INTERFACE WORKSHOP-APRIL 2004

RFtools-- for Predicting and Understanding Data

Leo Breiman
Statistics, UCB
leo@stat.berkeley.edu

Adele Cutler
Mathematics, USU
adele@math.usu

1. Overview of Features--Leo Breiman
2. Graphics and Case Sturies--Adele Cutler
3. Nuts and Bolts--Adele Cutler

The Data Avalanche

The ability to gather and store data has resulted in an avalanche of scientific data over the last 25 years. Who is trying to analyze this data and extract answers from it?

There are small groups of academic statisticians, machine learning specialists in academia or at high end places like IBM, Microsoft, NEC, ETC.

More numerous are the workers in many diverse projects trying to extract significant information from data.

Question (US Forestry Service). "We have lots of satellite data over our forests. We want to use this data to figure out what is going on"

Question (LA County) "We have many years of information about incoming prisoners and whether they turned violent. We want to use this data to screen incoming prisoners for potential violent behavior."

Tools Needed

Where are the tools coming from?

SAS $$$

S+ $$$

SPSS $$$

R 000 (free open source)

Other scattered packages

The most prevalent of these tools are two generations old--

General and non-parametric

CARTlike (binary decision trees)

Clustering

Neural Nets

What Kind Of Tools Are Needed to Analyze Data ?

An example--CART

The most successful tool (with lookalikes) of the last 20 years. Why?

1) Universally applicable to classification and regression problems with no assumptions on the data structure.

2) The picture of the tree structure gives valuable insights into which variables were important and where.

3) Terminal nodes gave a natural clustering of the data into homogenous groups.

4) Handles missing data and categorical variables efficiently.

4) Can handle large data sets--computational requirements are order of MNlogN where N is number of cases and M is number of variables

Drawbacks

accuracy: current methods such as SVMs and ensembles average 30% lower error rates than CART.

instability: change the data a little and you get a different tree picture. So the interpretation of what goes on is built on shifting sands.

In 2003 we can do better

What would we want in a tool to be a useful to the sciences.

Tool Specifications For Science

The Minimum

1) Universally applicable in classification and regression.

2) Unexcelled accuracy

3) Capable of handling large data sets

4) Handles missing values effectively

Much More

think of CART tree picture.

5) which variables are important?

6) how do variables interact?

7) what is the shape of the data--how does it cluster?

8) how does the multivariate action of the variables separate classes?

9) find novel cases and outliers

Toolmakers

Adele Cutler & Leo Breiman

free open source written in f77

www.stat.berkeley.edu/users/breiman/RFtools

The generic names of our tools is random forests (RF).

Characteristics as a classification machine:

1) Unexcelled accuracy-about the same as SVMs

2) Scales up to large data sets.

Unusually Rich

In the wealth of scientifically important insights it gives into the data It is a general purpose tool, not designed for any specific application

Outline of Part One (Leo Breiman)

I *The Basic Paradigm*

a. error, bias and variance

b. randomizing "weak" predictors

c. two dimensional illustrations

d. unbiasedness in higher dimensions

II. *Definition of Random Forests*

a) the randomization used

b) properties as a classification machine

c) two valuable by-products
oob data and proximities

III *Using Oob Data and Proximities*

a) using oob data to estimate error

b) using oob data to find important variables

c) using proximities to compute prototypes

d) using proximities to get 2-d data pictures

e) using proximities to replace missing values

f) using proximiites to find outliers

g) using proximities to find mislabeled data

IV *Other Capabilities*

a) balancing error

b) unsupervised learning

I. The Fundamental Paradigm

Given a training set of data

$$\mathbf{T}= (\mathbf{y}_n, \mathbf{x}_n)\, n = 1, \ldots, N\}$$

where the $\mathbf{y}_n$ are the response vectors and the $\mathbf{x}_n$ are vectors of predictor variables:

Problem: Find a function f on the space of prediction vectors with values in the response space such that the prediction error is small.

If the $(\mathbf{y}_n, \mathbf{x}_n)$ are i.i.d from the distribution **(Y,X)** and given a function L(**y,y'**) that measures the loss between **y** and the prediction **y'**: the prediction error is

$$PE(f, \mathbf{T}) = E_{\mathbf{Y},\mathbf{X}} L(\mathbf{Y}, f(\mathbf{X}, \mathbf{T}))$$

Usually **y** is one dimensional.

If numerical, the problem is regression.
the loss is squared error.

If unordered labels, it is classification.

Bias and Variance in Regression

For a specific predictor the bias measures its "systematic error".

The variance measures how much it "bounces around"

the Bias-Variance Decomposition

A random variable Y related to a random vector $\mathbf{x}$ can be expressed as

(1) $$Y = f^*(\mathbf{X}) + \varepsilon$$

where

$$f^*(\mathbf{X}) = E(Y|\mathbf{X}), \quad E(\varepsilon|\mathbf{X}) = 0$$

This decomposes Y into its structural part $f^*(\mathbf{x})$ which can be predicted in terms of $\mathbf{x}$, and the unpredictable noise component.

Mean-squared generalization error

of a predictor $f(\mathbf{x},T)$ is

(2) $$PE(f(\bullet,\mathbf{T}))=E_{Y,\mathbf{X}}(Y-f(\mathbf{X},\mathbf{T}))^2$$

where the subscripts indicate expectation with respect to $Y,\mathbf{X}$ holding **T f**ixed.

Take the expectation of (2) over all training sets of the same size drawn from the same distribution .

This is the mean-squared generalization error $PE^*(f)$.

Let $\bar{f}(\mathbf{x})$ be the average over training sets of the predicted value at **x**. That is;

(3) $$\bar{f}(\mathbf{x})=E_{\mathbf{T}}(f(\mathbf{x},\mathbf{T})).$$

The bias-variance decomposition

$$(4)\quad \begin{aligned} PE^*(f) &= E\varepsilon^2 + E_{\mathbf{X}}(f^*(\mathbf{X}) - \bar{f}(\mathbf{X}))^2 \\ &+ E_{\mathbf{X},T}(f(\mathbf{X},\mathbf{T}) - \bar{f}(\mathbf{X}))^2 \end{aligned}$$

the first term is the *noise variance*,

the second is the *bias squared*

the third is the *variance.*

Weak Learners

Definition: *a weak learner is a prediction function that has low bias.*

Generally, low bias comes at the cost of high variance.

A weak learner is usually not an accurate predictor because of high variance.

two dimensional example

The 100 case training set is generated by taking:

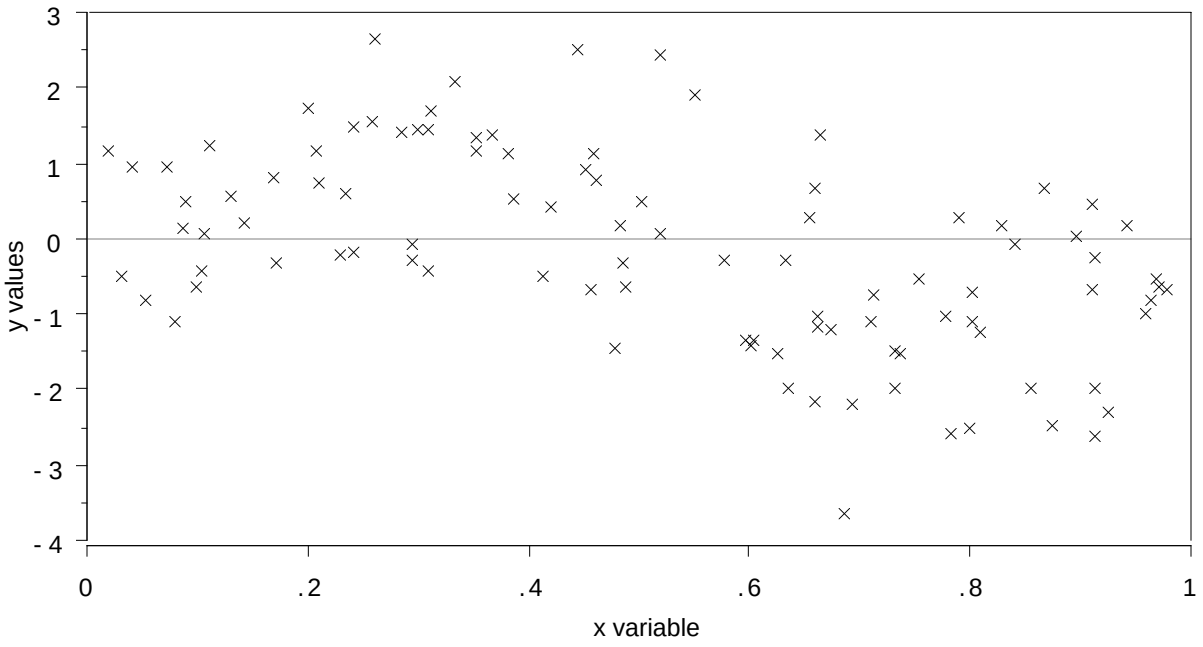
x(n) =100 uniformly spaced points on [0,1]
y(n)=sin(2*pi*x(n))+N(0,1)

The weak learner $f(\mathbf{x},\mathbf{T})$ is defined by:

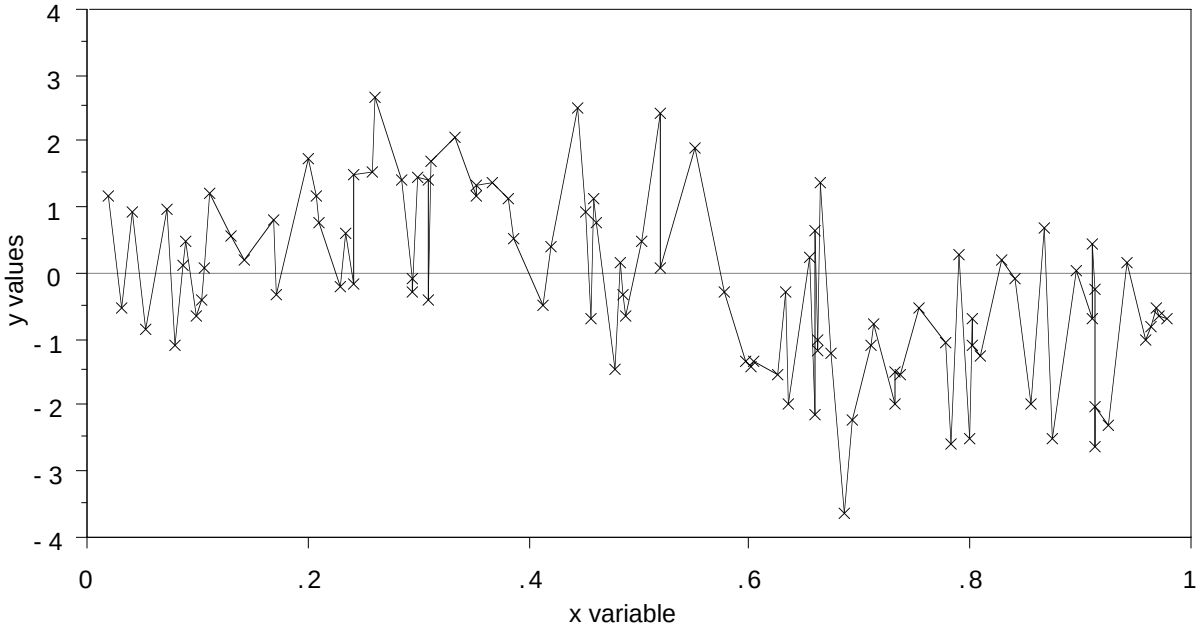
If $x(n) \le x < x(n+1)$ then $y(n), y(n+1)$ is linearly interpolated between $y(n), y(n+1)$

i.e. the weak learner is join the data dots by straight line segments.

FIRST TRAINING SET

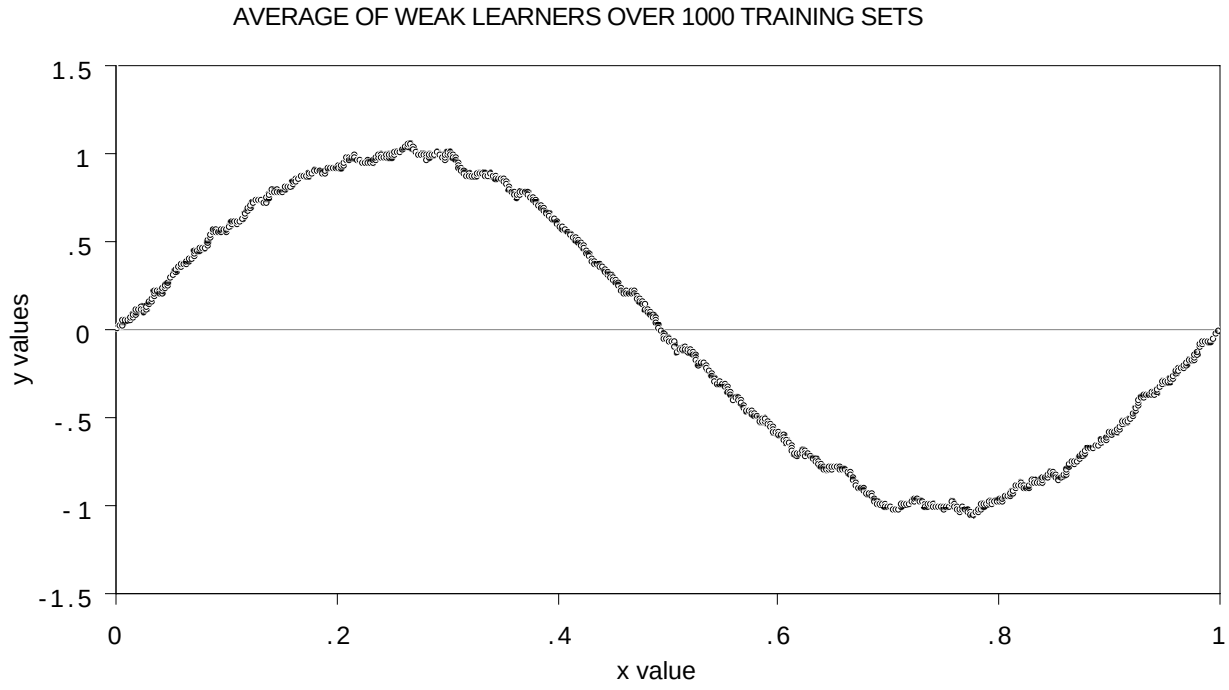


WEAK LEARNER FOR FIRST TRAINING SET



Bias

1000 training sets are generated in the same way and the 1000 weak learners averaged.



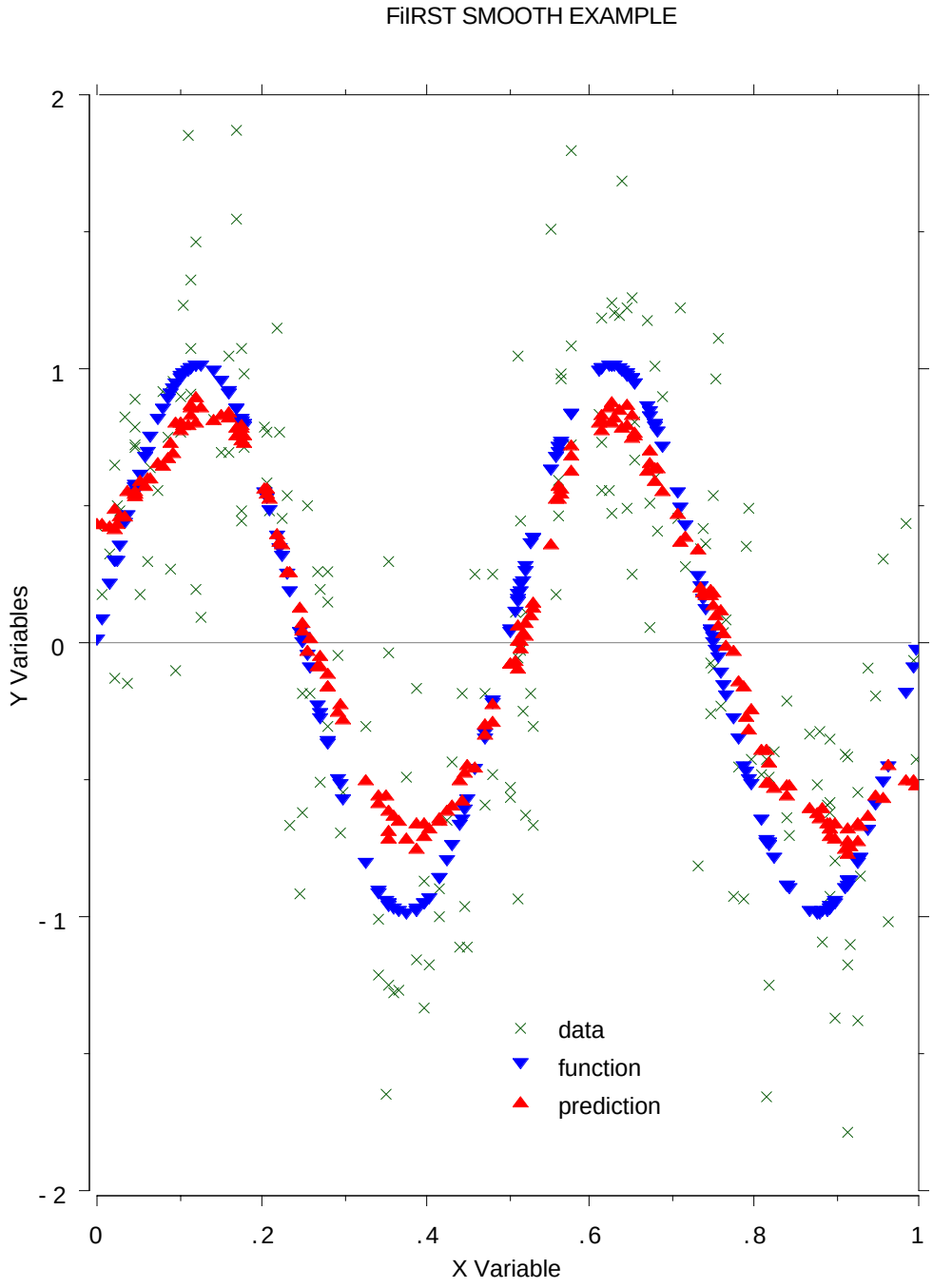
The averages approximate the underlying function sin(2*pi*x).

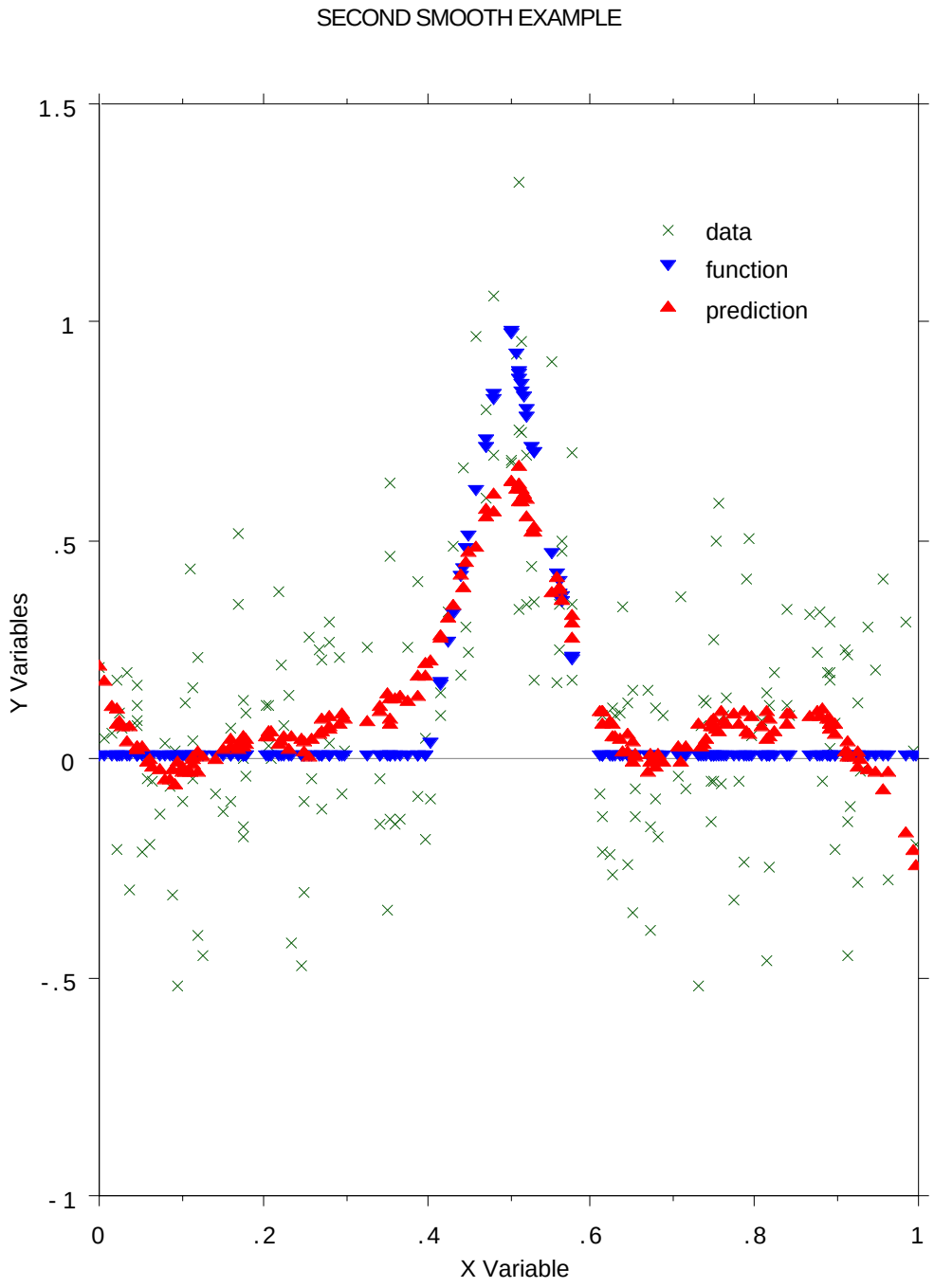
The weak learner is almost unbiased but with large variance. But 1000 replicate training sets are rarely available.

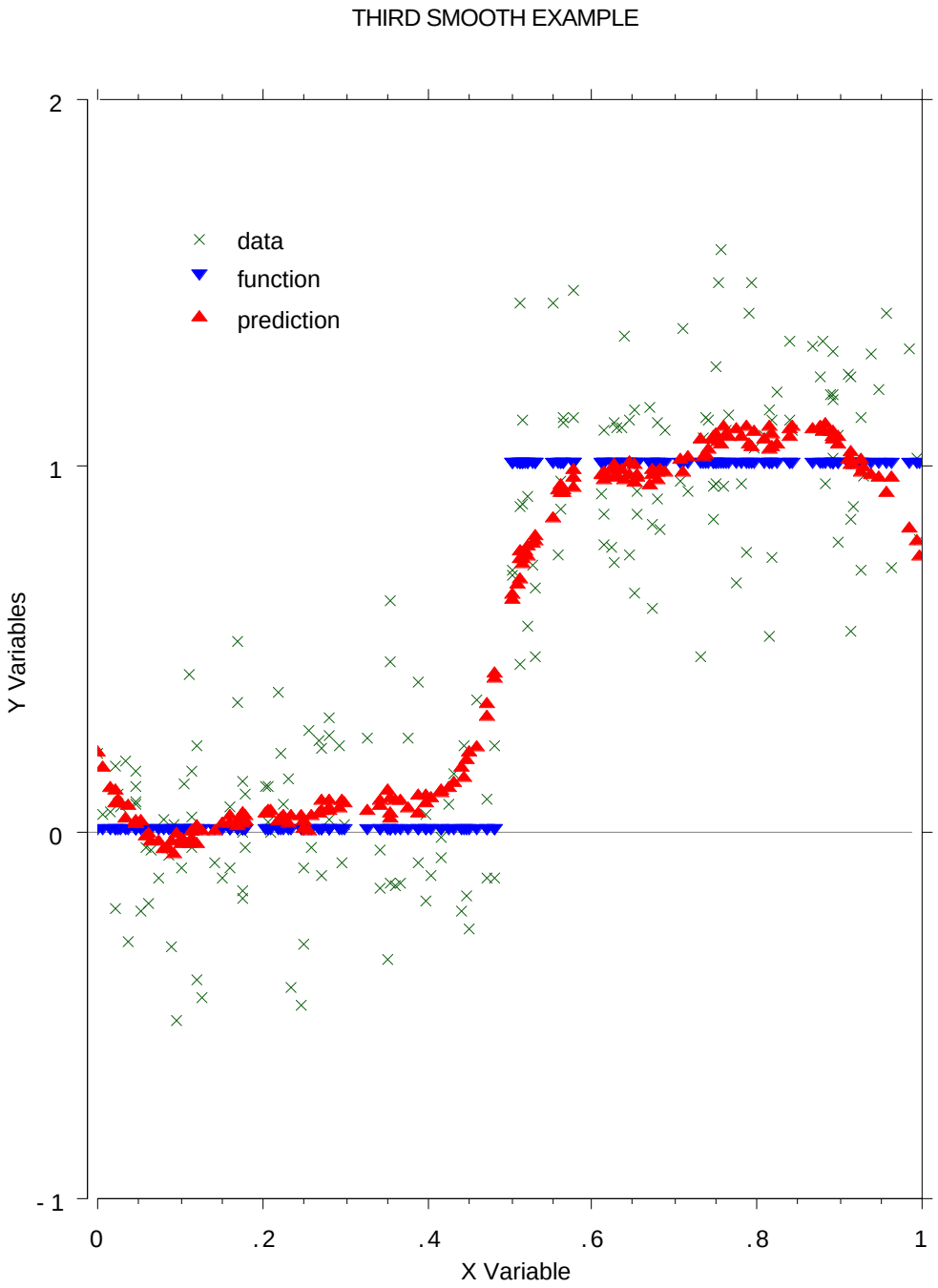
Making Silk Purses out of Weak Learners

Here are some examples of our fundamental paradigm applied to a single training set

using the same "connect-the -dots" weak learner.

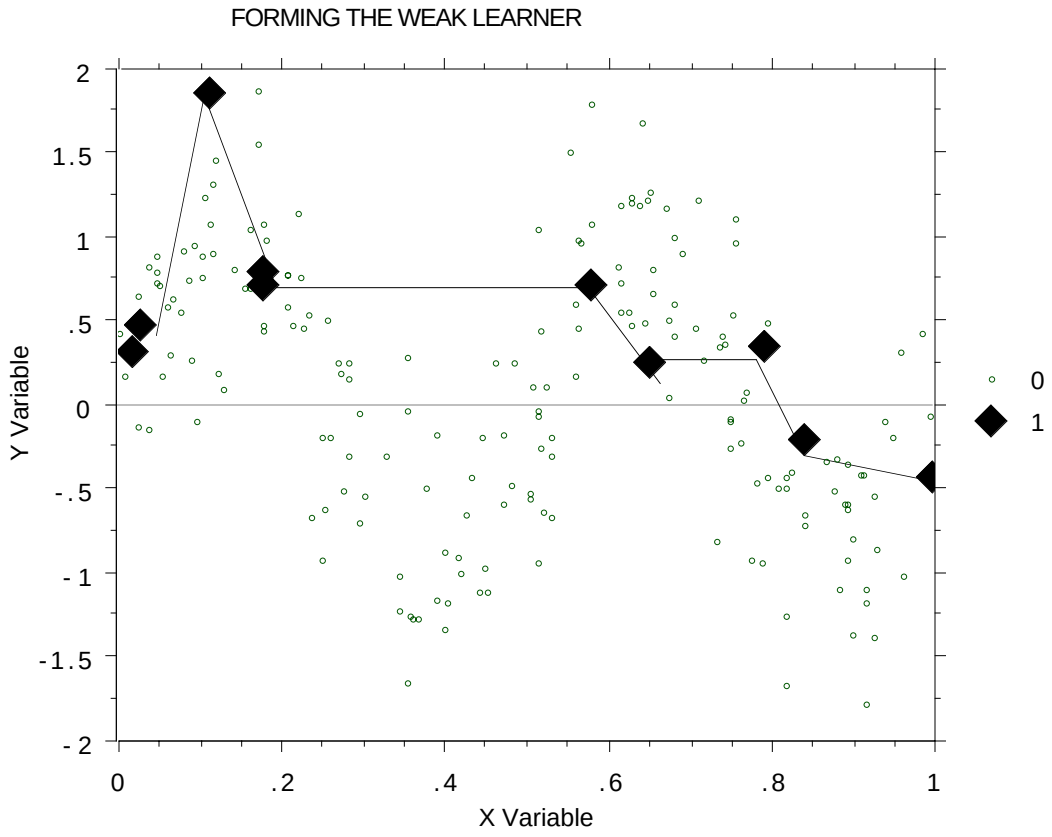

FiIRST SMOOTH EXAMPLE
2
1
0
- 1
- 2
Y Variables
0
.2
.4
.6
.8
1
X Variable
data
function
prediction


SECOND SMOOTH EXAMPLE
1.5
1
.5
0
-.5
-1
Y Variables
0
.2
.4
.6
.8
1
X Variable
data
function
prediction


THIRD SMOOTH EXAMPLE
data
function
prediction
Y Variables
X Variable
2
1
0
- 1
0
.2
.4
.6
.8
1

The Paradigm--
IID Randomization of Weak Learners

The predictions shown above are the averages of 1000 weak learners. Here is how the weak learners are formed:



Subsets of the training set consisting of two-thirds of the cases is selected at random. All the (y,x) points in the subset are connected by lines.

Repeat 1000 times and average the 1000 weak learners prediction.

The Paradigm-Continued

The kth weak learner is of the form:

$$f_k(\mathbf{x},\mathbf{T}) = f(\mathbf{x},\mathbf{T},\Theta_k)$$

where Θ_k is the random vector that selects the points to be in the weak learner.

The Θ_k are i.i.d.

If there are N cases in the training set, each Θ_k selects, at random, $2N/3$ integers from among the integers 1,...,N.

The values of y(n),x(n) for the selected n are deleted from the training set.

The ensemble predictor is:

$$F(\mathbf{x},\mathbf{T}) = \frac{1}{K}\Sigma_k f(\mathbf{x},\mathbf{T},\Theta_k)$$

Algebra and the LLN leads to:

$$Var(F) =$$
$$E_{\mathbf{X},\Theta,\Theta'}[\rho_{\mathbf{T}}(f(\mathbf{x},\mathbf{T},\Theta)f(\mathbf{x},\mathbf{T},\Theta'))Var_{\mathbf{T}}(f(\mathbf{x},\mathbf{T},\Theta)$$

where Θ,Θ' are independent. Applying the mean value theorem--

$$Var(F) = \bar{\rho} Var(f)$$

and

$$\begin{aligned} Bias^2(F) &= E_{Y,\mathbf{X}}(Y - E_{\mathbf{T},\Theta} f(\mathbf{x},\mathbf{T},\Theta))^2 \\ &\leq E_{Y,\mathbf{X},\Theta}(Y - E_{\mathbf{T}} f(\mathbf{x},\mathbf{T},\Theta))^2 \\ &= E_{\Theta} bias^2 f(\mathbf{x},\mathbf{T},\Theta) + E\varepsilon^2 \end{aligned}$$

Using the iid randomization of predictors leaves the bias approximately unchanged while reducing variance by a factor of $\bar{\rho}$

The Message

A big win is possible with using iid randomization of weak learners as long as their correlation and bias are low.

In sin curve example, base predictor is connect all points in order of x(n).

$bias^2$=.000
variance=.166

For the ensemble

$bias^2$ = .042
variance =.0001

Random forests is an example of iid randomization applied to binary classifiction trees (CART-like)

What is Random Forests

A random forest (RF) is a collection of tree predictors

$$f(\mathbf{x}, \mathbf{T}, \Theta_k), k = 1, 2, \ldots, K)$$

where the Θ_k *are i.i.d random vectors.*

In classification, the forest prediction is the unweighted plurality of class votes

The Law of Large Numbers insures convergence as $k \to \infty$

The test set error rates (modulo a little noise) are monotonically decreasing and converge to a limit.

That is: *there is no overfitting as the number of trees increases*

Bias and Correlation

The key to accuracy is low correlation and bias.

To keep bias low, trees are grown to maximum depth.

To keep correlation low, the current version uses this randomization.

i) Each tree is grown on a bootstrap sample of the training set.

ii) A number **m** is specified much smaller than the total number of variables M.

iii) At each node, **m** variables are selected at random out of the M.

iv) The split used is the best split on these **m** variables

The only adjustable parameter in RF is **m**. User setting of **m** will be discussed later.

Properties as a classification machine.

a) excellent accuracy

in tests on collections of data sets, has better accuracy than Adaboost and Support Vector Machines

b) is fast

with 100 variables, 100 trees in a forest can be grown in the same computing time as 3 single CART trees

c) handles

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

d) gives

internal unbiased estimate of test set error as trees are added to ensemble

e) cannot overfit

(already discussed)

Two Key Byproducts

The out-of-bag test set

For every tree grown, about one-third of the cases are out-of-bag (out of the bootstrap sample). Abbreviated *oob*.

The oob samples can serve as a test set for the tree grown on the non-oob data.

This is used to:

i) Form unbiased estimates of the forest test set error as the trees are added.

ii) Form estimates of variable importance.

The Oob Error Estimate

oob is short for out-of-bag meaning not in the bootstrap training sample.

the bootstrap training sample leaves out about a third of the cases.

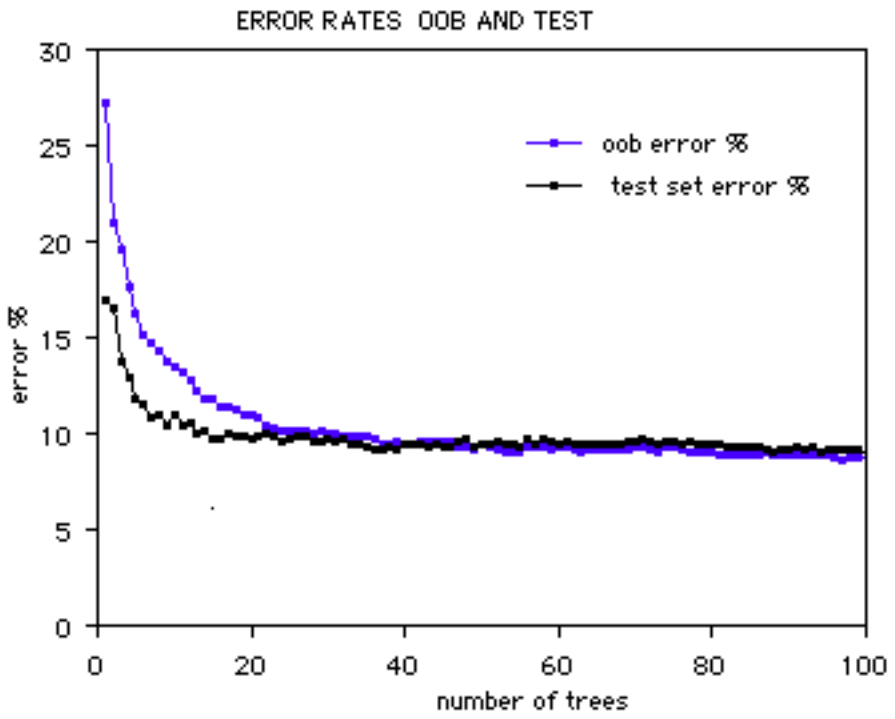
each time a case is oob, it is put down the corresponding tree and given a predicted classification.

for each case. as the forest is grown, the plurality of these predictions give a forest class prediction for that case.

this is compared to the true class, to give the oob error rate.

Illustration-satellite data

This data set has 4435 cases, 35 variables and a test set of 2000 cases. If the output to the monitor is graphed for a run of 100 trees, this is how it appears:



The oob error rate is larger at the beginning because each case is oob in only about a third of the trees.

The oob error ate is used to select **m** (called mtry in the code) by starting with **m** = $\sqrt{M}$, running about 25 trees, recording the oob error rate. Then increasing and decreasing **m** until the minimum oob error is found.

Using Oob for Variable Importance

to assess the importance of the mth variable, after growing the kth tree randomly permute the values of the mth variable among all oob cases.

put the oob cases down the tree.

compute the decrease in the number of votes for the correct class due to permuting the mth variable.

average this over the forest.

also compute the standard deviation of the decreases and the standard error.

dividing the average by the se gives a z-score.

assuming normality, convert to a significance value.

the importance of all variables is assessed in a single run

Illustration-breast cancer data

699 cases, 9 variables, two classes.
initial error rate is 3.3%.

added 10,000 independent unit normal variables to each case.

did a run to generate a list 10,009 long of variable importances and ordered them by z-score

here are the first 12 entries

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

2003 NIPS competition on feature selection in data sets with thousands of variables

over 1600 entries from some of the most prominent people in Machine Learning.

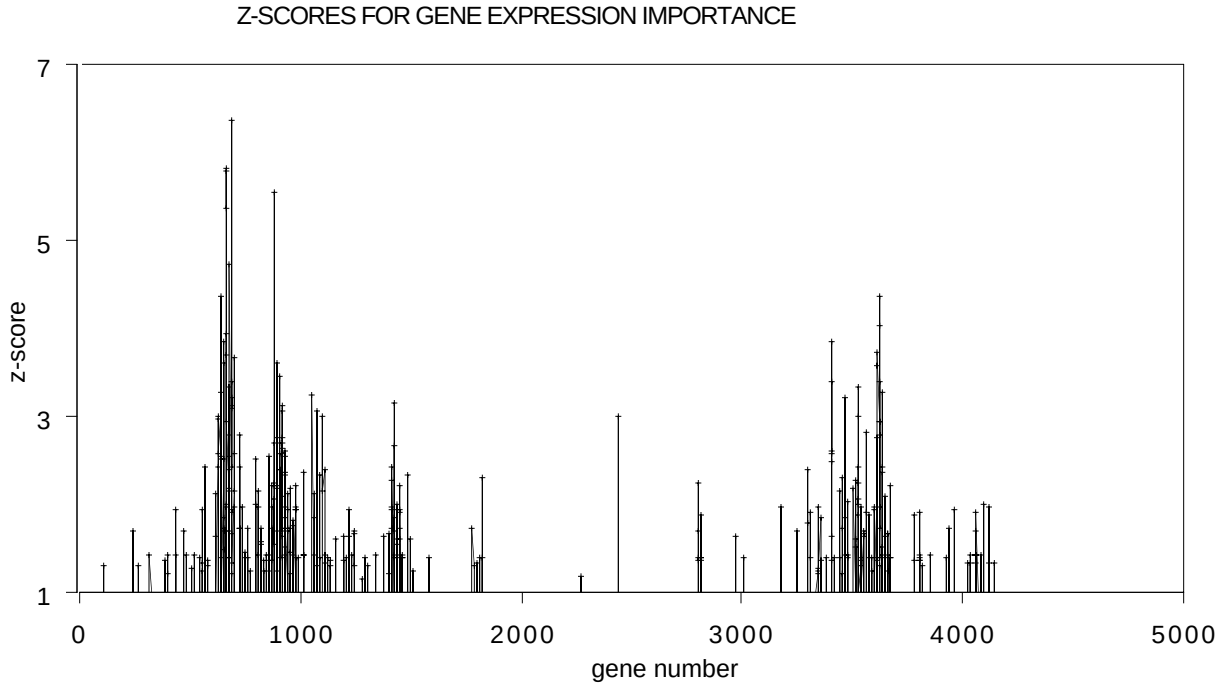
the top 2nd and 3rd entry used RF for feature selection.

Illustration-Microarray Data

81 cases, three classes, 4682 variables

This data set was run without variable deletion The error rate is 1.25% --one case misclassified.

Importance of all variables is computed in a single run of 1000 trees.



The Proximities

Since the trees are grown to maximum depth, the terminal nodes are small.

For each tree grown, pour all the data down the tree.

If two data points $\mathbf{x}_n$ and $\mathbf{x}_k$ occupy the same terminal node,

increase $prox(\mathbf{x}_n,\mathbf{x}_k)$ **by one**.

At the end of forest growing, these proximities are normalized by division by the number of trees.

They form an intrinsic similarity measure between pairs of data vectors.

These are used to:

i) Replace missing values.

ii) Give informative data views via metric scaling.

iii) Understanding how variables separate classes--prototypes

iv) Locate outliers and novel cases

Replacing Missing Values using Proximities

RF has two ways of replacing missing values.

The Cheap Way

Replace every missing value in the mth coordinate by the median of the non-missing values of that coordinate or by the most frequent value if it is categorical.

The Expensive Way

This is an iterative process. If the mth coordinate in instance $\mathbf{x}_n$ is missing then it is estimated by a weighted average over the instances $\mathbf{x}_k$ with non-missing mth coordinate where the weight is $prox(\mathbf{x}_n,\mathbf{x}_k)$.

The replaced values are used in the next iteration of the forest which computes new proximities.

The process it automatically stopped when no more improvement is possible or when five iterations are reached.

Tested on data sets , this replacement method turned out to be remarkably effective.

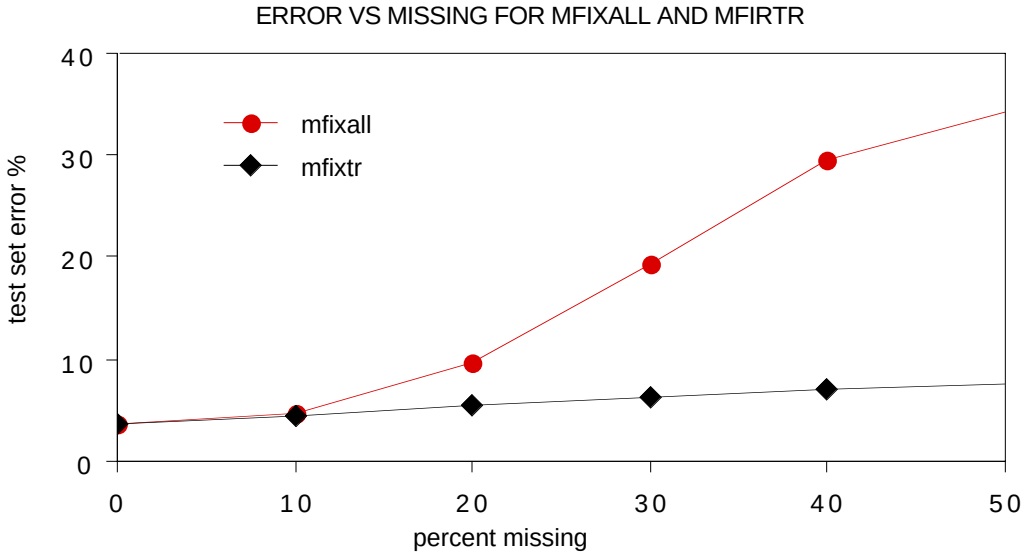
Illustration-DNA Splice Data

the DNA splice data set has 60 variables, all *four valued categorical*, three classes, 2000 cases in the training set and 1186 in the test set.

interesting as a case study because the categorical nature of variables makes many other methods, such as nearest neighbor, difficult to apply.

runs were done deleting 10%, 20%,30%, 40%, and 50%. at random and both methods used to replace.

forests were constructed using the replaced values and the test set accuracy of the forests computed,



It is remarkable how effective the proximity-based replacement process is. Similar results have been gotten on other data sets.

Using Proximites to Picture the Data

Clustering=getting a picture of the data.

To cluster, you have to have a distance, a dissimilarity, or a similarity between pairs of instances.

Challenge: find an appropriate distance measure between pairs of instances in 4691 dimensions. Euclidean? Euclidean normalized?

The values (1-proximity(k,j)) are distances squared in a high-dimensional Euclidean space.

They can be projected down onto a low dimensional space using metric scaling.

Metric scaling derives scaling coordinates which are related to the eigenvectors of a modified version of the proximity'

An Illustration: Microarray Data

81 cases, 4691 variables, 3 classes (lymphoma)

error rate (CV) 1.2%--no variable deletion

Others do as well, but only with extensive variable deletion.

So have a few algorithms that can give accurate classification.

But this is not the goal, more is needed for the science.

1) What does the data look like? how does it cluster?

2) Which genes are active in the discrimination?

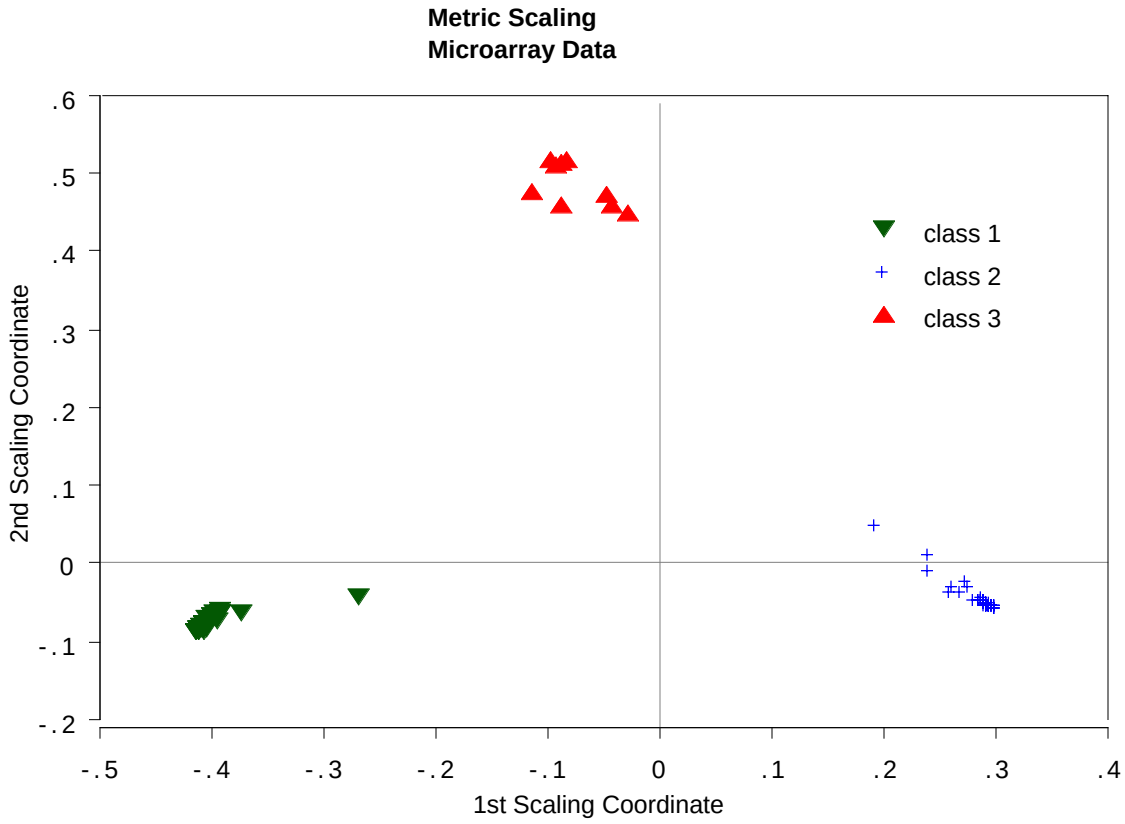
3) What multivariate levels of gene expressions discriminate between classes.

2) can be answered by using variable importance in RF.

now we work on 1) and 3)

Picturing the Microarray Data

The graph below is a plot of the 2nd scaling coordinate vs. the first:



consider the possiblilities of getting a picture by standard clustering methods.

i.e. find an appropriate distance measure between 4691 variables!

Using Proximites to Get Prototypes

Prototypes are a way of getting a picture of how the variables relate to the classification.

For each class j, it searches for that case n1 such that weighted class j cases is among its K nearest neighbors in proximity measure is largest.

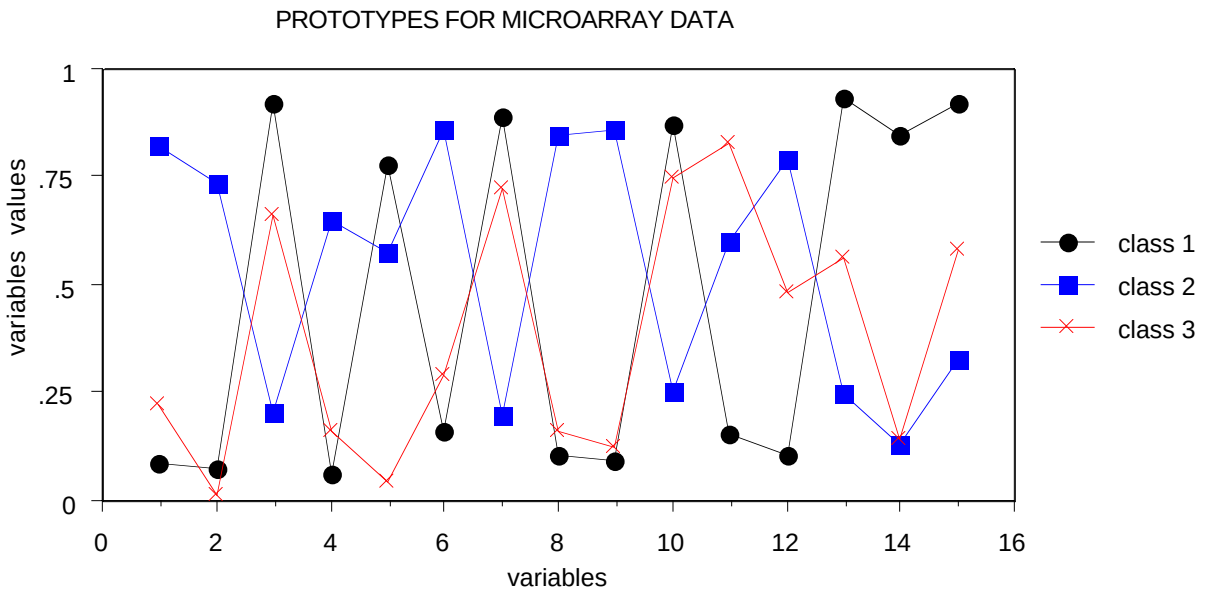
Among these K cases the median, 25th percentile, and 75th percentile is computed for each variable. The medians are the prototype for class j and the quartiles give an estimate of is stability.

For the second class j prototype, a search is made for that case n2 which is not a member of the K neighbors to n1 having the largest weighted number of class j among its K nearest neighbors.

This is repeated until all the desired prototypes for class j have been computed. Similarly for the other classes.

Illustration-Microarray Data

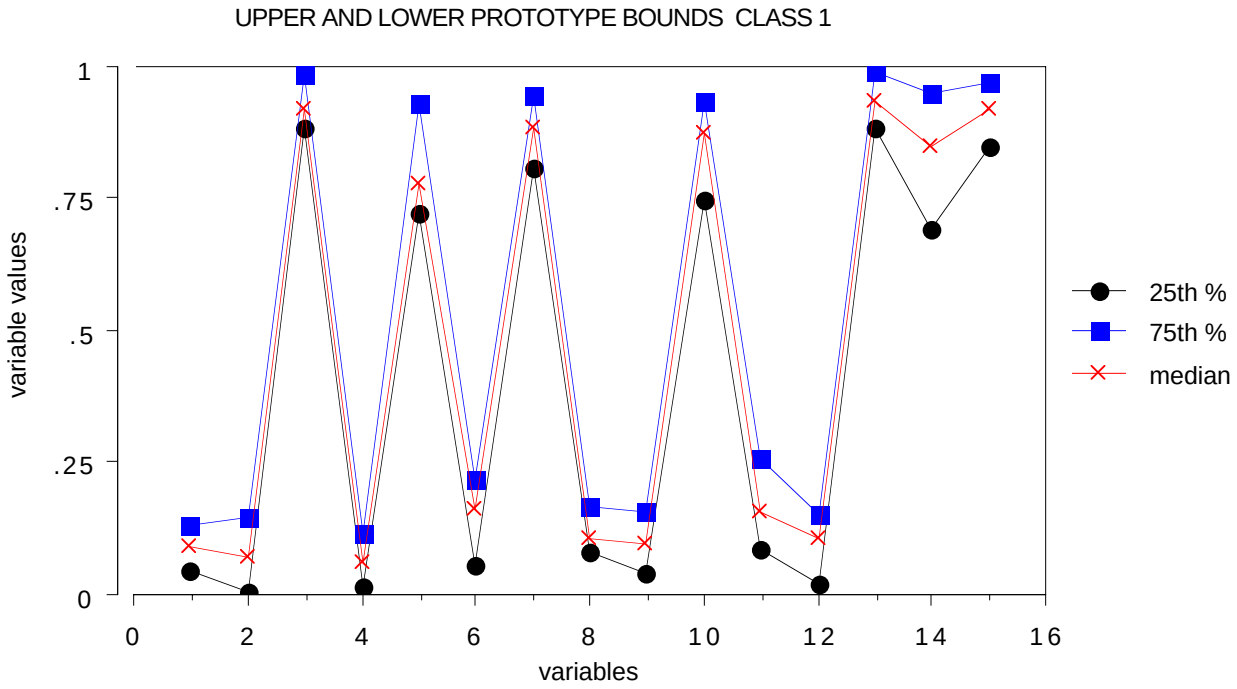
In the microarray data, the class sizes were 29 43 9. K is set equal to 20, and a single prototype is computed for each class using only the 15 most important variables.



It is easy to see from this graph how the separaration into classes works. For instance, cl;ass 1 is low on variables 1,2-high on 3, low on 4, etc.

Prototype Variability

In the same run the 25th and 75th percentiles are computed for each variable. Here is the graph of the prototype for class 2 together with percentiles



The prototypes show how complex the classification process may be, involving the need to look at multivariate values of the variables.

Using Proximities to Find Outliers

Qutliers can be found using proximities An outlier is a case whose proximities to all other cases is small.

Based on this concept, a measure of outlyingness is computed for each case in the training sample.

The measure for case **x**(n) is 1/(sum of squares of prox(**x**(n),**x**(k)) , k not equal to n

Our rule of thumb that if the measure is greater than 10, the case should be carefully inspected.

Outlyingness for the Microarray Data

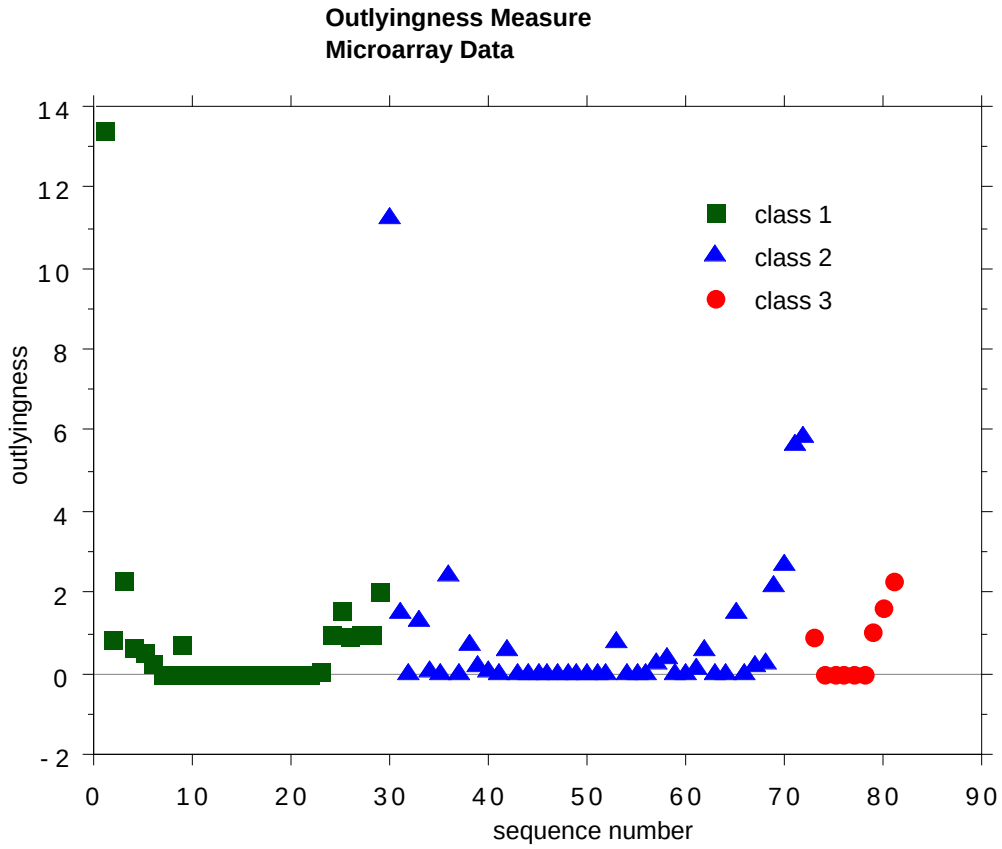
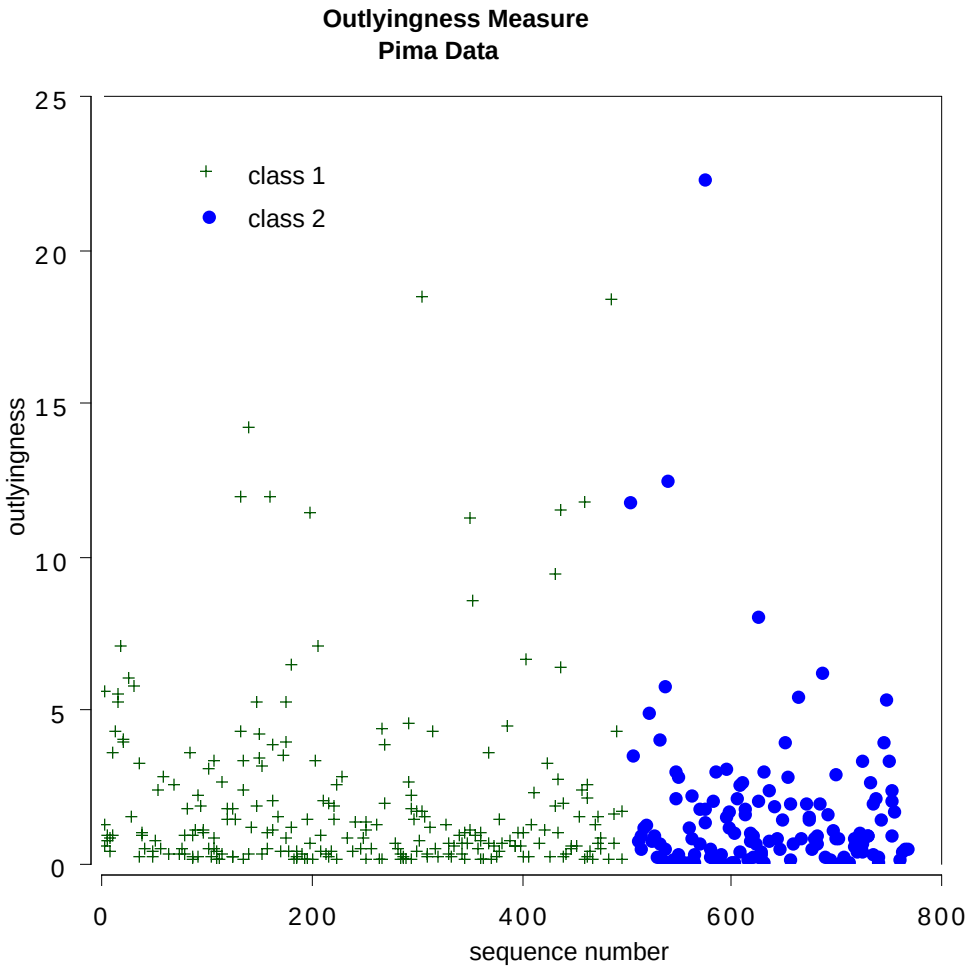


Illustration-- Pima Indian Data

As second example, we plot the outlyingness for the Pima Indians hepatitis data. This data set has 768 cases, 8 variables and 2 classes.

It has been used often as an example in Machine Learning research and is suspected of containing a number of outliers.

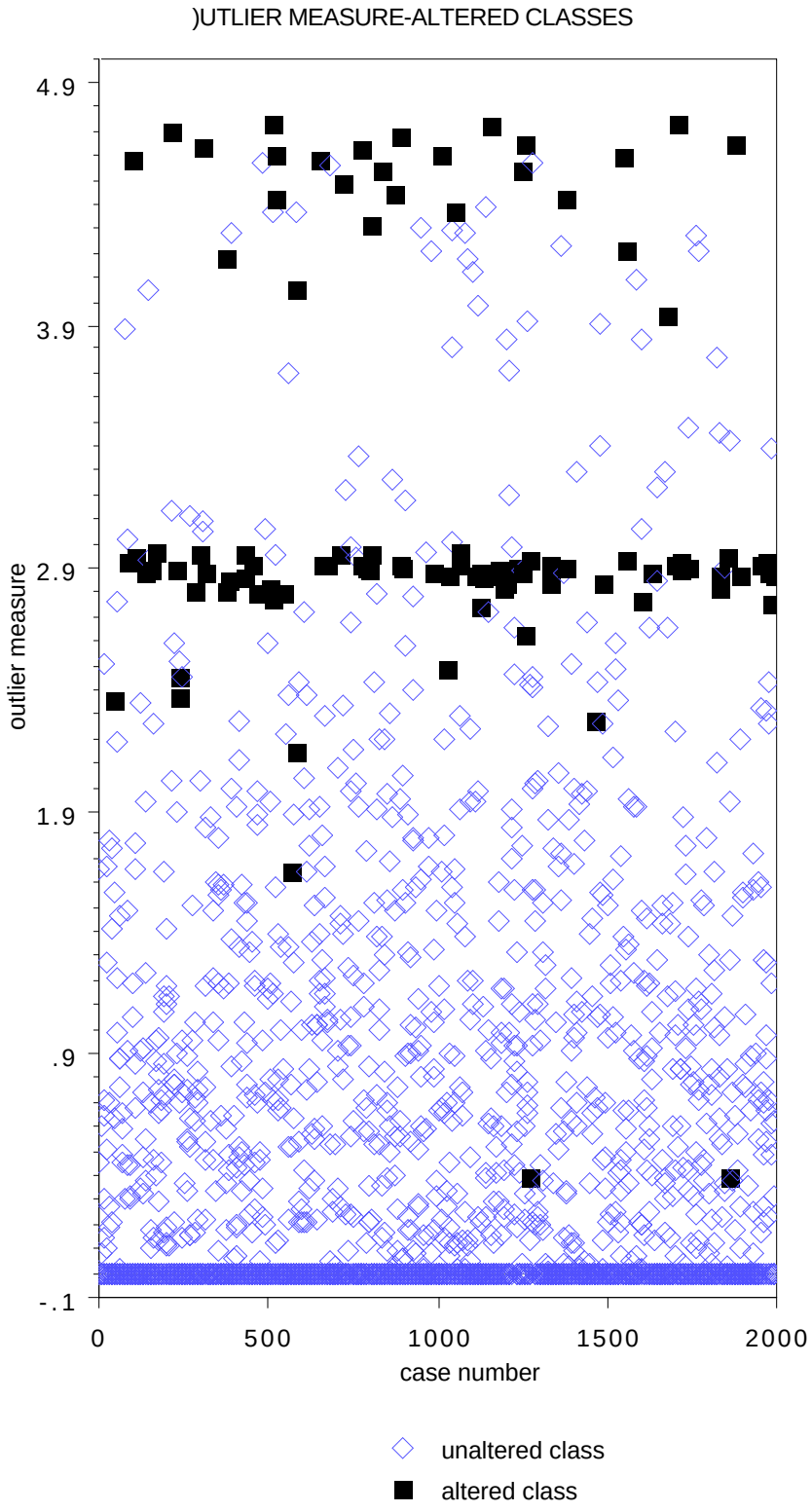


If 10 is used as a cutoff point, there are 12 cases suspected of being outliers.

Mislabeled Data as Outliers

Often, the instances in the training set are labeled by human hand. The results are often either ambiguous or downright incorrect.

Our experiment--change 100 labels at random in the DNA data. Maybe these will turn up as outliers.


)UTLIER MEASURE-ALTERED CLASSES
4.9
3.9
2.9
1.9
.9
-.1
outlier measure
0
500
1000
1500
2000
case number
unaltered class
altered class

Learning from Unbalanced Data Sets

Increasingly often, data sets are occurring where the class of interest has a population that is a small fraction of the total population.

In document classification, the number of relevant documents may be 1-2% of the total number.

In drug searches, the number of active drugs in the sample may be similarly small.

In such unbalanced data, the classifier will achieve great accuracy by classifying almost all cases as the majority case, thus--

completely misclassifying the class of interest.

Example--Small Class In Satellite Data

Class 4 in the satellite data has 415 cases. The other classes total 4020 cases.

objectives--considered as a two class problem (class 4 relabelled class 2) from the rest(class1)

1) equalize the error rates between the classes.

2) find wich variables are important in separating the two classes

1st run: no attention to unbalance

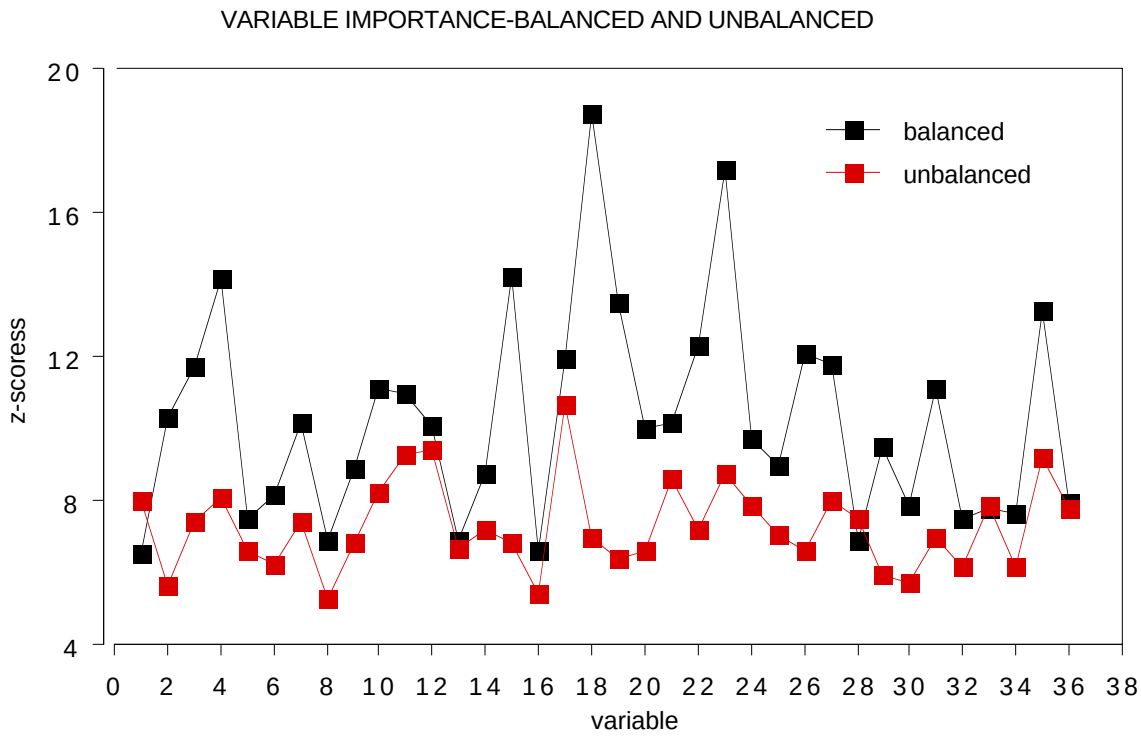
overall error rate 5.8%
error rate-class 2 51.0%
error rate-class 1 1.2%

2nd run: 8:1 weight on class 2

overall error rate 12.9%
error rate-class 2 13.1%
error rate-class 1 11.3%

Variables Importances Unweighted And Weighted

Here is a graph of both z-scores:



There are significant differences. For example, variable 18 becomes important. So does variable 23.

In the unbalanced data, because class 2 is virtually ignored, the variable importances tend to be more equal.

In the balanced case a small number stand out.

Unsupervised Learning Using RF

Unsupervised learning implies that the data has no class labels to guide the analysis.

The data consists of a set of N **x** vectors of the same dimension M.

The most common unsupervised effort is to try and cluster this data to find some "structure"--a most ambiguous project.

Still, random forests demands labels. So we trick it!

Label the original data class 1. I construct a synthetic data set of size N which will be labeled class 2.

Denote the value of the mth variable in the nth instance in the class 1 data as x(m,n).

Here is how each class two instance is constructed. Select the first coordinate at random from the N values {x(1,n)}. Select the 2nd coordinate at random from the N values {x(2,n)}, and so on.

Using the Second Class

The distribution of the 2nd class destroys the dependencies between variables.

It has the distribution of M independent random variables, the mth of which has the same univariate distribution as the mth variable in the original data.

Now we can run the data as a two class problem.

If the error rate is up near 50%, then RF cannot distinguish between the two classes.

Class 1 looks like a sampling from M independent random variables--not a very interesting distribution.

Bit if the separation is good, then all the tools in RF can be used on the original data set.

1) scaling views
2) outlier location
3) missing value replacement
4) prototypes
5) variable importance.

Unsupervised Clustering

Difficulty with clustering: no objective figure of merit.

A proposed test:

take data with class labels.

Erase the labels.

Cluster the data.

Do the clusters correspond to the original classes?

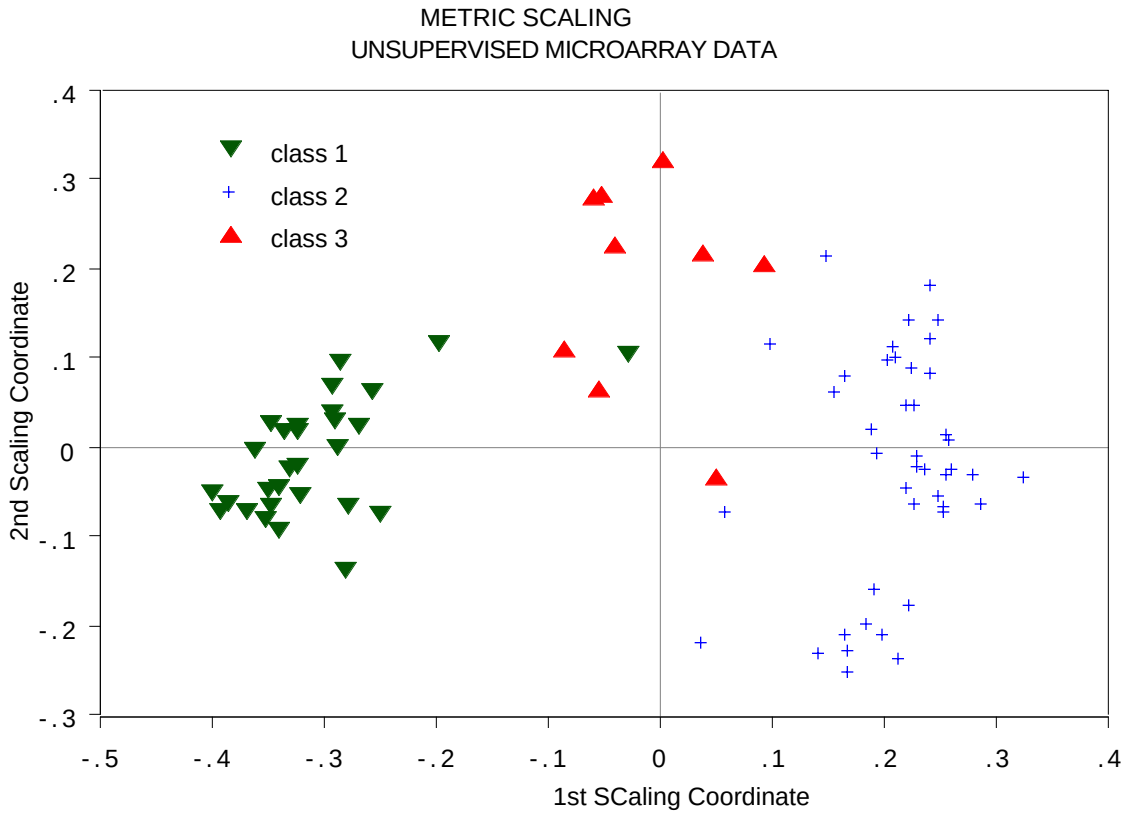
Why isn't this being used to check out the avalanche of clustering algorithms?

So here is random forests response to this test.

The Microarray Data Again

The labels were erased from the data, the synthetic 2nd class formed and RF run on the two class data.

The optimal mtry for the original labelled data is in the range 150-200. For the unsupervised run it is around 50. The error rate is 10%, Here is the scaling picture:

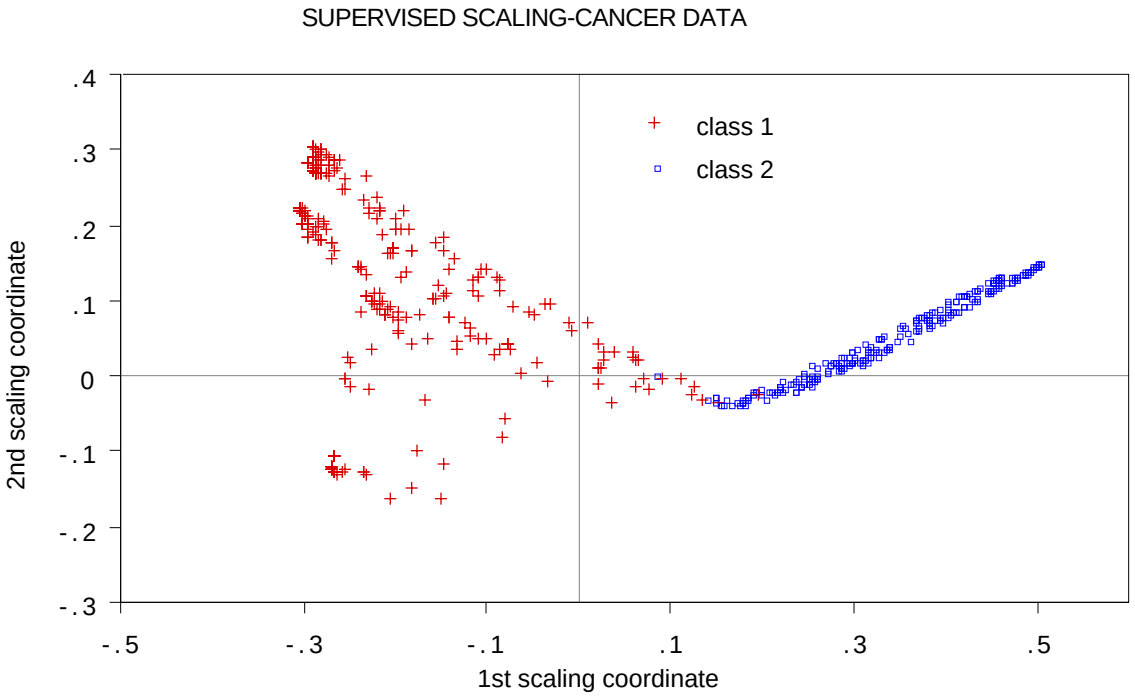


The three clusters appear again.

I*llustration-The Cancer Data*

The cancer data is another classic machine learning benchmark data set. It has 699 cases, 9 variables ,and 2 classes.

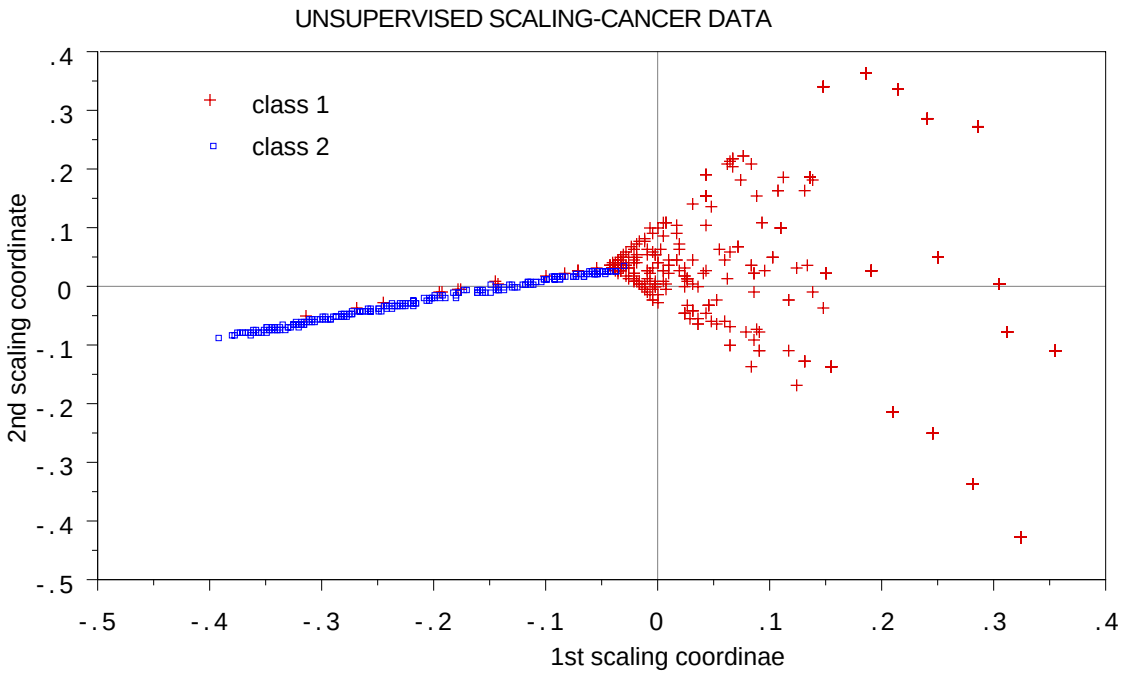
Using labels, the scaling projection is:



The often odd appeasrance of the scaling plots with arms reaching out is due to the nature of the proximities--unlike Euclidean metrics, proximities are locally variable dependent and tend to pull classes further apart.

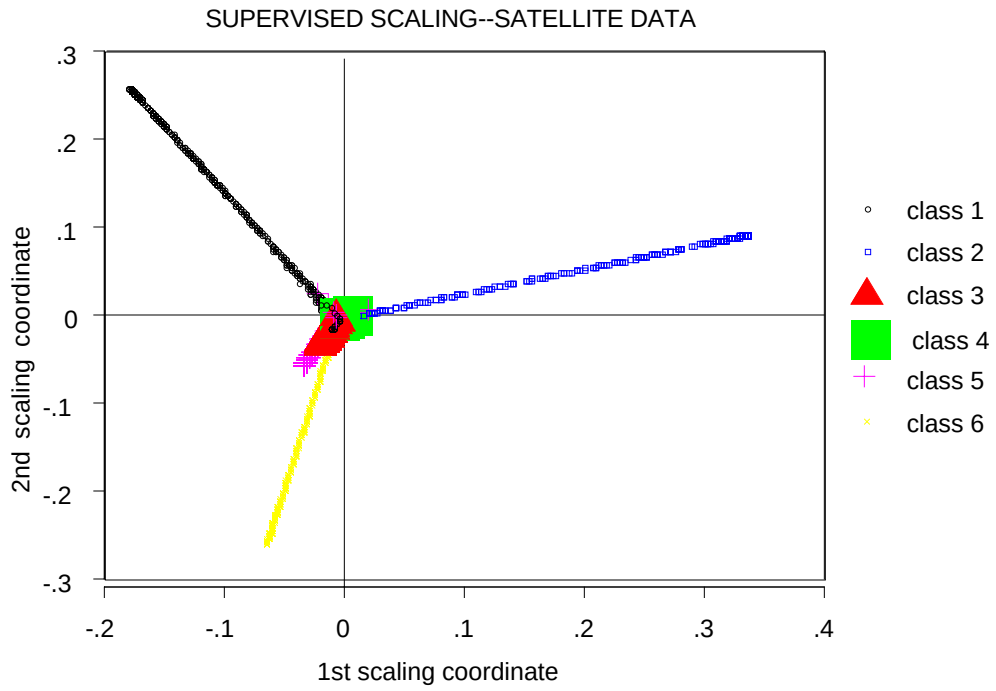
Erasing labels

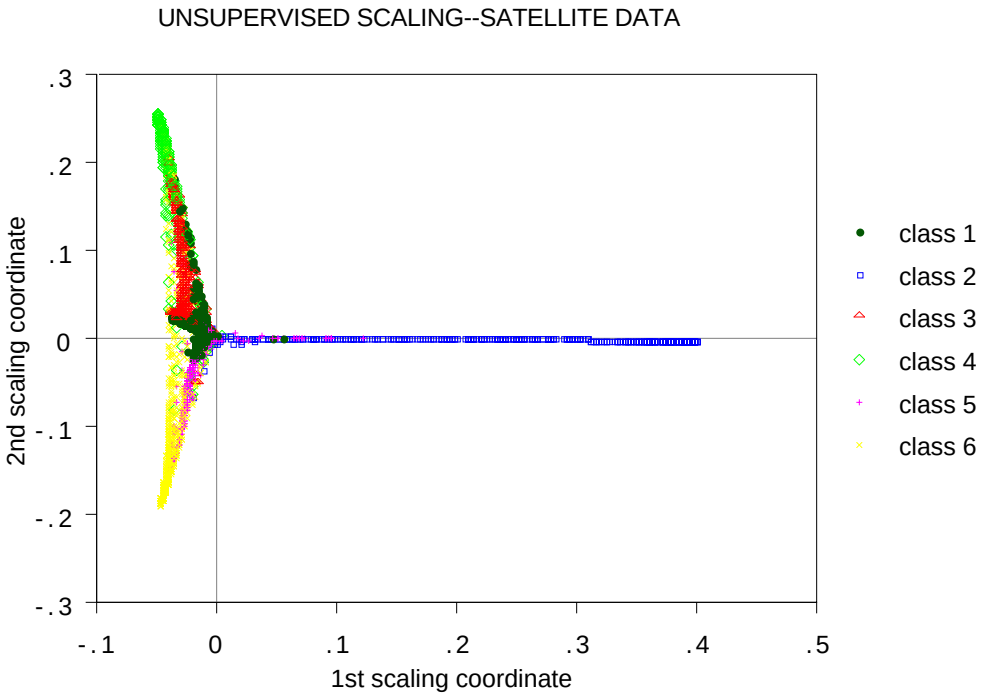
And doing unsupervised scaling gives this picture



The structure of the data is largely retained in unsupervised mode because of dependencies between variables.

Illustration: satellite data





Unsupervised clustering: spectral data

Another example uses data supplied by Merck consists of:

The first 468 spectral intensities in the spectrums of 764 compounds.

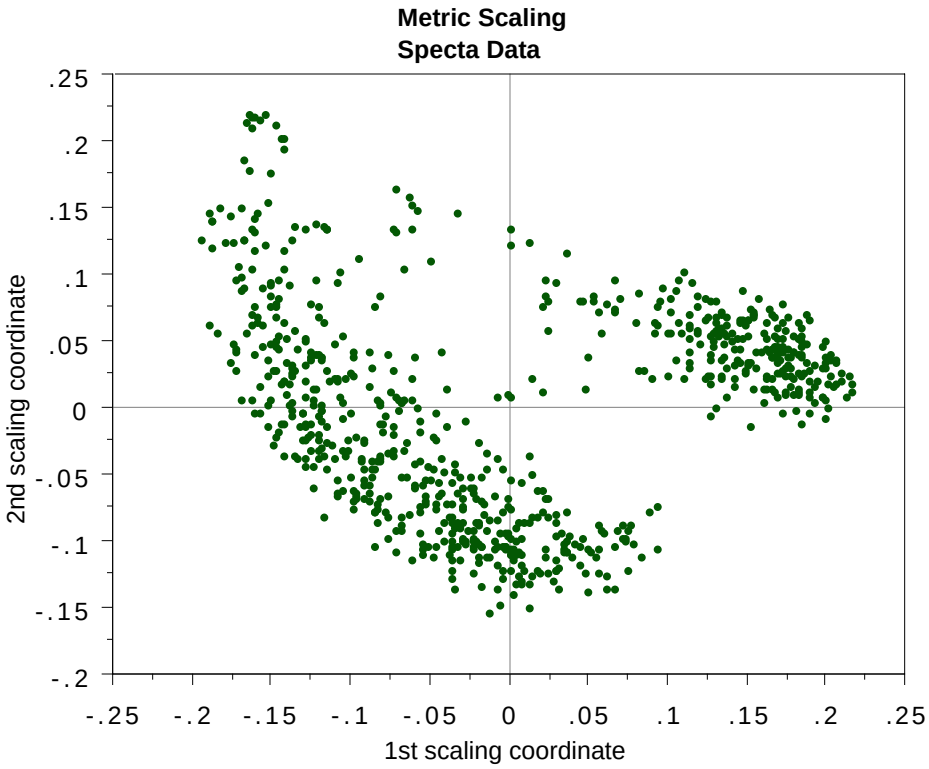
The challenge presented by Merck was to find small cohesive groups of outlying cases in this data.

There is excellent separation between the two classes, with an error rate of 0.5%.

We looked for outliers, and didn't find any.

But outliers must be fairly isolated to show up in the outlier display.

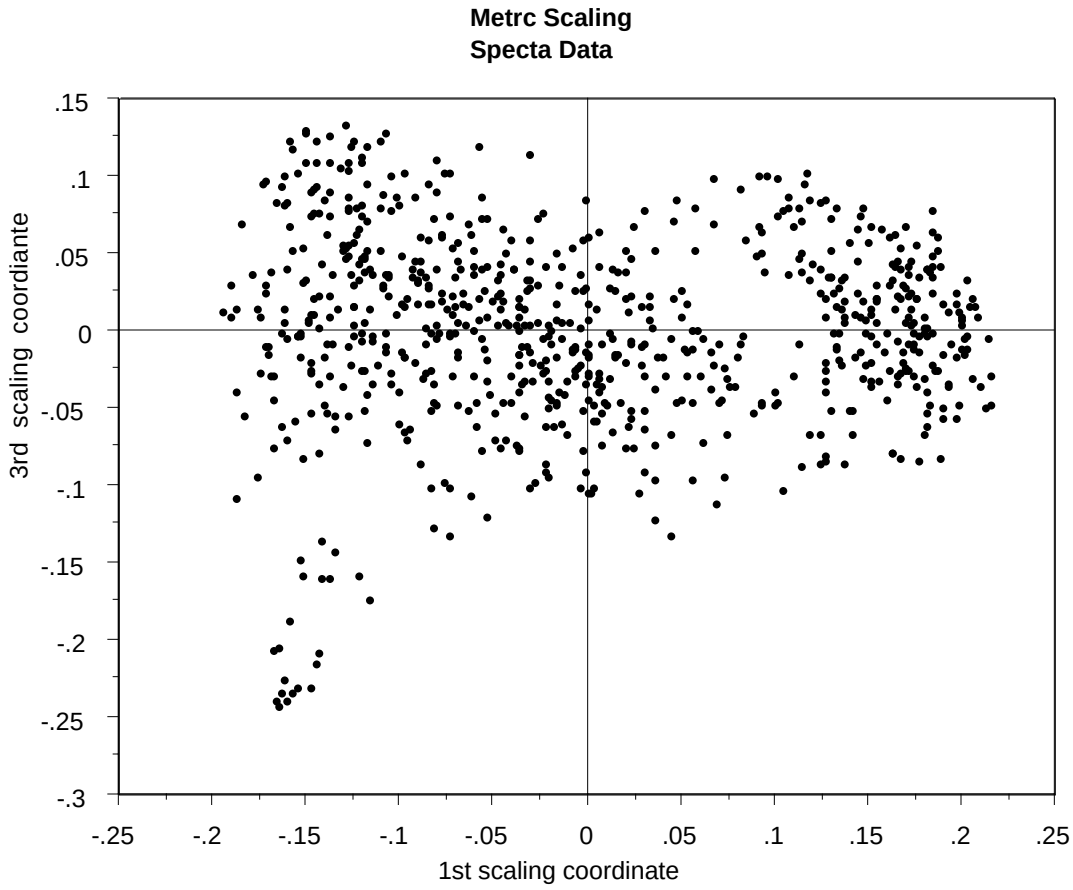
To search for outlying groups scaling coordinates were computed. The plot of the 2nd vs. the 1st is below:



The spectra fall into two main clusters.

There is a possibility of a small outlying group in the upper left hand corner.

To get another picture, the 3rd scaling coordinate is plotted vs. the 1st.



The group in question is now in the lower left hand corner.

It's separation from the main body of the spectra has become more apparent.

RF gives an answer to an non-trivial scientific question.

Explore with Random Forests

The experiment adding 10,000 variables to a data set has this point--

You can add almost as many features (functions of the original variables) as you want and RF will handle the increased dimensionality.

Then it will tell you which are the important features and which are not.

If quadratic interactions are suspected, add all terms of the form x(m)*x(k) and see what falls out.

If domain knowledge is available, use it to form features that you think might be significant.

Using RF gives freedom to explore.

END-PART I