

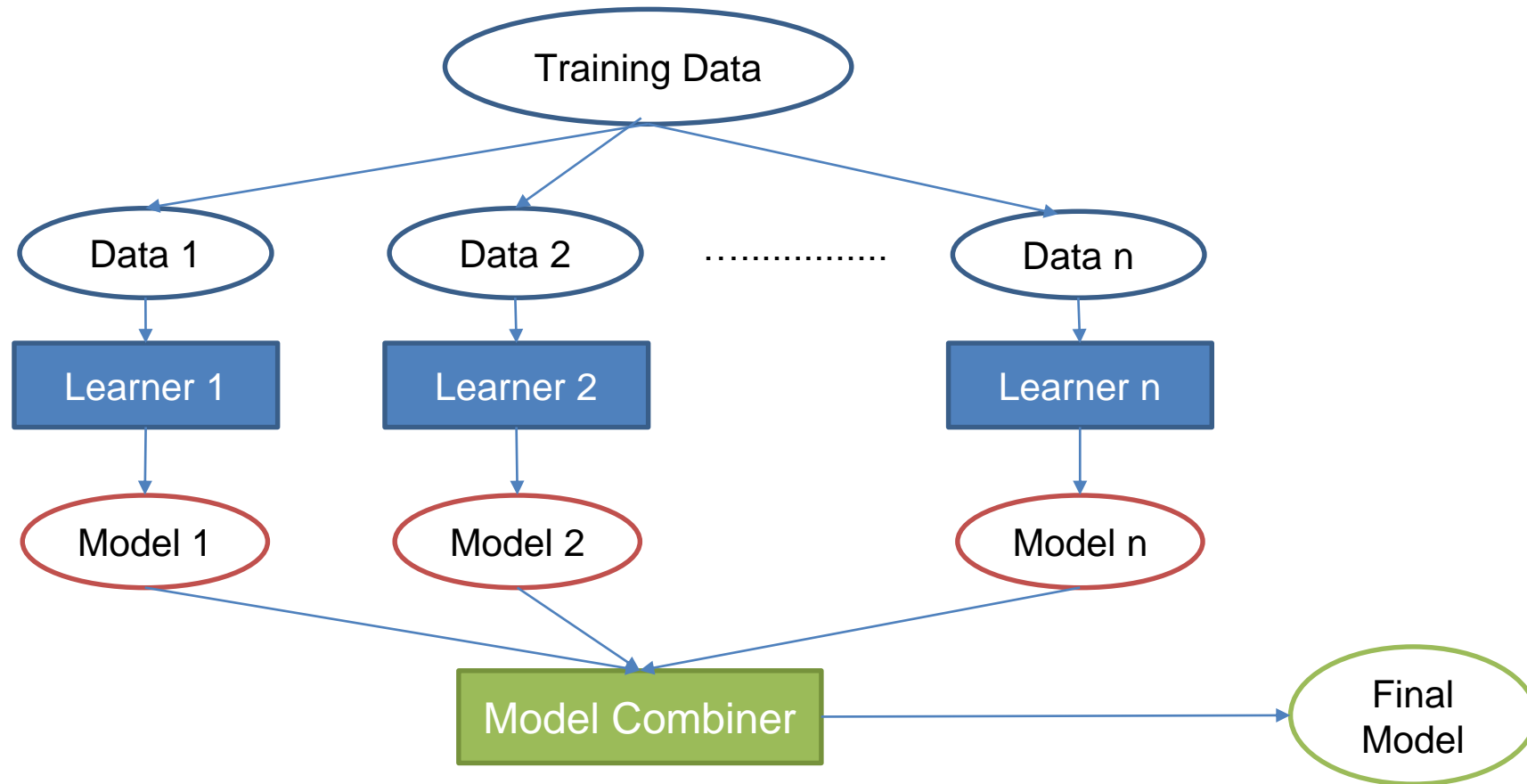
ENSEMBLE LEARNING

Ensemble Learning

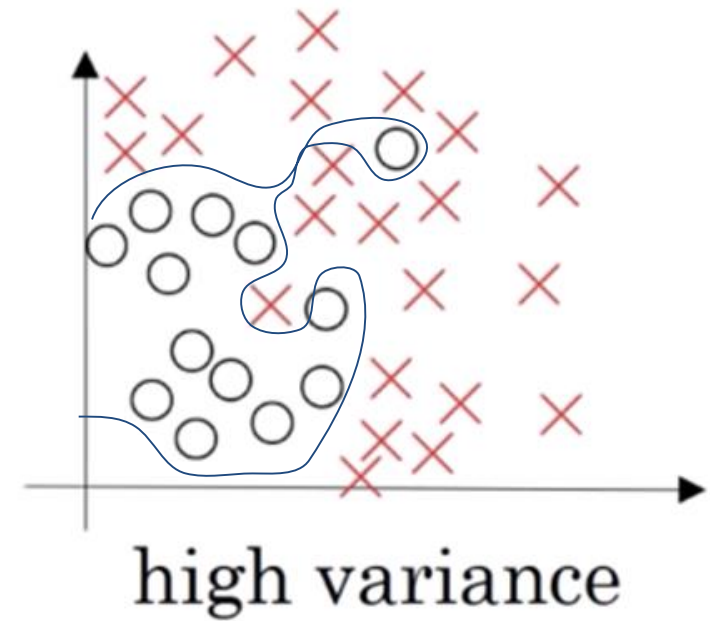
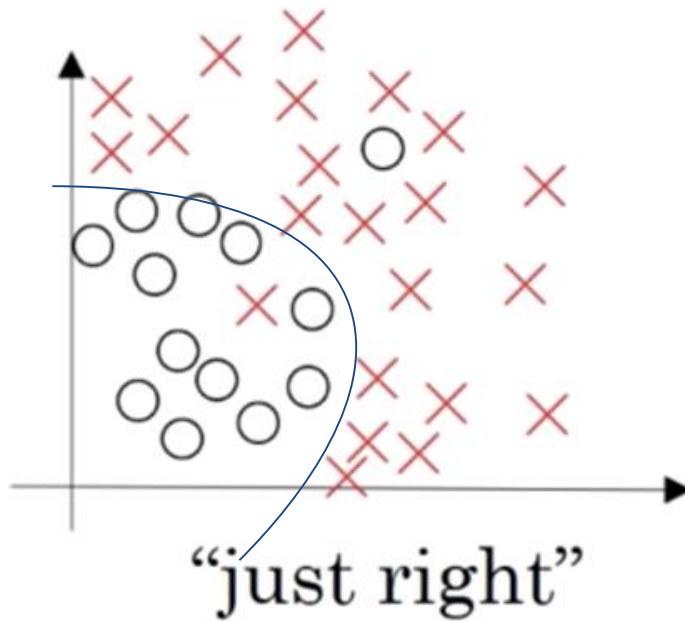
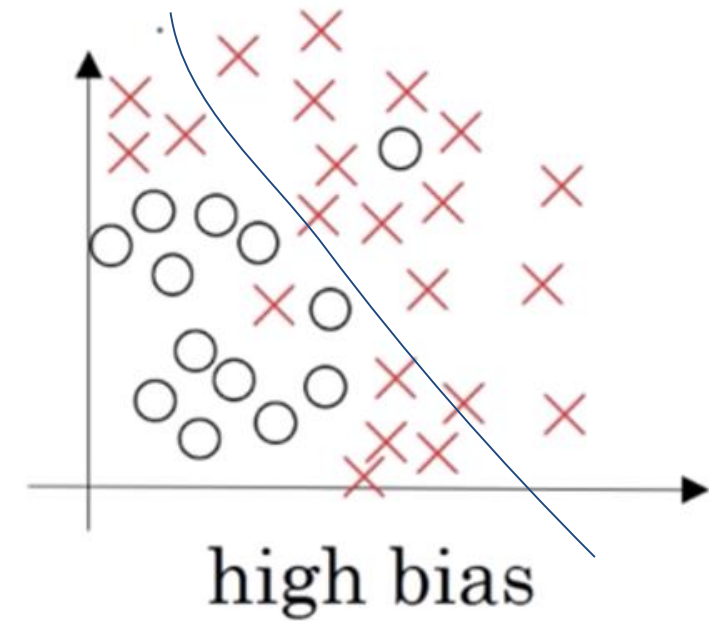
- So far – learning methods that learn a single hypothesis, chosen from a hypothesis space that is used to make predictions.
- Ensemble learning - > select a collection (ensemble) of hypothesis and combine their predictions.
- **Example 1** – generate 100 different decision trees from the same or different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



Bias Vs Variance



Bias Vs Variance

Lets assume we are building a model to perform binary classification.

We evaluated the model on the training and validation datasets.

Train: 1% error

Validation: 20% error

What does that tell you??

Bias Vs Variance

Now lets assume that we evaluated the model on the training and validation datasets with the following:

Train: 15% error

Validation: 17% error

Assuming that this task is simple, and humans would usually get 0 or 1 percent error.

What does that tell you??

Bias Vs Variance

Now lets assume that we evaluated the model on the training and validation datasets with the following:

Train: 30% error

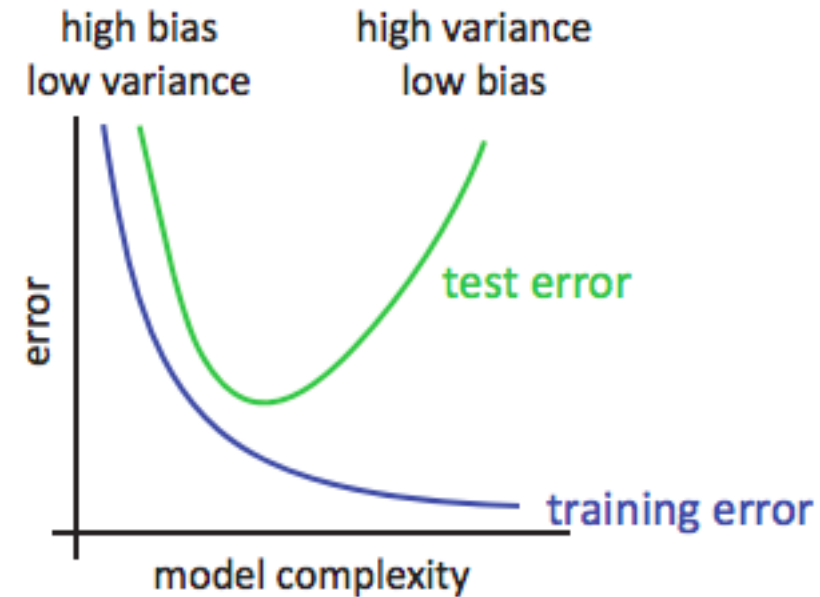
Validation: 40% error

Assuming that this task is simple, and humans would usually get 0 or 1 percent error.

What does that tell you??

Bias vs Variance

- ML balances
 - Estimation error (variance)
 - Precision of match
 - Sensitivity to training data
 - Structural error (bias)
 - Distance from true relationship
- Goals
 - Reduce variance without increasing bias
 - Reduce bias and variance



Debugging High Bias/Variance

High Bias

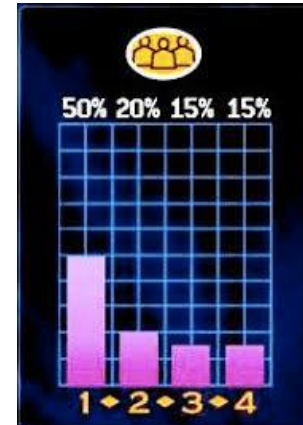
- Means the model is underfitting the data
- Should try different models

High Variance


















































- Means the model is overfitting the data
- Add more data (subset is small hence the model memorizes)
- Apply regularization (a technique to improve the generalizability of a learned model.)
- Evaluate model parameters

Value of Ensembles

- “No Free Lunch” Theorem
 - No single algorithm wins all the time!
- When combining 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
 - Who wants to be a millionaire: Audience vote



Example: Weather Forecast

Reality							
1							
2							
3							
4							
5							
Combine							

Homogenous Ensembles

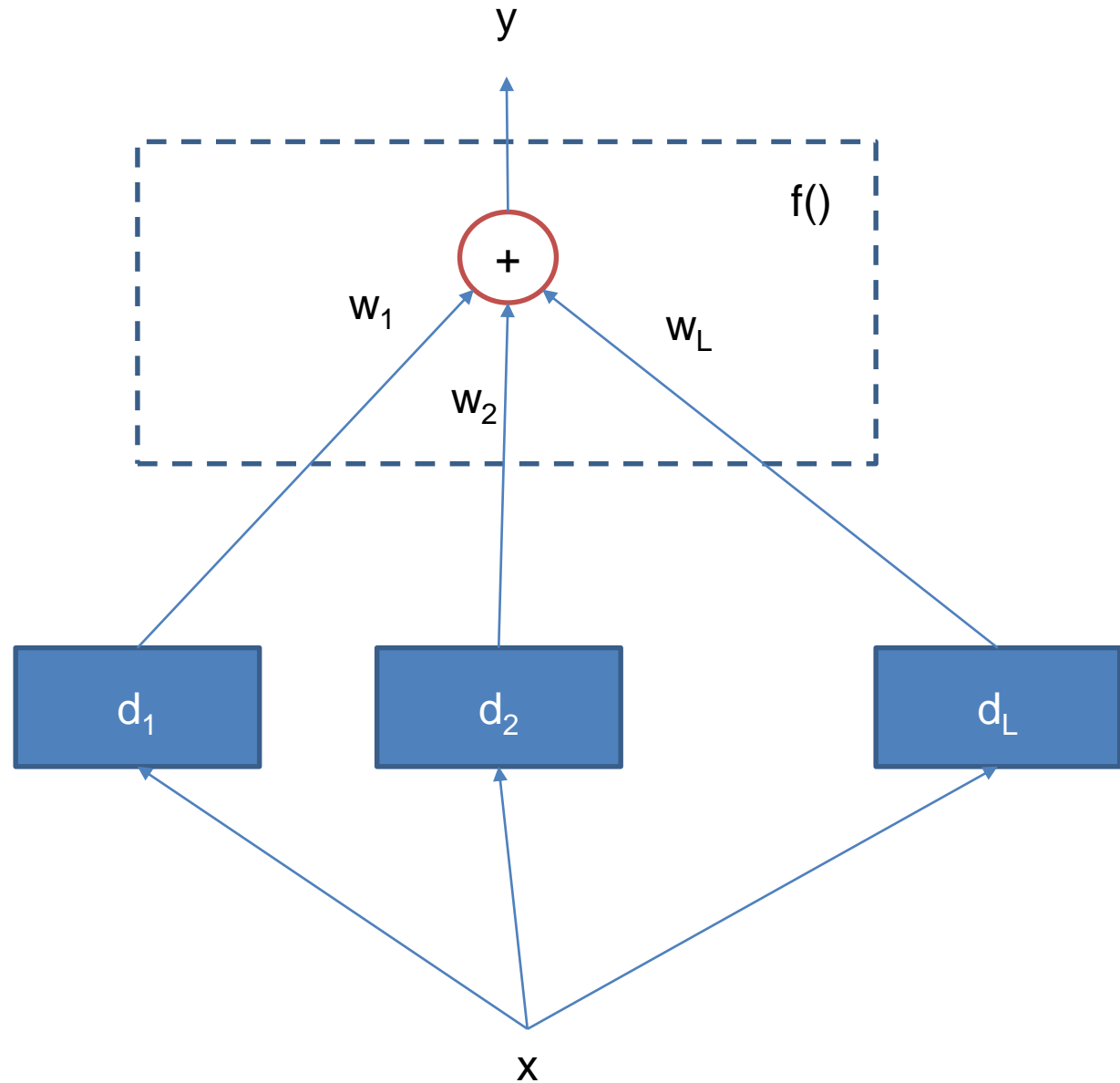
- Use a single, arbitrary learning algorithm but manipulate training data to make it learn multiple models.
 - $\text{Data}_1 \neq \text{Data}_2 \neq \dots \neq \text{Data}_m$
 - $\text{Learner}_1 = \text{Learner}_2 = \dots = \text{Learner}_m$
- Different methods for changing training data
 - **Bagging** : Resample training data
 - **Boosting** : Reweight training data

<https://towardsdatascience.com/ensemble-learning-using-scikit-learn-85c4531ff86a>

Voting

- Linear combination

$$y_i = \sum_j w_j d_{ji} \text{ where } w_j \geq 0, \sum_j w_j = 1$$



Bootstrap Resampling

- Idea: Get several samples from the training data
- Procedure:
 - Let the original sample be $X = (x_1, x_2, x_3, \dots, x_n)$
 - Repeat B time
 - Generate a sample X^* of size n from X by sampling with replacement
 - Compute θ for X^*
 - Now we end up with bootstrap values
 - $\theta^* = (\theta_1, \theta_2, \theta_3, \dots, \theta_B)$
- Use these values for calculating all the quantities of interest (e.g., standard deviation, confidence intervals)

A quick view of bootstrapping

- Introduced by Bradley Efron in 1979 [\[PDF\]](#)
- Became popular in 1980s due to the introduction of computers in statistical practice.
- It has minimum assumptions. It is merely based on the assumption that the sample is a good representation of the unknown population.

Bagging (bootstrap + aggregating)

- **Bagging** is a way to decrease the variance in the prediction by generating additional data for training from dataset using combinations with repetitions to produce multi-sets of the original data.
- i.e. Bagging averages a noisy fitted function refit to many bootstrap samples to reduce its variance.
 - Use bootstrapping to generate B training sets
 - We draw bootstrap samples S_1, \dots, S_b each of size N with replacement from the training data.
 - Train B base learners
 - During test, take the average

Bagging

- Training
 - In each iteration t , $t = 1, \dots, T$
 - Randomly sample with replacement N samples from the training set
 - Train a chosen “base model” (e.g. neural network, decision tree) on the samples
- Test
 - For each test example
 - Start all trained base models
 - Predict by combining results of all T trained models:
 - Regression: averaging
 - Classification: a majority vote

Boosting: Weak learner vs Strong learner

- **Boosting** is an iterative technique which adjusts the weight of an observation based on the last classification
- In boosting, we actively try to generate complementary base-learners by training the next learner on the mistakes of the previous learners.
- The original boosting algorithm (Schapire 1990) combines **three weak learners** to generate a strong learner.
- A weak learner has error probability less than $\frac{1}{2}$, which makes it better than random guessing on a two-class problem.
- A strong learner has arbitrarily small error probability.

Boosting (Alpaydin, 2012)

- Given a large training set, we randomly divide it into three.
- We use X_1 and train classifier d_1 .
- We then use X_2 and feed it to d_2 .
 - We take all instances misclassified by d_1 , and also as many instances on which d_1 is correct from X_2 .
 - Then, this training data is feed-in to d_2 .
- We then take X_3 and feed it to d_1 and d_2 .
 - The instances on which d_1 and d_2 disagree form the training set for d_3 .
- During testing, given an instance x :
 - x is evaluated using d_1 and d_2 .
 - If both agree on the result, it is returned immediately.
 - Otherwise, the response of d_3 is taken as the output.

Boosting: drawback

- Though it is quite successful, the disadvantage of the original boosting method is that it requires a very large training sample.

Adaboost (adaptive boosting)

- Use the same training set over and over and thus need not be large.
- Classifiers must be simple so they do not over-fit.
- Can combine an arbitrary number of base learners, not only three.

<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html>

AdaBoost

- The output of the other learning algorithms ('weak learners') is combined into a weighted sum that represents the final output of the boosted classifier.
- The probability of a correctly classified instance is decreased, and the probability of a misclassified instance increases. This has the effect that the next classifier focuses more on instances misclassified by the previous classifier.

Training:

For all $\{x^t, r^t\}_{t=1}^N \in \mathcal{X}$, initialize $p_1^t = 1/N$

For all base-learners $j = 1, \dots, L$

Randomly draw \mathcal{X}_j from \mathcal{X} with probabilities p_j^t

Train d_j using \mathcal{X}_j

For each (x^t, r^t) , calculate $y_j^t \leftarrow d_j(x^t)$

Calculate error rate: $\epsilon_j \leftarrow \sum_t p_j^t \cdot 1(y_j^t \neq r^t)$

If $\epsilon_j > 1/2$, then $L \leftarrow j - 1$; stop

$\beta_j \leftarrow \epsilon_j / (1 - \epsilon_j)$

For each (x^t, r^t) , decrease probabilities if correct:

If $y_j^t = r^t$ $p_{j+1}^t \leftarrow \beta_j p_j^t$ Else $p_{j+1}^t \leftarrow p_j^t$

Normalize probabilities:

$Z_j \leftarrow \sum_t p_{j+1}^t$; $p_{j+1}^t \leftarrow p_{j+1}^t / Z_j$

Testing:

Given x , calculate $d_j(x)$, $j = 1, \dots, L$

Calculate class outputs, $i = 1, \dots, K$:

$$y_i = \sum_{j=1}^L \left(\log \frac{1}{\beta_j} \right) d_{ji}(x)$$

Adaboost: Testing

- Given an instance, all the classifiers decide and a weighted vote is taken.
- The weights are proportional to the base learner's accuracies on the training set.
- The success of Adaboost is due to its property of increasing the margin. If the margin increases, the training instances are better separated and error are less likely.

Summary

- It is often a good idea to combine several learning methods.
- We want diverse classifiers, so their errors cancel out.
- However, remember, ensemble methods do not get a free lunch ...
- Readings
 - Dietterich (2000) : [\[PDF\]](#)
 - Alpaydin (2010) : Ch 17 [\[PDF\]](#)
 - Daume (2012) : Ch 11 [\[PDF\]](#)



RANDOM FOREST

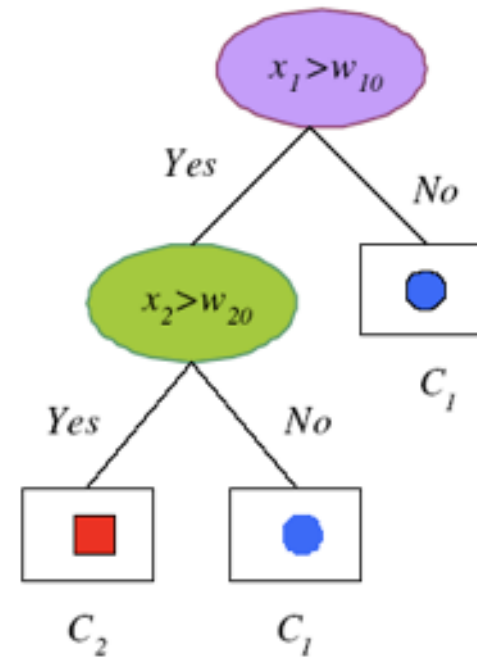
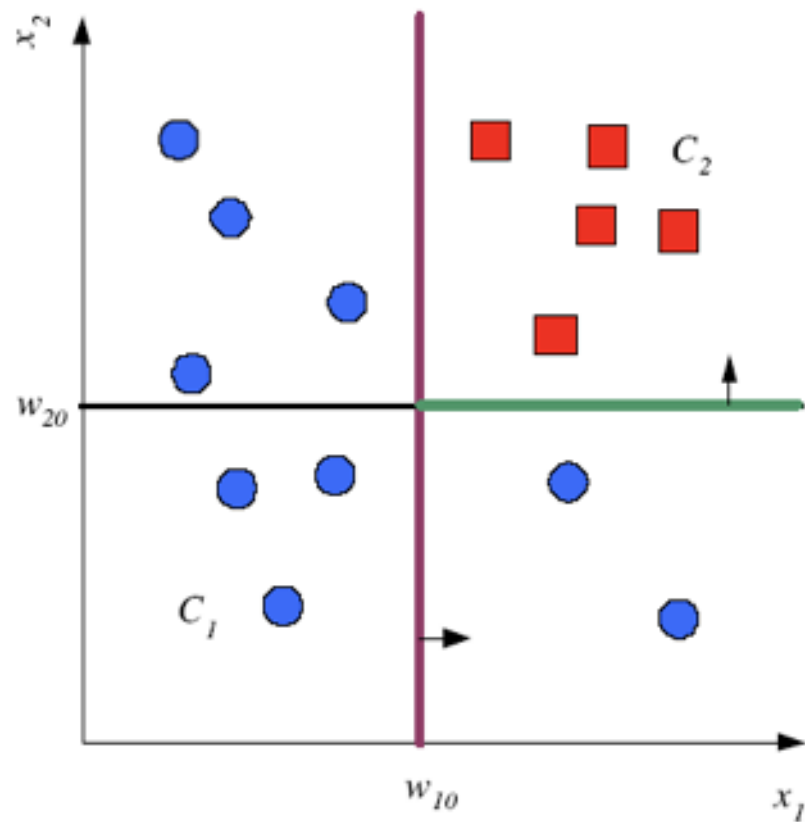
Definition

- **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 trees



Decision trees

- Internal decision nodes
 - Univariate: Uses a single attribute, x_j
 - Numeric x_j : Binary split : $x_j > w_m$
 - Discrete x_j : n -way split for n possible values
 - Multivariate: Uses all attributes, \mathbf{x}
- Leaves
 - Classification: Class labels, or proportions
 - Regression: Numeric; r average, or local fit
- Learning is **greedy**; find the best split recursively (Breiman et al, 1984; Quinlan, 1986, 1993) without backtracking (decision taken are final)

Algorithm

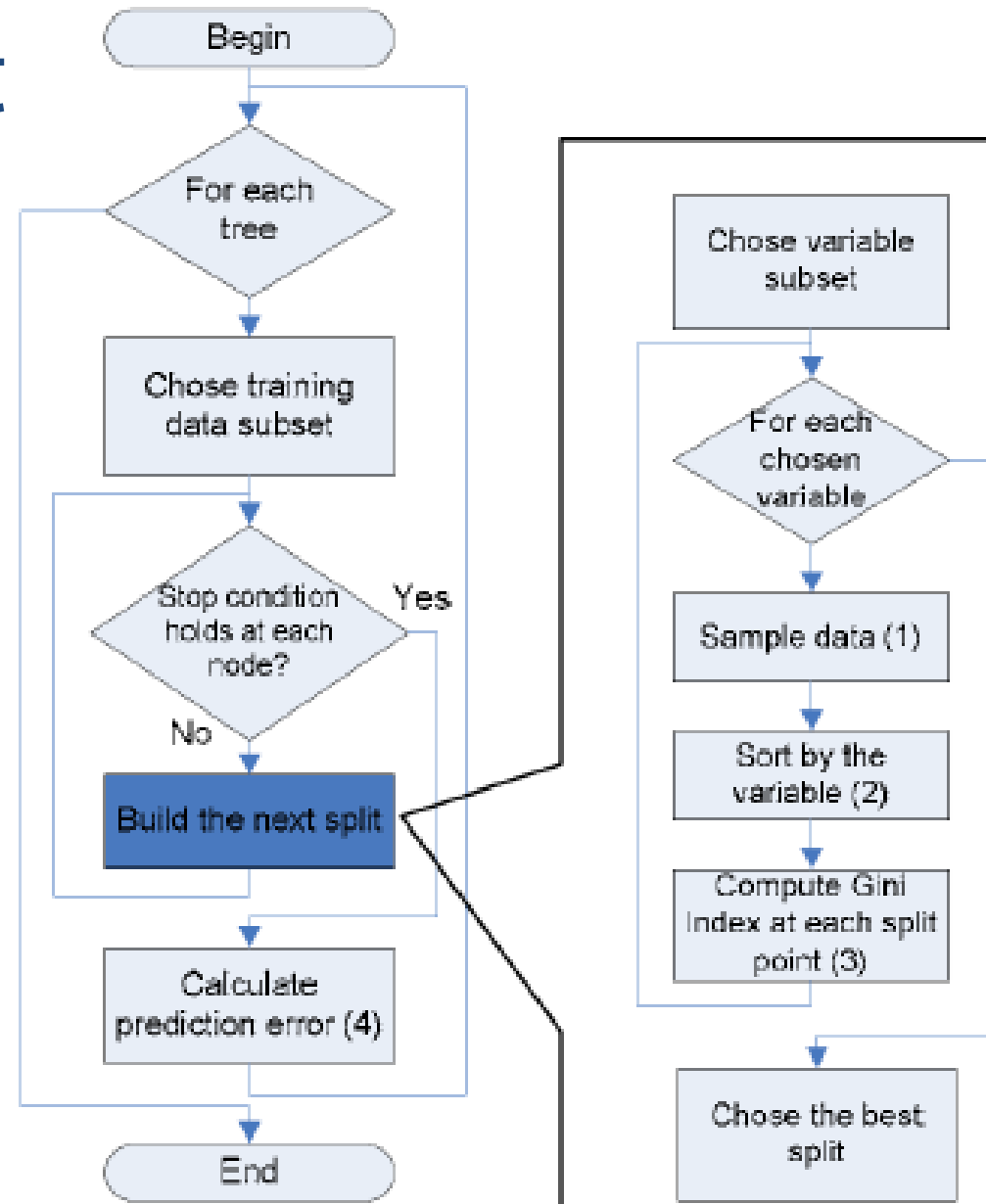
Each tree is constructed using the following algorithm:

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

- For computer scientists:



Random Forest – practical consideration

- Splits are chosen according to a purity measure:
 - E.g. squared error (regression), Gini index or deviance (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.

RM - 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

RM - Additional information

Estimating the importance of each predictor:

- Denote by \hat{e} the OOB estimate of the loss when using original training set, D .
- For each predictor x_p where $p \in \{1, \dots, k\}$
 - Randomly permute p th predictor to generate a new set of samples $D' = \{(y_1, x'_1), \dots, (y_N, x'_N)\}$
 - Compute OOB estimate \hat{e}_k of prediction error with the new samples
- A measure of importance of predictor x_p is $\hat{e}_k - \hat{e}$, the increase in error due to random perturbation of p th predictor

Conclusions & summary:

- Fast fast 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