

# COSC343: Artificial Intelligence

## Lecture 12 : Kernel methods

Lech Szymanski

Dept. of Computer Science, University of Otago

# In today's lecture

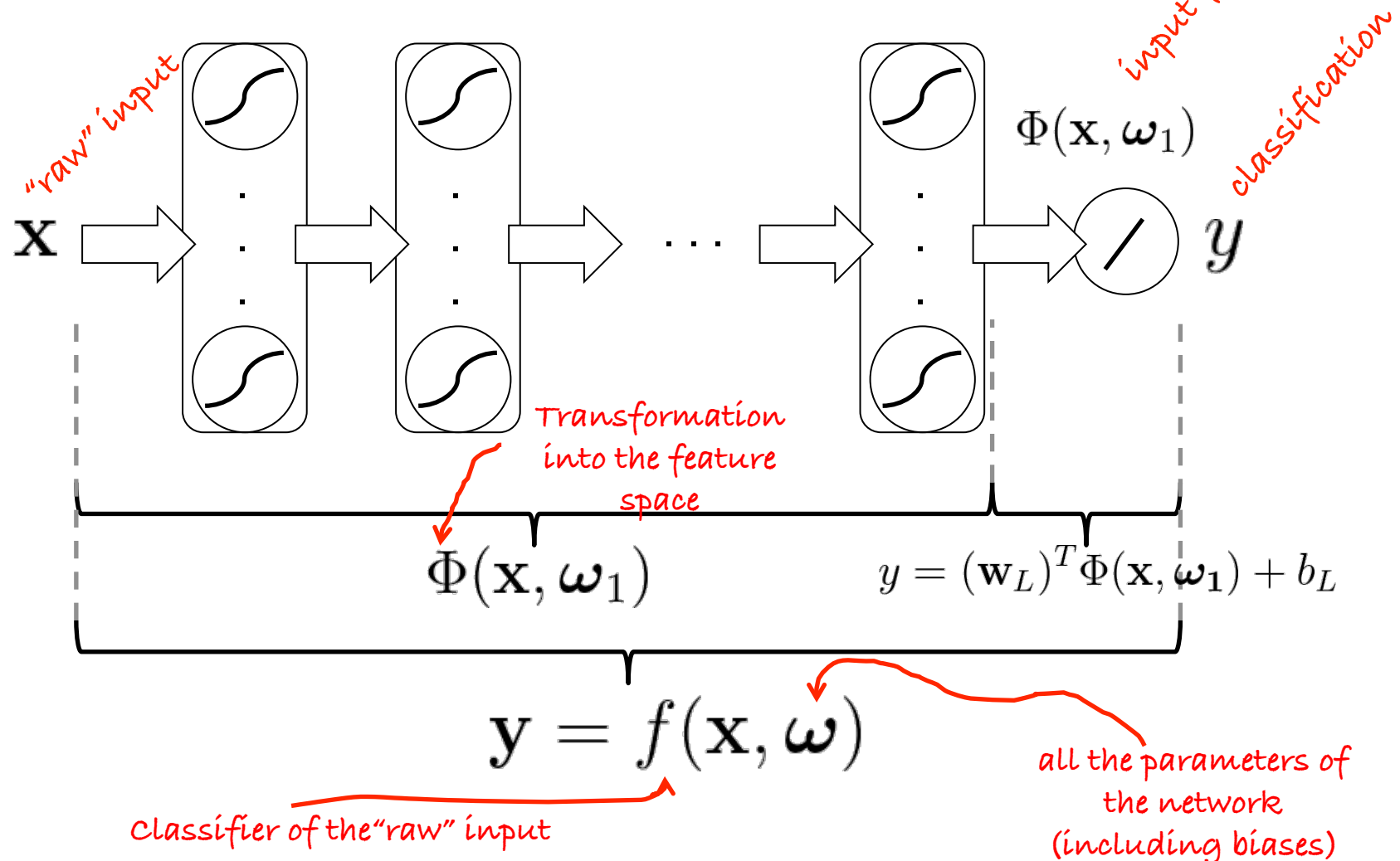
- Feature spaces
- Maximum margin of separation
- Support Vector Machine
- The kernel trick

# Feature space

$$\omega = \{W_1, b_1, \dots, W_{L-1}, b_{L-1}, w_L, b_L\}$$




$$\omega_1 = \{W_1, b_1, \dots, W_{L-1}, b_{L-1}\}$$

Consider a neural network with multiple layers, with output neuron that separates input data into two classes...



# Feature space

A classifier that processes data in a series of transformations can be seen as

- A complex classifier that maps “raw” input  $\mathbf{x}$  to desired output, or...  
  
The entire neural network
- A complex mapping of the “raw” input  $\mathbf{x}$  to a *feature space*  $\Phi(\mathbf{x})$ , followed by a simple linear classifier.  
  
Hidden layers  
  
Output neuron

# Feature space

The mapping to the feature space can be:

changeable parameters  
that adjust the  
mapping

- Learned,  $\Phi(\mathbf{x}, \omega_1)$ , as in multi-layer neural networks, or...
- Static,  $\Phi(\mathbf{x})$  – usually chosen by the user, as in the a linear model with a set of nonlinear base functions...

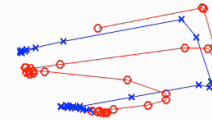
Recall, the linear model:

$$y = \sum_j w_j f_j(\mathbf{x}) = [w_1 \quad \dots \quad w_M] \begin{bmatrix} f_1(\mathbf{x}) \\ \vdots \\ f_M(\mathbf{x}) \end{bmatrix} = \mathbf{w}^T \Phi(\mathbf{x}) , \text{ where } \Phi(\mathbf{x}) = \begin{bmatrix} f_1(\mathbf{x}) \\ \vdots \\ f_M(\mathbf{x}) \end{bmatrix}$$

# An example: the spiral in a feature space

- After a transformation via a 2-8-8 neural network, the spiral pattern becomes linearly separable in the feature space
- Depending on the starting conditions, the actual transformation may vary

Visualisation of a 2-8-8 neural network transformation



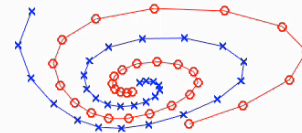
$$\Phi(\mathbf{x}) = f_{\text{logsig}} \left( \mathbf{W}_2 f_{\text{logsig}} (\mathbf{W}_1 \mathbf{x} - \mathbf{b}_1) - \mathbf{b}_2 \right)$$

# An example: the spiral in a feature space

- After a transformation via  $k=3$  polynomial, the spiral pattern is still NOT linearly separable in the feature space

$$\Phi(\mathbf{x}) = \begin{bmatrix} x_1^3 \\ x_2^3 \\ x_1^2 x_2 \\ x_1 x_2^2 \\ x_1^2 \\ x_2^2 \\ x_1 x_2 \\ x_1 \\ x_2 \\ 1 \end{bmatrix}$$

Visualisation of polynomial bases transformation



# Linear separation in feature space

Let's now assume that the model has learned, or been given, the appropriate transformation into the feature space for the purpose of classification. What next?



# Linear separation in feature space

Let's now assume that the model has learned, or been given, the appropriate transformation into the feature space for the purpose of classification. What next?

- We need to find the parameters of the linear model that maps data from the feature space to output that will label the input appropriately

Recall: Perceptron is a linear model that does such a split:

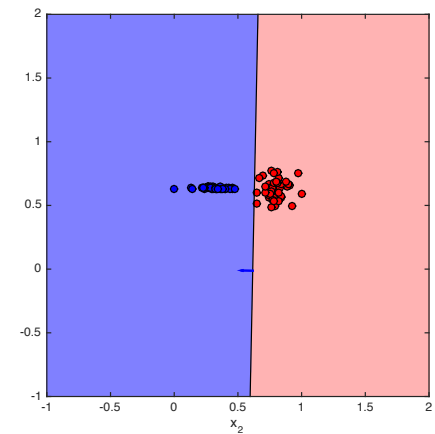
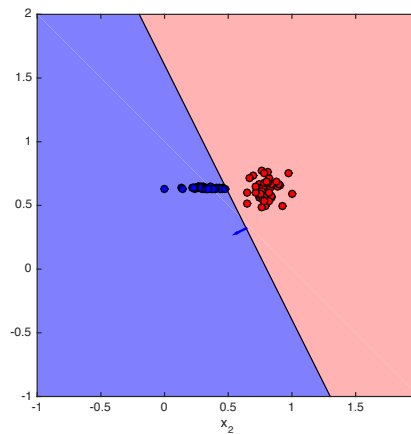
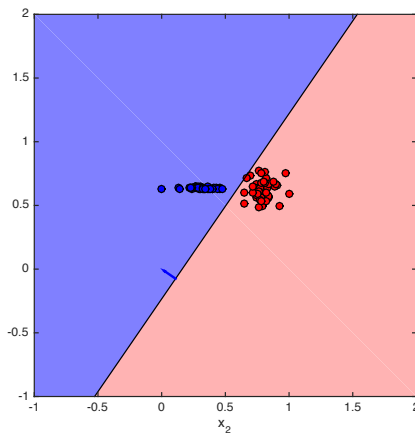
$$y = f_{\text{hardlim}} \left( \sum_i w_i \Phi(\mathbf{x})_i - b \right)$$

this perceptron operates in the feature space, so its input is the feature vector

# Problem with separating hyperplanes

Recall: that a perceptron specifies a separating hyperplane that divides space into two half-spaces.

- For any given linearly separable problem, there are many ways to split the space
- E.g. Below are 3 consistent separations of the same data sample. Each is a 0 training error solution to the problem, yet all are quite different (due to different choices of initial parameters for the perceptron learning rule).



Which one is the best solution?

# Maximum margin separation

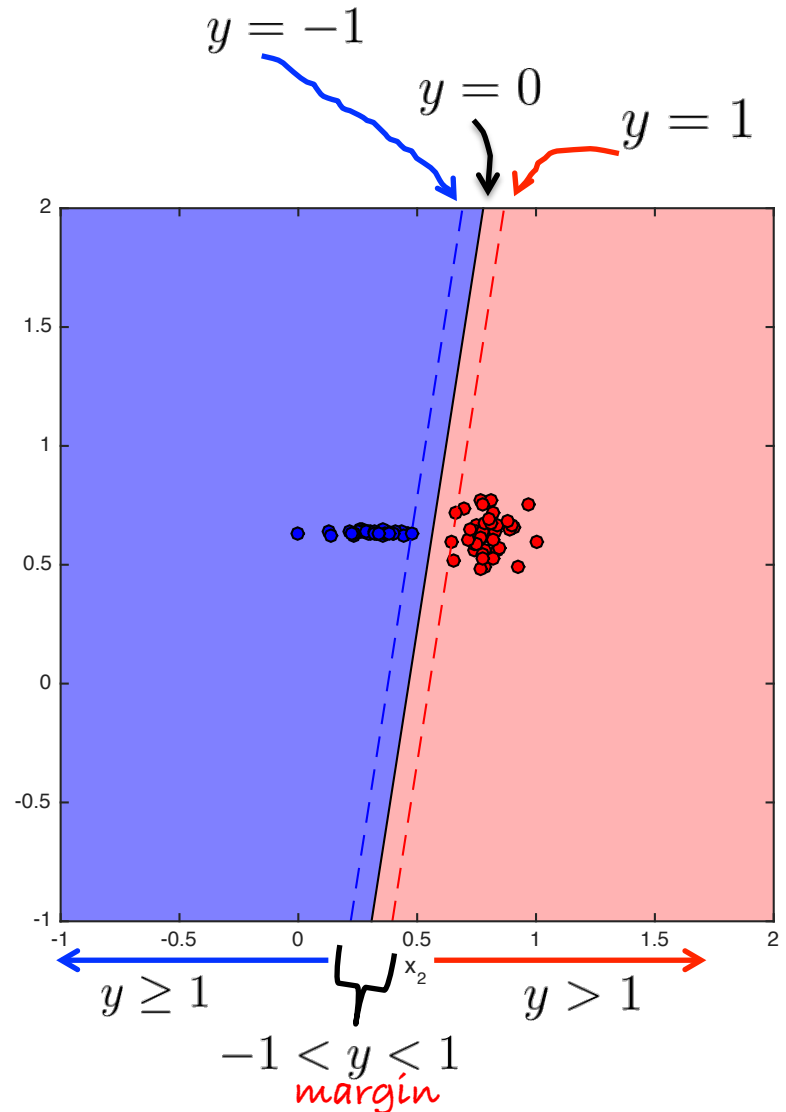
Given a feature space with points corresponding to two classes, and a linear separator (no hardlimiting function)

$$y = \mathbf{w}^T \Phi(\mathbf{x}) - b$$

the best separating hyperplane is the one where

$$J = \frac{1}{2} \mathbf{w}^T \mathbf{w}$$

is minimised while all points of one class produce output  $y \geq 1$ , while all points of other class produce output  $y \leq -1$ .



Geometrically, this corresponds to a separation with **maximum margin hyperplane**.

# Statistical learning theory

- Vapnik has demonstrated (mathematically) that a separating hyperplane with the **maximum margin** of separation is the solution that is most likely (in probability) going to give the best generalisation.
- The statistical learning theory is a mathematical proof that Occam's razor principle (simple solution is the best) is the correct approach to machine learning.
- The maximum margin separating hyperplane corresponds to the *simplest consistent hypothesis*.

# Support Vector Machines

Support Vector Machine is a linear model that maximises the margin of separation while at the same time separating the classes according to their labels (with some allowed errors).

*desired output labels must be either -1 or 1 for SVM equations to work*

Given training set  $\{(\mathbf{x}_i, \tilde{y}_i), \dots, (\mathbf{x}_N, \tilde{y}_N)\}$ , where  $\tilde{y}_i \in \{-1, 1\}$ , and some transformation to feature space  $\Phi(\mathbf{x})$ :

- Finding set of  $N$  parameters  $\alpha$  (one for each sample input) that maximise the following cost

$$J = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j \tilde{y}_i \tilde{y}_j \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j)^T$$

*not learning rate parameters*

under the constraints that  $0 \leq \alpha_i \leq C$ , for  $i = 1, 2, \dots, N$  and  $\sum_{i=1}^N \alpha_i \tilde{y}_i = 0$ .

*constant chosen by the user*

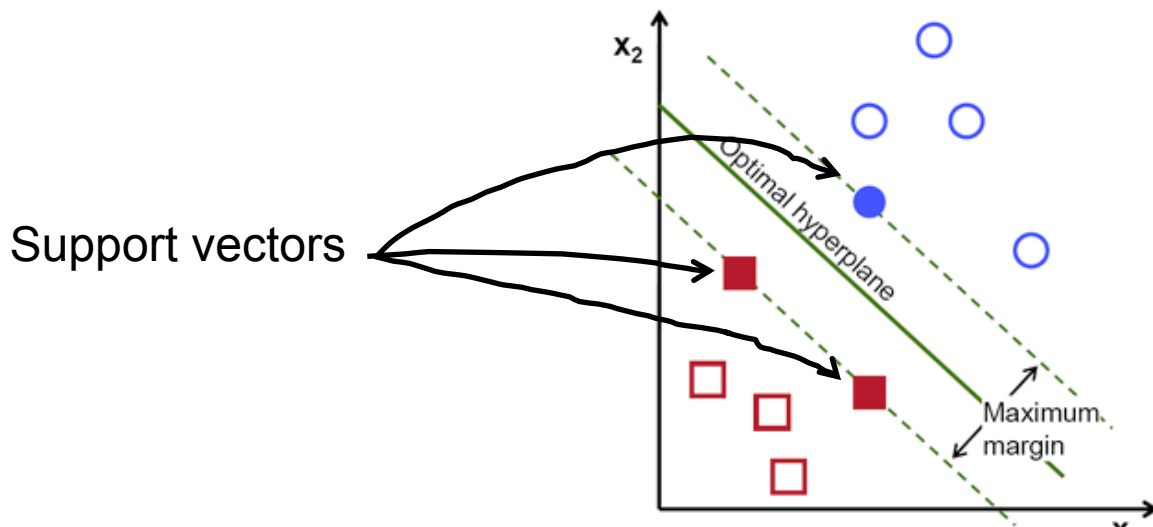
- The output of the machine is

$$y_k = f_{\text{hardlims}}(\mathbf{w}^T \Phi(\mathbf{x}_k) - b) \quad f_{\text{hardlims}}(\mathbf{w}^T \Phi(\mathbf{x}) - b) = \begin{cases} 1 & \mathbf{w}^T \Phi(\mathbf{x}) - b \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

where  $\mathbf{w}^T = \sum_{i=1}^N \alpha_i \tilde{y}_i \Phi(\mathbf{x}_i)^T$  and  $b$  is also a function of  $\alpha$ 's and target output.

# Support Vector Machines

The geometric significance of the  $\alpha$ 's found in training (using complicated optimisation algorithms that take constraints under consideration) is that non-zero  $\alpha_i$  correspond to point  $\Phi(\mathbf{x}_i)$  that lies on the margin – these points are referred to as the **support vectors**.



[http://docs.opencv.org/2.4/doc/tutorials/ml/introduction\\_to\\_svm/introduction\\_to\\_svm.html](http://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html)

# The Kernel Trick

If in the equation for SVM's output,  $y_k = f_{\text{hardlims}}(\mathbf{w}^T \Phi(\mathbf{x}_k) - b)$ , we substitute  $\mathbf{w}^T = \sum_{i=1}^N \alpha_i \tilde{y}_i \Phi(\mathbf{x}_i)^T$ , we get:

$$y_k = f_{\text{hardlims}} \left( \sum_{i=1}^N \alpha_i \tilde{y}_i \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_k) - b \right)$$

Notice that in both the SVM output, as well as the optimisation

$$J = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j \tilde{y}_i \tilde{y}_j \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j)^T$$

the feature vector never appears alone – it's always evaluated as the dot product against another feature vector:

$$\Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j)$$

# The Kernel Trick

Hence, SVM will still work on an *abstract* feature space where  $\Phi(\mathbf{x}_i)$  is not defined but the dot product  $\Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j)$  is.

- In these abstract spaces the relationship between two feature vectors can be quantified even though the concept of the individual feature vector has no meaning.
- The expression that computes the dot product of two feature vectors is called the **kernel function**.
- One such very powerful kernel is the Radial Basis Function (RBF)

$$\Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j) = e^{-\gamma(\mathbf{x}_i - \mathbf{x}_j)^T (\mathbf{x}_i - \mathbf{x}_j)}$$

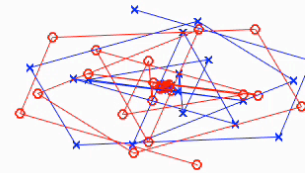
parameter chosen by  
the user



# An example: the spiral in a feature space

- Radial basis function allows computation of the dot product in feature space between two vectors, from which we can infer the relationship (distance) between the points after transformation to this space.

Visualisation of RBF kernel transformation

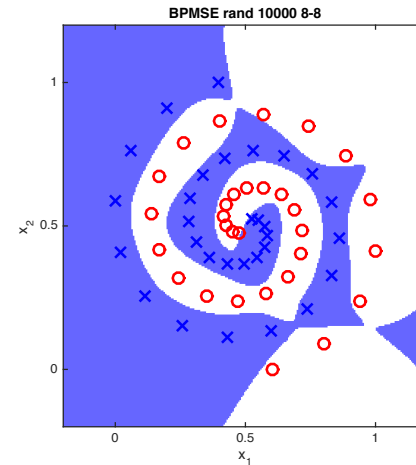
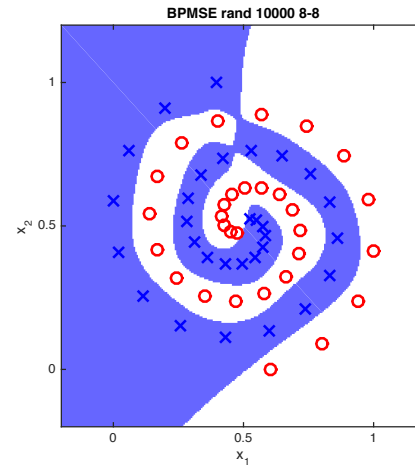
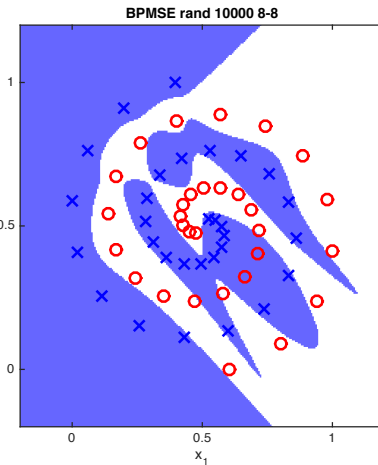


$$\gamma = 20$$

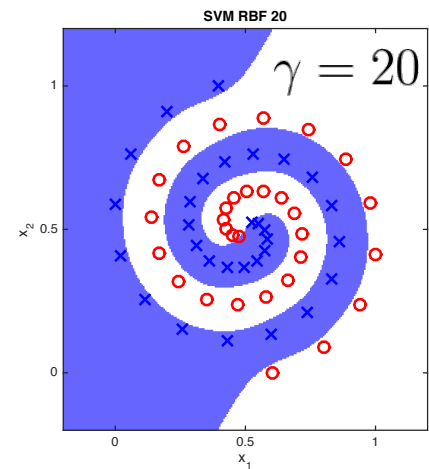
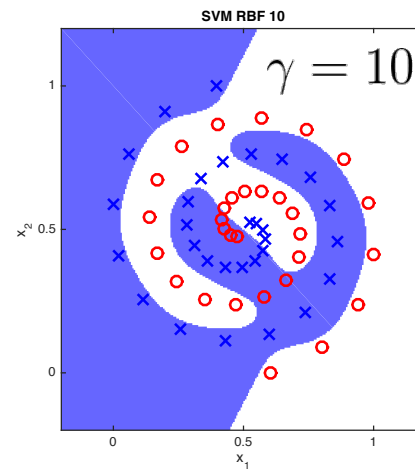
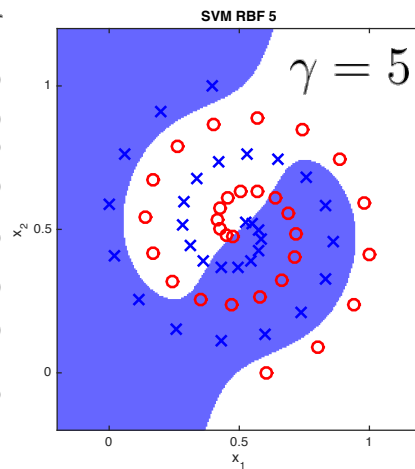
$$\Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j) = e^{-\gamma(\mathbf{x}_i - \mathbf{x}_j)^T(\mathbf{x}_i - \mathbf{x}_j)}$$

# An example: spiral classification

Neural networks  
with different  
starting  
conditions



SVMs with RBF  
kernel and  
different value of  $\gamma$

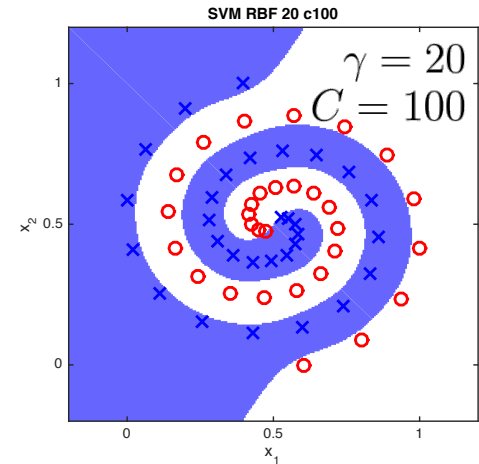
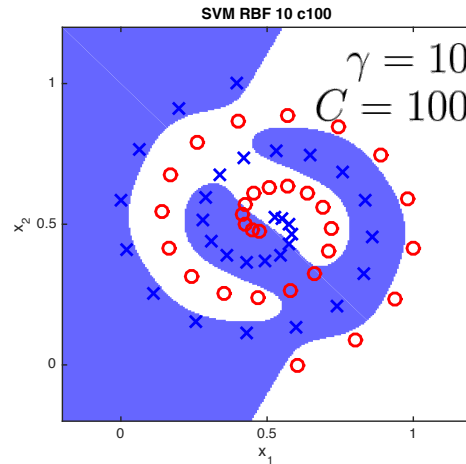
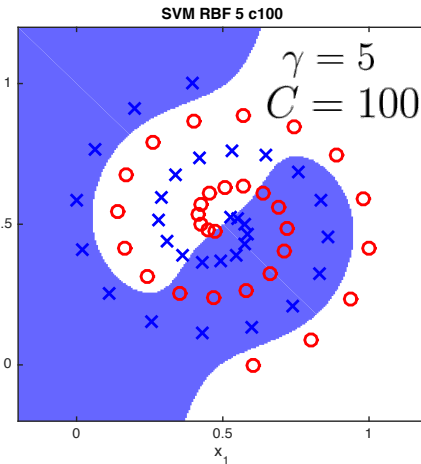


# An example: spiral classification

- The higher the value of  $C$ , the more insistence on consistency (making fewer errors during training)

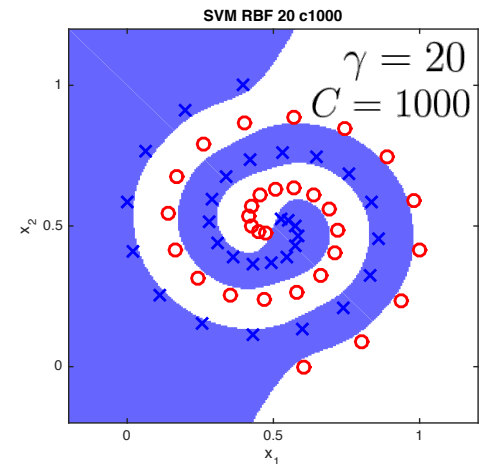
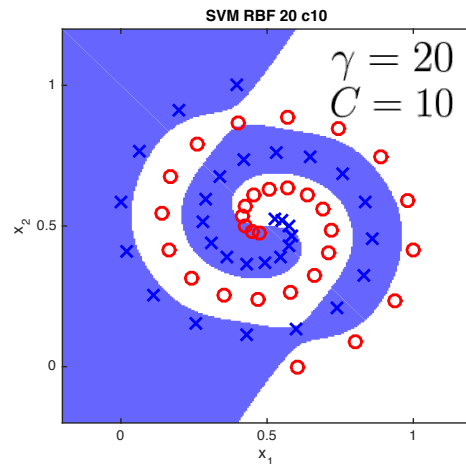
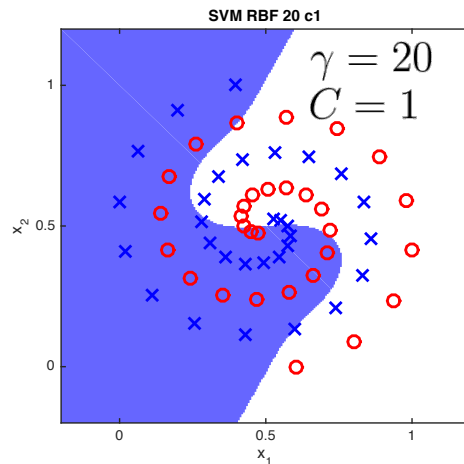
SVMs with RBF

kernel and  
different value of  $\gamma$



SVMs with RBF

kernel and different  
value of  $C$



# Summary and reading

- Feature space – transformation that makes a difficult problem an easy problem to solve
- Maximum margin separating hyperplane – best separation of linearly separable data
- Support Vector Machines – produce maximum margin separation
  - Must choose the kernel function (and its parameters) in order to get appropriate transformation into the feature space
  - RBF kernel is a kernel that often works well (if you pick the right gamma value)

Reading for the lecture: AIMA Chapter 18.9

Reading for next lecture: no reading