Decoding Machine Learning Algorithms

Gupta, Tvisha
tvisha.gupta@in.ey.com

Mukherjee, Satraajeet
satraajeet.mukherjee@in.ey.com

Rajasekaran, Prabakar
prabakar.rajasekaran@in.ey.com

May 15, 2018

Abstract

Just like a human being learns from his past experiences and modifies his future strategies, it is imperative that our computers do the same as well. But with the ever increasing amount of data available, learning from the past and applying in the future needs to be done in a smarter way. Machine Learning serves this exact purpose. With the entire analytics and IT industry rushing for adopting machine learning algorithms for their modelling purposes, the insurance industry is not far behind. Many of the pricing methodologies are already moving to more sophisticated methods of Gradient Boosting and Support vector machines than the traditional GLM. Optimising the maximum likelihood is a lot more than just numerical approximation. Predicting loss amounts and reserves is now done on a claim level basis. Cohort level reserving is done by clustering different variables based on their proximity and then simulating sample policies from each cohort. But while we are applying more and more techniques, the question is do we really understand them well?

In this research, we decode the black box in machine learning algorithms. This enables us in understanding if we are applying those techniques in the right place, in the right way and with the right parameters. We explain the mathematics of the fundamental concepts in machine learning, keeping our emphasis on variable selection, decision trees for regression and classification and regularisation. We determine how optimal parameters for each of them are obtained, how the predicted values and splits are calculated and what the sensitivities of each of them are with respect to changes in the data. We study why different techniques give different solutions to the same problem and what are the key drivers behind it. At last, we understand the appropriateness of different modelling techniques at different situations.

Contents

1 Introduction

Machine learning is an application of artificial intelligence that provides systems the ability to automatically learn, act and improve from experience (just like humans) without being explicitly programmed. It relies on data and information fed in the form of observations and real world interactions.

1.1 Difference between Statistics and Machine Learning

Traditionally, statistical learning refers to inference, testing and explaining the properties of the population from where the data comes. Statistics deals with a formally specified model and deriving closed form solutions. Machine learning, on the other hand is about letting the machine predict how the future data and the outcomes will look like. Classical statistical areas like data collection, sampling, unbiased estimators, marginal distributions of the explanatory variables do not hold in machine learning. Rather machine learning acts like a "black box" and focuses on learning algorithms.

However, that said, the two are not mutually exclusive. Many of the techniques used in machine learning (eg. expectation maximisation, optimisation) come from traditional statistical techniques. Statistics and machine learning attempt to solve many of the same problems in data analysis, and have begun converging in their approaches.

An interesting read on the differences can be found in a blog by Brendor O'Connor [1] where he gives a fight between statistics and machine learning.

1.2 When do we need machine learning

When both statistical and machine learning techniques can perform the same task, then why do we need the computers to identify the patterns and learn themselves rather than us adopting more robust and accurate statistical ways? The choice between statistical learning and machine learning depends on two parameters - the problem's complexity and the need for adaptivity.

- Problems solved by humans/animals which seem routinely but cannot be formulated in a well defined algorithm. For example, driving, speech recognition.
- Problems involving analysis of extremely large and complex datasets wherein a complete statistical analysis of the data is not feasible since the data might itself evolve by the time the analysis gets over.
- Problems involving the need for adaptive models. Statistical models get fixed once coded and need to be recalibrated with new data. Machine learning solves this problem by letting the model evolve based on the experiences.

1.3 Problems addressed by machine learning

Machine learning broadly addresses problems realted to three areas - supervised learning, unsupervised learning and reinforcement learning.

Supervised learning in Machine learning refers to the models that takes the form of predicting about a response variable, given information about some other explanatory variables. Classic examples of supervised learning are regression and classification. Supervised learning normally starts with dividing the data into training and testing sets and then training the model on the basis of the training data. The model is then applied on the test data and allowed to adapt itself from any new changes it see in the test data.

Unsupervised learning is the science of learning the relationships and structure of a data in the absence of any (supervising) response variable. The input data is processed with the goal of summarising the data or getting a compressed version of the data. Classic examples of unsupervised learning are clustering and principal component analysis.

Between supervised and unsupervised learning, lies another branch of machine learning - called reinforcement learning. This deals with goal oriented algorithms, which learn how to attain a goal by learning over multiple steps. These algorithms are penalised when they make the wrong decisions and rewarded when they make the right decisions. A classic example of reinforcement learning is neural network.
Consider a simple problem of a product suggestion on an e-commerce website. A supervised learning algorithm might take the ratings that an individual provides to the products and accordingly suggest products that he

would be interested in. An unsupervised learning algorithm might see the trends in the browsing activities of the person and would suggest products accordingly. A reinforcement learning algorithm might suggest products to the individual and ask him to rate. Based on his ratings to the suggestions, the suggestion algorithm will learn and change the suggestion criteria.

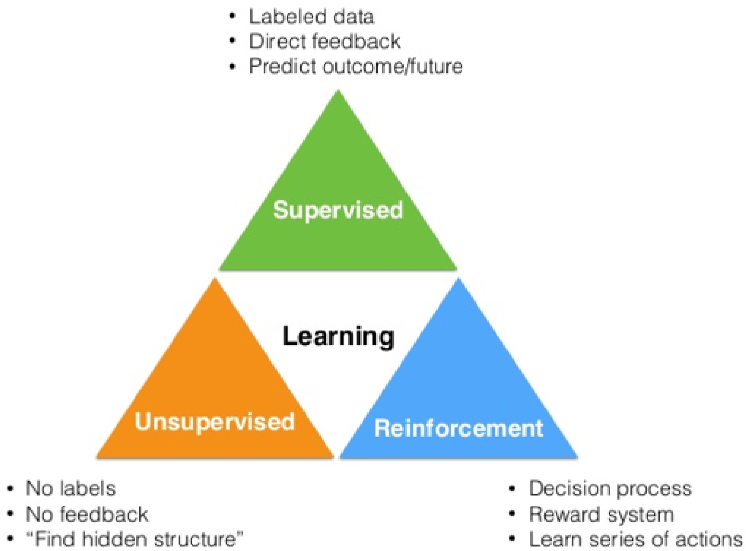

Figure 1: Types of learning

In this paper, we deal with explaining a few techniques, namely random forest, support vector machines, gradient boosting method and neural networks. We explain what these methods do and how prediction is performed using them.

1.4 Why is there a need to decode machine learning algorithms

Many ML techniques result in models which are not easily interpretable. By this we mean that, if an interested party were to ask for the exact formula to use to make future predictions, it would be difficult to give the answer in a simple form. For certain applications this can matter a lot. A particular patient might be prescribed a particular limit on the medicinal expenses but not the other patient, and, when an underwriter is challenged as to which features drive this conclusion and the magnitude of the effect of each feature, he should be able to give a simple answer. In an insurance context, a model which predicts that a customer should have a price increase in the premium, but leaves an underwriter unable to understand exactly why this is the case, may be difficult for the underwriter to implement in practice.

Hence even though the idea behind Machine Learning is "to let the data play the game", it is essential to understand the rules of the game and how it is being played.

2 Supervised Learning

Supervised learning is the machine learning task of inferring a function from supervised training data. The training data consist of a set of training examples in the form of (x_i, y_i). Here x_i belong to a Domain Set $\mathcal{X}$ and can be multi-dimensional. y_i belong to the Label Set $\mathcal{Y}$ that has the outcomes observed in the training set. A supervised learning algorithm analyzes the training data and produces an inferred function, which is called a classifier (if the output is discrete), or a regression function (if the output is continuous). The inferred function should predict the correct output value for any valid input object. This means that the learning algorithm learns and generalizes from the training data and applies the learnt algorithm to unseen situations (known as the test set) in order to predict for the test data.

The test data in supervised learning is another set of m measurements similar to the domain set $\mathcal{X}$ but without the label set. As described above, the goal is to make educated guesses about the labels for the test set by drawing inferences from the training set.

Supervised learning splits into two broad categories: classification and regression.

In classification, the goal is to assign a class (or label) from a finite set of classes to an observation. That is, responses are categorical variables. A classification problem is about determining the optimal hyperplanes that separate the data into different classes. Actuarial applications include the occurrence of an event (car accident, heart attack) in a year in the form of True/False events, classifying the claims into fraudulent or non-fraudulent claims, categorising the claims into varying levels of severity (low/ medium/ high) for investigative purposes. Classification can be binary or having multiple classes.

In regression, the goal is to predict a continuous measurement for an observation. That is, the responses variables are real numbers. Actuarial applications include predicting the claim severity and using it to assign weights, predicting the premium to be charged, predicting the development of a claim.

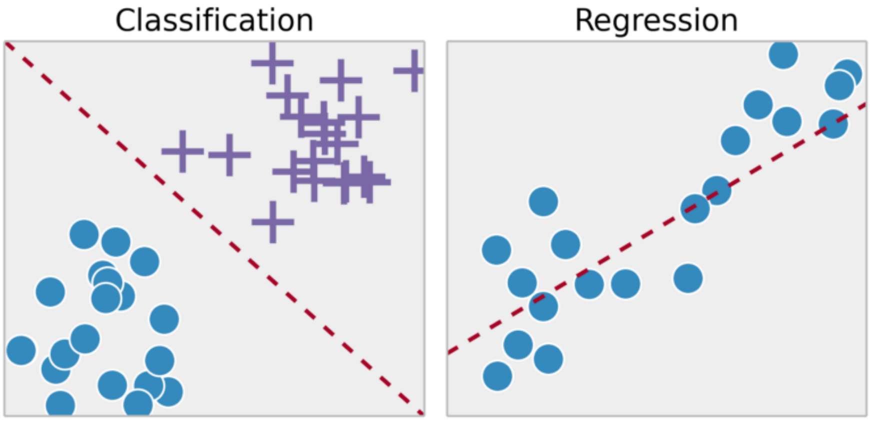

Figure 2: Classification vs Regression

2.1 Framework of Supervised Learning

A general framework of a supervised learning algorithm is described below:-

1. Determine the type of training set needed and gather the training set. This includes deciding the type of examples to be used in the training set. The training set needs to be representative of the real world use of the function. The set needs to cover all possible types of examples, without any known bias. Hence the training data must be large enough and balanced enough. Entries representing each value of the target variable should be a part of the training dataset and there should be several examples of every typical observation.

2. Determine the input feature representation of the learned function. This includes deciding the variables needed. One can start with a very simplistic approach of having the most intuitive variables first and then amending the list of variables. Typically a large number of features which are descriptive of the Label set (objective of the modelling exercise). Finding and exploring new features is critical to creating good or improved models. The number of features should not be too large, in order to avoid the issues with multiple dimensions and risk of over fitting; but should contain enough information to explain the output from all aspects.

3. Determine the structure of the learned function and corresponding learning algorithm. This step is about deciding the modelling methodology. The aim of the learning exercise is to determine a mapping function $f : \mathcal{X} \mapsto \mathcal{Y}$ such that $y_i = f(x_i)$ for all i with minimum error. The goal of any supervised machine learning algorithm is to best estimate the mapping function f for the label set $\mathcal{Y}$ given the input domain set $\mathcal{X}$. The mapping function is often called the target function because it is the function that a given supervised machine learning algorithm aims to approximate.

4. Complete the design. Run the learning algorithm on the gathered training set. Some supervised learning algorithms require the user to determine certain control parameters. These parameters may be adjusted by optimizing performance on a subset (called a validation set) of the training set, or via cross-validation.

5. Evaluate the accuracy of the learned function in terms of the success factors decided to judge the model performance. After parameter adjustment and learning, the performance of the resulting function should be measured on a set that is separate from the training set. This set is known as the holdout data.

2.2 No Free Lunch Theorem

Each supervised learning algorithm has its own strengths and weaknesses. There is no universal learner that performs the job correctly everytime. To this effect, we have the No Free Lunch Theorem.

According to this theorem, for every learner, there exists a task on which it fails even though that task can be successfully learned by another learner. As a result, there is no one model that works best for every problem. The assumptions of a great model for one problem may not hold for another problem, so it is common in machine learning to try multiple models and find one that works best for a particular problem. Depending on the problem, it is important to assess various trade offs of different models, algorithms and data selection methods and then find a combination that works best for that particular problem.

The contribution of No Free Lunch Theorem is that it tells us that choosing an appropriate algorithm requires making assumptions about the kinds of target functions the algorithm is being used for. With no assumptions, no scientific method performs better than random choice.

2.2.1 Bias-Variance tradeoff

Imagine there are several different, but equally good, training data sets. Unless one is careful, the training data set can mislead the learner and result in overfitting, biasness and hence inappropriate prediction. The prediction error for any machine learning algorithm can be broken down into three parts:

- Bias Error
- Variance Error
- Irreducible Error

The irreducible error cannot be reduced regardless of what algorithm is used. It is the error introduced from the chosen framing of the problem and may be caused by factors like unknown variables that influence the mapping of the input variables to the output variable. The tradeoff faced in choosing the right training data is the tradeoff between bias error and variance error.

A learning algorithm is biased for a particular input x if, when trained on each of the possible training data sets, it is systematically incorrect when predicting the correct output for x. A learning algorithm has high variance for a particular input x if it predicts different output values when trained on different training sets. Generally, there is a tradeoff between bias and variance. A learning algorithm with low bias must be flexible so that it can fit the data well. But if the learning algorithm is too flexible, it will fit each training data set differently, and hence have high variance.

So it is essential to determine the tolerance of bias error and variance error when choosing the training data and the modelling methodology.

Generally, parametric algorithms have a high bias making them fast to learn and easier to understand but generally less flexible. In turn, they have lower predictive performance on complex problems that fail to meet the simplifying assumptions of the algorithms bias. Decision trees, nearest problems are examples of low bias models whereas models based on linear regression are high bias models. Non-parametric machine learning algorithms like tree based models often have a low bias but a high variance.

2.2.2 Function complexity and amount of training data

The amount of training data required for modelling is dependent on the relative complexity of the "true" classifier or regression function. If the true function is simple, then an inflexible learning algorithm with high bias and low variance will be able to learn it from a small amount of data. But if the true function is highly complex (e.g., because it involves complex interactions among many different input features and behaves differently in different parts of the input space), then the function will only be learnable from a very large amount of training data and using a flexible learning algorithm with low bias and high variance.

2.2.3 Choice of error

Choice of error or the loss function is a measure of success of a machine learning algorithm. The loss function is a function that represents the deviation of the predicted outcome with the actual outcome. The loss minimization framework is to cast machine learning as an optimization problem. The aim is to minimise the overall loss of the learner rather than minimising example by example.

For a regression problem, the loss is how different the predicted value of the label is from the actual label. A popular and convenient loss function to use in regression is the squared loss, which penalizes the residual of the prediction quadratically. This is given by

$$\mathcal{L}_f(S) = \sum_{x_i \in S} (f(x_i) - y_i)^2$$

Another alternative is the absolute deviation loss, which is the absolute value of the residual. However this loss function suffers from the problem of not being differentiable and hence not easy to be minimised by mathematical techniques.

For a classification problem, error of a classifier is the probability that the learner does not predict the correct label (class) on a random data point from the test set.

$$\mathcal{L}_f(S) = \mathop{\mathbb{P}}_{x_i \in S}(f(x_i) \neq y_i)$$

Since the training sample is the snapshot of the world that is available to the learner, it makes sense to search for a solution that works well on that data. This learning paradigm (coming up with a predictor f that minimizes $\mathcal{L}_f(S)$) is called Empirical Risk Minimization (ERM).

Although the Empirical Risk Minimization rule seems very natural, it possesses the risk of increasing the chances of overfitting.

2.2.4 Noise in the output values

If the output values in the training dataset are known to be incorrect (because of human error or sensor errors), then the learning algorithm should not attempt to find a function that exactly matches the training examples. Instead, the problem should be dealt with two ways:

- Giving credibility weights to the known correct outcomes
- Deciding an error threshold while determining the appropriate model

2.2.5 Bias-Complexity tradeoff

Another tradeoff after the bias-variance tradeoff is the choice of how complex algorithm and modelling method we need. The total error term on a set S can be decomposed as

$$\mathcal{L}_f(S) = \epsilon_{approx} + \epsilon_{est}$$

where ϵ_{approx} stands for approximation error. It is the minimum risk achievable by a predictor in the hypothesis class of all possible mapping functions f. The approximation error does not depend on the sample size and is determined by the hypothesis class chosen. Enlarging the hypothesis class can decrease the approximation error. The difference between the approximation error and the error achieved by the ERM predictor is called the estimation error and is given by ϵ_{est}. It depends on the training set size and on the size, or complexity, of the hypothesis class.

Since our goal is to minimize the total risk, we face a tradeoff, called the bias-complexity tradeoff. On one hand, choosing the hypothesis class to be a very rich class decreases the approximation error but might increase the estimation error, as a rich hypothesis might lead to overfitting. On the other hand, choosing the hypothesis class to be a very small set reduces the estimation error but might increase the approximation error.

2.2.6 Redundancy in the data

If the input features contain redundant information like highly correlated features, some learning algorithms (e.g., linear regression, logistic regression, and distance based methods) will perform poorly because of numerical instabilities and from the fact that these methods assume that the data does not have such redundant information. These problems can often be solved by imposing some form of regularization.

Regularization is a technique that constrains/ regularizes or shrinks the coefficient estimates towards zero. In other words, this technique discourages learning a more complex or flexible model for the features that show redundant information. We will explore more about regularization and its uses in the next few sections.

If it is not possible to ignore the correlations between variables or to regularize them and the interactions between the variables are complex, it is better to use algorithms such as decision trees and neural networks, because they are specifically designed to discover these interactions.

2.2.7 Feature Selection and Dimensionality Reduction

Feature selection is the process of selection of attributes in the data that are most relevant to the modeling problem. It is different from dimensionality reduction. Both methods seek to reduce the number of attributes in the dataset, but a dimensionality reduction method does so by creating new combinations of attributes, where as feature selection methods include and exclude attributes present in the data without transforming them. Principal component analysis is a classic example of dimensionality reduction where the number of attributes are reduced by creating new variables from the existing variables.

Fewer attributes is desirable because it reduces the complexity of the model, and a simpler model is simpler to understand and explain. However explaining a model using variables that were not present in the first place becomes a difficult task, which is why dimensionality reduction needs to be done in a balanced way.

It is essential that feature selection is done as a part of the model selection process rather than first completing the features to be selected and then comparing the models to select one final model, i.e, feature selection must be included within the inner-loop when model accuracy estimation is analysed.

2.2.8 Accuracy Evaluation

The accuracy evaluation needs to be performed on two grounds - the model accuracy is good in terms of the insample and out of sample prediction, and that the accuracy is not too good to be true (the model is not overfitting). The overfitting is normally tested on test data sets (out of sample) and by cross validation wherein different combinations of train and test samples are created from the same dataset and model results are compared. A trained model is not exposed to the test dataset during training and any predictions made on that dataset are designed to be indicative of the performance of the model in general. Cross validation first involves separating the dataset into a number of equally sized groups of instances (called folds). The model is then trained on all folds exception one that was left out and the prepared model is tested on that left out fold. The process is repeated so that each fold gets an opportunity at being left out and acting as the test dataset. If the results are too sensitive to the combination chosen, there is clear evidence of overfitting in the model.

2.3 Parameter tuning

Having understood how to select the best model based on the given problem and addressing the various tradeoffs, the next step in supervised learning is to tune the model hyperparameters. The important thing to understand here is the difference between model parameters and model hyperparameters.

Model parameters are configuration variables that are internal to the model and whose value can be estimated from data. These are learned from the data using optimisation algorithms. Examples are coefficients in linear or generalised linear regression problems, support vectors in support vector machines.

Model hyperparameters on the other hand are configuration variables that are external to the model and whose values are specified by the model developer rather than being estimated from the data. The best values for the hyperparameters are not known in advance but are rather based on either rules of thumb or on trial and error. Examples are number of trees in random forest, sigma in support vector machines.

When a machine learning algorithm is tuned for a specific problem, there are two components to it:

- perform a search (described below in the methods of parameter tuning) on the hyperparameters of the model
- re-estimate the parameters of the model for each combination of the hyperparameters.

With this, the final aim is to obtain the model whose parameters and hyperparameters result in the best predictions.

2.3.1 Grid Search Parameter Tuning

Grid search most basic hyperparameter tuning method. It is an approach to hyperparameter tuning that will methodically build and evaluate a model for each combination of algorithm parameters specified in a grid provided by the developer. However, this is an exhaustive sampling of the hyperparameter space and can be quite inefficient.

2.3.2 Random Search Parameter Tuning

Random search differs from grid search in that we longer provide a discrete set of values to explore for each hyperparameter; rather, we provide a statistical distribution for each hyperparameter from which values may be randomly sampled. A model is constructed and evaluated for each combination of parameters sampled.

2.3.3 Optimisation based Parameter Tuning

Optimisation based methods take the domain of possible hyperparameters the same way as grid search and random search methods, but instead of evaluating on each possibility, it starts with a promising hyperparameter and starts to optimise it (either using Bayesian optimisation methods or gradient descent methods).

2.4 Ensembling Techniques

An ensemble is just a collection of predictors which come together (e.g. mean of all predictions) to give a final prediction. The reason we use ensembles is that many different predictors trying to predict same target variable will perform a better job than any single predictor alone. Ensembling techniques are further classified into Bagging and Boosting.

- **Bagging:** Bagging is a simple ensembling technique in which we build many independent predictors/ models/ learners and combine them using some model averaging techniques. We typically take random sub-sample/bootstrap of data for each model, so that all the models are little different from each other. Each observation has the same probability to appear in all the models. Because this technique takes many uncorrelated learners to make a final model, it reduces error by reducing variance. Example of bagging ensemble is Random Forest models.

- **Boosting:** Boosting is an ensemble technique in which the predictors are not made independently, but sequentially. This technique employs the logic in which the subsequent predictors learn from the mistakes of the previous predictors. Therefore, the observations have an unequal probability of appearing in subsequent models and ones with the highest error appear most. The predictors can be chosen from a range of models like decision trees, regressors, classifiers etc. Because new predictors are learning from mistakes committed by previous predictors, it takes less time/iterations to reach close to actual predictions. But we have to choose the stopping criteria carefully or it could lead to overfitting on training data. Gradient Boosting is an example of boosting algorithm.

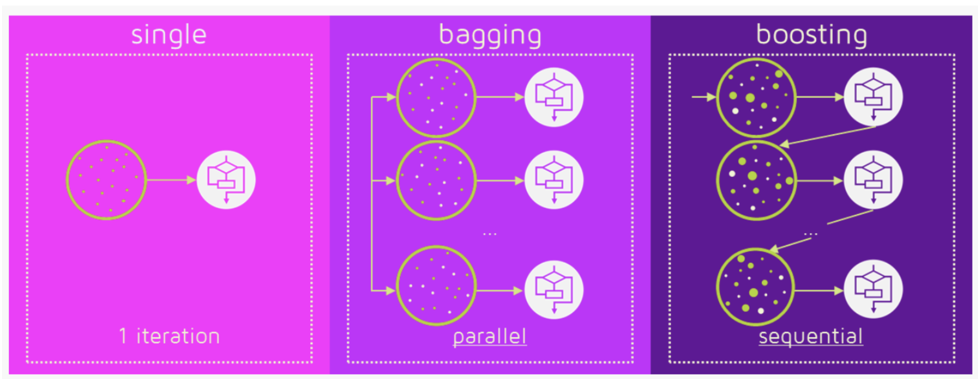

Figure 3: Ensembling

3 Unsupervised Learning

Unsupervised learning seems a much harder technique, since the goal is to have the machine learn how to do something that we don't tell it how to do and how it has been treated historically. An AI system may group unsorted information according to similarities and differences even though there are no categories or target values provided. Unsupervised learning algorithms can perform more complex processing tasks than supervised learning systems since the entire concept of training and testing does not hold here. However, unsupervised learning can be more unpredictable than the alternate model. It might add unforeseen and undesired information to the model, thereby creating clutter instead of order.

In contrast to supervised learning, it is not always easy to come up with metrics for how well an unsupervised learning algorithm is doing. Performance is often subjective and domain-specific. As such the framework and the issues to look out for are very specific to the modelling approach being considered and the data. Also, Unsupervised learning has a lot of subjectivity involved as there is no simple goal for the analysis, such as prediction of a response. Instead, unsupervised learning is considered for exploratory data analysis to extract information about the data that we do not have ourselves. However, unsupervised learning is more practical in the sense that it is closer to the reality. It is often easier to obtain unlabeled data from a lab instrument or a computer than labeled data, which can require human intervention and is subject to bias.

The most common unsupervised learning method is cluster analysis, which is used for exploratory data analysis to find hidden patterns or grouping in data. The clusters are modeled using a measure of similarity which is defined upon metrics such as Euclidean or probabilistic distance.

4 Exploring Some ML Algorithms

Having set the background of machine learning and the ideas behind it, we now move on to exploring some popular machine learning algorithms.

4.1 Random Forest

Random forest is a supervised machine learning algorithm. It is a combination of tree predictors (known as decision trees) such that each tree depends on the values of a random vector (vector of variables or features) sampled independently and with the same distribution for all trees in the forest. To understand how a random forest works, we first explore how decision trees work in general.

4.1.1 An Introduction to Decision Trees

A decision tree is a predictor, $\mathcal{H} : X \mapsto Y$, that predicts the label associated with an instance x by traveling from a root node of a tree to a leaf. It learns in a hierarchical fashion by repeatedly splitting the dataset into separate branches that maximize the information gain of each split.

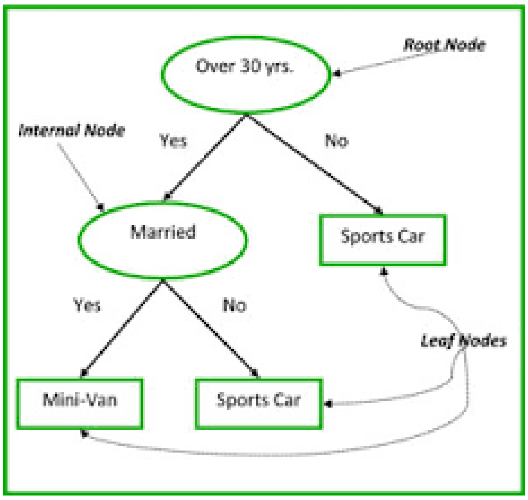

Figure 4: Example of a decision tree

Figure 4 shows a simple example of a decision tree. It show an individual's decision to buy a car based on his age and marital status.

A decision tree consists of decision nodes and leaf nodes. A decision node has two or more branches and specifies the criteria based on which a decision is supposed to be taken. A leaf node represents the decision. The top most decision node is called the root node.

4.1.2 Constructing a decision tree

Decision tree construction is based on heuristics such as a greedy approach, where the tree is constructed gradually, and locally optimal decisions are made at the construction of each node. A general framework for growing a decision tree is as follows. We start with a tree with a single leaf (the root) and assign this leaf a label according to a majority vote among all labels over the training set. We now perform a series of iterations. On each iteration, we examine the effect of splitting a single leaf. We define some gain measure that quantifies the improvement due to this split. Then, among all possible splits, we either choose the one that maximizes the gain and perform it, or choose not to split the leaf at all. A decision tree is dependent on the variable splits made. Therefore, the split decision is an important part of the decision tree construction.

4.1.3 Implementation of the gain measure

The gain can be defined in different ways. The most popular ways of defining the gain measure are:

- **Train Error Gain:** The simplest definition of gain is the decrease in training error. Let the training error before the split be ϵ_{full}. Let the training errors after splitting on feature i be ϵ_{split1} and $\epsilon_{split12}$. Then the gain is defined as

$$Gain = \epsilon_{full} - (\epsilon_{split1} + \epsilon_{split2})$$

- **Information gain:** The information gain is the difference between the entropy of the label before and after the split. Entropy is an indicator of how mixed the data is. It is a function that provides a minimum value if there is the same kind of objects in the set and a maximal value if there is a uniform mixing of objects with different labels (or categories) in the set. Lower the entropy, the easier it is to conclude what decision each node should correspond to. Mathematically entropy for a binary model is defined as

$$E = -p\log_2(p) - (1-p)\log_2(1-p)$$

  where p is the probability of category 1 items in the set.

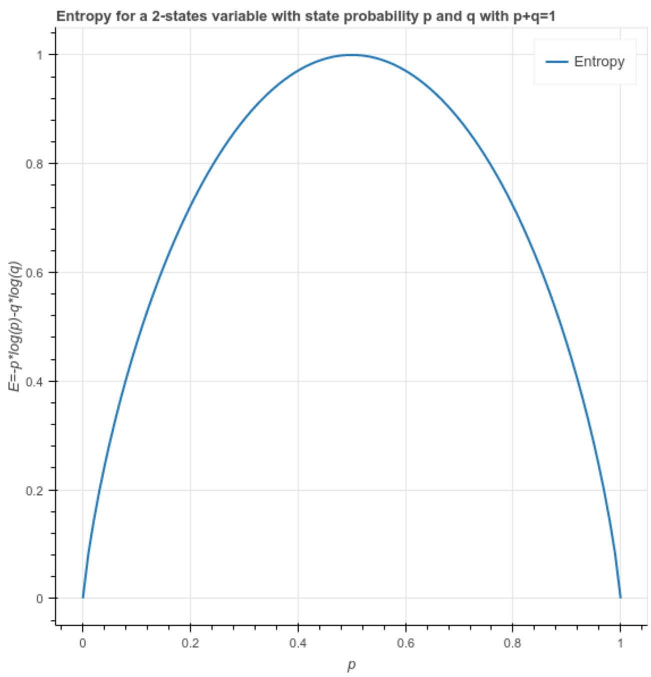

Figure 5: Graph of Entropy as p changes

When the set consists completely of category 1 ($p = 0$) or category 2 (p =1), E is 0 and when the set is uniformly split between category 1 and category 2 (p=0.5), E is 1. See Figure 5 for the complete graph of entropy as p changes.

In the presence of more than 2 labels, the Entropy is defined as

$$E = -\sum_i p_i \log_2 p_i$$

where p_i is the proportion of the elements of each label in the set.

If E_{full} is the entropy of the node before splitting and E_{Split1} and E_{Split2} are the entropies of the split nodes, then the gain is defined as

$$Gain = E_{full} - (E_{split1} + E_{split2})$$

- **Gini Index:** Another definition used for gain is the Gini index. It is the impurity measure of a node and is defined as

  $$Gini = \sum_{i \neq j} p_i p_j$$

  For a perfectly classified node, the Gini index is 0 and for an evenly distributed node, it is $1 - \frac{1}{n}$ where n is the number of labels in the node. Similar to the concept of information value, lower the Gini index, the better it is from a decision perspective.

  If G_{full} is the Gini index of the node before splitting and G_{Split1} and G_{Split2} are the Gini indices of the split nodes, then the gain is defined as

  $$Gain = G_{full} - (G_{split1} + G_{split2})$$

4.1.4 Variable Importance

A decision tree can start with any variable as the root and hence the order in which the variables occur affect the predictions. To understand which variable should occur first, a variable importance analysis is done. A variable is considered to be important if changing its values affects the predictive accuracy. To do this, the variable is permuted over the dataset and the same tree as before is used to predict the data. If the predictions get worse, the variable that was permuted played a significant role in the model and hence had a high importance.

4.1.5 Decision Tree Pruning

When the tree is constructed, it suffers from a major problem - it might be very big and hence might suffer from the risk of being overfitted. To solve this problem, the tree needs to be pruned. Pruning is a technique in machine learning that reduces the size of decision trees by removing sections of the tree that provide little power to classify instances. It reduces the complexity of the final classifier, and hence improves predictive accuracy by the reduction of overfitting.

Pruning can occur in a top down or bottom up fashion. A top down pruning will traverse nodes and trim subtrees starting at the root, while a bottom up pruning will start at the leaf nodes. In a top down pruning approach, an inner (non-root, non-leaf) node t is selected, all the successor nodes of t are deleted and the class most popular in the node is assigned to t. In a bottom up pruning approach, starting from the bottom (except the leaf), a node t is selected and assigned the most popular class. In both the cases, the prediction accuracy is calculated at each step of the pruning process and a choice is made at each node whether to prune it or let it stay as it was. As a result, Each node is replaced with one of its subtrees or with a leaf.

4.1.6 Prediction in a Random Forest

Now that we understand how a decision tree works, a random forest is simply a collection of decision trees. Starting from the root node, a feature is randomly selected and a decision tree is constructed out of it. The prediction of the random forest is obtained by a majority vote over the predictions of the individual trees.

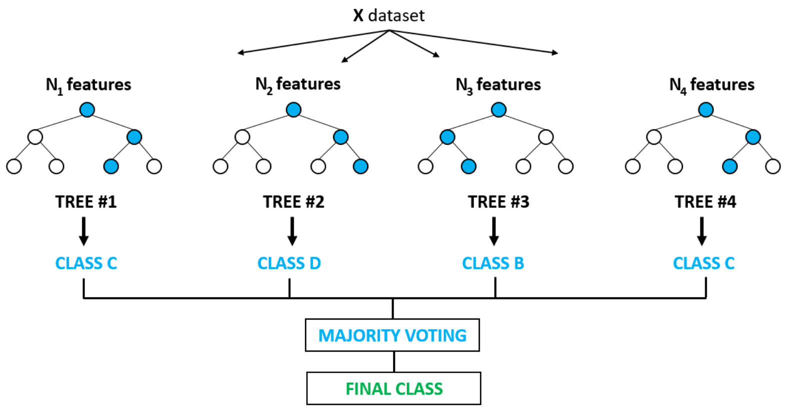

Figure 6: Illustration of Random Forest

4.2 Support Vector Machines

Support vector machines are again supervised learning algorithms that work on the principles of margin and the idea of separating data with a large gap. It is a classification technique in which optimal separating hyperplanes are determined in non-separable data.

The motivation to create support vector machines can be best understood by considering an example of a binary classification. If the data is plotted in the form of a scatterplot, the classification can be done by means of determining a hyperplane passing through the plot such that points on one part of the hyperplane belong to category 1 and the other part belong to category 2. This is as shown in Figure 7.

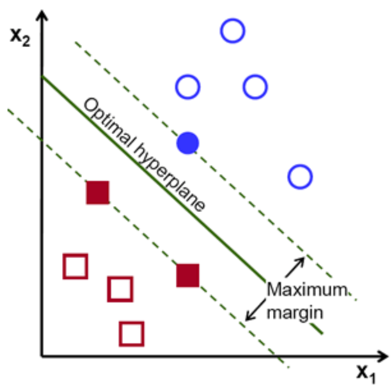

Figure 7: Illustration of Support Vector Machines

It is a non-probabilistic approach to classification since the new points fully determine their location in the scatterplot and hence the category can be determined from the learned model. An important thing to note here is that the hyperplanes need not be linear. Using different mapping functions (known as kernels), support vector machines provide different separating boundaries. If in a feature space some of the sets are not linearly separable and they overlap, then it is necessary to perform a mapping of the original feature space to a higher-dimensional space, in which the separation between the groups is clear. See Figure 8. The original separating boundary is then a non-linear boundary.

4.2.1 Separating Hyperplanes

In a p-dimensional feature space, a linear separating hyperplane is an object in a (p-1)-dimensional space such that it separates the feature space into two regions.

Let a vector x in the feature space be of the form $(x_1, x_2, \ldots, x_p)$. Then determining the hyperplane becomes equivalent to determining the optimal β_0 and $\beta = (\beta_1, \beta_2, \ldots, \beta_p)$ that satisfies

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p = 0$$

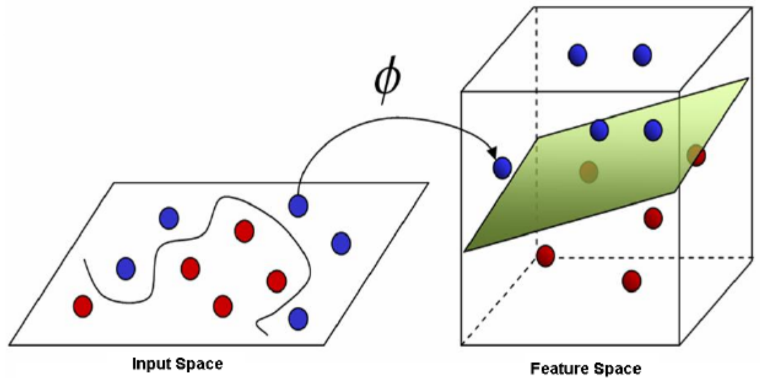

Figure 8: Illustration of overlapping feature space

This can be written as

$$\beta_0 + \beta \cdot x = 0$$

If an element x satisfies the equation, it lies on the hyperplane. If $\beta_0 + \beta \cdot x < 0$, it lies on one part of the plane and if $\beta_0 + \beta \cdot x > 0$, it lies on the other part of the plane. Note that all we need to find to determine which side of the plane x lies in is the sign of $\beta_0 + \beta \cdot x$.

Intuitively, the further from the hyperplane our data points lie, the more confident we are that they have been correctly classified. We therefore want our data points to be as far away from the hyperplane as possible, while still being on the correct side of it.

4.2.2 Maximum Margin

Separating hyperplanes are not unique, since it is possible to slightly rotate a hyperplane and still be able to divide the data into different categories. So it is important that out of the entire universe of possible hyperplanes, the optimal hyperplane is chosen. This brings the concept of maximum margin. Margin is defined as the smallest perpendicular distance of a vector x on the feature space from the separating hyperplane. The maximal margin hyperplane is the separating hyperplane where the margin is the highest.

We note that in the condition $\beta_0 + \beta \cdot x = 0$ for a hyperplane, multiplying the value of β and β_0 with a constant does not change the equation. Hence there can be infinitely many solutions possible for the same hyperplane. To solve this problem, we take β such that it is a unit vector.

Let there be n training observations $x_1, x_2, \ldots, x_n$ and n class labels $y_1, y_2, \ldots, y_n$ where y_i belongs to the set $-1, 1$ and indicates the category x_i belongs to. Let M be the margin. Now the problem of finding the maximal margin hyperplane boils down to an optimisation problem of the form

$$\text{maximise M such that}$$
$$\sum_{j=1}^{p} \beta_j^2 = 1$$
$$y_i(\beta \cdot x_i) \geq M$$

The first condition imposes that β is a unit vector and the second condition makes sure that the correct vectors are on the correct side of the hyperplane and at least as far from the hyperplane as M.

4.2.3 Support Vectors

Support vectors are the data points nearest to the hyperplane, the points of a data set that, if removed, would alter the position of the dividing hyperplane. Support vector machines are maximum-margin classifiers. Hence they find the hyperplane that has the largest perpendicular distance between the hyperplane and the closest samples on either side. The closest samples on either side are the support vectors.

Support vectors are defined as the vectors that are at a margin of 1 unit from the hyperplane. Therefore the vectors x that satisfy the condition $\beta_0 + \beta \cdot x = 1$ or $\beta_0 + \beta \cdot x = -1$ are the support vectors. They are crucial because it is very evident that the maximal margin hyperplane passes through the centre of the support vectors and hence is completely determined by them. Hence it is very sensitive to the support vector locations.

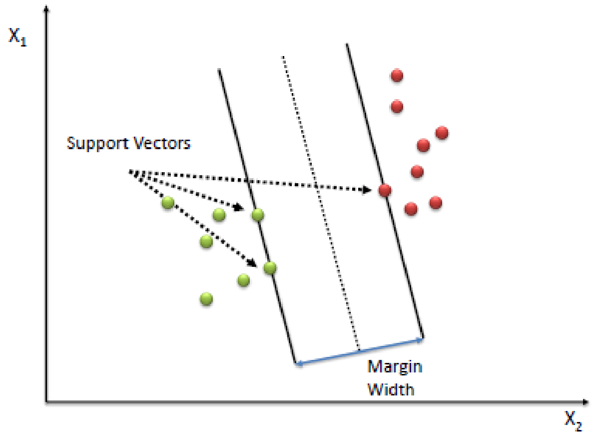

Figure 9: Illustration of Support Vectors and Margin in Support Vector Machines

4.2.4 Kernels

Mathematically, Kernal is a way to map or project one data sequence onto the other in 1-to-1 manner. It is called a generalised dot product function since it is a way of computing the dot product of two vectors x and y in some different (and possibly very high dimensional) feature space.

Support vector machines allow non-separable data to be mapped to another feature space using kernels where the data becomes separable by a hyperplane. This was shown in Figure 8. Kernel functions transform nonlinear spaces into linear ones. There are different possible kernel functions. Some of them are listed below:

- Linear Kernel - maps the points to themselves.
- Polynomial Kernel - maps the points to a d-degree polynomial using a polynomial mapping function.
- Radial Kernel - maps the points to exponentiated distance between the points in the original feature space.
- Sigmoid Kernel - maps the points to a hyperbolic tan space.

But often it is difficult to know beforehand which kernel would work the best. The optimal kernel is more of a trial and error technique. Often the rule of thumb is to start with a linear kernel and then move to complex methods based on how the data appears to be. The radial kernel is often the most used non-linear kernel.

4.2.5 When perfect separation is not possible

Often, the data is so much overlapping that no kernel function can separate the data points perfectly. Or even if we find a kernel function that does so, there is arisk that it might be overfitting the data and that the model might not be robust in case of new data points to be predicted. In such a situation, the algorithm needs to be trained for an acceptable threshold of errors. This threshold is called the budget of the algorithm and is denoted by C. Hence instead of being atleast M distance from the separating hyperplane, each vector x_i is at least $M(1-\epsilon_i)$ from the separating hyperplane such that $\epsilon_i \geq 0$ and

$$\sum_i \epsilon_i \leq C$$

Each ϵ_i denotes where the i^{th} is located relative to the margin and hyperplane. If $\epsilon_i = 0$, this means the the vector x_i is on the correct side of the margin. If $0 < \epsilon_i < 1$, it means that x_i is on the wrong side of the margin but on the correct side of the hyperplane. If $\epsilon_i \geq 1$, it means that x_i is on the wrong side of the hyperplane.

C is a non-negative tuning hyperparameter and controls how much the individual ϵ_i can be modified to violate the margin. C governs the bias-variance tradeoff. A small value of C indicates a low bias but high variance situation, whereas a large value of C indicates high bias and low variance situation.

4.2.6 Prediction in Support Vector Machines

Once the support vectors and separating hyperplanes have been determined, predicting the category of a new vector x using SVM is simply a matter of checking the sign of $\beta_0 + \beta \cdot x$.

4.3 Gradient Boosting Method

Gradient Boosting Method is a supervised learning iterative method which uses the steepest descent algorithm to approximate a function f such that the expected value of the loss function is minimized over the joint distribution of label set Y and domain set X. Hence the method aims to minimise the value of $\mathcal{L}_f$ over the training set and hence find an optimal function f that best approximates the value of Y with minimum loss. The loss is minimised using gradient descent optimisation methods.

At each iteration m of the method, Gradient Boosting assumes that we have a function $F^{(m)}$ that predicts Y. The algorithm does not change $F^{(m)}$ in any way. Instead, it improves it in the next iteration. In the next iteration m+1, it constructs a new model that adds an estimator h such that

$$F^{(m+1)}(X) = F^{(m)}(X) + h(X)$$

A perfect estimator h would mean $F^{(m)}(X) + h(X) = Y$. In other words, a perfect h would be that captures the residual $Y - F^{(m)}$ perfectly. Gradient boosting fits h to the residual. Hence each $F^{(m+1)}$ learns to predict its predecessor $F^{(m)}$. This way, $F^{(m+1)}(X)$ can be written as

$$F^{(m+1)}(X) = F^{(0)}(X) + \sum_{i=1}^{m+1} h_i(X)$$

4.3.1 Gradient Descent

Gradient descent is a first-order iterative optimization algorithm for finding the minimum of a function. To find a local minimum of a function using gradient descent, one takes steps proportional to the negative of the gradient (or approximate gradient) of the function at the current point.

The negative gradient of a function L at the m^{th} iteration is defined as

$$-g_m(X) = -\left[\frac{\partial L(y, F(X))}{\partial F(X)}\right]_{F^{(m-1)}(X)}$$

Gradient descent can be interpreted on the observation that an individual standing on top of a mountain will reach the base fastest if at each point, he moves down in the direction of the steepest slope.

4.3.2 How GBM works

The GBM is an algorithm which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It tries to make the learnt function closer to the true function based on steepest descent algorithm.

The process for GBM is described below:-

1. Fit a simple linear regressor or decision tree (or in fact any model) on the data.
2. Calculate error residuals - Actual target value, minus predicted target value, i.e, $h_1 = y - F^{(1)}(X)$ where $F^{(1)}(X)$ is the predicted vector using the fit in step 1.
3. Fit a new model on error residuals h_1 as target variable with same input variables. The fitted residuals are h_{pred1}.
4. Add the predicted residuals to the previous predictions to obtain $F^{(2)}(X)$, i.e, $F^{(2)}(X) = F^{(1)}(X) + h_{pred1}$.
5. Fit another model on residuals that is still left. i.e. $h_2 = y - F^{(2)}(X)$ and repeat steps 2 to 5 until it starts overfitting or the sum of residuals become constant.

Thus at each iteration, we minimise the error function. Although we can minimize this function directly, gradient descent is a mathematical technique that lets us minimize more complicated error functions that we can't minimise directly. Gradient Boosting Method takes this algorithm to minimise the error function.

4.3.3 Prediction in GBM

A GBM has the advantage that it can be based on numerical optimisation as well as tree based methods, since the ultimate aim is to minimise the errors. For every record for which prediction is needed, the first tree is traversed/ fitted value of the function is calculated. This gives the predicted value of the target variable. Using the residual as a new target variable, grow a second tree/ obtain the second function to predict the residuals from the first tree/ function. Hence a part of the error term that was not explained by the first tree/ function now gets explained from the second tree/ function. The new tree/ function is then weighted by a small weight. This means that the model prediction changes by very small amounts after a new tree/ function is added to the model. It is to reduce the impact of each additional step in the model. The weighting factor is also called shrinkage or learning rate.

GBM faces the issue of overfitting. Hence shrinkage provides the benefit of a late overfitting. Shrinked models have a flatter plateau beyond their error minimas, and it takes them many more iterations to initiate overfitting. Yet, it also means that these models will naturally take longer to learn. In practice one typically chooses the shrinkage parameter λ beforehand and varies the number of iterations M with respect to the chosen shrinkage. One possible approach to choosing the number of iterations M would be to use an information criterion like Akaike's AIC or some sort of minimum description length criteria.

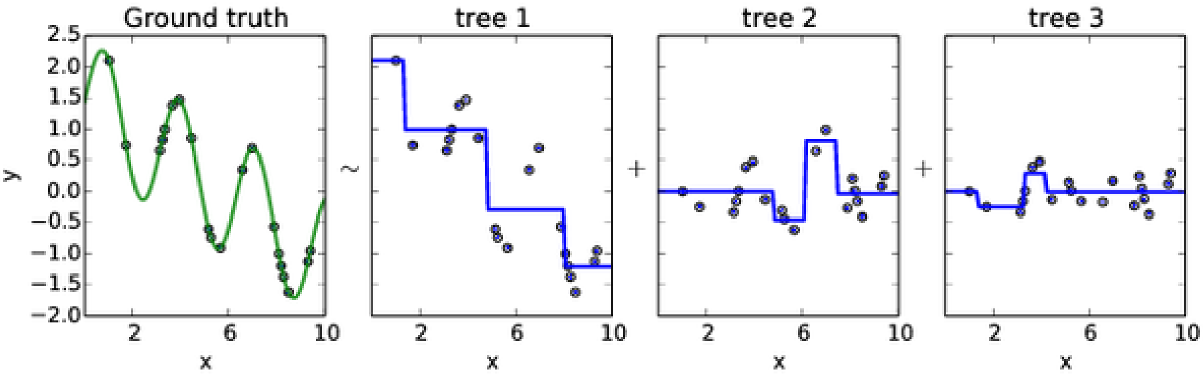

Figure 10: Illustration of prediction using GBM

5 Appropriateness of different Machine Learning algorithms

The No Free Lunch Theorem clearly suggests that there is no perfect model or methodology that fits well in all situations. As a result, it is essential to consider what models to apply based on the appropriateness of the models in the required situation. Accuracy, training time, linearity in the data, number of parameters (and hyperparameters), number of features play an important role in determining which algorithm should be used. Also, the need to get a closed form solution determines whether machine learning algorithms can even be used or not. We discuss the advantages and disadvantages of different methods:

- **Linear Regression:** Linear regression is straightforward to understand and explain. But it performs poorly when there are non-linear relationships. They are not naturally flexible enough to capture more complex patterns, and adding the right interaction terms or polynomials can be tricky and time-consuming.
- **Logistic Regression:** Logistic regression is the classification counterpart to linear regression. Predictions are mapped to be between 0 and 1 through the logistic function, which means that predictions can be interpreted as class probabilities. Outputs have a nice probabilistic interpretation. However just like linear regression, logistic regression tends to underperform when there are multiple or non-linear decision boundaries. They are not flexible enough to naturally capture more complex relationships.
- **Tree based methods:** Decision trees can learn non-linear relationships due to their hierarchial structure, and are fairly robust to outliers. However they are unconstrained. Individual trees are prone to overfitting because they can keep branching until they memorize the training data.
- **Support vector machines:** Support vector machines can model non-linear decision boundaries, and there are many kernels to choose from. They are also fairly robust against overfitting, especially in high-dimensional space. However they are memory intensive, trickier to tune due to the importance of picking the right kernel, and don't scale well to larger datasets.

In practice, many different algorithms for the problem are tried. The drawbacks seen from each method are observed and improvements are thought upon.

6 Conclusion

Actuarial science is the discipline that applies mathematical and statistical methods to assess risk in insurance, finance and other industries and professions. The role of an actuary goes far beyond interpreting just the numbers and rather involves a lot of judgemental inputs and knowledge. This brings the age-old debate of whether black boxes like machine learning can be used in the actuarial domain. Machine learning algorithms provide the advantage of including deeper relationships between variables, speedy analysis, handling large volumes of data and hence lesser expenses. On the other hand, the lack of actuaries who understand the data science methodologies and machine learning algorithms often means that the methodologies are used without appropriate background knowledge. This makes it difficult for them to interpret the results and apply their judgement to them. In order to bridge this gap between the sophisticated methods and potential benefits, actuaries will have to understand and appreciate the growing use of big data and the potential disruptive impacts on the insurance industry. Actuaries will also need to become more proficient with the underlying technology and tools required to use big data in business processes.

References

[1] Blog by Brendor O'Connor
http://brenocon.com/blog/2008/12/statistics-vs-machine-learning-fight.

[2] Understanding Machine Learning: From Theory to Algorithms - Shai Shalev-Shwartz, Shai Ben-David; Cambridge University Press - 2014

[3] An Introduction to Statistical Learning - Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani; Springer Texts in Statistics - 2015

[4] https://medium.com/mlreview/gradient-boosting-from-scratch-1e317ae4587d

[5] https://nlp.stanford.edu/IR-book/html/htmledition/support-vector-machines-the-linearly-separable-case-1.html

[6] A Gentle Introduction to Gradient Boosting - Cheng Li, Northeastern University

[7] https://www.analyticsvidhya.com/blog/2016/04/complete-tutorial-tree-based-modeling-scratch-in-python