Ministry of Education and Science of the Russian Federation
Peter the Great St. Petersburg State Polytechnic University
Institute of Computer Sciences and Technologies
Graduate School of Cyber-Physical Systems and Control

Report 6

Principal Component Analysis (PCA) and Support Vector Machines (SVM)
Discipline: Modern Problems of Informatics and Computer Science
19 May 2017

Student Group: 13541/8

Christopher W. Blake

Professor

Rodionova E.A.

St. Petersburg
2017

Contents

Christopher W. Blake

Problem – Classification of a Dataset

Problem Description

A dataset with various observations has various properties for each observation. Such a dataset may have 4+ dimensions, hence graphical methods are no longer possible or, at least, not intuitive to create. An example dataset with just this problem is provided and Principle Component Analysis (PCA) and Support Vector Machines (SVM), possible solutions should be explored.

Iris Dataset

To test these two methods the well-known Iris Flower dataset, from Fisher, is utilized. This dataset includes 150 observations (measurements), 4 properties (sepal length, sepal width, petal length, petal width), and 3 classifications (Setosa, Versicolor, Verginica). A pairwise comparison of these properties affects is shown in figure 4.

Figure 1: Versicolor

Figure 2: Verginica

Figure 3: Setosa

Table 1: Iris Flower Observations (examples)

Sepal Length	Sepal Width	Petal Length	Petal Width
7	3.2	4.7	1.4
6.4	3.2	4.5	1.5
6.9	3.1	4.9	1.5
5.5	2.3	4	1.3
6.5	2.8	4.6	1.5
5.7	2.8	4.5	1.3
6.3	3.3	4.7	1.6
4.9	2.4	3.3	1
6.6	2.9	4.6	1.3
5.2	2.7	3.9	1.4
…	…	…	…

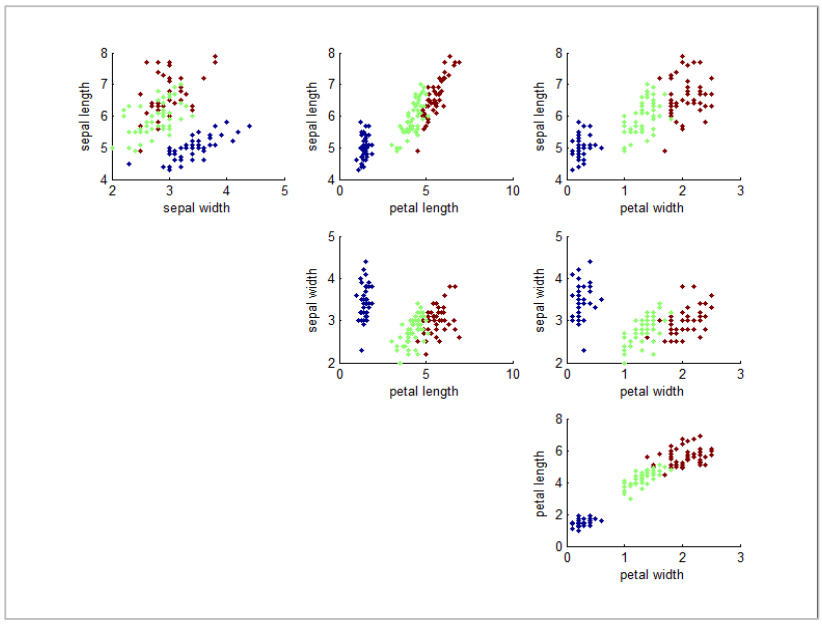

Figure 4:Property vs Property Analysis, Iris Dataset

Solution 1 – Principal Component Analysis

Overview

The first method discussed attempts to reduce the dimensionality of the property space. Through PCA, the correlation of the different properties is generated and using these correlation values, a new set of principle components are created. Each principle component has a relationship to the original properties, thereby requiring less properties to explain the same dataset variation. The process follows the following steps.

1. Calculate mean and standard deviation for each property.
2. Standardize all observations.
3. Calculate correlation matrix between properties.
4. Calculate eigen vectors for each correlation.
5. Calculate principle components.
6. Select components that make up 95% of variance.
7. Plot results

Step 1 – Mean and Standard Deviation

Using the below two equations, the mean and standard deviation are calculated for each property (sepal length, sepal width, petal length, petal width).

$$mean = \bar{x} = \frac{1}{n} * \sum_{i=1}^{n} X_i$$

$$standard\ deviation\ = s = \sqrt{\frac{\sum x_i - \bar{x}}{n-1}}$$

Where:
n = number of observations
x = property value for observation n

	Sepal Length	Sepal Width	Petal Length	Petal Width
Mean	5.843	3.057	3.758	1.199
Standard Deviation	0.828	0.436	1.765	0.762

Step 2 – Standardize Observations (Z)

Using the below equation on each property column, each measurement is standardized for each property, allowing later operations to be simpler. By standardizing, the mean becomes 0 and the standard deviation becomes 1. This matrix is referred to as the "Z" matrix.

$$Z = \sum_{i=1}^{n} \frac{x_i - \bar{x}}{s}$$

Where:
n = number of observations
x_i = property value for observation n
$\bar{x}$ = mean
s = standard deviation

Z: Sepal Length	Z: Sepal Width	Z: Petal Length	Z: Petal Width
-0.898	1.016	-1.336	-1.311
-1.139	-0.132	-1.336	-1.311
-1.381	0.327	-1.392	-1.311
-1.501	0.098	-1.279	-1.311
-1.018	1.245	-1.336	-1.311
-0.535	1.933	-1.166	-1.049
-1.501	0.786	-1.336	-1.180
-1.018	0.786	-1.279	-1.311
-1.743	-0.361	-1.336	-1.311
-1.139	0.098	-1.279	-1.442
…	…	…	…

Step 3 – Correlation Matrix (R)

Using the below equation on each property column (Z), an orthogonal matrix is generated which represents the correlations of each property onto each other property. This matrix is referred to as the "R" matrix.

$$R_{ab} = \frac{\sum_{b=1}^{p} z_a * z_b}{n - 1}$$

Where:
Z_a = Z for current property
Z_b = Z for comparison property column
p = number of columns (properties)
n = number of observations

	Sepal Length	Sepal Width	Petal Length	Petal Width
Sepal Length	1.000	-0.118	0.872	0.818
Sepal Width	-0.118	1.000	-0.428	-0.366
Petal Length	0.872	-0.428	1.000	0.963
Petal Width	0.818	-0.366	0.963	1.000

Step 4 – Eigen Vectors (E)

Using any of many techniques, such as power-iteration method, the eigen values are calculated for the correlation matrix, for each principle component. This eigen vectors matrix is referred to as the "E" matrix.

$$eigen = \lambda = b$$
$$when \lim_{n\to\infty}\left(b_{n+1} = \frac{Ab_n}{\|Ab_n\|}\right)$$

	PC1	PC2	PC3	PC4
Eigen Values	0.021	0.147	0.914	2.919

	PC1	PC2	PC3	PC4
Eigen Vectors	-0.261	0.720	0.377	0.521
	0.124	-0.244	0.923	-0.269
	0.801	-0.142	0.024	0.580
	-0.524	-0.634	0.067	0.565

Step 5 – Calculate Principle Components (PC)

Using the below matrix equation, calculate the possible principle components.

$$[PC] = [Z] * [E]$$

Where:
Z = matrix of standardized observations
E = matrix of eigen vectors

Step 6 – Select Top Components

Using the eigen values from step 4, the most important principle components are identified. The principle components are sorted in descending order by eigen value. The eigen values are normalized to show their percentage of the total.

	Eigen Value	% Total	% Sum
PC4	2.918	72.96	72.96
PC3	0.914	22.85	95.81
PC2	0.147	3.67	99.48
PC1	0.021	0.52	100.00

From the above principal components, the top components are PC4 and PC3, which together make up 95.81% of the variance.

Step 7 – Graph Results

The original dataset required 4 parameters to represent 1 observation. This was not graphically representable. However, after PCA, the observations are only dependent on 2 parameters. Using these two principle components (PC4 and PC3)

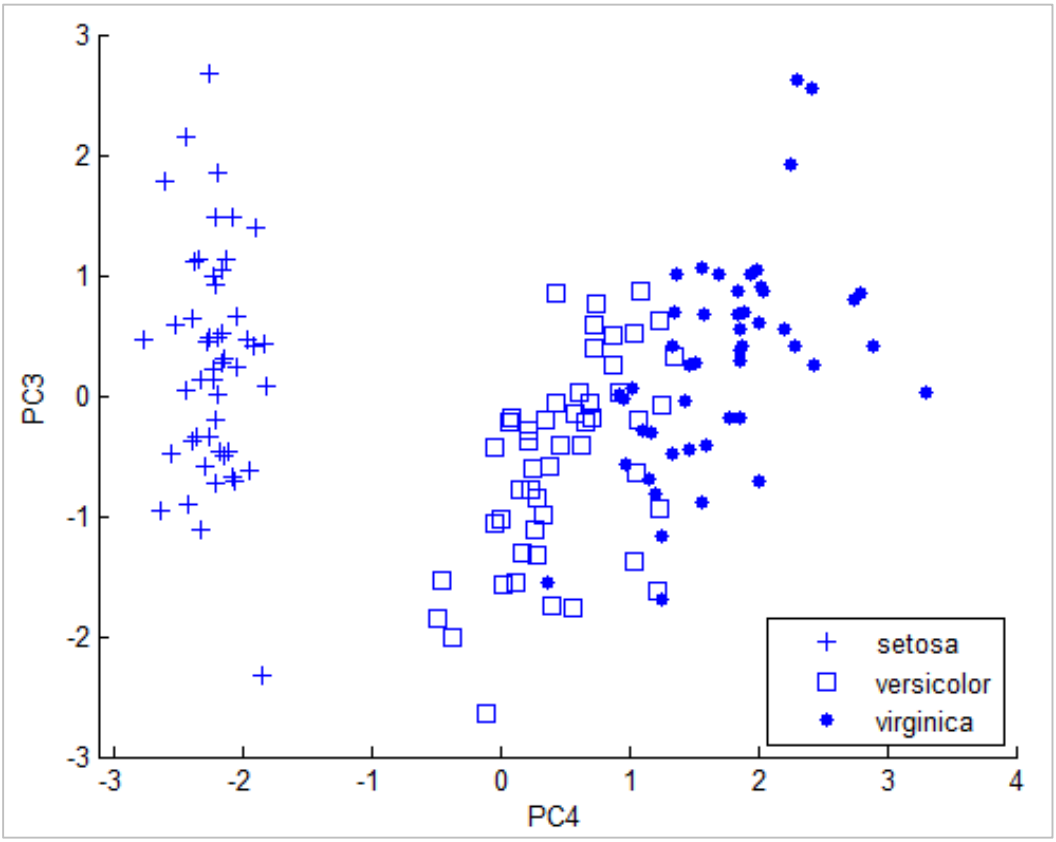

Figure 5: PCA Results, Iris Dataset

Christopher W. Blake

Solution 2 – Support Vector Machines

Overview

A support vector machine is a supervised algorithm mainly for classification tasks. A set of previously labeled data is used as training information, and then a line is produced which divides the space into two distinct regions. The idea is similar to regression, but instead of finding a line of best fit, a line of best separation is created. This is demonstrated in figure 1, with traingles on one side of the line and circles on the other.

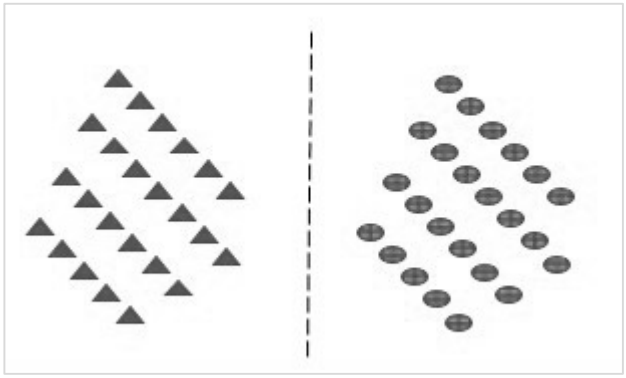

Figure 6: Linearly Separable Data

A SVM is inherently a linear classifier, but is extended to non-linear classifications using the kernel technique, which transforms a data set.

Process: Linear Method

The below method references figure 5, which shows two data sets separate by various lines. The different elements are described below to aid explanations of the method. For the purpose of this example, all data has been normalized to 1.

X1 – Characteristic 1 of the feature vector
X2 – Characteristic 2 of the feature vector
W – The normal vector of the separation lines
Y – The classification result (1 or -1)

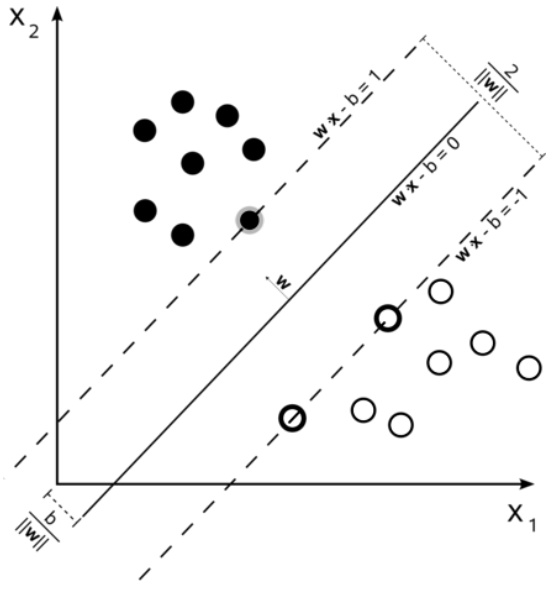

Figure 7: Datasets separated by maximum margin lines

Two parallel lines are selected next to each data set, and the line directly in the middle of them is calculated. The nearest lines are chosen by those points nearest the two data sets, which are called the support vectors.

The initial two lines can be represented as:

Black Dots Line: $\vec{w} \cdot \vec{x_i} - b = 1$

White Dots Line: $\vec{w} \cdot \vec{x_i} - b = -1$

These two equations are combined to calculate the distance between them, which needs to be maximized. This can be done by minimizing the normal vector W.

$$SeperationDistance = \frac{2}{\|\vec{W}\|}$$

The calculation space is further restricted using the previously specified lines separating the black and white does from the margin area.

$$\vec{w} \cdot \vec{x_i} - b \geq 1, if\ y_i = 1\ (black)$$
$$\vec{w} \cdot \vec{x_i} - b \leq -1, if\ y_i = -1\ (white)$$

These restraint equations can be combined as the following.

$$y_i(\vec{w} \cdot \vec{x_i} - b) \geq 1$$

Finally, the minimization of W is combined with the restraints to create the optimization problem. Any multi-dimensional optimization technique may be used at this point.

$$\min \left\| \vec{W} \right\| \ for \ \ y_i(\vec{w} \cdot \vec{x_i} - b) \geq 1$$

Step 1 – Data Preparation

The iris dataset has 3 classifications and four properties for each observation. SVM is a binary classifier, so one of the classes, "Setosa," is removed. SVM can also be extended using kernel methods to handle 3+ dimensions, but this removes the ability to graphically represent the results. So, for demonstration purposes, only the petal length and petal width properties will be used. Hence the data is filtered to look like below.

Petal Length	**Petal Width**	**Classification**
4.70	1.40	'versicolor'
4.50	1.50	'versicolor'
4.90	1.50	'versicolor'
4.00	1.30	'versicolor'
4.60	1.50	'versicolor'
…	…	…
6.00	2.50	'virginica'
5.10	1.90	'virginica'
5.90	2.10	'virginica'
5.60	1.80	'virginica'
5.80	2.20	'virginica'
…	…	…

Step 2 – SVM Training (manual)

Using the prepared dataset, the previous described process, without kernel functions, is implemented. The Matlab code is shown in appendix 1. Below is a listing of the steps.

1. Load Iris dataset
   1.1. Separate training data
   1.2. Count training samples
   1.3. Split into 2 classes
2. Prepare equations
   2.1. Create system of equations
   2.2. Reformat for quadratic solver
3. Solve system of equations
   3.1. Configure optimization options
   3.2. Perform optimization
   3.3. Get support vectors
   3.4. Solve for w (separation line slope) and b (offset)
   3.5. Create equations for margin lines
4. Plot Original Data
   4.1. Create graph
   4.2. Show data points for each class
   4.3. Show support vectors
5. Plot separation lines
   5.1. Create points for separation lines
   5.2. Show classification line
   5.3. Show margin line for class A
   5.4. Show margin line for class B

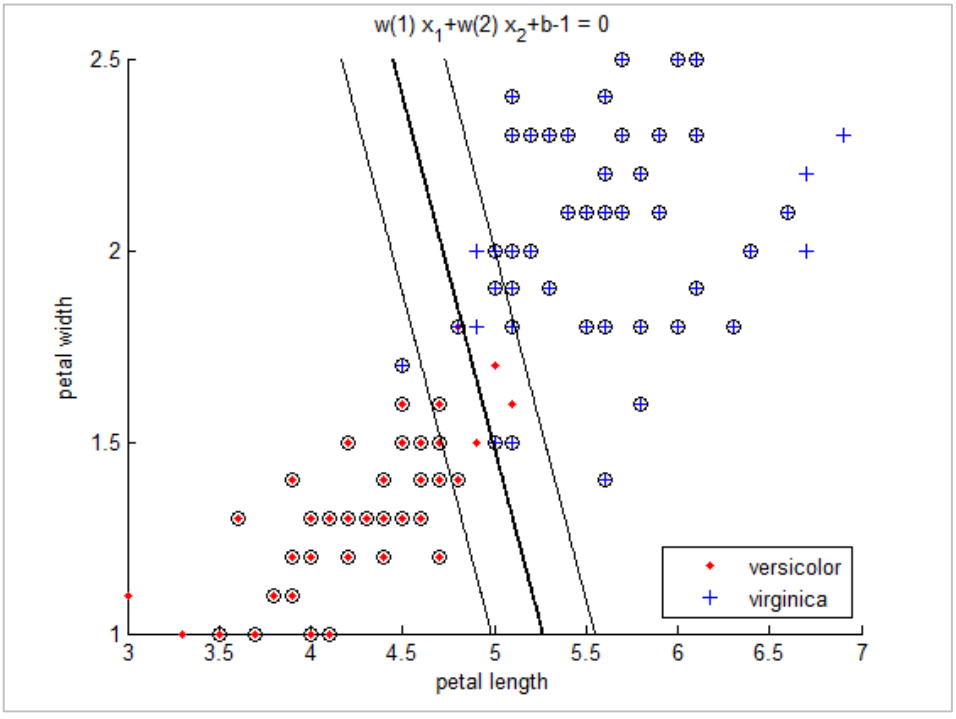

Figure 8: SVM - Iris Dataset - Manually

Step 2 – SVM Training (auto)

Using the prepared dataset, the "svmtrain" Matlab function is utilized. This function produces an "svmstruct", which is used for later classifying of new data. The below chart shows this separation. Points in the top-right represent flowers classified as "Virginica" while points in the bottom left represent flowers classified as "Versicolor". The points closest to the separation line, those in circles, are the support vectors. These are the critical points used for determining the best separation line.

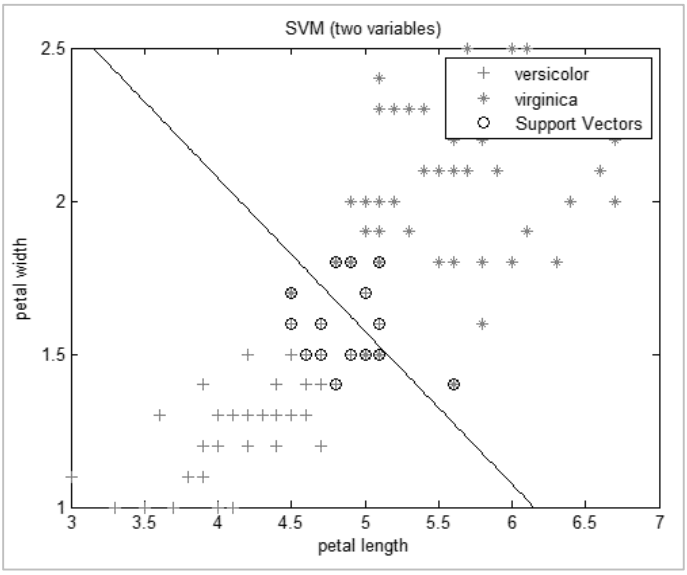

Figure 9: SVM Classification of Iris dataset

Step 3 – Classification of New Data

Using the "svmstruct" from step two, additional data can be classified. For example, the below two flowers are classified. "F1" is found to be "Vircginical" and "F2" is found to be "Versicolor"

Flower	Petal Length	Petal Width
F1	4.70	1.40
F2	4.50	1.50

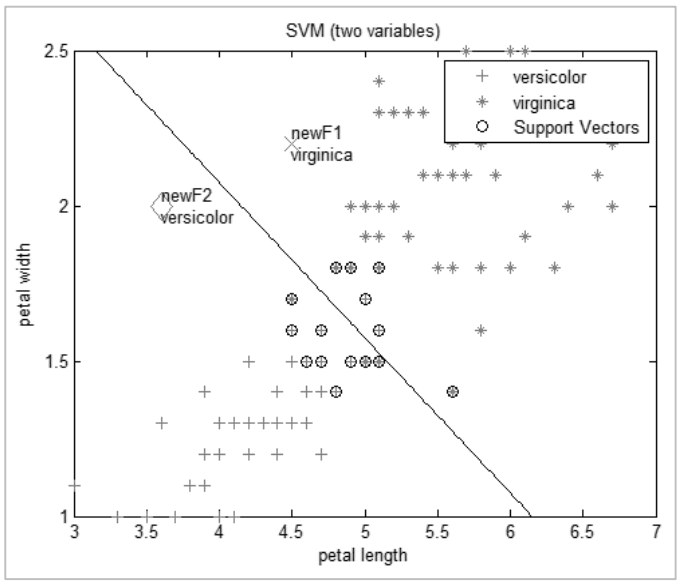

Figure 10: Classification of F1 and F2

Solution 3 – Combined PCA and SVM

Principle Component Analysis (PCA) has the ability to reduce the dimensionality of data. In the example of this iris dataset, the dimensions were reduced from 4 (sepal length, sepal width, petal length, petal width) to 2 (PC4, PC3). This allowed plotting of all information in a 2D graph. If these principle components are then used with SVM, classification can be performed using all 4 properties without usage of kernel methods, which increase dimensionality.

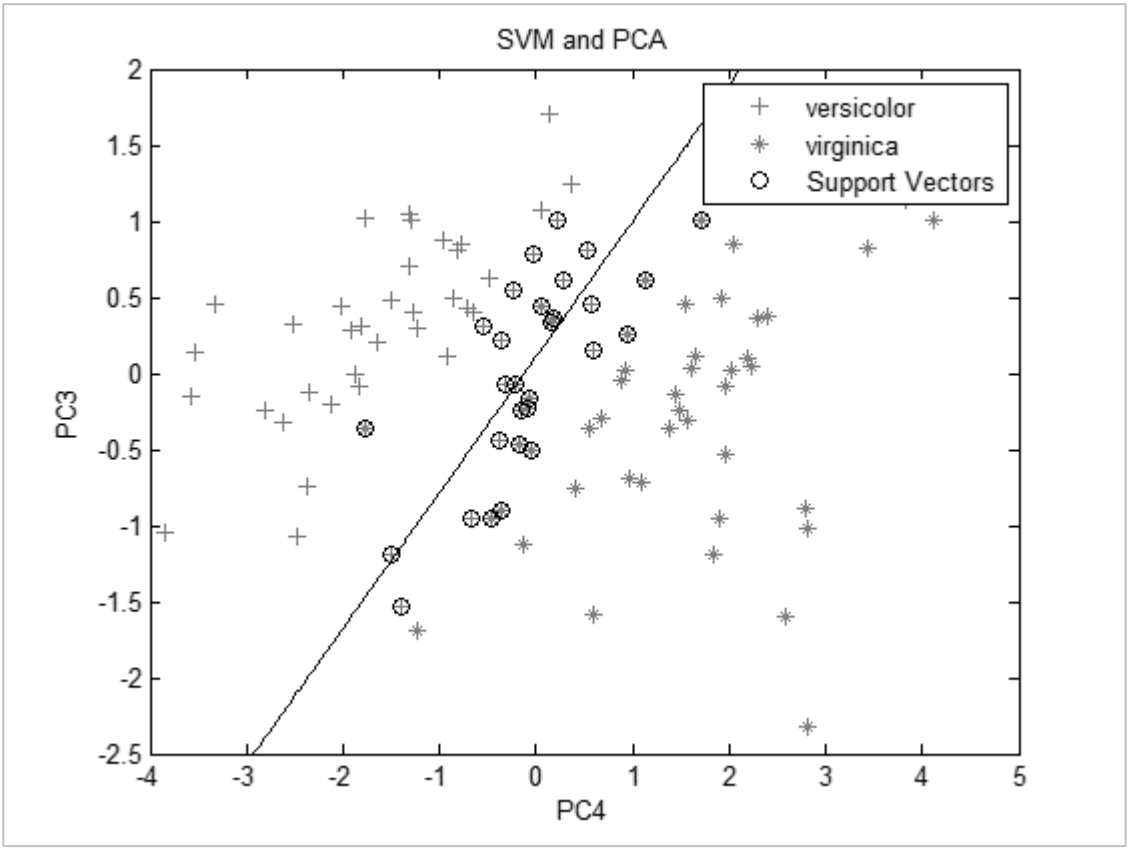

Figure 11: Iris Flower Classification using PCA and SVM

Conclusion

The well-known Iris flower dataset from Fisher is used with Principle Component Analysis (PCA) and Support Vector Machines (SVM). PCA is used to demonstrate that the 4 flower properties of sepal length, sepal width, petal length, petal width can be reduced to 2 dimensions. SVM is used to create a fast-linear classifier, which is dependent on two parameters. By combining these approaches, the SVM method is able to classify 4-dimensional data and display the results in a 2-dimenisional graphical form.

Appendix 1 – Matlab Code

irisPCA.m

```
clear; clc;

%% 1.0 Load dataset
load fisheriris
irisdata = meas;
properties = {'sepal length', 'sepal width', 'petal length', 'petal
width'};

%Convert species names to species ID numbers
speciesLabel = {'setosa', 'versicolor', 'virginica'};
species = strcmp(species, 'setosa')*1 + ...   %ID:0
    strcmp(species, 'versicolor')*2 + ...       %ID:1
    strcmp(species, 'virginica')*3;             %ID:2

%clear old variables
clear meas;

%% 2.0 Plot different features

%150 columns of length 4
X = irisdata(:,1:4)';
n = size(X,2);

figure(1)
% plot x_i vs x_j for all combinations of i,j
for i=1:4
  for j=1:i-1
     subplot(3,3,(i-1)+3*(j-1))
     scatter(X(i,:),X(j,:),7,species,'filled')
     xlabel(properties(i)); ylabel(properties(j))
  end
end

%% 3.0 Perform PCA (Step by Step)
[rows cols] = size(irisdata);

% 1. Mean and Standard Deviation
irisMean = mean(irisdata);
irisStdDev = std(irisdata);

% 2. Standardize measurements
for col=1:cols
    Z(:, col) = (irisdata(:, col) - irisMean(col)) / irisStdDev(col);
end

% 3. Correlation Coefficients
for colA=1:cols
    Za = Z(:,colA);
    for colB=1:cols
        Zb = Z(:,colB);
        R(colA, colB) = sum(Za .* Zb) / (rows-1);
    end
end
clear Za; clear Zb;
%R_matlab = corrcoef(Z); %Verification

% 4. Eigen Vectors
eigValues = eig(R)';
[eigVectors D] = eig(R);
```

```
% 5. Compute scores for each primary component
pcScore = Z * eigVectors;

% 6. Sort principle components according to greatest eigen values.
pcScore = sortrows([eigValues', pcScore'], -1)'; pcScore(1,:) = [];
eigValues = sort(eigValues, 'descend')
perEigValues = eigValues / sum(eigValues)*100

%% Plot Results

% Get principal component 1 & 2
PC1 = pcScore(:,1);
PC2 = pcScore(:,2);

% Plot results
figure(2)
gscatter(PC1, PC2, species, 'bbb', '+s.');
xlabel('PC1'); ylabel('PC2');
legend(char(speciesLabel), 'location', 'southeast')

%% Perform PCA (auto)

%Standardize raw data
irisdataZ = zscore(irisdata);

%Perform PCA
[COEFF, SCORE, LATENT, TSQUARED, EXPLAINED] = pca(irisdataZ, 'Algorithm',
'eig');

% Get principal component 1 & 2
 PC1 = SCORE(:,1);
 PC2 = SCORE(:,2);

% Plot results
figure(3)
gscatter(PC1, PC2, species);
xlabel('PC1'); ylabel('PC2');
legend(char(speciesLabel), 'location', 'southeast')
```

irisSVM-manual.m

```
clc; clear; close all;

%% 1.0 Load iris dataset
load fisheriris
%only take two of the species and two properties ('petal length', 'petal
width')
TrainData = meas(51:end,3:4);
TrainData(1:50, 3) = 1;
TrainData(51:end, 3) = -1;

%1.1 Seperate Training Data
x=TrainData(:,1:2)';
y=TrainData(:,3)';

%1.2 Count training samples
n=numel(y);

%2.3 Split into 2 classes
ClassA=find(y==1);
ClassB=find(y==-1);

%% 2.0 Prepare equations
%Margin size
C=0.1;

% 2.1 Create System of Equations
H=zeros(n,n);
for i=1:n
    for j=i:n
        H(i,j)=y(i)*y(j)*x(:,i)'*x(:,j);
        H(j,i)=H(i,j);
    end
end
f=-ones(n,1);

% 2.1 Reformat for quadratic solver
Aeq=y;
beq=0;
lb=zeros(n,1);
ub=C*ones(n,1);

%% 3.0 Solve system of equations

%3.1 Configure optimization options
Alg{1}='trust-region-reflective';
Alg{2}='interior-point-convex';
Alg{3}='active-set';
options=optimset('Algorithm',Alg{3}, 'Display','off', 'MaxIter',20);

%3.2 Perform optimization
alpha=quadprog(H,f,[],[],Aeq,beq,lb,ub,[],options)';

%3.3 Get support vectors
AlmostZero=(abs(alpha) < max(abs(alpha))/1e5);
alpha(AlmostZero)=0;
S=find(alpha>0 & alpha<C);

%3.4 Solve for w and b
w=0;
for i=S
    w=w+alpha(i)*y(i)*x(:,i);
```

```
end
b=mean(y(S)-w'*x(:,S));

% 3.5 Create equations for margin lines
Line=@(x1,x2) w(1)*x1+w(2)*x2+b;
LineA=@(x1,x2) w(1)*x1+w(2)*x2+b+1;
LineB=@(x1,x2) w(1)*x1+w(2)*x2+b-1;

%% 4.0 Plot Original Data

% 4.1 Create Graph
fh = figure(1);
movegui(fh,'east')
hold on;

% 4.2 Show data points for each class
plot(x(1,ClassA),x(2,ClassA),'r.');
plot(x(1,ClassB),x(2,ClassB),'b+');

% 4.3 Show support vectors
plot(x(1,S),x(2,S),'ko');

%% 5.0 Plot separation line

% 5.1 Create points for seperation line
x1min=min(x(1,:));
x1max=max(x(1,:));
x2min=min(x(2,:));
x2max=max(x(2,:));

% 5.2 Show Classification Line
handle=ezplot(Line,[x1min x1max x2min x2max]);
set(handle,'Color','k','LineWidth',2);

% 5.3 Show Margin Line - Class A
handleA=ezplot(LineA,[x1min x1max x2min x2max]);
set(handleA,'Color','k','LineWidth',1,'LineStyle',':');

% 5.4 Show Margin Line - Class B
handleB=ezplot(LineB,[x1min x1max x2min x2max]);
set(handleB,'Color','k','LineWidth',1,'LineStyle',':');

% Add Legend and Labels
xlabel('petal length');
ylabel('petal width');
legend('versicolor', 'virginica', 'Location', 'southeast');
```

irisSVM-auto.m

```
clear; clc; close all;

%% 1.0 Load iris dataset
load fisheriris

%only take two of the species and two properties ('petal length', 'petal
width')
xdata = meas(51:end,3:4);
group = species(51:end);

%% 2.0 Perform the svm
figure(1);
svmStruct = svmtrain(xdata,group,'ShowPlot',true);
title('SVM (two variables)')
xlabel('petal length'); ylabel('petal width');

%% 3.0 Classify new flowers
newF1 = [4.5 2.2];
newF2 = [3.6 2.0];

speciesF1 = svmclassify(svmStruct,newF1);%,'ShowPlot',true)
speciesF2 = svmclassify(svmStruct,newF2);%,'ShowPlot',true)

hold on;
plot(newF1(1),newF1(2),'rx','MarkerSize',12);
text(newF1(1),newF1(2),['newF1 ', speciesF1])
plot(newF2(1),newF2(2),'rd','MarkerSize',12);
text(newF2(1),newF2(2),['newF2 ', speciesF2])
hold off
```

irisPCA_SVM.m

```
clear; clc; close all;

%% 1.0 Load iris dataset
load fisheriris

%Only take two of the species (keep all four properties)
irisdata = meas(51:end, :);
species = species(51:end);
group = species;

%Convert names to ids
speciesLabel = {'versicolor', 'virginica'}
species = strcmp(species, 'versicolor')*1 + ...        %ID:1
          strcmp(species, 'virginica')*2;              %ID:2


%% 2.0 Use PCA to reduce from 4 dimensions to 2
irisdataZ = zscore(irisdata);
[COEFF, SCORE, LATENT, TSQUARED, EXPLAINED] = pca(irisdataZ, 'Algorithm',
'eig');

%% Plot PCA
% Get principal component 1 & 2
 PC1 = SCORE(:,1);
 PC2 = SCORE(:,2);

% Plot results
figure(1)
gscatter(PC1, PC2, species, 'rg', '+*');
title('PCA')
xlabel('PC1'); ylabel('PC2');
legend(char(speciesLabel), 'location', 'northeast')

%% 2.0 Perform the svm
figure(2);
%Select first two principle components
xdata = SCORE(:, 1:2);

%Generate SVM
svmStruct = svmtrain(xdata,group,'ShowPlot',true);
title('SVM and PCA')
xlabel('PC4'); ylabel('PC3');
```