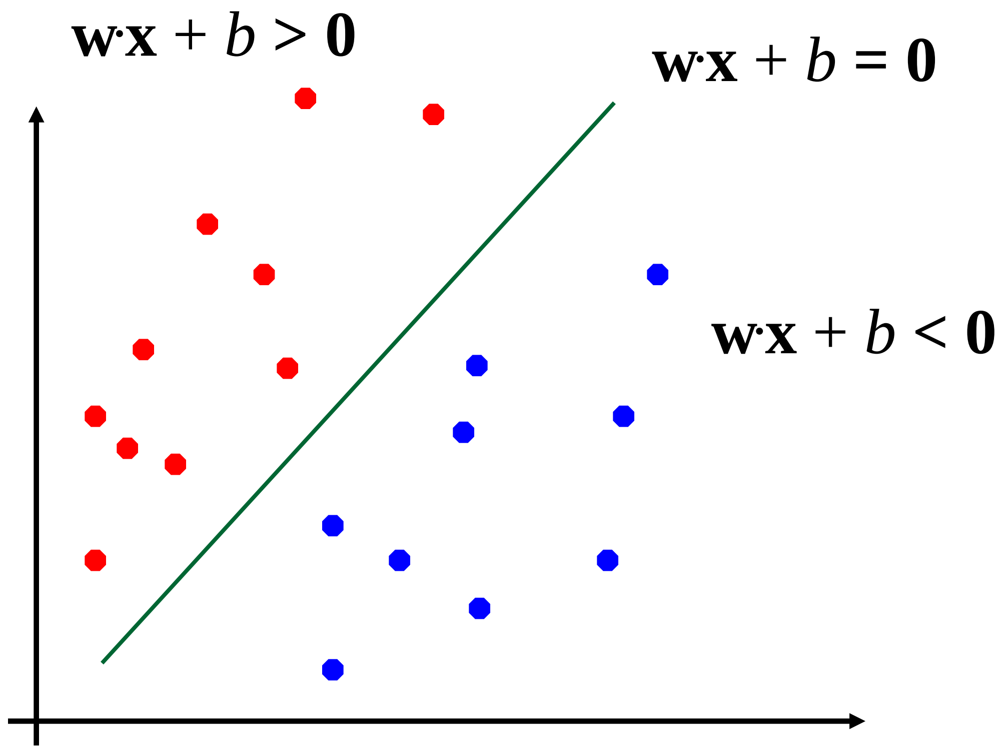
OI-OPPA. European Social Fund
Prague & EU: We invests in your future.

Support Vector Machines

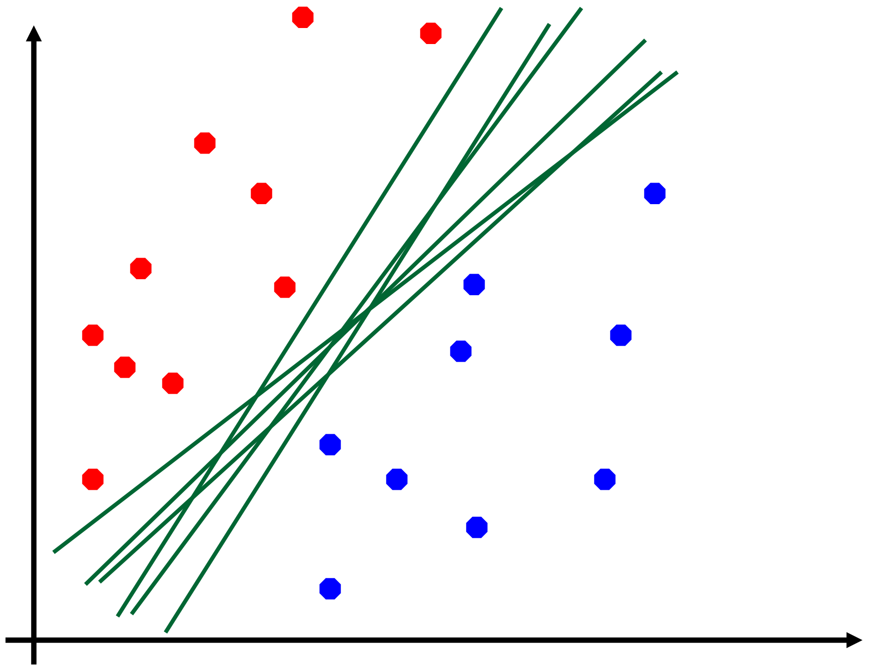
Perceptron Revisited:

- Linear Classifier: $y(\mathbf{x}) = \text{sign}(\mathbf{w}\cdot\mathbf{x} + b)$



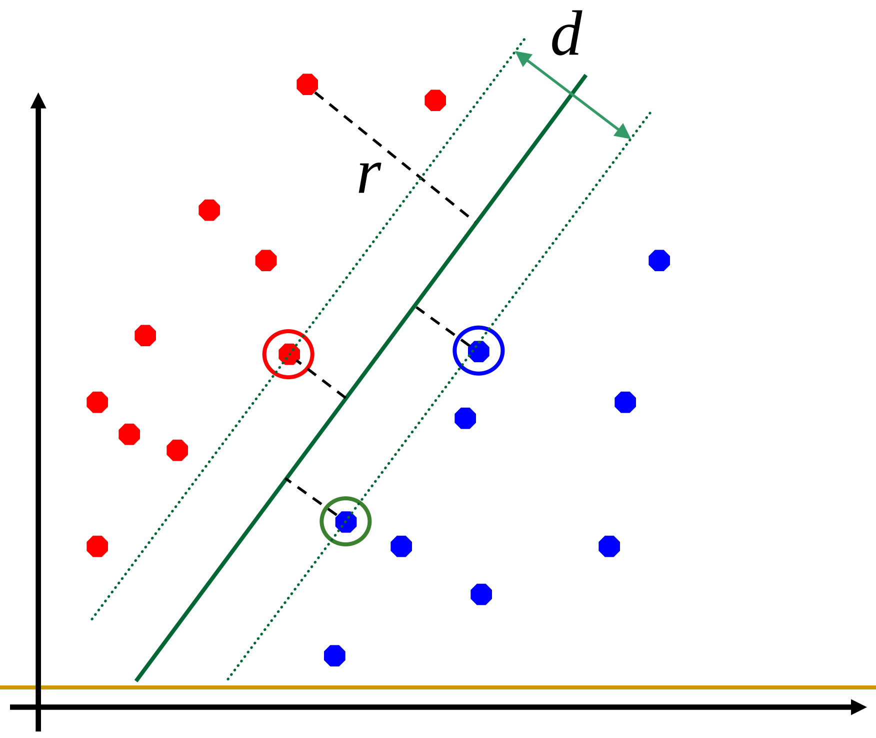
$\mathbf{w}\cdot\mathbf{x} + b > \mathbf{0}$
$\mathbf{w}\cdot\mathbf{x} + b = \mathbf{0}$
$\mathbf{w}\cdot\mathbf{x} + b < \mathbf{0}$

Which one is the best?



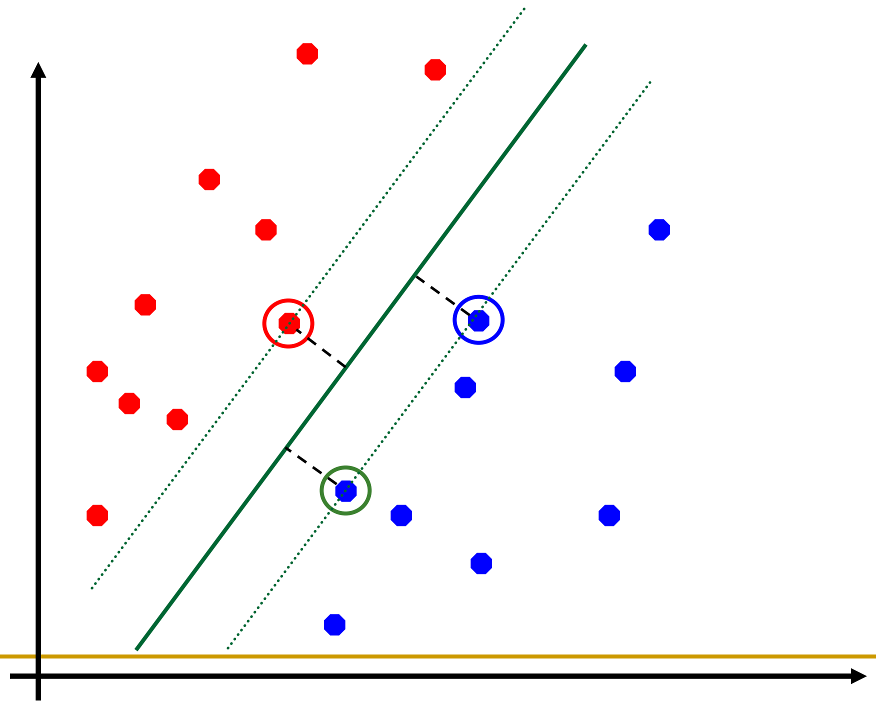
Notion of Margin

- Distance from a data point to the boundary: $r = \frac{|\mathbf{w} \cdot \mathbf{x} + b|}{\|\mathbf{w}\|}$
- Data points closest to the boundary are called support vectors
- ***Margin*** *d* is the distance between two classes.



Maximum Margin Classification

- Intuitively, the classifier of the maximum margin is the best solution
- Vapnik formally justifies this from the view of Structure Risk Minimization
- Also, it seems that only support vectors matter (is SVM a statistical classifier?)



Quantifying the Margin:

- Canonical hyper-planes:
  - Redundancy in the choice of **w** and b:

$$y(\mathbf{x}) = sign(\mathbf{w}\cdot\mathbf{x}+b)$$
$$= sign(k\mathbf{w}\cdot\mathbf{x}+k\cdot b)$$

  - To break this redundancy, assuming the closest data points are on the hyper-planes (canonical hyper-planes):

$$w\cdot x+b=\pm 1$$

- The margin is:

$$d=\frac{2}{\|\mathbf{w}\|}$$

- The condition of correct classification

$$\mathbf{w}\cdot\mathbf{x_i}+b\geq 1 \quad \text{if } y_i=1$$
$$\mathbf{w}\cdot\mathbf{x_i}+b\leq -1 \quad \text{if } y_i=-1$$

Maximizing Margin:

- The *quadratic optimization problem:*

Find **w** and *b* such that

$d = \frac{2}{\|\mathbf{w}\|}$ is maximized; and for all $\{(\mathbf{x_i}, y_i)\}$

$\mathbf{w} \cdot \mathbf{x_i} + b \geq 1$ if $y_i = 1$; $\mathbf{w} \cdot \mathbf{x_i} + b \leq -1$ if $y_i = -1$

- A simpler formulation:

Mimimizing $\frac{1}{2} \|\mathbf{w}\|^2$

Subject to : $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1$, for $i = 1, \ldots, N$

The dual problem (1)

- Quadratic optimization problems are a well-known class of mathematical programming problems, and many (rather intricate) algorithms exist for solving them.
- The solution involves constructing a *dual problem:*
  - The Lagrangian L:

$$L(\mathbf{w}, b; \mathbf{h}) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^{N} h_i [y_i (\mathbf{w} \cdot \mathbf{x}_i + b) - 1]$$

$$where\ \mathbf{h} = (h_1, \ldots, h_N) \text{ is the vector of non-negative Lagrange multipliers}$$

  - Minimizing L over **w** and b:

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{N} h_i y_i \mathbf{x}_i = 0$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{N} h_i y_i = 0$$

The dual problem (2)

- Therefore, the optimal value of **w** is:

$$\mathbf{w}^* = \sum_{i=1}^{N} h_i y_i \mathbf{x}_i$$

- Using the above result we have:

$$L(\mathbf{h}) = \sum_{i=1}^{N} h_i - \frac{1}{2} \| \mathbf{w}^* \|^2$$

$$= \sum_{i=1}^{N} h_i - \frac{1}{2} \mathbf{h} \cdot \mathbf{D} \cdot \mathbf{h}$$

$$where\ \mathbf{D} = y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$$

- The dual optimization problem

$$\text{Maximizing}: \ L(\mathbf{h}) = \sum_{i=1}^{N} h_i - \frac{1}{2} \mathbf{h} \cdot \mathbf{D} \cdot \mathbf{h}$$

$$\text{Subject to}: \mathbf{h} \cdot \mathbf{y} = 0$$

$$\mathbf{h} \geq 0$$

Important Observations (1):

- The solution of the dual problem depends on the *inner-product* between data points, i.e., $\mathbf{x}_i \cdot \mathbf{x}_j$ rather than data points themselves.

- The dominant contribution of support vectors:
  - The Kuhn-Tucker condition

At the solution, $(\mathbf{w}^*, \mathrm{b}^*, \mathbf{h})$, the follwoing relationships hold

$$h_i[y_i(\mathbf{w}^* \cdot \mathbf{x}_i + \mathrm{b}^*) - 1] = 0, \text{ for } i = 1, \ldots, N$$

  - Only support vectors have non-zero *h* values

$$y_i(\mathbf{w}^* \cdot \mathbf{x}_i + b^*) = 1, \ h_i > 0$$

Important Observations (2):

- The form of the final solution:

$$\mathbf{w}^* = \sum_{i \in SV} h_i y_i \mathbf{x}_i$$

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

$$= \sum_{i \in SV} h_i y_i \mathbf{x}_i \cdot \mathbf{x} + b^*$$

- Two features:
  - Only depending on support vectors
  - Depending on the inner-product of data vectors

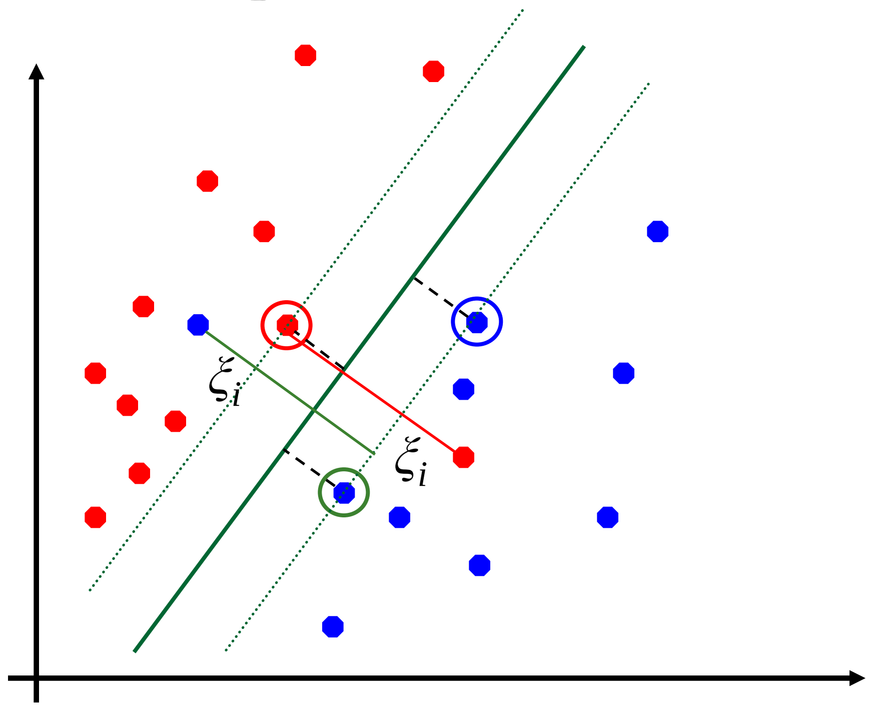
- Fixing b:

Choose any support vector, $\mathbf{x}_k$,

$$b^* = y_k - \mathbf{w}^* \cdot \mathbf{x}_k$$

Soft Margin Classification

- What if data points are not linearly separable?
- *Slack variables* ξ_i can be added to allow misclassification of difficult or noisy examples.



The formulation of soft margin

- The original problem:

$$\text{Mimimizing} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i$$

$$\text{Subject to}$$

$$y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 - \xi_i, \ \text{for } i = 1,\dots,N$$

$$\xi_i \geq 0, \qquad \text{for } i = 1,\dots,N$$

- The dual problem:

$$\text{Maximizing}: \ L(\mathbf{h}) = \sum_{i=1}^{N} h_i - \frac{1}{2}\mathbf{h}\cdot\mathbf{D}\cdot\mathbf{h}$$

$$\text{Subject to}: \mathbf{h}\cdot\mathbf{y} = 0$$

$$0 \leq \mathbf{h} \leq \mathrm{C}$$

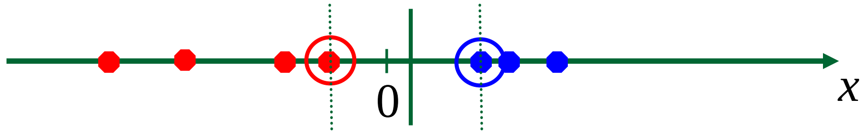
$$\text{where } D_{ij} = y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$$

Linear SVMs: Overview

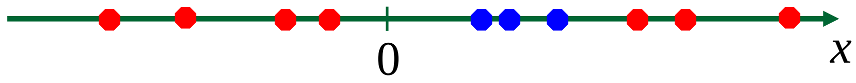
- The classifier is a *separating hyperplane.*
- Most "important" training points are support vectors; they define the hyperplane.
- Quadratic optimization algorithms can identify which training points $\mathbf{x}_i$ are support vectors with non-zero Lagrangian multipliers h_i.
- Both in the dual formulation of the problem and in the solution training points appear only inside inner-products.

Who really need linear classifiers

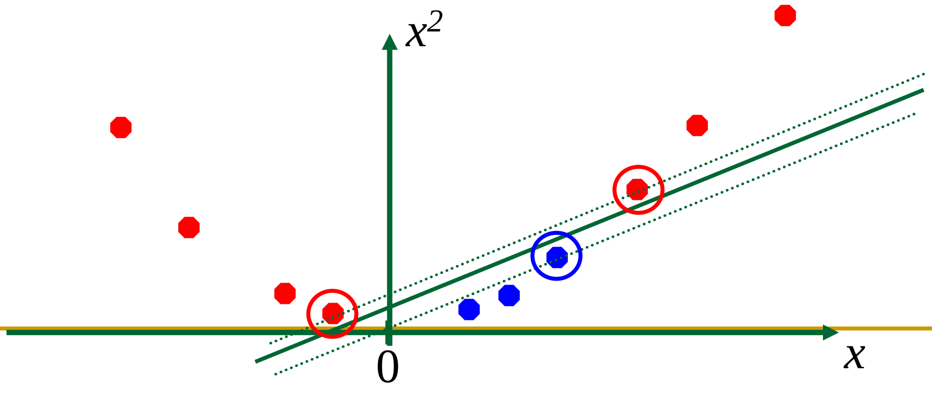
- Datasets that are linearly separable with some noise, linear SVM work well:



- But if the dataset is non-linearly separable?

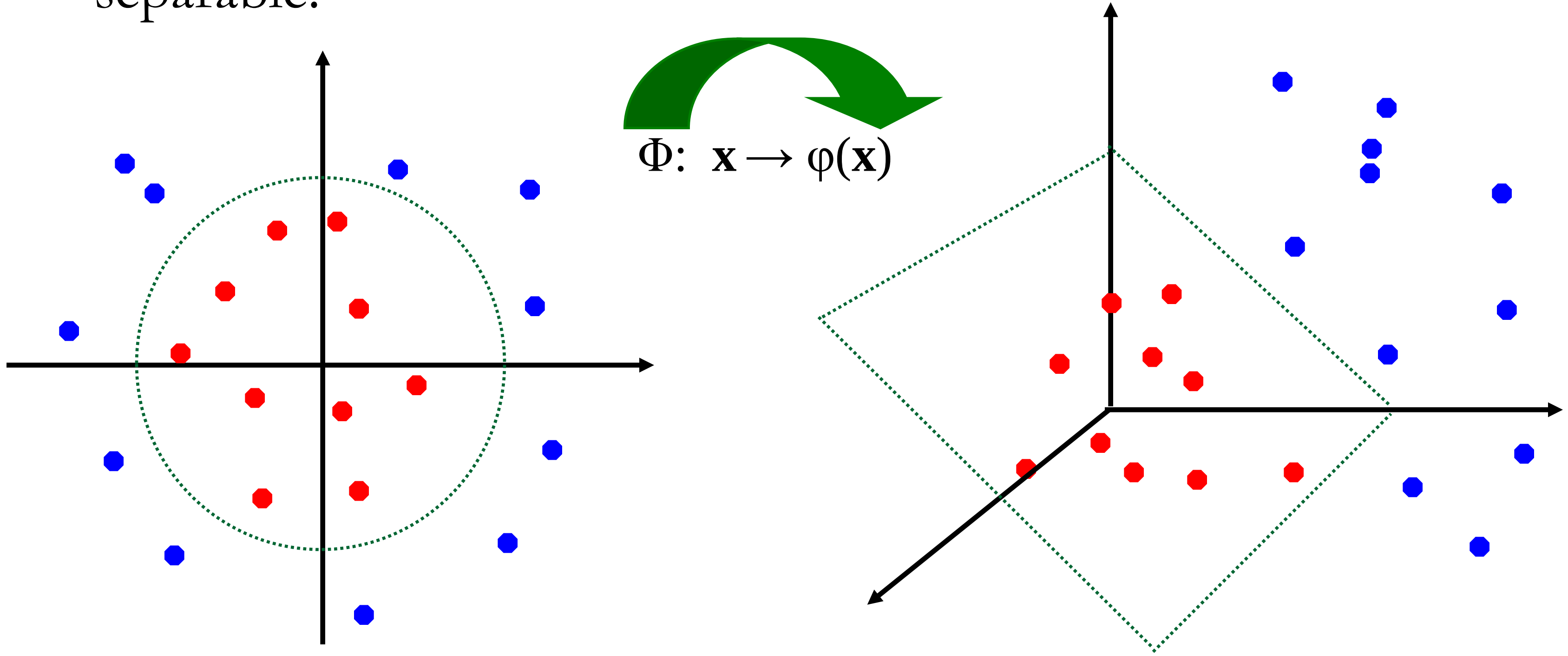


- How about… mapping data to a higher-dimensional space:



Non-linear SVMs: Feature spaces

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



The "Kernel Trick"

- The SVM only relies on the inner-product between vectors $\mathbf{x}_i \cdot \mathbf{x}_j$

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

$$K(\mathbf{x}_i,\mathbf{x}_j) = \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j)$$

- $K(\mathbf{x}_i,\mathbf{x}_j)$ is called the kernel function.
- For SVM, we only need specify the kernel $K(\mathbf{x}_i,\mathbf{x}_j)$, without need to know the corresponding non-linear mapping, $\varphi(\mathbf{x})$.

Non-linear SVMs

- The dual problem:

$$\text{Maximizing}: \; L(\mathbf{h}) = \sum_{i=1}^{N} h_i - \frac{1}{2}\mathbf{h}\cdot\mathbf{D}\cdot\mathbf{h}$$
$$\text{Subject to}: \mathbf{h}\cdot\mathbf{y} = 0$$
$$0 \le \mathbf{h} \le C$$
$$\text{where } D_{ij} = y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

- Optimization techniques for finding h_i's remain the same!
- The solution is:

$$\mathbf{w}^* = \sum_{i\in SV} h_i y_i \varphi(\mathbf{x}_i)$$
$$f(\mathbf{x}) = \mathbf{w}^* \cdot \varphi(\mathbf{x}) + b^*$$
$$= \sum_{i\in SV} h_i y_i K(\mathbf{x}_i, \mathbf{x}) + b^*$$

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

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

Examples of kernel trick (3)

- Gaussian kernel: $K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma^2}$

  - The mapping is of infinite dimension:

$$\varphi(\mathbf{x}) = (\ldots, \varphi_\omega(\mathbf{x}), \ldots), \quad \mathit{for} \;\; \omega \in R^d$$
$$\varphi_\omega(\mathbf{x}) = Ae^{-B\omega^2} e^{-i\mathbf{wx}}$$
$$K(\mathbf{x}, \mathbf{y}) = \int \varphi_\omega(\mathbf{x}) \varphi^*{}_\omega(\mathbf{y}) d\omega$$

- The moral: very high-dimensional and complicated non-linear mapping can be achieved by using a simple kernel!

What Functions are Kernels?

- For some functions $K(\mathbf{x}_i,\mathbf{x}_j)$ checking that $K(\mathbf{x}_i,\mathbf{x}_j)= \varphi(\mathbf{x}_i) \cdot\varphi(\mathbf{x}_j)$ can be cumbersome.
- Mercer's theorem:

Every semi-positive definite symmetric function is a kernel

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: $K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\alpha \mathbf{x}_i \cdot \mathbf{x}_j + \beta)$

SVM Overviews

- Main features:
  - By using the kernel trick, data is mapped into a high-dimensional feature space, without introducing much computational effort;
  - Maximizing the margin achieves better generation performance;
  - Soft-margin accommodates noisy data;
  - Not too many parameters need to be tuned.
- Demos(http://svm.dcs.rhbnc.ac.uk/pagesnew/GPat.shtml)

SVM so far

- SVMs were originally proposed by Boser, Guyon and Vapnik in 1992 and gained increasing popularity in late 1990s.
- SVMs are currently among the best performers for many benchmark datasets.
- SVM techniques have been extended to a number of tasks such as regression [Vapnik *et al.* '97].
- Most popular optimization algorithms for SVMs are SMO [Platt '99] and SVM^{light} [Joachims' 99], both use *decomposition* to handle large size datasets.
- It seems the kernel trick is the most attracting site of SVMs. This idea has now been applied to many other learning models where the inner-product is concerned, and they are called 'kernel' methods.
- Tuning SVMs remains to be the main research focus: how to an optimal kernel? Kernel should match the smooth structure of data.