Ensemble Learning

Training individual learners & combining their predictions

CS 6316 – Machine Learning
Fall 2017

OUTLINE

- Motivation for combining methods
- Ensemble Learning
- *Example*: Random Forests

Motivation for Combining Methods

- General setting (used in this course)
  - Given training data set
  - Flexible model parameterization ~ learning method
  - Empirical loss function
  - Optimization method
- Select the best model via single application of a learning method to data

Learning Method + Data → Predictive Model

Motivation (cont'd)

Learning Method + Data → Predictive Mode

- Theoretical and empirical evidence
  - No single "best" method exists
- Always possible to find:
  - Best method for given data set
  - Best data set for given method
- Many philosophical approaches (eastern philosophy / Bayesian averaging/etc) combine several theories (models) explaining the data

Collective Decision Making

Commonly used in our daily lives

- Jury trial
- Multiple expert opinions (in medicine, law, …)

Collective Decision Making

- When it works
  - Experts are indeed intelligent
  - Expert' opinions are different (not correlated)
  - Their decisions are combined intelligently

- When it does not work
  - Majority are not "smart"
  - Experts give similar opinions (highly correlated)
  - Their decisions are not combined intelligently

Strategies for Combining Methods

- Standard inductive learning setting
- **Two combining strategies** (for improved generalization)
  1. Apply different learning methods to the same data
     → Committee of Networks, Stacking, Bayesian averaging
  2. Apply the same method to different (modified) realizations of training data
     → Bagging, Boosting

Strategies for Combining Methods

- Combining methods are used for

Classification:

$$F(\mathbf{x}) = sign\left(\sum_{k=1}^{N} w_k f_k(\mathbf{x}) \right)$$

And **Regression**:

$$F(\mathbf{x}) = \sum_{k=1}^{N} w_k f_k(\mathbf{x})$$

What and Why: Ensemble Learning

- **What is ensemble learning?**
  - Ensemble learning refers to a collection of methods that learn a target function by training a number of individual learners and **combining their predictions**
- **Why ensemble learning?**
  - **Accuracy:** a more reliable mapping can be obtained by combining the output of multiple "experts"
  - **Efficiency:** [divide-and conquer approach] a complex problem can be decomposed into multiple subproblems that are easier to understand and solve

Ensemble Learning

- There is not a single model that works for all problems!
- "To solve really hard problems, we'll have to use several different representations..... It is time to stop arguing over which type of technique is best..... Instead we should work at a higher level of organization and discover how to build managerial systems to exploit the different virtues and evade the different limitations of each of these ways of comparing things." [Minsky, 1991]

Ensemble Learning

- **When to use ensemble learning?**
  - When you can build component classifiers that are more accurate than chance and,
  - more importantly, that are independent from each other

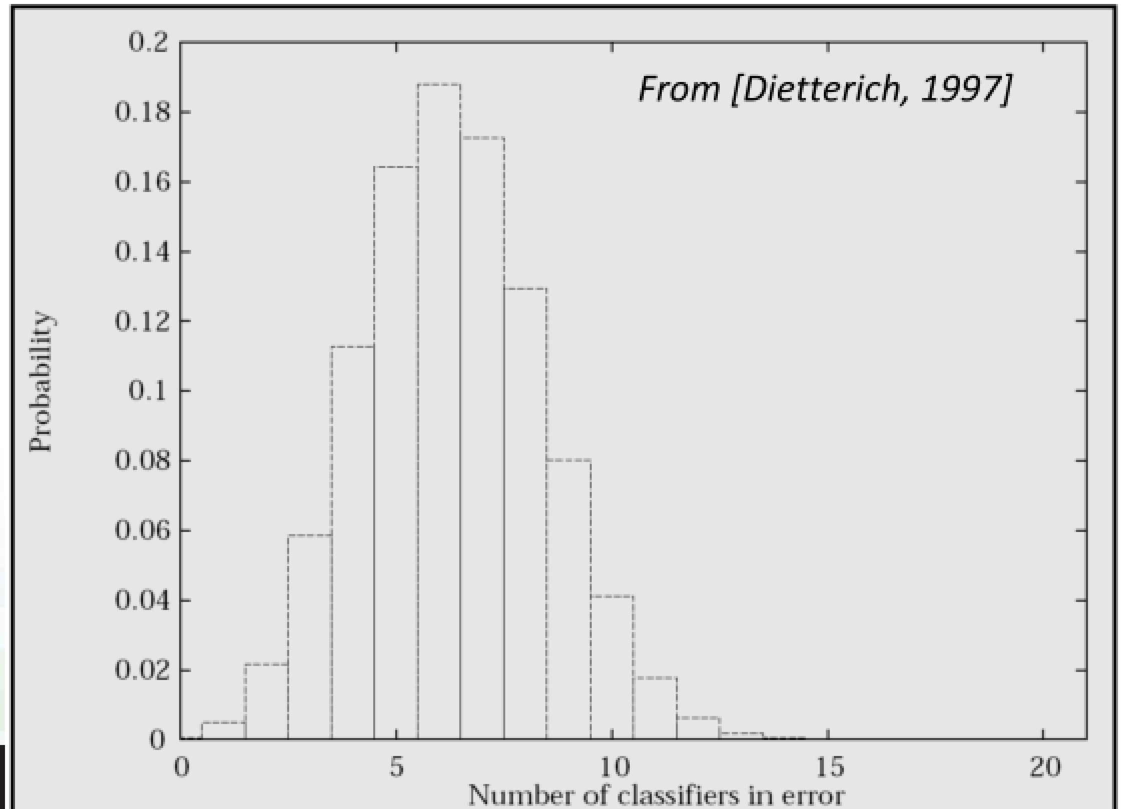
Why do ensembles work?

Because uncorrelated errors of individual classifiers can be eliminated through averaging

- Assume a binary classification problem for which you can train individual classifiers with **an error rate of 0.3**
- Assume that you **build an ensemble** by combining the prediction of 21 such classifiers **with a majority vote**
- What is the probability of error for the ensemble?

Why do ensembles work?

- What is the probability of error for the ensemble?
  - In order for the ensemble to misclassify an example, 11 or more classifiers have to be in error, or a probability of 0.026
  - The histogram here shows the distribution of the **number of classifiers that are in error** in the ensemble machine

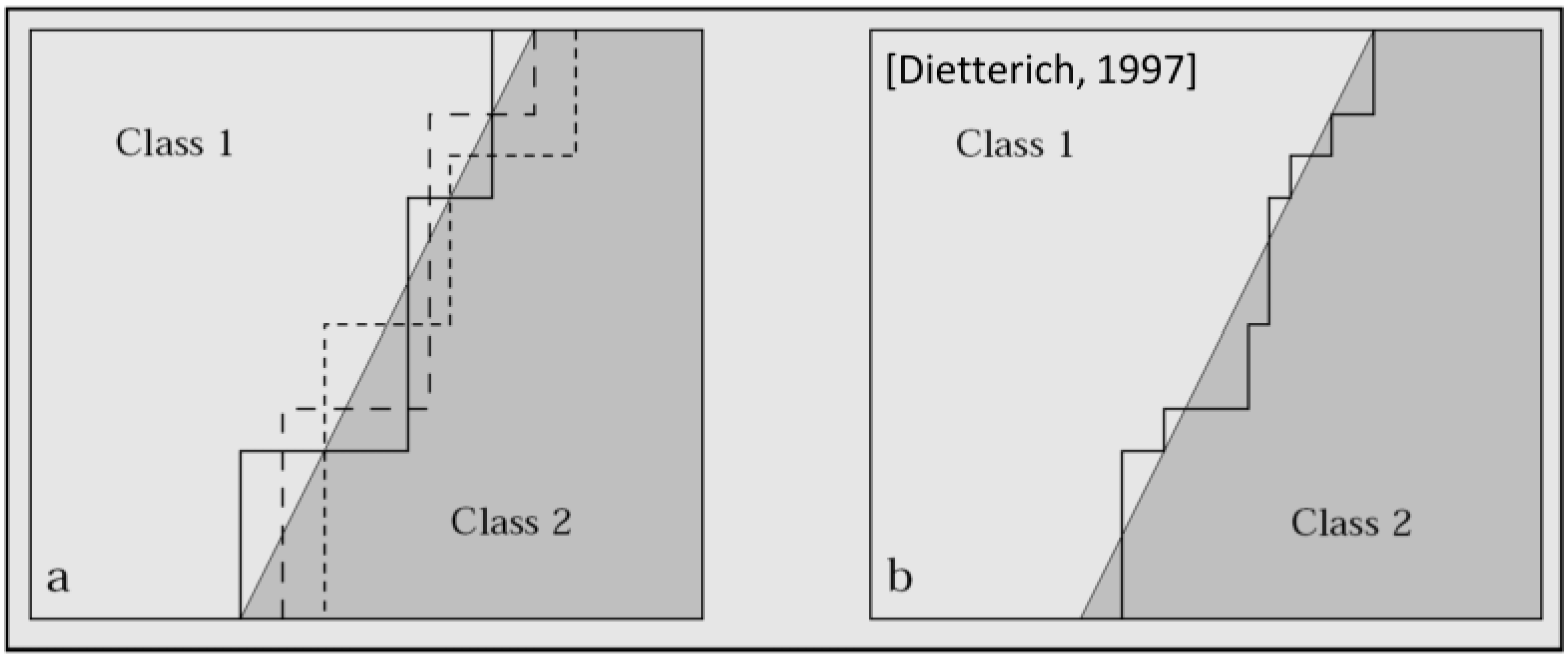


Ensemble Learning

- The target function may not be implementable with **single classifiers**, but may be approximated by ensemble averaging
  - Assume that you want to build a diagonal decision boundary with decision trees
  - The decision boundaries constructed by these machines are hyperplanes parallel to the coordinate axes, or "staircases" in the example (next slide)

Ensemble Learning

- By averaging a large number of such "staircases", the diagonal decision boundary can be approximated with arbitrarily small accuracy



Tree Based Ensemble Methods

- Ensemble methods involve group of predictive models to achieve a better accuracy and model stability
- Ensemble methods are known to impart supreme **boost** to tree based models
- Like every other model, a tree based model also suffers from the plague of bias and variance
  - Bias: *how much on an average are the predicted values different from the actual value*
  - Variance: *how different will the predictions of the model be at the same point if different samples are taken from the same population*

Tree Based Ensemble Methods

- If you have a small tree → you will get a model with **low variance and high bias**
- As you increase the complexity of your model, you will see a reduction in prediction error due to **lower bias** in the model
- As you continue to make your model more complex, you end up over-fitting your model and your model will start suffering from **high variance**

Tree Based Ensemble Methods

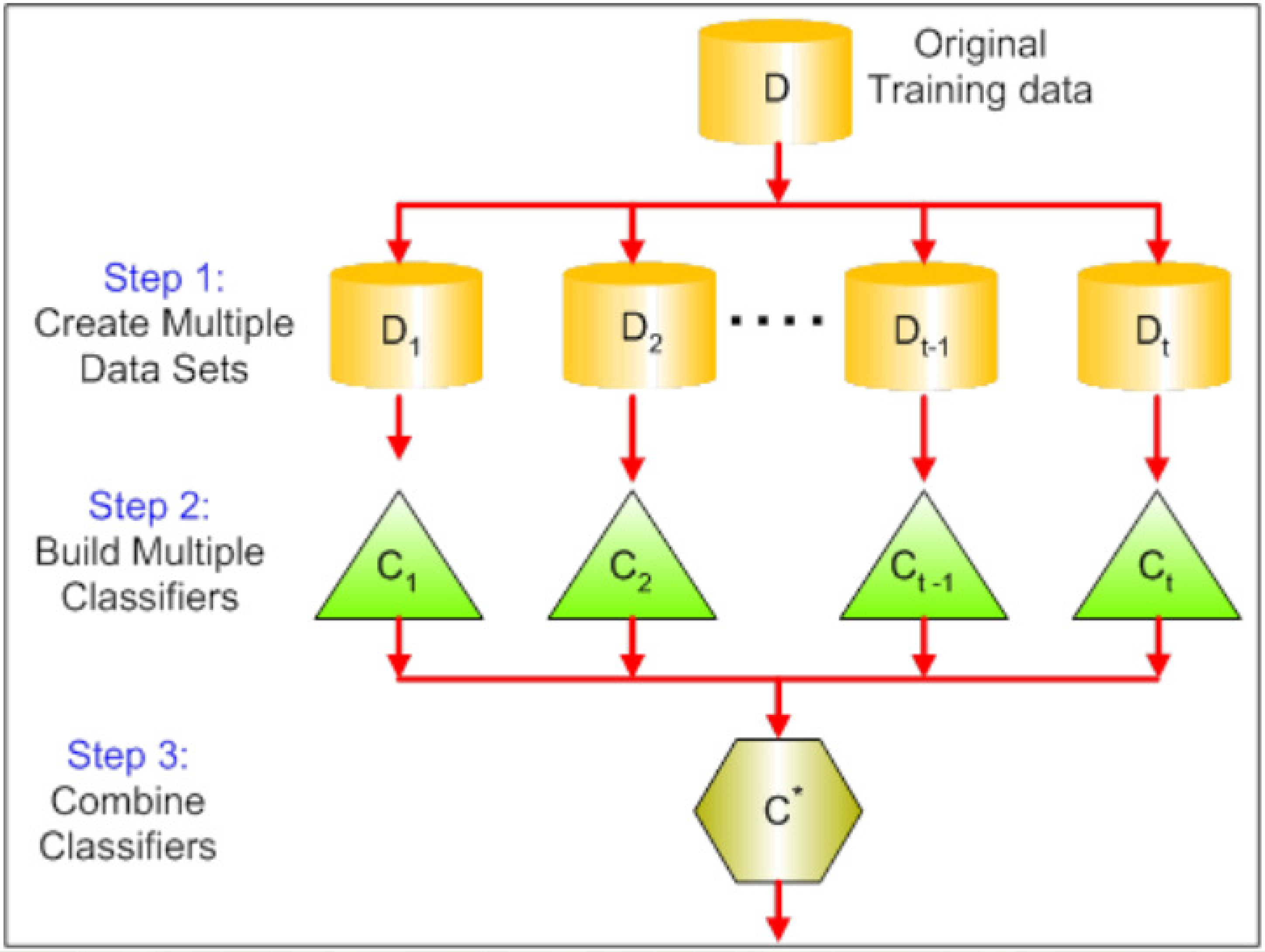
- An optimal model should maintain a balance between these two types of errors. This is known as the **trade-off management** of bias-variance errors
- Ensemble learning is one way to execute this trade off analysis

Bagging

- What is it? How does it work?

- **Bagging** is a technique used to **reduce the variance** of our predictions by combining the result of multiple classifiers modeled on different sub-samples of the same data set

Bagging


Original
Training data
D
Step 1:
Create Multiple
Data Sets
D_1
D_2
D_{t-1}
D_t
Step 2:
Build Multiple
Classifiers
C_1
C_2
C_{t-1}
C_t
Step 3:
Combine
Classifiers
C^*

Bagging

- **Create Multiple Data sets**:
  - Sampling is done *with replacement* on the original data and new datasets are formed
  - The new data sets can have a fraction of the columns as well as rows, which are generally hyper-parameters in a bagging model
  - Taking row and column fractions helps in making robust models, less prone to overfitting

Bagging

- **Build Multiple Classifiers:**
  - Classifiers are built on each data set
  - Generally the same classifier is modeled on each data set and predictions are made
- **Combine Classifiers:**
  - The predictions of all the classifiers are **combined** using a mean, median or mode value depending on the problem at hand
  - The combined values are generally more robust than a single model

Bagging

- The number of models built is not a hyper-parameter*. Higher number of models are always better or may give similar performance than lower numbers. It can be theoretically shown that the variance of the combined predictions are reduced to 1/n (n: number of classifiers) of the original variance, under some assumptions

*hyperparameters *are* **_parameters_** *whose values are set* *<u>prior</u> to the commencement of the learning process*

Bagging // Random Forest

- There are various implementations of bagging models
- **Random forest** is one of them


RANDOM FOREST

Random Forest

- Random Forest is a **versatile** machine learning method capable of performing both regression and classification tasks. It also undertakes dimensional reduction methods, handles outlier values and other essential steps of data exploration, and does a fairly good job
- It is a type of ensemble learning method, where a group of ***weak*** models combine to form a ***powerful*** model

RF: *How does it work?*

- In Random Forest (RF), multiple trees are grown (hence "***forest***") as opposed to a single tree

- To classify a new object based on attributes, each tree gives a classification (the tree "votes" for that class)
- The forest chooses the **classification** having the most votes (over all the trees in the forest)
- In the case of **regression**, it takes the average of outputs by different trees

Random Forest

- Assume number of cases in the training set is N. Then, sample of these N cases is taken at random but with replacement. This sample will be the **training set** for growing the tree
- If there are M input variables, a number ***m***<M is specified such that at each node, ***m*** variables are selected at random out of the M. The best split on these ***m*** is used to split the node. The value of ***m*** is held constant while we grow the forest
- Each tree is grown to the largest extent possible and there is no pruning
- Predict new data by **aggregating** the predictions of the ntree trees (i.e., majority votes for **classification**, average for **regression**)

Voting

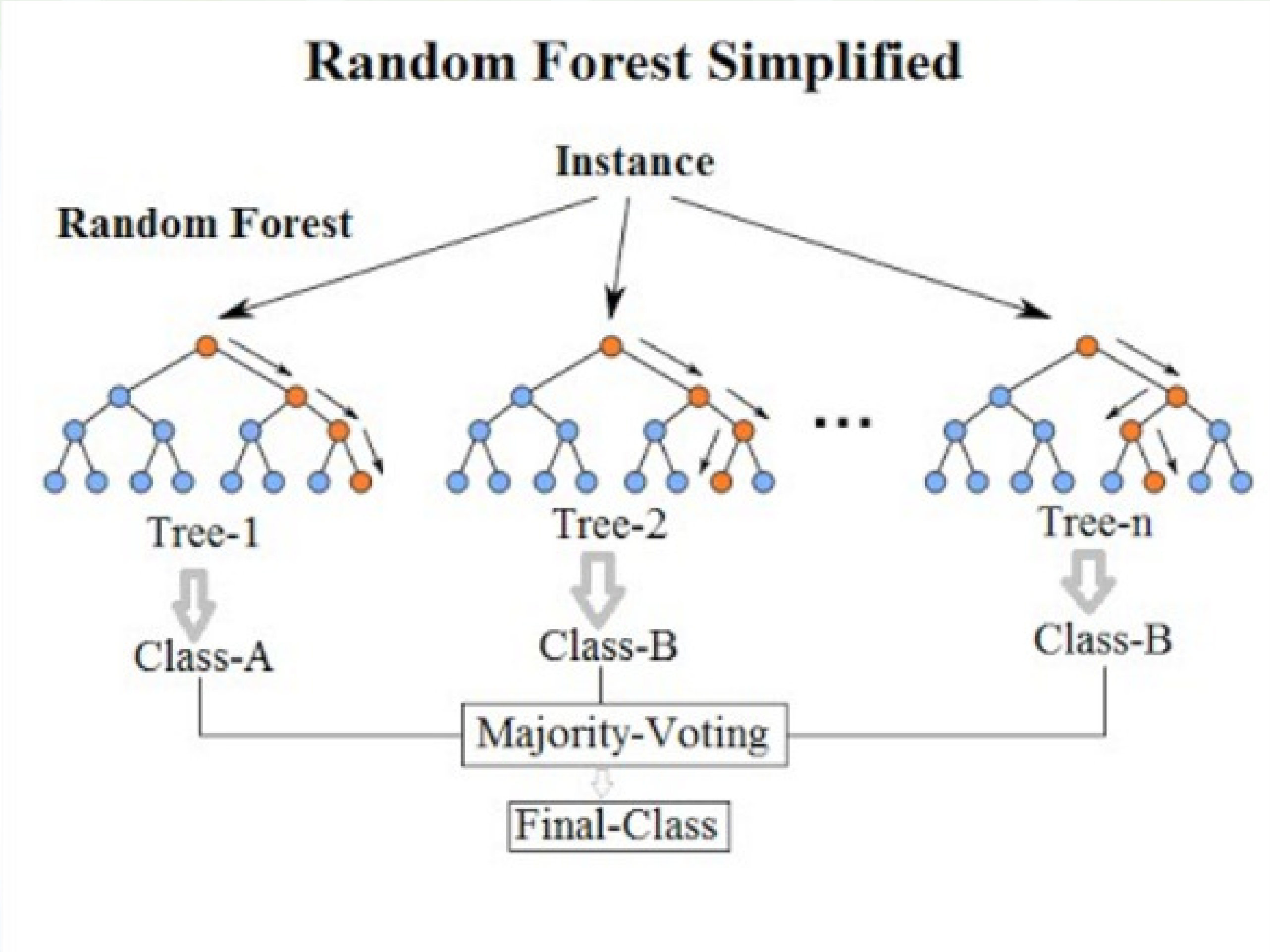
- So what good are 10000 (*probably*) **bad models**? Well it turns out that they really aren't that helpful. But what is helpful are the few really good decision trees that you also generated along with the bad ones

- When you make a prediction, the new observation gets pushed down each decision tree and assigned a predicted value/label. Once each of the trees in the forest have reported its predicted value/label, the predictions are tallied up and the mode vote of all trees is returned as the final prediction

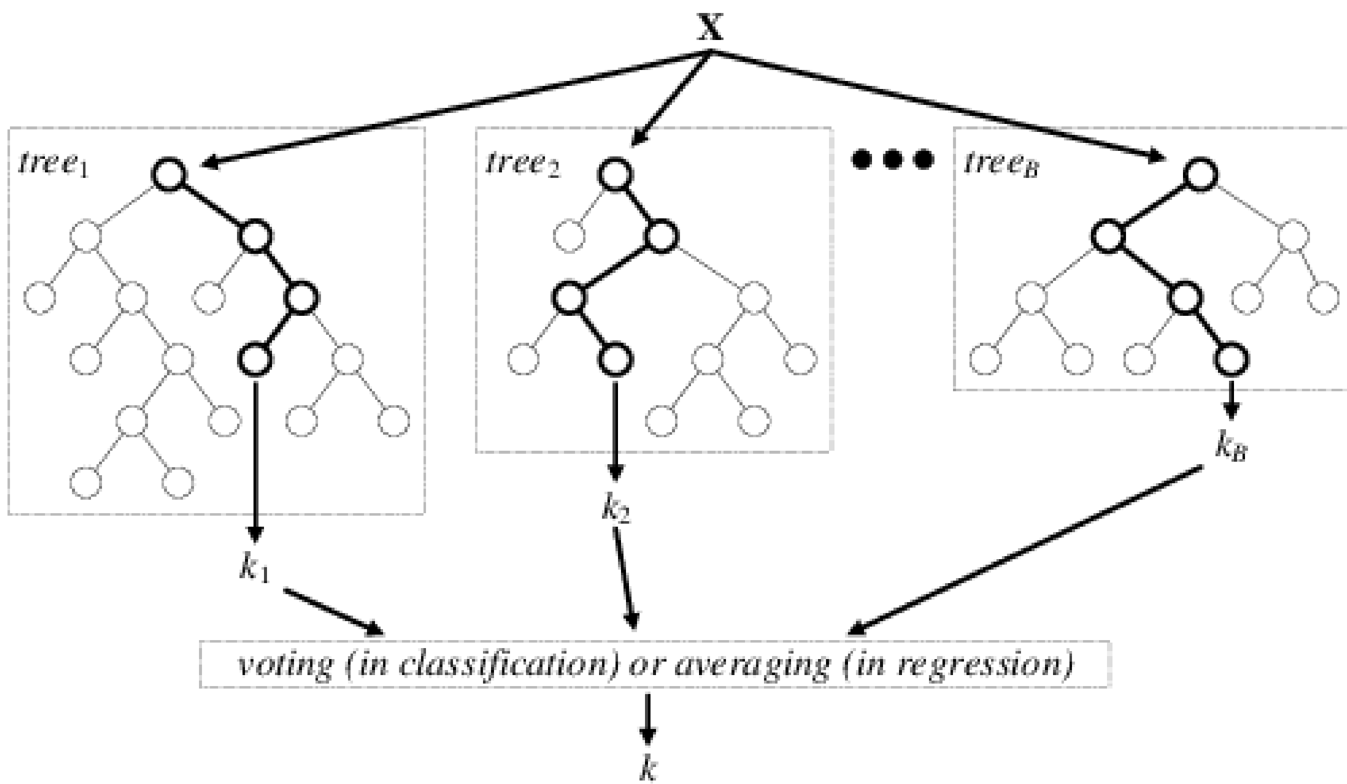
Voting

- Simply, the 99.9% of trees that are *irrelevant* make predictions that are all over the map and cancel each another out. The predictions of the minority of trees that are good top that noise, and yield a good prediction

At each node:
Choose some subset of variables at random find a variable (and a value for that variable) which optimizes the split



Random Forest Simplified
Instance
Random Forest
Tree-1
Tree-2
...
Tree-n
Class-A
Class-B
Class-B
Majority-Voting
Final-Class



X
$tree_1$
$tree_2$
$tree_B$
k_1
k_2
k_B
voting (in classification) or averaging (in regression)
k

Advantages of RF

- Suitable for both classification and regression
- Can handle large data sets with a large number of attributes (dimensions) ~ 1000s!
- Can assist with dimensionality reduction because often implementations help to identify the most significant variables
  - Can output importance of variables, which can be handy when exploring new/unknown data sets

Advantages of RF

- It has an effective method for estimating missing data and maintains accuracy when a large proportion of the data are missing
- It has methods for balancing errors in data sets where classes are imbalanced

Note

- RF involves sample of the input data with replacement
  - This is called ***bootstrap sampling***
- Here one third (for example) of the data is *not* used for training and can be used for testing. These are called the **out of bag** samples. Error estimated on these out of bag samples is known as *out of bag error*. Studies have shown that error estimates by Out of bag is as accurate as using a test set of the same size as the training set.
- Therefore, using the out-of-bag error estimate removes the need for a set aside test set

Out-of-Bag

- When the training set for the current tree is drawn by sampling with replacement, about one-third of the cases are left out of the sample. This *oob* (out-of-bag) data is used to get a running unbiased estimate of the classification error as trees are added to the forest. It is also used to get estimates of variable importance

Out-of-Bag

- So, for RF, there is ***no need*** for cross-validation or a separate test set to get an unbiased estimate of the test set error. It is estimated internally
  - Each tree is constructed using a different bootstrap sample from the original data
  - About one-third of the cases are left out of the bootstrap sample and not used in the construction of the kth tree

Disadvantages of RF

- While it does a good job with classification problems, it doesn't perform as well with regression problems
- In the case of regression, it doesn't predict beyond the range in the training data
  - May over fit data sets that are particularly noisy

References:

~Ensemble Learning ~ R. Gutierrez-Osuna ~ Pattern Analysis ~ TAMU

~Tree Based Modeling from Scratch ~ Analytics Vidhya

~Random Forests ~ L.Breiman and A.Cutler ~ stat.berkeley.edu