ISYE 4803 - Business Analytics
Homework 4 - Advanced Data-Modeling Tools
April 12, 2018

Cameron Anderson
Katie Butler
Kristen Drexinger
Nicholas Hardy
Langdon Hollingsworth
Alex McAuliffe
Will Olsson
Yijun (Emma) Wan

Table of Contents

1. Workload Distribution

Student Names	Topics
Katie Butler, Alex McAuliffe	Classification Procedures
Cameron Anderson, Emma Wan	Cluster Analysis
Langdon Hollingsworth, Will Olsson, Nick Hardy, Kristen Drexinger	Dimension Reduction Procedures

2. Classification Procedures

a. Summary

Classification procedures, along with cluster analysis, are used in machine learning, information retrieval, image investigation, and other similar topics. These procedures are crucial for managing algorithms in data mining processes. Specifically, classification procedures use a set of predefined classes to determine the class of a new object. For example, one may have a set of learning styles that include three classes: auditory, visual, and kinesthetic. Then, one may aim to place a student into one of these classes, therefore determining his learning style. Because classification procedures make use of data that has already been classified or labeled, it is considered to be a supervised learning technique. Machines learn from this labeled data and then classify new data.

b. Purpose and Functionality

Classifiers, algorithms that map information to a specific class, are used to analyze data. Classification aims to determine which class a certain object belongs to based on predefined labels. The goal of classification is to accurately predict the target class for each case in the data. There are a variety of classification procedures, and each one uses different techniques to determine the relationships between the values of the predictors and the values of the the target.

c. Assumptions

- Classification procedures depend on predefined labels or classes
- Predefined labels or classes are available
- In the classification process, determining the boundary conditions is crucial
- Uses both labeled and unlabeled data

d. Solution Methods

There are two schools of methods for classification procedures: statistical approaches and computing approaches. In each school of methods, there are several options. Each option should be used in a specific situation, and each has its own strengths and weaknesses.

i. Statistical Approaches:

- Linear Regression
- Logistics Regression
- Linear Discriminant Analysis (LDA)
- Quadratic Discriminant Analysis (QDA)
- Regulated Discriminant Analysis (RDA)
- Generalized Linear Discriminants
- Flexible Discriminants
- Mixture Discriminants
- Bayes Classifier and Naive Bayes Classifier
- Kernel-Based Classification

ii. Computing Approaches:

- Separating Hyperplanes
- Optimal Separating Hyperplane (Support Vector Machine)
- Artificial Neural Network
- Decision Tree
- Mathematical Programming Based Classification
- Nearest Neighbor (local) Classification

Although there are numerous approaches for each school of methods, only one popular procedure will be reviewed in depth from each school of methods. From the statistical approaches, Linear Discriminant Analysis (LDA) will be reviewed, and then Optimal Separating Hyperplanes will be discussed from the computing approaches.

Linear Discriminant Analysis (LDA) is a statistical method to make supervised learning decisions about the categorization of data. It uses X and Y data to assign classes.

LDA assumes that $\hat{x}^{(k)} = (x_1, x_2 ... x_p)^{T(k)} \sim N_p(\mu^{(k)}, \hat{\Sigma}^{(k)})$; that is, that data is equivalent to likelihood for k = 1,2,..,K classes.

Bayes Rule is applied to calculate the posterior probability Pr(Y=k | X = x)

For LDA, we further assume that all Σ are equal - that is, all classes have identical matrices. This is a strong assumption that can be relaxted by using QDA (Quandratic) or RDA (Regulated) Discriminant Analysis.

Optimal Separating Hyperplanes (OHS) is a computing method to make supervised decisions about the classification of data. Like other Classification procedures, it uses both Y and X data. This method works by separating the two classes and maximizing the distance to the closest point in either class. This provides a unique solution to the separating hyperplane problem, and further performs better on data for having maximized the margin of the training data.
OSH can be formulated in the following way:
Find $\widehat{\beta_0}$ and $\widehat{\beta}$ to maximize M

Subject to $(y_i * (x_i^T \widehat{\beta} + \widehat{\beta}_0)) \geq M$ for i=1,2,...,n; where M = margin

And $|\widehat{\beta}| = 1$

The first constraint ensures that the distance from any point to the boundary is greater than the margin. The second constraint ensures that the beta coefficients are normalized.

e. Available Software Programs with Syntax Examples

Excel and Matlab each have powerful capabilities for LDA, with some plug-ins and setup. A full Matlab implementation can be downloaded from the Matlab file sharing forum, and XLSTAT can be used in Excel.

f. Potential ISyE Applications

Classification procedures have many uses in industrial and systems engineering, and data mining more broadly. Some potential uses include:

- Recommendation engines for consumer profiles
- Social media content feeds
- Automatic organization of websites
- Medical diagnoses / triage
- Credit card approval classification
- Improving complex system performance (such as a factory line)
- Labeling astronomical bodies as stars, nebulas, etc.

3. Cluster Analysis

a. Summary

Cluster analysis is a data exploration (mining) tool for dividing a multivariate dataset into "natural" clusters or groups. It uses methods to explore whether previously undefined clusters exist in the dataset. The clusters are chosen by some inherent similarity between the data points

that is reasonable to separate them from the rest of the data. The greater similarity between data in a group or cluster, and similarly the greater the difference between each cluster of an entire data set, helps create meaningful results with the analysis. Selecting the "correct" clusters is often a more difficult process when in practice because of the sheer size and complexity of data that could be available.
Cluster analysis itself is not one specific algorithm, but the general task to be solved. Thus there are several different methods one can use to perform cluster analysis.

b. Purpose and Functionality

Cluster Analysis is used when it is believed that the sample units come from an unknown number of distinct populations or sub-populations. The objective is to describe those populations with the observed data.

c. Assumptions

Since there are many different types of cluster analysis, it is hard to make assumptions that apply to all methods. However, there are a few main assumptions for cluster analysis:

- Representative - Sample accurately reflects similarities of the general population
- Randomness - Variables are not correlated when plotted against each other
- We also assume that the sample units come from a number of distinct populations

d. Solution Methods

There are two schools of methods for cluster analysis: statistical approaches and computing approaches. In each school of methods, there are several options. Each option should be used in a specific situation, and each has its own strengths and weaknesses.

i. Statistical Approaches:
- Multivariate Normal Distribution

ii. Computing Approaches:
- Hierarchical Clustering
- K-means
- DBSCAN and OPTICS
- Biclustering or Co-Clustering
- Self-Organization Map and Multidimensional Scaling
- Clique

Although there are numerous approaches for each school of methods, only one popular procedure will be reviewed in depth from each school of methods. From the statistical

approaches, Multivariate normal distribution will be reviewed, and then K-means will be discussed from the computing approaches.

Multivariate normal distribution

In probability theory and statistics, the multivariate normal distribution is a generalization of the univariate normal distribution to higher dimensions. One definition is that a random vector is said to be k-variate normally distributed if every linear combination of its k components has a univariate normal distribution. Its importance derives mainly from the multivariate central limit theorem. The multivariate normal distribution is often used to describe, at least approximately, any set of (possibly) correlated real-valued random variables each of which clusters around a mean value.

The multivariate normal distribution of a k-dimensional random vector $\mathbf{X} = [X_1, X_2, \ldots, X_k]^{\mathrm{T}}$ can be written in the following notation:

$$\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}),$$

or to make it explicitly known that X is k-dimensional,

$$\mathbf{X} \sim \mathcal{N}_k(\boldsymbol{\mu}, \boldsymbol{\Sigma}),$$

with k-dimensional mean vector

$$\boldsymbol{\mu} = \mathrm{E}[\mathbf{X}] = [\mathrm{E}[X_1], \mathrm{E}[X_2], \ldots, \mathrm{E}[X_k]]^{\mathrm{T}},$$

And $k \times k$ covariance matrix

$$\boldsymbol{\Sigma} =: \mathrm{E}[(\mathbf{X} - \boldsymbol{\mu})(\mathbf{X} - \boldsymbol{\mu})^{\mathrm{T}}] = [\mathrm{Cov}[X_i, X_j]; 1 \leq i, j \leq k].$$

K-means:

K-means clustering is a method of vector quantization, originally from signal processing, that is popular for cluster analysis in data mining. k-means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells.

The problem is computationally difficult (NP-hard); however, there are efficient heuristic algorithms that are commonly employed and converge quickly to a local optimum. These are usually similar to the expectation-maximization algorithm for mixtures of Gaussian distributions via an iterative refinement approach employed by both k-means and Gaussian Mixture Modeling. Additionally, they both use cluster centers to model the data; however, k-means clustering tends to find clusters of comparable spatial extent, while the expectation-maximization mechanism allows clusters to have different shapes.

The algorithm has a loose relationship to the k-nearest neighbor classifier, a popular machine learning technique for classification that is often confused with k-means because of the k in the name. One can apply the 1-nearest neighbor classifier on the cluster centers obtained by k-means to classify new data into the existing clusters. This is known as nearest centroid classifier or Rocchio algorithm.

Given a set of observations ($\mathbf{x}_1$, $\mathbf{x}_2$, …, $\mathbf{x}_n$), where each observation is a d-dimensional real vector, k-means clustering aims to partition the n observations into k (≤ n) sets $\mathbf{S} = \{S_1, S_2, \ldots, S_k\}$ so as to minimize the within-cluster sum of squares (WCSS) (i.e. variance). Formally, the objective is to find:

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2 = \arg\min_{\mathbf{S}} \sum_{i=1}^{k} |S_i| \operatorname{Var} S_i$$

where $\boldsymbol{\mu}_i$ is the mean of points in Si. This is equivalent to minimizing the pairwise squared deviations of points in the same cluster:

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \frac{1}{2|S_i|} \sum_{\mathbf{x},\mathbf{y} \in S_i} \|\mathbf{x} - \mathbf{y}\|^2$$

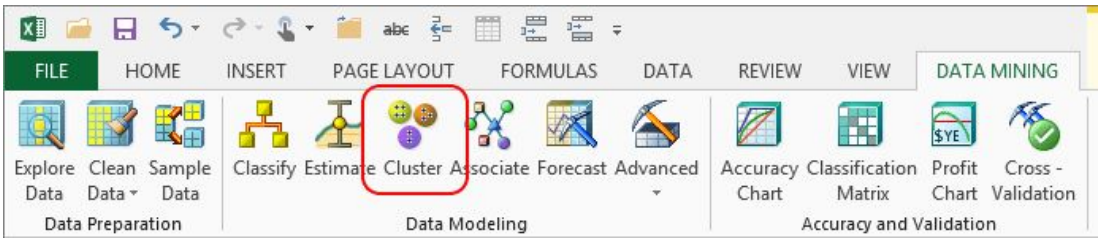
The Equivalence can be deduced from identity

$$\sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2 = \sum_{\mathbf{x} \neq \mathbf{y} \in S_i} (\mathbf{x} - \boldsymbol{\mu}_i)(\boldsymbol{\mu}_i - \mathbf{y})$$

Because the total variance is constant, this is also equivalent to maximizing the squared deviations between points in different clusters (between-cluster sum of squares, BCSS).

e. Available Software Programs with Syntax Examples

1. Cluster Wizard excel add-in
    a. The Cluster wizard helps you build a model that detects rows that share similar characteristics and groups them to maximize the distance between groups. This wizard is useful for finding patterns in all kinds of data.
    The Cluster wizard uses the Microsoft Clustering algorithm and can be extensively customized. It works on existing data from an Excel table, an Excel range, or an Analysis Services query. Similar functionality is provided by the Detect Categories tool, provided in the Table Analysis Tools for Excel. However, the Detect Categories tool cannot be customized and must use data in Excel tables.



b. How to use the Cluster Wizard:
   - In the Data Mining ribbon, click Cluster, and then click Next.
   - In the Select Source Data page, select an Excel table or range. Or specify and external data source.
   - If you use an external data source, you can create custom views or paste in custom query text, and save the data set as an Analysis Services data source.
   - On the Clustering page, you can customize the way the model is built.
   - For Number of segments, you can tell the wizard to create a fixed number of categories, or let it automatically detect the optimum number of groupings.
   - Review the list of columns in the Input columns list, and deselect any columns that are not useful in creating patterns. Columns you should exclude include ID numbers, customer names, and so on.
   - Optionally, click Parameters to change the algorithm parameters and customize the behavior of the clustering model.
   - In the Split data into training and testing sets page, specify how much data to hold out for testing. The remainder is always used for training the model.
   - The default setting is 30% testing data and 70% training data.
   - On the Finish page, provide a descriptive name for your data set and model, and set the following options that control how you work with the finished model:
   - Browse model. When this option is selected, as soon as the wizard finishes processing the model, it opens a Browse window to help you explore the results. The contents of the viewer depend on the type of model you built. For more information, see Browsing a Clustering Model.
   - Enable drillthrough. Select this option to view the underlying data from the finished model. This option is only available if you build a Decision Tree model.
   - Use temporary model. If you select this option, the model will not be saved to the server. Temporary models are deleted when you close Excel.

2. K-means Excel

a. Example: Apply the second version of the K-means clustering algorithm to the data in range B3:C13 of Figure 1 with k = 2.
   - Cluster analysis k-means

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
3		X	Y		Cluster		Centroid	1	2		Dist-sq	1	2		Cluster
4		5	0		1		X	2.6	3.2		7.72	7.72	12.24		1
5		5	2		2		Y	1.4	3		4.24	6.12	4.24		2
6		3	1		1						0.32	0.32	4.04		1
7		0	4		2		SSE	36.92			11.24	13.52	11.24		2
8		2	1		1						0.52	0.52	5.44		1
9		4	2		2		Converge	FALSE			1.64	2.32	1.64		2
10		2	2		1						0.72	0.72	2.44		1
11		2	3		2						1.44	2.92	1.44		2
12		1	3		1						4.84	5.12	4.84		2
13		5	4		2						4.24	12.52	4.24		2
14															
15		X	Y		Cluster		Centroid	1	2		Dist-sq	1	2		Cluster
16		5	0		1		X	3	2.833333		5	5	13.69444		1
17		5	2		2		Y	1	3		5	5	5.694444		1
18		3	1		1						0	0	4.027778		1
19		0	4		2		SSE	33.47222			9.027778	18	9.027778		2
20		2	1		1						1	1	4.694444		1
21		4	2		2		Converge	FALSE			2	2	2.361111		1
22		2	2		1						1.694444	2	1.694444		2
23		2	3		2						0.694444	5	0.694444		2
24		1	3		2						3.361111	8	3.361111		2
25		5	4		2						5.694444	13	5.694444		2

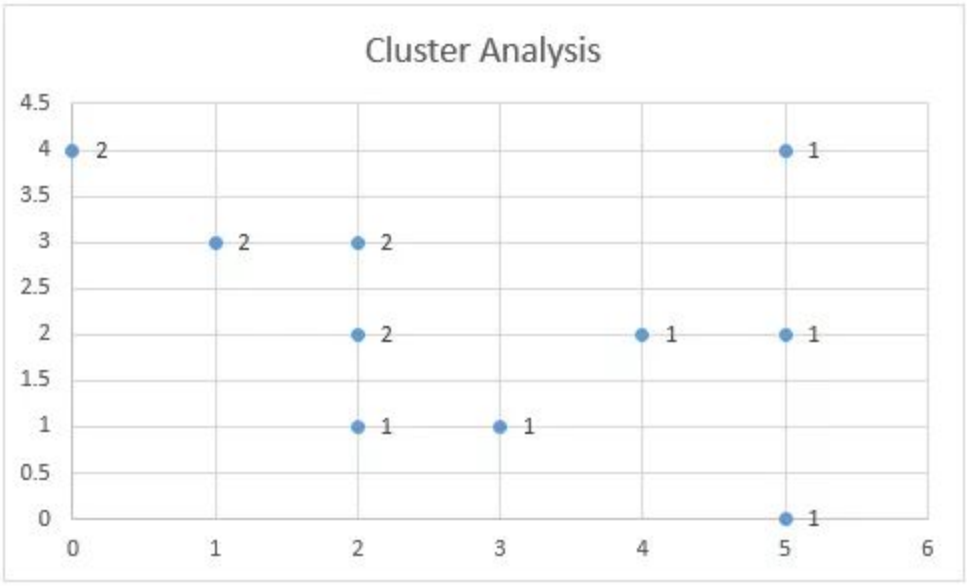
   - The data consists of 10 data elements which can be viewed as two-dimensional points (see Figure 3 for a graphical representation). Since there are two clusters, we start by assigning the first element to cluster 1, the second to cluster 2, the third to cluster 1, etc. (step 2), as shown in range E3:E13.
   - We next set the centroids of each cluster to be the mean of all the elements in that cluster. The centroid of the first cluster is (2.6, 1.4) where the X value (in cell H4) is calculated by the formula =AVERAGEIF(E4:E13,1,B4:B13) and the Y value (in cell H5) is calculated by the formula =AVERAGEIF(E4:E13,1,C4:C13). The centroid for the second cluster (3.2, 3.0) is calculated in a similar way.
   - We next calculate the squared distance of each of the ten data elements to each centroid. E.g. the squared distance of the first data element to the first centroid is 7.72 (cell L4) as calculated by =(B4-H4)^2+(C4-H5)^2 or equivalently =SUMXMY2($B4:$C4,H$4:H$5). Since the squared distance to the second cluster is 12.24 (cell M4) is higher we see that the first data element is closer to cluster 1 and so we keep that point in cluster

1 (cell O4). Here cell K4 contains the formula =MIN(L4:M4) and cell O4 contains the formula =IF(L4<=M4,1,2).

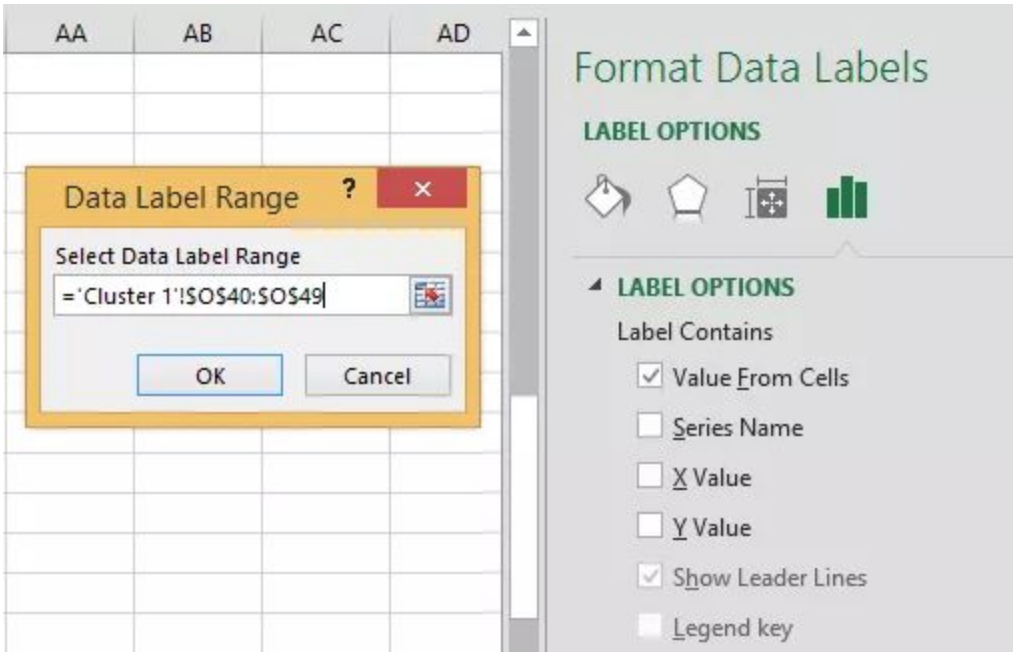
- We proceed in this way to determine a new assignment of clusters to each of the 10 data elements as described in range O4:O13. The value of for this assignment is 36.92 (cell H7). Since the original cluster assignment (range E4:E13) is different from the new cluster assignment, the algorithm has not yet converged and so we continue. We simply copy the latest cluster assignment into the range E16:E25 and repeat the same steps.
- After four steps we get convergence, as shown in Figure 2 (range E40:E49 contains the same values as O40:O49). The final assignment of data elements to clusters is shown in range E40:E49. We also see that and note that each step in the algorithm has reduced the value of SSE.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
27		X	Y		Cluster		Centroid	1	2		Dist-sq	1	2		Cluster
28		5	0		1		X	3.8	2		2.88	2.88	19.24		1
29		5	2		1		Y	1.2	3.2		2.08	2.08	10.44		1
30		3	1		1						0.68	0.68	5.84		1
31		0	4		2		SSE	26.04			4.64	22.28	4.64		2
32		2	1		1						3.28	3.28	4.84		1
33		4	2		1		Converge	FALSE			0.68	0.68	5.44		1
34		2	2		2						1.44	3.88	1.44		2
35		2	3		2						0.04	6.48	0.04		2
36		1	3		2						1.04	11.08	1.04		2
37		5	4		2						9.28	9.28	9.64		1
38															
39		X	Y		Cluster		Centroid	1	2		Dist-sq	1	2		Cluster
40		5	0		1		X	4	1.25		3.777778	3.777778	23.0625		1
41		5	2		1		Y	1.666667	3		1.111111	1.111111	15.0625		1
42		3	1		1						1.444444	1.444444	7.0625		1
43		0	4		2		SSE	22.08333			2.5625	21.44444	2.5625		2
44		2	1		1						4.444444	4.444444	4.5625		1
45		4	2		1		Converge	TRUE			0.111111	0.111111	8.5625		1
46		2	2		2						1.5625	4.111111	1.5625		2
47		2	3		2						0.5625	5.777778	0.5625		2
48		1	3		2						0.0625	10.77778	0.0625		2
49		5	4		1						6.444444	6.444444	15.0625		1

- The above figure graphically shows the assignment of data elements to the two clusters. This chart is created by highlighting the range B40:C49 and selecting Insert > Charts|Scatter.



- The labels (1 and 2) can be added to each point by opening the Format Data Labels dialog box and then deselecting the Y Value option and selecting the Value from Cells option as shown in Figure 1.4 (for Excel 2013). The Data Label Range dialog box then appears. Insert the range O40:O49 (which contains the cluster assignments) and press the OK button.



f. Potential ISyE Applications

Clustering has a large number of applications spread across various domains. Some of the most popular applications of clustering are:

- Recommendation engines
- Market segmentation
- Social network analysis
- Search result grouping
- Medical imaging
- Image segmentation
- Anomaly detection

4. Dimension Reduction Procedures

a. Summary

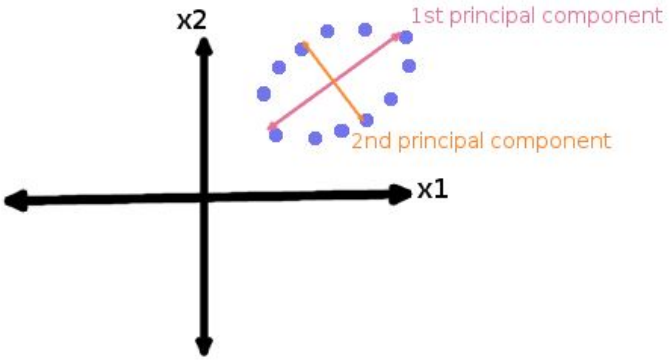
Dimension reduction is the process of reducing the number of random variables under consideration to a smaller number of principal variables. The new model should have uncorrelated principal variables and the data should be easier to interpret than the original model. Dimension reduction includes feature extraction and feature selection. Feature extraction techniques include Principal Component Analysis (PCA) and Partial Least Squares Regression (PLS). Feature selection refers to variable-selection. We will focus on problems where the number of x variables is less than the number of observations ($n<<p$).

b. Purpose and Functionality

Feature extraction transforms high dimensional data to a space with fewer dimension. PCA and PLS apply linear combinations to reduce the number of dimensions.

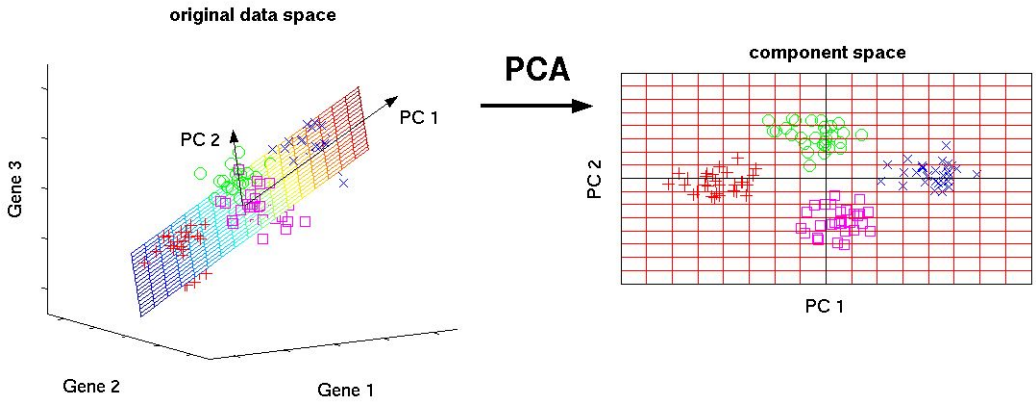
1) Principal Component Analysis

The reduced space PCA calculates provides a sequence of the best linear approximations of the data. The updated set of independent variables, or the principal components, are mutually uncorrelated and ordered in variance. Out of the components, the first principle component will account for the largest amount of the original data's possible variation. The next principal component should be orthogonal to the preceding principal component, thereby minimizing the possible covariance between principal components and capturing data not explained by the first principal component. The figure below shows how PCA would be applied to a two-dimensional dataset.



Our final principal components should explain the majority of the variance, while the insignificant principal components can be discarded with minimal loss of information.

The figure below shows how PCA can simplify a complicated data set.



c. Assumptions

PCA assumptions:

1) Data follows a normal distribution.
2) We find the most important information in the space where the x data variance is maximized.
3) X variables have a linear relation and linear transformation can efficiently reduce the data dimensionality.
4) Each data variable is numerical.
5) Outliers have been removed.
6) There is some correlation between the independent variables.

PLS assumptions:

1) Error terms are assumed to be independent and identically distributed normal variables
2) Linear transformation can efficiently reduce the data dimensionality.

d. Solution Methods

PCA:
Given a set of vectors $x_1, x_2, ..., x_n$, with each x defined as a p-dimensional vector. The observations stacked creates the vector X, a $n \cdot p$ matrix. We create a singular value decomposition of X as $X = UDV^T$.

U is a $N \cdot p$ orthogonal matrix, with columns u_j as left singular vectors.

V is a $p \cdot p$ orthogonal matrix, with columns v_j as left singular vectors.

D is a $p \cdot p$ diagonal matrix, where the diagonal elements $d_1, d_2, ..., d_n$ are the singular values.

The columns of UD are the principal components of X. The j-th principal component will represent data capturing d_j amount of variance. The U_j principal component will be a linear combination of X determined by the weights $w = [w_1, w_2, ..., w_n]$, represented as $U_j = w^T x$ in its matrix form. Var(U_j)= Var($w^T x$)= $w^T S w$. S is the sample covariance matrix. Then, we maximize $w^T S w$ subject to $w^T w = 1$. The resulting principal components are normalized eigenvectors.

PLS:
Given a set of predictors $x_1, x_2, ..., x_n$, with each x defined as a p-dimensional vector, and corresponding response y values. PLZ operates by finding a linear regression model by projecting the predicted variables and their corresponding observable values (y) to a new "space" to model the covariance structure.
The final model has to be formed through variable selection. There are many ways to select the final variables that will be included in the model with the most popular being forward stepwise, backwards stepwise, and best-subset regression. Forward stepwise regression starts by adding the most significant variable, adding it into the model and repeating, while removing any new non-significant terms. Backwards regression starts with all variables in the model and removes the least significant until a model is found. For these two procedures, the optimal model can be found by comparing r-squared(pred) and variance between models. In best-subset, the computer starts with one model and removes and adds them as seems fit and outputs a few models with statistics. For best subset, the subset with the lowest Cp is chosen.

e. Available Software Programs with Syntax Examples

PCA in R:

The r command prcomp() performs PCA, defaulting the mean to equal zero. We can use the parameter scale.=T to normalize the data to have a standard deviation of one. If we are evaluating the data set *testdata* that follows each assumption,
prin_comp<- prcomp(testdata, scale.=T)

The function *names(prin_comp)* will return *"sdev" "rotation" "center" "scale" "x"*.

To output the means of the variables, you can use the function *prin_comp$center*. To output the standard deviation of variables, use the function *prin_comp$scale*. *prin_comp$rotation* will output the principal component loading vector, which will give $U_j = w^T x$ for each principal component.

PCA in Python:
Python uses the package *sklearn* that you must import from the python library. The code below shows how to do this:

```
import numpy as np
from sklear.decomposition import PCA
import pandas as pd
import matplotlib.plot as plt
from sklearn.preprocessing import scale
%matplotlib inline
```

After loading the data set with the command *data=pd.read_csv("File name.csv")*, convert the data to numpy arrays: *X=data.values*. Scale the values and perform PCA analysis as shown below:

```
X=scale(X)
pca= PCA(n_components= n)
pca.fit(X)
var=pca.explained_variance_ratio_  #gives amount of variance each PC explains.
```

Pls package in R

The PLS-DA package in R allows for discrimant analysis using PLS. The code below outlines a classification procedure that creates a model on the Iris dataset, plots it, and tests it using R's pls package.

```
data(iris)
tmp.data<-iris[,-5]
tmp.group<-iris[,5] # species
tmp.y<-matrix(as.numeric(tmp.group),ncol=1) # make numeric matrix
```

```
train.test.index.main=test.train.split(nrow(tmp.data),n=1,strata=tmp.group,split.type="duplex",da
ta=tmp.data)
train.id<-train.test.index.main=="train"

#partition data to get the trainning set
tmp.data<-tmp.data[train.id,]
tmp.group<-tmp.group[train.id]
tmp.y<-tmp.y[train.id,]
mods<-make.OSC.PLS.model(tmp.y,pls.data=scaled.data,comp=2,OSC.comp=1, validation =
"LOO",method="oscorespls", cv.scale=TRUE, progress=FALSE)
#extract model
final<-get.OSC.model(obj=mods,OSC.comp=1)
#view out-of-bag error for cross-validation splits
plot.OSC.results(mods,plot="RMSEP",groups=tmp.group)
plot.OSC.results(mods,plot="scores",groups=tmp.group)
plot.PLS.results(obj=final,plot="scores",groups=tmp.group)

train.test.index=test.train.split(nrow(scaled.data),n=100,strata=as.factor(tmp.y)) # strata controls
if the species are sampled from equally
permuted.stats<-permute.OSC.PLS(data=scaled.data,y=as.matrix(tmp.y),n=50,ncomp=2,osc.co
mp=1, progress=FALSE,train.test.index=train.test.index)
#look how our model compares to random chance
OSC.validate.model(model=mods,perm=permuted.stats)

train.stats<-OSC.PLS.train.test(pls.data = scaled.data,pls.y  = tmp.y,train.test.index
,comp=2,OSC.comp=1,cv.scale=TRUE, progress=FALSE)

OSC.validate.model(model=mods,perm=permuted.stats,train=train.stats)

#reset data
scaled.data<-iris[,-5]
tmp.group<-iris[,5]
tmp.y<-matrix(as.numeric(tmp.group),ncol=1)

#make predictions for the test set
mods<-make.OSC.PLS.model(tmp.y,pls.data=scaled.data,comp=2,OSC.comp=1, validation =
"LOO",
method="oscorespls", cv.scale=TRUE, progress=FALSE,train.test.index=train.test.index.main)
```

```
#get the true (actual) and predicted values
#round them to integers to represent discreet species labels
plot.data=data.frame(predicted = round(mods$predicted.Y[[2]][,1],0),actual= mods$test.y)
#note these are numeric but we would prefer to interpret classification of species a class
plot.data$predicted<-factor(plot.data$predicted,labels=levels(iris[,5]),levels=1:3)
plot.data$actual<-factor(plot.data$actual,labels=levels(iris[,5]),levels=1:3)

table(plot.data)
pairs(iris[,-5],pch=21,bg=rainbow(nlevels(iris[,5]),alpha=.75)[iris[,5]],upper.panel=NULL,cex=2
)
par(xpd=TRUE)
legend(.75,1,levels(iris[,5]),fill=rainbow(nlevels(iris[,5]),alpha=.75),bty="n")
```

Scikit-learn PLSRegression package in python

The below python code imports the PLSRegression package and runs a simple PLS fit with

```
>>> from sklearn.cross_decomposition import PLSRegression
>>> X = [[0., 0., 1.], [1.,0.,0.], [2.,2.,2.], [2.,5.,4.]]
>>> Y = [[0.1, -0.2], [0.9, 1.1], [6.2, 5.9], [11.9, 12.3]]
>>> pls2 = PLSRegression(n_components=2)
>>> pls2.fit(X, Y)
...
PLSRegression(copy=True, max_iter=500, n_components=2, scale=True,
      tol=1e-06)
>>> Y_pred = pls2.predict(X)
```

PLS in MATLAB

The common syntax for Partial Least Squares analysis is as follows:

```
[XL,YL] = plsregress(X,Y,ncomp)
[XL,YL,XS] = plsregress(X,Y,ncomp)
[XL,YL,XS,YS] = plsregress(X,Y,ncomp)
```

[XL,YL,XS,YS,BETA] = plsregress(X,Y,ncomp,...)
[XL,YL,XS,YS,BETA,PCTVAR] = plsregress(X,Y,ncomp)
[XL,YL,XS,YS,BETA,PCTVAR,MSE] = plsregress(X,Y,ncomp)
[XL,YL,XS,YS,BETA,PCTVAR,MSE] = plsregress(...,param1,val1,param2,val2,...)
[XL,YL,XS,YS,BETA,PCTVAR,MSE,stats] = plsregress(X,Y,ncomp,...)

Each variable contributes to the PLS in the following way:

[XL,YL] = plsregress(X,Y,ncomp) computes a partial least-squares (PLS) regression of Y on X, using ncomp PLS components, and returns the predictor and response loadings in XL and YL, respectively. X is an n-by-p matrix of predictor variables, with rows corresponding to observations and columns to variables. Y is an n-by-m response matrix. XL is a p-by-ncomp matrix of predictor loadings, where each row contains coefficients that define a linear combination of PLS components that approximate the original predictor variables. YL is an m-by-ncomp matrix of response loadings, where each row contains coefficients that define a linear combination of PLS components that approximate the original response variables.

[XL,YL,XS] = plsregress(X,Y,ncomp) returns the predictor scores XS, that is, the PLS components that are linear combinations of the variables in X. XS is an n-by-ncomp orthonormal matrix with rows corresponding to observations and columns to components.

[XL,YL,XS,YS] = plsregress(X,Y,ncomp) returns the response scores YS, that is, the linear combinations of the responses with which the PLS components XS have maximum covariance. YS is an n-by-ncomp matrix with rows corresponding to observations and columns to components. YS is neither orthogonal nor normalized.

plsregress uses the SIMPLS algorithm, first centering X and Y by subtracting off column means to get centered variables X0 and Y0. However, it does not rescale the columns. To perform PLS with standardized variables, use zscore to normalize X and Y. If ncomp is omitted, its default value is min(size(X,1)-1,size(X,2)).

The relationships between the scores, loadings, and centered variables X0 and Y0 are:

XL = (XS\X0)' = X0'*XS

YL = (XS\Y0)' = Y0'*XS

*XL and YL are the coefficients from regressing X0 and Y0 on XS, and XS*XL' and XS*YL' are the PLS approximations to X0 and Y0.

plsregress initially computes YS as:

YS = Y0*YL = Y0*Y0'*XS,

By convention, however, plsregress then orthogonalizes each column of YS with respect to preceding columns of XS, so that XS'*YS is lower triangular

[XL,YL,XS,YS,BETA] = plsregress(X,Y,ncomp,...) returns the PLS regression coefficients BETA. BETA is a (p+1)-by-m matrix, containing intercept terms in the first row:

Y = [ones(n,1),X]*BETA + Yresiduals

Y0 = X0*BETA(2:end,:) + Yresiduals. Here Yresiduals is the vector of response residuals.

[XL,YL,XS,YS,BETA,PCTVAR] = plsregress(X,Y,ncomp) returns a 2-by-ncomp matrix PCTVAR containing the percentage of variance explained by the model. The first row of PCTVAR contains the percentage of variance explained in X by each PLS component, and the second row contains the percentage of variance explained in Y.

[XL,YL,XS,YS,BETA,PCTVAR,MSE] = plsregress(X,Y,ncomp) returns a 2-by-(ncomp+1) matrix MSE containing estimated mean-squared errors for PLS models with 0:ncomp components. The first row of MSE contains mean-squared errors for the predictor variables in X, and the second row contains mean-squared errors for the response variable(s) in Y

[XL,YL,XS,YS,BETA,PCTVAR,MSE] = plsregress(...,param1,val1,param2,val2,...) specifies optional parameter name/value pairs from the following table to control the calculation of MSE.

f. Potential ISyE Applications

Principal component analysis is a very common use in regreghyssion analysis, a common procedure in ISYE. Typically, PCA considers regressing the dependent variable on a set of independent variables based on a standard linear regression model, but uses PCA for estimating the unknown regression coefficients in the model. In PCR, instead of regressing the dependent variable on the explanatory variables directly, the principal components of the explanatory variables are used as regressors. One typically uses only a subset of all the principal components for regression, thus making PCR a regularized procedure. Often the principal components with higher variances are selected as regressors. However, for the purpose of predicting the outcome, the principal components with low variances may also be important, in some cases even more important.

5. Sources

https://onlinecourses.science.psu.edu/stat505/node/138

https://stackoverflow.com/questions/5064928/difference-between-classification-and-clustering-in-data-mining
http://www.differencebetween.net/technology/difference-between-clustering-and-classification/
http://ieeexplore.ieee.org/document/6726842/
https://www.analyticsvidhya.com/blog/2015/07/dimension-reduction-methods/
https://msdn.microsoft.com/en-us/library/dn282355.aspx
http://www.real-statistics.com/multivariate-statistics/cluster-analysis/k-means-cluster-analysis/
http://web.cs.ucdavis.edu/~vemuri/classes/ecs271/lecture3.pdf
https://www.analyticsvidhya.com/blog/2016/03/practical-guide-principal-component-analysis-python/
https://www.statisticssolutions.com/principal-component-analysis-pca/