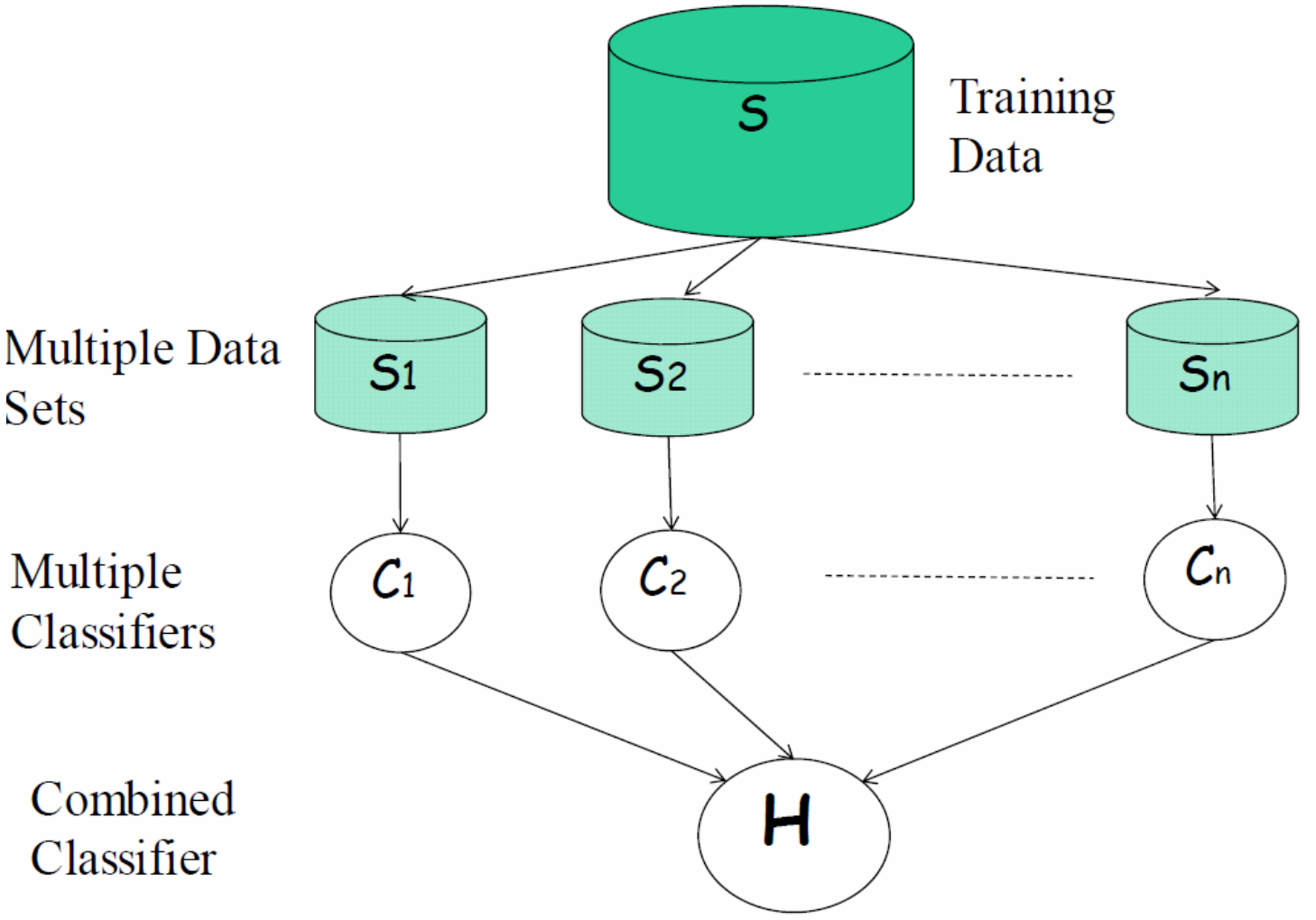
CS 383 – Machine Learning

Ensemble Learning

Agenda

- Ensemble Methods
  - Bagging
  - Voting
  - Boosting
  - Random Forests

Ensemble Learning: General Idea



Ensemble Learning

- Basic idea: Build different “experts” and let them collaborate to come up with a final decision
- Advantages:
  - Improve predictive performance
  - Different types of classifiers can be directly included
  - Easy to implement
  - Not too much parameter tuning (other than that of the individual classifiers themselves)
- Disadvantages
  - Not compact
  - Combine classifier not intuitive to interpret

Why do they work?

- Suppose there are 25 base classifiers
  - Each with an error rate of $\varepsilon = 0.35$
    - Which isn't that great of a rate
- If we assume independence among classifiers (which is a BIG assumption) to make a wrong decision at least 13 (the majority) of the classifiers would have to choose the wrong class.
- Therefore the probability that the ensemble classifier makes a wrong prediction is

$$\sum_{i=13}^{25} \binom{25}{i} \epsilon^i (1-\epsilon)^{25-i} = 0.06$$

Why do they work?

- Even without the independence assumption different classifiers can perform better on different data
  - Different parameters
  - Different features
  - Different training sets
  - Different classifier types
- Still there's no guarantee in all cases that ensemble classification will out perform single classifiers
- To maximize gain:
  - Make sure each classifier is accurate (at least better than average)
  - Make the classifiers have diverse errors (so they're as independent as possible)

Multiple Classifiers

- One way we can obtain our different classifiers for voting is to just choose and train T different *types* of classifiers.
- Another approach is to train classifiers using slightly different training sets.
- We could partition our dataset into T subsets and train a classifier on each
  - But then we'll have a greatly reduced number of training samples for each classifier.
- Alternatively, for each classifier we could we chose M training samples from our N possible samples *with replacement*
  - This approach is called *bagging.*
  - Of course with bagging…
    - We may get some samples several times.
    - We may not get others at all.

Voting

- Ok so we created T different classifiers $c_1, c_2, \ldots, c_T$
  - Now what?
- Classification: Given an unseen sample x
  - Each classifier c_j returns the probability that x belongs to class $i = 1, \ldots C$ as $P_{ji}(x)$
    - Or if they can't return probabilities, they will return $P_{ij}(x) \in \{0,1\}$
  - Now we can decide how to combine these "votes" to get a value (probability?) for each class, y_i
  - And make our final decision based on that!

Voting

- There are many different ideas on how to combine the opinions of classifiers.
  - We might also want to take the *training* accuracy into account for each classifier, α_j
- Here's some idea on how to compute the probability of belonging to class k, $\widehat{y_k}$,(using classifiers $c_1, .., c_T$)
  - Mean: $\widehat{y_k} = \frac{1}{T}\sum_{j=1}^{T} P_{jk}$
  - Weighted Mean: $\widehat{y_k} = \sum_{j=1}^{T} \alpha_j P_{jk}$ where $\sum_j \alpha_j = 1$
  - Median : $\widehat{y_k} = \text{median}_\text{j}(P_{jk})$
  - Minimum: $\widehat{y_k} = \min_\text{j}(P_{\text{j}k})$
  - Maximum: $\widehat{y_k} = \max_\text{j}\left(P_{jk}\right)$
  - Product: $\widehat{y_k} = \prod_j P_{jk}$

	Class 1	Class 2	Class 3
Classifier 1	0.2	0.5	0.3
Classifier 2	0.0	0.6	0.4
Classifier 3	0.4	0.4	0.2
Mean	0.2	**0.5**	0.3
Median	0.2	**0.5**	0.4
Min	0.0	**0.4**	0.2
Max	0.4	**0.6**	0.4
Product	0.0	**0.12**	0.032

Boosting

- Boosting takes the idea of bagging one step further!
- Hypothesis:
  - Detect which samples we did poorly on.
  - We want more of them in our next classifier!
  - So hopefully we'll get 'em next time!
- In the following slides let $c_t(x)$ be the class chosen for sample x by classifier c_t

General Boosting Algorithm

- Given:
  - Training set $\{X, Y\}_{i=1}^{N}$ where $y \in \{-1, +1\}$
  - Initialize sampling distribution D_1 on $\{1, \dots, N\}$
- For iteration $t = 1, \dots, T$
  - Create a training set S_t of size M from $\{X, Y\}_{i=1}^{N}$ using distribution D_t
  - Train a classifier c_t using this new training data
  - Compute the **training** error for this classifier
  - Update the sample distribution, D_{t+1}, by increasing the probability of selecting samples which are misclassified by c_t, for all $\{X, Y\}_{i=1}^{N}$
- Combine the T classifiers to make a final decision, weighting their "opinion" by their training error

AdaBoost

- AdaBoost (adaptive boosting) is a very popular boosting algorithm

Training Process:

- $D_1 = \frac{1}{N}$ (treat all samples equally at first)
- For $t = 1\ to\ T$
  - Train classifier c_t using samples drawn according to distribution D_t and obtain the training error ϵ_t
  - If $\epsilon_t > \frac{1}{2}$ stop (and discard this poor classifier)
  - Otherwise
    - Compute $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$
    - Go through all samples X_i in the full training set and
      - If $c_t(X_i) == Y_i$ then $D_{t+1}^i = \beta_t D_t^i$ (reduce probabilities of things positively classified)
      - Else $D_{t+1}^i = D_t^i$
    - Normalize the probabilities
      - $Z_{t+1} = \sum_{i=1}^{N} D_{t+1}^i$
      - $D_{t+1}^i = \frac{D_{t+1}^i}{Z_{t+1}}$

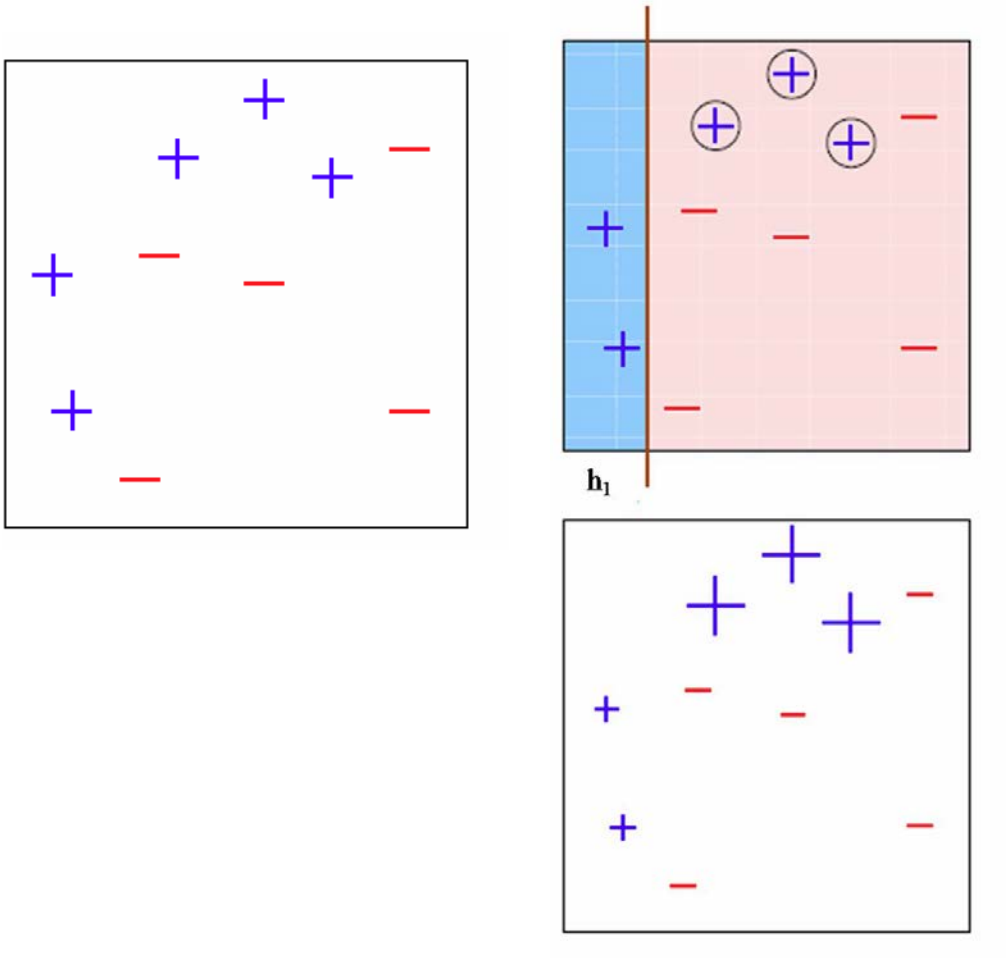
AdaBoost

- Testing:
  - Given x compute the probability of x belong to class i from classifier t as $P_{ti}(x)$ for $t = 1, \dots, T, i = 1, \dots, C$
  - Calculate the class outputs, $i = 1, \dots, C$ as

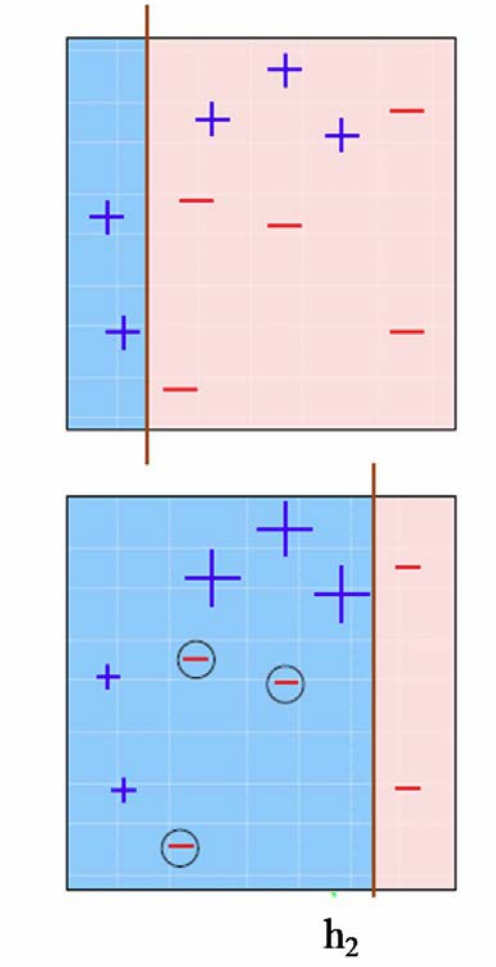
$$\hat{y}_i = \sum_{t=1}^{T} \log\left(\frac{1}{\beta_t}\right) P_{ti}(x)$$

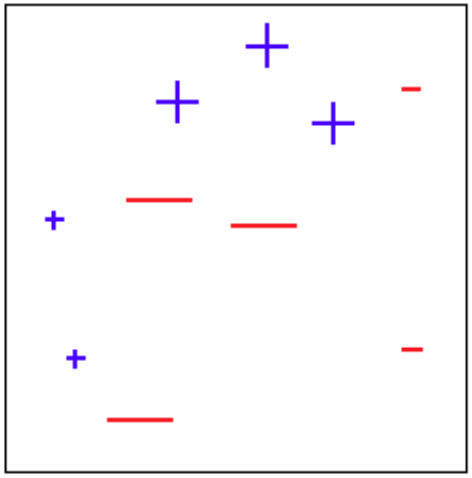
  - And choose label as $\hat{y} = argmax_i\ \hat{y}_i$

Toy Example

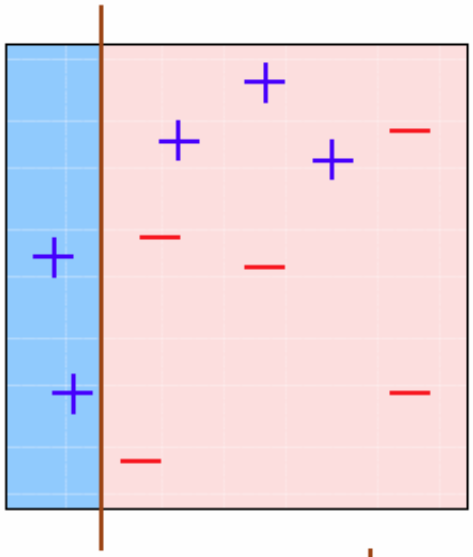


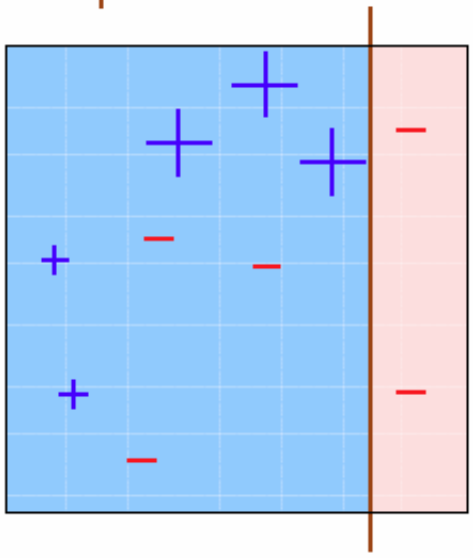
Toy Example

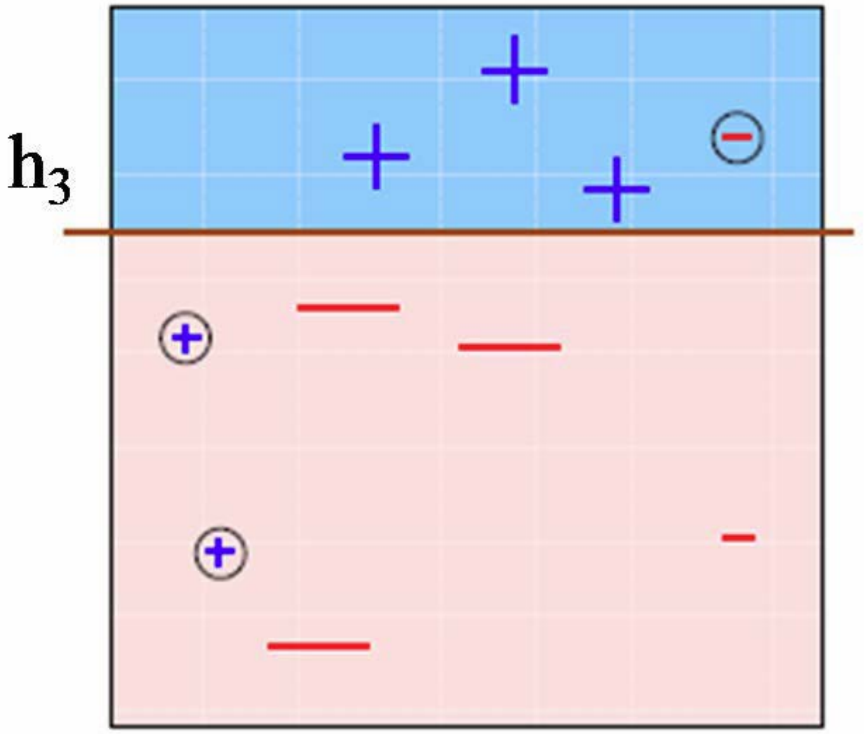




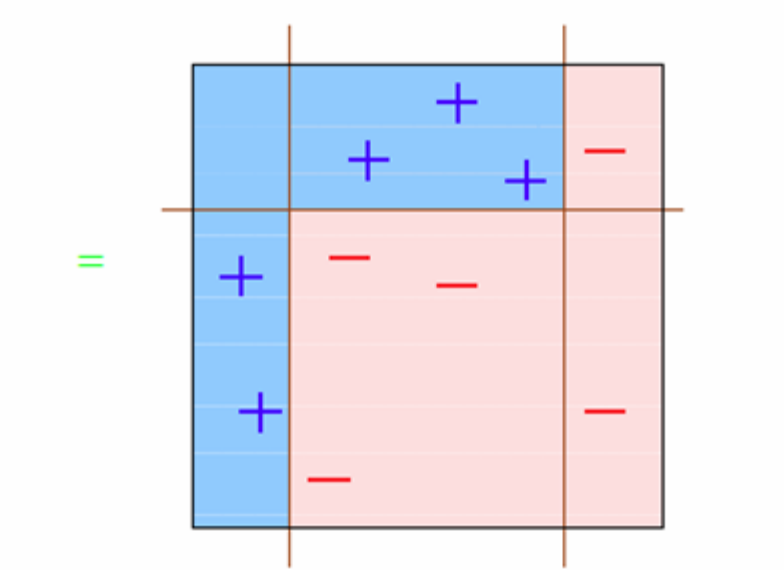
Toy Example







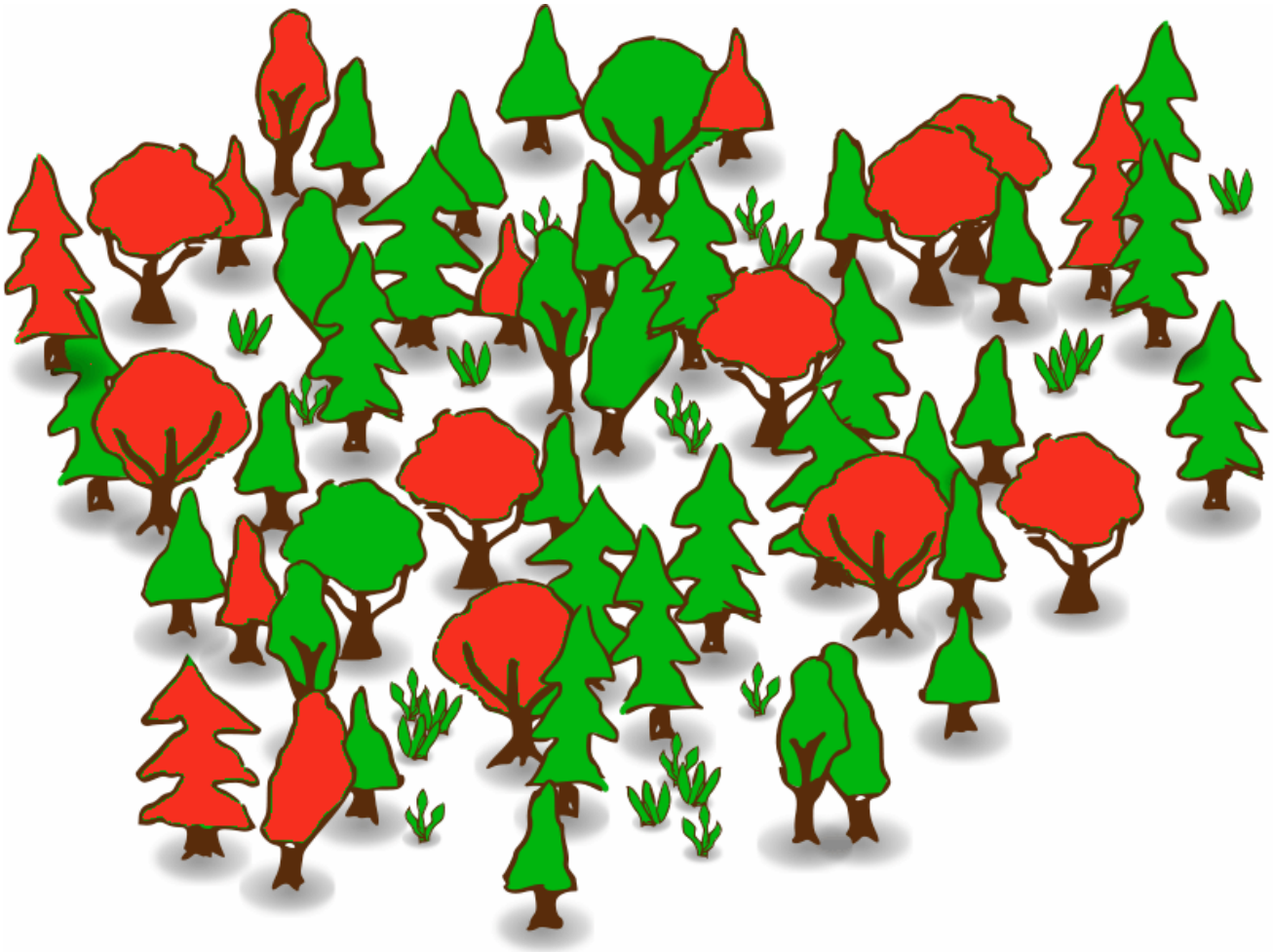
Toy Example: Final Hypothesis



Boosting

- Advantages
  - Simple and easy to implement
  - Flexible – can work for any learning algorithm
  - No parameters to tune
  - Versatile – can work on a lot of problems
- Disadvantages
  - Performance depends on data and weak learner
  - Can fail if weak classifier is too complex (overfitting) or too weak (under fitting)

Random Forests



Tree Bagging/Boosting

- One could of course just using the voting, bagging, or boosting ensemble approach with decision trees as the weak classifiers
  - For T iterations
    - Select a subset of the training set and train classifier t
  - For a new input x, get classification from each of the T trees and make a final decision (weighted sum, majority voting, etc..)

Random Forest

- Random forests take an approach of *feature bagging* where during training node splits are based on a random set of *features*

Random Forest Algorithm

- Repeat the following T times
  - Draw a training set S_t of size M from the original training data with replacement.
  - Grow a random-forest tree c_t using data set S_t by recursively repeating the following steps for each terminal node of the tree until there are no more features to pull from (build a full tree)
    - Select m variables at random from the remaining p variables
      - Empirical results suggest trying $m = \frac{1}{2}\sqrt{p}, \sqrt{p}\ and\ 2\sqrt{p}$
    - Pick the best variable/split-point among the m (information gain?)
    - Split the node into two daughter nodes
- Output the ensemble of trees $\{c_t\}^T$
- To make a prediction at a new point x:
  - Combine the trees to make a final decision
  - Just like with our other ensemble approaches

Example

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Example

- Let's decide to select $M = 7$ samples at random with replacement and each time we need to decide on which features to look at for splitting we'll look at $m = ceil(\sqrt{p})$ randomly selected ones.
- Below is out "randomly" selected training set for our first tree

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes

Example

- At the root of the tree we have $p = 4$ potential features to look at.
- Since $m = ceil(\sqrt{p}) = 2$ we choose two features at random.
  - Let's assume we got **outlook** and **windy**
- Now we need to figure out which has the higher information gain
  - IG(outlook)=0.5917, IG(windy)=0.0202
  - So choose to split on outlook

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes

Example

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes

- Continuing to build the tree….
- On the sunny branch
  - We have $p = 3$ remaining features {temp, hum, wind} to explore and therefore chose $m = ceil(\sqrt{3}) = 2$ at random and get **temp and hum**
  - For sunny branch
    - IG(temp)=0.5409, IG(hum)=1.0
    - So split on humidity. This provides "perfect" gain so it's children are leaves
- Now move over to the overcast branch…
  - Etc..

Example

- Eventually our tree will be built.
- Now repeat this T times to make a "random forest"
- Make final decision based on output of trees in the forest
  - Using some sort of voting scheme

Random Forest

- Advantages
  - Easy to use
  - Fast
  - No need for pruning trees
  - Accuracy and variable importance generated automatically
  - Overfitting is not a problem
  - Not very sensitive to outliers in training data
  - Easy to set parameters
  - Good performance

- Disadvantages
  - Difficult to interpret