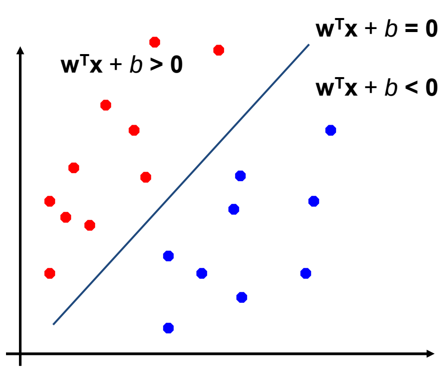
Classification with SVM

nanopoulos@ismll.de

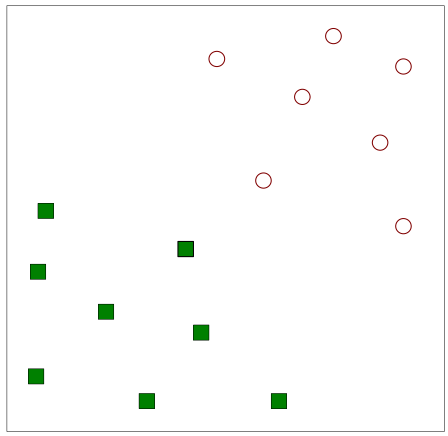
Perceptron Revisited

Binary classification can be viewed as the task of separating classes in feature space:



$y(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x} + b)$

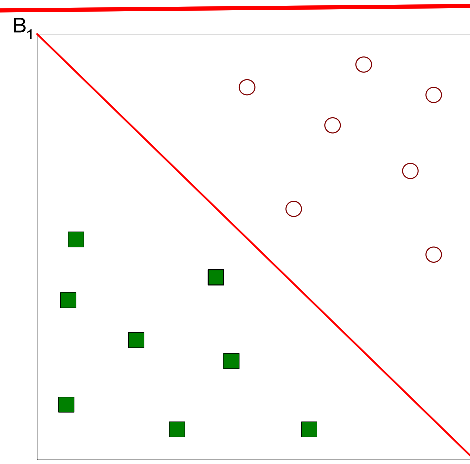
Linear Classification



- **Find a linear hyperplane (decision boundary) that will separate the data**

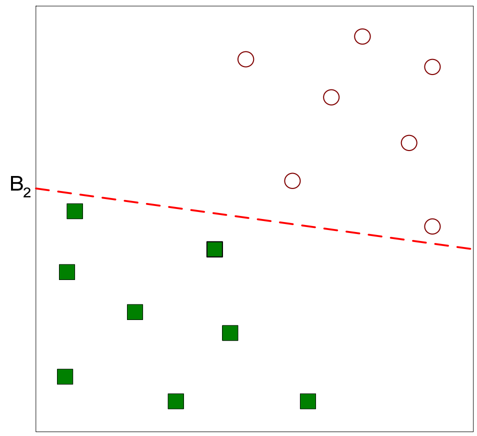
$$y(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b$$

Linear Classification



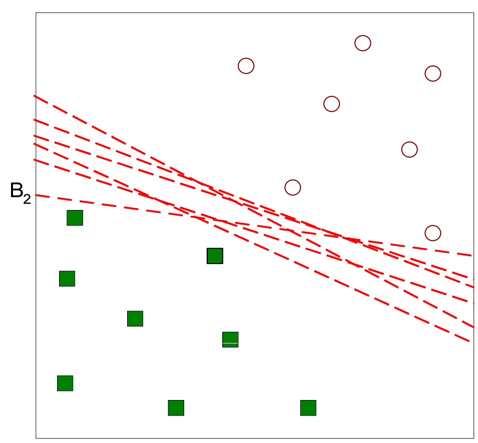
- **One Possible Solution**

Linear Classification



- **Another possible solution**

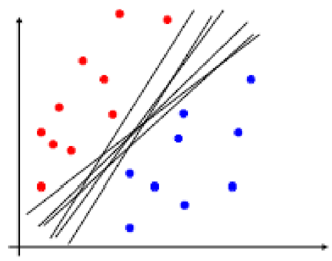
Linear Classification



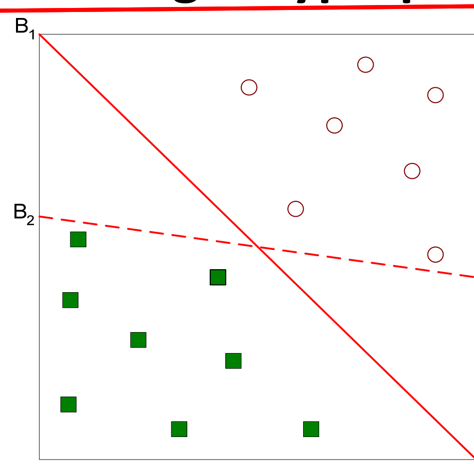
- **Other possible solutions**

(Slide from Perceptron Lecture)

- Which of these linear separators is optimal?

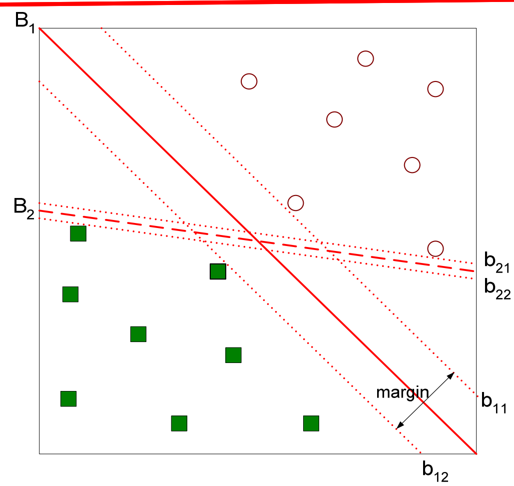


Maximum margin hyperplane



- Which one is better? B1 or B2?
- How do you define better?

Maximum margin hyperplane



- **Find hyperplane maximizes the margin => B1 is better than B2**

Theoretical Justification for Maximum Margins

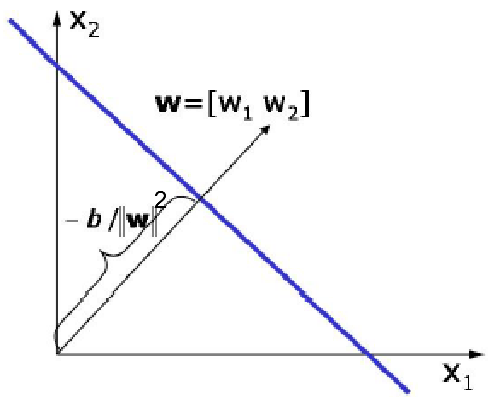
Vapnik has proved the following:

The class of optimal linear separators has VC dimension (complexity measure) h bounded from above as

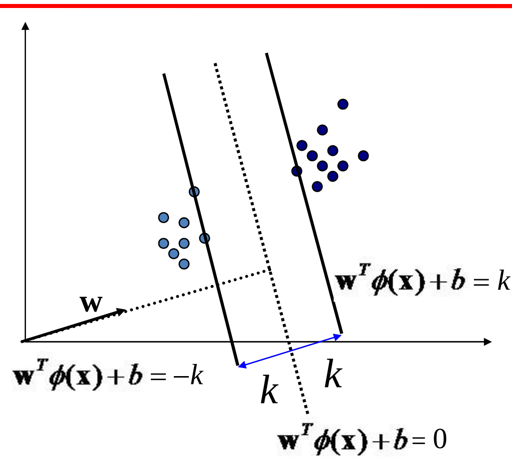
$$h \leq \min\left\{\left\lceil \frac{D^2}{\rho^2} \right\rceil, m_0\right\} + 1$$

where ρ is the margin, D is the diameter of the smallest sphere that can enclose all of the training examples, and m_0 is the dimensionality.

Distance of hyperplane from origin



Computing the size of margin

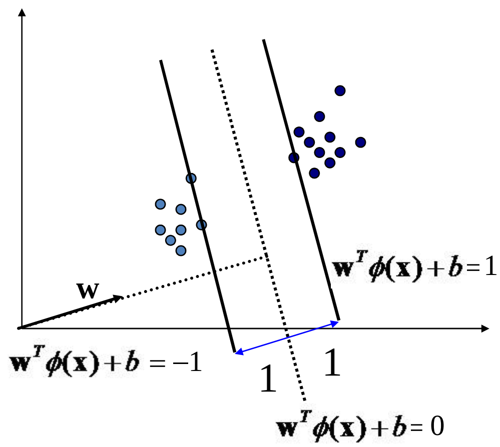


The width of the margin is:

$$\frac{2|k|}{\|w\|^2}$$

12

Setting Up the Optimization Problem



There is a scale and unit for data so that *k=1*. Then problem becomes:

$$\max \frac{2}{\|w\|^2}$$

The Optimization Problem

We want to maximize: $\text{Margin} = \dfrac{2}{\|w\|^2}$

Which is equivalent to minimizing: $L(w) = \dfrac{\|w\|^2}{2}$

But subjected to the following constraints:

$$\begin{cases} class\ 1: & y(x_n) = w^T\phi(x_n) + b \geq 1 \\ class\ 2: & y(x_n) = w^T\phi(x_n) + b \leq -1 \end{cases}$$

Restating the Optimization Problem

t_n = 1 for class 1 and t_n = -1 for class 2

For all data points: $t_n \, y(\boldsymbol{x}_n) \geq 1$

The optimization problem becomes:

$$\arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{subject to} \quad t_n(\mathbf{w}^T\phi(\mathbf{x}_n)+b) \geq 1, \; n=1,\ldots,N$$

15

Solution with Lagrange multipliers

$$L(\mathbf{w},b,\mathbf{a}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{n=1}^{N} a_n \left\{t_n(\mathbf{w}^T\phi(\mathbf{x}_n)+b)-1\right\}$$

Subject to $a_n \geq 0$ and

$$\sum_{n=1}^{N} a_n \{t_n y(\mathbf{x}_n) - 1\} = 0$$

Karush-Kuhn-Tucker conditions

$$\frac{\partial L}{\partial w} = 0 \Rightarrow w = \sum_{n=1}^{N} a_n t_n \phi(x_n)$$

16

Dual representation

$$\tilde{L}(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m)$$

subject to $a_n \geq 0, \; n = 1,\ldots,N$

$$\sum_{n=1}^{N} a_n \{t_n y(\mathbf{x}_n) - 1\} = 0$$

where $k(\mathbf{x}_n, \mathbf{x}_m) = \phi(\mathbf{x}_n)\phi(\mathbf{x}_m)$

Solve with quadratic programming in $O(N^3)$

Only function of Lagrange multipliers

The dual representation is for maximization

17

Classifying New Data

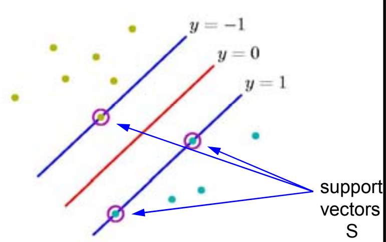
$$y(\mathbf{x}) = \mathbf{w}^T\phi(\mathbf{x}) + b \;\Rightarrow\; y(\mathbf{x}) = \sum_{n=1}^{N} a_n t_n k(\mathbf{x}, \mathbf{x}_n) + b$$

$$a_n \geq 0$$

$$t_n y(\mathbf{x}_n) - 1 \geq 0$$

$$a_n\{t_n y(\mathbf{x}_n) - 1\} = 0$$

$a_n = 0$ or $t_n y(\mathbf{x}_n) = 1$

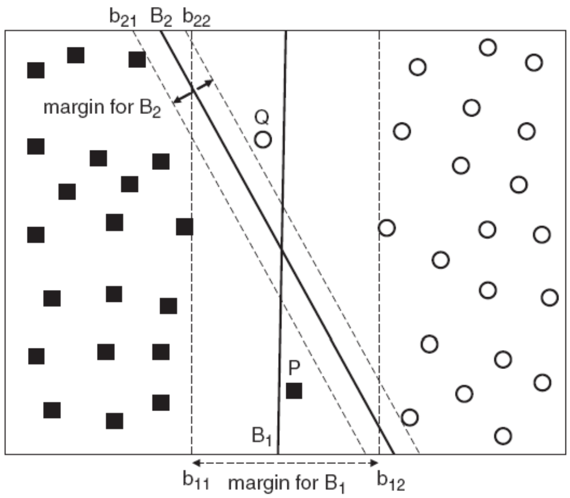


$$b = \frac{1}{N_S}\sum_{n\in S}\left(t_n - \sum_{m\in S} a_m t_m k(\mathbf{x}_n, \mathbf{x}_m)\right)$$

Average for all S (more stable)

18

Overlapping class distributions

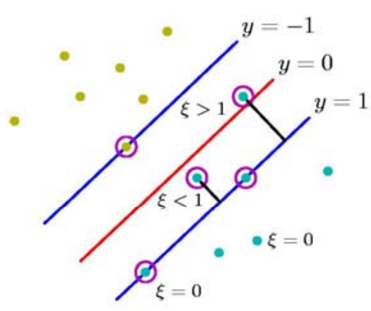


Tradeoff: Allow training errors to increase margin

Soft margin

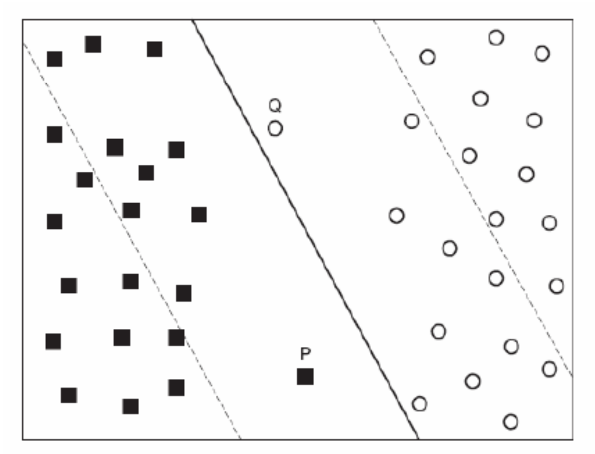
Allow some misclassified examples

Introduce slack variables $\xi_n \geq 0,\ n = 1,\ldots,N$



$$t_n y(\mathbf{x}_n) = t_n(\mathbf{w}^T\phi(\mathbf{x}_n) + b) \geq 1 \;\; \Rightarrow \;\; t_n y(\mathbf{x}_n) \geq 1 - \xi_n$$

Need to control slack variables



Soft Margin Solution

Minimize $C\sum_{n=1}^{N}\xi_n + \frac{1}{2}\|\mathbf{w}\|^2$

$C > 0$: constraints training errors

$$L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{n=1}^{N}\xi_n - \sum_{n=1}^{N} a_n\{t_n y(\mathbf{x}_n) - 1 + \xi_n\} - \sum_{n=1}^{N}\mu_n\xi_n$$

KKT conditions:

$$a_n \geq 0$$
$$t_n y(\mathbf{x}_n) - 1 + \xi_n \geq 0$$
$$a_n\left(t_n y(\mathbf{x}_n) - 1 + \xi_n\right) = 0$$
$$\mu_n \geq 0$$
$$\xi_n \geq 0$$
$$\mu_n\xi_n = 0$$

$a_n = 0$

or

$t_n y(\mathbf{x}_n) = 1 - \xi_n$

: support vectors

Dual representation

$$\tilde{L}(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m)$$

subject to $0 \le a_n \le C, \; n = 1,\ldots,N$ → This constraint stems from setting dev of L by slack equal to 0

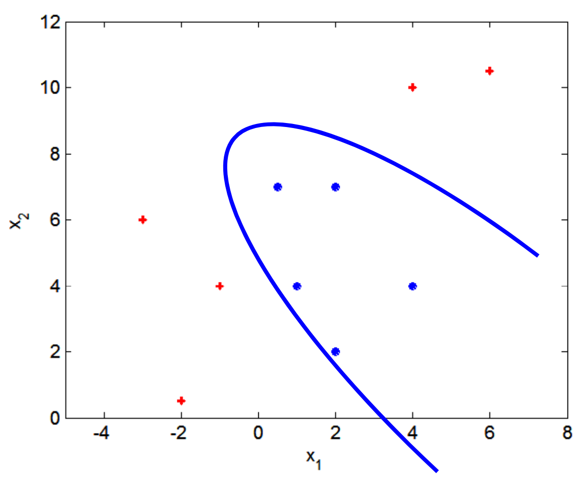
$$\sum_{n=1}^{N} a_n t_n = 0$$

Same equation for dual but different constraints for Lagrangian multipliers

23

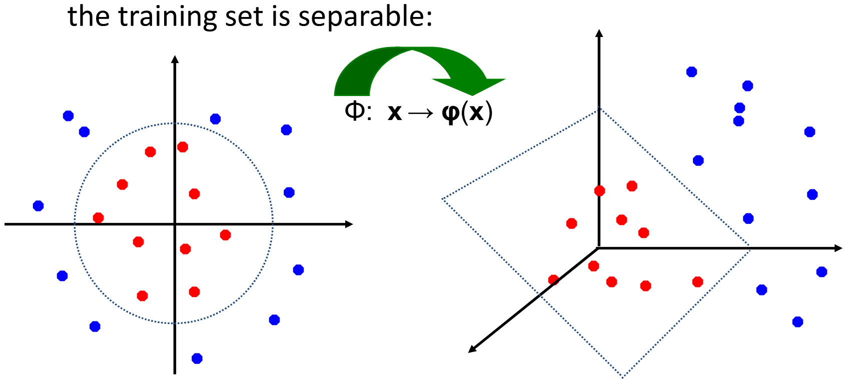
Nonlinear Support Vector Machines

What if decision boundary is not linear?



Non-linear SVMs: Feature spaces

General idea: the original feature space can always be mapped to some higher-dimensional feature space where the training set is separable:



The "Kernel Trick"

The SVM only relies on the inner-product between vectors $\phi(\mathbf{x}_n)\cdot\phi(\mathbf{x}_m)$

$$\tilde{L}(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m) \qquad y(\mathbf{x}) = \sum_{n=1}^{N} a_n t_n k(\mathbf{x}, \mathbf{x}_n) + b$$

If every datapoint is mapped into high-dimensional space via some transformation Φ: $\mathbf{x} \rightarrow \phi(\mathbf{x})$, the inner-product becomes: $k(\mathbf{x}_m,\mathbf{x}_n) = \phi(\mathbf{x}_m)\cdot\phi(\mathbf{x}_n)$

$k(\mathbf{x}_m,\mathbf{x}_n)$ is called the kernel function.

For SVM, we only need specify the kernel **without** need to know the corresponding non-linear mapping, $\phi(\mathbf{x})$.

Examples of Kernel Trick (1)

- **For the example in the previous figure:**
  - The non-linear mapping

$$x \rightarrow \varphi(x) = (x, x^2)$$

  - The kernel

$$\varphi(x_i) = (x_i, x_i^{\,2}), \quad \varphi(x_j) = (x_j, x_j^2)$$
$$K(x_i, x_j) = \varphi(x_i) \cdot \varphi(x_j)$$
$$= x_i x_j (1 + x_i x_j)$$

- **Where is the benefit?**

Examples of Kernel Trick (2)

- **Polynomial kernel of degree 2 in 2 variables**
  - The non-linear mapping:

$$\mathbf{x} = (x_1, x_2)$$
$$\varphi(\mathbf{x}) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

  - The kernel

$$\varphi(\mathbf{x}) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)$$
$$\varphi(\mathbf{y}) = (1, \sqrt{2}y_1, \sqrt{2}y_2, y_1^2, y_2^2, \sqrt{2}y_1y_2)$$
$$K(\mathbf{x}, \mathbf{y}) = \varphi(\mathbf{x}) \cdot \varphi(\mathbf{y})$$
$$= (1 + \mathbf{x} \cdot \mathbf{y})^2$$

Examples of Kernel Functions

- **Linear kernel:** $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j$
- **Polynomial kernel of power *p*:** $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i \cdot \mathbf{x}_j)^p$
- **Gaussian kernel:** $K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma^2}$
  - In the form, equivalent to RBFNN, but has the advantage of that the center of basis functions, i.e., support vectors, are optimized in a supervised.
- **Two-layer perceptron:** $K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\alpha \mathbf{x}_i \cdot \mathbf{x}_j + \beta)$

What Functions are Kernels?

For some functions $K(\mathbf{x}_i,\mathbf{x}_j)$ checking that $K(\mathbf{x}_i,\mathbf{x}_j)= \phi(\mathbf{x}_i)\, \phi(\mathbf{x}_j)$ can be cumbersome.

Mercer's theorem:

Every semi-positive definite symmetric function is a kernel

Semi-positive definite symmetric functions correspond to a semi-positive definite symmetric Gram matrix:

K=

$K(\mathbf{x}_1,\mathbf{x}_1)$	$K(\mathbf{x}_1,\mathbf{x}_2)$	$K(\mathbf{x}_1,\mathbf{x}_3)$	…	$K(\mathbf{x}_1,\mathbf{x}_n)$
$K(\mathbf{x}_2,\mathbf{x}_1)$	$K(\mathbf{x}_2,\mathbf{x}_2)$	$K(\mathbf{x}_2,\mathbf{x}_3)$		$K(\mathbf{x}_2,\mathbf{x}_n)$
…	…	…	…	…
$K(\mathbf{x}_n,\mathbf{x}_1)$	$K(\mathbf{x}_n,\mathbf{x}_2)$	$K(\mathbf{x}_n,\mathbf{x}_3)$	…	$K(\mathbf{x}_n,\mathbf{x}_n)$

http://en.wikipedia.org/wiki/Positive-definite_matrix