Artificial Intelligence

Kernel Methods and SVM

Donghui Wang
AI Institute@ZJU
2015.4

Contents

- Kernel methods
- Maximum margin classifiers

References:

1. *Bishop. "Pattern Recognition and Machine Learning", Chapter 6,7. 2006.*

Kernel Methods

Two approaches to use training data

- Use training data only in learning phase:
  - Obtain a point estimate of the parameter vector w
  - Determine a posterior distribution over w
  - Predictions for new inputs are based purely on the learned parameter vector w

Slow to learn but fast at making prediction.

- Use training data in both of learning phase and prediction phase:
  - Kept and use training data also during the prediction phase (memory-based)
  - Typically require to measure the similarity of any two vectors in input space
  - E.g. nearest neighbors classifier

Fast to learn but slow at making prediction.

Kernel function and kernel trick

- Similarity metric in feature space can be defined by *kernel function*:

$$k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^{\mathrm{T}}\phi(\mathbf{x}')$$

- The kernel is symmetric function: $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}', \mathbf{x})$
- The simplest example of a kernel function: $k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^{\mathrm{T}}\mathbf{x}'$ $\phi(\mathbf{x}) = \mathbf{x}$

- *Kernel trick* (kernel substitution)
  - The general idea is that, if we have an algorithm formulated in such a way that the input vector x enters only in the form of scalar products (e.g. inner products <x, y>), then we can replace that scalar product with some other choice of kernel k(x,y) and allow new learning algorithm to work efficiently in the high dimensional feature space.
  - E.g. perceptron, SVM, PCA, CCA, Fisher's linear discriminant analysis and cluster analysis

Dual representations

- In linear regression model, we have regularized SSE function:

M parameters

$$J(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\left\{\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n) - t_n\right\}^2 + \frac{\lambda}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w} = \frac{1}{2}\left\|\mathbf{\Phi}\mathbf{w} - \mathbf{t}\right\|_2^2 + \frac{\lambda}{2}\left\|\mathbf{w}\right\|_2^2$$

- *Let:*

N parameters

$$\mathbf{w} = \sum_{n=1}^{N} a_n \phi(\mathbf{x}_n) = \mathbf{\Phi}^{\mathrm{T}}\mathbf{a}$$

$$\mathbf{\Phi} = \begin{pmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \cdots & \phi_{M-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \cdots & \phi_{M-1}(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \cdots & \phi_{M-1}(\mathbf{x}_N) \end{pmatrix}$$

$N \times M$ design matrix

- *Substitute into J(w) to obtain dual representation:*

$$J(\mathbf{a}) = \frac{1}{2}\mathbf{a}^{\mathrm{T}}\mathbf{\Phi}\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\mathbf{\Phi}^{\mathrm{T}}\mathbf{a} - \mathbf{a}^{\mathrm{T}}\mathbf{\Phi}\mathbf{\Phi}^{\mathrm{T}}\mathbf{t} + \frac{1}{2}\mathbf{t}^{\mathrm{T}}\mathbf{t} + \frac{\lambda}{2}\mathbf{a}^{\mathrm{T}}\mathbf{\Phi}\mathbf{\Phi}^{\mathrm{T}}\mathbf{a}$$

$N \times N$ Gram matrix

$$\xrightarrow{\mathbf{K} = \mathbf{\Phi}\mathbf{\Phi}^{\mathrm{T}}} \quad J(\mathbf{a}) = \frac{1}{2}\mathbf{a}^{\mathrm{T}}\mathbf{K}\mathbf{K}\mathbf{a} - \mathbf{a}^{\mathrm{T}}\mathbf{K}\mathbf{t} + \frac{1}{2}\mathbf{t}^{\mathrm{T}}\mathbf{t} + \frac{\lambda}{2}\mathbf{a}^{\mathrm{T}}\mathbf{K}\mathbf{a}$$

$$\xrightarrow{\partial J(\mathbf{a})/\partial\mathbf{a} = 0} \quad \mathbf{a} = (\mathbf{K} + \lambda\mathbf{I}_N)^{-1}\mathbf{t}$$

$$\boxed{K_{nm} = \phi(\mathbf{x}_n)^{\mathrm{T}}\phi(\mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m)}$$

- *The prediction for a new input x:*

It's memory-based method!

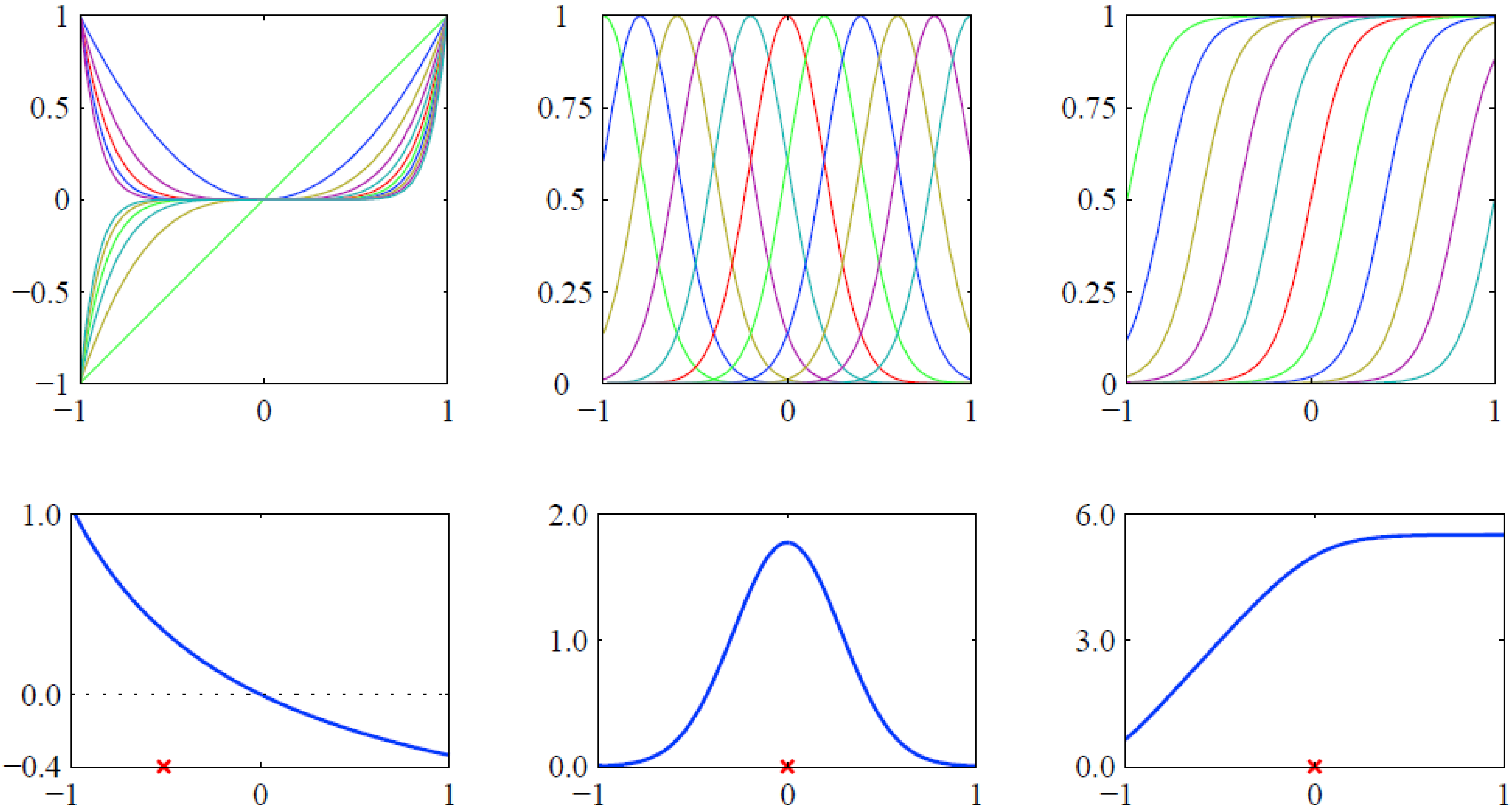
$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}) = \mathbf{a}^{\mathrm{T}}\mathbf{\Phi}\phi(\mathbf{x}) = \mathbf{k}(\mathbf{x})^{\mathrm{T}}(\mathbf{K} + \lambda\mathbf{I}_N)^{-1}\mathbf{t}$$

$$\mathbf{k}(\mathbf{x}) = \begin{pmatrix} k(\mathbf{x}_1, \mathbf{x}) \\ k(\mathbf{x}_2, \mathbf{x}) \\ \vdots \\ k(\mathbf{x}_N, \mathbf{x}) \end{pmatrix}$$

Constructing kernels

- Methods #1:
  - Choose a feature space mapping φ(x) and use it to find the corresponding kernel.

$$k(x, x') = \phi(x)^{\mathrm{T}}\phi(x') = \sum_{i=1}^{M} \phi_i(x)\phi_i(x')$$



Constructing kernels

- Methods #2:
  - Construct kernel function directly, but we must ensure the function we choose is a valid kernel (it can be represented by a scalar product in some feature space).

- *Example 1:* $k(\mathbf{x},\mathbf{z}) = \left(\mathbf{x}^{\mathrm{T}}\mathbf{z}\right)^2 = (x_1z_1 + x_2z_2)^2 = x_1^2z_1^2 + 2x_1z_1x_2z_2 + x_2^2z_2^2$

$$= (x_1^2, \sqrt{2}x_1x_2, x_2^2)(z_1^2, \sqrt{2}z_1z_2, z_2^2)^{\mathrm{T}} = \boxed{\phi(\mathbf{x})^{\mathrm{T}}\phi(\mathbf{z})}$$

  - Construct new kernels by using simpler kernels (building blocks).

Techniques for Constructing New Kernels.

Given valid kernels $k_1(\mathbf{x},\mathbf{x}')$ and $k_2(\mathbf{x},\mathbf{x}')$, the following new kernels will also be valid:

$$
\begin{aligned}
k(\mathbf{x},\mathbf{x}') &= ck_1(\mathbf{x},\mathbf{x}') \\
k(\mathbf{x},\mathbf{x}') &= f(\mathbf{x})k_1(\mathbf{x},\mathbf{x}')f(\mathbf{x}') \\
k(\mathbf{x},\mathbf{x}') &= q\left(k_1(\mathbf{x},\mathbf{x}')\right) \\
k(\mathbf{x},\mathbf{x}') &= \exp\left(k_1(\mathbf{x},\mathbf{x}')\right) \\
k(\mathbf{x},\mathbf{x}') &= k_1(\mathbf{x},\mathbf{x}') + k_2(\mathbf{x},\mathbf{x}')
\end{aligned}
$$

$$
\begin{aligned}
k(\mathbf{x},\mathbf{x}') &= k_1(\mathbf{x},\mathbf{x}')k_2(\mathbf{x},\mathbf{x}') \\
k(\mathbf{x},\mathbf{x}') &= k_3\left(\phi(\mathbf{x}),\phi(\mathbf{x}')\right) \\
k(\mathbf{x},\mathbf{x}') &= \mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x}' \\
k(\mathbf{x},\mathbf{x}') &= k_a(\mathbf{x}_a,\mathbf{x}'_a) + k_b(\mathbf{x}_b,\mathbf{x}'_b) \\
k(\mathbf{x},\mathbf{x}') &= k_a(\mathbf{x}_a,\mathbf{x}'_a)k_b(\mathbf{x}_b,\mathbf{x}'_b)
\end{aligned}
$$

where $c > 0$ is a constant, $f(\cdot)$ is any function, $q(\cdot)$ is a polynomial with nonnegative coefficients, $\phi(\mathbf{x})$ is a function from $\mathbf{x}$ to $\mathbb{R}^M$, $k_3(\cdot,\cdot)$ is a valid kernel in $\mathbb{R}^M$, $\mathbf{A}$ is a symmetric positive semidefinite matrix, $\mathbf{x}_a$ and $\mathbf{x}_b$ are variables (not necessarily disjoint) with $\mathbf{x} = (\mathbf{x}_a, \mathbf{x}_b)$, and k_a and k_b are valid kernel functions over their respective spaces.

Constructing kernels

- Methods #2:
  - Construct new kernels by using simpler kernels (building blocks).

$$
\begin{aligned}
k(\mathbf{x},\mathbf{x}') &= ck_1(\mathbf{x},\mathbf{x}') \\
k(\mathbf{x},\mathbf{x}') &= f(\mathbf{x})k_1(\mathbf{x},\mathbf{x}')f(\mathbf{x}') \\
k(\mathbf{x},\mathbf{x}') &= q\left(k_1(\mathbf{x},\mathbf{x}')\right) \\
k(\mathbf{x},\mathbf{x}') &= \exp\left(k_1(\mathbf{x},\mathbf{x}')\right) \\
k(\mathbf{x},\mathbf{x}') &= k_1(\mathbf{x},\mathbf{x}') + k_2(\mathbf{x},\mathbf{x}')
\end{aligned}
\qquad
\begin{aligned}
k(\mathbf{x},\mathbf{x}') &= k_1(\mathbf{x},\mathbf{x}')k_2(\mathbf{x},\mathbf{x}') \\
k(\mathbf{x},\mathbf{x}') &= k_3\left(\boldsymbol{\phi}(\mathbf{x}),\boldsymbol{\phi}(\mathbf{x}')\right) \\
k(\mathbf{x},\mathbf{x}') &= \mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x}' \\
k(\mathbf{x},\mathbf{x}') &= k_a(\mathbf{x}_a,\mathbf{x}'_a) + k_b(\mathbf{x}_b,\mathbf{x}'_b) \\
k(\mathbf{x},\mathbf{x}') &= k_a(\mathbf{x}_a,\mathbf{x}'_a)k_b(\mathbf{x}_b,\mathbf{x}'_b)
\end{aligned}
$$

- *Example 2:* $k(\mathbf{x},\mathbf{x}') = \exp\left(-\|\mathbf{x}-\mathbf{x}'\|^2/2\sigma^2\right)$ $\qquad \|\mathbf{x}-\mathbf{x}'\|^2 = \mathbf{x}^{\mathrm{T}}\mathbf{x} + (\mathbf{x}')^{\mathrm{T}}\mathbf{x}' - 2\mathbf{x}^{\mathrm{T}}\mathbf{x}'$

$$
\Rightarrow \quad k(\mathbf{x},\mathbf{x}') = \underbrace{\exp\left(-\mathbf{x}^{\mathrm{T}}\mathbf{x}/2\sigma^2\right)}_{f(x)} \underbrace{\boxed{\exp\left(\boxed{\mathbf{x}^{\mathrm{T}}\mathbf{x}'}/\sigma^2\right)}}_{\textit{Valid kernel}} \underbrace{\exp\left(-(\mathbf{x}')^{\mathrm{T}}\mathbf{x}'/2\sigma^2\right)}_{f(x')}
$$

$$
\exp(\mathbf{x}) = \sum_{m=0}^{\infty} \frac{\mathbf{x}^m}{m!} \quad \Rightarrow \quad \exp\left(\mathbf{x}^{\mathrm{T}}\mathbf{x}'/\sigma^2\right) = \sum_{m=0}^{\infty} \phi_m(\mathbf{x})^{\mathrm{T}}\phi_m(\mathbf{x}') = \underbrace{\psi(\mathbf{x})^{\mathrm{T}}\psi(\mathbf{x}')}_{\textit{Infinite dimension}}
$$

$$
\Rightarrow \quad k(\mathbf{x},\mathbf{x}') = \varphi(\mathbf{x})^{\mathrm{T}}\varphi(\mathbf{x}') \quad \text{where} \quad \varphi(\mathbf{x}) = \exp\left(-\frac{\mathbf{x}^{\mathrm{T}}\mathbf{x}}{2\sigma^2}\right)\psi(\mathbf{x}).
$$

Maximum margin classifiers

The problem of kernel-based algorithms

- Training phase:
  - Must evaluate all possible pairs x_n and x_m:

$$\mathbf{a} = (\mathbf{K} + \lambda \mathbf{I}_N)^{-1}\mathbf{t} \qquad K_{nm} = \phi(\mathbf{x}_n)^{\mathrm{T}}\phi(\mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m)$$

- Prediction phase:
  - Must compute kernel between input data x and all training data:

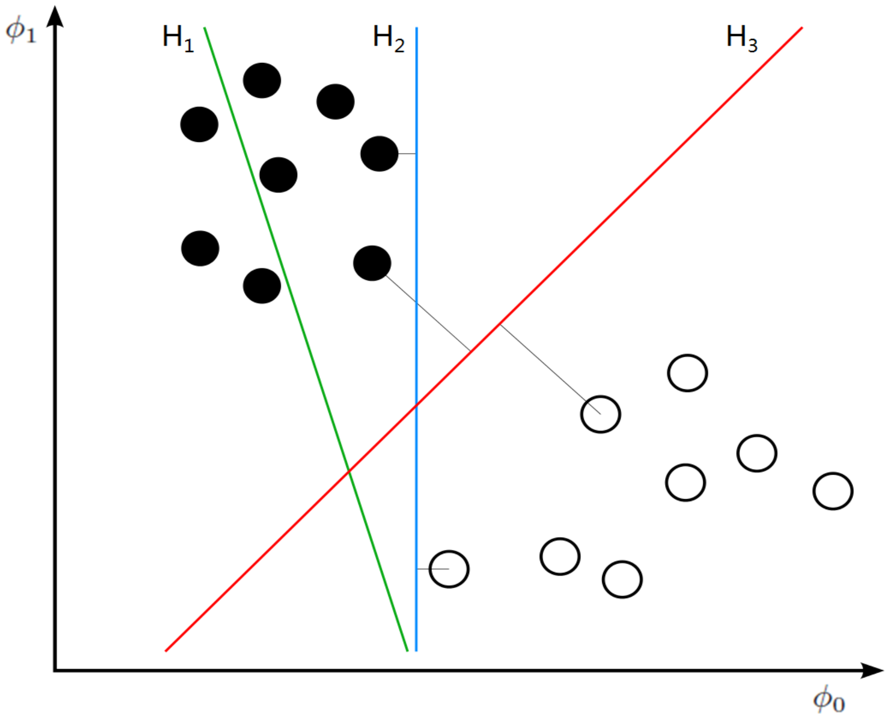
$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}) = \mathbf{a}^{\mathrm{T}}\Phi\phi(\mathbf{x}) = \mathbf{k}(\mathbf{x})^{\mathrm{T}}(\mathbf{K} + \lambda \mathbf{I}_N)^{-1}\mathbf{t} \qquad \mathbf{k}(\mathbf{x}) = \begin{pmatrix} k(\mathbf{x}_1, \mathbf{x}) \\ k(\mathbf{x}_2, \mathbf{x}) \\ \vdots \\ k(\mathbf{x}_N, \mathbf{x}) \end{pmatrix}$$

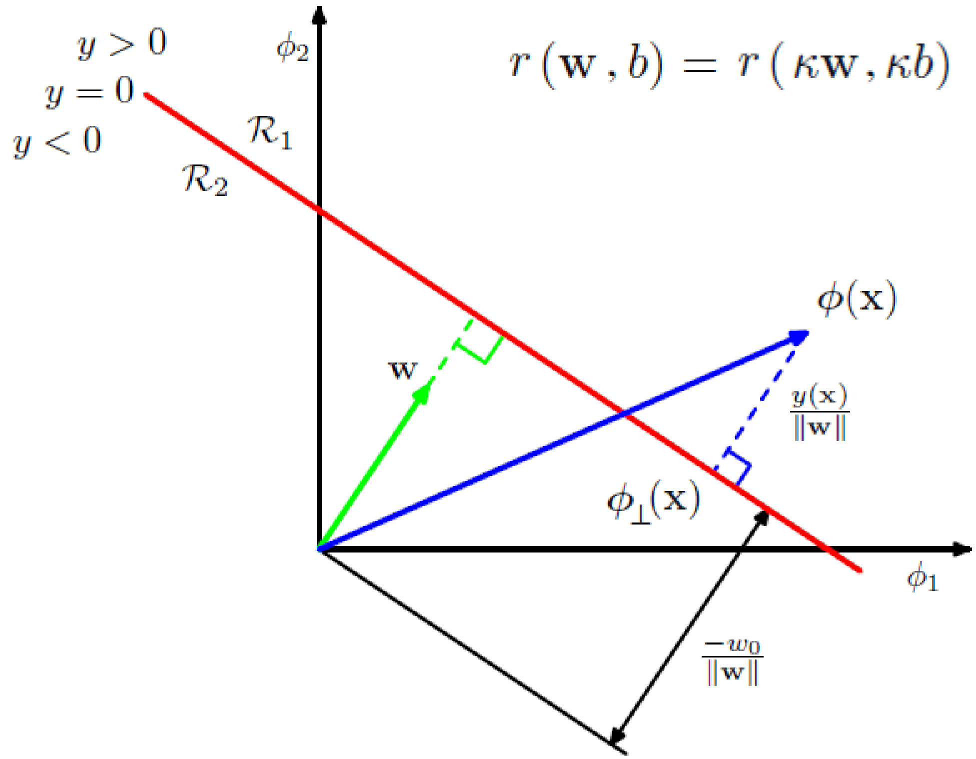
- Can the kernel-based algorithms work more effective?
  - We shall look at kernel-based algorithm that have *sparse* solutions!
  - E.g. SVM

Maximum margin classifiers

- For the two-class classification problem using linear models: $y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}) + b$
  - Assume the training data set is linearly separable in feature space;
  - And all data points are correctly classified, so that: $t_n y(\mathbf{x}_n) > 0$ $\quad t_n \in \{-1, 1\}$
- The definition of the *Margin* (rescaling w and b doesn't change *r*)

$$r(\mathbf{w}, b) = \min_n \left\{ \frac{t_n y(\mathbf{x}_n)}{\|\mathbf{w}\|} \right\} = \min_n \left\{ \frac{t_n(\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n) + b)}{\|\mathbf{w}\|} \right\}$$





Maximum margin classifiers

- Optimization problem: *find the solution of the maximum margin*

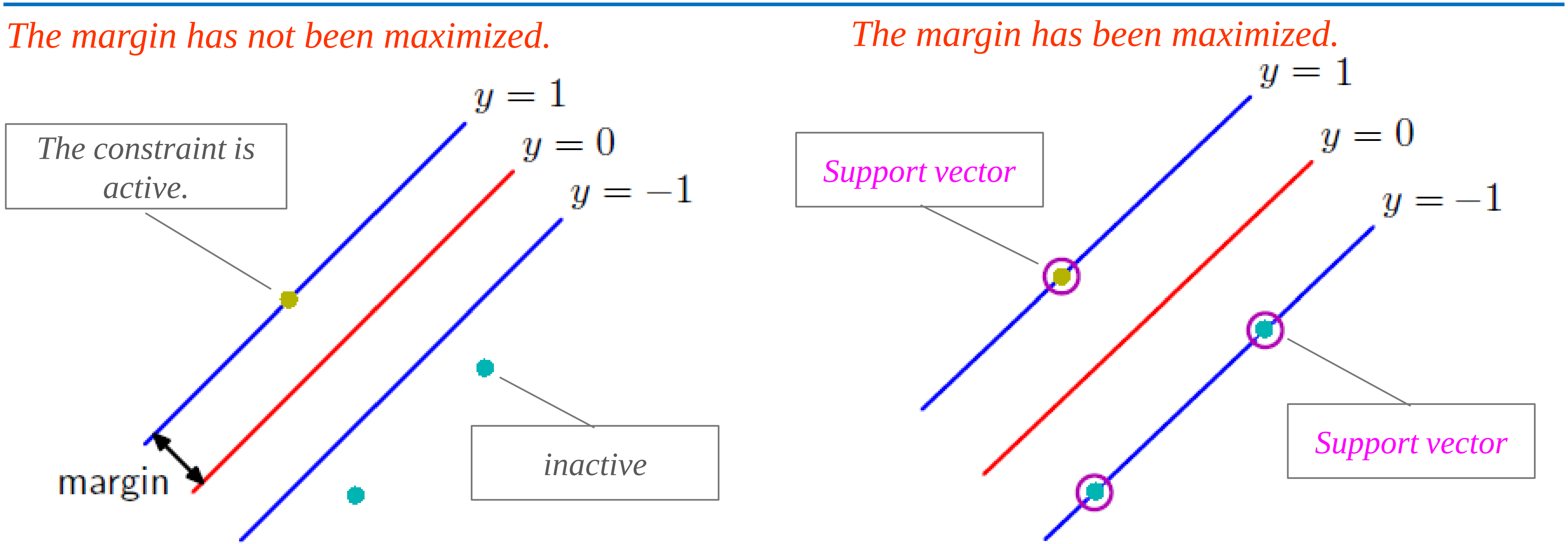
$$\arg\max_{\mathbf{w},b} \; r(\mathbf{w},b) \iff \arg\max_{\mathbf{w},b} \left\{ \frac{1}{\|\mathbf{w}\|} \min_n \left[t_n \left(\mathbf{w}^{\mathrm{T}} \phi(\mathbf{x}_n) + b \right) \right] \right\}$$

  – Because $r(\mathbf{w},b) = r(\kappa\mathbf{w},\kappa b)$, so we can set $t_n \left(\mathbf{w}^{\mathrm{T}} \phi(\mathbf{x}_n) + b \right) = 1$ for the point that is closest to the decision surface. Then, all data points will satisfy the constraints: $t_n y(\mathbf{x}_n) \geqslant 1$

- Equivalent constrained optimization problem:

$$\arg\max_{\mathbf{w},b} \; r(\mathbf{w},b) \iff \arg\max_{\mathbf{w},b} \left\{ \frac{1}{\|\mathbf{w}\|} \right\} \iff \arg\min_{\mathbf{w},b} \frac{1}{2} \|\mathbf{w}\|^2$$

$$\text{subject to } t_n y(\mathbf{x}_n) \geqslant 1, \quad n = 1, \ldots, N.$$



Maximum margin classifiers

- Solving the constrained optimization problem:

$$\arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \qquad \text{subject to } t_n y(\mathbf{x}_n) \geqslant 1, \; n = 1, \ldots, N.$$

  – Introduce Lagrange multipliers $a_n \geq 0$, the Lagrange function:

$$L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{n=1}^{N} a_n \left\{ t_n(\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n) + b) - 1 \right\} \quad \text{where } \mathbf{a} = (a_1, \ldots, a_N)^{\mathrm{T}}$$

$$\frac{\partial L(\mathbf{w}, b, \mathbf{a})}{\partial \mathbf{w}} = 0 \Longrightarrow \mathbf{w} = \sum_{n=1}^{N} a_n t_n \phi(\mathbf{x}_n) \qquad \frac{\partial L(\mathbf{w}, b, \mathbf{a})}{\partial b} = 0 \Longrightarrow 0 = \sum_{n=1}^{N} a_n t_n$$

  – Substitute into *L(w, b, a)* to obtain dual representation:

$$\boxed{k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^{\mathrm{T}}\phi(\mathbf{x}')}$$

$$\widetilde{L}(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m) \quad \text{subject to} \quad a_n \geqslant 0, \; \sum_{n=1}^{N} a_n t_n = 0.$$

  – Use the *Sequential Minimal Optimization* (SMO) algorithm to solve above problem.

- Classify new data:

memory-based

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

Maximum margin classifiers

- *Karush-Kuhn-Tucher* (KKT) conditions:

KKT conditions

$$\arg\max_{\mathbf{x}} f(\mathbf{x}) \quad \text{subject to } g(\mathbf{x}) \geqslant 0 \quad \Longleftrightarrow \quad \arg\max_{\mathbf{x},\lambda} L(\mathbf{x},\lambda) \equiv f(\mathbf{x}) + \lambda g(\mathbf{x}) \quad \text{subject to } \boxed{g(\mathbf{x}) \geqslant 0,\ \lambda \geqslant 0,\ \lambda g(\mathbf{x}) = 0}$$

$$\arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{subject to } t_n y(\mathbf{x}_n) \geqslant 1 \quad \Longleftrightarrow \quad \arg\min_{\mathbf{w},b} L(\mathbf{w},b,\mathbf{a}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{n=1}^{N} a_n \{t_n y(\mathbf{x}_n) - 1\}$$

$$\text{subject to} \quad \boxed{\begin{aligned} a_n &\geqslant 0 \\ t_n y(\mathbf{x}_n) - 1 &\geqslant 0 \\ a_n \{t_n y(\mathbf{x}_n) - 1\} &= 0 \end{aligned}}$$

Thus for every data point, either $a_n = 0$ or $t_n y(\mathbf{x}_n) = 1$.

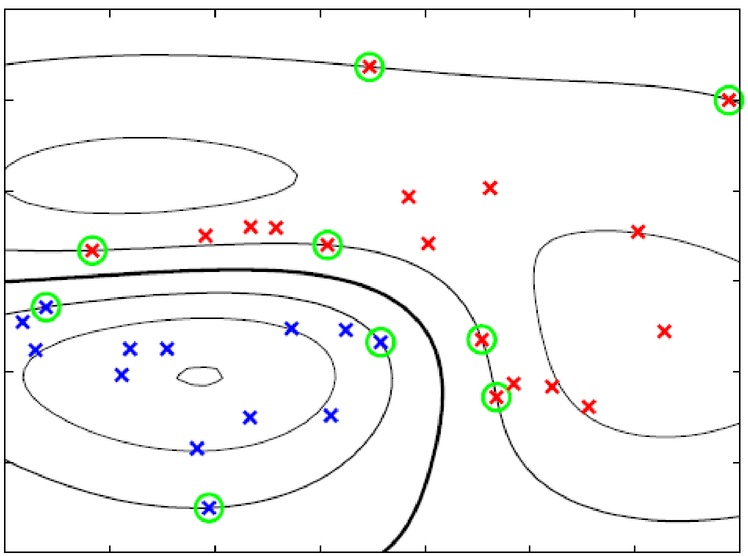
- Only *support vectors can make* $a_n > 0$ *, that means we don't need to keep all training data for prediction:*

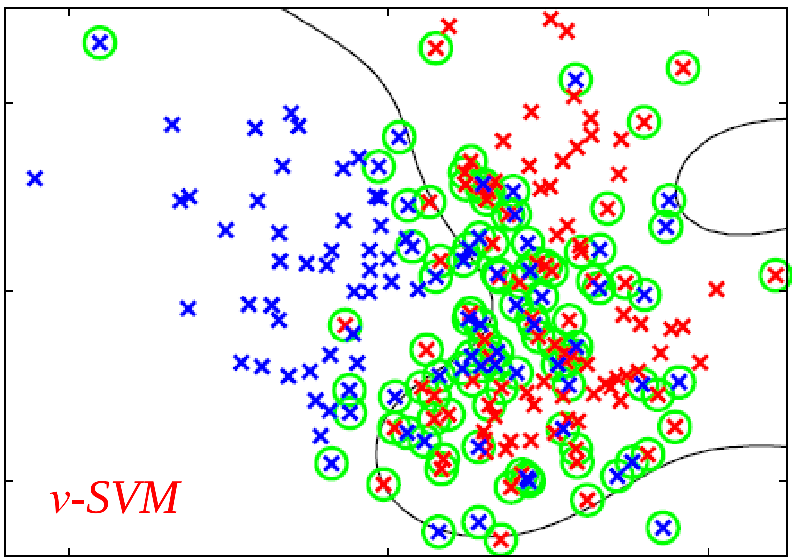
$$y(\mathbf{x}) = \sum_{n=1}^{N} a_n t_n k(\mathbf{x}, \mathbf{x}_n) + b \quad \Longrightarrow \quad y(\mathbf{x}) = \sum_{m \in \mathcal{S}} a_m t_m k(\mathbf{x}, \mathbf{x}_m) + b$$

Support Vectors set

Maximum margin classifiers

- *Support Vector Machines (SVM) learning algorithm*:
  1. *Choose a kernel function, e.g. Gaussian kernel function.*
  2. *Use SMO algorithm to solve* $\widetilde{L}(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m)$
  3. *Select support vectors with* $a_n > 0$
  4. *Compute the threshold parameter b by using SV set:* $b = \frac{1}{N_{\mathcal{S}}} \sum_{n \in \mathcal{S}} \left(t_n - \sum_{m \in \mathcal{S}} a_m t_m k(\mathbf{x}_n, \mathbf{x}_m) \right)$
  5. *Use SV set to classify new data point:* $y(\mathbf{x}) = \sum_{m \in \mathcal{S}} a_m t_m k(\mathbf{x}, \mathbf{x}_m) + b$





Maximum margin classifiers

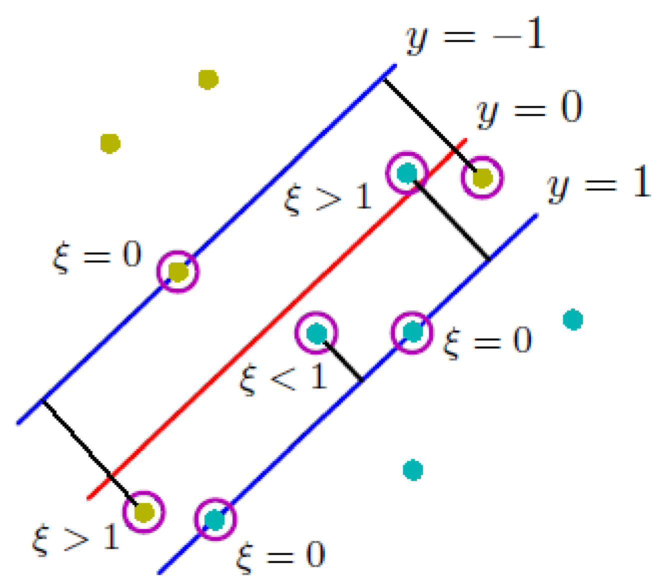
- Overlapping class distributions:
  - Slack variables $\xi_n = |t_n - y(\mathbf{x}_n)| \geqslant 0$
  - Now,

$$\arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2$$
$$\text{subject to } t_n y(\mathbf{x}_n) \geqslant 1$$

Hard margin

$$\arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{n=1}^{N}\xi_n$$
$$\text{subject to } t_n y(\mathbf{x}_n) \geqslant 1 - \xi_n$$
$$C > 0\,, \quad \xi_n \geqslant 0$$

Soft margin



- The corresponding Lagrange function:

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

KKT conditions:

$$\begin{aligned} a_n &\geqslant 0 \\ t_n y(\mathbf{x}_n) - 1 + \xi_n &\geqslant 0 \\ a_n\left(t_n y(\mathbf{x}_n) - 1 + \xi_n\right) &= 0 \\ \mu_n &\geqslant 0 \\ \xi_n &\geqslant 0 \\ \mu_n \xi_n &= 0 \end{aligned}$$

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

$$\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_{n=1}^{N} a_n t_n = 0 \qquad \frac{\partial L}{\partial \xi_n} = 0 \Rightarrow a_n = C - \mu_n$$

ν-SVM

Dual representation

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

$$0 \leqslant a_n \leqslant 1/N$$
$$\sum_{n=1}^{N} a_n t_n = 0, \qquad \sum_{n=1}^{N} a_n \geqslant \nu.$$

Next: Mixture Models and EM