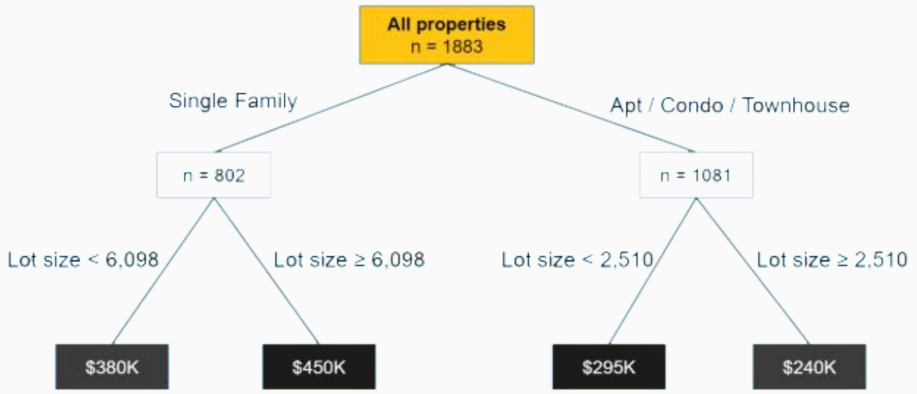
- **Among the most accurate models in Machine Learning.**
- Can handle non-linear relationships,
  - Tune kernel parameter to prevent overfitting.
- Can deal very effectively with high-dimensional data with something called the “kernel” trick.

SVM weaknesses

- Both **Training** and **Testing** are **SLOW**
- Don’t perform as well with Multi-class data.
    - Ex: Iris dataset
- **Kernel** function requires extensive tuning.
- Requires a lot of memory and processing power.

Decision Tree Algorithms

Decision trees model data as a "tree" of hierarchical branches. They make branches until they reach "leaves" that represent predictions.



Decision Tree Algorithms

Unfortunately, decision trees suffer from a major flaw as well. If you allow them to grow limitlessly, they can completely "memorize" the training data, just from creating more and more and more branches.

Tree Ensembles

Ensembles are machine learning methods for combining predictions from multiple separate models.

They allow us to take advantage of the flexibility of decision trees while preventing them from overfitting the training data.

Two most common methods for ensembling are Bagging and Boosting.

Tree Ensembles - Bagging

Bagging attempts to reduce the chance of overfitting.

- It trains a large number of "strong" learners in parallel.
- A strong learner is a model that's relatively unconstrained.
- Bagging then combines all the strong learners together in order to "smooth out" their predictions.

Tree Ensembles - Boosting

Boosting attempts to improve the predictive flexibility of simple models.

- It trains a large number of "weak" learners in sequence.
- A weak learner is a constrained model (i.e. you could limit the max depth of each decision tree).
- Each one in the sequence focuses on learning from the mistakes of the one before it.
- Boosting then combines all the weak learners into a single strong learner.

Tree Ensembles

Ensembling is a general term, but when the base models are decision trees, they have special names: random forests and boosted trees!

Decision Tree Algorithms
Random Forests

Random forests train a large number of "strong" decision trees and combine their predictions through bagging.

There are two sources of "randomness" for random forests:

1. Each tree is only allowed to choose from a random subset of features to split on (leading to feature selection).

2. Each tree is only trained on a random subset of observations (a process called resampling).

Decision Tree Algorithms
Random Forests

In practice, random forests tend to perform very well right out of the box.

- They often beat many other models that take up to weeks to develop.
- They are the perfect "swiss-army-knife"algorithm that almost always gets good results.
- They don’t have many complicated parameters to tune.

Decision Tree Algorithms
Boosted Trees

Boosted trees train a sequence of "weak", constrained decision trees and combine their predictions through boosting.

- Each tree is allowed a maximum depth, which should be tuned.
- Each tree in the sequence tries to correct the prediction errors of the one before it.

Decision Tree Algorithms
Boosted Trees

In practice, boosted trees tend to have the highest performance ceilings.

- They often beat many other types of models after proper tuning.
- They are more complicated to tune than random forests.

Artificial Neural Networks

Artificial neural networks are computational models which work similar to the functioning of a human nervous system.

There are several kinds of artificial neural networks.

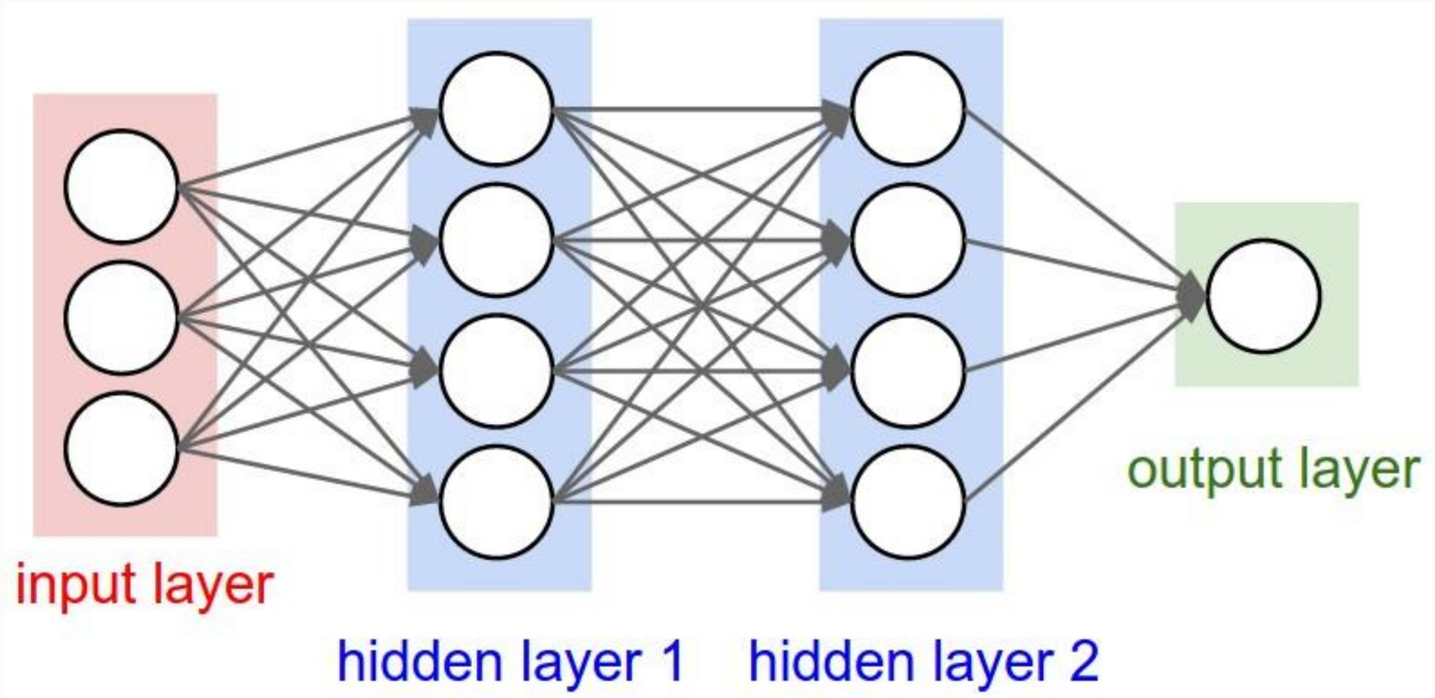
These type of networks are implemented based on the mathematical operations and a set of parameters required to determine the output.

Architecture of an Artificial Neural Network

ANN is a set of connected neurons organized in layers:

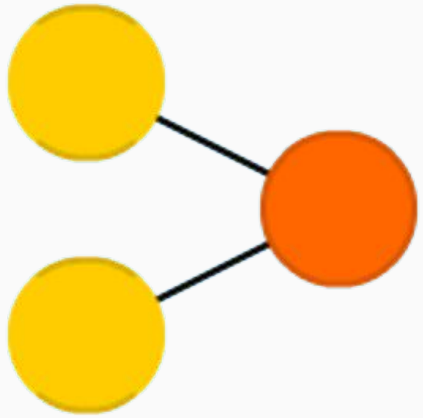
- **input layer**: brings the initial data into the system for further processing by subsequent layers of artificial neurons.
- **hidden layer**: a layer in between input layers and output layers, where artificial neurons take in a set of weighted inputs and produce an output through an activation function.
- **output layer**: the last layer of neurons that produces given outputs for the program.

Multi-Layer Neural Network



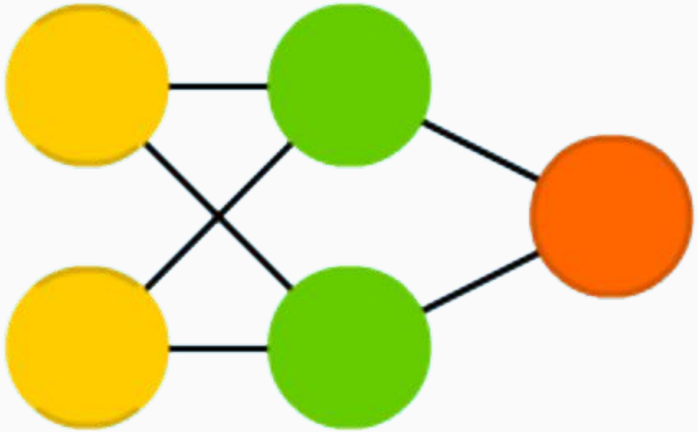
Perceptron

Perceptron (P)

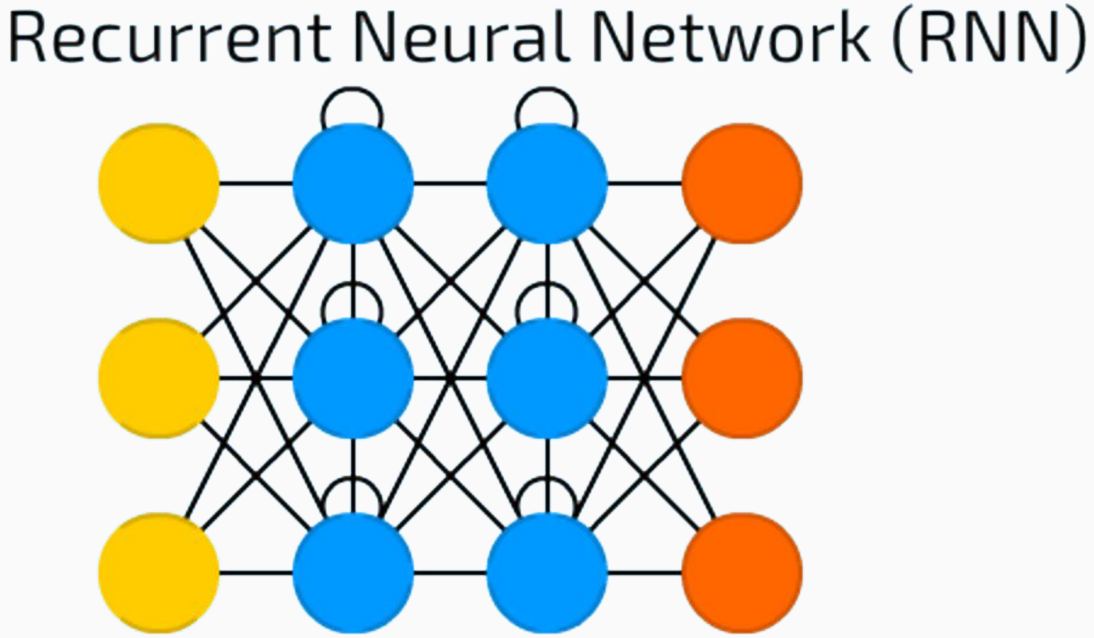


Feed Forward

Feed Forward (FF)

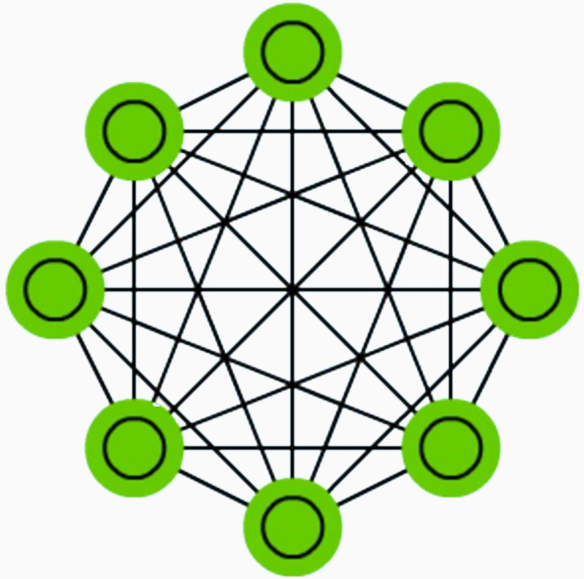


Recurrent Neural Network

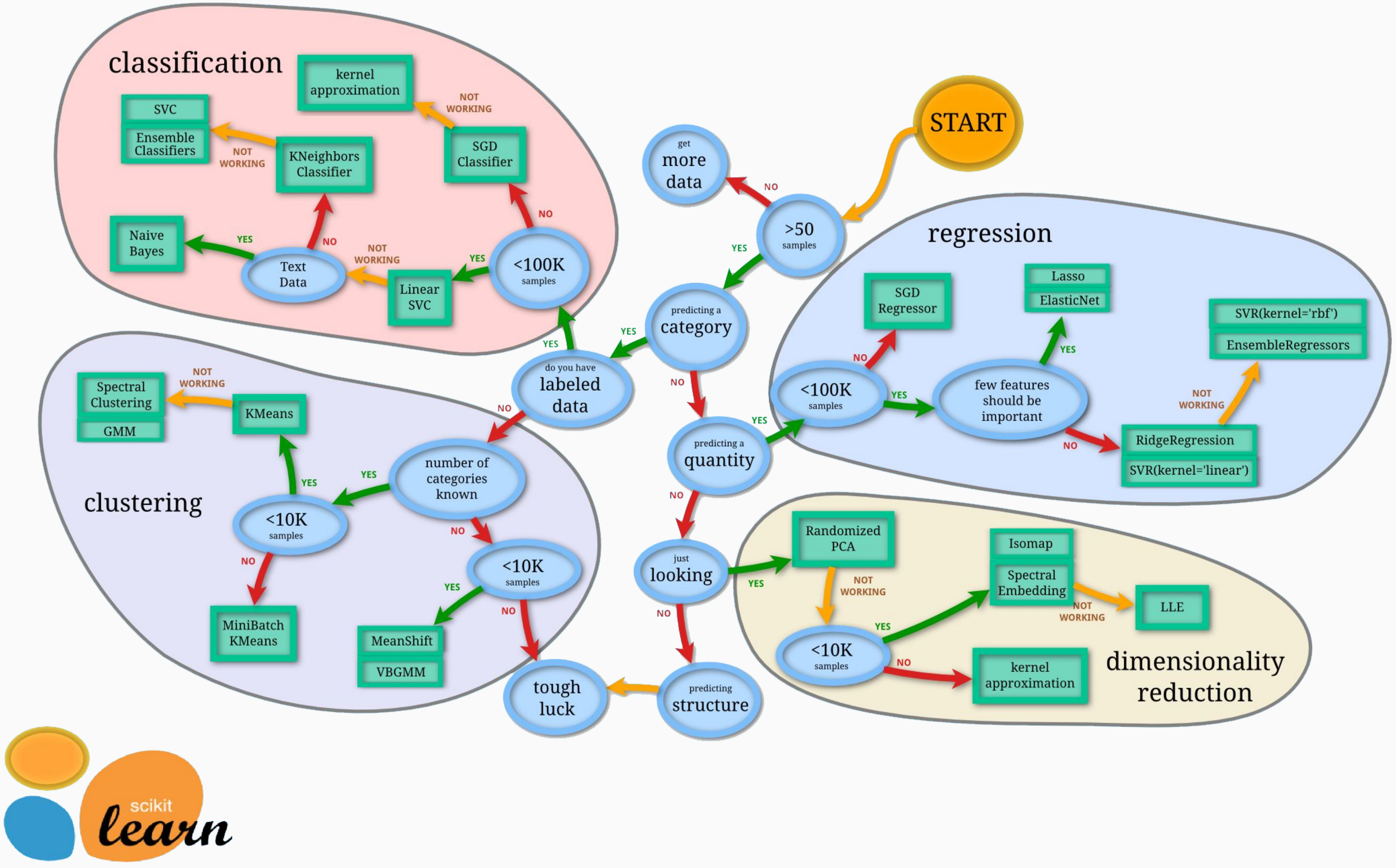


Markov Chain

Markov Chain (MC)



Neural Networks


START
>50 samples
NO
get more data
YES
predicting a category
YES
do you have labeled data
YES
classification
<100K samples
YES
Linear SVC
NOT WORKING
Text Data
YES
Naive Bayes
NO
KNeighbors Classifier
NOT WORKING
SVC
Ensemble Classifiers
NO
SGD Classifier
NOT WORKING
kernel approximation
NO
clustering
number of categories known
YES
<10K samples
YES
KMeans
NOT WORKING
Spectral Clustering
GMM
NO
MiniBatch KMeans
NO
<10K samples
YES
MeanShift
VBGMM
NO
tough luck
NO
predicting a quantity
YES
regression
<100K samples
NO
SGD Regressor
YES
few features should be important
YES
Lasso
ElasticNet
NO
RidgeRegression
SVR(kernel='linear')
NOT WORKING
SVR(kernel='rbf')
EnsembleRegressors
NO
just looking
YES
dimensionality reduction
Randomized PCA
NOT WORKING
<10K samples
YES
Isomap
Spectral Embedding
NOT WORKING
LLE
NO
kernel approximation
NO
predicting structure
scikit learn