Data-driven Intelligent Systems

Lecture 14
Ensemble Learning 1

KNOWLEDGE
TECHNOLOGY

http://www.informatik.uni-hamburg.de/WTM/

Ensemble Learning – Overview

- Benefits of ensembles
- How to combine their outputs
- Bagging
- Boosting
  - AdaBoost

Ensemble Learning

- So far – learning methods learn a single hypothesis (model), chosen from a hypothesis space to make predictions

- ***"There ain't no such thing as a free lunch"***
  - No single algorithm wins all the time!

- Ensemble learning
  - select a collection (*ensemble*) of hypotheses (*models*) and combine their predictions

- **Example:** Generate 100 different decision trees from the same or different training set and have them vote on the best classification for a new example.

Value of Ensembles

- Key motivation: ***reduce*** the ***error rate***!

  Hope: it is less likely that an *ensemble* misclassifies an example
- **Examples**: Human ensembles are demonstrably better:
  - How many jelly beans in the jar?:
    Individual estimates vs. group average
  - Who Wants to be a Millionaire: Audience vote
  - Diagnosis based on multiple doctors' majority vote

- **Theory behind**: We combine multiple ***independent*** and ***diverse*** decisions
  - each is at least more accurate than random guessing

    → random errors cancel each other out

    → correct decisions are more consistent and add up

Achieving Diversity (1)

1. Using different learning ***algorithms***

   ← *how many algorithms do we know?*
2. Using different ***hyper-parameters*** in the same algorithm

   ← *some parameters not as good as others*
3. Using different ***input representations***, e.g. different subsets of input *features*

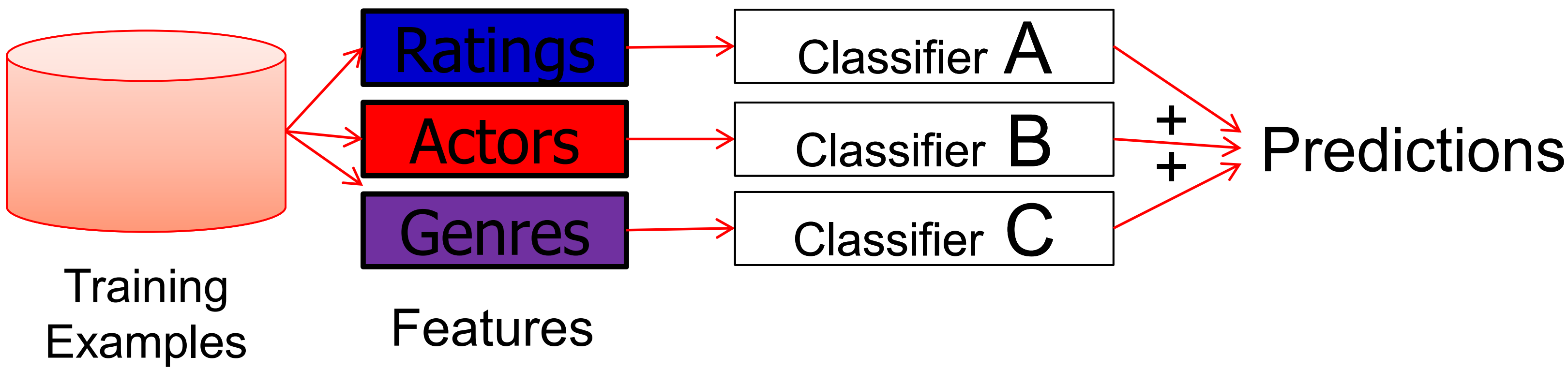
   ← *sometimes: diversity hand-designed*

   ← *requires redundant features*
   (e.g.: Random Subspace Method)
4. Using different ***subsets of training data***
   - e.g. bagging, boosting, and cascading

     ← *diversity achieved automatically*

Achieving Diversity (2)

3. Diversity from *differences in input features*:



Example: multimodal emotion classification:

Model 1 ← vision; Model 2 ← audio; Model 3 ← text

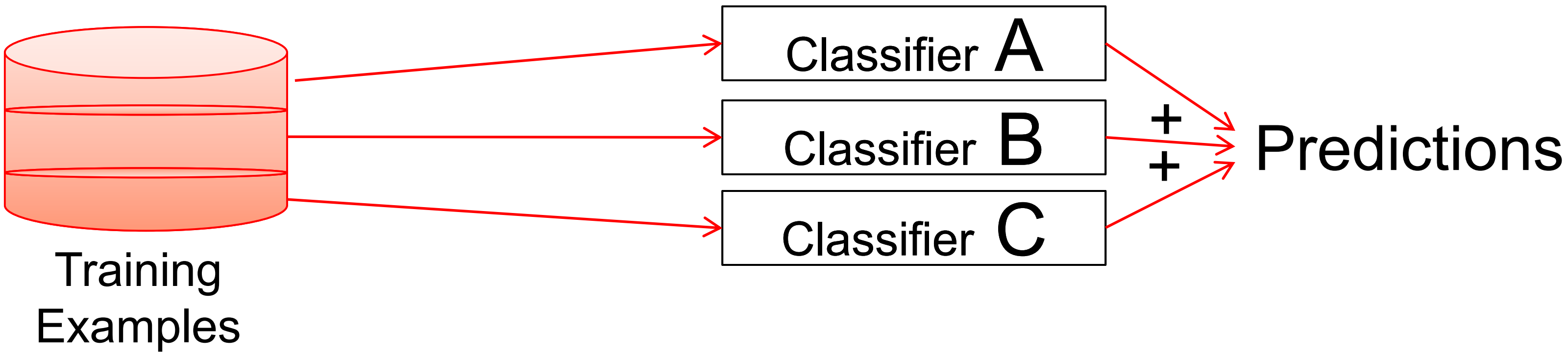
Automatic feature selection:

Random Subspace Method, works for large feature sets with redundant features

Achieving Diversity (3)

4. Diversity from *subsets of training data*:

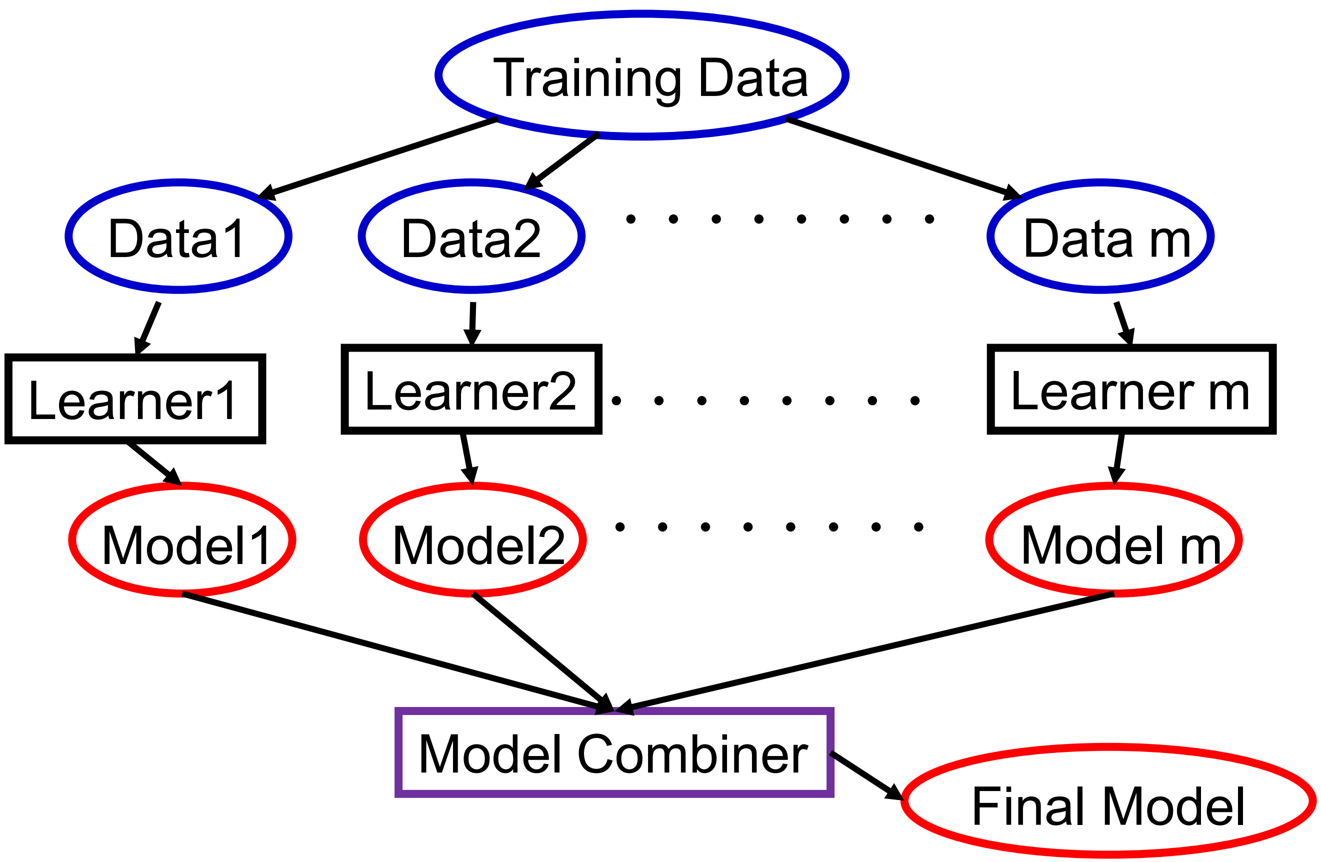
use ***different subsets of training data to*** learn multiple alternative definitions of a concept



Achieving Diversity (4)

4. Diversity from *subsets of training data*:

use ***different subsets of training data to*** learn multiple alternative definitions of a concept



Ensemble Learning – Overview

- Benefits of ensembles
- How to combine their outputs
- Bagging
- Boosting
  - AdaBoost

How to Combine the Outputs of Base Learners?

- ***Global approach*** is through fusion – the outputs of all learners are combined by *averaging*, *voting*, or *stacking**

- ***Local approach*** is based on *learner selection* – it examines the input and chooses the learner(s) responsible for generating the output

- ***Multistage combination*** use a serial approach where the next learner is trained with or tested on instances only where previous learners failed, or were inaccurate

**stacking*: a (simple) model classifies the learners' *outputs*

Global Approach: Averaging Example

- Guess: how many people are in the picture?

- Is the ensemble average better than the individual estimates?

Averaging Example

How many? True number = T; estimates = $\{x_i\}$, $i = 1, \ldots, N$

- Individual errors: $e_i^{ind} = |T - x_i|$
  - averaged: $L1^{ind} = \frac{1}{N}\sum_i^N e_i^{ind}$
- Ensemble estimate: $m^{arith} = \frac{1}{N}\sum_i^N x_i$
  - Ensemble error: $L1^{ens} = |T - m^{arith}|$

$$sq_i^{ind} = (T - x_i)^2$$

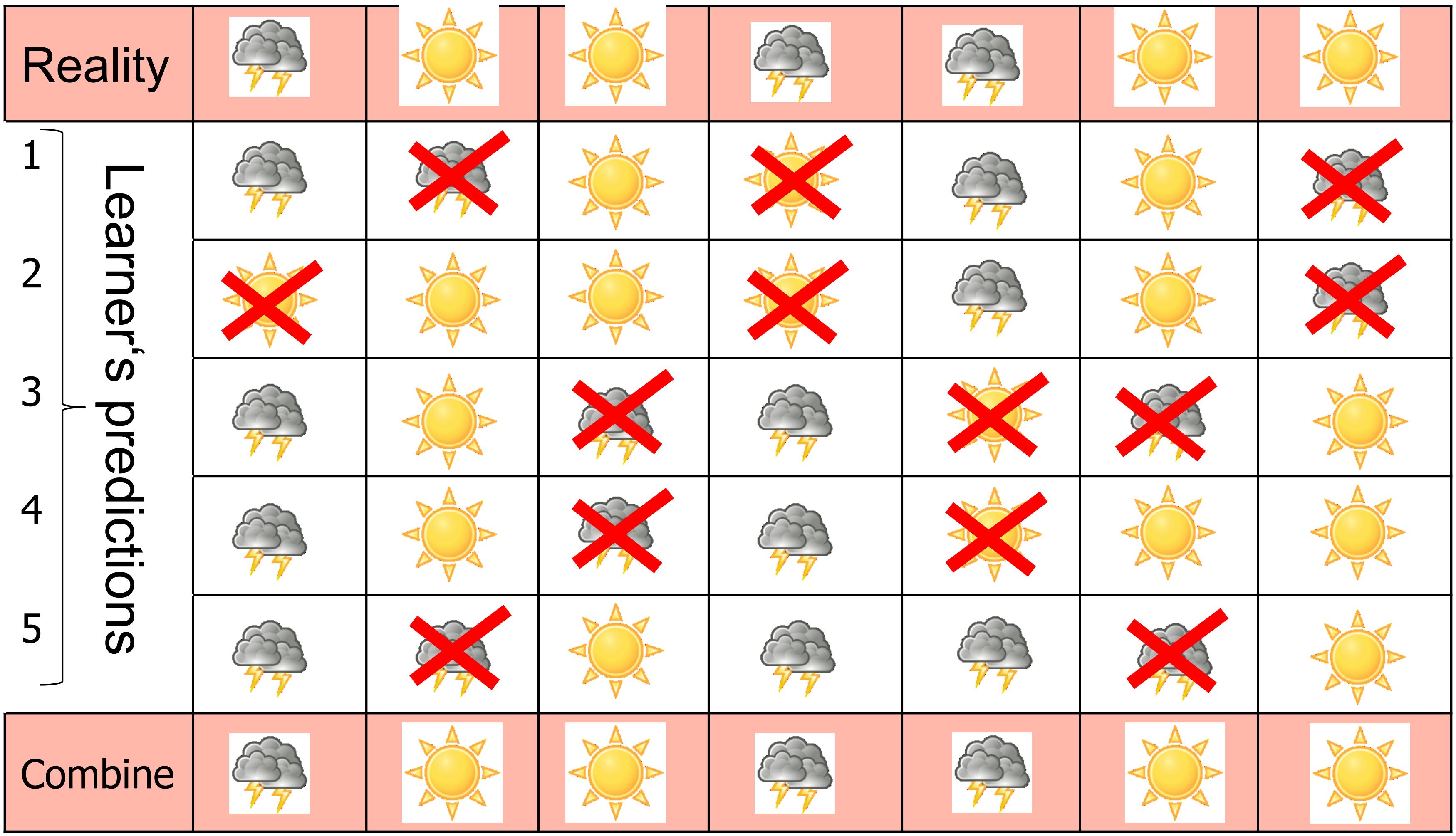
$$SQ^{ind} = \frac{1}{N}\sum_i^N sq_i^{ind}$$

$$SQ^{ens} = (T - m^{arith})^2$$

We find: $L1^{ens} \le L1^{ind}$ (equality if all $x_i \le T$ or all $x_i \ge T$)

We find: $SQ^{ens} \le SQ^{ind}$ (equality if all x_i equal)

Voting Example: Weather Forecast



- Combine decisions of multiple models using *voting* procedure!

Voting: Lower Error Than Individuals

- Assume, binary base classifiers with error rate ε = 0.3
- Then, majority vote of n=15 independent classifiers:

$$\varepsilon_{ensemble} = \sum_{k=\underbrace{ceil(15/2)}_{=8}}^{15} \binom{15}{k} \cdot \varepsilon^{k}(1-\varepsilon)^{15-k} = 0.05$$

number of k-subsets in 15 items (binomial coefficient)

probability of k outcomes in 15 draws, if p(outcome) = ε

→ *probability that exactly k classifiers make an error*

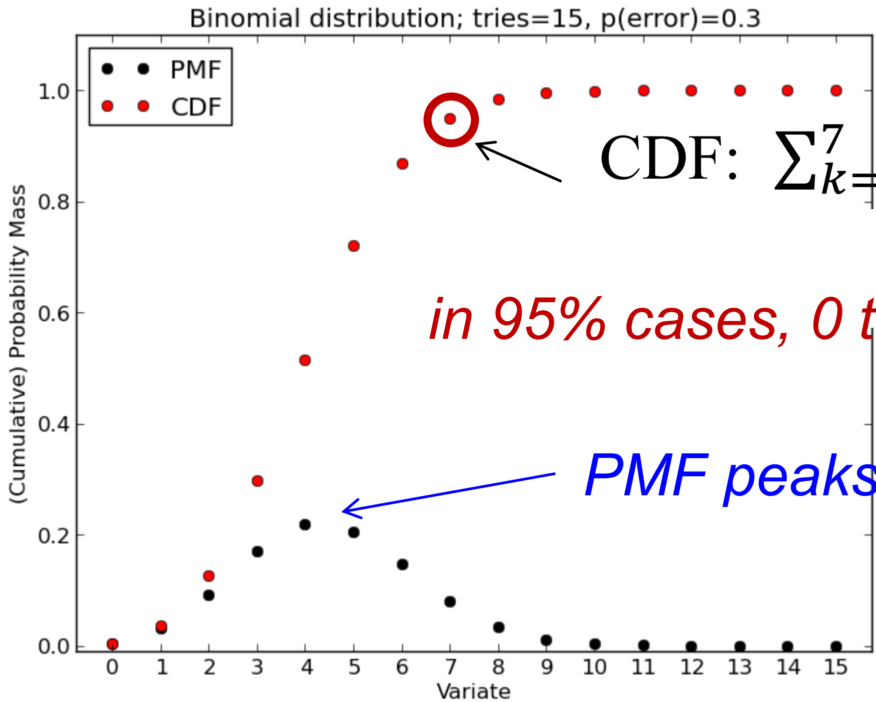
sum over where more than half
of the classifiers are wrong

- Binomial probability formula

Voting: Ensembles Give Better Results

- Majority vote of *n*=15 classifiers, error rate each ε=0.3:

$$\varepsilon_{ensemble} = \sum_{k=8}^{15} \underbrace{\binom{15}{k} \cdot \varepsilon^{k}(1-\varepsilon)^{15-k}}_{\textit{PMF}\text{, probability mass function}} = 0.05$$



CDF: $\sum_{k=0}^{7} PMF = 0.95 = 1 - \sum_{k=8}^{15} PMF$

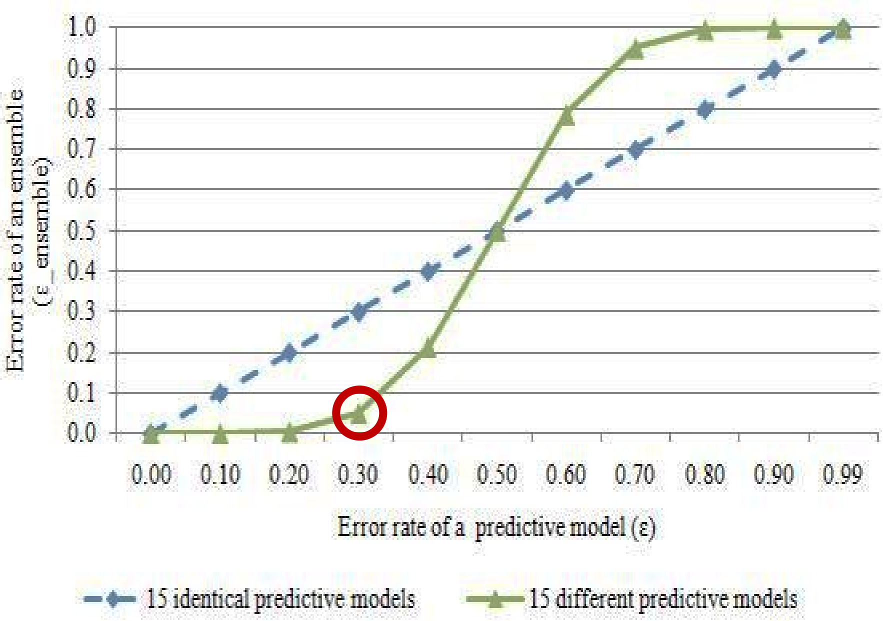
in 95% cases, 0 to 7 models are wrong

*PMF peaks at 0.3 * 15*
= 4.5

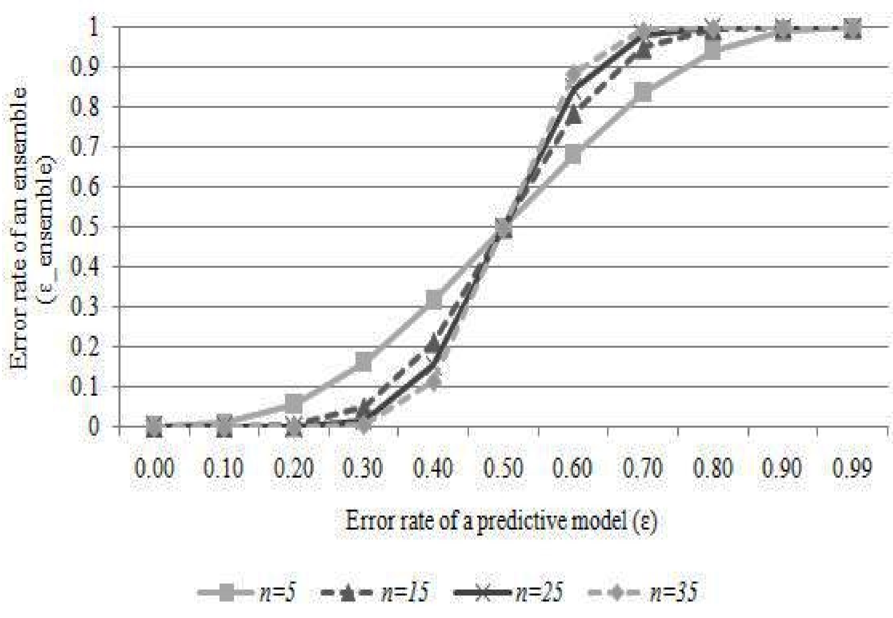
Voting: Ensembles Give Better Results

- Majority vote of n=15 classifiers, error rate each ε=0.3:

$$\varepsilon_{ensemble} = \sum_{k=8}^{15} \binom{15}{k} \cdot \varepsilon^k (1-\varepsilon)^{15-k} = 0.05$$



(a) Identical predictive models vs. different predictive models in an ensemble



(b) The different number of predictive models in an ensemble

Rank-Level Fusion Method

- Four-class problem (a,b,c,d)?

Rank / score	Classifier 1	Classifier 2	Classifier 3
4	c	a	d
3	b	b	b
2	d	d	c
1	a	c	a

$$r_a = r_a(C1) + r_a(C2) + r_a(C3) = 1 + 4 + 1 = 6$$

$$r_b = r_b(C1) + r_b(C2) + r_b(C3) = 3 + 3 + 3 = 9$$

$$r_c = r_c(C1) + r_c(C2) + r_c(C3) = 4 + 1 + 2 = 7$$

$$r_d = r_d(C1) + r_d(C2) + r_d(C3) = 2 + 2 + 4 = 8$$

- The winner-class is ***b*** because it has the maximum overall score

Global Approach: Combination Methods

Alternatives for combination are:

- Simple average (equal weights); for regression
- Majority voting; for classification
- Rank-level fusion; for multiple classes
- Average using weighted sum (unconstrained weights)
- Voting: linear combination of outputs d_j for learners j:

$$y = \sum w_j \cdot d_j \quad \text{where weights } w_j \geq 0 \text{ and } \sum w_j = 1$$

with trained weights, this is an example of stacking

- Median
- Maximum or minimum
- Geometric mean: $\sqrt[k]{d_1 \cdot d_2 \cdot \ldots \cdot d_k}$

Local Approach: Dynamic Classifier Selection

Algorithm

1. Define the local region
   - E.g. find the *k* nearest training points to the test input
2. Estimate the competence of the classifiers there
   - E.g. obtain accuracies of the classifiers on these points
3. Select
   - Choose the one that performs best
     on them
     (or vote over a few “competent” ones).

Ensemble Learning – Overview

- Benefits of ensembles
- How to combine their outputs
- Bagging
- Boosting
  - AdaBoost

Homogenous Ensembles

- Use a single, arbitrary learning algorithm but manipulate training data to make it learn multiple (diverse) models.
  - Data1 $\neq$ Data2 $\neq$ … $\neq$ Data n
  - Learner1 = Learner2 = … = Learner n

Methods to change training data:

- Bagging:
  - Resample training data
- Boosting:
  - Reweight training data

Bagging: Bootstrap Aggregation (1)

- Training
  - Given a set *D* of tuples
  - At each iteration *i*, a ***random subset*** D_i of tuples is sampled *with replacement* from *D* for training (bootstrap) *
  - A classifier model M_i is learned for each training set D_i
- **Classification**: classify an unknown sample *X*
  - Each classifier M_i returns its class prediction
  - The bagged classifier *M** counts the votes and assigns *X* to the class with the most votes
- **Regression** (predict continuous outputs): by taking the average value of each prediction for a given test sample

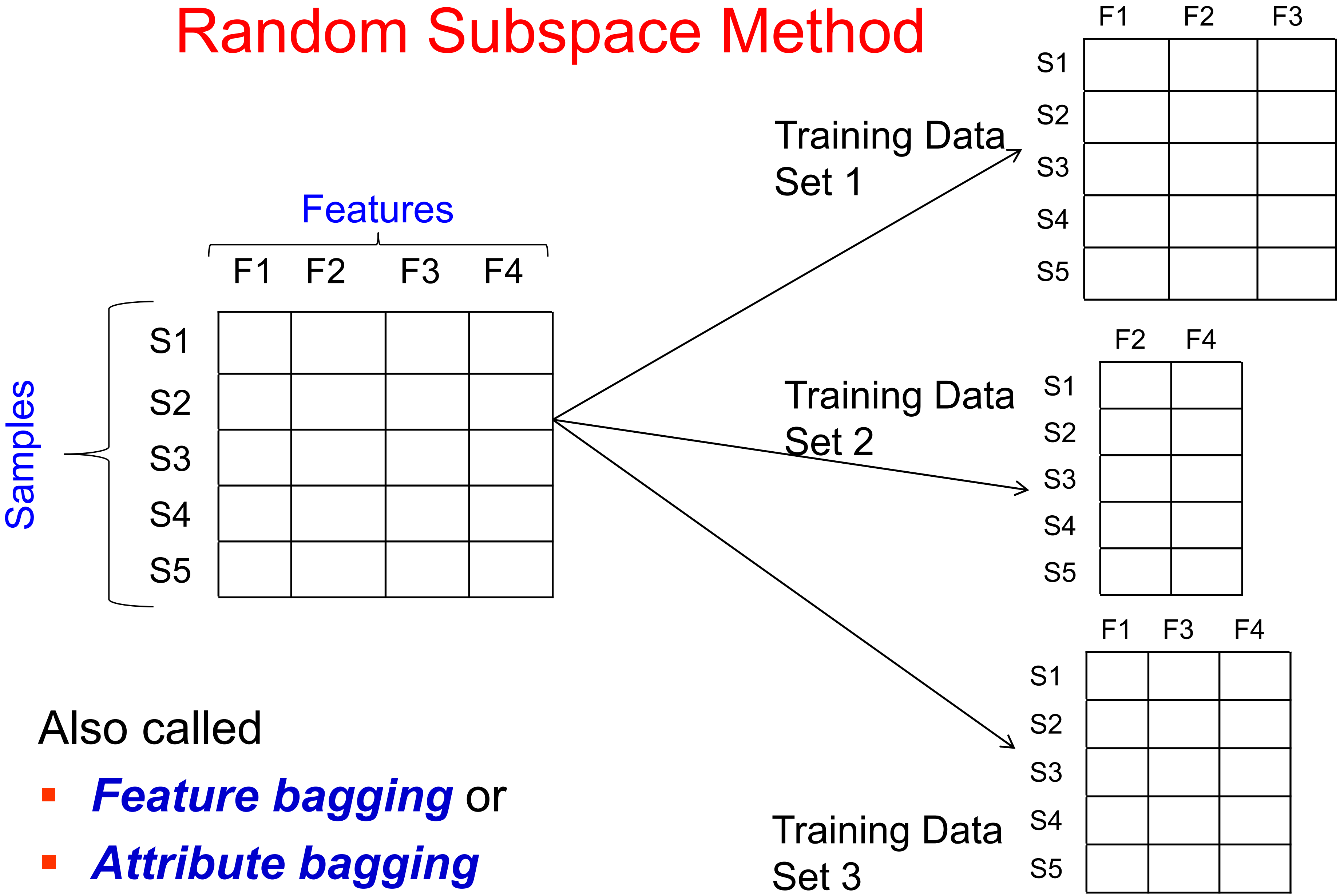
- → *each set D_i is expected to have ~2/3 unique tuples and ~1/3 duplicates*

Bagging: Bootstrap Aggregation (2)

- Accuracy
  - Often significantly better than a single classifier derived from D
  - For noisy data: not considerably worse, more robust
  - Improved accuracy in prediction
  - Decreases error by ***decreasing the variance*** in the results due to ***unstable learners***
    (some algorithms' output can change dramatically when the training data is slightly changed, e.g. decision trees, NNs, …)
  - Increases classifier stability,
    reduces variance!

(Breiman, 1996)

Random Subspace Method



Also called

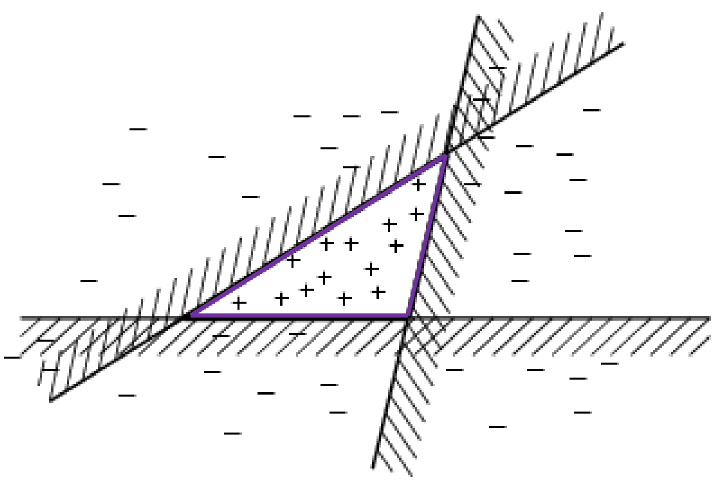
- ***Feature bagging*** or
- ***Attribute bagging***

Ensemble Learning – Overview

- Benefits of ensembles
- How to combine their outputs
- Bagging
- Boosting
  - AdaBoost

Ensemble Learning

- Another way of thinking about ensemble learning:
  - way of ***enlarging the hypothesis space,*** i.e., the ensemble itself is a hypothesis
  - the new hypothesis space is the set of all possible ensembles constructible from hypotheses of the original space
- Increased power of ensemble learning:



- Three linear threshold hypotheses (positive examples on the non-shaded sides)
- Ensemble classifies as positive any example classified positively by ***all*** three
- The resulting triangular region hypothesis is not expressible by any of the base hypotheses

Boosting

- How boosting works?
  - $D_t(i)$ **_Weights_** are assigned to each training tuple *i*
  - A series of *k* classifiers is iteratively learned ($t = 1,\ldots,k$)
  - After a classifier M_t is learned, the weights of tuples are updated to allow the subsequent classifier, M_{t+1}, to **_pay more attention to the training tuples that were misclassified_** by M_t
  - The final M^* combines the votes of each individual classifier, where the α_t **_weight of each classifier's vote is a function of its accuracy_**
- Boosting can be extended for numeric prediction
- Compared with bagging:
  Boosting tends to achieve greater accuracy,
  but it also risks overfitting the model

Boosting: Strong And Weak Learners (1)

- ***Strong Learner***
  - Take labeled data for training
  - Produce a classifier which can be ***arbitrarily accurate***
  - Strong learners are an objective of machine learning
- ***Weak Learner***
  - Take labeled data for training
  - Produce a classifier which is ***more accurate than random guessing***
  - Weak learners can be base classifiers for ensemble methods

Boosting: Strong And Weak Learners (2)

- ***Weak Learner:*** only needs to generate a hypothesis with a training accuracy greater than 0.5, i.e., < 50% error over any distribution
  - Strong learners are very difficult to construct
  - Constructing weaker learners is relatively easy
- Can a set of weak learners create a single strong learner?
  - Yes! Boost weak classifiers
    to a strong learner! (Shapire,1990)

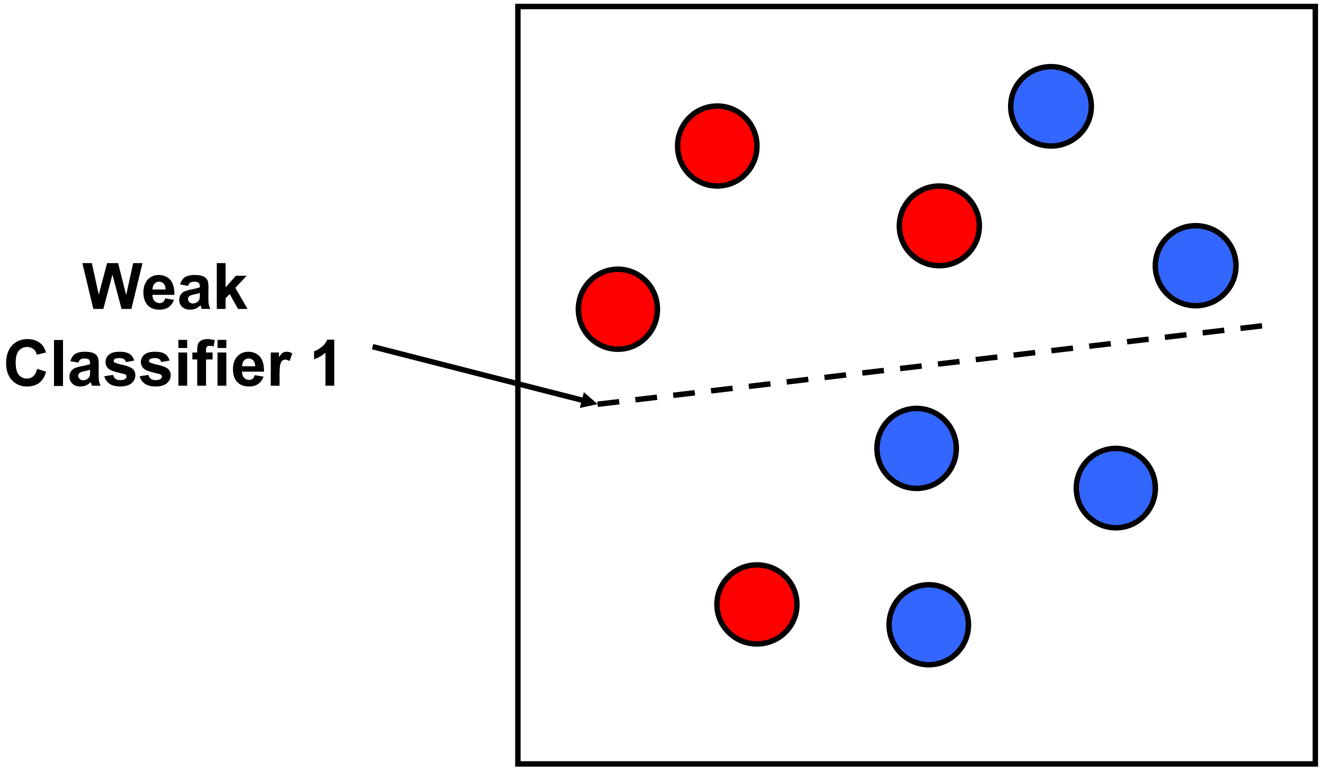
Boosting: Use Weak Classifiers

Idea of iterative learning: Focus on difficult samples which are not correctly classified in the previous steps.

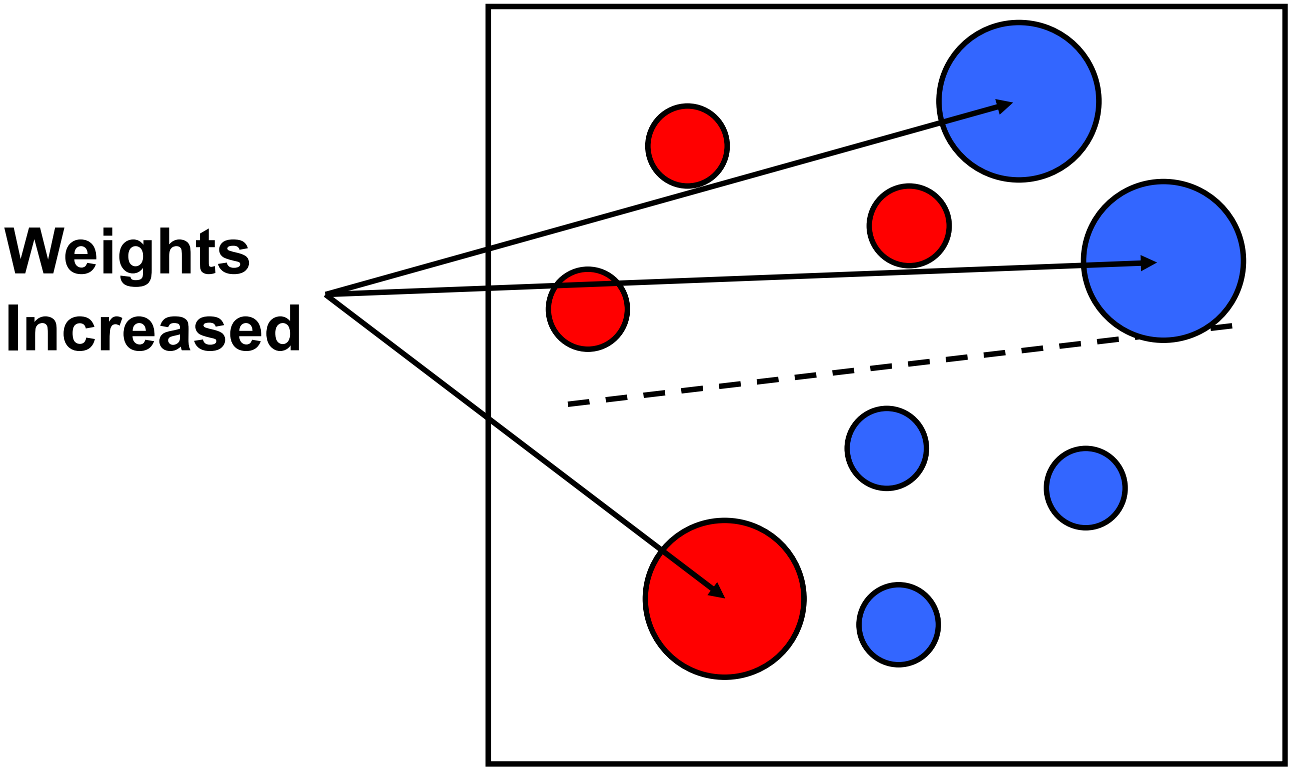
Use different data distribution:

- Start with ***uniform weighting*** of samples ($D_t(i)$)
- During each step of learning
  - Increase weights of the samples which are not correctly learned by the weak learner
  - Decrease weights of the samples which are correctly learned by the weak learner

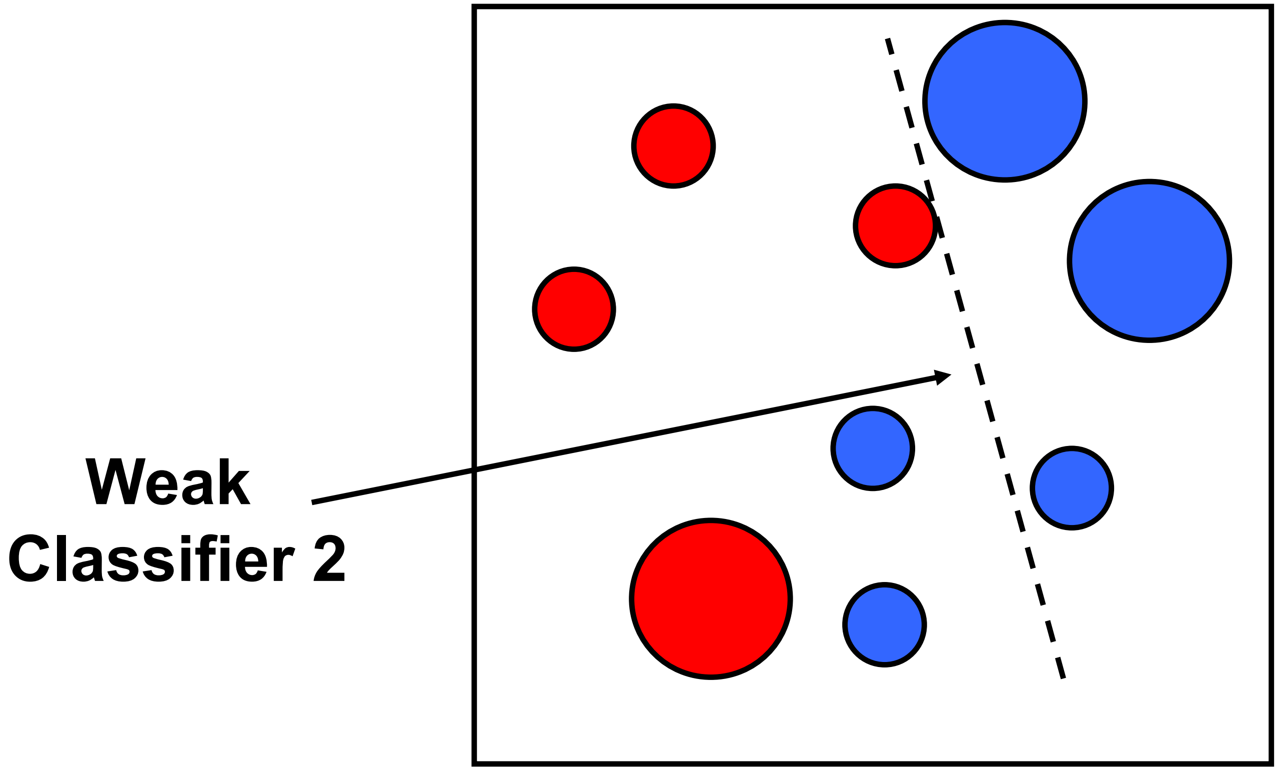
Boosting Idea



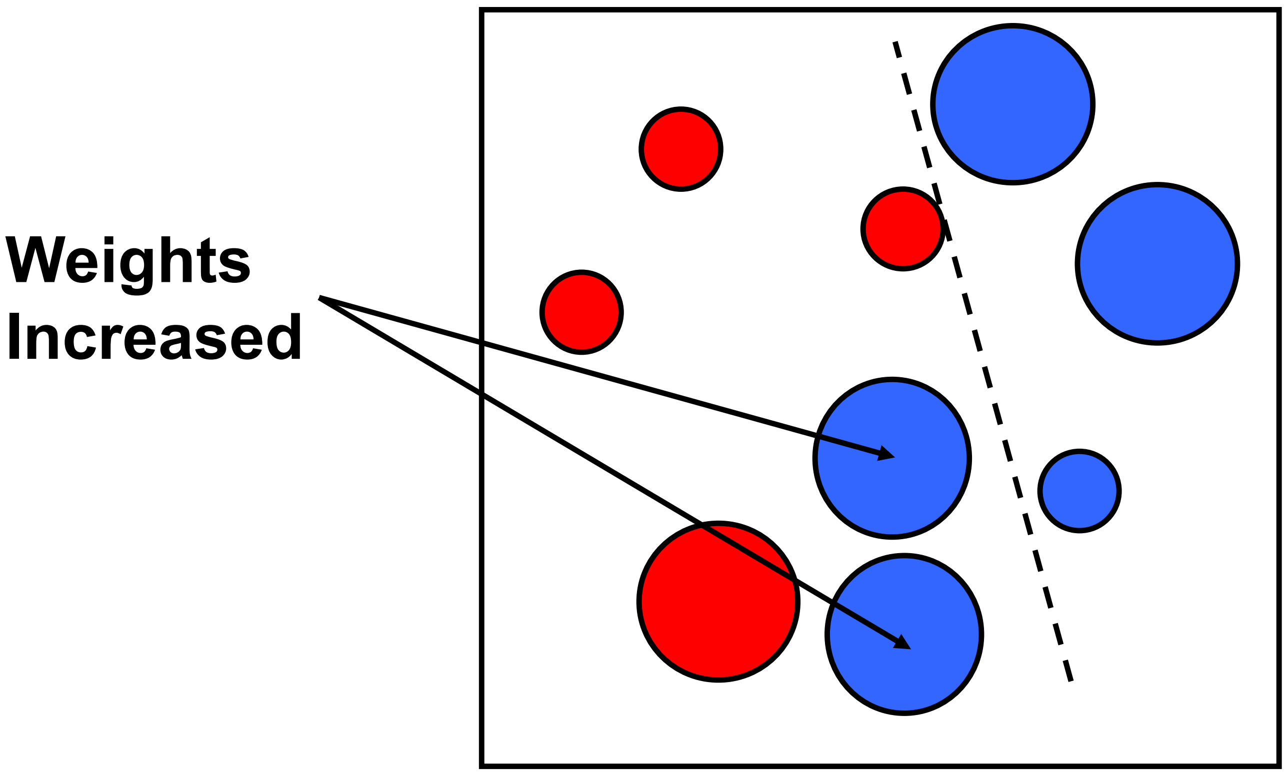
Boosting Idea



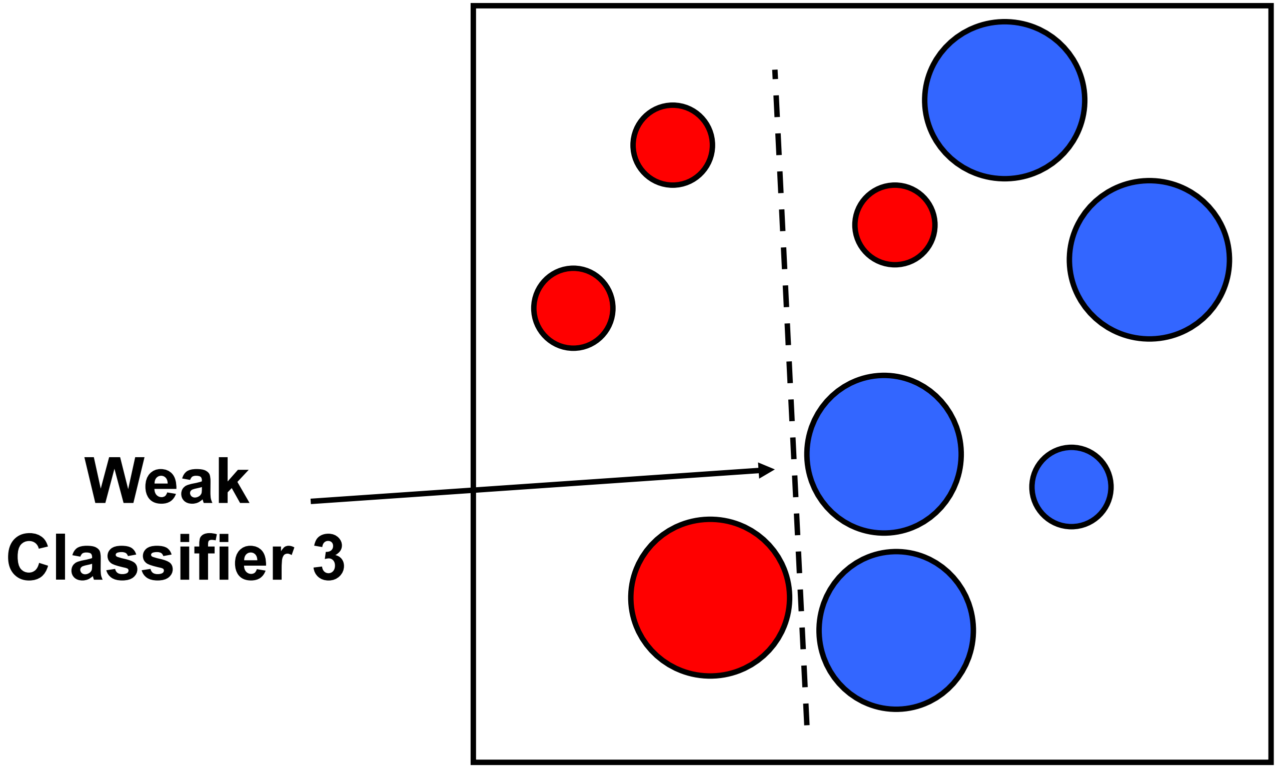
Boosting Idea



Boosting Idea

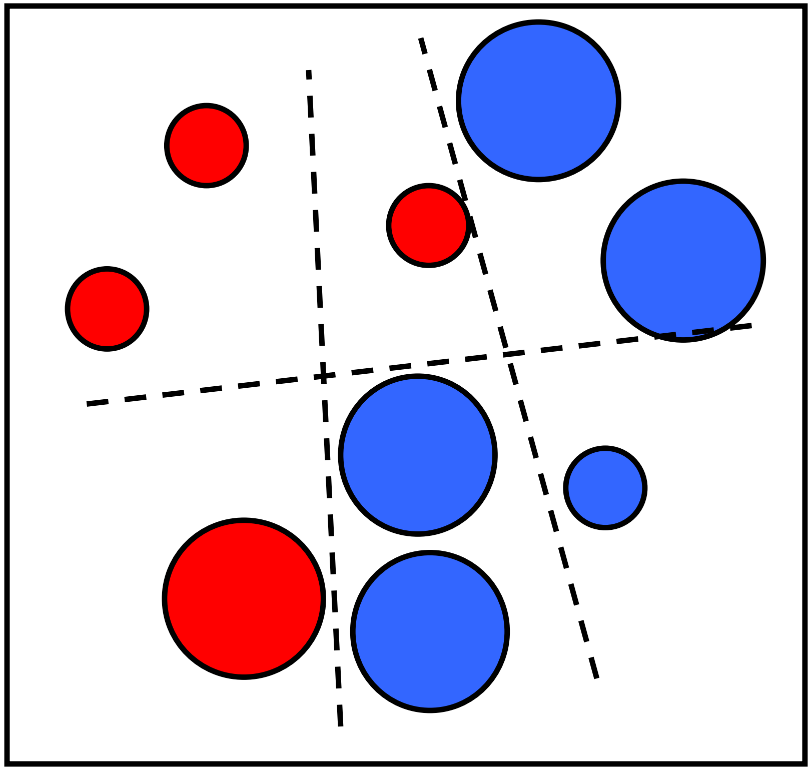


Boosting Idea



Boosting Idea

Final classifier is a combination of weak classifiers



Boosting: Combine Weak Classifiers

- Idea for combination: better weak classifier gets a larger weight!

- Weighted voting
  - Construct ***strong classifier*** by ***weighted voting*** of the weak classifiers
  - Weight of each learner is directly proportional to its accuracy

Ensemble Learning – Overview

- Benefits of ensembles
- How to combine their outputs
- Bagging
- Boosting
  - AdaBoost

AdaBoost: Adaptive Boosting

- Does not need to know the number of weak classifiers in advance
- Does not need to know error bounds on the weak classifiers, unlike earlier boosting algorithms

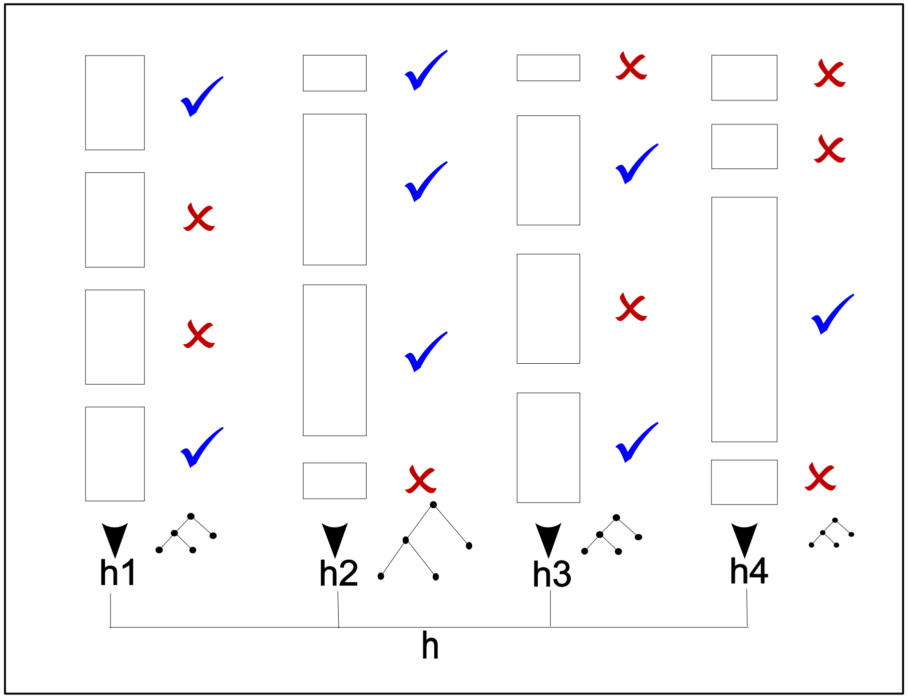
AdaBoost: Adaptive Boosting

- Each rectangle corresponds to an example, with weight proportional to its height.

$D_t(i)$

- Crosses correspond to misclassified examples.
- Size of "decision tree" indicates the weight of that hypothesis in the final ensemble.

α_t



Initialization

Given : $(x_1, y_1), \ldots (x_n, y_n)$, where $x_i \in X, y_i \in Y = \{-1,+1\}$

Initialze distribution (weight) $D_{t=1}(i) = 1/n$; such that n = M + L

M = number of positive $(+1)$ examples; L = number of negative (-1) examples

Main Loop

For $t = 1, \ldots T$

{ Step1a : Find the classifier $h_t : X \to \{-1,+1\}$ that minimizes the

error with respect to D_t, that means : $h_t = \arg\left[\min_q(\varepsilon_q)\right]$

Step1b : error $\varepsilon_t = \sum_{i=1}^{n} D_t(i) * I_{[h_t(x_i) \neq y_i]}$, where $I_{[h_t(x_i) \neq y_i]} = \begin{cases} 1 \text{ if } [h_t(x_i) \neq y_i] \text{(classified incorrectly)} \\ 0 \quad \textit{otherwise} \end{cases}$

checking step : prerequisite : $\varepsilon_t < 0.5$: (error smaller than 0.5 is ok) otherwise stop.

Step2 : $\alpha_t = \frac{1}{2} \ln \frac{1-\varepsilon_t}{\varepsilon_t}$, α_t = weight (or confidence value).

*Step*3 : $D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$, see next slide for explanation

Step4 : Current total cascaded classifier error $CE_t = \sum_{j=1}^{j=t} E_j\left(t, \alpha_\tau, h_\tau(x_i)\right)$

while the current classifier error $E_\tau = \frac{1}{n}\sum_{\tau=1}^{n} I\left(t, \alpha_\tau, h_\tau(x_i)\right)$,

and $I(\,)$ is defined as follows :

If x_i is correctly classified by the current cascaded classifier , i.e.

$y_i = sign\left(\sum_{\tau=1}^{t} \alpha_\tau h_\tau(x_i)\right)$, hence error $I\left(t, \alpha_\tau, h_\tau(x_i)\right) = 0$

If x_i is incorrectly classified by the current cascaded classifier i.e.

$y_i \neq sign\left(\sum_{\tau=1}^{t} \alpha_\tau h_\tau(x_i)\right)$, hence error $I\left(t, \alpha_\tau, h_\tau(x_i)\right) = 1$

If $CE_t = 0$ then T = t, break;

}

The output $o_t(x_i) = \sum_{\tau=1}^{t} \alpha_\tau h_\tau(x_i)$, and $S\left(t, \alpha_\tau, h_\tau(x_i)\right) = \begin{cases} 1 \text{ if } y_i = sign(o_t(i)) \\ 0 \quad \textit{otherwise} \end{cases}$

where Z_t = *normalization* factor, so D_t is a *probability distrubution*

$$Z_t = \sum_{i=1}^{n\_correctly\_classified} correct\_weight + \sum_{i=1}^{n\_incorrectly\_classified} incorrrect\_weight$$

$$= \sum_{i=1}^{n\_correctly\_classified} D_t(i) e^{-\alpha_t} y_i h_t(x_i) + \sum_{i=1}^{n\_incorrectly\_classified} D_t(i) e^{\alpha_t} y_i h_t(x_i)$$

Strong Classifier

Final strong classifier: $H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

enlarged versions on the following slides

Initialization

Given $(x_{1,} y_1), ..(x_n, y_n)$, where $x_i \in X, y_i \in Y = \{-1, +1\}$

Initialze weights of samples $D_{t=1}(i) = 1/n$;

such that $\mathrm{n} = \mathrm{M} + \mathrm{L}$

M = number of positive $(+1)$ examples;

L = number of negative (-1) examples

Adapted from:
Kin Hong Wong: Adaboost for building robust classifiers.
http://appsrv.cse.cuhk.edu.hk/~khwong/

Main Loop (Steps 1, 2, 3)

For $t = 1, \ldots T$

$\{$

Step1a : Find the classifier $h_t : X \to \{-1,+1\}$ that minimizes the

error with respect to D_t : $h_t = \arg\left[\min_q \left(\varepsilon_q\right)\right]$

Step1b : error $\varepsilon_t = \sum_{i=1}^{n} D_t(i) * I_{[h_t(x_i) \neq y_i]}$,

$\boxed{I = \text{incorrectness}}$

$$\text{where } I_{[h_t(x_i) \neq y_i]} = \begin{cases} 1 \text{ if } \left[h_t(x_i) \neq y_i\right] \text{ (classified incorrectly)} \\ 0 \qquad\qquad\qquad \textit{otherwise} \end{cases}$$

Check whether $\varepsilon_t < 0.5$, otherwise stop.

Step2 : $\alpha_t = \frac{1}{2} \ln \frac{1 - \varepsilon_t}{\varepsilon_t}$, α_t = weight of classifier (confidence).

*Step*3 : $D_{t+1}(i) = \dfrac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$

Main Loop (Step 4)

$$\text{Step4 : Current total cascaded classifier error } CE_t = \sum_{j=1}^{j=t} E_j\left(t, \alpha_\tau, h_\tau(x_i)\right)$$

$$\text{where the current classifier error } E_\tau = \frac{1}{n}\sum_{\tau=1}^{n} I\left(t, \alpha_\tau, h_\tau(x_i)\right),$$

and $I(\,)$ denotes incorrectness for x_i of the current cascaded classifier :

$$y_i = sign\left(\sum_{\tau=1}^{t} \alpha_\tau h_\tau(x_i)\right) \rightarrow I\left(t, \alpha_\tau, h_\tau(x_i)\right) = 0$$

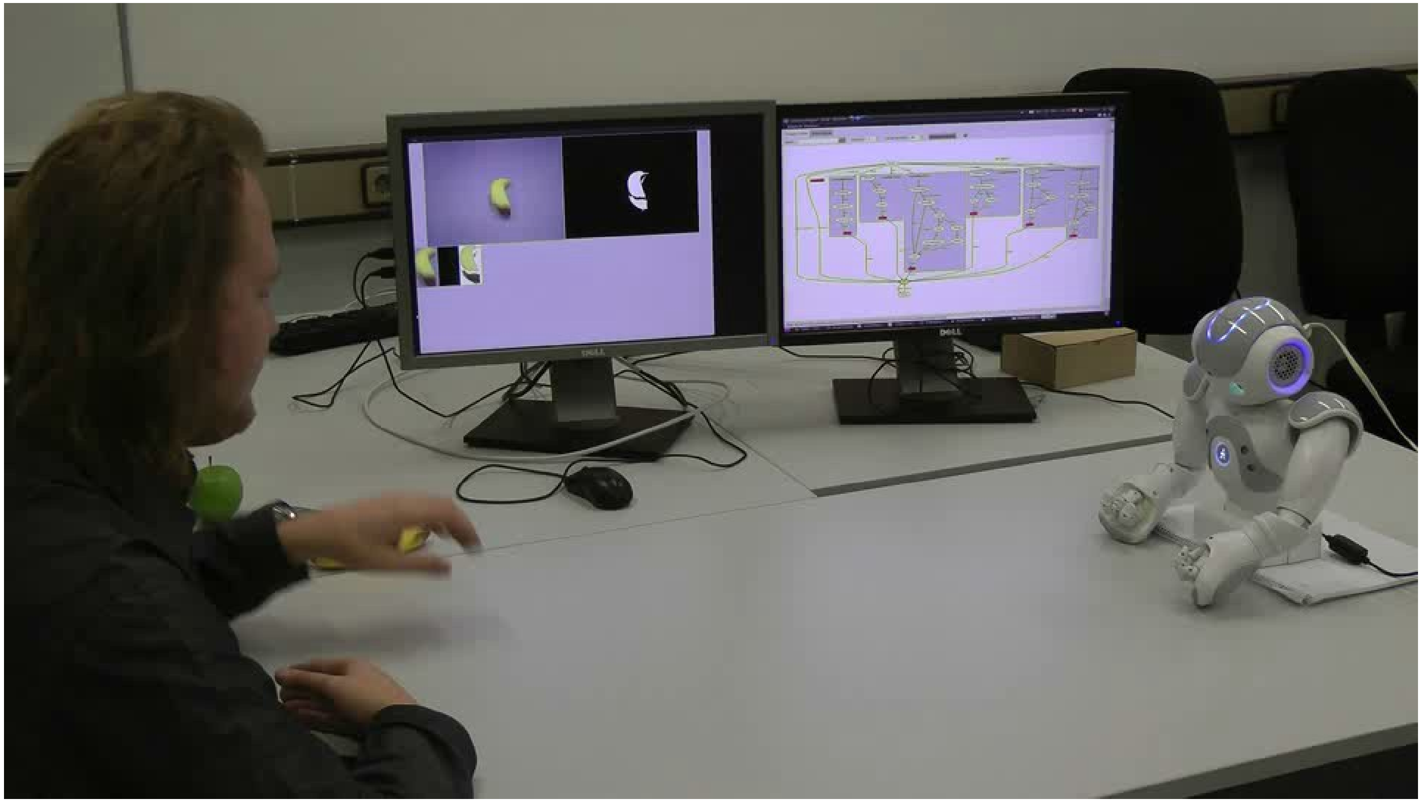
$$y_i \neq sign\left(\sum_{\tau=1}^{t} \alpha_\tau h_\tau(x_i)\right) \rightarrow I\left(t, \alpha_\tau, h_\tau(x_i)\right) = 1$$

If $CE_t = 0$ then T = t, break;

add threshold
if needed

$$\boxed{\text{Final strong classifier: } H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x) - 0\right)}$$

Hybrid Ensemble Learning with the NAO



NAO learns objects based on an ***ensemble*** of ***neural networks***

- Every network classifies based on ***different features***:
  pixel patterns, color & texture, or SURF features

http://www.informatik.uni-hamburg.de/WTM/teaching/WiSe11_HumanRobotInteraction_Pj.shtml