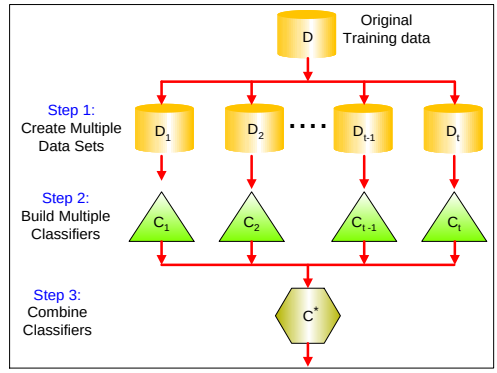
ENSEMBLE METHODS

EMMANUEL JAKOBOWICZ

ENSEMBLE METHODS

- Definition:
  - In statistics and machine learning, ensemble methods use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms
- One of the eager methods => builds model over the training set
- Construct a set of classifiers from the training data
- Predict class label of previously unseen records by aggregating predictions made by multiple classifiers

GENERAL IDEA

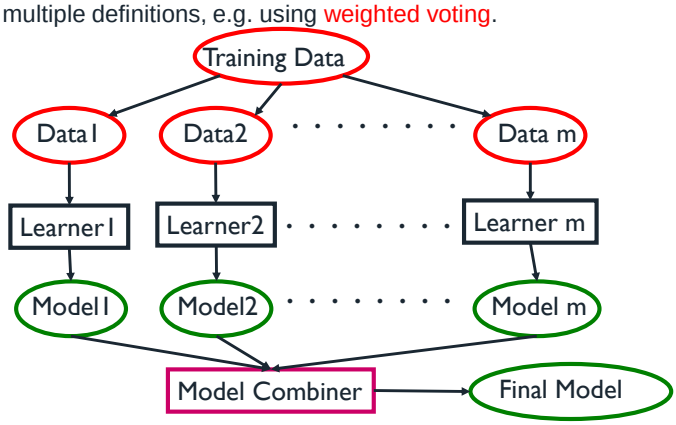


ENSEMBLE LEARNING

- So far – learning methods that learn a single hypothesis, chosen form a hypothesis space that is used to make predictions.
- Ensemble learning → select a collection (ensemble) of hypotheses and combine their predictions.
- Example 1 - generate 100 different decision trees from the same or different training set and have them vote on the best classification for a new example.
- Key motivation: reduce the error rate. Hope is that it will become much more unlikely that the ensemble of will misclassify an example.

LEARNING ENSEMBLES

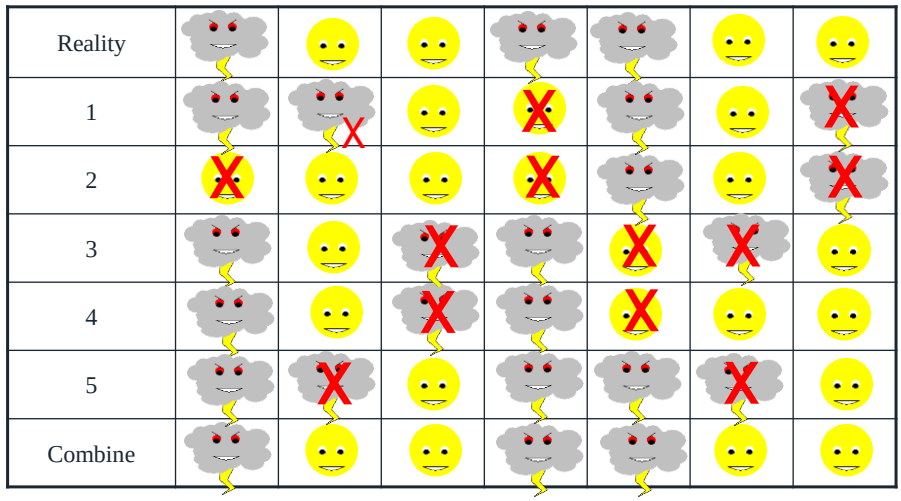
- Learn multiple alternative definitions of a concept using different training data or different learning algorithms.
- Combine decisions of multiple definitions, e.g. using weighted voting.



VALUE OF ENSEMBLES

- "No Free Lunch" Theorem
  - No single algorithm wins all the time!
- When combing multiple independent and diverse decisions each of which is at least more accurate than random guessing, random errors cancel each other out, correct decisions are reinforced.
- Examples: Human ensembles are demonstrably better
  - How many jelly beans in the jar?: Individual estimates vs. group average.
  - Who Wants to be a Millionaire: Audience vote.

EXAMPLE: WEATHER FORECAST



Intuitions

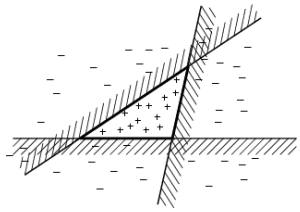
- Majority vote
- Suppose we have 5 completely independent classifiers…
  - If accuracy is 70% for each
    - $(.7^5)+5(.7^4)(.3)+ 10 (.7^3)(.3^2)$
    - **83.7% majority vote accuracy**
  - 101 such classifiers
    - **99.9% majority vote accuracy**

Note: Binomial Distribution: The probability of observing *x* heads in a sample of *n* independent coin tosses, where in each toss the probability of heads is *p*, is

$$P(X = x|p, n) = \frac{n!}{r!(n-x)!}p^x(1-p)^{n-x}$$

ENSEMBLE LEARNING

- Another way of thinking about ensemble learning:
- → way of enlarging the hypothesis space, i.e., the ensemble itself is a hypothesis and the new hypothesis space is the set of all possible ensembles constructible form hypotheses of the original space.



Increasing power of ensemble learning:

Three linear threshold hypothesis
(positive examples on the non-shaded side);
Ensemble classifies as positive any example classified positively be all three. The resulting triangular region hypothesis is not expressible in the original hypothesis space.

DIFFERENT LEARNERS

- Different learning algorithms
- Algorithms with different choice for parameters
- Data set with different features
- Data set = different subsets

HOMOGENOUS ENSEMBLES

- Use a single, arbitrary learning algorithm but manipulate training data to make it learn multiple models.
  - Data1 ≠ Data2 ≠ … ≠ Data m
  - Learner1 = Learner2 = … = Learner m
- Different methods for changing training data:
  - Bagging: Resample training data
  - Boosting: Reweight training data

AN EXAMPLE: SPAM FILTERING

- Problem: Set rules for classification of spam emails
- Solution: We can generate various rules for classification of spam emails
- Spam
  - Have total length less than 20 words
  - Have only image (promotional images)
  - Have specific key words like “make money and grow” and “reduce your fat”
  - More miss spelled words in the email
  - Not Spam
    - Email from known domain
    - Email from family members or anyone from e-mail address book
- Do you think that all these rules individually can predict the correct class?
- Combining these rules will provide robust prediction as compared to prediction done by individual rules.
- Random forest algorithm (having multiple CART models) performs better compared to individual CART model by classifying a new object where each tree gives “votes” for that class and the forest chooses the classification having the most votes (over all the trees in the forest). In case of regression, it takes the average of outputs of different trees.

EXAMPLES OF ENSEMBLE METHODS

- Most famous methods for ensemble learning:
  - Bagging
  - Boosting
  - Random Forests

BAGGING: BOOTSTRAP AGGREGATING

- Bootstrap: data resampling
  - Generate multiple training sets
    - Resample the original training data
    - With replacement
  - Data sets have different “specious” patterns
- Sampling with replacement
  - Each sample has probability $(1 – 1/n)^n$ of being selected
- Build classifier on each bootstrap sample
  - Specious patterns will not correlate
- Underlying true pattern will be common to many
- Combine the classifiers: Label new test examples by a majority vote among classifiers

BAGGING

- Create ensembles by "*bootstrap aggregation*", i.e., repeatedly randomly resampling the training data (Brieman, 1996).
- Bootstrap: draw N items from X with replacement
- Bagging
  - Train *M* learners on *M* bootstrap samples
  - Combine outputs by voting (e.g., majority vote)
- Decreases error by decreasing the variance in the results due to ***unstable learners***, algorithms (like decision trees and neural networks) whose output can change dramatically when the training data is slightly changed.

BAGGING - AGGREGATE BOOTSTRAPPING

- Given a standard training set *D* of size *n*
- For i = 1 .. M
  - Draw a sample of size $n^* < n$ from *D* uniformly and with replacement
  - Learn classifier C_i
- Final classifier is a vote of C_1 .. C_M
- Increases classifier stability/reduces variance

STRONG AND WEAK LEARNERS

- Strong Learner →Objective of machine learning
  - Take labeled data for training
  - Produce a classifier which can be *arbitrarily accurate*
- Weak Learner
  - Take labeled data for training
  - Produce a classifier which is more accurate than random guessing

BAGGING WITH PYTHON

- In scikit-learn, bagging methods are offered as a unified BaggingClassifier meta-estimator (resp. BaggingRegressor)
- taking as input a user-specified base estimator along with parameters specifying the strategy to draw random subsets.
- In particular, max_samples and max_features control the size of the subsets (in terms of samples and features), while bootstrap and bootstrap_features control whether samples and features are drawn with or without replacement.
- When using a subset of the available samples the generalization error can be estimated with the out-of-bag samples by setting oob_score=True.
- As an example, the snippet below illustrates how to instantiate a bagging ensemble of KNeighborsClassifier base estimators, each built on random subsets of 50% of the samples and 50% of the features.

```
from sklearn.ensemble import BaggingClassifier
from sklearn.neighbors import KNeighborsClassifier
bagging = BaggingClassifier(KNeighborsClassifier(), max_samples=0.5, max_features=0.5)
```

Try to do bagging with a different algorithm on a titanic dataset

BOOSTING

- Weak Learner: only needs to generate a hypothesis with a training accuracy greater than 0.5, i.e., < 50% error over any distribution
- Learners
  - Strong learners are very difficult to construct
  - Constructing weaker Learners is relatively easy
- Questions: Can a set of **weak learners** create a single **strong learner** ?

  YES ☺

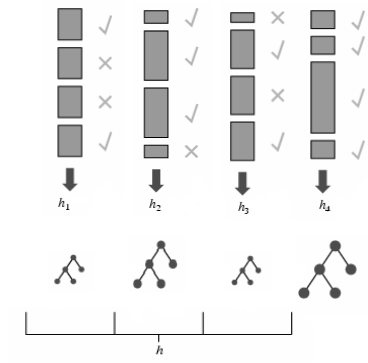
  Boost weak classifiers to a strong learner

BOOSTING

- Originally developed by computational learning theorists to guarantee performance improvements on fitting training data for a ***weak learner*** that only needs to generate a hypothesis with a training accuracy greater than 0.5 (Schapire, 1990).
- Revised to be a practical algorithm, AdaBoost, for building ensembles that empirically improves generalization performance (Freund & Shapire, 1996).
- Key Insights
- Instead of sampling (as in bagging) re-weigh examples!
- Examples are given weights. At each iteration, a new hypothesis is learned (weak learner) and the examples are reweighted to focus the system on examples that the most recently learned classifier got wrong.
- Final classification based on weighted vote of weak classifiers

ADAPTIVE BOOSTING

- Each rectangle corresponds to an example,
- with weight proportional to its height.
- Crosses correspond to misclassified examples.
- Size of decision tree indicates the weight of that hypothesis in the final ensemble.



CONSTRUCT WEAK CLASSIFIERS

- Using Different Data Distribution
  - Start with uniform weighting
  - During each step of learning
    - Increase weights of the examples which are not correctly learned by the weak learner
    - Decrease weights of the examples which are correctly learned by the weak learner
- Idea
  - Focus on difficult examples which are not correctly classified in the previous steps

COMBINE WEAK CLASSIFIERS

- Weighted Voting
  - Construct strong classifier by weighted voting of the weak classifiers
- Idea
  - Better weak classifier gets a larger weight
  - Iteratively add weak classifiers
    - Increase accuracy of the combined classifier through minimization of a cost function

ADAPTIVE BOOSTING: HIGH LEVEL DESCRIPTION

- C =0; /* counter*/
- M = m; /* number of hypotheses to generate*/

- 1. Set same weight for all the examples (typically each example has weight = 1);

- 2. While (C < M)
- 2.1 Increase counter C by 1.
- 2.2 Generate hypothesis h_C .
- 2.3 Increase the weight of the misclassified examples in hypothesis h_C
- 3. Weighted majority combination of all M hypotheses (weights according to how well it performed on the training set).

- Many variants depending on how to set the weights and how to combine the hypotheses. ADABOOST → quite popular!!!!

PERFORMANCE OF ADABOOST

- Learner = Hypothesis = Classifier
- Weak Learner: < 50% error over any distribution
- M number of hypothesis in the ensemble.
- If the input learning is a Weak Learner, then ADABOOST will return a hypothesis that classifies the training data perfectly for a large enough M, boosting the accuracy of the original learning algorithm on the training data.
- Strong Classifier: thresholded linear combination of weak learner outputs.

BOOSTING

- An iterative procedure to adaptively change distribution of training data by focusing more on previously misclassified records
  - Initially, all N records are assigned equal weights
  - Unlike bagging, weights may change at the end of boosting round
- The final classifier is the weighted combination of the weak classifiers.

BOOSTING

- Records that are wrongly classified will have their weights increased
- Records that are classified correctly will have their weights decreased

HOW CAN WE IDENTIFY THE WEIGHTS OF DIFFERENT MODELS FOR ENSEMBLE?

- One of the most common challenge with ensemble modeling is to find optimal weights to ensemble base models.
- In general, we assume equal weight for all models and takes the average of predictions. But, is this the best way to deal with this challenge?
- There are various methods to find the optimal weight for combining all base learners. These methods provide a fair understanding about finding the right weight. I am listing some of the methods below:
- Find the collinearity between base learners and based on this table, then identify the base models to ensemble.
- After that look at the cross validation score (ratio of score) of identified base models to find the weight.

- Find the algorithm to return the optimal weight for base learners. You can refer article Finding Optimal Weights of Ensemble Learner using Neural Network to look at the method to find optimal weight.
- We can also solve the same problem using methods like:
  - Forward Selection of learners
  - Selection with Replacement
  - Bagging of ensemble methods

WHAT ARE THE BENEFITS OF ENSEMBLE MODEL?

- There are two major benefits of Ensemble models:
  - Better prediction
  - More stable model
  - The aggregate opinion of a multiple models is less noisy than other models.

- In finance, we called it “Diversification” a mixed portfolio of many stocks will be much less variable than just one of the stocks alone. This is also why your models will be better with ensemble of models rather than individual. One of the caution with ensemble models are over fitting although bagging takes care of it largely.

NETFLIX EXAMPLE

NETFLIX

Users rate movies (1,2,3,4,5 stars);
Netflix makes suggestions to users based on previous rated movies.

Netflix Prize

http://www.netflixprize.com/index

"The Netflix Prize seeks to substantially improve the accuracy of predictions about how much someone is going to love a movie based on their movie preferences. Improve it enough and you win one (or more) Prizes. Winning the Netflix Prize improves our ability to connect people to the movies they love."

Netflix Prize

http://www.netflixprize.com/

- Supervised learning task
  - Training data is a set of users and ratings (1,2,3,4,5 stars) those users have given to movies.
  - Construct a classifier that given a user and an unrated movie, correctly classifies that movie as either 1, 2, 3, 4, or 5 stars

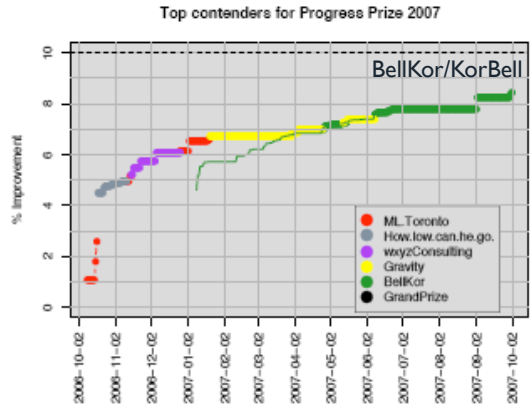
$1 million prize for a 10% improvement over Netflix's current movie recommender/classifier (MSE = 0.9514)

BellKor / KorBell

Scores of the leading team for the first 12 months of the Netflix Prize.

Colors indicate when a given team had the lead. The % improvement is over Netflix' Cinematch algorithm.

The million dollar Grand Prize level is shown as a dotted line at 10% improvement.



"Our final solution (RMSE=0.8712) consists of blending 107 individual results."

The BellKor solution to the Netflix Prize

Robert M. Bell, Yehuda Koren and Chris Volinsky
AT&T Labs – Research
BellKor@research.att.com

Our final solution (RMSE=0.8712) consists of blending 107 individual results. Since many of these results are close variants, we first describe the main approaches behind them. Then, we will move to describing each individual result.

The core components of the solution are published in our ICDM'2007 paper [1] (or, KDD-Cup'2007 paper [2]), and also in the earlier KDD'2007 paper [3]. We assume that the reader is familiar with these works and our terminology there.

Neighborhood-based model (k-NN)

A movie-oriented k-NN approach was thoroughly described in our KDD-Cup'2007 paper [kNN]. We apply it as a post-processor for most other models. Interestingly, it was most effective when applied on residuals of RBMs [5], thereby driving the Quiz RMSE from 0.9093 to 0.8888.
An earlier k-NN approach was described in the KDD'2007 paper ([3], Sec. 3) [Slow-kNN]. It appears that this earlier approach can achieve slightly more accurate results than the newer one, at the expense of a significant increase in running time. Consequently, we dropped the older approach, though some results involving it survive within the final blend.
We also tried more naïve k-NN models, where interpolation weights are based on pairwise similarities between movies (see [2], Sec. 2.2). Specifically, we based weights on $corr^2/(1-corr^2)$ [Corr-kNN], or on $mse^{-1.0}$ [MSE-kNN]. Here, $corr$ is the Pearson correlation coefficient between the two respective movies, and mse is the mean squared distance between two movies (see definition of s_{ij} in Sec. 4.1 of [2]). We also tried taking the interpolation weights as the "support-based similarities", which will be defined shortly [Supp-kNN].

Netflix Prize

2008-11-30

Leaderboard

Display top 40 leaders.

Rank	Team Name	Best Score	% Improvement	Last Submit Time
--	No Grand Prize candidates yet	--	--	--
Grand Prize - RMSE <= 0.8563				
1	BellKor in BigChaos	0.8613	9.47	2008-11-30 19:01:34
Progress Prize - RMSE <= 0.8625				
2	BigChaos	0.8626	9.33	2008-12-01 22:24:12
3	BellKor	0.8631	9.28	2008-11-30 18:44:54
4	PragmaticTheory	0.8638	9.21	2008-11-28 11:46:23
5	Gravity	0.8654	9.04	2008-11-27 21:18:37
6	My Brain and His Chain	0.8668	8.89	2008-09-30 02:19:47
7	Just a guy in a garage	0.8673	8.84	2008-11-26 17:00:27
8	When Gravity and Dinosaurs Unite	0.8675	8.82	2008-10-05 14:16:53
9	Opera Solutions	0.8676	8.81	2008-12-02 22:08:45
10	acmehill	0.8677	8.80	2008-11-26 10:15:28
11	scientist	0.8677	8.80	2008-12-02 01:10:13
12	Ces	0.8711	8.44	2008-08-25 05:00:23
Progress Prize 2007 - RMSE = 0.8712 - Winning Team: KorBell				
13	KorBell	0.8712	8.43	2007-10-01 23:25:23
14	basho	0.8714	8.41	2008-05-21 22:06:00

WHY DO ENSEMBLES WORK?

Dietterich(2002) showed that ensembles overcome three problems:

- ***The Statistical Problem*** arises when the hypothesis space is too large for the amount of available data. Hence, there are many hypotheses with the same accuracy on the data and the learning algorithm chooses only one of them! There is a risk that the accuracy of the chosen hypothesis is low on unseen data!
- ***The Computational Problem*** arises when the learning algorithm cannot guarantees finding the best hypothesis.
- ***The Representational Problem*** arises when the hypothesis space does not contain any good approximation of the target class(es).

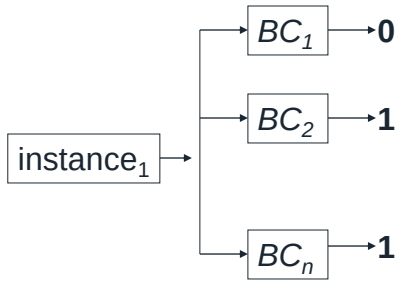
The statistical problem and computational problem result in the variance component of the error of the classifiers!

The representational problem results in the bias component of the error of the classifiers!

STACKING

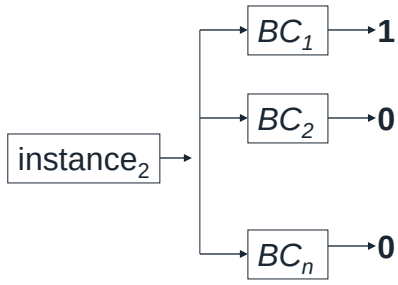
- Uses *meta learner* instead of voting to combine predictions of base learners
  - Predictions of base learners (*level-0 models*) are used as input for meta learner (*level-1 model)*
- Base learners usually different learning schemes
- Hard to analyze theoretically: “black magic”

Stacking



meta instances	BC_1	BC_2	...	BC_n	Class
instance$_1$	0	1		1	1

Stacking



meta instances	BC_1	BC_2	...	BC_n	Class
instance$_1$	0	1		1	1
instance$_2$	1	0		0	0

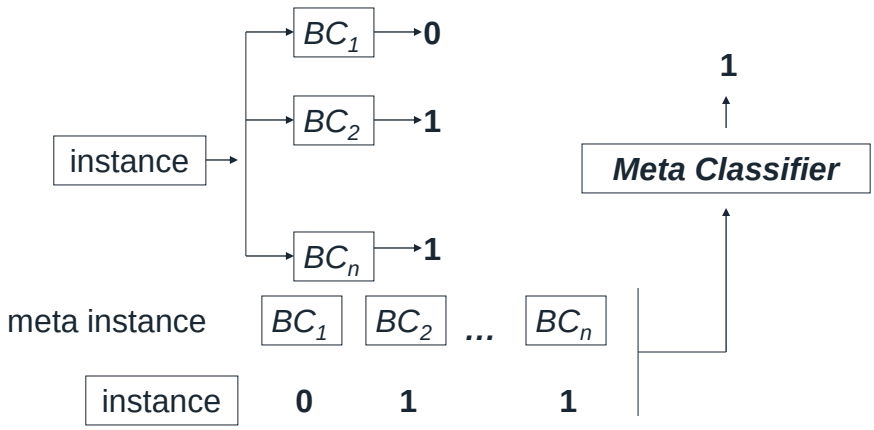
Stacking

Meta Classifier

⇧

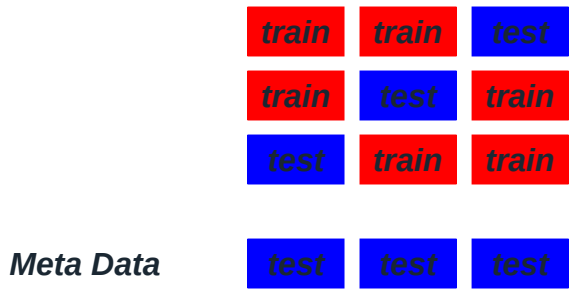
meta instances	BC_1	BC_2	…	BC_n	Class
$instance_1$	**0**	**1**		**1**	**1**
$instance_2$	**1**	**0**		**0**	**0**

Stacking



MORE ON STACKING

- Predictions on training data can't be used to generate data for level-1 model! The reason is that the level-0 classifier that better fit training data will be chosen by the level-1 model! Thus,
- k-fold cross-validation-like scheme is employed! An example for k = 3!



SOME PRACTICAL ADVICES

- If the classifier is unstable (high variance), then apply bagging!
- If the classifier is stable and simple (high bias) then apply boosting!
- If you have many classes and a binary classifier then try error-correcting codes! If it does not work then use a complex binary classifier!

RANDOM FOREST

DEFINITION

- **Random forest** (or **random forests**) is an ensemble classifier that consists of many decision trees and outputs the class that is the mode of the class's output by individual trees.
- The term came from **random decision forests** that was first proposed by Tin Kam Ho of Bell Labs in 1995.
- The method combines Breiman's "bagging" idea and the random selection of features.

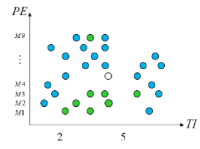
DECISION TREES

- Decision trees are individual learners that are combined. They are one of the most popular learning methods commonly used for data exploration.
- One type of decision tree is called CART… classification and regression tree.
- CART … greedy, top-down binary, recursive partitioning, that divides feature space into sets of disjoint rectangular regions.
  - Regions should be pure wrt response variable
  - Simple model is fit in each region – majority vote for classification, constant value for regression.

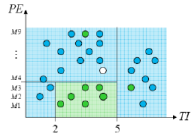
DECISION TRESS INVOLVE GREEDY, RECURSIVE PARTITIONING.

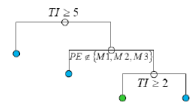
- Simple dataset with two predictors

TI	PE	Response
1.0	M2	good
2.0	M1	bad
...	...	...
4.5	M6	?



- Greedy, recursive partitioning along TI and PE





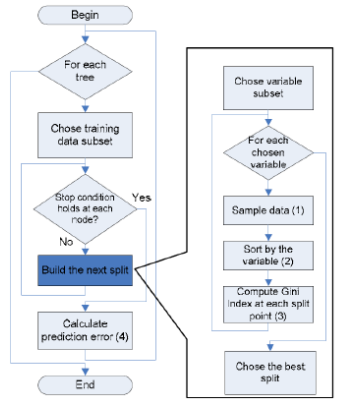
ALGORITHM

Each tree is constructed using the following algorithm:

1. Let the number of training cases be *N*, and the number of variables in the classifier be *M*.
2. We are told the number *m* of input variables to be used to determine the decision at a node of the tree; *m* should be much less than *M*.
3. Choose a training set for this tree by choosing *n* times with replacement from all *N* available training cases (i.e. take a bootstrap sample). Use the rest of the cases to estimate the error of the tree, by predicting their classes.
4. For each node of the tree, randomly choose *m* variables on which to base the decision at that node. Calculate the best split based on these *m* variables in the training set.
5. Each tree is fully grown and not pruned (as may be done in constructing a normal tree classifier).

For prediction a new sample is pushed down the tree. It is assigned the label of the training sample in the terminal node it ends up in. This procedure is iterated over all trees in the ensemble, and the average vote of all trees is reported as random forest prediction.

ALGORITHM FLOW CHART



RANDOM FOREST – PRACTICAL CONSIDERATION

- Splits are chosen according to a purity measure:
  - E.g. squared error (regression), Gini index or devinace (classification)
- How to select N?
  - Build trees until the error no longer decreases
- How to select M?
  - Try to recommend defaults, half of them and twice of them and pick the best.

FEATURES AND ADVANTAGES

The advantages of random forest are:

- It is one of the most accurate learning algorithms available. For many data sets, it produces a highly accurate classifier.
- It runs efficiently on large databases.
- It can handle thousands of input variables without variable deletion.
- It gives estimates of what variables are important in the classification.
- It generates an internal unbiased estimate of the generalization error as the forest building progresses.
- It has an effective method for estimating missing data and maintains accuracy when a large proportion of the data are missing.

FEATURES AND ADVANTAGES

- It has methods for balancing error in class population unbalanced data sets.
- Generated forests can be saved for future use on other data.
- Prototypes are computed that give information about the relation between the variables and the classification.
- It computes proximities between pairs of cases that can be used in clustering, locating outliers, or (by scaling) give interesting views of the data.
- The capabilities of the above can be extended to unlabeled data, leading to unsupervised clustering, data views and outlier detection.
- It offers an experimental method for detecting variable interactions.

DISADVANTAGES

- Random forests have been observed to overfit for some datasets with noisy classification/regression tasks.
- For data including categorical variables with different number of levels, random forests are biased in favor of those attributes with more levels. Therefore, the variable importance scores from random forest are not reliable for this type of data.

RF - ADDITIONAL INFORMATION

Estimating the test error:

- While growing forest, estimate test error from training samples
- For each tree grown, 33-36% of samples are not selected in bootstrap, called out of bootstrap (OOB) samples
- Using OOB samples as input to the corresponding tree, predictions are made as if they were novel test samples
- Through book-keeping, majority vote (classification), average (regression) is computed for all OOB samples from all trees.
- Such estimated test error is very accurate in practice, with reasonable N

CONCLUSIONS & SUMMARY:

- Fast!
  - RF is fast to build. Even faster to predict!
  - Practically speaking, not requiring cross-validation alone for model selection significantly speeds training by 10x-100x or more.
  - Fully parallelizable … to go even faster!
- Automatic predictor selection from large number of candidates
- Resistance to over training
- Ability to handle data without preprocessing
  - data does not need to be rescaled, transformed, or modified
  - resistant to outliers
  - automatic handling of missing values
- Cluster identification can be used to generate tree-based clusters through sample proximity

APPLICATION

- Run a random forrest model on the titanic dataset
- Compare the results based on hyper-parmeters
- Use scikit-learn from python