SHIRAZ UNIVERSITY
Computer Science and Engineering Department
Machine Learning Lab

Bagging & Boosting

Dr Sattar Hashemi

Based on:

Christopher M. Bishop, *Pattern recognition and machine learning*.----- 14.3

Dietterich, Thomas G. "Ensemble methods in machine learning." In Multiple classifier systems, pp. 1-15. Springer Berlin Heidelberg, 2000

Breiman, Leo. "Bagging predictors." Machine learning 24, no. 2 (1996): 123-140.

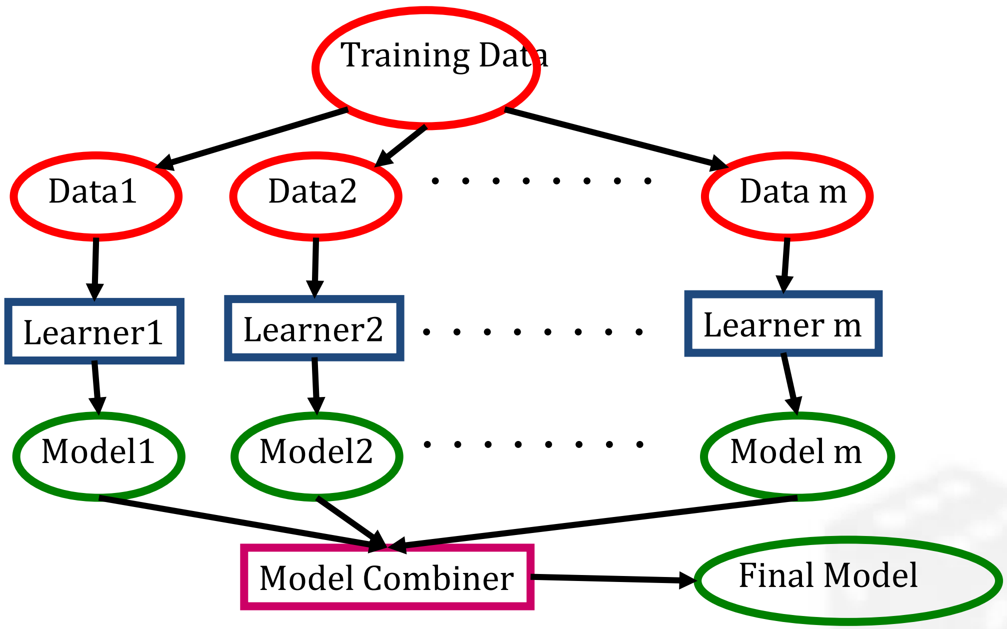
Freund, Yoav, and Robert E. Schapire. "A decision-theoretic generalization of on-line learning and an application to boosting." Journal of computer and system sciences 55, no. 1 (1997): 119-139.

Introduction

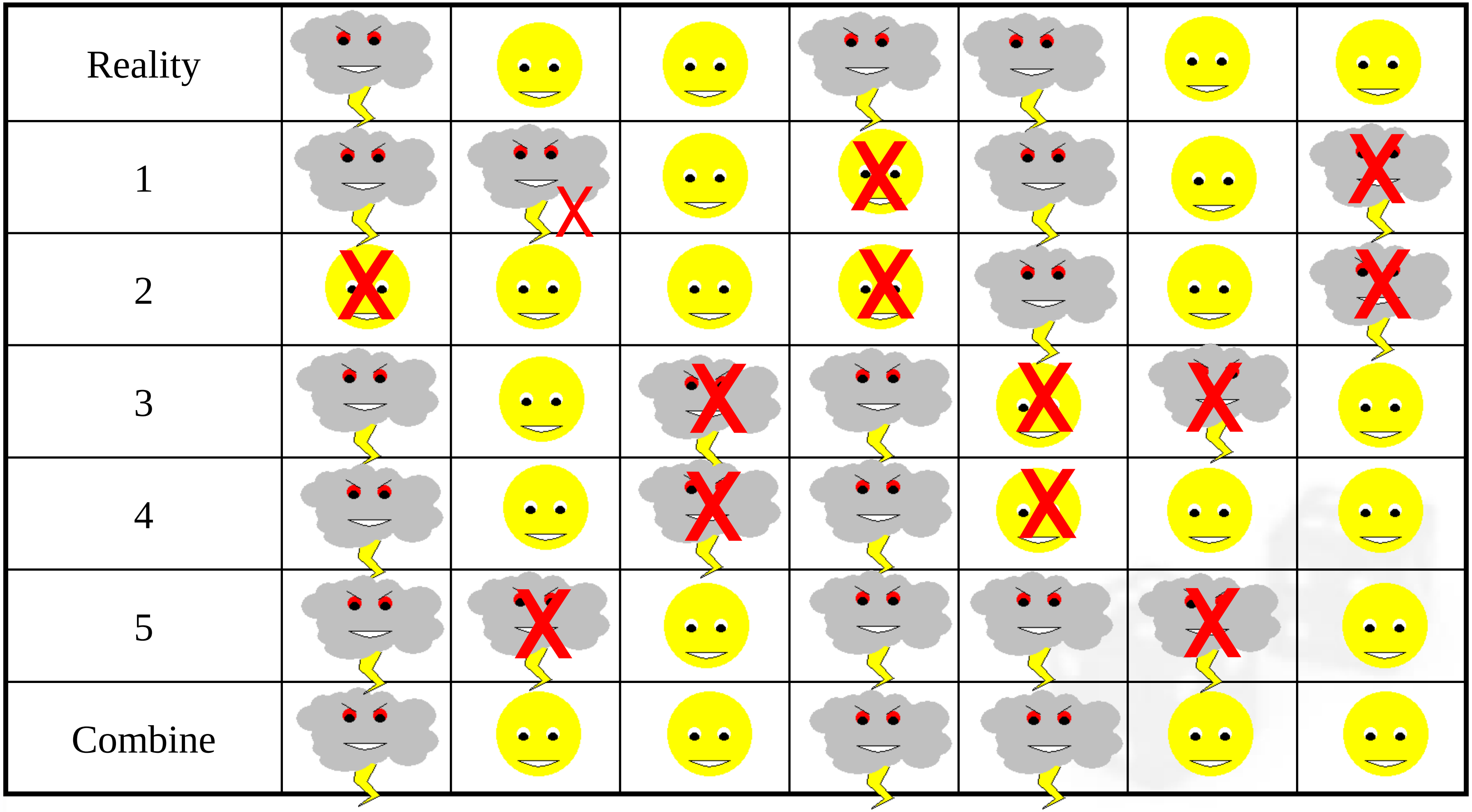
- **What is ensemble learning?**
  - So far – learning methods that learn a single hypothesis, chosen form a hypothesis space that is used to make predictions.
  - Ensemble learning: select a collection (ensemble) of hypotheses and combine their predictions
- **Why ensemble learning?**
  - Accuracy: a more reliable mapping can be obtained by combining the output of multiple "experts".
  - Efficiency: a complex problem can be decomposed into multiple sub-problems that are easier to understand and solve (divide-and-conquer approach).
- **When ensemble learning?**
  - When you can build component classifiers that are **more accurate than chance** and, more importantly, that are **independent** from each other.

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.



Example: Weather Forecast



Why do ensembles work?

- A necessary and sufficent condition for an ensemble of classifier to be more accurate that any of its individual members is if the classifiers are **accurate** and **diverse***

- An **accurate** classifier is one has an error rate of better than random guessing on new x values.

- Two classifiers are **diverse** if they make different errors on new data points.

- Why these two factors are important?

*Hansen, Lars Kai, and Peter Salamon. "Neural network ensembles." *IEEE transactions on pattern analysis and machine intelligence* 12 (1990): 993-1001.

Why do ensembles work?

- Why these two factors are important?
  - we have an ensemble of three classifiers: $\{h_1, h_2, h_3\}$ and consider a new case x. If the three classifiers are identical (i.e., not diverse), then when $h_1(x)$is wrong , $h_2(x)$ and $h_3(x)$ will also be wrong.
  - However, if the **errors** made by the classifiers are uncorrelated, then when $h_1(x)$ is wrong, $h_2(x)$ and $h_3(x)$ may be correct, so that a majority vote will correctly classify x.

- More formal example:
  - Suppose we have T different classifiers and error rate of all of them are equal to ε and if the errors are independent. So the probability that the majority vote will be wrong is equal to :

$$\sum_{i=\left\lceil\frac{T+1}{2}\right\rceil}^{T} \binom{T}{i} \varepsilon^i (1-\varepsilon)^{T-i}$$

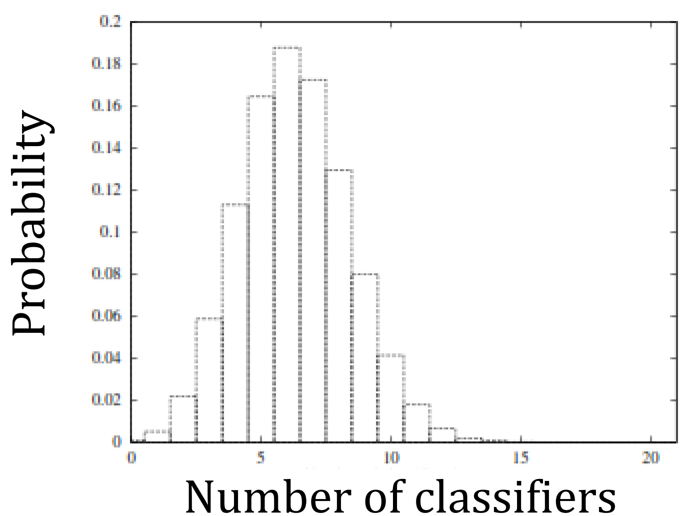
  - Consider two cases: $\varepsilon < \frac{1}{2}$ and $\varepsilon > \frac{1}{2}$

Why do ensembles work?

- Assume the error rate $\varepsilon = 0.3$ and $T = 21$
- 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 below shows the distribution of the number of classifiers that are in error in the ensemble machine.

$$\sum_{i=11}^{21} \binom{21}{i} \varepsilon^i (1-\varepsilon)^{21-i} = 0.026 \ll 0.3$$

- Now suppose $\varepsilon = 0.7$ and $T = 21$, What is the probability of error for this ensemble?



Overfitting

- Main advantage of ensemble is its generalization. Actually ensemble is created first to prevent of overfitting by reasonable computation.

- Suppose the error of classifier(ϵ) in ensemble is independent. This error follows an unknown distribution.

- If the variance of ϵ is v then variance of $\bar{\epsilon}$ is $\frac{v}{T}$

- As you see the variance of ensemble error is lower than the variance of each classifier alone.

But..

- Neither examples is real case.

- Because the error of classifier is not independent and accuracy and diversity is correlated phenomena.

- The real analysis for ensembles is more complicated that we don't talk about it here.

- There is not rigor definition for diversity and its role in classification yet.*,**

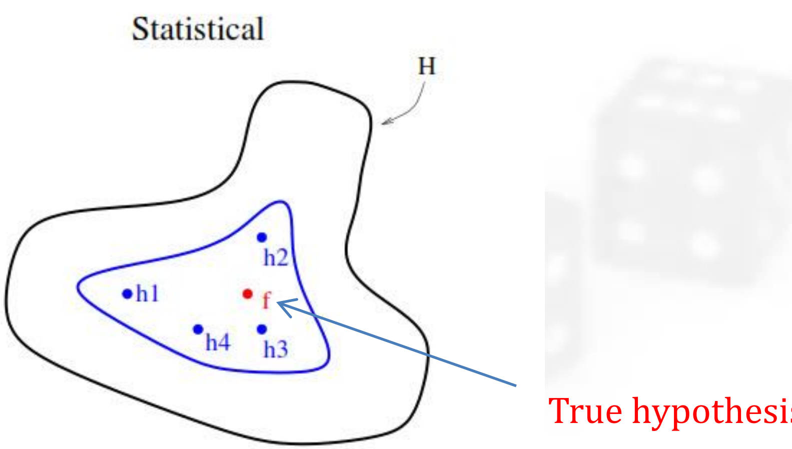
*Brown, Gavin, Jeremy L. Wyatt, and Peter Tiňo. "Managing diversity in regression ensembles." *The Journal of Machine Learning Research* 6- 2005

**Didaci, Luca, Giorgio Fumera, and Fabio Roli. "Diversity in classifier ensembles: fertile concept or dead end?." *Multiple Classifier Systems*. Springer Berlin Heidelberg, 2013. 37-48.

Statistical Problem: S.S.S.P

- A learning algorithm can viewed as searching a space H of hypotheses to identify best hypothesis in the space.
- **The Statistical Problem** arises when the amount of training data available is too small compared o the size of the hypothesis space.(small sample size problem) Without sufficient data, the learning algorithm can find many different hypothesis in H that all give the same accuracy on the training data.

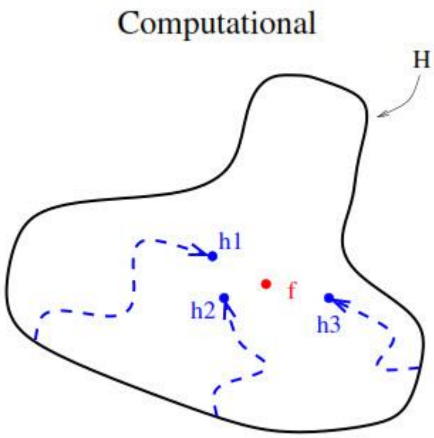
- By constructing an ensemble out of all of these accurate classifiers, the algorithm can average their votes and reduce the risk of choosing the wrong classifier.



Computational Problem

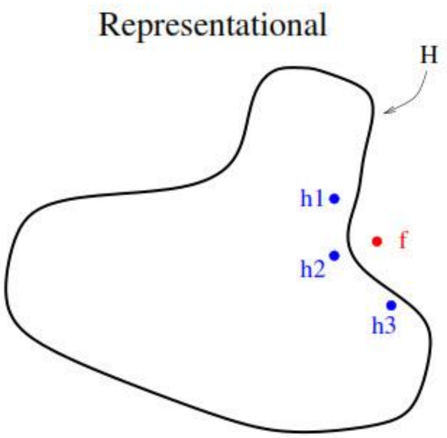
- Suppose there we have enough training data, so that the statistical problem is absent. It may be difficult computationally for the learning algorithm to find the best hypothesis. For example optimal training of Neural Network is NP-Hard.

- **The Computational Problem** arises when the learning algorithm cannot guarantee finding the best hypothesis. An ensemble constructed by running the local search from many different starting points may provide a better approximation of the true unknown function



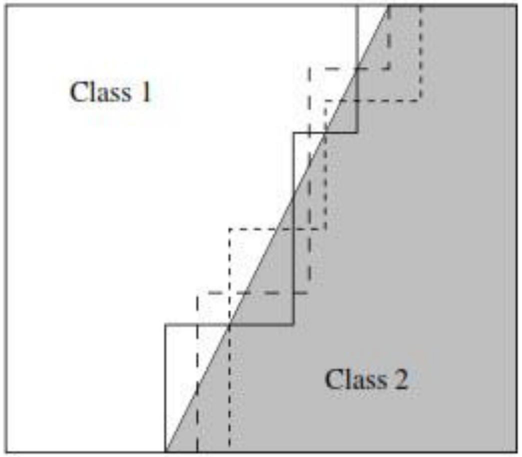
Representational Problem

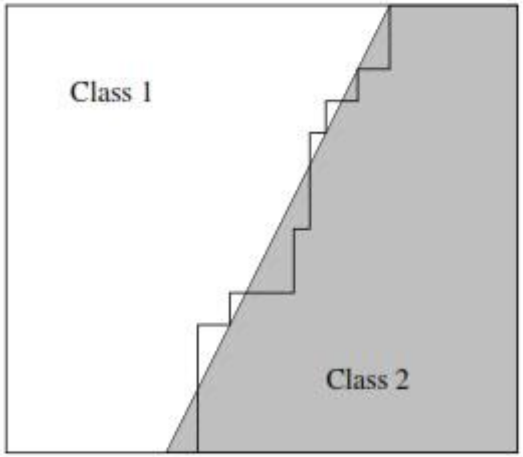
- **The Representational Problem** arises when the hypothesis space does not contain any good approximation of the target class(es). By forming weighted sums of hypotheses drawn from *H*, it may possible to expand the space of representable functions.



Example of representational problem

- If the true decision boundaries are not orthogonal to the coordinate axes, then C4.5 requires infinite size to represent those boundaries correctly.
- Decision boundaries constricted by decision trees →hyperplanes parallel to the coordinate axis – "staircases".
- By averaging a large number of "staircases" the diagonal boundary can be approximated with some accuracy.





Diversification of classifiers

Different training sets (different samples or splitting,...)

- Different classifiers (trained for the same data)
- Different attributes sets (e.g., identification of speech or images)
- Different parameter choices (e.g., amount of tree pruning, BP parameters, number of neighbors in KNN,…)
- Different architectures (like topology of ANN)
- Different initializations

How to measure diversity?

- Pairwise Measures
  - The Q statistics
  - The correlation coefficient
  - The Disagreement Measure
  - The Double-Fault Measure

- Non-pairwise Measures
  - The Entropy Measure
  - Kohavi–Wolpert Variance
  - Measurement of Interrater Agreement
  - The Measure of difficulty
  - Generalized Diversity
  - Coincident failure diversity

The Q-statistics

A 2×2 table of the relationship between a pair of classifiers.

	D_k correct (1)	D_k wrong (0)
D_i correct (1)	N^{11}	N^{10}
D_i wrong (0)	N^{01}	N^{00}
$Total, N = N^{00} + N^{10} + N^{01} + N^{11}$		

Yule's Q-statistic (1900) for independency check of two classifier is:

$$Q_{i,k} = \frac{N^{11}N^{00} - N^{01}N^{10}}{N^{11}N^{00} + N^{01}N^{10}}$$

For an ensemble of T classifier , we calculate the averaged Q-statistics:

$$Q_{av} = \frac{2}{T(T-1)} \sum_{i=1}^{T-1} \sum_{k=i+1}^{T} Q_{i,k}$$

Different kind of Ensembles

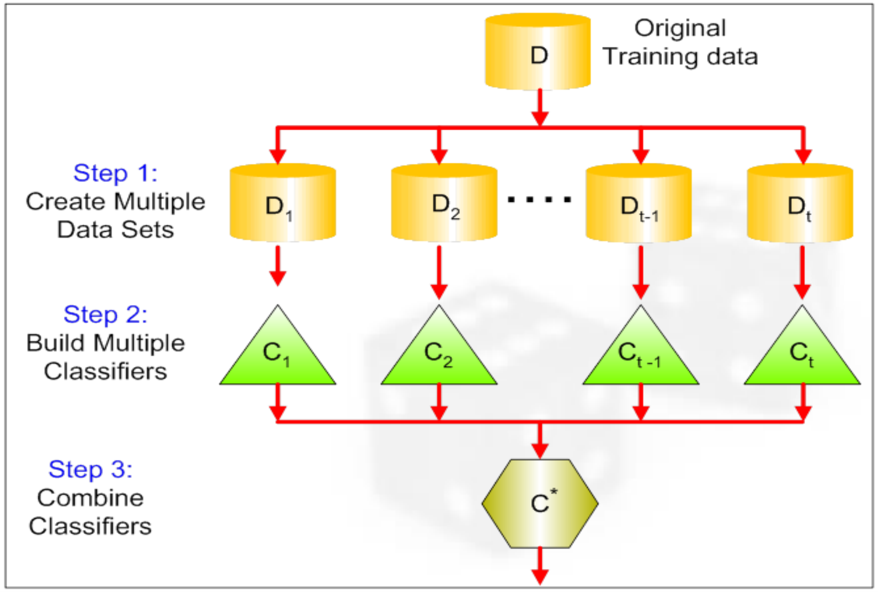
- Bagging
  - Resample training data
- Boosting
  - Reweight training data
- Voting:
  - For binary classification problems
- Averaging:
  - For regression problems
- Stacking
  - Combine the prediction of several other learning algorithm

Bagging [L. Breiman, 1996]

Bagging = **B**ootstrap **agg**regat**ing**

- Generates individual classifiers on bootstrap samples of the training set.

- Suppose original data has M member. By uniform distribution, M data is sampled from original training data.
- As a result of the sampling-with-replacement procedure, each classifier is trained on the average of 63.2% of the training examples.



Bagging [L. Breiman, 1996]

Boot strapping:
Each dataset is consisting of M example is drawn from the original dataset.
What is the probability of each example being in new training set?

the answer is

$$1-\left(1-\frac{1}{M}\right)^{M}$$

For large N this value is near $\left(1-\frac{1}{e}\right)$ or 0.632

More about "Bagging"

Training phase

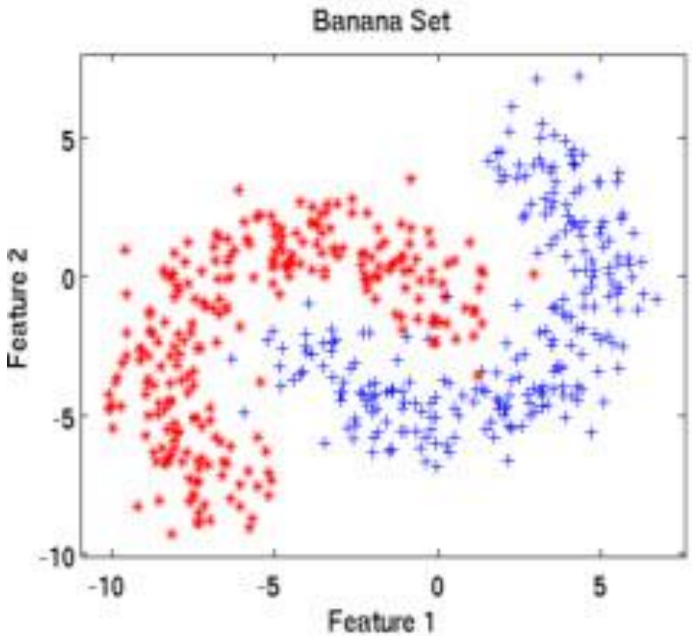
1. Initialize the parameters
   - $\mathcal{D} = \emptyset$, the ensemble.
   - L, the number of classifiers to train.
2. For $k = 1, \ldots, L$
   - Take a bootstrap* sample S_k from **Z.**
   - Build a classifier D_k using S_k as the training set.
   - Add the classifier to the current ensemble, $\mathcal{D} = \mathcal{D} \cup D_k$.
3. Return $\mathcal{D}$.

Classification phase

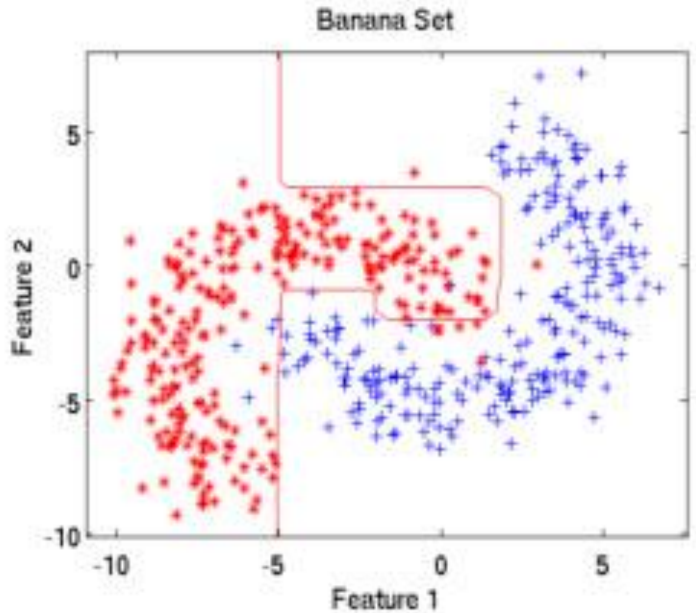
4. Run $\mathcal{D}_1, \ldots, \mathcal{D}_L$ on the input x.
5. The class with the maximum number of votes is chosen as the label for x.

*it allows estimation of the sampling distribution using a very simple method (like resampling)

Example: Bagging decision tree

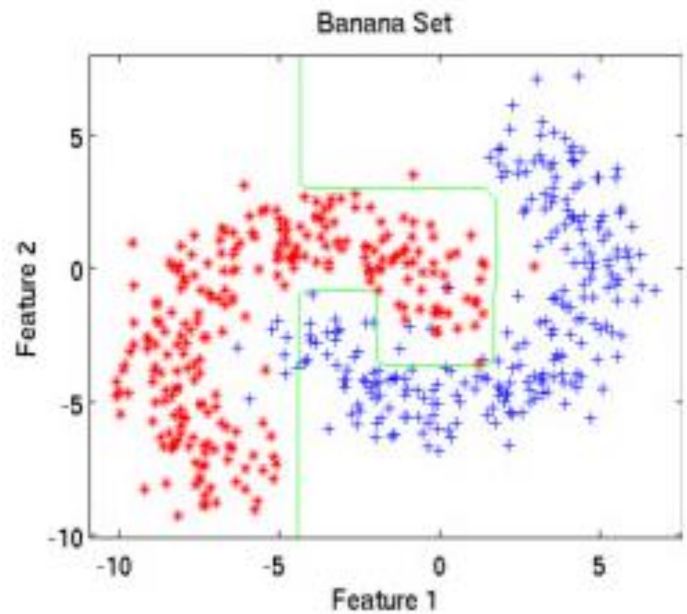


Training data

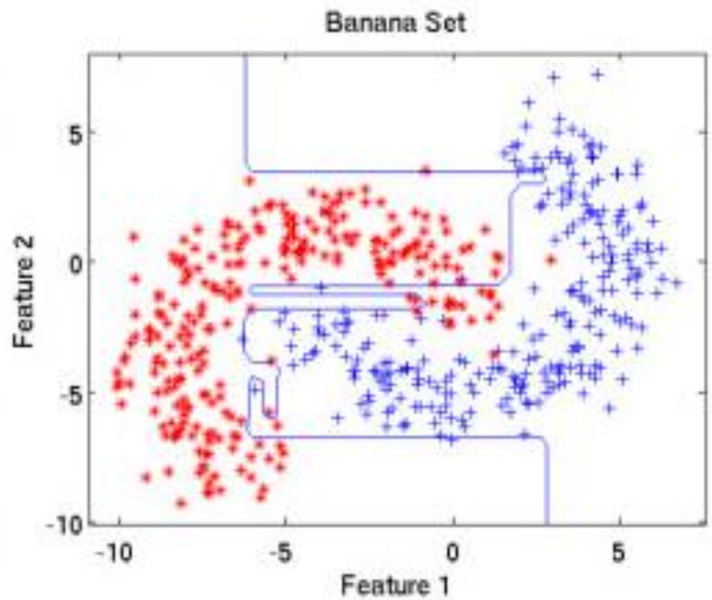


Decision boundary
produced by first tree

Example: Bagging decision tree

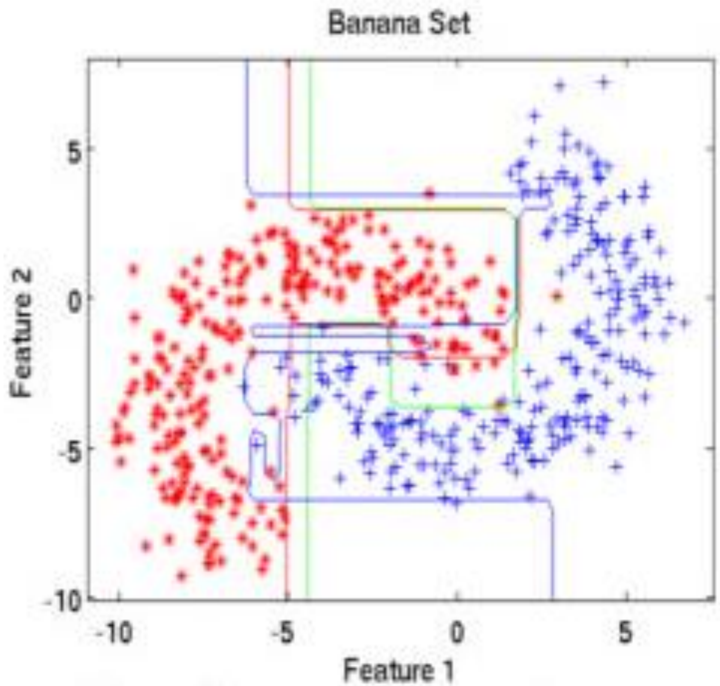


Decision boundary
produced by second tree

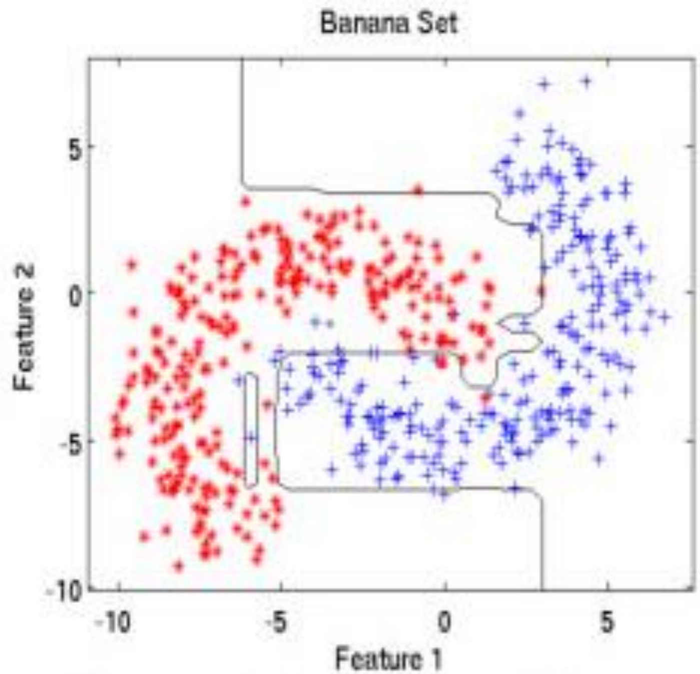


Decision boundary
produced by third tree

Example: Bagging decision tree



Three trees and final boundary overlaid



Final result from bagging all trees

Bagging

- Bagging works because it reduces variance by voting/averaging
  - Usually, the more classifiers the better
- Problem: we only have one dataset.
  - Solution: generate new ones of size n by bootstrapping, i.e. sampling it with replacement
- Can help a lot if data is noisy.
- Averaging over bootstrap samples can reduce error from variance especially in case of **unstable** classifiers.

Bagging variants

- Random Forests
  - A variant of bagging proposed by Breiman
  - It’s a general class of ensemble building methods using a decision tree as base classifier.
- Classifier consisting of a collection of tree-structure classifiers.

Boosting [Schapire 1990; Freund & Schapire 1996]

- In general takes a different weighting schema of resampling than bagging.
- Freund & Schapire: theory for "weak learners" in late 80's
- *Weak Learner: performance on any train set is slightly better than chance prediction.*
  - Schapire has shown that a weak learner can be converted into a strong learner by changing the distribution of training examples
- Iterative procedure:
  - The component classifiers are built sequentially, and examples that are misclassified by previous components are chosen more often than those that are correctly classified!
  - So, new classifiers are influenced by performance of previously built ones. New classifier is encouraged to become expert for instances classified incorrectly by earlier classifier.
- There are several variants of this algorithm – AdaBoost the most popular.

Boosting , AdaBoost

- Start with equally weighted data, apply first classifier.
- Increase weights on misclassified data, apply second classifier.
- Continue emphasizing misclassified data to subsequent classifiers until all classifiers have been trained.

AdaBoost

Given: $(x_1, y_1), \dots (x_m, y_m)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$
Initialize $D_1(i) = \frac{1}{m}$.
For $t = 1, \dots, T$:

Train weak learner using distribution D_t.
Get weak hypothesis $h_t : X \rightarrow \{-1, +1\}$ with error

$$\epsilon_t = \sum_{i=1}^{m} D_t(i) I(h_t(x_i) \neq y_i)$$

Choose $\alpha_t = \frac{1}{2} \ln(\frac{1-\epsilon_t}{\epsilon_t})$

Update

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} \; if \; h_t(x_i) = y_i \\ e^{\alpha_t} \; if \; h_t(x_i) \neq y_i \end{cases}$$

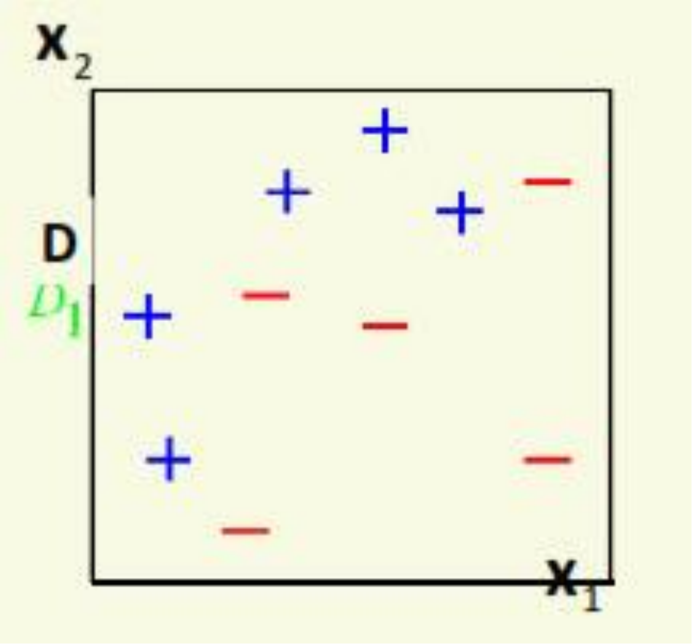
$$= \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

Where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).
Output the final hypothesis:

$$H(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x))$$

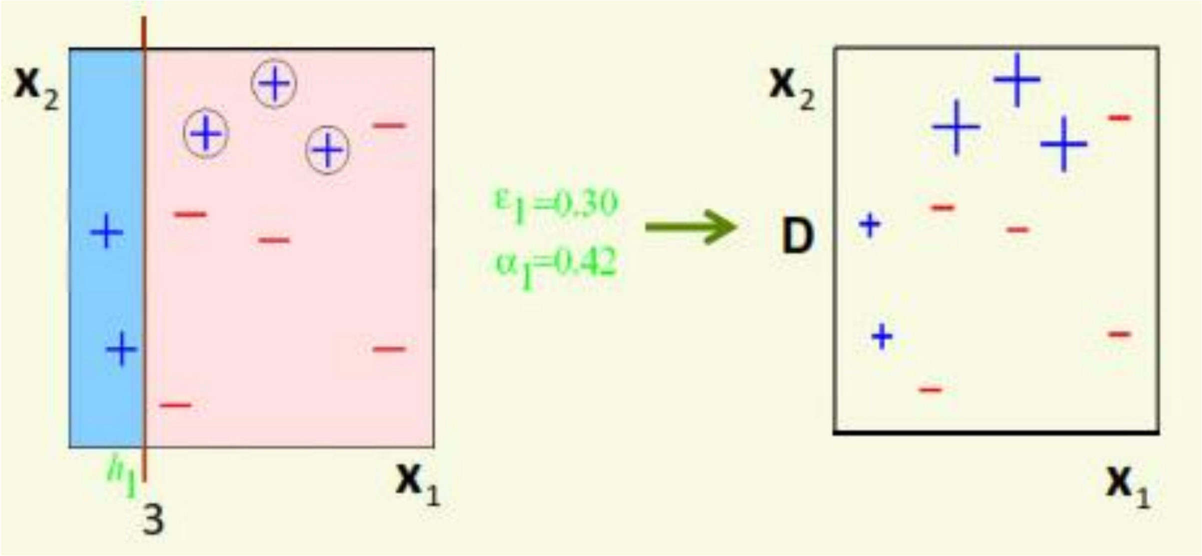
AdaBoost example

Initialization: all examples have equal weights



AdaBoost example

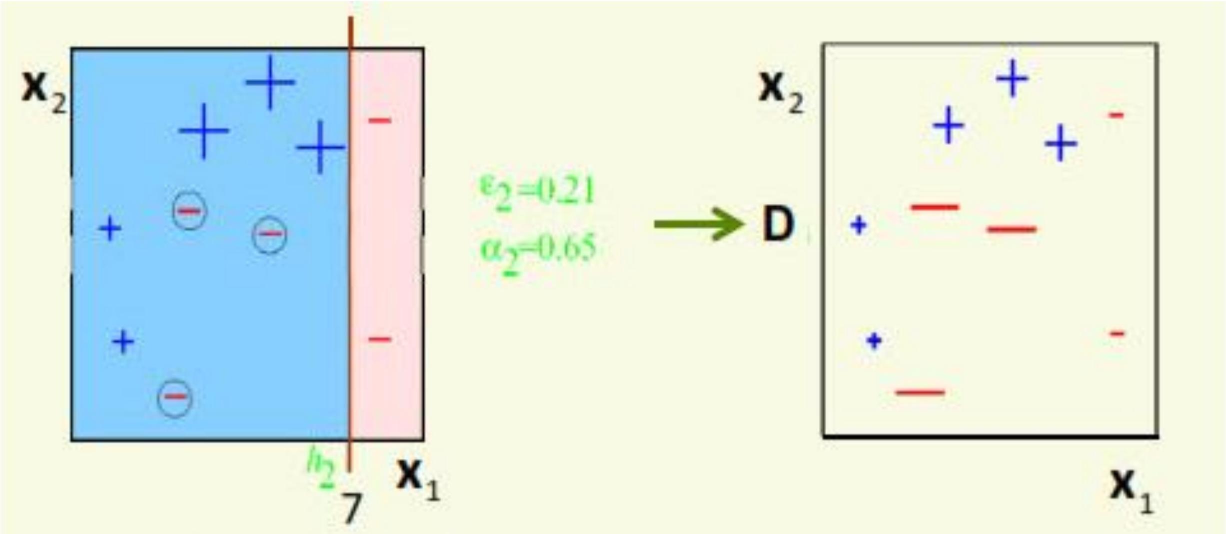
Round 1



$$h_1(x) = sign(3 - x_1)$$

AdaBoost example

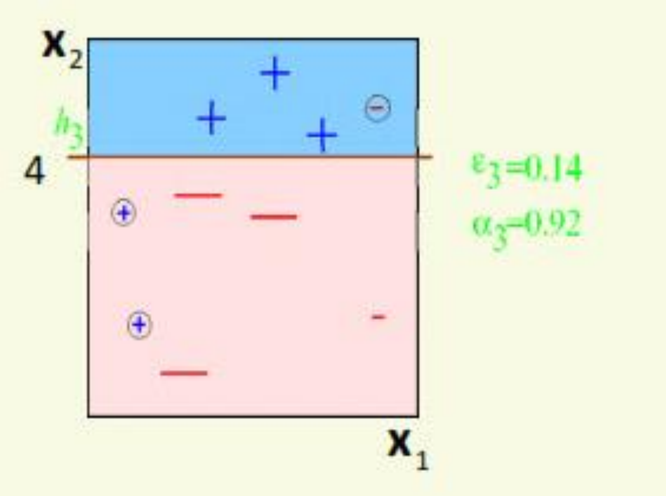
Round 2



$h_2(x) = sign(7 - x_1)$

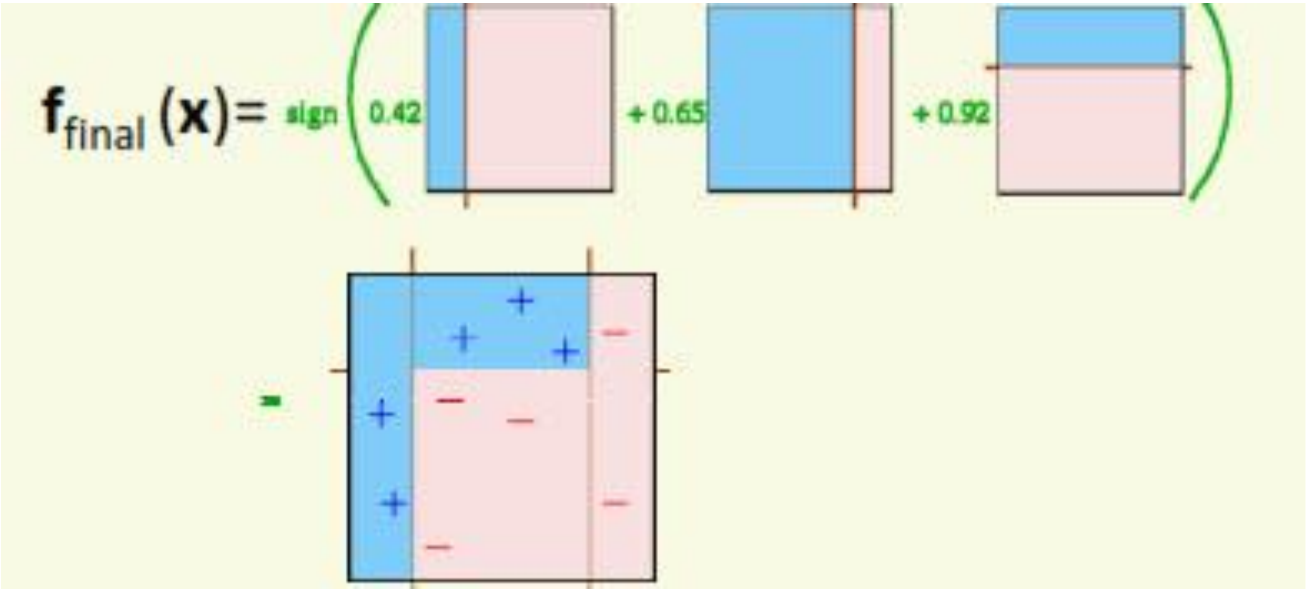
AdaBoost example

Round 3



$$h_3(x) = sign(x_2 - 4)$$

AdaBoost example



$$f_{final}(x) = sign\big(0.42 sign(3 - x_1) + 0.65\, sign(7 - x_1)\big) + 0.92\, sign(x_2 - 4))$$

Note non-linear decision boundary

AdaBoost comments

- We are really interested in the generalization properties of $f_{final}(x)$ not the training error
- AdaBoost was shown to have excellent generalization properties in practice
  - ✓ the more rounds, the more complex is the final classifier, so overfitting is expected as the training proceeds
  - ✓ but in the beginning researchers observed no overfitting of the data
  - ✓ It turns out it does overfit data eventually, if you run it really long

- It can be shown that boosting increases the margins of training examples, as iterations proceed
  - ✓ larger margins help better generalization.
  - ✓ margins continue to increase even when training error reaches zero.
  - ✓ helps to explain empirically observed phenomena: test error continues to drop even after training error reaches zero

Boosting vs. Bagging

- Bagging doesn't work so well with stable models. Boosting might still help.
- Boosting might hurt performance on noisy datasets. Bagging doesn't have this problem.
- On average, boosting helps more than bagging, but it is also more common for boosting to hurt performance.
- In practice bagging almost always helps.
- Bagging is easier to parallelize.