LVI Linear Discriminant Analyses

January 15, 2013

Prepared for:

Forest Analysis and Inventory Branch

BC Ministry of Forests, Lands and Natural Resource Operations

By

Ian Moss, PhD, RPF

Tesera Systems Inc.

Victoria, BC

LVI Linear Discriminant Analyses

Table of Contents

Prepared by: Ian Moss, Tesera Systems Inc.
January 15, 2013

LVI Linear Discriminant Analyses

1. Introduction

Assumes windows machine
All code tested using R 2.9.1
Assumes competency with computers and some low level familiarity with Python and R
R packages installed (see appendix)
All operating within an R working directory (Rwd)

Scripts
Data processing
Statistical analyses
Data visualization
Data interpretation (summary statistics)
Modular construction for ease of extension

Don't need to program in R
Can help if you want to learn how to program in R – some working code with documentation

2. Create an R working directory

1. Create an R working directory on your computer as follows: \\Rwd.
2. Copy the Rwd contents into the same directory you just established in the previous step; the Rwd directory should have three subdirectories: Rdata, Rdocs, and RScript.
3. Start up a new R session.
4. click **File**, **Open** Script at the top RHS of the R interpreter and navigate to the Rwd directory.
5. Navigate to the //Rwd// RScript file and open the SetRwd.R script file.
6. Change setwd("E:\\Rwd") to indicate the address of Rwd on your computer, e.g. setwd("C:\\Rwd"); note that you must use "\\" as subfile separators in the file address.
7. Click **Edit**, **Run all**.
8. A new file will now be stored in your Rwd directory called OpenSession.RData.
9. If you now close the R session, go to the Rwd directory, double click on OpenSession.RData a new R session will open up in the interpreter.
10. Now if you once again click on **File**, **Open** you will see that the interpreter automatically opens up in the Rwd directory.

3. Fuzzy C-Means Classification

1. Develop the data input file LVINEW (top of \\Rwd\\ directory).

2. Ensure that the data dictionary (PyRDataDict.csv in \\Rwd\\Python\\DATDICT\\ directory) is correctly filled out with variable names corresponding to those for LVINEW (and LVINORM if a normalized dataset is to be produced by subtracting all the continuous variable values from their respective means and dividing by the standard deviation) and with appropriate new variable names (NEWVARNAMES) and variable types. Note that the new unique key variable name should be identified as LVI_FCOID – these should all be integers.

3. Update XVARSELV1 with list of variables in LVINEW (and LVINORM). Indicate whether each variable is selected as a candidate X-variable (XVARSEL = X), or a Y-variable to be used in the Fuzzy C-Means clustering (XVARSEL = Y), or none of the above (XVARSEL = N).

4. Open FUZZYC_INITIALIZATION.csv to set the fuzzy c-means classification parameters. In particular select the desired classification routine (e.g. Euclidean distance by inputting FCM_E in row 4, FID = 3, ROUTINE, under column C, VARVALUE). Also enter the range of classifications to be developed using the Y-variable dataset (specified in step 3) by entering the lower limit number of classes (LLNCLASS, row 5, FID = 4; enter integer in row under column C, VARVALUE) and upper limit number of classes (ULNCLASS, row 6, FID = 5; enter integer in row under column C, VARVALUE). Note that ULNCLASS must be ≥ LLNCLASS. LLNCLASS must be ≥ 2. If ULNCLASS == LLNCLASS then 1 system of classification will be produced with the number of classes equal to LLNCLASS.

5. Open the Python interpreter (IDLE).

6. Go to **File … Open** and navigate to the \\Rwd\\Python\\ directory; click on LVI_CLASSIFY.py; Go to **Run … Run Module (F5)** to run the classification routine.

7. For each classification with N classes, the class assigned to each observation are written to FCLASS.csv (the file is overwritten) and the centroids associated with each of the selected Y-variables for each class in each system of classification are written to FCENTROID.csv .

8. The classes assigned to each of the observations should be manually transferred to LVINEW along with the corresponding labels (e.g. CLASS5 indicating a classification with 5 classes) and a corresponding update of variables in LVINEW to the data dictionary (PyRDataDict.csv) referred to in step 2 above. Updates are not required to be made in XVARSELV1, but one may do so.

4. Start with Variable Selection from a Large Number of Variables

1. Load Dataset

1.1. LoadDatasetAndAttachVariableNames.R (DATA in LVINEW.csv)

2. Select observation subset (Optional)

2.1. SelectObservationSubset.R (Optional)

Note that this is currently set to ensure that the bec zone (recorded in LVINEW under the variable name LVI_BECZ) is equal to a particular BEC_ZONE. In the Quesnel dataset this may be set to equal ‘SBPS’, ‘SBS’ or ‘MS’. Other variables and variable names may be applied. The script also reduces the original dataset down to a particular set of observations. If a different set of observations are desired then the process must be started from the beginning. Also issues may arise where certain zones or subzones do not have enough observations to support.

3. Declare (numerical) classification (Y-) variable as a “Factor”

3.1. DeclareClassificationVariableAsFactor.R

Note that this is an opportunity to change the classification Y-variable dataset – e.g. in the code where:

```
CLASSIFICATION = factor(CLASS5)
```

Change CLASS5 to CLASSS10 (I.e. one of the optional classifications in LVINEW) to indicate which system of classification you wish to analyse.

4. Select (independent) X variable subset

4.1. SelectXVariableSubset_v1 (primary option) (DATA in XVARSELV1.csv)
4.2. SelectXVariableSubset_v2 (DATA in XVARSELV2.csv)

Note that the second version (v2) is designed to accommodate multiple variable selection; however, when used in this procedure only header and the first row may be entered, otherwise only the last variable set in the list will be used in the next step. The first version (XVARSELV1.txt) was originally designed to identify the X variable superset from which specific variable sets would be extracted. However this could also be accomplished using the second version (XVARSEV2.txt) format.

(The first version is used currently as the starting point – it was also constructed manually and was intended for initial variable selection)

5. Load R library package, subseselect.

5.1. Loadsubselect-R-Package.R

(Note if you wish to view the data and make changes to it you can run the following script:

ViewLviNewDataset.R

This will open up a new screen and allow you to view the data. However, if this is deployed using a large dataset it may cause the interpreter to become non-responsive since it tries to write all of the data into the interpreter once the data editor is closed.

6. Run ldaHmat in subselect package

   6.1. RunLinearDiscriminantAnalysis_subselect_ldaHmat.R

7. Select chosen criteria and run ldaHmat variable selection routine

   7.1. Run-ldaHmat-VariableSelection-Improve.R

   Note that for variable selection you can select a range in the number of variables to be used in the model by changing the minimum (minNvar) and the maximum (maxNvar). You can also control the number of solutions that you would like to investigate for each number of variables (between minNvar and maxNvar, inclusive). Finally you can set different criteria for variable selection as follows:

   - Roy's first root statistic ("ccr12")
   - Wilks' Lamda ("Wilkes")
   - Chi squared ("x12")
   - And the Zeta 2 coefficient ("zeta2")

   Currently of the alternatives are listed in the script file. To deselect a choice put a **#** in front of the line (R recognizes these as comments). To select a criterion, remove the **#** sign from the front of the line. Note that if two of the criteria are selected, the last one listed (toward the bottom of the script) will be the one used.

   Note that in the process of testing this algorithm, the following error statement was encountered when the number of classes exceeded 16:

   Error in if (maxabssym > tolsym) { :

   missing value where TRUE/FALSE needed

   As a result the program does not complete properly. There may be an upper limit in the number of classes that can be handled using the subselect package improve function.

8. Extract all variable subsets derived from step 6 and put in data frame called SOLSUM (solution summary).

   8.1. ExtractVariableNameSubsets.R

9. Write SOLSUM from step 7 to VARSELECT.csv in R working directory

9.1. WriteDataframeToCsvFile.R

10. Python Code used here.

10.1. Using a Python module create a reformatted list of all of the unique combinations of variables and print it to a file called XVARSELV in the LVI directory; Run the following routine:

10.1.1. EXTRACT_RVARIABLE_COMBOS.py

OUTPUT (".csv" comma delimited files):

XVARSELV contains the list of unique combinations of variables developed from running the variable selection routine lda.Hmat.

UNIQUEVAR contains a unique list of variable names compiled from all variable sets produced in lda.Hmat.

Note that the output is assigned to the following directory:

"E\\Rwd\\"

5. Run Discriminant Analysis for a Single X-Variable Set

1. Load Dataset

1.1. LoadDatasetAndAttachVariableNames.R (DATA in LVINEW.txt)

2. Select observation subset

2.1. SelectObservationSubset.R (Optional)

Note that this is currently set to ensure that the bec zone (recorded in LVINEW under the variable name LVI_BECZ) is equal to a particular BEC_ZONE. In the Quesnel dataset this may be set to equal 'SBPS', 'SBS' or 'MS'. Other variables and variable names may be applied. The script also reduces the original dataset down to a particular set of observations. If a different set of observations are desired then the process must be started from the beginning. Also issues may arise where certain zones or subzones do not have enough observations to support.

3. Select and declare (numerical) classification variable as a "Factor"

3.1. DeclareClassificationVariableAsFactor.R

Note that this is an opportunity to change the classification Y-variable if there are a number of different classifications you would like to investigate.

4. Set prior classification distribution as being uniform or as per sample

   4.1. Uniform: ComputeUniformPriorClassificationProbabilityDistribution.R
   4.2. Sample: ComputeSamplePriorClassProbabilityDistribution.R

5. Select (independent) X variable subset

   5.1. SelecXVariableSubset_v1 (DATA in XVARSELV1.csv)
   5.2. SelectXVariableSubset_v2 (DATA in XVARSELV2.csv)

   These are loaded in the //Rwd//

   Note that the second version (v2) is designed to accommodate multiple variable selection; however, when used in this procedure only header and the first row may be entered, otherwise only the last variable set in the list will be used in the next step.

6. Load Discriminant Analysis R-package

   6.1. LoadMASS-R-Package.R

7. Run Linear Discriminant Analysis

   7.1. RunLinearDiscriminantAnalysis_MASS_lda.R
   7.2. RunLinearDiscriminantAnalysis_MASS_lda_TakeOneLeaveOne.R

   Notes:

   In *RunLinearDiscriminantAnalysis_MASS_lda.R* the Take-One-Leave-One option is disabled (CV = FALSE). As a result the following output is available following the discriminant analysis (note by typing the command, indicated in bold, the results can be printed out in the interpreter:

   lvi.lda$prior this produces the prior probability distribution (established in step 3 above) used to represent the distribution of observations amongst the classes (CLASSIFICATION).

   lvi.lda$counts the number of observations by class.

   lvi.lda$means the mean for each X-variable by class

   lvi.lda$scaling the discriminant functions are scaled so that the mean z-score for each function is 0. Note that this is equivalent to subtracting observed X-variable value from the mean for each of the variables and then multiplying by the discriminant functions.

lvi.lda$svd the ratio's of between to within-group standard deviations in the linear discriminant variables. These are also referred to as eigenvalues. The squares of these figures are the canonical F-statistics. When the squares of these figures are converted into proportions of the total – this is equivalent to the proportion of the total (Between-to-within) variance explained by each discriminant function.

lvi.lda$N is the number of observations contained in the dataset.

Within this same routine the following additional output is also available:

class.pred$class (the predict function) produces the class assignments to each observation (with all observations used in the discriminant analysis); also at the bottom of this output, the unique class names (or numbers) are listed.

class.pred$posterior the estimated posterior probability distributions based on the prior distribution calculated as follows (see Hora and Wilcox 1982, Dillon and Goldstein 1984 pp. 392 – 393.

class.pred$x the scores for each if the test cases associated with each variate (function)

class.table the (table function produces a) classification contingency table with the original class distribution in rows and the predicted class distribution in columns.

In *RunLinearDiscriminantAnalysis_MASS_lda_TakeOneLeaveOne.R* the Take-One-Leave-One option is enabled (CV =TRUE). The following output may be obtained:

lvi.lda$class the class assigned to each observation

lvi.lda$posterior the posterior probabilities developed using Take-One-Leave-One; these are superior to those not involving the Take-One-Leave-One process (Bates and Wilcox 1982; Dillon and Goldstein 1984, pp. 406-409).

class.table the (table function produces a) classification contingency table with the original class distribution in rows and the predicted class distribution in columns.

6. Run Linear Discriminant Analysis for Multiple X-Variable Sets

1. Load Dataset

   1.1. LoadDatasetAndAttachVariableNames.R (DATA in LVINEW.csv)

2. Select observation subset (Optional)

   2.1. SelectObservationSubset.R (Optional)

Note that this is currently set to ensure that the bec zone (recorded in LVINEW under the variable name LVI_BECZ) is equal to a particular BEC_ZONE. In the Quesnel dataset this may be set to equal 'SBPS', 'SBS' or 'MS'. Other variables and variable names may be applied. The script also reduces the original dataset down to a particular set of observations. If a different set of observations are desired then the process must be started from the beginning. Also issues may arise where certain zones or subzones do not have enough observations to support.

3. Declare (numerical) classification variable as a "Factor"

   3.1. DeclareClassificationVariableAsFactor.R

   Note that this is an opportunity to change the classification Y-variable if there are a number of different classifications you would like to investigate. If this process is used after having run process number 4 (Start with Variable Selection from a Large Number of Variables) then make sure that the correct (same) class label is selected as before.

4. Compute prior classification distribution as being uniform or as per sample

   4.1. Uniform: ComputeUniformPriorClassificationProbabilityDistribution.R
   4.2. Sample: ComputeSamplePriorClassProbabilityDistribution.R

5. Write prior distribution to file

   5.1. WritePriorDistributionToFile.R

   OUTPUT (.csv file)

   PRIOR Contains a list of classes (CLASS) and associated prior probabilities (PROIRD).

6. Load Discriminant Analysis R-package

   6.1. LoadMASS-R-Package.R

7. Select (independent) X variable subset

   7.1. SelectXVariableSubset_v2.1 (DATA in XVARSELV.csv)

8. Run (Multiple) Linear Discriminant Analysis for Multiple Sets of Variables – Take One Leave One

   8.1. RunMultipleLinearDiscriminantAnalysis_MASS_lda_TakeOneLeaveOne.R

   WARNING this is easily confused with another script that will not work in this context:

   - RunLinearDiscriminantAnalysis_MASS_lda_TakeOneLeaveOne.R

Note also that this routine uses the Take-One-Leave-One routine for the purpose of rating the quality of the variable sets in terms of their accuracies in classification based on producing contingency tables as 1 output, and in terms of the posterior estimation of error as another output.

Note that the posterior error of estimation is calculated following the procedures of Hora and Wilcox (1982; Equation 9):

$$\hat{e} \quad = 1 - \ N^{-1} \sum_{i=1}^{N} max\,[P(Y \ \ |X_i)] \qquad \text{Eq. 1}$$

Where,

$\hat{e}$ is the estimated (posterior) error

N Is the total number of observations

$P(Y \ \ |X_i)$ is the probability of class Y, where Y is equal to 1 to m classes, given a set of variables, X_i , where i equals 1 to n observations.

9. Write (Multiple Linear) Results from Step 6 to Files

9.1. WriteMultipleLinearDiscriminantAnalysis_MASS_lda_TOLO_File.R

Note this is easily confused with:

OUTPUT (.csv files)

CTABULATION: contains the contingency table data from which the Cohen's (1960) Coefficient of Agreement can be calculated.

POSTERROR: contains the errors estimated using Eq. 1 above.

10. Run (Multiple) Linear Discriminant Analysis for Multiple Sets of Variables – All Observations

10.1. RunMultipleLinearDiscriminantAnalysis_Mass_lda.R

Note this can be confused with RunLinearDiscriminantAnalysis_Mass_lda.R … missing the "Multiple."

11. Write Multiple Linear results from step 10 to files.

11.1. WriteMultipleLinearDiscriminantAnalysis_MASS_lda.R

OUTPUT (.csv files)

CTABALL: contains the contingency table data from which the Cohen's (1960) Coefficient of Agreement can be calculated.

VARMEANS: contains the mean values for each variable by class.

DFUNCT: contains the discriminant functions for each axis and combination of variables.

BWRATIO: contains the between-to-within variance ratio of the differences in class Z-statistics associated with each discriminant function (i.e. eignevalues).

12. Compile Take-One-Leave-One CTABULATION Classification Accuracy Statistics

12.1. COHENS_KHAT.py (Data in CTABULATION.csv)

Note that the input file was produced using step 7 in these procedures.

OUTPUT (.csv files in Rwd directory)

CTABSUM provides statistics as indicated in Table 2.

Table 2. A description of variables included in the output file: CTABSUM.

Variable	Name
VARSET	Variable Set
OA	Overall Accuracy
KHAT	Coefficient of Agreement
MINPA	Minimum Producer Accuracy
MAXPA	Maximum Producer Accuracy
MINUA	Minimum User Accuracy
MAXUA	Maximum User Accuracy

A variable set is associated with a given combination of different kinds and numbers of variables that as developed in steps 5, 6 and 7 above. The overall accuracy indicates the proportion of observations that were correctly classified according to the original (reference) classification. Cohen's coefficient of agreement is an indicator of the overall success rate after having removed the potential for a certain level of agreement to occur by chance.

The minimum producer accuracy indicates the minimum number of correctly classified observations given the total number of observations assigned to any given class, amongst all classes by way of discriminant analysis in this case, and expressed as a proportion. The maximum producer accuracy is similarly derived but with respect to the maximum. These figures are also related to the maximum and errors of omission amongst all of the classes (e.g. 1 – MINPA, and 1 – MAXPA).

The minimum user accuracy indicates the minimum number of correctly classified observations given the total number of observations as originally assigned to any given class, amongst all classes by way of discriminant analysis in this case, and expressed as a proportion. The maximum user accuracy is similarly derived but with respect to the maximum. These figures are also related to the maximum and errors of commission amongst all of the classes (e.g. 1 – MINPA, and 1 – MAXPA).

7. Produce Unique X-Variable Subset Correlation Matrix

Note that as a guideline you may wish to exclude any one variable in pairs with correlations > 0.7 (or < -0.7) a priori. The step of removing correlated variables may best be done

1. Load dataset

   1.1. LoadDatasetAndAttachVariableNames.R (DATA in LVINEW.csv)

2. Select observation subset (Optional)

   2.1. SelectObservationSubset.R (Optional)

   Note that this is currently set to ensure that the bec zone (recorded in LVINEW under the variable name LVI_BECZ) is equal to a particular BEC_ZONE. In the Quesnel dataset this may be set to equal 'SBPS', 'SBS' or 'MS'. Other variables and variable names may be applied. The script also reduces the original dataset down to a particular set of observations. If a different set of observations are desired then the process must be started from the beginning. Also issues may arise where certain zones or subzones do not have enough observations to support.

3. Select unique X variable subset

   3.1. SelectUniqueXVariableSubset.R (DATA in UNIQUEVAR.csv)

4. Compile Unique Variable Correlation Matrix

   4.1. CompileUniqueXVariableCorrelationMatrixSubset.R

5. Create a unique variable correlation matrix file for printing

   5.1. CreateUniqueVarCorrelationMatrixFileForPrinting.R

6. Select new X variable subset (v2.1;This must be included prior to next step)

   6.1. SelectXVariableSubset_v2.1.R (DATA in XVARSELV21.csv)

7. Add variable subset indicators to correlation matrix file if option 5.1 selected

   7.1. AddVariableSubsetIndicatorsToCorrelationMatrix.R

   Note that this routine labels each variable indicator set as I1, I2, I3 … in the order that they are produced in step 5.

8. Write unique variable correlation matrix to a file

   8.1. WriteUniqueVarCorrelationMatrix.R

   OUTUPT (.csv file)

   UCORCOEF: This produces a table of correlation coefficients and indicator variables for each variable subset with 1's assigned to variable pairs that exist in the subset, and 0's assigned to all other variable pairs.

   MINMAXCOR: This is a compilation of the maximum and minimum correlations within each variable set across all variable pairs (excluding identical pairs for which the correlations are 1) within each variable subset. This is one criterion that can be used to select preferred sets of variables.

8. Produce Original Classification Unique X- or Y-Variable Subset Box and Scatter Plots

1. Load dataset

   1.1. LoadDatasetAndAttachVariableNames.R (DATA in LVINEW.csv)

2. Select observation subset (Optional)

   2.1. SelectObservationSubset.R (Optional)

   Note that this is currently set to ensure that the bec zone (recorded in LVINEW under the variable name LVI_BECZ) is equal to a particular BEC_ZONE. In the Quesnel dataset this may be set to equal 'SBPS', 'SBS' or 'MS'. Other variables and variable names may be applied. The script also reduces the original dataset down to a particular set of observations. If a different set of observations are desired then the process must be started from the beginning. Also issues may arise where certain zones or subzones do not have enough observations to support.

3. Declare (numerical) classification (Y-) variable as a "Factor"

   3.1. DeclareClassificationVariableAsFactor.R

(Interaction of User)

4. Select unique X or Y variable subset

   4.1. SelectUniqueXVariableSubset.R (DATA in UNIQUEVAR.csv)
   4.2. SelectXVariableSubset_v1.R (DATA in XVARSELV1.csv)
   4.3. SelectYVariableSubset_v1.R (DATA in XVARSELV1.csv)

5. Load R package – graphics

   5.1. LoadGraphics-R-Package.R

6. Run the box plot script

   If X variables have been selected then use:

   6.1. CreateUniqueVariableClassificationBoxPlots.R

   Else Y-variables were selected; use:

   6.2. CreateYVariableClassificationBoxPlots.R

   Note that you must right-click on each graph to proceed to the next graph. The first graph will be blank.

7. Run the scatter plot script

   7.1. CreateUniqueVariableScatterPlots.R

   Note that this can produce a lot of graphs. Specifically if there a total of K unique variables, then (k*(K-1))/2 graphs will be produced. For 20 variables that is equal to 190 graphs. Having said that you can end the graphics session at any time, by closing the graphics box (right clicking on the "X" button in the top right hand corner of the window).

   The figures illustrate the scatter of observations for y-axis versus x-axis variables. A regression line (red, y vs. x) and a lowess line (blue; a locally weighted polynomial regression line – similar to a moving average) are also shown on each figure to facilitate interpretation of the trends.

8. Produce X vs. Y variable scattergrams

   If X variables have already been selected then select Y-variables:

   8.1. SelectYVariableSubset_v1.R (DATA in XVARSELV1)

   Else Y variables previously selected; Select X-variables:

8.2. SelectUniqueXVariableSubset.R (DATA in UNIQUEVAR.csv)
8.3. SelectXVariableSubset_v1.R (DATA in XVARSELV1.csv)

Then run the following script:

8.4. CreateYvXVariableScatterPlot.R

9. Combine Evaluation Datasets

Use a Python program, COMBINE_EVALUATION_DATASETS.py to combine the following datasets for purposes of overall assessment (OUTPUT: ASSESS.csv):

Table 3. A list of files combined into one file: ASSESS.csv.

Input	Description
PyRDataDict.csv	This contains the data types (e.g. string, float, integer) associated with each of the input tables, except XVARSELV
MINMAXCOR	This contains the minimum and maximum correlation coefficients amongst all pairs of variables contained within a variable set.
CTABSUM	See Table 2.
POSTERIOR	See Eq. 1 above.
XVARSELV	The actual variable sets produced during the variable selection process.

10. Alternative Variable Selection Procedure (Under Development)

1. Load dataset

    1.1. LoadDatasetAndAttachVariableNames.R (DATA in LVINEW.csv)

2. Select observation subset (Optional)

    2.1. SelectObservationSubset.R

    Note that this is currently set to ensure that the bec zone (recorded in LVINEW under the variable name LVI_BECZ) is equal to a particular BEC_ZONE. In the Quesnel dataset this may be set to equal ‘SBPS’, ‘SBS’ or ‘MS’. Other variables and variable names may be applied. The script also reduces the original dataset down to a particular set of observations. If a different set of observations are desired then the process must be started from the beginning. Also issues may arise where certain zones or subzones do not have enough observations to support.

3. Declare (numerical) classification (Y-) variable as a “Factor”

3.1. DeclareClassificationVariableAsFactor.R

4. Set prior classification distribution as being uniform or as per sample

4.1. Uniform: ComputeUniformPriorClassificationProbabilityDistribution.R
4.2. Sample: ComputeSamplePriorClassProbabilityDistribution.R

5. Select X variable subset (v1)

5.1. SelectXVariableSubset_v1.R

6. Load klaR Package

6.1. LoadklaR-R-Package.R

7. Load combinat Package

7.1. LoadCombinat-R-Package.R

8. Run pairwise class variable selection

8.1. RunLinearDiscriminantAnalysis_klaR_pvs.R

Note that this routine uses linear dsicriminant analysis (or alternatives such as quadratic or reduced discriminant procedures – making it more flexible than the subselect package which only provides for the standard linear discriminant analysis procedure). The procedure then compares each possible pair of classes in turn (using the pvs command in klaR) and selects the best variable sets according to certain criteria. The basic criteria are “stepclass” (forward, backward, or both), “ks.test” (Kolmogorov-Smirnov test) , or “greedy.wilks” (Wilks’ lamda). This procedure could be applied within a bootstrap procedure to generate multiple variable sets for further testing.

11. Calculate Z-Scores, Nearest Neighbours, and Root Mean Squared Errors associated with Y-variables for Reference Dataset Using Discriminant Functions

1. Apply the discriminant functions to the reference dataset

1.1. Run NN_ZSCORE.py (DATA in LVINEW.csv, DFUNCT.csv, BWRATIO.csv)

Note that this routine can take a long time to run. A summary of analysis steps is as follows:

1.1.1. For each variable set, compute the Z-scores for each observation for each of the associated discriminant functions. Note that these functions were derived from the reference dataset; bootstrapping has not been invoked; nor a take-one-leave-one strategy.

1.1.2. Normalize the Z-scores (subtract from the mean and divide by the standard deviation for each function).

1.1.3. An option is then provided to weight the results based on the proportion of between-to-within variance explained by each discriminant function associated with a variable set. This option is embedded in the program for identifying nearest neighbours. However preliminary tests of this option indicate no differences in the results when using mahalanobis distances primarily because this distance metric immediately restores the weight assigned in proportion to the inverse of the covariance matrix. Furthermore the mahalanobis distances tended to consistently produce the lowest root mean squared errors associated with nearest neighbours when compared with the use of absolute or Euclidean distances. In this case all of the observations were used to derive the discriminant function but each observation was then, in turn, excluded from having itself identified as being the nearest neighbor. As a consequence the current default is set so that the Z-values assigned by the discriminant function are given equal weight in determining the nearest neighbor. Under this scenario Euclidean and Mahalanobis distances produce equivalent results; absolute differences are generally associated with higher nearest neighbor root mean squared errors. The option is always there to test this further with other datasets.

1.1.4. Identify nearest neighbours for each reference observation (excluding identification of self) and calculate root mean squared errors for the associated Y-variable set.

2. Run nearest neighbour analysis using selected X-Variables: absolute difference, Euclidean Distance, Mahalanobis Distance.

3. Run nearest neighbour analysis using selected Y-Variables: absolute difference, Euclidean Distance, Mahalanobis Distance.

Remaining on the list:

(Need to do)

1. Nearest Neighbour and RMSE Statistics (Both NN Classification with Centroids; pre and post Discriminant Classifications – and complete NN RMSE Statistics)
    1.1. Euclidean
    1.2. MSN (Mahalanobis)

(Like to do)

2. Classification of Target Dataset Using desired Prior and Posterior Distributions)
3. Apply Results to Target Dataset
4. Bootstrap Analyses
5. Use the concept of tolerance to eliminate collinearity – i.e. regress each X-variable on the remaining variables and remove those with a tolerance <= a given threshold. Do so in order of least to most significant variables based on an ANOVA. This would be a useful screening tool to eliminate issues with Multicollinearity
6. Extend process to quadratic discriminant analysis and reduced discriminant analysis (use klaR package – a start has been made on this)
7. For graphical analysis – get unique pairs of variables used in equations – eliminate pairs that don't actually occur in any one equation (reduces number of graphs).
8. Determine optimal number of classes in classification given Y-variables (used to develop the classification and given X-variables.
9. Select Observation Subsets based on certain criteria.
10. Output actual versus predicted class for each observation for further graphical analysis
11. Optimizing ROC thresholds for purposes of classification.

References

Cohen, J. 1960. A Coefficient of Agreement for Nominal Scales. Educational and Psychological Measurement XX(1): 37 – 46.

Dillon, W.R., and Goldstein, M. 1984. *Multivariate analysis.* Methods and applications. John Wiley & Sons, New York, NY, US.

Hora, S.C. and Wilcox, W.B. 1982. Estimation of error rates in several-population discriminant analysis. *Journal of Marketing Research* 19(1):57-61.

Additional References

Birkal, D. 2006. Regularized Discriminant Aanalysis. Lecture Notes. http://www.uni-leipzig.de/~strimmer/lab/courses/ss06/seminar/slides/daniela-2x4.pdf [accessed Dec 19 2012]

Davis, H.Z., Mesznik, R. and Lee, J.Y. 2009. Finding an internal optimum in the classification of management accounting information: The role of fuzzy sets. Management Accounting 17:203-216.

Liu, H. and Yu, L. 2005. Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering* 17(4):491-502.

Appendix: A listing of Python Modules and R Scripts and R Working Directory Structure

Rwd Working Directory Structure

This is a dictionary created to maintain LVI data management processes. It is created separately from the location of the R interface, packages, etc.

- 1. E:
- 1.1. Rwd
  - 1.1.2. Python
    - 1.1.2.1. Admin
    - 1.1.2.2. DATDICT
      - 1.1.2.2.1. PyRDataDict.csv
    - 1.1.2.3. PyReadError
  - 1.1.3. Rdata
    - 1.1.3.1. Archived
  - 1.1.4. Rdocs
    - 1.1.4.1. LOGISTIC
    - 1.1.4.2. MANOVA
    - 1.1.4.3. MDA
    - 1.1.4.4. Packages
    - 1.1.4.5. R-Language
  - 1.1.5. RScript
    - 1.1.5.1. Additional

R-packages

1. MASS
2. subselect
3. klaR
4. combinat

Note: to install a new packages you can use the following script:

installPackages.R

Enter the name of the package in quotation marks after the “packageName =” command and then run the script.

Python Module in the Main Python Directory (\\Python26\\)

1. routineLviApplications.py

Python Module Library (\\Rwd\\Python\\)

1. COHENS_KHAT.py

2. COMBINE_EVALUATION_DATASETS.py
3. EXTRACT_RVARIABLE_COMBOS.py
4. LVI_CLASSIFY.py

Python Module Admin (Support) Files (\\Rwd\\Python\\Admin\\:

1. addToDataMachette20121026.py
2. BASE.py
3. dictionaryDBUtilities.py
4. fileUtilities.py
5. fuzzyC_v2.py
6. innerJoinDictDBs.py
7. makeDD.py
8. MAN.py
9. MATRIX_MAN.py
10. PRINTv1.py
11. readCSV.py
12. typeDataset.py

Python Data Dictionary (\\Rwd\\Python\\DATDICT\\)

1. PyRDataDict.csv

Essential Applications in Python Site Packages (\\Python26\\Lib\\site-packages\\)

1. routineLviApplications.py

   Note that this module must be placed in the top of the site packages directory. When installing the program the routine must be opened up and the default directories set to correctly identify the R working directory locations and structure. The directory structure below the R working directory (Rwd) should remain unchanged.

2. numpy (A Python package to be down loaded from the internet).

3. scipy (A Python package to be down loaded from the internet).

R Script Library (\\Rwd\\RScript\\)

1. AddVariableSubsetIndicatorsToCorrelationMatrix.R
2. CompileUniqueVariableCorrelationMatrixSubset.R
3. ComputeSamplePriorClassProbabilityDistribution.R
4. ComputeUniformPriorClassificationProbabilityDistribution.R
5. CreateUniqueVarCorrelationMatrixFileForPrinting.R
6. CreateUniqueVariableClassificationBoxPlots.R
7. CreateUniqueVariableScatterPlots.R
8. CreateYVariableClassificationBoxPlots.R
9. CreateYvXVariableScatterPlots.R
10. DeclareClassificationVariableAsFactor.R
11. ExtractVariableNameSubsets.R
12. installPackages.R
13. LoadCombinat-R-Package.R
14. LoadDatasetAndAttachVariableNames.R
15. LoadGraphics-R-Package.R
16. LoadklaR-R-Package.R
17. LoadMASS-R-Package.R
18. Loadsubselect-R-Package.R
19. Run-ldaHmat-VariableSelection-Improve.R
20. RunLinearDiscriminantAnalysis_klaR_pvs.R
21. RunLinearDiscriminantAnalysis_MASS_lda.R
22. RunLinearDiscriminantAnalysis_MASS_lda_TakeOneLeaveOne.R
23. RunLinearDiscriminantAnalysis_subselect_ldaHmat.R
24. RunMultipleLinearDiscriminantAnalysis_Mass_lda.R
25. RunMultipleLinearDiscriminantAnalysis_MASS_lda_TakeOneLeaveOne.R
26. SelectUniqueXVariableSubset.R
27. SelectObservationSubset.R
28. SelectXVariableSubset_v2.1.R
29. SelectXVariableSubset_v2.R
30. SelecXVariableSubset_v1.R
31. SelectYVariableSubset_v1.R
32. SetRwd.R
33. ViewLviNewDataset.R
34. WriteDataframeToCsvFile.R
35. WriteMultipleLinearDiscriminantAnalysis_MASS_lda_to_File.R
36. WriteMultipleLinearDiscriminantAnalysis_MASS_lda_TOLO_to_File.R
37. WritePriorDistributionToFile.R
38. WriteUniqueVarCorrelationMatrix.R

The data (.csv) files used as input and output:

1. OpenSession.RData (after it has initially been created following procedure number 1 above)

2. Output (.csv) files in the Rwd directory:

ID	File	Source	Description
1	ASSESS	Python	File with summary statistics for assessment of results from modeling
2	BWRATIO	R	Ratios: Square root of between to within variance for each discriminant function
3	CTABALL	R	Cross tabulation of results using all data in model calibration.
4	CTABSUM	Python	Cross tabulation statistic summary derived from CTABULATION
5	CTABULATION	R	Cross tabulation results using Take-OneLeave-One
6	DFUNCT	R	Discriminant functions
7	FCENTROID	Python	Centroids associated with each classification and set of selected variables
8	FCLASS	Python	Class assignments to each observation in LVINEW
9	FUZZYC_INITIALIZATION	Manual	User defined inputs to developing classification
10	LVINEW	Manual	User defined base data including X, Y and CLASSIFICATION variables
11	LVINORM	Manual	User defined normalized (mean and standard deviation) data (continuous variables only) derived from LVINEW
12	MINMAXCORR	R	Minimum and Maximum Correlations within each X variable set
13	POSTERIOR	R	Error of estimation using posterior probabilities (see Hora and Wilcox 1982)
14	PRIOR	R	Prior probability selected by user
15	PyDataDict	Python	This is a special file in the \\Rwd\\Python\\DATDICT\\ used to control how data is brought into the Python environment and transformed for further analyses (i.e. declared as a string, integer, or float value).
16	UCORCOEF	R	Correlation Coefficients amongst variables in UNIQUEVAR
17	UNIQUEVAR	Python	Unique variable set associated with variables listed in VARSELECT
18	VARMEANS	R	Variable means by original CLASS assignments
19	VARSELECT	R	String table describing variable sets used to derive XVARSELV
20	XVARSELV	Python	Variable sets used as further input into discriminant analysis
21	XVARSELV1	Manual	Initial file for selecting eligible X (and Y) variable datasets for use in analysis
22	XVARSELV2	Manual	Same format as XVARSELV but with only 1 variable set
23	ZCOV	Python	Contains the covariance matrix amongst the discriminant scores for each function associated with normalized Z-scores

			for each variable set; the inverse of the covariance matrix is used in calculating Mahalanobis distances. Note that this tends to be a diagonal matrix (i.e. 1's on the diagonal and 0's in the off diagonal.
24	ZERROR	Python	Squared errors in Y-variable estimation associated with each variable set and (Target or Test) observation. Note that this is not a take-one leave-one process since all of the observations were used in this instance to derive the discriminant functions.
25	ZNNDICT	Python	A list of nearest neighbours for each variable set, excluding the reference observation for different distance measures (absolute, Euclidean, and mahalanobis differences).
26	ZNSCORE	Python	Normalized discriminant scores (subtracted from the mean and divided by the standard deviation for each discriminant function).
27	ZSCORE	Python	Un-normalized discriminant scores for each observation associated with a variable set – discriminant function combination.
28	ZRMSE	Python	A summary of the root mean squared errors associated with the nearest neighbor Y-variables according to nearest neighbours identified according to each discriminant variable set.
29	ZSTAT	Python	Mean and standard deviation in Z-values for each variable set – discriminant function combination.

Appendix: B: Standard LVI Analysis Process

1. Prepare LVINEW dataset and update PyRDtataDict.csv with new attributes; ensure that key variable name is labeled 'LVI_FCOID'.

2. Make sure PyRDataDict.csv is in the \\Rwd\\Python\\DATDICT\\ directory and that the variable names associated with LVINEW are correctly identified.

3. Put the list of variables (minus the Key Variable) in LVINEW into VARSELV1.csv. Select the candidate X variables by entering an X next to those selected variables. Select the desired Y variables (variables of interest that are to be estimated in some way using the X-variable set) by entering a Y next to those variables. Enter N beside any remaining variables that are not to be included in the analysis. At minimum the desired X and Y variables must be included in the list with the appropriate entries. LVI_FCOID may or may not be included; if it is included it must have an N entered next to it indicating that it is not a Y or X variable of interest.

4. Make sure PyRDataDict.csv is in the \\Rwd\\Python\\DATDICT\\ directory and that the variable names associated with VARSEV1 are correctly identified.

5. Check to make sure the following files are in the Rwd Directory:
    - FUZZY_C_INITIALIZATION.csv
    - LVINEW.csv
    - OpenSession.RData
    - XVARSELV1.csv with indication of selected Y-variables and selected X-Variable candidates

6. Start with process number 3: Fuzzy C-means Classification

6.1. Open FUZZY_C_INITIALIZATION .csv and set initial parameters, particularly the range of classifications to be produced in terms of number of classes.

6.2. Run LVI_CLASSIFY.py

- The following files will be added to the Rwd directory:
    - FCLASS. Csv
    - FCENTROID.csv

6.3. Add the classifications and associated labels identified in FCLASS.csv to the LVINEW.csv dataset including the classification labels. Make sure that the assignments are based on matching LVI_FCOID.

6.4. Update the classification filed names associated with LVINEW in the PySDataDict.csv file.

7. Start with Variable Selection

7.1. Open R-session

7.1.1. OpenSession.RData

Click on this in the Rwd directory

7.2. Implement procedure number 4 (Start with Variable Selection from a Large Number of Variables) above

7.2.1. Use XVARSELV1.csv in the process

- At the end of the R part of the session the following file will be added to the Rwd directory
    - OpenSession.RData
- At the end of the Python part of the session the following files will be added to the Rwd directory:
    - XVARSELV.csv
    - UNIQUEVAR.csv

7.3. Close R session – do not save the results

8. Run process number 7 (Produce Unique X-Variable Subset Correlation Matrix)

8.1. Open R Session

8.1.1. OpenSession.RData

- At the end of the R session the following files will have been added to the Rwd directory:
    - MINMAXCOR
    - UCORCOEF

Check UCORCOEF for variables with correlations ≤ - 0.8 or ≥ 0.8. Select the preferred variable associated with these pairs of variables to be used in the analysis. Use process number 8 (Produce Original Classification Unique X- or Y-Variable Subset Box and Scatter Plots) to help with this by comparing variable pairs and selecting the one that seems produce the greatest differentiation amongst the classes, while excluding the others. Start by selecting the best x- variable and then eliminate any variables highly correlated with that variable. Then select the next best available variable and repeat this process until all of the high correlations are removed. For those variables that are to be removed go back into XVARSELV1 and change the corresponding variable names with XVARSEL = X to XVARSEL = N instead. Then repeat stem 7 in this overall procedure.

9. Run process number 6 (Run Linear Discriminant Analysis for Multiple X-Variable Sets)

9.1. Open R Session

9.1.1. OpenSession.RData

- At the end of the R part of the session the following files will have been added to the Rwd directory:
  - BWRATIO.csv
  - CTABALL.csv
  - CTABULATION.csv
  - DFUNCT.csv
  - POSTERIOR.csv
  - PRIOR.csv
  - VARMEANS.csv
- At the end of running the Python part of the session the following files have been added to the Rwd directory:
  - CTABSUM.csv

10. Run process number 9 (Combine Evaluation Datasets)

10.1. Python Session

- At the end of running the Python part of the session the following files have been added to the Rwd directory:
  - ASSESS.csv

11. Run nearest neighbor analysis

11.1