More on Classification

- More Classification Models
  - Bayes classifier, Association-based classifier, k-nearest neighbor (kNN), Neural network (shallow and deep models), etc.
- Classification Measures
- Ensemble Methods
  - Ideas and Examples
  - Bagging: Random Forest
  - Boosting: Adaboost
- Take-home messages

More Classification Models

Bayesian Classification: Why?

- Probabilistic learning: Calculate explicit probabilities for hypothesis, among the most practical approaches to certain types of learning problems
- Incremental: Each training example can incrementally increase/decrease the probability that a hypothesis is correct. Prior knowledge can be combined with observed data.
- Probabilistic prediction: Predict multiple hypotheses, weighted by their probabilities
- Standard: Even when Bayesian methods are computationally intractable, they can provide a standard of optimal decision making against which other methods can be measured

Bayesian Theorem

- Given training data *D, posteriori probability of a hypothesis h, P(h|D)* follows the Bayes theorem

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)} \qquad \text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}$$

- MAP (maximum a-posteriori) hypothesis

$$h_{MAP} \equiv \underset{h \in H}{\arg\max}\, P(h|D) = \underset{h \in H}{\arg\max}\, P(D|h)P(h).$$

- Practical difficulty: require initial knowledge of many probabilities, significant computational cost

Bayesian classification

- The classification problem may be formalized using a-posteriori probabilities:
- P(C|X) = probability that the sample tuple
  $X=<x_1,\ldots,x_k>$ is of class C.

  e.g. P(buys_computer=Yes |Age≤30, Income=high, Student=no, Credit_rating=fair)

  e.g. P(class=N |outlook=sunny,windy=true,…)
- Idea: assign to sample X the class label C such that P(C|X) is maximal

Estimating a-posteriori probabilities

- Bayes theorem:

$$P(C|X) = P(X|C) \cdot P(C) \,/\, P(X)$$

- P(X) is constant for all classes
- P(C) = relative freq. of class C samples
- choose C such that $P(C|X)$ is maximum =

  choose C such that $P(X|C) \cdot P(C)$ is maximum
- Problem: computing $P(X|C)$ is unfeasible!

X is multidimensional
(multiple attributes)!

Naïve Bayes Classifier

- A simplified assumption: attributes are conditionally independent:

$$P(C_j|X) \propto P(C_j)\prod_{i=1}^{n} P(x_i|C_j)$$

- Greatly reduces the computation cost, only count the class distribution.

Naïve Bayesian Classification

- Naïve assumption: attribute independence

$$P(x_1,\ldots,x_k|C) = P(x_1|C)\cdot\ldots\cdot P(x_k|C)$$

- If i-th attribute is categorical:
  $P(x_i|C)$ is estimated as the relative freq of samples having value x_i as i-th attribute in class C
- If i-th attribute is continuous:
  $P(x_i|C)$ is estimated thru a Gaussian density function
- Computationally easy in both cases

Play-tennis example: estimating $P(x_i|C)$

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	n
sunny	hot	high	true	n
overcast	hot	high	false	p
rain	mild	high	false	p
rain	cool	normal	false	p
rain	cool	normal	true	n
overcast	cool	normal	true	p
sunny	mild	high	false	n
sunny	cool	normal	false	p
rain	mild	normal	false	p
sunny	mild	normal	true	p
overcast	mild	high	true	p
overcast	hot	normal	false	p
rain	mild	high	true	n

P(p) = 9/14
P(n) = 5/14

outlook	
P(sunny|p) = 2/9	P(sunny|n) = 3/5
P(overcast|p) = 4/9	P(overcast|n) = 0
P(rain|p) = 3/9	P(rain|n) = 2/5
temperature	
P(hot|p) = 2/9	P(hot|n) = 2/5
P(mild|p) = 4/9	P(mild|n) = 2/5
P(cool|p) = 3/9	P(cool|n) = 1/5
humidity	
P(high|p) = 3/9	P(high|n) = 4/5
P(normal|p) = 6/9	P(normal|n) = 1/5
windy	
P(true|p) = 3/9	P(true|n) = 3/5
P(false|p) = 6/9	P(false|n) = 2/5

Play-tennis example: Classifying X

- An unseen sample X = <rain, hot, high, false>
- P(X|p)·P(p) =
  P(rain|p)·P(hot|p)·P(high|p)·P(false|p)·P(p) =
  3/9·2/9·3/9·6/9·9/14 = 0.010582
- P(X|n)·P(n) =
  P(rain|n)·P(hot|n)·P(high|n)·P(false|n)·P(n) =
  2/5·2/5·4/5·2/5·5/14 = 0.018286
- Sample X is classified as class n (don't play tennis)

Association-Based Classification

- Several methods for association-based classification
  - Associative classification: (Liu et al’98)
    - It mines high support and high confidence rules in the form of “cond_set => y”, where y is a class label
  - ARCS: Quantitative association mining and clustering of association rules (Lent et al’97)
    - It beats C4.5 in (mainly) scalability and also accuracy
  - CAEP (Classification by aggregating emerging patterns) (Dong et al’99)
    - Emerging patterns (EPs): the itemsets whose support increases significantly from one class to another
    - Mine EPs based on minimum support and growth rate

Eager learning vs lazy learning

- Decision tree is a representative eager learning approach which takes proactive steps to build up a hypothesis of the learning task.
- It has an explicit description of target function on the whole training set
- Let's take a look at a more "relaxed" supervised learning approach, i.e. lazy learning, which can be mostly manifested by the so-called instance-based models.

Instance-Based Classifiers

Set of Stored Cases

Atr1	………	AtrN	Class
			A
			B
			B
			C
			A
			C
			B

Basic idea:

- Store the training records/cases
- Use training records to predict the class label of unseen cases, typically with majority rule applied.

Similar records

Unseen Case

Atr1	………	AtrN

Got 2 votes of class B and 1 vote of class C, so classify the unseen case as class B.

k-Nearest Neighbor (kNN) Classification

ID	Food	Chat	Fast	Price	Bar	BigTip
1	great	yes	yes	normal	no	yes
2	great	no	yes	normal	no	yes
3	mediocre	yes	no	high	no	no
4	great	yes	yes	normal	yes	yes

← Stored Records For BigTip Classification

Similarity metric: Number of matching attributes

Number of nearest neighbors: k=2

Classifying new data:

- New data (x, great, no, no, normal, no)

  → most similar: ID=2 (1 mismatch, 4 match) → yes

  → Second most similar example: ID=1 (2 mismatch, 3 match) → yes

  So, classify it as "Yes".

- New data (y, mediocre, yes, no, normal, no)

  → Most similar: ID=3 (1 mismatch, 4 match) → no

  → Second most similar example: ID=1 (2 mismatch, 3 match) → yes

  So, classify it as "Yes/No"?!

Classification by Neural Networks

- Advantages
  - prediction accuracy is generally high
  - robust, works when training examples contain errors
  - output may be discrete, real-valued, or a vector of several discrete or real-valued attributes
  - fast evaluation of the learned target function
- Criticisms
  - long training time (model construction time)
  - difficult to understand the learned function (weights)
  - not easy to incorporate domain knowledge

Classification by Neural Networks

- Shallow models:
  - Perceptron
  - Multilayer Perceptron (MLP)
  - Support Vector Machine (SVM)…
- Deep models:
  - Convolutional Neural Network (CNN)
  - Recurrent Neural Network (RNN), e.g. Long Short-Term Memory (LSTM)
  - Graph Neural Network (GNN)
  - Transformer…

Classification Measures

Measuring Error

	Predicted class	
True Class	Yes	No
Yes	TP: True Positive	FN: False Negative
No	FP: False Positive	TN: True Negative

- Error rate = # of errors / # of instances = (FN+FP) / N
- Recall = # of found positives / # of positives
  = TP / (TP+FN) = sensitivity = hit rate
- Precision = # of found positives / # of found
  = TP / (TP+FP)
- Specificity = TN / (TN+FP)
- False alarm rate = FP / (FP+TN) = 1 - Specificity

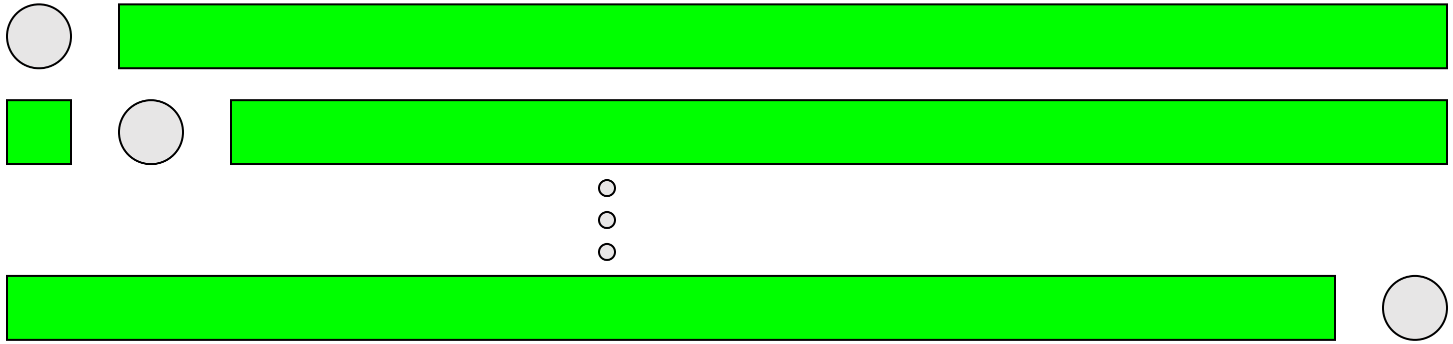
Classification Accuracy:
Estimating Error Rates

- Partition: Training-and-testing
  - use two independent data sets, e.g., training set (2/3), test set(1/3)
  - used for data set with large number of samples
- Cross-validation
  - divide the data set into *k* subsamples
  - use *k-1* subsamples as training data and one sub-sample as test data --- *k*-fold cross-validation
  - for data set with moderate size

Classification Accuracy:
Estimating Error Rates

- k-fold cross-validation

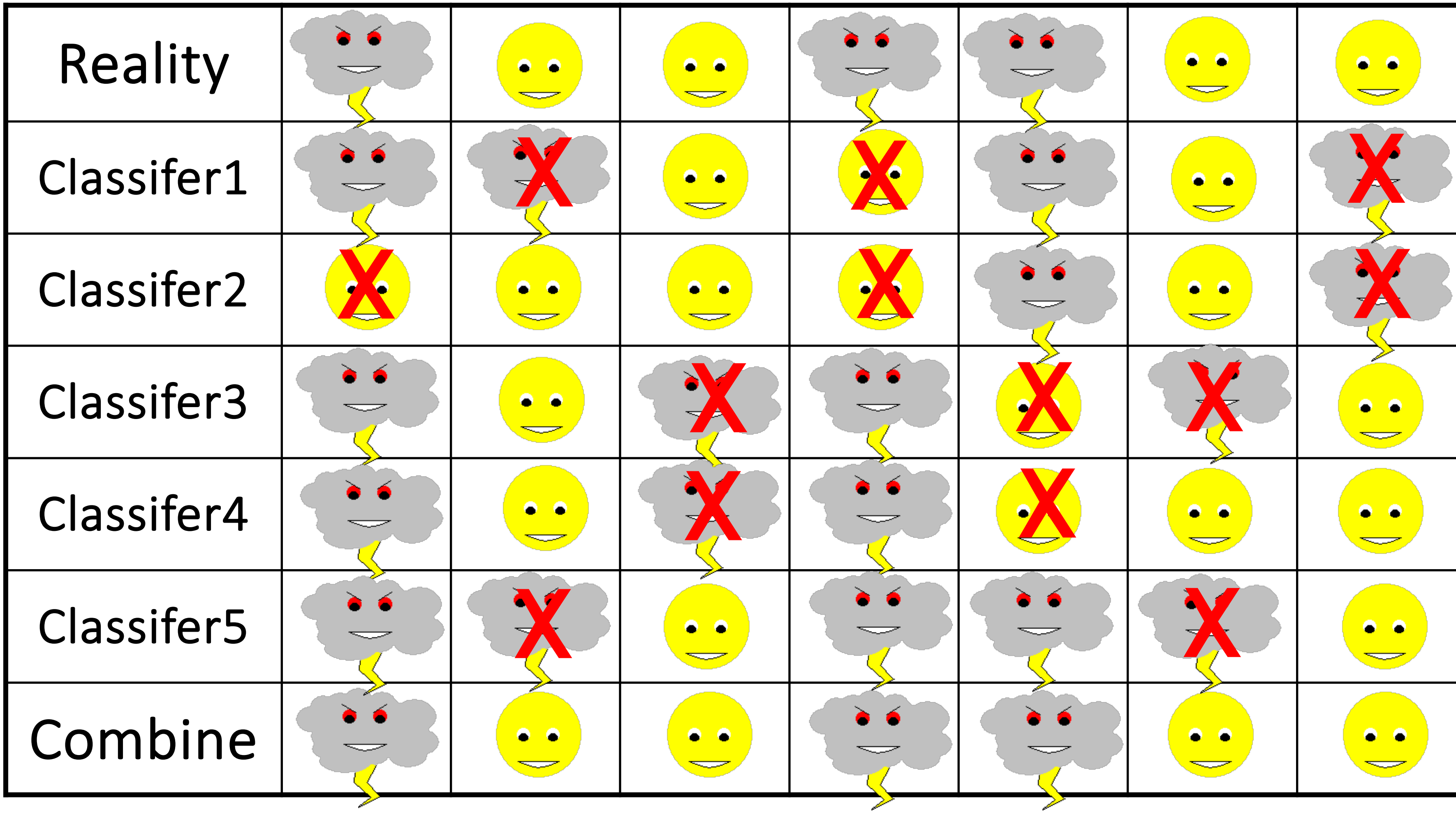
- Leave-one-out (n-fold cross validation)



Ensemble Methods

Collective wisdom

An illustrative example: Weather forecasting



Collective wisdom ——> better performance!!

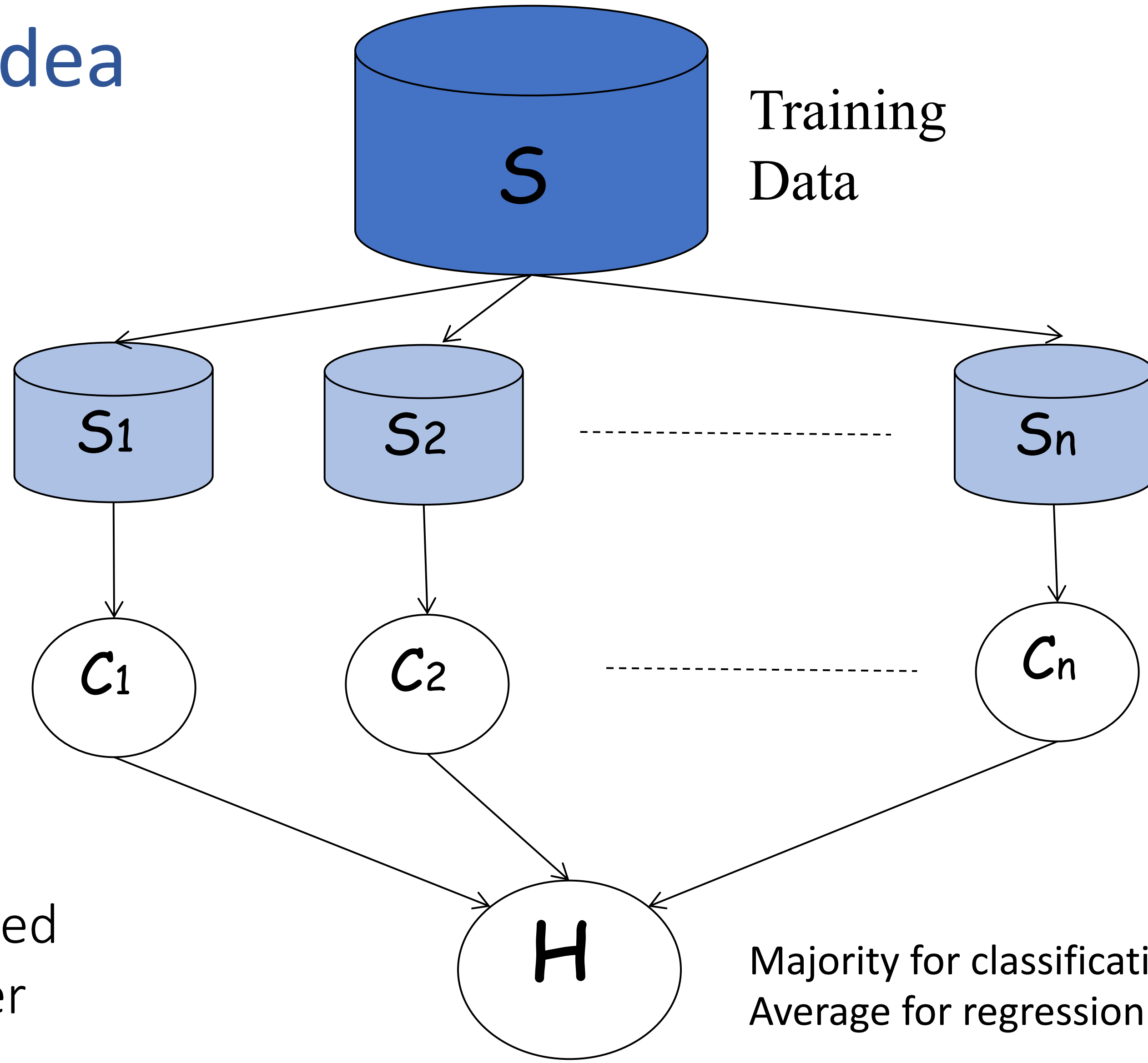
Collective Wisdom: General Idea

Training Data

Multiple Data Sets

Multiple Classifiers

Combined Classifier



Majority for classification
Average for regression

Why does it work?

- Suppose there are 25 base classifiers
- Each classifier has error rate $\varepsilon = 0.35$
- Assume independence among classifiers
- Probability that the ensemble classifier makes a wrong prediction:

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

Building Ensemble Classifiers

- Basic idea:
  Build different "experts", and let them vote
- Advantages:
  Improve predictive performance
  Easy to implement
  No too much *parameter tuning*
- Disadvantages:
  The combined classifier is not so transparent (black box; low interpretability)
  Not a compact representation
- Two approaches: Bagging (Bootstrap Aggregating) and Boosting

Bagging Ensemble Methods:
Random Forest (RF)

- A single decision tree does not perform well
- But, it is super fast
- What if we learn multiple trees?

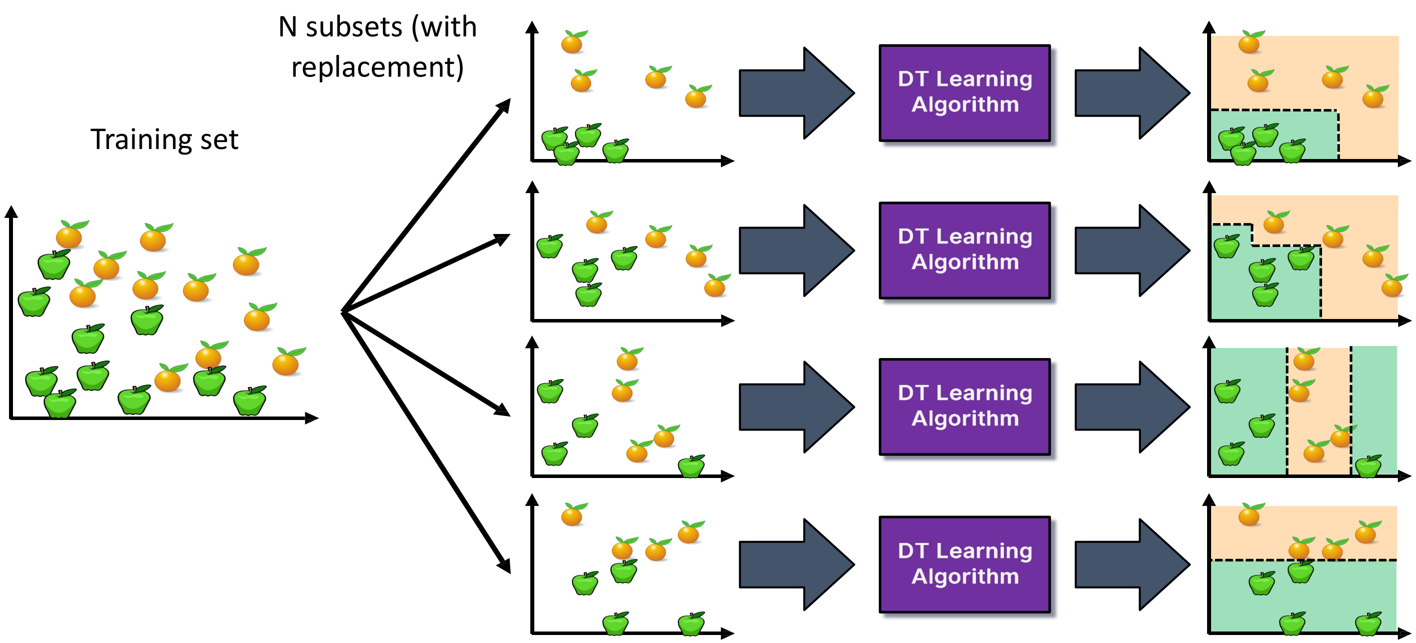
We need to make sure they do not all just learn the same

What is a Random Forest?

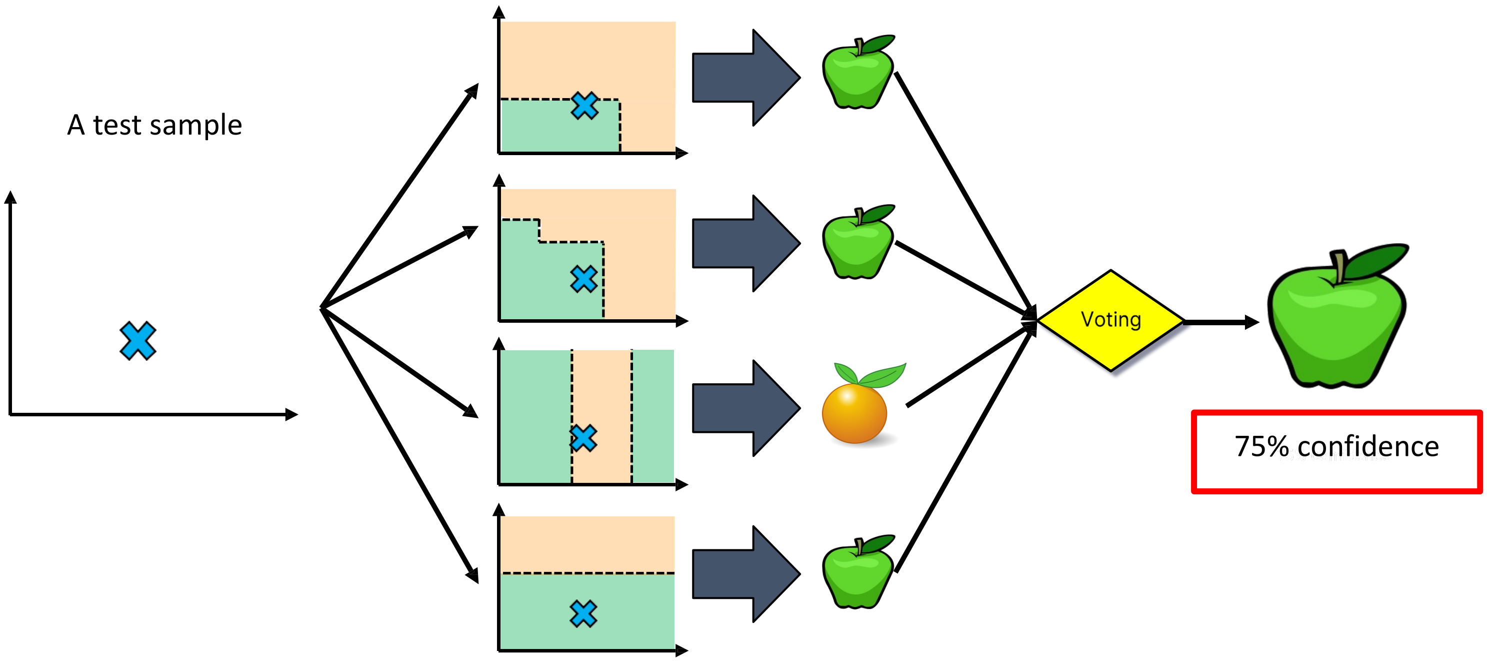
Definition 1.1. A random forest is a classifier consisting of a collection of tree-structured classifiers $\{h(\mathbf{x}, \Theta_k),\ k = 1, \ldots\}$ where the $\{\Theta_k\}$ are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input $\mathbf{x}$.

- Suppose we have a single data set, so how do we obtain slightly different trees?
  1. Basic Bagging:
     - Take random subsets of data points from the training set to create N smaller data sets
     - Fit a decision tree on each subset
  2. Feature Bagging:
     - Fit N different decision trees by constraining each one to operate on a random subset of features

Basic bagging at training time (model construction)



Basic bagging at inference time (model usage)



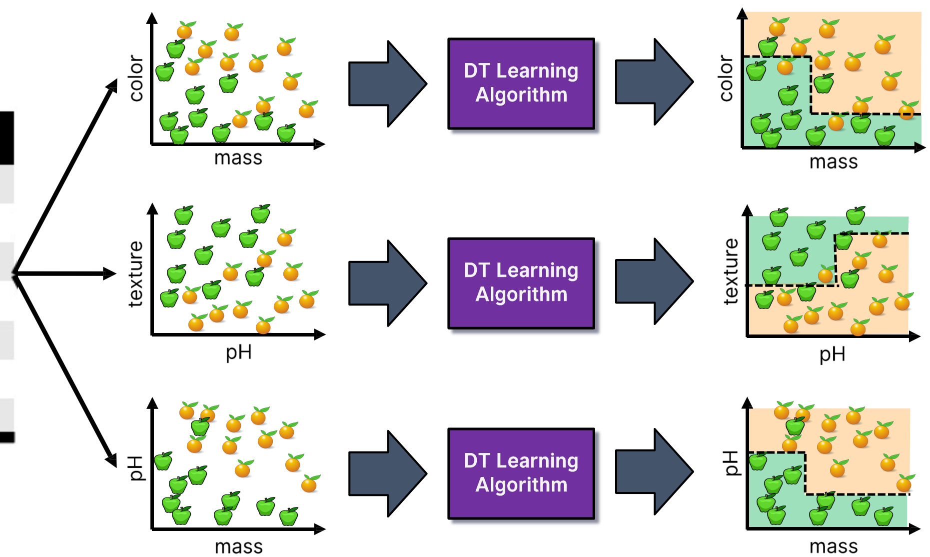
Feature bagging at training time (model construction)

Training data

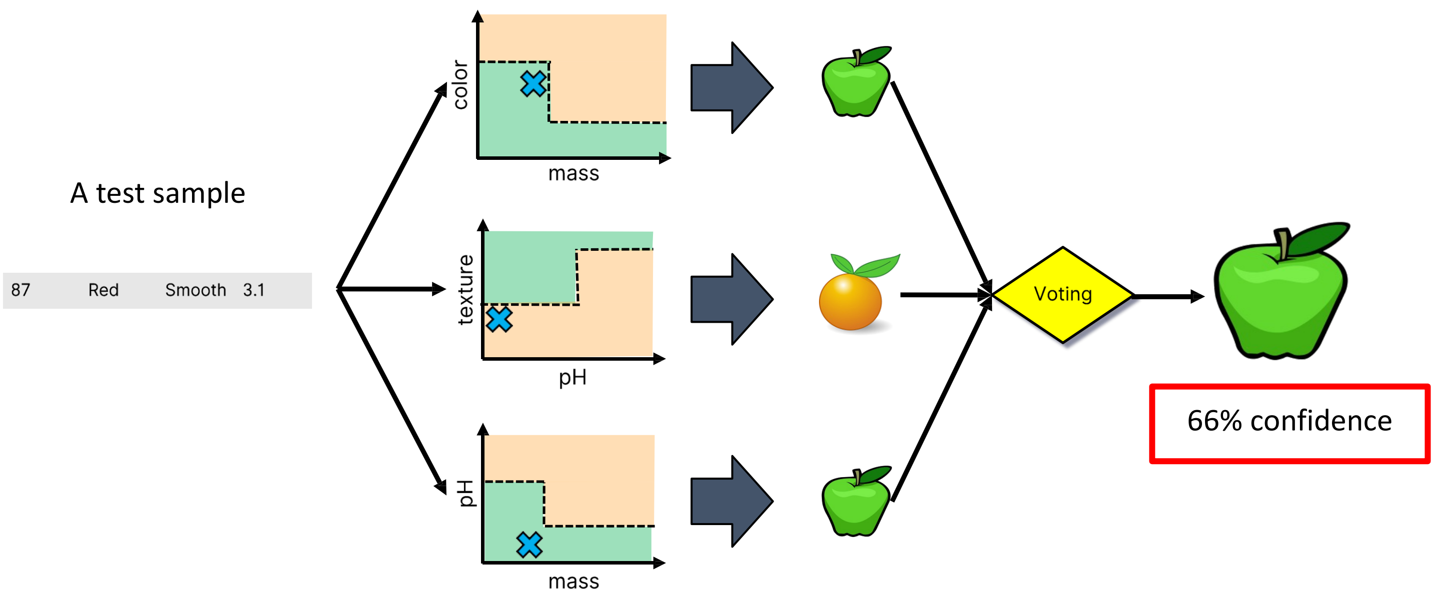
Mass (g)	Color	Texture	pH	Label
84	Green	Smooth	3.5	**Apple**
121	Orange	Rough	3.9	**Orange**
85	Red	Smooth	3.3	**Apple**
101	Orange	Smooth	3.7	**Orange**
111	Green	Rough	3.5	**Apple**
...				
117	Red	Rough	3.4	**Orange**

4-D Data

(4 features: Mass, Color, Texture, pH)



Feature bagging at inference time (model usage)



Random Forests

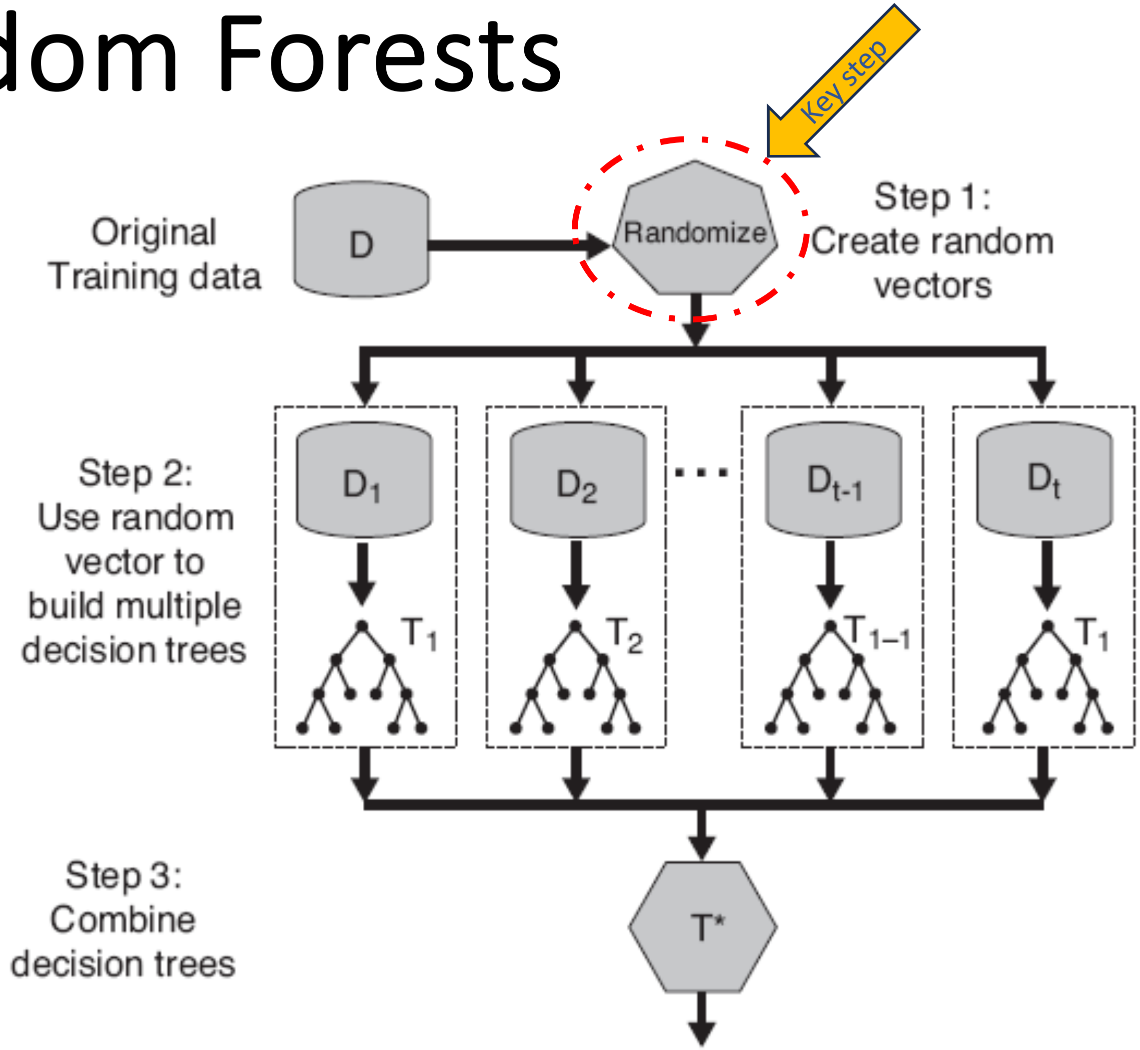


Figure 5.40. Random forests.

Random Forests

Random forests are popular. Leo Breiman's and Adele Cutler maintains a random forest website where the software is freely available, and of course it is included in every ML/STAT package

http://www.stat.berkeley.edu/~breiman/RandomForests/

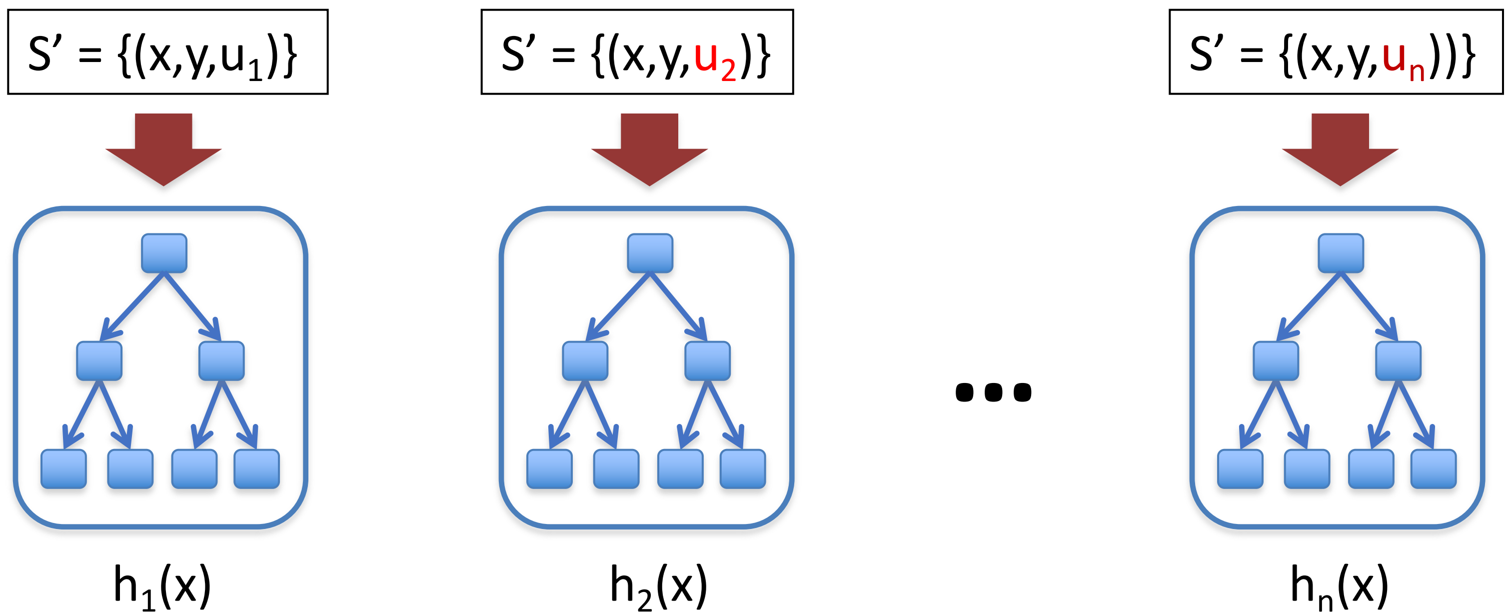
Boosting Ensemble Methods: Adaboost

Most boosting algorithms follow these steps:

1. Train a weak model on some training data
2. Compute the error of the model on each training example
3. Give higher importance to examples on which the model made mistakes
4. Re-train the model using “importance weighted” training examples
5. Go back to step 2

Boosting (AdaBoost)

$$h(x) = a_1 h_1(x) + a_2 h_2(x) + \ldots + a_n h_n(x)$$



u – weighting on data points
a – weight of linear combination

Stop when validation performance plateaus

https://towardsdatascience.com/a-comprehensive-mathematical-approach-to-understand-adaboost-f185104edced

AdaBoost (Adaptive Boosting): Algorithm

- Given: Training data $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$ with $y_n \in \{-1, +1\}$, $\forall n$
- Initialize weight of each example $(\boldsymbol{x}_n, y_n)$: $D_1(n) = 1/N$, $\forall n$
- For round $t = 1 : T$
  - Learn a weak $h_t(\boldsymbol{x}) \rightarrow \{-1, +1\}$ using training data weighted as per D_t
  - Compute the weighted fraction of errors of h_t on this training data

$$\epsilon_t = \sum_{n=1}^{N} D_t(n) \mathbb{1}[h_t(\boldsymbol{x}_n) \neq y_n]$$

  - Set "importance" of h_t: $\alpha_t = \frac{1}{2} \log(\frac{1-\epsilon_t}{\epsilon_t})$(gets larger as ϵ_t gets smaller)
  - Update the weight of each example

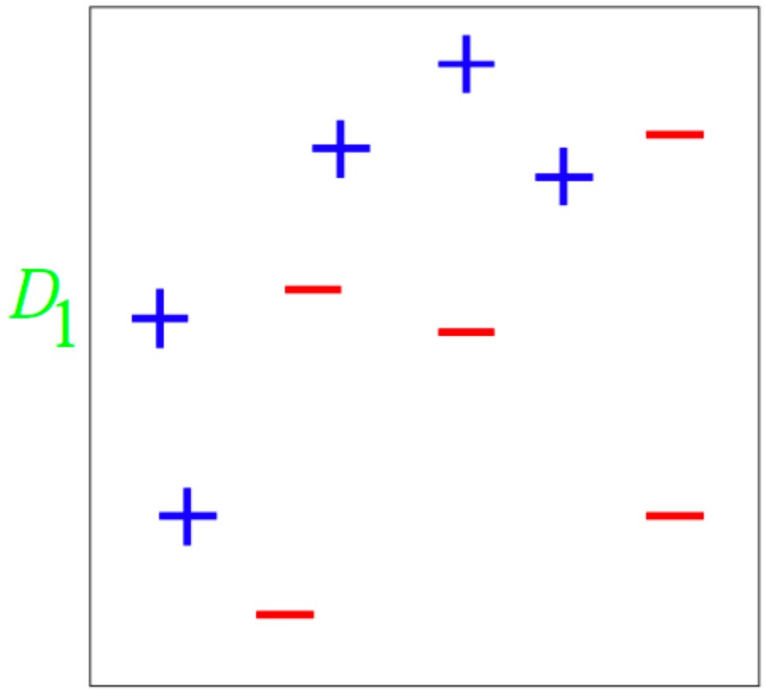
$$\begin{aligned} D_{t+1}(n) &\propto \begin{cases} D_t(n) \times \exp(-\alpha_t) & \text{if } h_t(\boldsymbol{x}_n) = y_n \quad \text{(correct prediction: decrease weight)} \\ D_t(n) \times \exp(\alpha_t) & \text{if } h_t(\boldsymbol{x}_n) \neq y_n \quad \text{(incorrect prediction: increase weight)} \end{cases} \\ &= D_t(n) \exp(-\alpha_t y_n h_t(\boldsymbol{x}_n)) \end{aligned}$$

  - Normalize D_{t+1} so that it sums to 1: $D_{t+1}(n) = \frac{D_{t+1}(n)}{\sum_{m=1}^{N} D_{t+1}(m)}$
- Output the "boosted" final hypothesis $H(\boldsymbol{x}) = \text{sign}(\sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x}))$

AdaBoost: An Example

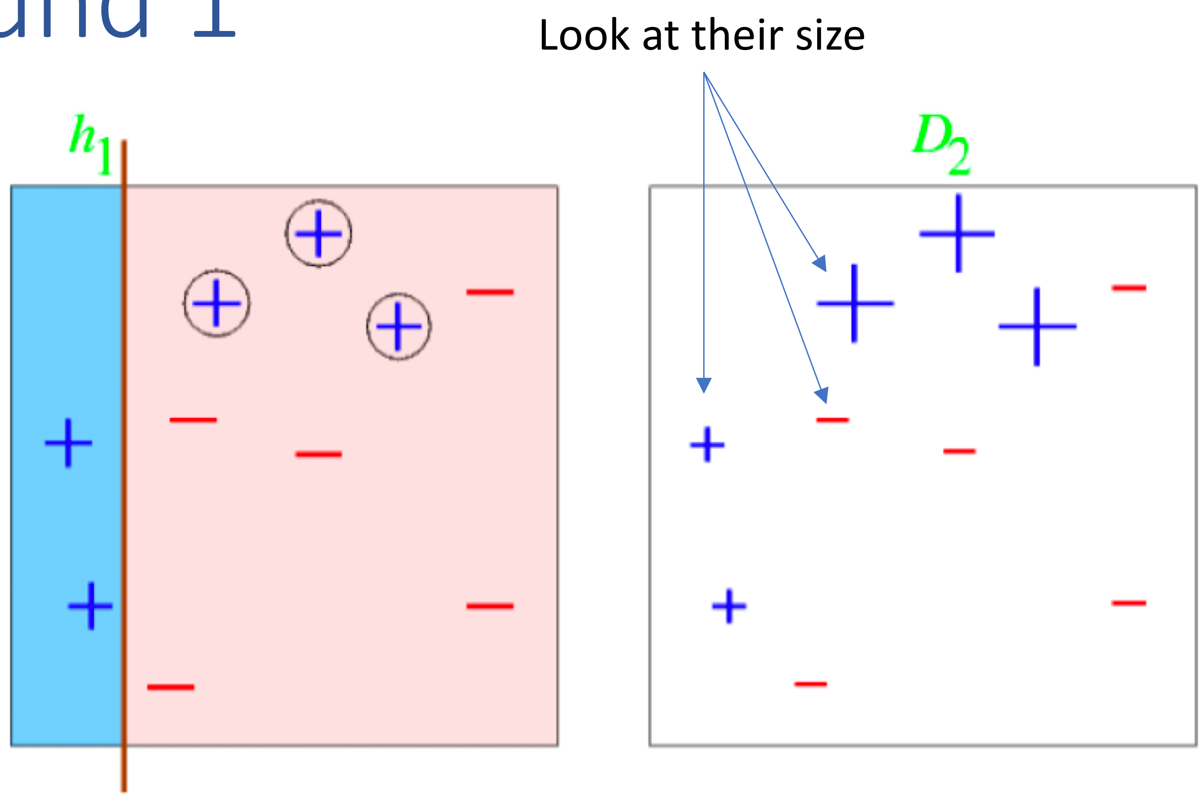
Consider binary classification with 10 training examples

Initial weight distribution D_1 is uniform (each point has equal weight $= 1/10$)



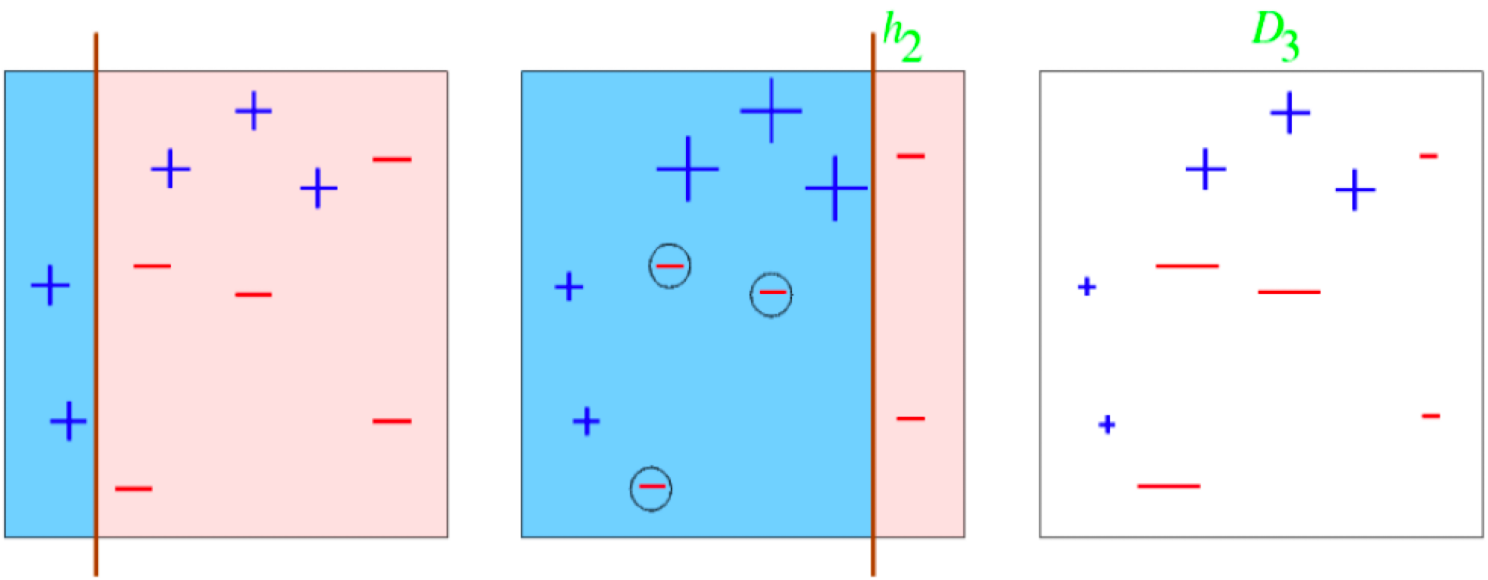
Each of our weak classifers will be an **axis-parallel linear classifier**

After Round 1



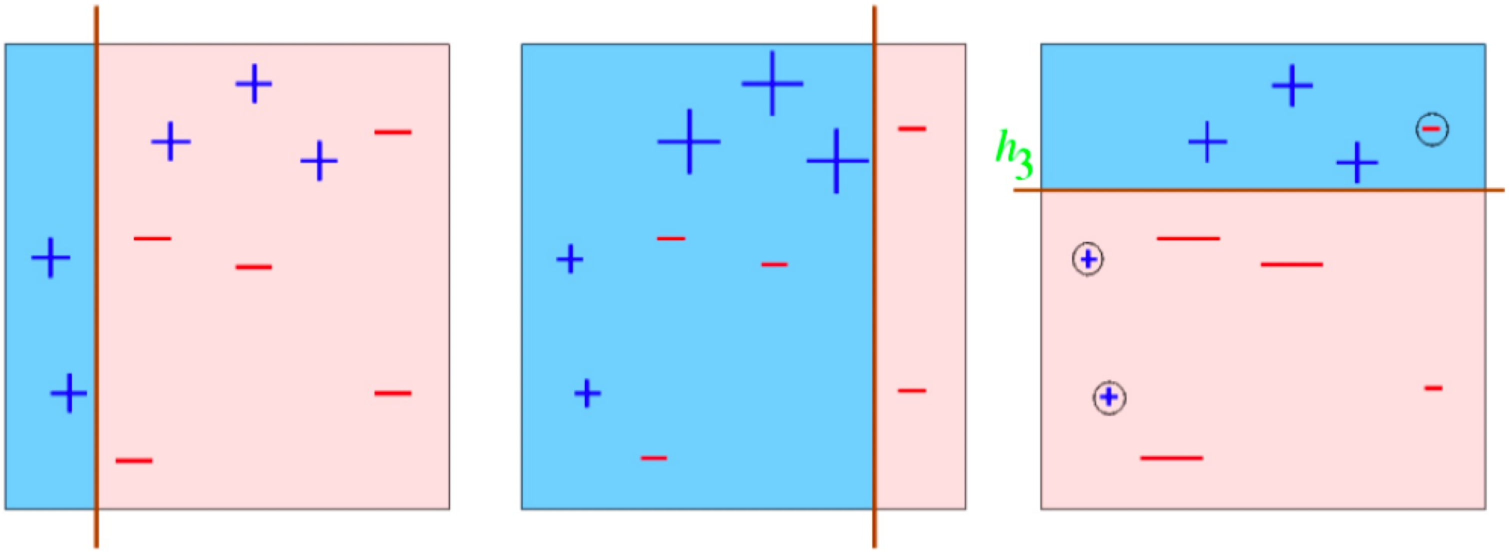
- Error rate of h_1: $\epsilon_1 = 0.3$; weight of h_1: $\alpha_1 = \frac{1}{2}\ln((1-\epsilon_1)/\epsilon_1) = 0.42$
- Each misclassified point upweighted (weight multiplied by $\exp(\alpha_2)$)
- Each correctly classified point downweighted (weight multiplied by $\exp(-\alpha_2)$)

After Round 2



- Error rate of h_2: $\epsilon_2 = 0.21$; weight of h_2: $\alpha_2 = \frac{1}{2}\ln((1-\epsilon_2)/\epsilon_2) = 0.65$
- Each misclassified point upweighted (weight multiplied by $\exp(\alpha_2)$)
- Each correctly classified point downweighted (weight multiplied by $\exp(-\alpha_2)$)

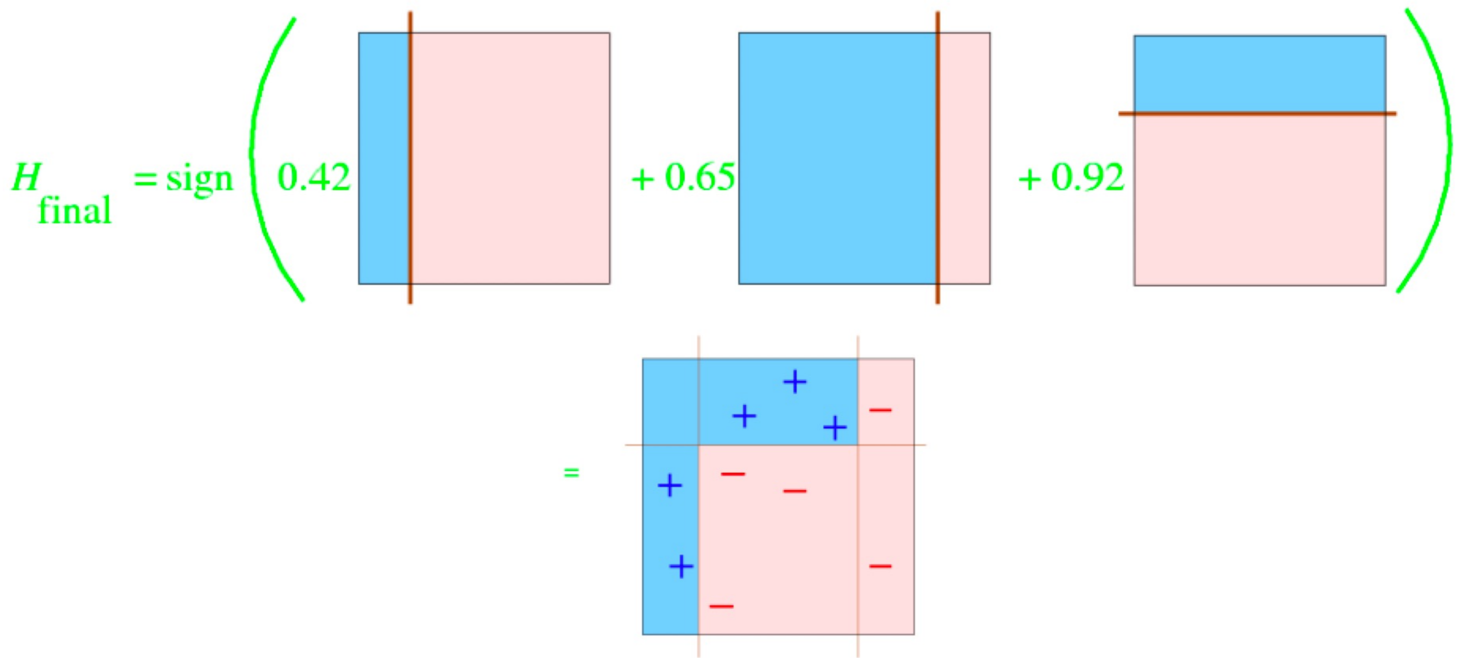
After Round 3



- Error rate of h_3: $\epsilon_3 = 0.14$; weight of h_3: $\alpha_3 = \frac{1}{2}\ln((1-\epsilon_3)/\epsilon_3) = 0.92$
- Suppose we decide to stop after round 3
- Our ensemble now consists of 3 classifiers: h_1, h_2, h_3

Finally, we have

- Final classifier is a weighted linear combination of all the classifiers
- Classifier h_i gets a weight α_i



- Multiple **weak, linear classifiers combined** to give a strong, nonlinear classifier

A bit more … XGBoost

Why use XGBoost?

- All of the advantages of gradient boosting (a kind of learning tree ensembles), plus more.
- Frequent Kaggle data competition champion.
- Utilizes CPU Parallel Processing by default.
- Two main reasons for use:
  1. Low Runtime
  2. High Model Performance

Remarks

- Competitor-LightGBM (Microsoft)
  - Very similar, not as mature and feature rich
  - Slightly faster than XGBoost – much faster when it was published
- Hardly find an example where Random Forest outperforms XGBoost
- XGBoost Github: https://github.com/dmlc/xgboost
- XGBoost documentation: http://xgboost.readthedocs.io

Take-home Messages

- Classification is an extensively studied problem (mainly in statistics, machine learning & neural networks)
- Scalability is still an important issue for database applications: thus combining classification with database techniques should be a promising topic
- Research directions: classification of non-relational data, e.g., text, spatial, multimedia, etc..
- Ensemble methods are effective approaches to boost the performance of ML/DM system.
- One particular advantage is that they typically do not involve too many parameters and too involved tuning.

Acknowledgement

- Slides/Materials of
  - Yisong Yue’s Presentation at University of Luebeck
  - P. Rai, IIT
  - Zhuowen Tu, UCLA
- Photos from Internet