Lecture 3: SVM dual, kernels and regression

C19 Machine Learning | Hilary 2015 | A. Zisserman

- Primal and dual forms
- Linear separability revisted
- Feature maps
- Kernels for SVMs
- Regression
    - Ridge regression
    - Basis functions

SVM – review

- We have seen that for an SVM learning a linear classifier

$$f(x) = \mathbf{w}^\top \mathbf{x} + b$$

is formulated as solving an optimization problem over $\mathbf{w}$:

$$\min_{\mathbf{w} \in \mathbb{R}^d} ||\mathbf{w}||^2 + C \sum_i^N \max\left(0, 1 - y_i f(\mathbf{x}_i)\right)$$

- This quadratic optimization problem is known as the primal problem.

- Instead, the SVM can be formulated to learn a linear classifier

$$f(\mathbf{x}) = \sum_i^N \alpha_i y_i (\mathbf{x}_i{}^\top \mathbf{x}) + b$$

by solving an optimization problem over α_i.

- This is know as the dual problem, and we will look at the advantages of this formulation.

Sketch derivation of dual form

The Representer Theorem states that the solution $\mathbf{w}$ can always be written as a linear combination of the training data:

$$\mathbf{w} = \sum_{j=1}^{N} \alpha_j y_j \mathbf{x}_j$$

Proof: see example sheet .

Now, substitute for $\mathbf{w}$ in $f(x) = \mathbf{w}^\top \mathbf{x} + b$

$$f(x) = \left(\sum_{j=1}^{N} \alpha_j y_j \mathbf{x}_j \right)^\top \mathbf{x} + b = \sum_{j=1}^{N} \alpha_j y_j \left(\mathbf{x}_j{}^\top \mathbf{x} \right) + b$$

and for $\mathbf{w}$ in the cost function $\min_{\mathbf{w}} ||\mathbf{w}||^2$ subject to $y_i \left(\mathbf{w}^\top \mathbf{x}_i + b \right) \geq 1, \forall i$

$$||\mathbf{w}||^2 = \left\{ \sum_j \alpha_j y_j \mathbf{x}_j \right\}^\top \left\{ \sum_k \alpha_k y_k \mathbf{x}_k \right\} = \sum_{jk} \alpha_j \alpha_k y_j y_k (\mathbf{x}_j{}^\top \mathbf{x}_k)$$

Hence, an equivalent optimization problem is over α_j

$$\min_{\alpha_j} \sum_{jk} \alpha_j \alpha_k y_j y_k (\mathbf{x}_j{}^\top \mathbf{x}_k) \quad \text{subject to } y_i \left(\sum_{j=1}^{N} \alpha_j y_j (\mathbf{x}_j{}^\top \mathbf{x}_i) + b \right) \geq 1, \forall i$$

and a few more steps are required to complete the derivation.

Primal and dual formulations

N is number of training points, and d is dimension of feature vector $\mathbf{x}$.

Primal problem: for $\mathbf{w} \in \mathbb{R}^d$

$$\min_{\mathbf{w}\in\mathbb{R}^d} ||\mathbf{w}||^2 + C\sum_i^N \max\left(0, 1 - y_i f(\mathbf{x}_i)\right)$$

Dual problem: for $\boldsymbol{\alpha} \in \mathbb{R}^N$ (stated without proof):

$$\max_{\alpha_i \geq 0} \sum_i \alpha_i - \frac{1}{2}\sum_{jk} \alpha_j \alpha_k y_j y_k (\mathbf{x}_j{}^\top \mathbf{x}_k) \text{ subject to } 0 \leq \alpha_i \leq C \text{ for } \forall i, \text{ and } \sum_i \alpha_i y_i = 0$$

- Need to learn d parameters for primal, and N for dual
- If $N << d$ then more efficient to solve for α than $\mathbf{w}$
- Dual form only involves $(\mathbf{x}_j{}^\top \mathbf{x}_k)$. We will return to why this is an advantage when we look at kernels.

Primal and dual formulations

Primal version of classifier:

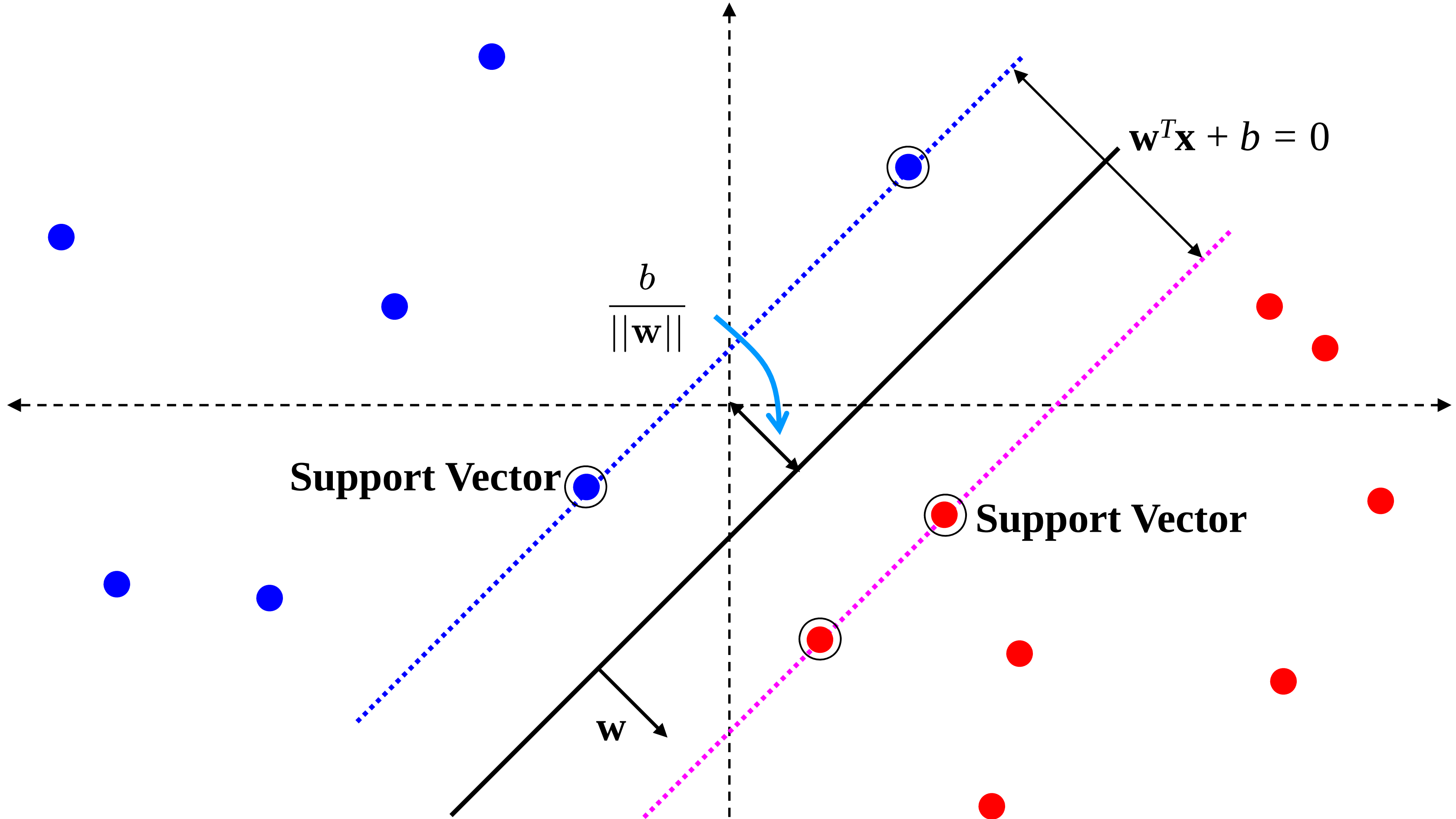
$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$$

Dual version of classifier:

$$f(\mathbf{x}) = \sum_i^N \alpha_i y_i (\mathbf{x}_i{}^\top \mathbf{x}) + b$$

At first sight the dual form appears to have the disadvantage of a K-NN classifier – it requires the training data points $\mathbf{x}_i$. However, many of the α_i's are zero. The ones that are non-zero define the support vectors $\mathbf{x}_i$.

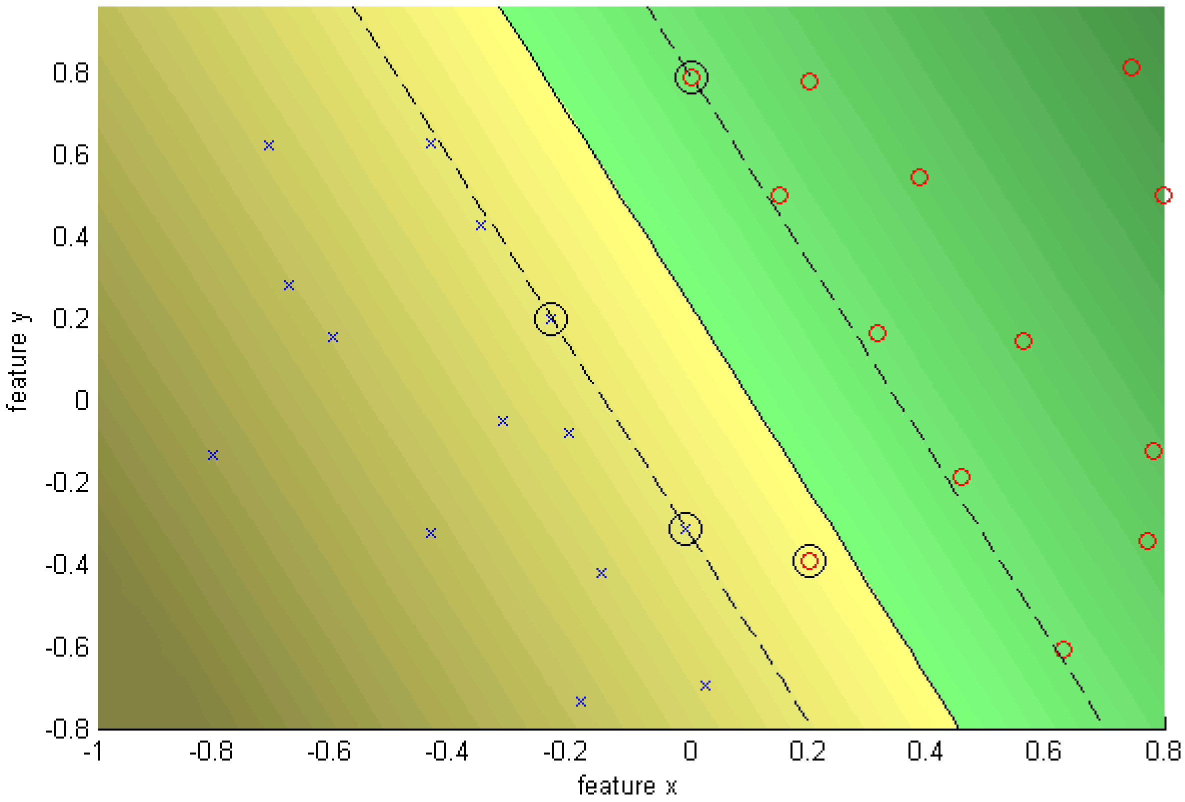
Support Vector Machine



$$f(\mathbf{x}) = \sum_i \alpha_i y_i (\mathbf{x}_i{}^\top \mathbf{x}) + b$$

support vectors

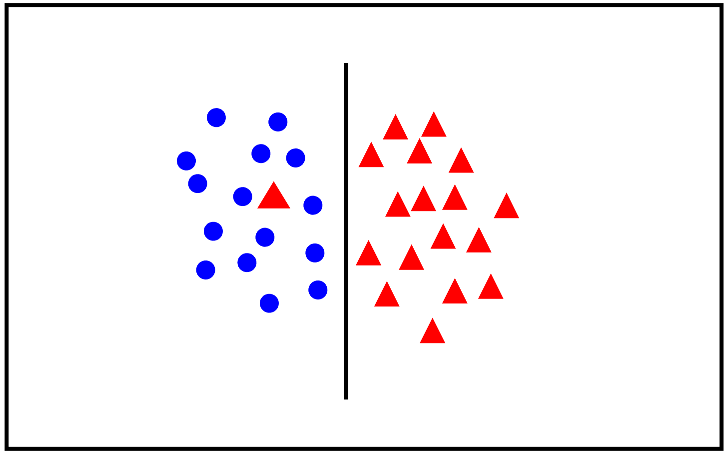
C = 10   soft margin



Comment Window

SVM (L1) by Sequential Minimal Optimizer
Kernel: linear (-), C: 10.0000
Kernel evaluations: 2645
Number of Support Vectors: 4
Margin: 0.2265
Training error: 3.70%

Handling data that is not linearly separable

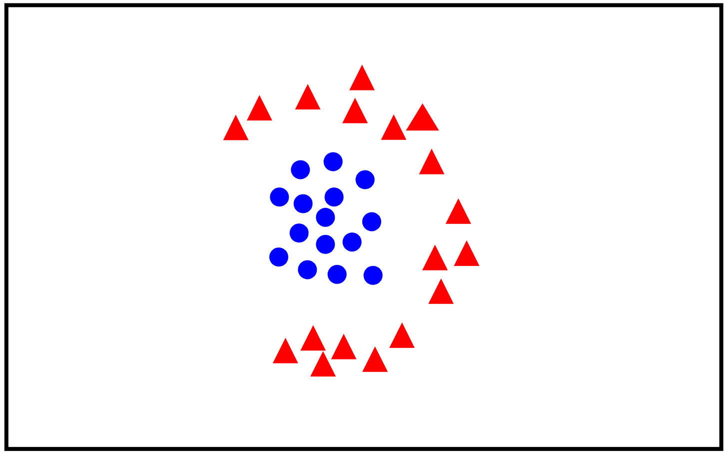


- introduce slack variables

$$\min_{\mathbf{w}\in\mathbb{R}^d, \xi_i \in \mathbb{R}^+} ||\mathbf{w}||^2 + C \sum_i^N \xi_i$$

subject to

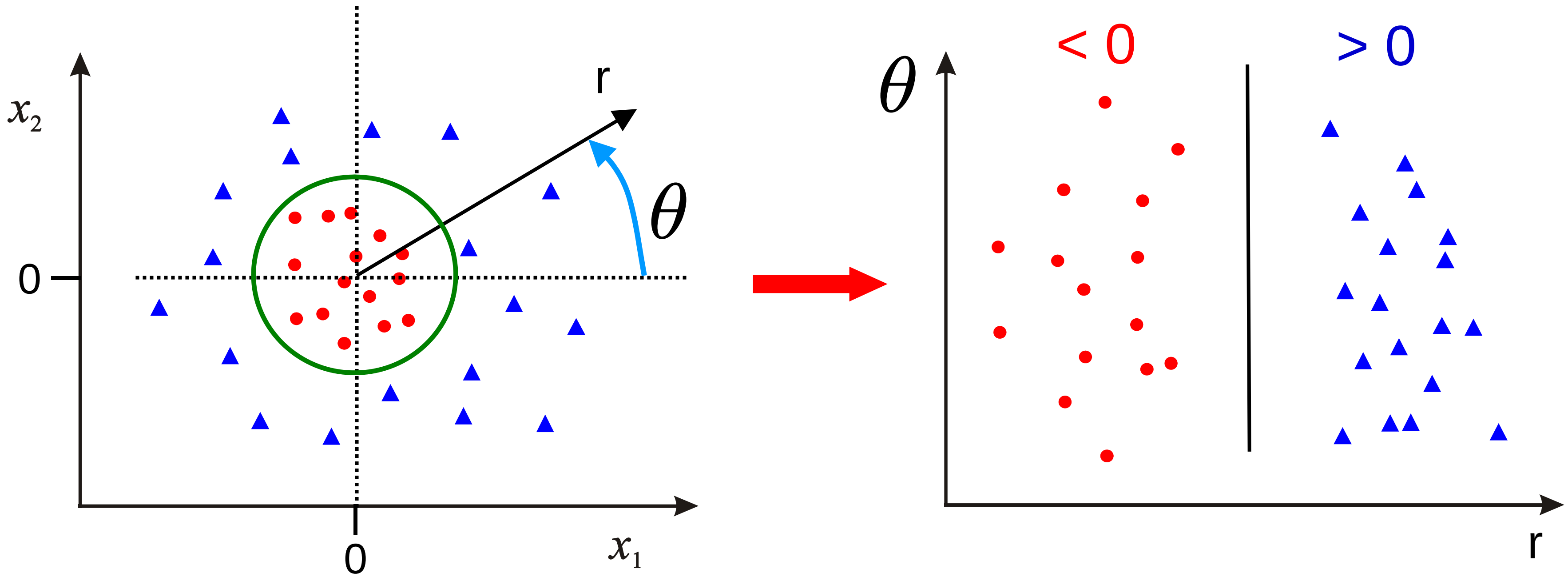
$$y_i \left(\mathbf{w}^\top \mathbf{x}_i + b\right) \geq 1 - \xi_i \text{ for } i = 1 \ldots N$$



- linear classifier not appropriate

??

Solution 1: use polar coordinates

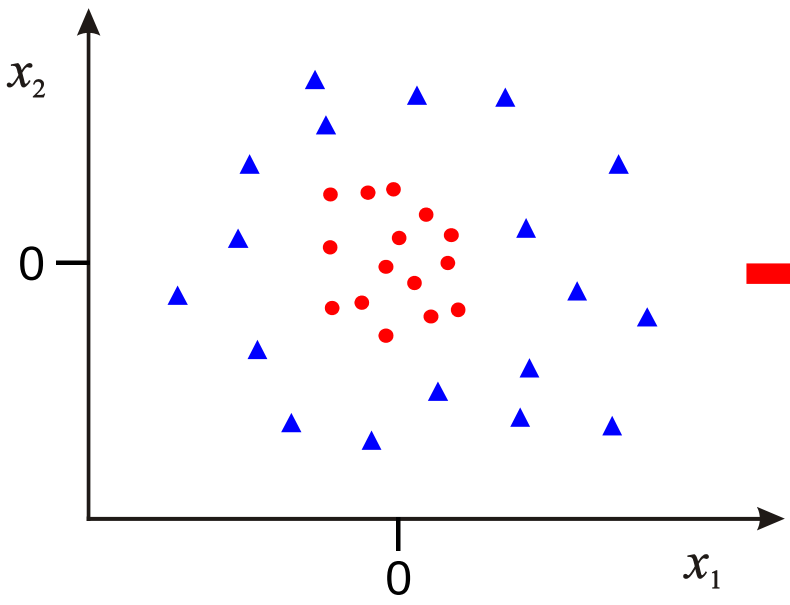


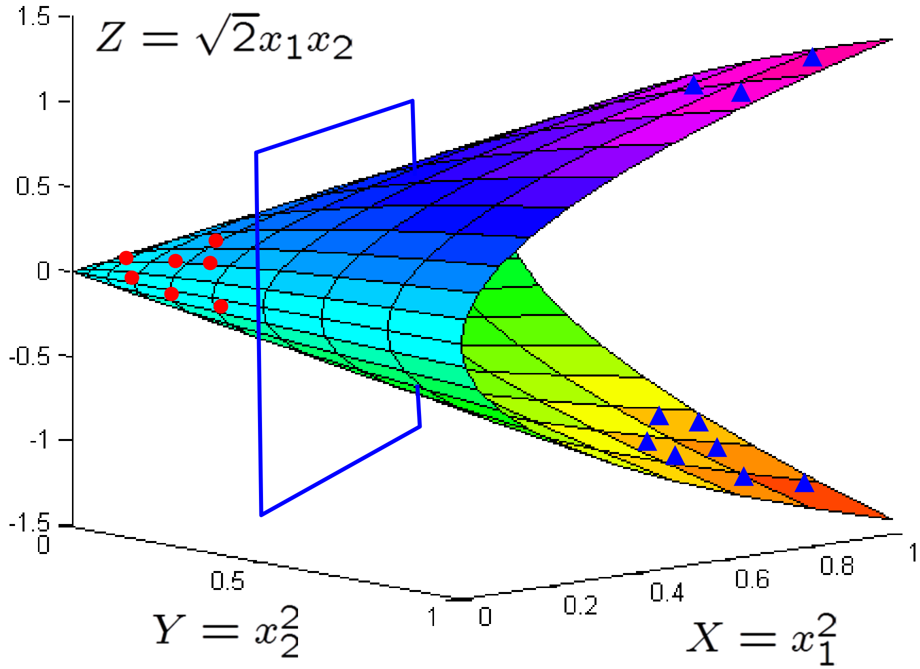
- Data is linearly separable in polar coordinates
- Acts non-linearly in original space

$$\Phi : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} r \\ \theta \end{pmatrix} \qquad \mathbb{R}^2 \rightarrow \mathbb{R}^2$$

Solution 2: map data to higher dimension

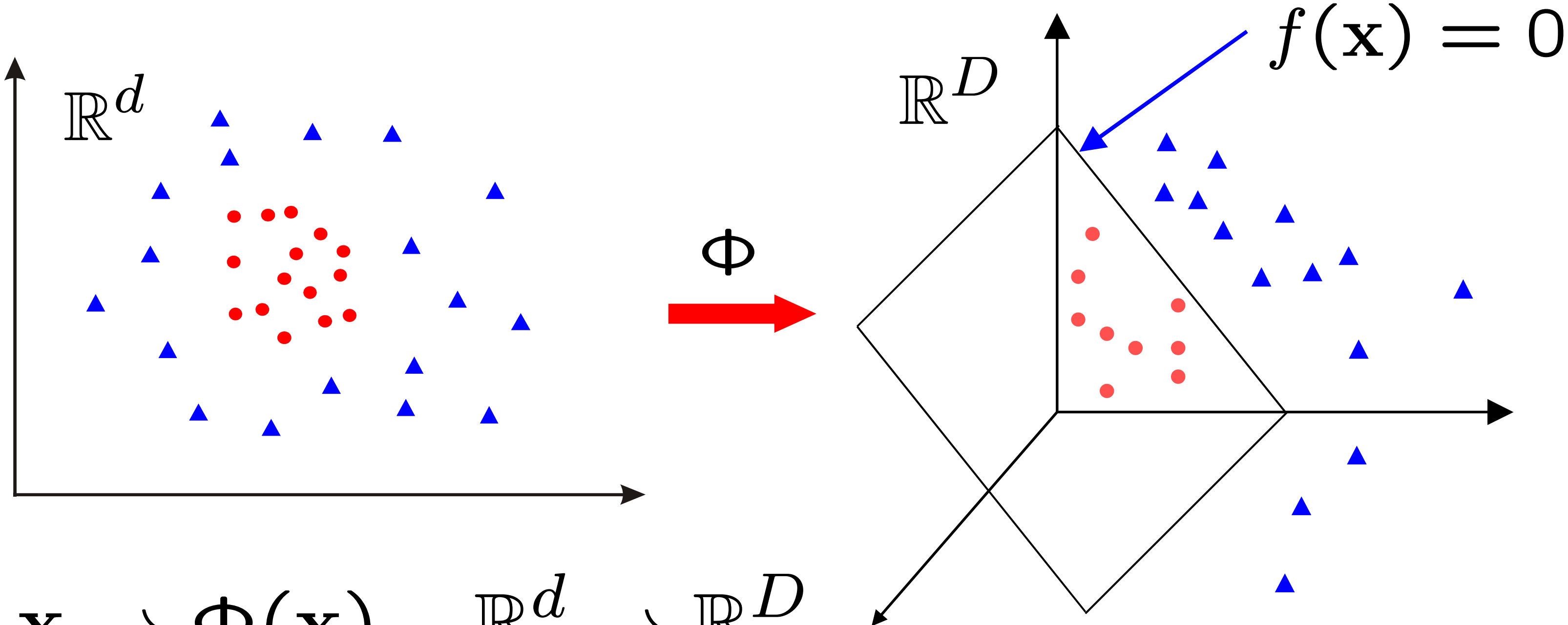
$$\Phi : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{pmatrix} \qquad \mathbb{R}^2 \rightarrow \mathbb{R}^3$$





- Data is linearly separable in 3D
- This means that the problem can still be solved by a linear classifier

SVM classifiers in a transformed feature space



$\Phi : \mathbf{x} \to \Phi(\mathbf{x}) \quad \mathbb{R}^d \to \mathbb{R}^D$

Learn classifier linear in $\mathbf{w}$ for $\mathbb{R}^D$:

$$f(\mathbf{x}) = \mathbf{w}^\top \Phi(\mathbf{x}) + b$$

$\Phi(\mathbf{x})$ is a feature map

Primal Classifier in transformed feature space

Classifier, with $\mathbf{w} \in \mathbb{R}^D$:

$$f(\mathbf{x}) = \mathbf{w}^\top \Phi(\mathbf{x}) + b$$

Learning, for $\mathbf{w} \in \mathbb{R}^D$

$$\min_{\mathbf{w} \in \mathbb{R}^D} ||\mathbf{w}||^2 + C \sum_i^N \max\left(0, 1 - y_i f(\mathbf{x}_i)\right)$$

- Simply map $\mathbf{x}$ to $\Phi(\mathbf{x})$ where data is separable
- Solve for $\mathbf{w}$ in high dimensional space $\mathbb{R}^D$
- If $D >> d$ then there are many more parameters to learn for $\mathbf{w}$. Can this be avoided?

Dual Classifier in transformed feature space

Classifier:

$$
\begin{aligned}
f(\mathbf{x}) &= \sum_i^N \alpha_i y_i \, \mathbf{x}_i{}^\top \mathbf{x} + b \\
\rightarrow f(\mathbf{x}) &= \sum_i^N \alpha_i y_i \, \mathbf{\Phi}(\mathbf{x}_i)^\top \mathbf{\Phi}(\mathbf{x}) + b
\end{aligned}
$$

Learning:

$$
\begin{aligned}
& \max_{\alpha_i \geq 0} \sum_i \alpha_i - \frac{1}{2} \sum_{jk} \alpha_j \alpha_k y_j y_k \mathbf{x}_j{}^\top \mathbf{x}_k \\
\rightarrow \quad & \max_{\alpha_i \geq 0} \sum_i \alpha_i - \frac{1}{2} \sum_{jk} \alpha_j \alpha_k y_j y_k \mathbf{\Phi}(\mathbf{x}_j)^\top \mathbf{\Phi}(\mathbf{x}_k)
\end{aligned}
$$

subject to

$$
0 \leq \alpha_i \leq C \text{ for } \forall i, \text{ and} \sum_i \alpha_i y_i = 0
$$

Dual Classifier in transformed feature space

- Note, that $\Phi(\mathbf{x})$ only occurs in pairs $\Phi(\mathbf{x}_j)^\top \Phi(\mathbf{x}_i)$
- Once the scalar products are computed, only the N dimensional vector $\boldsymbol{\alpha}$ needs to be learnt; it is not necessary to learn in the D dimensional space, as it is for the primal
- Write $k(\mathbf{x}_j, \mathbf{x}_i) = \Phi(\mathbf{x}_j)^\top \Phi(\mathbf{x}_i)$. This is known as a Kernel

Classifier:

$$f(\mathbf{x}) = \sum_i^N \alpha_i y_i \, k(\mathbf{x}_i, \mathbf{x}) + b$$

Learning:

$$\max_{\alpha_i \geq 0} \sum_i \alpha_i - \frac{1}{2} \sum_{jk} \alpha_j \alpha_k y_j y_k \, k(\mathbf{x}_j, \mathbf{x}_k)$$

subject to

$$0 \leq \alpha_i \leq C \text{ for } \forall i, \text{ and} \sum_i \alpha_i y_i = 0$$

Special transformations

$$\Phi : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{pmatrix} \qquad \mathbb{R}^2 \rightarrow \mathbb{R}^3$$

$$\begin{aligned} \Phi(\mathbf{x})^\top\Phi(\mathbf{z}) &= \left(x_1^2, x_2^2, \sqrt{2}x_1x_2\right) \begin{pmatrix} z_1^2 \\ z_2^2 \\ \sqrt{2}z_1z_2 \end{pmatrix} \\ &= x_1^2z_1^2 + x_2^2z_2^2 + 2x_1x_2z_1z_2 \\ &= (x_1z_1 + x_2z_2)^2 \\ &= (\mathbf{x}^\top\mathbf{z})^2 \end{aligned}$$

Kernel Trick

- Classifier can be learnt and applied without explicitly computing $\Phi(\mathbf{x})$
- All that is required is the kernel $k(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^\top\mathbf{z})^2$
- Complexity of learning depends on N (typically it is $O(N^3)$) not on D

Example kernels

- Linear kernels $k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{x}'$
- Polynomial kernels $k(\mathbf{x}, \mathbf{x}') = \left(1 + \mathbf{x}^\top \mathbf{x}'\right)^d$ for any $d > 0$
  - Contains all polynomials terms up to degree d
- Gaussian kernels $k(\mathbf{x}, \mathbf{x}') = \exp\left(-||\mathbf{x} - \mathbf{x}'||^2/2\sigma^2\right)$ for $\sigma > 0$
  - Infinite dimensional feature space

SVM classifier with Gaussian kernel

N = size of training data

$$f(\mathbf{x}) = \sum_{i}^{N} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) + b$$

weight (may be zero)

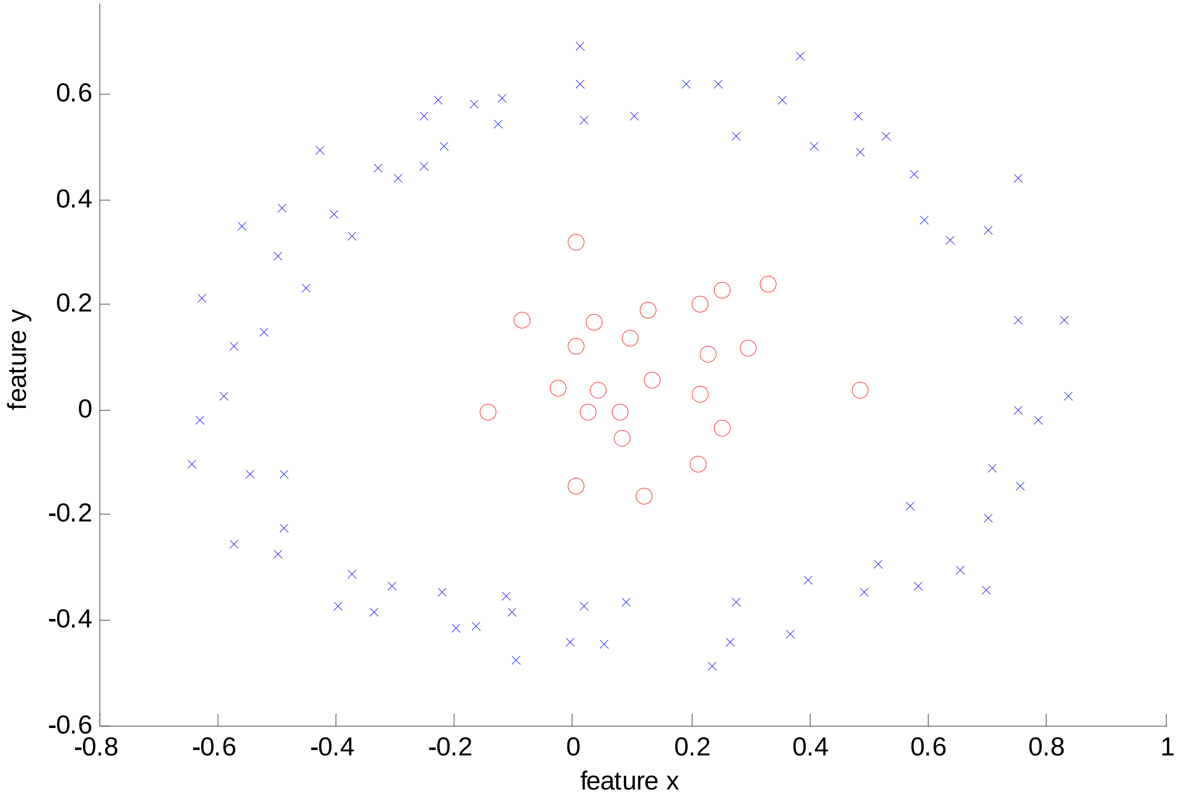
support vector

Gaussian kernel $k(\mathbf{x}, \mathbf{x}') = \exp\left(-||\mathbf{x} - \mathbf{x}'||^2 / 2\sigma^2\right)$

Radial Basis Function (RBF) SVM

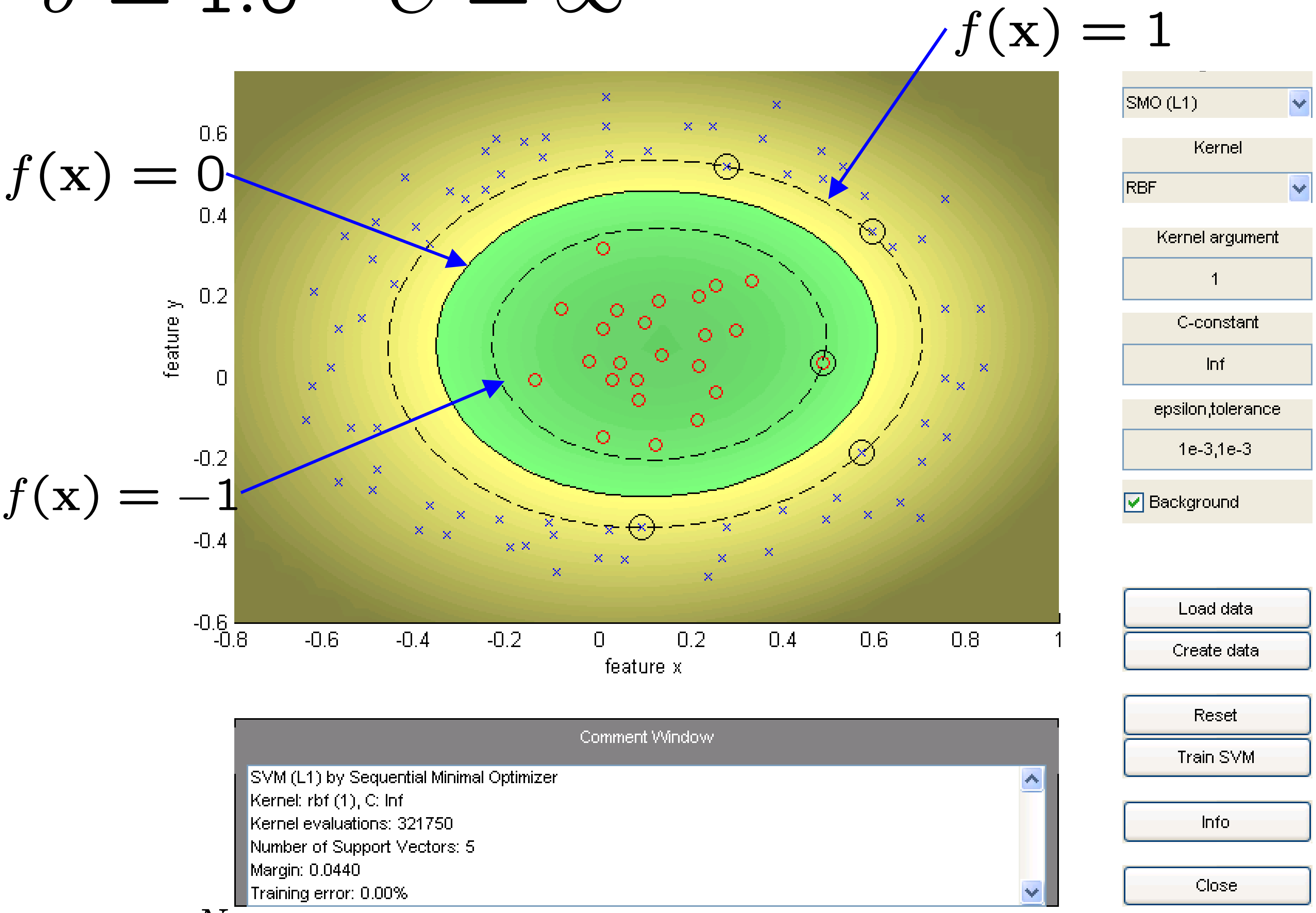
$$f(\mathbf{x}) = \sum_{i}^{N} \alpha_i y_i \exp\left(-||\mathbf{x} - \mathbf{x}_i||^2 / 2\sigma^2\right) + b$$

RBF Kernel SVM Example



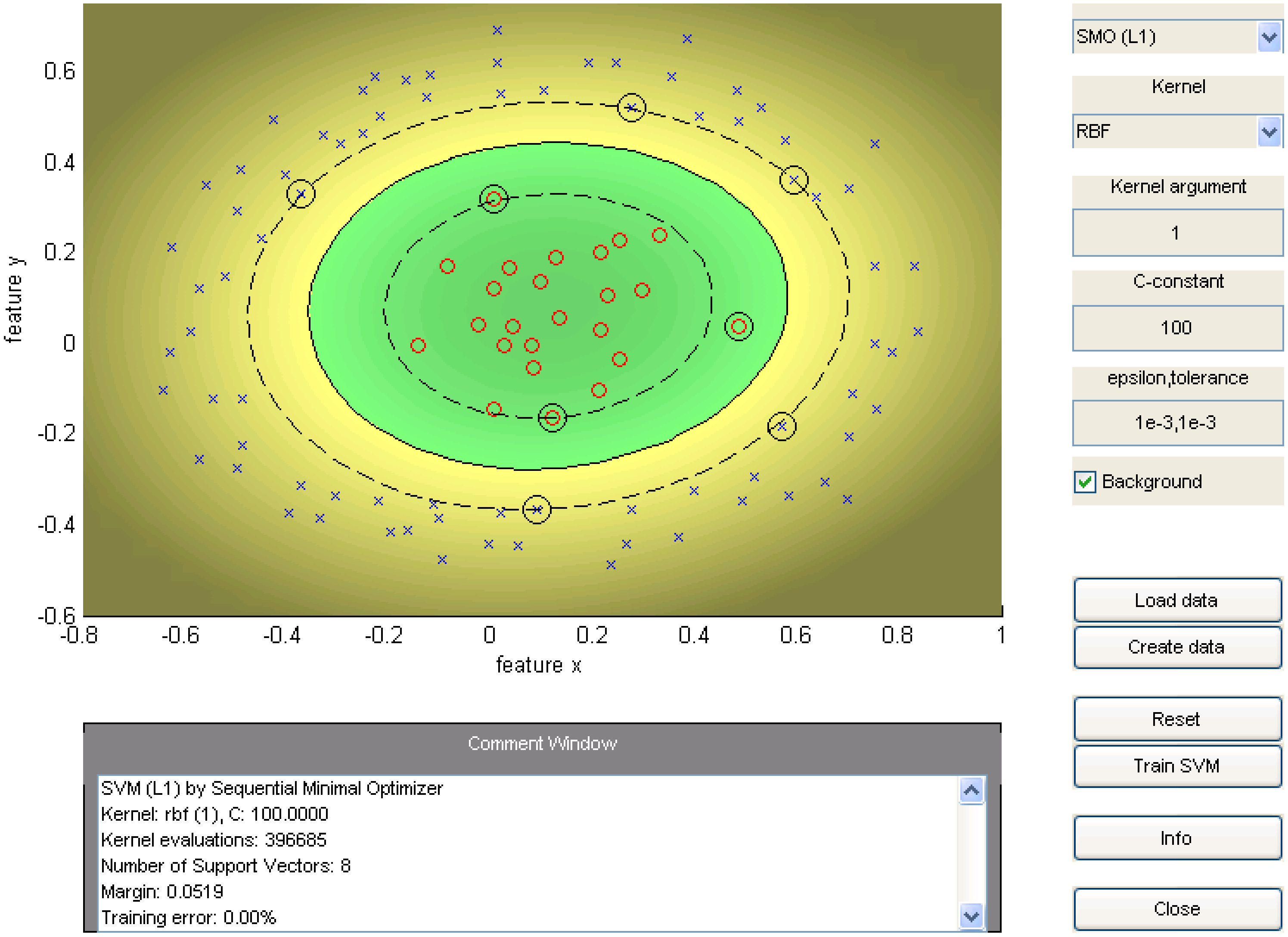
- data is not linearly separable in original feature space

$$\sigma = 1.0 \quad C = \infty$$



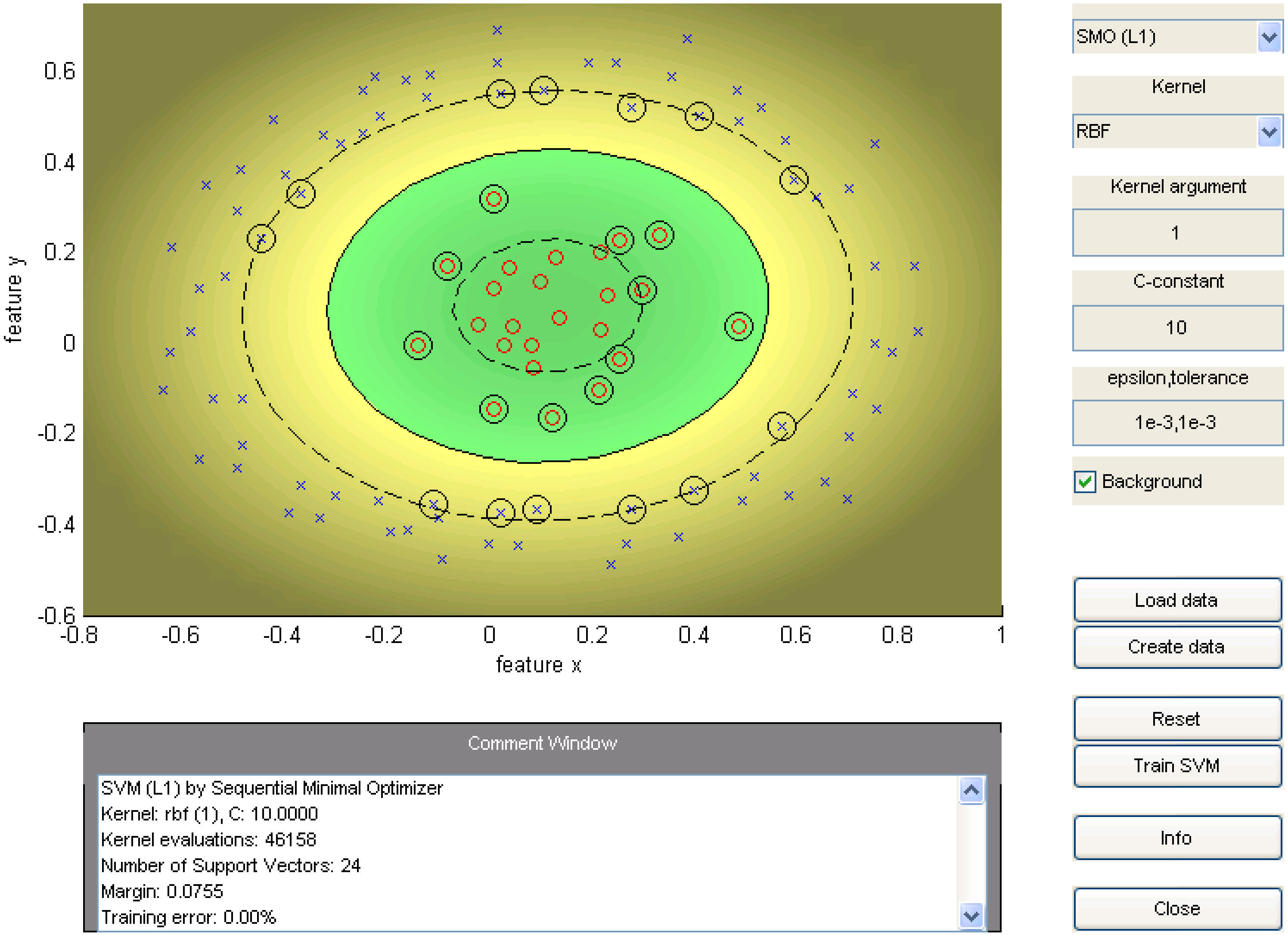
$$f(\mathbf{x}) = \sum_{i}^{N} \alpha_i y_i \exp\left(-||\mathbf{x} - \mathbf{x}_i||^2 / 2\sigma^2\right) + b$$

$$\sigma = 1.0 \quad C = 100$$



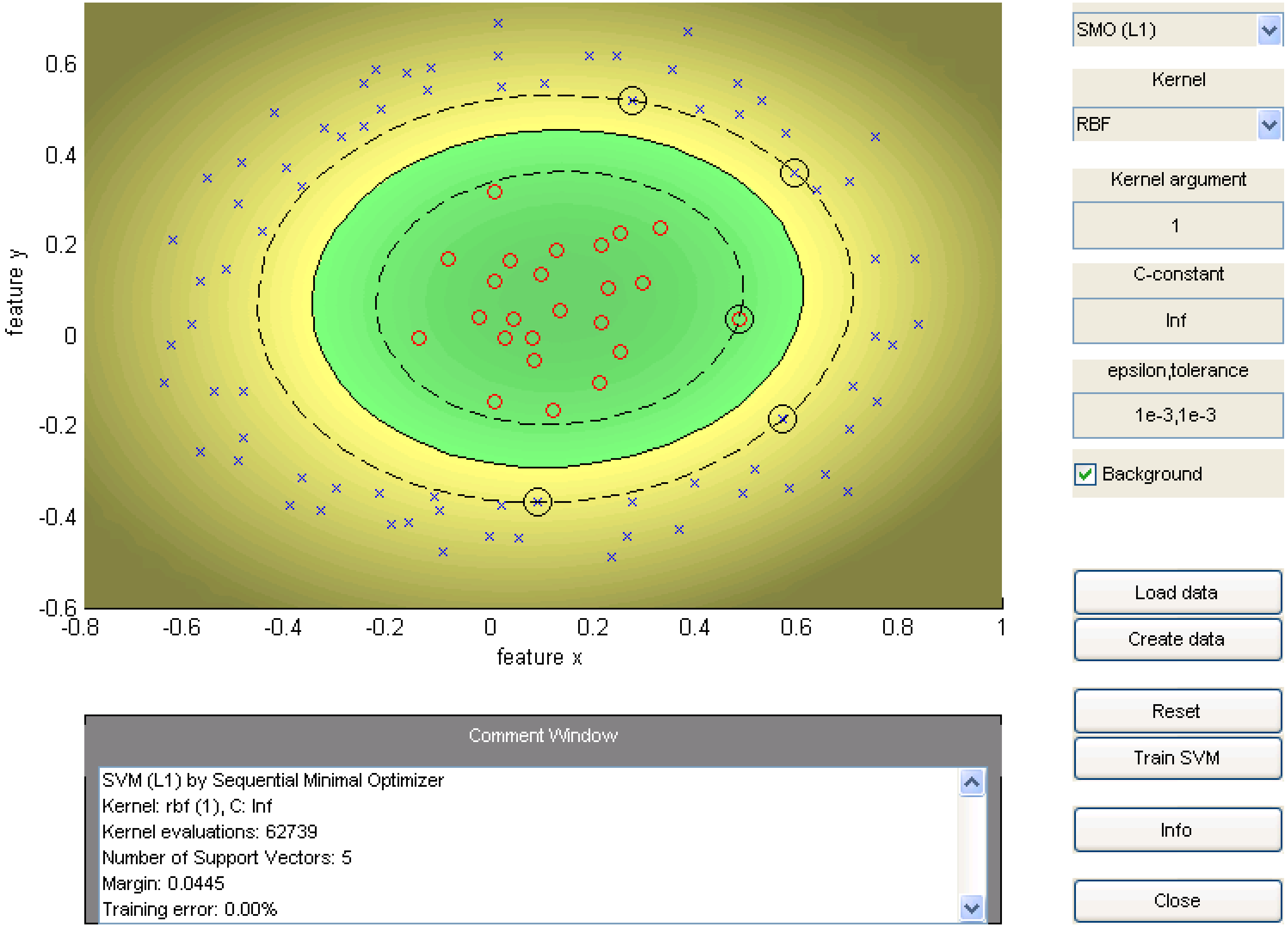
Decrease C, gives wider (soft) margin

$\sigma = 1.0 \quad C = 10$



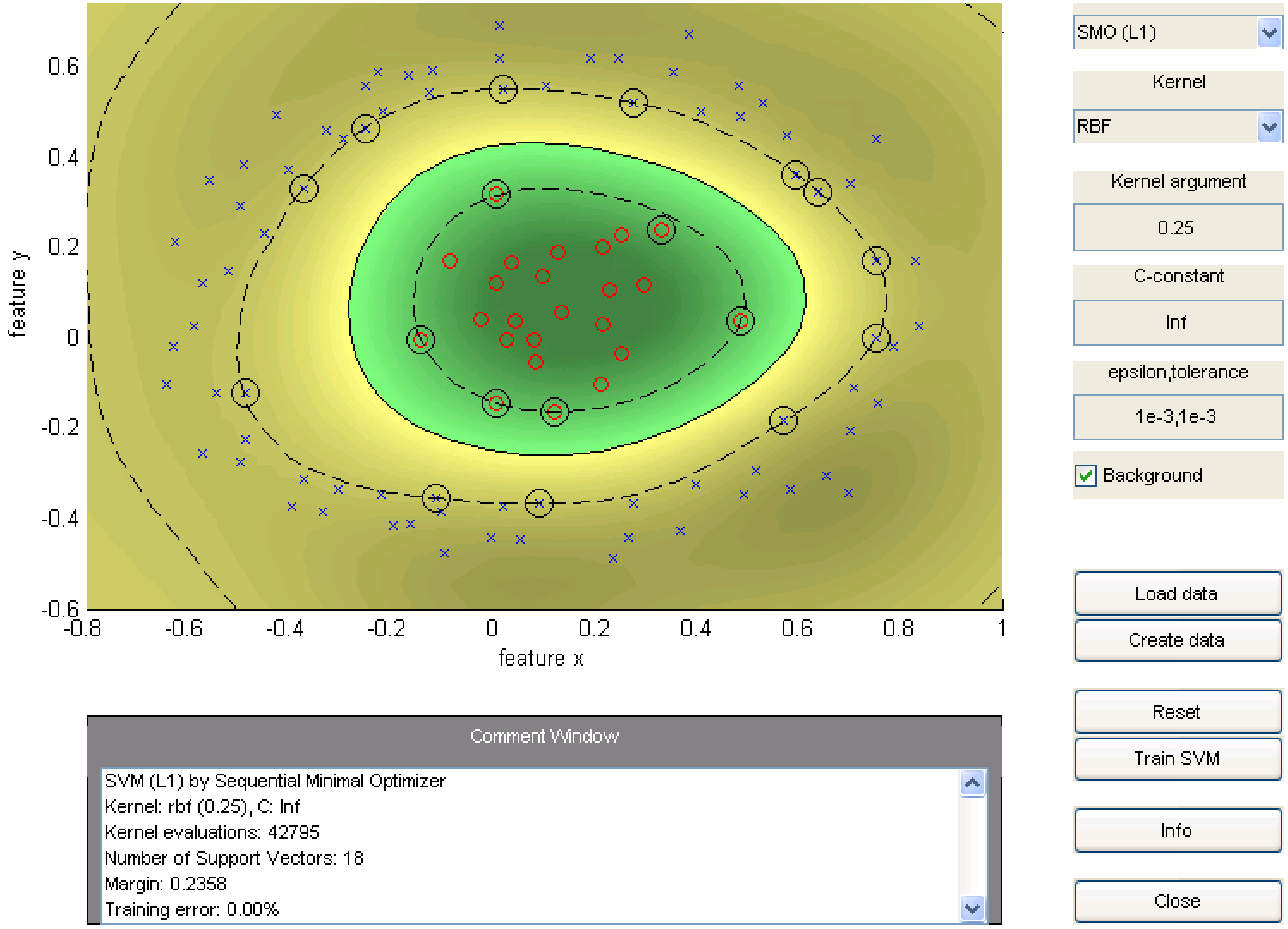
$$f(\mathbf{x}) = \sum_{i}^{N} \alpha_i y_i \exp\left(-||\mathbf{x} - \mathbf{x}_i||^2 / 2\sigma^2\right) + b$$

$$\sigma = 1.0 \quad C = \infty$$



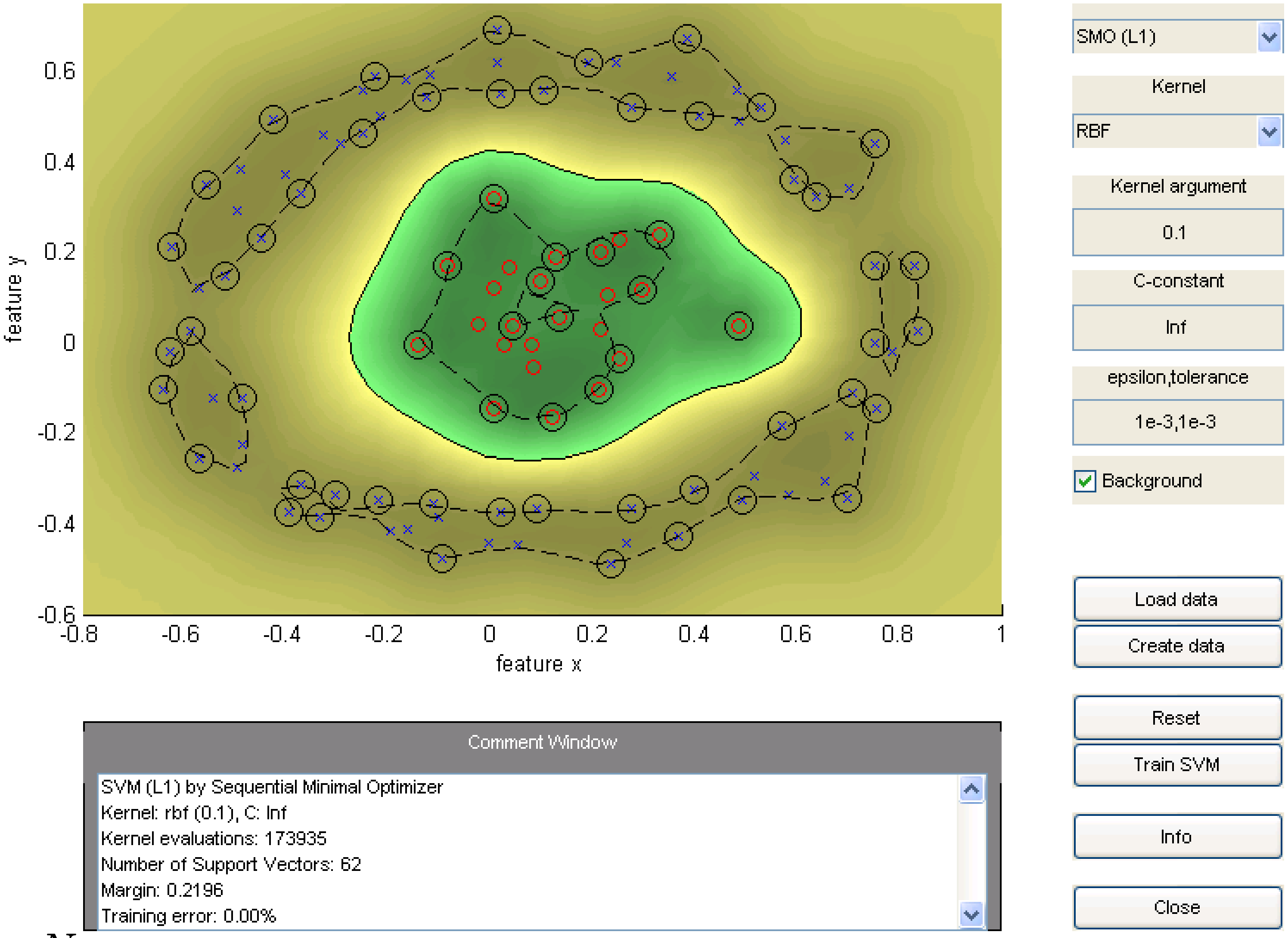
$$f(\mathbf{x}) = \sum_{i}^{N} \alpha_i y_i \exp\left(-||\mathbf{x} - \mathbf{x}_i||^2 / 2\sigma^2\right) + b$$

$$\sigma = 0.25 \quad C = \infty$$



Decrease sigma, moves towards nearest neighbour classifier

$$\sigma = 0.1 \quad C = \infty$$



$$f(\mathbf{x}) = \sum_{i}^{N} \alpha_i y_i \exp\left(-||\mathbf{x} - \mathbf{x}_i||^2 / 2\sigma^2\right) + b$$

Kernel Trick - Summary

• Classifiers can be learnt for high dimensional features spaces, without actually having to map the points into the high dimensional space

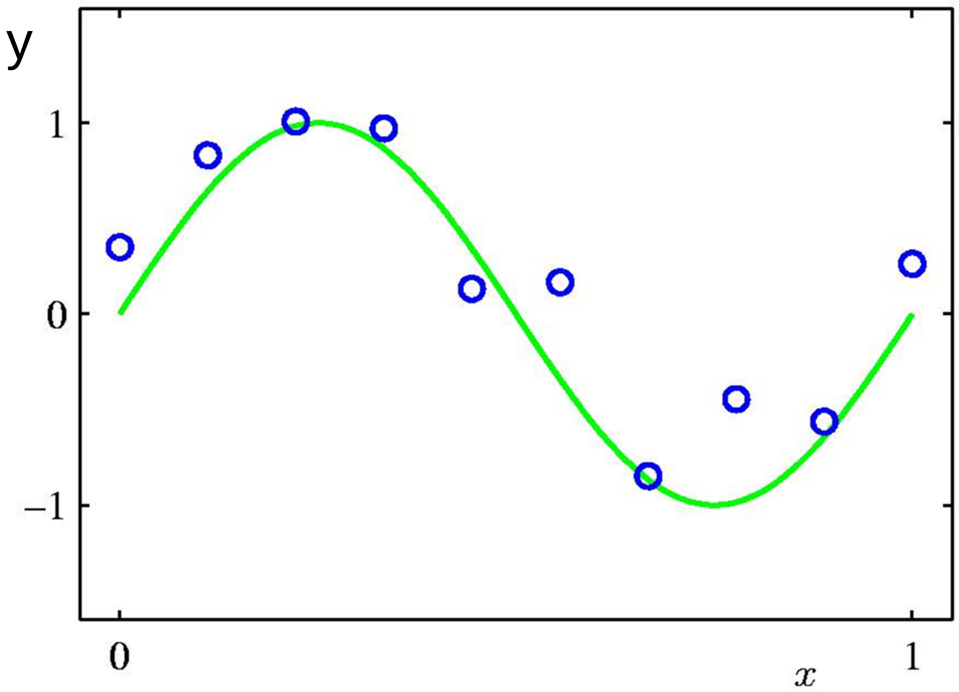
• Data may be linearly separable in the high dimensional space, but not linearly separable in the original feature space

• Kernels can be used for an SVM because of the scalar product in the dual form, but can also be used elsewhere – they are not tied to the SVM formalism

• Kernels apply also to objects that are not vectors, e.g.

$$k(h, h') = \sum_k \min(h_k, h'_k) \text{ for histograms with bins } h_k, h'_k$$

Regression



- Suppose we are given a training set of N observations

$$((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)) \text{ with } \mathbf{x}_i \in \mathbb{R}^d, y_i \in \mathbb{R}$$

- The regression problem is to estimate $f(\mathbf{x})$ from this data such that

$$y_i = f(\mathbf{x}_i)$$

Learning by optimization

- As in the case of classification, learning a regressor can be formulated as an optimization:

Minimize with respect to $f \in \mathcal{F}$

$$\sum_{i=1}^{N} \underbrace{l\left(f(\mathbf{x}_i), y_i\right)}_{\text{loss function}} + \underbrace{\lambda R\left(f\right)}_{\text{regularization}}$$

- There is a choice of both loss functions and regularization
  - e.g. squared loss, SVM “hinge-like” loss
  - squared regularizer, lasso regularizer

Choice of regression function – non-linear basis functions

- Function for regression $y(\mathbf{x}, \mathbf{w})$ is a non-linear function of $\mathbf{x}$, but linear in $\mathbf{w}$:

$$f(\mathbf{x}, \mathbf{w}) = w_0 + w_1 \phi_1(\mathbf{x}) + w_2 \phi_2(\mathbf{x}) + \ldots + w_M \phi_M(\mathbf{x}) = \mathbf{w}^\top \Phi(\mathbf{x})$$

- For example, for $x \in \mathbb{R}$, polynomial regression with $\phi_j(x) = x^j$:

$$f(x, \mathbf{w}) = w_0 + w_1 \phi_1(\mathbf{x}) + w_2 \phi_2(\mathbf{x}) + \ldots + w_M \phi_M(\mathbf{x}) = \sum_{j=0}^{M} w_j x^j$$

e.g. for $M = 3$,

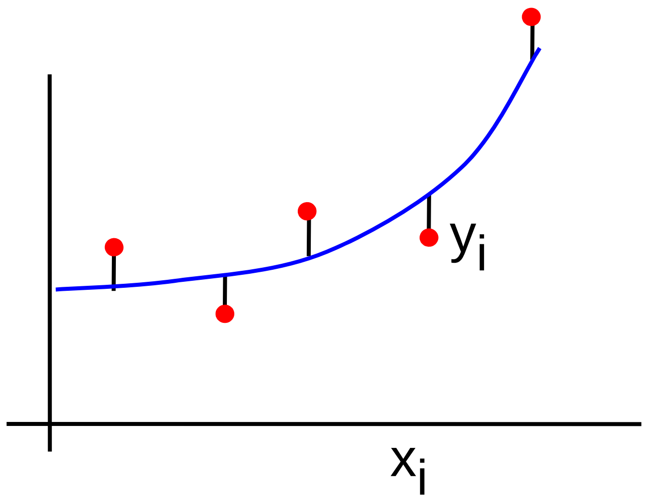
$$f(x, \mathbf{w}) = (w_0, w_1, w_2, w_3) \begin{pmatrix} 1 \\ x \\ x^2 \\ x^3 \end{pmatrix} = \mathbf{w}^\top \Phi(x)$$

$$\Phi : x \to \Phi(x) \quad \mathbb{R}^1 \to \mathbb{R}^4$$

Least squares "ridge regression"

- Cost function – squared loss:

$$\widetilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N} \underbrace{\{f(x_i,\mathbf{w}) - \overset{\text{target value}}{y_i}\}^2}_{\text{loss function}} + \underbrace{\frac{\lambda}{2}\|\mathbf{w}\|^2}_{\text{regularization}}$$



- Regression function for x (1D):

$$f(\mathbf{x},\mathbf{w}) = w_0 + w_1\phi_1(\mathbf{x}) + w_2\phi_2(\mathbf{x}) + \ldots + w_M\phi_M(\mathbf{x}) = \mathbf{w}^\top \Phi(\mathbf{x})$$

- NB squared loss arises in Maximum Likelihood estimation for an error model

$$\underset{\text{measured value}}{y_i} = \underset{\text{true value}}{\tilde{y}_i} + n_i \qquad n_i \sim \mathcal{N}(0,\sigma^2)$$

Solving for the weights w

Notation: write the target and regressed values as N-vectors

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ . \\ . \\ y_N \end{pmatrix} \qquad \mathbf{f} = \begin{pmatrix} \Phi(x_1)^\top \mathbf{w} \\ \Phi(x_2)^\top \mathbf{w} \\ . \\ . \\ \Phi(x_N)^\top \mathbf{w} \end{pmatrix} = \mathbf{\Phi w} = \begin{bmatrix} 1 & \phi_1(x_1) & \dots & \phi_M(x_1) \\ 1 & \phi_1(x_2) & \dots & \phi_M(x_2) \\ . & & & . \\ . & & & . \\ 1 & \phi_1(x_N) & \dots & \phi_M(x_N) \end{bmatrix} \begin{pmatrix} w_0 \\ w_1 \\ . \\ . \\ w_M \end{pmatrix}$$

$\mathbf{\Phi}$ is an $N \times M$ design matrix

e.g. for polynomial regression with basis functions up to x^2

$$\mathbf{\Phi w} = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ . & & . \\ . & & . \\ 1 & x_N & x_N^2 \end{bmatrix} \begin{pmatrix} w_0 \\ w_1 \\ w_2 \end{pmatrix}$$

$$
\begin{aligned}
\widetilde{E}(\mathbf{w}) &= \frac{1}{2}\sum_{i=1}^{N}\{f(x_i,\mathbf{w})-y_i\}^2+\frac{\lambda}{2}\|\mathbf{w}\|^2 \\
&= \frac{1}{2}\sum_{i=1}^{N}\left(y_i-\mathbf{w}^\top\Phi(x_i)\right)^2+\frac{\lambda}{2}\|\mathbf{w}\|^2 \\
&= \frac{1}{2}(\mathbf{y}-\mathbf{\Phi}\mathbf{w})^2+\frac{\lambda}{2}\|\mathbf{w}\|^2
\end{aligned}
$$

Now, compute where derivative w.r.t. $\mathbf{w}$ is zero for minimum

$$
\frac{\widetilde{E}(\mathbf{w})}{d\mathbf{w}} = -\mathbf{\Phi}^\top(\mathbf{y}-\mathbf{\Phi}\mathbf{w})+\lambda\mathbf{w}=\mathbf{0}
$$

Hence

$$
\left(\mathbf{\Phi}^\top\mathbf{\Phi}+\lambda\mathtt{I}\right)\mathbf{w}=\mathbf{\Phi}^\top\mathbf{y}
$$

$$
\mathbf{w}=\left(\mathbf{\Phi}^\top\mathbf{\Phi}+\lambda\mathtt{I}\right)^{-1}\mathbf{\Phi}^\top\mathbf{y}
$$

M basis functions, N data points

$$\mathbf{w} = \left(\mathbf{\Phi}^\top \mathbf{\Phi} + \lambda \mathrm{I}\right)^{-1} \mathbf{\Phi}^\top \mathbf{y}$$

=

assume N > M

M x 1 M x M M x N N x 1

- This shows that there is a unique solution.

- If $\lambda = 0$ (no regularization), then

$$\mathbf{w} = (\mathbf{\Phi}^\top \mathbf{\Phi})^{-1} \mathbf{\Phi}^\top \mathbf{y} = \mathbf{\Phi}^{+} \mathbf{y}$$

  where $\mathbf{\Phi}^{+}$ is the pseudo-inverse of $\mathbf{\Phi}$ (`pinv` in Matlab)

- Adding the term $\lambda \mathrm{I}$ improves the conditioning of the inverse, since if $\mathbf{\Phi}$ is not full rank, then $(\mathbf{\Phi}^\top \mathbf{\Phi} + \lambda \mathrm{I})$ will be (for sufficiently large λ)

- As $\lambda \to \infty$, $\mathbf{w} \to \frac{1}{\lambda} \mathbf{\Phi}^\top \mathbf{y} \to \mathbf{0}$

- Often the regularization is applied only to the inhomogeneous part of $\mathbf{w}$, i.e. to $\tilde{\mathbf{w}}$, where $\mathbf{w} = (w_0, \tilde{\mathbf{w}})$

$$\mathbf{w} = \left(\mathbf{\Phi}^\top \mathbf{\Phi} + \lambda \mathtt{I}\right)^{-1} \mathbf{\Phi}^\top \mathbf{y}$$

$$\begin{aligned} f(x, \mathbf{w}) &= \mathbf{w}^\top \Phi(x) = \Phi(x)^\top \mathbf{w} \\ &= \Phi(x)^\top \left(\mathbf{\Phi}^\top \mathbf{\Phi} + \lambda \mathtt{I}\right)^{-1} \mathbf{\Phi}^\top \mathbf{y} \\ &= \mathbf{b}(x)^\top \mathbf{y} \end{aligned}$$

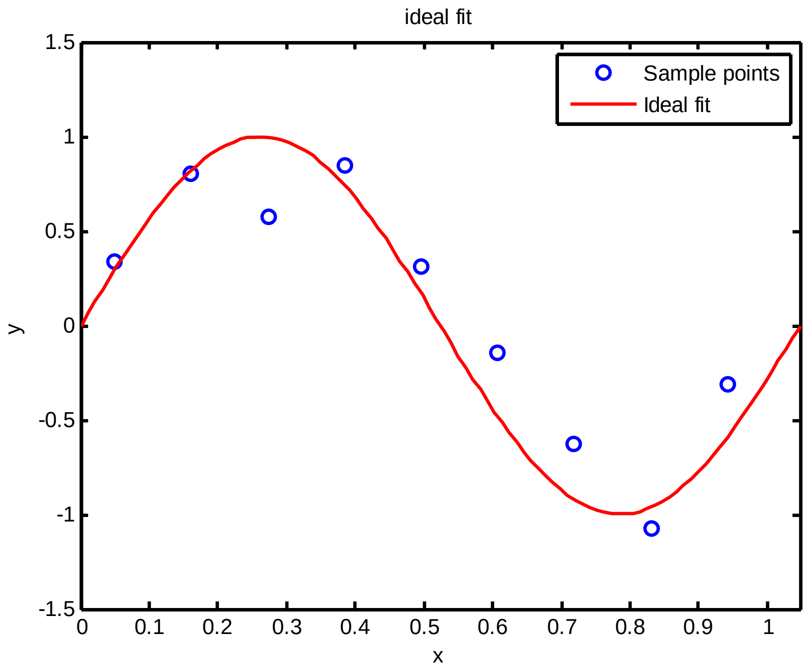
Output is a linear blend, $\mathbf{b}(x)$, of the training values $\{y_i\}$

Example 1: polynomial basis functions

• The red curve is the true function (which is not a polynomial)

• The data points are samples from the curve with added noise in y.

• There is a choice in both the degree, M, of the basis functions used, and in the strength of the regularization

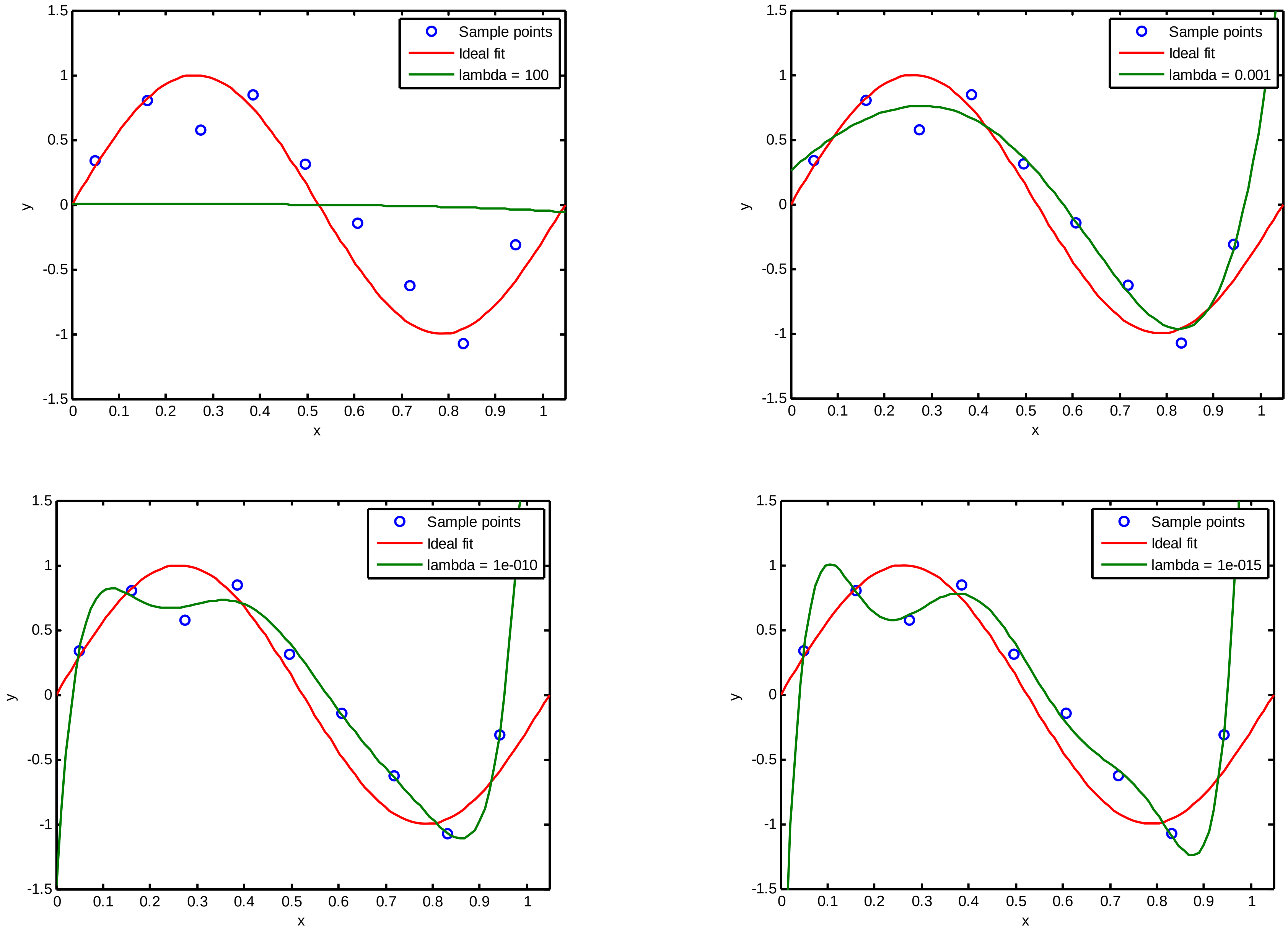


$$f(x, \mathbf{w}) = \sum_{j=0}^{M} w_j x^j = \mathbf{w}^\top \Phi(x) \qquad \Phi : x \to \Phi(x) \quad \mathbb{R} \to \mathbb{R}^{M+1}$$

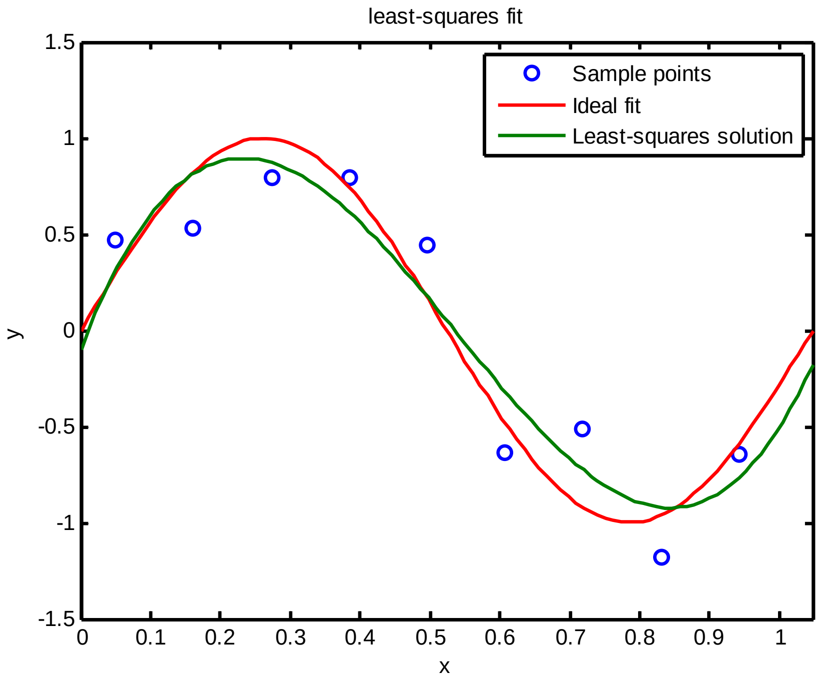
$$\widetilde{E}(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^{N} \{f(x_i, \mathbf{w}) - y_i\}^2 + \frac{\lambda}{2} \|\mathbf{w}\|^2$$

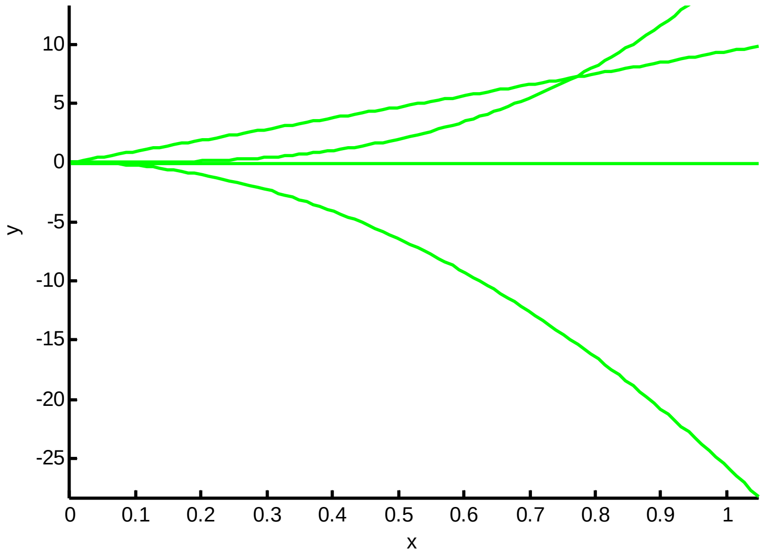
w is a M+1 dimensional vector

N = 9 samples, M = 7

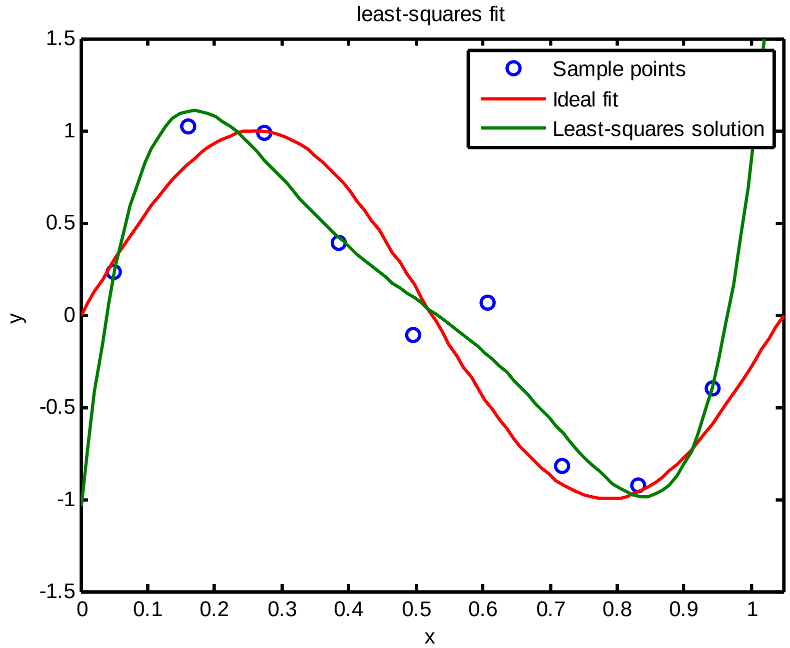


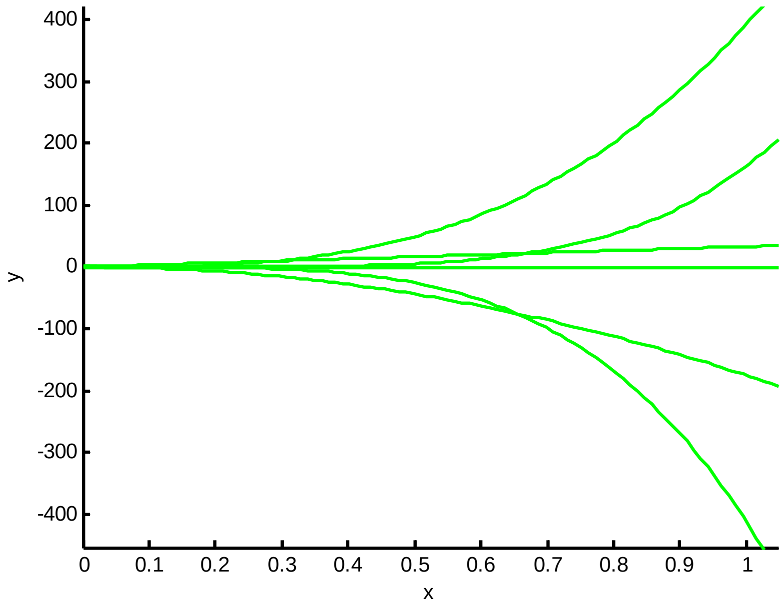
M = 3


least-squares fit
Sample points
Ideal fit
Least-squares solution
x
y


x
y

M = 5

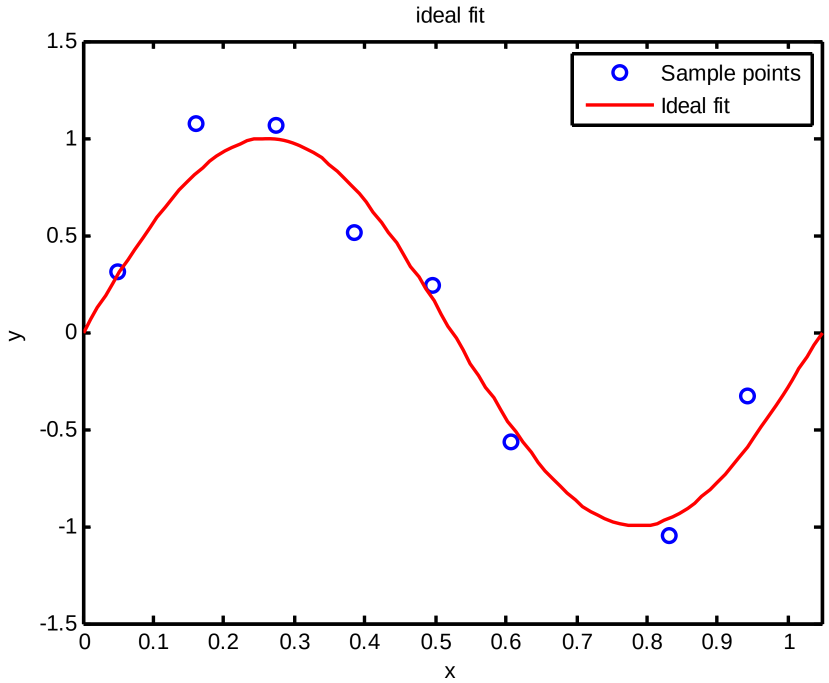

least-squares fit
Sample points
Ideal fit
Least-squares solution
x
y


x
y

Example 2: Gaussian basis functions

- The red curve is the true function (which is not a polynomial)
- The data points are samples from the curve with added noise in y.

- Basis functions are centred on the training data (N points)
- There is a choice in both the scale, sigma, of the basis functions used, and in the strength of the regularization

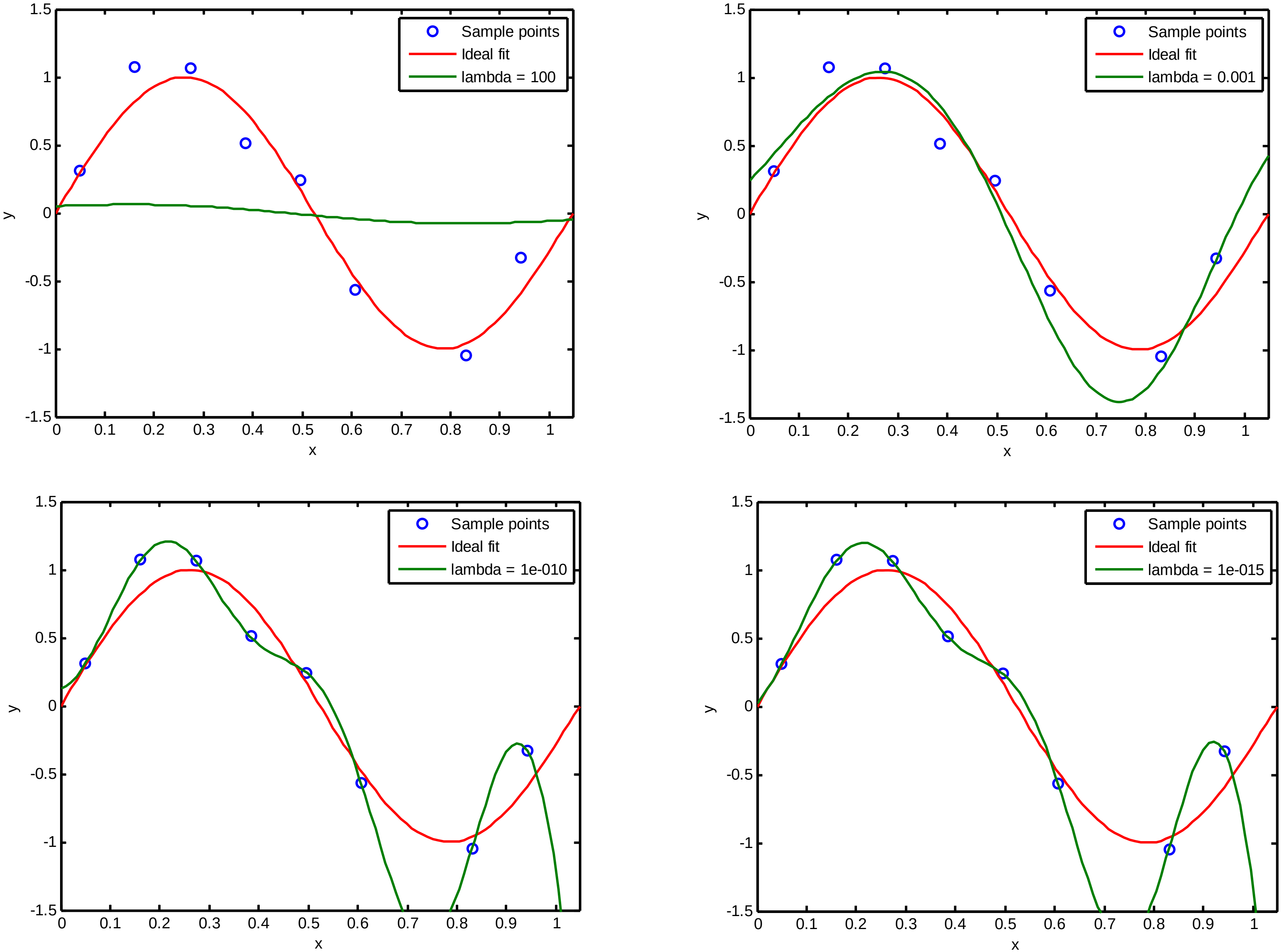


$$f(x, \mathbf{w}) = \sum_{i=1}^{N} w_i e^{-(x-x_i)^2/\sigma^2} = \mathbf{w}^\top \Phi(x) \qquad \Phi : x \rightarrow \Phi(x) \quad \mathbb{R} \rightarrow \mathbb{R}^N$$

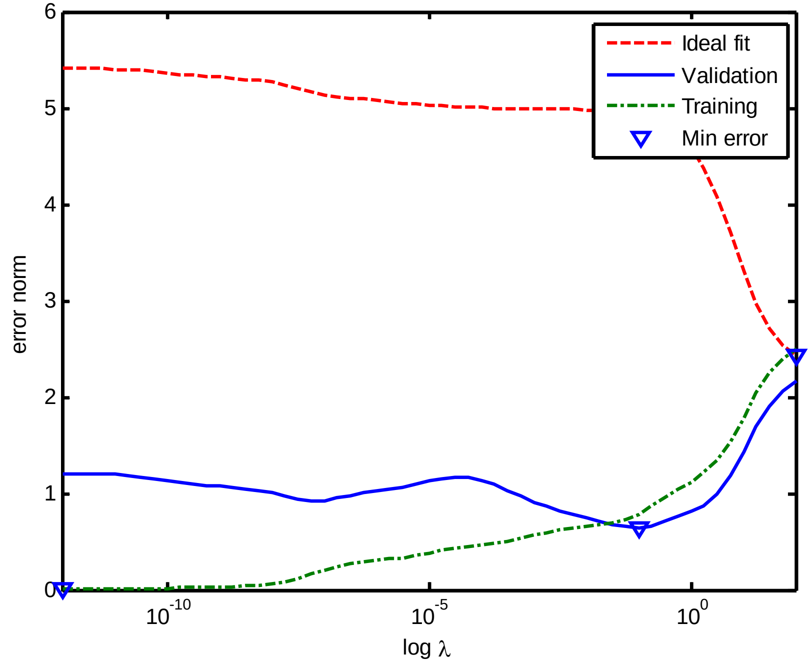
$$\widetilde{E}(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^{N} \{f(x_i, \mathbf{w}) - y_i\}^2 + \frac{\lambda}{2} \|\mathbf{w}\|^2$$

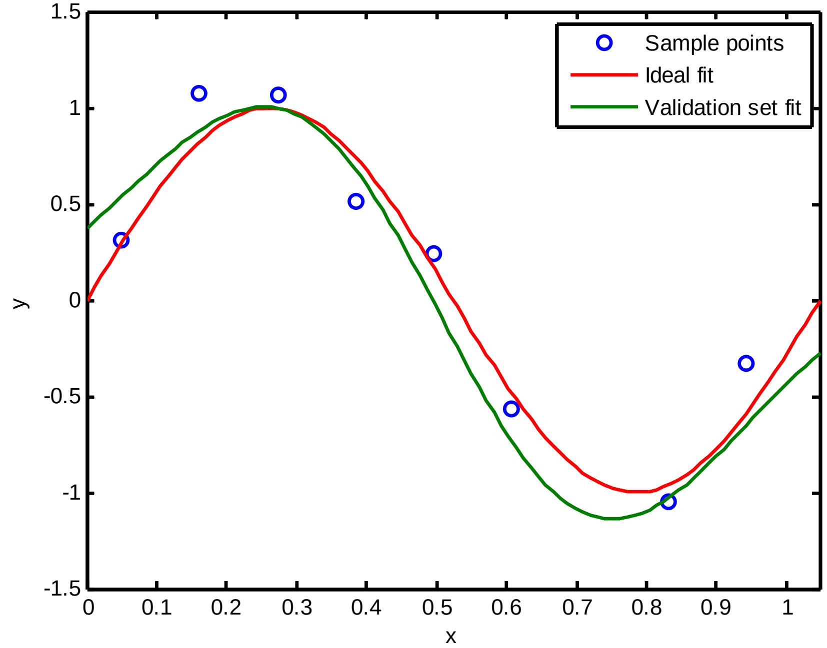
w is a N-vector

N = 9 samples, sigma = 0.334

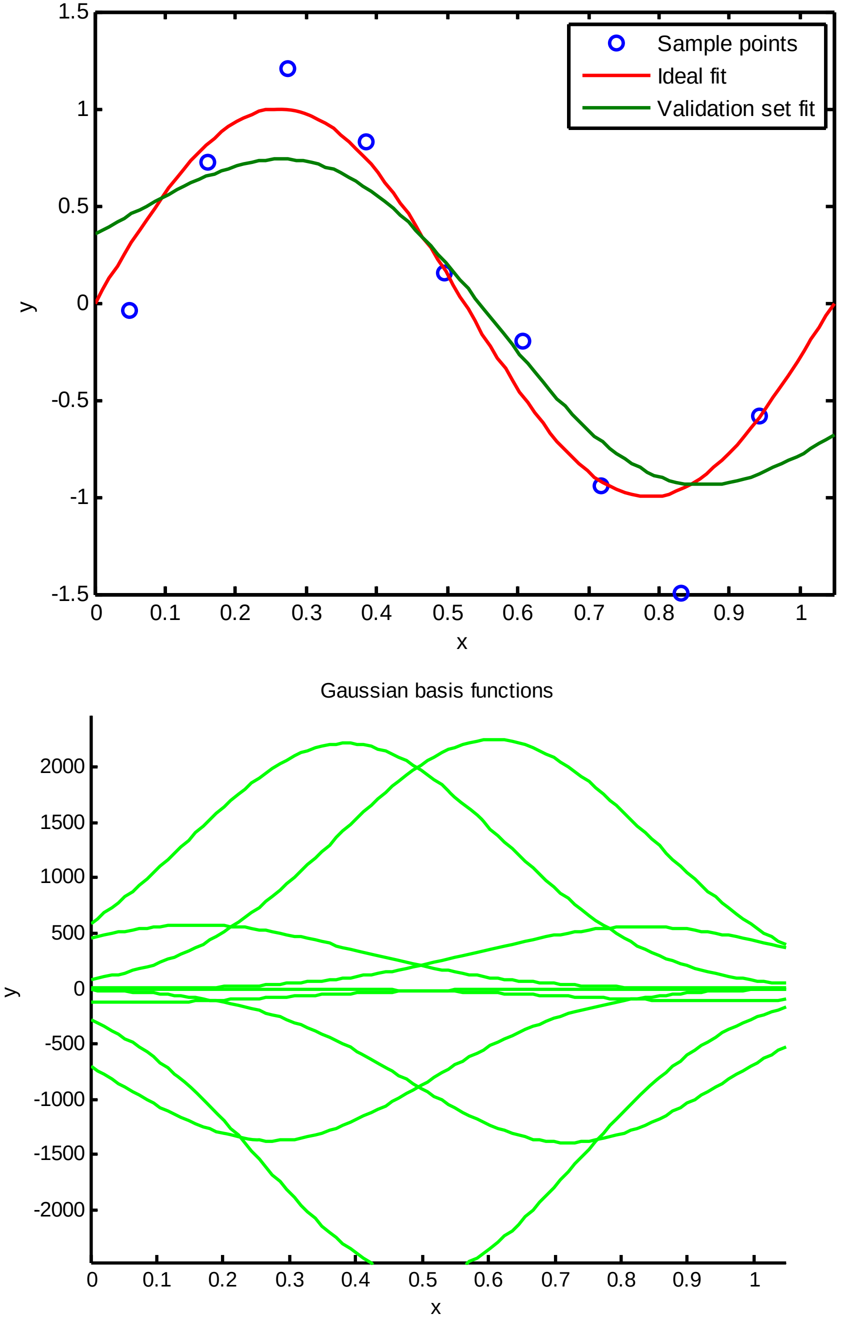


Choosing lambda using a validation set

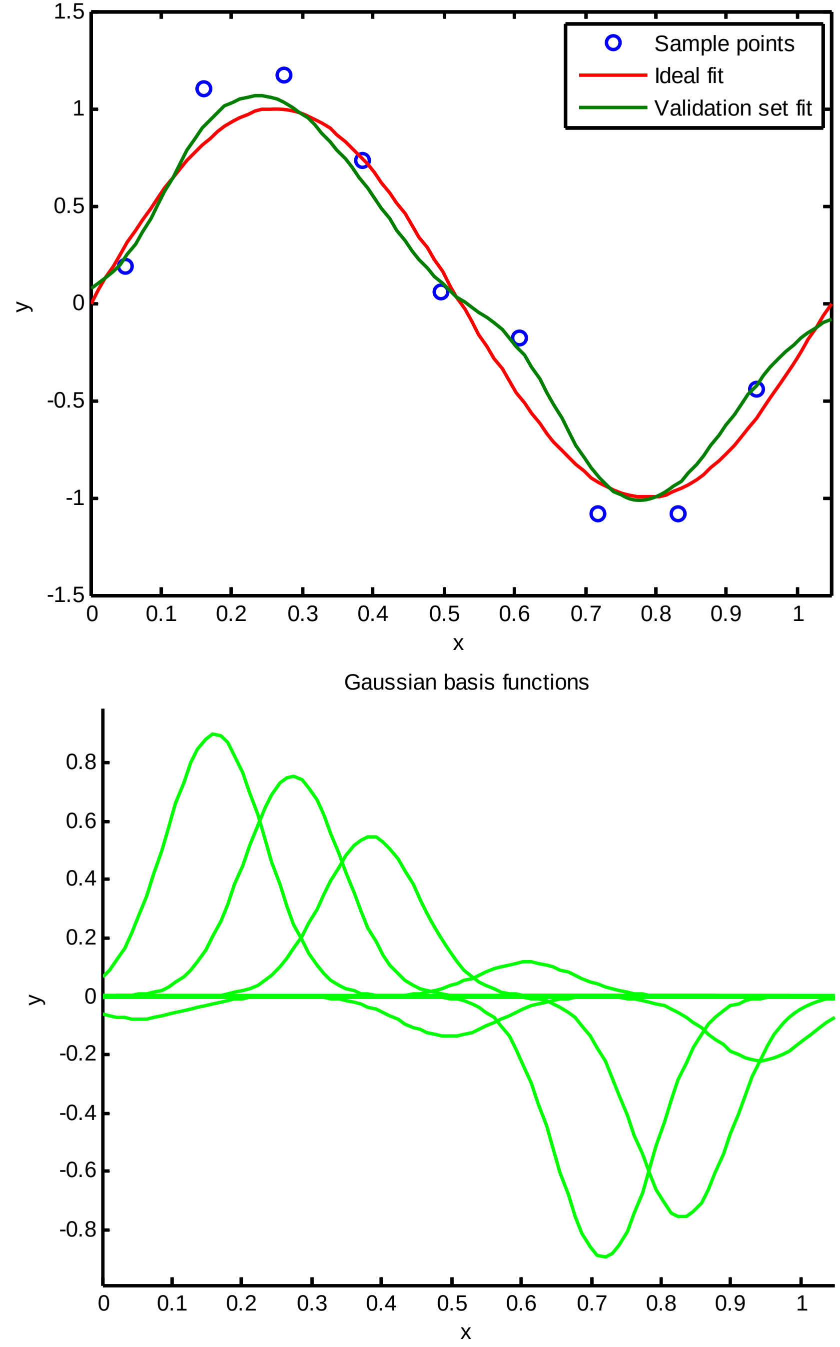




Sigma = 0.334

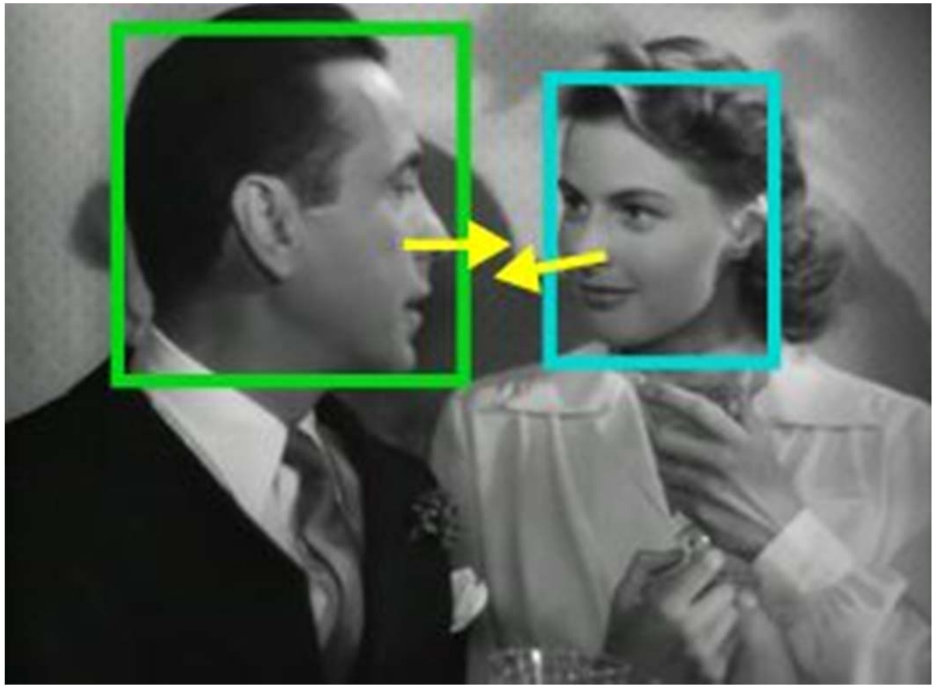


Sigma = 0.1



Application: regressing face pose

- Estimate two face pose angles:
    - yaw (around the Y axis)
    - pitch (around the X axis)



- Compute a HOG feature vector for each face region

- Learn a regressor from the HOG vector to the two pose angles

Summary and dual problem

So far we have considered the primal problem where

$$f(\mathbf{x}, \mathbf{w}) = \sum_{i=1}^{M} w_i \phi_i(\mathbf{x}) = \mathbf{w}^\top \mathbf{\Phi}(\mathbf{x})$$

and we wanted a solution for $\mathbf{w} \in \mathbb{R}^M$

As in the case of SVMs, we can also consider the dual problem where

$$\mathbf{w} = \sum_{i=1}^{N} a_i \mathbf{\Phi}(x_i) \quad \text{and} \quad f(\mathbf{x}, \mathbf{a}) = \sum_{i} a_i \mathbf{\Phi}(x_i)^\top \mathbf{\Phi}(x)$$

and obtain a solution for $\mathbf{a} \in \mathbb{R}^N$.

Again

- there is a closed form solution for $\mathbf{a}$,
- the solution involves the $N \times N$ Gram matrix $k(x_i, x_j) = \mathbf{\Phi}(x_i)^\top \mathbf{\Phi}(x_j)$,
- so we can use the kernel trick again to replace scalar products

Background reading and more

- Bishop, chapters 6 & 7 for kernels and SVMs
- Hastie et al, chapter 12
- Bishop, chapter 3 for regression
- More on web page:

  http://www.robots.ox.ac.uk/~az/lectures/ml