LECTURE 5: DUAL PROBLEMS AND KERNELS

* Most of the slides in this lecture are from http://www.robots.ox.ac.uk/~az/lectures/ml

”

Optimization

Learning an SVM has been formulated as a constrained optimization problem over $\mathbf{w}$ and ξ

$$\min_{\mathbf{w}\in\mathbb{R}^d, \xi_i \in \mathbb{R}^+} ||\mathbf{w}||^2 + C\sum_i^N \xi_i \text{ subject to } y_i\left(\mathbf{w}^\top \mathbf{x}_i + b\right) \geq 1 - \xi_i \text{ for } i = 1 \ldots N$$

The constraint $y_i\left(\mathbf{w}^\top \mathbf{x}_i + b\right) \geq 1 - \xi_i$, can be written more concisely as

$$y_i f(\mathbf{x}_i) \geq 1 - \xi_i$$

which, together with $\xi_i \geq 0$, is equivalent to

$$\xi_i = \max\left(0, 1 - y_i f(\mathbf{x}_i)\right)$$

Hence the learning problem is equivalent to the unconstrained optimization problem over $\mathbf{w}$

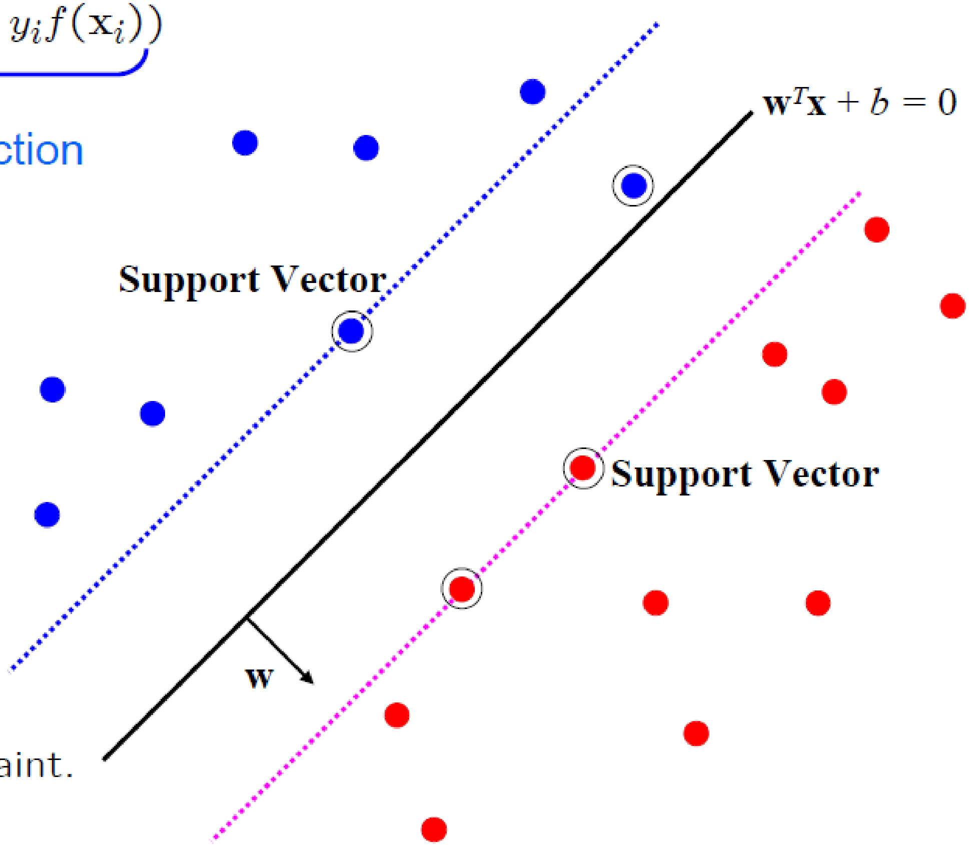
$$\min_{\mathbf{w}\in\mathbb{R}^d} \underbrace{||\mathbf{w}||^2}_{\text{regularization}} + C\sum_i^N \underbrace{\max\left(0, 1 - y_i f(\mathbf{x}_i)\right)}_{\text{loss function}}$$

Loss function

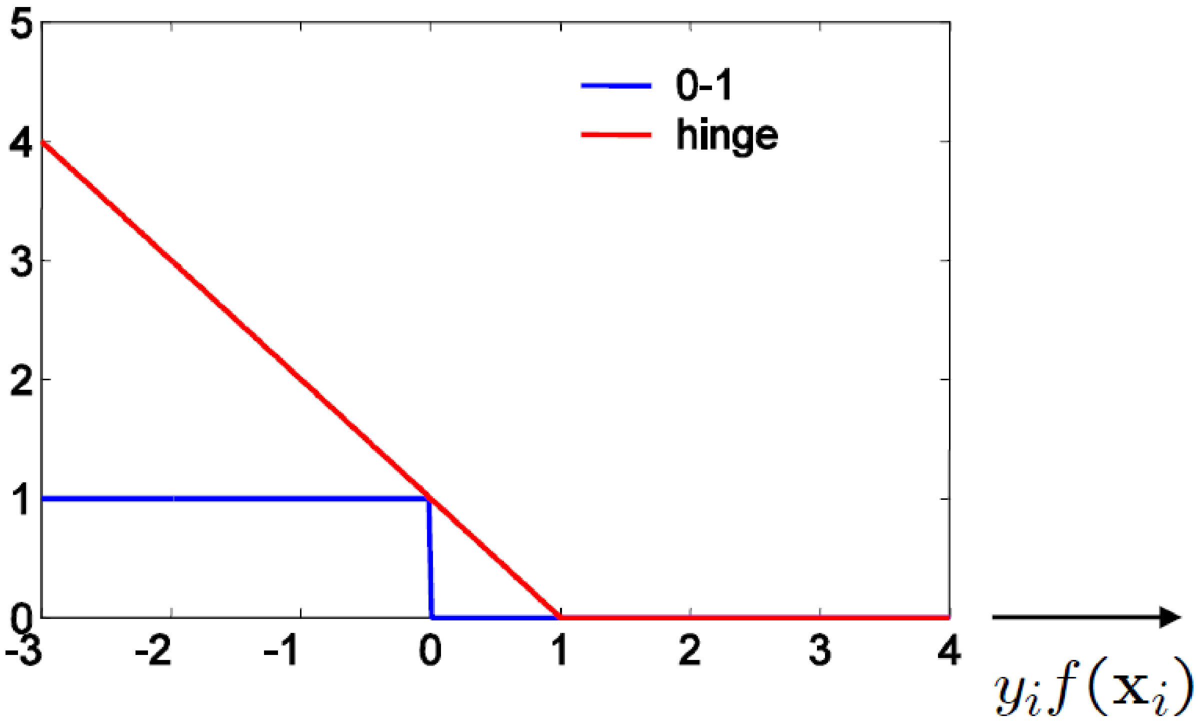
$$\min_{\mathbf{w}\in\mathbb{R}^d} ||\mathbf{w}||^2 + C\sum_i^N \underbrace{\max\left(0, 1 - y_i f(\mathbf{x}_i)\right)}_{\text{loss function}}$$

Points are in three categories:

1. $y_i f(x_i) > 1$
   Point is outside margin.
   No contribution to loss
2. $y_i f(x_i) = 1$
   Point is on margin.
   No contribution to loss.
   As in hard margin case.
3. $y_i f(x_i) < 1$
   Point violates margin constraint.
   Contributes to loss



Loss functions



- SVM uses "hinge" loss $\max(0, 1 - y_i f(\mathbf{x}_i))$
- an approximation to the 0-1 loss

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 $\mathbf{w}$:

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

- 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.

PRIMAL-DUAL PROBLEM

Max−min inequality

$$\max_{\lambda} \min_{x} f(x, \lambda) \leq \min_{x} \max_{\lambda} f(x, \lambda)$$

$$g(\lambda) \doteq \min_{x} f(x, \lambda)$$

$$g(\lambda) \leq f(x, \lambda), \forall x$$

$$\max_{\lambda} g(\lambda) \leq \max_{\lambda} f(x, \lambda), \forall x$$

$$\max_{\lambda} g(\lambda) \leq \min_{x} \max_{\lambda} f(x, \lambda)$$

$$\sup_{x \in X} \inf_{y \in Y} f(x, y) \leq \inf_{y \in Y} \sup_{x \in X} f(x, y).$$

The reasoning goes quite straightforwardly from the definitions of sup and inf,

$$\begin{aligned} f(x,y) &\leq \sup_{x \in X} f(x,y), \forall x, y \\ \inf_{y \in Y} f(x,y) &\leq \sup_{x \in X} f(x,y) \\ \sup_{x \in X} \inf_{y \in Y} f(x,y) &\leq \sup_{x \in X} f(x,y) \\ \sup_{x \in X} \inf_{y \in Y} f(x,y) &\leq \inf_{y \in Y} \sup_{x \in X} f(x,y). \end{aligned}$$

$$\min_{x,y} \frac{1}{2}\left(x^2+y^2\right) \quad s.t. \;\; x+y=1$$

$$g(x,y) \doteq \begin{cases} \frac{1}{2}\left(x^2+y^2\right) & x+y=1 \\ \infty & \text{otherwise} \end{cases}$$

$$\begin{aligned} \min_{x,y} g(x,y) &= \min_{x,y} \max_{\lambda} \frac{1}{2}\left(x^2+y^2\right) + \lambda(x+y-1) \\ &\geq \max_{\lambda} \min_{x,y} \frac{1}{2}\left(x^2+y^2\right) + \lambda(x+y-1) \\ &= \max_{\lambda}(-\lambda-\lambda^2) = \frac{1}{4} \end{aligned}$$

$$\min_{x,y} \frac{1}{2}\left(x^2 + y^2\right) \quad s.t. \;\; x + y \geq 1$$

$$g(x,y) \doteq \begin{cases} \frac{1}{2}\left(x^2 + y^2\right) & x + y \geq 1 \\ \infty & \text{otherwise} \end{cases}$$

$$p^* = \min_{x,y} g(x,y) = \min_{x,y} \max_{\lambda \geq 0} \frac{1}{2}\left(x^2 + y^2\right) - \lambda(x + y - 1)$$

$$\geq \max_{\lambda \geq 0} \min_{x,y} \frac{1}{2}\left(x^2 + y^2\right) - \lambda(x + y - 1) = d^*$$

$$= \max_{\lambda \geq 0} (\lambda - \lambda^2) = \frac{1}{4}$$

$$\min_{x,y} \frac{1}{2}\left(x^2+y^2\right) \qquad s.t. \quad x+y \geq 1$$
$$y \geq 0$$

$$g(x,y) \doteq \begin{cases} \frac{1}{2}\left(x^2+y^2\right) & x+y \geq 1 \;\; \& \;\; y \geq 0 \\ \infty & \text{otherwise} \end{cases}$$

$$p^* = \min_{x,y} g(x,y) = \min_{x,y} \max_{\lambda\geq 0, \mu \geq 0} \frac{1}{2}\left(x^2+y^2\right) - \lambda(x+y-1) - \mu(y-1/2)$$

$$\geq \min_{x,y} \max_{\lambda\geq 0, \mu \geq 0} \frac{1}{2}\left(x^2+y^2\right) - \lambda(x+y-1) - \mu(y)$$

$$x = \lambda, y = \lambda + \mu$$

$$= \max_{\lambda\geq 0, \mu \geq 0} \left(\lambda - \lambda^2 - \frac{1}{2}\mu^2 - \mu\lambda\right)$$

$$\lambda = \frac{1}{2}, \mu = 0$$

$$= \frac{1}{4}$$

Duality gap

$$p^* - d^*$$

Example

$$\min_{x,y} \frac{1}{2}\left(x^2 + y^2\right)$$

$$\begin{aligned} s.t. \quad & x + y \geq 1 \\ & x \geq 0 \end{aligned}$$

$$\begin{aligned} p^* &= \min_{x,y} \max_{\lambda\geq 0,\mu\geq 0} \frac{1}{2}\left(x^2 + y^2\right) - \lambda(x + y - 1) - \mu(x) \\ &\geq \max_{\lambda\geq 0,\mu\geq 0} \min_{x,y} \frac{1}{2}\left(x^2 + y^2\right) - \lambda(x + y - 1) - \mu(x) = d^* \end{aligned}$$

PRIMAL-DUAL PROBLEM:GEOMETRIC INTERPRETATION

Necessary conditions [edit]

Suppose that the objective function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ and the constraint functions $g_i : \mathbb{R}^n \rightarrow \mathbb{R}$ and $h_j : \mathbb{R}^n \rightarrow \mathbb{R}$ are continuously differentiable at a point x^*. If x^* is a local optimum and the optimization problem satisfies some regularity conditions (see below), then there exist constants $\mu_i \ (i = 1, \ldots, m)$ and $\lambda_j \ (j = 1, \ldots, \ell)$, called KKT multipliers, such that

Stationarity

For maximizing $f(x)$: $\nabla f(x^*) = \sum_{i=1}^{m} \mu_i \nabla g_i(x^*) + \sum_{j=1}^{\ell} \lambda_j \nabla h_j(x^*)$,

For minimizing $f(x)$: $-\nabla f(x^*) = \sum_{i=1}^{m} \mu_i \nabla g_i(x^*) + \sum_{j=1}^{\ell} \lambda_j \nabla h_j(x^*)$,

Primal feasibility

$g_i(x^*) \leq 0, \text{ for } i = 1, \ldots, m$

$h_j(x^*) = 0, \text{ for } j = 1, \ldots, \ell$

Dual feasibility

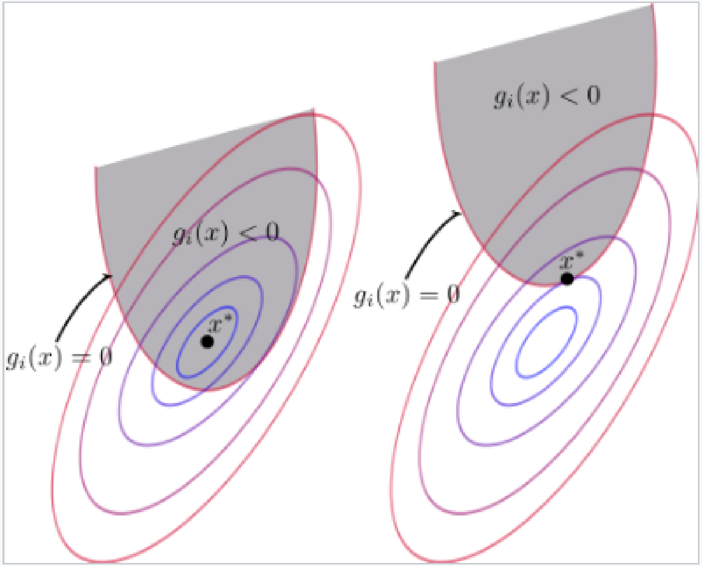
$\mu_i \geq 0, \text{ for } i = 1, \ldots, m$

Complementary slackness

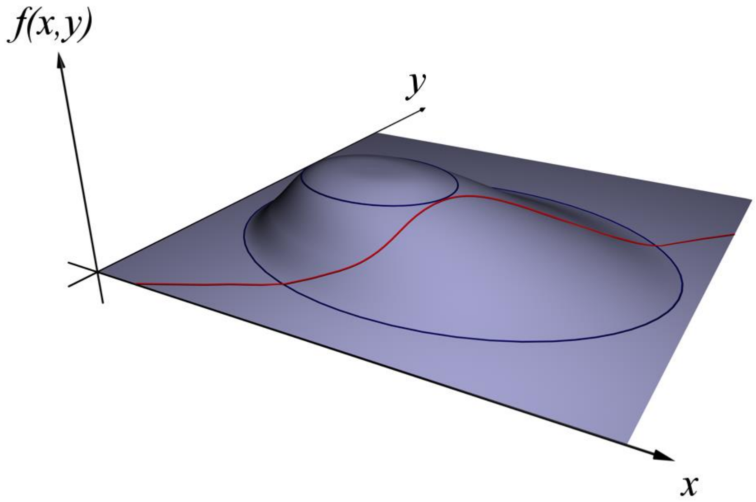
$\mu_i g_i(x^*) = 0, \text{ for } i = 1, \ldots, m.$

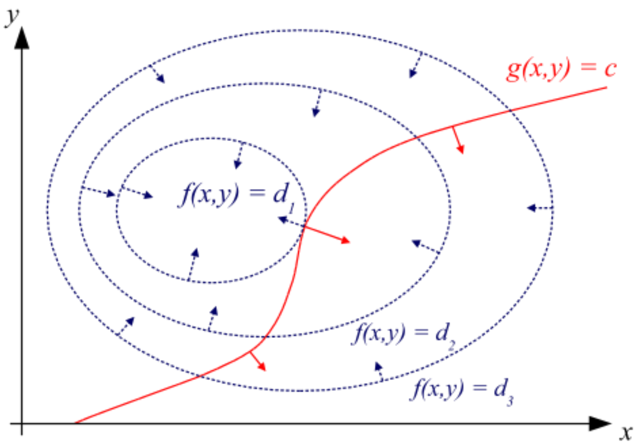
In the particular case $m = 0$, i.e., when there are no inequality constraints, the KKT conditions turn into the Lagrange conditions, and the KKT multipliers are called Lagrange multipliers.

If some of the functions are non-differentiable, subdifferential versions of Karush–Kuhn–Tucker (KKT) conditions are available.[5]



Inequality constraint diagram for optimization problems





Find x and y to maximize $f(x, y)$ subject to a constraint (shown in red) $g(x, y) = c$.

The red line shows the constraint $g(x, y) = c$. The blue lines are contours of $f(x, y)$. The point where the red line tangentially touches a blue contour is the solution. Since $d_1 > d_2$, the solution is a maximization of $f(x, y)$.

DUAL FORM OF SVM

Primal Form

$$\min_{w,b,\xi_i \geq 0} \left(\frac{1}{2} \|w\|^2 + C \sum_{i=1}^{N} \xi_i \right)$$

$$\begin{aligned} s.t. \quad & y_i(w^\top x_i + b) \geq 1 - \xi_i \\ & \xi_i \geq 0 \qquad\qquad\qquad (i = 1, \ldots, N) \end{aligned}$$

$$\min_{w,b,\xi_i\geq 0}\max_{\alpha_i\geq 0,\beta_i\geq 0}\left(\frac{1}{2}\|w\|^2+\sum_i\left(C\xi_i-\alpha_i(y_i(w^\top x_i+b)-1+\xi_i)-\beta_i\xi_i\right)\right)$$

$$\geq\max_{\alpha_i\geq 0,\beta_i\geq 0}\min_{w,b,\xi_i\geq 0}\left(\frac{1}{2}\|w\|^2+\sum_i\left(C\xi_i-\alpha_i(y_i(w^\top x_i+b)-1+\xi_i)-\beta_i\xi_i\right)\right)$$

$$=\max_{\alpha_i\geq 0,\beta_i\geq 0}\min_{w,b,\xi_i\geq 0}\left(\frac{1}{2}\|w\|^2+\sum_i\left((C-\alpha_i-\beta_i)\xi_i-\alpha_i(y_i(w^\top x_i+b)-1)\right)\right)$$

$$w=\sum_i\alpha_i y_i x_i$$

$$C=\alpha_i+\beta_i$$

$$\sum_i\alpha_i y_i=0$$

$$=\max_{\alpha_i\geq 0,\beta_i\geq 0}\left(\sum_i\alpha_i-\frac{1}{2}\|w\|^2\right)=\max_{\alpha_i\geq 0,\beta_i\geq 0}\left(\sum_i\alpha_i-\frac{1}{2}\sum_i\sum_j\alpha_i\alpha_j y_i y_j x_i^\top x_j\right)$$

$$=\max_{0\leq\alpha_i\leq C}\left(\sum_i\alpha_i-\frac{1}{2}\sum_i\sum_j\alpha_i\alpha_j y_i y_j x_i^\top x_j\right)$$

The Representer Theorem

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

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

Primal and dual formulations

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

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

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

Dual problem: for $\boldsymbol{\alpha} \in \mathbb{R}^N$

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

- 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:

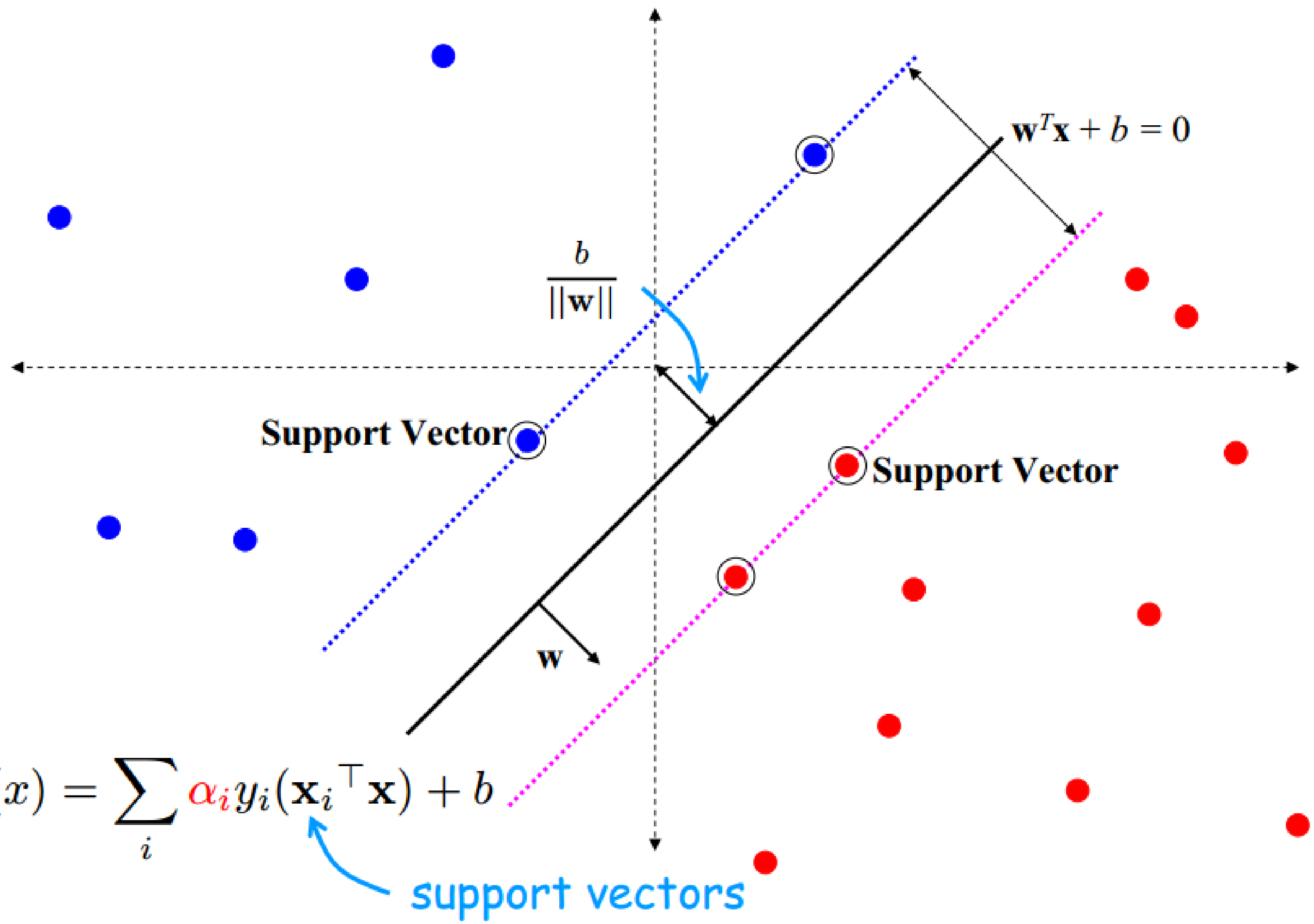
$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$$

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 $\mathbf{x}_i$.

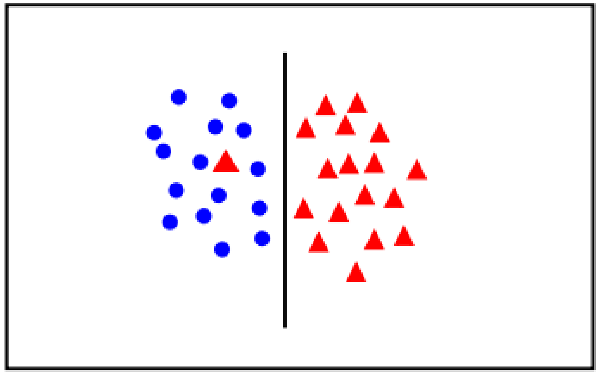
Support Vector Machine



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

KERNEL TRICK

Handling data that is not linearly separable

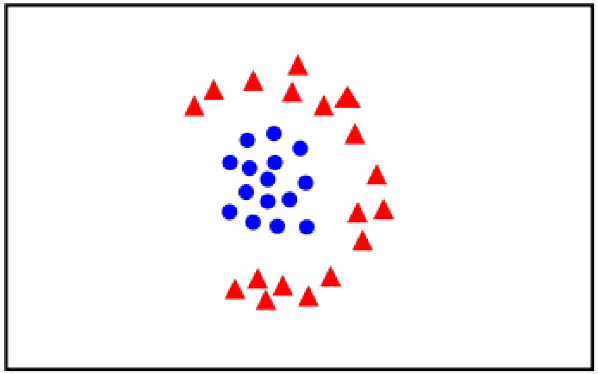


- introduce slack variables

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

subject to

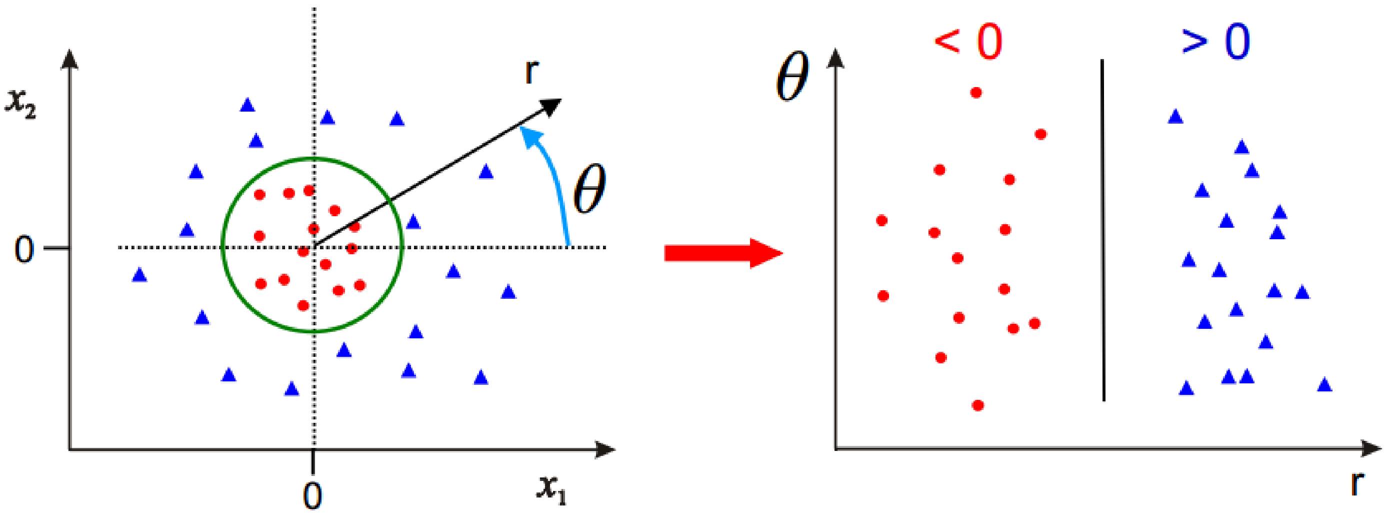
$$y_i \left(\mathbf{w}^\top \mathbf{x}_i + b \right) \geq 1 - \xi_i \text{ for } i = 1 \dots N$$



- linear classifier not appropriate

??

Solution 1: use polar coordinates

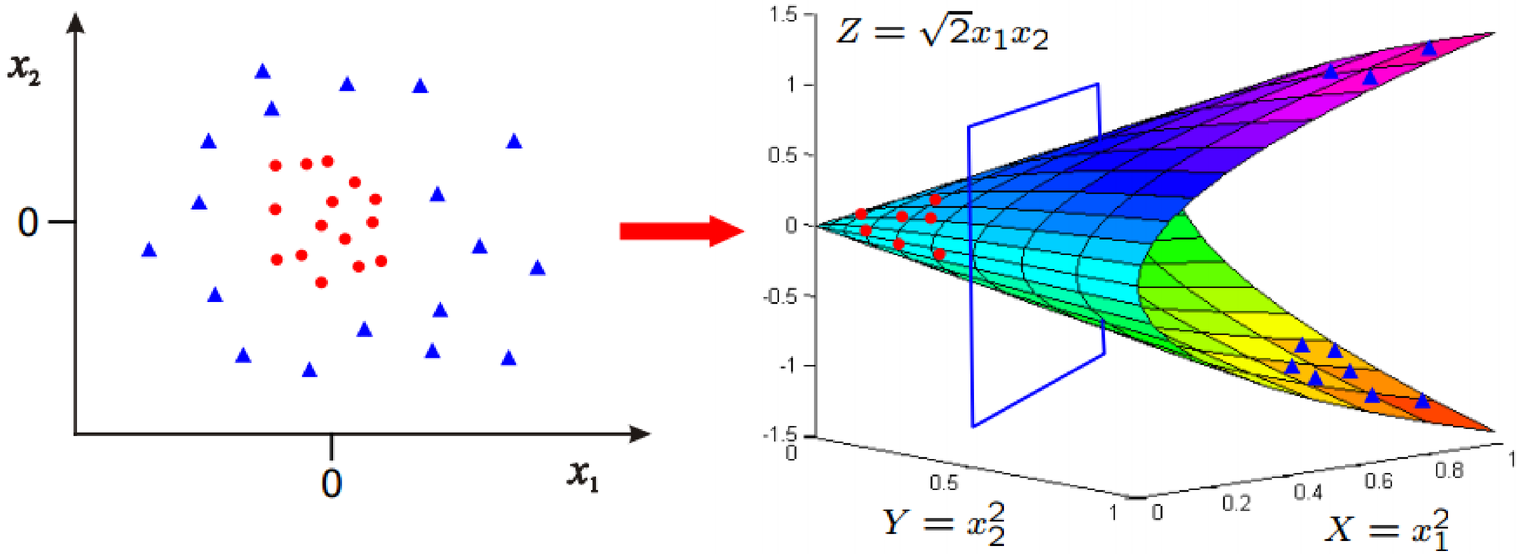


- Data is linearly separable in polar coordinates
- Acts non-linearly in original space

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

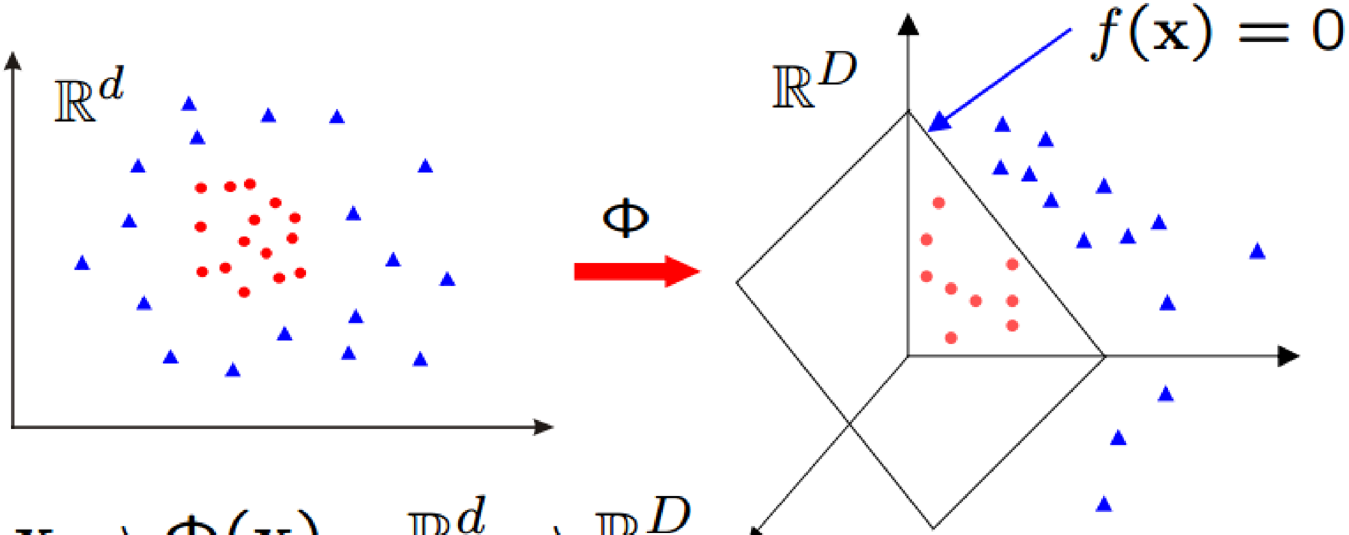
Solution 2: map data to higher dimension

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



- 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



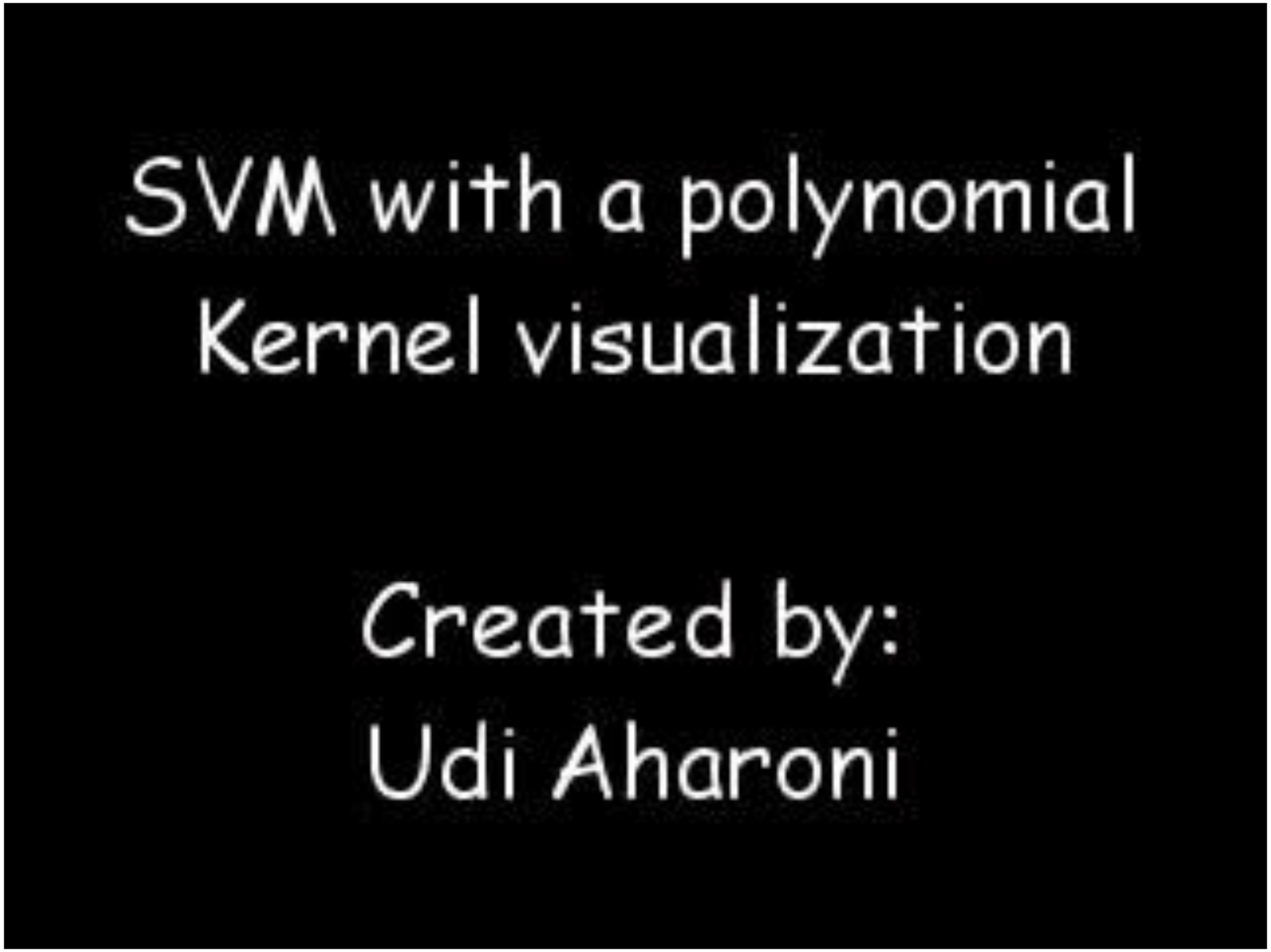
$\Phi : \mathbf{x} \rightarrow \Phi(\mathbf{x}) \quad \mathbb{R}^d \rightarrow \mathbb{R}^D$

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

Kernel trick visualization



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 $\mathbf{w} \in \mathbb{R}^D$

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

- 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:

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

$$\rightarrow f(\mathbf{x}) = \sum_{i}^{N} \alpha_i y_i \Phi(\mathbf{x}_i)^\top \Phi(\mathbf{x}) + b$$

Learning:

$$\max_{\alpha_i \geq 0} \sum_i \alpha_i - \frac{1}{2} \sum_{jk} \alpha_j \alpha_k y_j y_k \mathbf{x}_j{}^\top \mathbf{x}_k$$

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

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

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

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

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

Valid kernels – when can the kernel trick be used?

- Given some arbitrary function $k(\mathbf{x}_i, \mathbf{x}_j)$, how do we know if it corresponds to a scalar product $\Phi(\mathbf{x}_i)^\top \Phi(\mathbf{x}_j)$ in some space?

- Mercer kernels: if $k(,)$ satisfies:
  - Symmetric $k(\mathbf{x}_i, \mathbf{x}_j) = k(\mathbf{x}_j, \mathbf{x}_i)$
  - Positive definite, $\boldsymbol{\alpha}^\top \mathtt{K} \boldsymbol{\alpha} \geq 0$ for all $\boldsymbol{\alpha} \in \mathbb{R}^N$, where $\mathtt{K}$ is the $N \times N$ Gram matrix with entries $\mathtt{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$.

  then $k(,)$ is a valid kernel.

- e.g. $k(\mathbf{x}, \mathbf{z}) = \mathbf{x}^\top \mathbf{z}$ is a valid kernel, $k(\mathbf{x}, \mathbf{z}) = \mathbf{x} - \mathbf{x}^\top \mathbf{z}$ is not.

Kernel Trick – Summary

- Classifiers can be learned 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

KERNEL SVM EXAMPLE

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)

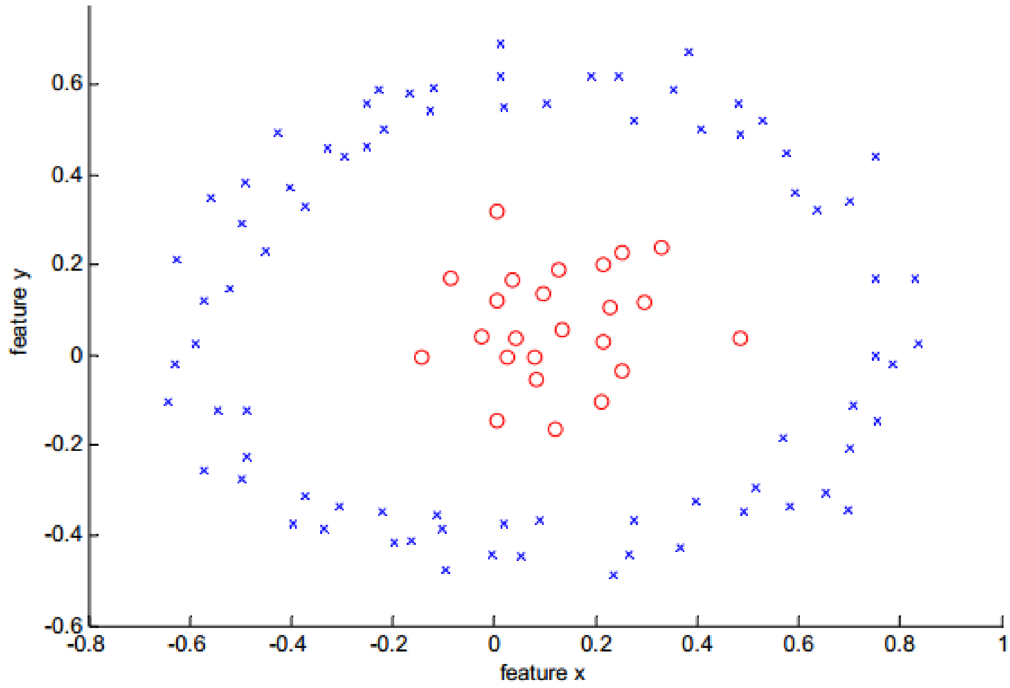
support vector

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



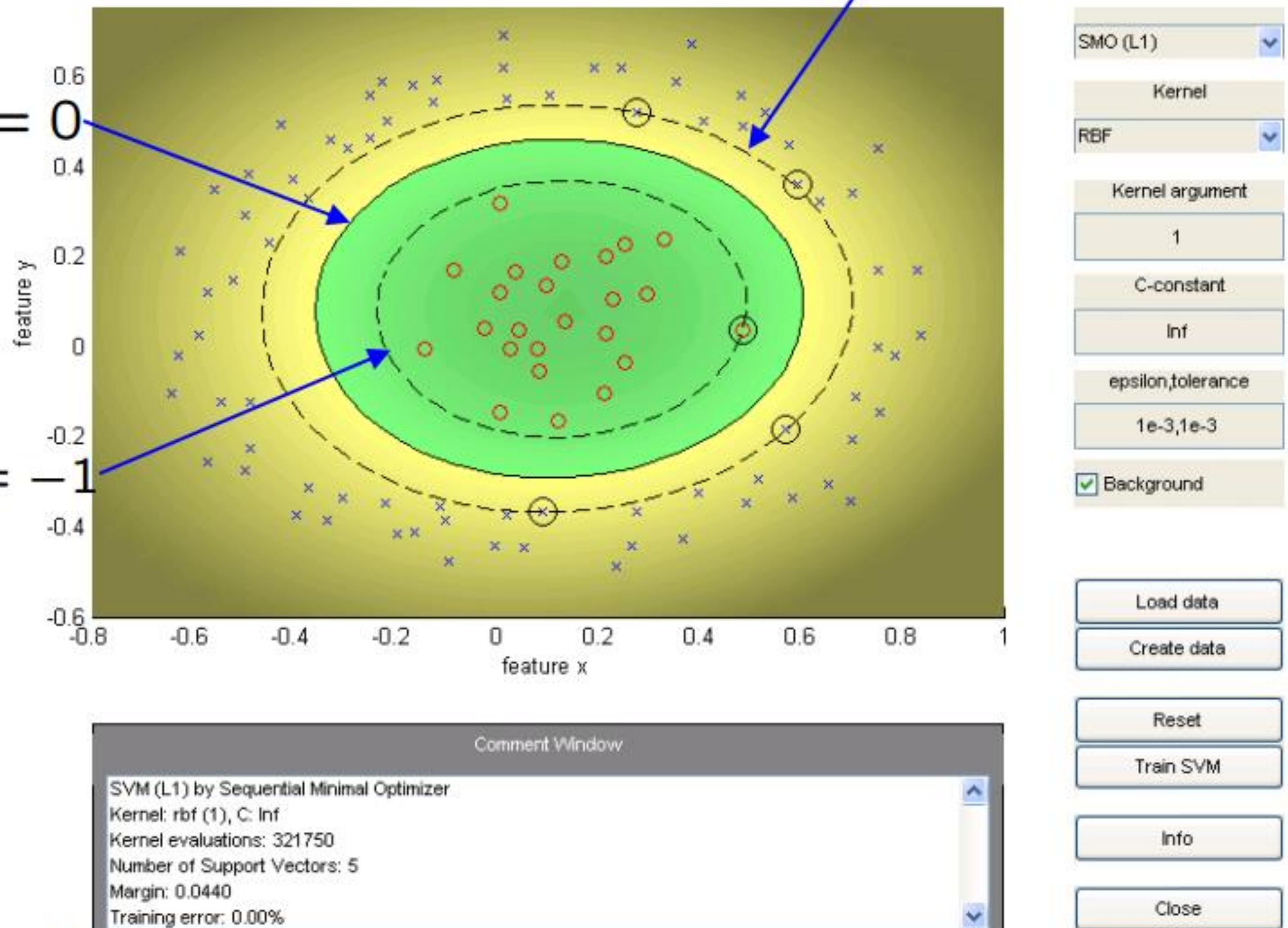
- data is not linearly separable in original feature space

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

$f(\mathbf{x}) = 1$

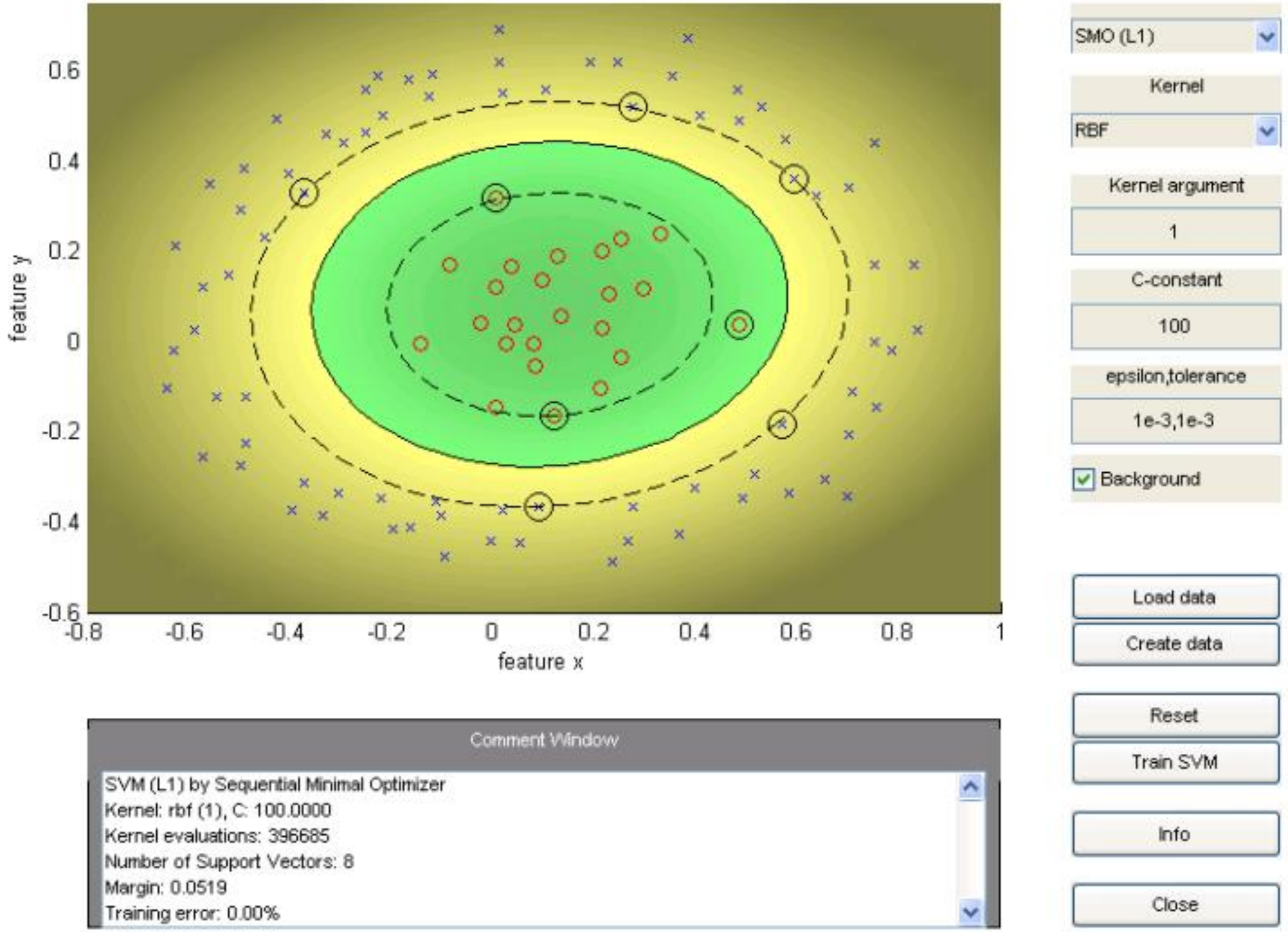
$f(\mathbf{x}) = 0$

$f(\mathbf{x}) = -1$



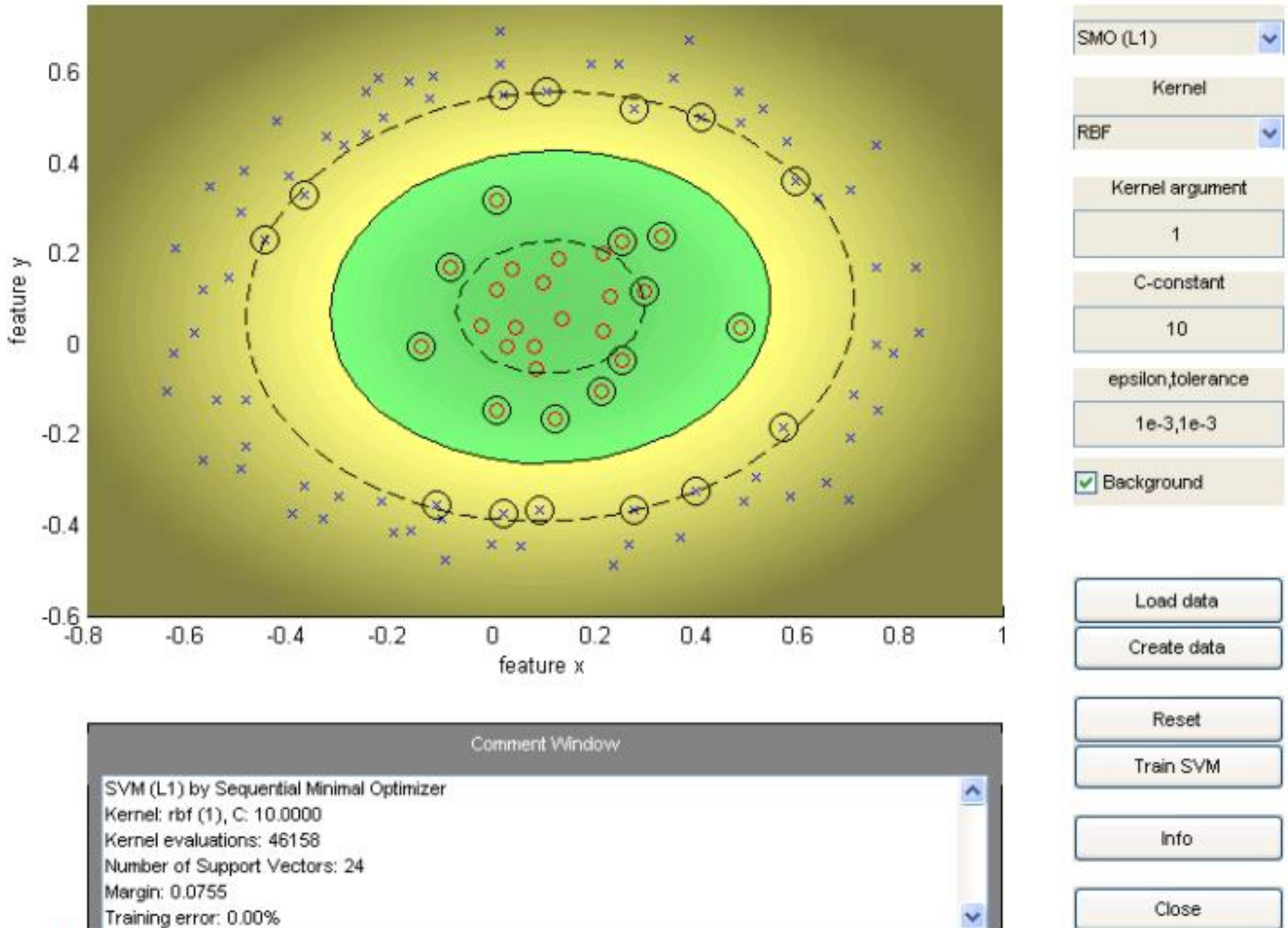
$$f(\mathbf{x}) = \sum_{i}^{N} \alpha_i y_i \exp\left(-||\mathbf{x} - \mathbf{x}_i||^2 / 2\sigma^2\right) + b$$

$\sigma = 1.0 \quad C = 100$



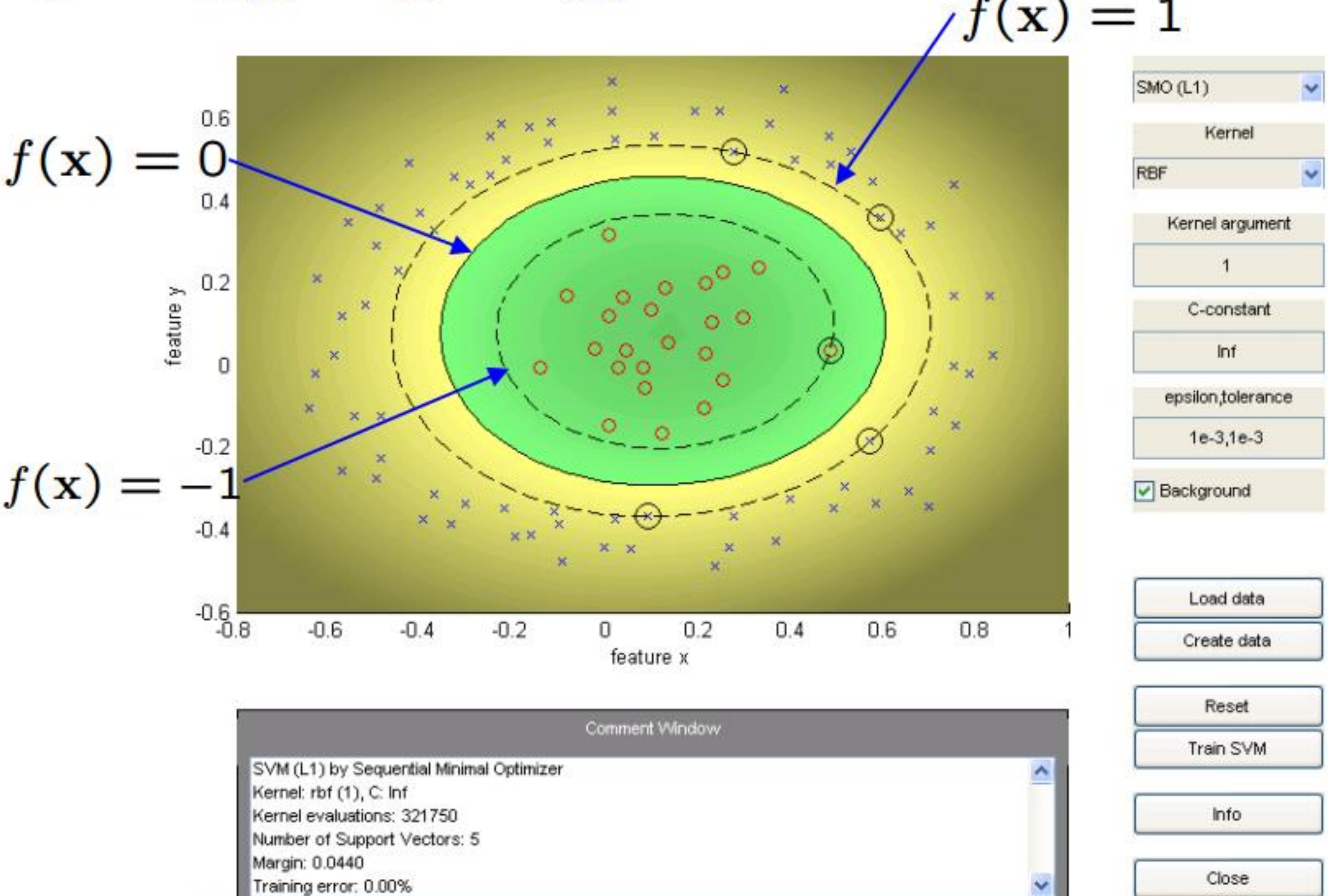
Decrease C, gives wider (soft) margin

$$\sigma = 1.0 \quad C = 10$$



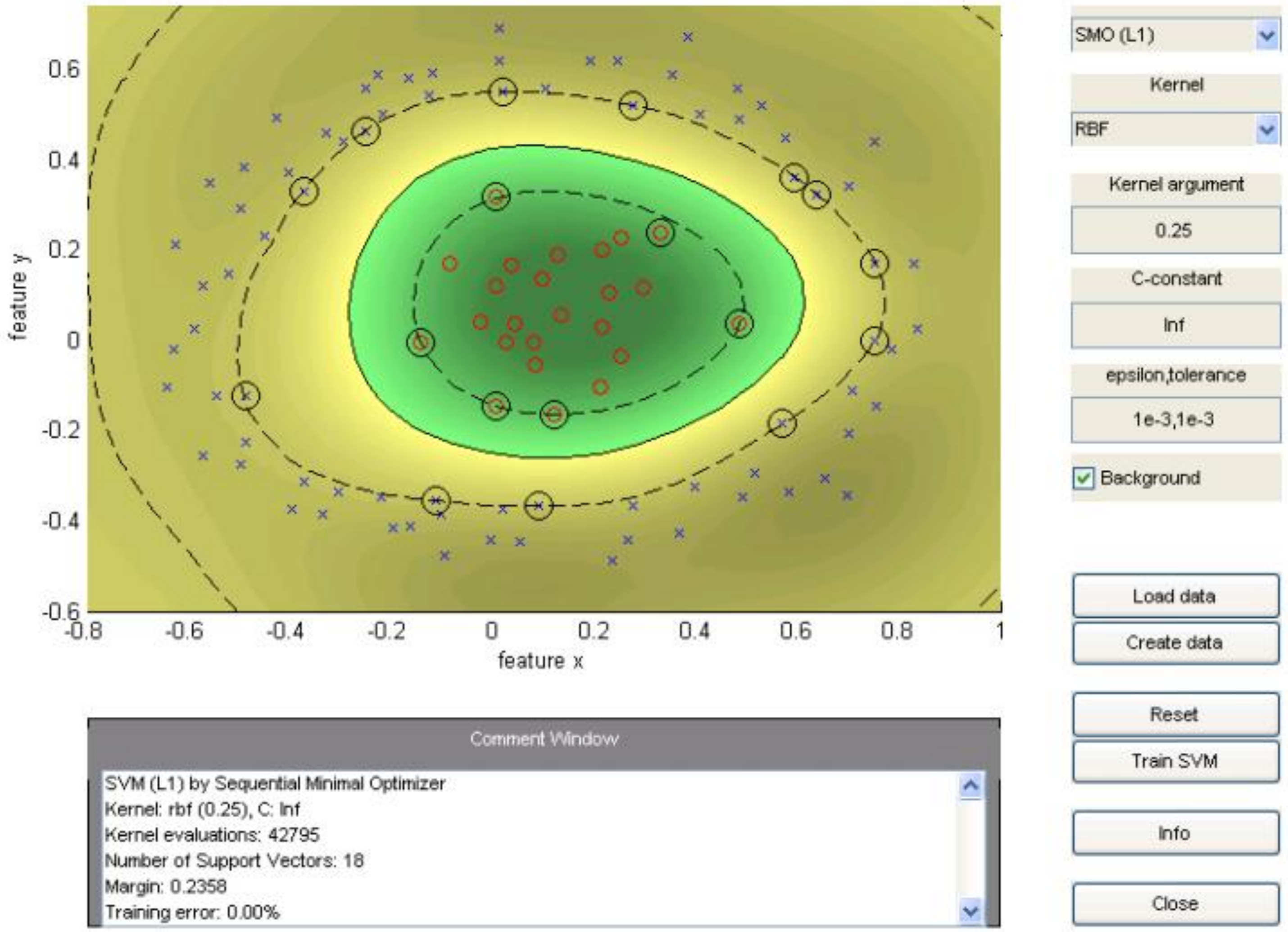
$$f(\mathbf{x}) = \sum_{i}^{N} \alpha_i y_i \exp\left(-||\mathbf{x} - \mathbf{x}_i||^2 / 2\sigma^2\right) + b$$

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



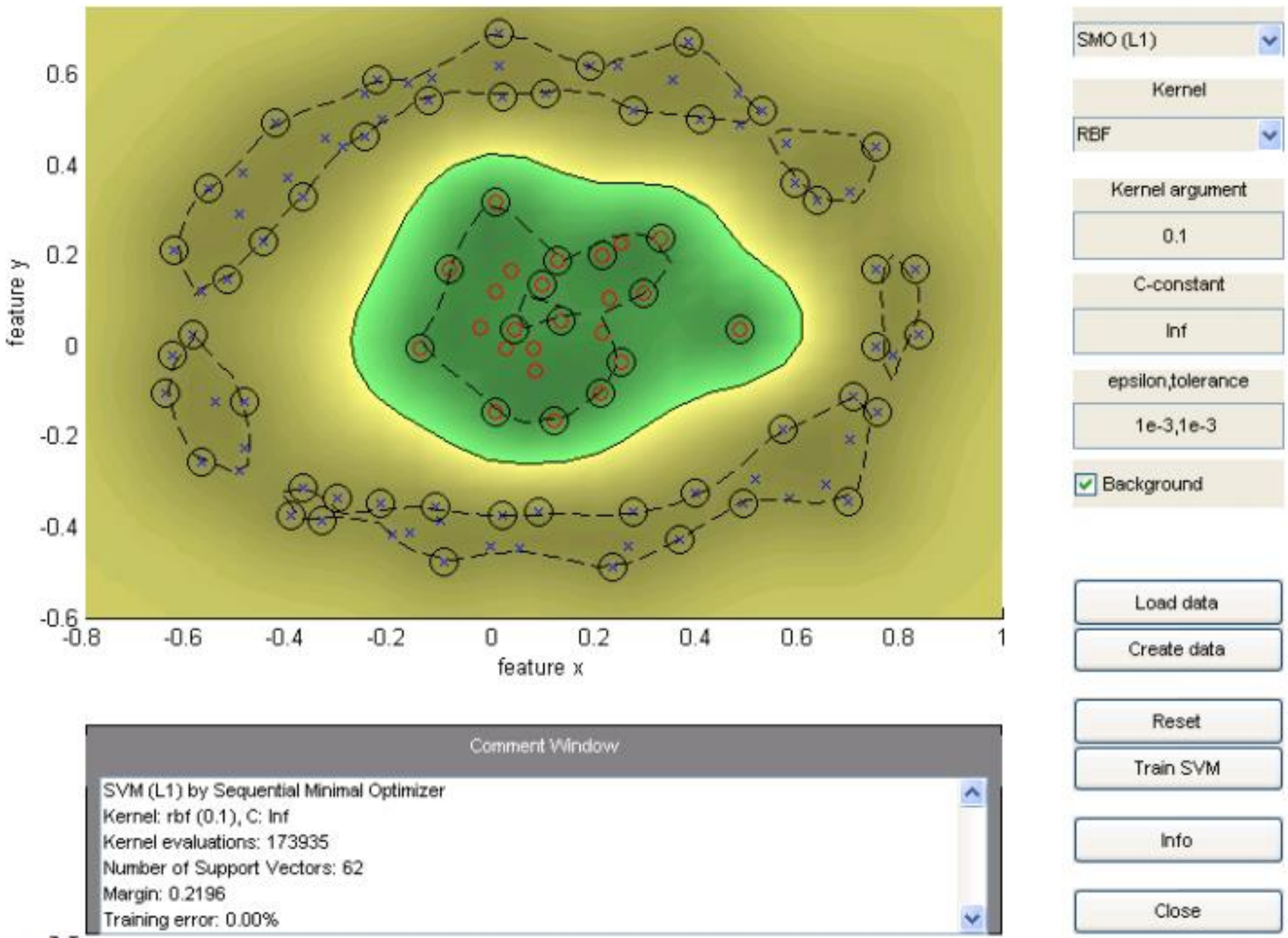
$$f(\mathbf{x}) = \sum_{i}^{N} \alpha_i y_i \exp\left(-||\mathbf{x} - \mathbf{x}_i||^2 / 2\sigma^2\right) + b$$

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



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 SVM EXAMPLE (XOR PROBLEM)

XOR example

- $K(x, y) = (1 + x^T y)^2$
  $= 1 + x_1^2 y_1^2 + 2x_1 x_2 y_1 y_2 + x_2^2 y_2^2 + 2x_1 y_1 + 2x_2 y_2$
  $= \phi^T(x)\phi(y)$

- $\phi(x) = \left[1\ x_1^2\ \ \sqrt{2}x_1 x_2\ \ x_2^2\ \ \sqrt{2}x_1\ \ \sqrt{2}x_2\right]^T$

TABLE 6.2 XOR Problem

Input vector, $\mathbf{x}$	Desired response, d
$(-1, -1)$	-1
$(-1, +1)$	$+1$
$(+1, -1)$	$+1$
$(+1, +1)$	-1

$$\mathbf{K} = \{K(\mathbf{x}_i, \mathbf{x}_j)\}_{(i,j)=1}^{N} \longrightarrow \mathbf{K} = \begin{bmatrix} 9 & 1 & 1 & 1 \\ 1 & 9 & 1 & 1 \\ 1 & 1 & 9 & 1 \\ 1 & 1 & 1 & 9 \end{bmatrix}$$

$$Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j d_i d_j K(\mathbf{x}_i, \mathbf{x}_j)$$

$$Q(\alpha) = \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 - \frac{1}{2}(9\alpha_1^2 - 2\alpha_1\alpha_2 - 2\alpha_1\alpha_3 + 2\alpha_1\alpha_4$$

$$+ 9\alpha_2^2 + 2\alpha_2\alpha_3 - 2\alpha_2\alpha_4 + 9\alpha_3^2 - 2\alpha_3\alpha_4 + 9\alpha_4^2)$$

$$9\alpha_1 - \alpha_2 - \alpha_3 + \alpha_4 = 1$$
$$-\alpha_1 + 9\alpha_2 + \alpha_3 - \alpha_4 = 1$$
$$-\alpha_1 + \alpha_2 + 9\alpha_3 - \alpha_4 = 1$$
$$\alpha_1 - \alpha_2 - \alpha_3 + 9\alpha_4 = 1$$

$$\mathbf{w}_o = \frac{1}{8}\left[-\varphi(\mathbf{x}_1) + \varphi(\mathbf{x}_2) + \varphi(\mathbf{x}_3) - \varphi(\mathbf{x}_4)\right]$$

$$= \frac{1}{8}\left[-\begin{bmatrix} 1 \\ 1 \\ \sqrt{2} \\ 1 \\ -\sqrt{2} \\ -\sqrt{2} \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ -\sqrt{2} \\ 1 \\ -\sqrt{2} \\ \sqrt{2} \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ -\sqrt{2} \\ 1 \\ \sqrt{2} \\ -\sqrt{2} \end{bmatrix} - \begin{bmatrix} 1 \\ 1 \\ \sqrt{2} \\ 1 \\ \sqrt{2} \\ \sqrt{2} \end{bmatrix}\right] = \begin{bmatrix} 0 \\ 0 \\ -1/\sqrt{2} \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Optimal hyperplane (XOR)

$$\mathbf{w}_o^T \boldsymbol{\varphi}(\mathbf{x}) = 0$$

$$\Rightarrow \quad \left[0, 0, \frac{-1}{\sqrt{2}}, 0, 0, 0\right] \begin{bmatrix} 1 \\ x_1^2 \\ \sqrt{2}\,x_1 x_2 \\ x_2^2 \\ \sqrt{2}\,x_1 \\ \sqrt{2}\,x_2 \end{bmatrix} = 0$$

MULTICLASS SVM

Multiclass SVMs

One-versus-the-rest approach: trains K separate SVMs, in which the k-th model $y_k(\mathbf{x})$ is trained using the data from class $\mathcal{C}_k$ as the positive examples and the data from the remaining $K-1$ classes as the negative examples.

The prediction for new input $\mathbf{x}$ is by

$$y(\mathbf{x}) = \max_k y_k(\mathbf{x}).$$

Problems: 1) the output values $y_k(\mathbf{x})$ for different classifiers have no appropriate scales. 2) the training sets are imbalanced.

Multiclass SVMs

One-versus-one approach: is to train $K(K-1)/2$ different 2-class SVMs on all possible pairs of classes, and then to classify test points according to which class has the highest number of 'votes'.

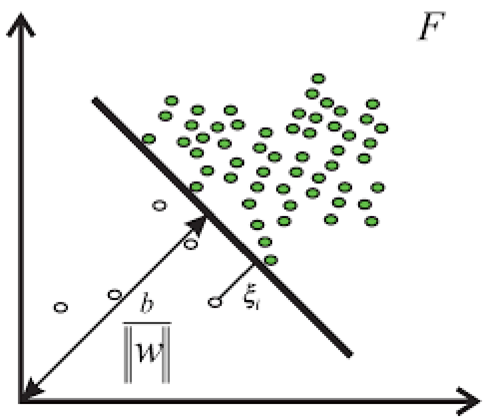
Problems: it requires more training time and evaluation time.

ONE SVM AND SVDD

One-class SVM

- To maximize the distance from the hyperplane to the origin

$$distance = \frac{\rho}{\|w\|}$$



$$w^\top x = \rho$$

$$\min_{w,\,\xi_i,\,\rho} \frac{1}{2}\|w\|^2 + \frac{1}{\nu n}\sum_{i=1}^{n}\xi_i - \rho$$

subject to:

$$(w \cdot \phi(x_i)) \geq \rho - \xi_i \qquad \text{for all } i = 1, \ldots, n$$

$$\xi_i \geq 0 \qquad \text{for all } i = 1, \ldots, n$$

$$f(x) = \operatorname{sgn}((w \cdot \phi(x_i)) - \rho) = \operatorname{sgn}(\sum_{i=1}^{n} \alpha_i K(x, x_i) - \rho)$$

Dual form of One-Class SVM

$$\max_{0 \le \alpha_i \le \frac{1}{\nu l}} \left(- \sum_i \sum_j \alpha_i \alpha_j (x_i^\top x_j) \right)$$

$$\Big\downarrow \; \text{Kernel trick}$$

$$\max_{0 \le \alpha_i \le \frac{1}{\nu l}} \left(- \sum_i \sum_j \alpha_i \alpha_j k(x_i, x_j) \right)$$

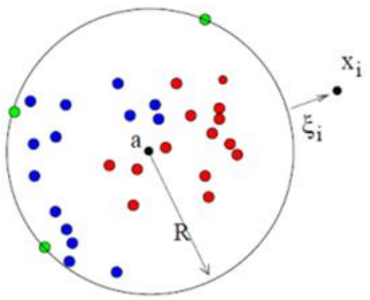
SVDD

- Support vector data description
  - A method to find the boundary around a data set

$$\min_{R,a,\xi_i \geq 0} \left(R^2 + C \sum_i \xi_i \right)$$

$$s.t. \quad \|x_i - a\|^2 \leq R + \xi_i$$

$$\xi_i \geq 0$$



Dual form of SVDD

$$\max_{0 \le \alpha_i \le C} \left(\sum \alpha_i (x_i^\top x_i) - \sum_i \sum_j \alpha_i \alpha_j (x_i^\top x_j) \right)$$

Kernel trick

$$\max_{0 \le \alpha_i \le C} \left(\sum \alpha_i k(x_i, x_i) - \sum_i \sum_j \alpha_i \alpha_j k(x_i, x_j) \right)$$

SUMMARY

SVM parameter selection

- The effectiveness of SVM depends on the selection of kernel, the kernel's parameters, and soft margin parameter C.
- Typically, each combination of parameter choices is checked using cross validation, and the parameters with best cross-validation accuracy are picked.
- The final model, which is used for testing and for classifying new data, is then trained on the whole training set using the selected parameters.

Choosing the Kernel Function

- Probably the most tricky part of using SVM.
- The kernel function is important because it creates the kernel matrix, which summarizes all the data
  - Many principles have been proposed (diffusion kernel, Fisher kernel, string kernel, …)
  - In practice, a low degree polynomial kernel or RBF kernel with a reasonable width is a good initial try

Software

- A list of SVM implementation can be found at
- http://www.kernel-machines.org/software

- Some implementation (such as LIBSVM) can handle multi-class classification
- SVMLight is among one of the earliest implementation of SVM
- Several Matlab toolboxes for SVM are also available

Summary: Steps for Classification

- Select the kernel function to use
- Select the parameter of the kernel function and the value of C
  - You can use the values suggested by the SVM software, or you can set apart a validation set to determine the values of the parameter
- Execute the training algorithm and obtain the α_i
- Unseen data can be classified using the α_i and the support vectors

$$f(x) = \sum \alpha_i \, y_i k(x_i, x) + b$$

Strengths and Weaknesses of SVM

- Strengths
  - Training is relatively easy
    - No local optimal, unlike in neural networks
    - It scales relatively well to high dimensional data
    - Tradeoff between classifier complexity and error can be controlled explicitly
- Weaknesses
  - Need to choose a “good” kernel function.

Conclusions

- SVM is a useful alternative to neural networks
- Two key concepts of SVM:
  - maximize the margin and the kernel trick
- Many SVM implementations are available on the web for you to try on your data set!