Contents

[Which model would work 1](#_Toc520825629)

[Logistic Regression 2](#_Toc520825630)

[Decision Trees 3](#_Toc520825631)

[Tree Ensembles 4](#_Toc520825632)

[Bagged Trees 4](#_Toc520825633)

[Boosted Trees 5](#_Toc520825634)

[Random Forest 5](#_Toc520825635)

[SVM 5](#_Toc520825636)

[Naive Bayes 6](#_Toc520825637)

[Deep Learning 7](#_Toc520825638)

[KNN 7](#_Toc520825639)

[Logistic vs Decision 7](#_Toc520825640)

Which model would work best will depend on certain questions

So you have to ask yourself:

* what kind of decision boundary makes more sense in your particular problem?
* how do you want to balance bias and variance?
* are there interactions between my features?

**Parametric method**

* Logistic Regression
* Linear Discriminant Analysis
* Perceptron
* Naive Bayes
* Simple Neural Networks
* Non Parametric method

**Non Parametric method**

* k-Nearest Neighbors
* Decision Trees like CART and C4.5
* Support Vector Machines

# Logistic Regression

It searches for single linear decision boundary

* + Pros
    - low variance
    - provides probabilities for outcomes which gives good interpretability
* features roughly linear, problem roughly linearly separable
* robust to noise, use l1,l2 regularization for model selection, avoid overfitting
* can be used as a baseline for other algorithms
  + - works well with diagonal (feature) decision boundaries
    - NOTE: logistic regression can also be used with kernel methods
    - Logistic models can be updated easily with new data using stochastic gradient descent.??
  + Cons
    - high bias (difference between actual and predicted value) when there are multiple or non linear boundaries. They are not flexible enough to capture more complex relationship
    - can hardly handle categorical features

**It should be used**

Logistic regression assumes there is only one decision boundary that is smooth and non-linear.

Logistic regression work best when there is single decision boundary. If the signal to noise ratio is low (it is a ‘hard’ problem) logistic regression is likely to perform best. In technical terms, if the AUC of the best model is below 0.8, logistic very clearly outperformed tree induction.

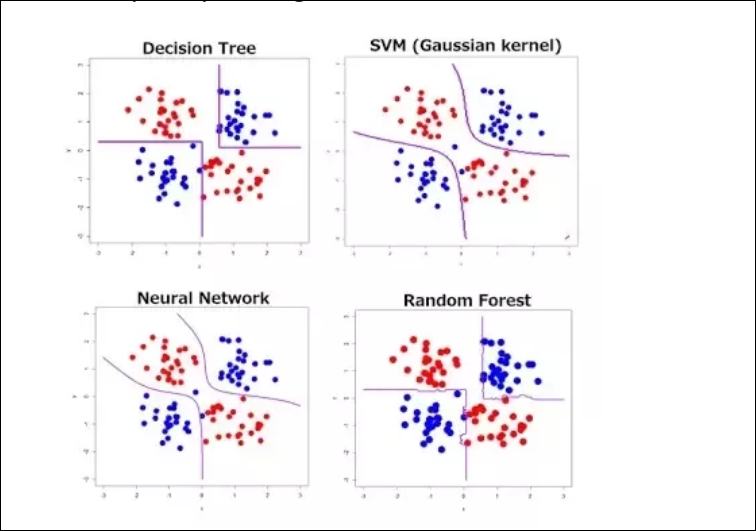
# Decision Trees

* Regular (not bagged or boosted)
* **Pros**
  + easy to interpret visually when the trees only contain several levels
  + Can easily handle qualitative (categorical) features
  + Works well with decision boundaries parellel to the feature axis
* **Cons**
  + prone to overfitting
  + possible issues with diagonal decision boundaries

Where to use –

The decision tree assumes the splits are axis parallel and will become more complex with the increase in number of features and multiple decision.

decision tree will partition you feature space into half spaces using axis aligned linear decision boundaries. So using a decision tree you get a non linear decision boundary, sometimes more than one.



# Tree Ensembles

* good for large N and large P, can deal with categorical features very well
* non parametric, so no need to worry about outliers

Bagged Trees : train multiple trees using bootstrapped data to reduce variance and prevent overfitting

* **Pros**
  + reduces variance in comparison to regular decision trees
  + Can provide variable importance measures
    - classification: Gini index
    - regression: RSS
  + Can easily handle qualitative (categorical) features
  + Out of bag (OOB) estimates can be used for model validation
  + RF works out of the box, but usually performs worse than GBT
* **Cons**
  + Not as easy to visually interpret
  + Does not reduce variance if the features are correlated

**In what condition we should use it –**

Boosted Trees : Similar to bagging, but learns sequentially and builds off previous trees

* + Pros
    - Somewhat more interpretable than bagged trees/random forest as the user can define the size of each tree resulting in a collection of stumps (1 level) which can be viewed as an additive model
    - Can easily handle qualitative (categorical) features
* GBT’s work better but the parameters are harder to tune
  + - Cons
      * Unlike bagging and random forests, can overfit if number of trees is too large

# Random Forest

* + Pros
    - Decorrelates trees (relative to bagged trees)
      * important when dealing with mulitple features which may be correlated
    - reduced variance (relative to regular trees)
  + Cons
    - Not as easy to visually interpret

# SVM

* + Pros
    - Performs similarly to logistic regression when linear separation
    - Performs well with non-linear boundary depending on the kernel used
    - They are also fairly robust against overfitting, especially in high-dimensional space.Cons
  + Cons
    - Susceptible to overfitting/training issues depending on kernel
    - However, SVM's are memory intensive, trickier to tune due to the importance of picking the right kernel, and don't scale well to larger datasets. Currently in the industry, random forests are usually preferred over SVM's.
    - slow to train, for most industry scale applications, not really efficient

**When to use**

SVM is basically logistic regression with L2 regularization and a slightly different loss function (SVM uses hinge loss while logistic uses log loss).

[Support Vector Machines](https://www.quora.com/topic/Support-Vector-Machines) (SVMs) use a different loss function (Hinge) from LR. They are also interpreted differently (maximum-margin). However, in practice, an SVM with a linear kernel is not very different from a Logistic Regression

The main reason you would want to use an SVM instead of a Logistic Regression is because your problem might not be linearly separable. In that case, you will have to use an SVM with a non linear kernel (e.g. RBF)

Linear SVMs and logistic regression generally perform comparably in practice.

Use SVM with a nonlinear kernel if you have reason to believe your data won't be linearly separable (or you need to be more robust to outliers than LR will normally tolerate). Otherwise, just try logistic regression first and see how you do with that simpler model. If logistic regression fails you, try an SVM with a non-linear kernel like a RBF.

**Key Differences**

Logistic Regression fits the data points as if they are along a continuous function.

This isn't always the case for single-class classification, and so the function may have trouble classifying where P = 0.5

SVM fits a function (hyperplane) that attempts to separate two classes of data that could be of multiple dimensions.

SVM could have difficulty when the classes are not separable or there is not enough margin to fit a (n\_dimensions - 1) hyperplane between the two classes.

# Naive Bayes

Pros

* computationally efficient when P is large by alleviating the curse of dimensionality
* works surprisingly well for some cases even if the condition doesn’t hold
* with word frequencies as features, the independence assumption can be seen reasonable. So the algorithm can be used in text categorization
  + - Simple to implement
    - Works well with high dimensions

Cons

* Relies on independence assumption and will perform badly if this assumption is not met

# Deep Learning

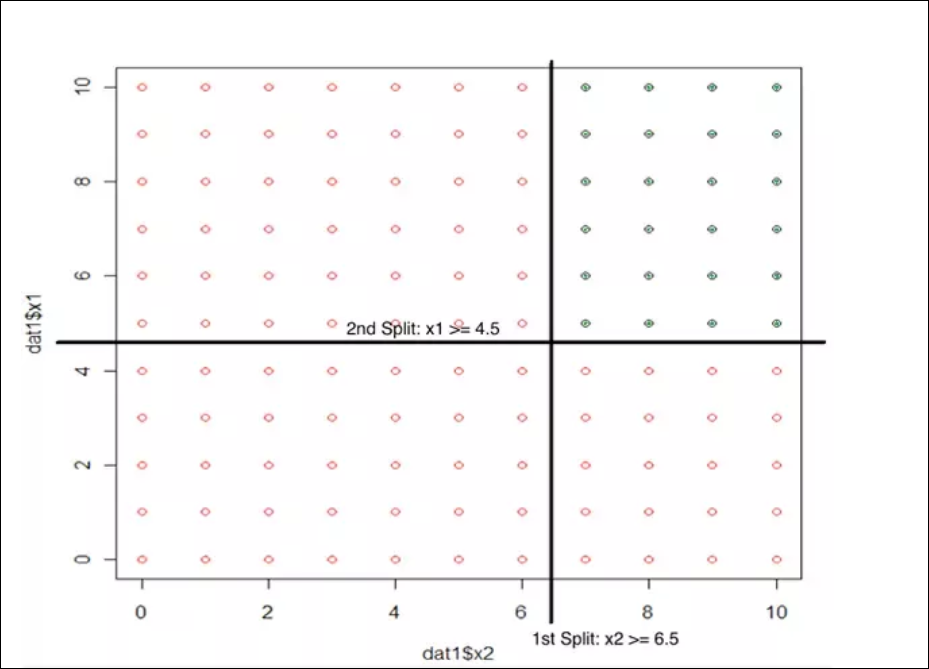
To continue the trend, deep learning is also easily adapted to classification problems. In fact, classification is often the more common use of deep learning, such as in image classification.

* **Strengths:** Deep learning performs very well when classifying for audio, text, and image data.
* **Weaknesses:** As with regression, deep neural networks require very large amounts of data to train, so it's not treated as a general-purpose algorithm.

# KNN

# Logistic vs Decision

**Decision trees assume** that our decision boundaries are parallel to the axes, for example if we have two features (x1, x2) then it can only create rules such as x1>=4.5, x2>=6.5 etc. which we can visualize as lines parallel to the axis. We see this in practice in the diagram below.



So decision trees chop up the feature space into rectangles (or in higher dimensions, hyper-rectangles). There can be many partitions made and so decision trees naturally scale up to creating more complex (say, higher VC) functions - which can be a problem with over-fitting.

What assumptions does logistic regression make? Despite the probabilistic framework of logistic regression, all that **logistic regression assumes** is that there is one smooth linear decision boundary. It finds that linear decision boundary by making assumptions that the P(Y|X) of some form, like the inverse logit function applied to a weighted sum of our features. Then it finds the weights by a maximum likelihood approach.   
  
However people get too caught up on that... The decision boundary it creates is a linear\* decision boundary that can be of any direction. So if you have data where the decision boundary is not parallel to the axes, regression picks it out pretty well, whereas a decision tree will have problems

**So in terms of performane**

* If the signal to noise ratio is low (it is a ‘hard’ problem) logistic regression is likely to perform best. In technical terms, if the AUC of the best model is below 0.8, logistic very clearly outperformed tree induction.
* You have low signal to noise for a number of reasons - the problem is just inherently unpredictable (think stock market) dataset or it is too small to ‘find the signal’. The latter is an interesting case - we observe that the performance order of the two algorithms can cross - meaning, logistic performs better on a small version of the dataset but eventually is beaten by the tree when the dataset gets large enough.
* Trees generally have a harder time coming up with calibrated probabilities. This can be helped somewhat with bagging and Laplace correction.
* Trees tend to have problems when the base rate is very low. In the worst case, it will not split at all. While this might maximize accuracy it is obviously useless for ranking or probability estimation. You can try to fix this with downsampling, but then your probability estimates are off.

# Difference between SVM and Logistic Regression

# Difference between Decision Tree and Random Forest

# Interview Questions

## You are working on a time series data set. You manager has asked you to build a high accuracy model. You start with the decision tree algorithm, since you know it works fairly well on all kinds of data. Later, you tried a time series regression model and got higher accuracy than decision tree model. Can this happen? Why?

**Answer:** Time series data is known to posses linearity. On the other hand, a decision tree algorithm is known to work best to detect non – linear interactions. The reason why decision tree failed to provide robust predictions because it couldn’t map the linear relationship as good as a regression model did. Therefore, we learned that, a linear regression model can provide robust prediction given the data set satisfies its linearity assumptions.

## Q8. You are assigned a new project which involves helping a food delivery company save more money. The problem is, company’s delivery team aren’t able to deliver food on time. As a result, their customers get unhappy. And, to keep them happy, they end up delivering food for free. Which machine learning algorithm can save them?

**Answer:** You might have started hopping through the list of ML algorithms in your mind. But, wait! Such questions are asked to test your machine learning fundamentals.

This is not a machine learning problem. This is a route optimization problem. A machine learning problem consist of three things:

1. There exist a pattern.
2. You cannot solve it mathematically (even by writing exponential equations).
3. You have data on it.

Always look for these three factors to decide if machine learning is a tool to solve a particular problem.

## Q9. You came to know that your model is suffering from low bias and high variance. Which algorithm should you use to tackle it? Why?

**Answer:**  Low bias occurs when the model’s predicted values are near to actual values. In other words, the model becomes flexible enough to mimic the training data distribution. While it sounds like great achievement, but not to forget, a flexible model has no generalization capabilities. It means, when this model is tested on an unseen data, it gives disappointing results.

In such situations, we can use bagging algorithm (like random forest) to tackle high variance problem. Bagging algorithms divides a data set into subsets made with repeated randomized sampling. Then, these samples are used to generate  a set of models using a single learning algorithm. Later, the model predictions are combined using voting (classification) or averaging (regression).

Also, to combat high variance, we can:

1. Use regularization technique, where higher model coefficients get penalized, hence lowering model complexity.
2. Use top n features from variable importance chart. May be, with all the variable in the data set, the algorithm is having difficulty in finding the meaningful signal.

**Q36.** **Considering the long list of machine learning algorithm, given a data set, how do you decide which one to use?**

**Answer:** You should say, the choice of machine learning algorithm solely depends of the type of data. If you are given a data set which is exhibits linearity, then linear regression would be the best algorithm to use. If you given to work on images, audios, then neural network would help you to build a robust model.

If the data comprises of non linear interactions, then a boosting or bagging algorithm should be the choice. If the business requirement is to build a model which can be deployed, then we’ll use regression or a decision tree model (easy to interpret and explain) instead of black box algorithms like SVM, GBM etc.

In short, there is no one master algorithm for all situations. We must be scrupulous enough to understand which algorithm to use.

**Q40.** **OLS is to linear regression. Maximum likelihood is to logistic regression. Explain the statement.**

**Answer:** OLS and Maximum likelihood are the methods used by the respective regression methods to approximate the unknown parameter (coefficient) value. In simple words,

Ordinary least square(OLS) is a method used in linear regression which approximates the parameters resulting in minimum distance between actual and predicted values. Maximum Likelihood helps in choosing the the values of parameters which maximizes the likelihood that the parameters are most likely to produce observed data.

#### **41. What are parametric models? Give an example.**

Parametric models are those with a finite number of parameters. To predict new data, you only need to know the parameters of the model. Examples include linear regression, logistic regression, and linear SVMs.

Non-parametric models are those with an unbounded number of parameters, allowing for more flexibility. To predict new data, you need to know the parameters of the model and the state of the data that has been observed. Examples include decision trees, k-nearest neighbors, and topic models using latent dirichlet analysis.

# How should I choose between SVM and decision tree for a classification problem?

The biggest difference between the two algorithms is that SVM uses the kernel trick to turn a linearly nonseparable problem into a linearly separable one (unless of course we use the linear kernel), while decision trees (and forests based on them, and boosted trees, both to a lesser extent due to the nature of the ensemble algorithms) split the input space into hyper-rectangles according to the target.

Usually one will work better than another in a given situation, but it's hard to tell in most cases in high dimensional spaces unless there is something about the data that suggests one over the other. This is the preferred method, but hardly obvious in most cases.

Most of the time, people use a validation set to not only optimize hyperperameters but also to choose between algorithms. It's not perfect, but often it works.

Oh - if you have categories in your inputs, you can't use SVMs. They only work with numeric data.

# Generative VS Discriminative Models

## ****Generative classifiers****

* Assume some functional form for **P(Y), P(X|Y)**
* Estimate parameters of **P(X|Y), P(Y)** directly from training data
* Use Bayes rule to calculate **P(Y |X)**

## ****Discriminative Classifiers****

* Assume some functional form for **P(Y|X)**
* Estimate parameters of **P(Y|X)** directly from training data

**Examples:**

## Generative classifiers

* ‌Naïve Bayes
* Bayesian networks
* Markov random fields
* ‌Hidden Markov Models (HMM)

## Discriminative Classifiers

* ‌Logistic regression
* Scalar Vector Machine
* ‌Traditional neural networks
* ‌Nearest neighbour
* Conditional Random Fields (CRF)s