Machine Learning: Ensemble Learning

Introduction to Ensemble Learning

Problem!

- Decision trees discussed earlier suffer from high variance!
  - If we randomly split the training data into 2 parts, and fit decision trees on both parts, the results could be quite different
- We would like to have models with low variance
- To solve this problem, we can use Ensemble methods

Ensemble Learning - Definition

- Ensemble learning is a process that uses a set of models, each of them obtained by applying a learning process to a given problem. This set of models(ensemble) is integrated in some way to obtain final prediction

- Aggregation of multiple learned models with the goal of improving accuracy

- Intuition: Simulate what we do when we combine an expert panel in a human decision-making process.

Types of ensembles

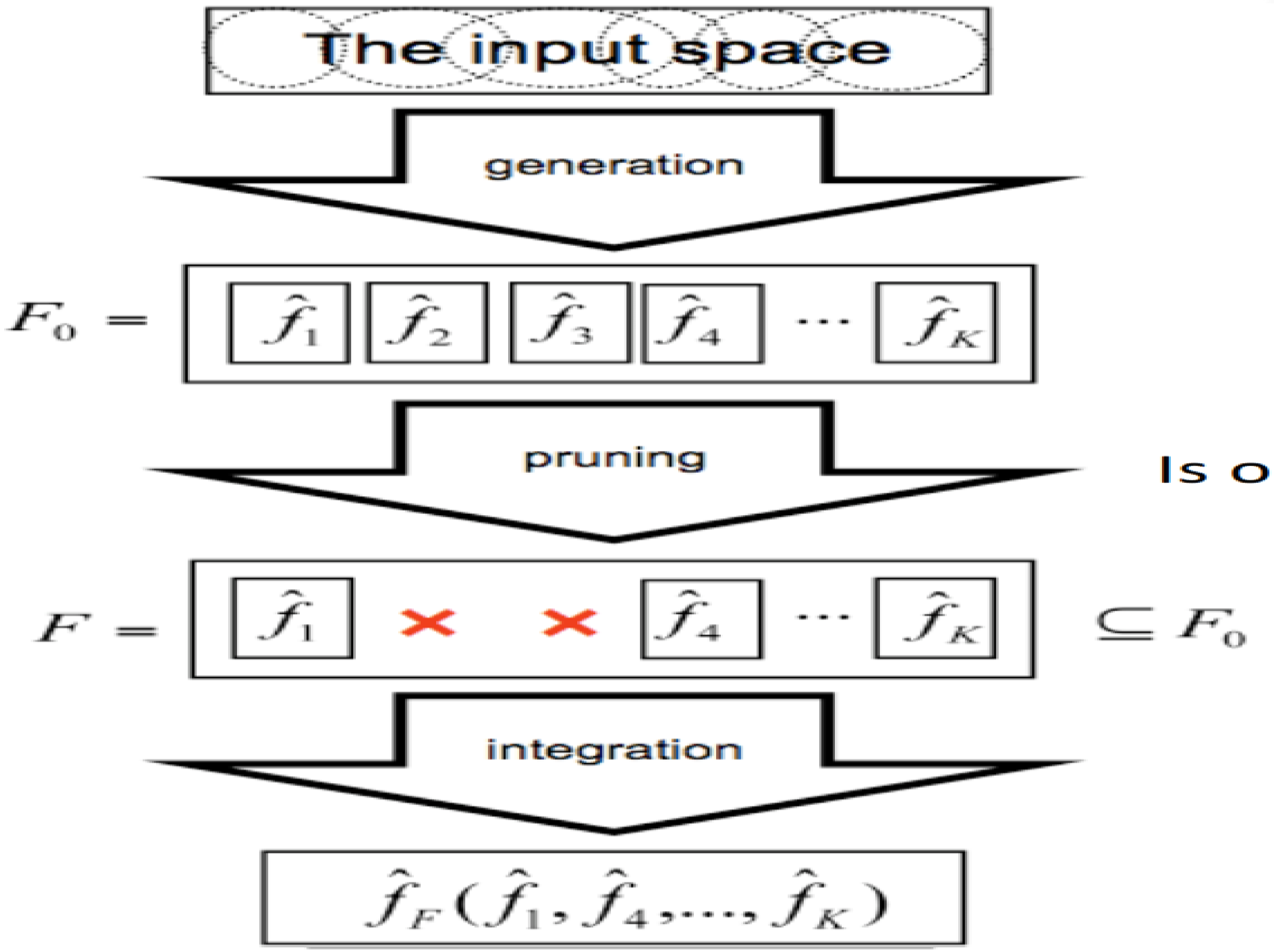
- There are ensemble methods for:
  - Classification
  - Regression
  - Clustering(also known as consensual clustering)
- We will only discuss ensemble methods for supervised learning (classification and Regression)
- Ensembles can also classified as:
  - Homogeneous: It uses only one induction algorithm
  - Heterogeneous: It uses different induction algorithms

Some comments

- Combining models adds complexity
  - It is, in general, more difficult to characterize and explain predictions
  - The accuracy may increase

- Violation of Ockham's Razor
  - Simplicity leads to greater accuracy
  - Identifying the best model requires identifying the proper model complexity

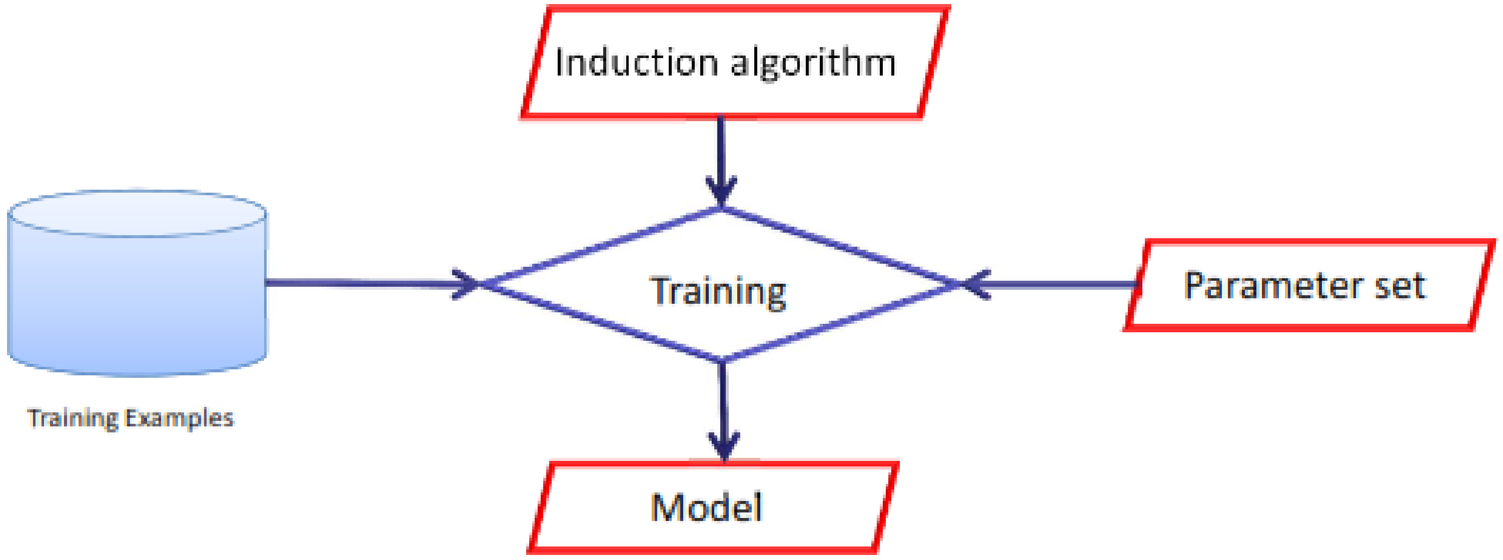
The ensemble learning process


The input space
generation
$F_0 = \hat{f}_1 \; \hat{f}_2 \; \hat{f}_3 \; \hat{f}_4 \cdots \hat{f}_K$
pruning
$F = \hat{f}_1 \; \hat{f}_4 \cdots \hat{f}_K \subseteq F_0$
integration
$\hat{f}_F(\hat{f}_1, \hat{f}_4, \ldots, \hat{f}_K)$

Is optional

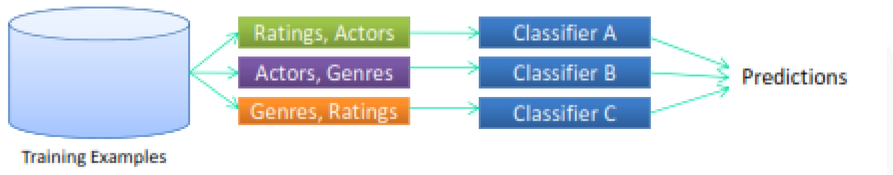
Methods to generate homogeneous ensembles

- Data Manipulation: It changes the training set in order to obtain different models
- Modeling Process manipulation: it changes the induction algorithm, the parameter set or the model (the last one is uncommon) in order to obtain different models

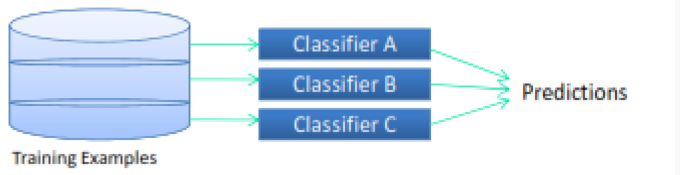


Data Manipulation

- Manipulating the input features

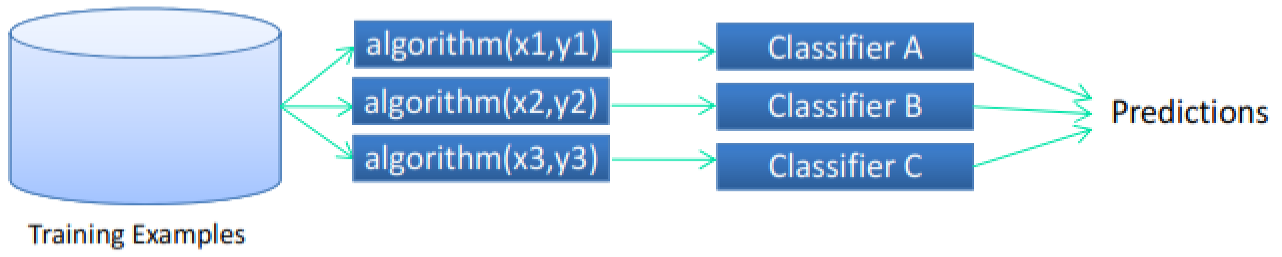


- Sub-sampling from the training set

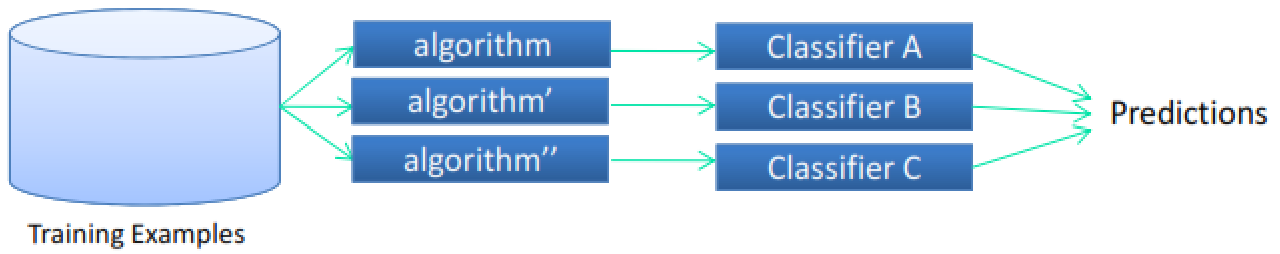


Modeling process manipulation

- Manipulating the parameter sets



- Manipulating induction algorithm



How to combine models (the integration phase)

Algebraic Methods

- Average
- Weighted average
- Sum
- Weighted sum
- Product
- Maximum
- Minimum
- Median

Voting Methods

- Majority voting
- Weighted majority voting
- Borda count

Characteristics of the base models

- For classification:
  - The base classifiers should be as accurate as possible and having diverse errors as much the true class is the majority class.
  - It is not possible to obtain the optimum ensemble of classifier based on the knowledge of the base learners
- For Regression:
  - It is possible to express the error of the ensemble in fun of the error of the base learners
  - Assuming the average as the combination method,

$$\boxed{E[(\hat{f}_{\mathcal{F}} - f)^2] = \overline{bias}^2 + \frac{1}{K} \times \overline{var} + (1 - \frac{1}{K}) \times \overline{covar}}$$

The goal is to minimize $E[(\hat{f}_F - f)^2]$, so:

- The average error of the base learners ($\overline{bias}$) should be as small as possible, i.e., the base learners should be as accurate (in average) as possible;
- The average variance of the base learners ($\overline{var}$) should be as small as possible;
- The average covariance of the base learners ($\overline{covar}$) should be as small as possible, i.e., the base learners should have negative correlation.

Popular Ensemble methods

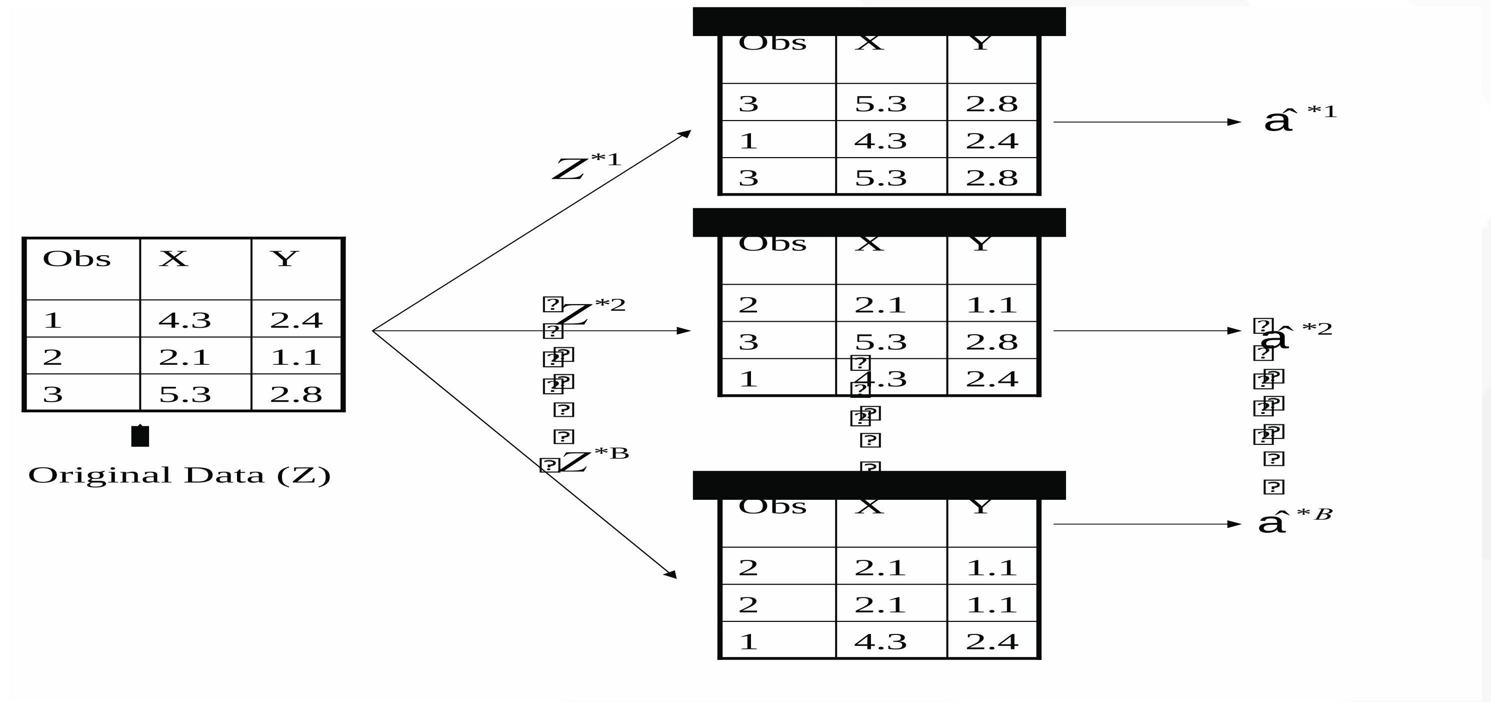
- **Bagging:** Averaging the prediction over a collection of unstable predictors generated from bootstrap samples(both classification and regression)
- **Boosting:** Weighted vote with a collection of classifiers that were trained sequentially from training sets given priority to instances wrongly classified (classification)
- **Random Forest:** Averaging the prediction over a collection of trees splitted using a randomly selected subset of features (both classification and Regression)
- **Ensemble Learning via negative correlation learning:** Generating sequentially new predictors negatively correlated with the existing ones (Regression)
- **Heterogeneous ensembles:** Combining a set of heterogeneous predictors (Both classification and Regression)

What is bagging?

- Bootstrap AGGregatING
- Bagging is an extremely powerful idea based on two things:
  - Averaging: reduces variance!
  - Bootstrapping: plenty of training datasets!
- Why does averaging reduces variance?
  - Averaging a set of observations reduces variance. Recall that given a set of n independent observations $Z_1, \ldots, Z_n$, each with variance s^2, the variance of the mean $\overline{Z}$ of the observations is given by s^2/n

Bootstrapping is simple!

- Resampling of the observed dataset (and of equal size to the observed dataset), each of which is obtained by random sampling with replacement from the original dataset.



Obs	X	Y
1	4.3	2.4
2	2.1	1.1
3	5.3	2.8

Obs	X	Y
3	5.3	2.8
1	4.3	2.4
3	5.3	2.8

Obs	X	Y
2	2.1	1.1
3	5.3	2.8
1	4.3	2.4

Obs	X	Y
2	2.1	1.1
2	2.1	1.1
1	4.3	2.4

Bagging: Bootstrap AGGregatING

Analogy: Diagnosis based on multiple doctors' majority vote

Training:

Given a set D of *d* tuples, at each iteration *i*, a training set D_i of *d* tuples is sampled with replacement from D (i.e., bootstrap)

A classifier model M_i is learned for each training set D_i

Classification: classify an unknown sample **X**

Each classifier M_i returns its class prediction

The bagged classifier M* counts the votes and assigns the class with the most votes to **X**

Prediction: can be applied to the prediction of continuous values by taking the average value of each prediction for a given test tuple

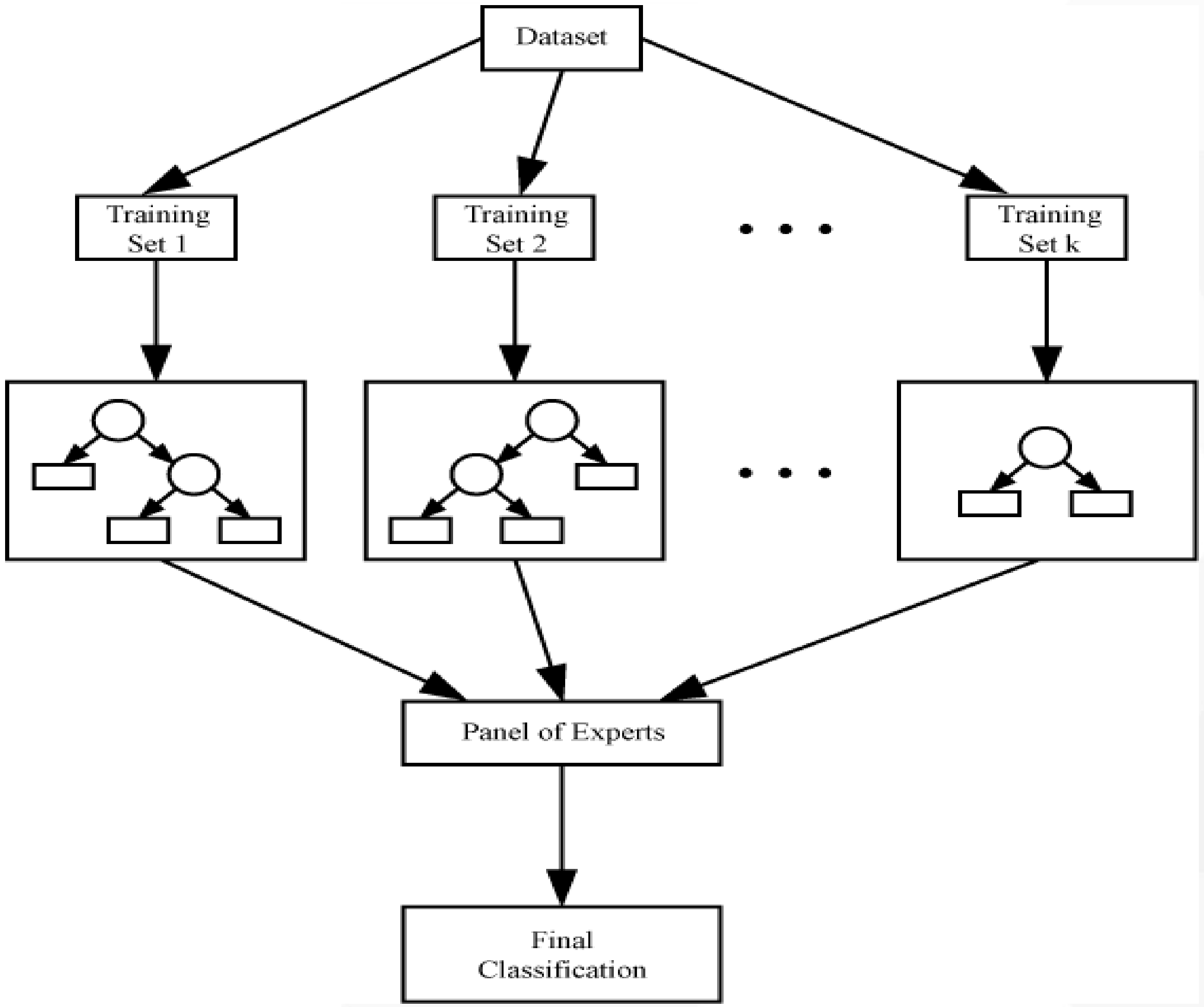
How does bagging work?

- Generate B different bootstrapped training datasets

- Train the statistical learning method on each of the B training datasets, and obtain the prediction

- For prediction:
  - Regression: average all predictions from all B trees
  - Classification: majority vote among all B trees

How does bagging work?



http://en.wikibooks.org/wiki/File:DTE_Bagging.png

Bagging for Regression Trees

- Construct B regression trees using B bootstrapped training datasets
- Average the resulting predictions

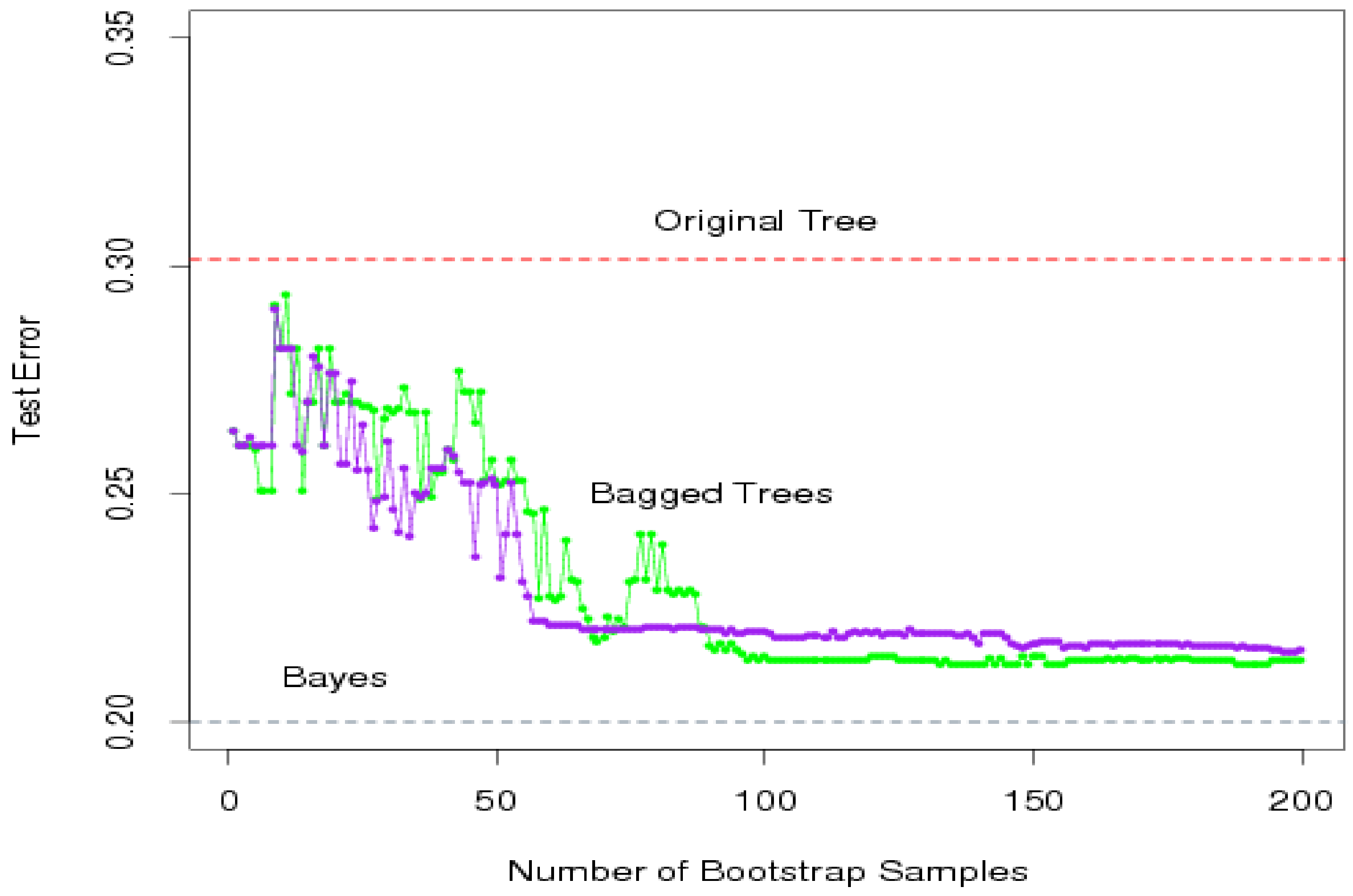
- Note: These trees are not pruned, so each individual tree has high variance but low bias. Averaging these trees reduces variance, and thus we end up lowering both variance and bias ☺

Bagging for Classification Trees

- Construct B regression trees using B bootstrapped training datasets
- For prediction, there are two approaches:
    1. Record the class that each bootstrapped data set predicts and provide an overall prediction to the most commonly occurring one (majority vote).
    2. If our classifier produces probability estimates we can just average the probabilities and then predict to the class with the highest probability.
- Both methods work well.

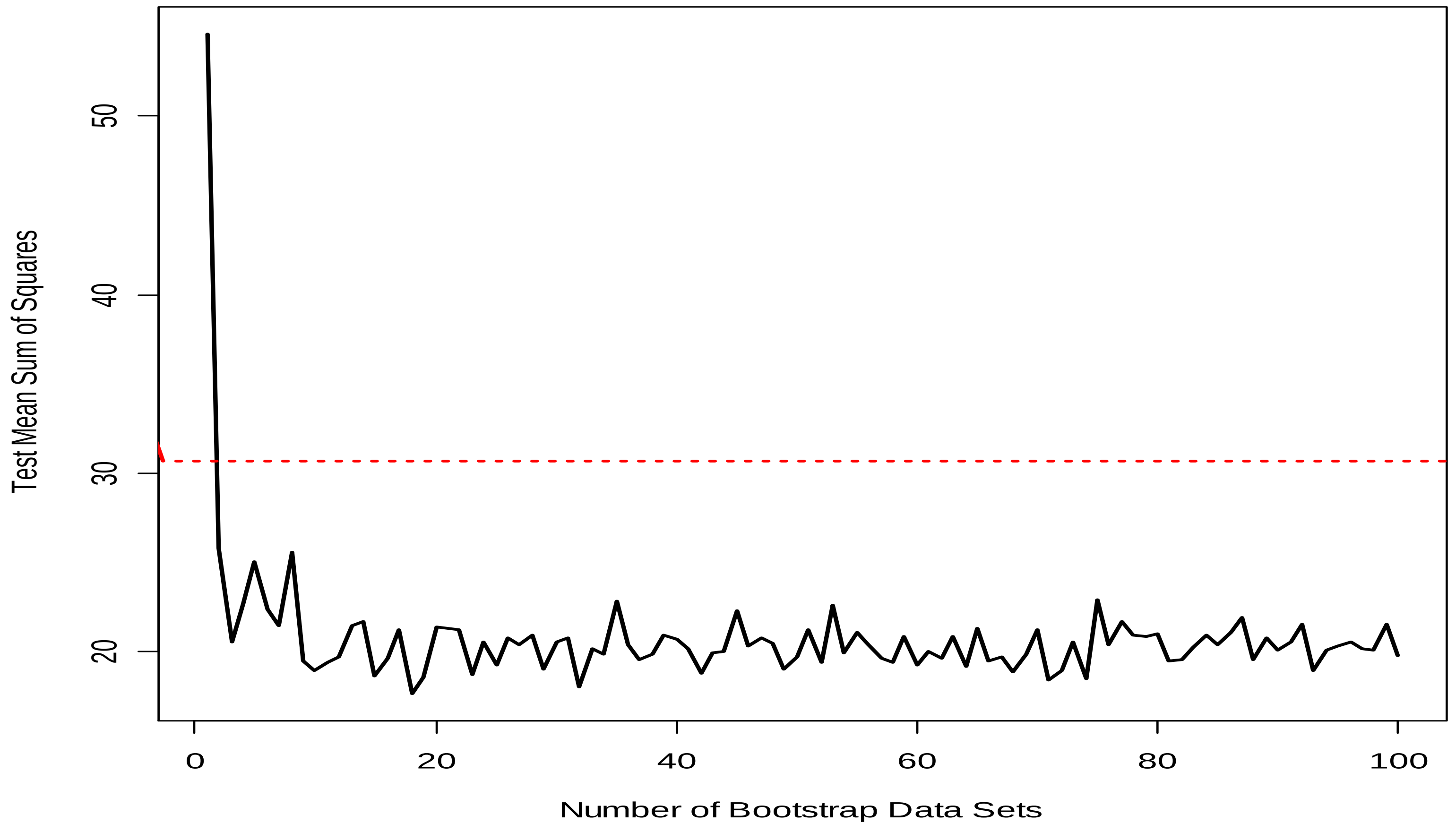
A Comparison of Error Rates

- Here the green line represents a simple majority vote approach
- The purple line corresponds to averaging the probability estimates.
- Both do far better than a single tree (dashed red) and get close to the Bayes error rate (dashed grey).



Example 1: Housing Data

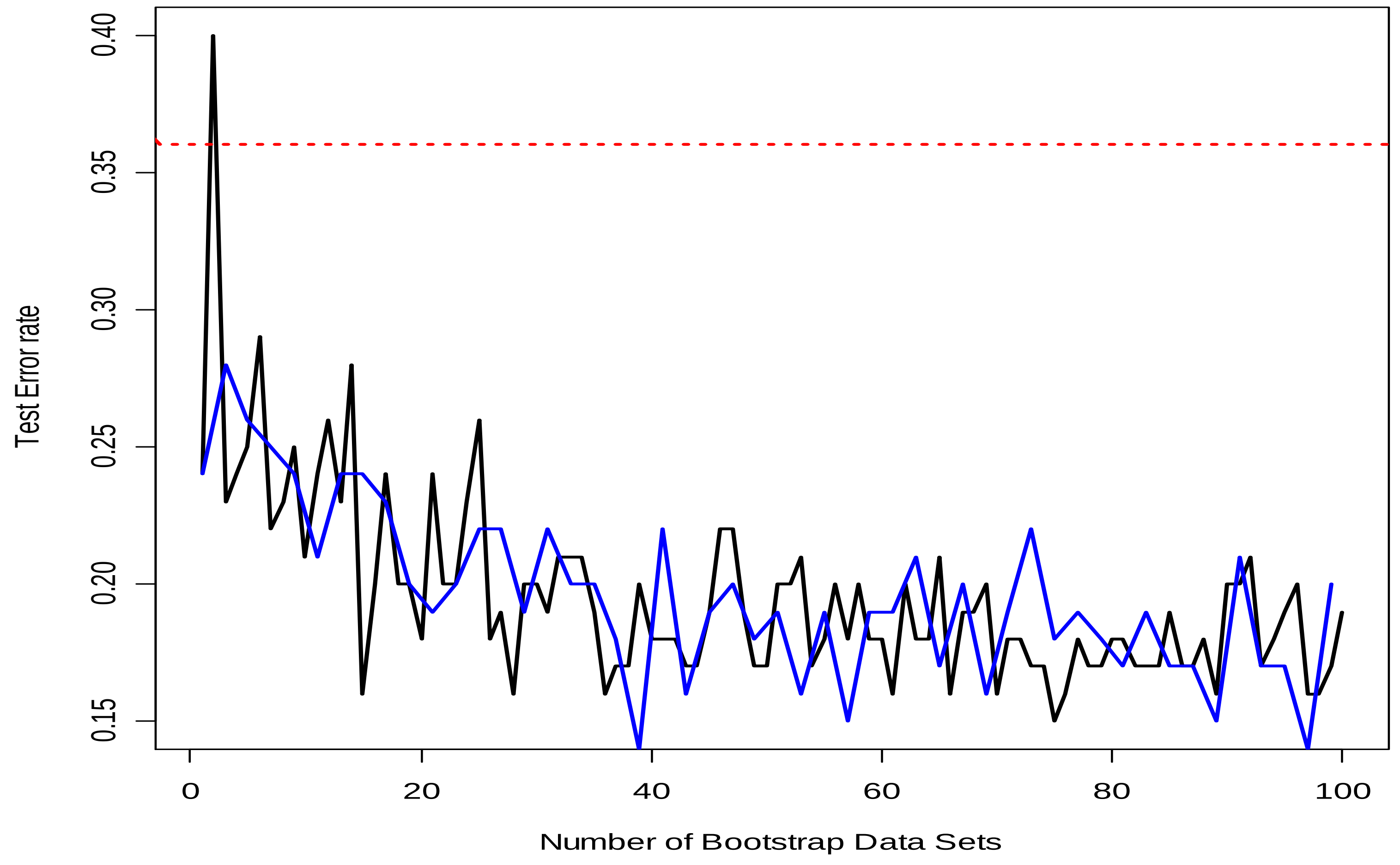
- The red line represents the test mean sum of squares using a single tree.
- The black line corresponds to the bagging error rate



Example 2: Car Seat Data

- The red line represents the test error rate using a single tree.

- The black line corresponds to the bagging error rate using majority vote while the blue line averages the probabilities.



Out-of-Bag Error Estimation

- Since bootstrapping involves random selection of subsets of observations to build a training data set, then the remaining non-selected part could be the testing data.
- On average, each bagged tree makes use of around 2/3 of the observations, so we end up having 1/3 of the observations used for testing

Variable Importance Measure

- Bagging typically improves the accuracy over prediction using a single tree, but it is now hard to interpret the model!
- We have hundreds of trees, and it is no longer clear which variables are most important to the procedure
- Thus bagging improves prediction accuracy at the expense of interpretability
- But, we can still get an overall summary of the importance of each predictor using Relative Influence Plots

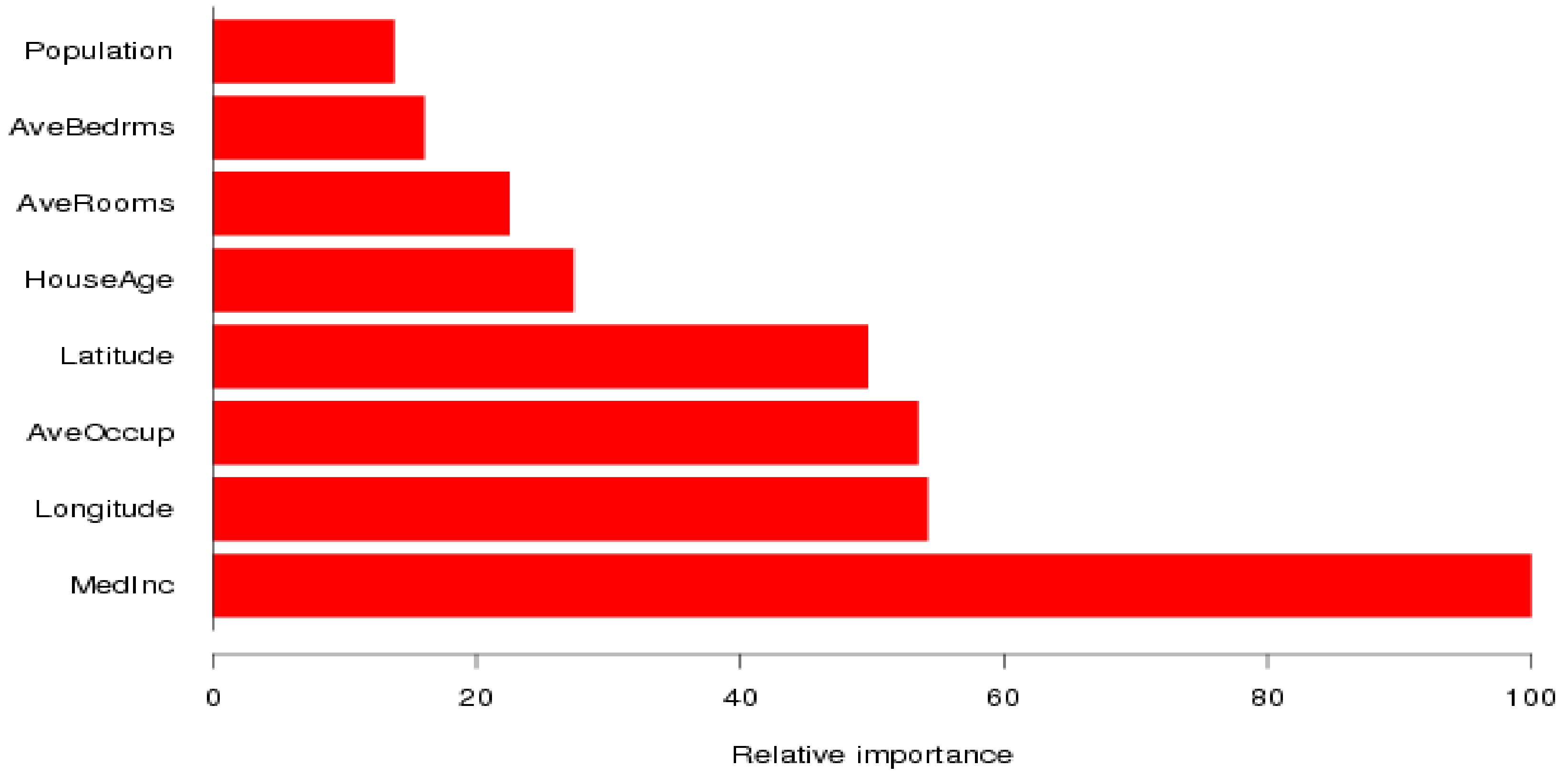
Relative Influence Plots

- How do we decide which variables are most useful in predicting the response?
    - We can compute something called relative influence plots.
    - These plots give a score for each variable.
    - These scores represents the decrease in MSE when splitting on a particular variable
    - A number close to zero indicates the variable is not important and could be dropped.
    - The larger the score the more influence the variable has.

Example: Housing Data

- Median Income is by far the most important variable.

- Longitude, Latitude and Average occupancy are the next most important.



Random Forest

Random Forest

- Random Forest: A variation of the **bagging algorithm**
- Created from individual decision trees.
- Diversity is guaranteed by selecting randomly at each split, a subset of the original features during the process of tree generation.
- During classification, each tree votes and the most popular class is returned
- During regression, the result is the averaged prediction of all generated trees

Random Forest

- Two Methods to construct Random Forest:
  - Forest-RI (random input selection): Randomly select, at each node, F attributes as candidates for the split at the node. The CART methodology is used to grow the trees to maximum size
  - Forest-RC (random linear combinations): Creates new attributes (or features) that are a linear combination of the existing attributes (reduces the correlation between individual classifiers)
- Comparable in accuracy to Adaboost, but more robust to errors and outliers
- Insensitive to the number of attributes selected for consideration at each split, and faster than bagging or boosting

Random Forests

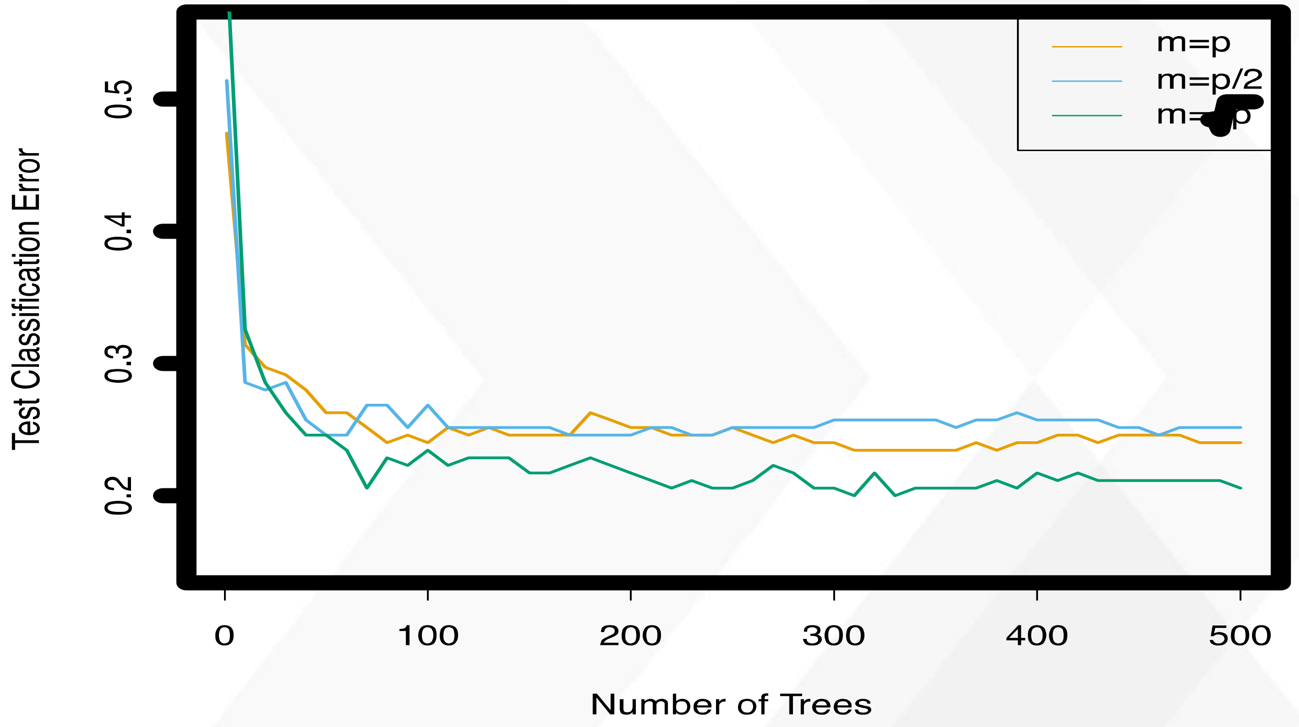
- It is a very efficient statistical learning method
- It builds on the idea of bagging, but it provides an improvement because it de-correlates the trees
- How does it work?
  - Build a number of decision trees on bootstrapped training sample, but when building these trees, each time a split in a tree is considered, a random sample of *m* predictors is chosen as split candidates from the full set of *p* predictors (Usually $m » \sqrt{p}$)

Why are we considering a random sample of m predictors instead of all p predictors for splitting?

- Suppose that we have a very strong predictor in the data set along with a number of other moderately strong predictor, then in the collection of bagged trees, most or all of them will use the very strong predictor for the first split!
- All bagged trees will look similar. Hence all the predictions from the bagged trees will be highly correlated
- Averaging many highly correlated quantities does not lead to a large variance reduction, and thus random forests “de-correlates” the bagged trees leading to more reduction in variance

Random Forest with different values of “m”

- Notice when random forests are built using m = p, then this amounts simply to bagging.



Ensemble learning via negative correlation learning

- Negative correlation learning can be used only in rnsemble regression algorithms that try to minimize/maximize a given objective function (e.g., neural networks, support vector regression)
- The idea is: a model should be trained in order to minimize the error function of the ensemble, i.e., it adds to the error function a penalty term with the averaged error of the models already trained.
- This approach will produce models negatively correlated with the averaged error of the previously generated models.

BOOSTING

Boosting

Analogy: Consult several doctors, based on a combination of weighted diagnoses—weight assigned based on the previous diagnosis accuracy

- Incrementally create models selectively using training examples based on some distribution.

How boosting works?

- Weights are assigned to each training example
- A series of k classifiers is iteratively learned
- After a classifier Mi is learned, the weights are updated to allow the subsequent classifier, Mi+1, to pay more attention to the training examples that were misclassified by Mi
- The final M* combines the votes of each individual classifier, where the weight of each classifier's vote is a function of its accuracy

Boosting: 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

Boosting: 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

Boosting

Differences with Bagging:

- Models are built sequentially on modified versions of the data
- The predictions of the models are combined through a weighted sum/vote

- Boosting algorithm can be extended for numeric prediction
- Comparing with bagging: Boosting tends to achieve greater accuracy, but it also risks overfitting the model to misclassified data

AdaBoost: a popular boosting algorithm

- Given a set of d class-labeled examples, (X1, y1), …, (Xd, yd)
- Initially, all the weights of examples are set the same (1/d)
- Generate k classifiers in k rounds. At round i,
  - Tuples from D are sampled (with replacement) to form a training set Di of the same size
  - Each example's chance of being selected is based on its weight
  - A classification model Mi is derived from Di and its error rate calculated using Di as a test set
  - If a tuple is misclassified, its weight is increased, otherwise it is decreased
- Error rate: err(Xj) is the misclassification error of example Xj. Classifier Mi error rate is the sum of the weights of the misclassified examples.

Adaboost comments

- This distribution update ensures that instances misclassified by the previous classifier are more likely to be included in the training data of the next classifier.
- Hence, consecutive classifiers’ training data are geared towards increasingly hard-to-classify instances.
- Unlike bagging, AdaBoost uses a rather undemocratic voting scheme, called the weighted majority voting. The idea is an intuitive one: those classifiers that have shown good performance during training are rewarded with higher voting weights than the others.

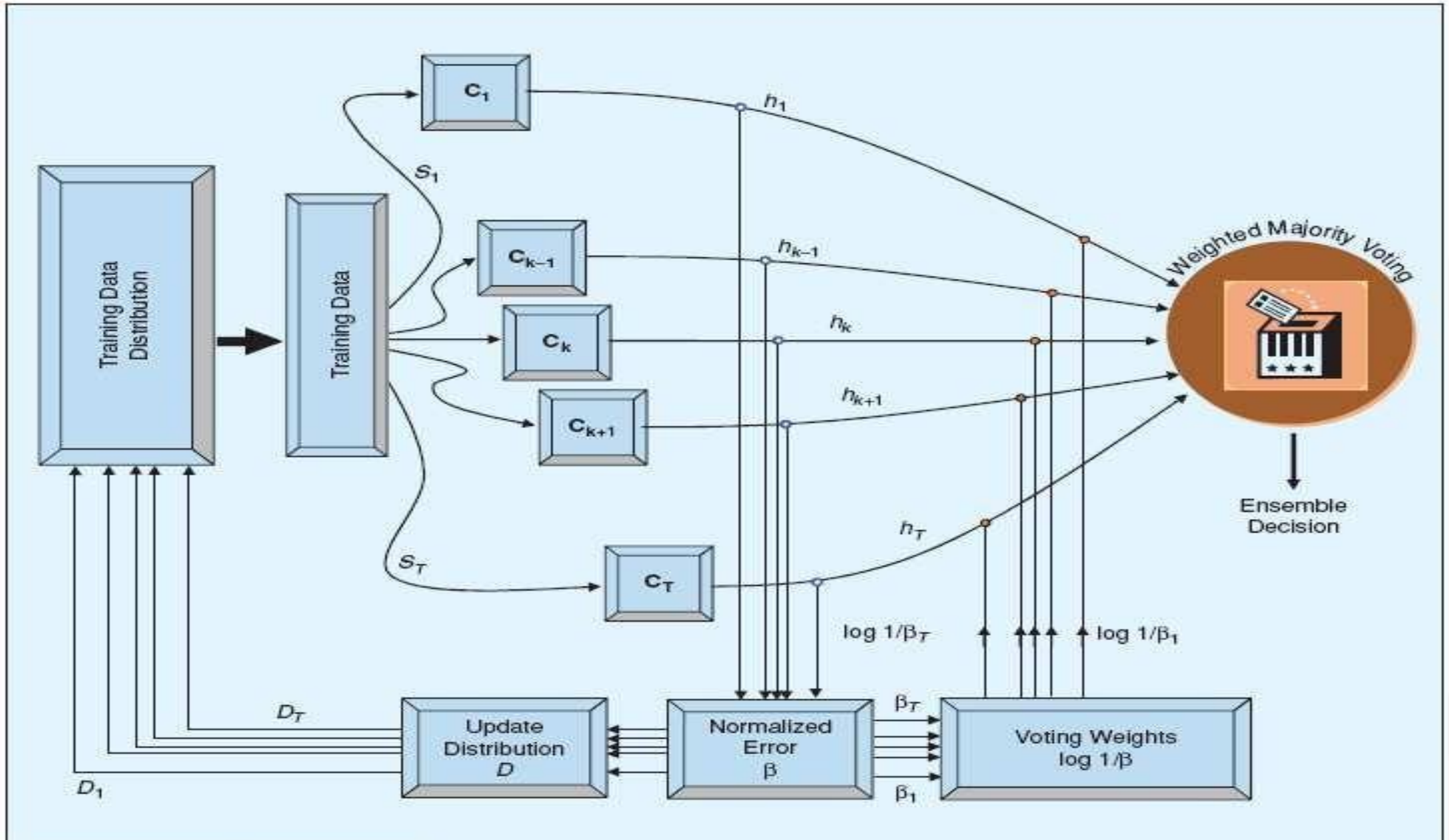


Figure 9. The AdaBoost.M1 algorithm.

The diagram should be interpreted with the understanding that the algorithm is sequential: classifier *CK is created before classifier CK+1, which in turn requires* that *βK and the current distribution DK be available.*

Contact Us

Visit us on: http://www.analytixlabs.in/

For more information, please contact us: http://www.analytixlabs.co.in/contact-us/

Or email: info@analytixlabs.co.in

Call us we would love to speak with you: (+91) 88021-73069

Join us on:

Twitter - http://twitter.com/#!/AnalytixLabs

Facebook - http://www.facebook.com/analytixlabs

LinkedIn - http://www.linkedin.com/in/analytixlabs

Blog - http://www.analytixlabs.co.in/category/blog/