

Unsupervised Learning

Supervised Learning: Learn to predict y from \mathbf{x} from examples of (\mathbf{x}, y) . Performance is measured by error rate.

Unsupervised Learning: Learn a representation from exs. of \mathbf{x} . Learn how the exs. are similar to/different from each other. No standard way to measure performance.

- Clustering. Group the data maximizing similarity.
 - Component Analysis. Extract the most descriptive features.
 - Association Rules. Find common combinations of attributes.
-

k -Means Algorithm

Goal: Find k centers $\mathbf{c}_1, \dots, \mathbf{c}_k$ to minimize:

$$\sum_i \min_j d(\mathbf{x}_i, \mathbf{c}_j)$$

where d is a distance measure (typically Euclidean).

Choose k that makes a big decrease.

k -Means

Repeatedly

Initialize centers $\mathbf{c}_1, \dots, \mathbf{c}_k$ to k examples.

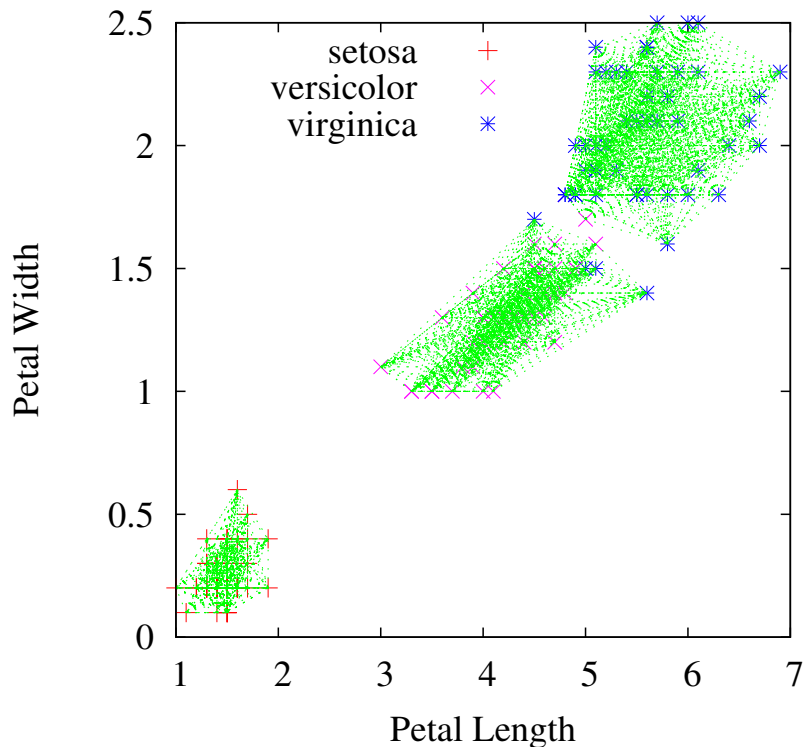
Repeat until convergence

For each example \mathbf{x}_i , find its closest center.

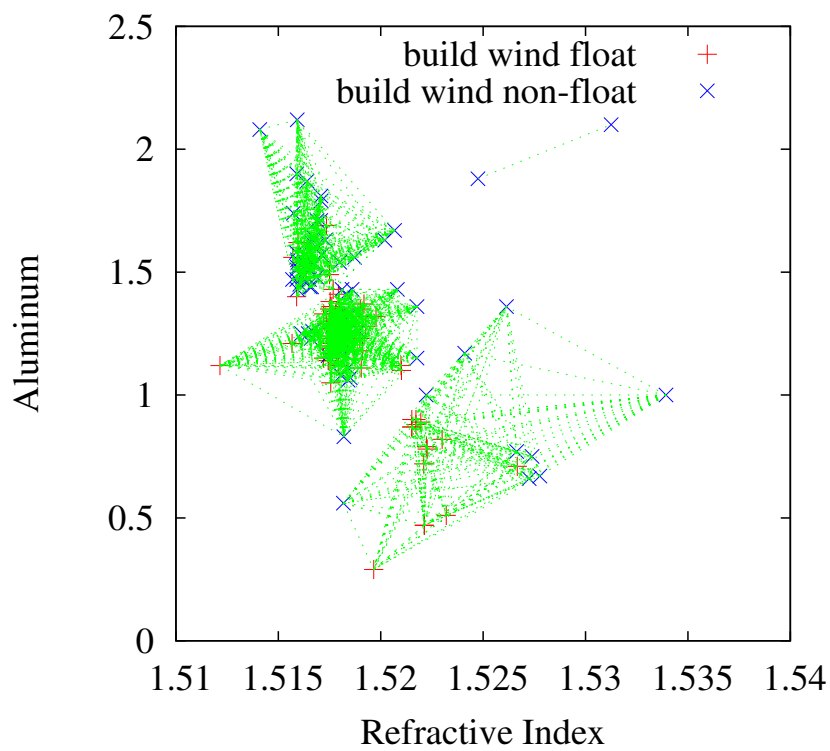
Set each center to the mean of its examples.

Return the best centers found

3 Clusters in Iris 2-Attribute Dataset



4 Clusters in Glass 2-Attribute 2-Class Dataset



Expectation-Maximization

Goal: Find a (local) maximum-likelihood estimate of unknown parameters Z governed by a probability distribution.

Restated: Over a set of examples, find a hypothesis h that maximizes:

$$\sum_i \Pr(\mathbf{x}_i, \mathbf{z}_i \mid h)$$

where each \mathbf{x}_i is known and each \mathbf{z}_i is unknown. We require:

- \mathbf{z}_i can be predicted from h .
- h can be predicted from \mathbf{z}_i .

EM Algorithm

EM

Repeatedly (in case of local maxima)

Initialize hypothesis h

Repeat until convergence

// Expectation Step

For each example \mathbf{x}_i ,

$\mathbf{z}_i \leftarrow$ predictions from h .

// Maximization Step

$h \leftarrow$ hypothesis assuming \mathbf{z}_i

Return the best hypothesis found

Learning Mixtures of k Normal Distributions

Suppose we assume the examples are in k clusters, each corresponding to a normal distribution.

Each distribution is defined by a mean vector \mathbf{u}_j (the means of the attributes) and a covariance matrix Σ_j .

For each example \mathbf{x}_i , we can predict:

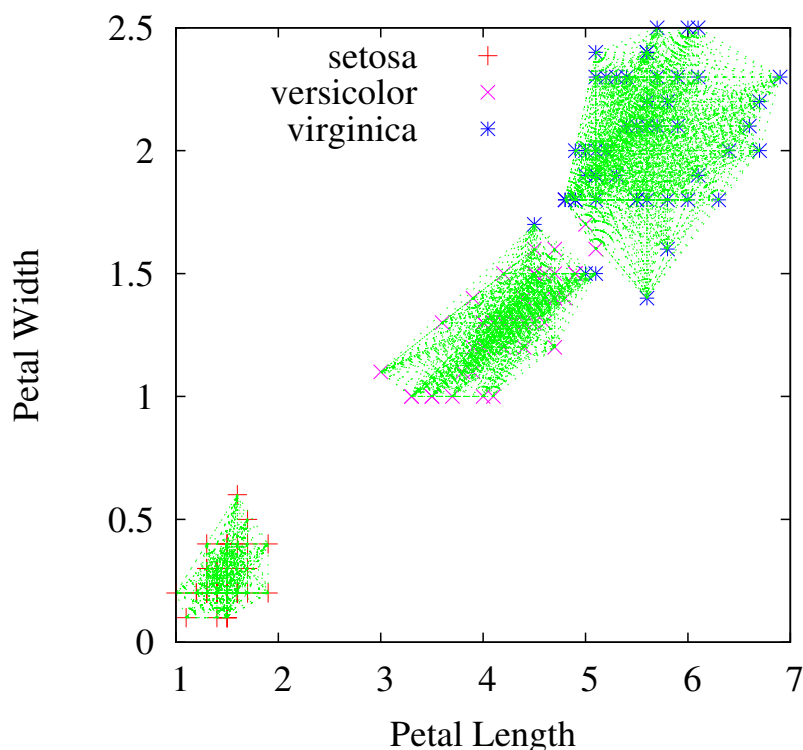
$$z_{ij} = \Pr(\mathbf{x}_i \mid \mathbf{u}_j, \Sigma_j)$$

the probability that \mathbf{x}_i belongs to cluster j .

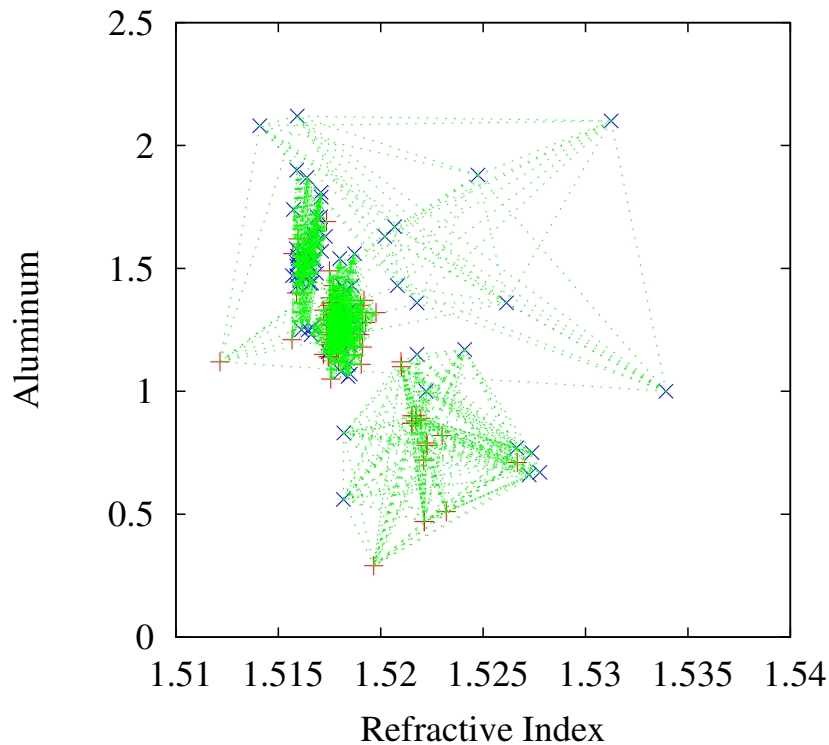
Given \mathbf{z}_i , we can predict means by (similarly covariances):

$$\mathbf{u}_j = \left(\sum_i z_{ij} \mathbf{x}_i \right) / \left(\sum_i z_{ij} \right)$$

3 Clusters in Simplified Iris Dataset



4 Clusters in Simplified Glass Dataset



Hierarchical Clustering

In hierarchical clustering, the examples are leaves in a tree and each internal node is a cluster.

This algorithm is agglomerative (bottom-up) hierarchical clustering.

Hierarchical-Clustering

For each example \mathbf{x}_i , $C_i \leftarrow \{\mathbf{x}_i\}$

Repeat until one cluster remains

 Find the two nearest clusters C_{i1} and C_{i2}

 Merge C_{i1} and C_{i2} into one cluster

Return the cluster hierarchy

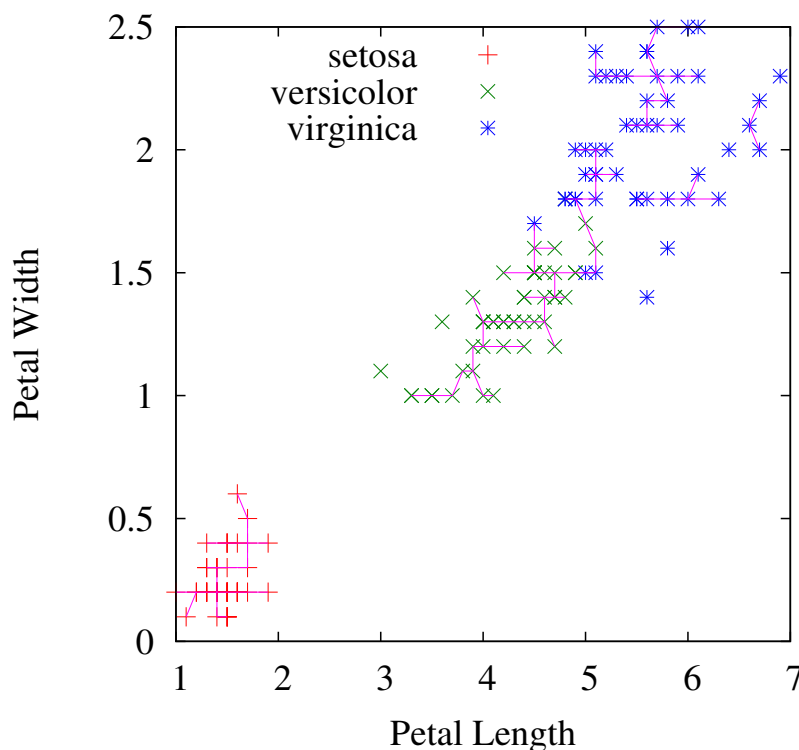
Distances between Clusters

There are several distance measures between clusters.

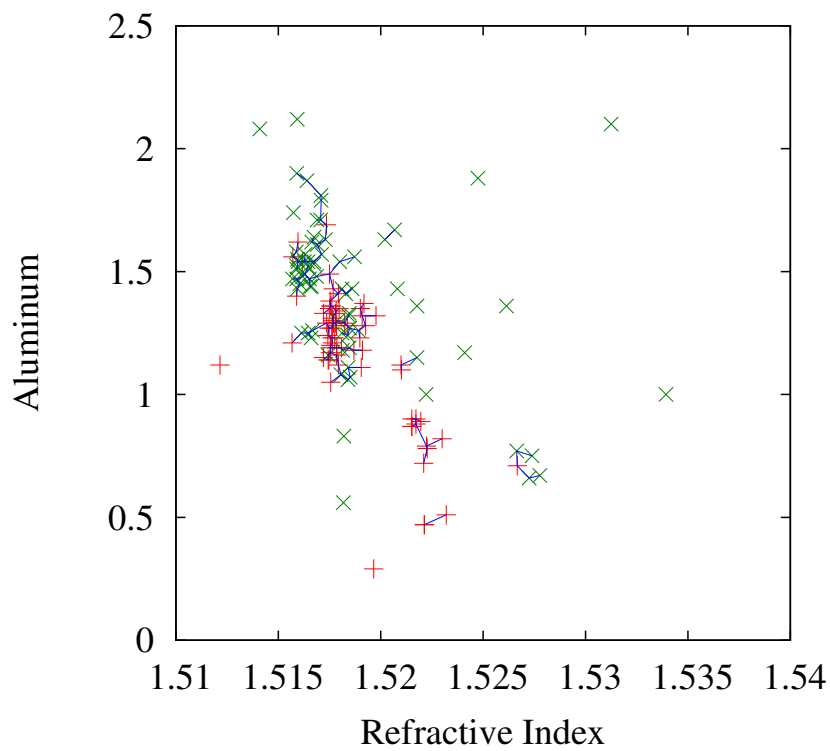
- Nearest Neighbor. The minimize distance between an example in C_{i1} and an example in C_{i2} . This encourages elongated clusters.
- Farthest Neighbor. The maximum distance between an example in C_{i1} and an example in C_{i2} .
- Mean. The distance between the means of C_{i1} and C_{i2} .

In the following, intermediate results are shown for measuring distance by nearest neighbor (first two graphs) and mean (last two graphs).

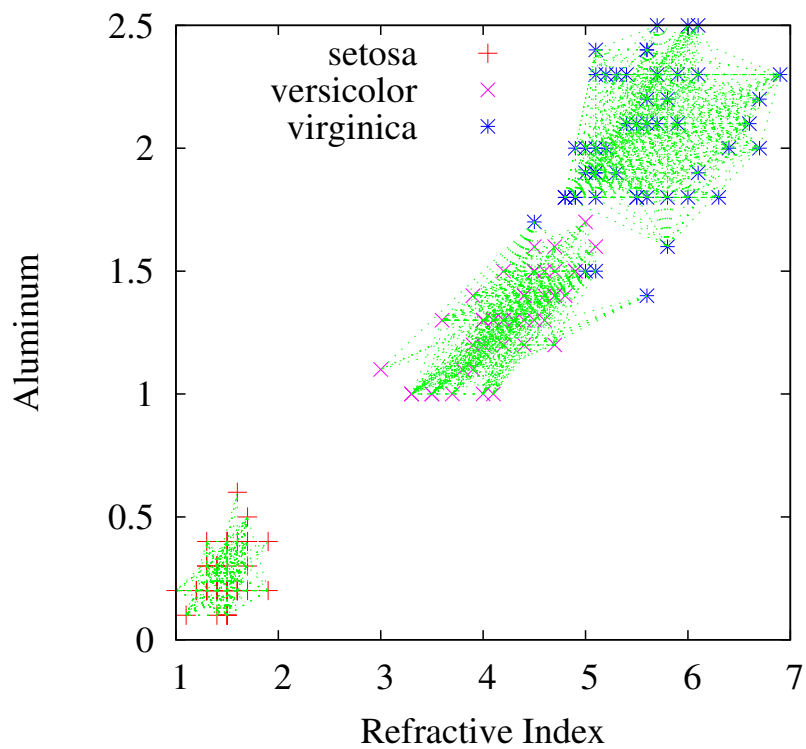
Simplified Iris Dataset (Nearest Neighbor)



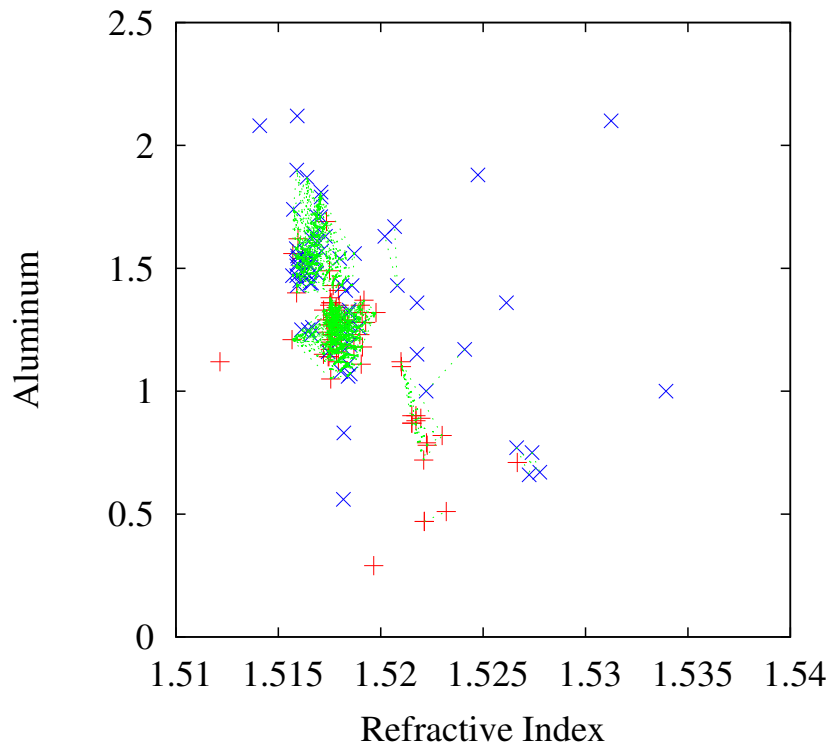
Simplified Glass Dataset (Nearest Neighbor)



Simplified Iris Dataset (Mean)

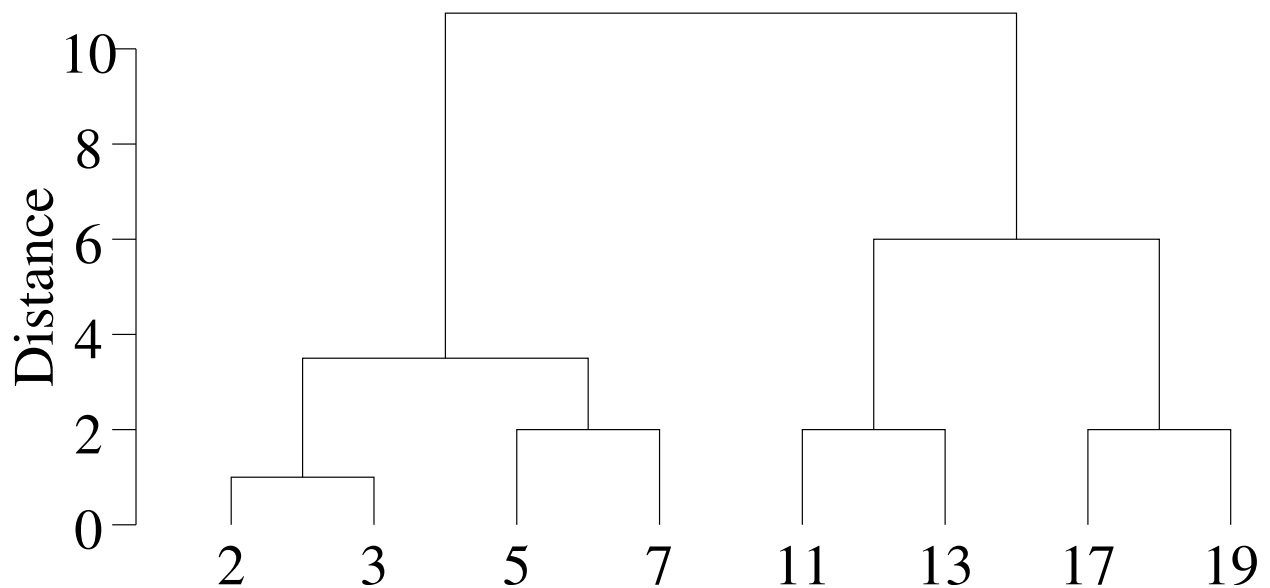


Simplified Glass Dataset (Mean)



Dendograms

A dendrogram illustrates the hierarchical clustering. Here is a dendrogram for the numbers 2, 3, 5, 7, 11, 13, 17, 19 using mean distance.



Association Rules

An association rule indicates that a set of items implies another item with levels of confidence and support.

In market basket analysis, the goal is to determine what items are bought with other items, e.g., cereal \rightarrow milk.

For our purposes, an *item* specifies attribute=value (later discuss $<$ and $>$).

Each example \mathbf{x} is treated as a *transaction*, which is a set of items.

An *association rule* is an implication $A \rightarrow B$, where A and B are sets of items (typically though $|B| = 1$).

Confidence and Support

The *support* of a set of items A is the number of examples \mathbf{x} where $A \subseteq \mathbf{x}$.

The *support* of an association rule $A \rightarrow B$ is the support of $A \cup B$.

The *confidence* of an association rule $A \rightarrow B$ is $\text{support}(A \cup B) / \text{support}(A)$.

$\text{support}(A)$ corresponds to $\Pr(A)$.

$\text{confidence}(A \rightarrow B)$ corresponds to $\Pr(B|A)$.

No.	Attributes				Class
	Outlook	Temp	Humidity	Windy	
1	sunny	hot	high	false	neg
2	sunny	hot	high	true	neg
3	overcast	hot	high	false	pos
4	rain	mild	high	false	pos
5	rain	cool	normal	false	pos
6	rain	cool	normal	true	neg
7	overcast	cool	normal	true	pos
8	sunny	mild	high	false	neg
9	sunny	cool	normal	false	pos
10	rain	mild	normal	false	pos
11	sunny	mild	normal	true	pos
12	overcast	mild	high	true	pos
13	overcast	hot	normal	false	pos
14	rain	mild	high	true	neg

Examples

Here are some association rules from the weather dataset.

Association Rule	Conf	Sup
overcast \rightarrow pos	4/4	4
cool \rightarrow normal	4/4	4
(sunny \wedge neg) \rightarrow high	3/3	3
normal \rightarrow pos	6/7	6
neg \rightarrow high	4/5	4
cool \rightarrow (normal \wedge pos)	3/4	3
mild \rightarrow high	4/6	4
(normal \wedge pos) \rightarrow false	4/6	4

Frequent Itemsets

The Apriori algorithm for finding association rules is based on finding *frequent k -itemsets*, a set of k items with \geq *minimum support*, which is a parameter to Apriori.

If we require a minimum support of 5, then:

Frequent 1-itemsets		Frequent 2-itemsets	
sunny	true	normal \wedge pos	
rainy	false	false \wedge pos	
mild	pos		
high	neg		
normal			

Apriori Algorithm

Apriori

$L_1 =$ frequent 1-itemsets

For k from 2 on up

$C_k \leftarrow$ Apriori-gen(L_{k-1}) // Candidates

For each example \mathbf{x}_i

$C_{k,i} \leftarrow \{c : c \in C_k \wedge c \subseteq \mathbf{x}_i\}$

For each candidate $c \in C_{k,i}$

Increment c .count

$L_k \leftarrow \{c : c \in C_k \wedge c.\text{count} \geq \text{min. support}\}$

Exit loop if $L_k = \emptyset$

Return L_1, \dots, L_{k-1}

Apriori Candidate Generation

```
Apriori-gen( $L_{k-1}$ )
  // Assuming  $L_{k-1}$  and each  $c \in L_{k-1}$  is sorted
  // Join step
  For each itemset pair  $p \in L_{k-1}$  and  $q \in L_{k-1}$ 
    If  $p = q$  except for their last items
      Then insert  $p \cup q$  into  $C_k$ 
  // Prune step
  For each candidate  $c \in C_k$ 
    If some  $(k-1)$ -subset of  $c$  is not in  $L_{k-1}$ 
      Then remove  $c$  from  $C_k$ 
  Return  $C_k$ 
```

Sorted sets allows efficient candidate generation.

Principal Components Analysis

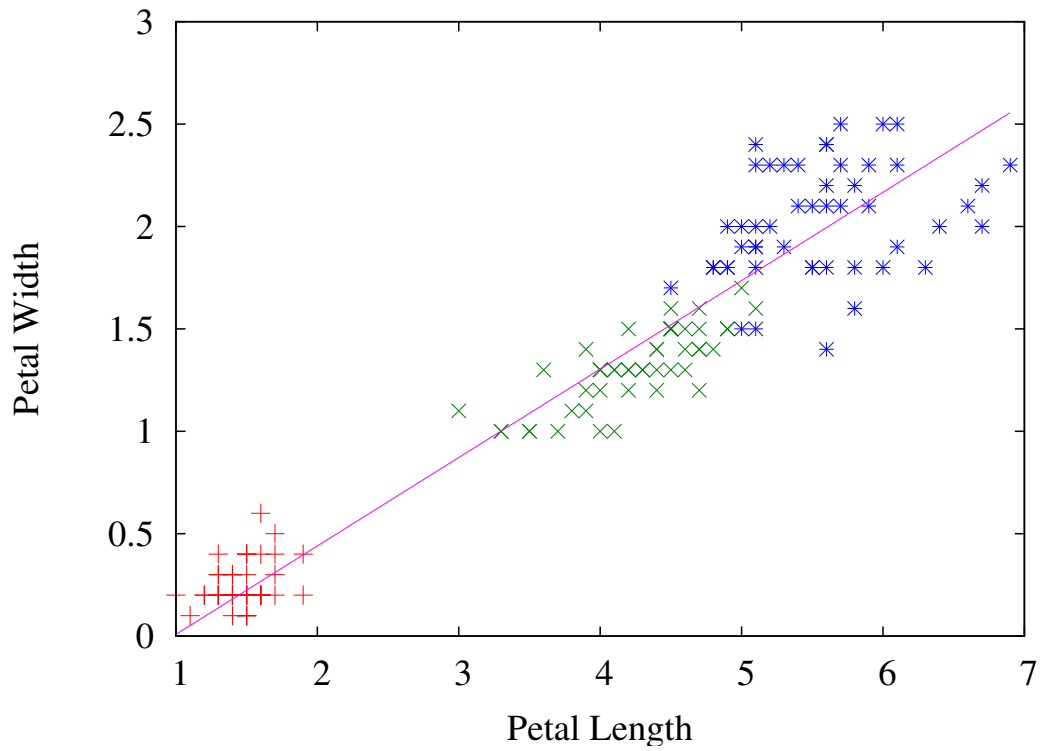
PCA is one of several techniques to analyze and/or extract the most relevant attributes from a dataset.

PCA identifies and ranks n new attributes in the dataset that are linear combinations of the n old attributes.

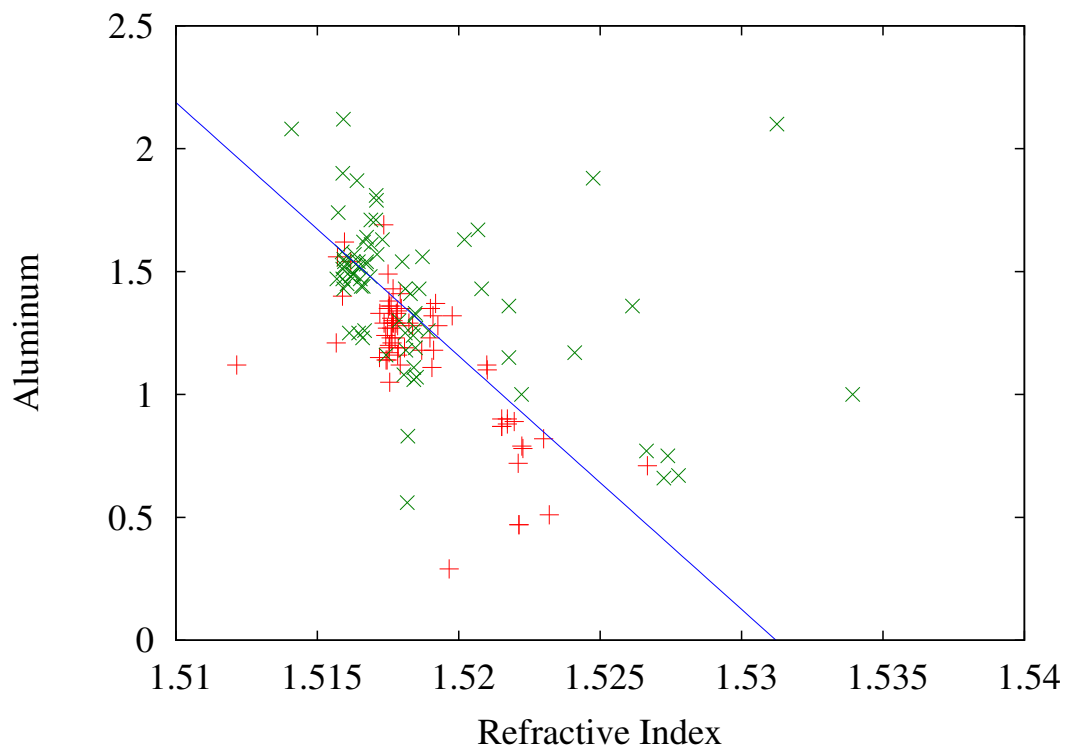
The ranking is by variance reduction, i.e, by the variance in the new attributes.

The new attributes are orthogonal to each other, so the full result is just a linear transformation; however, we can select those attributes which account for the most variance in the data (dimensionality reduction).

Principal Component of Simplified Iris



Principal Component of Simplified Glass



Finding Principal Components

Principal components are eigenvectors of the correlation matrix. If the examples \mathbf{x} have n attributes with mean \mathbf{u} , then the $n \times n$ correlation matrix A is defined by:

$$\mathbf{A}_{ij} = \frac{\sum_{\mathbf{x}} (x_i - u_i)(x_j - u_j)}{\sqrt{\sum_{\mathbf{x}} (x_i - u_i)^2 \sum_{\mathbf{x}} (x_j - u_j)^2}}$$

An $n \times 1$ eigenvector \mathbf{e} with eigenvalue λ satisfies:

$$\mathbf{A}\mathbf{e} = \lambda\mathbf{e}$$

The eigenvectors are ranked by decreasing eigenvalues. Algorithms are in the online *Numerical Recipes* book.

Examples

In each dataset, the attributes have been standardized:

Using 4-attribute Iris, top two principal components are:

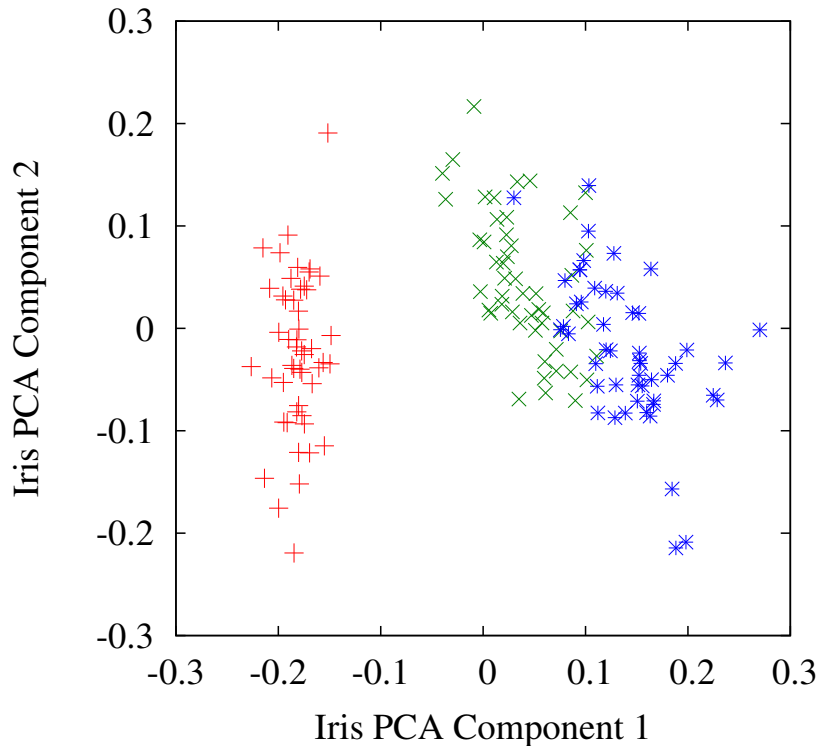
$$0.522 * slen - 0.269 * swid + 0.580 * plen + 0.565 * pwid \\ -0.377 * slen - 0.923 * swid - 0.024 * plen - 0.067 * pwid$$

Using 9-attribute Glass (2 classes), the top two are:

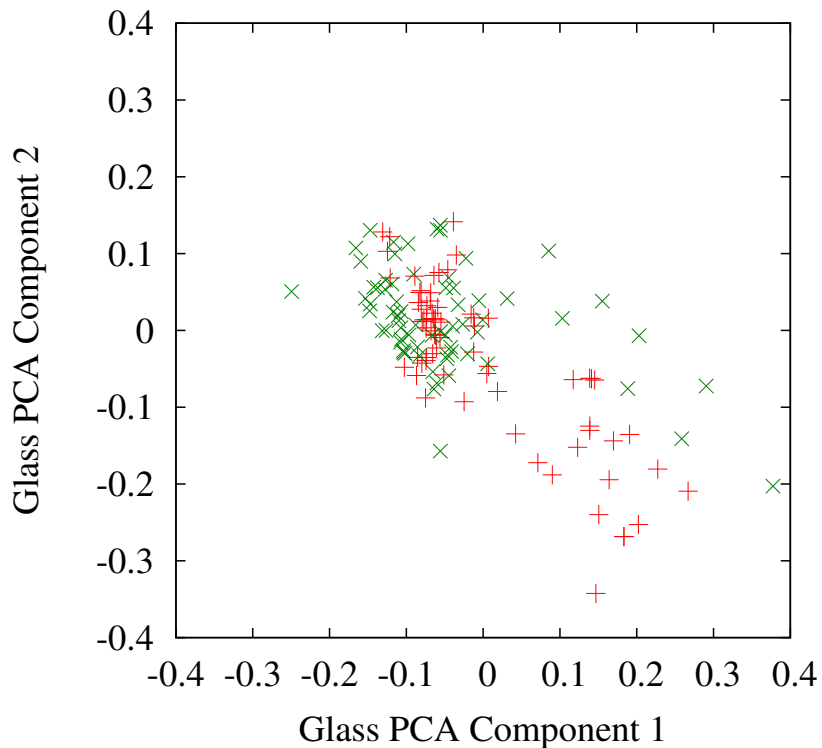
$$0.505 * RI - 0.038 * Na - 0.359 * Mg - 0.268 * Al - 0.344 * \\ Si - 0.381 * K + 0.491 * Ca - 0.182 * Ba + 0.068 * Fe$$

$$0.004 * RI - 0.562 * Na - 0.314 * Mg + 0.417 * Al - 0.124 * \\ Si + 0.371 * K + 0.141 * Ca + 0.379 * Ba + 0.308 * Fe$$

Top Two Principal Components of Iris



Top Two PCs of Glass (2 classes)



(Some green X's not shown above and right)

Comments on Unsupervised Learning

Unsupervised learning can be useful for finding interesting patterns; however they can often find irrelevant patterns.

City = San Antonio \rightarrow State = Texas

Difficult to determine number of clusters or components.

Clustering: Choose k if higher does not improve much.

For PCA, eliminate eigenvalues with low eigenvalues.

This is a small sample of unsupervised techniques.

Besides the Duda, Hart, Stork book, consider:

T. Hastie, R. Tibshirani and J. Friedman (2001). *The Elements of Statistical Learning*. Springer-Verlag.

R. E. Neapolitan (2004). *Learning Bayesian Networks*. Prentice-Hall.