An Introduction to 6 Machine Learning Models

by Ricky Ho MVB · Mar. 12, 12 · Database Zone

Whether you work in SQL Server Management Studio or Visual Studio, Redgate tools integrate with your existing infrastructure, enabling you to align DevOps for your applications with DevOps for your SQL Server databases. Discover true Database DevOps, brought to you in partnership with Redgate.

I was motivated to write this blog from a discussion on the Machine Learning Connection group.

For classification and regression problem, there are different choices of Machine Learning Models each of which can be viewed as a blackbox that solve the same problem. However, each model come from a different algorithm approaches and will perform differently under different data set. The best way is to use cross-validation to determine which model perform best on test data.

Here I'll try to provide a high level summary of its underlying algorithmic approach and hopefully can give a sense of whether it will be a good fit for your particular problem.

Decision Tree based methods

The fundamental learning approach is to recursively divide the training data into buckets of homogeneous members through the most discriminative dividing criteria. The measurement of "homogeneity" is based on the output label; when it is a numeric value, the measurement will be the variance of the bucket; when it is a category, the measurement will be the entropy or gini index of the bucket. During the learning, various dividing criteria based on the input will be tried (using in a greedy manner); when the input is a category (Mon, Tue, Wed ...), it will first be turned into binary (isMon, isTue, isWed ...) and then use the true/false as a decision boundary to evaluate the homogeneity; when the input is a numeric or ordinal value, the lessThan, greaterThan at each training data input value will be used as the decision boundary. The training process stops when there is no significant gain in homogeneity by further split the Tree. The members of the bucket represented at leaf node will vote for the prediction; majority wins when the output is a category and member average when the output is a numeric.

The good part of Tree is that it is very flexible in terms of the data type of input and output variables which can be categorical, binary and numeric value. The level of decision nodes also indicate the degree of influences of different input variables. The limitation is each decision boundary at each split point is a concrete binary decision. Also the decision criteria only consider one input attributes at a time but not a combination of multiple input variables. Another weakness of Tree is that once learned it cannot be updated incrementally. When new training data arrives, you have to throw away the old tree and retrain every data from scratch.

However, Tree when mixed with Ensemble methods (e.g. Random Forest, Boosting Trees) addresses a lot of the limitations mentioned above. For example, Gradient Boosting Decision Tree consistently beat the performance of other ML models in many problems and is one of the most popular method these days

Boosting Decision Tree consistently beat the performance of other ML models in many problems and is one of the most popular method these days.

Linear regression based methods

The basic assumption is that the output variable (a numeric value) can be expressed as a linear combination (weighted sum) of a set of input variable (which is also numeric value).
y = w1x1 + w2x2 + w3x3 ....

The whole objective of the training phase is to learn the weights w1, w2 ... by minimizing the error function lost(y, w1x1 + w2x2 + ...). Gradient descent is the classical technique of solving this problem with the general idea of adjusting w1, w2 ... along the direction of the maximum gradient of the loss function.

The input variable is required to be numeric. For binary variable, this will be represented as 0, 1. For categorical variable, each possible value will be represented as a separate binary variable (and hence 0, 1). For the output, if it is a binary variable (0, 1) then a logit function is used to transform the range of -infinity to +infinity into 0 to 1. This is called logistic regression and a different loss function (based on maximum likelihood) is used.

To avoid overfitting, regularization technique (L1 and L2) is used to penalize large value of w1, w2 ... L1 is by adding the absolute value of w1 into the loss function while L2 is by adding the square of w1 into the loss function. L1 has the property that it will penalize redundant features or irrelevant feature more (with very small weight) and is a good tool to select highly influential features.

The strength of Linear model is that it has very high performance in both scoring and learning. The Stochastic gradient descent-based learning algorithm is highly scalable and can handle incremental learning.

The weakness of linear model is linear assumption of input features, which is often false. Therefore, an important feature engineering effort is required to transform each input feature, which usually involved domain expert. Another common way is to throw different transformation functions 1/x, x^2, log(x) in the hope that one of them will have a linear relationship with the output. Linearity can be checked by observing whether the residual (y - predicted_y) is normally distributed or not (using the QQplot with the Gaussian distribution).

Neural Network

Neural Network can be considered as multiple layer of perceptron (each is a logistic regression unit with multiple binary input and one binary output). By having multiple layers, this is equivalent to : z = logit(v1.y1 + v2y2 + ...), while y1 = logit(w11x1 + w12x2 + ...)

This multi-layer model enables Neural Network to learn non-linear relationship between input x and output z. The typical learning technique is "backward error propagation" where the error is propagate from the output layer back to the input layer to adjust the weight.

Notice that Neural Network expect binary input which means we need to transform categorical input into multiple binary variable. For numeric input variable, we can transform that into binary encoded 101010 string. Categorical and numeric output can be transformed in a similar way.

Bayesian Network

It is basically a dependency graph where each node represents a binary variable and each edge (directional) represents the dependency relationship. If NodeA and NodeB has an edge to NodeC. This means the probably of C is true depends on different combinations of the boolean value of A and B. NodeC can point to NodeD and now NodeD depends on NodeA and NodeB as well.

The learning is about finding at each node the join-probability distribution of all incoming edges. This is done by counting the observed values of A, B and C and then update the joint probability distribution table at NodeC.

Once we have the probability distribution table at every node, then we can compute the probability of any hidden node (output variable) from the observed nodes (input variables) by using the Bayes rule.

The strength of Bayesian network is it is highly scalable and can learn incrementally because all we do is to count the observed variables and update the probability distribution table. Similar to Neural Network, Bayesian network expects all data to be binary, categorical variable will need to be transformed into multiple binary variable as described above. Numeric variable is generally not a good fit for Bayesian network.

Support Vector Machine

Support Vector Machine takes numeric input and binary output. It is based on finding a linear plane with maximum margin to separate two class of output. Categorical input can be turned into numeric input as before and categorical output can be modeled as multiple binary output.

With a different lost function, SVM can also do regression (called SVR). I haven't used this myself so I can't talk much.

The strength of SVM is it can handle large number of dimensions. With the kernel function, it can handle non-linear relationship as well.

Nearest Neighbor

We are not learning a model at all. The idea is to find K similar data point from the training set and use them to interpolate the output value, which is either the majority value for categorical output, or average (or weighted average) for numeric output. K is a tunable parameter which needs to be cross-validated to pick the best value.

Nearest Neighbor require the definition of a distance function which is used to find the nearest neighbor. For numeric input, the common practice is to normalize them by minus the mean and divided by the standard deviation. Euclidean distance is commonly used when the input are independent, otherwise mahalanobis distance (which account for correlation between pairs of input features) should be used instead. For binary attributes, Jaccard distance can be used.

The strength of K nearest neighbor is its simplicity as no model needs to be trained. Incremental learning is automatic when more data arrives (and old data can be deleted as well). Data, however, needs to be organized in a distance-aware tree such that finding the nearest neighbor is O(logN) rather than O(N). On the other hand, the weakness of KNN is it doesn't handle high number of dimensions well. Also, the weighting of different factors needs to be hand tuned (by cross-validation on different weighting combination) and can be a very tedious process.

Topics:

Database Partner Resources

Replacing proprietary database with open source? Read the Cost Analysis Guide for a Total Cost of Ownership Comparison
MariaDB

Strategies for modernizing DBMS infrastructure and reducing costs: View IT Market Clock Report
MariaDB

New to NoSQL? Learn the key factors driving NoSQL adoption and top 10 use cases
Couchbase

MySQL vs. MariaDB: Guide to Open Source Database Selection
MariaDB