Extracting Information from Large Datasets

The Informatics side of Proteomics, Chemometrics, Metabolomics, Metabonomics, …

Brian T. Luke
ABCC

Outline

- Goal of the Study
- Source of the Data
- Preprocess the Dataset
- Construct a Classification Model
- Examination of the Model
- Verification of the Model
- Conclusions

Goal of the Study

- Various experimental techniques can generate a large volume of data (Microarray, MS, NMR,…).
- Applied mathematicians (statisticians, numerical analysts,…) can “slice and dice” the data beyond recognition.
- The goal is to find biologically/chemically relevant information from the data.

Analysis

Determine the goal of the analysis and apply the appropriate mathematical technique(s).

Numerical Analysis

"Find a mathematical model that separates one histological state from another"

Bioinformatic Analysis

"Find one or more biomarkers that separate one histological state from another"

Source of the Data

Microarray Experiments

Florescence intensity is proportional to the concentration of gene-specific mRNA. It is assumed that this is also proportional to the expression level of the particular gene.

NMR/Mass Spectra

Material obtained from particular cells – Metabolomics

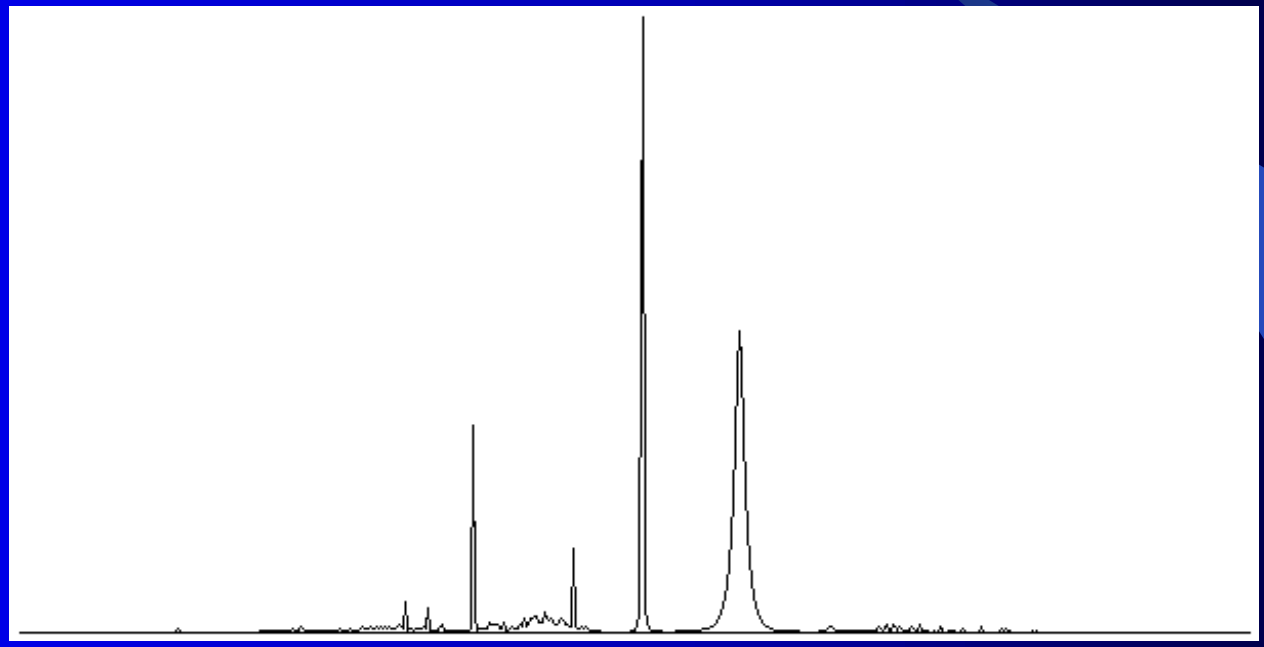
Spectra taken of plasma or urine – Metabonomics

Spectra produces a *fingerprint* of the sample.

Conditioning Data

- Make sure there is no missing data.
- Remove background intensities (NMR and MS)
- Remove effects of added markers, substrate or solvent, and any other "constant" features.

NMR Spectra



Preprocessing Dataset

In many cases, the original data can be reduced.

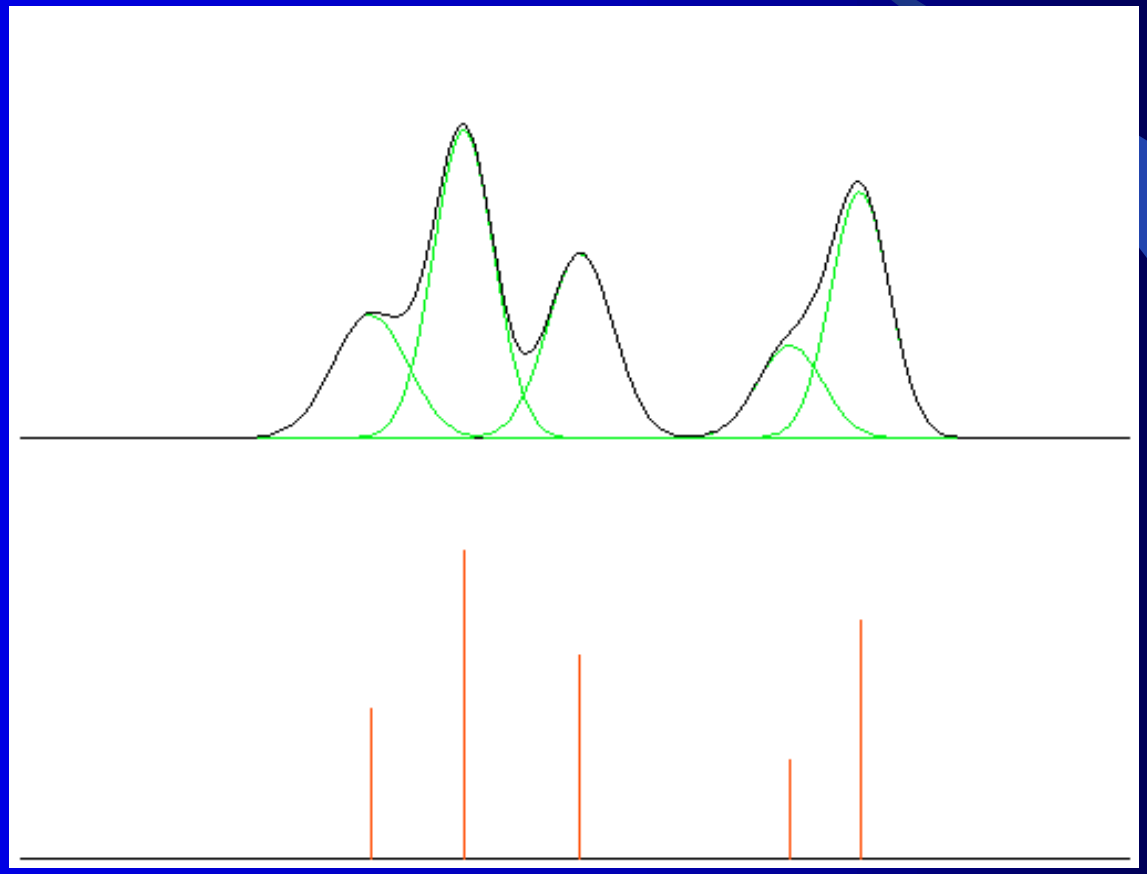
Microarray

Remove any gene with a constant expression level across different histologies.

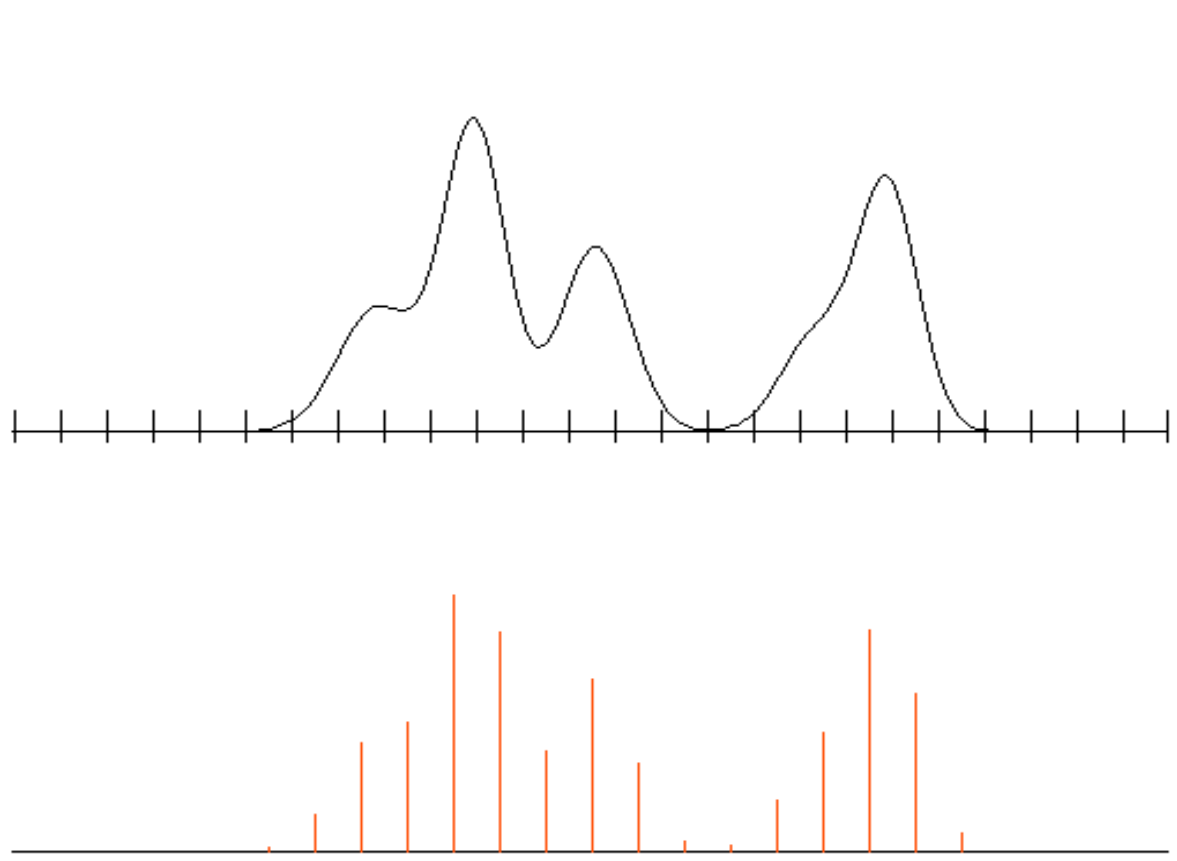
Spectra (NMR/MS)

Replace peaks by single lines or separate integrated intensities into bins (*binning*).

Spectral Fitting



Binning



Preprocessing Dataset

Normalize the data so that different runs can be combined and new results added.

Microarray

Use “control” spots to normalize intensities.

NMR and MS

Constant integrated intensity, constant maximum intensity,…

Search for Outliers

An outlier is a run/sample/cohort that, when taken as a whole, is not similar to any others.

- Normal Distribution
- Principal Component Analysis
- Sammon Map
- Closest Neighbor Plot

Statistical Method of Finding outliers

Statistics textbooks state that an outlier has a feature value that is more than two or three standard deviations from the mean.

The underlying assumptions are

- The number of samples is large.
- Each sample has a small number of features.
- The values for a feature over all samples forms a *normal* or *gaussian distribution*.

Statistical Method of Finding outliers

For most datasets, none of these assumptions hold.

- The number of samples is small so the mean and standard deviation are not well defined.
- There are a large number of features per sample so examining any one is not relevant.
- The hope is to find multi-modal features that can distinguish one *Class* of samples from another.

Principal Component Analysis

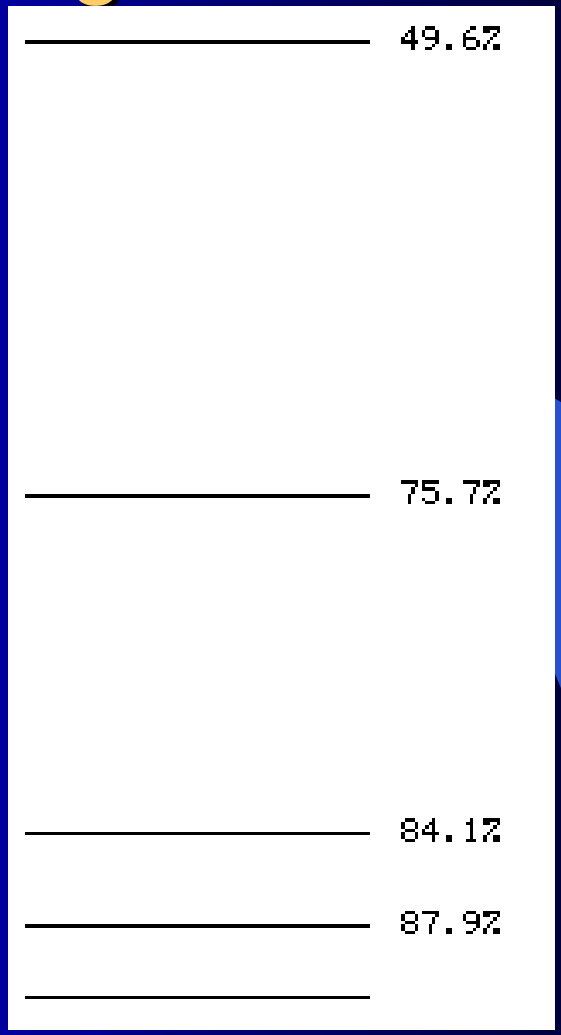
PCA is also known as Factor Analysis, or Karhunen-Loeve Transform, or Eigenanalysis and are the eigenvectors of the variance/covariance, or dispersion, matrix that correspond to the largest eigenvalues.

Really, Principal Components are linear combinations of features that do the best job of spreading out the data.

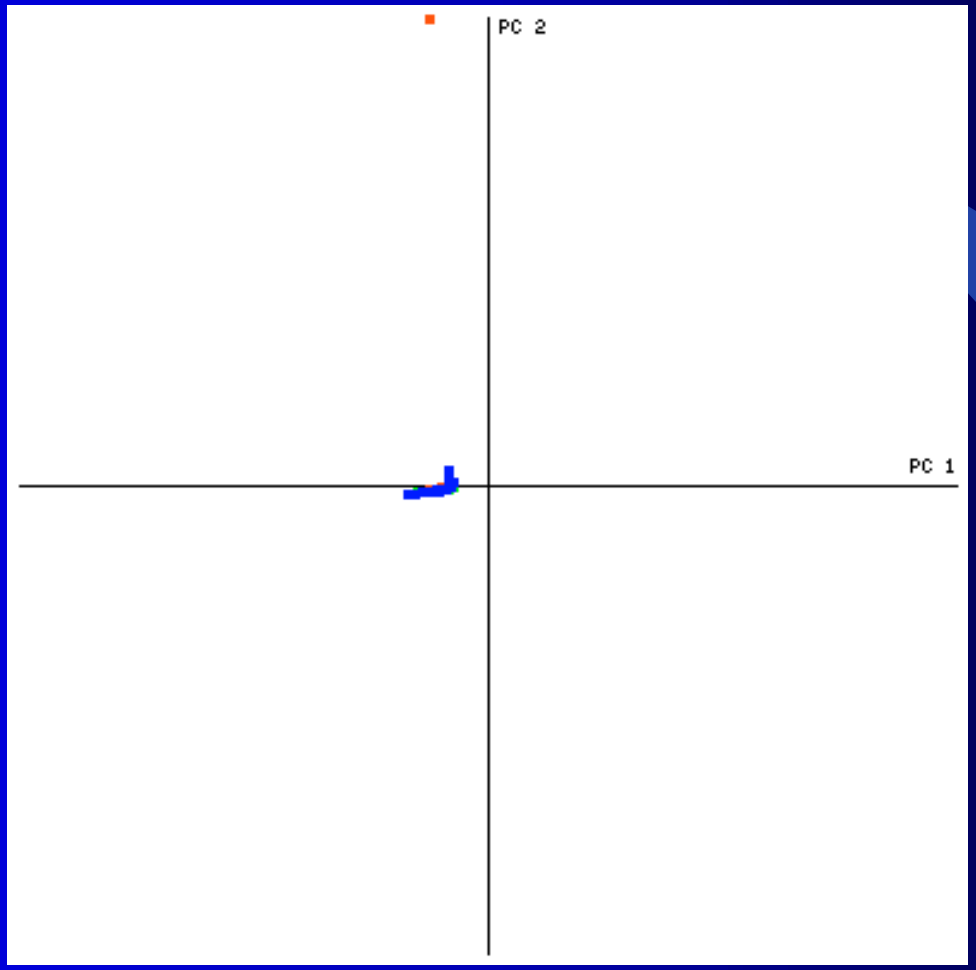
Principal Component Eigenvalues

Each eigenvalue divided by their sum represents the fraction of the total variation explained by that linear combination.

A plot of the first few Principal Components will hopefully identify an outlier.



First Two Principal Components

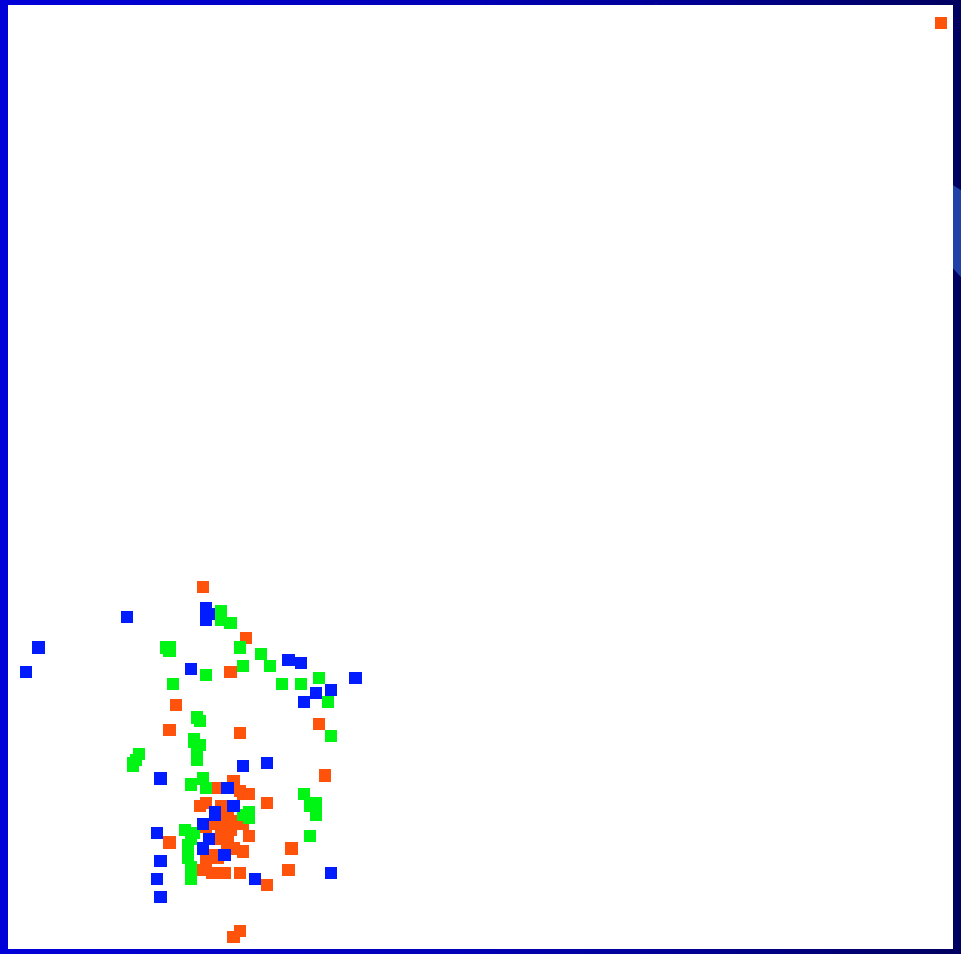


Sammon Map

A Sammon Map is a (non-unique) projection of samples from a high dimensional space (the number of features/samples) to a lower dimensional space such that the distances between all sample-pairs is preserved to the greatest possible extent.

This distance can be a Euclidean distance using all features.

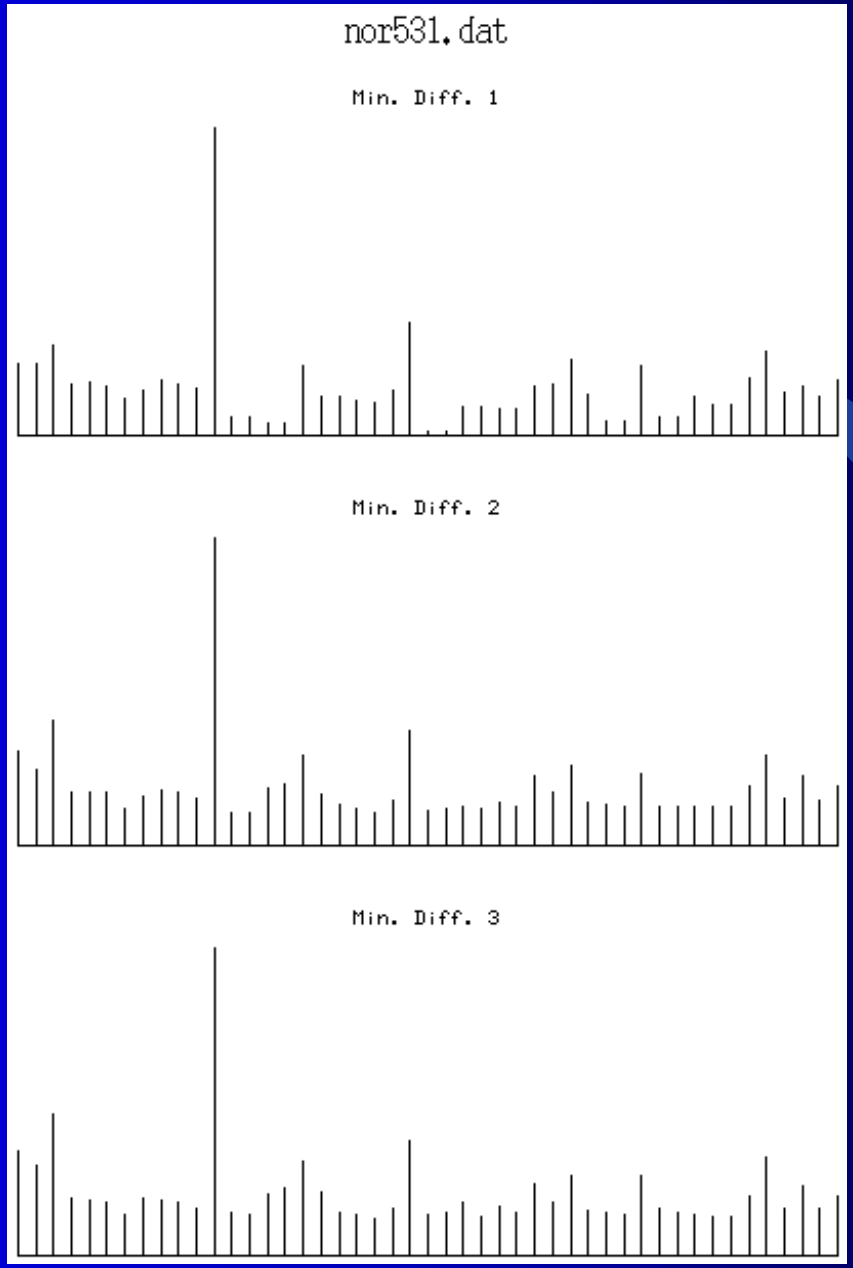
Sammon Map

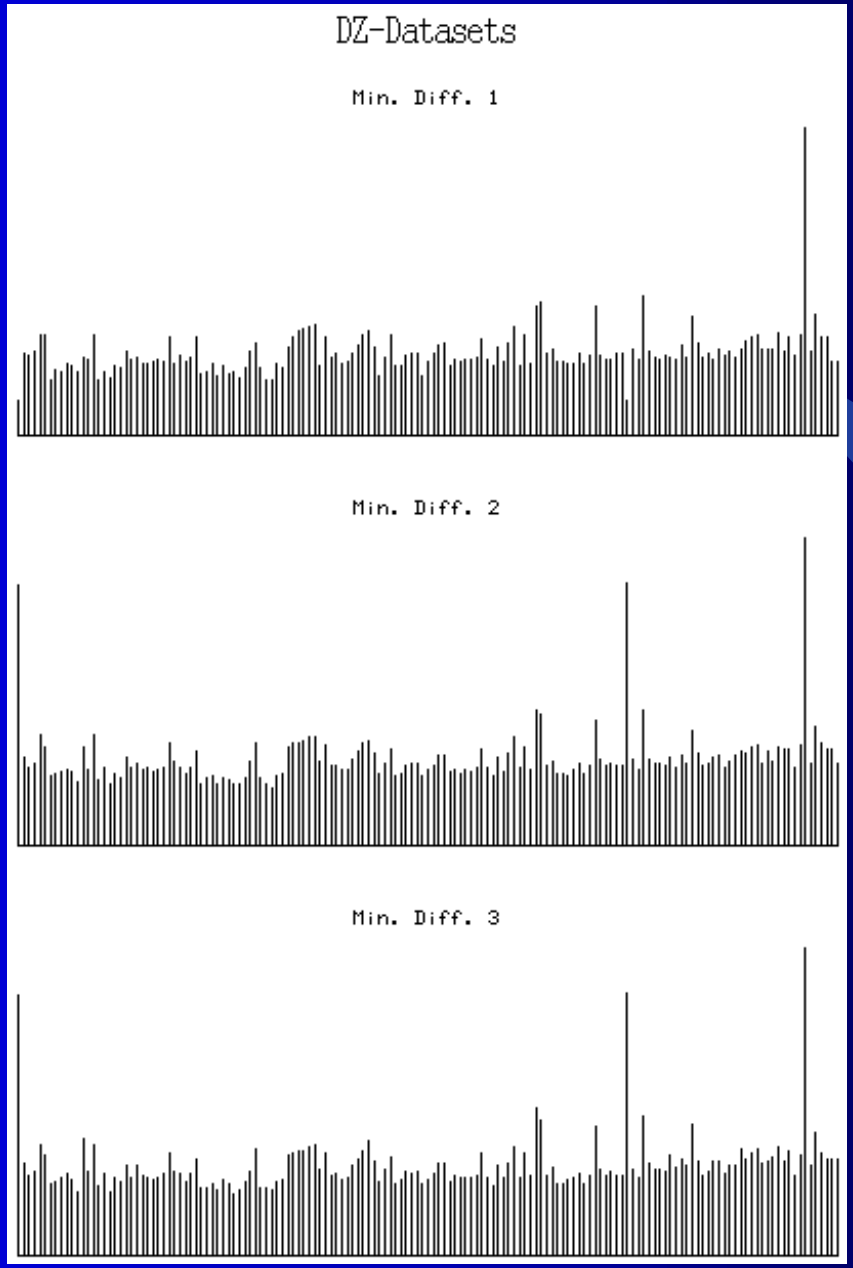


Closest Neighbor Plot

Plots of the distance to the first, second, and third closest neighbors can show if there are zero, one, or two outliers.

This procedure also identifies which sample is the outlier

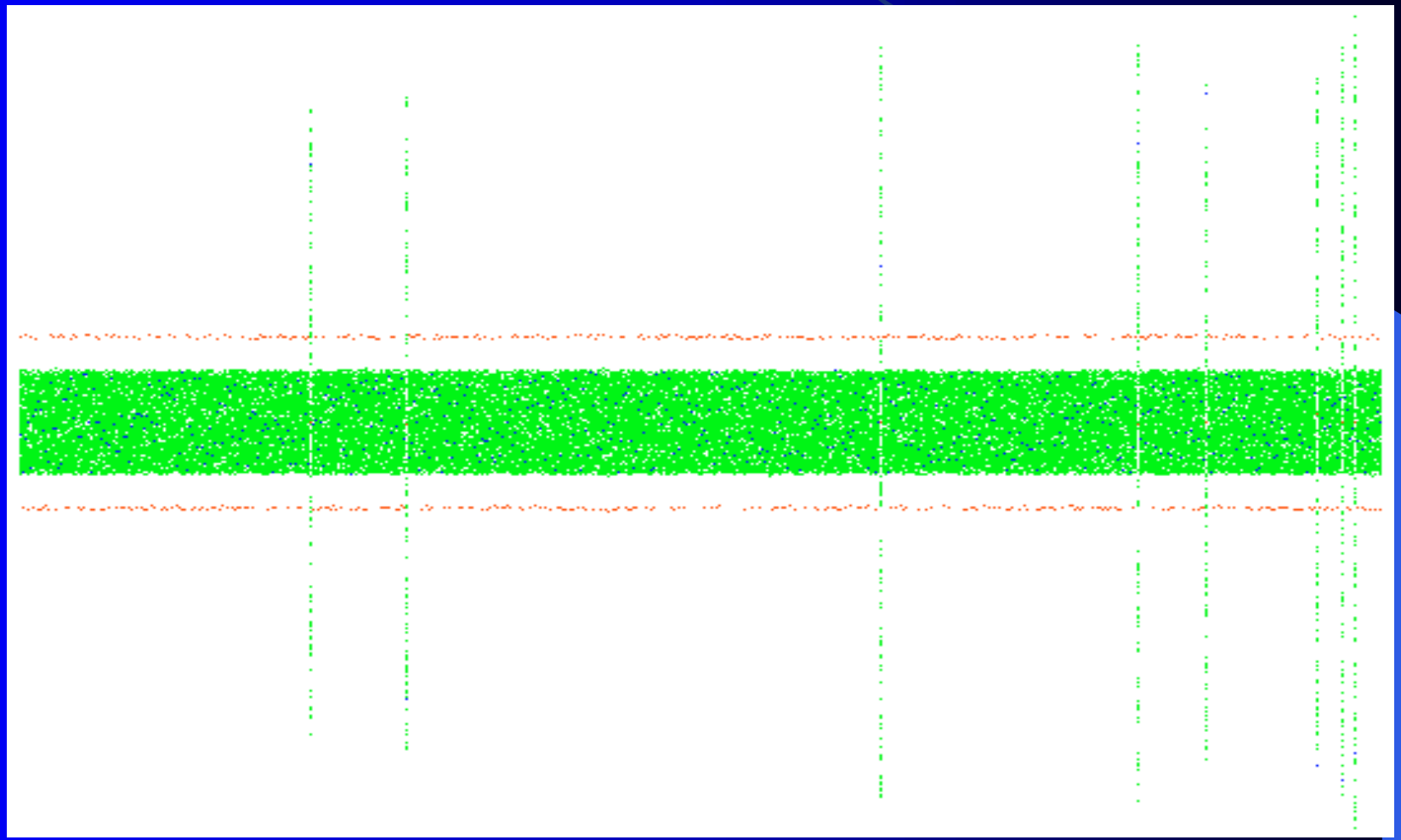

nor531.dat
Min. Diff. 1
Min. Diff. 2
Min. Diff. 3


DZ-Datasets
Min. Diff. 1
Min. Diff. 2
Min. Diff. 3

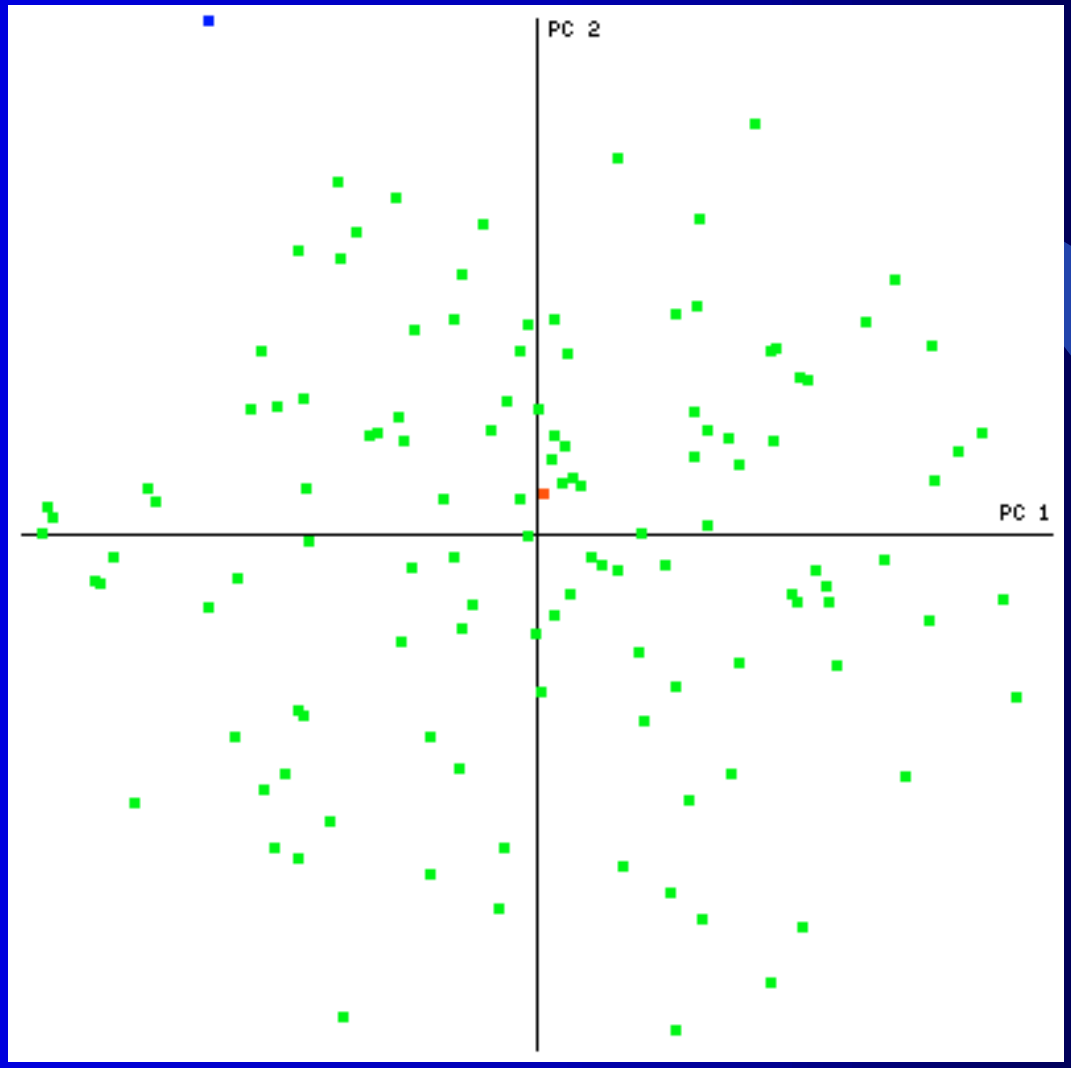
PCA Does Not Always Work

- The first few Principal Components are linear combinations of features that result in the largest variation (spread) in the data.
- Distances are not preserved unless all Principal Components are used.
- The large number of features allow an outlier to be spread across many features.

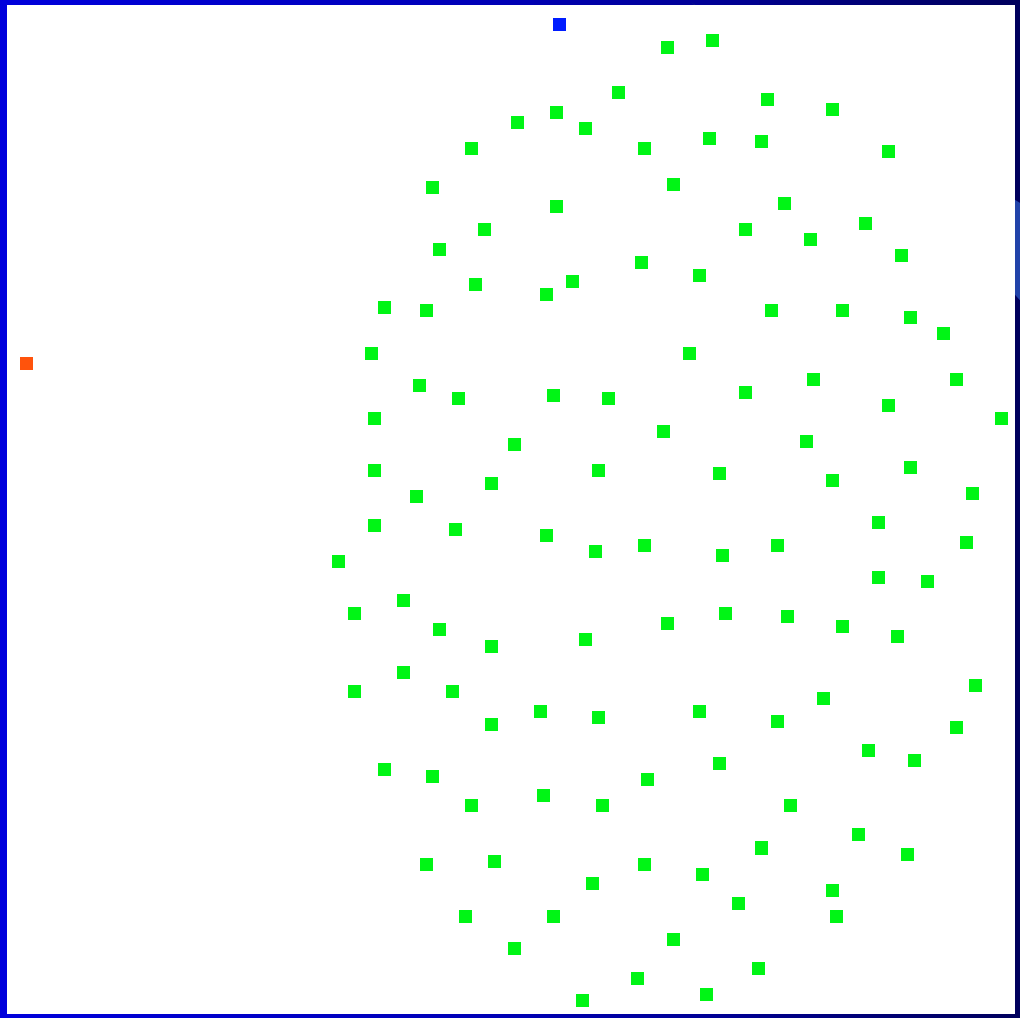
Variations in Small and Large Peaks/Bins



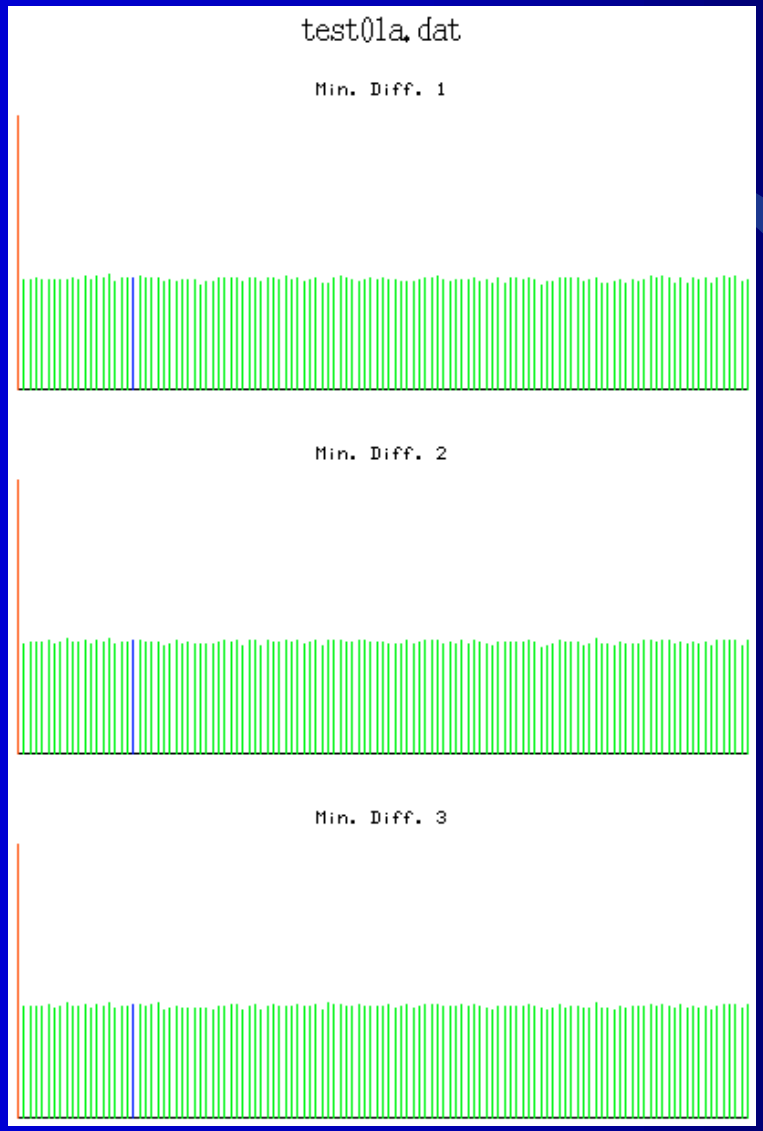
PCA Results



Sammon Map Results



Nearest Distances Results



Finding Outliers

- Spectra are dominated by a few large peaks which therefore have a large variance.
- PCA will only find an outlier if it has extreme values in the large peaks or if its total variance is concentrated in a single feature.
- Other techniques should also be used.

Construct a Classification Model

Notation:

N is the number of samples.

L is the number of features ($L>>N$).

J is the number of features used in a classification model ($J<<L$).

$\mathbf{X}$ is an $N \times L$ matrix containing the dataset.

$x_{i,j}$ is a particular feature (I=1,N; j=1,L).

X_i is the sum of features for Sample-I ($\Sigma_{j=1,L}\ x_{i,j}$)

Construct a Classification Model

- Use a Feature Selection method to efficiently choose sets of *J* features.
- Choose a Distance Metric to determine how similar/different two samples are.
- Construct a Classification Model using these distances.

Feature Selection Methods

Choosing the best J features from a group of L scales as $O(L^J)$, which means that trying all unique sets is impractical.

Feature Selection Methods include

- Heuristic searches
- Stochastic searches using a single solution
- Stochastic searches using a population of solutions

Heuristic Feature Selection Methods

Greedy Search

- Try each feature and find the one that classifies the samples the best.
- Keeping this "best" feature, sequentially try all others and find the best combination of two.
- Continue adding one feature at a time to the best set so far until all J features are located.

Scales as $L \times J$, so it is fast but the final feature set is sub-optimal.

Branch and Bound

Like Greedy only you keep *NBB* solutions instead of just the best.

- Try all unique sets of two features and store the best *NBB*.
- For each set, try all unique features and keep the best *NBB* sets of three.
- Continue process until you have the best *NBB* sets of *J*.

The computational time grows very rapidly with *NBB*, but this must be large to ensure that you find the best solution at the end.

Stochastic Feature Selection Using a Single Solution

These are guided random searches that allow a single feature set to travel through search space.

- Tabu Search
- Simulated Annealing
- Gibbs Sampling

Tabu Search

This procedure searches a “local region” of feature space and selects the best new solution (even if it is worse that the original).

The region is placed on the top of a tabu list and cannot be searched again.

The best feature set found during the search is the final solution.

For example, given a tabu list of length less than *J*, a randomly generated set of J features and a maximum number of local searches *MLS*, a search is

- Set *ILS*=0
- Increment *ILS* by 1
- Randomly select a position in the feature set list.
- If this position is on the tabu list return to Step 3.
- Try all other features in this position and select the new set that categorizes the samples the best.
- If this is the "best-to-date" feature set store it.
- Place this position at the top of the tabu list, moving all other positions down and dropping off the last one.
- If *ILS* is less than *MLS*, return to Step 2.

Simulated Annealing

Starting with a randomly generated feature set, FSold, and its quality, Qold, the search is as follows.

- Set T=Ti.
- Set *ISTEP*=0.
- Increment *ISTEP* by 1.
- Copy FSold to FSnew and randomly change one or two of its features.
- Calculate its quality, Qnew.
- If Qnew is greater than Qold, change FSold and Qold. Otherwise if $e^{(Qnew-Qold)/T}$ is greater than a random number in [0.0,1.0] change FSold and Qold.
- If *ISTEP* is less than *MAXSTEP* go to Step 3.
- If T is greater than Tf, reduce T slightly and go to Step 2.

In this process, T is an "effective temperature" and $e^{(Qnew-Qold)/T}$ is the Boltzmann acceptance probability. The temperature slowly changes from and initial (high) value Ti to a final (low) value Tf using a *cooling schedule*.

Simulated Annealing is as much an art as math since good choices need to be made for Ti, Tf, the cooling schedule and *MAXSTEP*, the number of Monte Carlo steps at each temperature.

Gibbs Sampling

Gibbs sampling is basically a combination of Tabu Search (with or without the tabu list) and Simulated Annealing.

As in the Tabu Search, a position in the feature set list is randomly chosen and all other features are tried in this position. The quality of the initial set, Qo, and all subsequent sets, Qi, are used to generate un-normalized acceptance probabilities, $UAPi=e^{Qi/T}$ (i=0,*N*-*J*).

They are then normalized by dividing by their sum to give NAPi. A random number between 0.0 and 1.0 is chosen and each NAPi is subtracted from this number.

As soon as this number becomes zero or negative this is the feature set that is selected to start the next search.

As with Simulated Annealing, the effective temperature starts at a high value and is slowly decreased to a final, small value.

With this procedure you still have to worry about the initial and final temperatures, the cooling schedule, and the number of searches at each temperature.

In addition, each feature set is compared to the "best-to-date" set, and only this best feature set is reported at the end.

Stochastic Feature Selection Using a Population of Solutions

Instead of using FSold(J) and Qold as before, we have FSold(J,NPOP) and Qold(NPOP) where NPOP is the size of the population.

Population based methods include

- Genetic Algorithms
- Evolutionary Programming
- Ant Colony Optimization
- Particle Swarm Optimization

Genetic Algorithms

Start with a randomly generated population of *J* feature sets FSold(J,NPOP) and qualities Qold(NPOP). Then

- Choose two feature sets using a probabilistic selection procedure. The probability of being selected is proportional to the feature set's quality.
- Create a complimentary pair of offspring using a *mating operator*, and determine their qualities.
- Have the higher quality offspring replace the feature set in the population with the lowest quality.
- If all feature sets in the population are not the same, return to Step 1.

This is only one example of possible search heuristics. Options can include

- generating a mating population
- using *mutation* and/or *maturation* operators
- placing offspring in a new population and, once full, building a new parent population using either a deterministic or probabilistic (μ,λ) or $(\mu+\lambda)$ selection procedure.

The most popular mating operator is the 1-point crossover.

(A,B,C,D,E) Parent 1
| cut point
(a,b,c,d,e) Parent 2

(A,B,c,d,e) Offspring 1
(a,b,C,D,E) Offspring 2

Other mating operators are possible.

Evolutionary Programming

This independently developed procedure uses asexual reproduction and a generational algorithm. Start with a population of feature sets FSold(J,NPOP) and their qualities Qold(NPOP).

- Set IGEN=0.
- Increment IGEN by 1.
- For each parent, copy the feature set to an offspring and then randomly change one or two of the features. Calculate its quality and place it in a new population.
- Combine the parent and offspring populations and deterministically or probabilistically choose the next generation's parents.
- If IGEN is less than MAXGEN, go to Step 2.

I have modified the offspring generation step slightly by including a *uniqueness operator* as a special maturation operator.

For each parent *m*

- Copy FSold(J,m) to FSnew(J,m).
- Randomly change one or two features in FSnew(J,m).
- Compare FSnew(J,m) with all FSold(J,m) and FSnew(J,n) [n=1,m-1]. If it is the same, return to Step 1.
- Determine Qnew(m) and place this offspring in the new population.

Ant Colony Optimization

Ant Colony Optimization uses the pheromone-trail model of ants going to and from a food source.

ACO is like EP only the mutation is not random and there is no uniqueness operator.

QFS(L) is an array that stores the quality of each feature and is originally set to a random value.

After a generation, the new parents are examined and each feature's QFS'(l) is increased by an amount proportional to the quality of the feature set.

$$QFS(l) = \lambda\ QFS(l) + QFS'(l)$$

$$0.0 < \lambda < 1.0$$

λ is the evaporation rate.

Starting with a randomly generated FSold(J,NPOP) and their qualities Qold(NPOP), set QFS(L) to random values.

- Examine the feature sets in each parent and update QFS'(L).
- Update QSF(L)
- Each parent produces an offspring FSnew(J,L) by copying their feature set and then changing one or to randomly selected features. The probability that feature *l* is selected is proportional to QFS(l). Calculate its quality and place it in a new population.
- Combine the parent and offspring populations and select the next generation's parent population
- If all parent feature sets are not the same, go to Step 1.

Particle Swarm Optimization

Particle Swarm Optimization (PSO) is modeled after independent particles whose motion is influenced by local and global attractors.

Each particle has a position vector FS(J,NPOP) and a velocity vector V(NPOP).

This is like EP with V(NPOP) representing the mutation operator, but each offspring replaces its own parent.

FS'(J,NPOP) is the best feature set found by each particle to date and FSg(J) is the best feature set found by any particle.

Initially load FS(J,NPOP) with random feature sets and V(J,NPOP) with random numbers. Store FS(J,NPOP) in FS'(J,NPOP) and Q(NPOP) in Q'(NPOP). Store the best FS'(J,NPOP) in FSg(J) and its quality in Qg.

- Mutate each particle: FS(J,n)=FS(J,n) + V(J,n)
- Construct the feature set by taking the nearest integer to each element in FS(J,n) and determine its quality Q(n). If Q(n)>Q'(n) update FS'(J,n) and Q'(n).
- If the best Q'(n)>Qg, update FSg(J) and Qg.
- Update the mutation vector of each particle using V(J,n)=V(J,n) + C×R(J)×(FS'(J,n)-FS(J,n)) + C' ×R'(J)×(FS'(J,n)-FS(J,n)) where R(J) and R' are random arrays in [0.0,1.0] and C and C' are constants.
- If all of the particles are not the same, return to Step 1.

Feature Selection Methods

Each method has its strengths and weaknesses.

I prefer EP because, except for an exhaustive search or Branch and Bound which take too long, it is the only method that can generate multiple models.

- Final population can be used to search for biologically-relevant markers.
- Final population can be used as the initial population for model updates. All other methods would have to start from scratch.

Distance Metrics

Many distance metrics are possible.

I generally use L_n-Norms.

$$L_n(i,k)=\{\sum_j |x_{i,j} - x_{k,j}|^n\}^{1/n}$$

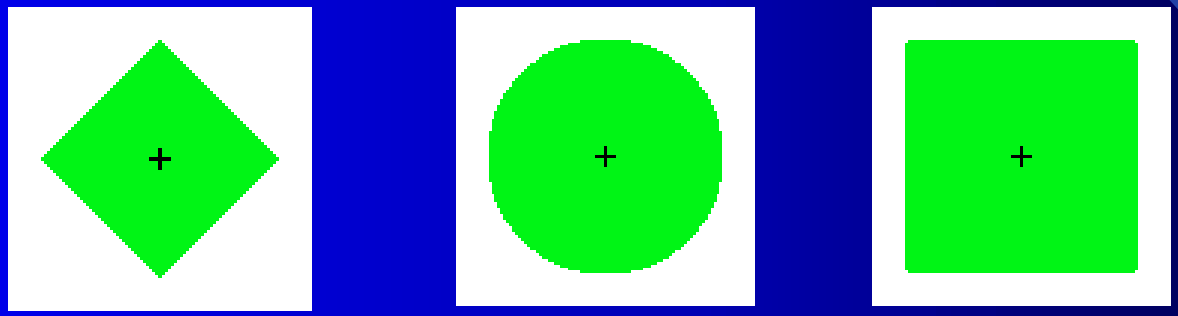
L_1 is called a Manhattan Distance

L_2 is the Euclidean Distance

L_∞ is the Chebyschev Distance

L_n ¡ Constant

n=1 n=2 n=□

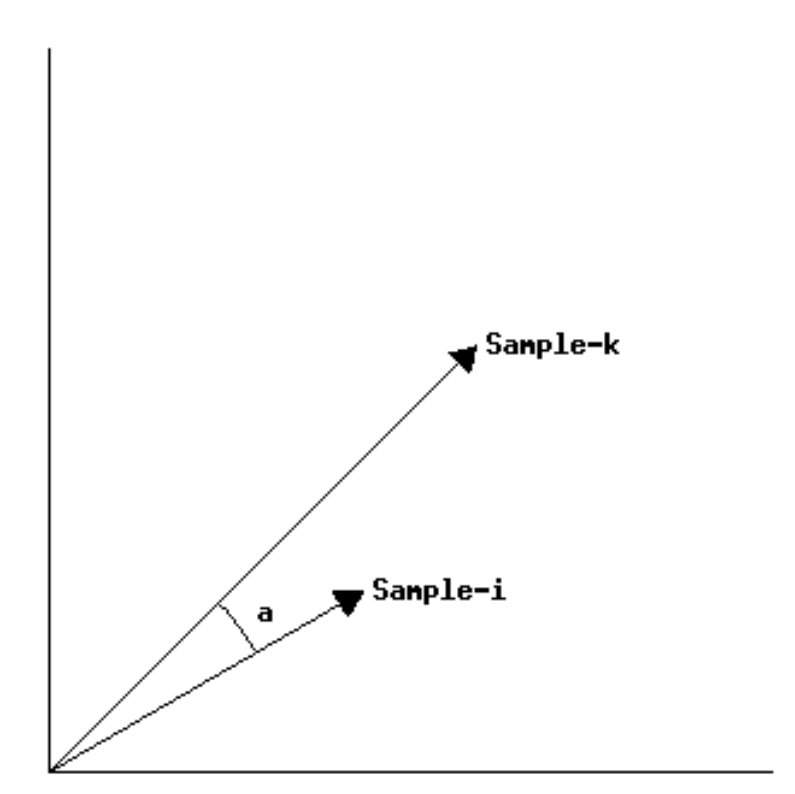


Sample Positions are also vectors.

Cos(a) = Similarity

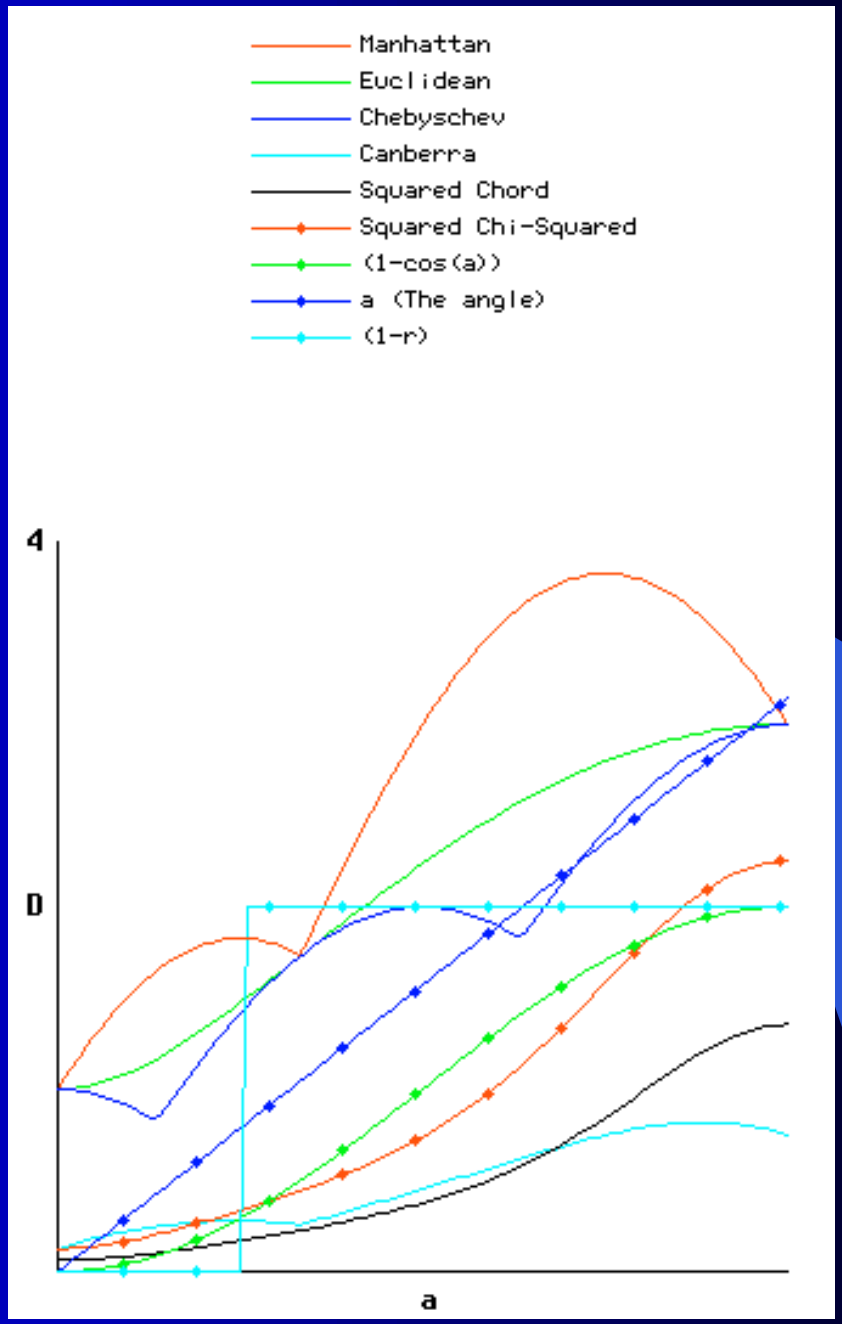
Dot Product
(normalized vectors)

Pearson's *r*
(standardized vectors)



S1 = (1.0, 0.0)

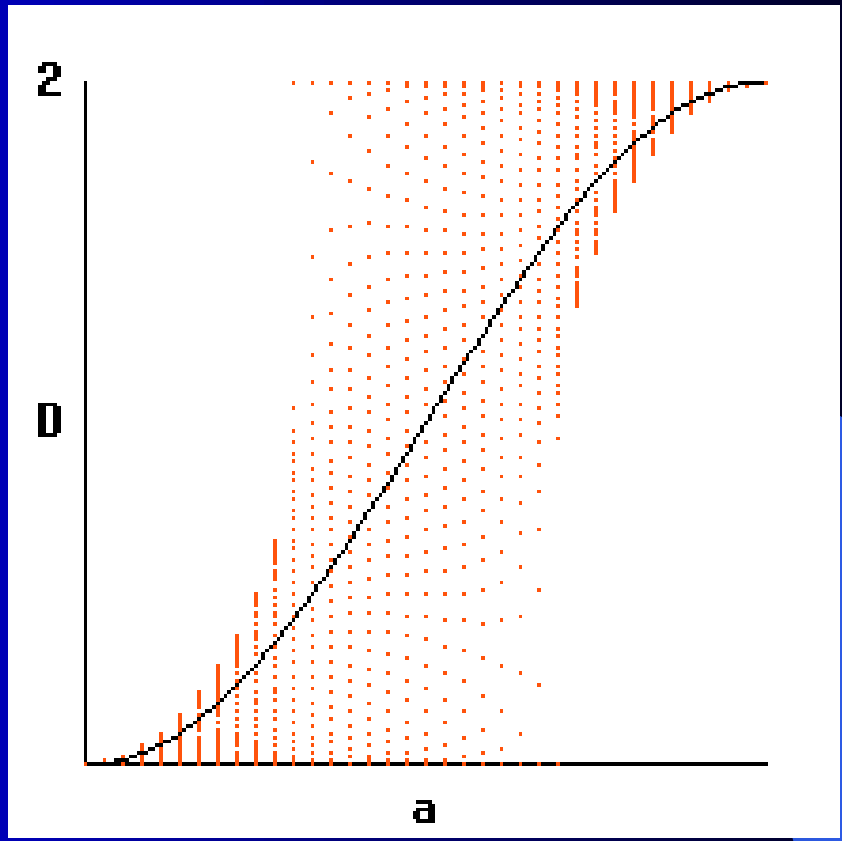
S2 = (X2, Y2)
X2 = 2 cos(a)
Y2 = 2 sin(a)



(1-cos(a)) black
(1-r) red

S1 = (0.0,0.0,1.0)

S2 = (X2,Y2,Z2)
X2 = sin(a) cos(p)
Y2 = sin(a) sin(p)
Z2 = cos(a)



Building a Classification Model

We now have a method of effectively choosing J features from the set of L and a means of measuring a distance in this J-dimensional sample space.

The classification model will be used on training samples whose Class is known. The major classification methods are

- K-Nearest Neighbors
- Clustering
- Neural Networks

Crisp versus Fuzzy Classifiers

If an unknown sample is contained in a group of 10 known samples. If 7 of the samples are Class 1 and the other 3 are Class 2, what is the classification of the unknown?

Crisp Classification: The unknown is Class 1 with 100% certainty.

Fuzzy Classification: The unknown is 70% Class 1 and 30% Class 2.

Maximum Likelihood is just a nice way of saying "winner take all" so this is a crisp classification and the unknown is 100% Class 1.

Beware of a crisp classification. Though the model gives you a definitive classification and may be right all of the time, the uncertainty in the assignment is unknown and can be quite large.

K-Nearest Neighbors

The known samples represent points in *J*-dimensional space.

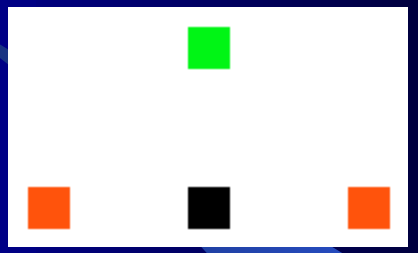
The unknown sample is then placed in this sample space.

KNN finds the *K* known samples that are closest to the unknown and uses their Classes to predict the Class of the unknown.

KNN classification is independent of the distances to these neighbors.

Should the fuzzy classification be the same in each case?

Case 1: 66.7% red, 33.3% green

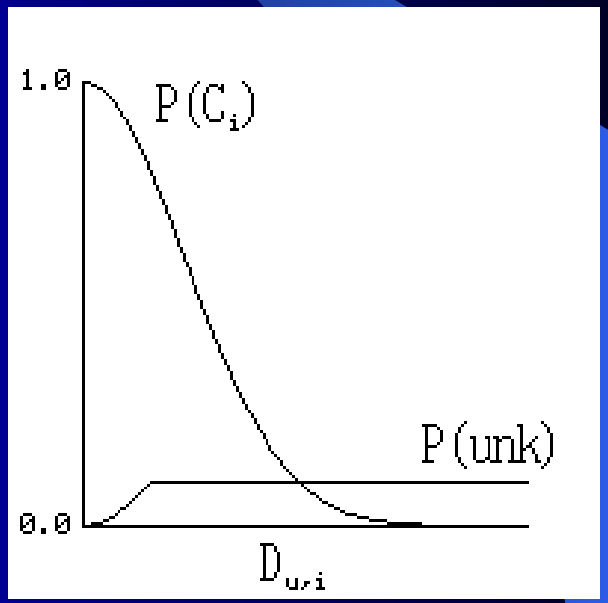


Case 2: >66.7% red, <33.3% green

Case 3: ???

There should be a distance dependence in the extent to which the unknown agrees with each neighbor.

DD-KNN: The extent to which the unknown is in the same Class as neighbor *I*, P(Ci), is a decreasing function of their separation. P(Ci) = f(Du,i)



This takes care of Cases 1 and 2, but not 3. Add an *unknown* Class.

P(unk) is constant until P(Ci) is large enough. P(unk) then monotonically Decreases to zero.

Classification by Clustering

The goal is to choose the feature set, distance metric, and clustering method such that the clusters are as homogeneous as possible.

The prediction of the unknown will then be as certain as possible.

Three types of clustering algorithms.

- Agglomerative Hierarchical Clustering
- Divisive Hierarchical Clustering
- Non-Hierarchical Clustering

Agglomerative Hierarchical Clustering

All samples start in their own cluster (singletons).

A rule is used to decide which two clusters are merged.

This continues until the desired number of clusters is obtained.

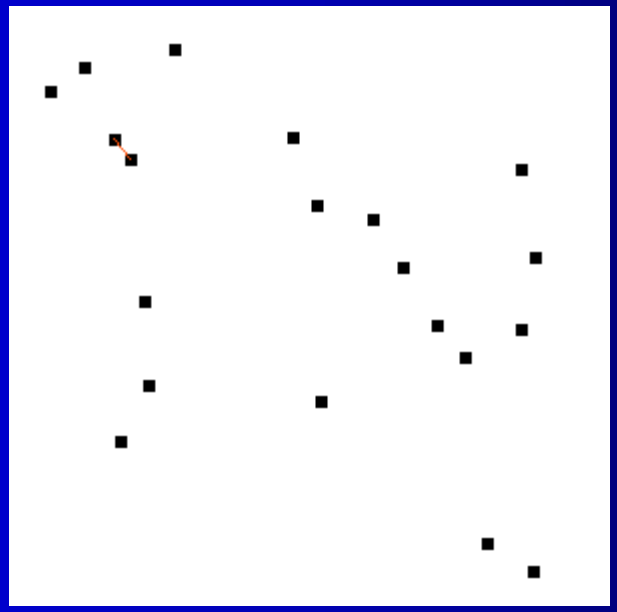
Common Agglomerative Hierarchical Clustering methods are

- Single Linkage Clustering
- Average Linkage Clustering
- Complete Linkage Clustering
- Ward's Method
- Jarvis-Patrick Clustering

Single Linkage Clustering

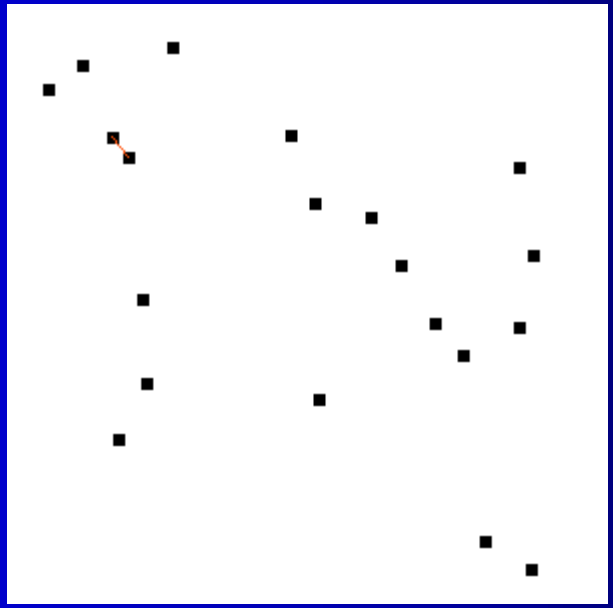
- Each pair of samples from different clusters is examined.
- The pair with the smallest distance is located.
- The clusters containing these samples are merged.

This allows a cluster to “snake” its way through the sample space.



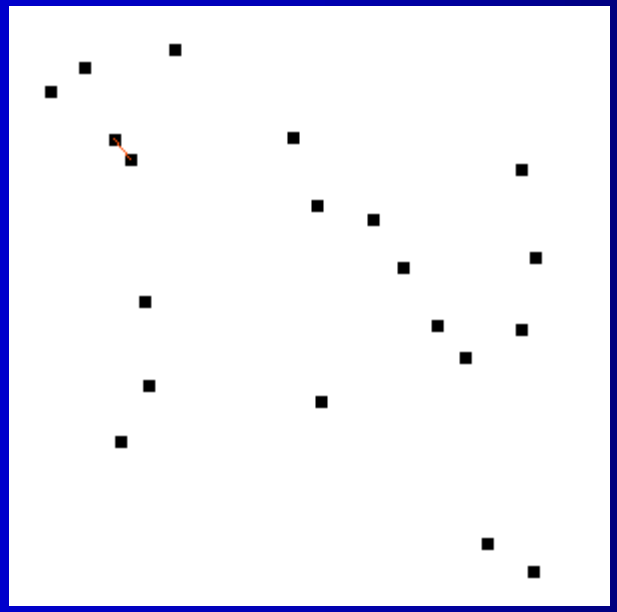
Average Linkage Clustering

- Every pair of *clusters* is examined and the average inter-cluster distance is calculated.
- The cluster-pair with the smallest average distance is merged.



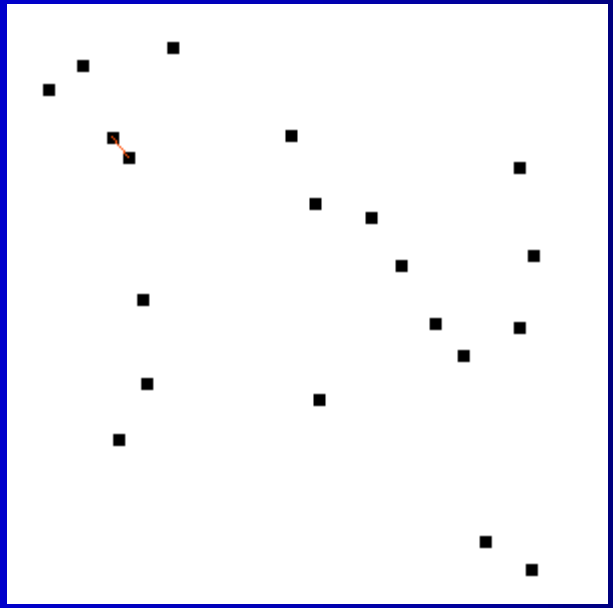
Complete Linkage Clustering

- Every pair of *clusters* is examined and the maximum inter-cluster distance is calculated.
- The cluster-pair with the smallest maximum distance is merged.



Ward's Method

- Every pair of *clusters* is temporarily merged.
- The centroid of the merged clusters is determined and the variance is calculated.
- The cluster-pair with the smallest combined variance is merged.

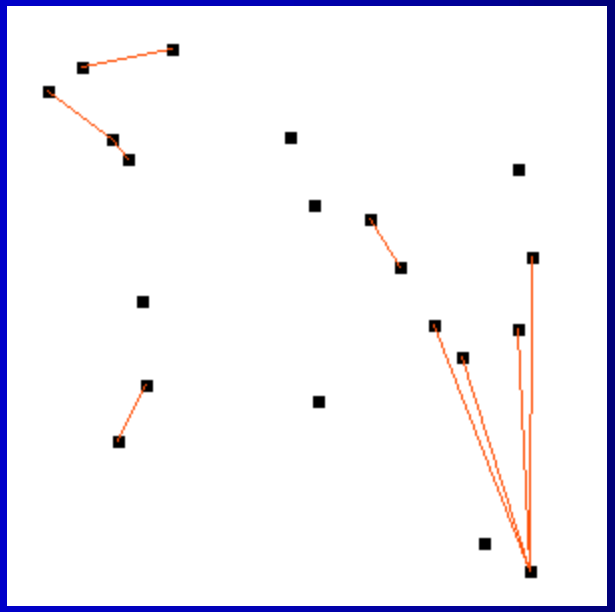


Jarvis-Patrick Clustering

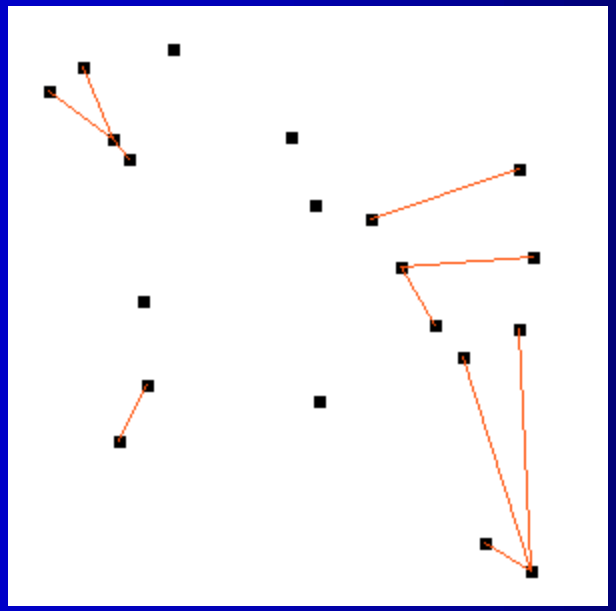
- A neighbor list of length *NJP* is calculated for each sample.
- Every pair of samples from different clusters is examined and the pair with the most common neighbor lists is selected.
- The clusters containing this pair are merged.

Like Single Linkage but distance-independent.

NJP=10

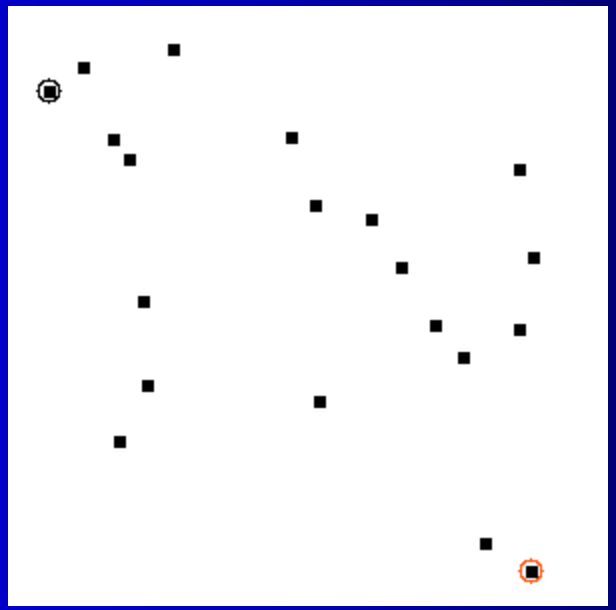


NJP=8

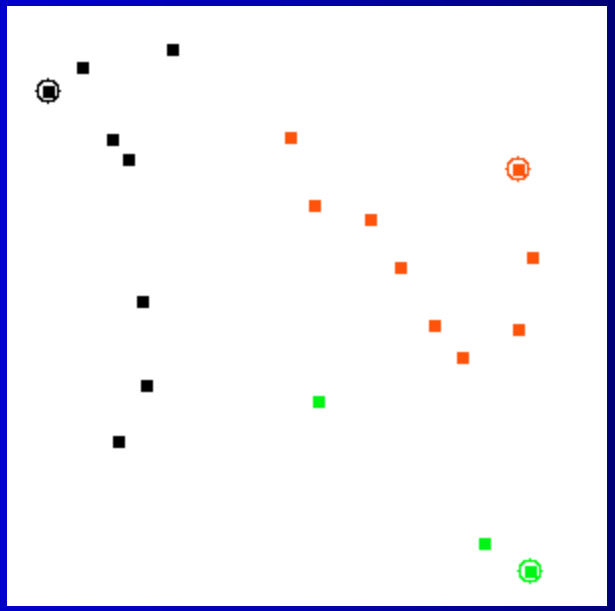


Divisive Hierarchical Clustering

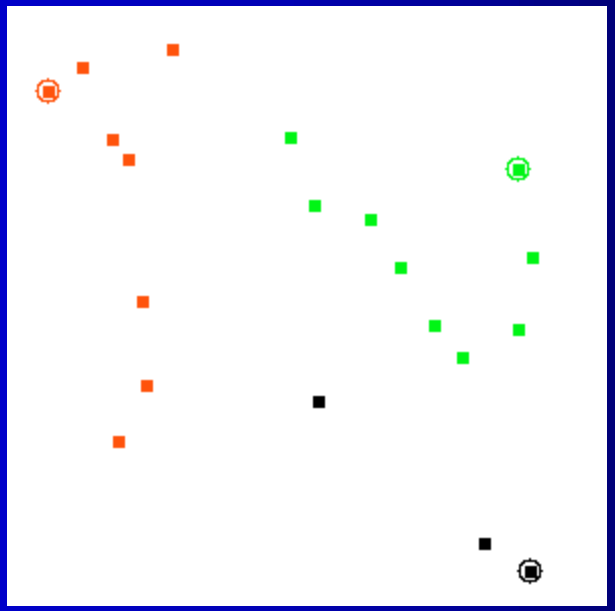
- All samples are placed in one cluster.
- In each cycle, each cluster is examined and the sample-pair with the largest within-cluster distance is found.
- The cluster with the largest within-cluster distance is split using this pair as seeds.
- All samples in this cluster are placed in the split cluster containing the closest seed.



Instead the K seeds (K>2) can be used to distribute all samples,



or a non-hierarchical procedure can be used to find the *K* point with the largest total distance.



Non-Hierarchical Clustering

The bet known Non-Hierarchical Clustering is K-Means Clustering (a.k.a. c-Means Clustering and others).

The goal is to select *K centroids* so that the sum of the distance-squared between each sample and their centroid is a minimum, though we are more interested in finding homogeneous clusters.

A true K-Means Clustering assigns each sample to a cluster, determines the position of the centroid, and calculates their summed distance-squared.

Each sample is moved to a new cluster, the new centroids are determined, and the new summed-distance squares.

This sample is placed into the cluster with the smallest sum of squares.

This process continues until no samples change clusters.

Finding the optimal cluster assignments is very difficult and other algorithms have been proposed (H-Means Clustering, J-Means Clustering, Variable Neighborhood Search, etc.).

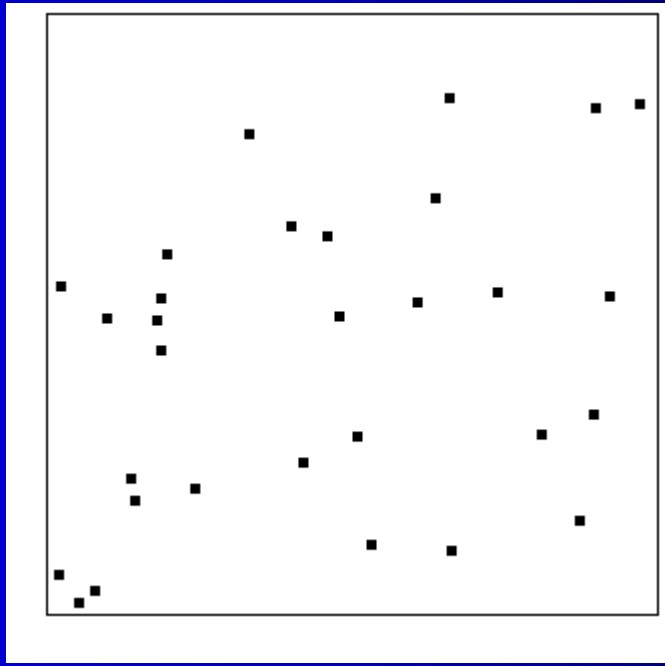
In K-Means Clustering, the final set of clusters depend on

- The initial distribution of samples amongst clusters.
- The ordering of the samples.

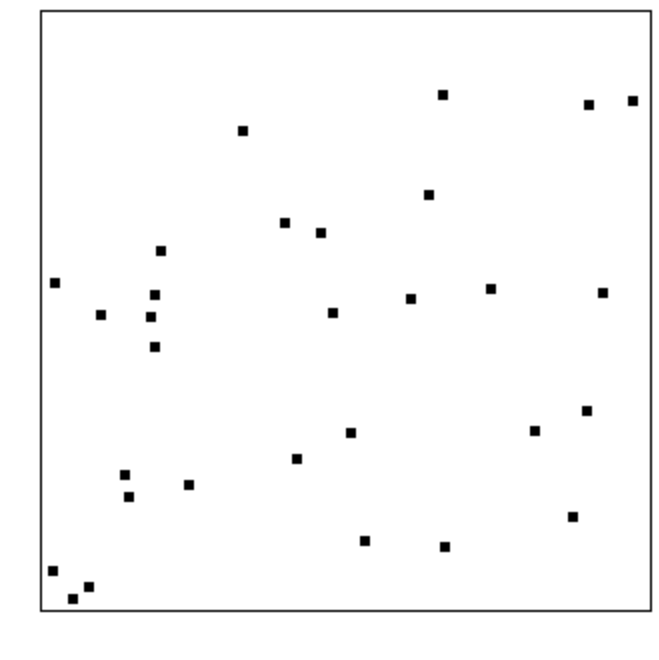
There are K^N ways to initially distribute the points amongst the clusters and $N!$ orders of the points, so trying all is impossible.

I use the following procedure.

- Choose K samples as initial centroids.
- Assign all samples to the nearest centroid.
- Use the assigned points to determine the new centroids.
- Re-assign the samples and continue the process until no samples change clusters.



The resulting clusters depends upon which *K* samples are chosen as initial centroids.



This procedure is independent of the sample order and finding the best initial set of *K* centroids (samples) is a much easier problem.

I use Evolutionary Programming to search for the best set of *K* samples to use as the initial centroids.

If the number of samples is large, the centroid will be close enough to a sample point that the iteration is not necessary. The *K* samples are then known as *medoids*.

PAM (Partitioning Around Medoids) searches through all unique sets of *K* samples.

CLARA (Clustering LARge Applications) and CLARANS (Clustering Large Applications based on RANdomized Search) speed up the search by using a series of reduced-dimensional searches.

Comparison of Clustering Methods

In general, each clustering method can generate a different partitioning of the samples.

Simple Linkage and Jarvis-Patrick Clustering are basically 1-Nearest Neighbor methods. They do not generally form neat clusters, but they are the only methods which guarantee that “close” samples will end up in the same cluster.

Neural Networks

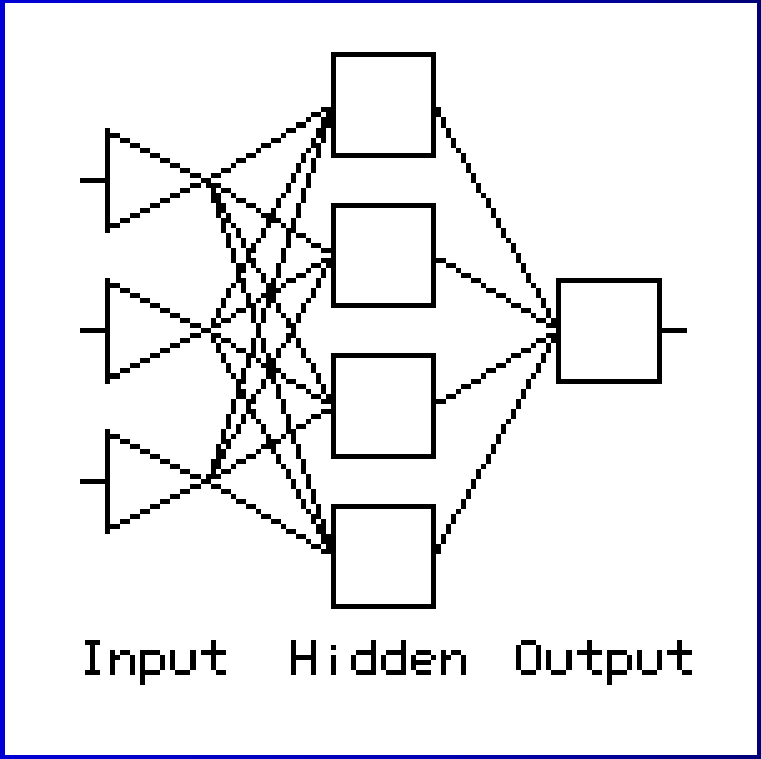
Neural networks are algorithms that learn from a set of training samples. The hope is that the trained algorithm will cause an unknown sample that is similar to a training sample to produce approximately the same result.

There are two training methods.

- Supervised
- Unsupervised

Supervised Learning Algorithm

Feedforward Backpropagation Multilayer Neural Network

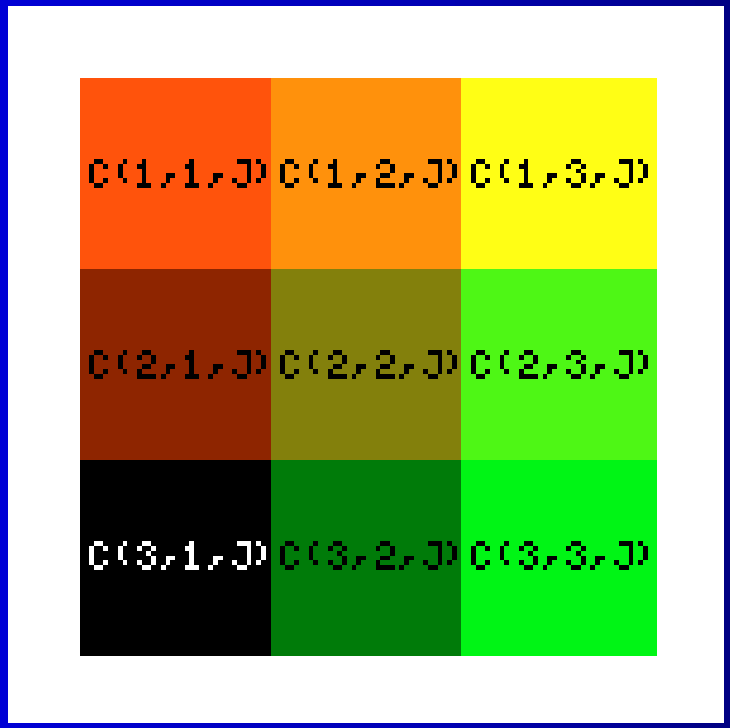


In a classification problem, this network can be used by assigning diseased samples a value of 0.0 and non-diseased samples a value of 1.0.

Once trained, an unknown sample is put in and its response can be used in either a crisp or fuzzy classification.

Unsupervised Learning Algorithm

Self-Organizing Map (SOM) or
Kohonen Map.



Each node has an associated array C(i,j,*J*) where the first two give its location and the third its *J*-dimensional coordinates.

A training sample is assigned to the node with the closest coordinates and the coordinates of all nodes are moved towards this sample.

This continues though the training samples and as the training proceeds the size of the step and nodes affected decrease.

At the end, the step size is small and only the assigned node and the adjacent neighbors have their coordinates moved.

A SOM is therefore similar to a K-Means Clustering with the centroids plotted using a Sammon Map.

A SOM suffers from the same problems.

- The final results depend upon the initial, random coordinates of each node.
- The final results depend upon the ordering of the training samples.
- Two closely-spaced samples are not guaranteed to end up on the same node.

Examination of the Model

Has a good model been produced or has it numerically done a good job on a small number of samples?

Robustness: Jackknife or Bootstrap

Cluster Statistics: Average Silhouette Width, Kelley Analysis

Fuzzy Classifier: Receiver Operating Characteristic

Robustness

A Jackknife Analysis is simply a *leave-one-out cross-validation*. For small sample sizes this may be all that you can do, but the standard deviation of the quality is not well defined.

An n^{th}-Order Bootstrap Analysis yields more tests. Remove n samples, build model on the (N-n) remaining and test those removed. Repeat this process a large number of times.

Cluster Statistics

The Average Silhouette Width (ASW) is a measure of the extent each sample should be in their assigned cluster.

For sample *i*, A(i) is the average distance to all other samples in its cluster and B(i) is the smallest average distance to samples in another cluster.

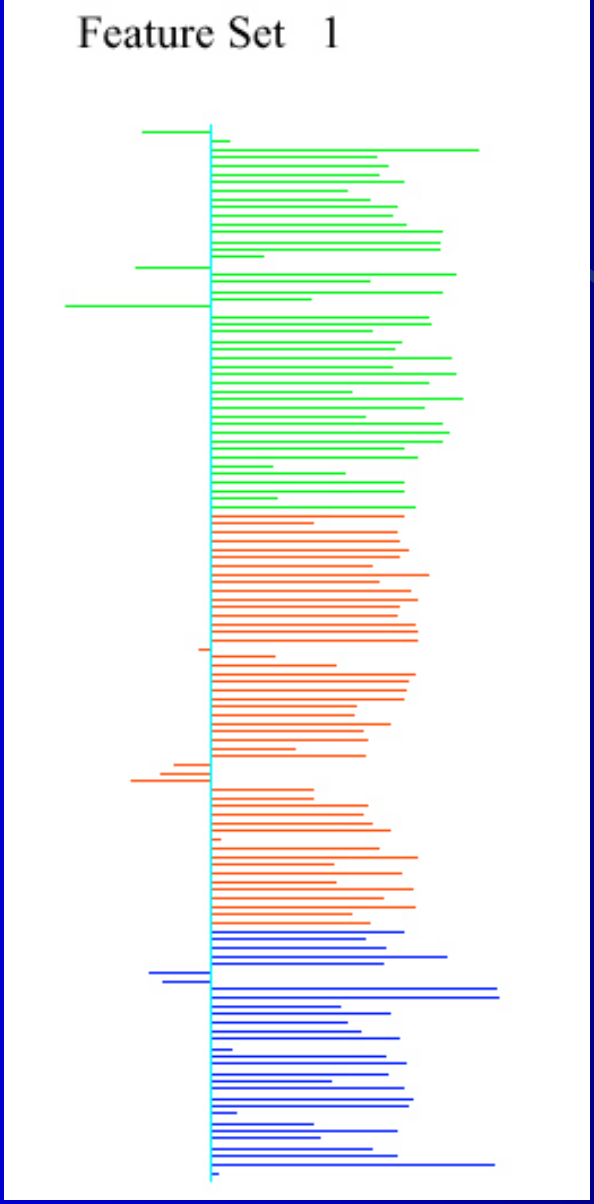
$$SW(i) = [B(i) - A(i)]/MAX[A(i),B(i)]$$

If SW(i) is near 1.0 the sample is well clustered.

If SW(i) is near 0.0 the sample is between clusters.

If SW(i) is negative it is probably in the wrong cluster.

ASW is the average of SW(i) over all samples.


Feature Set 1

A Kelley Analysis is a measure of the compactness of the clusters and uses the *average spread* (AS) of clusters containing more than one sample (non-singletons).

SPi is the average separation between all samples in cluster *i*.

The average spread of *K* cluster with *Kns* non-singletons is the sum of the spreads divided by *Kns*.

$$AS(K) = \ddagger \; SPi / Kns$$

ASn(K) is AS(K) normalized to lie between 1 and *N*-1.

$$ASn(K) = [(N-2)(AS(K)-ASmin) / (ASmax-ASmin)]+1$$

The Kelley Penalty Function is

$$P(K) = ASn(K) + K$$

and the goal is to minimize P(K).

At this point the clusters are dense and compact.

Using Fuzzy Classifications

Since a fuzzy classifier yields the probability that a sample is in a given state, a threshold value (T) is used determine if it is or isn't in that state.

Sensitivity: True positive fraction (TPF).

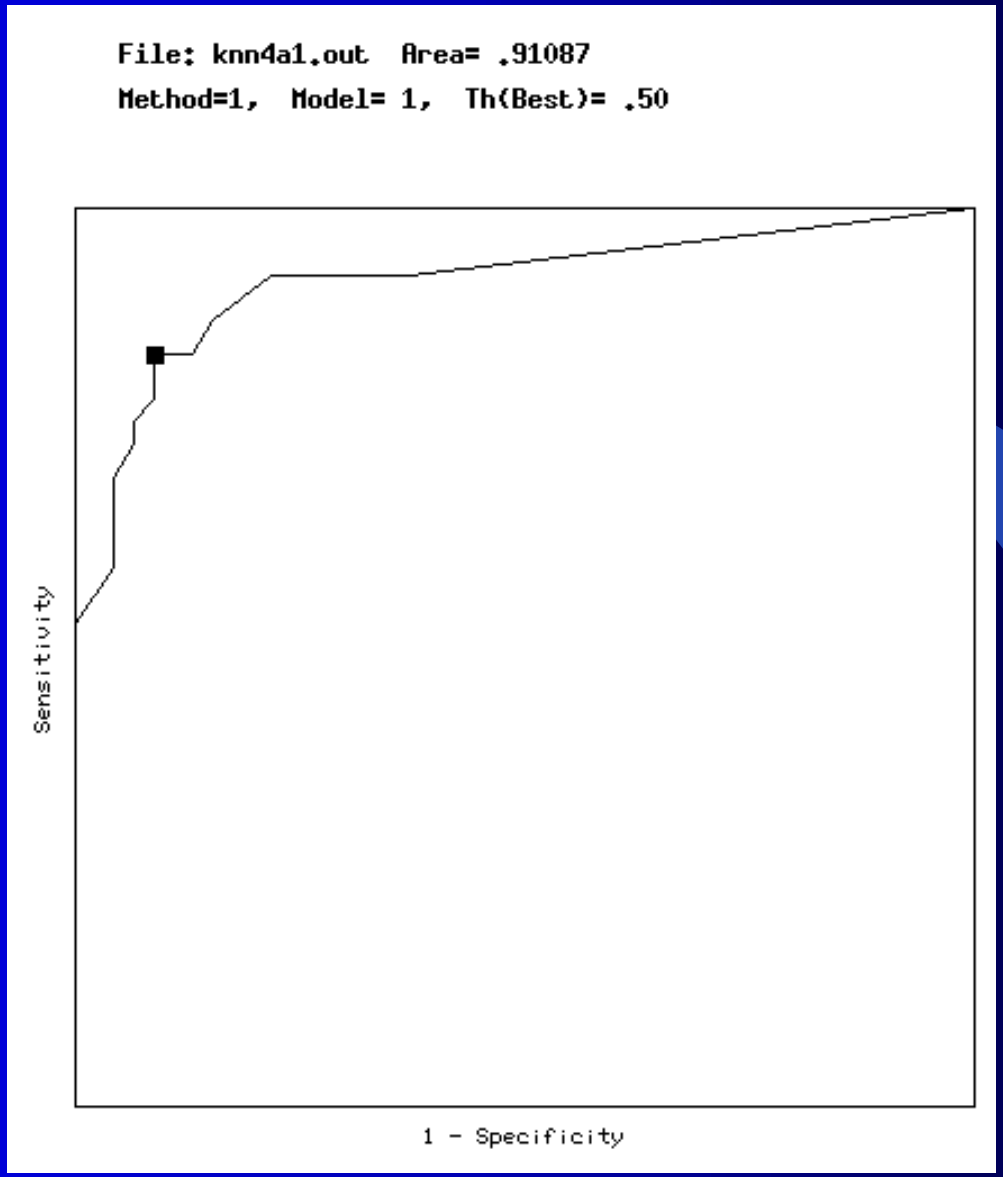
Specificity: True negative fraction (TNF).

(1-specificity): False positive fraction (FPF).

Both the sensitivity and specificity are functions of *T*.

A Receiver Operating Characteristic (ROC) Analysis graphically displays the quality of a diagnosis as a function of *T*. As *T* decreases from 1.0, the sensitivity increases and the specificity decreases.

The goal is to find *T* such that the sensitivity and specificity are maximized.


File: knn4a1.out Area= .91087
Method=1, Model= 1, Th(Best)= .50
Sensitivity
1 - Specificity

The area under a ROC curve is a measure of the average quality of the model for all values of *T*.

Because the sample size is usually small, this is not a smooth (monotonic) curve and the area may be of limited value.

Verification of the Model

We are still left with the problem of determining whether a model is a valid classifier, or if its simply a good *numerical* procedure that separates one class from another.

Since the number of features far exceeds the number of samples the latter is likely.

Tests of robustness and cluster quality can help.

The best model yields a biological basis for the classification.

Microarray: Likely

Metabolomics: Possible

Metabonomics: Unlikely

Mass Spectra: Impossible

In the real world, the only option is to produce multiple, quantitatively good models and blind-test them on a large number of subjects.

As the number of tests increases, so does the confidence in the models.

Conclusions

- Many ways to generate classification models.
- Most models are probably nothing more than good *numerical* separations of a small number of samples.
- Experimental and computational scientists must work together to ensure that each choice makes physical sense and the verification/analysis of the models are sufficient.

The scientists and staff of the ABCC are here to help.

Thank you for your attention.

lukeb@ncifcrf.gov

www.abcc.ncifcrf.gov