Exercise 5

Philipp Hanslovsky, Robert Walecki

May 31, 2012

5.1.1

$$
\begin{aligned}
\boldsymbol{x} &= (x_1, x_2)^T \\
k(\boldsymbol{x}, \boldsymbol{z}) &= (\boldsymbol{x}^T \boldsymbol{z})^3 \\
&= x_1^3 z_1^3 + 3 x_1^3 z_1^2 x_2 z_2 + 3 x_1 z_1 x_2^2 z_2^2 + x_2^3 z_2^3 \\
\Rightarrow \boldsymbol{\Phi}(\boldsymbol{x}) &= (x_1^3, \sqrt{3} x_1^2 x_2, \sqrt{3} x_1 x_2^2, x_2^3)^T \\
\Phi : & \quad \mathbb{R}^2 \mapsto \mathbb{R}^4
\end{aligned}
$$

5.1.2

$$
\begin{aligned}
& k(\boldsymbol{x}, \boldsymbol{z}) = c * k_1(\boldsymbol{x}, \boldsymbol{z}) && c = const && (1) \\
& k(\boldsymbol{x}, \boldsymbol{z}) = \exp\left(k_1(\boldsymbol{x}, \boldsymbol{z})\right) && && (2) \\
& k(\boldsymbol{x}, \boldsymbol{z}) = k_1(\boldsymbol{x}, \boldsymbol{z}) + k_2(\boldsymbol{x}, \boldsymbol{z}) && && (3) \\
& k(\boldsymbol{x}, \boldsymbol{z}) = k_1(\boldsymbol{x}, \boldsymbol{z}) \cdot k_2(\boldsymbol{x}, \boldsymbol{z}) && && (4) \\
& && && (5)
\end{aligned}
$$

$$
\begin{aligned}
k(\boldsymbol{x}, \boldsymbol{z}) &= \exp\left(-\frac{(\boldsymbol{x} - \boldsymbol{z})^2}{2\sigma^2}\right) && (6) \\
&= \prod_{i=1}^{3} \exp\left(-\frac{k_i}{2\sigma^2}\right) && (7) \\
k'(\boldsymbol{x}, \boldsymbol{z}) &= \boldsymbol{x}^T \boldsymbol{z} = \boldsymbol{\Phi}^T \boldsymbol{\Phi} && (8) \\
k_1 &= k'(\boldsymbol{x}, \boldsymbol{x}) && (9) \\
k_2 &= k'(\boldsymbol{z}, \boldsymbol{z}) && (10) \\
k_3 &= k'(\boldsymbol{x}, \boldsymbol{z}) && (11) \\
\Phi &: (x_1, \ldots, x_n) \mapsto (x_1, \ldots, x_n) && (12) \\
& && (13)
\end{aligned}
$$

k_i are valid kernels as k' can be expressed by a scalar product of a mapping of $\boldsymbol{x}$ and $\boldsymbol{z}$. According to equation 1 $\tilde{k}_i = -\frac{k_i}{2\sigma^2}$ are valid kernels as well with $c = -\frac{1}{2\sigma^2}$. From equation 2 we can see that $\exp\left(\tilde{k}_i\right)$ are valid kernels as well. Finally a product of valid kernels results in a valid kernel (equation 4).

Listing 1: Gaussian kernel response in Matlab

```
function k = gaussian_kernel(x, z, sigma)
% x are matrices of size pxn, containing n (number of samples) feature
% vectors of size p. z is a vector of size p. sigma is a scalar.

n = size(x, 2);
k = zeros(1, n);
SGM = 2*sigma^2;
z = z';

for i=1:n
    arg = x(:,i) - z;
    k(i) = exp(-(arg'*arg)/(SGM));
end
end
```

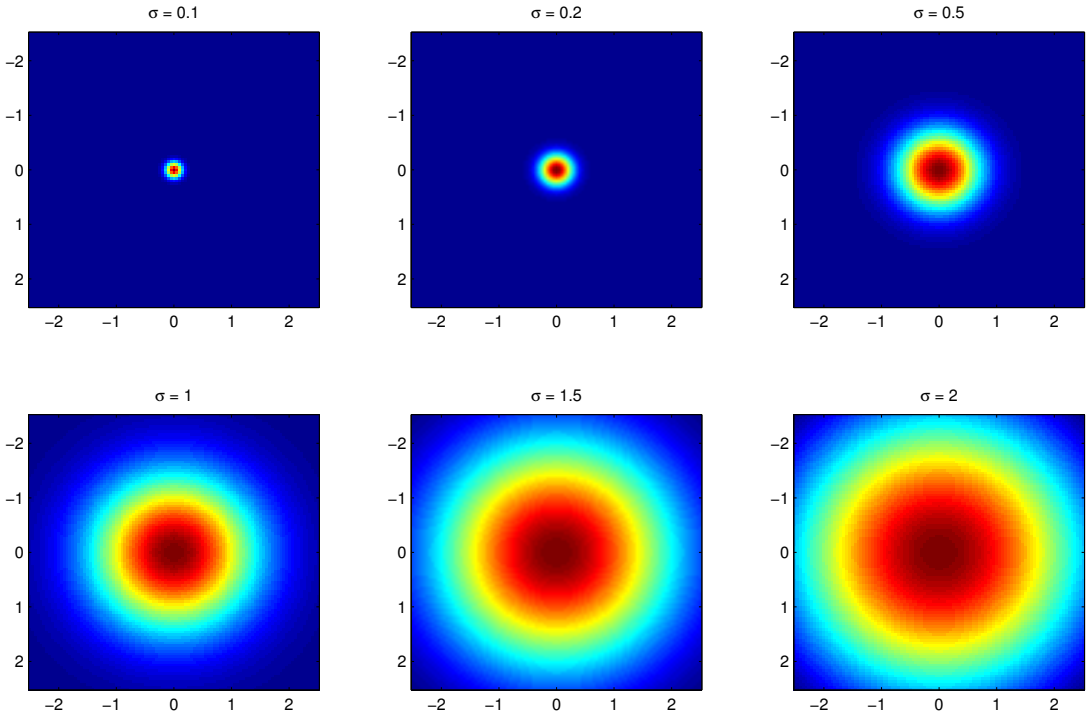


Figure 1: Responses for various values of σ (see plot titles).

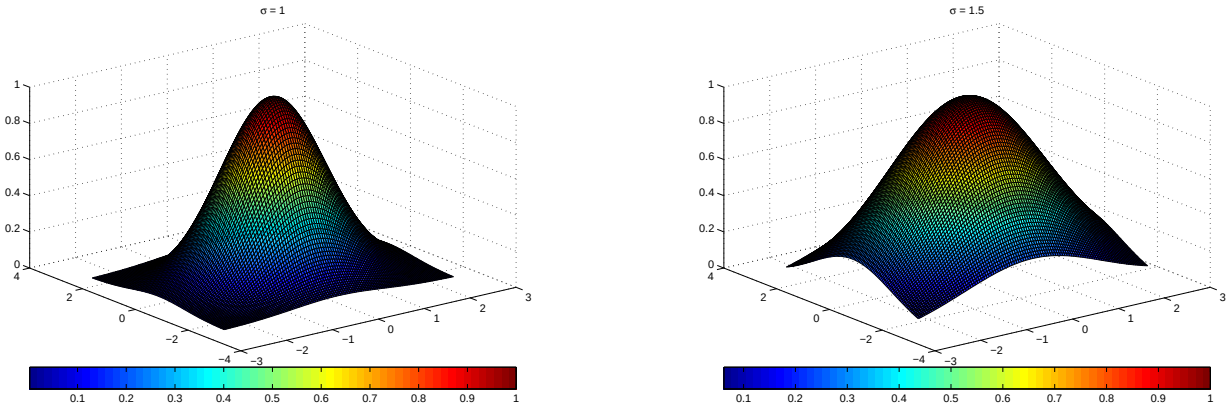
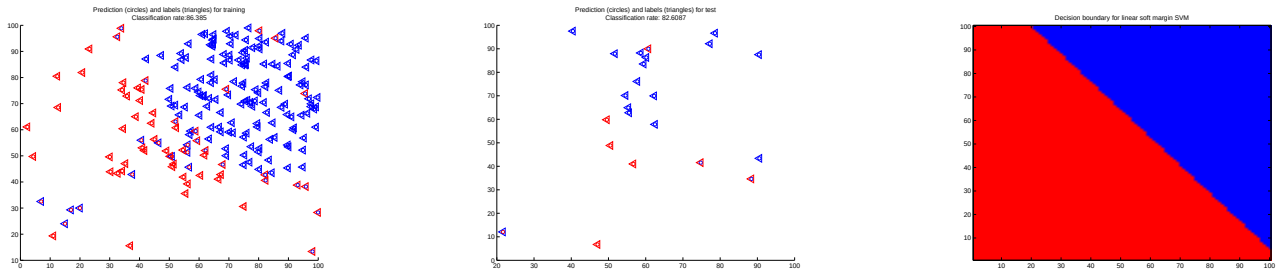


Figure 2: surface plots for $\sigma = 1$ and $\sigma = 1.5$

5.2.1

The linear SVM classifies 83%/86% of test/training correctly (see figure 3). That leaves room for improvement.



(a) training **(b)** test **(c)** test

Figure 3: Classification rates on test/training as well as decision boundary for a linear SVM

5.2.2

Increasing γ will make the decision boundary approximate the samples of the training dataset very closely (will go up to 100% classification rate). That agrees with the Gaussian kernel, that will be very narrow (δ distribution like) for large values of γ/small values of σ. However, du to this approximation of the training dataset, the classification rate for the test dataset will drop significantly (as low as 70%).Therefore γ should not be chosen too large. $\gamma = 0.01$ resulted in the best classification rate for the test dataset and will be used for investigations on C (see figures 4 - 6). C was set to 1 while varying γ.

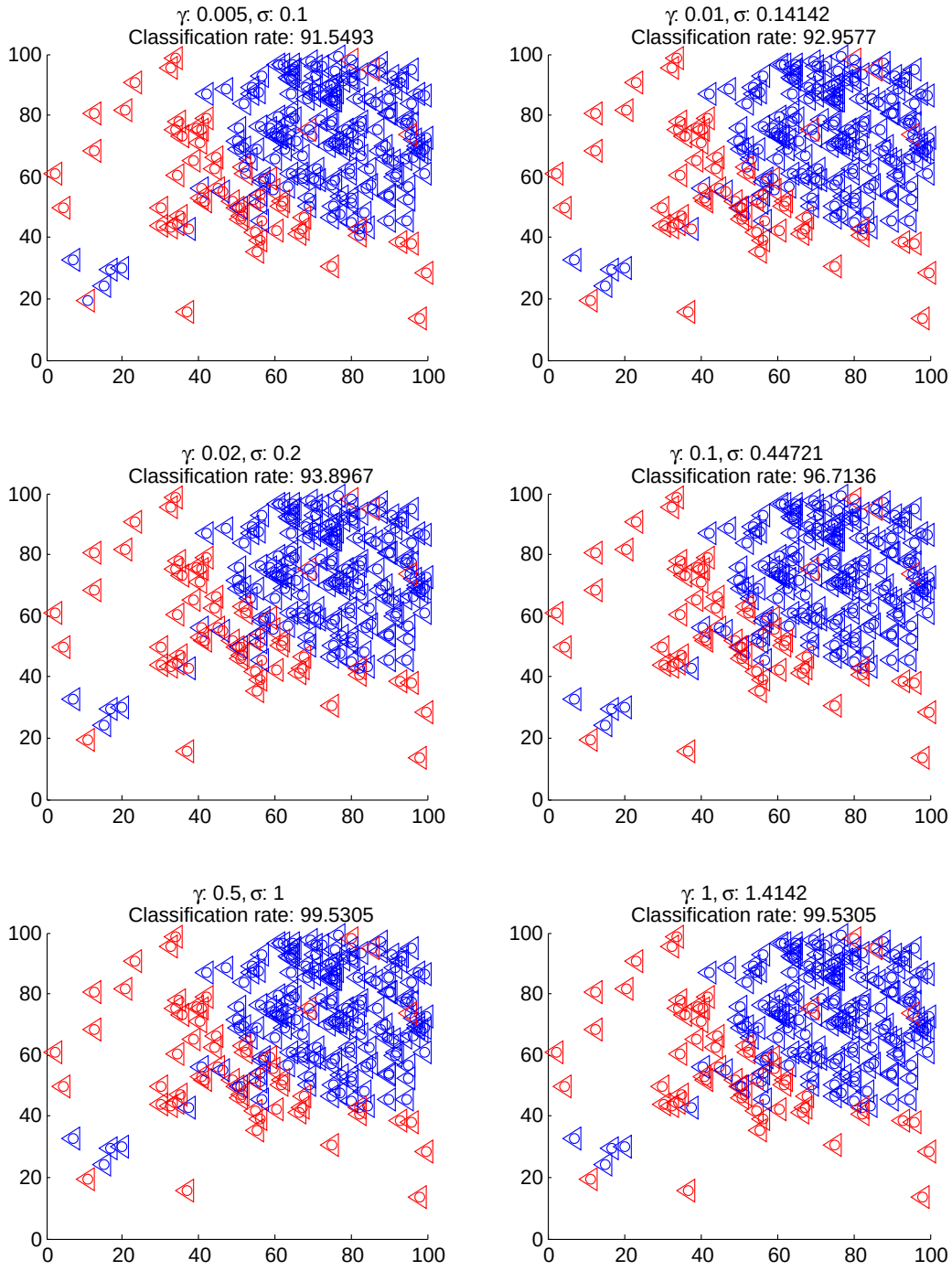


Figure 4: Prediction and labels for training dataset for various γ

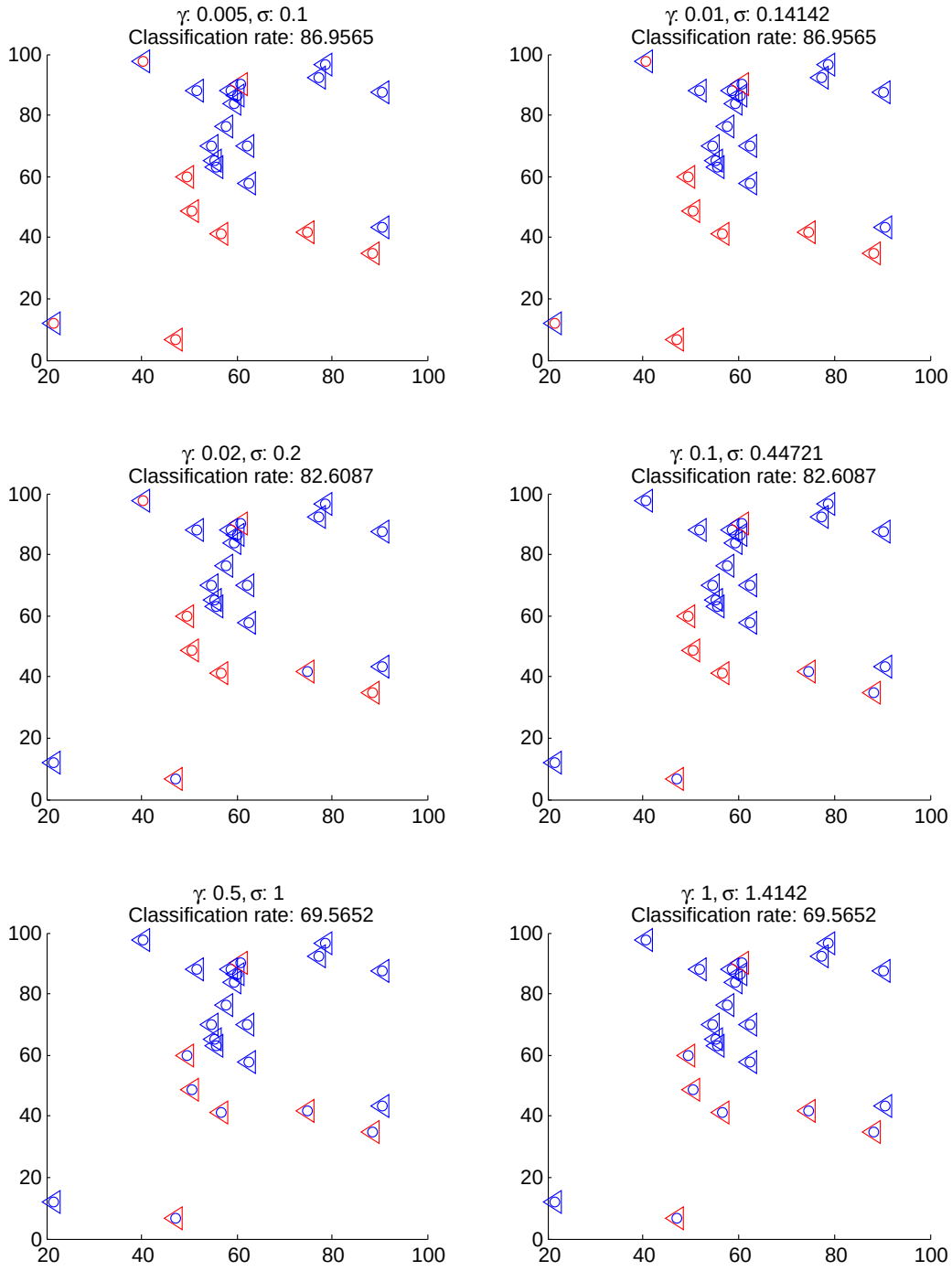


Figure 5: Prediction and labels for test dataset for various γ

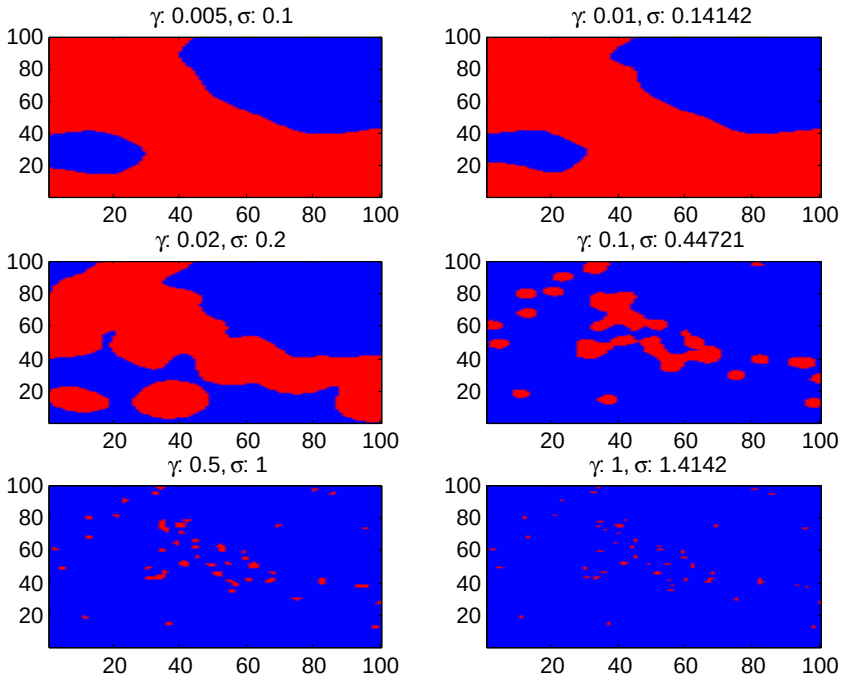


Figure 6: Decision boundary for various γ

Increasing C, which can be interpreted as the cost for wrong decisions, one can force the SVM to try and get at 100% classification rate on the training set. In contrast to increasing γ this does not lead to an approximation of the training dataset as close as for high a γ and therefore the classification rate in the test dataset does not drop as dramatically as for (given an appropriate choice for γ). There is an optimal rate for $C = 0.5$ (91%) and for larger values of C the rate seems to be constant at 87% (see figures 7 - 9). Therefore a grid search for parameter optimzation is a good idea for finding optimal parameters γ and C.

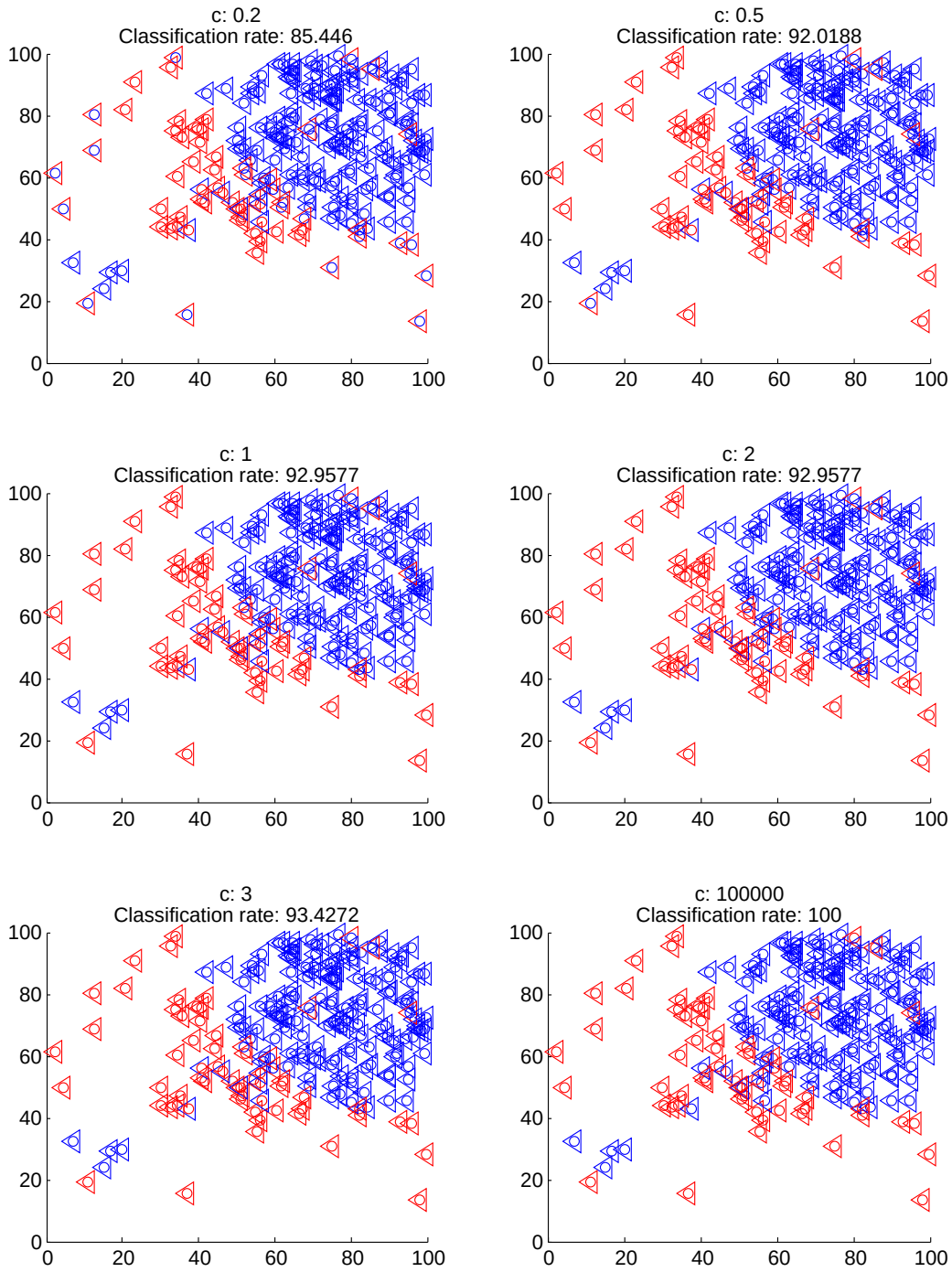


Figure 7: Prediction and labels for training dataset for various γ

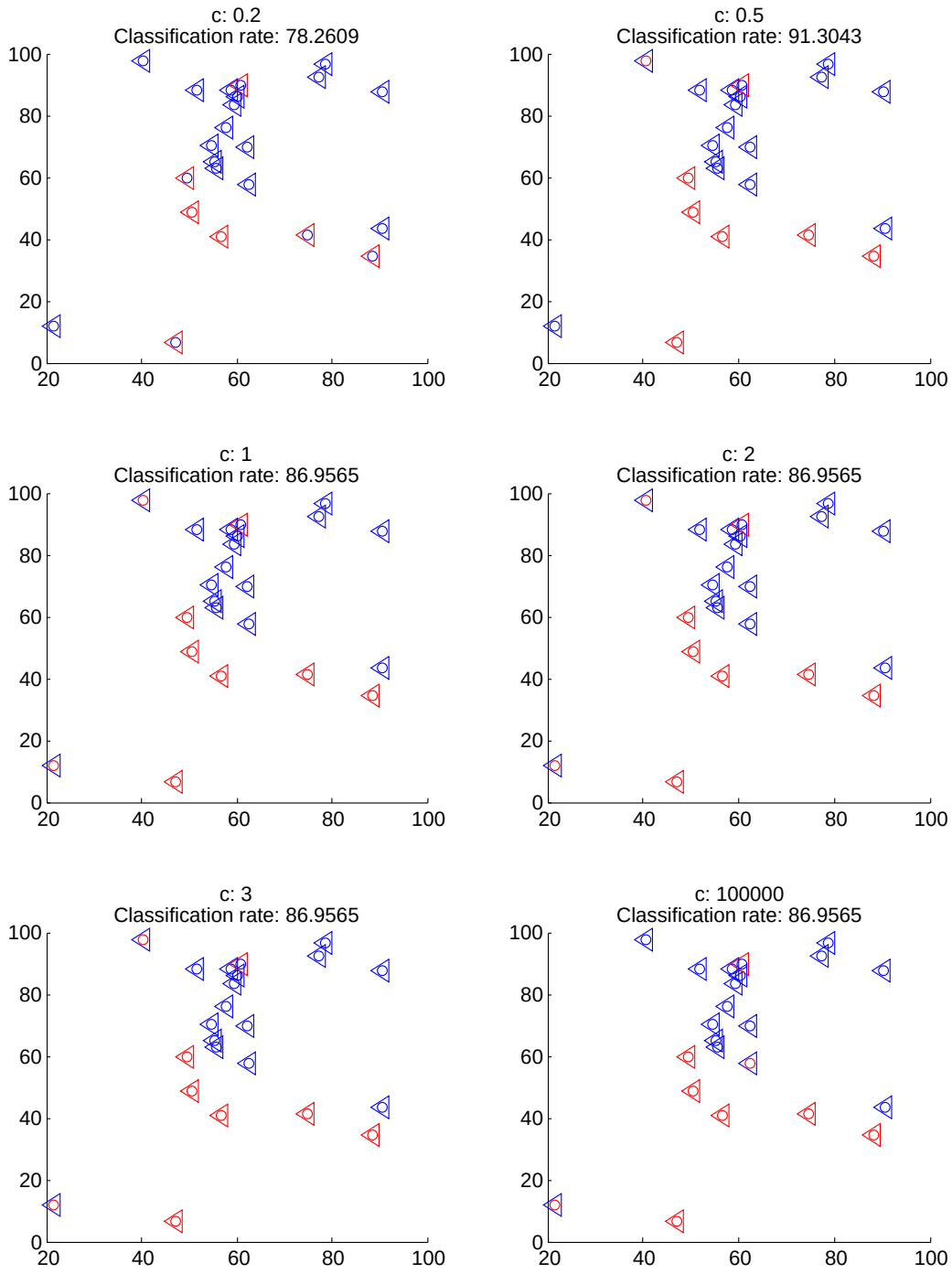


Figure 8: Prediction and labels for test dataset for various γ

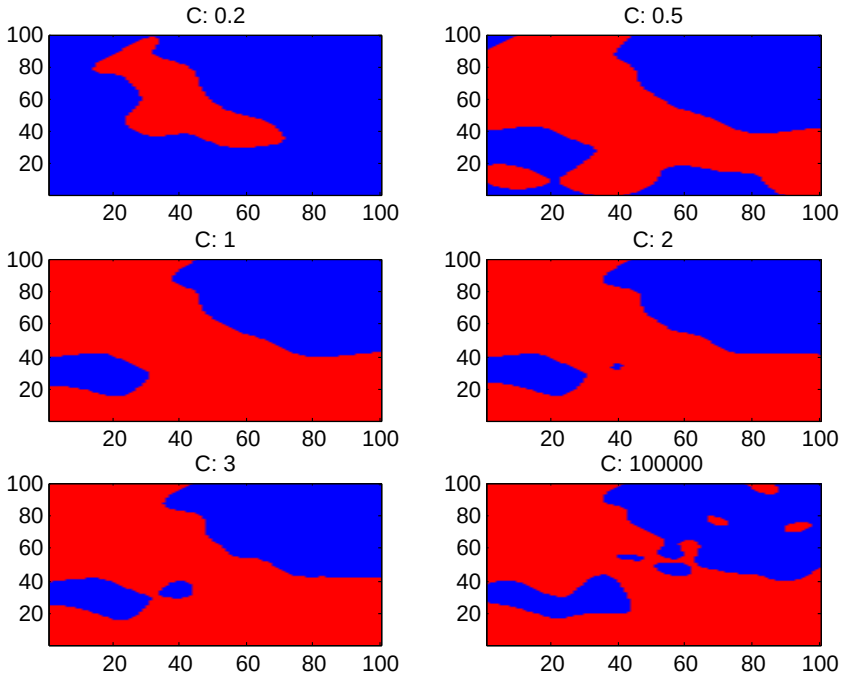
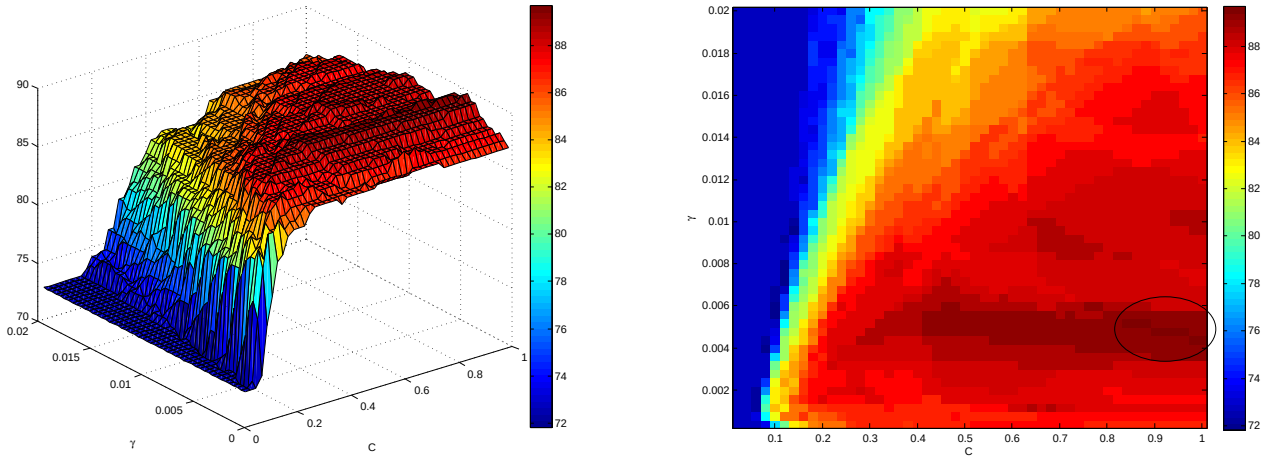


Figure 9: Decision boundary for various γ

5.2.3

We used a 50x50 grid for $\gamma \in [0.0004, 0.02]$ and $C \in [0.02, 1]$. The resulting cross-validation rates are shown in a surface plot as well as in a 2D score plot (see figure 10). 9 compostions of γ and C result in the best classification rate of 90% (see table 1).



(a) surface plot **(b)** score plot

Figure 10: Surface/score plots for cross-validation. The 9 compositions with the best scores are indicated by the ellipse in the score plot.

γ	0.0052	0.0052	0.0052	0.0048	0.0048	0.0048	0.0048	0.0044	0.0044
C	0.8400	0.8600	0.8800	0.9000	0.9200	0.9400	0.9600	0.9800	1.0000

Table 1: Coordinates for best cross-validation rate