

|  |
| --- |
| MODEL COMPARISM  2018 |
|  |
| July 12  Authored by: Arijit Raha |

# Learning Algorithm Perspectives

|  |
| --- |
| Model Comparism – Know Your Best Model To get started right away, the application of learning algorithm to a business problem depends on few underlying conceptualizations such as stability, computational time, and the nature of the data.  I will touch upon some of the aspects of ML modelling that is critical in decision making of model selection. |
| *“Alternatively, Consider Maximum No. of Models and Test Them With Selected Performance Metrics.”* |
| Stability and Online Learning - Decision trees are inherently unstable as with slight change in input, not only the output changes but it gives rise to a completely different tree structure. The reason is over fitting and the solution is ensembling technology. It overfits by learning from noise data as well and optimizes for that particular sample, which causes its variable importance order to change significantly. As a result it is not proper for online learning or incremental learning. Any data that does not reflect the structure of the tree will cause the decision tree to fall apart and requires you to rebuild the model again.  Robust to Outliers - Decision Tree are not sensitive to outliers or robust to outliers because the nodes are determined based on the sample proportions (that is cut point range) in each split region and not on their absolute values. So the data is divided into regions and prediction is based on average Y values of that region. An outlier lying in any region will therefore have no role in bringing any change in prediction by itself. Also, the variable and the respective cut points chosen for splitting criteria are with respect to 1. The most important variable – by weights / co-efficient and 2. The cut points that results in homogeneous classification. Now choosing an outlier as cut point will not result in homogeneous classification and the algorithm understands these underlying phenomena well enough to push the outlier into any regions and not as splitting cut points such that the impact of outlier in the model outcome prediction is negligible. Also, since, extreme values or outliers, never cause much reduction in RSS, they are never involved in split. However, that said it also depends on the choice of algorithm around the splitting criteria, for example variance. The outlier will increase the variance which means outlier will have influence in the choice of feature on which node is split into child nodes. Despite of the fact that the variable is not the most important variable in the data set the top variable is defined by variance increased due to outlier which could probably be a potential noise.  Feature Scaling - Scaling is only needed to be done for distance based algorithms. For tree based algorithms, scaling is not required. This algorithm requires partitioning, and are rule based even if you apply Normalization then also the result would be the same. Random Forest is tree based (Decision Trees), which typically uses something similar to if statements, so it doesn't matter whether you scale the columns or not. If you scale the values, the “if” statements will be same, which means it is robust to variance. Also, you're don't get any analogue of a regression coefficient, which measures the relationship between each predictor variable and the response. Because of this, you also don't need to consider how to interpret such coefficients which is something that is affected by variable measurement scales. Because Support Vector Machine (SVM) optimization occurs by minimizing the decision vector w, the optimal hyper plane is influenced by the scale of the input features and it's therefore recommended that data be standardized. We need to perform Feature Scaling when we are dealing with Gradient Descent Based algorithms (Linear and Logistic Regression, Neural Network) and Distance-based algorithms (KNN, K-means, SVM) as these are very sensitive to the range of the data points.    Accuracy - One of the reason for high accuracy for random forest is the embedded feature selection in the model generation process and other reason is variance reduction through ensembling techniques. ANN and deep learning has higher accuracy because it encourages the collection of large data sets and the systematic integration of performance evaluation through back propagation and another factor is integration of feature extraction within the training process.  Imbalance - Because of bootstrap sampling in RF, the random bootstrapped samples will not consider the skewed class distribution for imbalanced classification datasets. In addition, in RF there are two types of class weighting. The first technique is to weight the tree splitting criterion. The other technique is to either oversample or under sample data points during the bootstrap sampling process. The weighted decision tree that is boosted tree applies boosting to reduce the impact of imbalance on impact of imbalance class on accuracy – appropriate chosen measure. How this happens – When a small weight is assigned to a majority class and larger weights to minority / misclassified data , then it has the effect of improving the purity score of a node (bringing class balance) or reducing cost / cost function. Decision tree and Logistic Regression are sensitive to imbalance class unlike RF and ensemblers.  Computation Time - The Naive Bayes classifier employs a very simple (linear) hypothesis function, the function it uses to model data. It suffers from high bias, or error resulting from inaccuracies in its hypothesis class, because its hypothesis function is so simple it cannot accurately represent many complex situations. On the other hand, it exhibits low variance, or failure to generalize to unseen data based on its training set, because its hypothesis class' simplicity prevents it from over fitting to its training data.  It is inaccurate to say the Naive Bayes classifier converges faster than Logistic Regression because the Naive Bayes classifier does not converge at all. Rather than learning its parameters by iteratively tweaking them to minimize a loss function using gradient descent like the vast majority of machine learning models, the Naive Bayes classifier learns it parameters by explicitly calculating them. The Naive Bayes classifier trains faster than logistic regression for this reason; the simple counting the calculation of its parameters consists run much faster than gradient decent.  Increasing learning rate, using RBF instead of sigmoidal function, normalizing and data transformation, and using stochastic learning as opposed to batch learning helps model to converge faster. Stochastic learning is usually much faster than batch learning. SGD is stochastic in nature i.e. it picks up a “random” instance of training data at each step and then computes the gradient making it much faster as there is much fewer data to manipulate at a single time, unlike Batch GD.  Networks learn the fastest from the most unexpected sample. Therefore, it is advisable to choose a sample at each iteration that is the most unfamiliar to the system. Shuffle the training set so that successive training examples never (rarely) belong to the same class. Convergence is usually faster if the average of each input variable over the training set is close to zero. Input variables should be uncorrelated if possible for faster convergence.  One of the primary reasons popular libraries SVM algorithms are slow is because they are not incremental. They require the entire dataset to be in RAM all at once. So if you have a million data points, it's going to run kind of slow. SVM works well with high dimensional data because of kernel trick but perform badly when applied to large dataset because the training complexity of SVM is highly dependent on the size of data set. Storing the kernel matrix requires memory that scales quadratically with the number of data points. Training time for traditional SVM algorithms also scales super linearly with the number of data points. So, these algorithms feasible aren't for large data sets.  Scalability - Random forests is great with high dimensional data since we are working with subsets of data. It is faster to train than decision trees because we are working only on a subset of features in this model, so we can easily work with hundreds of features.  *With these few things in mind it might become relatively easy to find meaning in selecting appropriate learning algorithm.* |