Topics in Data Science

(Lecture 4)

Mohammad Pourhomayoun
Assistant Professor
Computer Science Department
California State University, Los Angeles

Continue:
Ensemble Learning
Bagging, Boosting, and Random Forests

Review: Ensemble Learning

- **Ensemble Learning** is a popular and effective approach to improve the accuracy and performance of a machine learning problem.
- **Ensemble Learning** uses a **group of machine learning algorithms** (called **base** learners), and then combine the results of them to achieve higher accuracy.
  - **Example**: Construct a Strong Classifier by combining several Weak Classifiers!
- Each learner (e.g. classifier) alone may have very poor performance. But, a group of them together can achieve very accurate results.

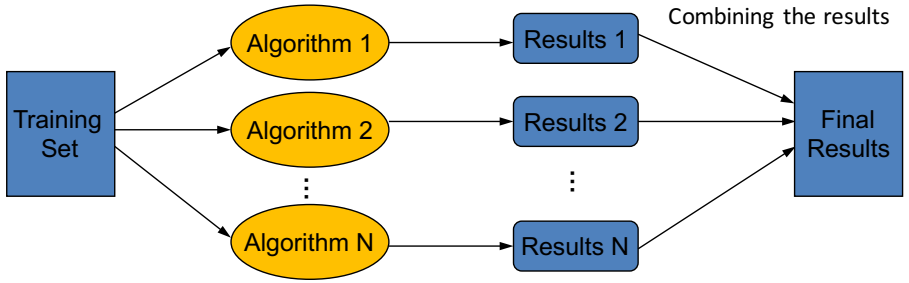
Review: Different types of Ensemble Learning

- **Different learners of the Ensemble Learning may use:**

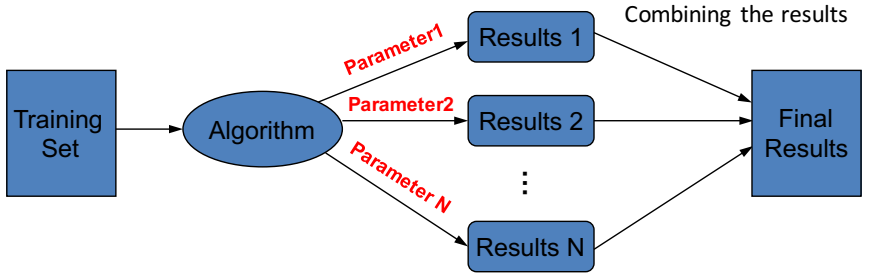
1. **Different learning Algorithms**
   - E.g. Combination of decision tree, KNN, and logistic regression
2. **Different choice of learning Parameters**
   - E.g. Several KNNs with various K's
3. **Different Features**
   - E.g. Several decision trees, each for a set of features
4. **Different Data Subsets**
   - E.g. Several decision trees, each for a section of the dataset
5. **Different Subproblems**
   - E.g. Several logistic regression classifiers, each for a part of the problem

Review: Different types of Ensemble Learning

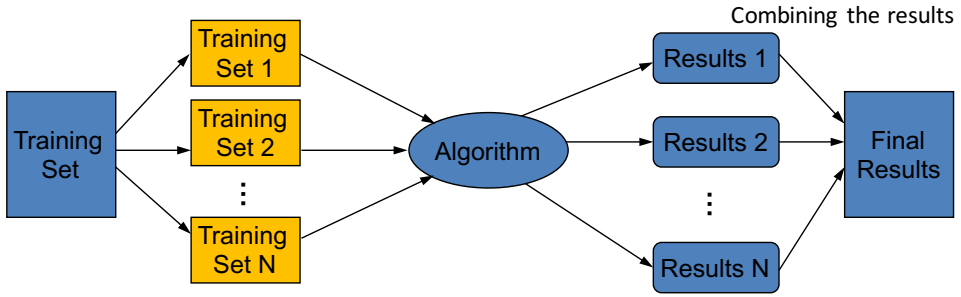
- **Different learning Algorithms:**



- **Different choice of learning Parameters:**



- **Different Features/Subsets:**



An Important Note about Ensemble Learning

- The key in designing ensembles is **diversity** and **not necessarily high accuracy** of the base classifiers.

- Members of an ensemble group **should vary in the examples they misclassify, so that they cover each other's mistakes!**

- In other word, if we have several classifiers that are pretty accurate but they all misclassify the **same samples**, then ensemble learning will not achieve any better results! Therefore, most ensemble approaches, seek to promote diversity among the models they combine.

Three Popular Approaches for Ensemble Learning

- **Bagging**: Bagging (stands for **B**ootstrap **Agg**regating) was first proposed by ***Leo Breiman*** to improve the classifier results by combining classifications of randomly generated training sets.

- **Boosting:** Originally proposed by ***Robert Schapire*** to build a strong classifier using a set of extremely weak base classifiers (with accuracy of slightly better than random guess).

- **Random SubSpace (Random Forest):** First proposed by ***Leo Breiman*** to improve the accuracy of decision tree classifiers and address the overfitting problem.

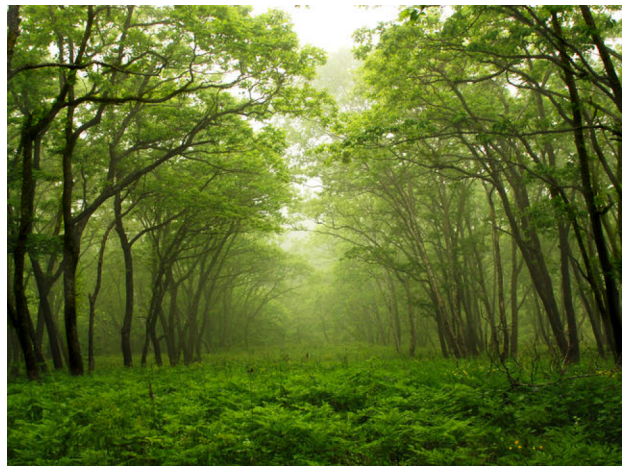

Random Forest

Random Forest

- **Random Forest** (also called Random Decision Forest) is an ensemble learning method, that operates by constructing **several decision trees** at training stage, and then combining the prediction results.
- First proposed by **Leo Breiman** in 2001.
- Random Forest algorithm resolves the well-known problem of **overfitting** in decision tree algorithms by reducing variance and instability.

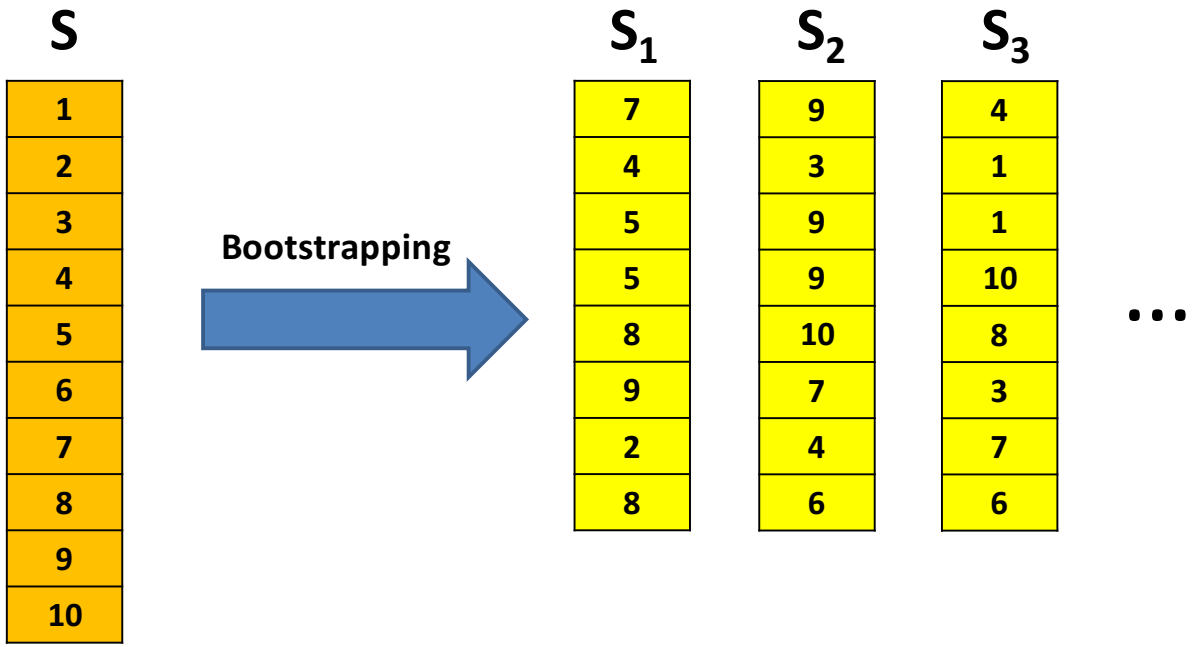
Random Forest

- **Key Note1:** Random Forest uses **Bootstrap Sampling (randomly sampling of data)** to generate several training datasets used to train several decision trees.
- **Key Note2**: Random Forest uses a **random selection of Features** for split on at each node of each decision tree.
- Both of these approaches will help the algorithm build a more accurate predictor, which is also robust to overfitting!



Random Forest Algorithm

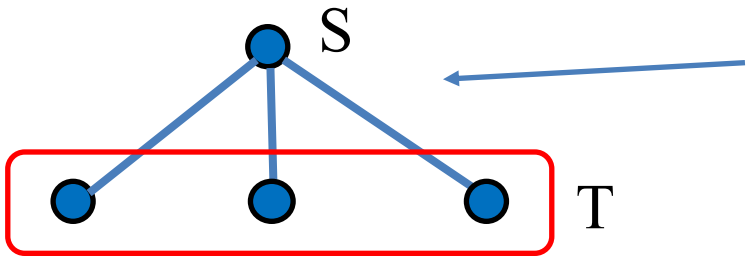
- **Step1 (Bootstrapping):** Suppose we have a training set with ***N* samples** and ***D* features**. Random Forest first uses Bootstrap Sampling (randomly sampling of data) to generate ***L*** training datasets $\mathbf{S_1, S_2, \ldots, S_L}$.



Random Forest Algorithm

- **Step2 (Training):** The *L* new training sets $S_1, S_2, \ldots, S_L$ will be used to **train *L* decision trees. HOWEVER, for each branch split, the algorithm first randomly selects a small number of features, and then uses the best of them for splitting.**

- In other word, unlike regular decision tree that selects the best feature in the entire feature set, "random forest" select the best feature in a very small random subset of features! **Thus, the features that are used in each tree may differ from another tree!**



At each node, this is the best feature selected from a **small random subset of features.**

- **Note:** If *d* is the size of random feature subset at each node, the *d* is much smaller than the size of the entire feature set (*d*<<*D*, e.g. *d* = sqrt (*D*)).

Random Forest Algorithm

- **Step3 (Base Learner Prediction):** Given a new unknown data sample, all trained decision trees (*L* trees) make their prediction for the new sample.
- **Step4 (Voting):** The final decision will be made using **Voting** method. The final prediction is based on the majority vote of the *L* decision trees.

Pseudo Code for Random Forest

Given a training set S

For i = 1 to L do:

Build subset S_i by sampling with replacement from S.

Learn tree T_i from S_i:

At each node of tree T_i :

Choose best split from a random subset of *d* features.

Each trained tree makes prediction about a new sample.

Make final prediction according to majority vote of the *L* trees.

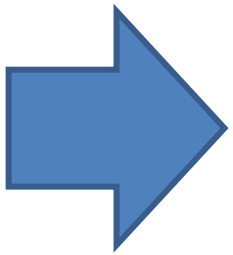
Advantages of random forest

- One of the most accurate classification algorithms!
- Very Robust to Noise and Overfitting.
- It can handle big data including hundreds of features.
- It can handle missing value!

Why does Random Forest work so well?

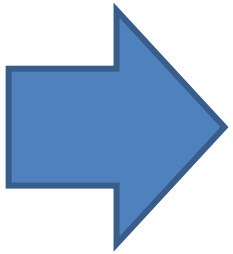
- **Question1: Why is Random Forest much better than a single Decision Tree?**

- It can reduce variance and resolve the overfitting problem.



Why does Random Forest work so well?

- Question2: Why is Random Forest much better than Bagging with Decision Trees?
- In other word, Why does using a subset of features at each node (rather than all features) help a lot?



Why does Random Forest work so well?

- **Question2: Why is Random Forest much better than Bagging with Decision Trees?**
- **In other word, Why does using a subset of features at each node help a lot?**
- **Review:**
  - The key of designing ensembles is **diversity!**
  - Members of the ensemble **should be uncorrelated and vary in the examples they misclassify, so that they can compensate for each others mistakes!**

Why does Random Forest work so well?

- **Question2: Why is Random Forest much better than Bagging with Decision Trees?**

- If each time we select from the entire feature set, every tree always selects the **best features one after another**, and consequently, the structure of all trees will be **the same**.

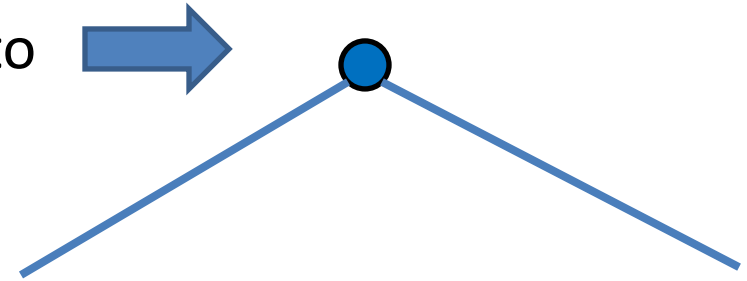
- On the other hand, Selecting only from a random subset of the features at each node, makes our trees **different**.

- Random Forest tries to generate the set of tress that are **different from each other**. Nonetheless, each one does **its best in its own way**.

- After voting, the trees will **compensate for each others mistakes**, and provide the best results together.

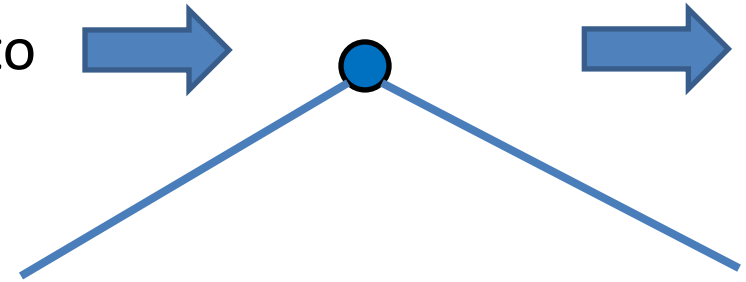
Example: Tree1 in Random Forest for Titanic (using total 3 features {age, gender, pclass})

At this node, let's limit our feature subset to {gender,pclass}



Example: Tree1 in Random Forest for Titanic (using total 3 features {age, gender, pclass})

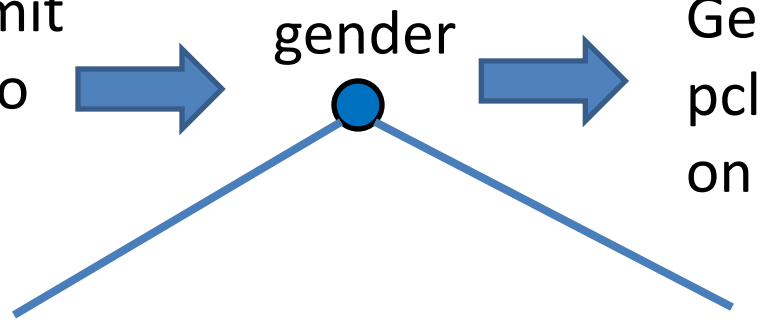
At this node, let's limit our feature subset to {gender,pclass}



Gender is better than pclass, So I split based on gender!

Example: Tree1 in Random Forest for Titanic

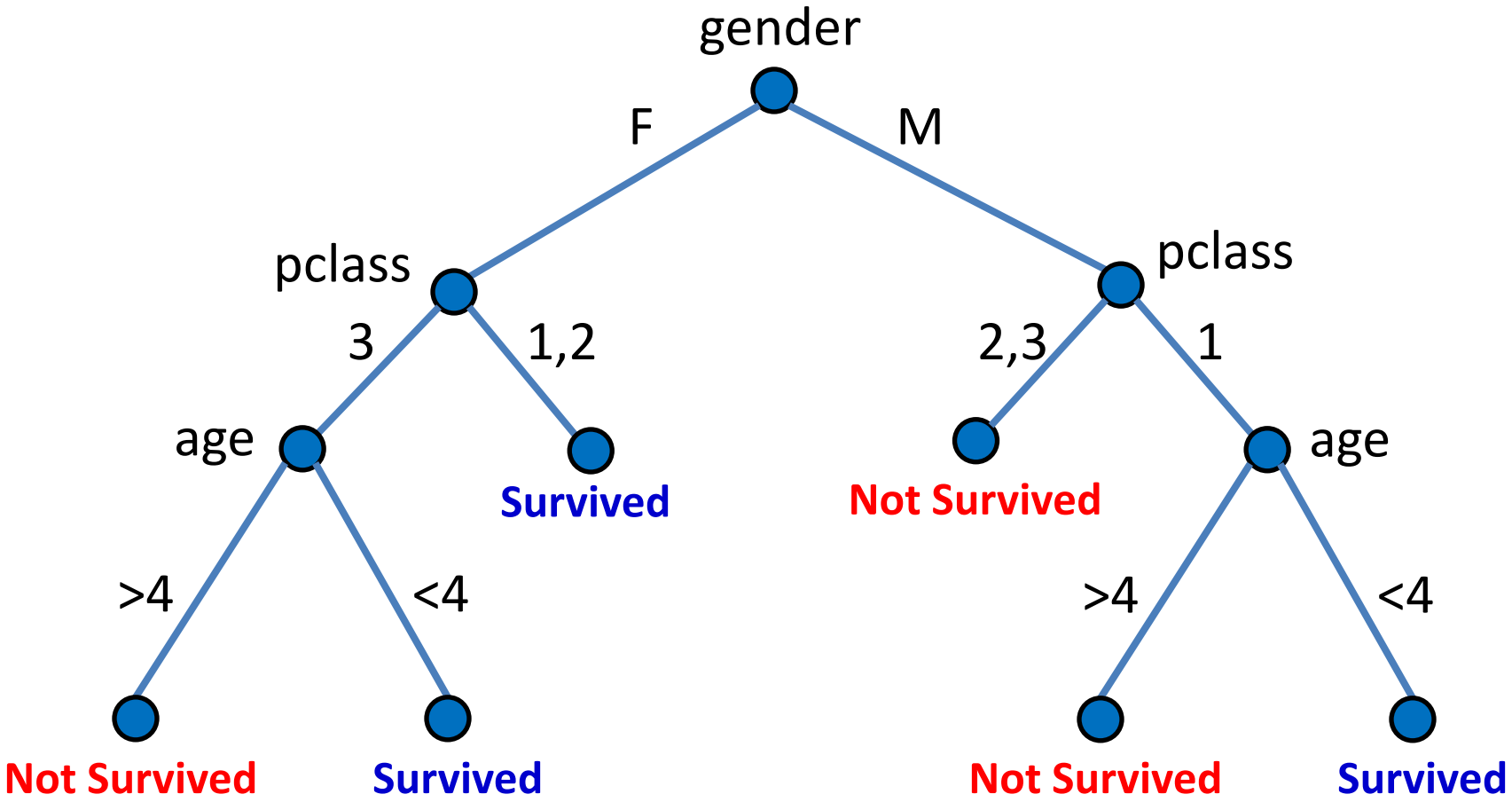
At this node, let's limit our feature subset to {gender,pclass}



Gender is better than pclass, So I split based on gender!

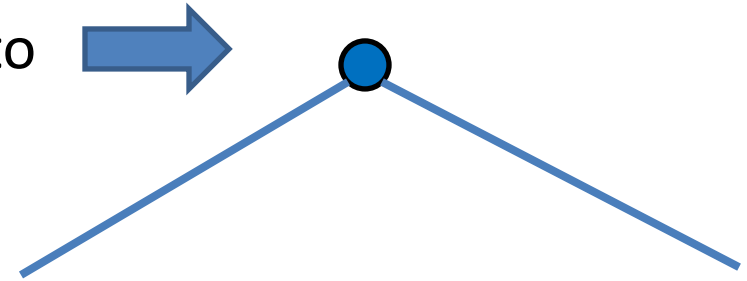
And similarly for the rest of the tree …

Example: Tree1 in Random Forest for Titanic



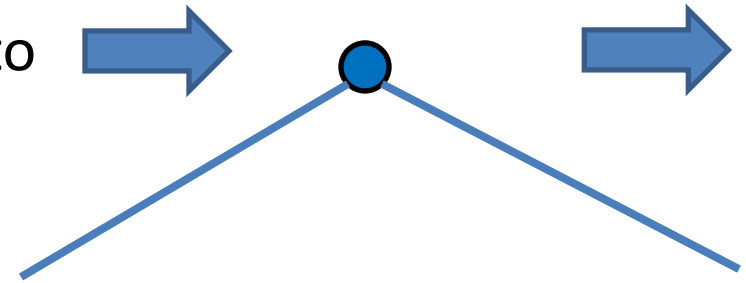
Example: Tree2 in Random Forest for Titanic (using total 3 features {age, gender, pclass})

At this node, let's limit our feature subset to {age,pclass}



Example: Tree2 in Random Forest for Titanic (using total 3 features {age, gender, pclass})

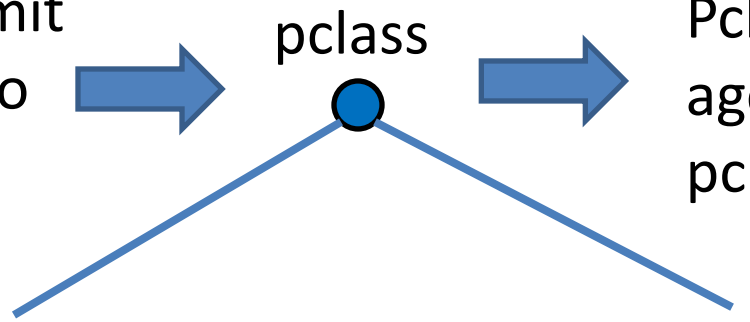
At this node, let's limit our feature subset to {age,pclass}



Pclass is better than age, So I split based on pclass!

Example: Tree2 in Random Forest for Titanic

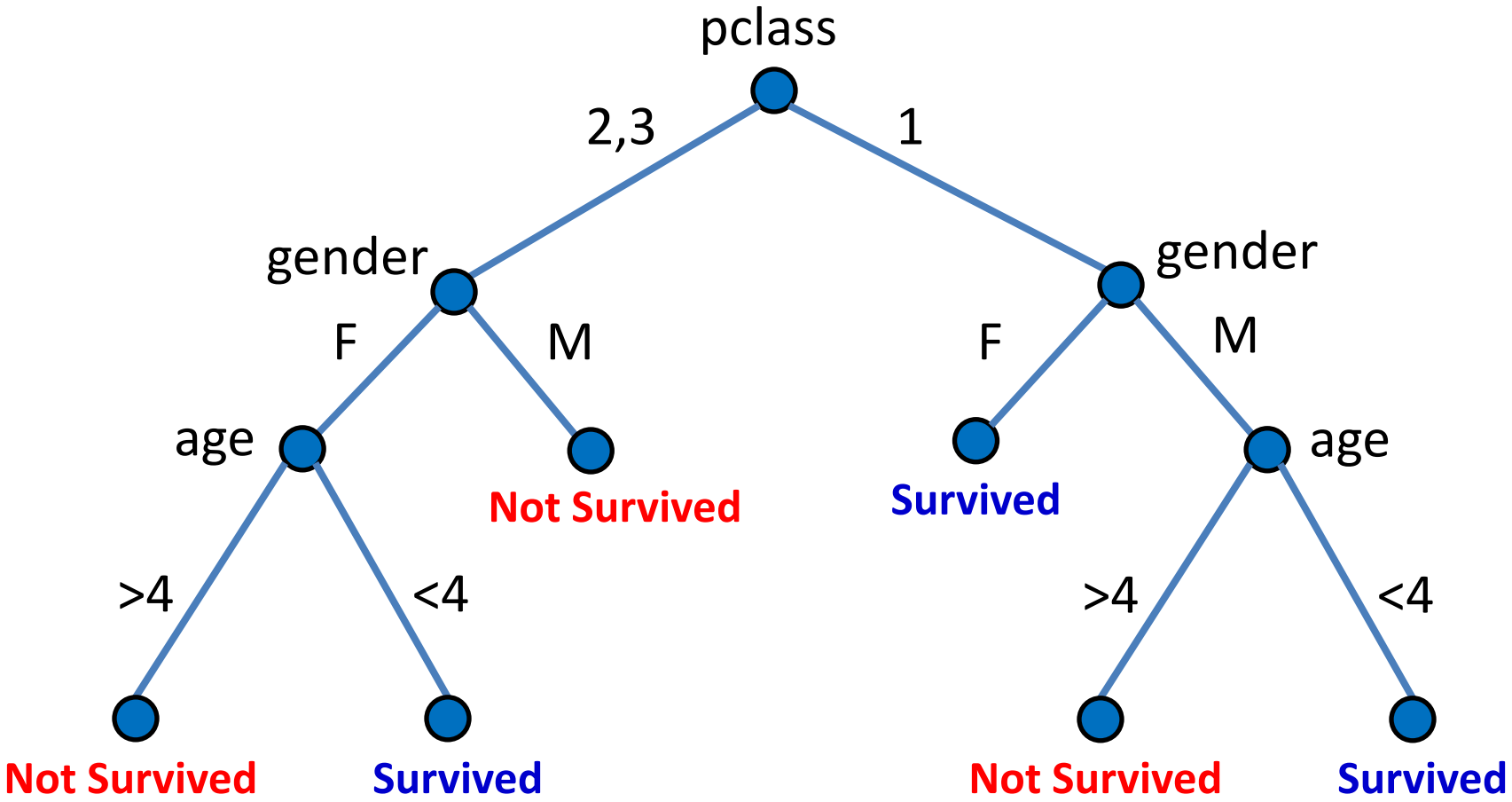
At this node, let's limit our feature subset to {age,pclass}



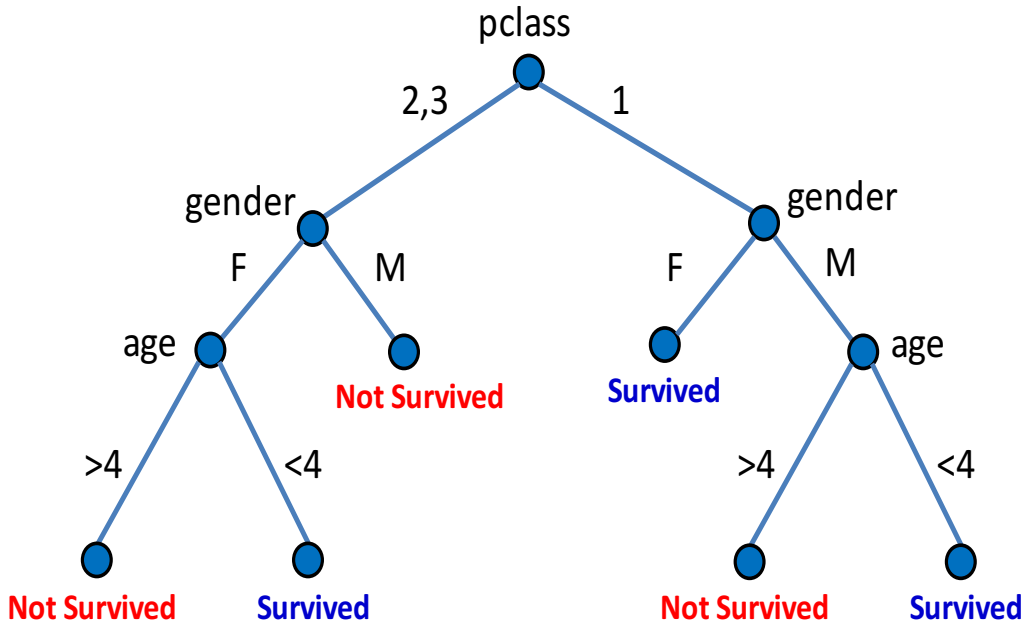
Pclass is better than age, So I split based on pclass!

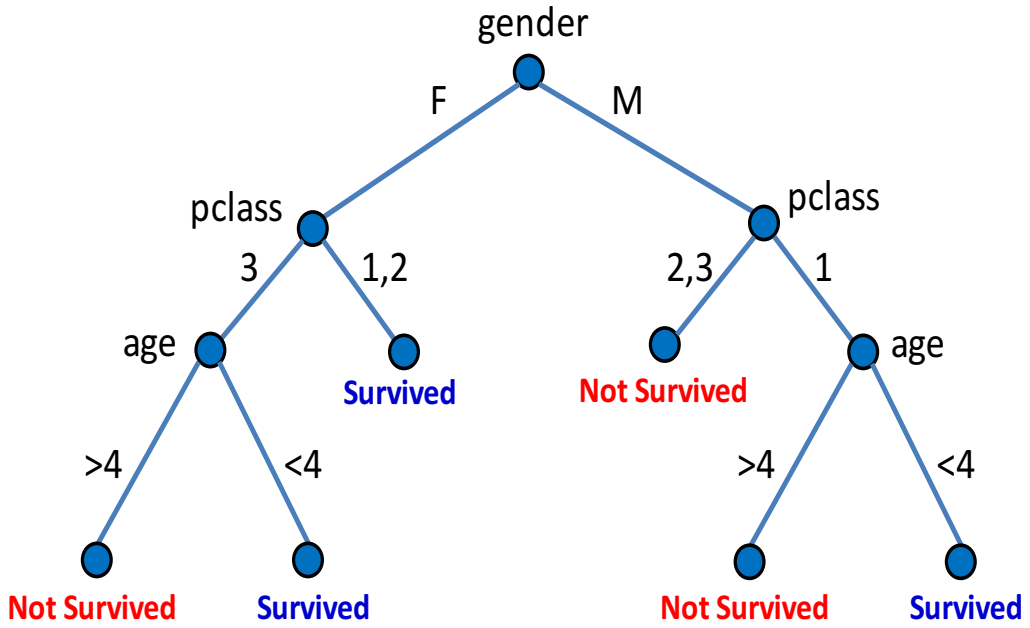
And similarly for the rest of the tree …

Example: Tree2 in Random Forest for Titanic



Example: Random Forest for Titanic





Question: What is your prediction for a passenger who is adult female and has 2nd class ticket?

Boosting

Boosting

- Boosting also tries to build a strong classifier using a set of weak base classifiers.

- Boosting is an **iterative** procedure to **adaptively** change distribution of **training data** by focusing more on previously **misclassified** data samples.

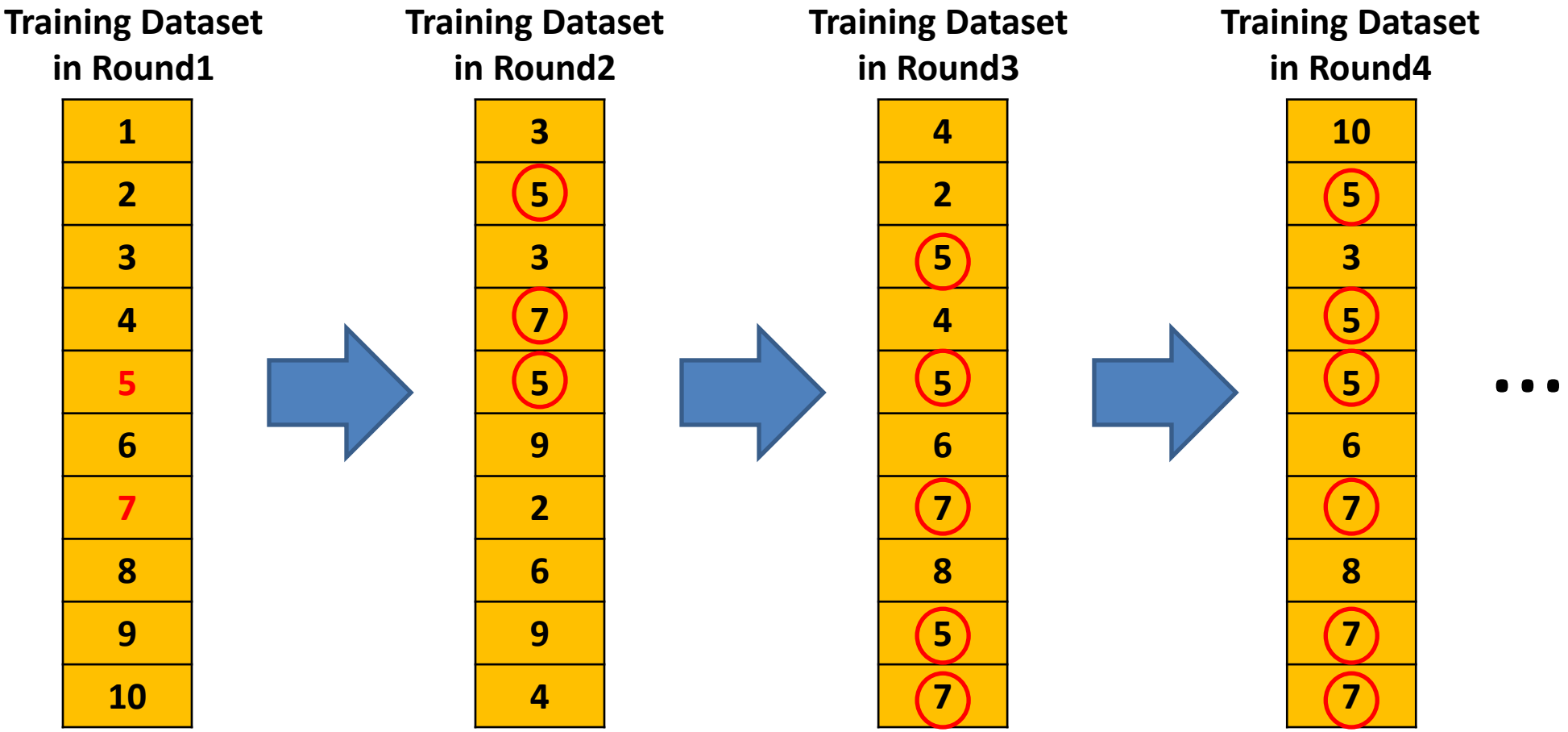
- Instead of sampling randomly and constructing several classifiers independently (as we saw in Bagging and Random Forest), Boosting tries to construct a sequence of predictive models such that new models **focus on data samples that were problematic for previous predictors.**

Boosting

- To make sure that a learner will focus more on the problematic samples, in each iteration we will give **extra weights** **to the samples that misclassified in previous round.**

- In each iteration we can change the distribution of **training set** such that:
  - Records that are wrongly classified will have their weights increased.
  - Records that are classified correctly will have their weights decreased

- In the following example, suppose that **samples 5** and **7** are hard to be classified correctly, and misclassified in the several first rounds.
- Thus, we increase the weight of them in **next training sets** (in the next rounds) to push the **new classifiers** to learn how to classify them correctly!



Boosting

- **Combining:** After training a sequence of **base classifiers** using the above **Adaptive Training Sets**, the algorithm uses a **Weighted Voting** system to make the final decision.
  - Each base classifier has a vote.
  - Better base classifier gets a larger weight.

- Several different implementations of Boosting are available.

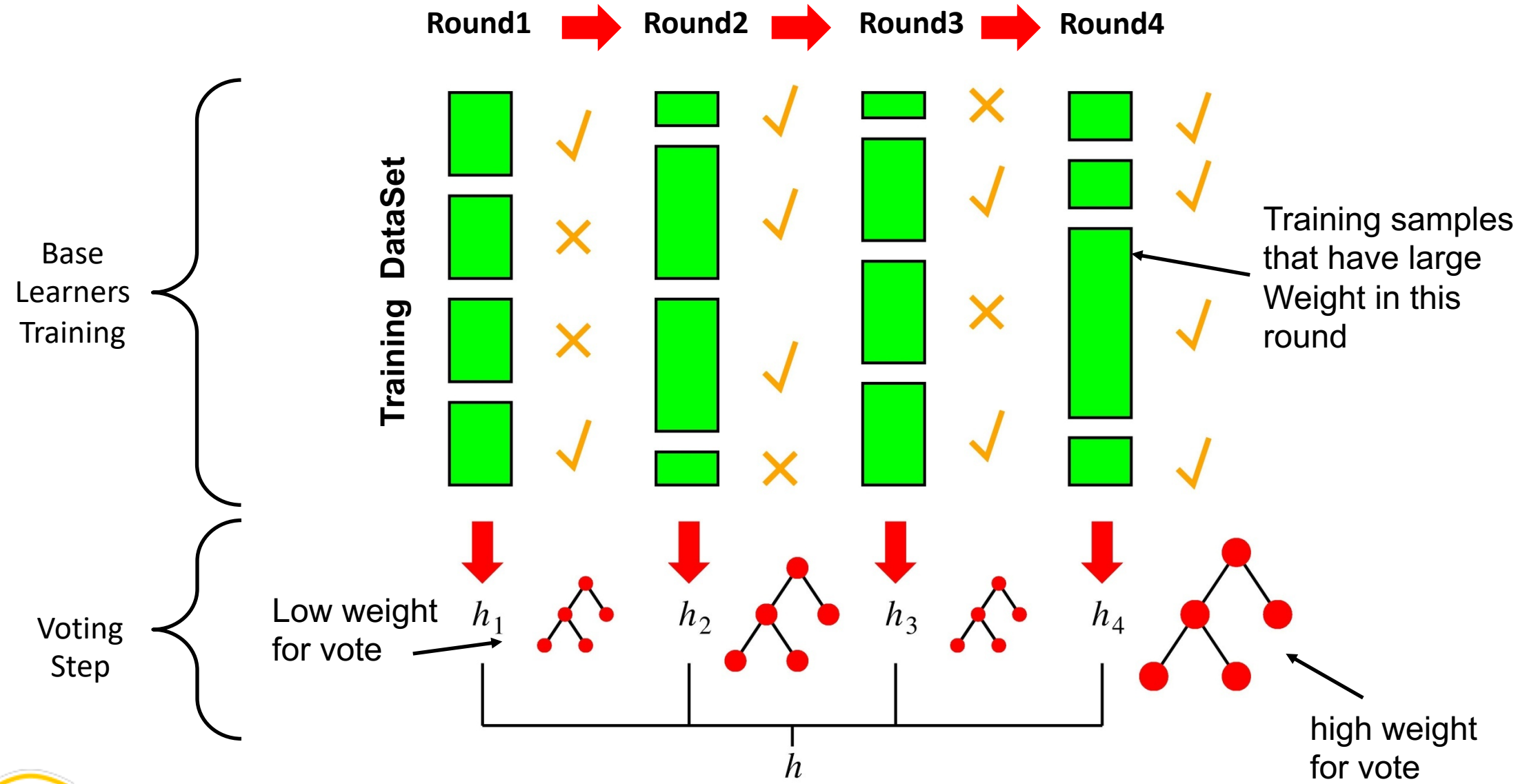
  AdaBoost (**Ada**ptive **Boost**ing) algorithm is one of the most popular approaches.

Boosting

- **In summary, Boosting includes 2 main steps:**

1. Base learners training in an iterative and adaptive fashion by focusing more on previously misclassified data samples in each iteration.
2. Combining the prediction results of the base learners based on a Weighted Voting system to make the final decision.

Boosting in a Picture



Ref: Max Welling, machine learning, UCI

AdaBoost Pseudocode

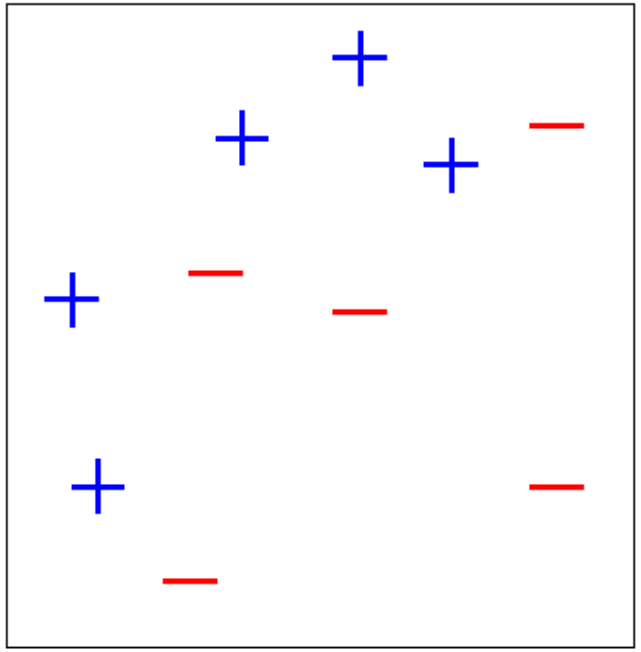
TrainAdaBoost(S, BaseClassifier)

For each example d_i in S let its weight w_i=1/(size of S)
Let H be an empty set of classifiers
For t from 1 to T do:
    train a classifier c_t, from dataset S
    Add c_t to H
    Calculate the error, ε_t, of the classifier c_t as the total sum weight of the examples that it classifies incorrectly.
    Let $\beta_t = \varepsilon_t / (1 - \varepsilon_t)$
    Multiply the weights of the examples that c_t classifies correctly by β_t
    Rescale the weights of all of the examples so the total sum weight remains 1.
Return H

TestAdaBoost(*x*, *H*)

Let each classifier, c_t, in H vote for x's classification with weight $\alpha_t = 0.5 \ln(1/\beta_t)$
Return the class with the highest weighted vote total.

Example

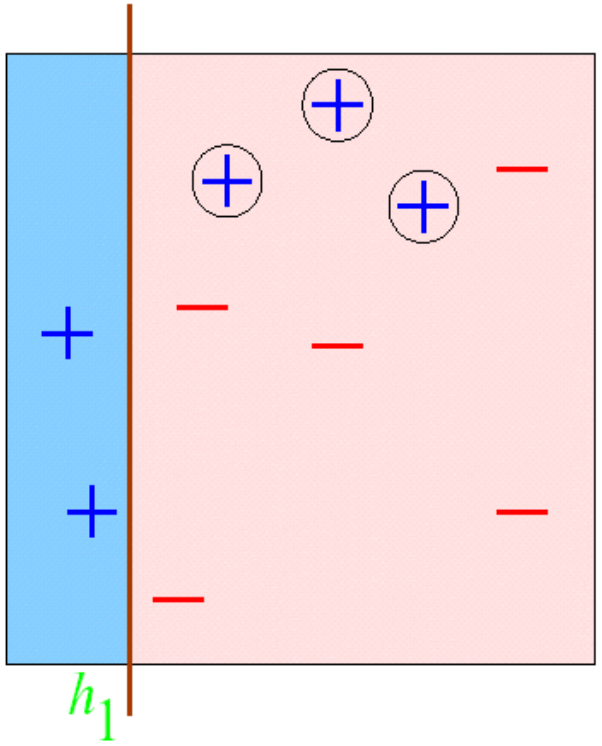


Initial Training Set: Equal Weights to all training samples

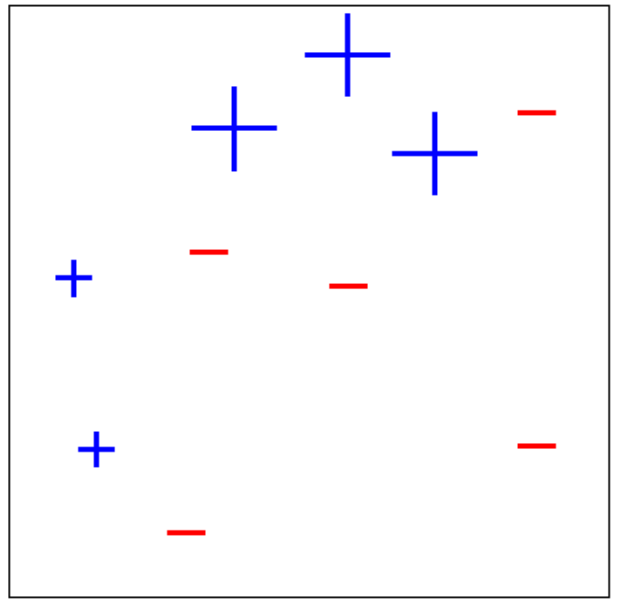
Ref: "A Tutorial on Boosting" by Robert Schapire and Yoav Freund

Example: Round1

Classifier in Round1

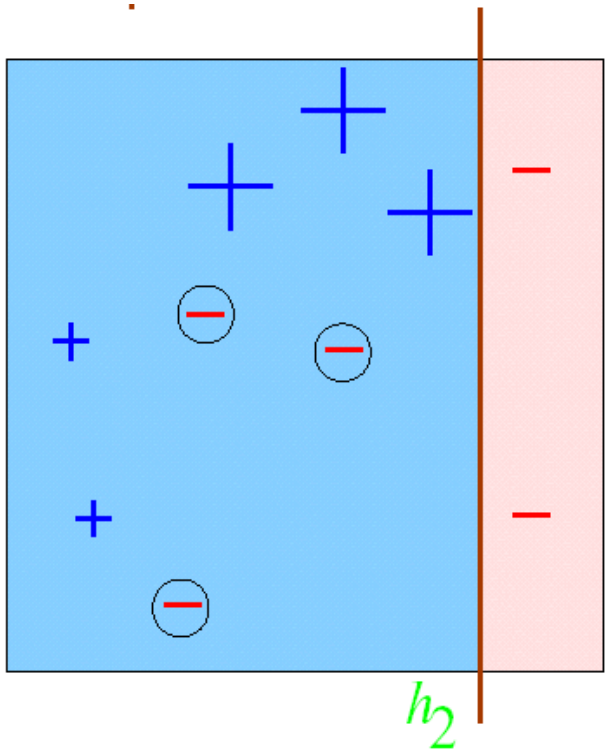


Training Set for next Round

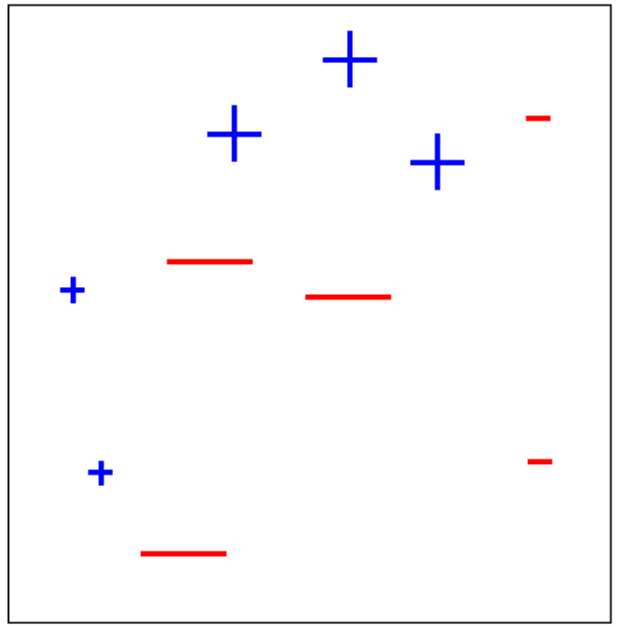


Example: Round2

Classifier in Round2

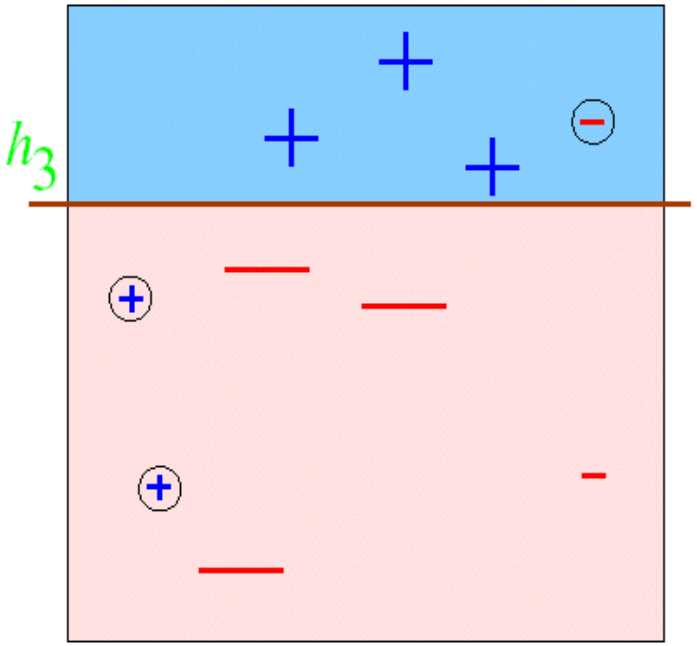


Training Set for next Round

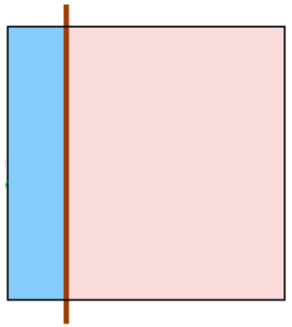
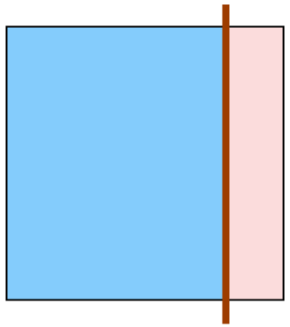
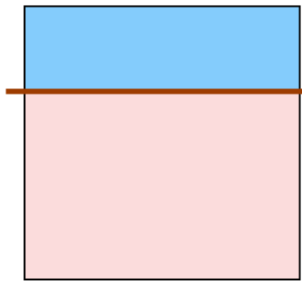


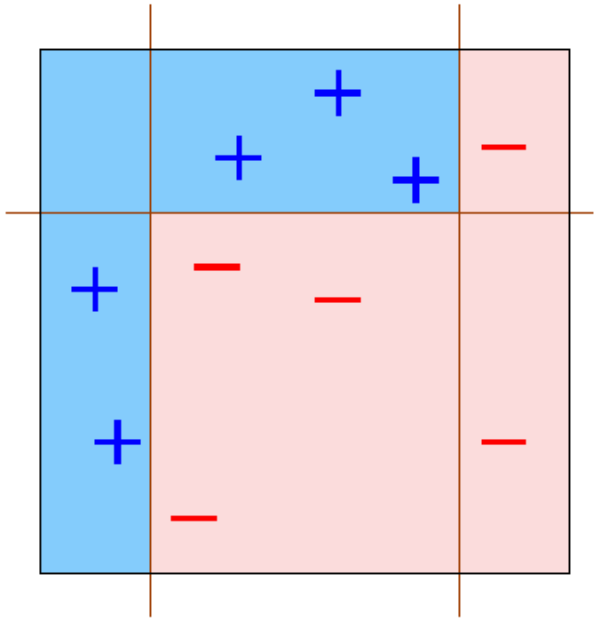
Example: Round3

Classifier in Round3

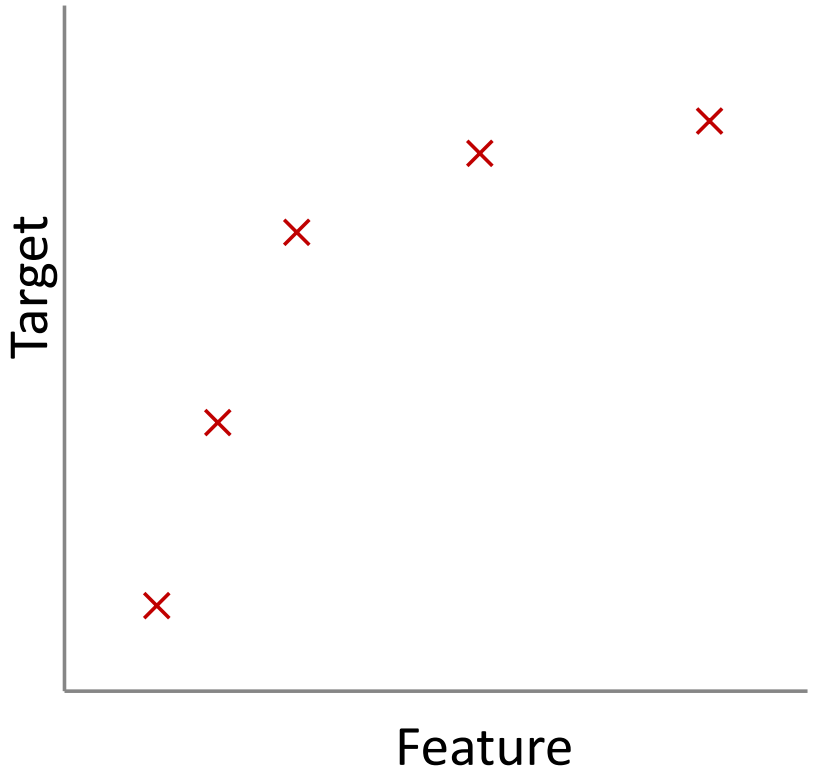


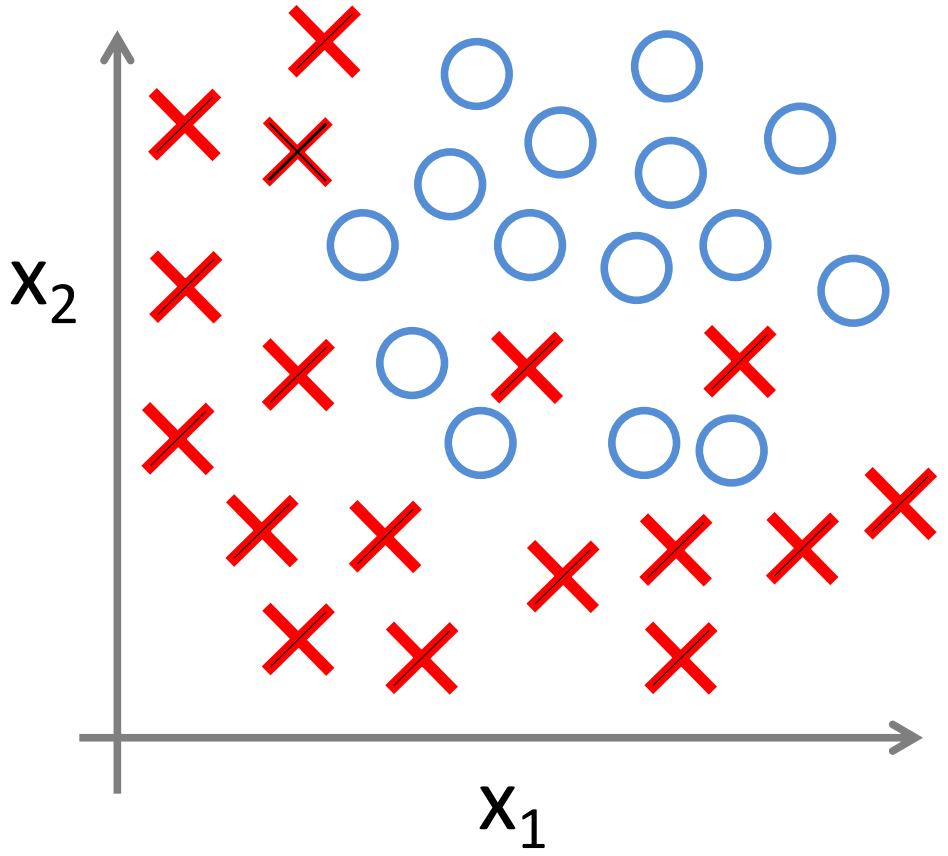
Example: Final Classifier

$sign\,(\,\alpha_1$  $+\alpha_2$  $+\alpha_3$  $)$

$=$ 

How Can Boosting Improve the Accuracy?





Thank You!

Questions?