Support Vector Machines

Lecturer:
Jiří Matas

Authors:
Ondřej Drbohlav, Jiří Matas

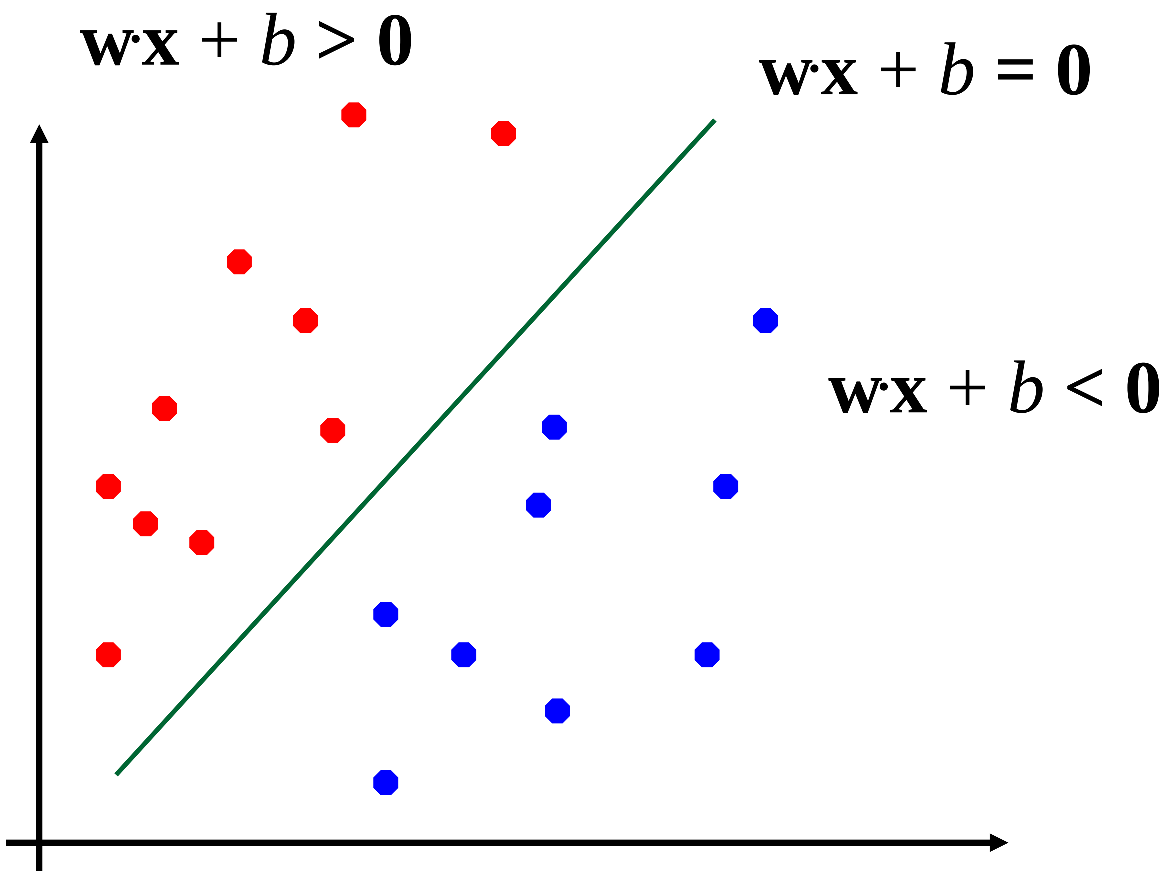
Centre for Machine Perception
Czech Technical University, Prague
http://cmp.felk.cvut.cz

Slide credits:
Alexander Apartsin

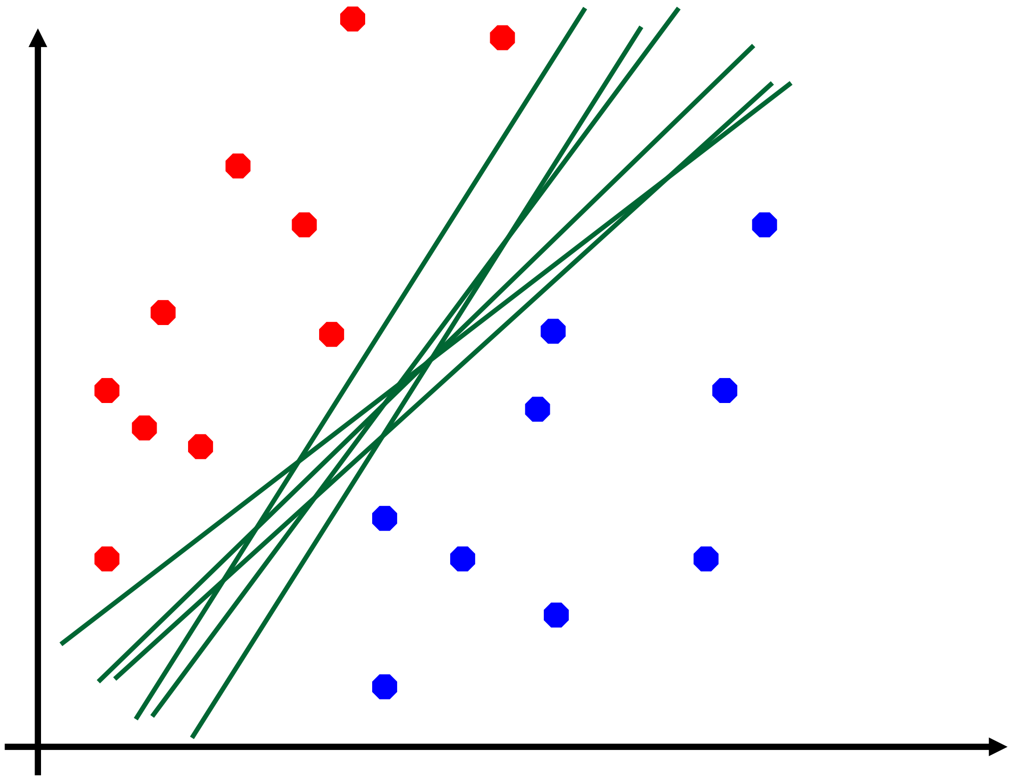
Last update: 13.11.2017

Perceptron Revisited:

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

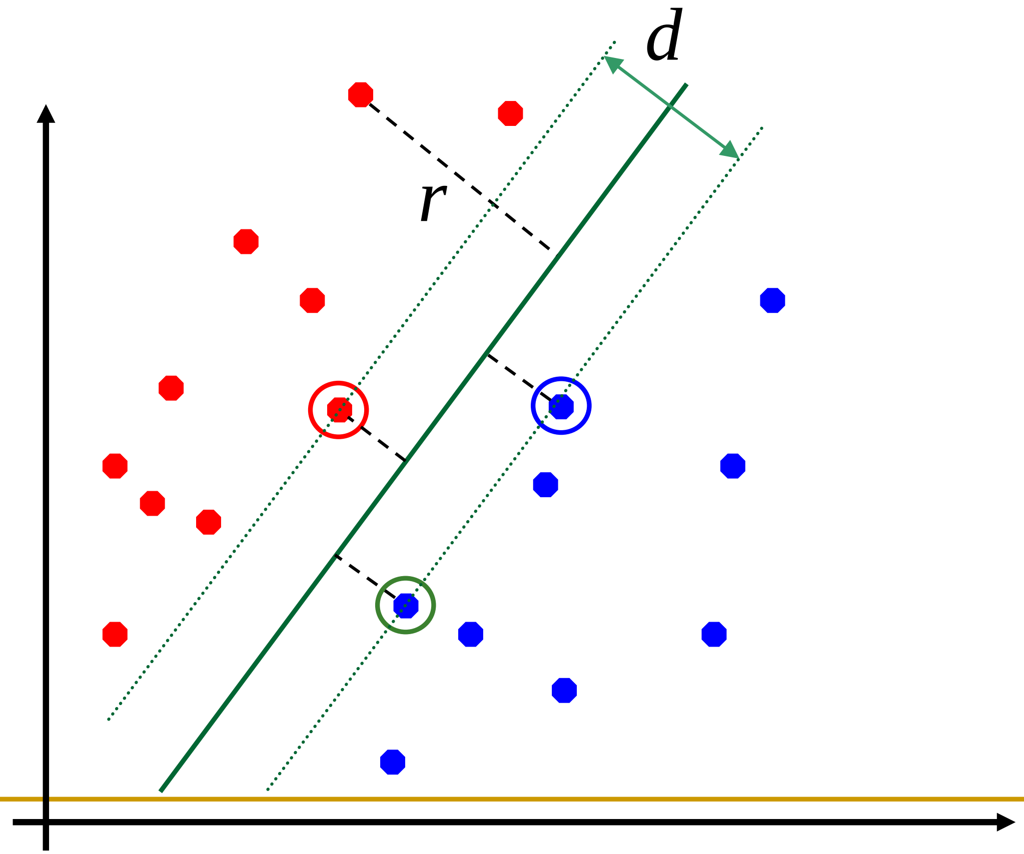


Which one is the best?



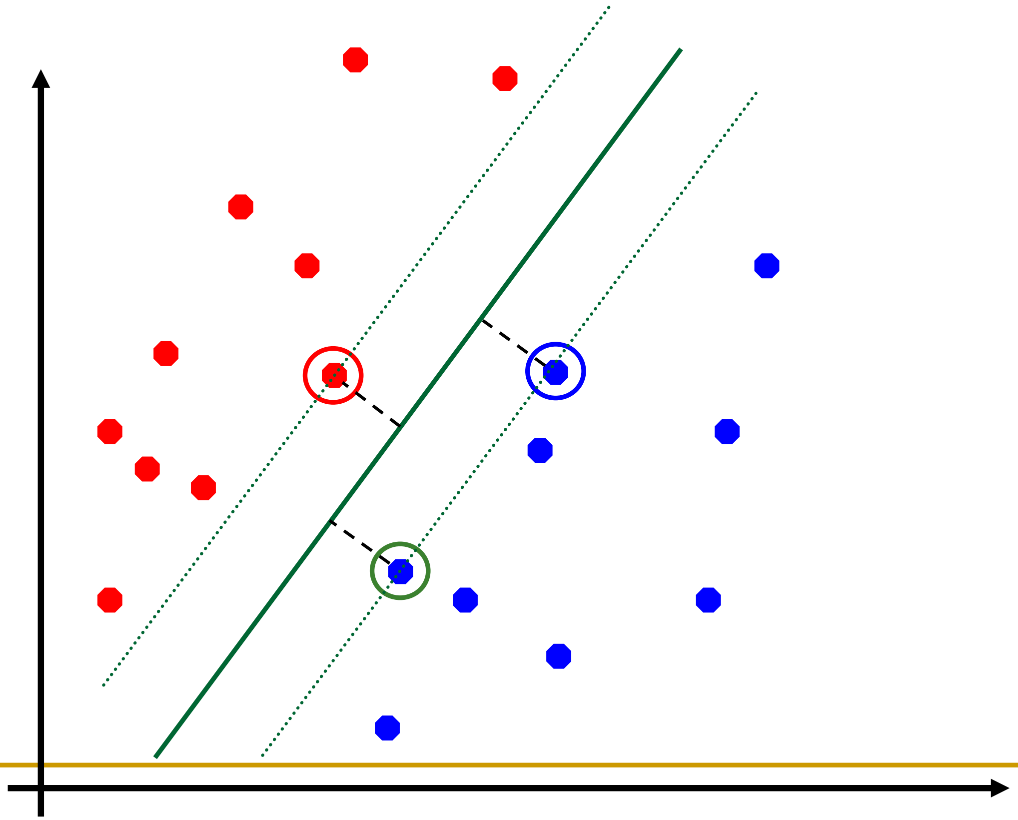
Notion of Margin

- Distance from a data point to the boundary: $r = \dfrac{|\mathbf{w} \cdot \mathbf{x} + b|}{\|\mathbf{w}\|}$
- 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:

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

  - 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

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

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 $\mathbf{h} = (h_1, \ldots, h_N)$ 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:

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

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

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

- 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

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

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

  - 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

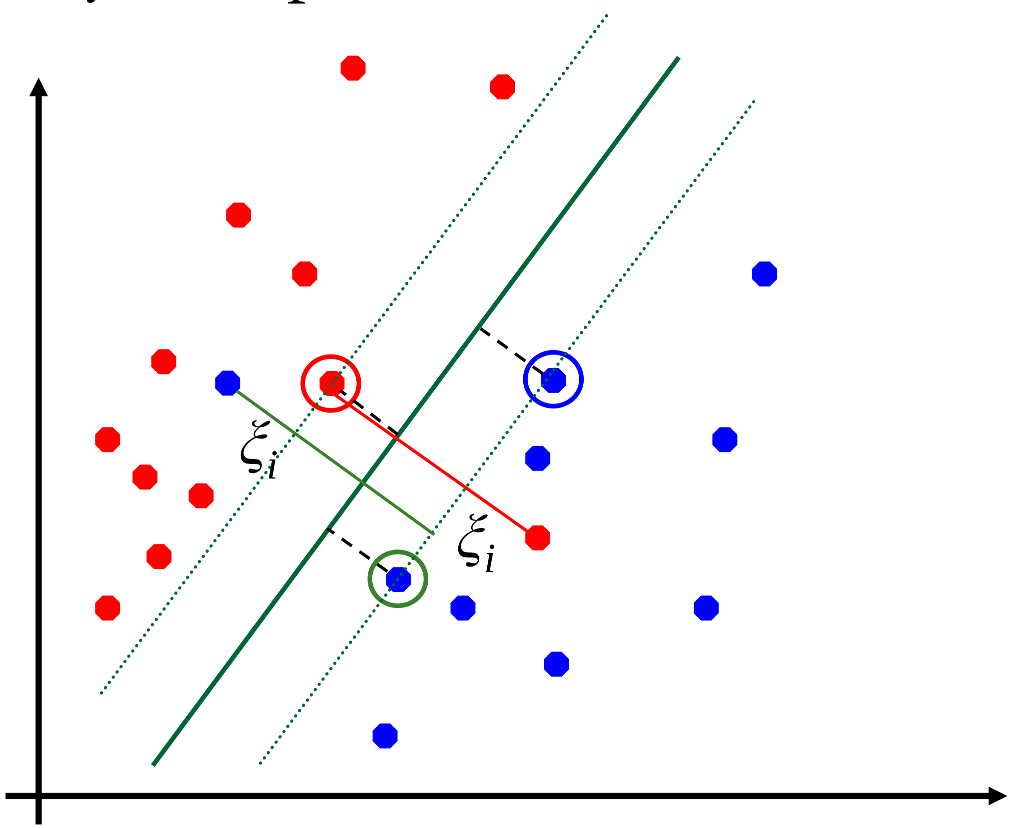
- Fixing b:

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:

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

Subject to

$$y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 - \xi_i, \text{ for } i = 1,\ldots,N$$
$$\xi_i \geq 0, \quad \text{for } i = 1,\ldots,N$$

- The dual problem:

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

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

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

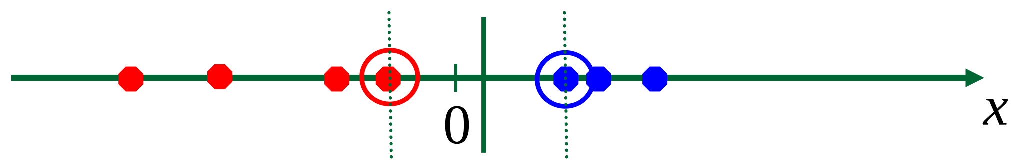
where $D_{ij} = y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$

Linear SVMs: Overview

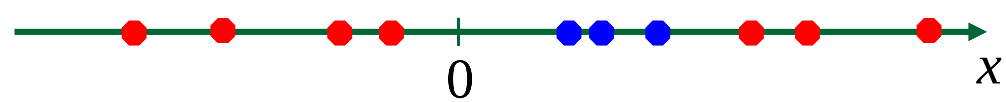
- The classifier is a *separating hyperplane.*
- Most "important" training points are support vectors; they define the hyperplane.
- Quadratic optimization algorithms can identify which training points $\mathbf{x_i}$ are support vectors with non-zero Lagrangian multipliers h_i.
- Both in the dual formulation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

Who really need linear classifiers

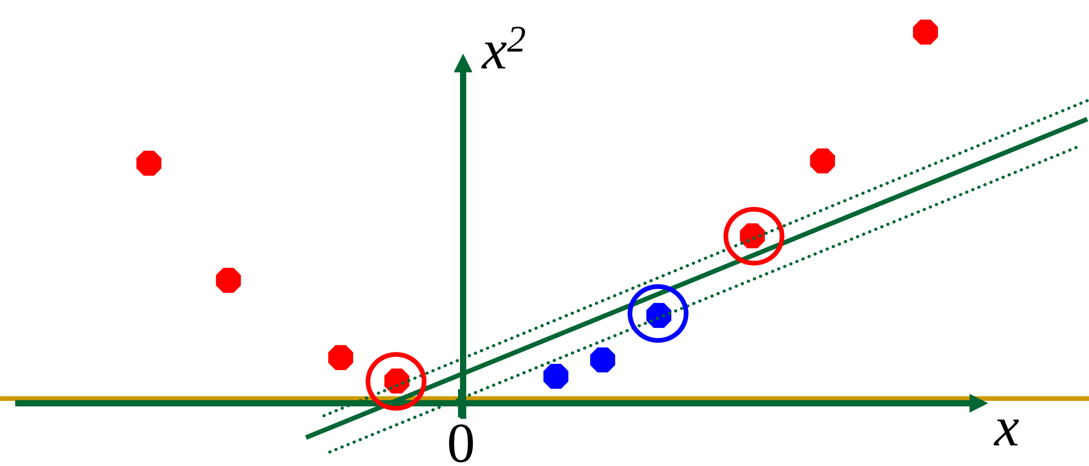
- Datasets that are linearly separable with some noise, linear SVM work well:



- But if the dataset is non-linearly separable?

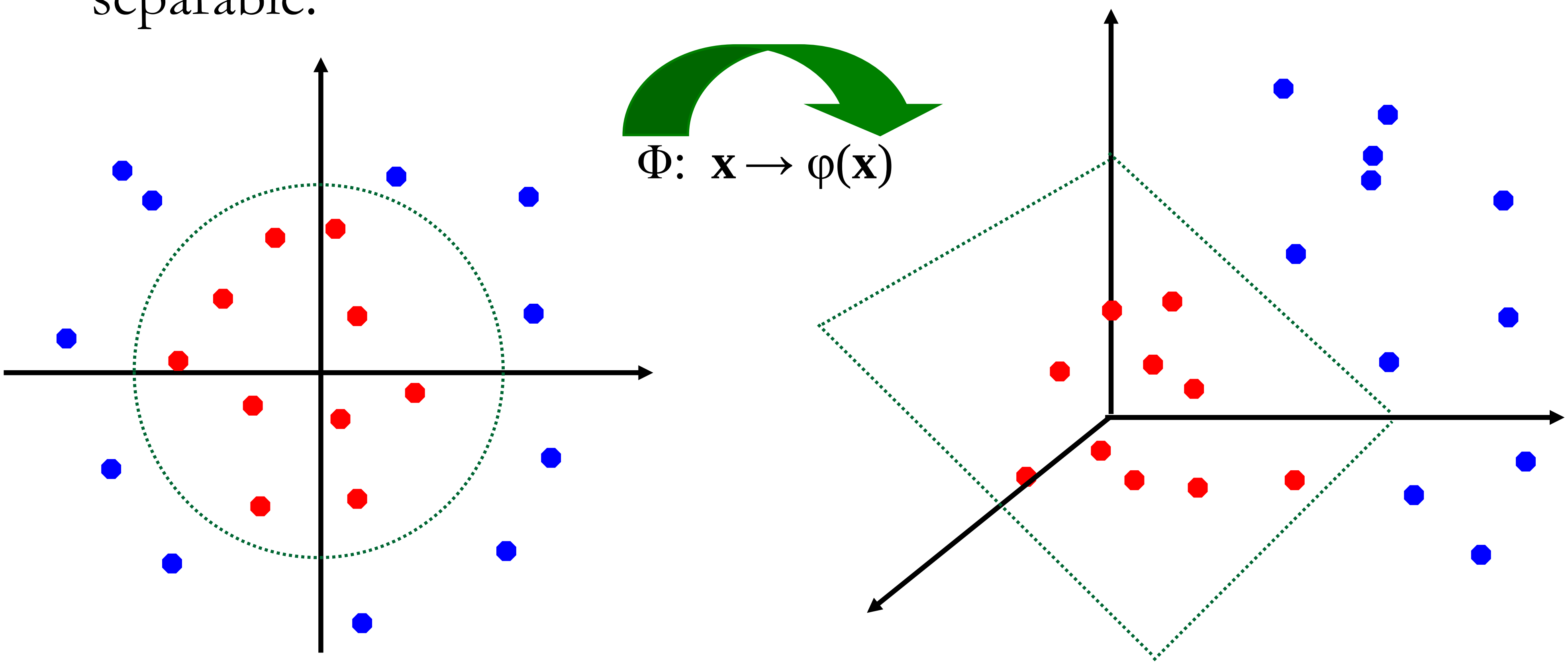


- How about… mapping data to a higher-dimensional space:



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: $\mathbf{x} \longrightarrow \varphi(\mathbf{x})$, 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:

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

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

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

$$\text{where } D_{ij} = y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

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

Lifting Dimension by Polynomial Mapping of Degree d

Let $d \in \mathbb{N}$ and $\mathbf{x} = [x_1, x_2, ..., x_D]^\top \in \mathbb{R}^D$.

Let $\phi_d(\mathbf{x})$ denote the mapping which lifts $\mathbf{x}$ to the space containing all monomials of degree d', $1 \le d' \le d$ in the components of $\mathbf{x}$:

For example, when $\mathbf{x} = [x_1, x_2]^\top \in \mathbb{R}^2$,

$$\phi_1(\mathbf{x}) = [x_1, x_2]^\top , \tag{1}$$

$$\phi_2(\mathbf{x}) = [x_1, x_2, x_1^2, x_1 x_2, x_2^2]^\top , \tag{2}$$

$$\phi_3(\mathbf{x}) = [x_1, x_2, x_1^2, x_1 x_2, x_2^2, x_1^3, x_1^2 x_2, x_1 x_2^2, x_2^3]^\top . \tag{3}$$

The number of monomials of degree d' of $\mathbf{x} \in \mathbb{R}^D$ is $\binom{d'+D-1}{d'}$. The dimensionality L of the output space of $\phi_d(\mathbf{x})$ is thus

$$L = \sum_{d'=1}^{d} \binom{d' + D - 1}{d'} . \tag{4}$$

Lifting Dimension by Polynomial Mapping of Degree d

Feature space dimensionality D, lifting by $\phi_d(\mathbf{x})$

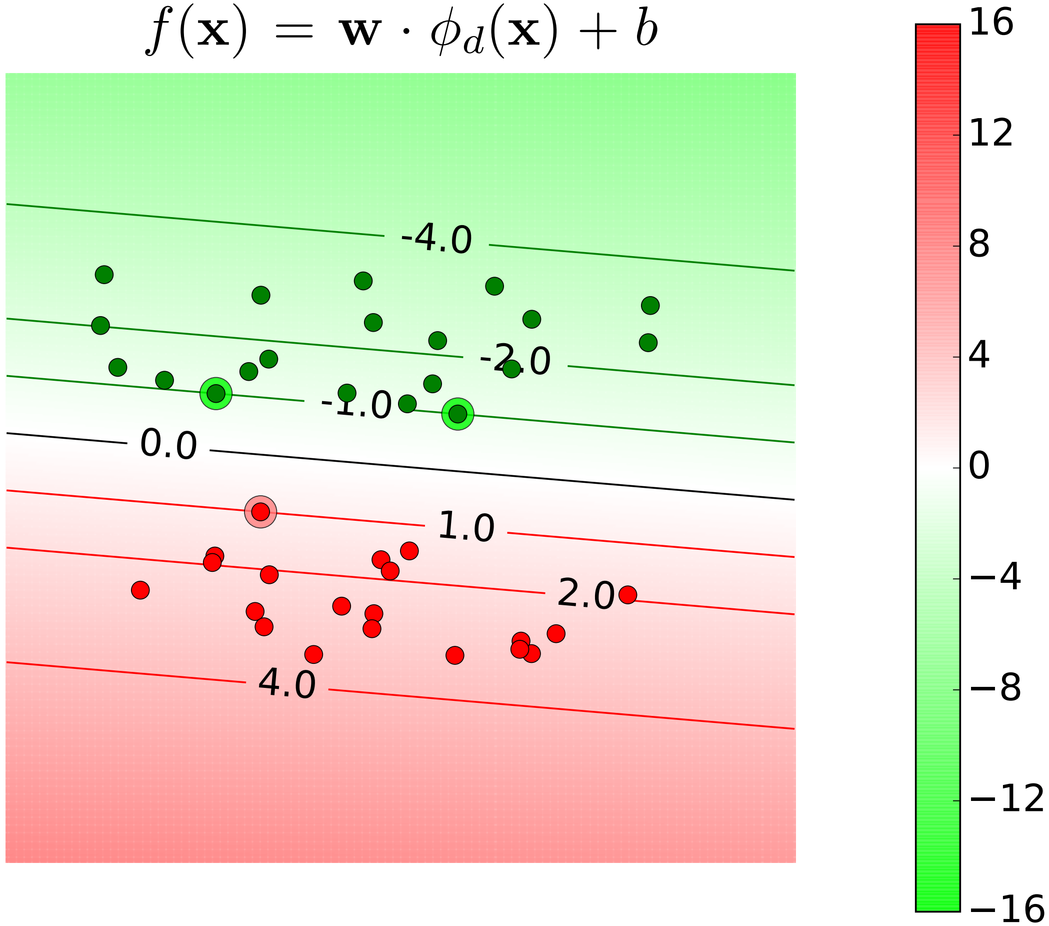
dimensionality of feature space after lifting (L)

D \ d	1	2	3	4	5	6	7	8
1	1	2	3	4	5	6	7	8
2	2	5	9	14	20	27	35	44
3	3	9	19	34	55	83	119	164
4	4	14	34	69	125	209	329	494
5	5	20	55	125	251	461	791	1286
6	6	27	83	209	461	923	1715	3002
7	7	35	119	329	791	1715	3431	6434
8	8	44	164	494	1286	3002	6434	12869

Lifting by Polynomial Mapping of Degree d, Example

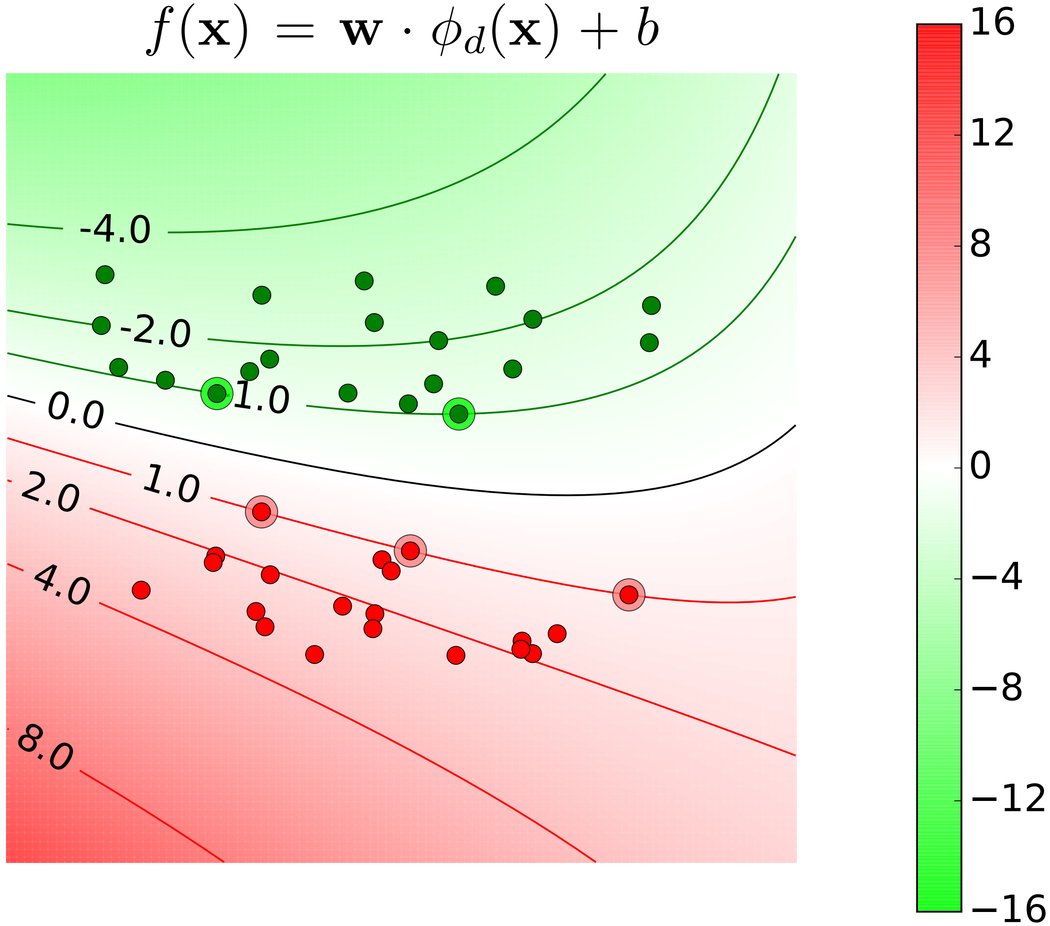
$d = 1$, dim$(\phi_d(\mathbf{x})) = 2$
support vectors : 3

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



$d = 2$, dim$(\phi_d(\mathbf{x})) = 5$
support vectors : 5

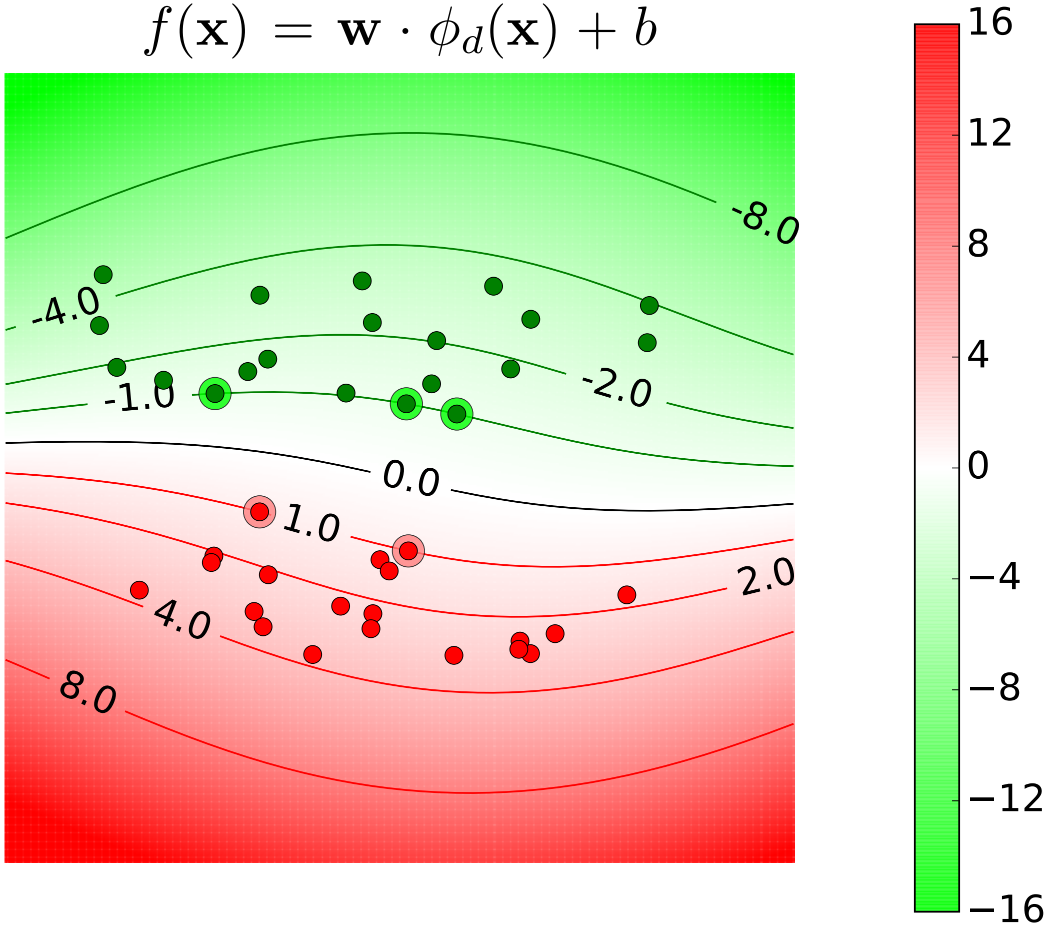
$f(\mathbf{x}) = \mathbf{w} \cdot \phi_d(\mathbf{x}) + b$



Lifting by Polynomial Mapping of Degree d, Example

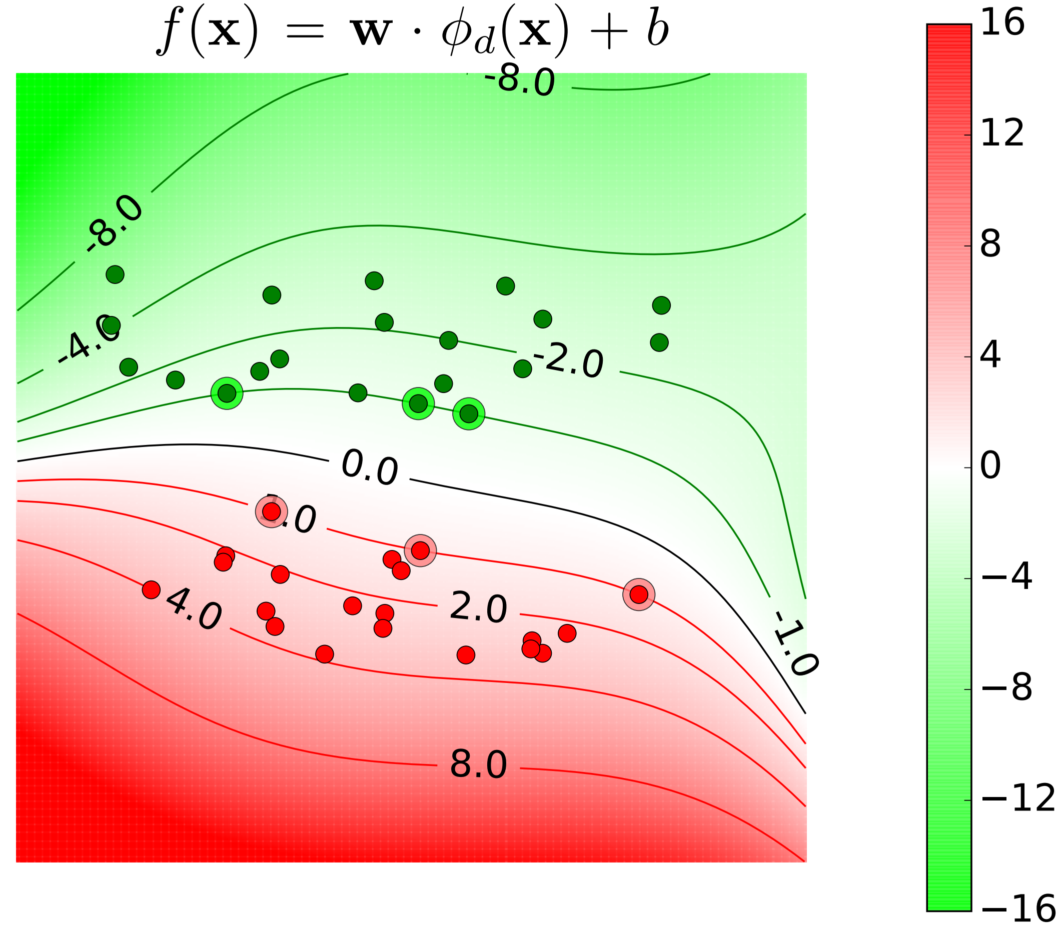
$d = 3$, $\dim(\phi_d(\mathbf{x})) = 9$
support vectors : 5

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



$d = 4$, $\dim(\phi_d(\mathbf{x})) = 14$
support vectors : 6

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



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.

Appendix

Online demo: http://cs.stanford.edu/people/karpathy/svmjs/demo/

The dual formulation (1)

$$\begin{aligned}&\text{Minimizing } \tfrac{1}{2}\|\mathbf{w}\|^2\\ &\text{subject to: } y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1, \quad \forall i \in \{1,2,...,N\}\end{aligned} \tag{5}$$

Let $f(\mathbf{w}, b)$ be defined as follows:

$$f(\mathbf{w}, b) = \begin{cases} \frac{1}{2}\|\mathbf{w}\|^2, & \text{if } y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1, \quad \forall i \in \{1,2,...,N\} \\ \infty, & \text{otherwise} \end{cases} \tag{6}$$

Then $\min_{\mathbf{w},b} f(\mathbf{w}, b)$ surely has the same minimum as (5). Now, $f(\mathbf{w}, b)$ can be rewritten as follows (h_i's are non-negative Lagrange multipliers):

$$f(\mathbf{w}, b) = \max_{\substack{\{h_i\}\\ h_i \geq 0\\ i\in\{1,..,N\}}} \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} h_i[y_i(\mathbf{w}\cdot\mathbf{x}_i + b) - 1] \tag{7}$$

The dual formulation (2)

The original optimization problem is thus equivalent to:

$$\min_{\mathbf{w},b} \max_{\substack{\{h_i\}\\ h_i \geq 0 \\ i\in\{1,..,N\}}} \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} h_i[y_i(\mathbf{w}\cdot\mathbf{x}_i + b) - 1] \tag{8}$$

There holds that $\max_x \min_y f(x,y) \leq \min_y \max_x f(x,y)$. For our case,

$$\begin{aligned} &\min_{\mathbf{w},b} \max_{\substack{\{h_i\}\\ h_i \geq 0 \\ i\in\{1,..,N\}}} \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} h_i[y_i(\mathbf{w}\cdot\mathbf{x}_i + b) - 1] \geq \\ &\geq \max_{\substack{\{h_i\}\\ h_i \geq 0 \\ i\in\{1,..,N\}}} \min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} h_i[y_i(\mathbf{w}\cdot\mathbf{x}_i + b) - 1] \end{aligned} \tag{9}$$

For our problem, strong duality holds and the two terms are equal (duality gap is zero).

The dual formulation (3)

Lagrangian:

$$L(\mathbf{w}, b, \mathbf{h}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} h_i[y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1] \tag{10}$$

$\mathbf{h} = (h_1, h_2, ..., h_N)$ vector of non-negative Lagrange multipliers.

$$\min_{\mathbf{w},b} \max_{\substack{\{h_i\} \\ h_i \geq 0 \\ i \in \{1,..,N\}}} L(\mathbf{w}, b, \mathbf{h}) = \max_{\substack{\{h_i\} \\ h_i \geq 0 \\ i \in \{1,..,N\}}} \min_{\mathbf{w},b} L(\mathbf{w}, b, \mathbf{h}) \tag{11}$$

Minimize $L(\mathbf{w}, b, \mathbf{h})$ over $\mathbf{w}$ and b:

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{N} h_i y_i \mathbf{x}_i = 0 \tag{12}$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{N} h_i y_i = 0 \tag{13}$$

The dual formulation (4)

The optimal value for $\mathbf{w}$ is $\mathbf{w} = \sum_{i=1}^{N} h_i y_i \mathbf{x}_i$ and $\sum_i h_i y_i = 0$, thus

$$\min_{\mathbf{w},b} L(\mathbf{w}, b, \mathbf{h}) = \min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} h_i [y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1] \tag{14}$$

$$= -\frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{N} h_i = \sum_{i=1}^{N} h_i - \frac{1}{2} \sum_{i,j=1}^{N} h_i h_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \tag{15}$$

$$= \mathbf{1}^\top \mathbf{h} - \frac{1}{2}\mathbf{h}^\top \mathbf{D}\mathbf{h}, \qquad D_{ij} = y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \tag{16}$$

and the dual optimization problem is:

$$\max_{\{h_i\}} \mathbf{1}^\top \mathbf{h} - \frac{1}{2}\mathbf{h}^\top \mathbf{D}\mathbf{h} \tag{17}$$

$$\text{subject to: } \sum_i h_i y_i = 0; \quad h_i \geq 0, \quad \forall i \in \{1, 2, ..., N\} \tag{18}$$