Preliminary Report on MDA work

Dan Beatty

May 28, 2007

The first part of the assignment was

> You are given a total of 150 four dimensional samples of three Iris species. Use Multiple Discriminant Analysis (MDA) to separate the samples into three Iris species. Show the resulting misclassifications. Repeat MDA choosing any three-dimensional features at a time and show the resulting misclassifications.

The Mathematica version is long mostly from listing the samples. Other than that, the steps mimic the algorithm for MDA very closely. Once the structure allows for a classifier to is constructed on the assumption that the data is Gaussian.

```
barSetosa = Mean[setosa]
barSetosaArray = PadLeft[{barSetosa}, First[Dimensions[setosa]], {barSetosa}]
normSetosa = setosa - barSetosaArray
ScatterSetosa = Transpose[normSetosa].normSetosa/First[Dimensions[setosa]]

barVersicolor = Mean[versicolor]
barVersicolorArray = PadLeft[{barVersicolor}, First[Dimensions[
  versicolor]], {barVersicolor}]

normVersicolor = versicolor - barVersicolorArray
ScatterVeriscolor =
        Transpose[normVersicolor].normVersicolor/First[Dimensions[versicolor]\
]

barVirginica = Mean[virginica]
barVirginicaArray = PadLeft[{barVirginica}, First[Dimensions[virginica]], \
{barVirginica}]
normVirginica = virginica - barVirginicaArray
ScatterVirginica = \
Transpose[normVirginica].normVirginica/First[Dimensions[virginica]]
whiteningScatter = ScatterVirginica + ScatterSetosa + ScatterVeriscolor
```

```
flowers = Join[setosa, versicolor, virginica]
combinedMean = Mean[flowers]
flowers = Join[setosa, versicolor, virginica]
combinedMean = Mean[flowers]
meanArray = Join[{barSetosa}, {barVersicolor}, {barVirginica}]
combineMeanArray = PadLeft[{combinedMean}, Dimensions[meanArray], \
{combinedMean}]
meanDifference = (meanArray - combineMeanArray)
scatterBackground = Transpose[meanDifference] .(50*meanDifference)
{lambda, W} = Eigensystem [{scatterBackground, whiteningScatter}, 2]
ListPlot[flowers.Transpose[W]]
```

0.1 Explanation of the Mathematica Code

In the Mathematica example, we use setosa, viriginica, and versicolor as variables containing the original measurements. We next construct an row matrix contain at each of its rows, called barSetosaArray. This array is used to generate $\vec{x_i} - \mu$ for row vector $\vec{x_i}$ in the setosa collection, misnomered as normSetosa. This allows mean to generate the scatter matrix for setosa.

The same thing was done for versicolor and virginica. As stated in the MDA algorithm, $\mathbf{S_w} = \sum_i S_i$ where S_i are the scatter matrices for each of the classes. Here also, we combine the flowers using the Join method in Mathematic, and take a mean of the combined whole. From there, we use the mean array concept to reduce finding the background scatter to matrix addition, scalar multiplication, matrix multiplication and a transpose.

Fortunately, Mathematica has an intuitive method for computing eigenvectors of a generalized system as below.

$$\mathbf{A}\vec{e_i} = \lambda_i \mathbf{B}\vec{e_i}$$

Thus mapping the original data source via the $\mathbf{W}$ obtained is simply a matter of matrix multiplication

0.2 Plot Results from Mathematica

1 Objective-C Version of the MDA Classifier

In order to satisfy the Multiple Discriminant Analysis (MDA) section of the problem, this solution has taken on two forms. One, a solution in Mathematica, solve the preconditioner by hand and shows demonstration of concept. The other is an automated library to be service that can be used in general. Lessons from the Mathematica demonstration provides means for the unit tests of the object class. For the sake of this discussion data class shall refer set for classifying the data into. Object class shall refer to the Objective-C structure used to construct objects.

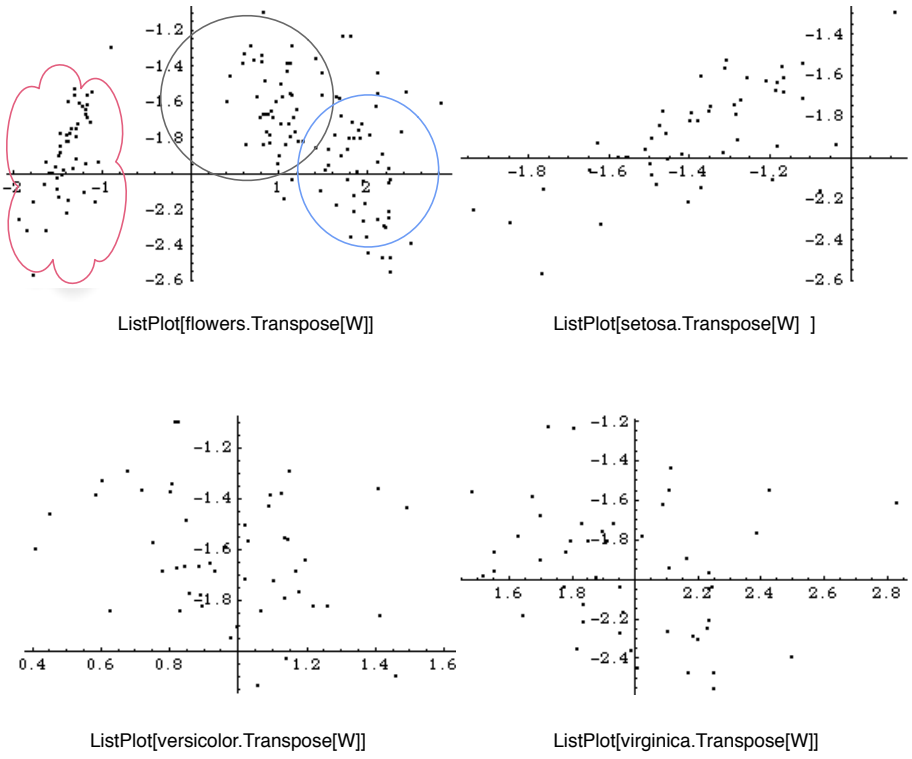

Figure 1: mathematica Plots using a manual MDA

The classic definition of Multiple Discriminant Analysis is defined by Algorithm 1. In this definition, D_i are the data classes, $\vec{m}_i$ are the sample mean of these data classes, n_i are the number of samples in these data classes, $\vec{m}_t$ is the total mean for the samples of all of the data classes, and

Algorithm 1 Multiple Discriminant Analysis

Determine $\vec{m}_t$
for all Classes D_i in Discriminant Set D **do**
  Compute $\vec{m}_i$
  Determine n_i
  Determine $\hat{m}_i = \vec{m}_i - \vec{m}_t$
  Compute $S_i = \sum_{\vec{x_i} \in D_i} (\vec{x_i} - \vec{m_i})(\vec{x_i} - \vec{m_i})^T$
end for
$S_w = \sum_{S_i \in D} S_i$
Compute $S_B = \sum_{\hat{m_i} \in D} n_i \hat{m}_i$
Compute Top eigenvectors for equation:

$$\mathbf{S_B w_i} = \lambda_i \mathbf{S_W w_i}$$

return $\mathbf{W}, \mathbf{\Lambda}$

The MDA is all about conditioning the data to one less component than there is a data class. The components are usually arranged as the columns of a matrix, $\mathbf{W}$. If the original data is arranged as row vectors in a matrix $\mathbf{X}$, then

$$\mathbf{XW} = \mathbf{Y} \tag{1}$$

where $\mathbf{Y}$ is a matrix the same number of rows, m, as sample matrix and $c-1$ columns where c are the number of data classes. Each of the columns are mappings onto an ortho-normal basis.

The Objective-C version has two constructors. One takes an array of "sample classes" which are simply structures containing the original samples, the mean, covariance, and scatter matrix. The other takes a matrix containing the samples and a vector containing a enumerated mapping of row vector to class. In the second, the "sample classes" are constructed so that the preconditioning matrix W may be computed. The preconditioning matrix W is defining feature of the object class, and should be the thing requested from the structure.

In order to construct the preconditioning matrix, there are two sets of methods based on data classes (sample classes) to generate the matrices required. These two matrices are the background scatter matrix ($\mathbf{S_b}$) and whitening scatter matrix ($\mathbf{S_w}$).

Compute Whitening Scatter is derived on the basis that a data class structure produces a scatter matrix on initialization. Thus computing whitening scatter matrix is merely an

addition of all of the scatter matrices.

$$\mathbf{S_w} = \sum_i \mathbf{S_i} \tag{2}$$

In order to construct the background scatter matrix, one needs a few supplemental matrices. Most mathematical definitions of MDA do not state this explicitly, yet it is a necessary step for efficiency.

The computation for the scatter matrices used to construct the pre-conditioner is as follows:

- Number of Samples Vector $\vec{n}$
- Number of Samples Matrix $\mathbf{N}$
- Mean Transformation Matrix $\mathbf{M}$

The background scatter matrix is defined from these structures as:

$$\mathbf{S_b} = (\mathbf{N} \cdot \mathbf{M})(\mathbf{M}^T) \tag{3}$$

where the cdot indicates a dot-Multiply between $\mathbf{N}$ and $\mathbf{M}$.

Real trick is how the eigenvectors are computed for S_w and S_b. In this case, it is convenient to note the characteristic equation being solved.

$$\mathbf{S_B}\vec{e_i} = \lambda_i \mathbf{S_w}\vec{e_i} \tag{4}$$
$$\mathbf{S_w}^{-1}\mathbf{S_B}\vec{e_i} = \lambda_i \mathbf{S_w}^{-1}\mathbf{S_w}\vec{e_i} \tag{5}$$
$$\mathbf{S_w}^{-1}\mathbf{S_B}\vec{e_i} = \lambda_i \mathbf{I}\vec{e_i} \tag{6}$$
$$\mathbf{S_w}^{-1}\mathbf{S_B}\vec{e_i} = \lambda_i \vec{e_i} \tag{7}$$

Most would argue that solving equation 4 would be the proper choice. However, the Objective-C version does not have such a method. It can compute inverses of square matrices, and eigenvectors of the characteristic equation 8 which is the same as equation 7.

$$\mathbf{A}\vec{e_i} = \lambda_i \vec{e_i} \tag{8}$$

2 Plotting Libraries

One library necessary for plotting these results come from the CGContext and CGContextArcToPoint. These are presented in Laden and Gelphman chapter 5 and page 57.