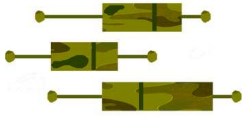
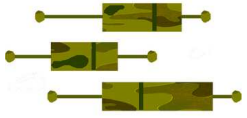
The Curse of Dimensionality



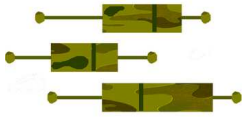
Curse of Dimensionality

- The basic idea of the curse of dimensionality is that high dimensional data is difficult to work with for several reasons:
  - Adding more features can increase the noise, and hence the error.
  - There aren't enough observations to get good estimates.
  - Most of the data is in the tails.



Curse of Dimensionality

- Consider a two class problem, with each class distributed as a multivariate Normal with (known) identity covariance and means $\mu_1 = (1, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{3}}, \ldots, \frac{1}{\sqrt{d}})$ and $\mu_2 = (-1, -\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{3}}, \ldots, -\frac{1}{\sqrt{d}})$.
- Note that the discriminant information gained by increasing the dimension goes to zero.
- Trunk showed that:
  1. If $\mu = \mu_1 - \mu_2$ is known, the probability of error goes to 0.
  2. If μ is unknown the probability of error goes to $\frac{1}{2}$.



The Curse and Kernel Estimators

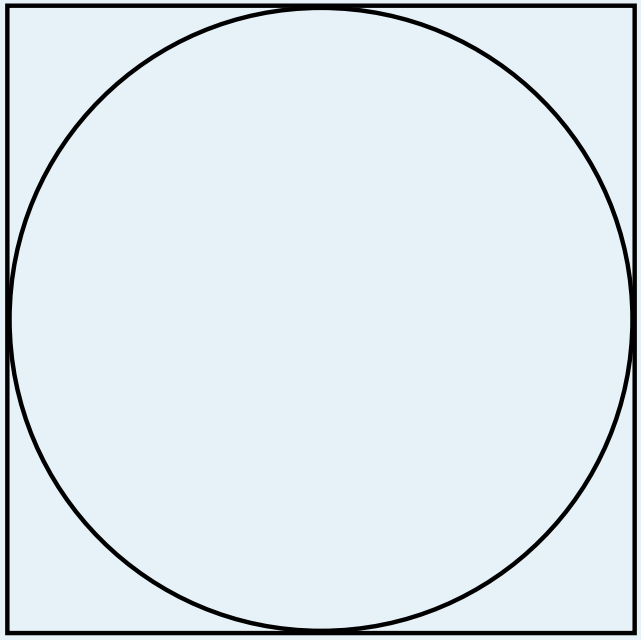
Silverman provides a table illustrating the difficulty of kernel estimation in high dimensions. To estimate the density at 0 with a given accuracy, he reports:

Dimensionality	Required Sample Size
1	4
2	19
5	786
7	10,700
10	842,000

Silverman, *Density Estimation for Statistics and Data Analysis*, 1986, Chapman & Hall.

Ball in Square

- Consider a sphere of radius r inscribed inside a hypercube of dimension d and sides of length $2r$.
- The volume of the hypercube is $(2r)^d$.
- The volume of the sphere is $\frac{2r^d \pi^{d/2}}{d\Gamma(d/2)}$.

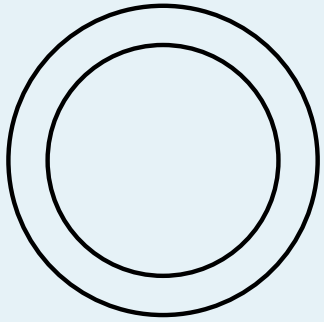


Ball in Square

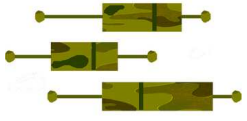
- Thus the proportion of the volume of the square that is inside the sphere is

$$\frac{\pi^{d/2}}{d2^{d-1}\Gamma(d/2)} \to 0$$

  as $d \to \infty$.
- For large d all the mass is in the corners of the hypercube.
- Similar calculations show that all the mass in a ball is near the edge (like a soccer ball).



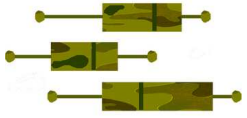
Dimensionality Reduction



Dimensionality Reduction

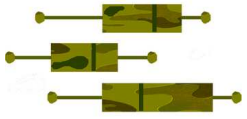
We want to reduce the number of dimensions because:

- Efficiency.
  - Measurement costs.
  - Storage costs.
  - Computation costs.
- Classification performance.
- Ease of interpretation/modeling.

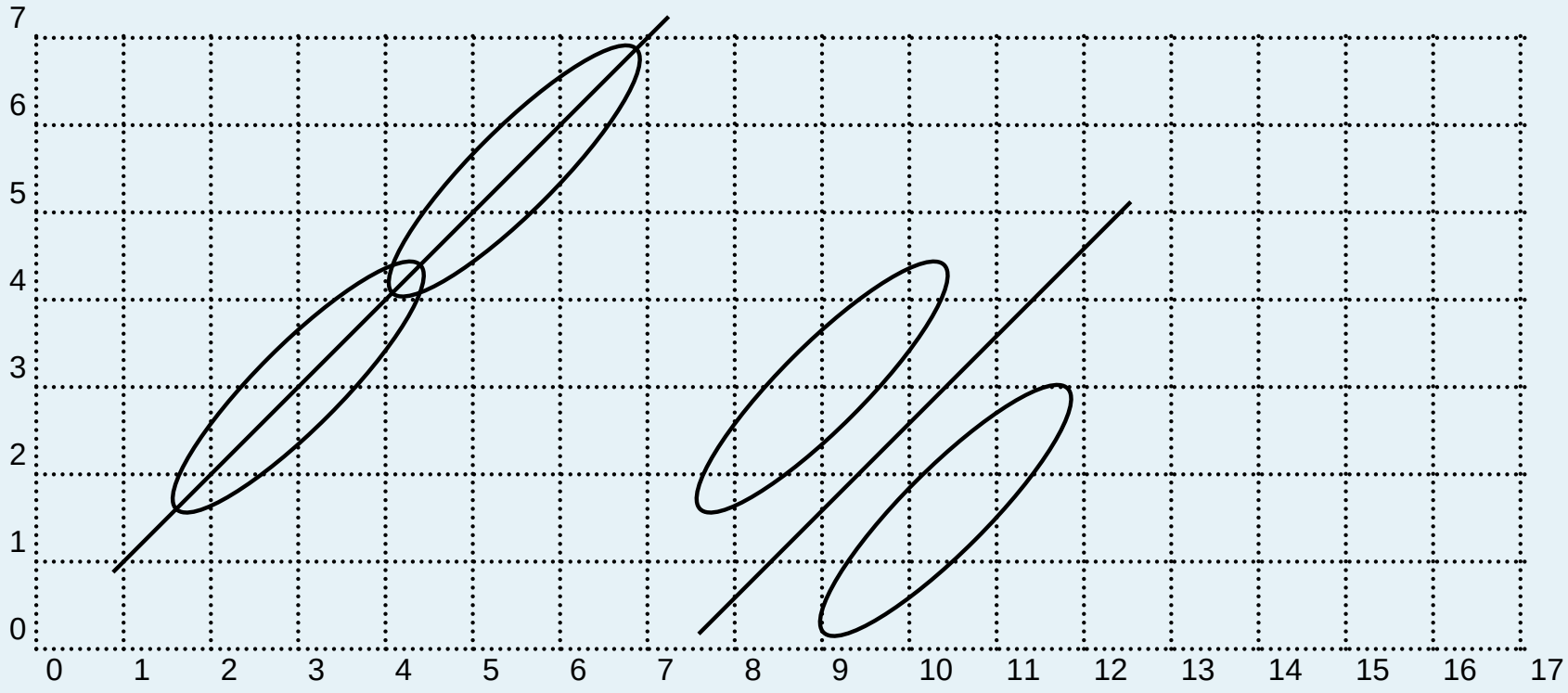


Principal Components

- The idea is to project onto the subspace which accounts for most of the variance.
- This is accomplished by projecting onto the eigenvectors of the covariance matrix associated with the largest eigenvalues.
- This is generally **not** the projection best suited for classification.
- It can be a useful method for providing a first-cut reduction in dimension from a high dimensional space.



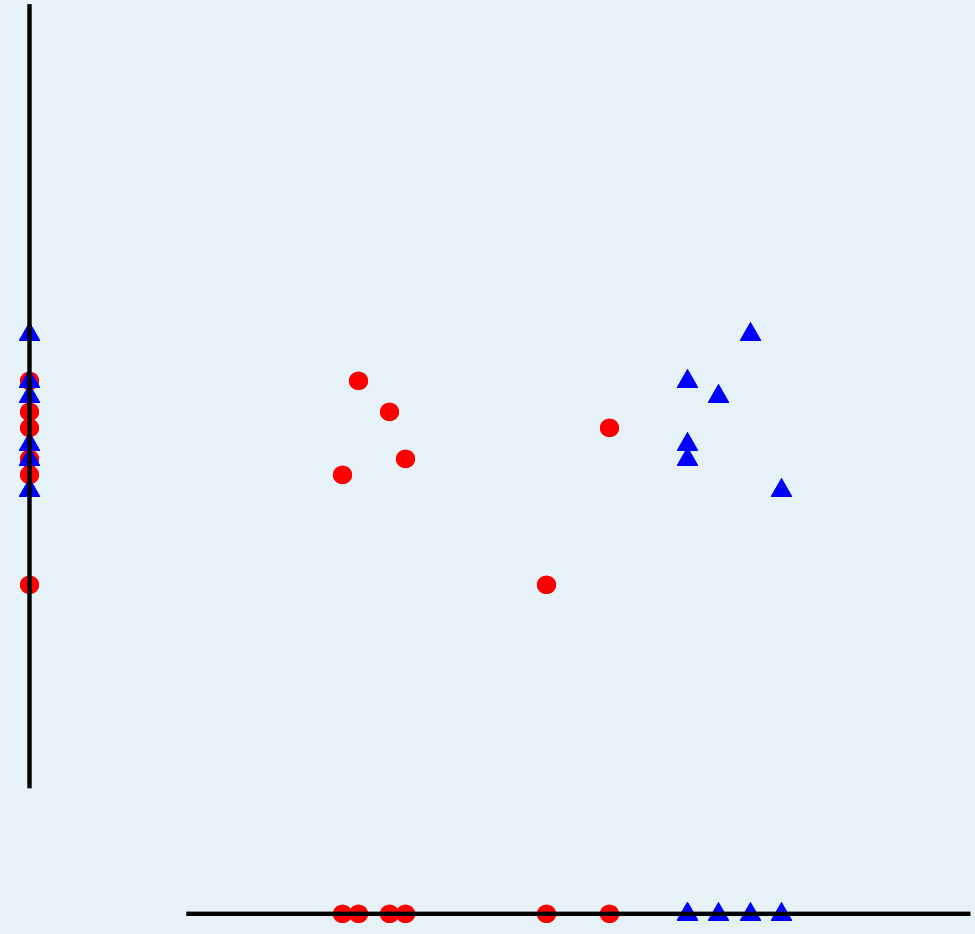
Principal Components

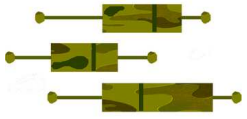


- In the first case the PC is the optimal projection to 1D.
- In the second case it is perpendicular to the optimal projection.

Fisher's Linear Discriminant - Motivation

- Another look at the effect of projection axis. Want to find the optimal projection for classification.





Fisher's Linear Discriminant - Methodology

- Let $x_1, x_2, \cdots, x_n$ be a set of d-dimensional samples with n_1 of these in class 1 and n_2 of these in class 2
- Given a projection vector w define
  - $|\tilde{m}_1 - \tilde{m}_2| = |w^t(m_1 - m_2)|$
  - $\tilde{s}_i^2 = \sum_{y \in class_i} (y - \tilde{m}_i)^2$
- FLD seeks to maximize
  - $J(w) = \frac{|\tilde{m}_1 - \tilde{m}_2|^2}{\tilde{s}_1^2 + \tilde{s}_2^2}$

FLD - Bottom Line

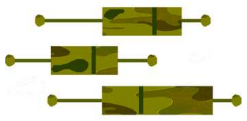
- The optimization is easier than it looks. Let

$$S_i = \sum_{x \in class_i} (x - m_i)(x - m_i)^t$$

$$S_W = S_1 + S_2$$

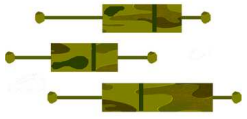
Then

$$w = S_W^{-1}(m_1 - m_2).$$



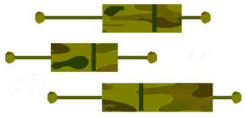
Local Fisher's Linear Discriminant

- The FLD is optimal if the data are Normal, with equal covariances.
- One can extend this idea to local projections as follows.
  - For each training observation x_i, define a local region (say, the k nearest training observations from each class).
  - Construct the FLD for that local region.
  - For new data, use the FLD appropriate to the region in which the new data lies.
- This results in a local-linear classifier, which can approximate much more general decision regions.



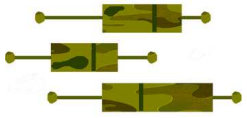
Multi-class Fisher's Linear Discriminant

- If there are c classes, one can compute the FLD in a manner similar to the 2 class case.
- The optimization is a little more complicated.
- The projection is $c-1$ dimensional.
- Alternatively, one can perform pairwise 2 class FLDs:
  - Class i versus class j.
  - Class i versus all other classes.



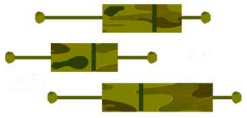
Projection Pursuit Exploratory Data Analysis

- A search for and exploration of nonlinear structure in a multi-dimensional data set via a sequence of two-dimensional projections (Friedman and Tukey (1974)).
- The hope is that two-dimensional projections will reveal structure that is resident in the original data.
- Purposes of projection pursuit
  - Exploratory data analysis
  - Clustering
  - Discriminant analysis



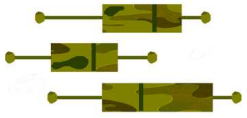
Projection Pursuit Exploratory Data Analysis in a Nutshell

- Formulate an index that measures the degree that a particular projection reveals the property of interest (departure from normality, cluster structure, class separation).
- Formulate a method to solve for the projection that yields the largest projection pursuit index.
- Proceed.



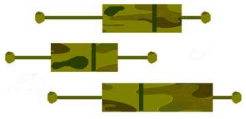
Projection Pursuit - EDA Notation - I

- X is a $n \times d$ matrix, where each row (X_i) corresponds to a d-dimensional observation and n is the sample size.
- Z is the sphered version of X.
- $\hat{\mu}$ is the $1 \times d$ sample mean:$\hat{\mu} = \frac{1}{n}\Sigma X_i$.
- $\hat{\Sigma}$ is the sample covariance matrix: $\hat{\Sigma_{ij}} = \frac{1}{n-1}\sum(X_i - \hat{\mu})(X_j - \hat{\mu})^T$.
- α, β are orthonormal ($\alpha^T\alpha = 1 = \beta^T\beta$ and $\alpha^T\beta = 0$) d-dimensional vectors that span the projection plane.
- $P(\alpha, \beta)$ is the projection plane spanned by α, β.



Projection Pursuit - EDA Notation - II

- z_i^α, z_i^β are the sphered observations projected onto the vectors α and β:
  $z_i^\alpha = z_i^T \alpha$
  $z_i^\beta = z_i^T \beta$.
- (α^*, β^*) denotes the plane where the index is maximum.
- $PI_{\chi^2}(\alpha, \beta)$ denotes the chi-squared projection index evaluated using the data projected onto the plane spanned by α and β.
- ϕ_2 is the standard bivariate Normal density.

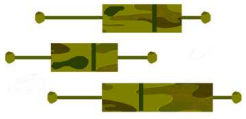


Projection Pursuit - EDA Notation - III

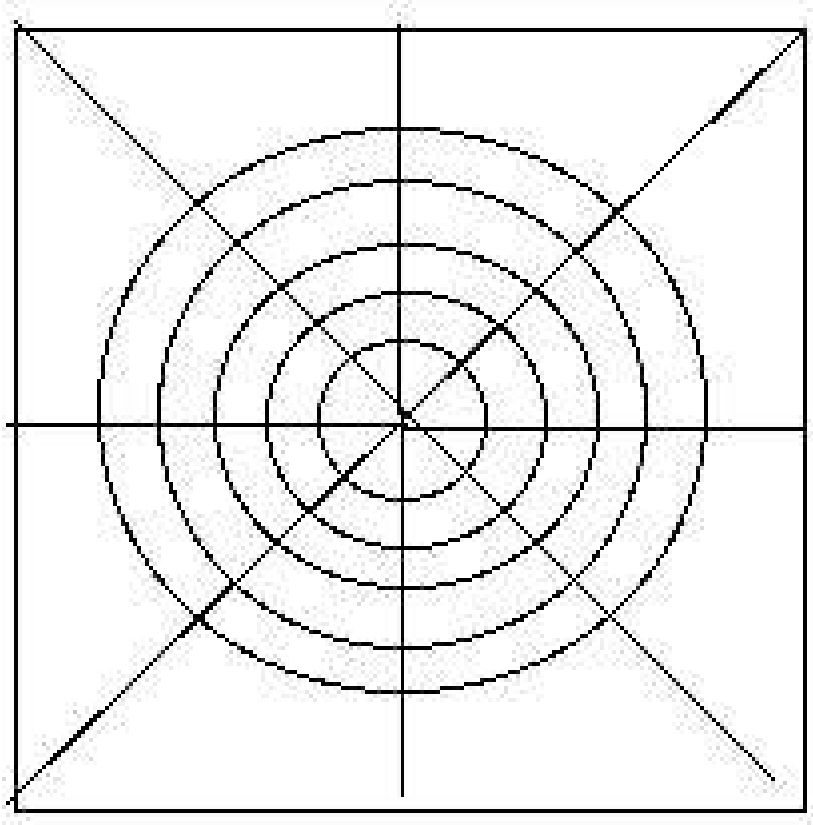
- c_k is the probability evaluated over the k-th region using the standard bivariate Normal density,
  $c_k = \int \int_{B_k} \phi_2 dz_1 dz_2$.
- B_k is a box in the projection plane.
- I_{B_k} is the indicator function for region B_k.
- $\eta_j = \pi_j/36, j = 0, \cdots, 8$ is the angle by which the data are rotated in $\alpha(\eta_j)$ and $\beta(\eta_j)$ are given by
  $\alpha(\eta_j) = \alpha cos\eta_j - \beta sin\eta_j$
  $\beta(\eta_j) = \alpha sin\eta_j + \beta cos\eta_j$.

Projection Pursuit - EDA Notation - IV

- c is a scalar that determines the size of the neighborhood around (α^*, β^*) that is visited in the search for planes that provide better values for the projection pursuit index.
- ν is a vector uniformly distributed on the unit d-dimensional sphere.
- $half$ specifies the number of steps without an increase in the projection index, at which time the value of the neighborhood is halved.
- m represents the number of searches or random starts to find the best plane.



B_k Regions for the χ^2 Projection Pursuit Index



The Empirical Projection Pursuit Index

- For Computer Implementation

$$PI_{\chi^2}(\alpha,\beta) = \frac{1}{9}\sum_{j=1}^{8}\sum_{k=1}^{48}\frac{1}{c_k}\left[\frac{1}{n}\sum_{i=1}^{n} I_{B_k}(z_i^{\alpha(\eta_j)}, z_i^{\beta(\eta_j)}) - c_k\right]^2$$

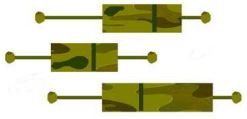
- Robust to outliers.
- Sensitive to distributions with holes and to clusters.

Finding the Structure

- How do we optimize the projection index over all possible projections to R^2?
- Method of Posse

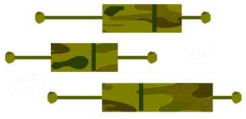
$$a_1 = \frac{\alpha^* + c\nu}{\|\alpha^* + c\nu\|} \qquad b_1 = \frac{\beta^* - (a_1^T\beta^*)a_1}{\|\beta^* - (a_1^T\beta^*)a_1\|}$$

$$a_2 = \frac{\alpha^* - c\nu}{\|\alpha^* - c\nu\|} \qquad b_2 = \frac{\beta^* - (a_2^T\beta^*)a_2}{\|\beta^* - (a_2^T\beta^*)a_2\|}$$



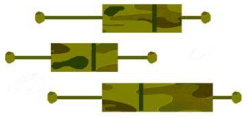
Algorithm - Projection Pursuit Exploratory Data Analysis-I

1. Sphere the data using
   $Z_i = \Lambda^{-1/2} Q^T (X_i - \hat{\mu})\ i = 1, \cdots, n$
   Q = eigenvectors of $\hat{\Sigma}$; Λ = diagonal matrix of corresponding eigenvalues, X_i = i-th observation.
2. Generate a random starting plane, (α_0, β_0)
   $(\alpha^*, \beta^*) = (\alpha_0, \beta_0)$.
3. Calculate the projection index for the starting plane $PI_{\chi^2}(\alpha_0, \beta_0)$.
4. Generate two candidate planes (a_1, b_1) and (a_2, b_2).
5. Evaluate the value of the projection index for these planes, $PI_{\chi^2}(a_1, b_1)$ and $PI_{\chi^2}(a_2, b_2)$.



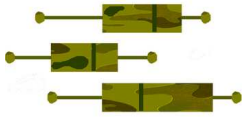
Algorithm - Projection Pursuit Exploratory Data Analysis-II

6. If one of the candidate planes has a higher index then it becomes the new best plane (α^*, β^*).
7. Repeat steps 4 through 6 while there are improvements in the projection pursuit indices.
8. If the index does not improve for half times, then decrease the value of c by half.
9. Repeat steps 4 through 8 until c is some small number set by the user.

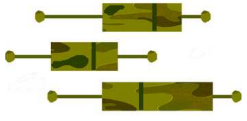


Projection Pursuit Exploratory Data Analysis - Conclusion

- Typically an iterative process with structure removed at each step of the process.
- Discriminant analysis needs might lead one to consider alternate projection pursuit indices.

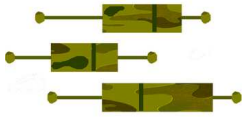


Trees, Graphs and Combining Classifiers

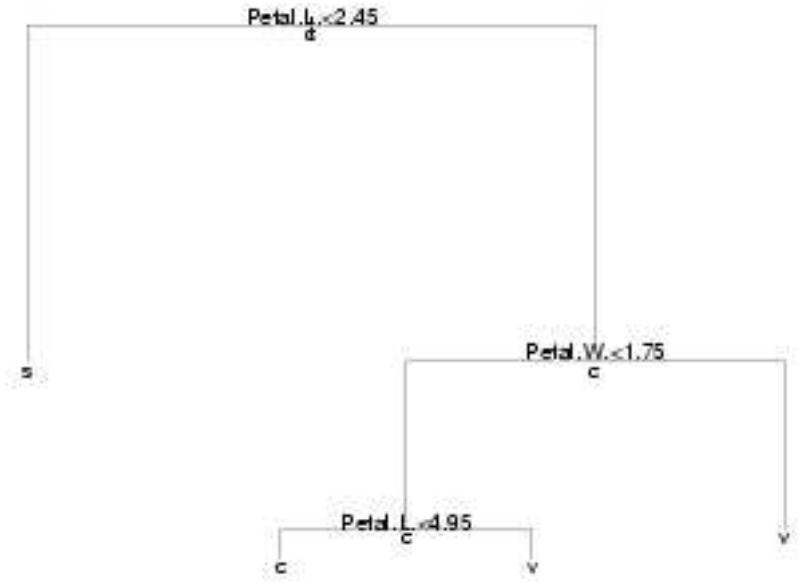


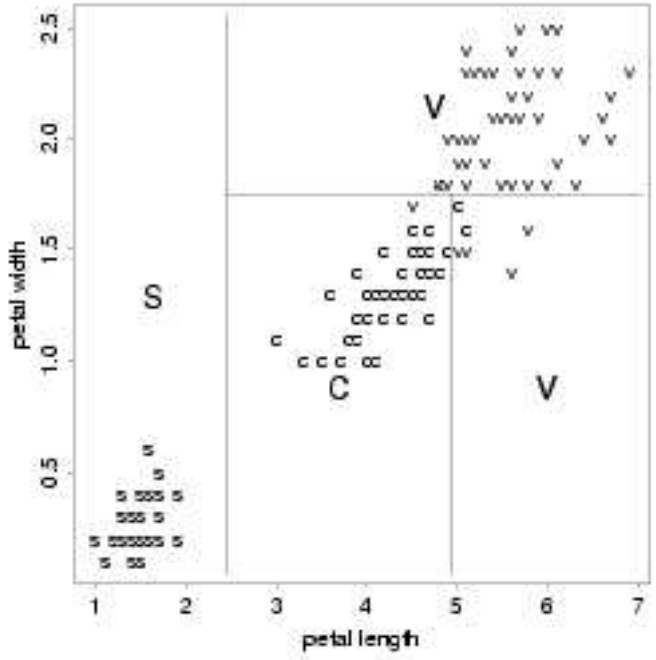
Classification Trees

- One of the first, and most popular is CART (Classification and Regression Trees).
- The idea is to build a tree of simple decision rules.
- Each branch of the tree is taken based on a comparison of a single variable with a value or threshold.
- CART implementation issues
  - How many splits shall we allow at a node?
  - What properties should be tested at a node?
  - When should a node be declared a leaf or terminal node?
  - How shall we prune trees?
  - If a leaf is impure then how shall we assign the class label?
  - What shall we do with missing data?



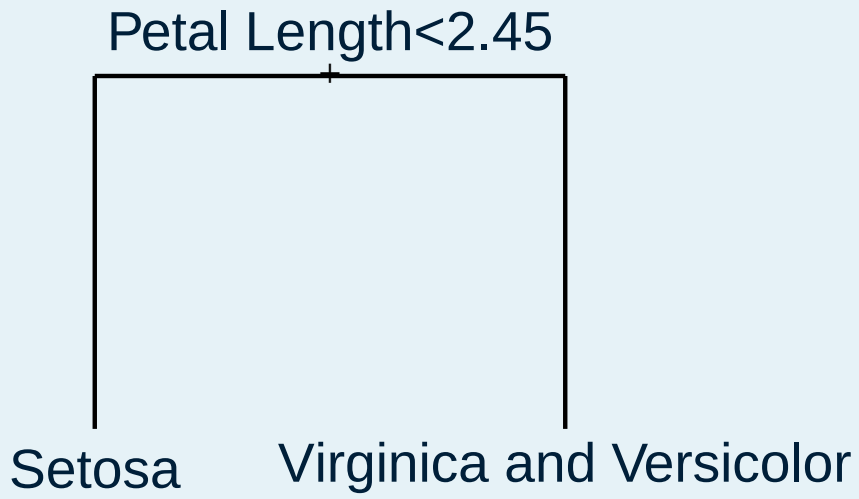
CART Iris Analysis





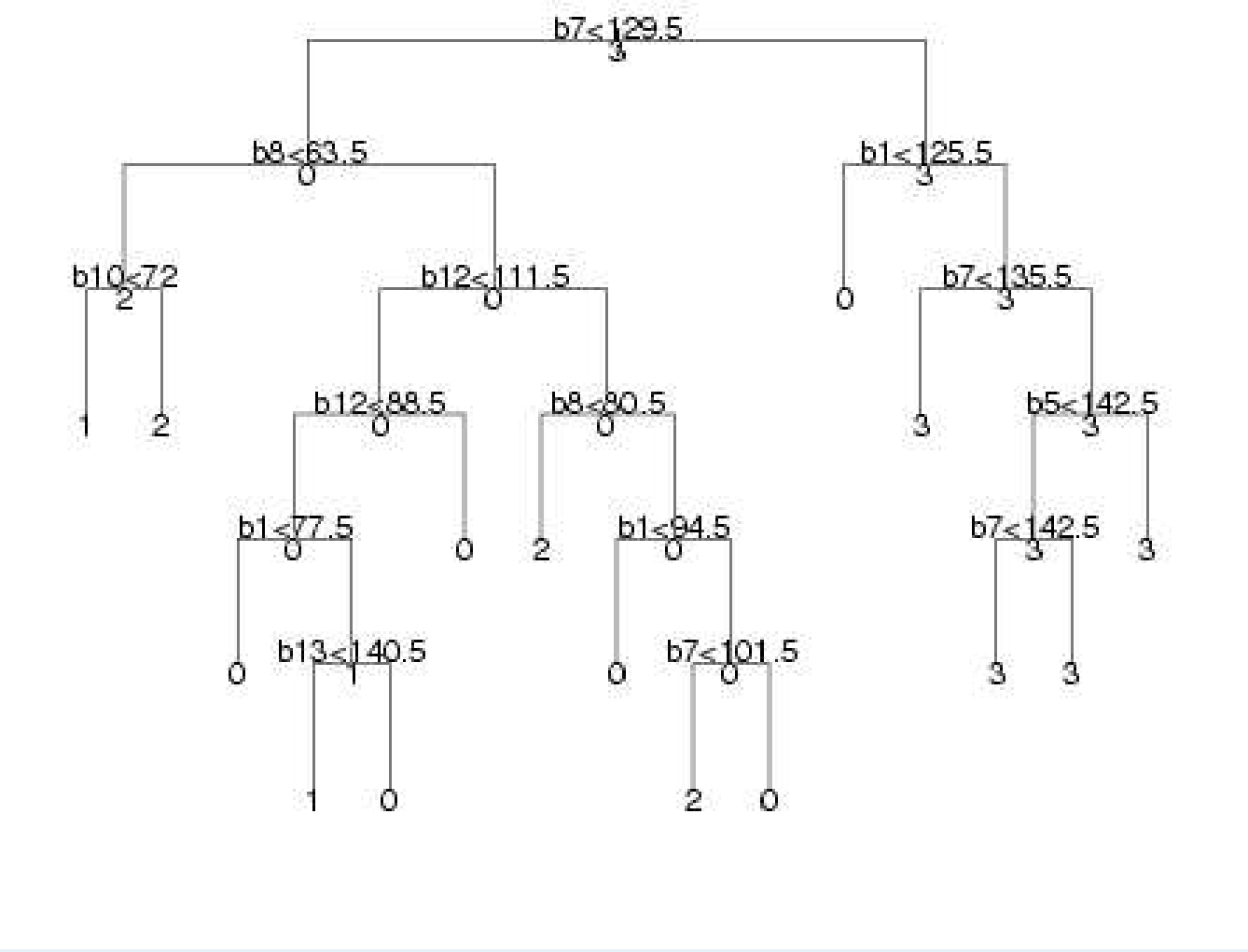
CART Details

- At each node in the tree, the data is split according to a single variable.
- The variable and split value are selected in order to best split the data.



- The tree is grown until each leaf is pure (single class).
- The tree is then pruned to reduce overfitting.
- The variables can be continuous or categorical.

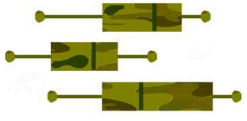
SALAD CART Analysis on Full 13 Bands



This tree uses bands 1,5,7,8,10,12,13.

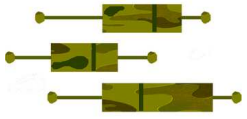
CART Properties

- Benefits
  - CART is an appropriate technique when not much is known about the form of an optimal classifier.
  - CART is an appropriate technique when presented with non-metric data.
  - The classifier can often be easily understood as a sequence of rules, and is thus easily understood by a client.
  - Like nearest neighbor classifiers, it often produces a reasonable classifier for comparison with other techniques.
  - Missing values are easily incorporated in the classifier.



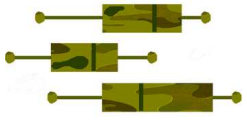
CART Properties

- Drawbacks
  - CART is highly sensitive to the training data set. Small changes in the training set can have a marked effect on the nature of the classifier.
  - Large trees are not really more “understandable” than other classifiers.
  - The decision region is not smooth (an intersection of rectangles).



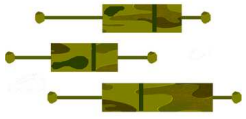
ID3 and C4.5

- The machine learning community has also developed decision trees.
- ID3 was designed to work on nominal data.
- It handles real-valued data via binning.
- The algorithm continues until all of the terminal nodes are pure or there are no more variables to split upon.
- C4.5, a follow-on to ID3, is similar to CART, except that different choices are made in the design of the tree.
- For example, it allows:
  - Variable numbers of branches from a node.
  - “Soft” thresholds, to account for observations near the threshold.
- Machine learning is also interested in generating rules, and there are methods for turning a tree into a set of rules, which can be evaluated by experts, or used in an expert system.



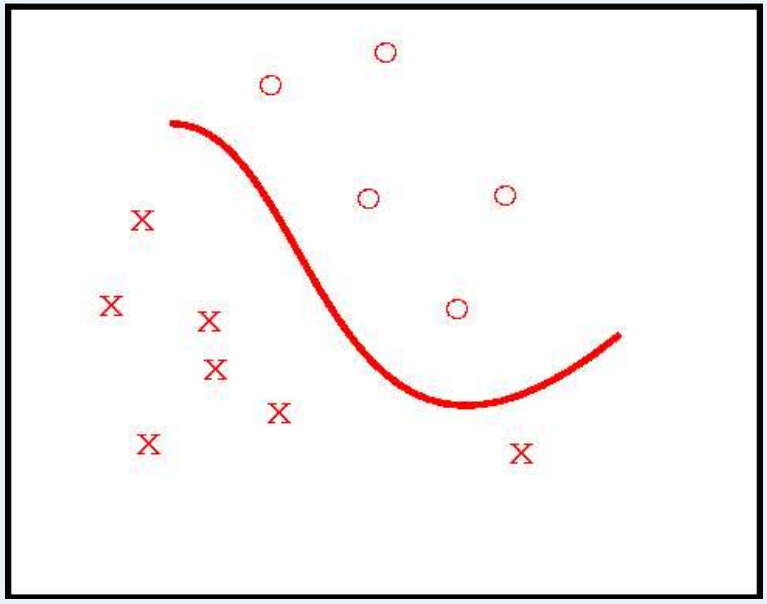
Dual Representation of the Linear Classifier

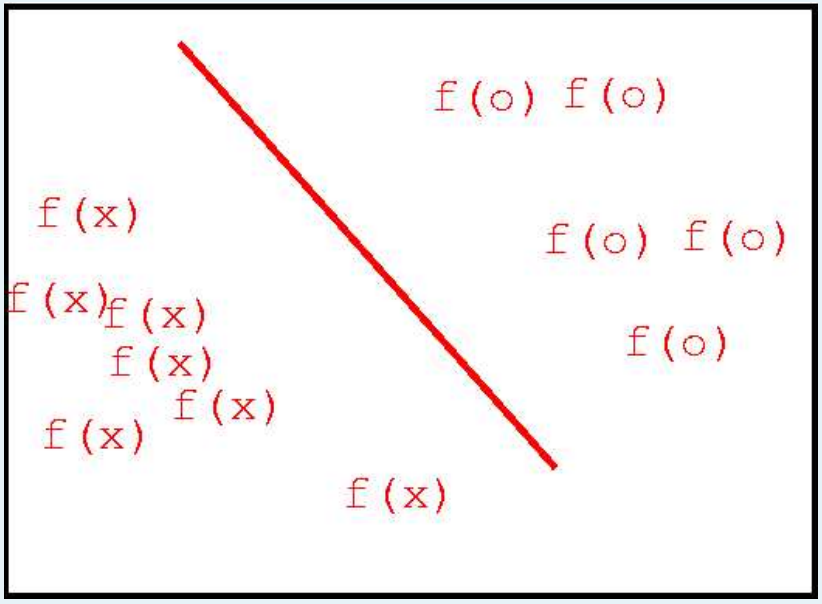
- The linear decision function may be rewritten as follow
- $f(x) =< w, x > +b = \Sigma\alpha_i y_i < x_i, x > +b \;\; w = \Sigma\alpha_i y_i x_i$
- Support vector machines are linear classifiers represented in a dual fashion.
- Data appears within dot products in the decision and training functions



Support Vector Machines (The Cartoon)

- We seek a mapping f where we may separate our points via a simple linear classifier



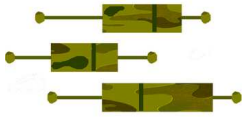


Kernels

- A function that returns the dot product between the images of the two arguments

$$K(x_1, x_2) = < \phi(x_1), \phi(x_2) >$$

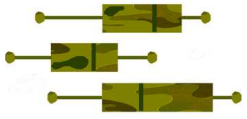
- Given a function K it is possible to verify that it is a kernel.
- One can use linear classifiers in a feature space by simply rewriting it in dual representation and replacing dot products with kernels.



Kernel Examples

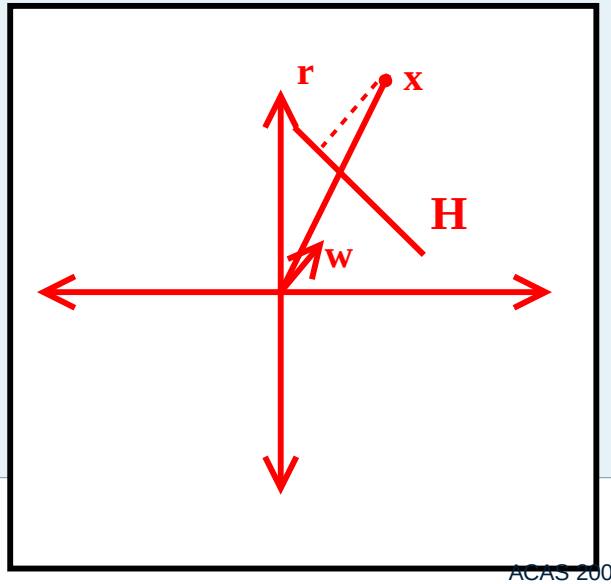
- $K(x,z) = < x, z >^d$
- $K(x,z) = e^{\frac{-||x-z||^2}{2\sigma}}$
- Polynomial kernel:
  - Let $x = (x_1, x_2)$ and $z = (z_1, z_2)$.
  - We may write

$$\begin{aligned} < x, z > &= (x_1 z_1 + x_2 z_2)^2 \\ &= x_1^2 z_1^2 + x_2^2 z_2^2 + 2 x_1 z_1 x_2 z_2 \\ &= < (x_1^2, x_2^2, \sqrt{2} x_1 x_2), (z_1^2, z_2^2, \sqrt{2} z_1 z_2) > \\ &= < \phi(x), \phi(z) > . \end{aligned}$$



Linear Discriminant Functions and Their Surfaces

- We may form a discriminant function that is a linear combination of the components of x.
  - $g(x) = w^t x + w_0$
  - w is the weight vector and w_0 is the bias or threshold.
- g(x) measures the distance r from x to the discriminant hyperplane.
  - $r = \frac{g(x)}{\|w\|}$.

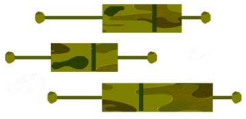


Homogeneous Form of the Linear Discriminant Functions

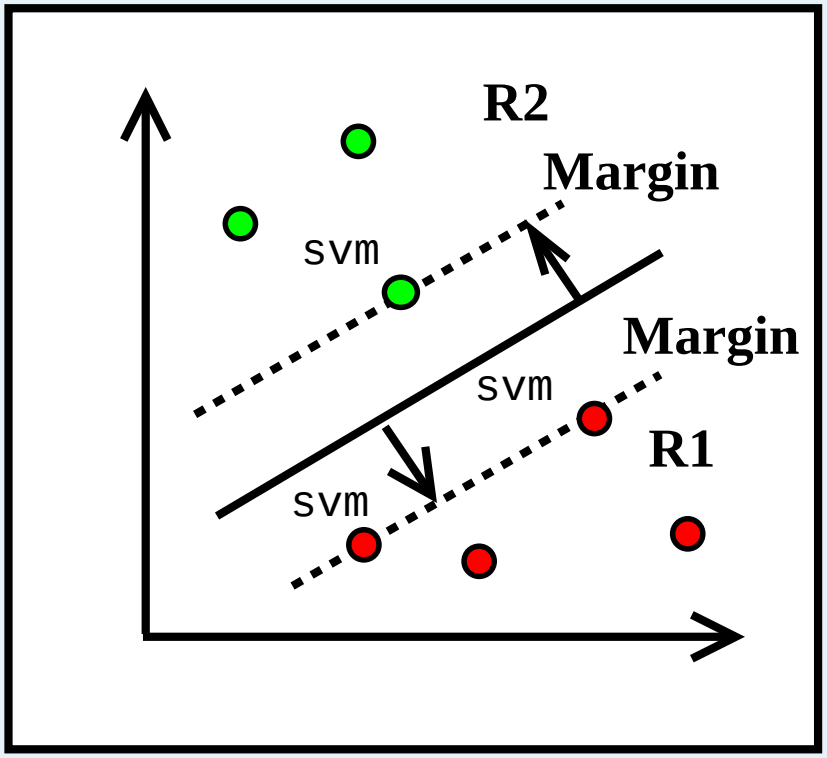
$$g(x) = w^t x + w_0$$

$$g(x) = w_0 + \sum_{i=1}^{d} w_i x_i = \sum_{i=0}^{d} w_i x_i$$

$$y = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_d \end{bmatrix} \quad a = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix}$$



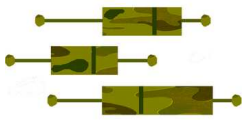
SVM Training Cartoon



- We seek a hyperplane with maximum margin.
- The training vectors that define the margins are the support vectors.

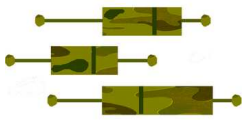
SVM Criterion Function - I

- Let our training patterns be x_k where $k = 1, \cdots, n$.
- Assume the patterns have been subjected to a transformation ϕ to produce $y_k = \phi(x_k) \forall k$.
- Each y_k has an associated class $\omega_k = \pm 1$.
- Our LD in the augmented space is given by $g(y) = a^t y$.
- $z_k g(y_k) \geq 1, k = 1, \cdots, n$.



SVM Criterion Function - II

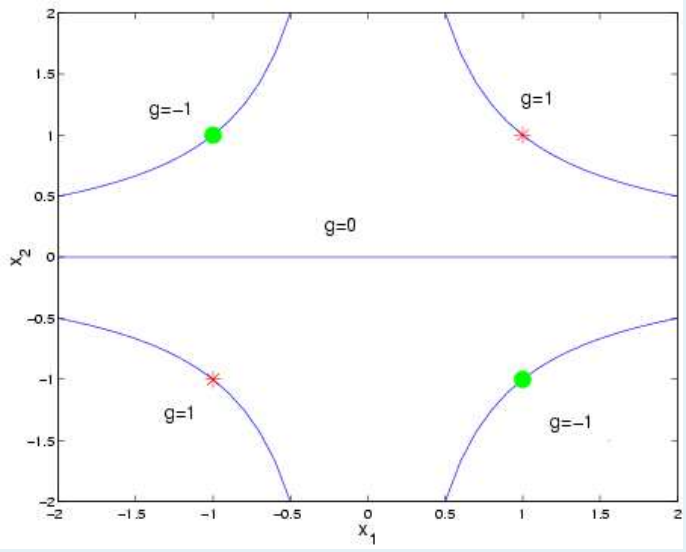
- Recall the distance from our hyperplane to a transformation point y is given by $\frac{|g(y)|}{||a||}$.
- Assuming b a positive margin exists we have $\frac{z_k g(y_k)}{||a||}$
- $||a|| = 1$ to ensure uniqueness.
- Minimize $||a||^2$.



Solving for the Support Vectors

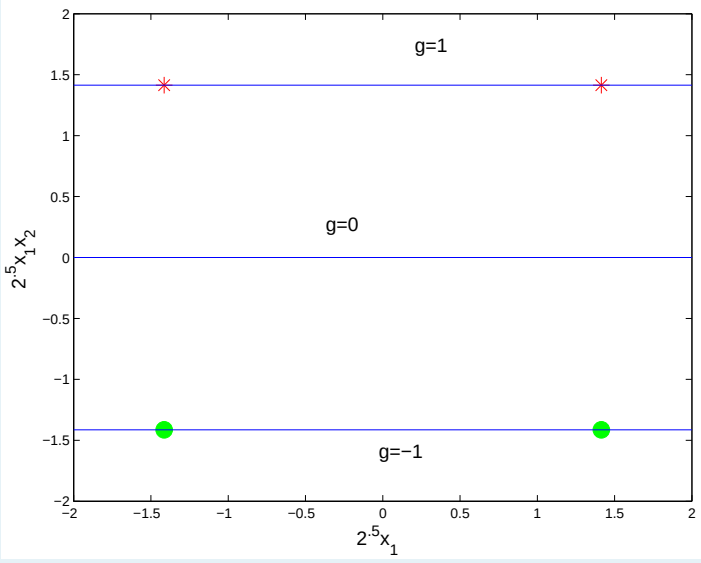
- Choosing ϕ
  - Domain knowledge
  - Polynomials
  - Gaussians
- $L(a, \alpha) = \frac{1}{2}\|a\|^2 - \sum_{k=1}^{n} \alpha_k \left[z_k a^t y_k - 1\right]$
- $L(\alpha) = \sum_{k=1}^{n} \alpha_i - \frac{1}{2} \sum_{k,j}^{n} \alpha_k \alpha_j z_k z_j y_j^t y_k.$
- Subject to $\sum_{k=1}^{n} z_k \alpha_k = 0$ $\alpha_k \geq 0, k = 1, \cdots, n.$

A Trivial Example SVM: X-OR



- $\phi : R^2 \rightarrow R^6 \; where \; \phi(x_1, x_2) = (1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2, x_1^2, x_2^2)$
- We seek to maximize $\sum_{k=1}^{4} \alpha_k - \frac{1}{2} \sum_{k,j}^{n} \alpha_k \alpha_j z_k z_j y_j^t y_k$
- Subject to $\alpha_1 - \alpha_2 + \alpha_3 - \alpha_4 = 0 \; where \; 0 \leq \alpha_k \; k = 1, 2, 3, 4.$

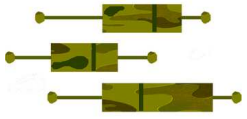
Two-Dimensional Projection of Six-Dimensional Space



- $g(x_1, x_2) = x_1 x_2 = 0.$
- $b = \sqrt{2}.$
- Hyperplanes through the support vectors $\sqrt{(2)}x_1 x_2 = \pm 1$ in the mapped space correspond to hyperbolas $x_1 x_2 = \pm 1$ in the original space.

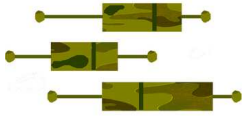
Boosting and Bagging

- Many classifiers can be improved by taking aggregates of classifiers (ensemble classifiers).
- The idea is to construct many classifiers, and combine their results into some overall classification.
- The simplest of these ideas is Bagging (bootstrap aggregation):
  - Draw a bootstrap sample of size $n' < n$ and construct a *component classifier* on the sample.
  - The overall classifier is a vote of the component classifiers.



Boosting

- The idea of boosting is to construct an ensemble of classifiers, each specialized to the points that previous classifiers had difficulty with.
  - Construct a classifier and determine the observations that it makes errors on.
  - Reweight the observations to give these difficult observations greater weight.
  - Construct a new classifier on these reweighted observations, and repeat.
  - The final classifier is a weighted average of the individual ones.



AdaBoost

Initialize k_{max}, $W = \{\frac{1}{n}, \ldots, \frac{1}{n}\}$, $k = 0$

Do

k=k+1

Train classifier C_k using weights W_k.

Variation: train on a bootstrap sample according to W.

$E_k =$ weighted error of C_k.

$\alpha_k = \ln\left(\frac{1-E_k}{E_k}\right)$.

$W_i = W_i e^{I(C_k(x_i)=y_i)\alpha_k}$.

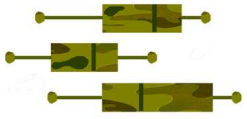
Normalize W.

While $k < k_{max}$.

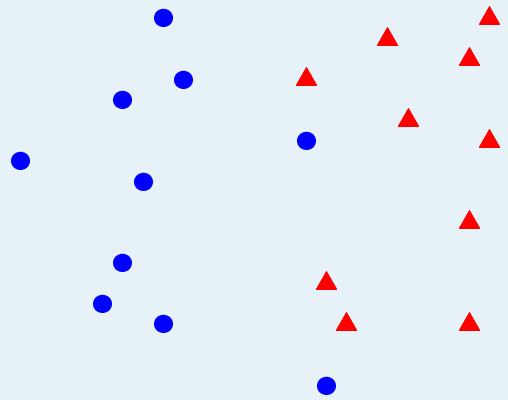
Return $C(x) = \text{sign} \sum_{k=1}^{k_{max}} \alpha_k C_k(x)$

Class Cover

- Assume (for now) that we have two classes whose supports do not intersect (so they are perfectly separable).
- Assume further that we have no model for the classes (or their separating surface).
- The class cover problem is to cover one class (X) with balls such that none of the balls intersect the other class (Y).
- One way to approximate this is to center balls at X observations with radii no larger than the distance from the observation to any Y observation.

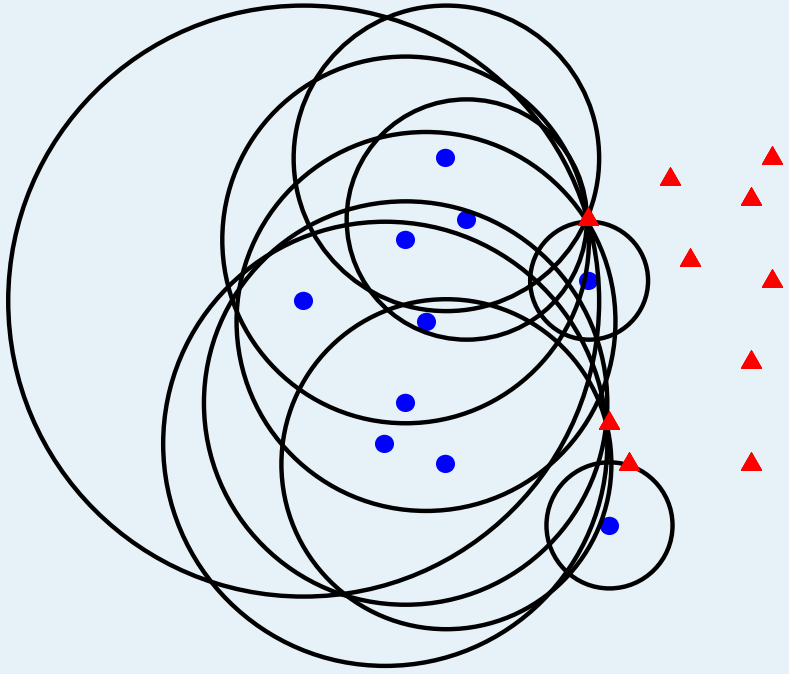


Class Cover Example



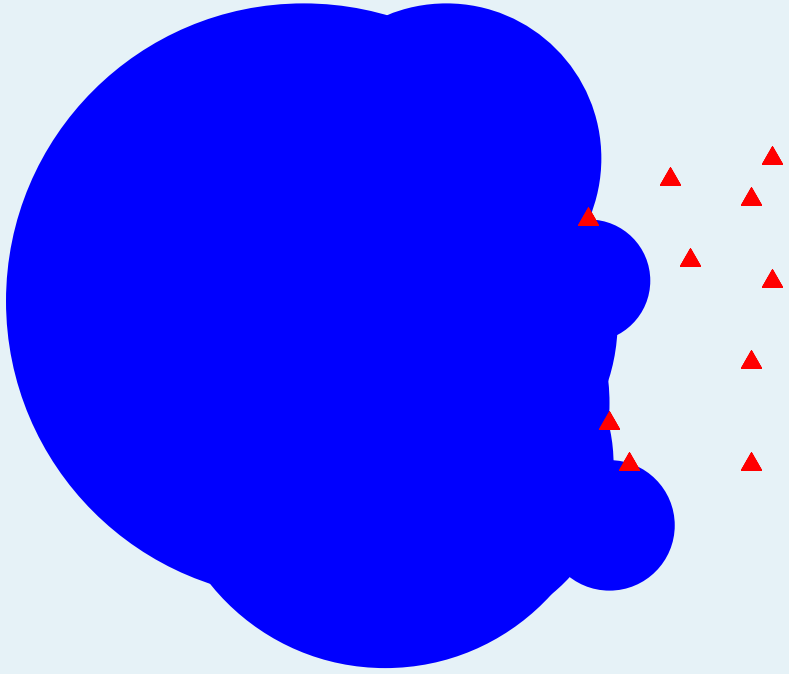
- We want to cover the support of one class, without covering that of the other.

Class Cover Example



- Circles indicate regions “near” blue observations. They are defined to be the largest circle centered on a blue that does not cover a red.

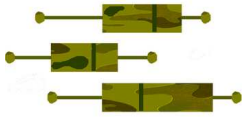
Class Cover Example



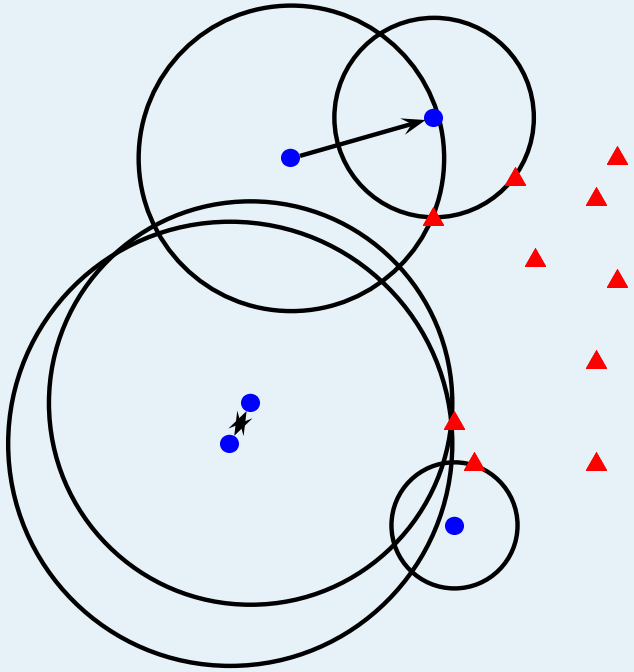
- Blue area shows the decision region for the blue class.

Class Cover Catch Digraph

- Note that we have one ball per observation, which is more than necessary. We want to reduce the complexity of the cover.
- Define a directed graph (digraph) as follows:
  - The vertices are the observations from X.
  - There is an edge from x_1 to x_2 if the ball at x_1 covers x_2.
- We **cover** the **class** by a **digraph** defined by the balls that **catch** (contain) the vertices.
- We abbreviate this CCCD.

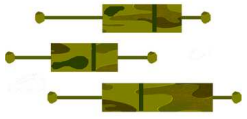


CCCD Example

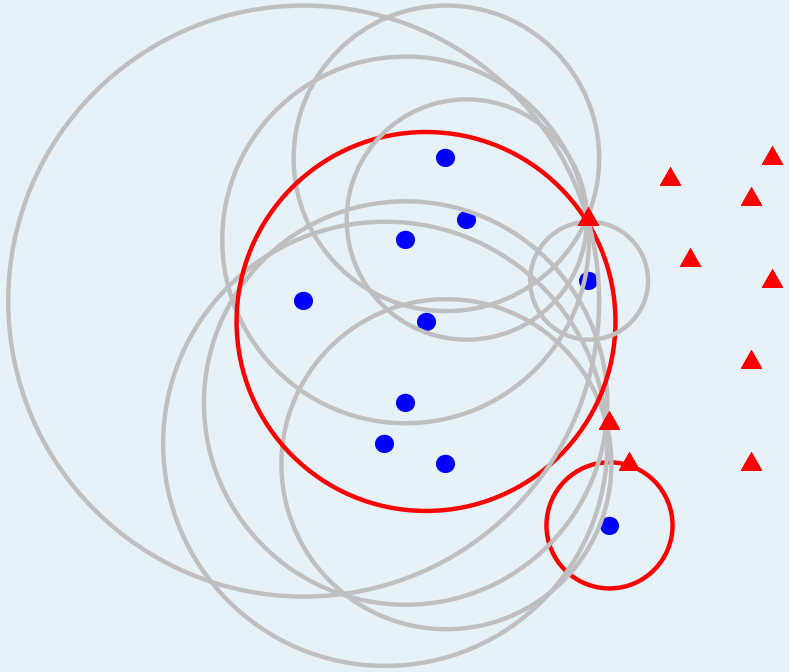


Complexity Reduction

- A **dominating set** of a graph is a set of vertices S such that every vertex is either a member of S or has an edge to it from a vertex in S.
- A **minimum** dominating set is one which has smallest cardinality.
- Denote the cardinality of a minimum dominating set by γ.
- Note that if we take a minimum dominating set of the CCCD, we still cover all the training observations, thus producing a reduced cover.



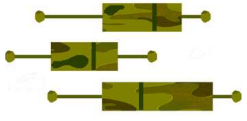
A Dominating Set



Note that in this case we obtain a large reduction in complexity and a “better” class cover.

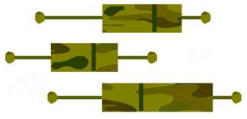
Choosing the Dominating Set

- Selecting a minimum dominating set is hard.
- A greedy algorithm works well to select a (nearly) minimum dominating set.
- The algorithm is fast (given that the interpoint distance matrix has already been calculated).
- A fast algorithm has been developed which allows the selection of a small dominating set with large data sizes.



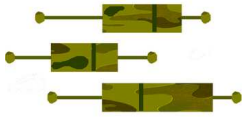
The Greedy Algorithm

- Select the ball that covers the most X points.
- While there are still points that are uncovered:
- Select the ball that covers the most points not yet covered.
- Return the balls selected.
- Notes
  - We do not restrict our selection of balls to those whose centers are not already covered.
  - Ties are broken arbitrarily (we can break them in favor of balls with a higher statistical depth (local density) if we choose).



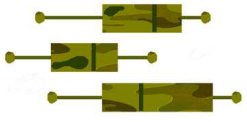
CCCD Classification

- The dominating set for the CCCD provides a set of centers and radii.
- Compute these for each class.
- Compute the minimum scaled distance between a new observation x and the centers ($d(x, x_i)/r_i$).
- The class of the minimum is the class assigned the new observation.
- The CCCD is like a reduced nearest neighbor classifier with a variable metric.



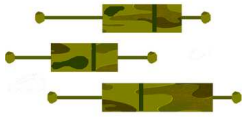
CCCD Extensions

- One can relax the constraint that the reduced set be a dominating set by allowing a small number of observations to not be covered. This can simply be done by stopping the greedy algorithm early.
- Similarly, one can relax the purity constraint, and allow a small number of other-class observations to be covered. Various algorithms are possible, such as only covering a Y observation if doing so covers "enough" new X observations.

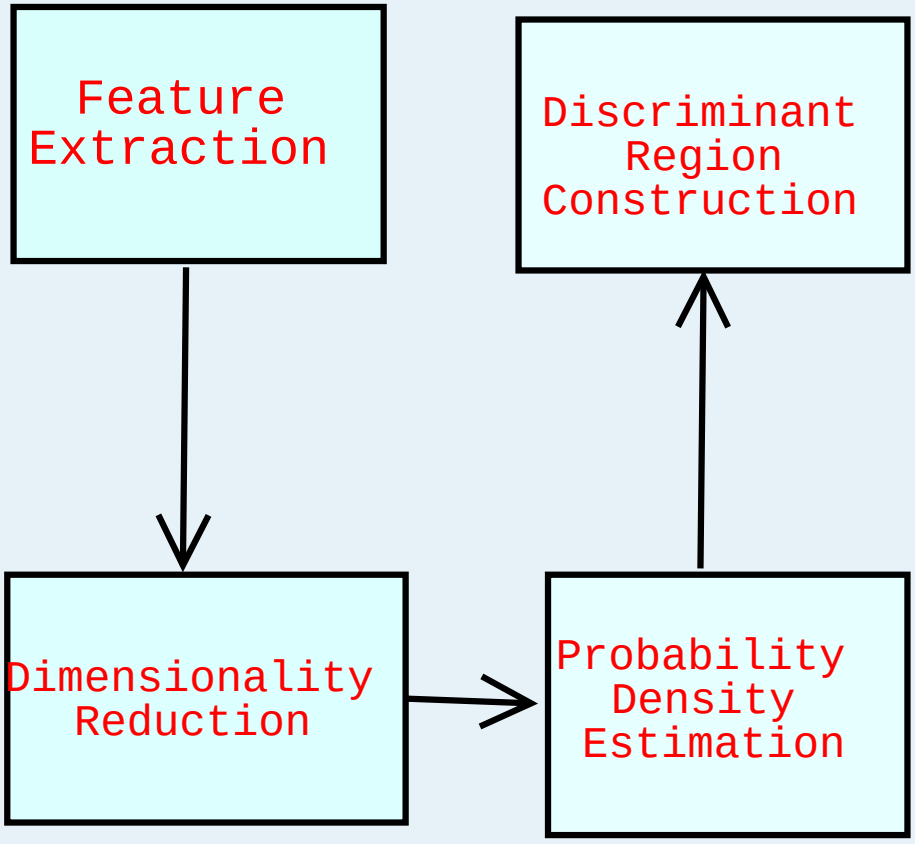


Putting It All Together

- We now give an example of a tool that puts all the steps described together.
- The tool was designed to operate on images and allow the user to construct classifiers to find certain types of objects within images.
- This was a “low level” image processing application.

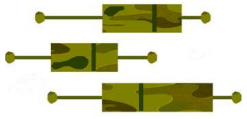


Advanced Distributed Region of Interest Tool (ADROIT)



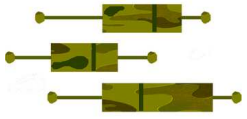
ADROIT Algorithms - I

- Feature Extraction
  - Wavelets, Sobel
  - Coefficient of Variation
  - Fractal Dimension, and Boundary Gated Fractal Dimension
- Dimensionality Reduction
  - Fisher Linear Discriminant
  - Iterated Fisher Linear Discriminant
  - Evolutionary Algorithm Methodology

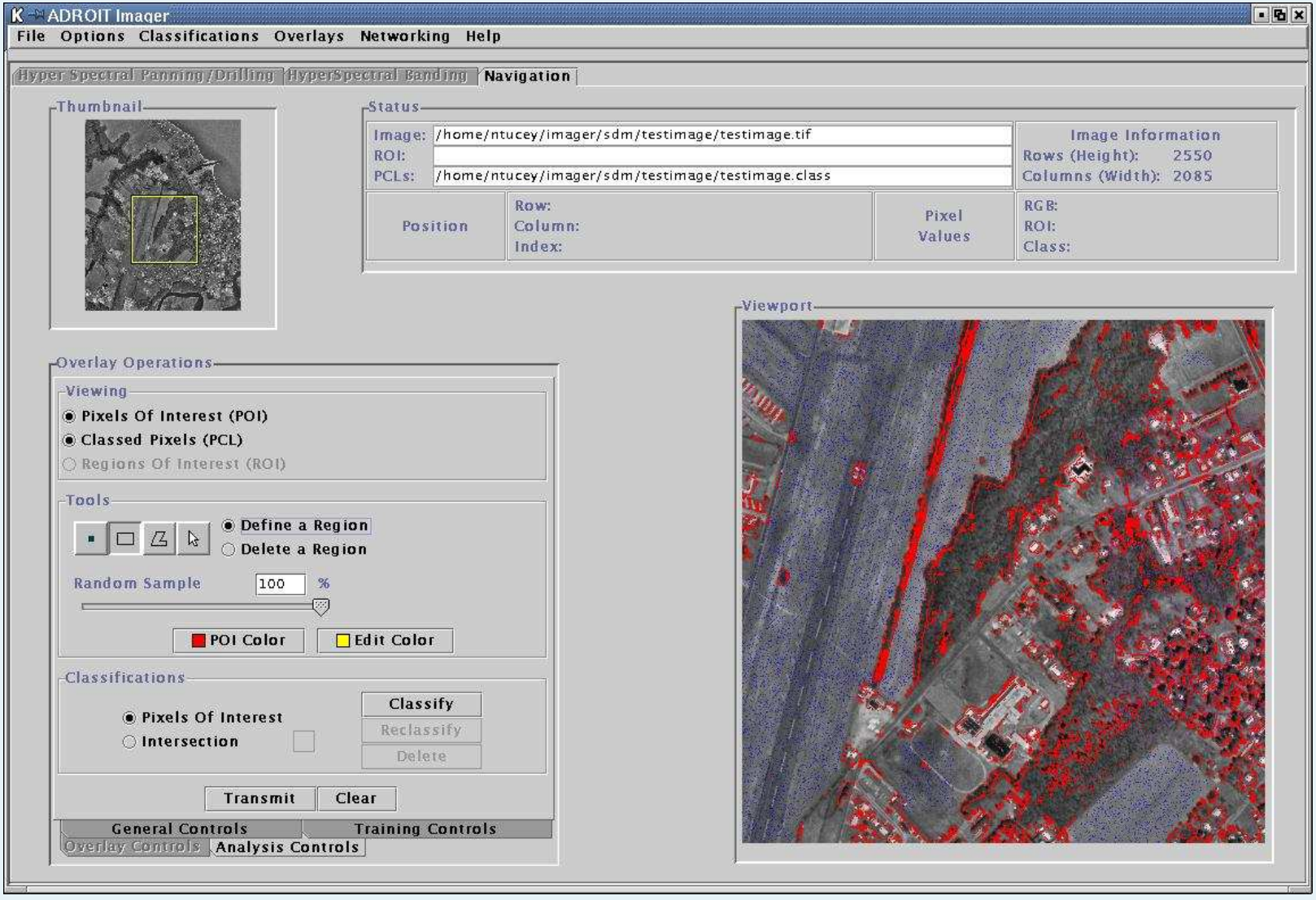


ADROIT Algorithms - II

- Probability Density Estimation
  - Finite Mixture
  - Adaptive Mixtures
- Classifier Construction
  - Likelihood Ratio (Bayes Rule)



ADROIT Analysis



ADROIT Multiclass