Pattern Recognition
Exercise 4 Report

Piotr Latusek, #212613
latusekpiotr@gmail.com

***Abstract* — This document covers the topic of feature extraction and projections. Two important techniques for analyzing data are described – Principal Component Analysis and Multiple Discriminant Analysis. There are also some examples presented of how they can be used in practice for feature extraction.**

I. Introduction

A feature space which has too many dimensions is usually difficult to work with. In this case, it is sometimes possible to reduce the number of features without losing performance of the classifier (or even enhancing it). The easiest way to do this is to combine features together using their linear combinations.

There are two different algorithms which can be used to decide how to combine the features. Principal Component Analysis tries to find a feature set which describes the samples within one class in an optimal way. The second algorithm, Multiple Discriminant Analysis seeks for a feature set which allows us to discriminate between different classes as easy as possible.

II. Component Analysis and Discriminants

A. Principal Component Analysis (PCA)

The aim of PCA is to find a set of unit vectors $\mathbf{e}_1$, $\mathbf{e}_2$, ..., $\mathbf{e}_d$ (d being the desired dimensionality), which can be used as new axes i.e. samples will be projected on them to get a new set of features. PCA tries to find these vectors in such a way, that a new set of features will be describing samples from one class in an optimal way.

Let us assume the mean value **m** being:

$$m = \frac{1}{n}\sum_{k=1}^{n} x_k$$

where n is a number of training samples in a set **X**. Our approximation of any sample **x** is then:

$$x = m + \sum_{i=1}^{d} a_i e_i$$

where a_i are the values of a new features corresponding to respective axes $\boldsymbol{e_i}$. a_i can be also seen as a distance of a projection of the sample on the ax $\boldsymbol{e_i}$. Having these assumptions, to find vectors $\boldsymbol{e_i}$, PCA minimizes the criterion function of that form:

$$J_d = \sum_{k=1}^{n} \left\| \left(m + \sum_{i=1}^{d} a_{ki} e_i \right) - x_k \right\|^2$$

with respect to $\boldsymbol{e_i}$. *J* can be interpreted as a sum of squared distances between the original sample and our approximation of it. After employing some mathematics, $\boldsymbol{e}_i$ vectors are found to be first *d* eigenvectors of a scatter matrix (scaled covariance matrix) with the biggest eigenvalues corresponding to them.

A. Multiple Discriminant Analysis (MDA)

While PCA looks for components which can describe the data from one class in an optimal way, these components might not be necessarily good for discriminating between two or more different classes. Such components can be found with a use of MDA algorithm.

To project the samples from *d*-dimensional to (*c-1*)-dimensional space, we will use the equation:

$$y = W^t x$$

where ***W*** is a *(c-1)* x *d* matrix of coefficients which we will be looking for and **x** is a sample we want to project. We will use the same mean value **m** in this algorithm. Furthermore, we need to define some extra values:

$$\tilde{S}_i = \sum_{x \in D_i} (y - \tilde{m}_i)(y - \tilde{m}_i)^t$$

$$\widetilde{S_W} = \sum_{i=1}^{c} \widetilde{S_i}$$

where D_i is the set of samples which belong to class i and $\widetilde{m_i}$ is a mean value of all the samples belonging to that class. We call $\widetilde{S_i}$ within-class scatter matrix. $\widetilde{S_W}$ is a sum of scatter matrices of all c classes and shows how much the samples are spread within classes. We also need to define between-class scatter matrix:

$$\widetilde{S_B} = \sum_{i=1}^{c} n_i(\widetilde{m_i} - \widetilde{m})(\widetilde{m_i} - \widetilde{m})^t$$

where n_i is the number of samples within class i. $\mathbf{S}_B$ simply shows how far are the the centers of classes to each other.

Having these definitions, MDA maximizes the criterion function:

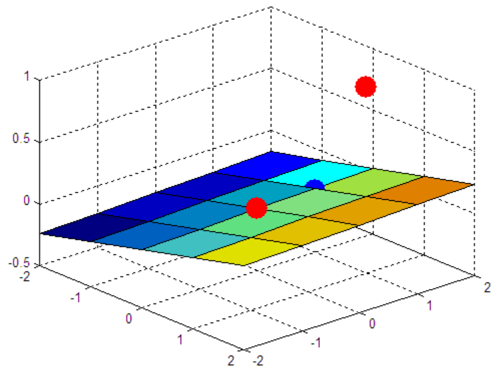
$$J(W) = \frac{|\widetilde{S_B}|}{|\widetilde{S_W}|}$$

It means it looks for a new representation of the data such that the distances between the mean values of the classes will be as big as possible comparing to sum of distances between samples within one class. Such new components makes the separation of the classes easier.

III. Results

In the first section, the linear classifier was unable to separate samples before mapping them to a higher dimension. It is because linear classifiers draws the boundaries between classes as hyperplanes and hence cannot separate non-linearly separable data (like our samples).

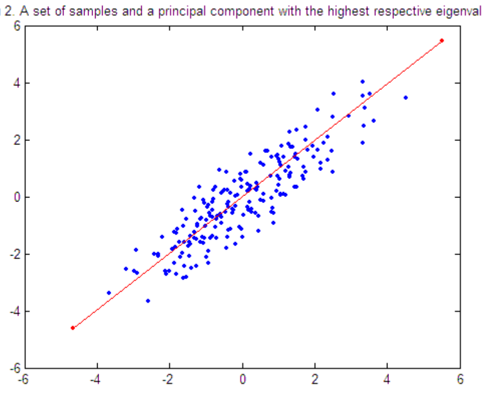
Fig 1. Hyperplane dividing mapped data.



After mapping data to 3-dimensional space, they were successfully separated as shown in Figure 1. It means that extending the number of dimensions can help in recognition task but it is not certain for any extension. In other words, extension has to be done carefully and new features has to be chosen in such a way that they really help. This decision can be done manually only in lower dimensionality case. For higher number of dimension the problem becomes too complex to be solved manually.

The idea of PCA is to find new set of features which will be a linear combination of old features. These new features have to represent the set of samples from a class in an optimal way in a least-square sense as described in [1].

Fig 2. A set of samples and a principal component with the highest respective eigenvalue.



For the sample data from *simple_data_PCA.mat* file, PCA algorithm found two principal components. Out of them, the one with a bigger corresponding eigenvalue has been chosen and was then visualized in Figure 2, together with the data. Its direction is along the longer diagonal of the "data cloud" oval. Because the second principal component corresponds to much smaller eigenvalue, it is probably a vector orthogonal to the first one, so that the variation of the samples projected on it is small too.

The difference between MDA and PCA is that MDA seeks for components which will show the differences between classes as good as possible while PCA returns components which can describe each of the class the best (separately). That is why the components found by PCA and MDA does not have to be the same.

There is a good example of this in [1]. If we had two letters O and Q, PCA would probably found for both of them the circle which mostly characterizes each of these letters separately. However, to decide upon a class these letters belong to, this feature would not be good. Better feature for that case would be found by MDA i.e. probably the tail of

"Q" would be this feature which discriminates between these two letters the best.

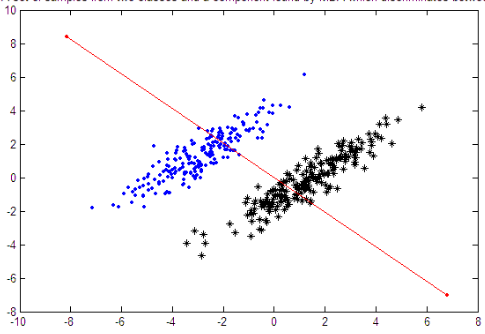


Fig 3. A set of samples from two classes and a component found by MDA which discriminates between them.

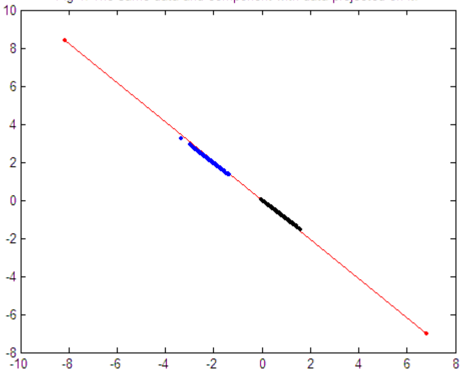


Fig 4. The same data and component with data projected on it.

The above two figures shows the results of MDA used in practice. The red line in figure 3 is a component found by MDA to discriminate between two classes – blue and black ones. Figure 4 shows the same data set which was projected on the found component. The classification of these data can be still performed and is even easier as there is only one dimension.

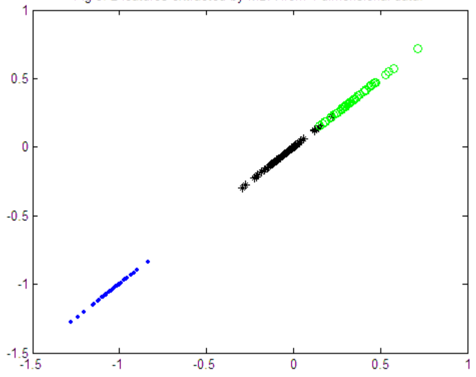


Fig 5. 2 features extracted by MDA from 4 dimensional data.

Figure 5 presents the result of using MDA algorithm for reducing the number of dimensions from four to two (the axes corresponds to each of the feature). The classes are still separable and the problem is now much easier. It seems that even one feature would be enough to separate these two classes.

IV. Conclusions

The number of features can be simply creating a new feature set where all the features are liner combinations of the old ones. It can be done either manually (recommended only in low dimensions) or automatically. MDA algorithm seem to be a really good automatic tool to lower the dimensionality of a feature space. PCA algorithm, which has a different aim than MDA, can be used for example for compression purposes.

V. MATLAB Code

```
% 1.1

fv1 = [ 1 0 0;
        1 1 1 ]';
fv2 = [ 1 1 0;
        1 0 1 ]';
features = [ fv1(2:3, :)
fv2(2:3, :) ];

a = perceptron(fv1, fv2, 0.01,
0.1);

% 1.2

fv1 = [ fv1(1, :)' fv1(2, :)'
fv1(3, :)' (fv1(2, :) .*
fv1(3, :))' ]';
fv2 = [ fv2(1, :)' fv2(2, :)'
fv2(3, :)' (fv2(2, :) .*
fv2(3, :))' ]';
features = [ fv1(2:4, :)
fv2(2:4, :) ];

a = perceptron(fv1, fv2, 0.01,
0.05);

% 1.3

[X, Y] = meshgrid(-2:1:2, -2:1:2);
Z = - ((a(1) + a(2) .* X + a(3) .*
Y) ./ a(4));

figure;
surf(X, Y, Z); hold on;
scatter3(fv1(2, :), fv1(3, :),
fv1(4, :), 300, [1 0 0],
'filled'); hold on;
scatter3(fv2(2, :), fv2(3, :),
fv2(4, :), 300, [0 0 1],
'filled');
title('Fig 1. Hyperplane dividing
mapped data.');


% 2.2

load simple_data_PCA.mat;
```

```
figure;
plot(simple_data_PCA(:, 1),
simple_data_PCA(:, 2), 'b.'); hold
on;

% 2.3

C = cov(simple_data_PCA);
[EVectors EValues] = eig(C);

% 2.4

largestPC = EVectors(:,
find(sum(EValues) ==
max(sum(EValues))));
X = [min(simple_data_PCA(:, 1)) -
1 max(simple_data_PCA(:, 1)) + 1];
Y = (largestPC(1) .* X) ./
largestPC(2);
plot(X, Y, 'r.-');
title('Fig 2. A set of samples and
a principal component with the
highest respective eigenvalue.');

% 3.2

load simple_data_MDA.mat;
fv1 = simple_data_MDA(1:200, :);
fv2 = simple_data_MDA(201:400, :);
figure;
plot(fv1(:, 1), fv1(:, 2), 'b.');
hold on;
plot(fv2(:, 1), fv2(:, 2), 'k*');
hold on;

% 3.3

targets = ones(400, 1);
targets(201:400) = 2;
[W] = MDAMod(simple_data_MDA,
targets, 1);

X = [(min(simple_data_MDA(:, 1)) -
1) (max(simple_data_MDA(:, 1)) +
1)];
Y = (W(1) .* X) ./ W(2);
plot(X, Y, 'r.-');
title('Fig 3. A set of samples
from two classes and a component
found by MDA which discriminates
between them.');

% 3.4

figure;
plot(X, Y, 'r.-'); hold on;
projectedPoints = (W' *
simple_data_MDA')' * W';
scatter(projectedPoints(1:200, 1),
projectedPoints(1:200, 2), 10, [0
0 1], 'filled'); hold on;
scatter(projectedPoints(201:400,
1), projectedPoints(201:400, 2),
10, [0 0 0], 'filled'); hold on;
title('Fig 4. The same data and
component with data projected on
it.');

% 3.5

load iris.mat;

% 3.6

[W] = MDAMod(X, y, 2);
projectedData = (W' * X')';
figure;
plot(projectedData(find(y == 1),
1), projectedData(find(y == 1),
1), 'b.'); hold on;
plot(projectedData(find(y == 2),
1), projectedData(find(y == 2),
1), 'k*'); hold on;
plot(projectedData(find(y == 3),
1), projectedData(find(y == 3),
1), 'go'); hold on;
title('Fig 5. 2 features extracted
by MDA from 4 dimensional data.');
```

VI. References

[1] R. Duda and P. Hart and D. Stork *Pattern Classification*, 2nd ed, 2001