Short Summaries

SVM

The objective of SVM algorithm is to find a hyperplane in an N-dimensional space that distinctly classifies the data points. The dimension of the hyperplane depends upon the number of features. If the number of input features is two, then the hyperplane is just a line. If the number of input features is three, then the hyperplane becomes a 2-D plane. It becomes difficult to imagine when the number of features exceeds three.

How do you select kernels

First: Why is the RBF Kernel the most widely used? Because SVM is intrinsically a linear separator when the classes are not linearly separable we can project the data into a high dimensionality space and with a high probability find a linear separation. This is Cover's theorem and the RBF Kernel does exactly that: it projects the data into infinite dimensions and then finds a linear separation.

Advantages of SVM:

- SVM works relatively well when there is a clear margin of separation between classes.
- SVM is more effective in high dimensional spaces.
- SVM is effective in cases where the number of dimensions is greater than the number of samples.
- SVM is relatively memory efficient

Disadvantages of SVM:

- SVM algorithm is not suitable for large data sets.
- SVM does not perform very well when the data set has more noise i.e. target classes are overlapping.
- In cases where the number of features for each data point exceeds the number of training data samples, the SVM will underperform.
- As the support vector classifier works by putting data points, above and below the classifying hyperplane there is no probabilistic explanation for the classification.

Decision Trees

Decision tree is a flowchart-like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.

Construction of Decision Tree: A tree can be “learned” by splitting the source set into subsets based on an attribute value test. This process is repeated on each derived subset in a recursive manner called recursive partitioning. The recursion is completed when the subset at a node all has the same value of the target variable, or when splitting no longer adds value to the predictions. The construction of a decision tree classifier does not require any domain knowledge or parameter setting, and therefore is appropriate for exploratory knowledge discovery. Decision trees can handle high-dimensional data. In general decision tree classifier has good accuracy. Decision tree induction is a typical inductive approach to learn knowledge on classification.

Gini Index:

Gini Index is a score that evaluates how accurate a split is among the classified groups. Gini index evaluates a score in the range between 0 and 1, where 0 is when all observations belong to one class, and 1 is a random distribution of the elements within classes. In this case, we want to have a Gini index score as low as possible. Gini Index is the evaluation metrics we shall use to evaluate our Decision Tree Model.

Advantages of the Decision Tree approach

The strengths of decision tree methods are:

- Decision trees are able to generate understandable rules.
- Decision trees perform classification without requiring much computation.
- Decision trees are able to handle both continuous and categorical variables.
- Decision trees provide a clear indication of which fields are most important for prediction or classification.

Disadvantages of decision tree methods

- Decision trees are less appropriate for estimation tasks where the goal is to predict the value of a continuous attribute.
- Decision trees are prone to errors in classification problems with many classes and a relatively small number of training examples.
- Decision tree can be computationally expensive to train. The process of growing a decision tree is computationally expensive. At each node, each candidate splitting field must be sorted before its best split can be found. In some algorithms, combinations of fields are used and a search must be made for optimal combining weights. Pruning algorithms can also be expensive since many candidate sub-trees must be formed and compared.

Random Forest

The random forest algorithm is an extension of the bagging method as it utilizes both bagging and feature randomness to create an uncorrelated forest of decision trees. Feature

randomness, also known as feature bagging or “the random subspace method”, generates a random subset of features, which ensures low correlation among decision trees. This is a key difference between decision trees and random forests. While decision trees consider all the possible feature splits, random forests only select a subset of those features.

Random forest algorithms have three main hyperparameters, which need to be set before training. These include node size, the number of trees, and the number of features sampled. From there, the random forest classifier can be used to solve for regression or classification problems.

The random forest algorithm is made up of a collection of decision trees, and each tree in the ensemble is comprised of a data sample drawn from a training set with replacement, called the bootstrap sample. Of that training sample, one-third of it is set aside as test data, known as the out-of-bag (oob) sample, which we’ll come back to later. Another instance of randomness is then injected through feature bagging, adding more diversity to the dataset and reducing the correlation among decision trees. Depending on the type of problem, the determination of the prediction will vary. For a regression task, the individual decision trees will be averaged, and for a classification task, a majority vote—i.e. the most frequent categorical variable—will yield the predicted class. Finally, the oob sample is then used for cross-validation, finalizing that prediction.

Key Benefits

- Reduced risk of overfitting: Decision trees run the risk of overfitting as they tend to tightly fit all the samples within training data. However, when there’s a robust number of decision trees in a random forest, the classifier won’t overfit the model since the averaging of uncorrelated trees lowers the overall variance and prediction error.
- Provides flexibility: Since random forest can handle both regression and classification tasks with a high degree of accuracy, it is a popular method among data scientists. Feature bagging also makes the random forest classifier an effective tool for estimating missing values as it maintains accuracy when a portion of the data is missing.
- Easy to determine feature importance: Random forest makes it easy to evaluate variable importance, or contribution, to the model. There are a few ways to evaluate feature importance. Gini importance and mean decrease in impurity (MDI) are usually used to measure how much the model’s accuracy decreases when a given variable is excluded. However, permutation importance, also known as mean decrease accuracy (MDA), is another importance measure. MDA identifies the average decrease in accuracy by randomly permutating the feature values in oob samples.

Key Challenges

- Time-consuming process: Since random forest algorithms can handle large data sets, they can be provide more accurate predictions, but can be slow to process data as they are computing data for each individual decision tree.
- Requires more resources: Since random forests process larger data sets, they'll require more resources to store that data.
- More complex: The prediction of a single decision tree is easier to interpret when compared to a forest of them.

LSTM

It is a variety of recurrent neural networks (RNNs) that are capable of learning long-term dependencies, especially in sequence prediction problems. LSTM has feedback connections, i.e., it is capable of processing the entire sequence of data, apart from single data points such as images. This finds application in speech recognition, machine translation, etc. LSTM is a special kind of RNN, which shows outstanding performance on a large variety of problems.

The central role of an LSTM model is held by a memory cell known as a 'cell state' that maintains its state over time. The cell state is the horizontal line that runs through the top of the below diagram. It can be visualised as a conveyor belt through which information just flows, unchanged.

Information can be added to or removed from the cell state in LSTM and is regulated by gates. These gates optionally let the information flow in and out of the cell. It contains a pointwise multiplication operation and a sigmoid neural net layer that assist the mechanism.

The sigmoid layer gives out numbers between zero and one, where zero means 'nothing should be let through,' and one means 'everything should be let through.'

LSTM vs RNN

Consider, you have the task of modifying certain information in a calendar. To do this, an RNN completely changes the existing data by applying a function. Whereas, LSTM

makes small modifications on the data by simple addition or multiplication that flow through cell states. This is how LSTM forgets and remembers things selectively, which makes it an improvement over RNNs.

Now consider, you want to process data with periodic patterns in it, such as predicting the sales of coloured powder that peaks at the time of Holi in India. A good strategy is to look back at the sales records of the previous year. So, you need to know what data needs to be forgotten and what needs to be stored for later reference. Else, you need to have a really good memory. Recurrent neural networks seem to be doing a good job at this, theoretically. However, they have two downsides, exploding gradient and vanishing gradient, that make them redundant.

Here, LSTM introduces memory units, called cell states, to solve this problem. The designed cells may be seen as differentiable memory.

RNNs are particularly suited for tasks that involve sequences (thanks to the recurrent connections). For example, they are often used for machine translation, where the sequences are sentences or words. In practice, an LSTM is often used, as opposed to a vanilla (or standard) RNN, because it is more computationally effective. In fact, the LSTM was introduced to solve a problem that standard RNNs suffer from, i.e. the vanishing gradient problem. (Now, for these tasks, there are also the transformers, but the question was not about them).

Precision vs Recall

Models need high recall when you need output-sensitive predictions like cancer predictions, terrorist identifications

Similarly, we need high precision in places such as recommendation engines, spam mail detection, etc. Where you don’t care about false negatives but focus more on true positives and false positives. It is ok if spam comes into the inbox folder but a really important mail shouldn’t go into the spam folder.

Generally, if you want higher precision you need to restrict the positive predictions to those with highest certainty in your model, which means predicting fewer positives overall (which, in turn, usually results in lower recall).
If you want to maintain the same level of recall while improving precision, you will need a better classifier.

Oversampling or undersampling will ensure better recall and keep precision manageable

MLE

In statistics, maximum likelihood estimation (MLE) is a method of estimating the parameters of a statistical model given observations, by finding the parameter values that maximize the likelihood of making the observations given the parameters. MLE can be seen as a special case of the maximum a posteriori estimation (MAP) that assumes a uniform prior distribution of the parameters, or as a variant of the MAP that ignores the prior and which therefore is unregularized.

DBSCAN

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is a base algorithm for density-based clustering. It can discover clusters of different shapes and sizes from a large amount of data, which is containing noise and outliers.

The DBSCAN algorithm uses two parameters:

minPts: The minimum number of points (a threshold) clustered together for a region to be considered dense.

eps (ε): A distance measure that will be used to locate the points in the neighborhood of any point.

These parameters can be understood if we explore two concepts called Density Reachability and Density Connectivity.

Reachability in terms of density establishes a point to be reachable from another if it lies within a particular distance (eps) from it.

Connectivity, on the other hand, involves a transitivity based chaining-approach to determine whether points are located in a particular cluster. For example, p and q points could be connected if p->r->s->t->q, where a->b means b is in the neighborhood of a.

There are three types of points after the DBSCAN clustering is complete:

ADVANTAGES	DISADVANTAGES	APPLICATIONS
• Can discover arbitrarily shaped clusters • Find cluster completely surrounded by different clusters. • Robust towards outlier detection (noise) • Require just two points which are very insensitive to the ordering of the points in the database.	• Not partitionable for multiprocessor systems. • Datasets with altering densities are tricky. • Sensitive to clustering parameters minPoints and EPS. • Fails to identify cluster if density varies and if the dataset is too sparse. • Sampling affects density measures.	• Scientific literature • Images of satellite • Crystallography of x-ray • Anomaly detection in temperation data

PCA

The Principal Components are a straight line that captures most of the variance of the data. They have a direction and magnitude. Principal components are orthogonal projections (perpendicular) of data onto lower-dimensional space.

Principal Component Analysis (PCA) is a statistical procedure that uses an orthogonal transformation that converts a set of correlated variables to a set of uncorrelated variables. PCA is the most widely used tool in exploratory data analysis and in machine learning for predictive models. Moreover, PCA is an unsupervised statistical technique used to examine the interrelations among a set of variables. It is also known as a general factor analysis where regression determines a line of best fit.

Advantages of using PCA:

- Removes correlated features. PCA will help you remove all the features that are correlated, a phenomenon known as multi-collinearity. Finding features that are correlated is time consuming, especially if the number of features is large.
- Improves machine learning algorithm performance. With the number of features reduced with PCA, the time taken to train your model is now significantly reduced.
- Reduce overfitting. By removing the unnecessary features in your dataset, PCA helps to overcome overfitting.

Disadvantages of PCA:

- Independent variables are now less interpretable. PCA reduces your features into smaller number of components. Each component is now a linear combination of your original features, which makes it less readable and interpretable.
- Information loss. Data loss may occur if you do not exercise care in choosing the right number of components.

- Feature scaling. Because PCA is a variance maximizing exercise, PCA requires features to be scaled prior to processing.
- PCA is useful in cases where you have a large number of features in your dataset.