University of Ljubljana
Faculty of Mechanical Engineering

Neural Networks course

Seminar on theme:

“Classification of 4-class spiral problem by SVM (Support Vector Machine) methodology”.

Student: Ivan Saprunov

Mentor: Primož Potočnik, assist. prof.

Date: 14.01.09

Semester 2008/2009

Ljubljana 2009

CONTENT

1. INTRODUCTION.

SVM (Support Vector Machine) is a useful technique for data classification. Even though people consider that it is easier to use than Neural Networks, however, users who are not familiar with SVM often get unsatisfactory results.

A classification task usually involves with training and testing data which consist of some data instances. Each instance in the training set contains one “target value” (class labels) and several “attributes" (features). The goal of SVM is to produce a model which predicts target value of data instances in the testing set which are given only the attributes.

Given a training set of instance-label pairs (x_i, y_i); i = 1… l, where $x_i \in R_n$ and $y \in \{1,-1\}^l$, the support vector machines (SVM) require the solution of the following optimization problem:

$$\begin{aligned} \min_{\mathbf{w},b,\xi} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0. \end{aligned}$$

Here training vectors x_i are mapped into a higher (maybe infinite) dimensional space by the function ϕ. Then SVM finds a linear separating hyperplane with the maximal margin in this higher dimensional space. C > 0 is the penalty parameter of the error term. Furthermore $K(x_i, x_j) \equiv \phi(x_i)^T \phi(x_j)$ is called the kernel function. Though new kernels are being proposed by researchers, beginners may find in SVM books the following four basic kernels:

- linear: $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T\mathbf{x}_j$.
- polynomial: $K(\mathbf{x}_i, \mathbf{x}_j) = (\gamma\mathbf{x}_i^T\mathbf{x}_j + r)^d$, $\gamma > 0$.
- radial basis function (RBF): $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2)$, $\gamma > 0$.
- sigmoid: $K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\gamma\mathbf{x}_i^T\mathbf{x}_j + r)$.

Here γ, r and d are kernels parameters.

2. PROBLEM STATEMENT.

2.1. Task description.

The exist function models 4-class data set which have shape of spiral. This data sets can have 2 main differs (Figure 1):

- number of points (correspond to the length of spiral)
- level of spreading of data points

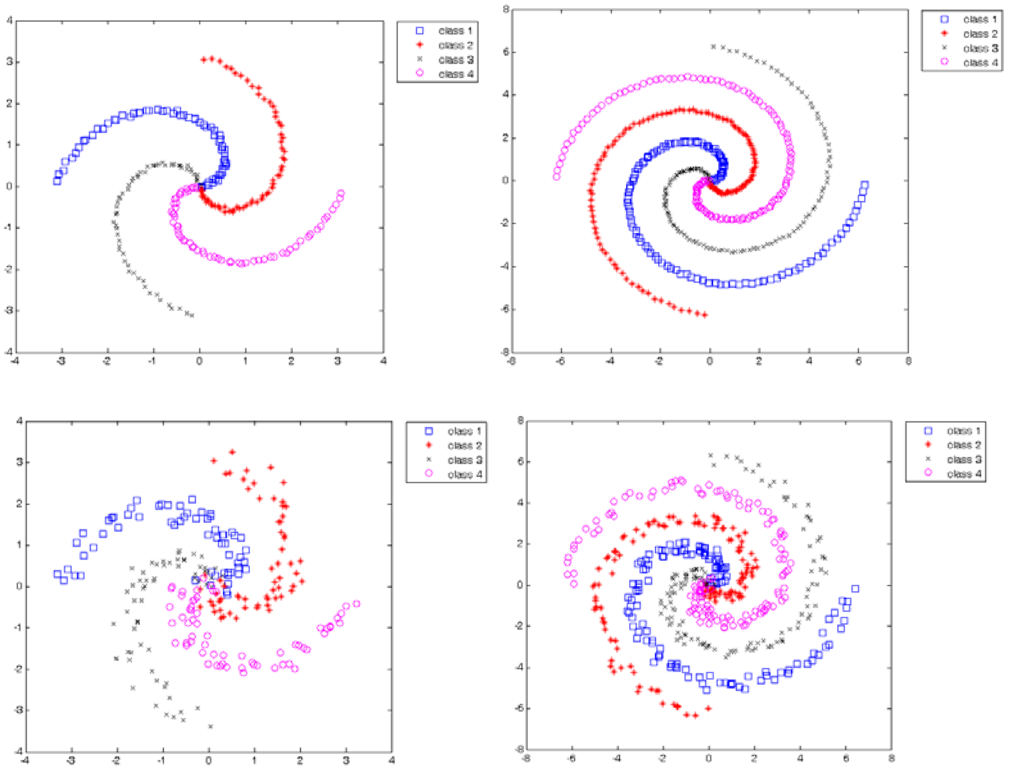


Figure 1. Examples of different spiral data sets.

Problem with such classification by using neural networks can be that border between classify classes sometimes is not so smooth which means bad generalization of obtained solution.

2.2. Goal of the seminar.

First of all the goal of the seminar is to show how SVM can be used to solve such kind of classification problem. So it's necessary to learn procedure of applying such methodology, find useful libraries which realize SVM algorithms and finally obtain good solution of the task.

3. SOLUTION STEPS.

3.1. Tool for realize SVM algorithms.

To apply SVM approach for our task was necessary to find already written functions, which is possible to use in Matlab. Such library was taken from internet: http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/

In short "Practical guide for SVM classification" written that usually many beginners follows this procedure:

- Transform data to the format of an SVM software
- Randomly try different kernels and parameters
- Test

From other hand authors of guide recommended their own more sophisticated procedure:

- Transform data to the format of an SVM software
- Conduct simple scaling of the data
- Consider the RBF kernel $K(x,y) = e^{-\gamma\|x-y\|^2}$
- Use cross validation to find the best parameters C and γ
- Use C and γ to train the whole training set
- Test

To solve our problem we try 2 types of kernels – RBF and polynomial. But at the beginning use "newbie" method and see what result we obtain.

3.1.1. Data creation and preprocessing.

At the beginning we create spiral data, make simple scaling of our data to [-1,1] range, code 4-class data and split whole data set to 2 parts: training and testing. Results of working this code represents on figure 2.

```
%% Create initial data.
clc;
clear all;
close all;

figure(1);
subplot(121);
N = 500;                                             % Number of points
deviat = .5;                                         % spreading of spirals
```

```
data_init = data_spiral(N,deviat,1);            % generate data
i = randperm(N);
data_init.x = Scale(data_init.x,-1,1);          % simple Scaling data to [-1,1]
range
hold on;
                                                % x - coordinates of points
x_tr = data_init.x(i(1:(0.7*N)),:);             % 70% of training data
x_test = data_init.x(i((0.7*N):N),:);           % 30% of testing data

y_tr = data_init.y(i(1:(0.7*N)));               % y - class Labels
y_test = data_init.y(i((0.7*N):N));

x_tr = x_tr';                                   % Transpose matrix (preprocess for
SVM functions)
x_test = x_test';

y_tr = y_tr';
y_test = y_test';

% Plot train, validation and test data sets
subplot(122);
plot(x_tr(1,:),x_tr(2,:),'go',x_test(1,:),x_test(2,:),'rp');
ylabel('Y coordinate');
xlabel('X coordinate');
legend('Train data','Validation data','Test data',-2);
```

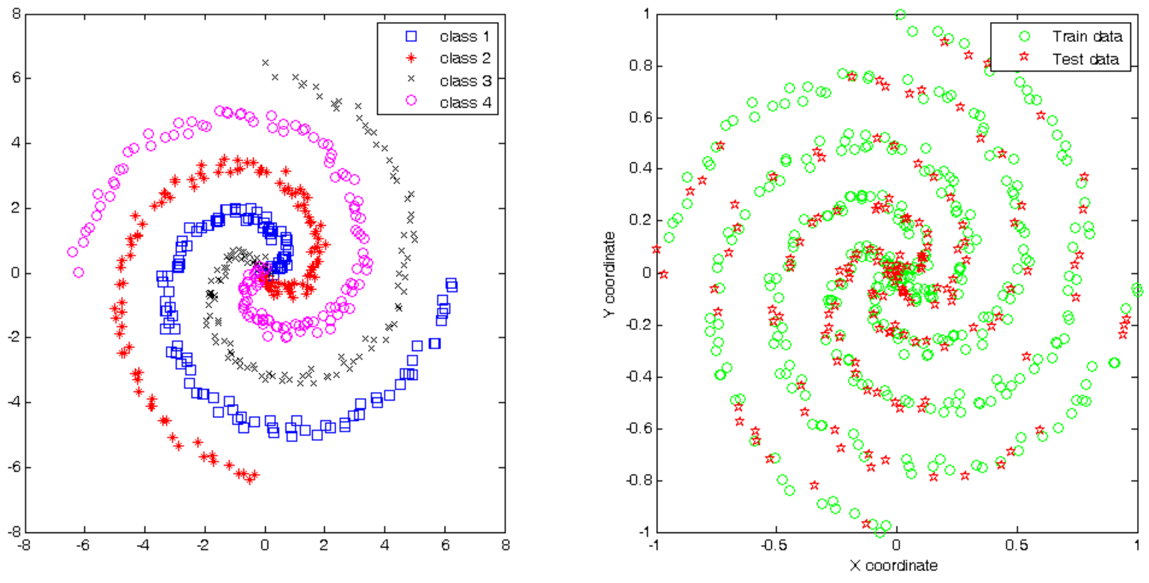


Figure 2. Creation of initial data and split it into 2 different sets.

Than build simplest form of a RBF Classifier with one parameter (Gamma = 1) and save it:

```
%% Create classifier
Tr_Samples = x_tr;
Tr_Labels = y_tr;
Gamma = 1;
[AlphaY, SVs, Bias, Parameters, nSV, nLabel] = RbfSVC(Tr_Samples, Tr_Labels,
Gamma)
save  IVAN_SVMClassifier_01 AlphaY SVs Bias Parameters nSV nLabel;
```

3.1.2. Build classifier and apply it to testing data. Preliminary results.

Perform classification procedure with testing data, plot results (Fig. 3):

```
%% Classify input patterns using the constructed nonlinear SVM Classifier
load IVAN_SVMClassifier_01;
Test_Samples = x_test;
Test_Labels = y_test;
[Labels, DecisionValue]= SVMClass(Test_Samples, AlphaY, SVs, Bias, Parameters,
nSV, nLabel);
[Labels_new,IX] = sort(Labels,2);
N=size(Labels);
for i = 1:N(2)
Test_Labels_new(i)=Test_Labels(IX(i))
end;
%Compare the resultant labels with the true labels of the data
figure(2)
plot(Test_Labels_new(1,1:N(2)),'r.');
hold on;
plot(Labels_new(1,1:N(2)),'-b');
ylabel('Class Index');
xlabel('Pattern Index');
legend('Resultant Labels','True Labels',0);

%% Plot results
figure(3);
init_Samples = data_init.x;
init_Labels = data_init.y;
aspect = 1;
mag = 0.5;
xaxis = 1;
yaxis = 2;
step = 200;
IVAN_SVMPlot(AlphaY, SVs, Bias, Parameters,Test_Samples, Test_Labels, nSV, nLabel,
aspect,mag,xaxis,yaxis,step)
```

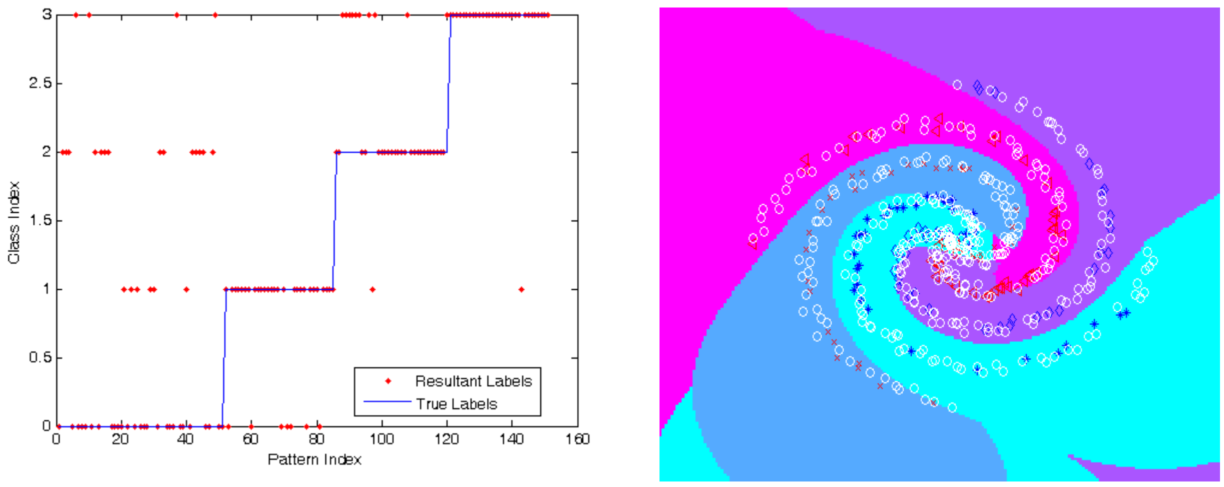


Figure 3. Results of “simple” methodology, Gamma = 1, N = 500.

Here white dots are support vectors. As we can see obtained results not really bad, but far away from ideal.

Just try another value of Gamma = 10 and see what result we can get (Figure 4).

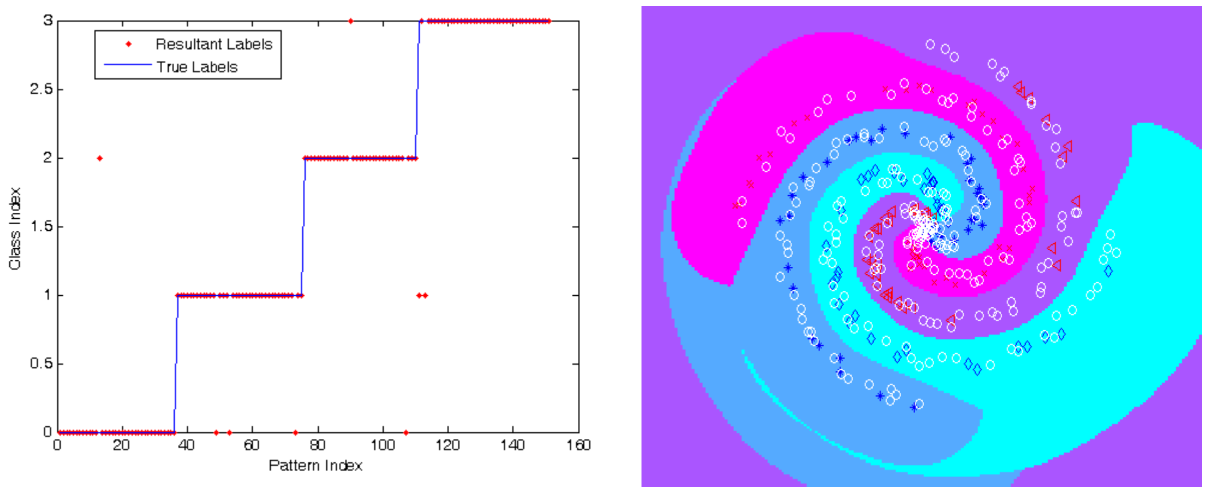


Figure 4. Results of “simple” methodology, Gamma = 10, N = 500.

From other hand increase number of data points (N = 1000) and observe result (Figure 5).

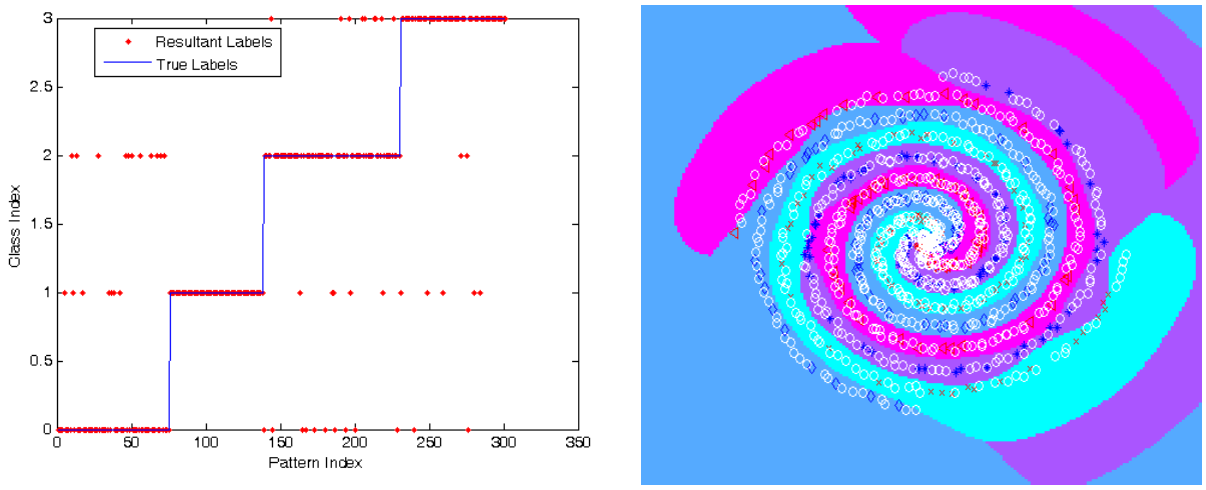


Figure 5. Results of “simple” methodology, Gamma = 10, N = 1000.

From such view can be clearly seen that by changing initial condition of task we should try different values of parameters of the kernel to get more or less good result. But from other side it’s possible to use more smart technique, which should guarantee not bad result. As we start to work with RBF Kernel, let’s continue to use this type of kernel, also it’s recommendation on guide to consider such kernel.

3.2. Advanced methodology. RBF kernel.

As was proposed let's start to use advanced technique of building Classifier. For realization necessary to include some improvement into a code:

- divide initial data to 3 sets (training, validation, testing)
- prepare "grid search" of 2 main parameters of RBK kernel (C, Gamma)
- use cross validation to obtained Classifiers with best parameters (C, Gamma)
- train Classifier with best parameters on the whore (training+validation) set
- apply Classifier on testing data

3.2.1. Data preprocessing.

Create another set of validation data (Figure 6):

```
x_tr = data_init.x(i(1:(0.5*N)),:);              % 50% of training data
x_val = data_init.x(i((0.5*N):(0.7*N)),:);       % 20% of validation data
x_test = data_init.x(i((0.7*N):N),:);            % 30% of testing data

y_tr = data_init.y(i(1:(0.5*N)));                % y - class Labels
y_val = data_init.y(i((0.5*N):(0.7*N)));
y_test = data_init.y(i((0.7*N):N));

x_tr = x_tr';                                    % Transpose matrix (preprocess for
SVM functions)
x_val = x_val';
x_test = x_test';

y_tr = y_tr';
y_val = y_val';
y_test = y_test';

TR_X = [x_tr,x_val];                             % complete set of "training data"
(include validation data)
TR_Y = [y_tr,y_val];
% Plot train, validation and test data sets
subplot(122);
plot(x_tr(1,:),x_tr(2,:),'go',x_val(1,:),x_val(2,:),'bx',x_test(1,:),x_test(2,:),'
rp');
ylabel('Y coordinate');
xlabel('X coordinate');
legend('Train data','Validation data','Test data',-2);
```

3.2.2. Cross-validation procedure. Grid search of parameters.

For cross validation procedure was organized special cycle where were verified C and Gamma parameters (recommended to verify parameters in log scale). By other words for different C and Gamma training data were validate on validation set, than was calculated **ClassRate** parameter, which show the percentage of right classification. By maximization of this parameter we choose proper C and Gamma. Algorithm and result of it present below.

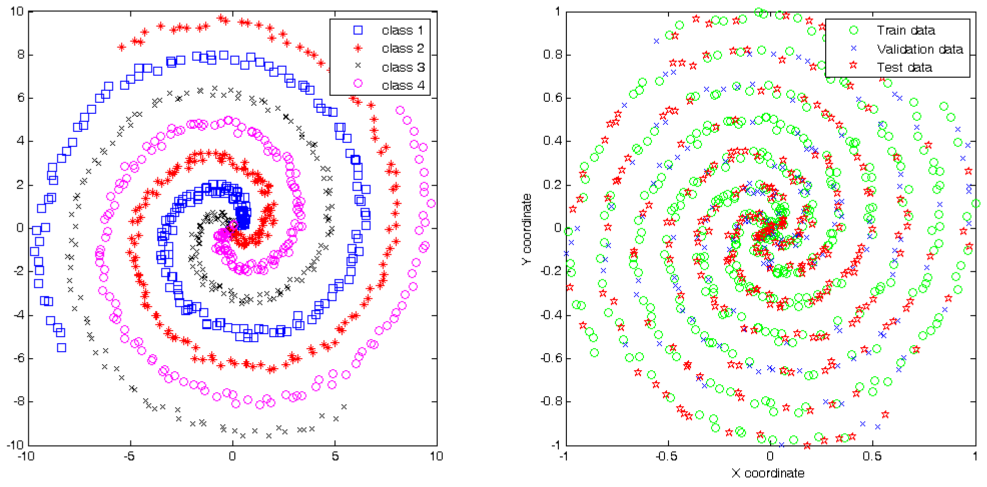


Figure 6. Creation of initial data and split it into 3 different sets.

```
%% Create classifier
Tr_Samples = x_tr;
Tr_Labels = y_tr;
Val_Samples = x_val;
Val_Labels = y_val;

nn = 11;
mm = 13;
C_in = 10^(-5);
Gamma_in = 10^(-10);
CC = zeros(1,nn);
GG = zeros(1,mm);
ClassRateMatr = zeros(nn,mm);
ClassRate=0;
fig=figure(2);
set(fig,'DoubleBuffer','on');
set(gca,'xlim',[-80 80],'ylim',[-80 80],...
       'NextPlot','replace','Visible','off')
mov = avifile('example.avi')                         % Create Movie

for j = 1:mm
     GG(j) = Gamma_in*(10^(j-1))
   for i = 1:nn
           CC(i) = C_in*(10^(i-1))
                [AlphaY, SVs, Bias, Parameters, nSV, nLabel] = RbfSVC(Tr_Samples,
Tr_Labels, GG(j), CC(i))
                [ClassRate, DecisionValue, Ns, ConfMatrix, PreLabels]=
SVMTest(Val_Samples, Val_Labels, AlphaY, SVs, Bias,Parameters, nSV, nLabel)
                ClassRateMatr(i,j) = ClassRate
               % MAke an AVI
                plot(SVs(1,:),SVs(2,:),'go',x_tr(1,:),x_tr(2,:),'rx');
                F = getframe(gca);
                mov = addframe(mov,F);
  end;
end;
mov = close(mov);
 figure(3);
 subplot(121);
 surf(log10(GG),log10(CC),ClassRateMatr);
 hold on;
 ylabel('log(C)');
 xlabel('log(Gamma)');
 zlabel('ClassRate');
 legend('ClassRate',1);
 [A,I] = max(ClassRateMatr);
 [A2,I2] = max(A);
 Gamma = GG(I2);
 C=CC(I(I2));
%% Precise determening of C and Gamma
for i = 1:5
    C_pr(i) = C*(10^(-1+0.5*(i-1)))
    for j = 1:5
            G_pr(j) = Gamma*(10^(-1+0.5*(j-1)))
             [AlphaY, SVs, Bias, Parameters, nSV, nLabel] = RbfSVC(Tr_Samples,
Tr_Labels, G_pr(j), C_pr(i))
             [ClassRate_pr, DecisionValue, Ns, ConfMatrix, PreLabels]=
SVMTest(Val_Samples, Val_Labels, AlphaY, SVs, Bias,Parameters, nSV, nLabel)
             ClassRateMatr_pr(i,j) = ClassRate_pr
    end;
end;
figure(3);
subplot(122);
 surf(log10(G_pr),log10(C_pr),ClassRateMatr_pr);
 ylabel('log(C)');
 xlabel('log(Gamma)');
 zlabel('ClassRate');
 legend('ClassRate',1);
 [A_pr,I_pr] = max(ClassRateMatr_pr);
 [A2_pr,I2_pr] = max(A_pr);
 Gamma2 = G_pr(I2_pr);
 C2=C_pr(I_pr(I2_pr));

%% Save Classifier
[AlphaY, SVs, Bias, Parameters, nSV, nLabel] = RbfSVC(TR_X, TR_Y, Gamma2, C2);
 save IVAN_SVMClassifier_01 AlphaY SVs Bias Parameters nSV nLabel;
```

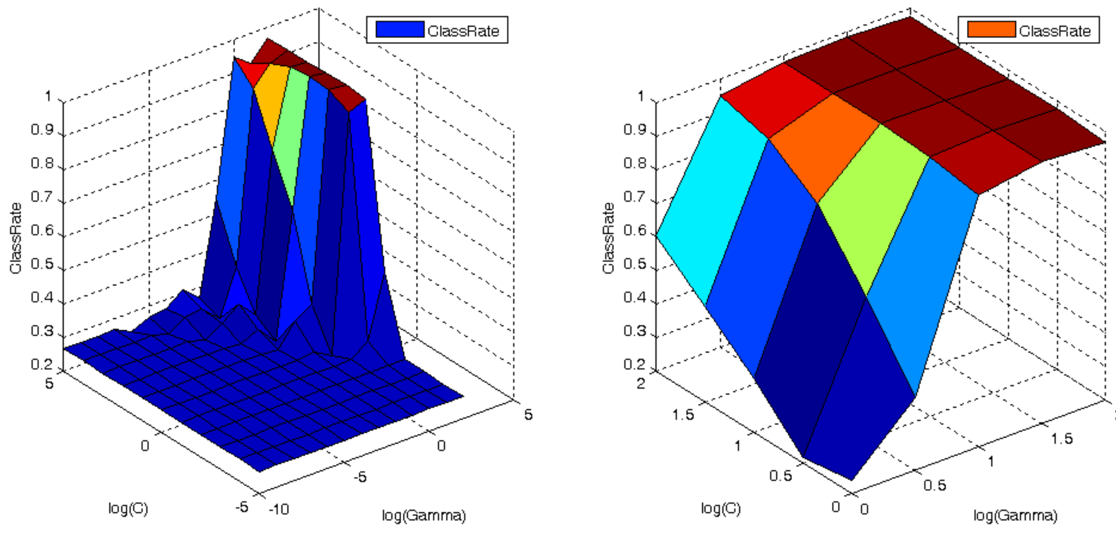


Figure 7. ClassRate dependence of C and Gamma.

Used grid search with step of one decade on logarithmic scale.

From this grid search was found the best ClassRate = 0.9752 and corresponds parameters C = 100, Gamma = 31.6228.

3.2.3. Applying Classifier and results of classification.

After Classifier is ready, we can try to apply it to the testing set of data, also shoe result of classification (code of classification plotting look at previous chapter):

```
%% Classify input patterns using the constructed nonlinear SVM Classifier
load IVAN_SVMClassifier_01;
Test_Samples = x_test;
Test_Labels = y_test;
[Labels, DecisionValue]= SVMClass(Test_Samples, AlphaY, SVs, Bias, Parameters,
nSV, nLabel);
[ClassRate_fin, DecisionValue, Ns, ConfMatrix, Labels]= SVMTest(Test_Samples,
Test_Labels, AlphaY, SVs, Bias,Parameters, nSV, nLabel);

[Labels_new,IX] = sort(Labels,2);
N=size(Labels);
figure(4);
for i = 1:N(2)
Test_Labels_new(i)=Test_Labels(IX(i))
end;
%Compare the resultant labels with the true labels of the data
plot(Test_Labels_new(1,1:N(2)),'r.');
hold on;
plot(Labels_new(1,1:N(2)),'-b');
ylabel('Class Index');
xlabel('Pattern Index');
legend('True Labels','Resultant Labels',0);
```

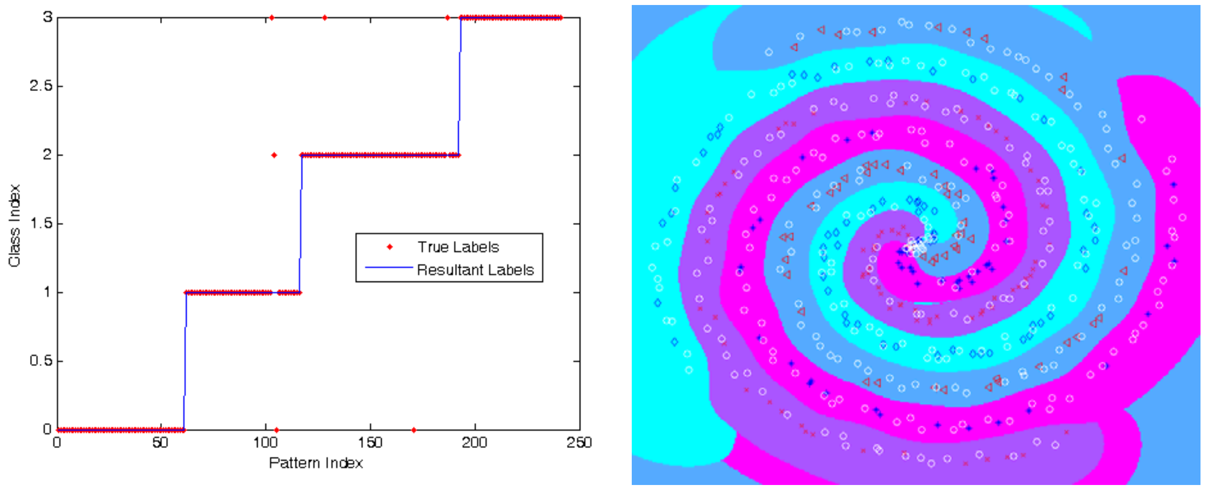
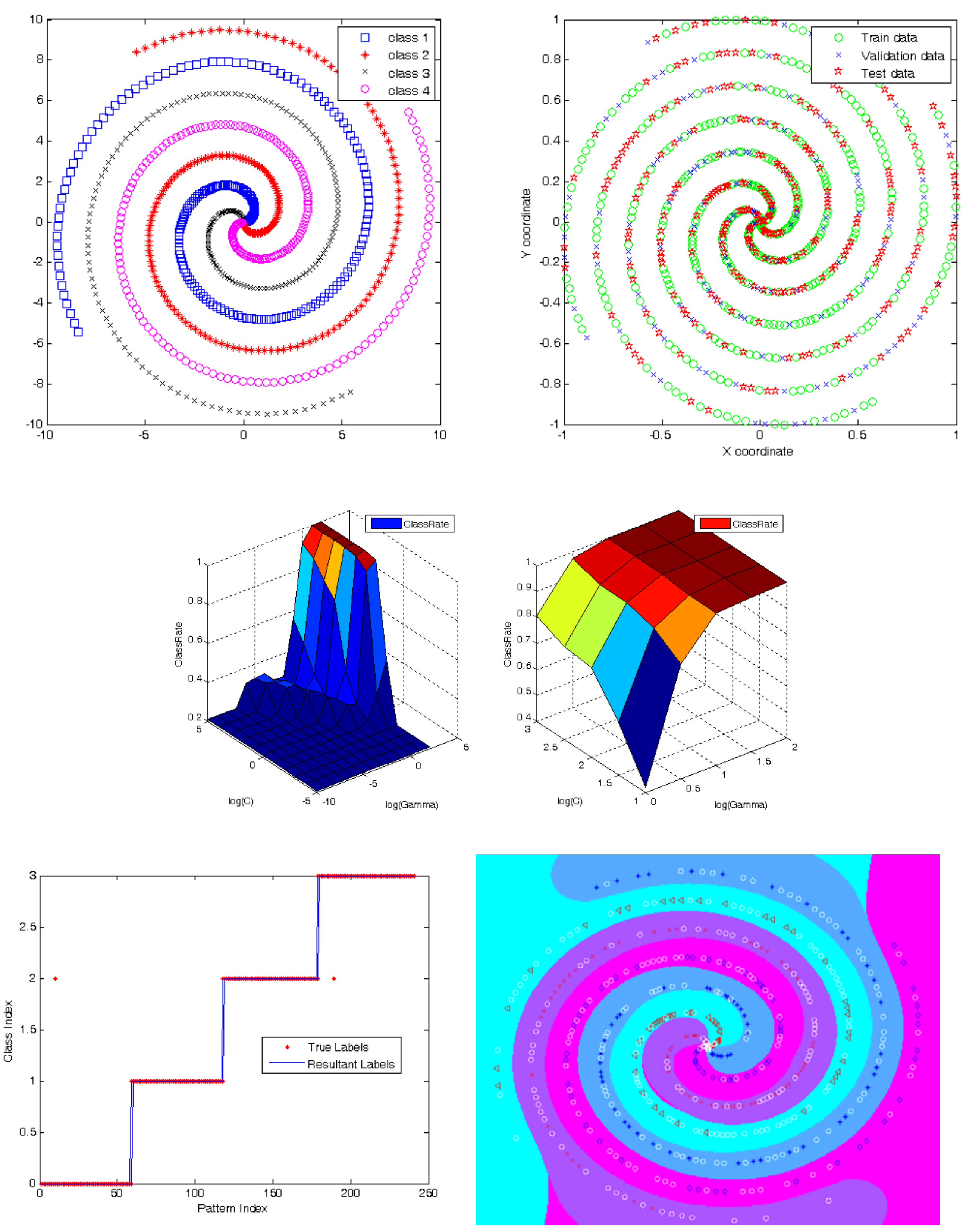


Figure 8. Result of advanced methodology (RBF kernel, N = 800, Spread parameter = 0.5, C = 100, Gamma = 31.63, ClassRate = 0.9751 = 97.51%).

Also this result show quite good generalization of obtained solution because the borders between separated classes are smooth. Also the small error on classification means that we don't have overfitting of solution.

3.2.4. Case of very smooth spirals.

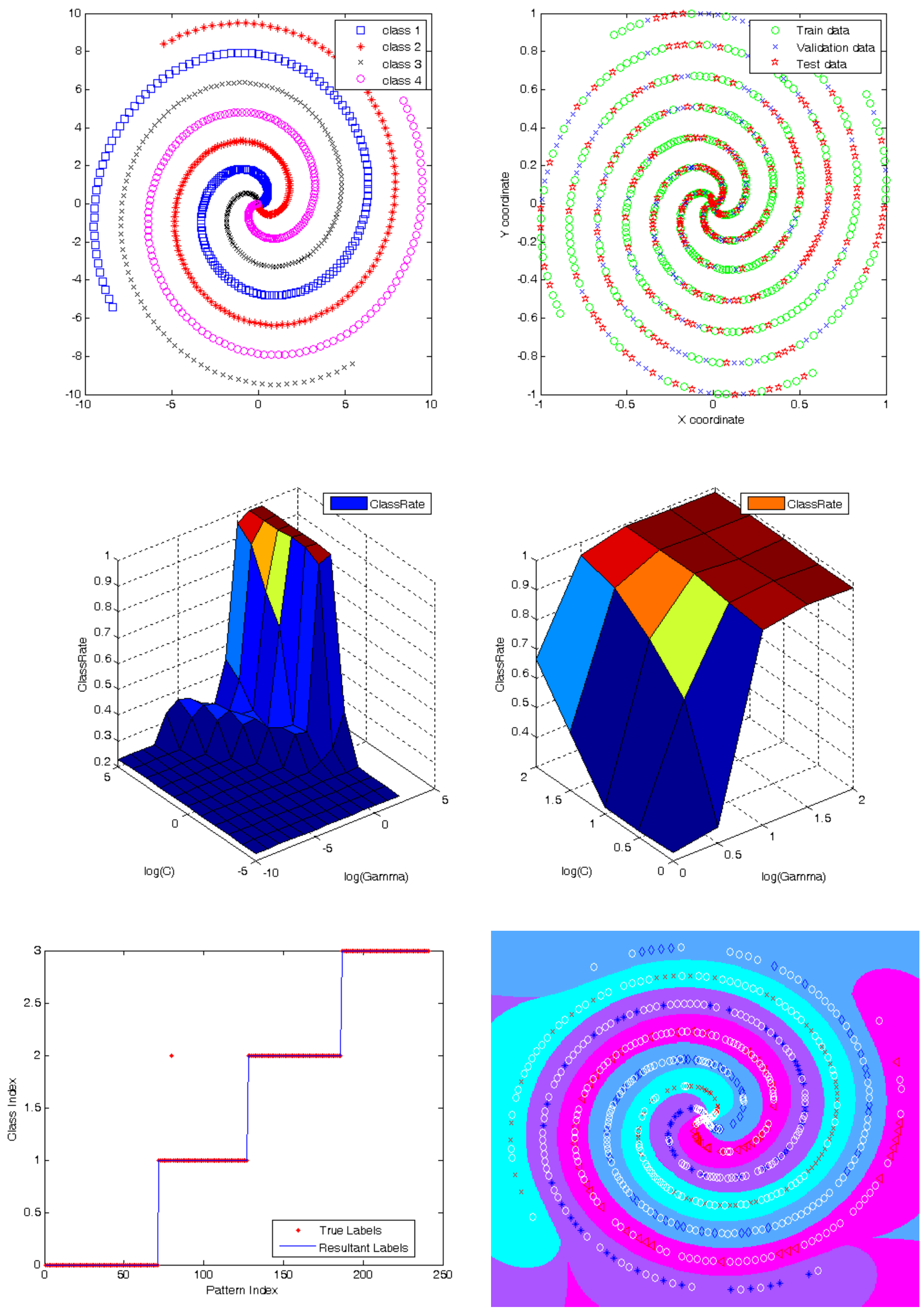
For smooth spirals (spread parameter = 0):



Final ClassRate = 99.18%

Figure 9. Results of RBF kernel realization for smooth spirals. N = 800.

Another repetition for smooth spiral:

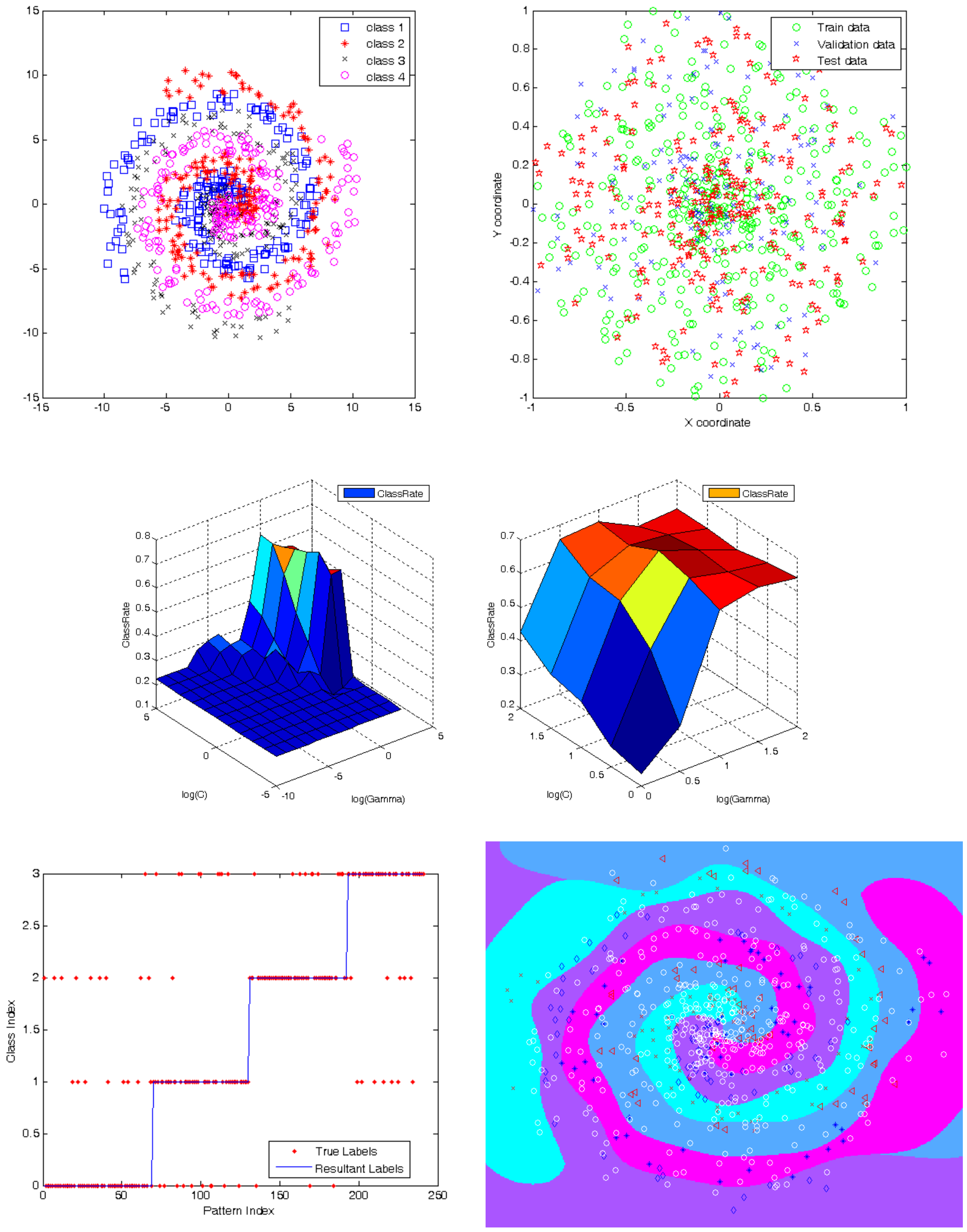


Final ClassRate = 99.59%

Figure 10. Results of RBG kernel realization for smooth spirals. N = 800.

3.2.5. Case of very spread spirals.

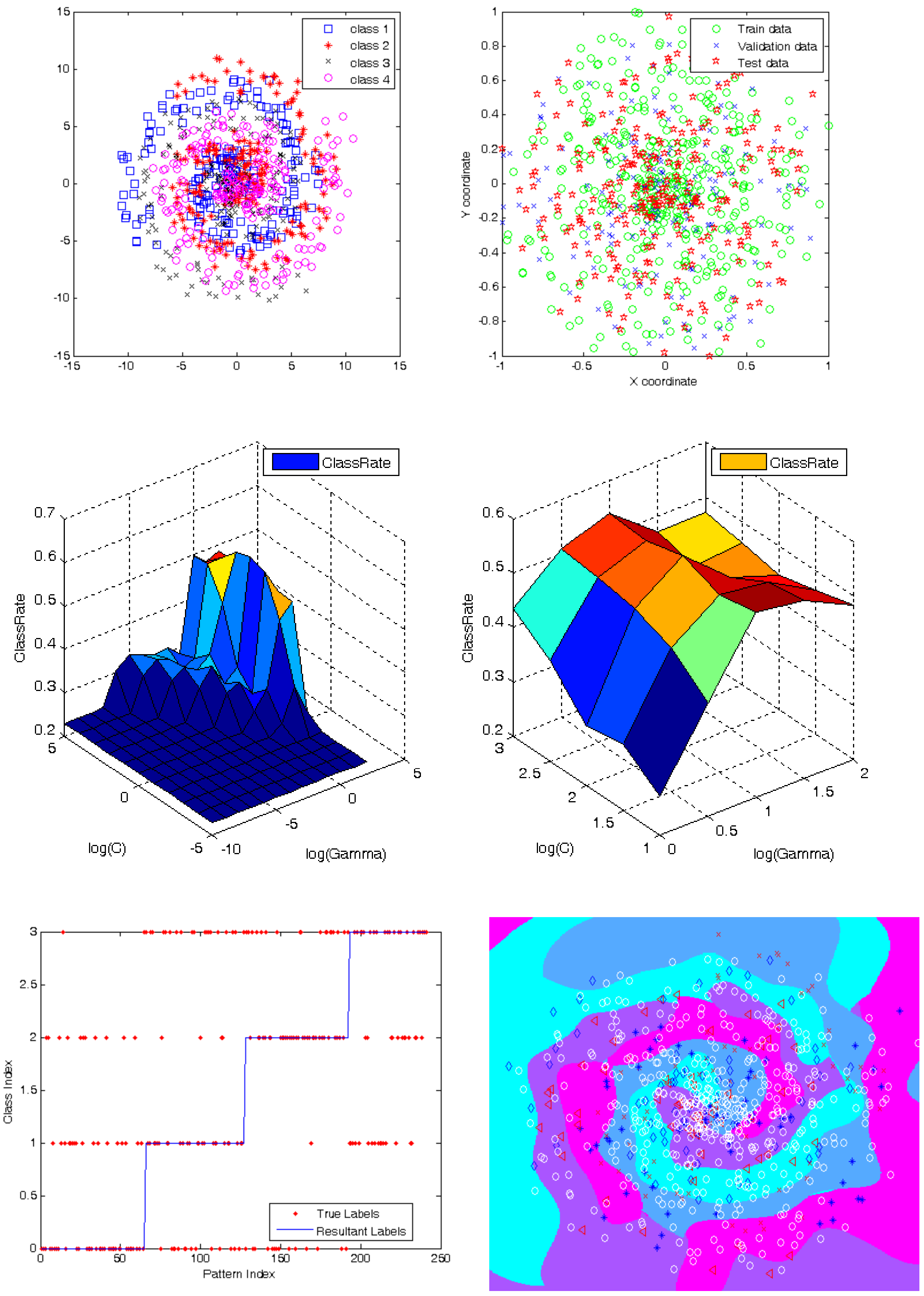
For spread spirals (spread parameter = 2):



Final ClassRate = 73.44%

Figure 11. Results of RBF kernel realization for spread spirals. N = 800.

For VERY spread spirals (spread parameter = 3):



Final ClassRate = 46.89%

Figure 12. Results of RBF kernel realization for very spread spirals. N = 800.

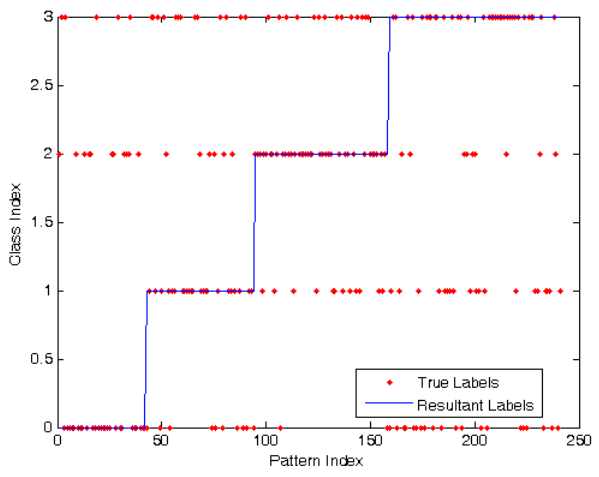
3.3. Advanced methodology. Polynomial kernel.

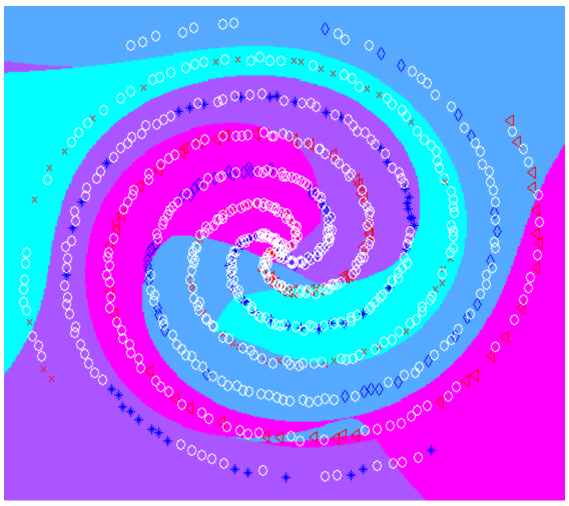
3.3.1. Difference with RBF Kernel.

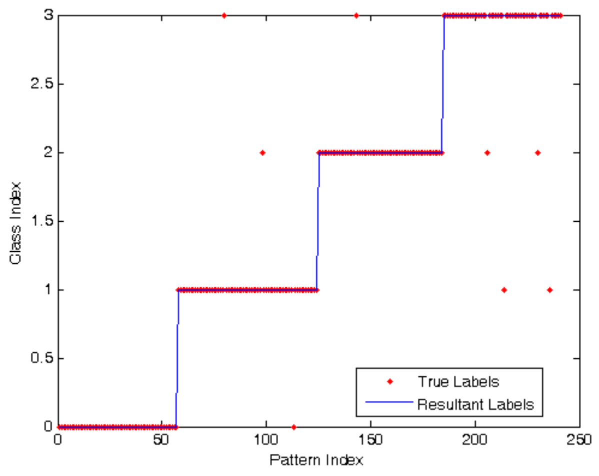
In case of using polynomial kernel whole procedure is almost the same, but we have not just 2 parameters (Gamma and C) like in previous case, but 3 parameters: C, Gamma and Degree (see page 3). So the procedure of Grid-Search can by modify, or one of parameters (for example Degree) ban be a constant.

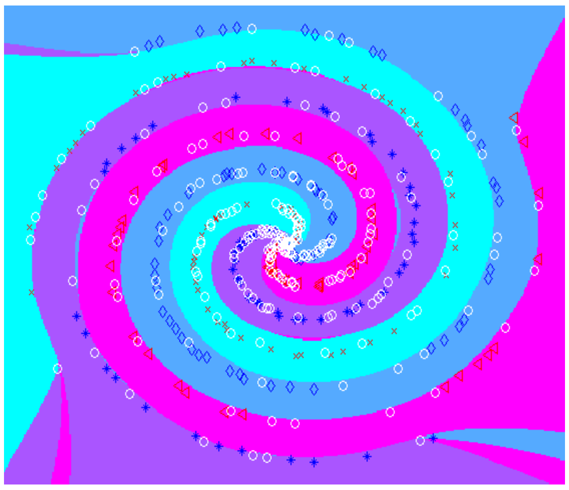
After some trying of the same procedure with one constant parameter we can make some conclusions. For fix Degree parameter (use the same grid search and cross-validation procedure) and Number of data points N = 800 (Figure 13):

- small Degree (below 10) bad result
- Degree about 10 – good result
- Degree higher than 10 – very long performance of classifier and not the best result of classification.









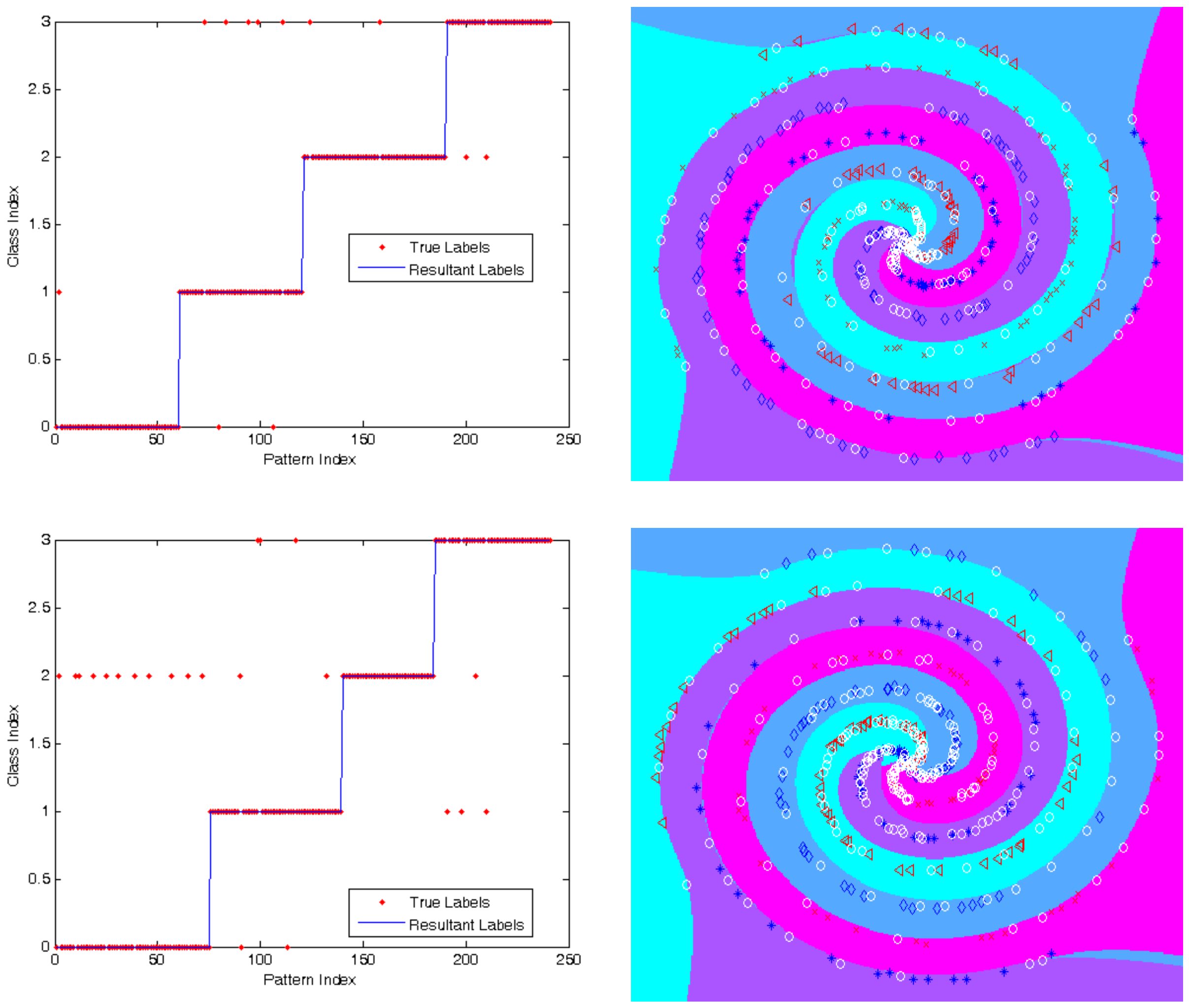


Figure 13. Polynomial Classifier for 4 cases:

Degree = 5, 10, 15 and 20

ClassRate = 49.79%, 97.51%, 95.02% and 90.87%.

This results means that high Degree doesn't really improve classifier (but logically it can be so)! At the same time the time of preparing classifier becomes much higher which slow your process. Actually this also mean that the more "bended" data we have – the bigger value of degree we need, in our case this "bending" of data is just number of points.

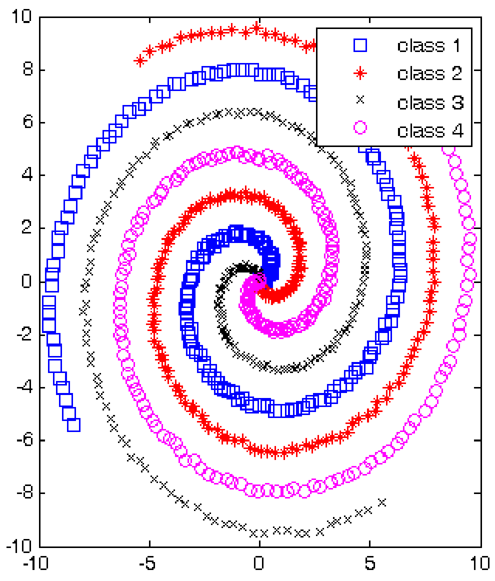
So we should perform grid search for 3 parameters: C, Gamma and Degree of polynomial kernel. Perform such calculations and compare result with previous obtained.

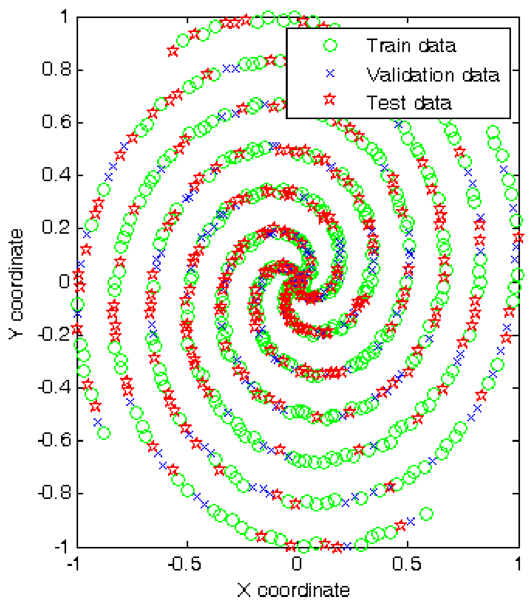
3.3.2. Advanced performance of polynomial classifier.

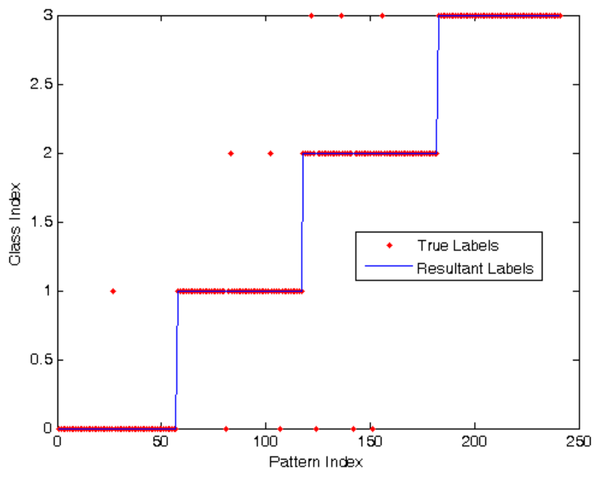
After preparing polynomial classifier with 3 parameters search result of classification more or less guarantee.

```
C_in = 1;
Degree = 2;
Coeff = 1;

nn = 5;
mm = 6;
kk = 6;
Deg_in = 5;
Gamma_in = 1;
DD = zeros(1,nn);
GG = zeros(1,mm);
ClassRateMatr = zeros(nn,mm);
ClassRate = 0;
for k = 1:kk
    CC(k) = C_in*0.5*k
for i = 1:nn
    DD(i) = Deg_in + (i-1)*4
   for j = 1:mm
              GG(j) = (Gamma_in*(0.3*j))
              [AlphaY, SVs, Bias, Parameters, nSV, nLabel] = PolySVC(Tr_Samples,
Tr_Labels, DD(i), CC(k), GG(j), Coeff)

              [ClassRate, DecisionValue, Ns, ConfMatrix, PreLabels]=
SVMTest(Val_Samples, Val_Labels, AlphaY, SVs, Bias,Parameters, nSV, nLabel)
              ClassRateMatr(i,j,k) = ClassRate;

   end;
end;
end;
```

So non precise search of course give us not the best possible results, but with smaller grid step performance will take much more time. Best parameters from this concrete search was: Degree = 9, C = 3, Gamma = 1.8, ClassRate = 95.44%.







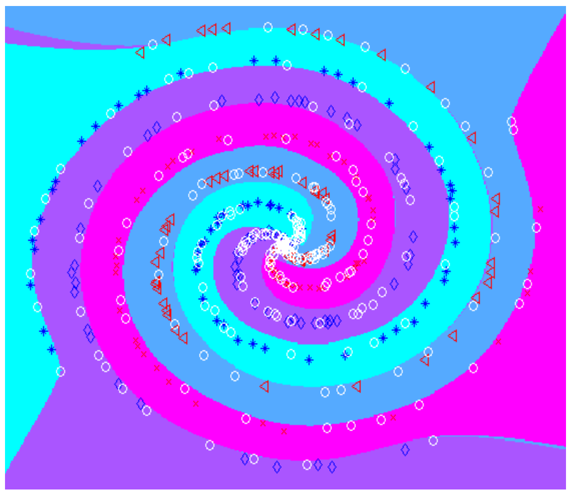
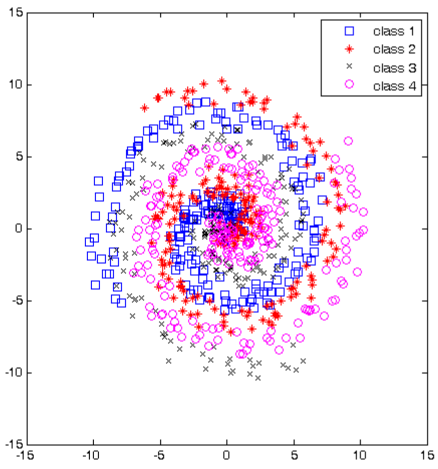
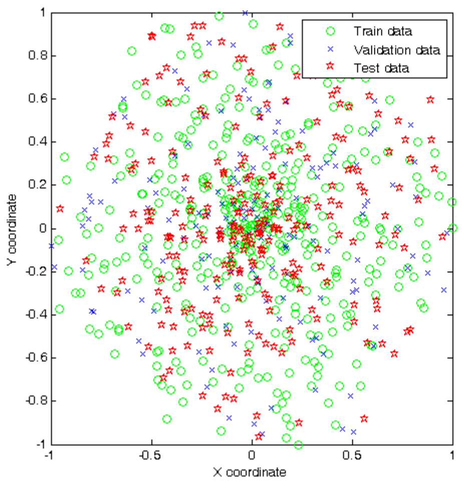
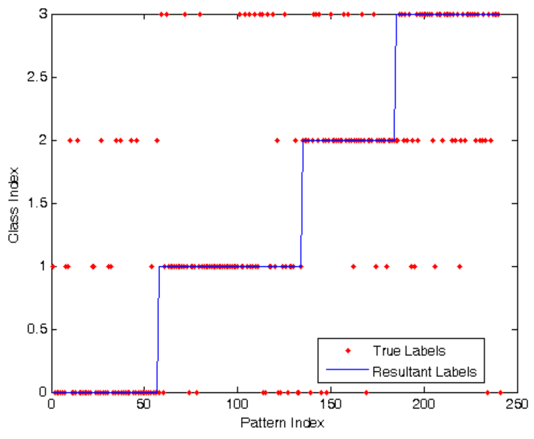


Figure 14. Result of polynomial kernel classification with grid search method. N = 800, Spread parameter = 0.2, ClassRate = 95.44%

Change parameters of initial data sets and verify procedure once again:







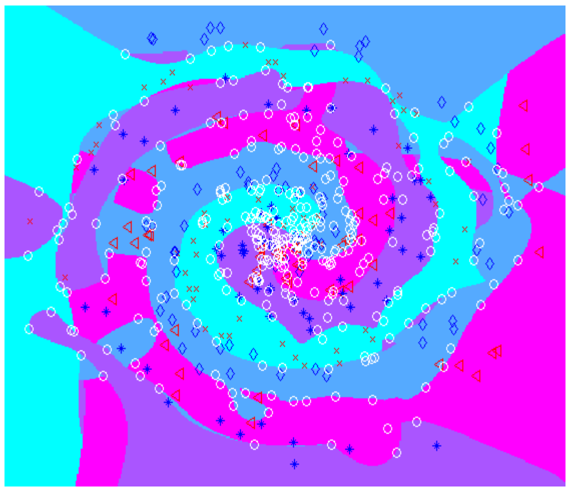


Figure 15. Classification result od spread spirals by polynomial kernel. N = 800, Spread parameter = 2, ClassRate = 66.8 %

3.4. Verification of RBF classifier for different initial data sets .

Train Classifier based on RBF kernel and that change initial data for such classification problem. So let’s check how good is obtained classifier. Parameters for initial training of classifier was: N = 800, spread = 2. Result of first training present on figure 16.

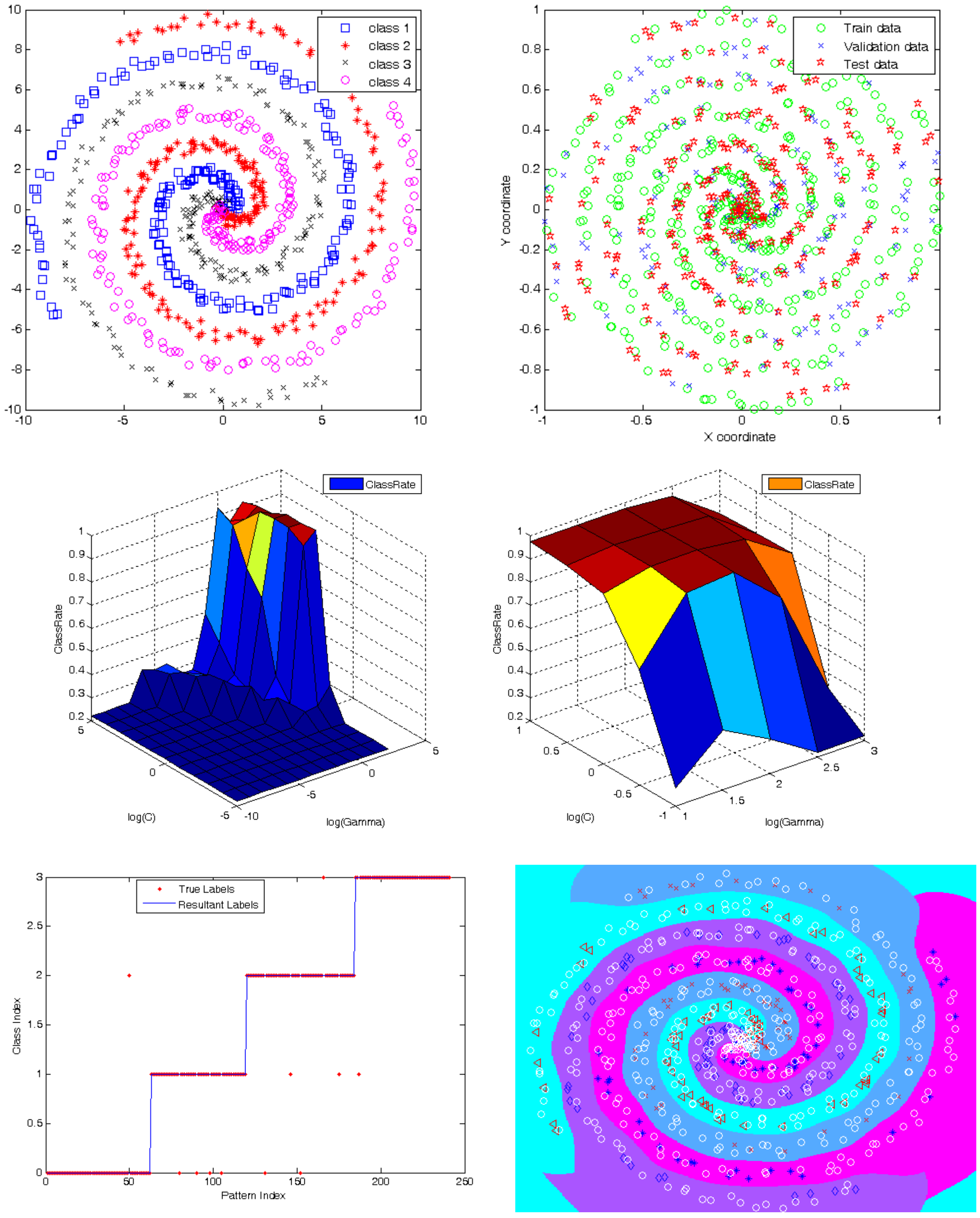


Figure 16. Result of performance RBF classifier. ClassRate = 95.44 %.

New initial data has parameters: N = 800, spread = 0.1. Result of applying trained classifier (train on initial data but with already found C and Gamma) is on Figure 17.

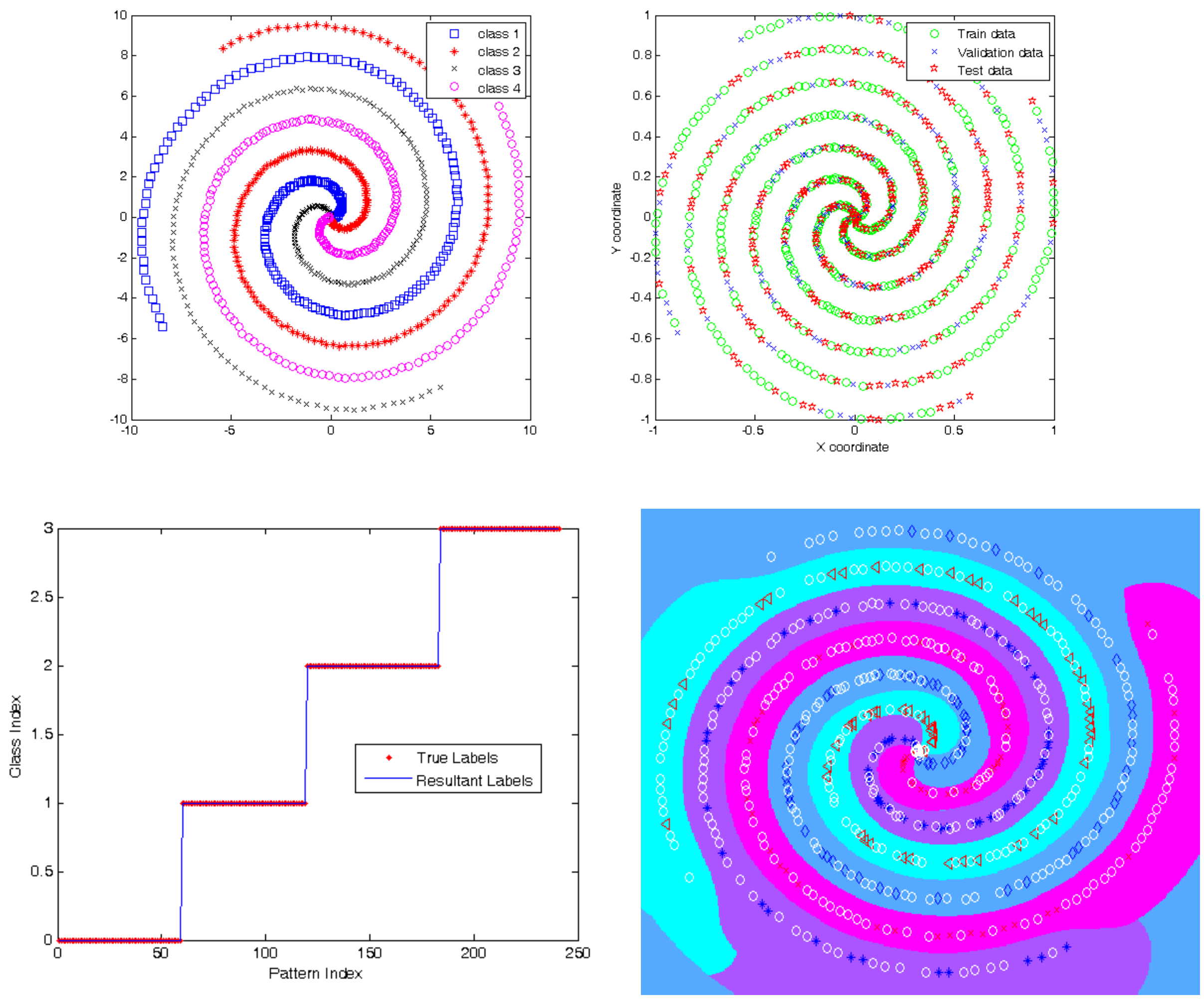
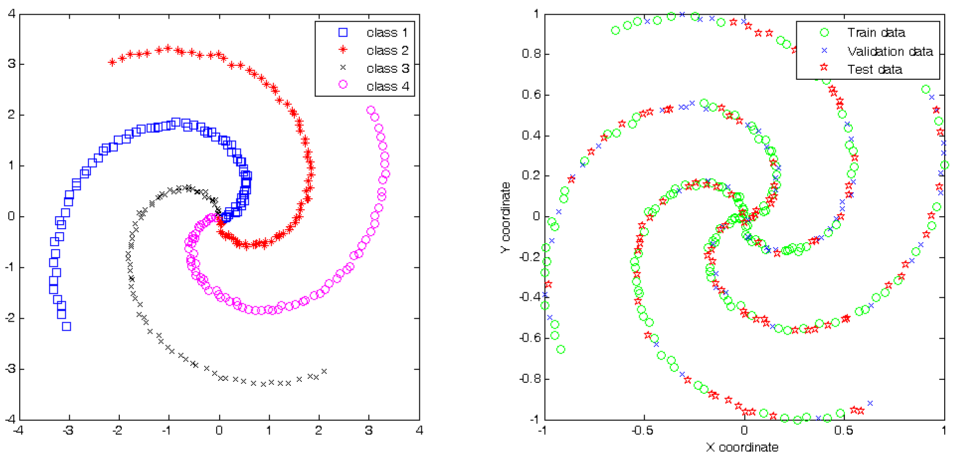
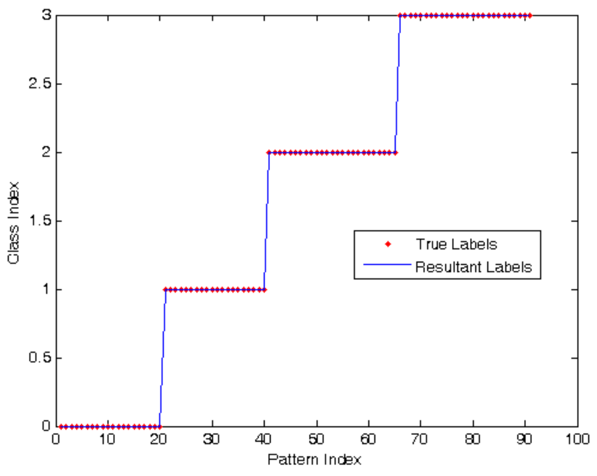


Figure 17. Result of applying trained classifier to new data. ClassRate = 100 %.

If we decrease number of data points (length of spirals) classification will be still perfect: N = 300, spread = 0.1





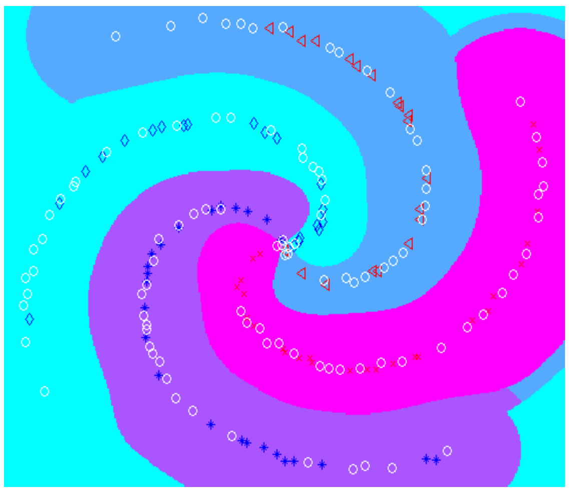
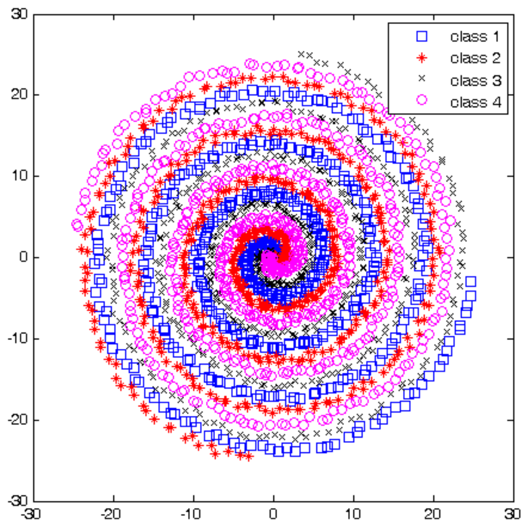
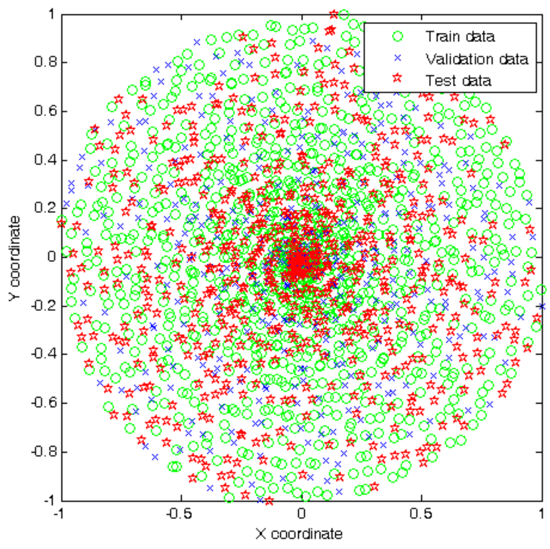
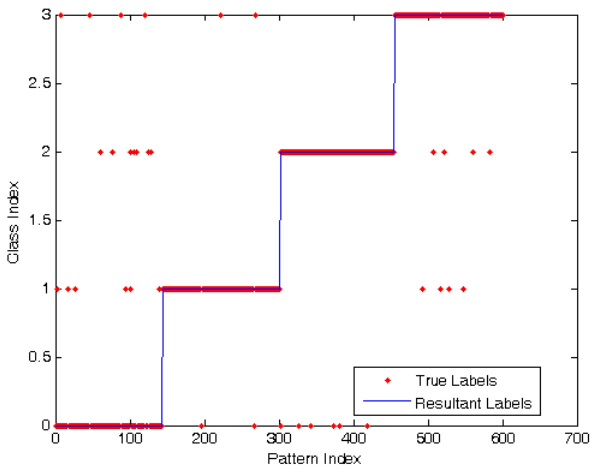


Figure 18. Result of applying trained classifier to new data. ClassRate = 100 %.

Let's consider increasing numbers of elements and spread: N = 2000, spread = 1







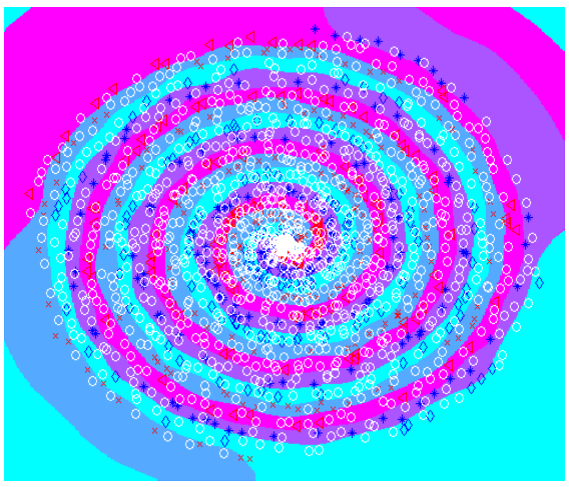


Figure 19. Result of applying trained classifier to new data. ClassRate = 93.79 %.

If we increase numbers to 5000 points – result will be unstable and bad. That means that found parameters C and Gamma quite good for some range near initial set parameters, but with much increases procedure of cross-validation train and grid search should be repeated.

4. CONCLUSIONS.

During work on seminar was learned approach of using SVM, basic theoretical knowledge in this field, some features to get more or less “good result”, realize SVM methods in Matlab package.

Was shown that for 4-class spiral problem better to use RBF kernel of SVM. Procedure of grid-search and cross-validation, as was recommended in guide, really works.

Other parameters of Train functions of SVM, especially type of SVM (c-, nu-, one-class, etc,) actually have no effect on result (it wasn’t shown in report). So we can use default settings of this function.

Performance of RBF-Classifier can take some time, and it’s similar to find best structure of Neural Networks. But training of Classifier is much faster than training of NN.