

Combining Classifiers and Feature Selection

Matteo Pardo

Sensor Lab, CNR-INFM Brescia

pardo@ing.unibs.it

<http://sensor.ing.unibs.it/>

Outline

- Bias-variance decomposition
- Ensemble methods
- Feature selection



Bias and variance definitions

- o Since the estimator is a random variable, it is possible to take its expectation (over the data it was generated from) or mean value:

$$E[\hat{\theta}] \doteq \int \hat{\theta}(x_1, \dots, x_N) p(x_1, \dots, x_N) dx_1, \dots, dx_N$$

- o The bias of the estimator of the parameter θ :

$$\textit{bias} = E[\hat{\theta}] - \theta$$

- o The variance:

$$\textit{variance} = E[\hat{\theta} - E[\hat{\theta}]]^2$$

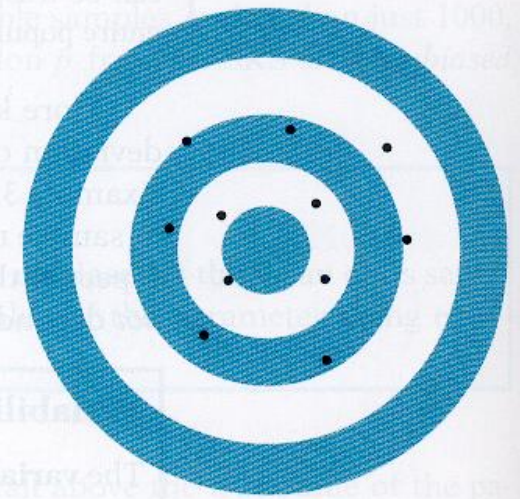


Bias and variance: example



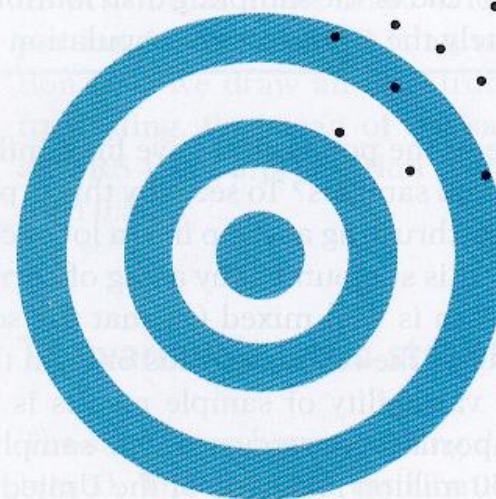
High bias, low variability

(a)



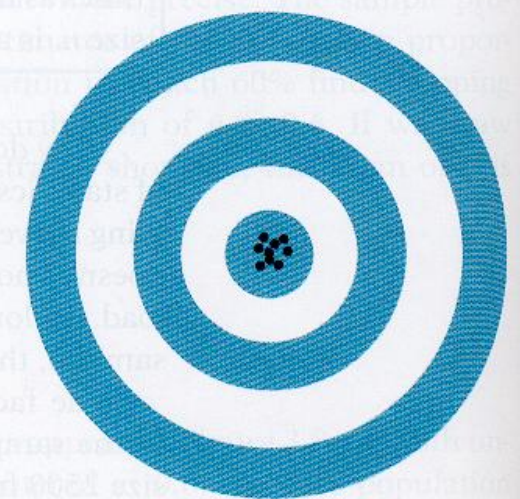
Low bias, high variability

(b)



High bias, high variability

(c)



The ideal: low bias, low variability

(d)

Bias-variance decomposition (1)

- Until now we saw **point** estimators, in learning problems we estimate **functions** $y(x)$
- The generalization error can be **decomposed** in a sum of a bias and a variance terms
- Hypothesis: a generic **regression** model trained using sum-of-squares error



Bias-variance decomposition (2)

$\hat{y}(x, w^*)$ is itself a random variable since it depends on the data

- o For every **fixed** point x the correspondence holds:

$$\theta \rightarrow \bar{y}(x)$$

$$\hat{\theta} \rightarrow \hat{y}(x, w),$$

- o Therefore it is well possible, for every point x , to consider the average value over the data distribution of the error

$$(\hat{y}(x, w(D)) - \bar{y}(x))^2$$



Bias-variance decomposition (3)

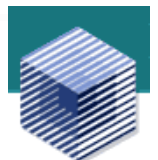
$$E_D[(\hat{y}(x, w) - \bar{y}(x))^2] = \text{bias} + \text{variance}$$
$$= \{E_D[\hat{y}(x, w)] - \bar{y}(x)\}^2 + E_D[\{\hat{y}(x, w) - E_D[\hat{y}(x, w)]\}^2]$$

- Finally we can average the whole expected error by integrating over x weighted by $p(x)$:

$$(\text{bias})^2 = \frac{1}{2} \int \{E_D[\hat{y}(x, w)] - \bar{y}(x)\}^2 p(x) dx$$

$$\text{variance} = \frac{1}{2} \int E_D[\{\hat{y}(x, w) - E_D[\hat{y}(x, w)]\}^2] p(x) dx.$$

- Bias-variance decomposition less straightforward for classification



Next 5 slides taken from

Lectures on
**Advanced Topics on
Machine Learning**

Joachim M. Buhmann
*Institute for Computational Science
Swiss Federal Institute of Technology ETHZ,
8092 Zurich, Switzerland*



If you consider the whole error

$$\begin{aligned} & \mathbb{E}_D \mathbb{E}_{X,Y} \left(\hat{f}(X) - Y \right)^2 \\ &= \mathbb{E}_D \mathbb{E}_X \left(\hat{f}(X) - \mathbb{E}(Y|X) \right)^2 + \mathbb{E}_{X,Y} \left(Y - \mathbb{E}(Y|X) \right)^2 \\ &= \mathbb{E}_X \mathbb{E}_D \left(\hat{f}(X) - \mathbb{E}_D \hat{f}(X) \right)^2 && \text{(variance)} \\ &+ \mathbb{E}_X \left(\mathbb{E}_D \hat{f}(X) - \mathbb{E}(Y|X) \right)^2 && \text{(bias)}^2 \\ &+ \mathbb{E}_{X,Y} \left(Y - \mathbb{E}(Y|X) \right)^2 && \text{(noise)} \end{aligned}$$



Combining Regressors - Bias

Set of estimators: $\hat{f}_1(x), \hat{f}_2(x), \dots, \hat{f}_B(x)$

Simple average: $\hat{f}(x) = \frac{1}{B} \sum_{i=1}^B \hat{f}_i(x)$

Bias:

$$\begin{aligned} B_f(x) &= \mathbb{E}_D \hat{f}(x) - \mathbb{E}(Y|x) \\ &= \frac{1}{k} \sum_{i=1}^k \mathbb{E}_D \hat{f}_i(x) - \mathbb{E}(Y|x) \end{aligned}$$

Unbiased estimators remain unbiased



Combining Regressors - Variance

$$\begin{aligned}\mathbb{V}\{f(x)\} &= \mathbb{E}_D \left(\hat{f}(X) - \mathbb{E}_D \hat{f}(X) \right)^2 \\ &= \mathbb{E}_D \left(\frac{1}{B} \sum_{i=1}^B \hat{f}_i(x) - \frac{1}{B} \sum_{i=1}^B \mathbb{E}_D \hat{f}_i(x) \right)^2 \\ &= \mathbb{E}_D \left(\frac{1}{B} \sum_{i=1}^B (\hat{f}_i(x) - \mathbb{E}_D \hat{f}_i(x)) \right)^2 \\ &= \frac{1}{B^2} \sum_{i=1}^B \mathbb{V} \left\{ \hat{f}_i(x) \right\} + \frac{1}{B^2} \sum_{i \neq j} \sum \text{Cov} \left\{ \hat{f}_i(x), \hat{f}_j(x) \right\}\end{aligned}$$



Combining Regressors – Variance 2

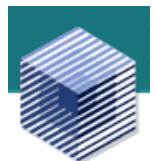
Assume:

Covariances small: $\text{Cov} \left\{ \hat{f}_i(x), \hat{f}_j(x) \right\} \approx 0$

Variances similar: $\mathbb{V} \left\{ \hat{f}_i(x) \right\} \approx v$

Then

$$\mathbb{V}\{f(x)\} \approx \frac{v}{B}$$



Similarly for classifiers

Input: A pool of binary classifiers c_1, c_2, \dots, c_B

Objective: A composite classifier

$$\hat{c}_B(x) = \text{sgn} \left(\sum_{i=1}^k \alpha_i c_i(x) \right)$$

Question: When can this procedure succeed?

Require **diversity** of the classifiers

Diversity: Use **different subsets** of the data for each c_i

Use **different features**

Decorrelate classifiers during training



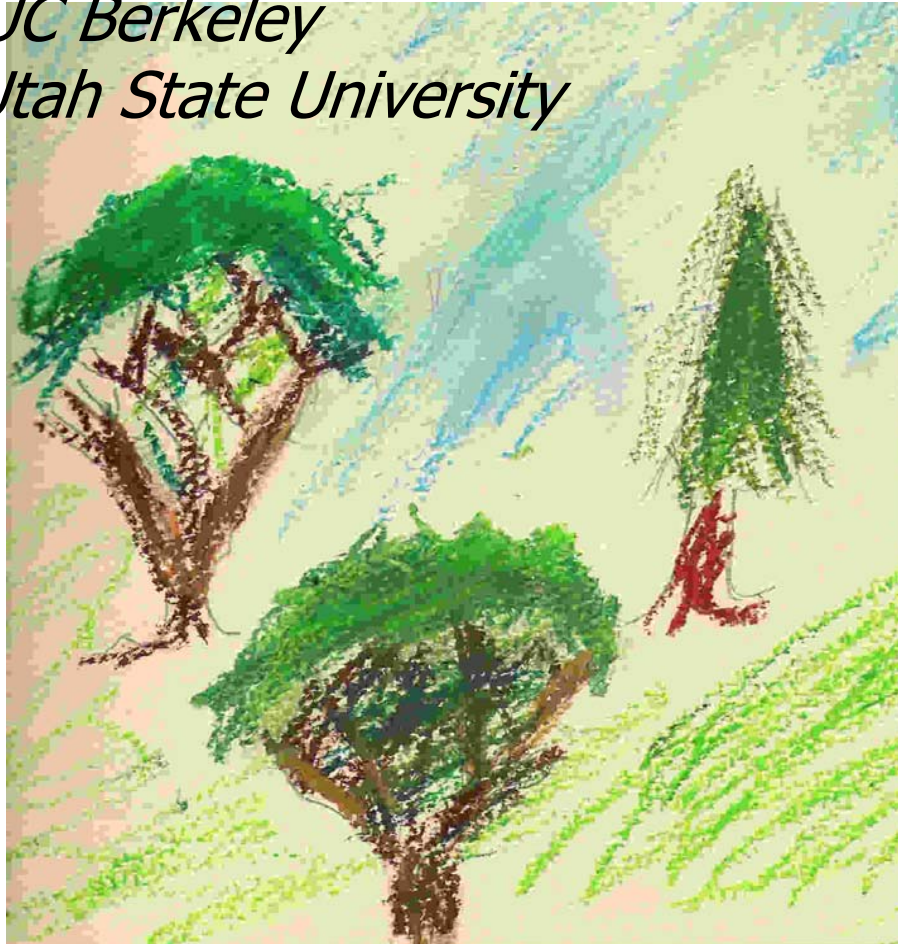
Ensemble Methods

- Train several sufficiently diverse predictors, and use an aggregation scheme
- Result is lower bias and variance.
- **Bagging** (bootstrap aggregation) proposed by Breiman (1996)
- **Random forests** (Breiman 2001): bagging on trees with a twist
- **Arcing** (adaptive resampling and combining) and **Boosting** extend bagging (Freund and Schapire, 1995): focus on examples incorrectly predicted



Random Forests for Scientific Discovery

Leo Breiman, UC Berkeley
Adele Cutler, Utah State University

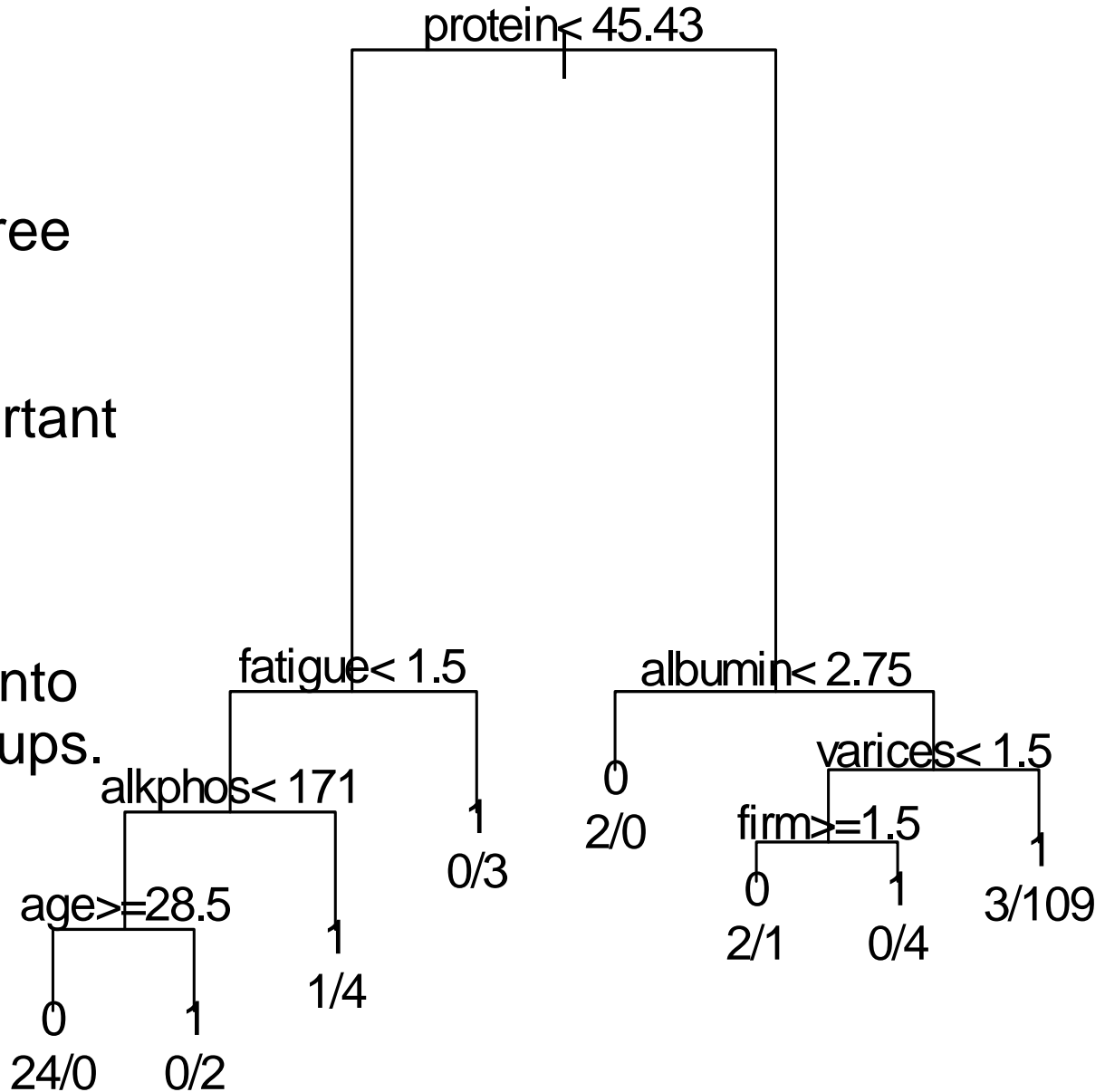


Arguably one of the most successful tools of the last 20 years. Why?

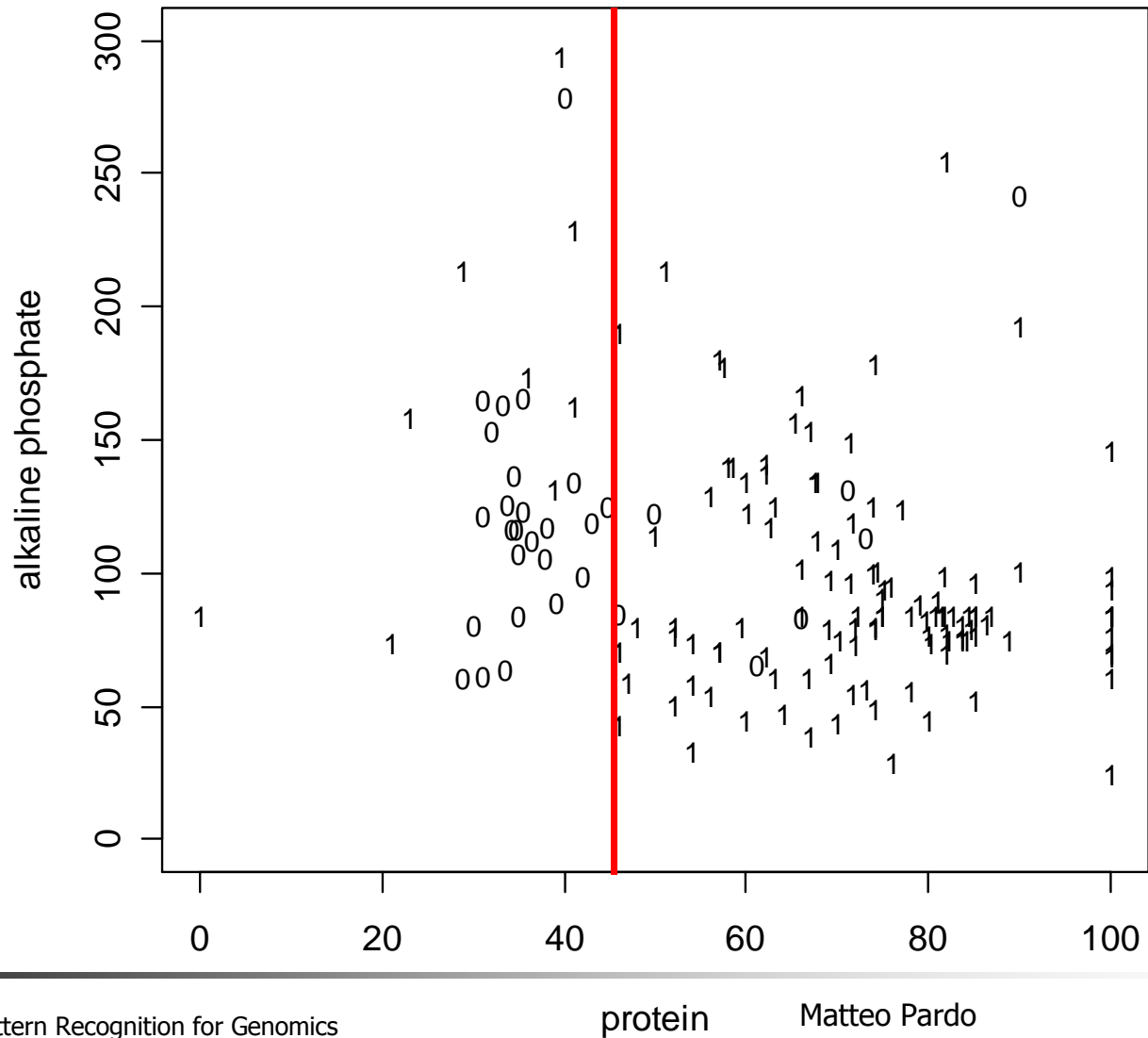
1. Applicable to both classification and regression problems with no formal assumptions on the data structure.
2. Can be applied to large datasets.
3. Handles missing data effectively.
4. Deals with categorical variables efficiently.



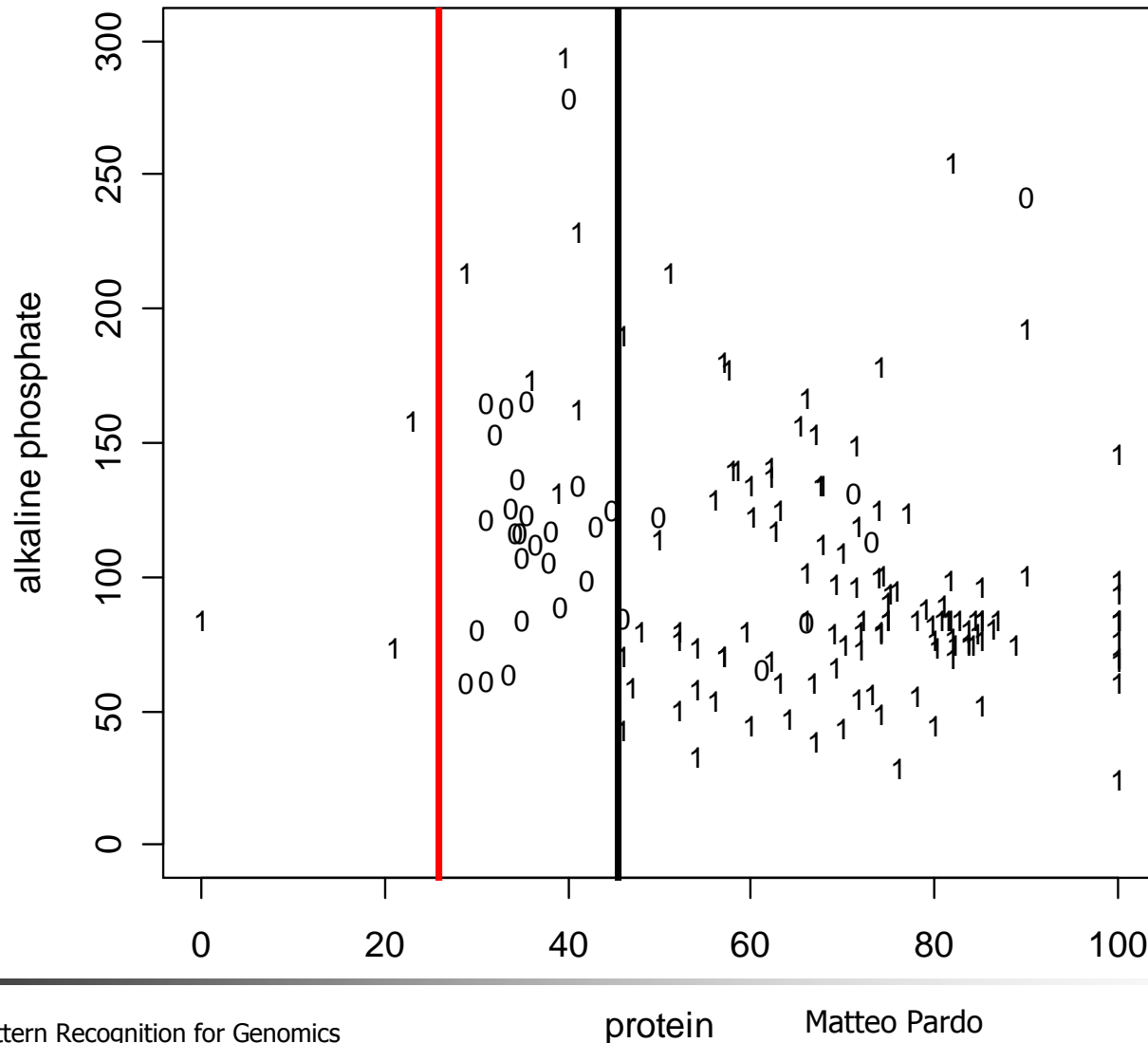
5. The picture of the tree can give valuable insights into which variables are important and where.
6. The terminal nodes suggest a natural clustering of data into homogeneous groups.



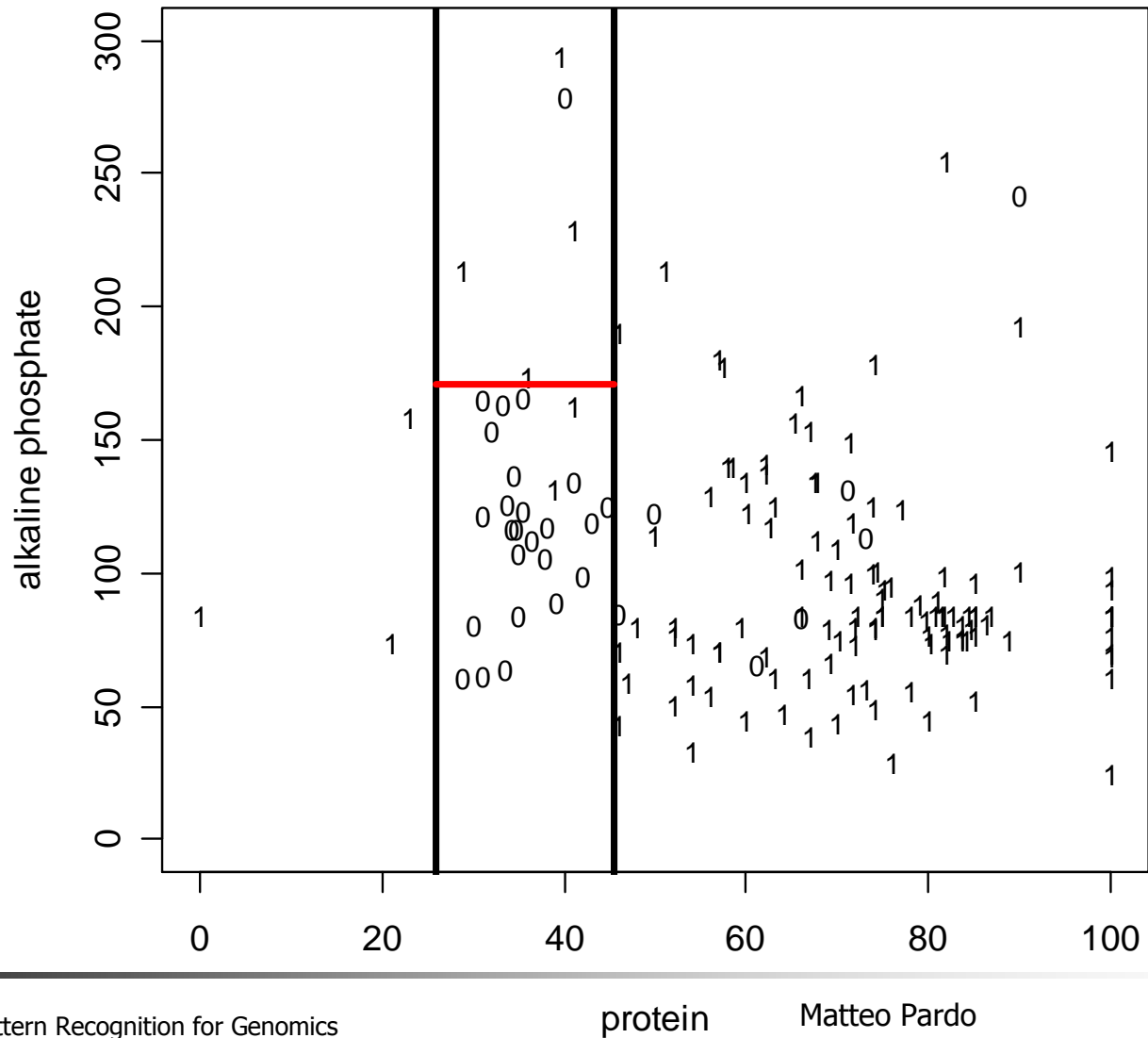
CART at work



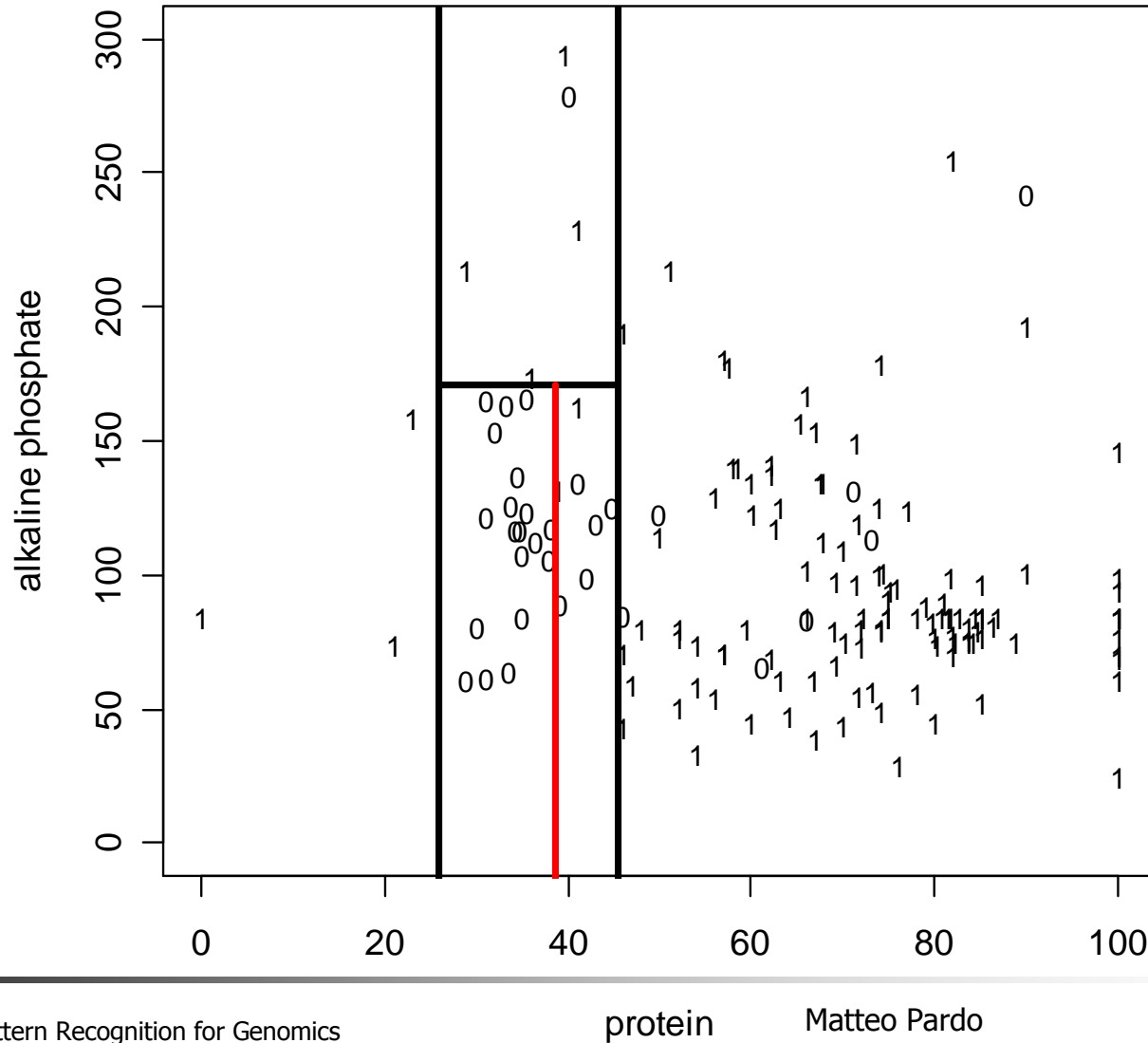
CART at work



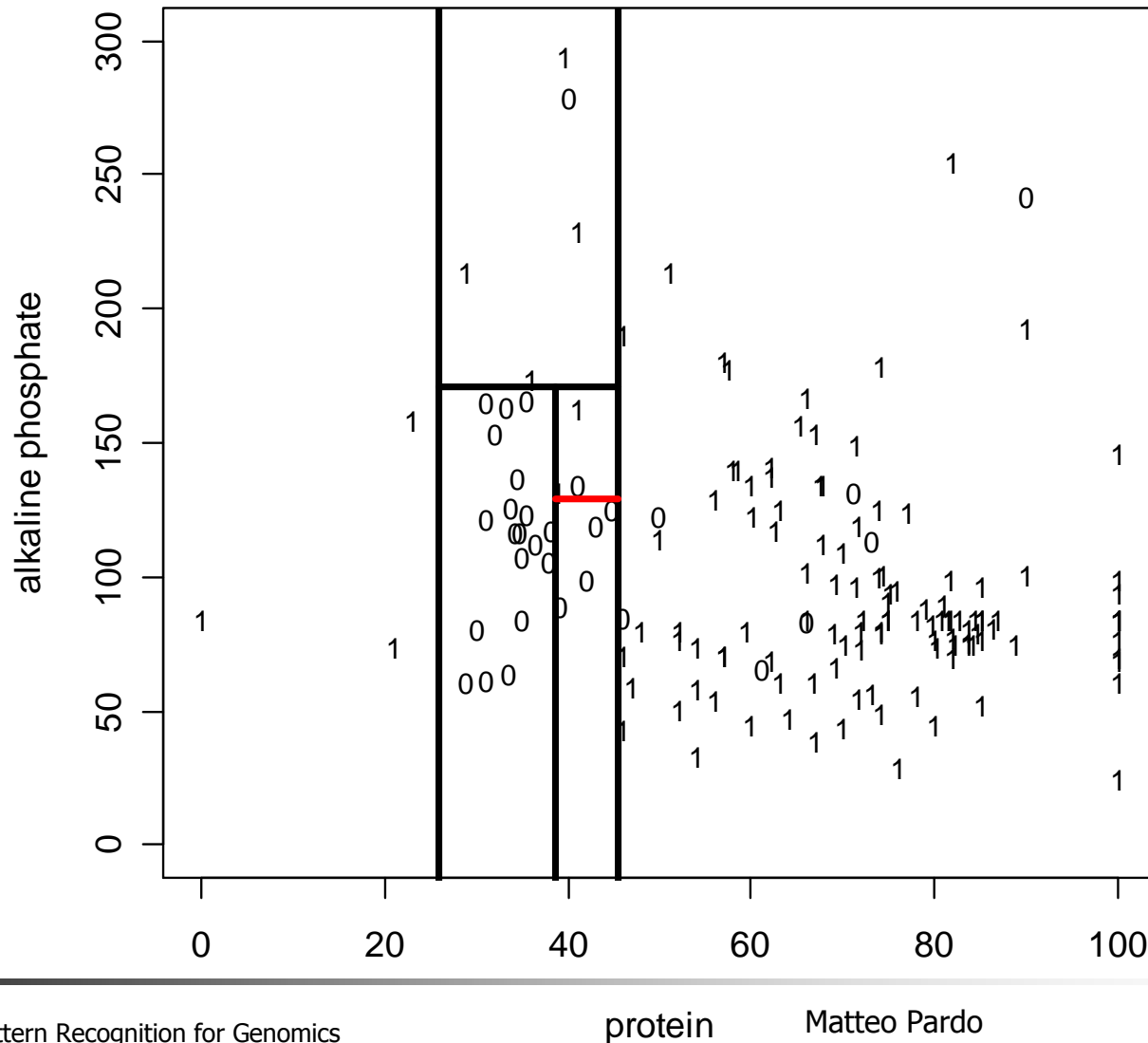
CART at work



CART at work



CART at work



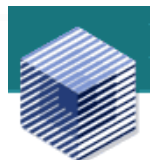
Drawbacks of CART

- *Accuracy* - current methods, such as support vector machines and ensemble classifiers often have 30% lower error rates than CART.
- *Instability* – if we change the data a little, the tree picture can change a lot. So the interpretation is not as straightforward as it appears.
- Today, we can do better!



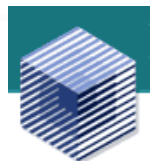
Random Forests

- Grow each tree on an independent bootstrap sample from the training data.
- At each node:
 1. Randomly select m variables out of all M possible variables (independently for each node).
 2. Find the best split on the selected m variables.
- Grow the tree to maximum depth.
- Vote or average the trees to get predictions.



Random Forests

1. Excellent accuracy.
 - In tests on collections of data sets it has accuracy at least as good as the best known machine learning methods.
2. Fast.
 - With 100 variables, 100 trees in a forest can be grown in the same time as growing 3 single CART trees.
3. Does not overfit as we add more trees.

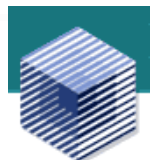


(ctnd)

4. Handles

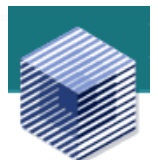
- thousands of variables
- many-valued categoricals
- extensive missing values
- badly unbalanced data sets.

5. Gives an internal unbiased estimate of test set error as trees are added to the ensemble.



Out-of-bag Data

- For each tree in the forest, we select a bootstrap sample from the data.
- The bootstrap sample is used to grow the tree.
- The remaining data are said to be “out-of-bag” (about one-third of the cases).
- The out-of-bag data serve as a test set for the tree grown on the bootstrap sample.



The out-of-bag Error Estimate

- Think of a *single case* in the training set.
- It will be out-of-bag in about one-third of the trees.
- Each time it is out of bag, pass it down the tree and get a predicted class.
- The RF prediction is the class that is chosen the most often.
- For each case, the RF prediction is either correct or incorrect.
 - Average over the cases within each class to get a *classwise* out-of-bag error rate.
 - Average over all cases to get an *overall* out-of-bag error rate.



The out-of-bag error Estimate

- For example, suppose we fit 1000 trees, and a case is out-of-bag in 339 of them, of which:
 - 283 say "class 1"
 - 56 say "class 2"
- The RF predictor is class 1.
- The out-of-bag error rate (%) is the percentage of the time that the RF predictor is correct.



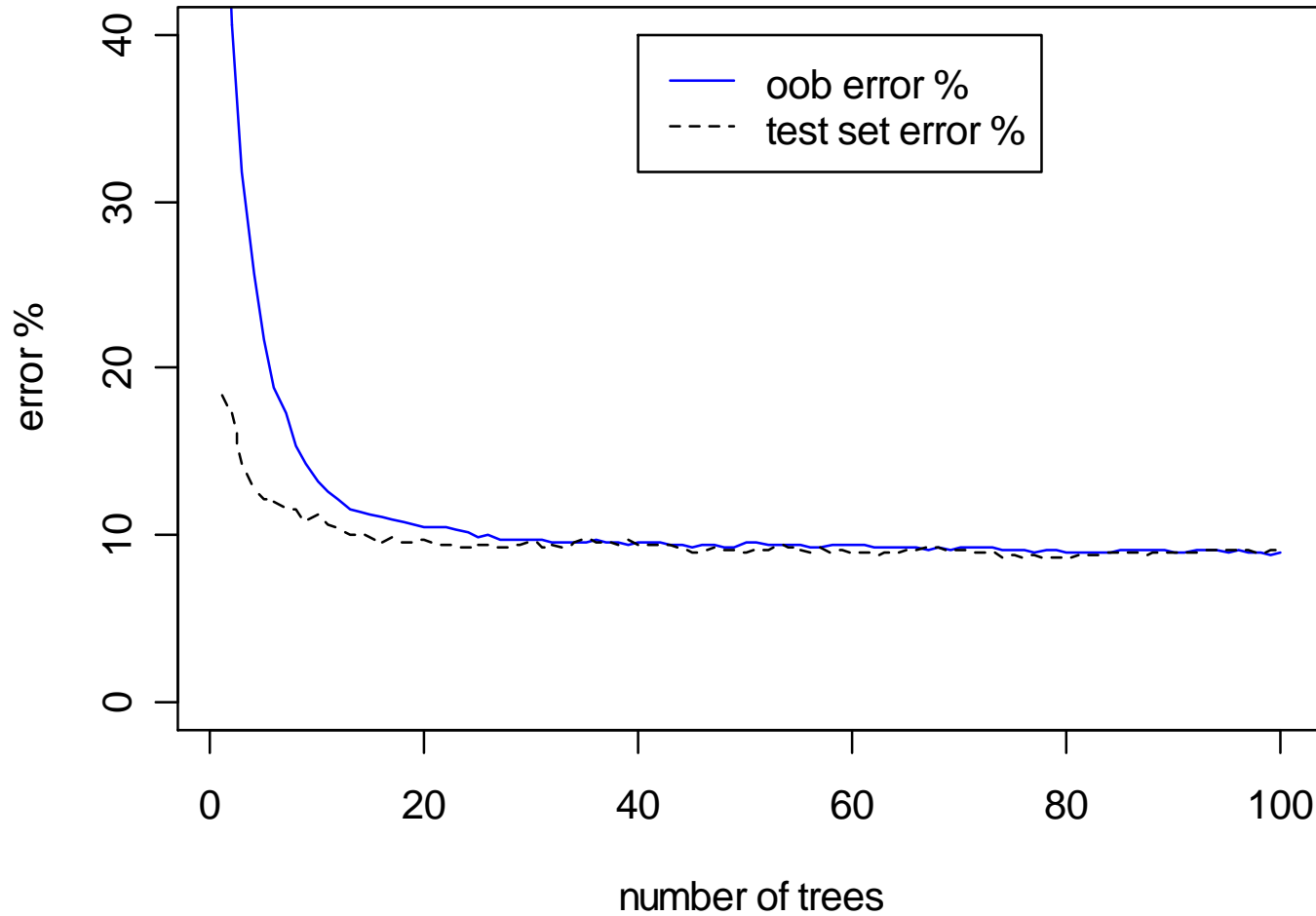
Illustration – Satellite data

- Training set: 4435 cases.
- Test set: 2000 cases.
- 36 variables.
- 100 trees.



The out-of-bag error rate is larger at the beginning because there are not enough trees for good predictions.

Error rates, oob and test, satellite data



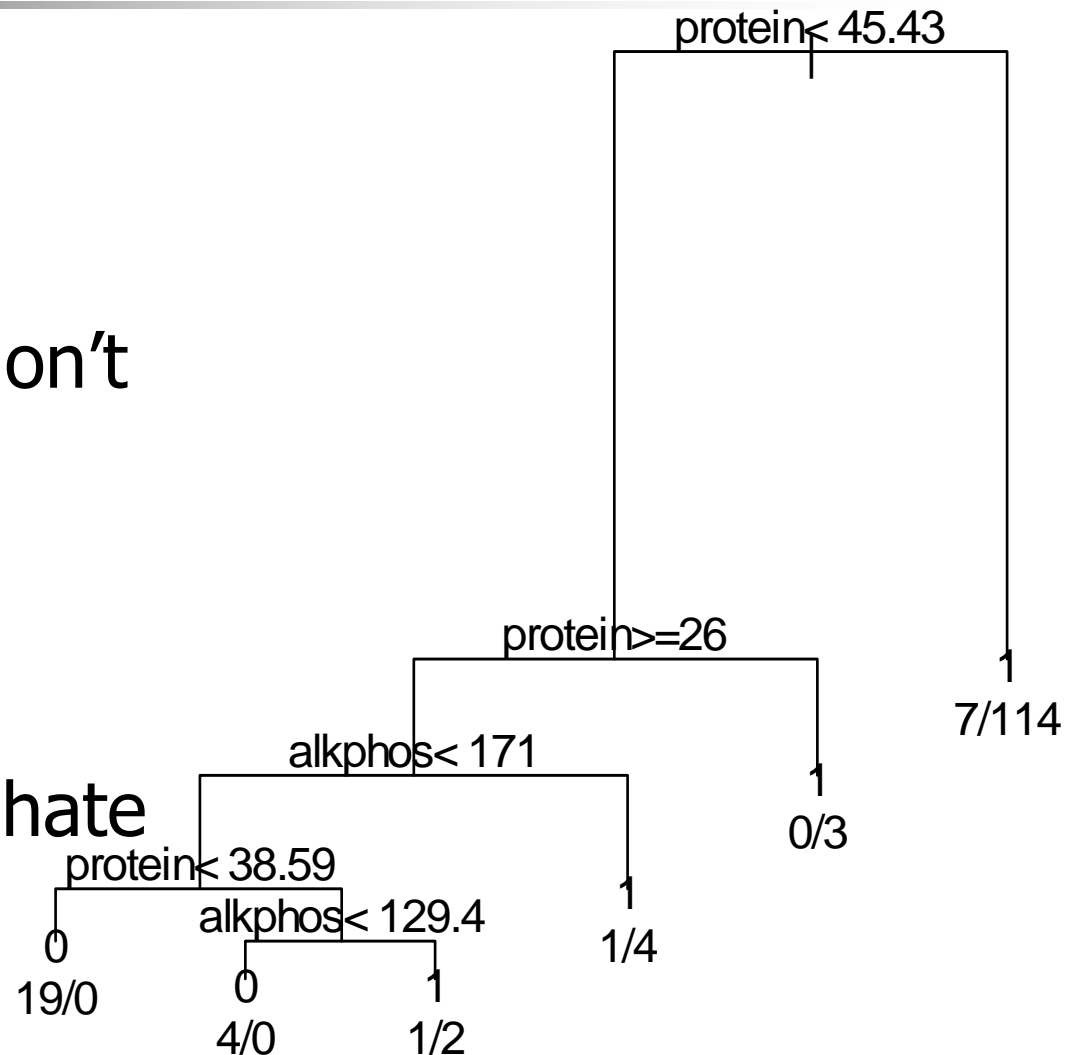
Variable Importance

- We usually think about variable importance as an overall measure. In part, this is probably because we fit models with global structure (linear regression, logistic regression).
- In CART, variable importance is local.



LOCAL Variable Importance

- Different variables are important in different regions of the data.
- If protein is high, we don't care about alkaline phosphate. Similarly if protein is low. But for intermediate values of protein, alkaline phosphate is important.



Variable importance for a single class 2 case (i.e. instance from class 2, predicted class shown)

TREE	No permutation	Permute variable 1	...	Permute variable m
1	2	2	...	1
3	2	2	...	2
4	1	1	...	1
9	2	2	...	1
...
992	2	2	...	2
% Error	10%	11%	...	35%

Variable Importance

- For case i and variable j find
 - error rate with variable j permuted
 - error rate with no permutation
 - do the difference

where the error rates are taken over all trees for which case i is out-of-bag.

- Average over all cases i (or class-wise)



Scoring variables

- The principle of permuting the values of the objects for each variable and look at how this changes the predictions is the same
- There are different ways to score the changes
- E.g. the margin:
 - For the n th case in the data, its margin at the end of a run is the proportion of votes for its true class minus the maximum of the proportion of votes for each of the other classes.
 - The measure of importance of the m th variable is the average lowering of the margin across all cases when the m th variable is randomly permuted.



Illustration – Breast Cancer data

- 699 cases, 9 variables, two classes
- Initial out-of-bag error rate is 3.3%

Added 10,000 independent standard normals to each case, making 10,009 variables.



The twelve most important variables, chosen by RF

variable #	raw score	z-score	significance
6	3.274	0.936	0.175
3	3.521	0.910	0.181
2	3.484	0.902	0.183
1	2.369	0.898	0.185
7	2.811	0.879	0.190
8	2.266	0.847	0.199
5	2.164	0.829	0.204
4	1.853	0.814	0.208
9	0.825	0.700	0.242
8104	0.016	0.204	0.419
430	0.005	0.155	0.438
5128	0.004	0.147	0.441



Boosting in words

- Fit many base learners to reweighted versions of the training data:
 - Determine a base learner
 - Look at the misclassified samples and give them a greater weight in the determination of the subsequent learner
- Classify by weighted majority vote
- ❖ Boosting **adaptively** concentrates on the harder examples (boundary)
- ❖ Reduces the classification bias and variance
- ❖ Has been very effective in real world classification tasks
- ❖ Can overfit very noisy datasets (outliers)

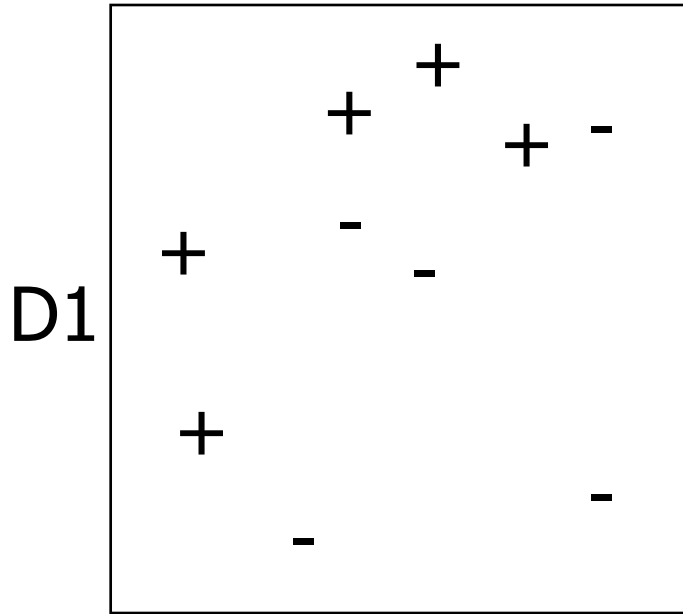


Boosting: pseudocode (AdaBoost)

1. Start with weights $w_i = 1/N, i=1,\dots,N; y_i \in \{1,-1\}$
2. Repeat for $m=1,\dots,M$:
 - a) Estimate the base (weak) learner $f_m(x)$ from the training data with weights w_i
 - b) Compute the weighted misclassification error $e_m = \sum_j(w_j)$; j index of misclassified samples
 - c) Compute the weight of the m -th classifier $f_m(x)$: $c_m = \log((1 - e_m) / e_m)$
 - d) Change the weights of the misclassified examples: $w_j = w_j \exp(c_m)$ and renormalize so that $\sum(w_i) = 1$
3. Output weighted majority classifier:
 $C(x) = \text{sign}[\sum_m(w_m f_m(x))]$

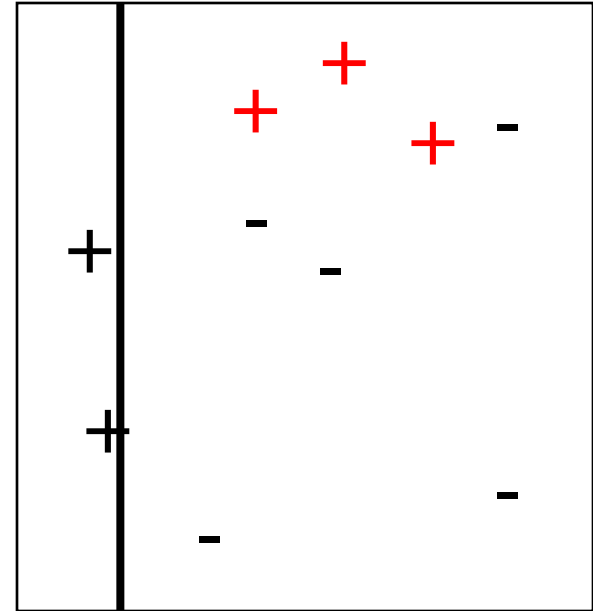


Toy example (Freund and Schapire)

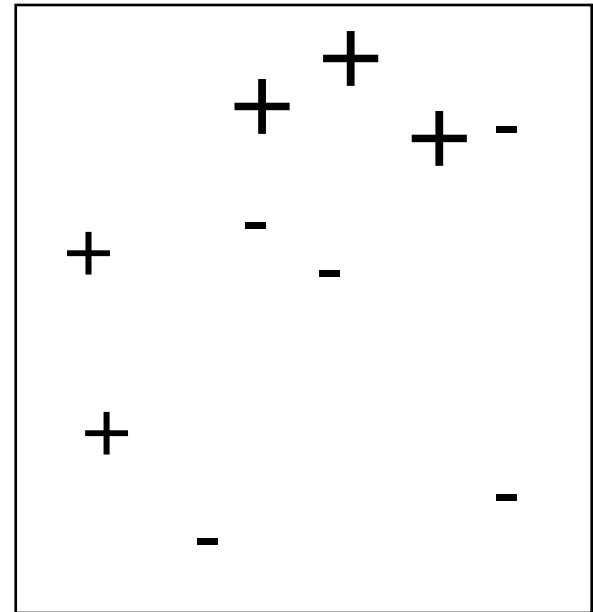


$e_1=0.3$
 $c_1=0.42$

Round 1

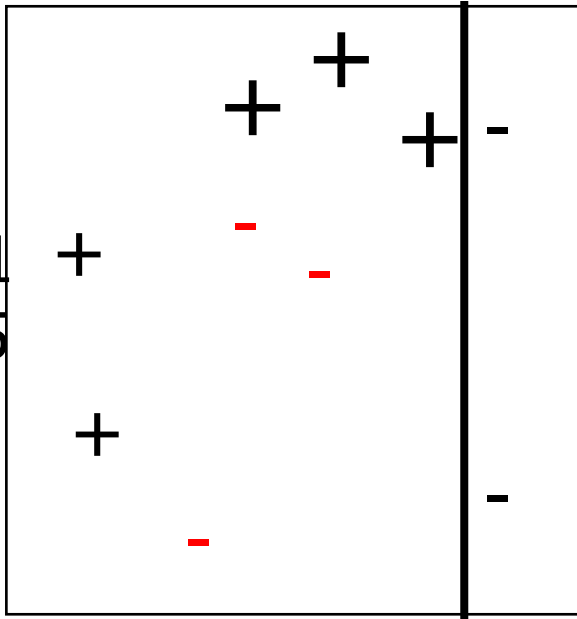


D2



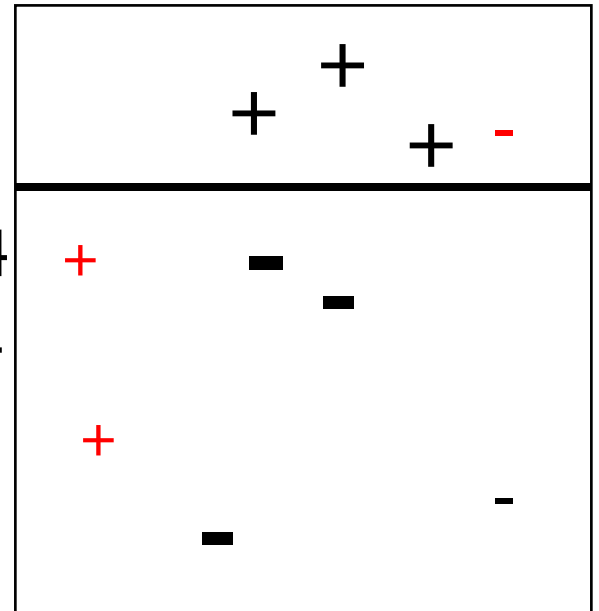
Round 2

$e1=0.21$
 $c1=0.65$

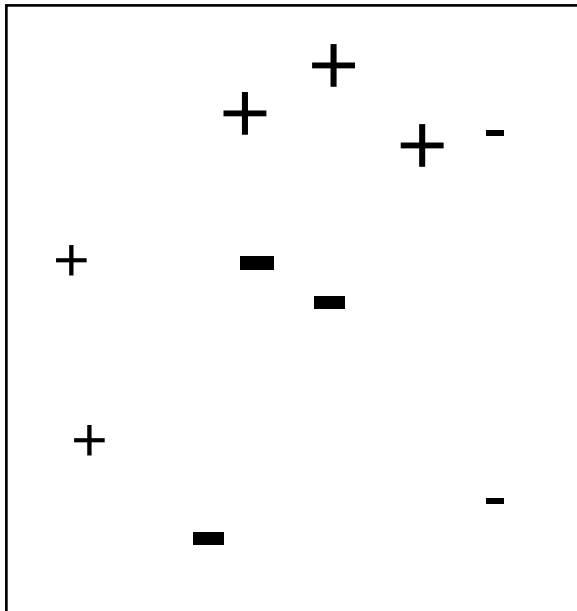


Round 3

$e1=0.14$
 $c1=0.92$

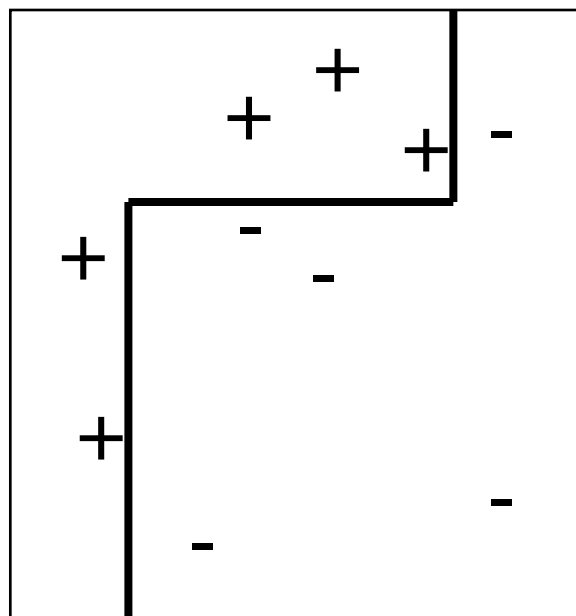


D3



Final classifier

$$C = \text{sign}(0.42 \begin{array}{|c|} \hline \\ \hline \end{array} + 0.65 \begin{array}{|c|} \hline \\ \hline \end{array} + 0.92 \begin{array}{|c|} \hline \\ \hline \end{array}) =$$



Example: e-nose discrimination between coffee blends

A single MLP

hidden #	Percent error rate on different runs						Best	Mean	Stdev
5	20.97	19.35	16.13	19.35	20.97	16.13	16.13	18.82	2.01
7	11.29	14.52	14.52	11.29	19.35	19.35	11.29	15.05	3.31
9	16.13	17.74	16.13	17.74	17.74	17.74	16.13	17.20	0.76

Boosted MLPs

hidden #	Percent error rate on different runs						Best	Mean	Stdev
5	9.68	11.29	9.68	11.29	11.29	11.29	9.68	10.75	0.83
7	6.45	9.68	9.68	9.68	6.45	9.68	6.45	8.60	1.67
9	11.29	9.68	6.45	9.68	6.45	11.29	6.45	9.14	2.20



Example: e-nose discrimination between coffee monovarieties

A single MLP

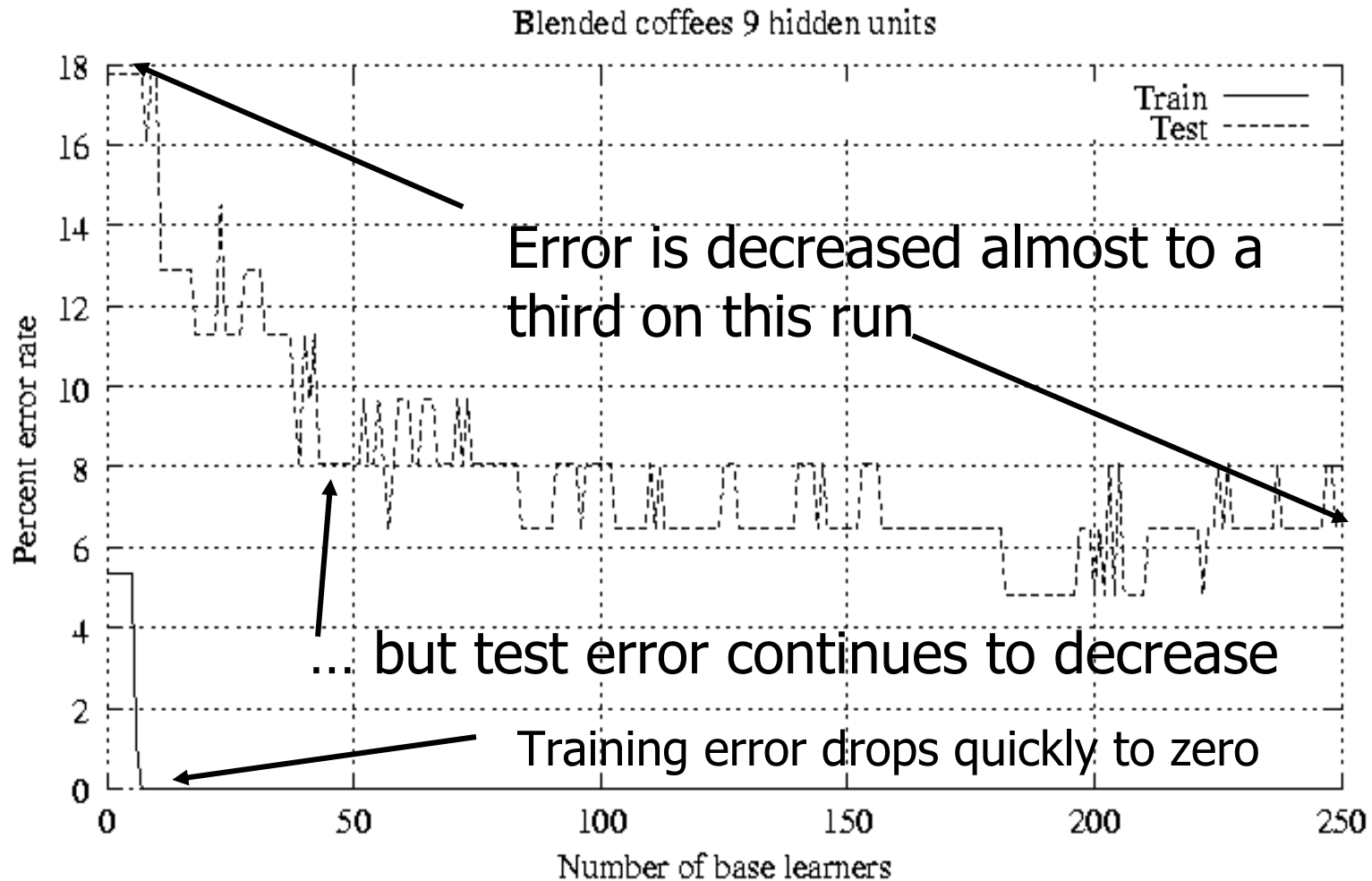
hidden #	Percent error rate on different runs						Best	Mean	Stdev
	20	21.43	23.21	21.43	21.43	21.43	23.21	21.43	22.02
30	23.21	23.21	21.43	23.21	21.43	23.21	21.43	22.62	0.84
40	25.00	25.00	23.21	23.21	23.21	23.21	23.21	23.81	0.84

Boosted MLPs

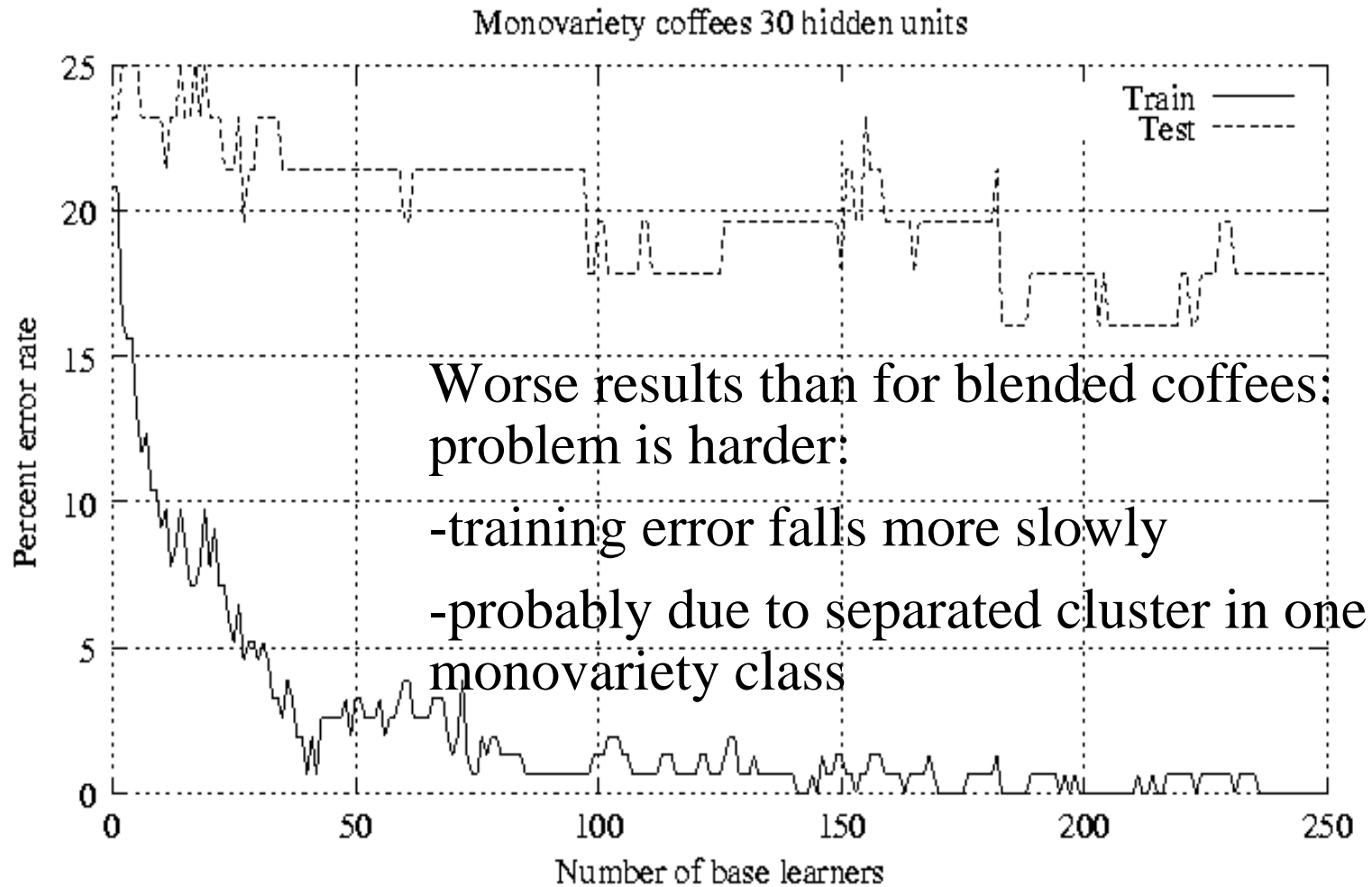
hidden #	Percent error rate on different runs						Best	Mean	Stdev
	20	21.43	21.43	19.64	21.43	19.64	21.43	19.64	20.83
30	21.43	17.86	19.64	17.86	19.64	21.43	17.86	19.64	1.60
40	23.21	17.86	19.64	19.64	23.21	17.86	17.86	20.24	2.44



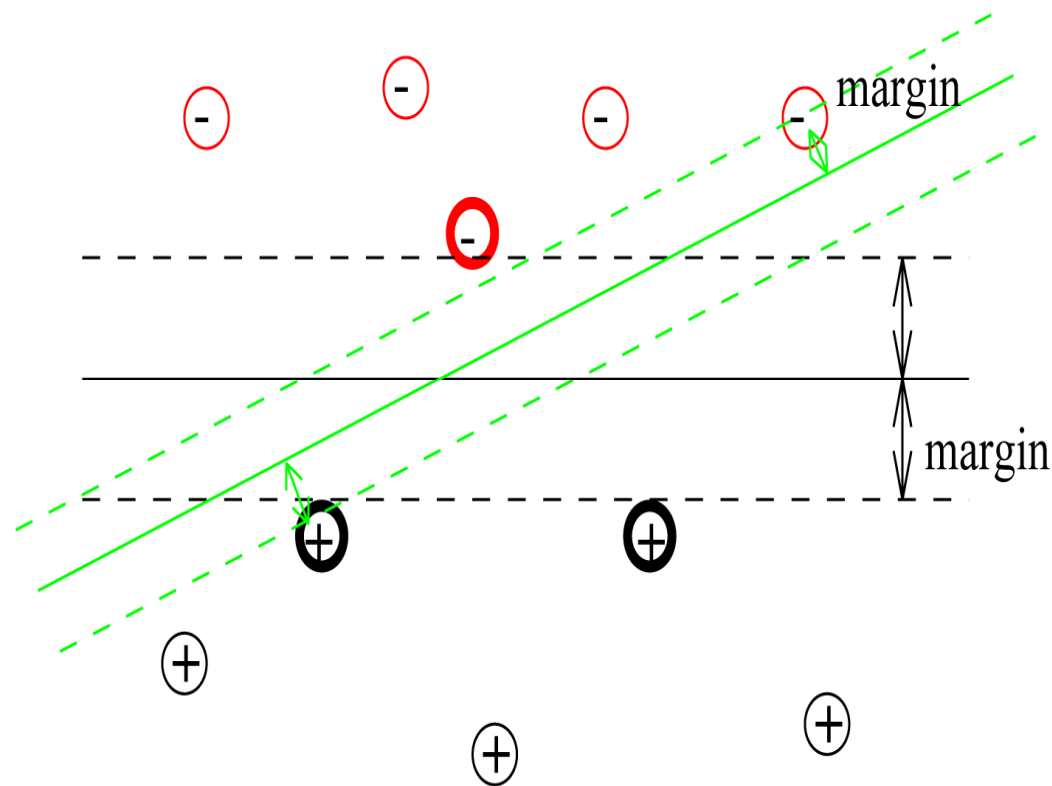
Results: graphs: blends



Results: graphs: monovarieties



Comments: why does the test error continue to decrease?



This behavior can be explained in the framework of **large margin classifiers**:

- Boosting tries not only to reduce the training error but to reduce the margin, that is the distance between the separating surface and the classes
- The margin is a key concept in machine learning: e.g. SVM try to optimize it directly



Motivation: datasets are a-changing

- Special issue on “Relevance”: Artificial Intelligence [Volume 97, Issues 1-2](#), Pages 1-402 (December 1997).
Editorial: “The relevance of relevance”
- Special issue on “Variable Selection”: Journal of Machine Learning Research, Vol. 3 Issue 7/8 (October 2003)

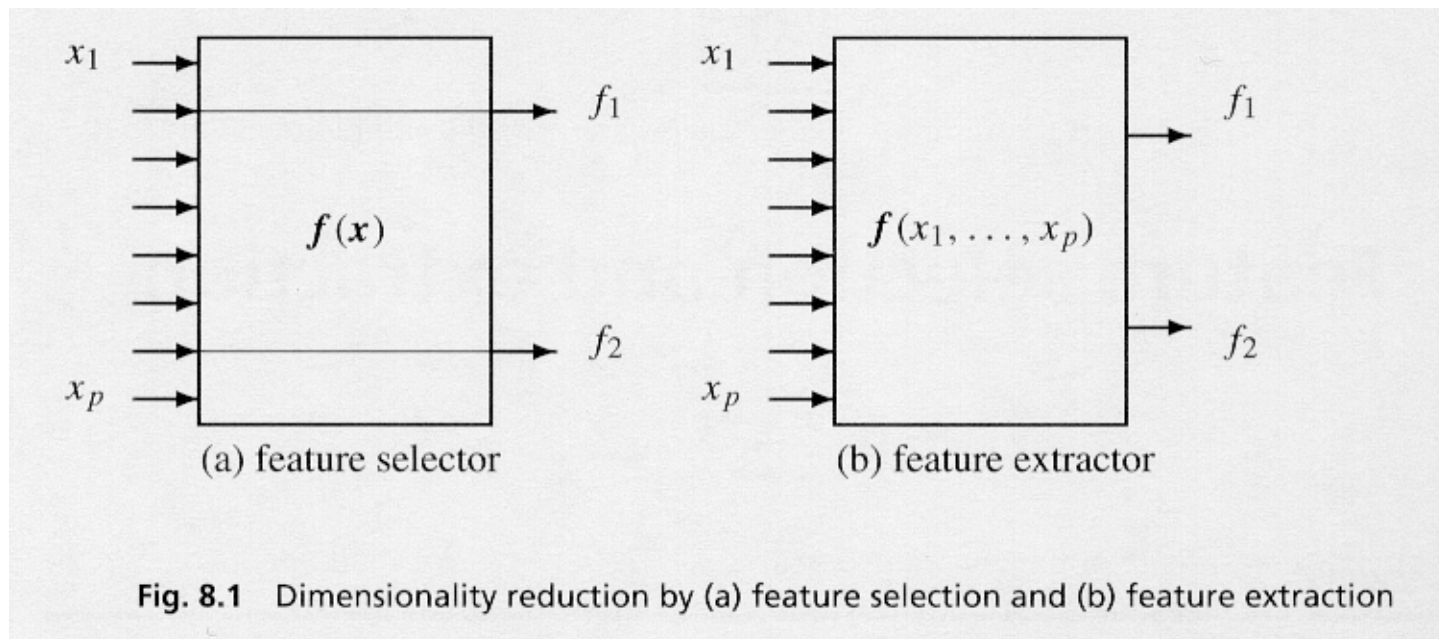
Difference:

- In 1997 few domains used more than 40 features;
- In 2003 most papers explore domains with 10^2 to 10^4 variables



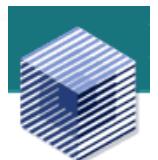
Difference between feature selection and extraction

- Selection: subset of original features
- Extraction: generic function of original features



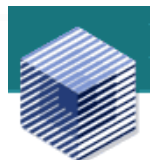
Why would I want to reduce dimensionality?

- Dimensionality reduction in general:
 - Improved classification performance through more stable representation (avoid **curse of dimensionality**)
 - Graphical representation (in 2D, e.g. PCA)
- Feature selection in particular:
 - **Easier interpretation**, e.g. possibility to discover underlying structure:
 - E-nose: which sensors contribute more to odor discrimination (not necessarily the most sensitive)?
 - Microarray: which genes contribute more to illness discrimination (not necessarily the most diff. expressed)?
 - Subsequently reduce effort (less measurements)



Some Heuristics

- Dimensionality reduction will generally involve loss of information. One typically wants to reduce the number of features in such a way that this loss is minimized. Some heuristics are:
 - Features that are functions of other features should be discarded.
 - Features that are nearly constant (small-variance) should be discarded.
 - Features weakly correlated with output (i.e., “noisy features”) should be discarded.



Excursus on the Curse of Dimensionality

- In high dimensions learning gets difficult:
 - Data are sparse, it gets impossible to calculate local *mean* values
 - Many inputs require approximating functions with many parameters → risk of overfitting (unless regularization)

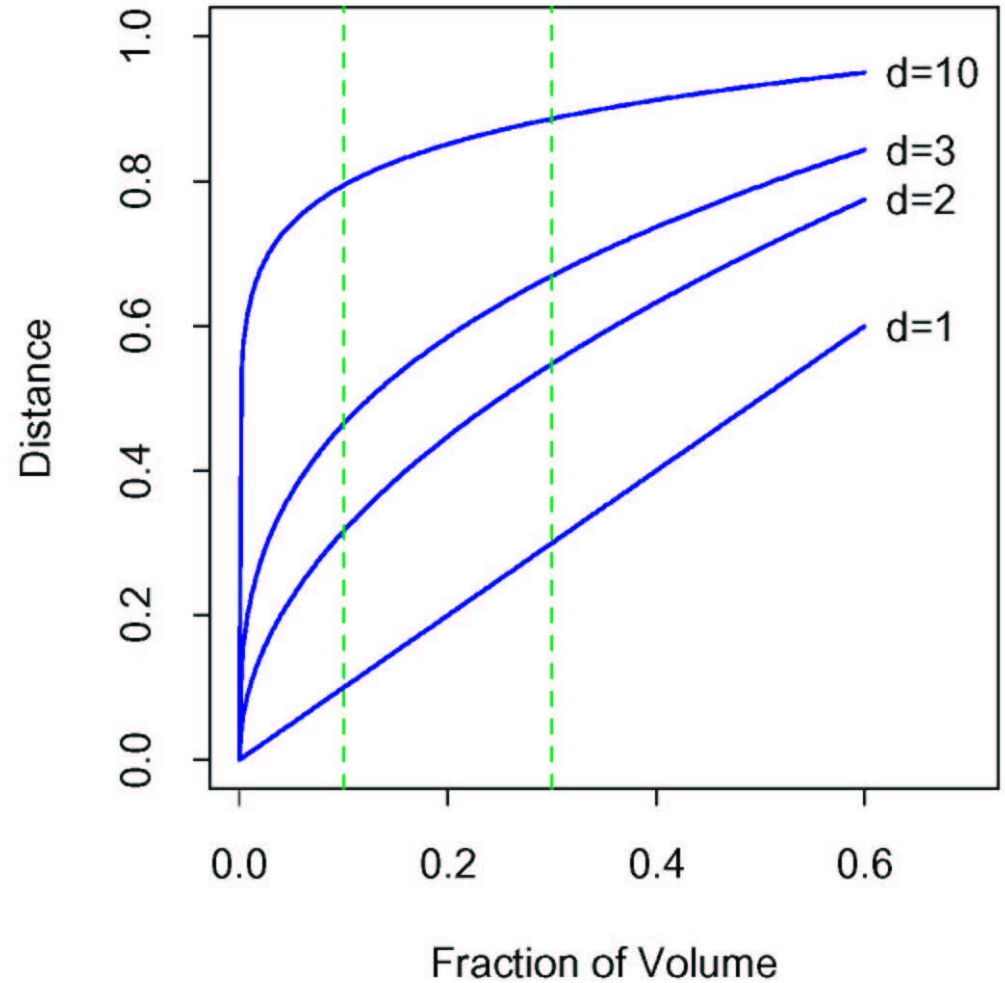
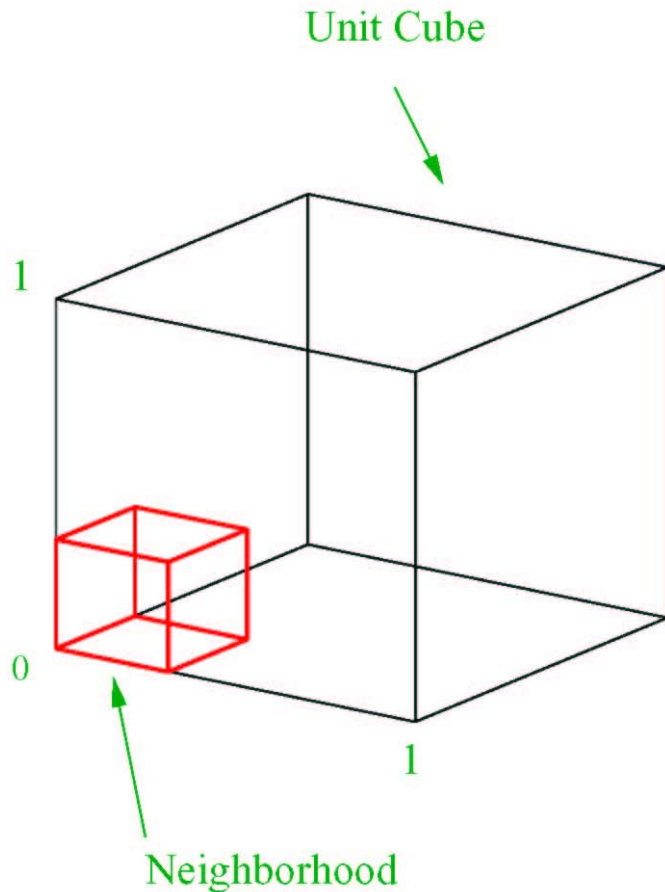


CD 1: loss of locality

- Consider points uniformly distributed in a p -dimensional unit hypercube.
- Consider hypercubical neighborhoods capturing a fraction r of observations, e.g. for the KNN algorithm.
- This corresponds to a fraction r of the unit volume \rightarrow edge is $e_p(r)=r^{1/p}$.
- Examples: 10D $\rightarrow e_{10}(0.01)=0.63$; $e_{10}(0.1)=0.8$
- In words: to capture 1% or 10% of the data (for a local average) we must cover 63% or 80% of the range of each variable.
- Such neighborhoods are hardly local

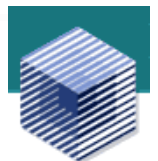


Loss of locality - illustration



CD 2: data sparsity

- Density proportional to $N^{1/p}$
(= #points/volume)
- **If 100 points are dense in 1D, 100^{10} points have same density in 10D**
- solar system will not last long enough for such a data collection → the measurement space is sparsely populated



CD 3: data are forced to the boundary

- N data uniformly distributed in a p dim unit ball
- We want nearest neighbor estimate at origin.
- Median distance from origin to closest point is:

$$d(p, N) = \left(1 - \frac{1}{2}^{1/N}\right)^{1/p}$$

- $D(10, 5000) = 0.52$
- This is more than half the way to boundary



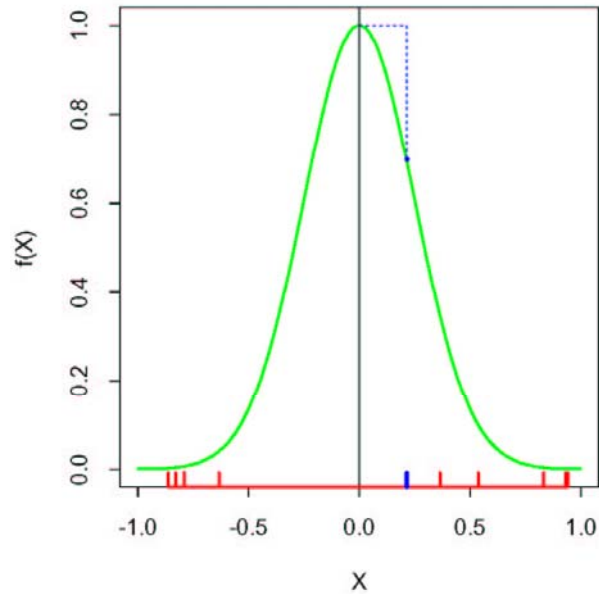
Consequence of CD: prediction error for 1NN

- Consider 1000 points generated uniformly over $[-1,1]^p$
- True data generator is $Y=f(X)=\exp(-8|X|^2)$ (no error)
- Use 1NN to predict y_0 at $x_0=0$

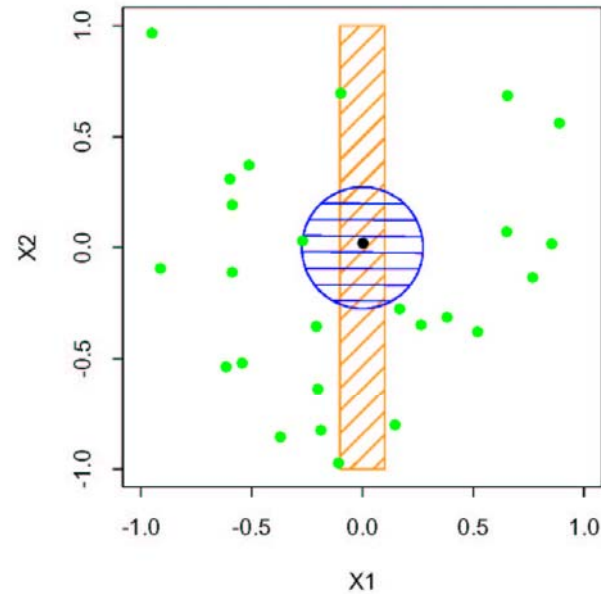
$$\begin{aligned}\text{MSE}(x_0) &= \mathbb{E}_{\mathcal{T}}[f(x_0) - \hat{y}_0]^2 \\ &= \mathbb{E}_{\mathcal{T}}[\hat{y}_0 - \mathbb{E}_{\mathcal{T}}(\hat{y}_0)]^2 + [\mathbb{E}_{\mathcal{T}}(\hat{y}_0) - f(x_0)]^2 \\ &= \text{Var}_{\mathcal{T}}(\hat{y}_0) + \text{Bias}^2(\hat{y}_0).\end{aligned}$$



1-NN in One Dimension



1-NN in One vs. Two Dimensions



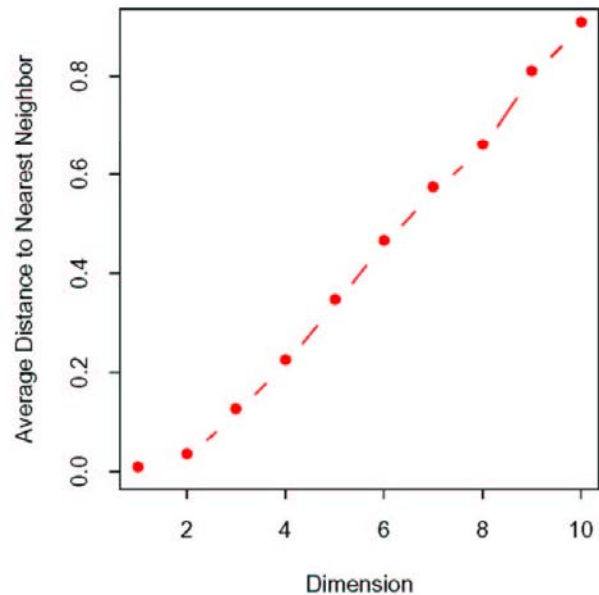
By $p=10$, 99% of the data are more than 0.5 away from the origin

$\rightarrow \hat{y}(0)=0$

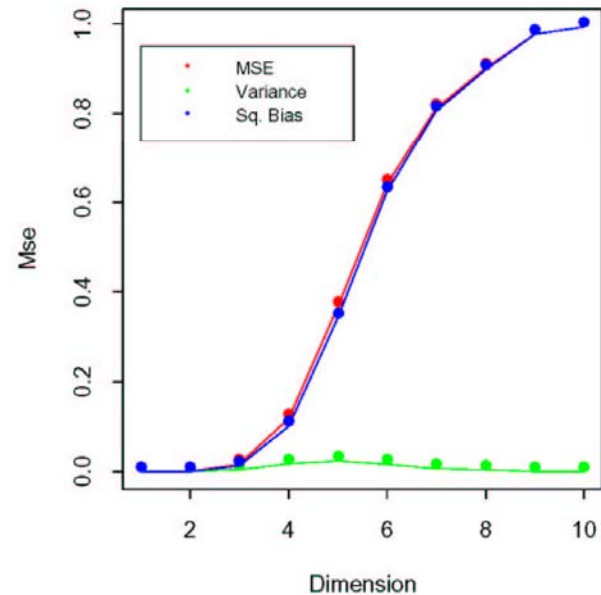
$\rightarrow \text{bias} = 1$

$\rightarrow \text{variance}=0$

Distance to 1-NN vs. Dimension



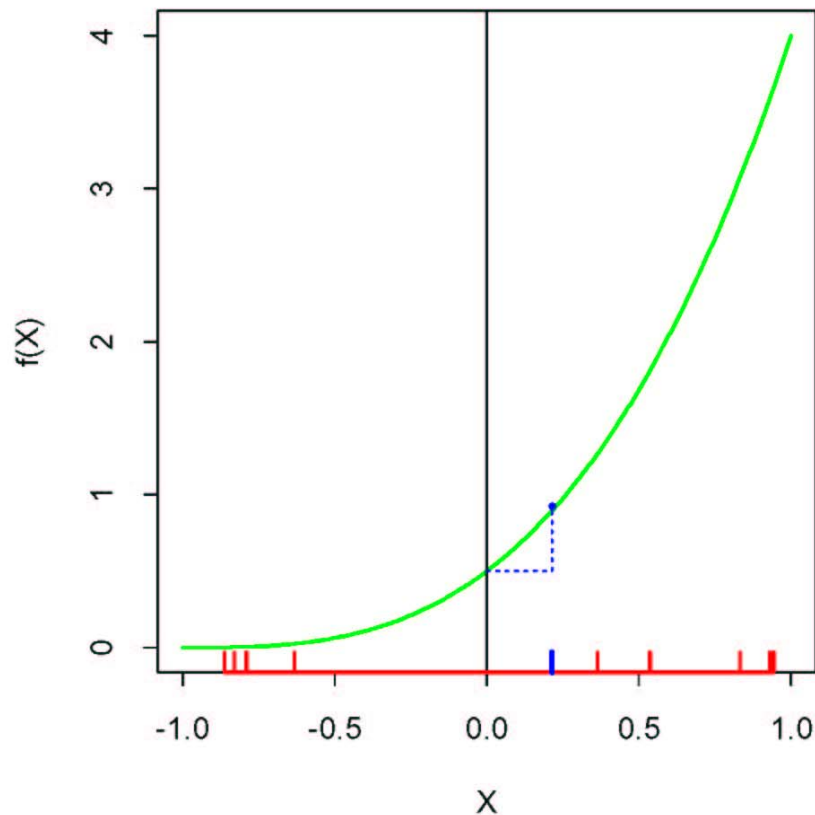
MSE vs. Dimension



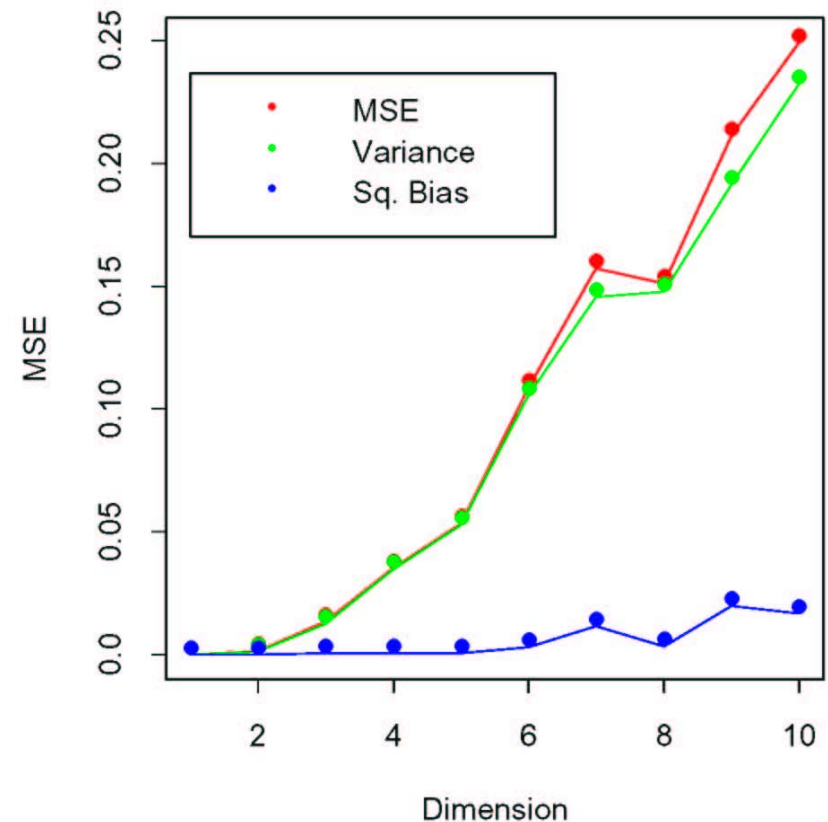
Example 2: variance dominates

$$Y=f(X)=1/2 (X_1+1)^3$$

1-NN in One Dimension



MSE vs. Dimension



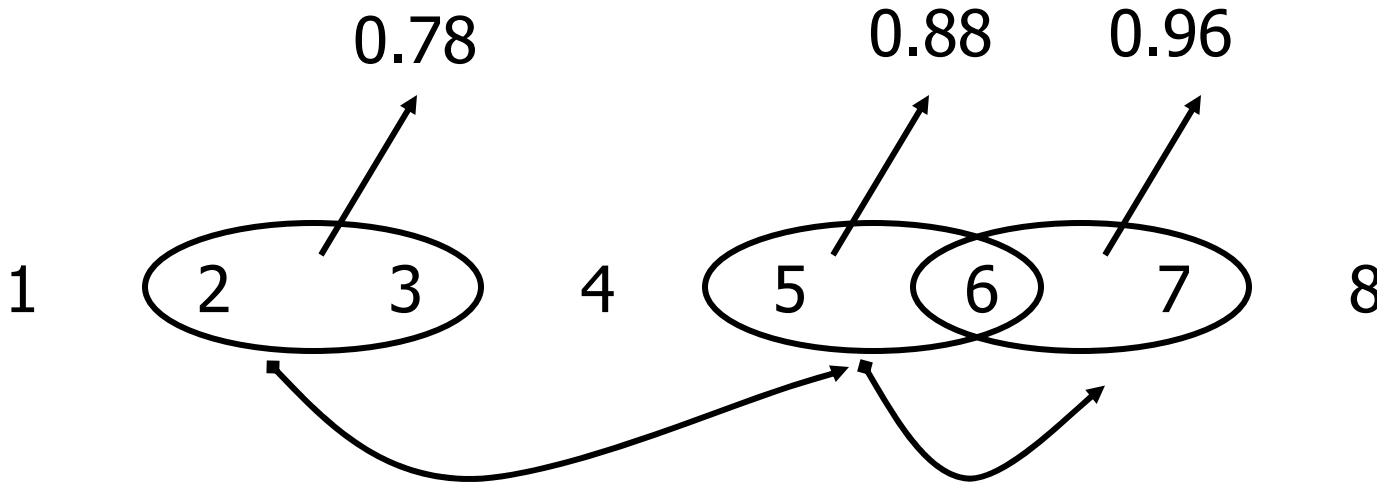
Feature extraction: Principal Component Analysis

- Probably the most well known data analysis technique
- Part of exploratory data analysis
- Linear transformation
- Unsupervised, i.e. does not use class labels
- In words:
 - 1st PC maximizes projected variance,
 - 2nd PC is orthogonal to 1st and maximizes residual variance...
- Depends on data standardization



Feature selection: two ingredients

- Feature **subset** evaluation criterion: e.g. test classification error
- Search algorithm (inside subset space)



Taxonomy of criterion function

- With regard to the final classifier (used to assess error):
 - **Wrappers:** the criterion is the same
 - Classification rate for your classifier (e.g. multilayer perceptron)
 - Results are best since the feature are selected using the same criterion used to finally validate them
 - A different classifier may give a different feature set
 - **Filters:** the criterion is different
 - Typically based on single feature tests or scatter matrices
 - Cheaper to implement, yet assumption in criterion can lead to poor test results
 - **Embedded methods:** FS is part of classifier optimization (e.g. NSC)
- Practically: trade off between time and final classification rate



Criterion function: scatter matrices

$$J_1 = \text{Tr} \left\{ S_W^{-1} S_B \right\}, \quad \text{where} \quad :$$

S_W pooled within – class scatter matrix

$$S_W = \sum_{i=1}^C \frac{n_i}{n} \hat{\Sigma}_i$$

S_B between – class scatter matrix

$$S_B = \sum_{i=1}^C \frac{n_i}{n} (\mathbf{m}_i - \mathbf{m})(\mathbf{m}_i - \mathbf{m})^T$$

$$\text{Else} \quad : \quad J_2 = \frac{\text{Tr} \left\{ S_B \right\}}{\text{Tr} \left\{ S_W \right\}}, \quad J_3 = \frac{|\hat{\Sigma}|}{|S_W|}$$



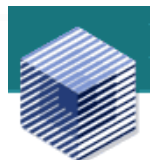
Search strategy

- Optimal methods:
 - exhaustive
 - branch and bound (for monotonic criteria)
- Suboptimal methods: all the rest



Exhaustive search

- X_d set of all possible subsets of size d out of p indexes
- Cardinality of X_d is
$$n_d = \frac{p!}{(p-d)!d!}$$
- Electronic Nose: 5 sensors sets out of 20:
 $n_d = 3 \times 10^4$
→ **may** use exhaustive methods
- Microarray: 10 genes out of 8000: $n_d = 3 \times 10^{32}$
→ **must** use suboptimal methods



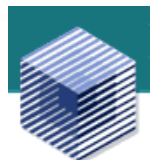
Sub-Optimal Search

- A sub-optimal solution in many cases cannot be avoided due to the sheer size of the problem. There are a number of fast feature selection algorithms that execute sub-optimal searches.
 - Best Individual d Features
 - Sequential Forward Search
 - Sequential Backward Search
 - Generalized Sequential Forward Search
 - Generalized Sequential Backward Search
 - Plus-I Take-r Search
 - Generalized Plus-I Take-r Sequential Search
 - Floating Search



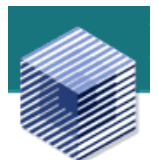
Best Individual d Features

- *Choose best single features*
- Intuitive idea, but ignores multivariate relationships.
- Counter-example: the best d features are equal:
 $X'_1 = \dots = X'_d$
- In practice, if the best d features are highly correlated with each other the result is similar.
- Nevertheless a very common method, at least as *filter*.
- Usually performed by using the correlation of individual features with Y or a hypothesis test as the criterion for the univariate search.



Sequential selection

- *Sequential forward selection (SFS)*: add one feature at a time.
 - Let $X_{(0)} = \emptyset$.
 - Given the current feature set $X_{(k)}$, the criterion $J(X_{(k)} \cup X_i, Y)$ is evaluated for each $X_i \notin X_{(k)}$ and the X_i^* that maximizes this is added to the feature set: $X_{(k+1)} = X_{(k)} \cup X_i^*$.
 - Stop if $k = d$ or if no improvement is possible.
- Defect: once added a feature cannot be deleted (nesting).
- *Analogous: Sequential backward selection (SBS)*: computationally more demanding



Other suboptimal methods

- *Generalized SFS (SBS)*: add r features at a time.
 - Takes into account statistical relationship between variables to add. What r ?
- *Plus / take away r selection*: l steps of SFS then r steps of SBS
 - Allows backtracking
 - Removes the problem of nesting: set of features at step n is not necessarily subset of set at step $n-1$.
- *Generalized l - r selection*: use GSFS(l) then GSBS(r)
- *Floating search methods*: l and r are allowed to float, i.e. change at each step
 - In comparative studies gave best results



CFS: classical (forward) floating search

CFS (specifically SFFS) first:

1. Add the *most significant feature* to the current subset of size k . Let $k = k + 1$.
2. Conditionally remove the *least significant feature* from the current subset.
3. If the current subset is the best subset of size $k - 1$ found so far, let $k = k - 1$ and go to step 2. Else return the conditionally removed feature and go to step 1.

- Difference with Plus I take away r selection: features are only *conditionally* removed
- *Adaptive floating search:*
 - Do *generalized* search (and not sequential)
 - Determine size of step *adaptively* (from 1 to a maximum r which is redefined at each step – maximal near the target dimension)

Unbiased Error Estimation



Statistical Applications in Genetics and Molecular Biology

Volume 3, Issue 1

2004

Article 37

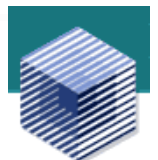
A Compendium to Ensure Computational Reproducibility in High-Dimensional Classification Tasks

Markus Ruschhaupt*

Wolfgang Huber†

Annemarie Poustka‡

Ulrich Mansmann**



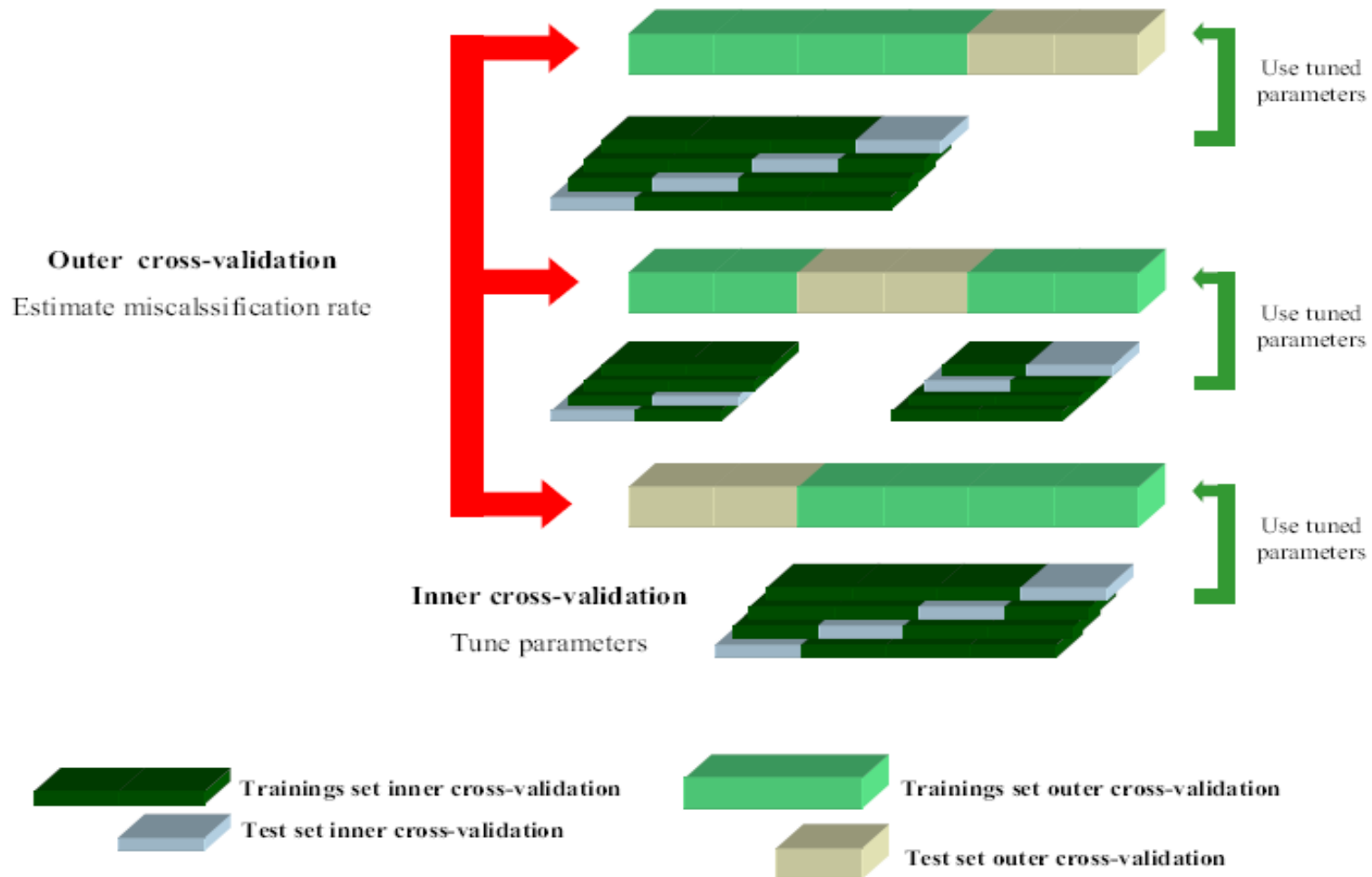
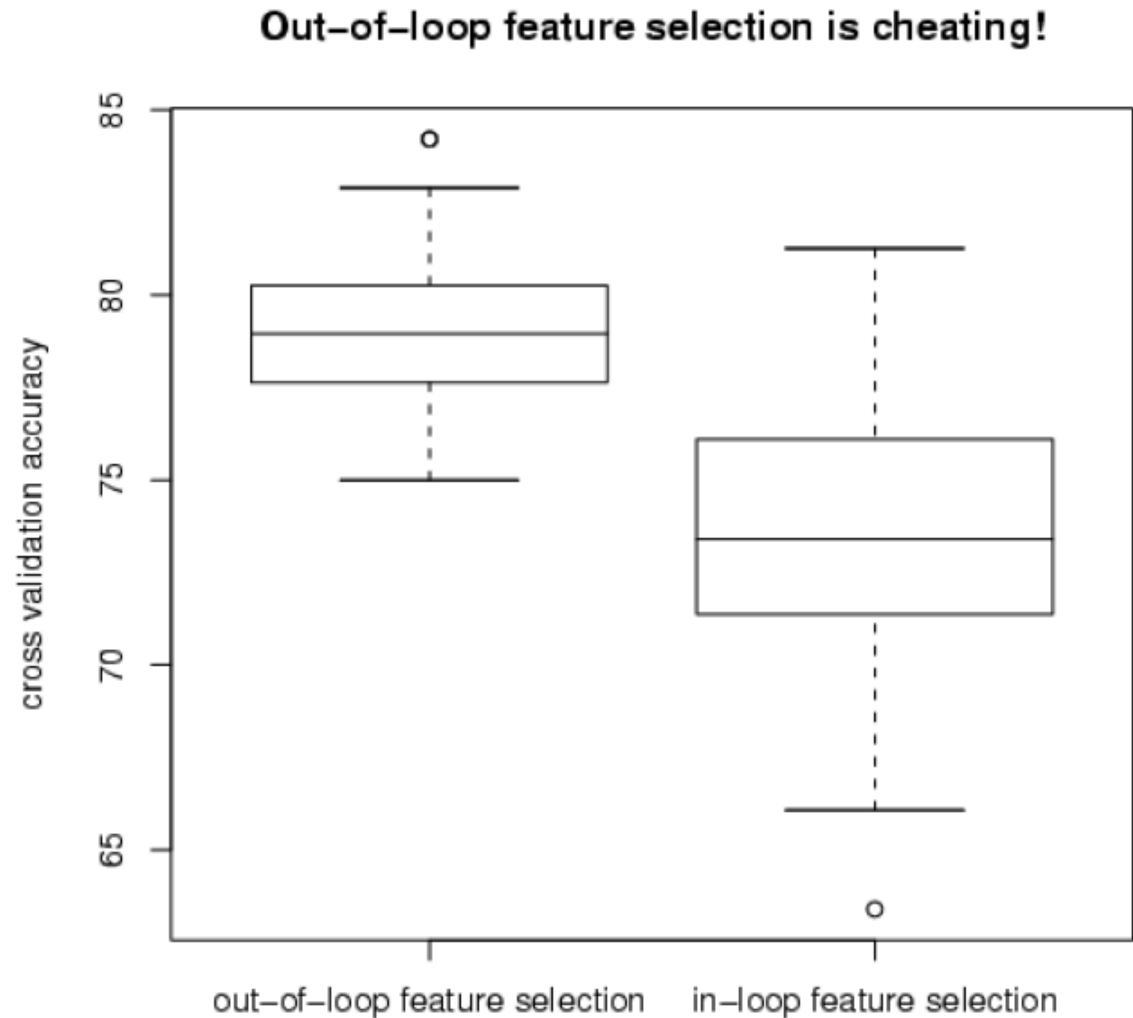


Figure 1: The cross-validation strategy implemented in MCRestimate



In-loop vs out-of-loop FS evaluation

- Nested 10-fold-CV
- Variance from 100 random partitions



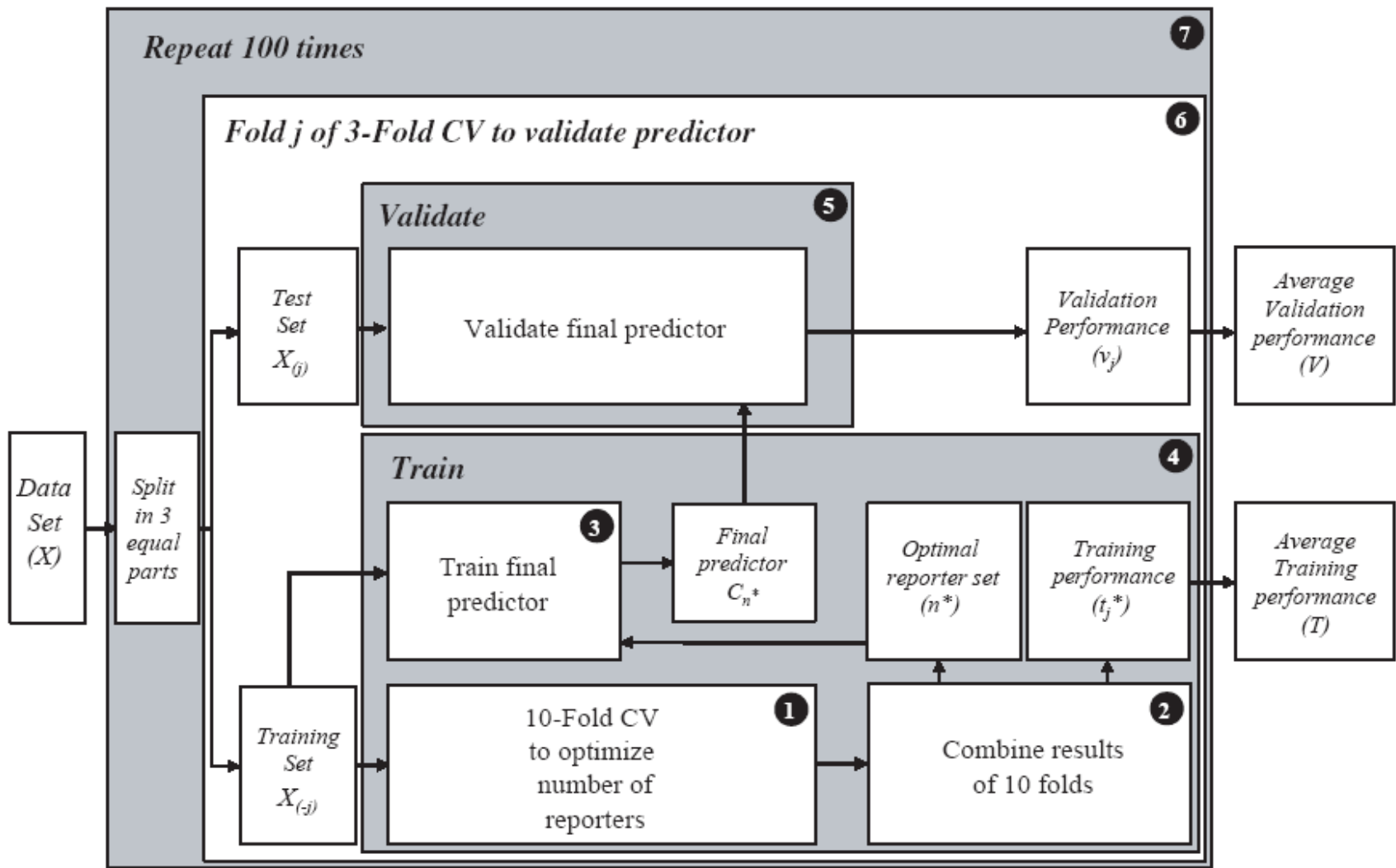
Gene expression

A protocol for building and evaluating predictors of disease state based on microarray data

Lodewyk F. A. Wessels^{1,2,*}, Marcel J. T. Reinders¹, Augustinus A. M. Hart³,
Cor J. Veenman¹, Hongyue Dai⁴, Yudong D. He⁴ and Laura J. van't Veer²

¹Department of Mediamatics, Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, Mekelweg 4, 2628 CD Delft, The Netherlands, ²Department of Pathology and ³Department of Radiotherapy, The Netherlands Cancer Institute, Plesmanlaan 121, 1066 CX Amsterdam, The Netherlands and ⁴Rosetta Inpharmatics LLC (a wholly owned subsidiary of Merck & Co., Inc.), 401 Terry Avenue N. Seattle, Washington 98109, USA





Results

- Simple selection strategies (forward filtering and shrunken centroids) outperform partial least squares in four of the six datasets.
- Similarly, simple predictors, such as the nearest mean classifier, outperform more complex classifiers.

