Ensemble Learning

Machine Learning Group, TU Berlin

Outline

- Decision Trees
- Resampling schemes
- Random Forests / Bagging
- Boosting
- Relation to SVMs and Learning Theory

Classification

Many approaches derive from a theory

- LDA, QDA: Bayes decision theory
- SVM: VC theory

Classification

Many approaches derive from a theory

- LDA, QDA: Bayes decision theory
- SVM: VC theory

Some primarily motivated by intuition/common sense

- Nearest-neighbor rule
- Fisher criterion

Classification

Many approaches derive from a theory

- LDA, QDA: Bayes decision theory
- SVM: VC theory

Some primarily motivated by intuition/common sense

- Nearest-neighbor rule
- Fisher criterion

- Decision trees
- Ensemble methods

Decision trees

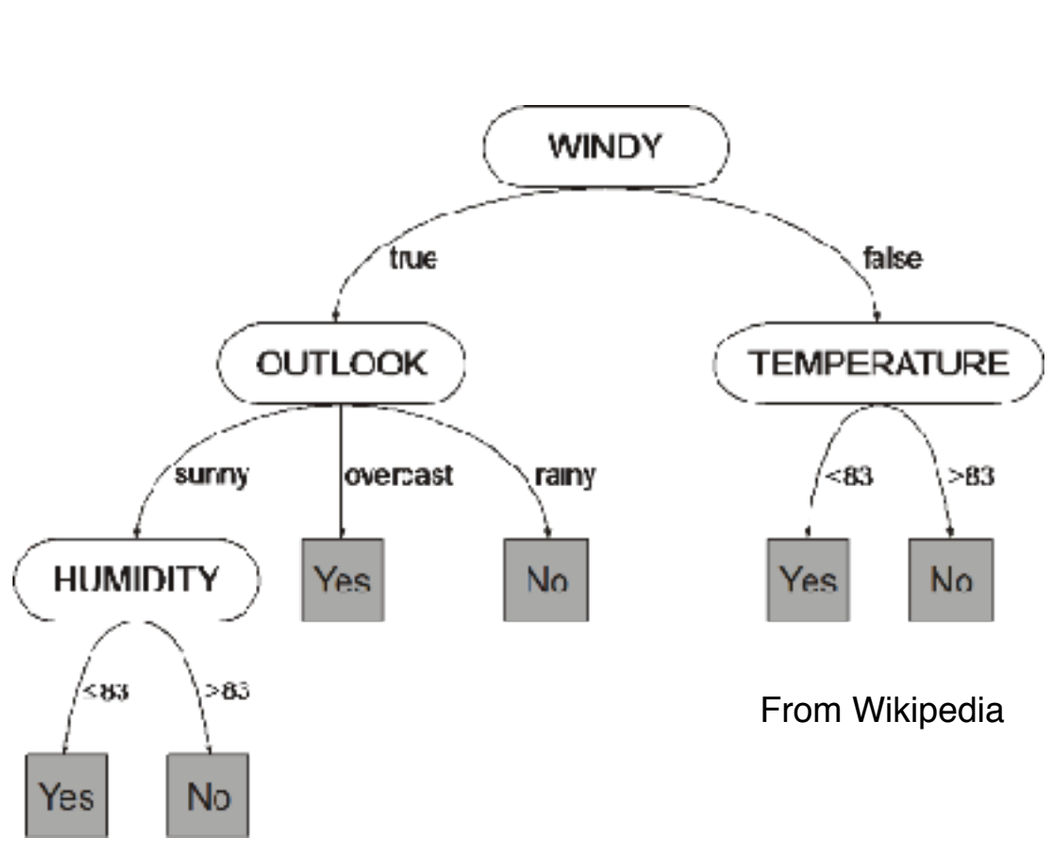
How do humans make decisions?

Decision trees

How do humans make decisions?

- Logical conjunction of many simple rules.
- Iterative procedure can be represented by a tree.

Play outside or not?



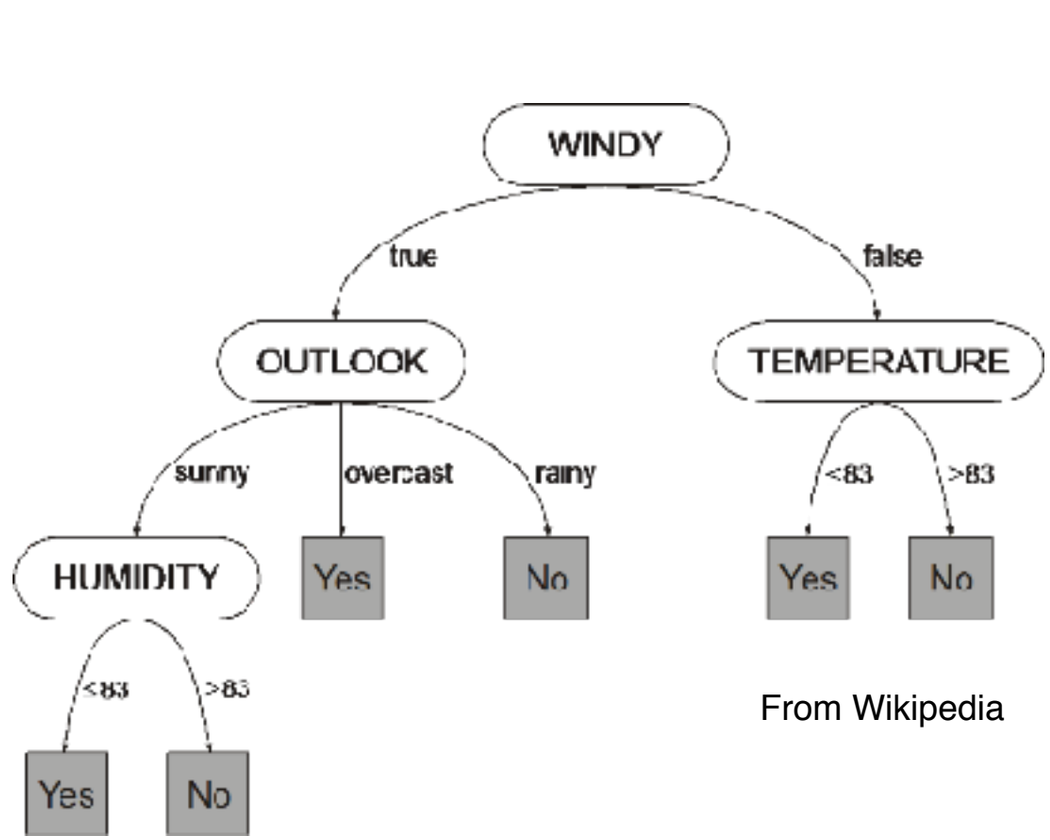
From Wikipedia

Decision trees

How do humans make decisions?

- Logical conjunction of many simple rules.
- Iterative procedure can be represented by a tree.

Play outside or not?



From Wikipedia

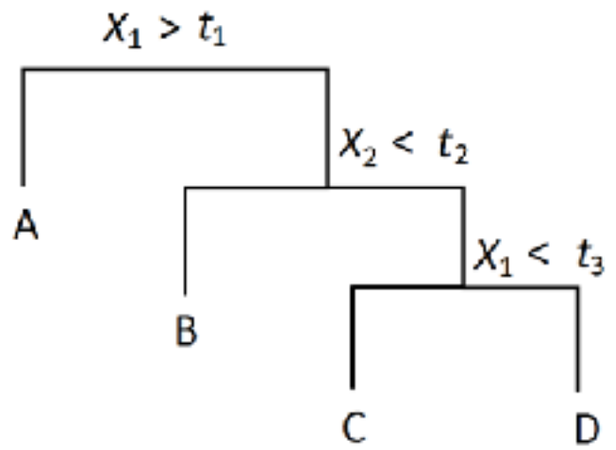
Decision trees

Formal definition

- Each leaf corresponds to a value of the target variable *y*
  - Categorial case: classification
  - Continuous case: regression

- Each interior node corresponds to a data dimension (feature) x_i
- Edges correspond to decisions
- Features may occur multiple times

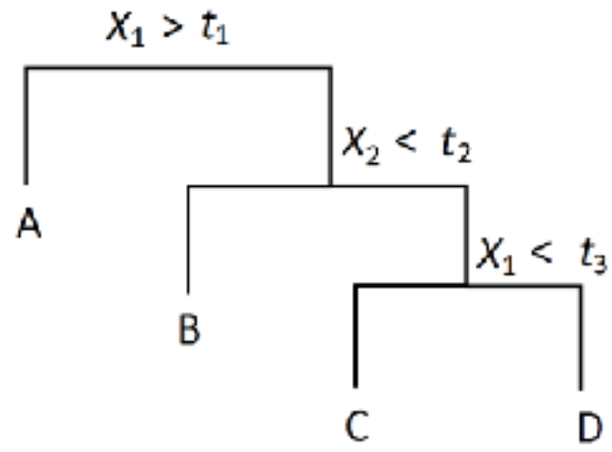
Defines recursive partitioning of the feature space.



From Wikipedia

Decision trees

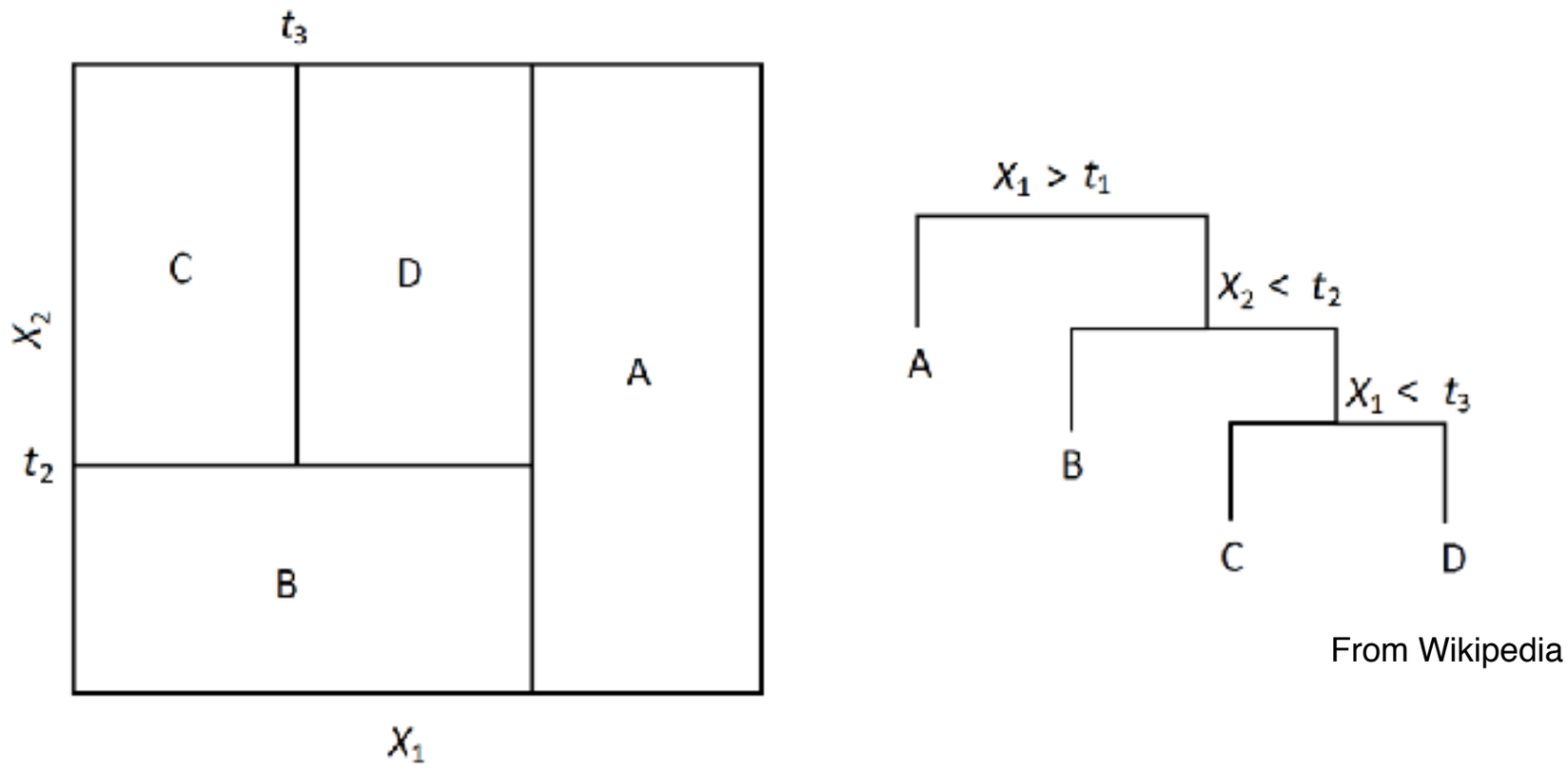
How does the decision boundary look like?



From Wikipedia

Decision trees

How does the decision boundary look like?

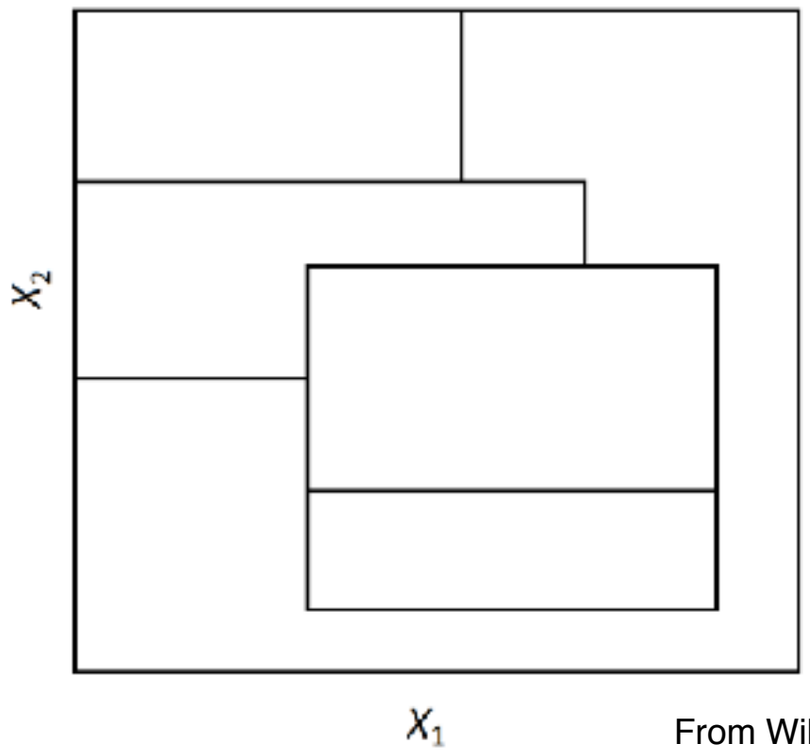


From Wikipedia

- Features treated separately → axes-parallel compartments.
- Nevertheless, can represent (arbitrary) complex functions.

Decision trees

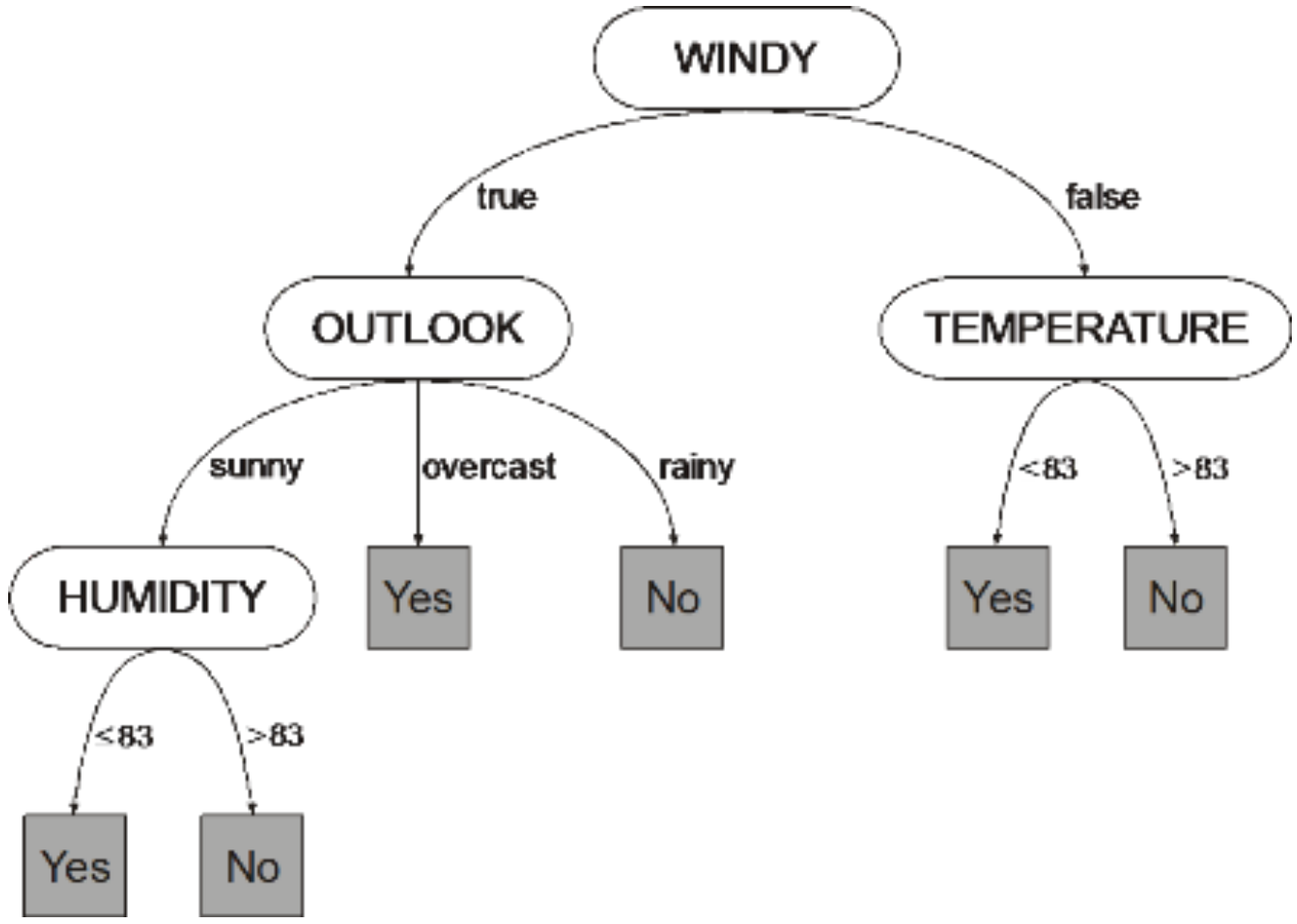
What tree represents these decision boundaries?



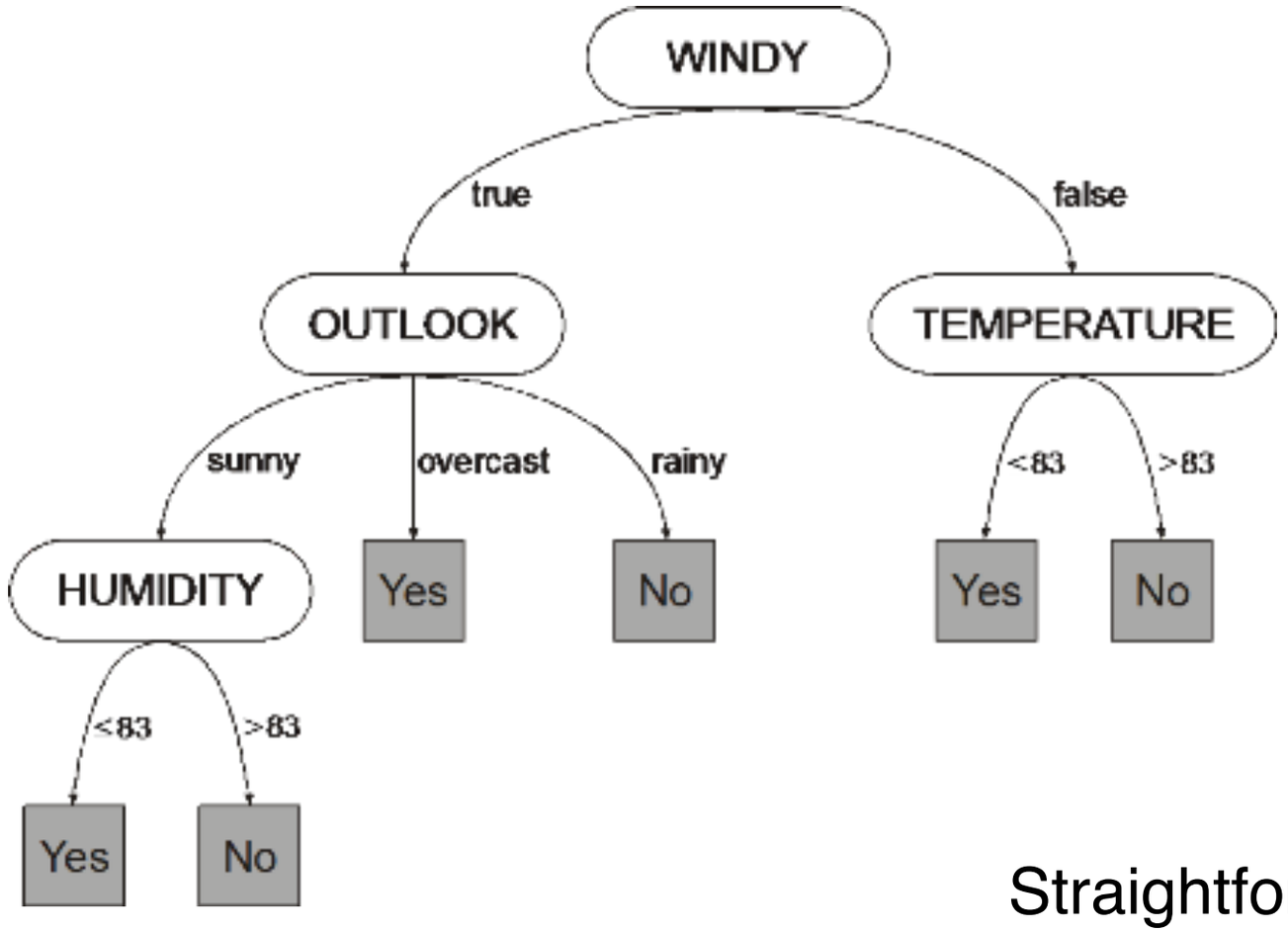
From Wikipedia

Possible to construct one, but requires many splits of the same features.

Predicting from a given tree



Predicting from a given tree



Straightforward

‘Growing’ a decision tree

- Many ways to choose splittings, order of variables.
- Low tree complexity leads to high generalization performance.

‘Growing’ a decision tree

- Many ways to choose splittings, order of variables.
- Low tree complexity leads to high generalization performance.
- Smallest tree that minimizes training error: NP hard.

‘Growing’ a decision tree

- Many ways to choose splittings, order of variables.
- Low tree complexity leads to high generalization performance.
- Smallest tree that minimizes training error: NP hard.

→ Greedy approaches

- ID3/C4.5
- CART

ID3

= Iterative Dichotomiser 3 (Ross Quinlan, 1986)

ID3

= Iterative Dichotomiser 3 (Ross Quinlan, 1986)

- Start with entire dataset $\mathcal{X}$ and features $x_1, \ldots, x_d$
- If all samples in $\mathcal{X}$ belong to class c: add leaf node for class c and stop.

ID3

= Iterative Dichotomiser 3 (Ross Quinlan, 1986)

- Start with entire dataset $\mathcal{X}$ and features $x_1, \ldots, x_d$
- If all samples in $\mathcal{X}$ belong to class c: add leaf node for class c and stop.
- Otherwise:
  - For each categorial feature x_i , split $\mathcal{X}$ into subsets $\mathcal{T}_k = \{\mathbf{x} \in \mathcal{X} | x_i = k\}$
  - Compute the information gain that is due to this splitting.
  - Select feature with largest information gain to split tree.

ID3

= Iterative Dichotomiser 3 (Ross Quinlan, 1986)

- Start with entire dataset $\mathcal{X}$ and features $x_1, \ldots, x_d$
- If all samples in $\mathcal{X}$ belong to class c: add leaf node for class c and stop.
- Otherwise:
  - For each categorial feature x_i , split $\mathcal{X}$ into subsets $\mathcal{T}_k = \{\mathbf{x} \in \mathcal{X} | x_i = k\}$
  - Compute the information gain that is due to this splitting.
  - Select feature with largest information gain to split tree.
  - Recurse on subsets $\mathcal{T}_k$ using remaining features.
  - If no more feature exists, add leaf node for majority class and stop.

Splitting criterion

Entropy: measures the uncertainty in a set $\mathcal{X}$

$$H(\mathcal{X}) = -\sum_{c \in C} p(c) \log p(c) \qquad \text{, where}$$

- C is the set of classes occurring in $\mathcal{X}$
- p(c) is the proportion of samples $\mathcal{X}$ that are in class c
- Low Entropy: all samples in the same class
- High Entropy: uniform distribution of classes

Splitting criterion

Entropy: measures the uncertainty in a set $\mathcal{X}$

$$H(\mathcal{X}) = -\sum_{c \in C} p(c) \log p(c) \quad \text{, where}$$

- C is the set of classes occurring in $\mathcal{X}$
- p(c) is the proportion of samples $\mathcal{X}$ that are in class c
- Low Entropy: all samples in the same class
- High Entropy: uniform distribution of classes

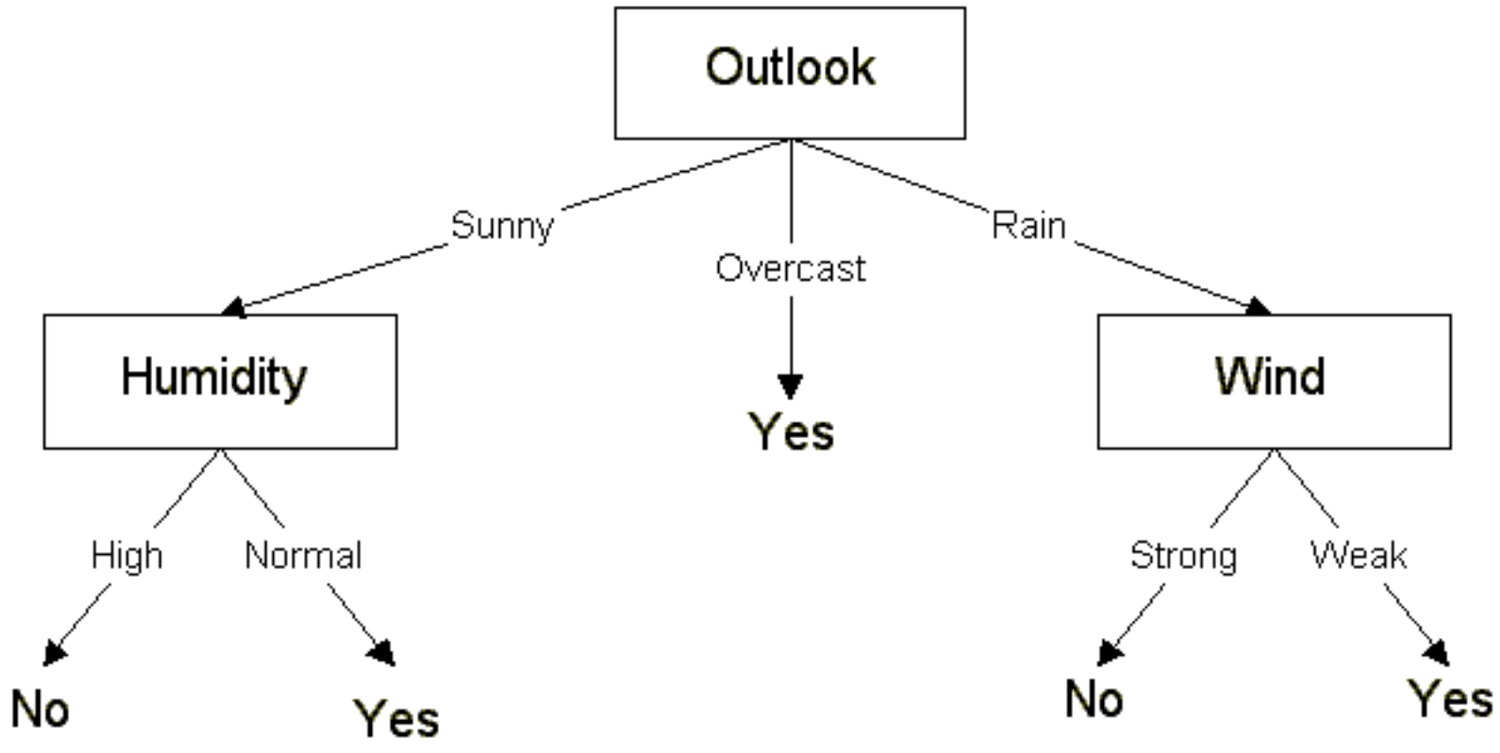
Information Gain:

$$IG(x_i, \mathcal{X}) = H(\mathcal{X}) - \sum_{k} p(\mathcal{T}_k) H(\mathcal{T}_k) \quad \text{, where}$$

- $p(\mathcal{T}_k)$ is the proportion of samples in $\mathcal{X}$ that are in $\mathcal{T}_k$.

Example

https://www.cise.ufl.edu/~ddd/cap6635/Fall-97/Short-papers/2.htm



C4.5

Extension of ID3 (Quinlan, 1993)

1. Handles continuous features by automatically setting a threshold base on maximal information gain.
2. Handles missing values.
3. Handles attributes with differing costs.
4. Pruning: Removes branches that do not help by replacing them with leaf nodes.
5. If no feature provides sufficient information gain, a leaf for the class of the majority is inserted instead of a split node.

C4.5

Extension of ID3 (Quinlan, 1993)

1. Handles continuous features by automatically setting a threshold base on maximal information gain.
2. Handles missing values.
3. Handles attributes with differing costs.
4. Pruning: Removes branches that do not help by replacing them with leaf nodes.
5. If no feature provides sufficient information gain, a leaf for the class of the majority is inserted instead of a split node.

   → 4 and 5 reduce overfitting.

CART

CART = Classification and Regression Trees (Leo Breiman, 1984)

- Only binary splits
- Different criteria for regression and classification

CART

CART = Classification and Regression Trees (Leo Breiman, 1984)

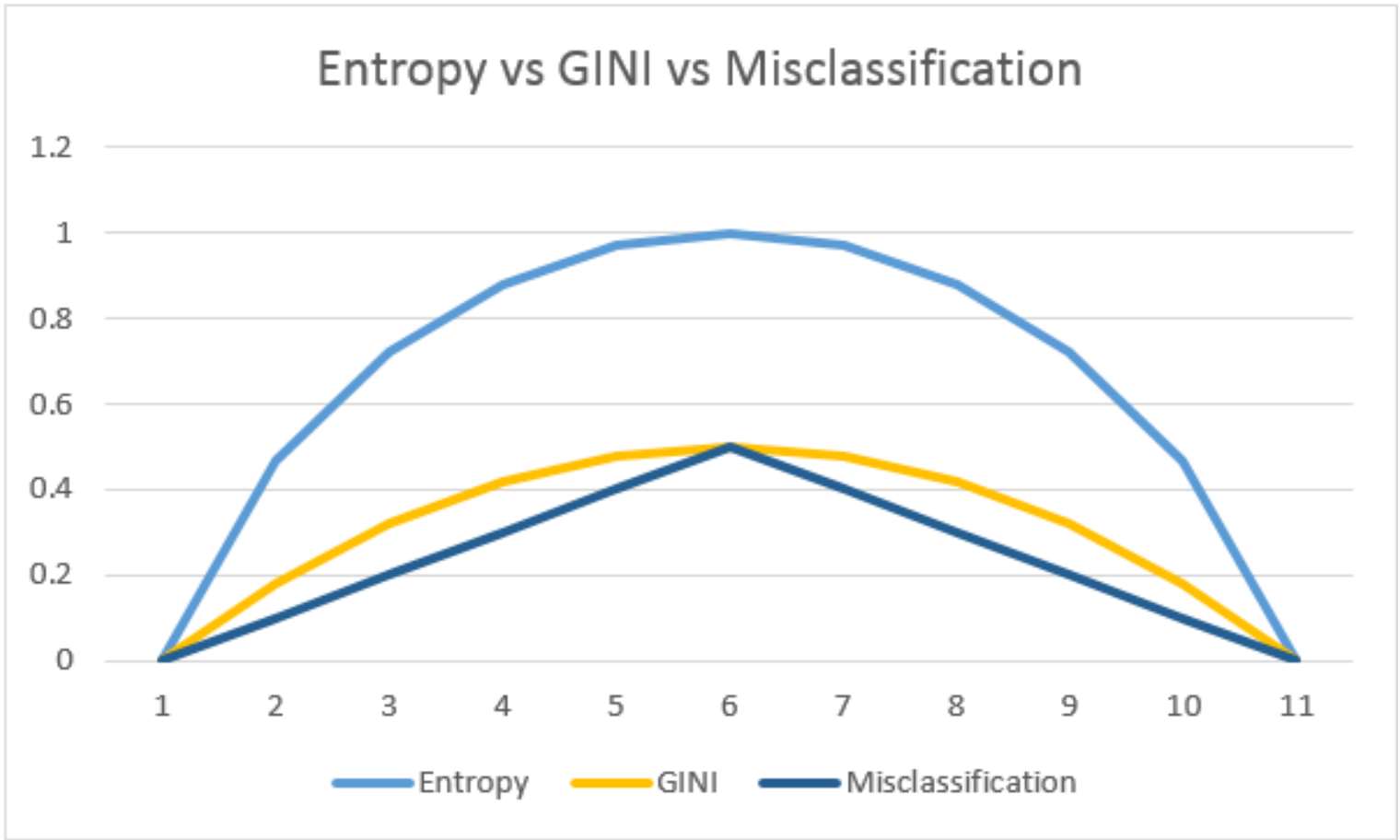
- Only binary splits
- Different criteria for regression and classification

Classification: Gini Impurity — measures how often a randomly chosen sample from $\mathcal{X}$ would be incorrectly labeled if it was randomly labeled according to the distribution of labels in $\mathcal{X}$.

$$GI(\mathcal{X}) = \sum_{c=1}^{C} p(c)(1 - p(c)) = \sum_{c \neq \tilde{c}} p(c)p(\tilde{c})$$

→ Similar to entropy, zero if all samples belong to same class.

CART


Entropy vs GINI vs Misclassification
1.2
1
0.8
0.6
0.4
0.2
0
1
2
3
4
5
6
7
8
9
10
11
Entropy
GINI
Misclassification

CART

Regression: Variance reduction

$$VR(\mathcal{X}, x_i, h) = \text{Var}(\mathcal{X}) - \left(p(\mathcal{T}_n)\text{Var}(\mathcal{T}_n) + p(\mathcal{T}_p)\text{Var}(\mathcal{T}_p)\right) \text{, where}$$

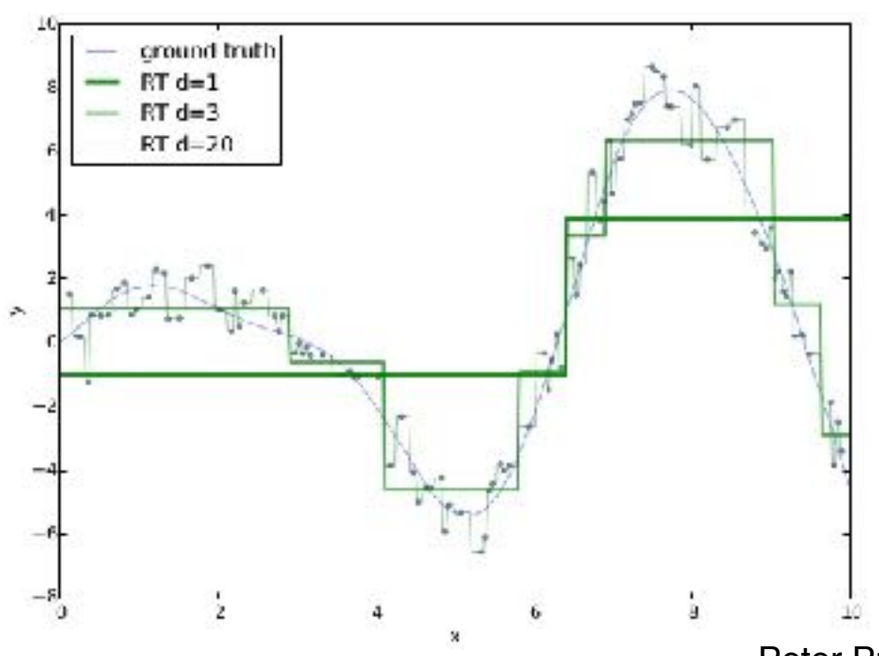
$$\mathcal{T}_n = \{\mathbf{x} \in \mathcal{X} | x_i \leq h\}, \mathcal{T}_p = \{\mathbf{x} \in \mathcal{X} | x_i > h\}$$

CART

Regression: Variance reduction

$$VR(\mathcal{X}, x_i, h) = \text{Var}(\mathcal{X}) - \left(p(\mathcal{T}_n)\text{Var}(\mathcal{T}_n) + p(\mathcal{T}_p)\text{Var}(\mathcal{T}_p)\right) \text{, where}$$

$$\mathcal{T}_n = \{\mathbf{x} \in \mathcal{X} | x_i \leq h\}, \mathcal{T}_p = \{\mathbf{x} \in \mathcal{X} | x_i > h\}$$



Peter Prettenhofer

Finiteness of tree leads to stepwise constant functions.

Advantages

- Simple to understand and interpret.
- Mirrors human decision making more closely than other approaches.
- Graphical representation.
- Non-parametric, few assumptions.
- Able to handle both numerical and categorical data.
- Requires little data preparation, no normalization/dummy variables.
- ‘White box’ model. Classification is easily explained by Boolean logic.
- Performs well with large datasets due to sequential univariate treatment.

Limitations

- Greedy, local optima.
- Non-robustness: small changes in training data and greedy strategy can change results.
- Overfitting, low test accuracy

  → Pruning
- Large trees needed to represent some simple functions (e.g. a sphere).

  → Multivariate Approaches (linear DT, PCA)
- No continuous regression.

Limitations

- Greedy, local optima.
- Non-robustness: small changes in training data and greedy strategy can change results.
- Overfitting, low test accuracy

  → Pruning

- Large trees needed to represent some simple functions (e.g. a sphere).

  → Multivariate Approaches (linear DT, PCA)

- No continuous regression.

→ One solution for all problems: ensemble methods

Ensemble methods

Idea: combine results of multiple estimators into ensemble estimate.

Ensemble methods

Idea: combine results of multiple estimators into ensemble estimate.

Expected benefits:

- Aggregate estimate has reduced variance but same bias
- Smooth functions, more ‘regular’ structure

Ensemble methods

Idea: combine results of multiple estimators into ensemble estimate.

Expected benefits:

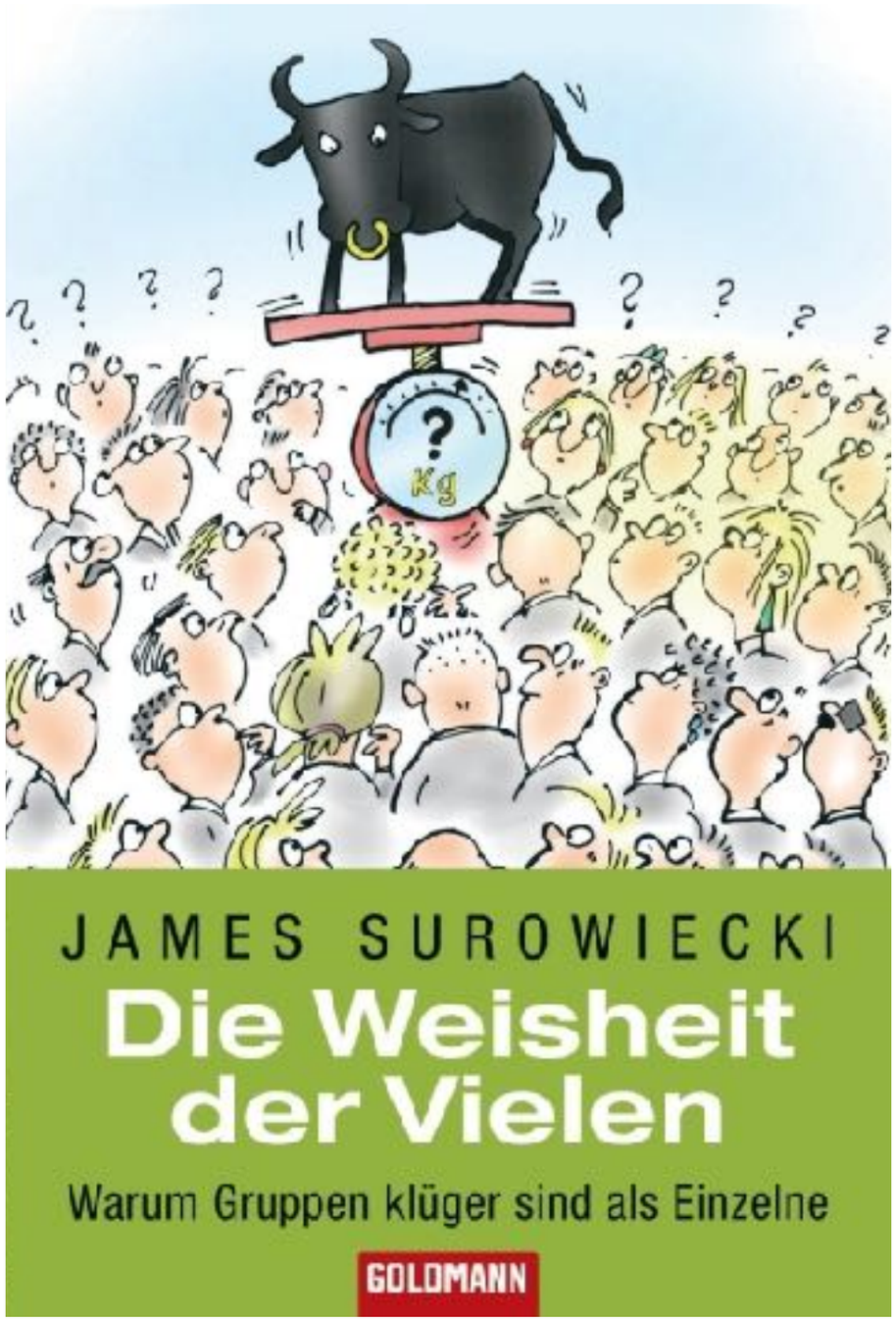
- Aggregate estimate has reduced variance but same bias
- Smooth functions, more ‘regular’ structure

Motivation: ‘The Wisdom of Crowds’

‘The Wisdom of Crowds’

Francis Galton Experiment (1906):

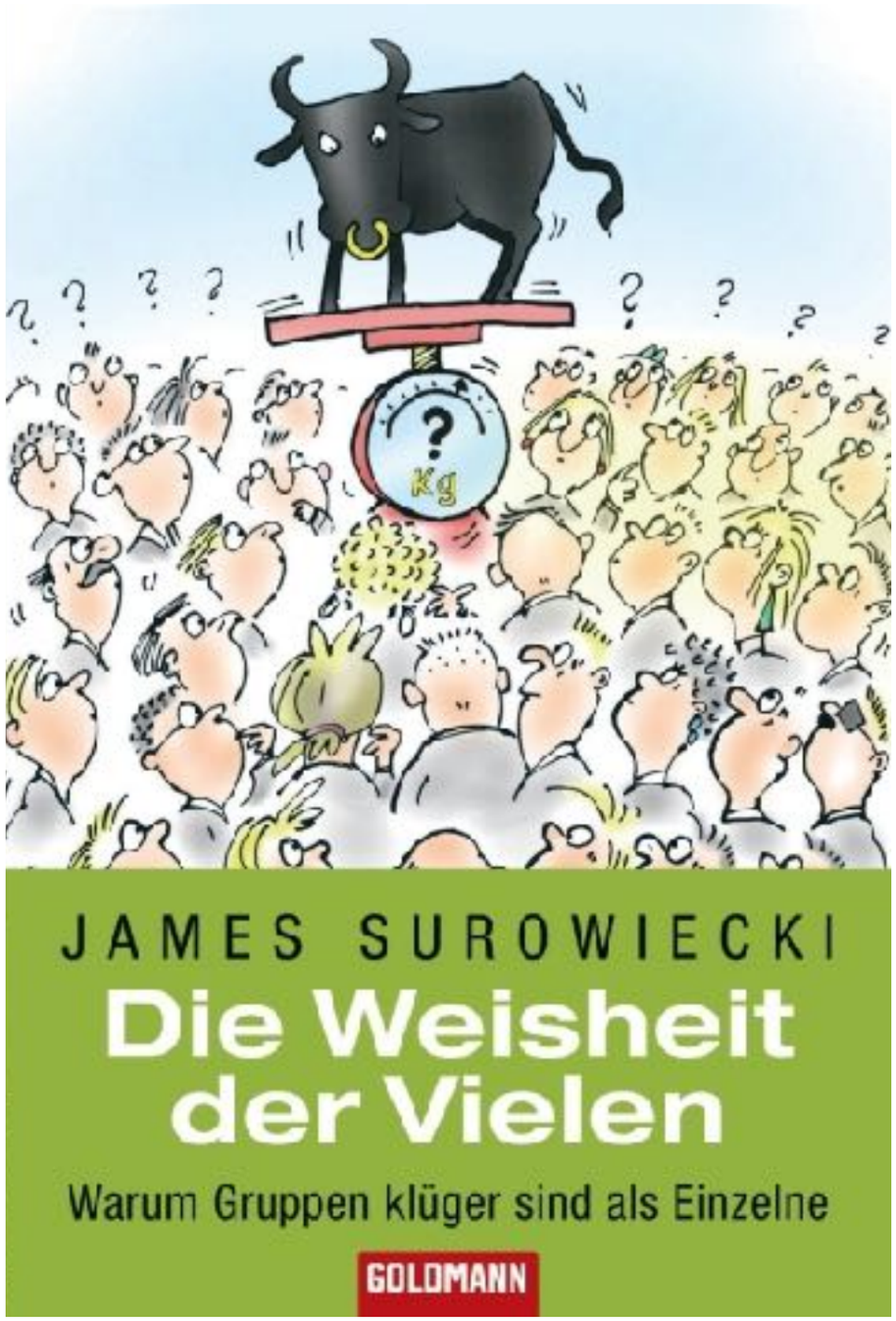
- 800 (lay) people guessed the weight of an ox at a country fair.
- Median guess deviated only 0.8% from the truth (mean even less).
- Better than most expert estimates.



‘The Wisdom of Crowds’

Francis Galton Experiment (1906):

- 800 (lay) people guessed the weight of an ox at a country fair.
- Median guess deviated only 0.8% from the truth (mean even less).
- Better than most expert estimates.



Example: mean

Mean: $\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} x_i$

Variance: $\hat{\sigma}^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \hat{\mu})^2$

Example: mean

Mean: $\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} x_i$ Variance: $\hat{\sigma}^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \hat{\mu})^2$

Variance of the mean is minimal: $\mathrm{Var}(\hat{\mu}) = \frac{\hat{\sigma}^2}{N} = \frac{1}{N(N-1)}\sum_{i=1}^{N}(x_i - \hat{\mu})^2$

Using that variances are linear for independent x_i : $\mathrm{Var}(\sum_i x_i) = \sum_i \mathrm{Var}(x_i)$

$\mathcal{X}$

Example: mean

Mean: $\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} x_i$ Variance: $\hat{\sigma}^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \hat{\mu})^2$

Variance of the mean is minimal: $\mathrm{Var}(\hat{\mu}) = \frac{\hat{\sigma}^2}{N} = \frac{1}{N(N-1)}\sum_{i=1}^{N}(x_i - \hat{\mu})^2$

Using that variances are linear for independent x_i : $\mathrm{Var}(\sum_i x_i) = \sum_i \mathrm{Var}(x_i)$

How to robust (low-variance) estimators for other statistics:

- Median, mode, percentiles, classifier outputs, … ?

Example: mean

Mean: $\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} x_i$ Variance: $\hat{\sigma}^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \hat{\mu})^2$

Variance of the mean is minimal: $\mathrm{Var}(\hat{\mu}) = \frac{\hat{\sigma}^2}{N} = \frac{1}{N(N-1)}\sum_{i=1}^{N}(x_i - \hat{\mu})^2$

Using that variances are linear for independent x_i : $\mathrm{Var}(\sum_i x_i) = \sum_i \mathrm{Var}(x_i)$

How to robust (low-variance) estimators for other statistics:

- Median, mode, percentiles, classifier outputs, … ?

→ **Resampling:** generate ‘new’ datasets by drawing samples from $\mathcal{X}$.

Jackknife (Maurice Quenouille, 1949)

Construct leave-one-out datasets $\mathcal{X}_i = \mathcal{X} \setminus \mathbf{x}_i$, $i \in \{1, \ldots, N\}$.

For any statistic θ, replace $\hat{\theta}(\mathcal{X})$ by Jackknife estimate $\hat{\theta}_J(\mathcal{X}) = \frac{1}{N} \sum_{i=1}^{N} \hat{\theta}(\mathcal{X}_i)$.

The variance of this estimator is $\mathrm{Var}\left[\hat{\theta}_J(\mathcal{X})\right] = \frac{N-1}{N} \sum_{i=1}^{N} \left(\hat{\theta}(\mathcal{X}_i) - \hat{\theta}_J(\mathcal{X})\right)^2$.

Jackknife (Maurice Quenouille, 1949)

Construct leave-one-out datasets $\mathcal{X}_i = \mathcal{X} \setminus \mathbf{x}_i$, $i \in \{1, \ldots, N\}$.

For any statistic θ, replace $\hat{\theta}(\mathcal{X})$ by Jackknife estimate $\hat{\theta}_J(\mathcal{X}) = \frac{1}{N} \sum_{i=1}^{N} \hat{\theta}(\mathcal{X}_i)$.

The variance of this estimator is $\mathrm{Var}\left[\hat{\theta}_J(\mathcal{X})\right] = \frac{N-1}{N} \sum_{i=1}^{N} \left(\hat{\theta}(\mathcal{X}_i) - \hat{\theta}_J(\mathcal{X})\right)^2$.

Notes:

- $N - 1$ appears in the numerator, because the $\hat{\theta}(\mathcal{X}_i)$ are highly dependent.
- Jackknife estimators for the mean are identical to the usual $\hat{\mu}$ and $\hat{\sigma}^2/N$.
- But can be used for any statistic → very handy tool.
- Can also be used to estimate and remove the bias of $\hat{\theta}(\mathcal{X})$.

Bootstrap (Bradley Efron, 1979)

Construct new datasets $\mathcal{X}_i$, $|\mathcal{X}_i| = N$, $i \in \{1, \ldots, M\}$, by sampling from $\mathcal{X}$ with replacement.

Bootstrap estimate: $\hat{\theta}_B(\mathcal{X}) = \frac{1}{M} \sum_{i=1}^{M} \hat{\theta}(\mathcal{X}_i)$

Variance: $\text{Var}\left[\hat{\theta}_B(\mathcal{X})\right] = \frac{1}{M} \sum_{i=1}^{M} \left(\hat{\theta}(\mathcal{X}_i) - \hat{\theta}_B(\mathcal{X})\right)^2$

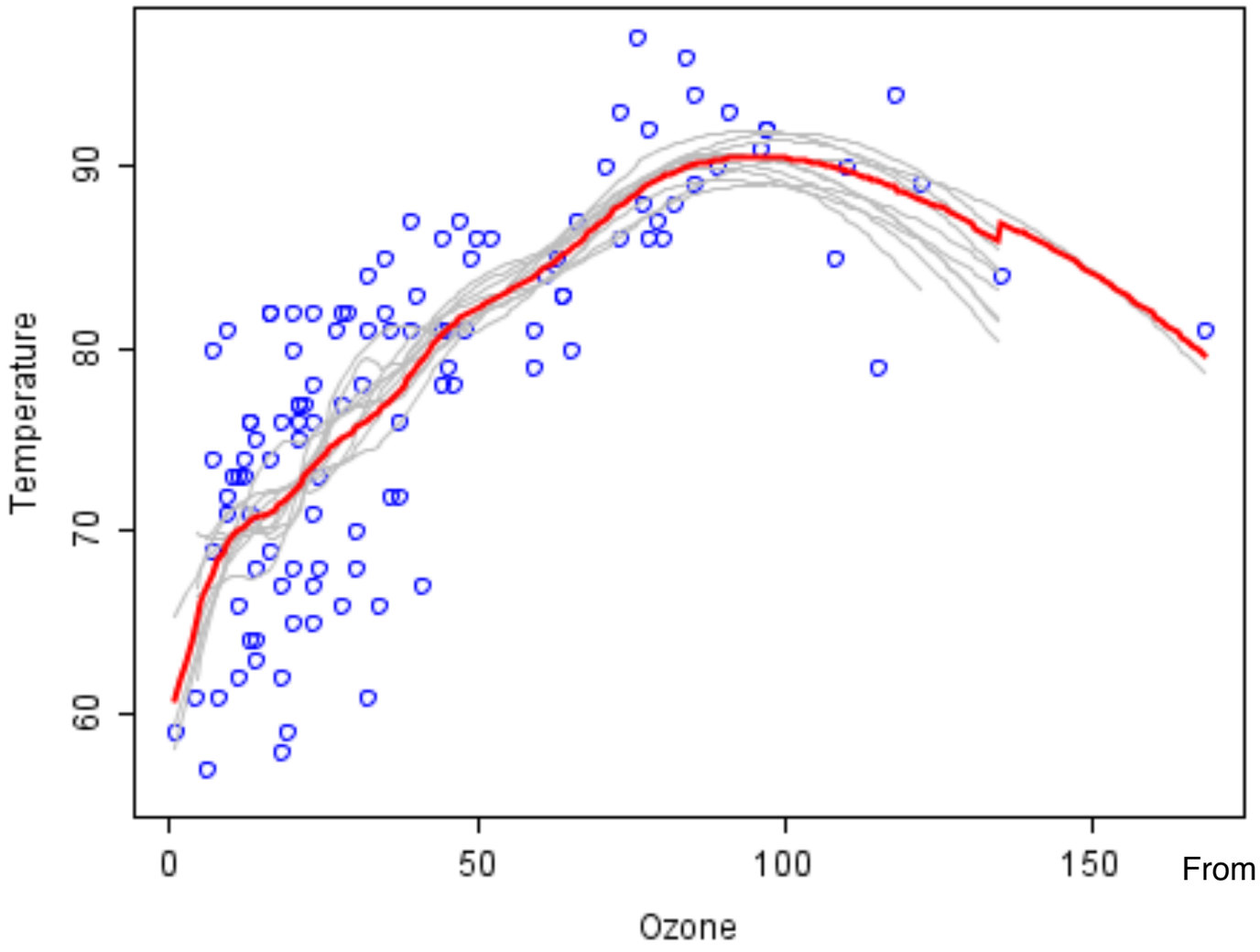
Similar properties as Jackknife.

Bagging

Bootstrap aggregating (Breiman, 1996): use Bootstrap to reduce the variance of predictions

- Construct Bootstrap samples $\mathcal{X}_i$, $|\mathcal{X}_i| = N$, $i \in \{1, \ldots, M\}$
- Train any model of choice on M datasets
- For a given test data point $\boldsymbol{x}$, obtain M predictions.
- Aggregate predictions
  - Regression: average
  - Classification: majority vote

Bagging



From Wikipedia

Random Subspace Method

- Problem with Bagging: datasets $\mathcal{X}_i$ are highly overlapping (~63%).
- Decision trees trained on $\mathcal{X}_i$ and their predictions become correlated.

→ Bagging estimators do not reduce variance as much as desired.

Random Subspace Method

- Problem with Bagging: datasets $\mathcal{X}_i$ are highly overlapping (~63%).
- Decision trees trained on $\mathcal{X}_i$ and their predictions become correlated.

→ Bagging estimators do not reduce variance as much as desired.

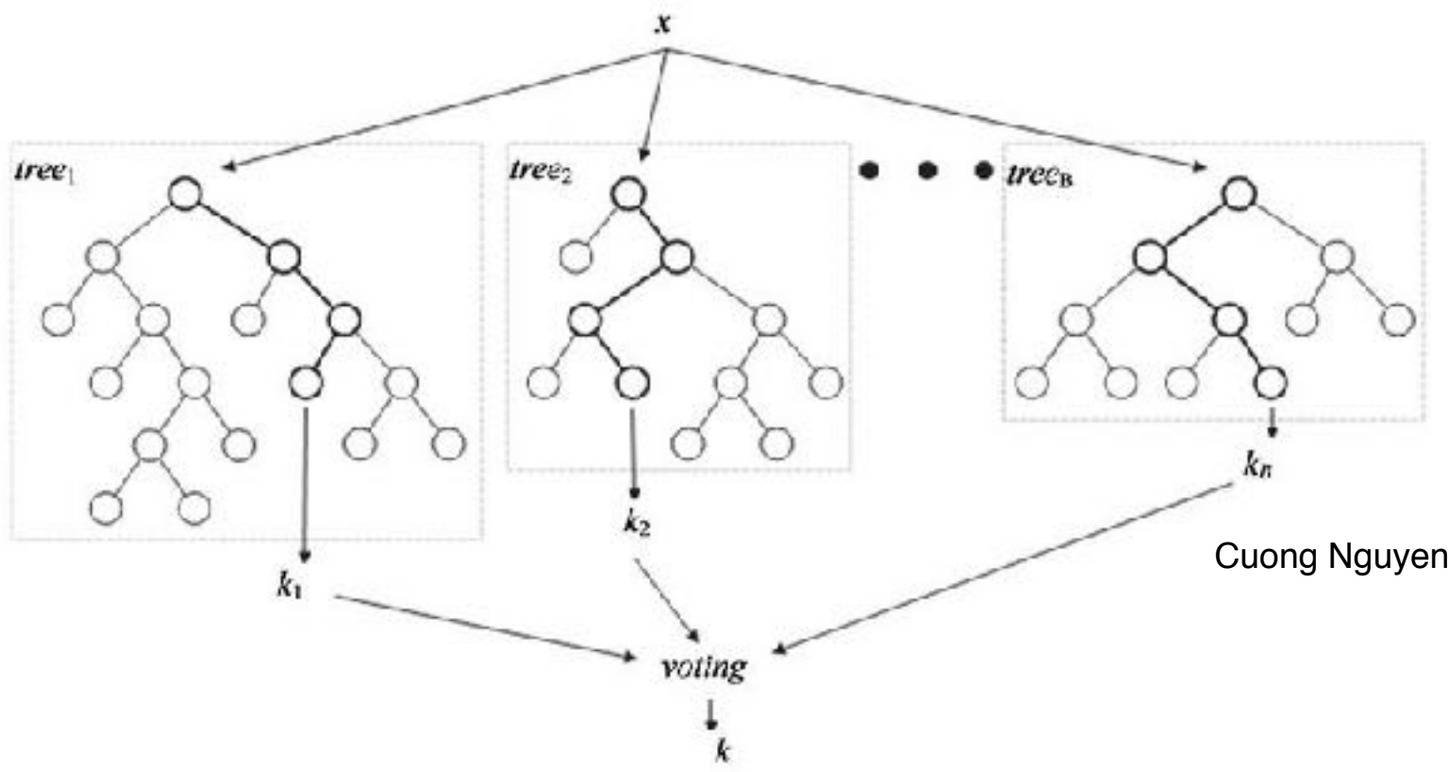
Random subspace method: construct new datasets by randomly sampling $d_B \leq d$ features without replacement. (Tin Kam Ho, 1998)

→ De-correlates estimators and decreases variance of the aggregate.

Random Forests

Combination of Bagging and random subspace method applied to decision trees. (Breiman, 1999)

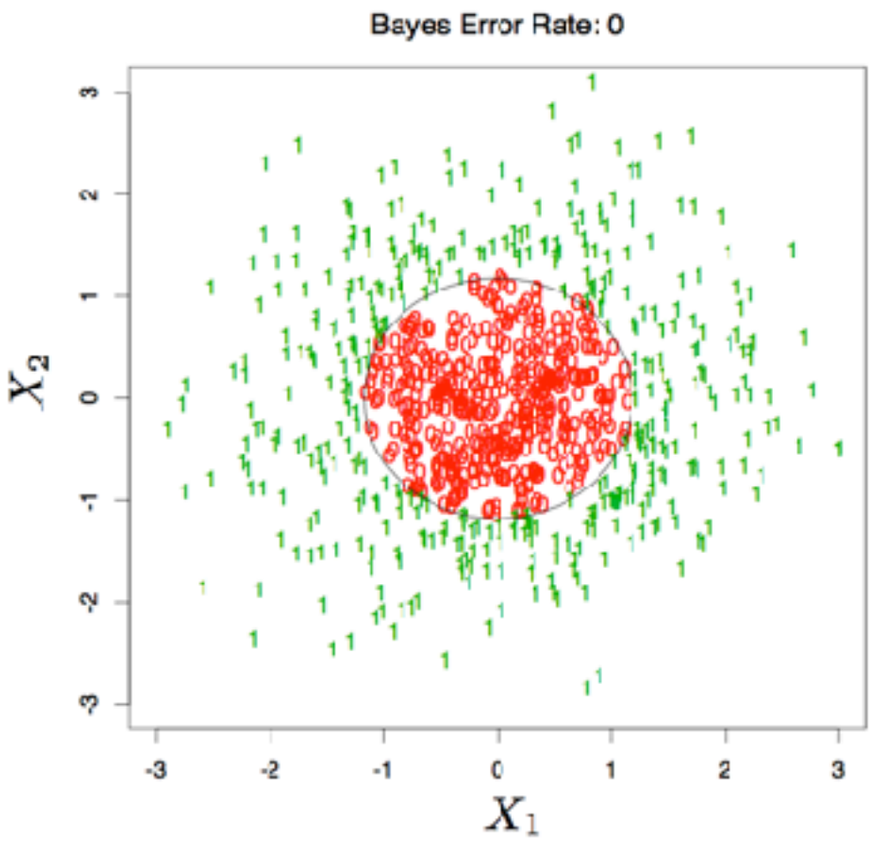
- Complexity control through out of bag monitoring.
- Random feature selection either at tree or split level.



Cuong Nguyen

Toy example

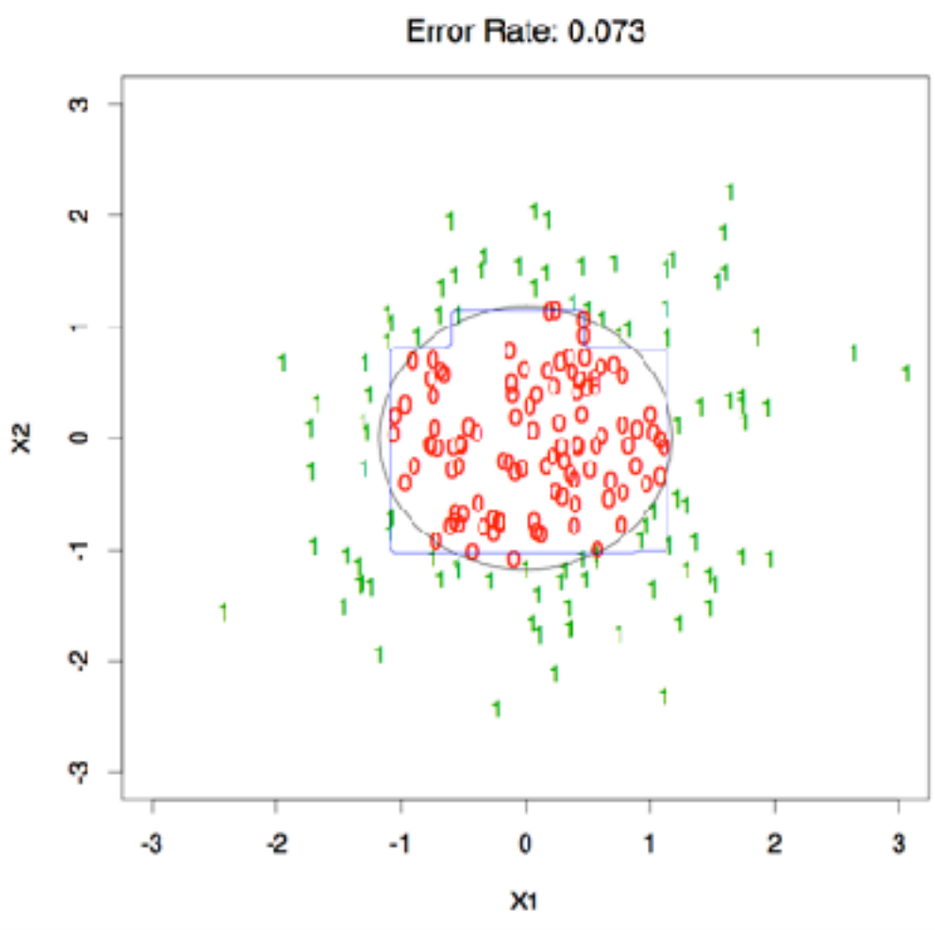
Toy Example - No Noise



- Deterministic problem; noise comes from sampling distribution of X.
- Use a training sample of size 200.
- Here Bayes Error is 0%.

Toy example

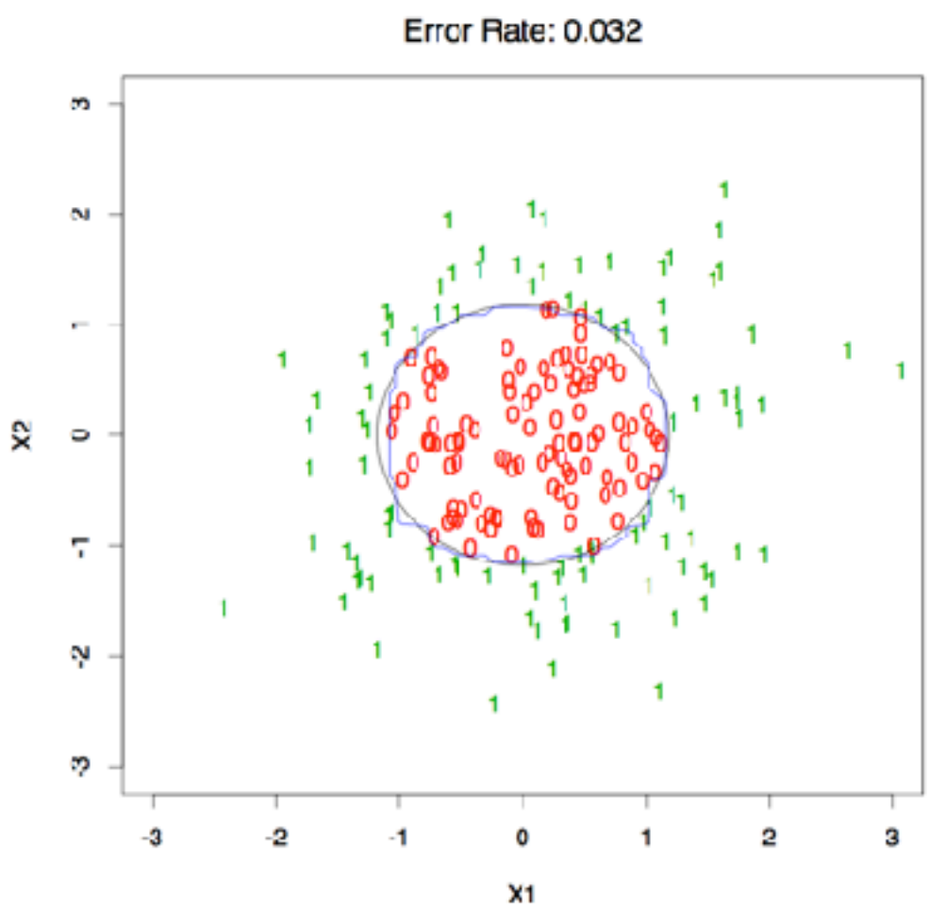
Decision Boundary: Tree



When the nested spheres are in 10-dimensions, Classification Trees produces a rather noisy and inaccurate rule $\hat{C}(X)$, with error rates around 30%.

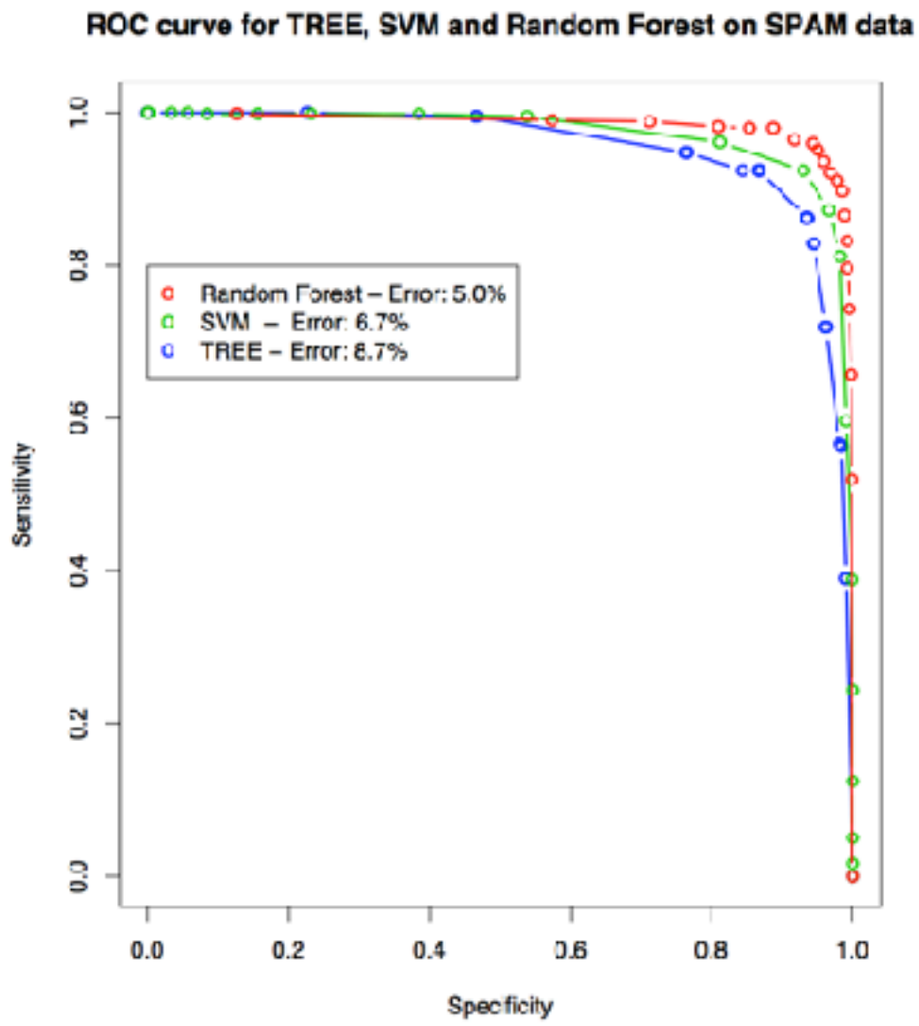
Toy example

Decision Boundary: Bagging



Bagging averages many trees, and produces smoother decision boundaries.

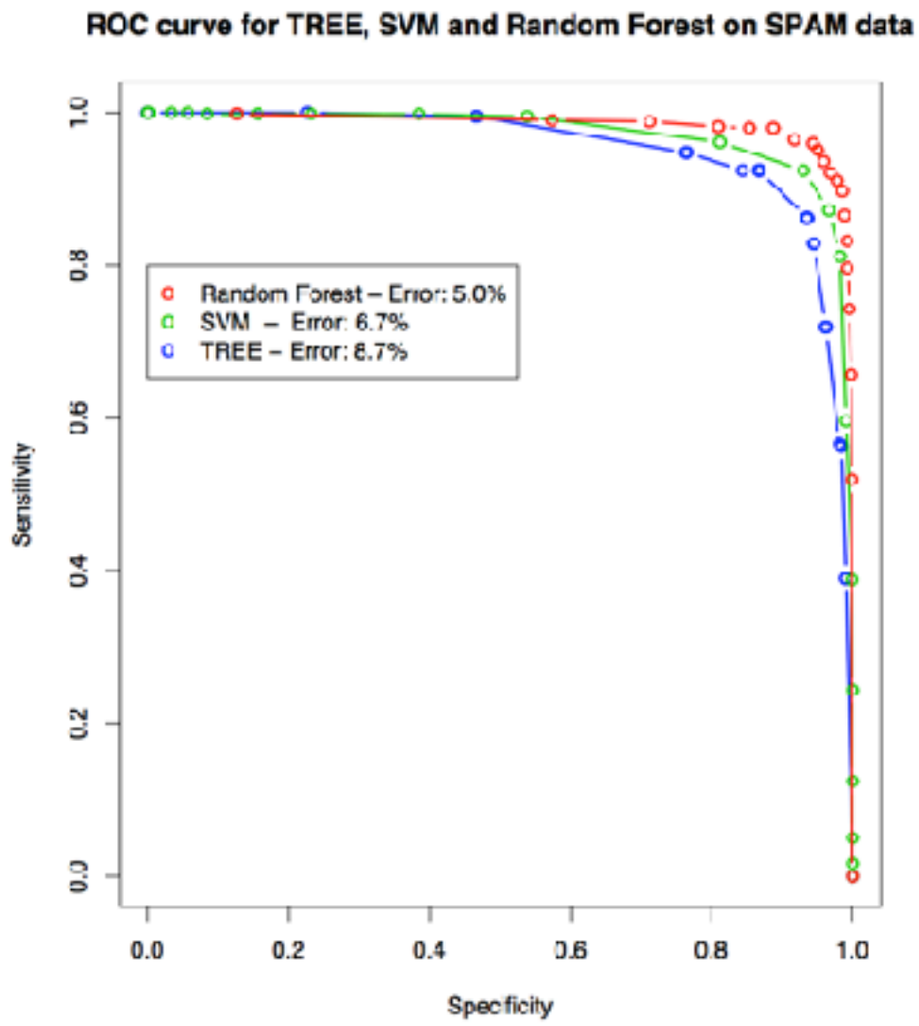
Toy example



TREE, SVM and RF

Random Forest dominates both other methods on the SPAM data — 5.0% error. Used 500 trees with default settings for `randomForest` package in `R`.

Toy example

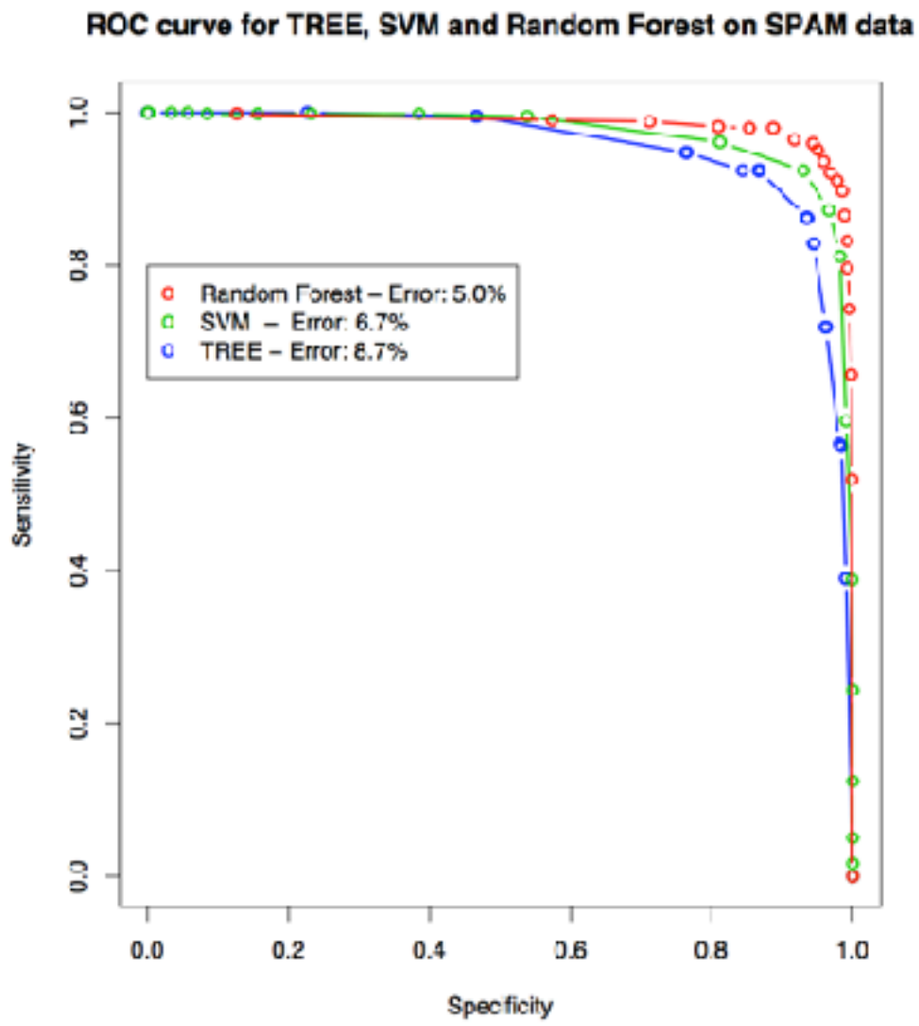


TREE, SVM and RF

Random Forest dominates both other methods on the SPAM data — 5.0% error. Used 500 trees with default settings for `randomForest` package in `R`.

In general Boosting $\succ$ Random Forests $\succ$ Bagging $\succ$ Single Tree.

Toy example



TREE, SVM and RF

Random Forest dominates both other methods on the SPAM data — 5.0% error. Used 500 trees with default settings for `randomForest` package in `R`.

In general Boosting $\succ$ Random Forests $\succ$ Bagging $\succ$ Single Tree.

Boosting

Consider the binary classification setting

$$\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{R}^d \, ; y_1, \ldots, y_N \in \{+1, -1\}$$

A classifier, or 'hypothesis': $f(\mathbf{x}) : \mathbb{R}^d \mapsto \{+1, -1\}$, $f \in \mathcal{F}$

$\mathcal{F}$ is some hypothesis set (e.g. $f(\mathbf{x}) = \text{sign}(\mathbf{w}^\top \mathbf{x} + b)$)

Boosting

Consider the binary classification setting

$$\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{R}^d \, ; y_1, \ldots, y_N \in \{+1, -1\}$$

A classifier, or 'hypothesis': $f(\mathbf{x}) : \mathbb{R}^d \mapsto \{+1, -1\}$, $f \in \mathcal{F}$

$\mathcal{F}$ is some hypothesis set (e.g. $f(\mathbf{x}) = \text{sign}(\mathbf{w}^\top \mathbf{x} + b)$)

0/1 classification loss: $l(f(x), y) = \begin{cases} 0 & \text{if } f(x) = y \\ 1 & \text{if } f(x) \neq y \end{cases}$

Empirical risk minimization: $R_{\text{emp}}(f) = \frac{1}{N} \sum_{i=1}^{N} l(f(\mathbf{x}_i), y_i)$

Boosting

Weighted ensemble learning: weighted majority vote over M hypotheses.

$$f_{\text{Ens}}(\mathbf{x}) = \sum_{i=1}^{M} a_i f_i(\mathbf{x}) \ , \ \ a_i \geq 0$$

Boosting

Weighted ensemble learning: weighted majority vote over M hypotheses.

$$f_{\text{Ens}}(\mathbf{x}) = \sum_{i=1}^{M} a_i f_i(\mathbf{x}) \ , \ a_i \geq 0$$

Bagging:

- $a_i = {}^1/_M$
- f_i are obtained using Bootstrap samples.

Boosting

Weighted ensemble learning: weighted majority vote over M hypotheses.

$$f_{\text{Ens}}(\mathbf{x}) = \sum_{i=1}^{M} a_i f_i(\mathbf{x}) \ , \ a_i \geq 0$$

Bagging:

- $a_i = {}^1/_M$
- f_i are obtained using Bootstrap samples.

Boosting:

- f_i are obtained from all samples, weighted by importance (soft selection)
- Sample weights: $d_i \in [0, 1] \ , \ i \in \{1, \ldots, N\} \ , \ \sum_{i=1}^{N} d_i = 1$
- Sample and hypothesis weights adapted over iterations 1, …, M

Motivation

- Difficult to achieve single classifier with high accuracy.
- But easy to devise ‘weak learners’ with non-random performance.
- **Definition:** $\exists \gamma \in [0, 1] \; \forall \{d_i\} \; \forall \mathcal{X} \forall \mathcal{Y} \; \varepsilon(f, \mathbf{d}) < \frac{1}{2}(1 - \gamma)$,

  where $\varepsilon(f, \mathbf{d}) = \sum_{i=1}^{N} d_i l(f(\mathbf{x}_i), y_i)$ is the weighted training error.

Motivation

- Difficult to achieve single classifier with high accuracy.
- But easy to devise ‘weak learners’ with non-random performance.
- **Definition:** $\exists \underset{N}{\gamma} \in [0, 1]\ \forall\{d_i\}\ \forall\mathcal{X}\forall\mathcal{Y}\ \varepsilon(f, \mathbf{d}) < \frac{1}{2}(1-\gamma)$,
  where $\varepsilon(f, \mathbf{d}) = \sum_{i=1} d_i l(f(\mathbf{x}_i), y_i)$ is the weighted training error.
- **Idea:** combine the hypotheses of different weak learners.
- **In practice:** same algorithm trained on differently weighted samples.
- Iterate with higher emphasis (weight) on misclassified examples.
- **Final hypothesis:** sum of weak hypotheses, weighted by training acc.

Analogy: horse-racing bets

- Nobody is really good in predicting a horse race
- But everybody has some rule of thumb that is better than random (teeth, chest, history, jockey weight)
- Strategy: present learner with races that are hardest to predict according to current model

→ ‘Maximal information gain’ per expert consultation

AdaBoost

Algorithm 1 (Discrete) AdaBoost (Freund and Schapire, 1996)

Input: sample $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, number of iterations M

Initialize: $d_i^{(1)} = \frac{1}{N}$, $i = 1, \ldots, N$

1: **for** $m = 1, \ldots, M$ **do**

2: Train classifier on weighted sample $\left\{(d_1^{(t)}, \mathbf{x}_1, y_1), \ldots, (d_N^{(t)}, \mathbf{x}_N, y_N)\right\}$ and obtain hypothesis $f^{(m)}(\mathbf{x})$

3: Calculate weighted training error: $\varepsilon^{(m)} = \sum_{i=1}^{N} d_i^{(t)} I(f^{(m)}(\mathbf{x}_i), y_i)$

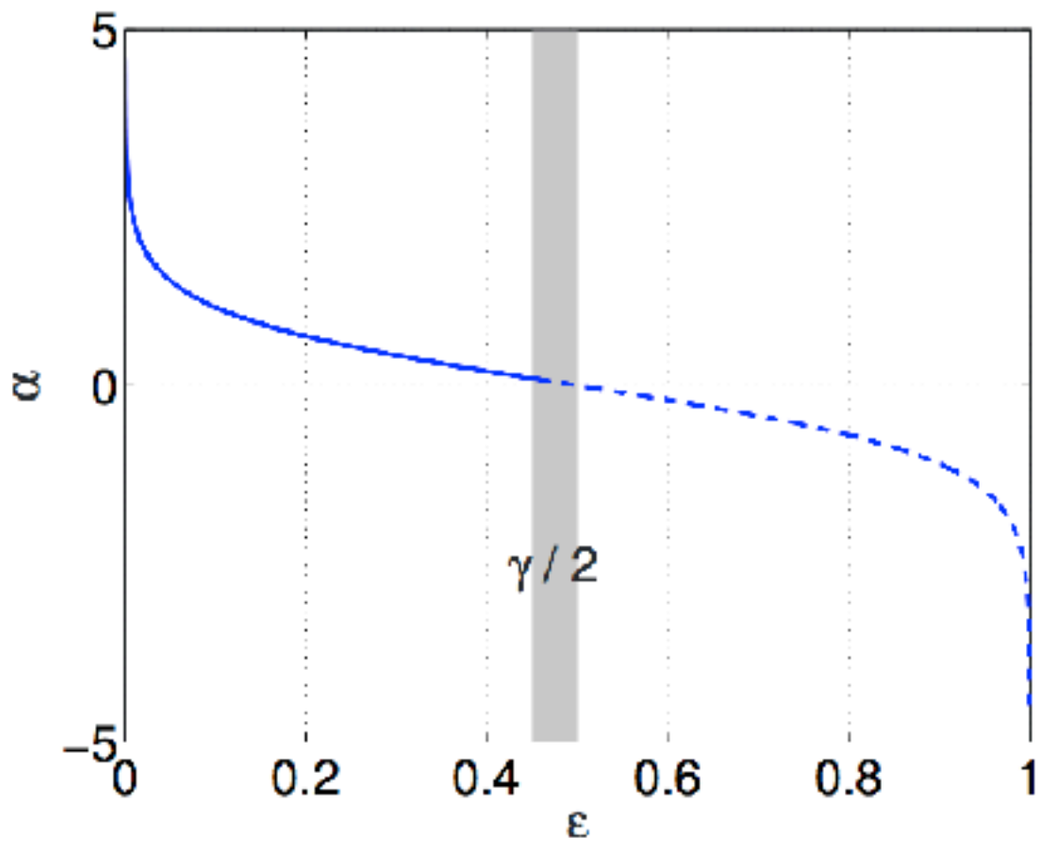
4: Assign weight to $f^{(m)}$: $\alpha^{(m)} = \frac{1}{2} \ln \frac{1-\varepsilon^{(m)}}{\varepsilon^{(m)}}$

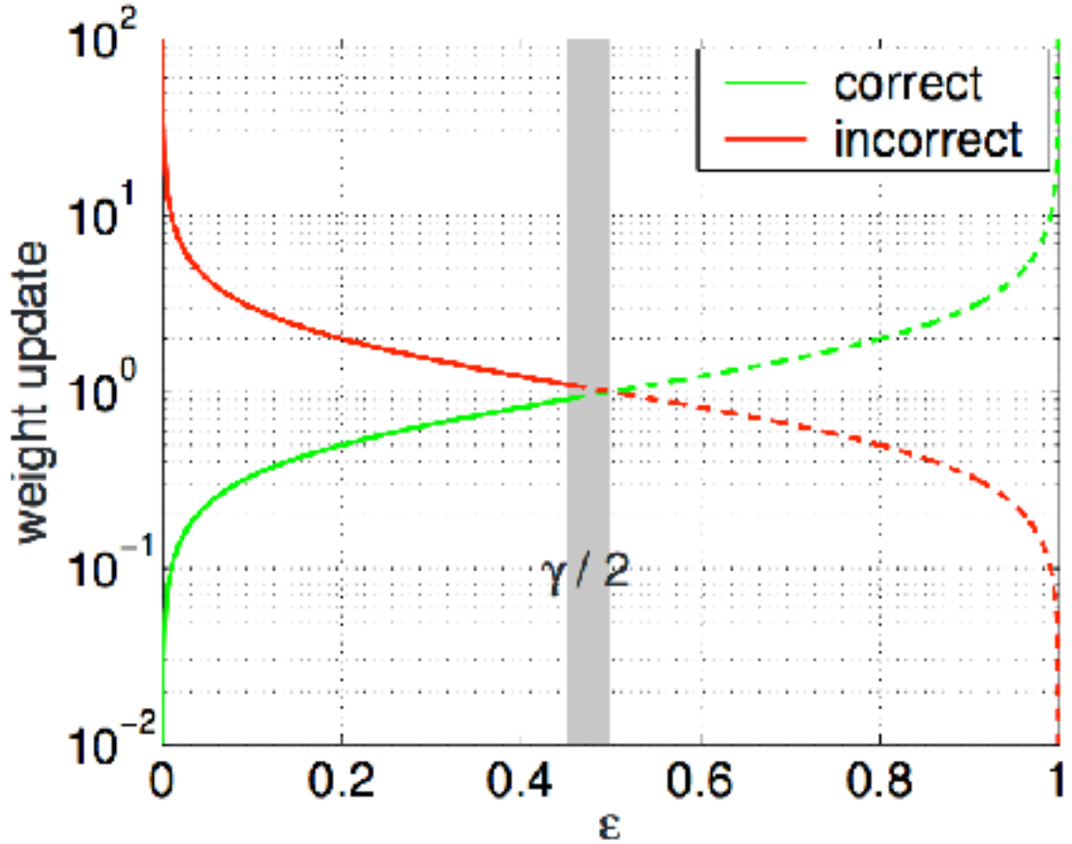
5: Update sample weights: $d_i^{(m+1)} = d_i^{(m)} \exp(-\alpha_m y_i f^{(m)}(\mathbf{x}_i))/Z^{(m)}$,
$Z^{(m)} = \sum_{i=1}^{N} d_i^{(m)} \exp(-\alpha_m y_i f^{(m)}(\mathbf{x}_i))$

6: **end for**

Final hypothesis: $f(x) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m f^{(m)}(\mathbf{x})\right)$

AdaBoost: weights





AdaBoost

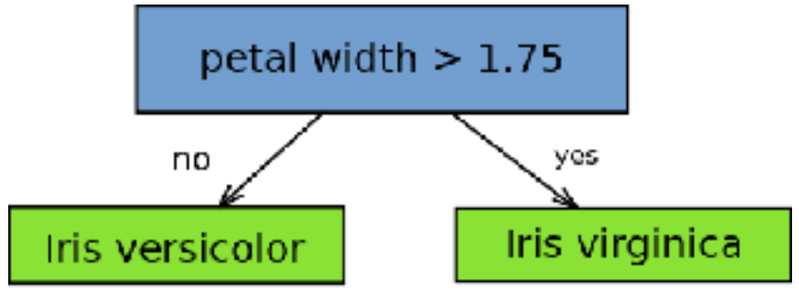
Choices of weak learners

- Trees

AdaBoost

Choices of weak learners

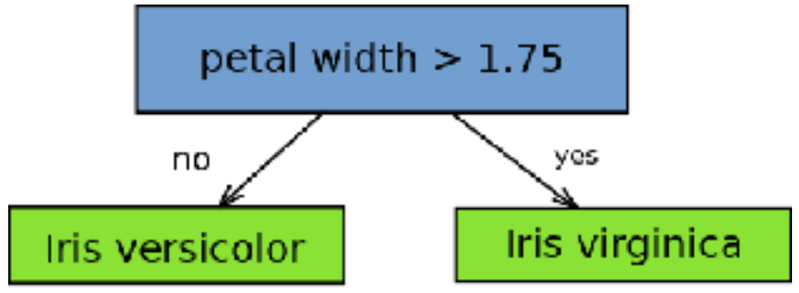
- Trees
- Stumps



AdaBoost

Choices of weak learners

- Trees
- Stumps



How to train on a weighted sample?

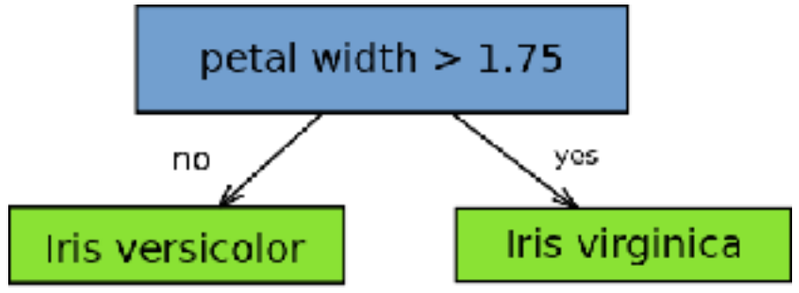
- Incorporate d in the loss function of the classifier.

e.g. $\min_{w,\xi,b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} p_i \xi_i \quad \text{s.t. } \forall i \ : \ y_i(w^\top x_i + b) \geq 1 - \xi_i, \ \ \xi_i \geq 0.$

AdaBoost

Choices of weak learners

- Trees
- Stumps



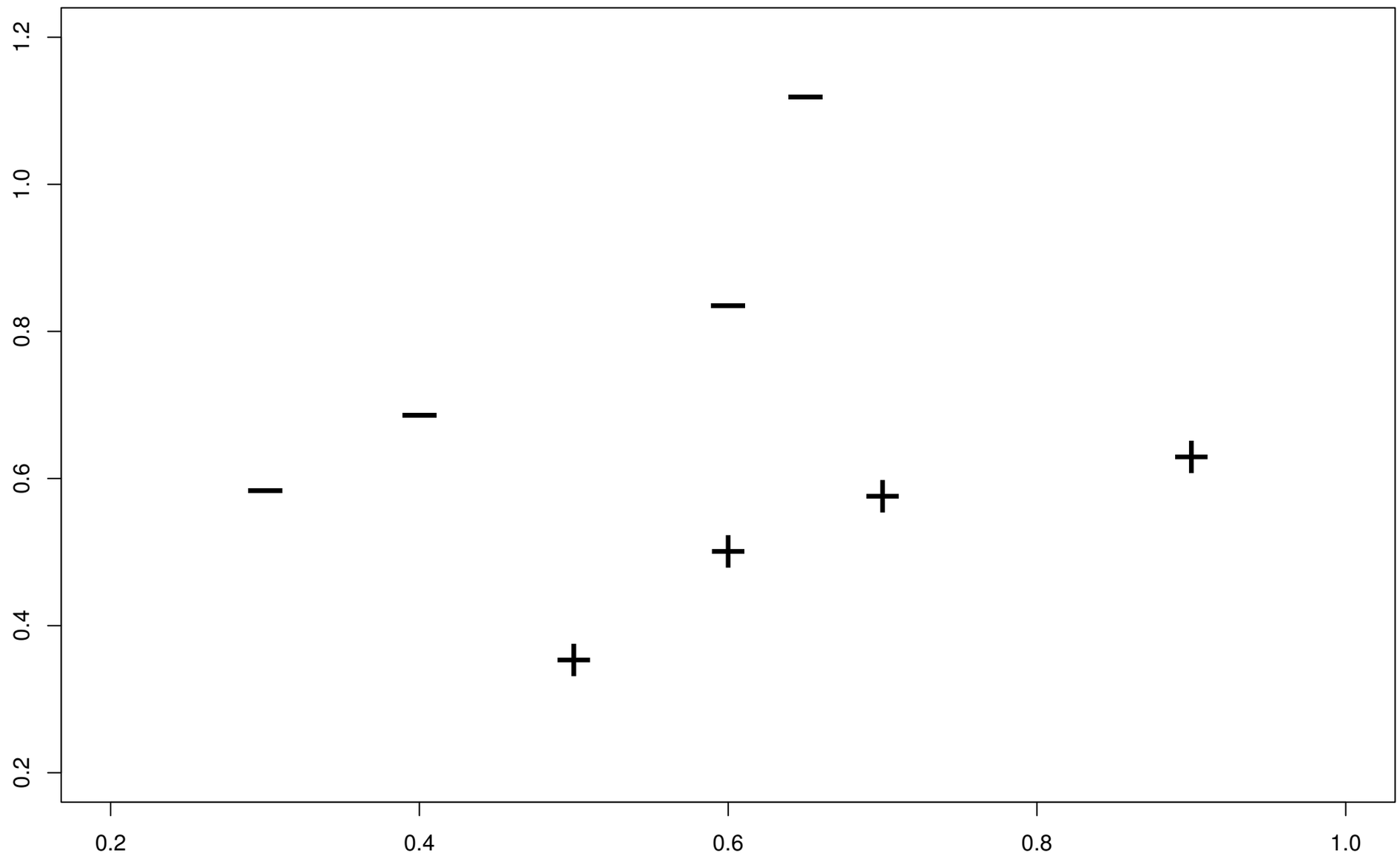
How to train on a weighted sample?

- Incorporate d in the loss function of the classifier.

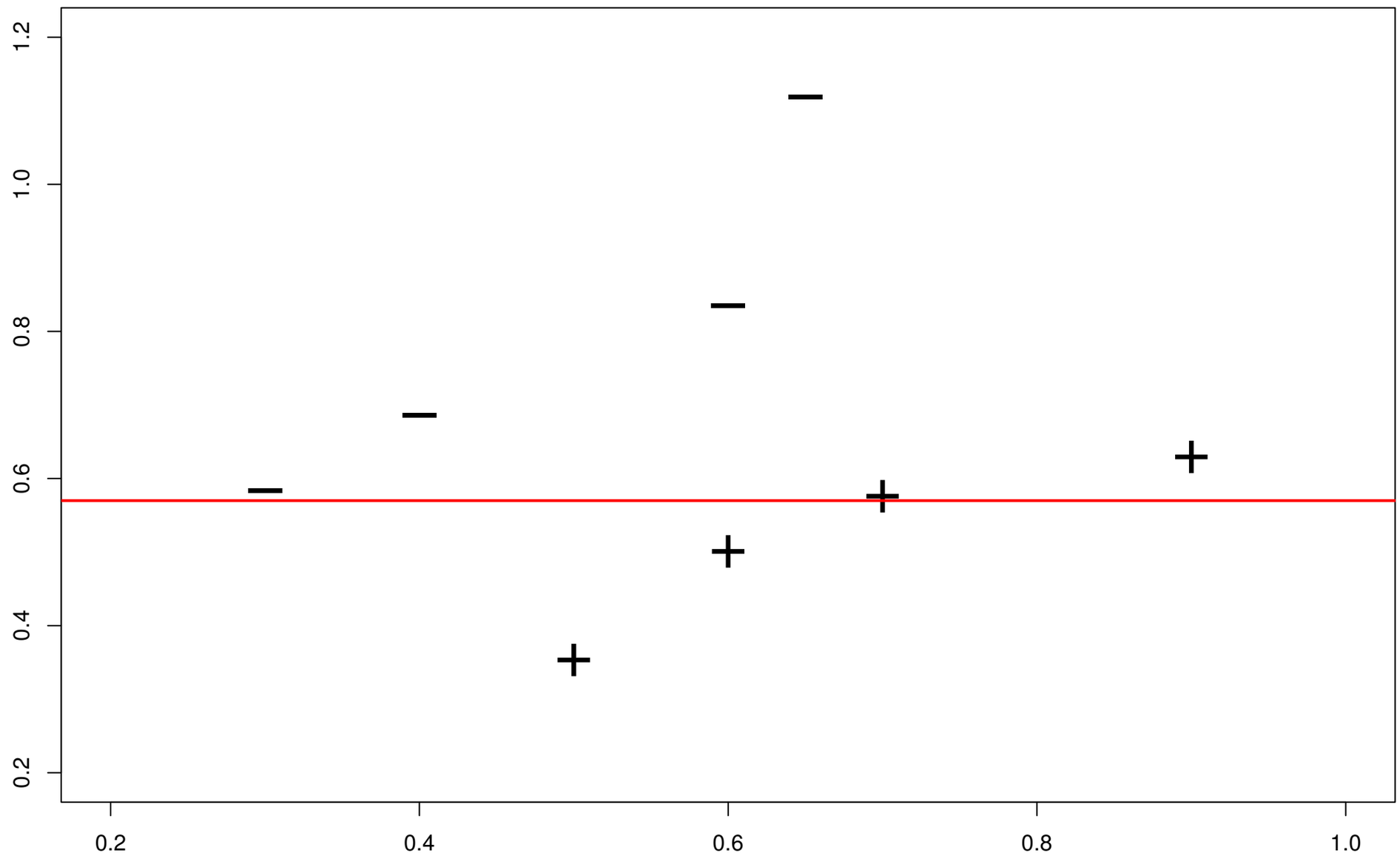
  e.g. $\min_{w,\xi,b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} p_i \xi_i \quad \text{s.t. } \forall i \ : \ y_i(w^\top x_i + b) \geq 1 - \xi_i, \ \ \xi_i \geq 0.$

- Over-sample data to follow ‘distribution’ d.

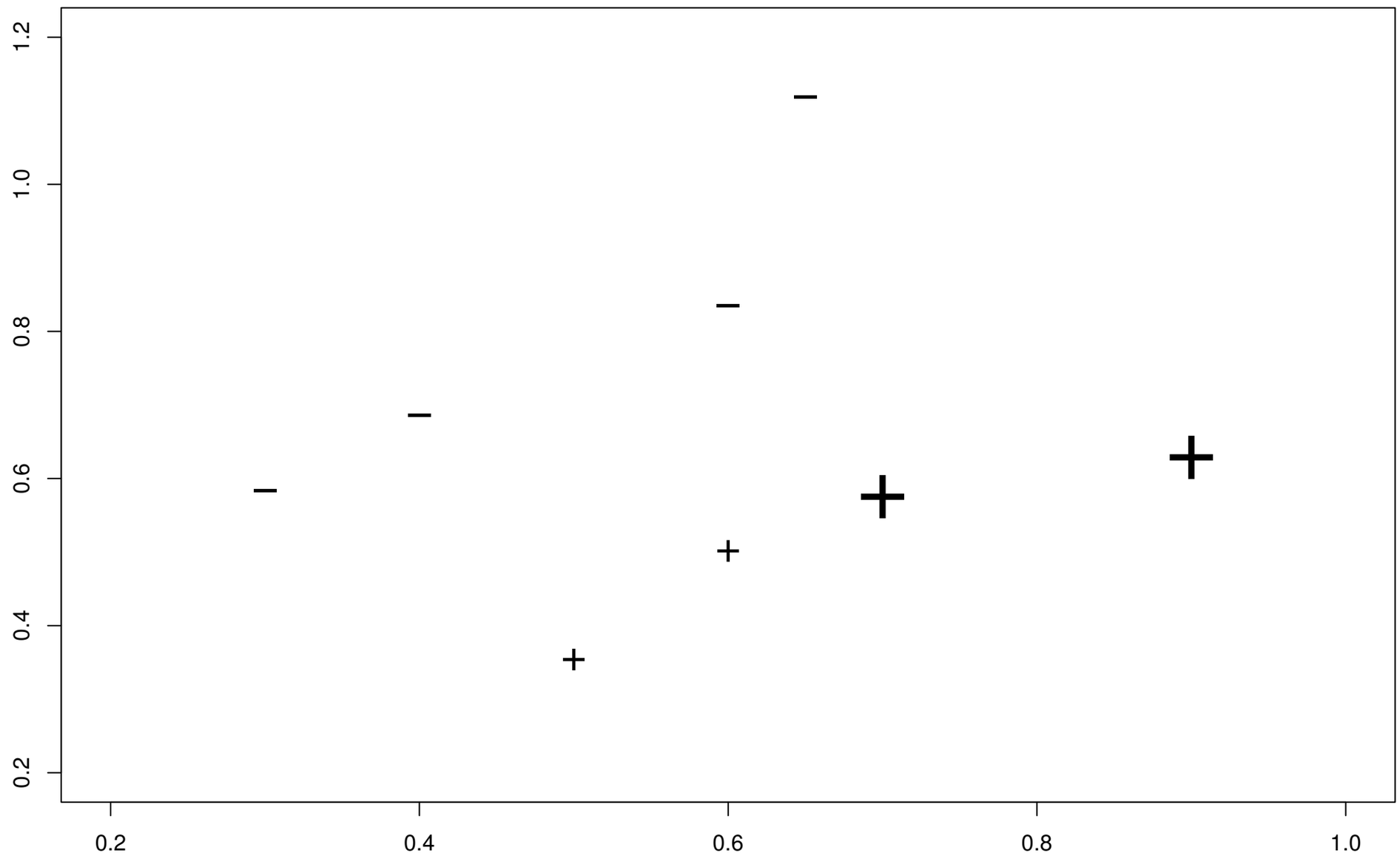
Example



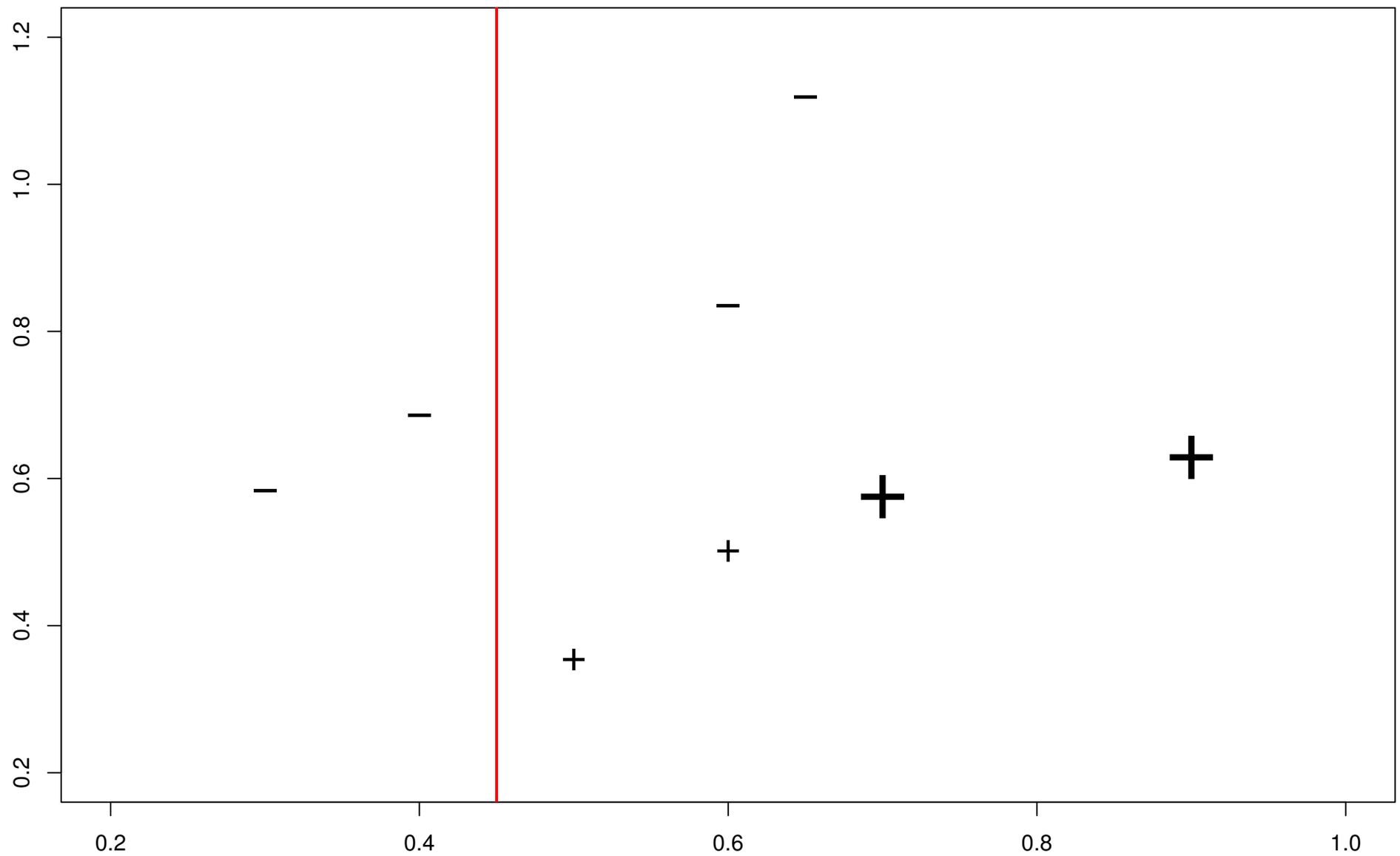
Example



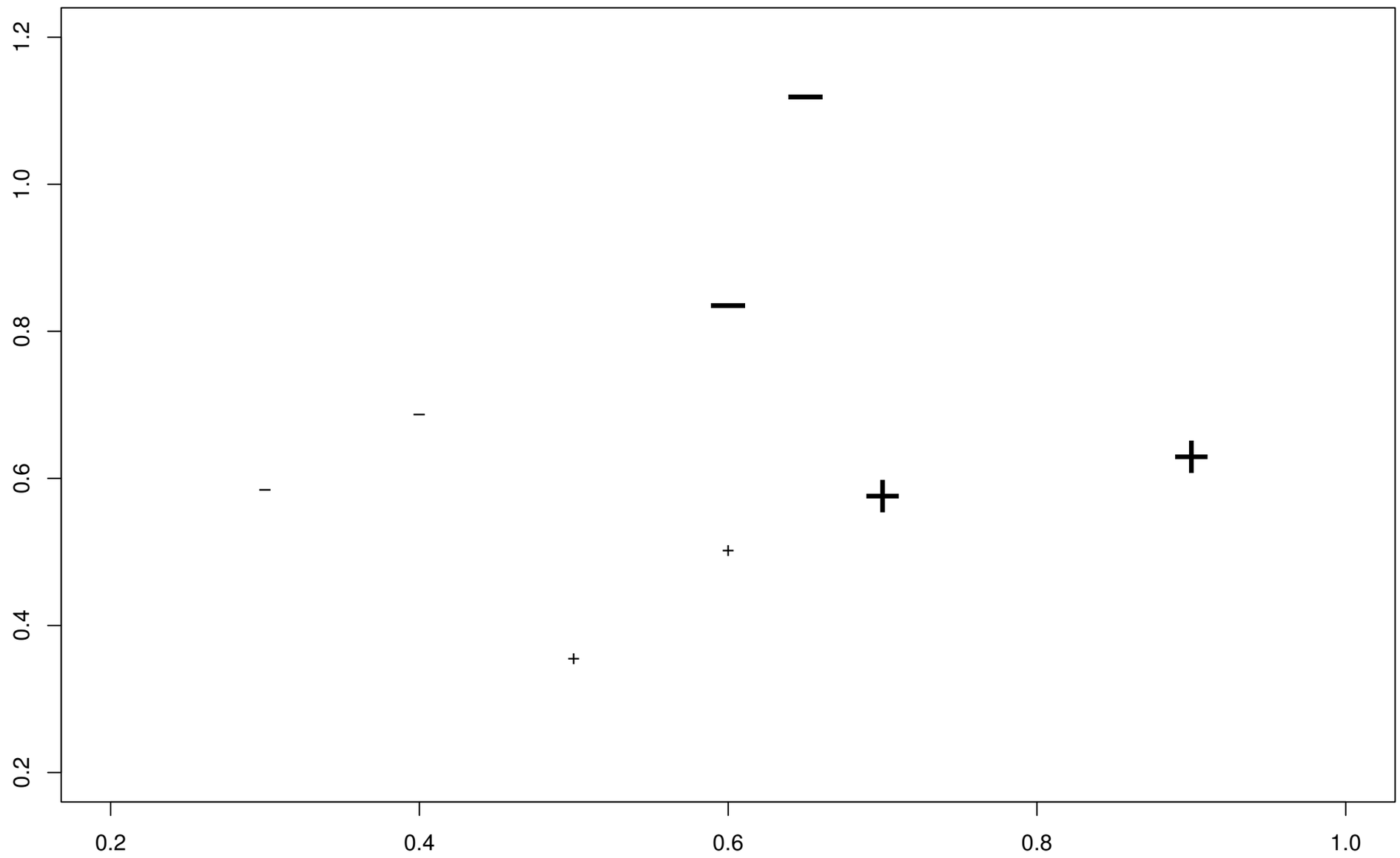
Example



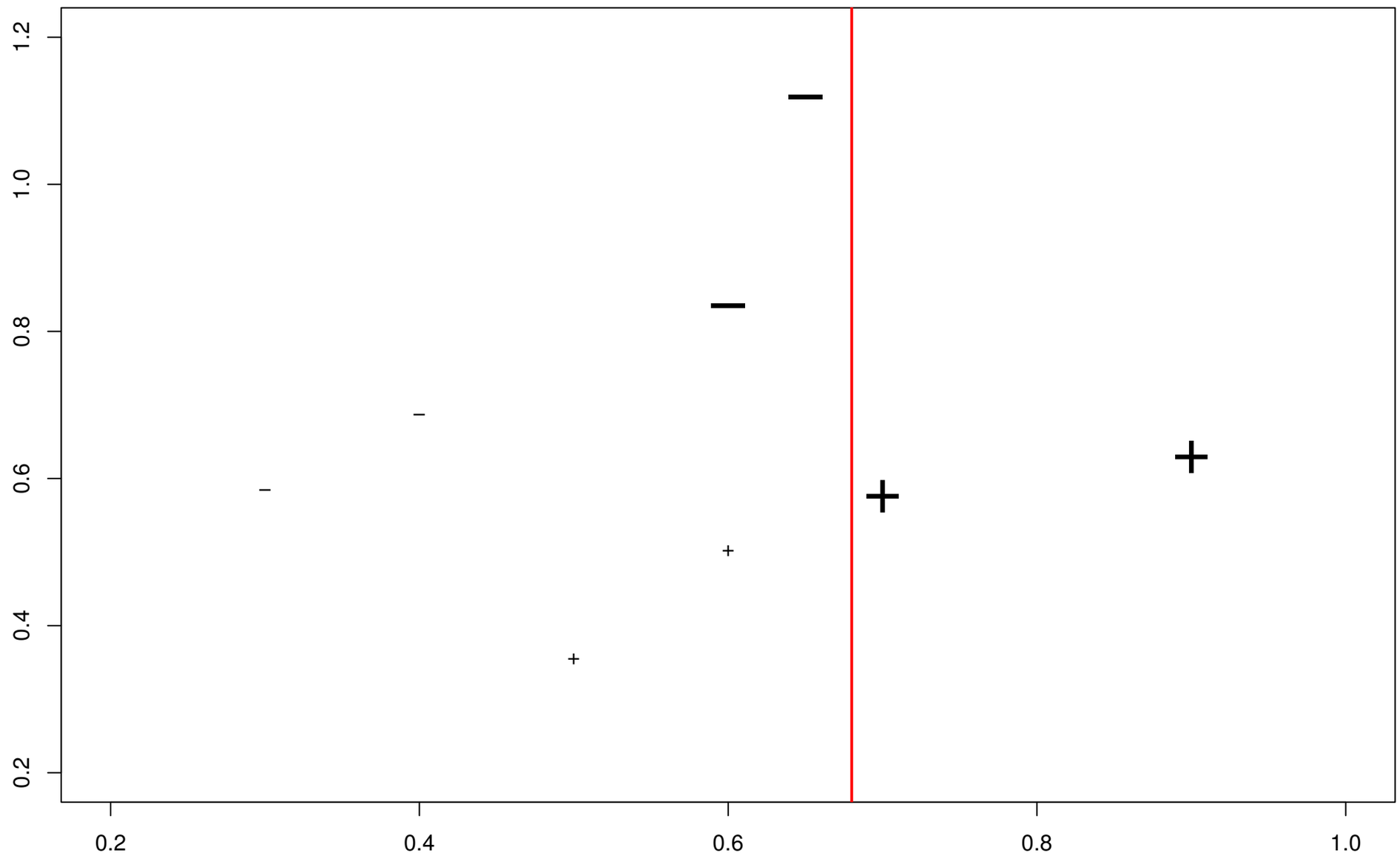
Example



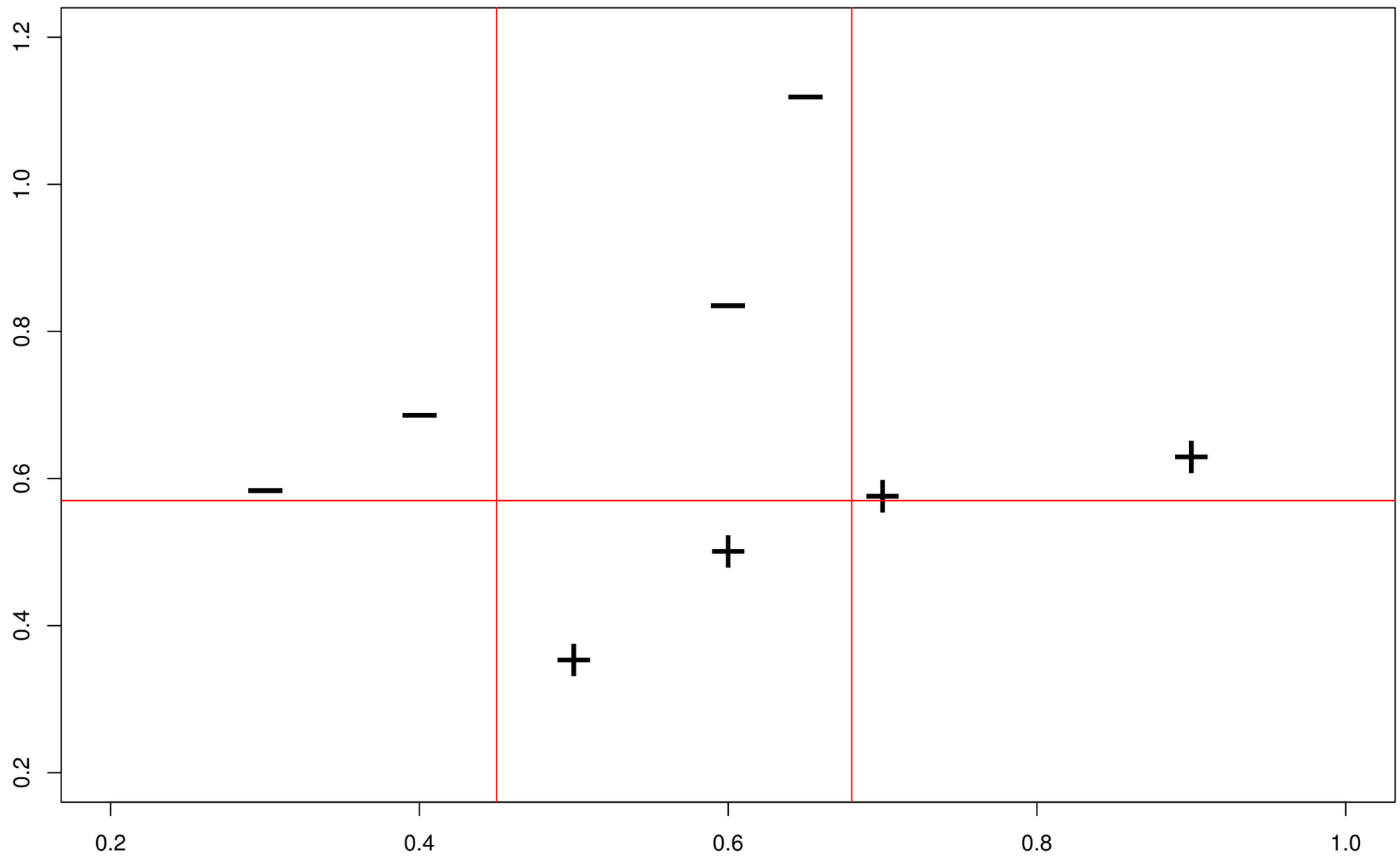
Example



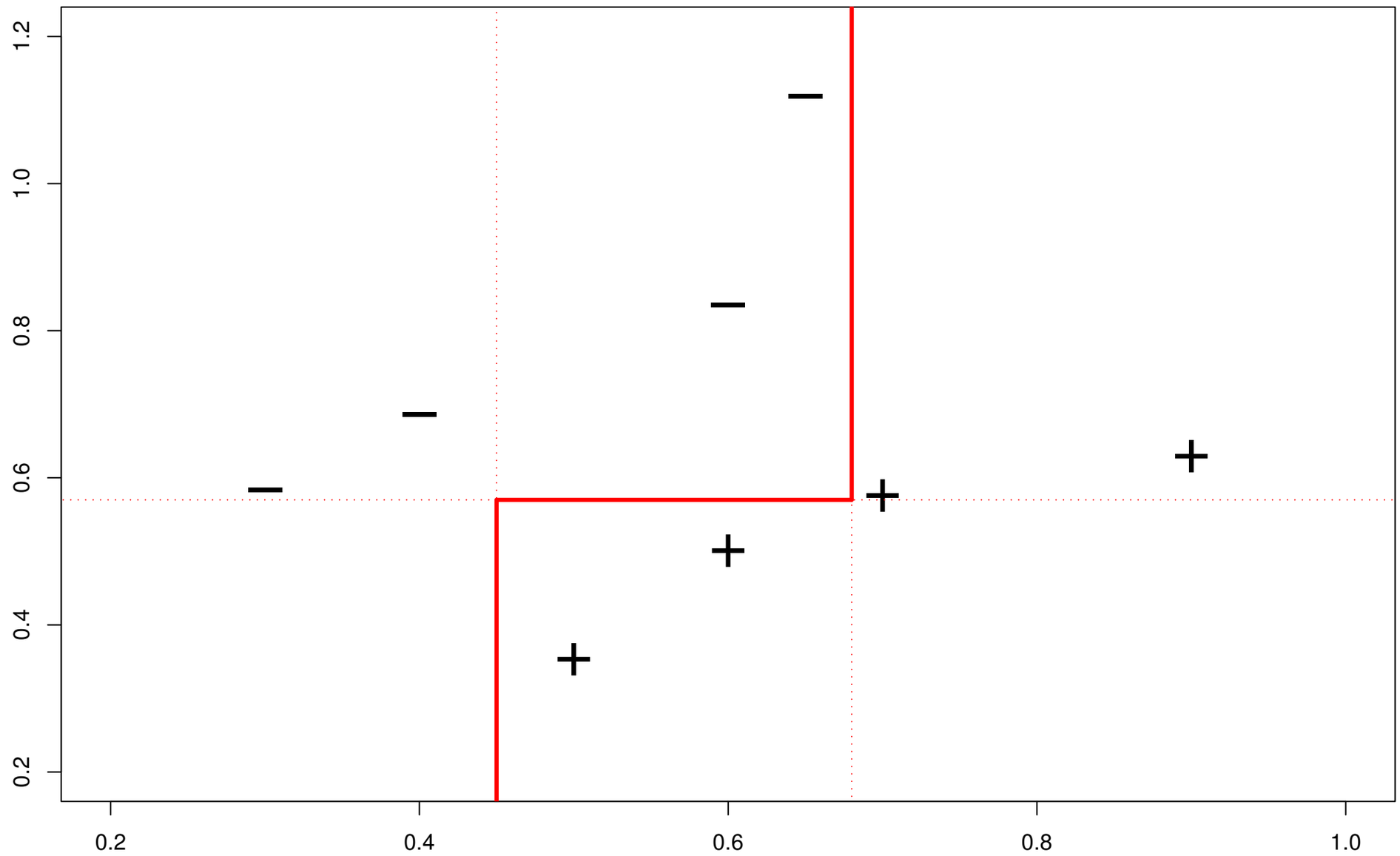
Example



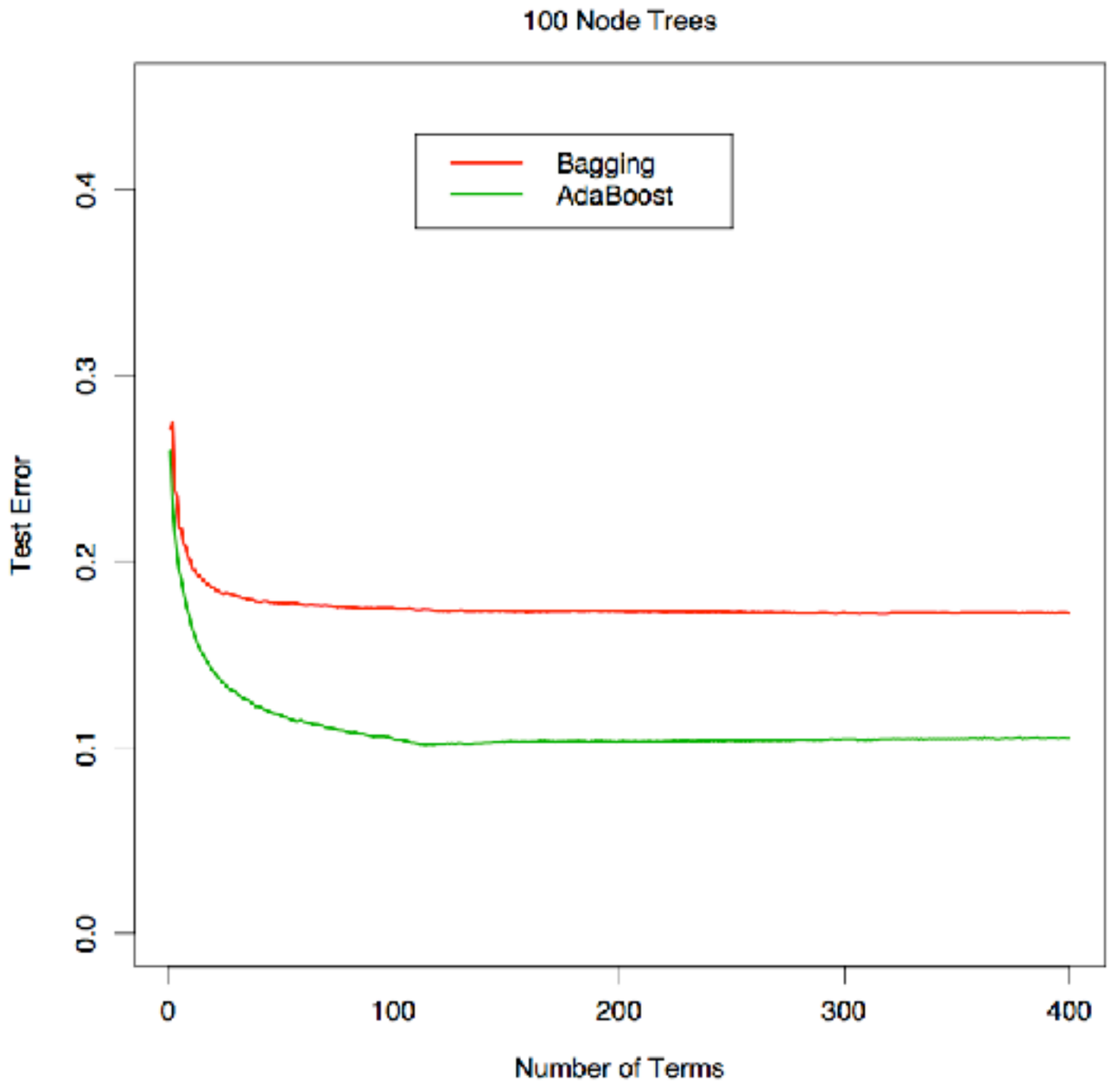
Example



Example



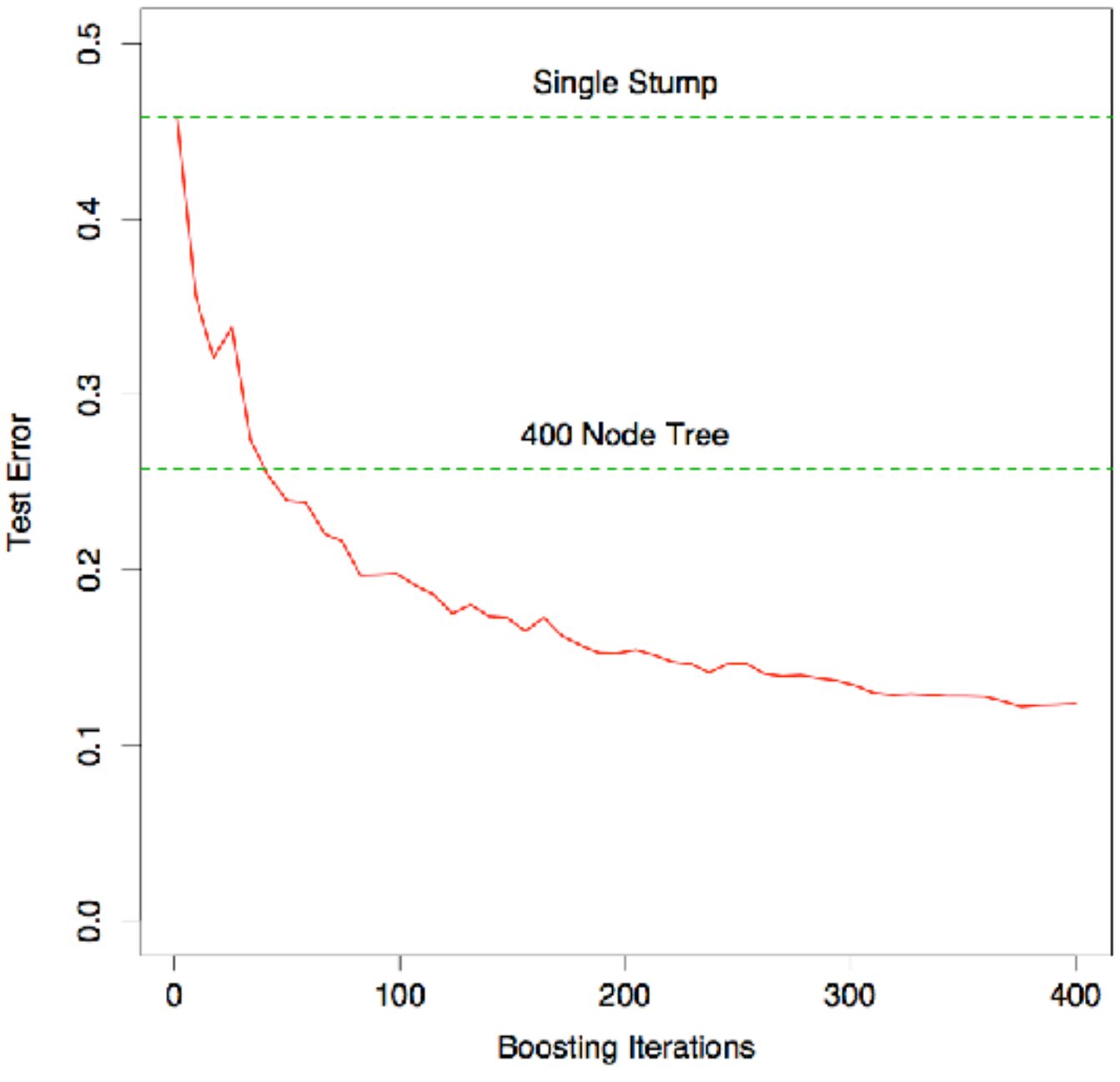
Toy example revisited



Boosting vs Bagging

- 2000 points from Nested Spheres in R^{10}
- Bayes error rate is 0%.
- Trees are grown best first without pruning.
- Leftmost term is a single tree.

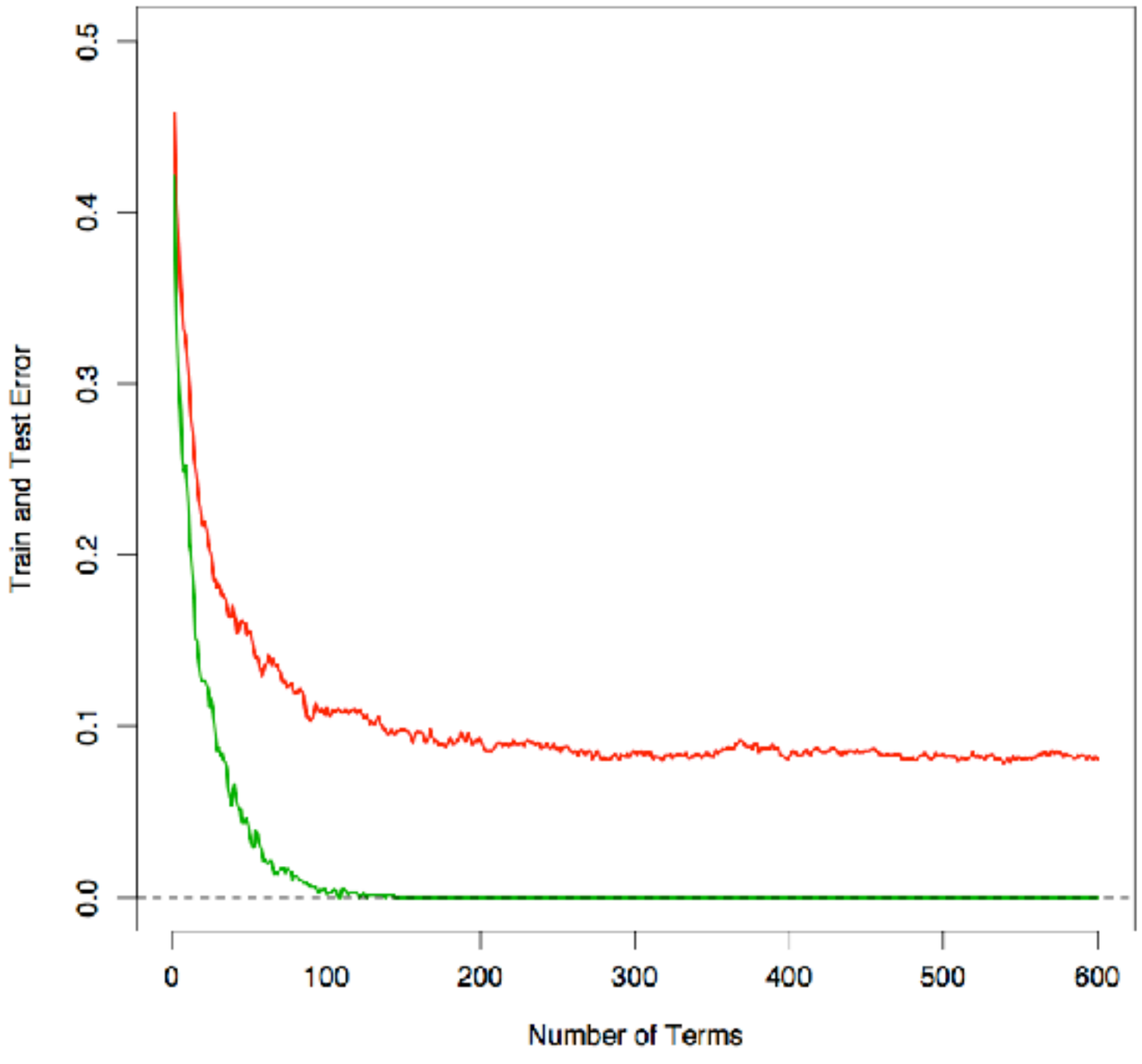
Toy example revisited



Boosting Stumps

A stump is a two-node tree, after a single split. Boosting stumps works remarkably well on the nested-spheres problem.

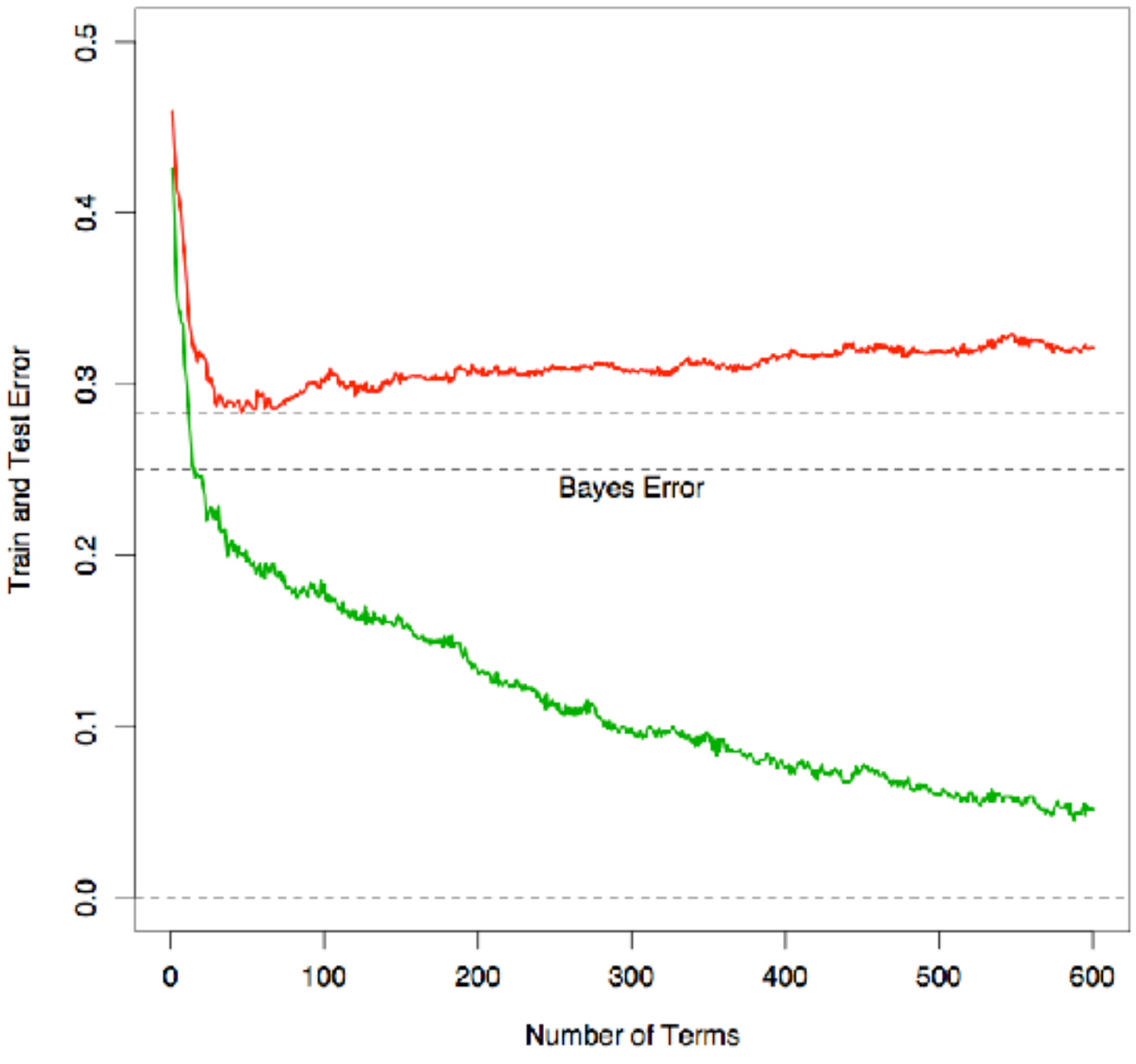
Toy example revisited



Training Error

- Nested spheres in 10-Dimensions.
- Bayes error is 0%.
- Boosting drives the training error to zero.
- Further iterations continue to improve test error in many examples.

Toy example revisited



Noisy Problems

- Nested Gaussians in 10-Dimensions.
- Bayes error is 25%.
- Boosting with stumps
- Here the test error does increase, but quite slowly.

Boosting as gradient descent

- AdaBoost stepwise minimizes a function of $y_i f_{\boldsymbol{\alpha}}(x_i) = y_i \sum_t \alpha_t h_t(\mathbf{x}_i)$

$$\mathcal{G}(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \exp\{-y_i f_{\boldsymbol{\alpha}}(\mathbf{x}_i)\}$$

- The gradient of $\mathcal{G}(\boldsymbol{\alpha}^{(t)})$ gives exactly the example weights used for AdaBoost:

$$\frac{\partial \mathcal{G}(\boldsymbol{\alpha}^{(t)})}{\partial f(\mathbf{x}_i)} \sim \exp\{-y_i f_{\boldsymbol{\alpha}}(\mathbf{x}_i)\} \sim d_i^{(t+1)}$$

- The hypothesis coefficient α_t is chosen, such that $\mathcal{G}(\boldsymbol{\alpha}^{(t)})$ is minimized:

$$\alpha_t = \underset{\alpha_t \geq 0}{\operatorname{argmin}}\, \mathcal{G}(\boldsymbol{\alpha}^{(t)}) = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$$

- **AdaBoost is a gradient descent method to minimizes $\mathcal{G}(\boldsymbol{\alpha})$.**

Convergence

Theorem 1 (Schapire et al. 1997) *Suppose AdaBoost generates hypotheses with weighted training errors* $\epsilon_1, \ldots, \epsilon_T$. *Then we have*

$$\sum_{i=1}^{N} \mathrm{I}(y_i \neq \mathrm{sign}(f_{Ens}(\mathbf{x}_i))) \leq 2^T \prod_{t=1}^{T} \sqrt{\epsilon_t(1-\epsilon_t)}$$

If $\epsilon_t < \frac{1}{2} - \frac{1}{2}\gamma$ (for all $t = 1, \ldots, T$), then the training error will decrease exponentially fast, i.e. will be **zero** after only

$$\frac{2\log(N)}{\gamma^2} = \mathcal{O}(\log(N))$$

iterations.

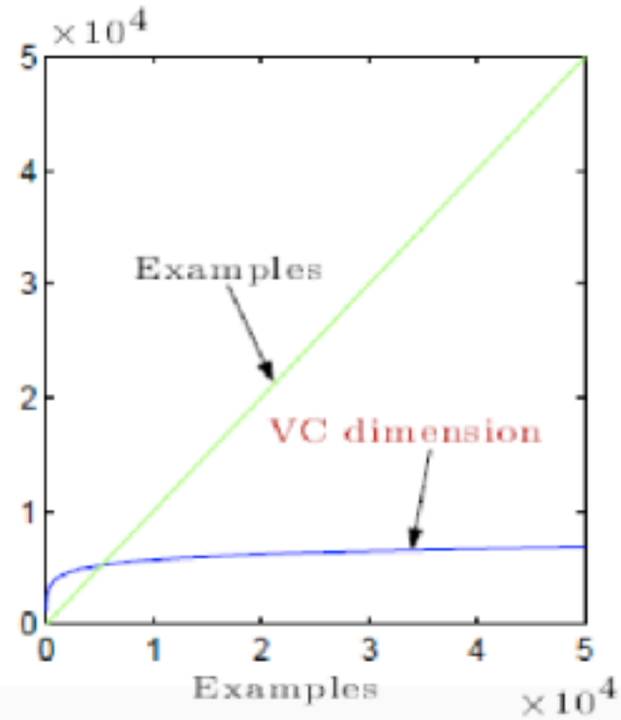
VC dimension

Let d be the VC dimension of the base hypothesis class $\mathcal{H}$.
Then the VC dimension of the class of combined functions is

$$d_{Ens}(N, \gamma) = \mathcal{O}\Big(d \underbrace{\frac{\log(N)}{\gamma^2}}_{\sim T} \log\left(\frac{\log(N)}{\gamma^2}\right)\Big) = \mathcal{O}\left(d \log(N) \log^2(N)\right).$$

An Example

- VC dimension $d = 2$
  (e.g. decision stumps)
- $\epsilon_t \leq 0.4 = \frac{1}{2} - \frac{1}{2}\gamma$
  $\Rightarrow \gamma \geq 0.2$



Theory

- properties of weak learner imply exponential convergence to a consistent hypothesis
- Fast convergence ensures small VC dimension of the combined hypothesis
- small VC implies small deviation from the empirical risk
- for any $\varepsilon > 0$ and $\delta > 0$ exists a sample size N, such that with probability $1 - \delta$ the expected risk is smaller than ε
- Any weak learner can be boosted to achieve an arbitrary high accuracy! ($\rightsquigarrow$ strong learner)

Similarities to SVM's

- AdaBoost maximizes the margin.
- Examples with stable large weights emerge during optimization

  → Support Vectors
- Formulation as a mathematical program similar to SVM possible

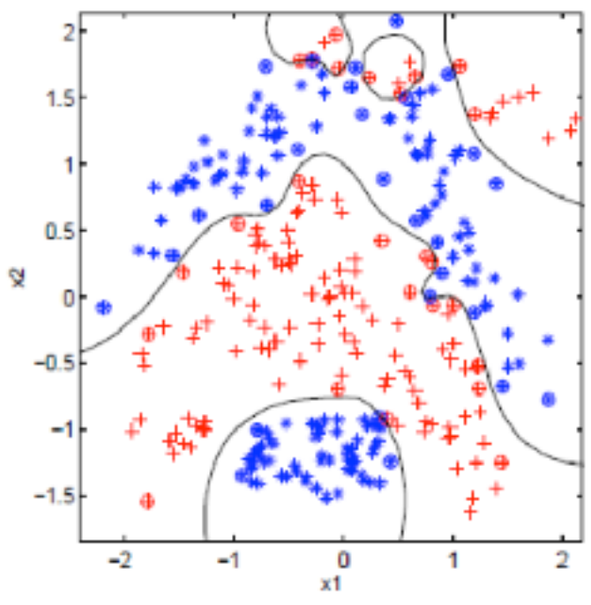
  → Regularized Boosting possible

Similarities to SVM's

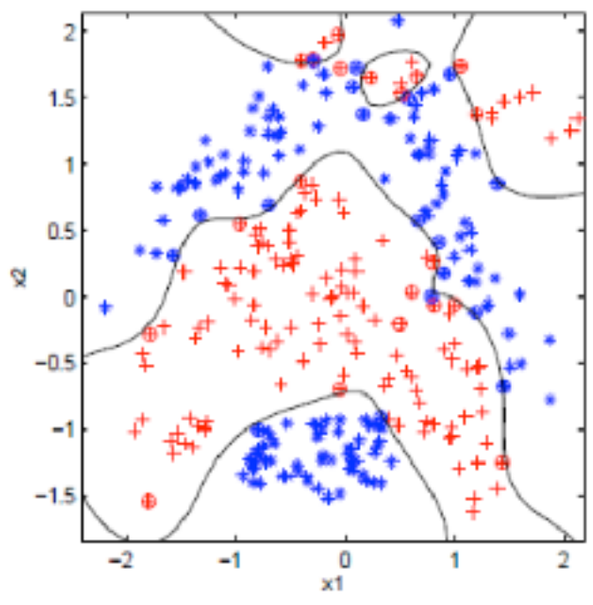
- AdaBoost maximizes the margin.
- Examples with stable large weights emerge during optimization

  → Support Vectors
- Formulation as a mathematical program similar to SVM possible

→ Regularized Boosting possible



AdaBoost's decision line



SVM's decision line

Real data

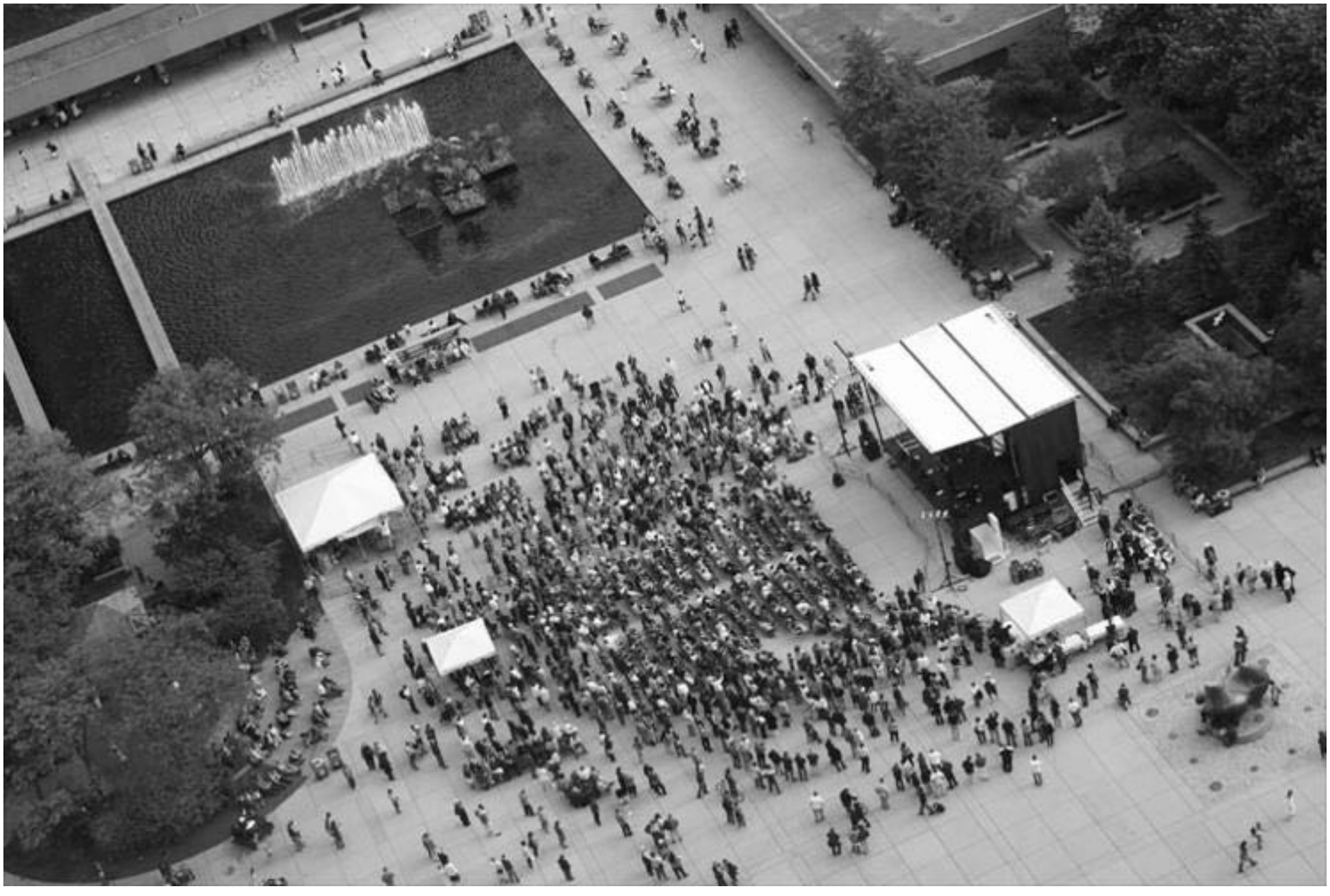
- 10 datasets (from UCI, DELVE and STATLOG repositories)
- Non binary problems partitioned into two-class problems.
- 100 partitions into test and training set (about 60%:40%).
- On each data sets we trained and tested all classifiers. Results are average test errors over 100 runs and standard deviations.
- Parameters estimated by 5-fold cross validation on first 5 realizations of dataset.
- For SVM we used Gaussian kernel
- For Boosting we used RBF networks as base learner

Real data

Experimental Results

	KNN	C4.5	RBF	AB	AB_R	SVM
Banana	$15.0_{\pm 1.0}$	$16.1_{\pm 2.8}$	$10.8_{\pm 0.6}$	$12.3_{\pm 0.7}$	$10.9_{\pm 0.4}$	$11.5_{\pm 0.7}$
B.Cancer	$28.4_{\pm 4.4}$	$24.6_{\pm 4.5}$	$27.6_{\pm 4.7}$	$30.4_{\pm 4.7}$	$26.5_{\pm 4.5}$	$26.0_{\pm 4.7}$
Diabetes	$28.9_{\pm 2.4}$	$26.0_{\pm 2.4}$	$24.3_{\pm 1.9}$	$26.5_{\pm 2.3}$	$23.8_{\pm 1.8}$	$23.5_{\pm 1.7}$
German	$28.9_{\pm 1.9}$	$28.1_{\pm 2.4}$	$24.7_{\pm 2.4}$	$27.5_{\pm 2.5}$	$24.3_{\pm 2.1}$	$23.6_{\pm 2.1}$
Heart	$15.8_{\pm 3.3}$	$20.4_{\pm 4.6}$	$17.6_{\pm 3.3}$	$20.3_{\pm 3.4}$	$16.5_{\pm 3.5}$	$16.0_{\pm 3.3}$
Ringnorm	$35.9_{\pm 1.3}$	$15.3_{\pm 1.5}$	$1.7_{\pm 0.2}$	$1.9_{\pm 0.3}$	$1.6_{\pm 0.1}$	$1.7_{\pm 0.1}$
F.Solar	$37.8_{\pm 2.8}$	$33.2_{\pm 1.9}$	$34.4_{\pm 2.0}$	$35.7_{\pm 1.8}$	$34.2_{\pm 2.2}$	$32.4_{\pm 1.8}$
Thyroid	$5.8_{\pm 2.8}$	$8.7_{\pm 3.3}$	$4.5_{\pm 2.1}$	$4.4_{\pm 2.2}$	$4.6_{\pm 2.2}$	$4.8_{\pm 2.2}$
Titanic	$25.5_{\pm 3.8}$	$22.9_{\pm 1.5}$	$23.3_{\pm 1.3}$	$22.6_{\pm 1.2}$	$22.6_{\pm 1.2}$	$22.4_{\pm 1.0}$
Waveform	$11.4_{\pm 0.8}$	$17.8_{\pm 1.0}$	$10.7_{\pm 1.1}$	$10.8_{\pm 0.6}$	$9.8_{\pm 0.8}$	$9.9_{\pm 0.4}$
Mean%	$2400_{\pm 6800}$	$1200_{\pm 2700}$	$5.8_{\pm 3.7}$	$13.4_{\pm 9.2}$	$2.7_{\pm 2.5}$	$2.9_{\pm 3.5}$

How many people?



Summary

- Ensemble learners combine simple learners to improve prediction.
- Easy to use.
- Typically perform well on real data.
- Based on intuitive concepts.
- But very strong links to theory and other methods (e.g. VC/SVM).

- Not much research anymore.
- But used in practice.

Margin

Theorem 2 *Suppose the base learning algorithm generates hypothesis with weighted training errors* $\epsilon_1, \ldots, \epsilon_T$. *Then we have for any* θ

$$P_{\mathbf{z}}(y f_{Ens}(\mathbf{x}) \leq \theta) \leq 2^T \prod_{t=1}^{T} \sqrt{{\epsilon_t}^{1-\theta}(1-\epsilon_t)^{1+\theta}}$$

Corollary 3 *If the base learning algorithm always achives* $\epsilon_t \leq \frac{1}{2} - \frac{1}{2}\gamma$ *then AdaBoost will generate a combined hyperplane with margin at least* $\frac{1}{2}\gamma$.

Margin bound

Theorem 4 *Let D be a distribution over $X \times \{\pm 1\}$ and let $\mathbf{Z}$ be a sample of N examples chosen independently at random according to D. Suppose the base-hypothesis space $\mathcal{H}$ has VC-dimension d, and let $\delta > 0$. Then with probability at least $1 - \delta$, the expected risk is bounded for $\theta > 0$ by*

$$R[f] \leq R^{\theta}_{emp}[f] + \mathcal{O}\left(\sqrt{\frac{d \log^2\left(\frac{N}{d}\right)}{\theta^2 N} + \frac{\log(1/\delta)}{N}}\right)$$

Compare to SVMs:

$$R[f] \leq R_{emp}[f] + \mathcal{O}\left(\sqrt{\frac{\log\left(N\theta^2\right)}{\theta^2 N} + \frac{\log(1/\eta)}{N}}\right).$$

- Bounds are independent of dimensionality of feature space