ECE 57000

Kernel method and Support vector machine

Chaoyue Liu

Fall 2024

Linear models have limited performance (e.g., decision boundary is <u>linear</u>)

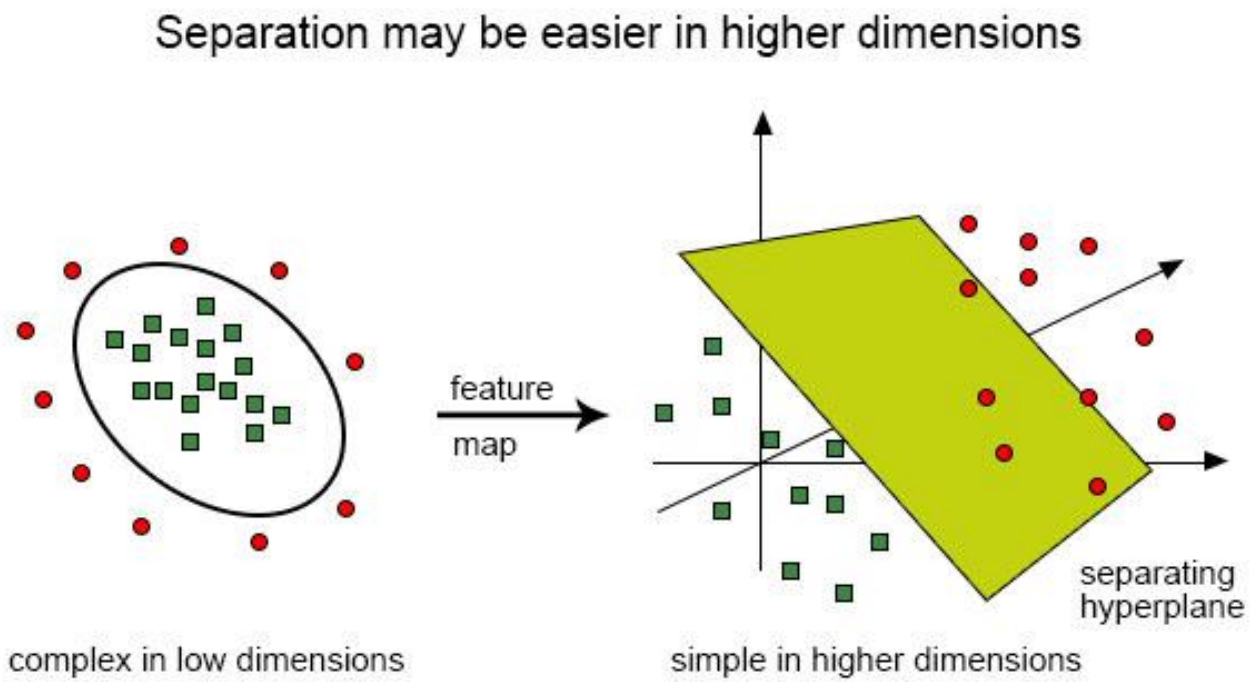
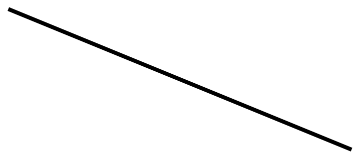
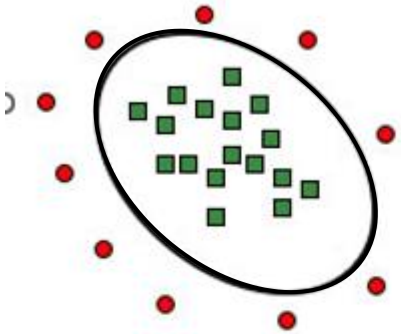


Figure source: https://www.engati.com/glossary/kernel-method

Feature map

	Input space	Feature space
	$\mathbf{x} = (x_1, x_2)^T$	$\phi(\mathbf{x}) = \left(1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, \sqrt{2}x_1x_2, x_2^2\right)^T$
Model	$f_{\mathbf{w}}(\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x})$	$f_{\mathbf{w}'}(\mathbf{x}) = \sigma(\mathbf{w}'^T\phi(\mathbf{x}))$
Decision boundary	$\mathbf{w}^T\mathbf{x} = 0$ $(w_1x_1 + w_2x_2 = 0)$	$\mathbf{w}'^T\phi(\mathbf{x}) = 0$ e.g. $x_1^2 + x_2^2 = 1$
Decision boundary plot		

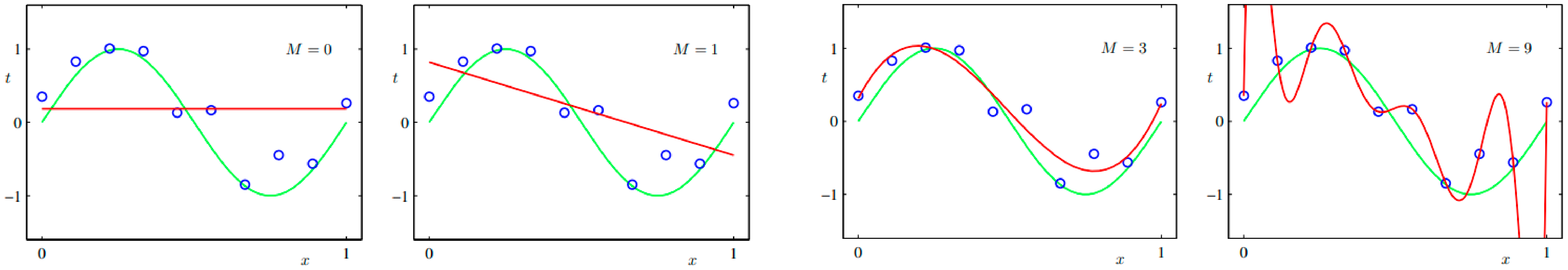
Feature map

Input space: x

Feature space: $\phi(x) = (1, x, x^2, x^3, \cdots, x^M)^T$

Model: $f_{\mathbf{w}'}(x) = \mathbf{w}'^T \phi(x)$

The model is basically a M-degree polynomial of x



Larger M gives better fit of the training data

Downside of feature map

In practice, input $\mathbf{x}$ has a large dimension d. (for example, $d > 1000$)

What if we want the M-degree polynomial feature?

$$\phi(\mathbf{x}) = \underbrace{(1, x_1, \cdots, x_d, x_1^2, x_1 x_2, \cdots, x_1^M, \cdots, x_d^M)}_{d^M}$$

Dimension of the feature vector $\phi(\mathbf{x})$, and weight $\mathbf{w}$: $\sim d^M$ (a huge number)

Downside of feature map

In practice, input $\mathbf{x}$ has a large dimension d. (for example, $d > 1000$)

Dimension of the feature vector $\phi(\mathbf{x})$, and weight $\mathbf{w}$: $\sim d^M$ (a huge number)

Q: Do we really need to compute the whole vectors $\phi(\mathbf{x})$ and $\mathbf{w}$?

A: we only care about the <u>inner product</u> $\mathbf{w}^T\phi(\mathbf{x})$ which is a <u>scalar</u>.

recall: gradient descent: $\mathbf{w}^{t+1} - \mathbf{w}^t = \eta \frac{1}{n}\sum_{i=1}^{n}(f(\mathbf{x}_i) - y_i) \cdot \phi(\mathbf{x}_i)$

$\mathbf{w} = \sum_{i=1}^{n} \alpha_i \cdot \phi(\mathbf{x}_i)$, for some coefficients α_i

$$f(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i \cdot \phi^T(\mathbf{x}_i)\phi(\mathbf{x})$$

We only need the inner products of feature vectors

The "kernel trick"

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

$$\phi(\mathbf{z}) = (1, \sqrt{2}z_1, \sqrt{2}z_2, z_1^2, \sqrt{2}z_1z_2, z_2^2)$$

$$\begin{aligned}\phi^T(\mathbf{x})\phi(\mathbf{z}) &= 1 + 2x_1z_1 + 2x_2z_2 + x_1^2z_1^2 + 2x_1x_2z_1z_2 + x_2^2z_2^2 \\ &= (1 + x_1z_1 + x_2z_2)^2 \\ &= (1 + \mathbf{x}^T\mathbf{z})^2 \quad =: K(\mathbf{x}^T\mathbf{z})\end{aligned}$$

More generally, for M-degree polynomials:

$$\phi(\mathbf{x}) = (1, x_1, \cdots, x_d, x_1^2, x_1x_2, \cdots, x_1^M, \cdots, x_d^M) \qquad \sim d^M \text{ dimensional}$$

$$\phi^T(\mathbf{x})\phi(\mathbf{z}) = K(\mathbf{x}^T\mathbf{z}) = (1 + \mathbf{x}^T\mathbf{z})^M$$

The "kernel trick"

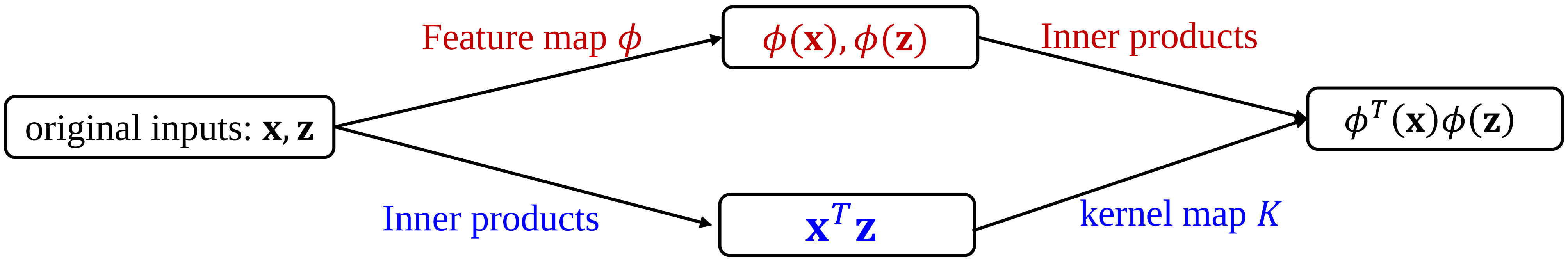
More generally, for *M*-degree polynomials:

$$\phi(\mathbf{x}) = (1, x_1, \cdots, x_d, x_1^2, x_1 x_2, \cdots, x_1^M, \cdots, x_d^M)$$

$$\phi(\mathbf{z}) = (1, z_1, \cdots, z_d, z_1^2, z_1 z_2, \cdots, z_1^M, \cdots, z_d^M)$$

$\sim d^M$ dimensional

$$\phi^T(\mathbf{x})\phi(\mathbf{z}) = K(\mathbf{x}^T\mathbf{z}) = (1 + \mathbf{x}^T\mathbf{z})^M$$



Red: costly
Blue: cheap

The “kernel trick”

$$f(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i \cdot \phi^T(\mathbf{x}_i)\phi(\mathbf{x})$$

$$= \sum_{i=1}^{n} \alpha_i \cdot K(\mathbf{x_i}, \mathbf{x})$$

Instead of learning a high-dimensional weight vector **w**, we just need to learn the coefficients α_i.

The "kernel trick"

One step further:

- we don't even need to know what feature maps are used
- we only need the kernel information

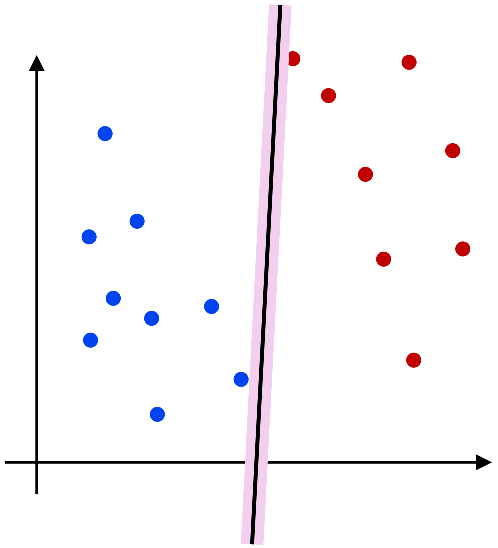
Kernel: $K(\cdot,\cdot)\colon\ \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$

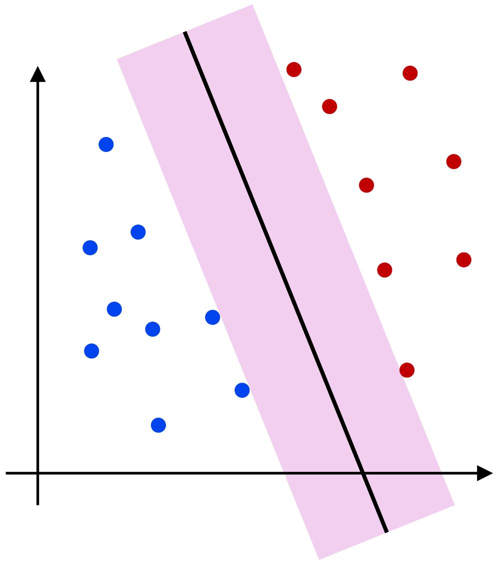
need a pair of inputs

Polynomial kernel: $K(\mathbf{x}, \mathbf{z}) = (\mathbf{1} + \mathbf{x}^T\mathbf{z})^M$
Gaussian kernel: $K(\mathbf{x}, \mathbf{z}) = \exp(-\|\mathbf{x} - \mathbf{z}\|^2/\sigma^2)$
Laplace kernel: $K(\mathbf{x}, \mathbf{z}) = \exp(-\gamma\|\mathbf{x} - \mathbf{z}\|)$

Support vector machine (SVM)

Setting: consider a linearly separable data with two classes





The latter is preferable, because it has a larger margin.

Support vector machine (SVM)

Decision boundary: $\mathbf{w}^T\mathbf{x} + b = 0$

Q: does $(\mathbf{w}, b)$ and $\alpha(\mathbf{w}, b)$ give the same decision boundary?

Normalize ($\mathbf{w}$,b):

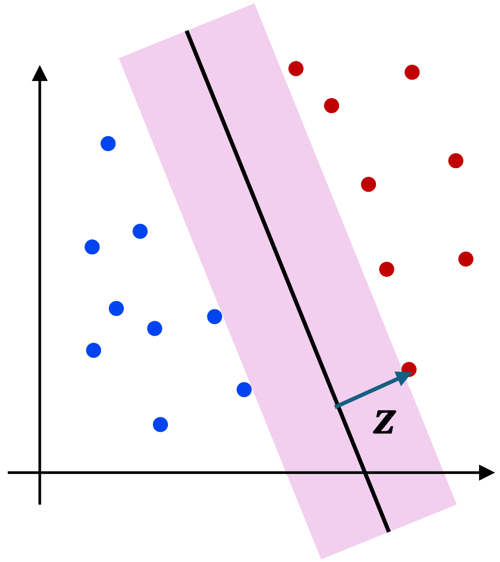
let the closest data points $\mathbf{x}$ satisfy $y \cdot (\mathbf{w}^T\mathbf{x} + b) = 1$, and others with $y' \cdot (\mathbf{w}^T\mathbf{x}' + b) > 1$

Compute the margin:

$$\begin{aligned} \mathbf{w}^T\mathbf{x} + b &= 1 \\ \mathbf{w}^T(\mathbf{x} - \mathbf{z}) + b &= 0 \end{aligned} \longrightarrow \mathbf{w}^T\mathbf{z} = \|\mathbf{w}\|\|\mathbf{z}\| = 1$$

$$\Downarrow$$

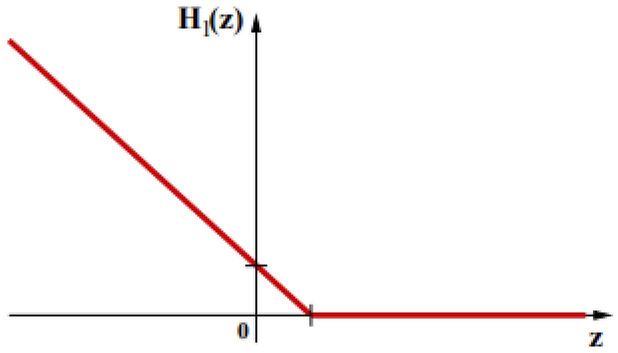
margin: $\|\boldsymbol{z}\| = \frac{1}{\|\mathbf{w}\|}$



Support vector machine (SVM)

Goal: maximize the margin $\frac{1}{\|\mathbf{w}\|}$, equivalent to minimize $\|\mathbf{w}\|^2$

$$\text{minimize } \|\mathbf{w}\|^2$$
$$\text{subject to:} \quad \mathrm{y}_i \cdot (\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \forall i$$



What if the data is <u>not</u> linearly separable?

Soft margin

$$\text{minimize } \|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \xi_i$$
$$\text{subject to:}$$
$$\mathrm{y}_i \cdot (\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \forall i$$
$$\xi_i \geq 0, \forall i$$

Minimize

$$\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} Hinge(\mathrm{y}_i(\mathbf{w}^T\mathbf{x}_i + b))$$

$$Hinge(z) = \max(0,1 - z)$$