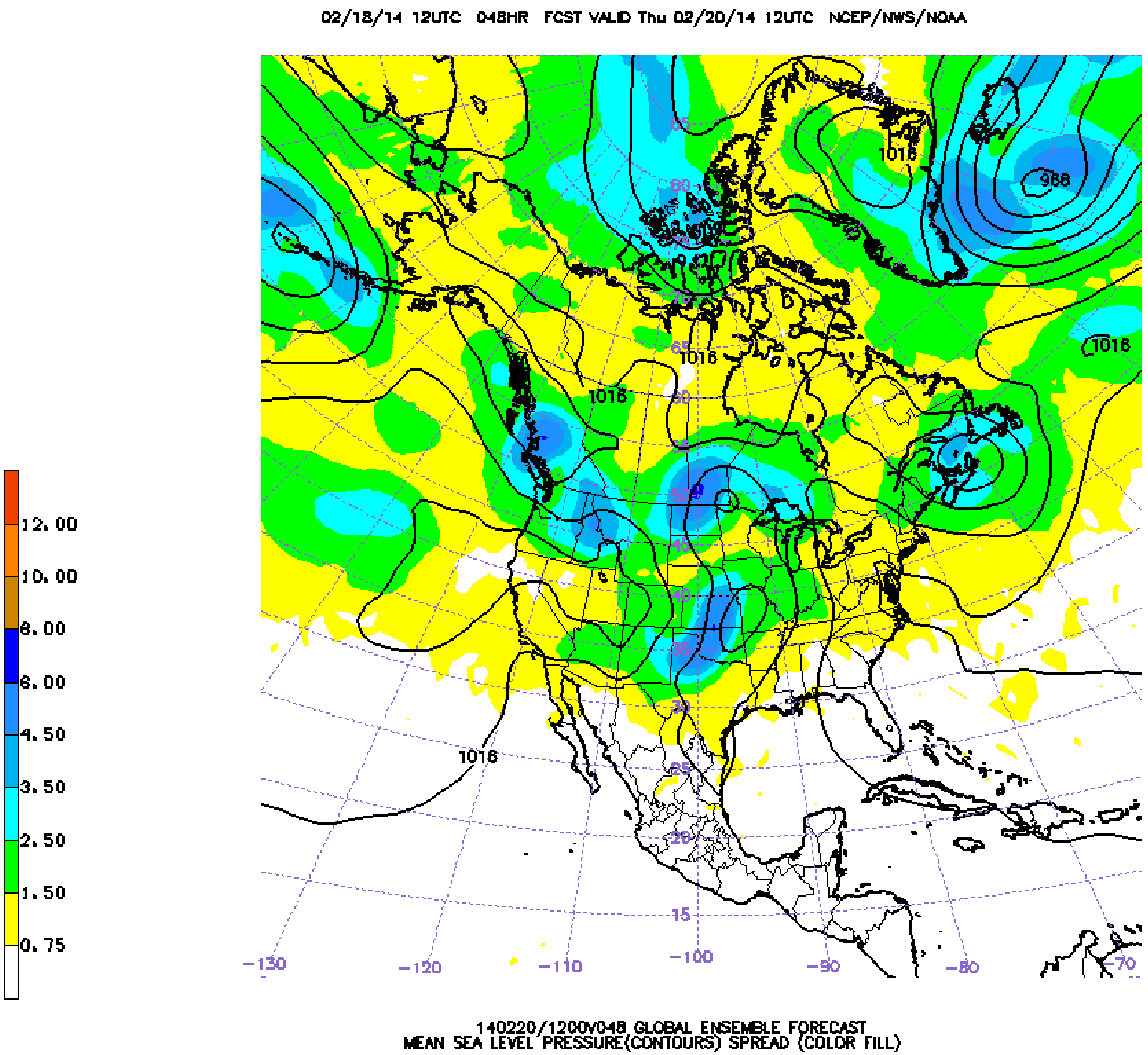
Classification

Ensemble Methods 1

Ensemble methods

- Basic idea of ensemble methods:
  - Combining predictions from competing models often gives better predictive accuracy than individual models.
- Shown to be empirically successful in wide variety of applications.
  - See table on p. 294 of textbook.
- Also now some theory to explain why it works.

Ensemble weather forecasting



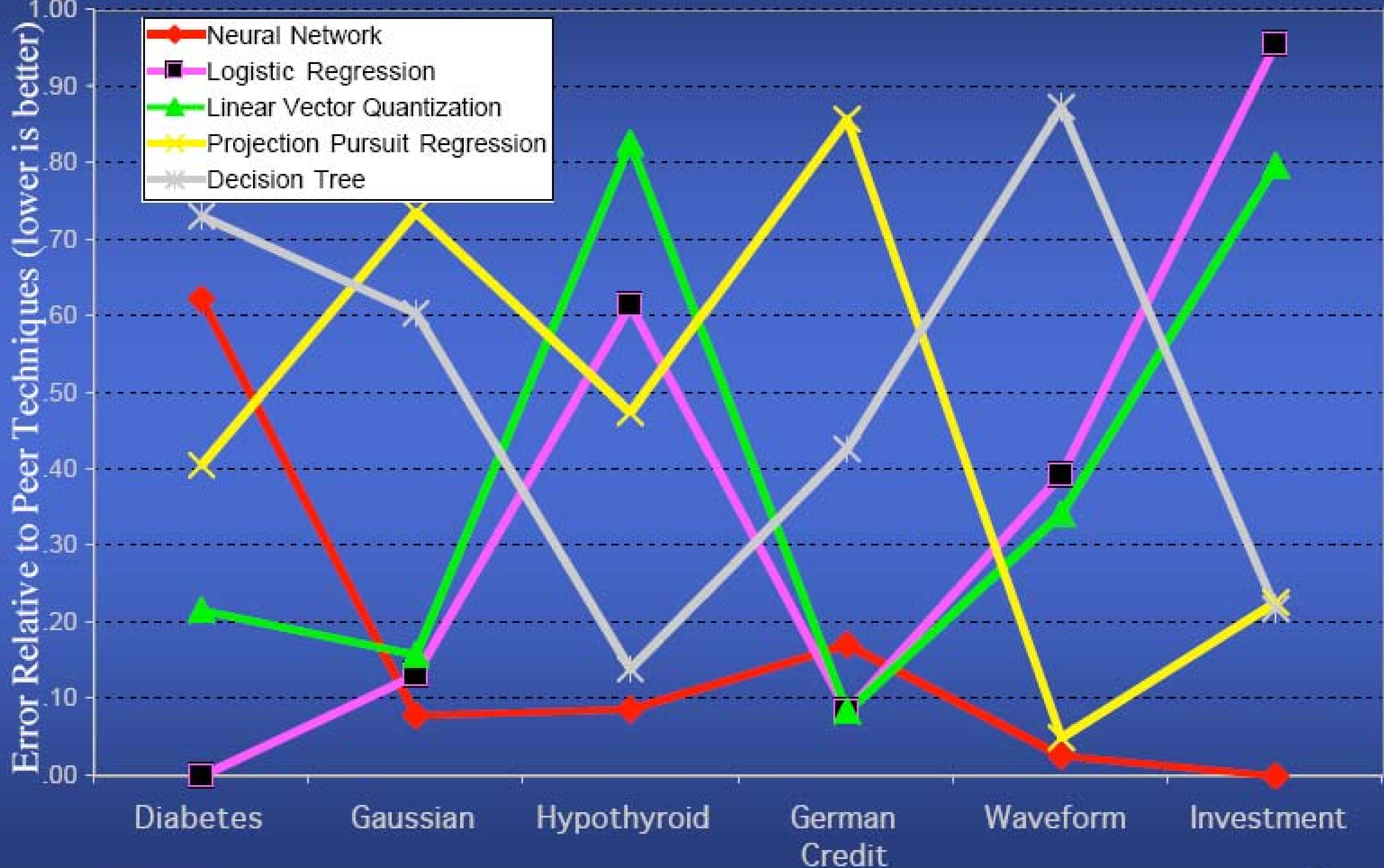
Build and using an ensemble

1) Train multiple, separate models using the training data.

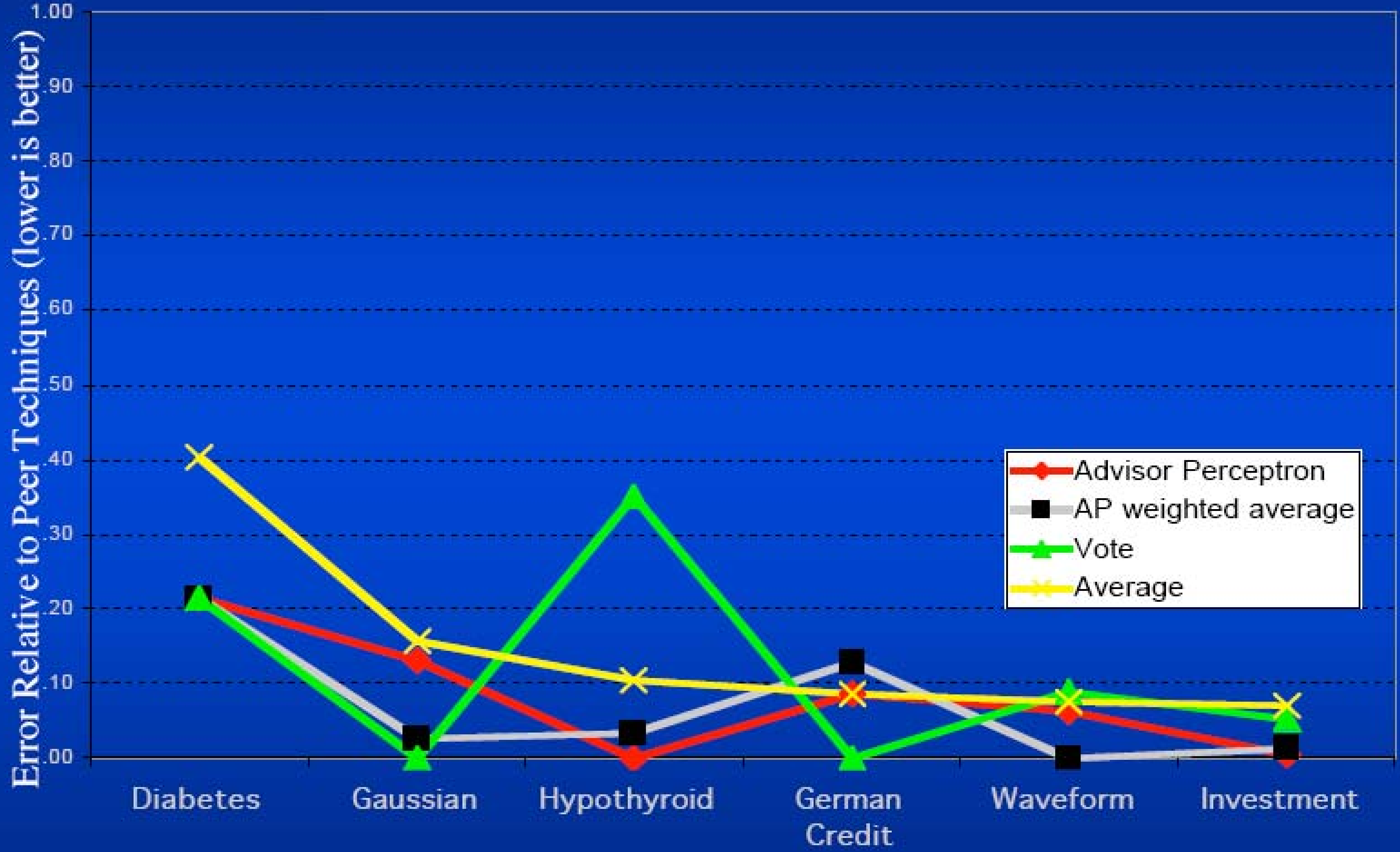
2) Predict outcome for a previously unseen sample by aggregating predictions made by the multiple models.

Relative Performance Examples: 5 Algorithms on 6 Datasets

(John Elder, Elder Research & Stephen Lee, U. Idaho, 1997)



Essentially every Bundling method improves performance



Estimation surfaces of five model types

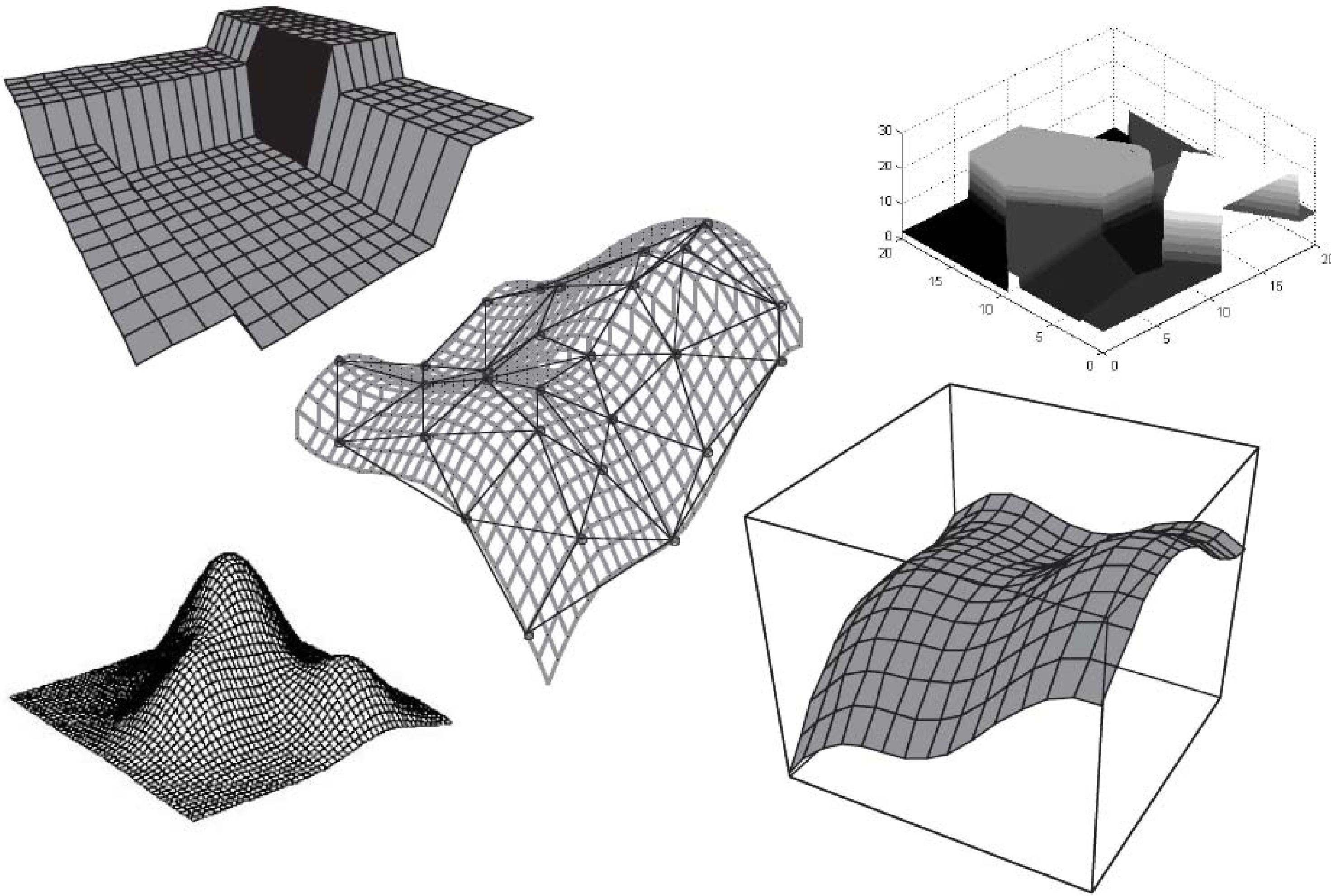


Figure 3. Estimation surfaces of five modeling algorithms. Clockwise from top left: decision tree, nearest neighbor, polynomial network, kernel; center: Delaunay planes (Elder 1993).

Ensemble methods

- Useful for classification or regression.
  - For classification, aggregate predictions by *voting*.
  - For regression, aggregate predictions by *averaging*.
- Model types can be:
  - Heterogeneous
    - Example: neural net combined with SVM combined decision tree combined with …
  - Homogeneous – most common in practice
    - Individual models referred to as base classifiers (or regressors)
    - Example: ensemble of 1000 decision trees

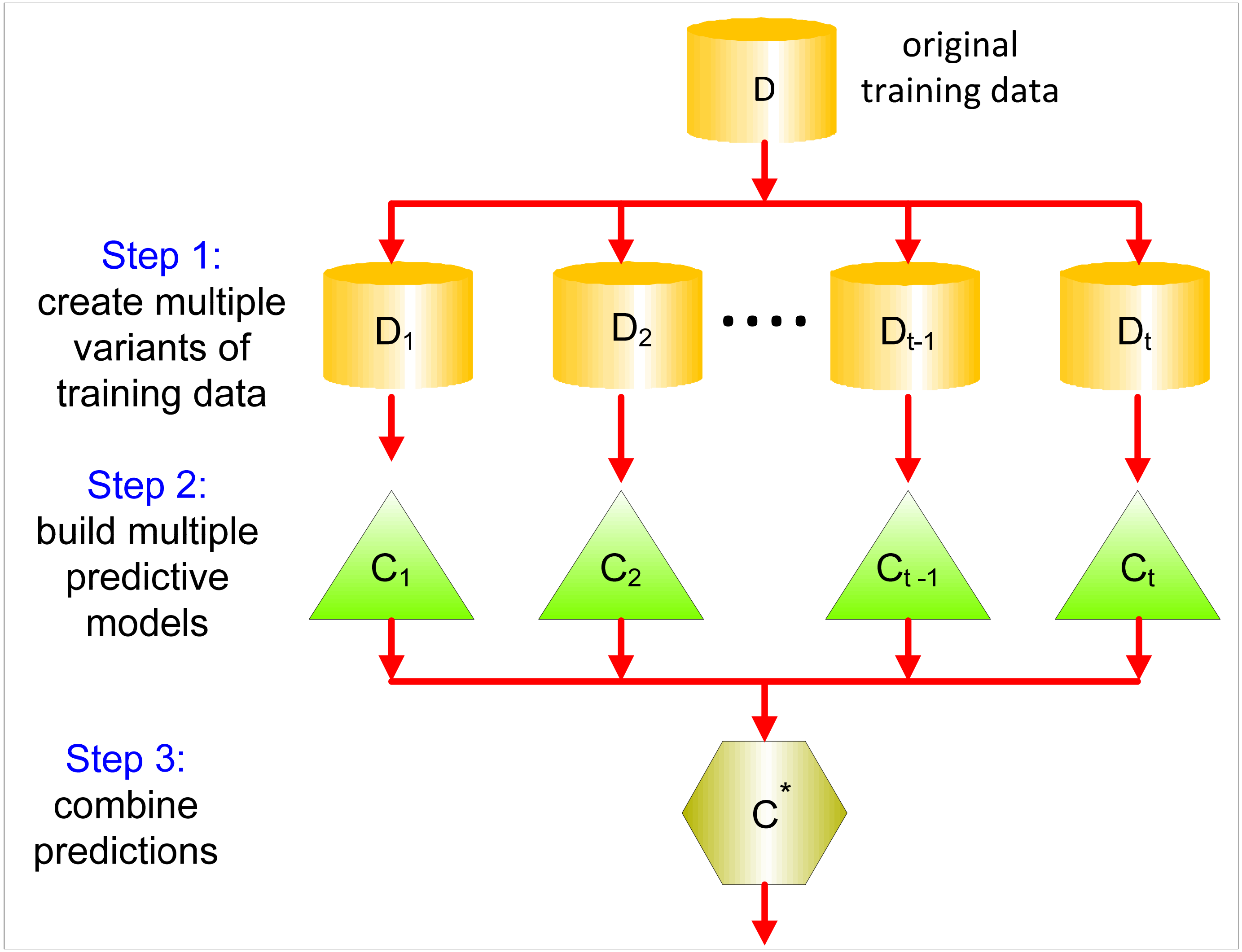
Classifier ensembles

- Committee methods
  - *m* base classifiers trained independently on different samples of training data
  - Predictions combined by unweighted voting
  - Performance:

    $$E[\text{ error }]_{\text{ave}} / m \;\leq\; E[\text{ error }]_{\text{committee}} \;\leq\; E[\text{ error }]_{\text{ave}}$$
  - Example: bagging

- Adaptive methods
  - *m* base classifiers trained sequentially, with reweighting of instances in training data
  - Predictions combined by weighted voting
  - Performance: $E[\text{ error }]_{\text{train}} + O(\,[\,md / n\,]^{1/2}\,)$
  - Example: boosting

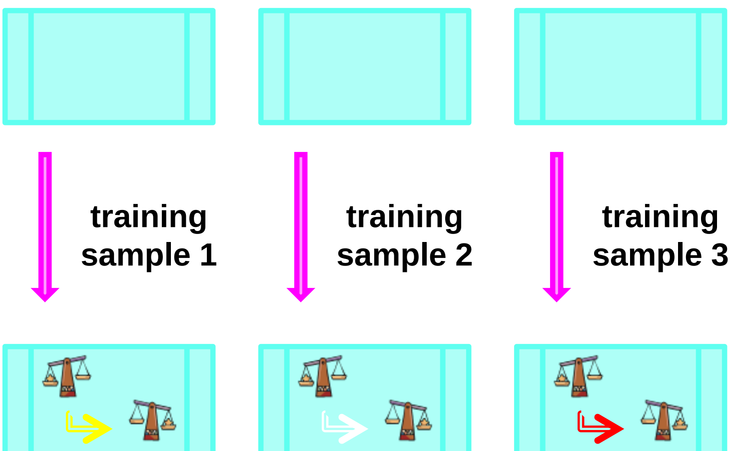
Building and using a committee ensemble



Building and using a committee ensemble

TRAINING

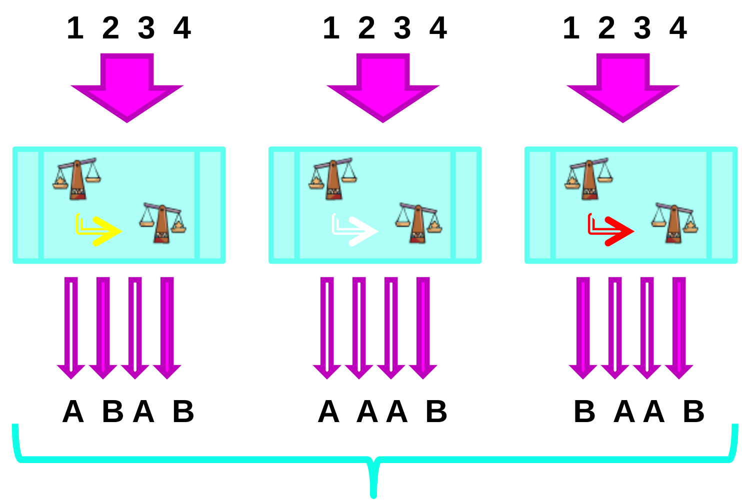
1) Create samples of training data
2) Train one base classifier on each sample



USING

1) Make predictions with each base classifier separately
2) Combine predictions by voting

Test or new data



$1 \rightarrow A$ $\quad 2 \rightarrow A$ $\quad 3 \rightarrow A$ $\quad 4 \rightarrow B$

Binomial distribution (a digression)

- The most commonly used discrete probability distribution.
- Givens:
  - a random process with two outcomes, referred to as *success* and *failure* (just a convention)
  - the probability p that outcome is success
    - probability of failure = $1 - p$
  - n trials of the process
- Binomial distribution describes probabilities that m of the n trials are successes, over values of m in range $0 \leq m \leq n$

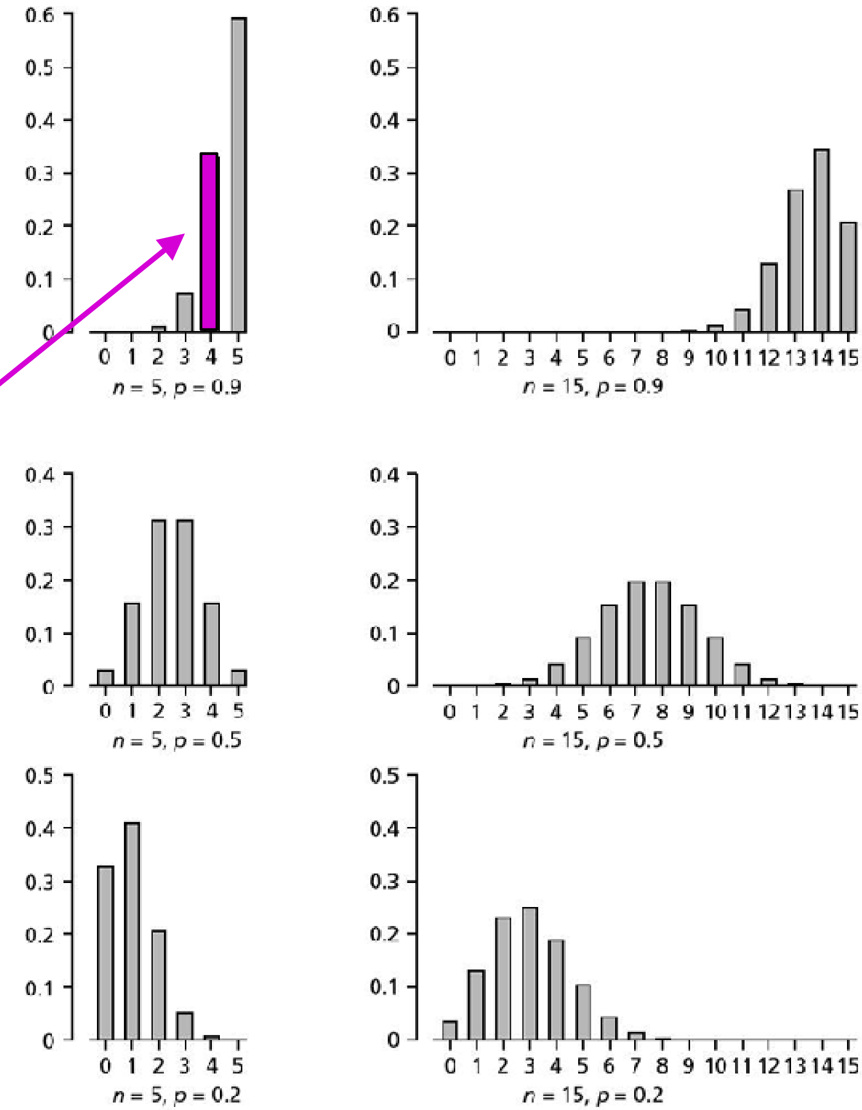
Binomial distribution

$$p(m\ \text{successes}) = \binom{n}{m} p^m (1-p)^{n-m}$$

Example:

$p = 0.9$, $n = 5$, $m = 4$

$$p(4\ \text{successes}) = \binom{5}{4} 0.9^4 0.1^1 = 0.328$$



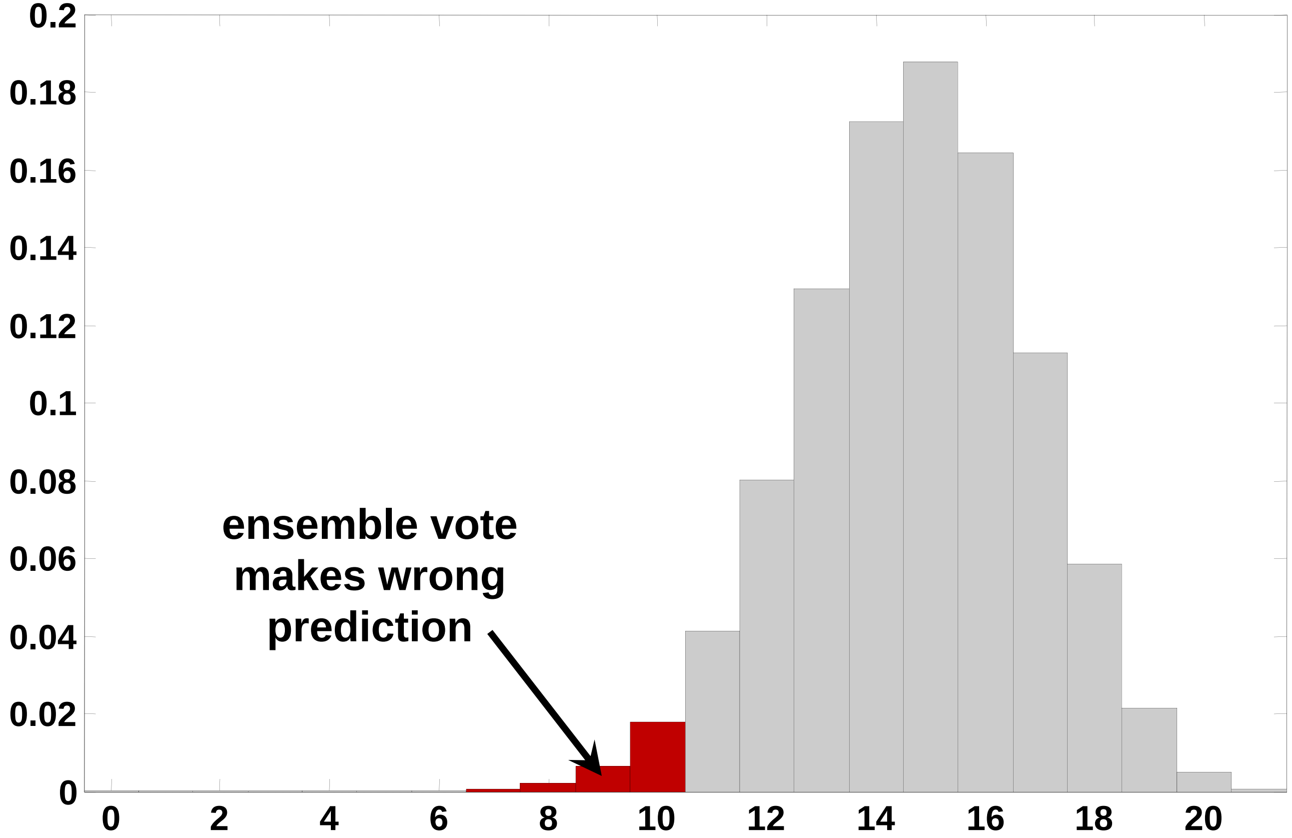
Why do ensembles work?

- A highly simplified example …
  - Suppose there are 21 base classifiers
  - Each classifier is correct with probability $p = 0.70$
  - Assume classifiers are independent
  - Probability that the *ensemble* classifier makes a correct prediction:

$$\sum_{i=11}^{21} \binom{21}{i} p^i (1-p)^{21-i} = 0.97$$

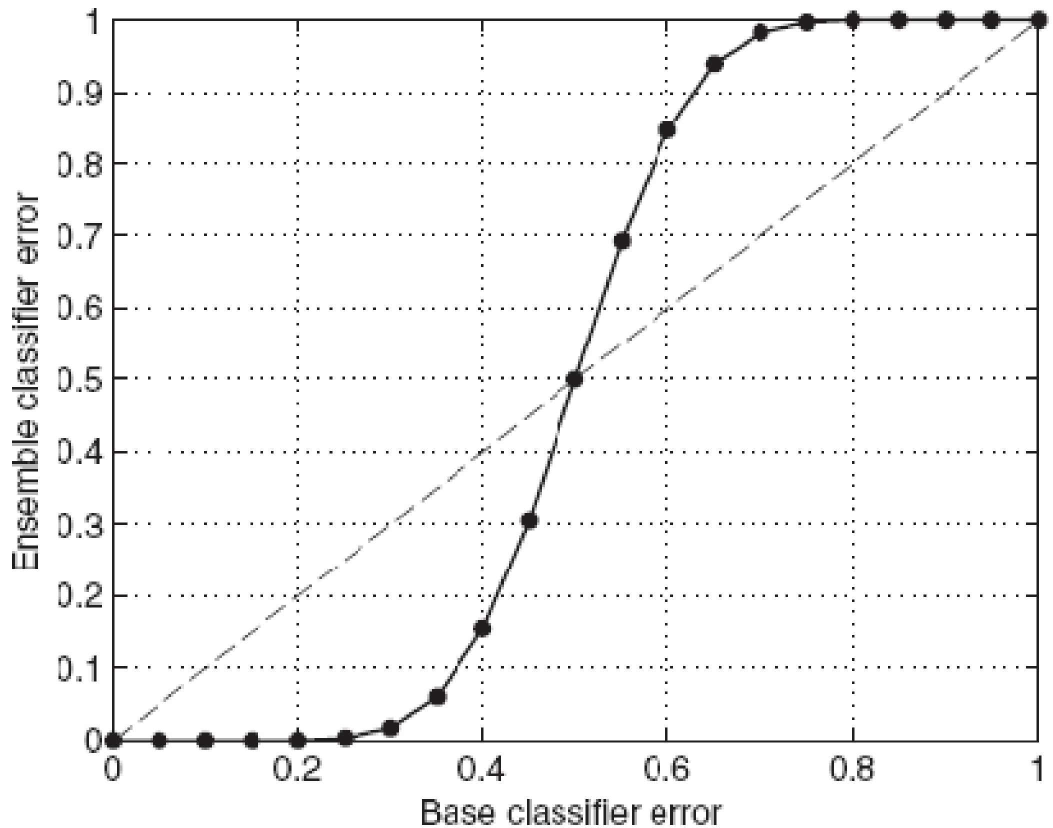
Why do ensembles work?

Voting by 21 independent classifiers, each correct with p = 0.7



Probability that exactly k of 21 classifiers will make be correct, assuming each classifier is correct with p = 0.7 and makes predictions independently of other classifiers

Ensemble vs. base classifier error



As long as base classifier is better than random (error < 0.5), ensemble will be superior to base classifier

Why do ensembles work?

- In real applications …
  - "Suppose there are 21 base classifiers …"
    - You do have direct control over the number of base classifiers.
  - "Each classifier is correct with probability $p = 0.70$ …"
    - Base classifiers will have variable accuracy, but you can establish *post hoc* the mean and variability of the accuracy.
  - "Assume classifiers are independent …"
    - Base classifiers always have some significant degree of correlation in their predictions.

Why do ensembles work?

- In real applications …
  - "Assume classifiers are independent …"
    - Base classifiers always have some significant degree of correlation in their predictions.
  - But the expected performance of the ensemble is guaranteed to be no worse than the average of the individual classifiers:

$$E[\text{ error }]_{\text{ave}} / m \leq E[\text{ error }]_{\text{committee}} \leq E[\text{ error }]_{\text{ave}}$$

$\Rightarrow$ The more uncorrelated the individual classifiers are, the better the ensemble.

Base classifiers: important properties

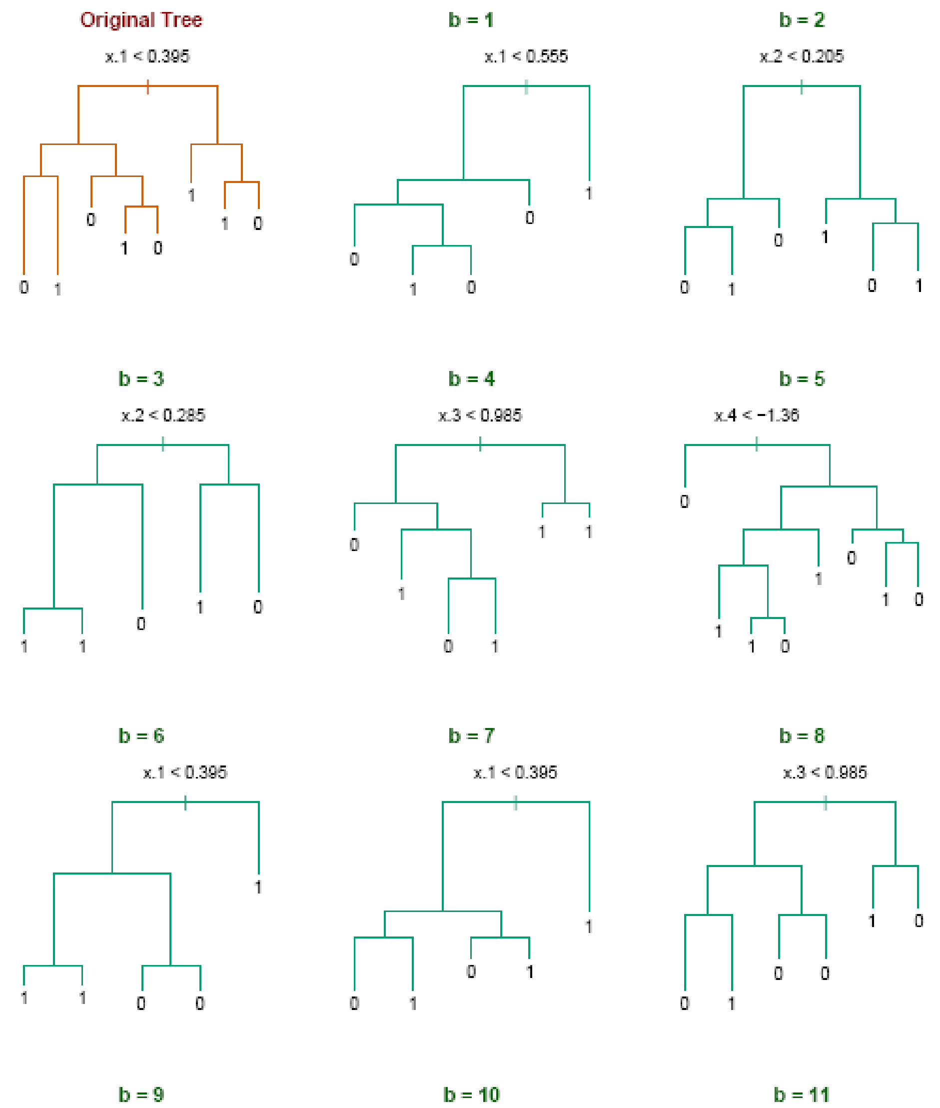
- Diversity (lack of correlation)
- Accuracy
- Computationally fast

Base classifiers: important properties

Diversity

- Predictions vary significantly between classifiers
- Usually attained by using unstable classifier
  - small change in training data (or initial model weights) produces large change in model structure
- Examples of unstable classifiers:
  - decision trees
  - neural nets
  - rule-based
- Examples of stable classifiers:
  - linear models: logistic regression, linear discriminant, etc.

Diversity in decision trees



- Bagging trees on simulated dataset.
  - Top left panel shows original tree.
  - Eight of trees grown on bootstrap samples are shown.

Base classifiers: important properties

Accurate

- Error rate of each base classifier better than random

Tension between diversity and accuracy

Computationally fast

- Usually need to compute large numbers of classifiers

How to create diverse base classifiers

- Random initialization of model parameters
  - Network weights
- Resample / subsample training data
  - Sample instances
    - Randomly with replacement (e.g. bagging)
    - Randomly without replacement
    - Disjoint partitions
  - Sample features (random subspace approach)
    - Randomly prior to training
    - Randomly during training (e.g. random forest)
  - Sample both instances and features
- Random projection to lower-dimensional space
- Iterative reweighting of training data

Common ensemble methods

- Bagging
- Boosting

Bootstrap sampling

- Given: a set S containing N samples
- Goal: a sampled set T containing N samples
- Bootstrap sampling process:

  for $i = 1$ to N

  – randomly select from S one sample *with replacement*

  – place sample in T
- If S is large, T will contain ~ (1 - 1 / e) = 63.2% *unique* samples.

Bagging

- Bagging = bootstrap + aggregation

1. Create *k* bootstrap samples.

   Example:

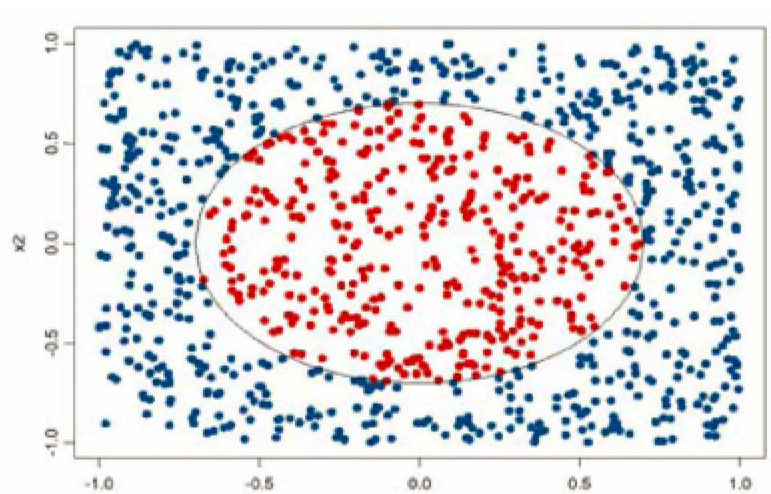
original data	1	2	3	4	5	6	7	8	9	10

bootstrap 1	7	8	10	8	2	5	10	10	5	9
bootstrap 2	1	4	9	1	2	3	2	7	3	2
bootstrap 3	1	8	5	10	5	5	9	6	3	7

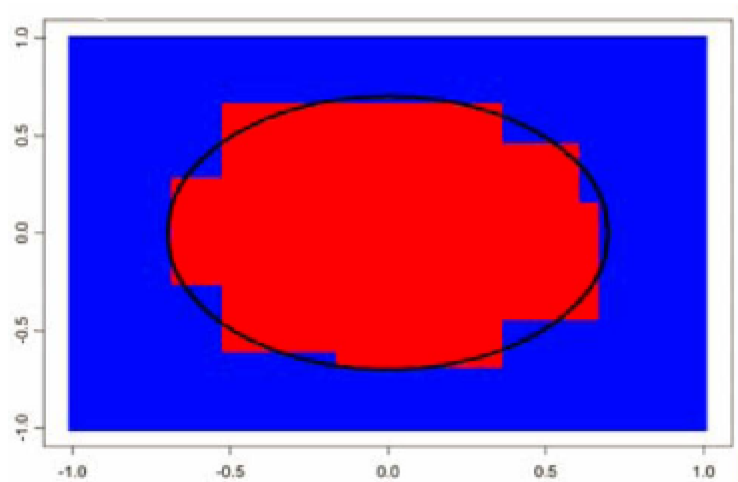
2. Train a classifier on each bootstrap sample.
3. Vote (or average) the predictions of the *k* models.

Bagging with decision trees

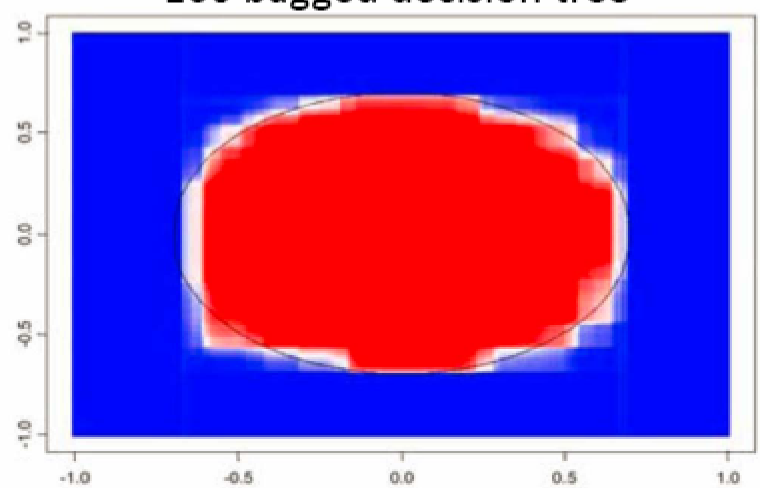
Target concept



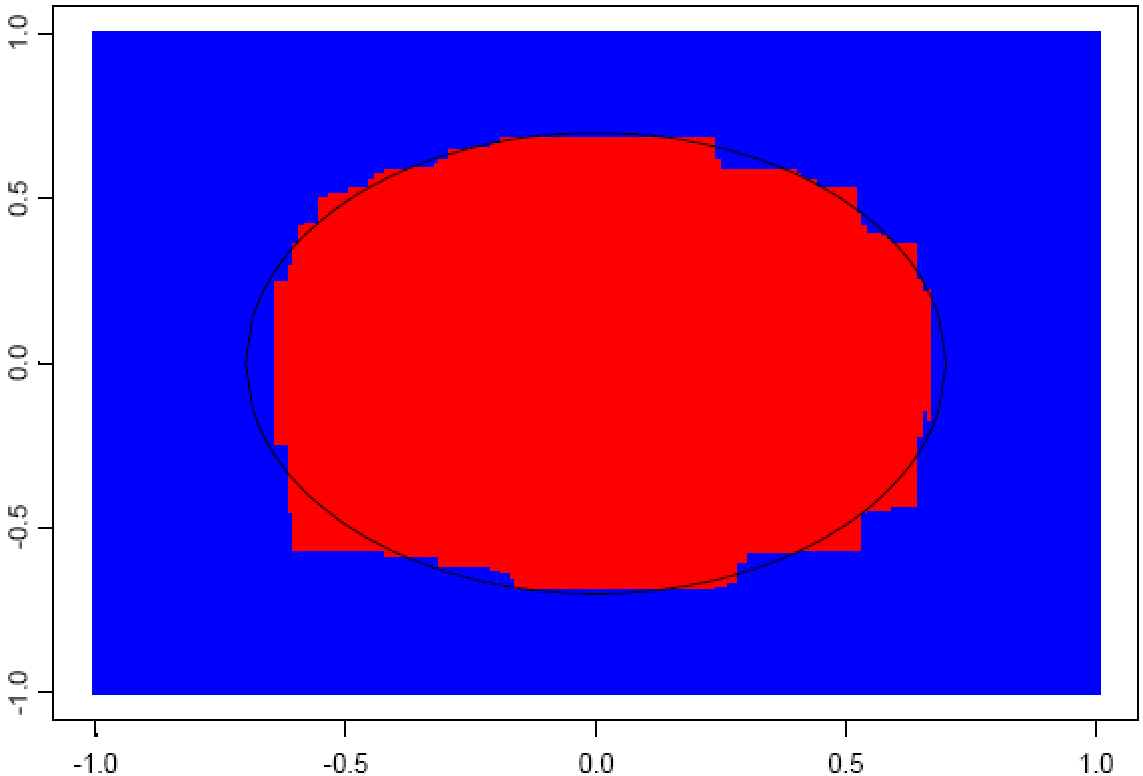
Single decision tree



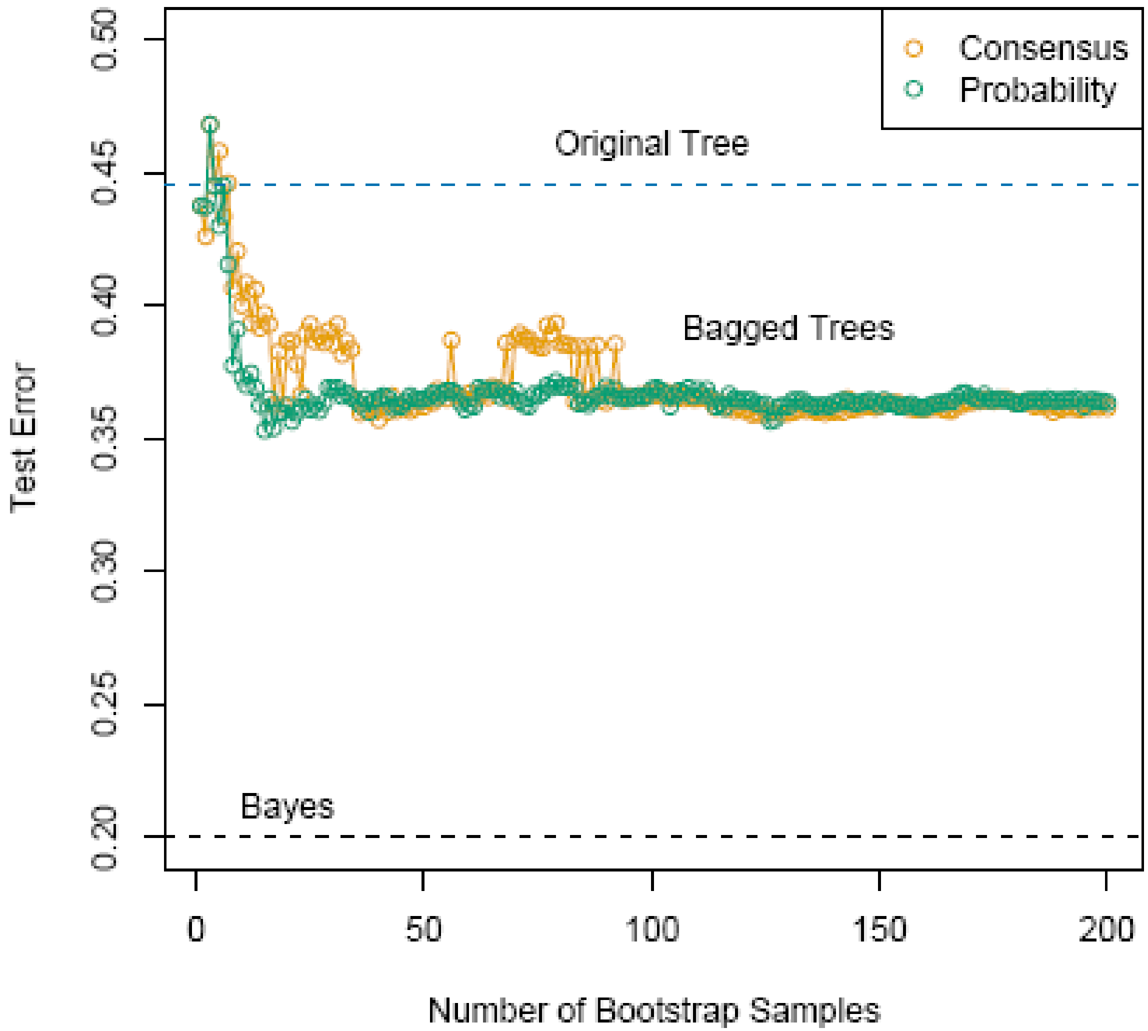
100 bagged decision tree



Bagged tree decision boundary



Bagging with decision trees



Boosting

- Key difference:
  - Bagging: individual classifiers trained independently.
  - Boosting: training process is <u>sequential</u> and <u>iterative</u>.
- Look at errors from previous classifiers to decide what to focus on in the next training iteration.
  - Each new classifier depends on its predecessors.
- Result: more weight on 'hard' samples (the ones where we committed mistakes in the previous iterations).

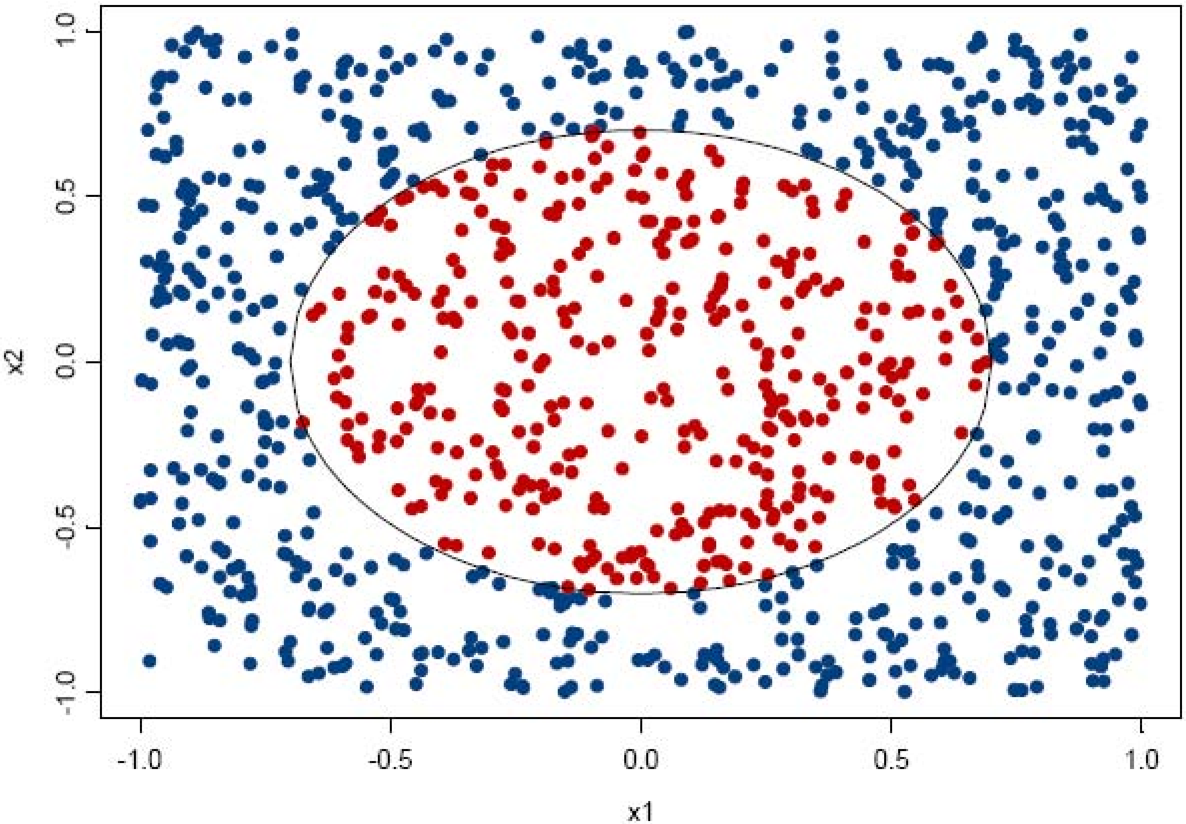
Boosting

- Initially, all samples have equal weights.
- Samples that are wrongly classified have their weights increased.
- Samples that are classified correctly have their weights decreased.
- Samples with higher weights have more influence in subsequent training iterations.
  - Adaptively changes training data distribution.

Original Data	1	2	3	4	5	6	7	8	9	10
Boosting (Round 1)	7	3	2	8	7	9	4	10	6	3
Boosting (Round 2)	5	4	9	4	2	5	1	7	4	2
Boosting (Round 3)	4	4	8	10	4	5	4	6	3	4

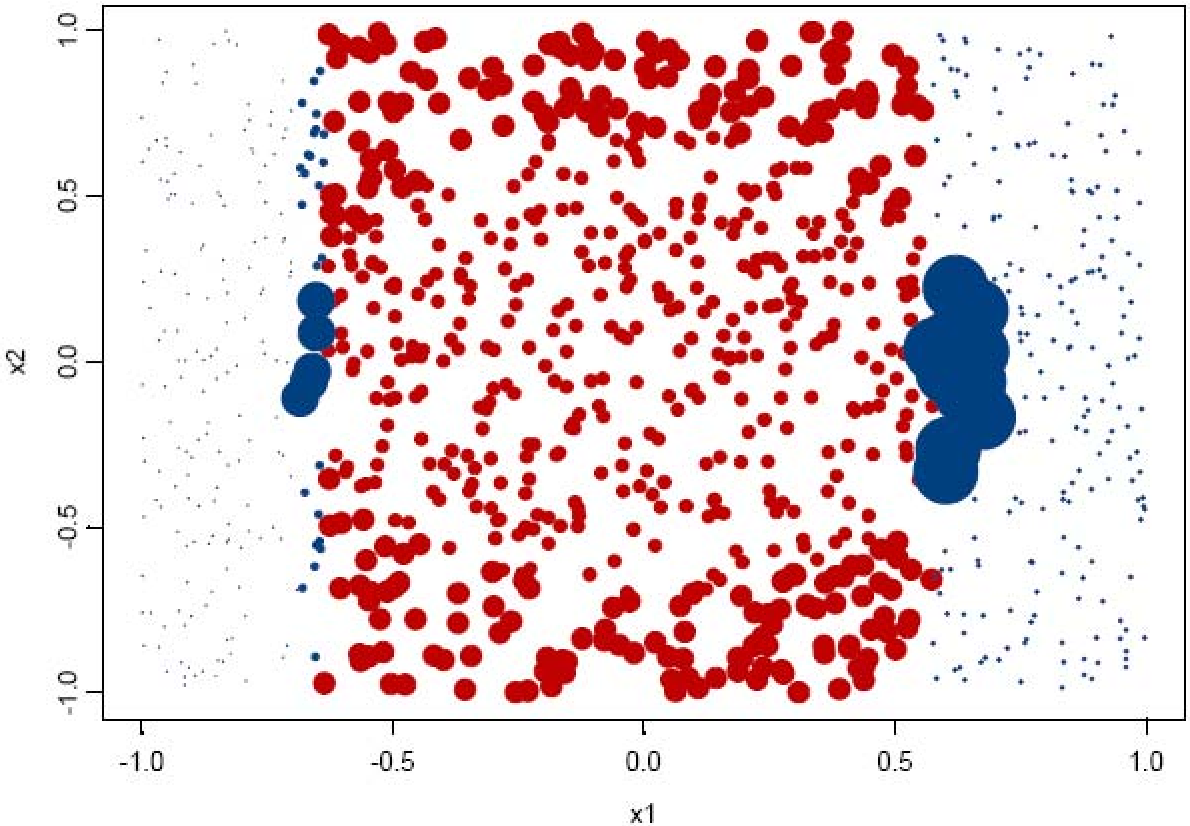
sample 4 is hard to classify → its weight is increased

Boosting Example

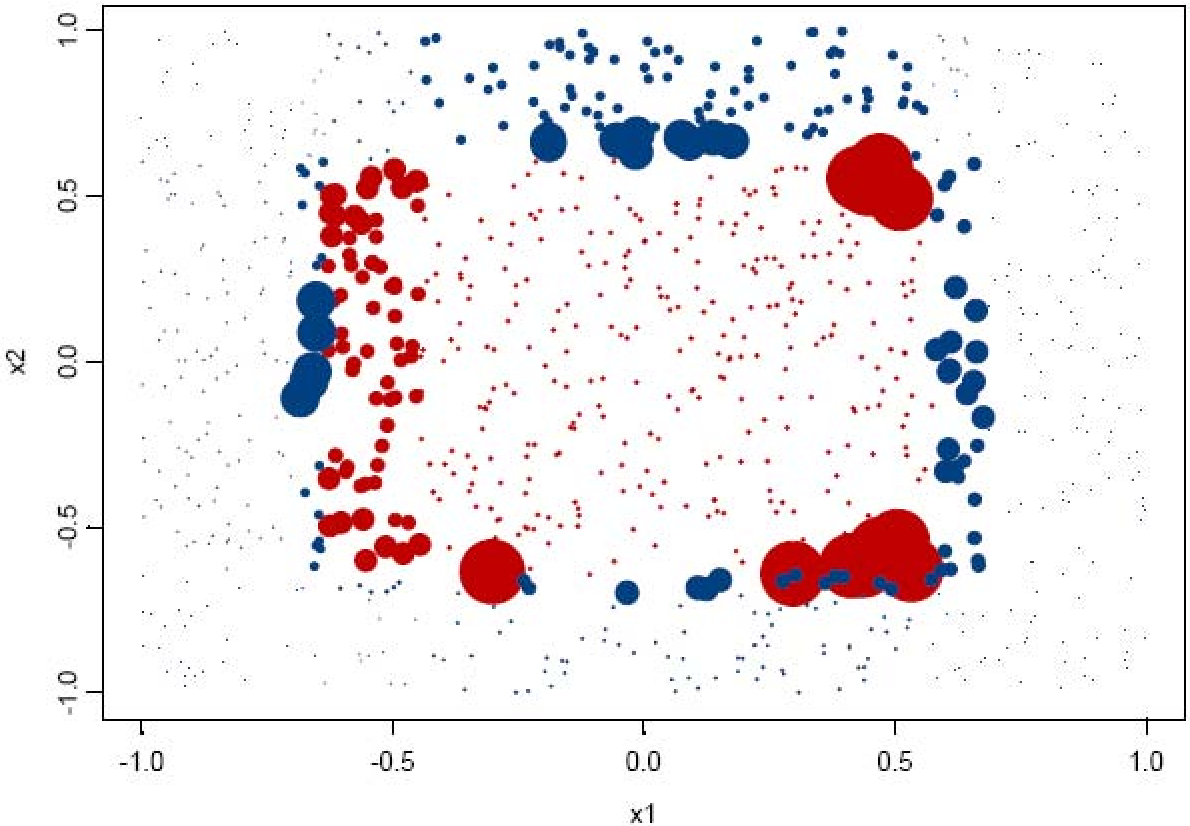


After one iteration

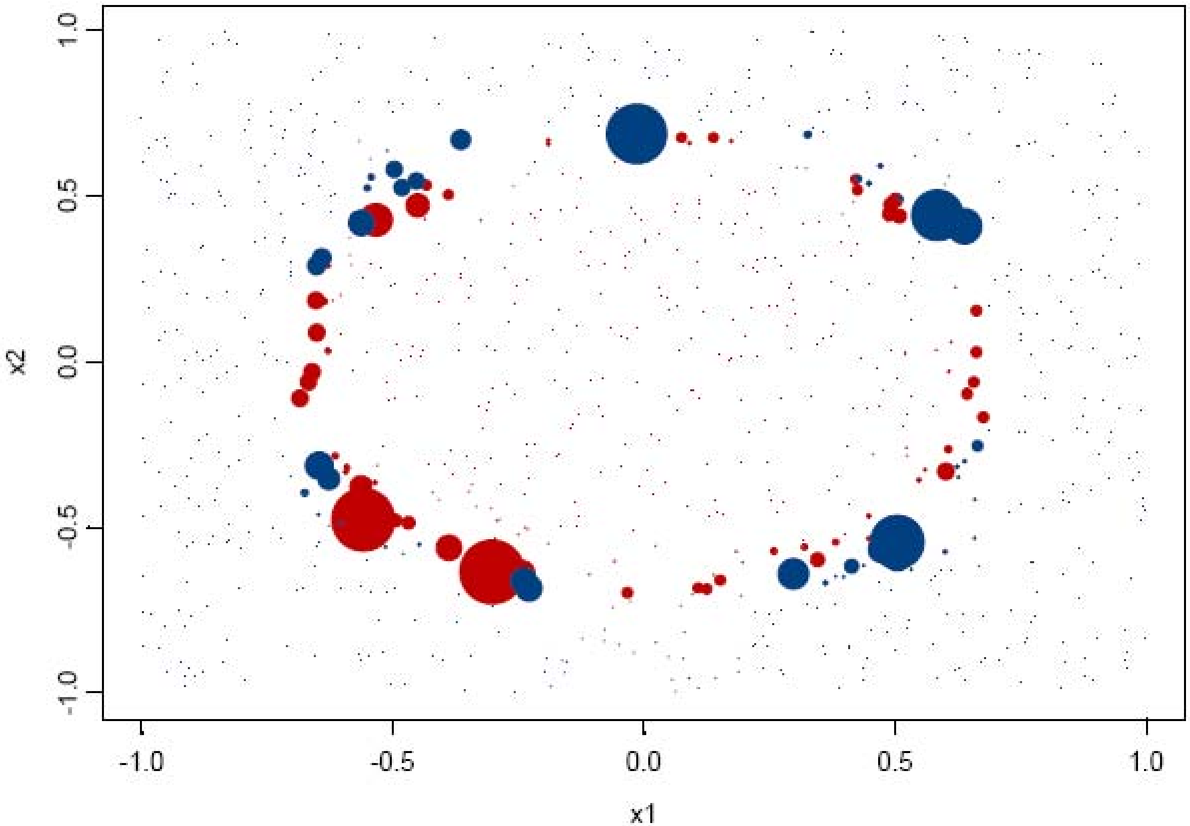
CART splits, larger points have great weight



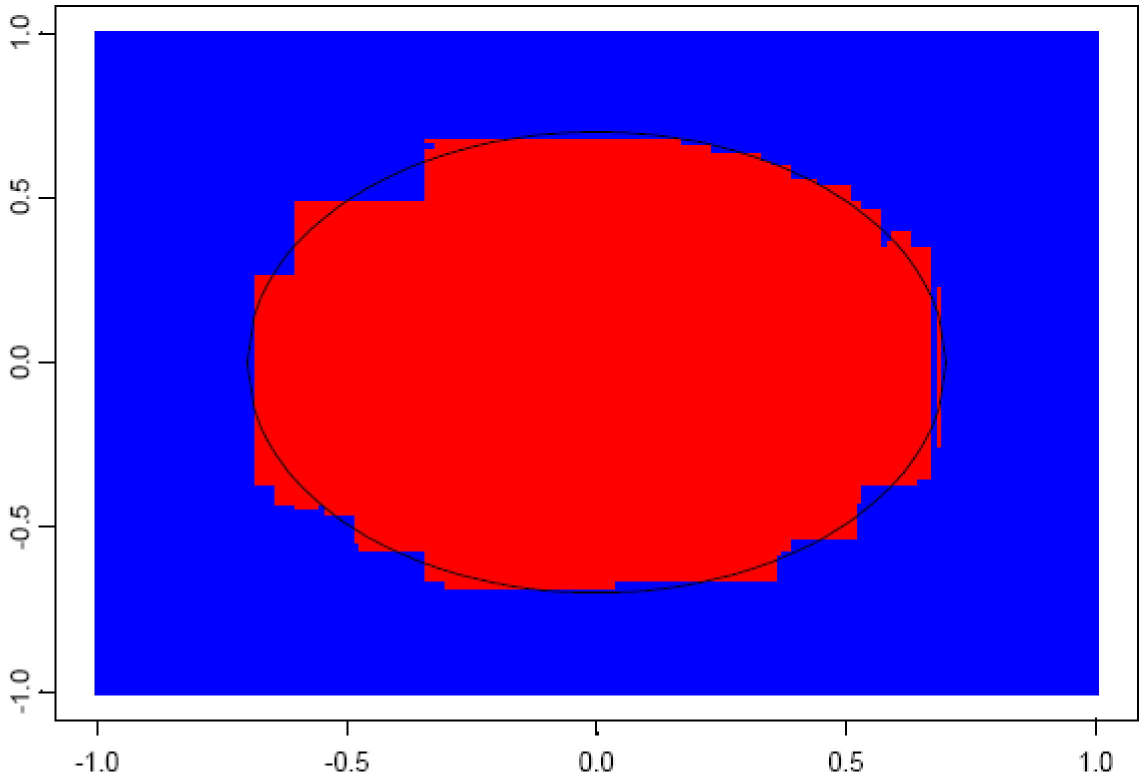
After 3 iterations



After 20 iterations



Decision boundary after 100 iterations



AdaBoost

- Training data has N samples
- K base classifiers: $C_1, C_2, \ldots, C_K$
- Error rate ε_i on i^{th} classifier:

$$\varepsilon_i = \frac{1}{N} \sum_{j=1}^{N} w_j \delta\left(C_i(x_j) \neq y_j\right)$$

where

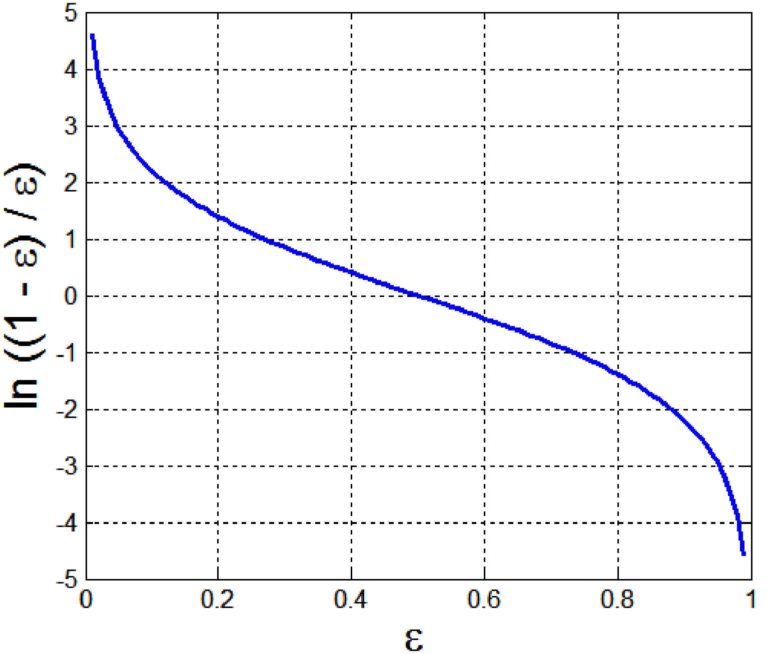
- w_j is the weight on the j^{th} sample
- δ is the indicator function for the j^{th} sample
  - $\delta(C_i(x_j) = y_j) = 0$ (no error for correct prediction)
  - $\delta(C_i(x_j) \neq y_j) = 1$ (error = 1 for incorrect prediction)

AdaBoost

- Importance of classifier *i* is:

$$\alpha_i = \frac{1}{2}\ln\left(\frac{1-\varepsilon_i}{\varepsilon_i}\right)$$

- α_i is used in:
  - formula for updating sample weights
  - final weighting of classifiers in voting of ensemble



Relationship of classifier importance α to training error ε

AdaBoost

- Weight updates:

$$w_j^{(i+1)} = \frac{w_j^{(i)}}{Z_i} \begin{cases} \exp^{-\alpha_i} & \text{if } C_i(x_j) = y_j \\ \exp^{\alpha_i} & \text{if } C_i(x_j) \neq y_j \end{cases}$$

where Z_i is a normalization factor

- If any intermediate iteration produces error rate greater than 50%, the weights are reverted back to 1 / n and the reweighting procedure is restarted.

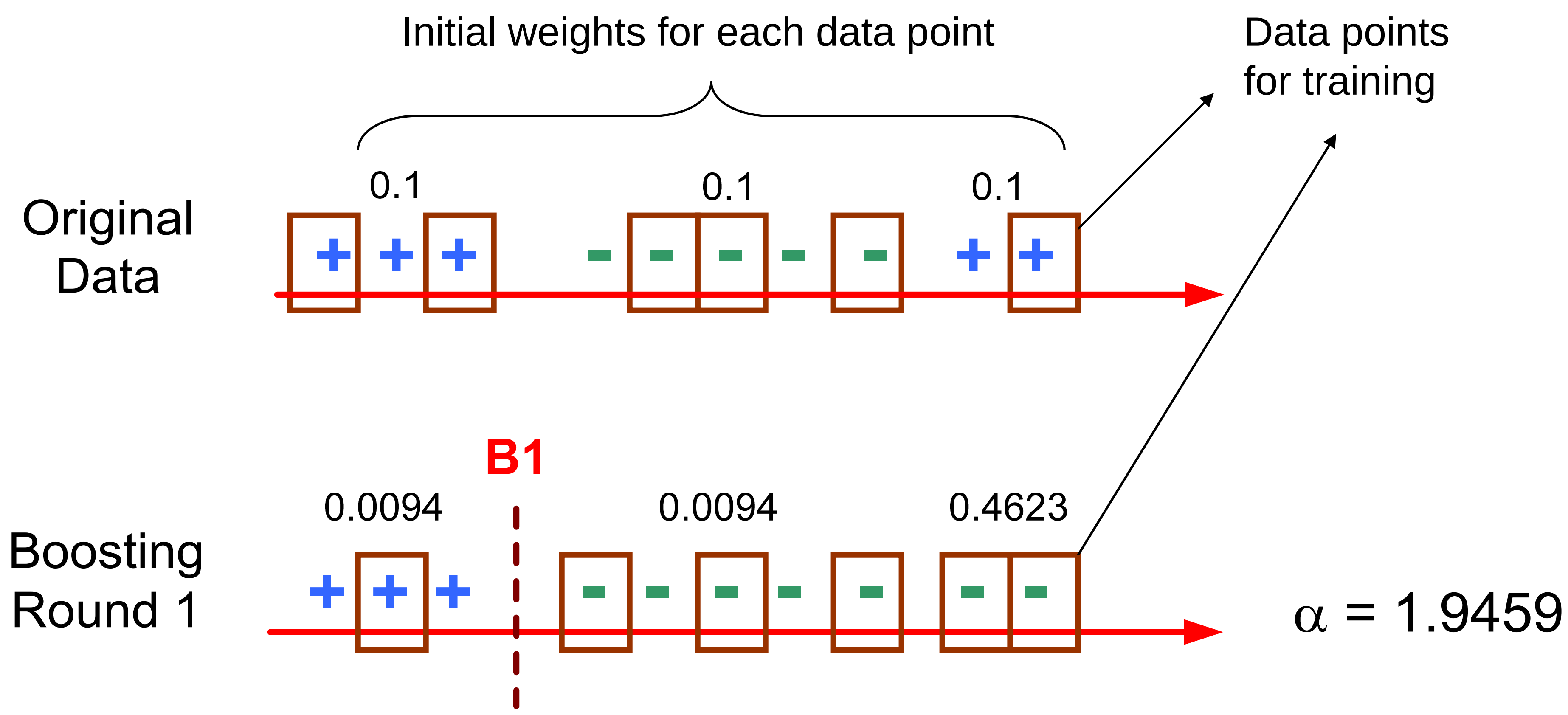
AdaBoost

- Final classification model:

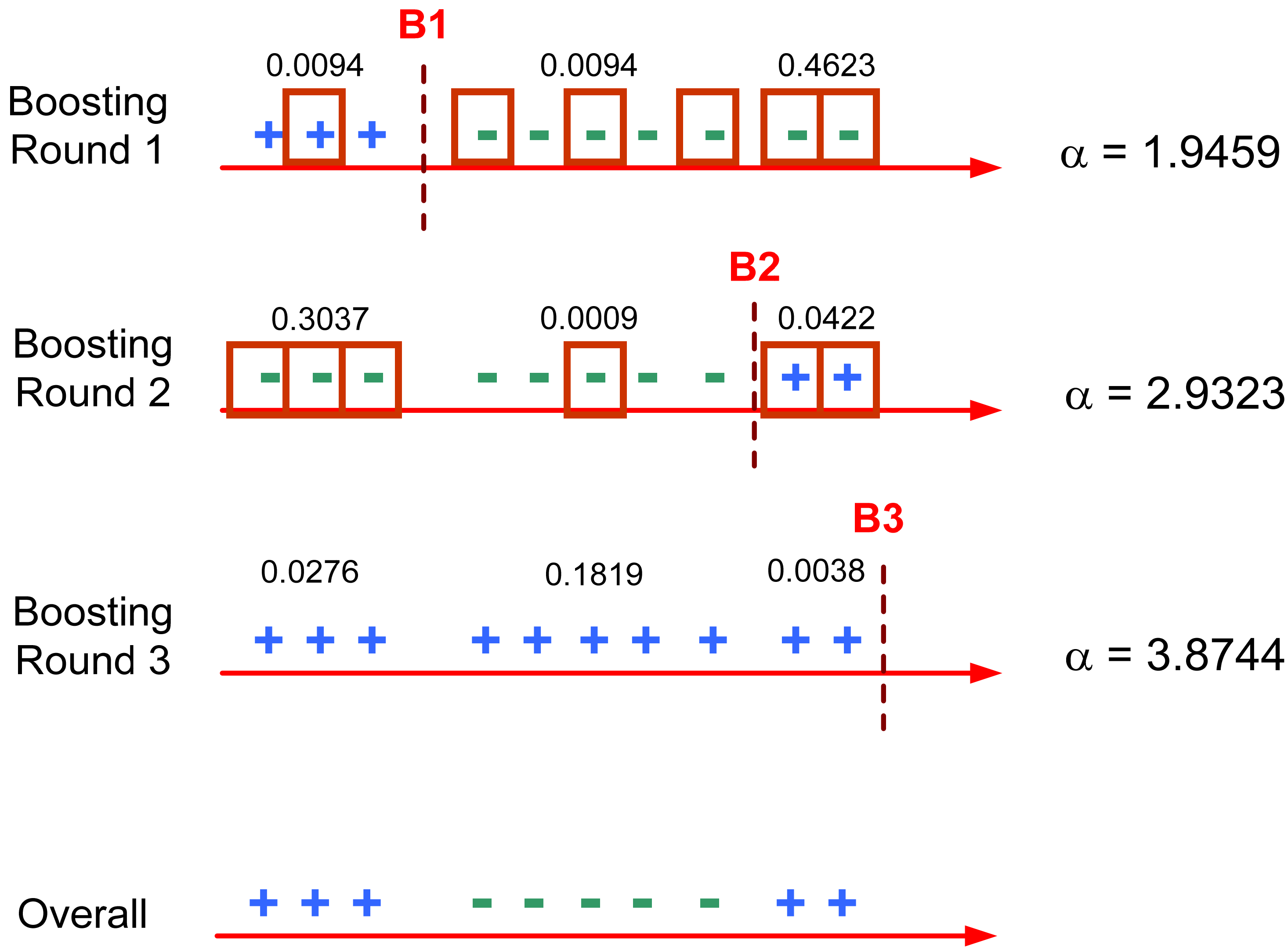
$$C^*(x) = \arg\max_y \sum_{i=1}^{K} \alpha_i \delta\left(C_i(x) = y\right)$$

i.e. for test sample *x*, choose the class label *y* which maximizes the importance-weighted vote across all classifiers.

Illustrating AdaBoost



Illustrating AdaBoost

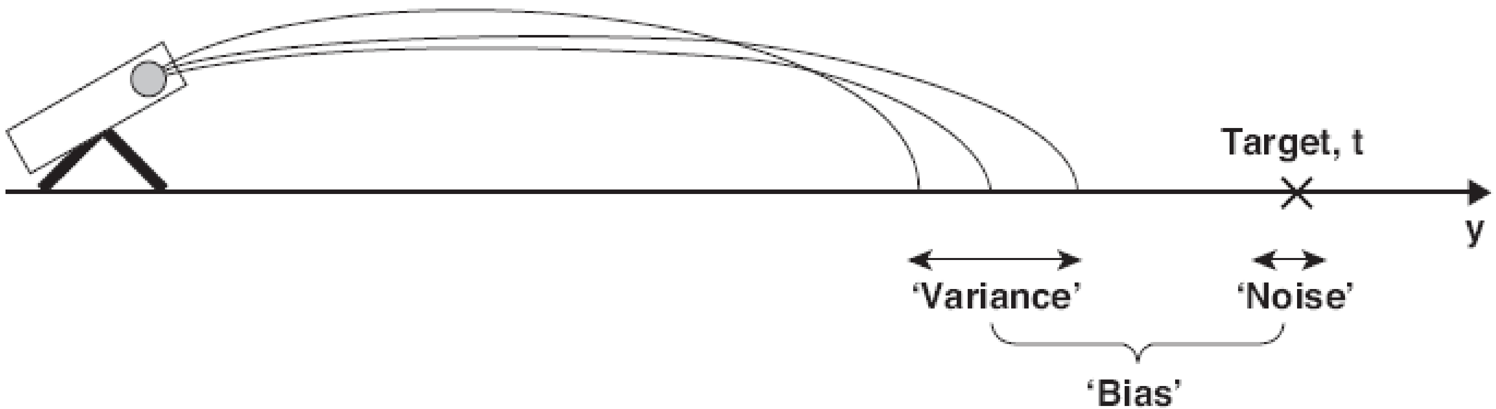


Summary: bagging and boosting

- Bagging
  - Resample data points
  - Weight of each classifier is same
  - Only reduces variance
  - Robust to noise and outliers
  - Easily parallelized
- Boosting
  - Reweight data points (modify data distribution)
  - Weight of a classifier depends on its accuracy
  - Reduces both bias and variance
  - Noise and outliers can hurt performance

Bias-variance decomposition

- An analogy from the Society for Creative Anachronism …



- Ideally, want to have low bias *and* low variance.

Bias-variance decomposition

expected error = bias^2 + variance + noise

where "expected" means the average behavior of the models trained on all possible samples of underlying distribution of data

Where

y(x; D) is the learned hypothesis given training set D

h(x) is the target underlying function

$$(\text{bias})^2 = \int \{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})] - h(\mathbf{x})\}^2 p(\mathbf{x})\,\mathrm{d}\mathbf{x}$$

$$\text{variance} = \int \mathbb{E}_{\mathcal{D}}\left[\{y(\mathbf{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})]\}^2\right] p(\mathbf{x})\,\mathrm{d}\mathbf{x}$$

$$\text{noise} = \iint \{h(\mathbf{x}) - t\}^2 p(\mathbf{x}, t)\,\mathrm{d}\mathbf{x}\,\mathrm{d}t$$

t(x) = h(x) + noise is the observed noisy target value

Bias-variance decomposition

- Examples of utility for understanding classifiers
  - Decision trees generally have low bias but high variance.
  - Bagging reduces the variance but not the bias of a classifier.

  $\Rightarrow$ Therefore expect decision trees to perform well in bagging ensembles.

Bias-variance decomposition

- General relationship to model complexity

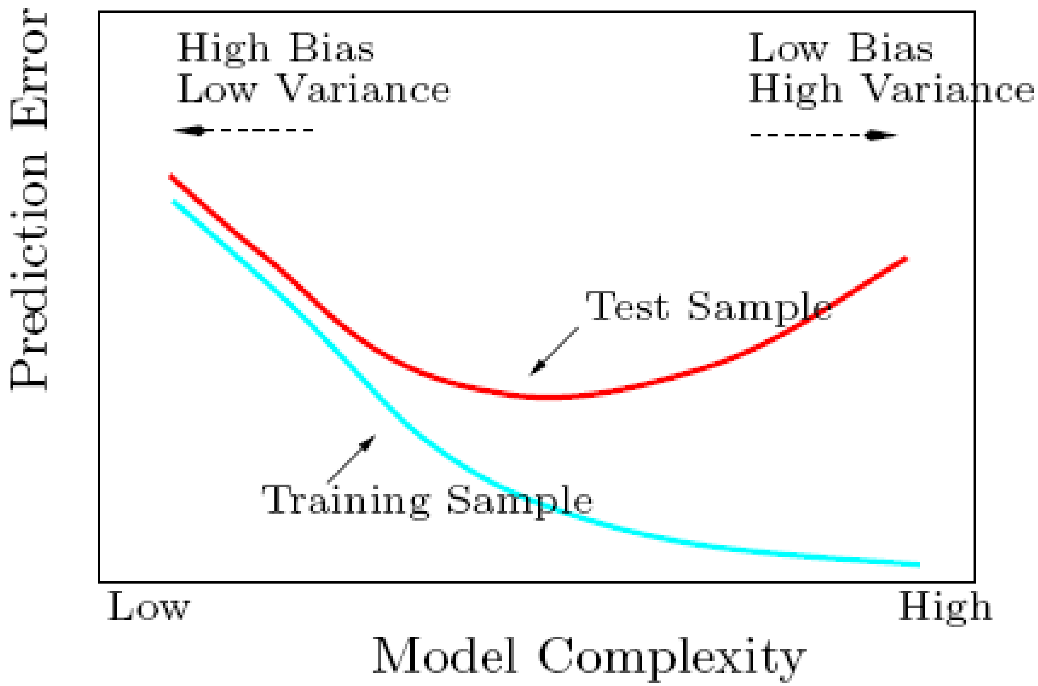


FIGURE 2.11. *Test and training error as a function of model complexity.*