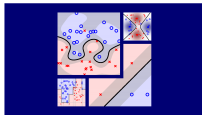


# Machine Learning Techniques (機器學習技法)



## Lecture 3: Kernel Support Vector Machine

Hsuan-Tien Lin (林軒田)

htlin@csie.ntu.edu.tw

Department of Computer Science  
& Information Engineering

National Taiwan University  
(國立台灣大學資訊工程系)



# Roadmap

## ① Embedding Numerous Features: Kernel Models

### Lecture 2: Dual Support Vector Machine

**dual** SVM: another **QP** with **valuable geometric messages** and almost **no dependence on  $\tilde{d}$**

### Lecture 3: Kernel Support Vector Machine

- Kernel Trick
- Polynomial Kernel
- Gaussian Kernel
- Comparison of Kernels

## ② Combining Predictive Features: Aggregation Models

## ③ Distilling Implicit Features: Extraction Models

## Dual SVM Revisited

goal: SVM **without dependence on  $\tilde{d}$**

half-way done:

高維度的矩陣內積仍然藏在  $Q_d$  矩陣裡面！

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2} \alpha^T Q_D \alpha - \mathbf{1}^T \alpha \\ \text{subject to} \quad & \mathbf{y}^T \alpha = 0; \\ & \alpha_n \geq 0, \text{ for } n = 1, 2, \dots, N \end{aligned}$$

- $q_{n,m} = y_n y_m \mathbf{z}_n^T \mathbf{z}_m$ : inner product in  $\mathbb{R}^{\tilde{d}}$
- need:  $\mathbf{z}_n^T \mathbf{z}_m = \Phi(\mathbf{x}_n)^T \Phi(\mathbf{x}_m)$  calculated faster than  $O(\tilde{d})$

簡化 inner product in  $\mathbb{R}^n$

**can we do so?**

我們能不能簡化  $Q$  矩陣裡面的高維度特徵內積運算？

Fast Inner Product for  $\Phi_2$ 

2nd order polynomial transform 二次多項式轉換：

$$\Phi_2(\mathbf{x}) = (1, x_1, x_2, \dots, x_d, x_1^2, x_1x_2, \dots, x_1x_d, x_2x_1, x_2^2, \dots, x_2x_d, \dots, x_d^2)$$

—include both  $x_1x_2$  &  $x_2x_1$  for ‘simplicity’ :-)

我們現在來看看先用這個 $\Phi$ 轉換、再做內積，會不會算得比較快：

$$\begin{aligned} \Phi_2(\mathbf{x})^T \Phi_2(\mathbf{x}') &= 1 + \sum_{i=1}^d x_i x'_i + \sum_{i=1}^d \sum_{j=1}^d x_i x_j x'_i x'_j \\ &= 1 + \sum_{i=1}^d x_i x'_i + \sum_{i=1}^d x_i x'_i \sum_{j=1}^d x_j x'_j \\ &= 1 + \mathbf{x}^T \mathbf{x}' + (\mathbf{x}^T \mathbf{x}') (\mathbf{x}^T \mathbf{x}') \end{aligned}$$

只需要花 $d$ 的力氣就好、不需要花到 $d^2$ 的力氣

for  $\Phi_2$ , transform + inner product can be carefully done in  $O(d)$  instead of  $O(d^2)$

# Kernel: Transform + Inner Product

transform  $\Phi \iff$  kernel function:  $K_{\Phi}(\mathbf{x}, \mathbf{x}') \equiv \Phi(\mathbf{x})^T \Phi(\mathbf{x}')$

$$\Phi_2 \iff K_{\Phi_2}(\mathbf{x}, \mathbf{x}') = 1 + (\mathbf{x}^T \mathbf{x}') + (\mathbf{x}^T \mathbf{x}')^2$$

- quadratic coefficient  $q_{n,m} = y_n y_m \mathbf{z}_n^T \mathbf{z}_m = y_n y_m K(\mathbf{x}_n, \mathbf{x}_m)$
- optimal bias  $b$ ? from SV  $(\mathbf{x}_s, y_s)$ ,

$$b = y_s - \mathbf{w}^T \mathbf{z}_s = y_s - \left( \sum_{n=1}^N \alpha_n y_n \mathbf{z}_n \right)^T \mathbf{z}_s = y_s - \sum_{n=1}^N \alpha_n y_n \left( K(\mathbf{x}_n, \mathbf{x}_s) \right)$$

有了kernel，不需要算到z空間裡面那麼多向量的維度

- optimal hypothesis  $g_{\text{SVM}}$ : for test input  $\mathbf{x}$ ,

$$g_{\text{SVM}}(\mathbf{x}) = \text{sign}(\mathbf{w}^T \Phi(\mathbf{x}) + b) = \text{sign} \left( \sum_{n=1}^N \alpha_n y_n K(\mathbf{x}_n, \mathbf{x}) + b \right)$$

kernel trick: plug in efficient kernel function  
to avoid dependence on  $\tilde{d}$

## Kernel SVM with QP

## Kernel Hard-Margin SVM Algorithm

- ①  $q_{n,m} = y_n y_m K(\mathbf{x}_n, \mathbf{x}_m)$ ;  $\mathbf{p} = -\mathbf{1}_N$ ;  $(\mathbf{A}, \mathbf{c})$  for equ./bound constraints
- ②  $\alpha \leftarrow \text{QP}(\mathbf{Q}_D, \mathbf{p}, \mathbf{A}, \mathbf{c})$
- ③  $b \leftarrow \left( y_s - \sum_{\text{SV indices } n} \alpha_n y_n K(\mathbf{x}_n, \mathbf{x}_s) \right)$  with SV  $(\mathbf{x}_s, y_s)$
- ④ return SVs and their  $\alpha_n$  as well as  $b$  such that for new  $\mathbf{x}$ ,
 
$$g_{\text{SVM}}(\mathbf{x}) = \text{sign} \left( \sum_{\text{SV indices } n} \alpha_n y_n K(\mathbf{x}_n, \mathbf{x}) + b \right)$$

- ①: time complexity  $O(N^2)$  · (kernel evaluation)
- ②: QP with  $N$  variables and  $N + 1$  constraints
- ③ & ④: time complexity  $O(\#\text{SV})$  · (kernel evaluation)

kernel SVM:

use computational shortcut to avoid  $\tilde{d}$  & predict with SV only

# Fun Time

Consider two examples  $\mathbf{x}$  and  $\mathbf{x}'$  such that  $\mathbf{x}^T \mathbf{x}' = 10$ . What is  $K_{\Phi_2}(\mathbf{x}, \mathbf{x}')$ ?

- ① 1
- ② 11
- ③ 111
- ④ 1111

# Fun Time

Consider two examples  $\mathbf{x}$  and  $\mathbf{x}'$  such that  $\mathbf{x}^T \mathbf{x}' = 10$ . What is  $K_{\Phi_2}(\mathbf{x}, \mathbf{x}')$ ?

- ① 1
- ② 11
- ③ 111
- ④ 1111

Reference Answer: ③

Using the derivation in previous slides,

$$K_{\Phi_2}(\mathbf{x}, \mathbf{x}') = 1 + \mathbf{x}^T \mathbf{x}' + (\mathbf{x}^T \mathbf{x}')^2.$$



在每個次項前面的  
係數做放縮的動作

## General Poly-2 Kernel

常用的二次多項式kernel轉換

$$\Phi_2(\mathbf{x}) = (1, x_1, \dots, x_d, x_1^2, \dots, x_d^2) \Leftrightarrow K_{\Phi_2}(\mathbf{x}, \mathbf{x}') = 1 + \mathbf{x}^T \mathbf{x}' + (\mathbf{x}^T \mathbf{x}')^2$$

$$\Phi_2(\mathbf{x}) = (1, \sqrt{2}x_1, \dots, \sqrt{2}x_d, x_1^2, \dots, x_d^2) \Leftrightarrow K_2(\mathbf{x}, \mathbf{x}') = 1 + 2\mathbf{x}^T \mathbf{x}' + (\mathbf{x}^T \mathbf{x}')^2$$

$$\Phi_2(\mathbf{x}) = (1, \sqrt{2\gamma}x_1, \dots, \sqrt{2\gamma}x_d, \gamma x_1^2, \dots, \gamma x_d^2) \\ \Leftrightarrow K_2(\mathbf{x}, \mathbf{x}') = 1 + 2\gamma \mathbf{x}^T \mathbf{x}' + \gamma^2 (\mathbf{x}^T \mathbf{x}')^2$$

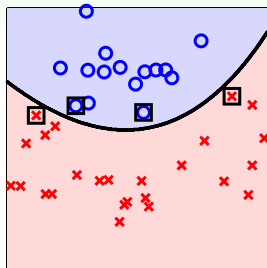
$$K_2(\mathbf{x}, \mathbf{x}') = (1 + \gamma \mathbf{x}^T \mathbf{x}')^2 \text{ with } \gamma > 0$$

- $K_2$ : somewhat 'easier' to calculate than  $K_{\Phi_2}$
- $\Phi_2$  and  $\Phi_2$ : equivalent power, different inner product  $\Rightarrow$  different geometry

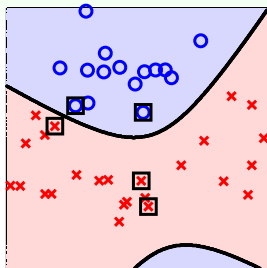
藍色跟綠色的其實對應到的是同樣的空間，所以其實有對等的power，只是幾何解看起來會有點不一樣而已

$K_2$  commonly used

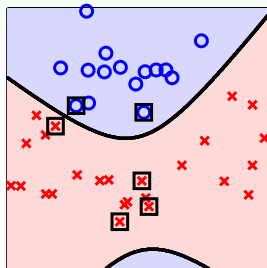
## Poly-2 Kernels in Action



$$(1 + 0.001 \mathbf{x}^T \mathbf{x}')^2$$



$$1 + \mathbf{x}^T \mathbf{x}' + (\mathbf{x}^T \mathbf{x}')^2$$



$$(1 + 1000 \mathbf{x}^T \mathbf{x}')^2$$

- $g_{\text{SVM}}$  **different**, **SVMs different** — ‘hard’ to say which is better before learning  
前面係數scalar的大小只是對距離的定義會不同、算出來的margin可能有所差異，但對於模型表現不影響
- change of **kernel**  $\Leftrightarrow$  change of **margin definition**

need selecting  $K$ , just like selecting  $\Phi$

# General Polynomial Kernel

$$K_2(\mathbf{x}, \mathbf{x}') = (\zeta + \gamma \mathbf{x}^T \mathbf{x}')^2 \text{ with } \gamma > 0, \zeta \geq 0$$

$$K_3(\mathbf{x}, \mathbf{x}') = (\zeta + \gamma \mathbf{x}^T \mathbf{x}')^3 \text{ with } \gamma > 0, \zeta \geq 0$$

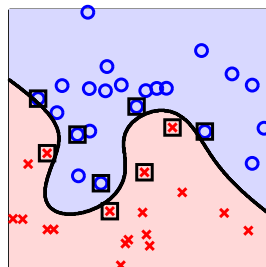
$$\vdots$$

$$K_Q(\mathbf{x}, \mathbf{x}') = (\zeta + \gamma \mathbf{x}^T \mathbf{x}')^Q \text{ with } \gamma > 0, \zeta \geq 0$$

- embeds  $\Phi_Q$  specially with parameters  $(\gamma, \zeta)$   
因為有large margin的幫助，可以多少避免over-fitting的問題
- allows computing large-margin polynomial classification without dependence on  $\tilde{d}$

模型的複雜度可由 $\zeta$ 跟 $Q$ 來做調整！

SVM + Polynomial Kernel: Polynomial SVM

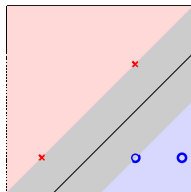


10-th order polynomial  
with margin 0.1

## Special Case: Linear Kernel

$$\begin{aligned}K_1(\mathbf{x}, \mathbf{x}') &= (0 + 1 \cdot \mathbf{x}^T \mathbf{x}')^1 \\&\vdots \\K_Q(\mathbf{x}, \mathbf{x}') &= (\zeta + \gamma \mathbf{x}^T \mathbf{x}')^Q \text{ with } \gamma > 0, \zeta \geq 0\end{aligned}$$

- $K_1$ : just **usual inner product**, called **linear kernel**
- 'even easier': can be solved (often in primal form) **efficiently**



**linear first, remember? :-)**

# Fun Time

Consider the general 2-nd polynomial kernel  $K_2(\mathbf{x}, \mathbf{x}') = (\zeta + \gamma \mathbf{x}^T \mathbf{x}')^2$ . Which of the following transform can be used to derive this kernel?

- 1  $\Phi(\mathbf{x}) = (1, \sqrt{2\gamma}x_1, \dots, \sqrt{2\gamma}x_d, \gamma x_1^2, \dots, \gamma x_d^2)$
- 2  $\Phi(\mathbf{x}) = (\zeta, \sqrt{2\gamma}x_1, \dots, \sqrt{2\gamma}x_d, x_1^2, \dots, x_d^2)$
- 3  $\Phi(\mathbf{x}) = (\zeta, \sqrt{2\gamma\zeta}x_1, \dots, \sqrt{2\gamma\zeta}x_d, x_1^2, \dots, x_d^2)$
- 4  $\Phi(\mathbf{x}) = (\zeta, \sqrt{2\gamma\zeta}x_1, \dots, \sqrt{2\gamma\zeta}x_d, \gamma x_1^2, \dots, \gamma x_d^2)$

# Fun Time

Consider the general 2-nd polynomial kernel  $K_2(\mathbf{x}, \mathbf{x}') = (\zeta + \gamma \mathbf{x}^T \mathbf{x}')^2$ . Which of the following transform can be used to derive this kernel?

- ①  $\Phi(\mathbf{x}) = (1, \sqrt{2\gamma}x_1, \dots, \sqrt{2\gamma}x_d, \gamma x_1^2, \dots, \gamma x_d^2)$
- ②  $\Phi(\mathbf{x}) = (\zeta, \sqrt{2\gamma}x_1, \dots, \sqrt{2\gamma}x_d, x_1^2, \dots, x_d^2)$
- ③  $\Phi(\mathbf{x}) = (\zeta, \sqrt{2\gamma\zeta}x_1, \dots, \sqrt{2\gamma\zeta}x_d, x_1^2, \dots, x_d^2)$
- ④  $\Phi(\mathbf{x}) = (\zeta, \sqrt{2\gamma\zeta}x_1, \dots, \sqrt{2\gamma\zeta}x_d, \gamma x_1^2, \dots, \gamma x_d^2)$

Reference Answer: ④

We need to have  $\zeta^2$  from the 0-th order terms,  $2\gamma\zeta\mathbf{x}^T\mathbf{x}'$  from the 1-st order terms, and  $\gamma^2(\mathbf{x}^T\mathbf{x}')^2$  from the 2-nd order terms.

有沒有辦法做一個無限大維度的kernel轉換??

## Kernel of Infinite Dimensional Transform

infinite dimensional  $\Phi(\mathbf{x})$ ? Yes, if  $K(\mathbf{x}, \mathbf{x}')$  **efficiently computable**!

$$\begin{aligned}
 \text{when } \mathbf{x} = (x), K(x, x') &= \exp(-(x - x')^2) \\
 &= \exp(-(x)^2) \exp(-(x')^2) \exp(2xx') \\
 &\stackrel{\text{Taylor}}{=} \exp(-(x)^2) \exp(-(x')^2) \left( \sum_{i=0}^{\infty} \frac{(2xx')^i}{i!} \right) \\
 &= \sum_{i=0}^{\infty} \left( \exp(-(x)^2) \exp(-(x')^2) \sqrt{\frac{2^i}{i!}} \sqrt{\frac{2^i}{i!}} (x)^i (x')^i \right) \\
 &= \Phi(x)^T \Phi(x')
 \end{aligned}$$

exponential的泰勒展開

with infinite dimensional  $\Phi(x) = \underbrace{\exp(-x^2)}_{\text{某個scaling}} \cdot \left( 1, \sqrt{\frac{2}{1!}}x, \sqrt{\frac{2^2}{2!}}x^2, \dots \right)$

這裡藏了無限多維度，這邊其實是個衰減的數列，所以如果gamma越大衰減越慢、代表後面的多項式越複雜，所以呈現出來的圖形會是個別獨立的小島！

more generally, **Gaussian kernel**

$$K(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2) \text{ with } \gamma > 0$$

# Hypothesis of Gaussian SVM

Gaussian kernel  $K(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2)$

$$\begin{aligned} g_{\text{SVM}}(\mathbf{x}) &= \text{sign} \left( \sum_{\text{SV}} \alpha_n y_n K(\mathbf{x}_n, \mathbf{x}) + b \right) \\ &= \text{sign} \left( \sum_{\text{SV}} \alpha_n y_n \exp(-\gamma \|\mathbf{x} - \mathbf{x}_n\|^2) + b \right) \end{aligned}$$

- linear combination of Gaussians centered at SVs  $\mathbf{x}_n$
- also called Radial Basis Function (RBF) kernel

Gaussian SVM:

find  $\alpha_n$  to combine Gaussians centered at  $\mathbf{x}_n$   
& achieve large margin in infinite-dim. space



# Support Vector Mechanism

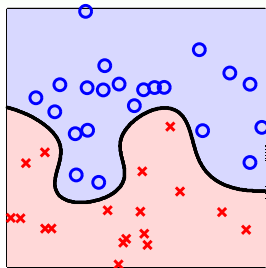
	<b>large-margin hyperplanes</b> <b>+ higher-order transforms with kernel trick</b>
#	<b>not many</b>
boundary	<b>sophisticated</b>

- transformed vector  $\mathbf{z} = \Phi(\mathbf{x}) \implies$  efficient kernel  $K(\mathbf{x}, \mathbf{x}')$
- store optimal  $\mathbf{w} \implies$  store a few SVs and  $\alpha_n$

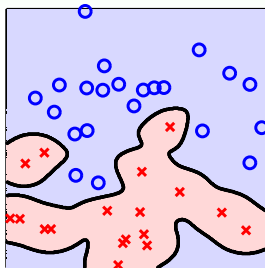
new possibility by Gaussian SVM:  
infinite-dimensional linear classification, with  
generalization 'guarded by' large-margin :-)

# Gaussian SVM in Action

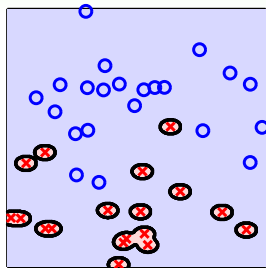
最後的圖形 $g\_svm(x)$ 其實是很多gaussian的線性組合，所以如果 $\gamma$ 越大代表是越多很尖很細的gaussian的線性組合



$$\exp(-1 \|\mathbf{x} - \mathbf{x}'\|^2)$$



$$\exp(-10 \|\mathbf{x} - \mathbf{x}'\|^2)$$



$$\exp(-100 \|\mathbf{x} - \mathbf{x}'\|^2)$$

- large  $\gamma \Rightarrow$  sharp Gaussians  $\Rightarrow$  'overfit'?
- warning: SVM can still overfit :-)

Gaussian SVM: need careful selection of  $\gamma$

# Fun Time

Consider the Gaussian kernel  $K(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2)$ . What function does the kernel converge to if  $\gamma \rightarrow \infty$ ?

- 1  $K_{\text{lim}}(\mathbf{x}, \mathbf{x}') = 0$
- 2  $K_{\text{lim}}(\mathbf{x}, \mathbf{x}') = \mathbb{I}[\mathbf{x} = \mathbf{x}']$
- 3  $K_{\text{lim}}(\mathbf{x}, \mathbf{x}') = \mathbb{I}[\mathbf{x} \neq \mathbf{x}']$
- 4  $K_{\text{lim}}(\mathbf{x}, \mathbf{x}') = 1$

# Fun Time

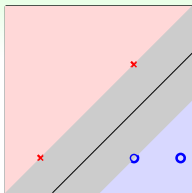
Consider the Gaussian kernel  $K(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2)$ . What function does the kernel converge to if  $\gamma \rightarrow \infty$ ?

- ①  $K_{\text{lim}}(\mathbf{x}, \mathbf{x}') = 0$
- ②  $K_{\text{lim}}(\mathbf{x}, \mathbf{x}') = \mathbb{I}[\mathbf{x} = \mathbf{x}']$
- ③  $K_{\text{lim}}(\mathbf{x}, \mathbf{x}') = \mathbb{I}[\mathbf{x} \neq \mathbf{x}']$
- ④  $K_{\text{lim}}(\mathbf{x}, \mathbf{x}') = 1$

Reference Answer: ②

If  $\mathbf{x} = \mathbf{x}'$ ,  $K(\mathbf{x}, \mathbf{x}') = 1$  regardless of  $\gamma$ . If  $\mathbf{x} \neq \mathbf{x}'$ ,  $K(\mathbf{x}, \mathbf{x}') = 0$  when  $\gamma \rightarrow \infty$ . Thus,  $K_{\text{lim}}$  is an impulse function, which is an extreme case of how the Gaussian gets sharper when  $\gamma \rightarrow \infty$ .

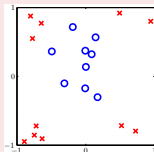
# Linear Kernel: Cons and Pros



$$K(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{x}'$$

## Cons

- restricted  
—**not always separable?!**

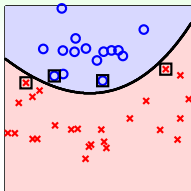


## Pros

- safe—**linear first, remember? :-)**
- fast—with **special QP solver** in primal
- very explainable—**w and SVs** say something

linear kernel: an important **basic** tool

# Polynomial Kernel: Cons and Pros



$$K(\mathbf{x}, \mathbf{x}') = (\zeta + \gamma \mathbf{x}^T \mathbf{x}')^Q$$

## Cons

- **numerical difficulty** for large  $Q$ 
  - $|\zeta + \gamma \mathbf{x}^T \mathbf{x}'| < 1: K \rightarrow 0$
  - $|\zeta + \gamma \mathbf{x}^T \mathbf{x}'| > 1: K \rightarrow \text{big}$
- three parameters ( $\gamma, \zeta, Q$ )  
—**more difficult to select**

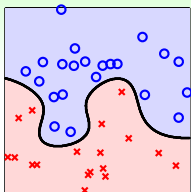
## Pros

- **less restricted** than linear
- strong physical control  
—‘knows’ **degree  $Q$**

主觀認定會是幾次方關係？

polynomial kernel: perhaps **small- $Q$  only**  
—sometimes efficiently done by **linear on  $\Phi_Q(\mathbf{x})$**

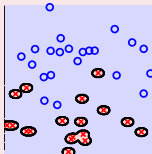
# Gaussian Kernel: Cons and Pros



$$K(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2)$$

## Cons

- **mysterious**—no  $\mathbf{w}$
- **slower** than linear
- **too powerful?!**



## Pros

- **more powerful** than linear/poly.
- bounded—**less numerical difficulty** than poly.
- one parameter only—**easier to select** than poly.

實務上，Gaussian Kernel表現得挺不錯的

Gaussian kernel: **one of most popular** but shall **be used with care**

# Other Valid Kernels

- **kernel** represents **special** similarity:  $\Phi(\mathbf{x})^T \Phi(\mathbf{x}')$
- any similarity  $\implies$  valid kernel? **not really**
- necessary & **sufficient** conditions for valid kernel:  
**Mercer's condition**
  - symmetric
  - let  $k_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$ , the matrix  $\mathbf{K}$

$$\begin{aligned}
 &= \begin{bmatrix} \Phi(\mathbf{x}_1)^T \Phi(\mathbf{x}_1) & \Phi(\mathbf{x}_1)^T \Phi(\mathbf{x}_2) & \dots & \Phi(\mathbf{x}_1)^T \Phi(\mathbf{x}_N) \\ \Phi(\mathbf{x}_2)^T \Phi(\mathbf{x}_1) & \Phi(\mathbf{x}_2)^T \Phi(\mathbf{x}_2) & \dots & \Phi(\mathbf{x}_2)^T \Phi(\mathbf{x}_N) \\ \vdots & \vdots & \ddots & \vdots \\ \Phi(\mathbf{x}_N)^T \Phi(\mathbf{x}_1) & \Phi(\mathbf{x}_N)^T \Phi(\mathbf{x}_2) & \dots & \Phi(\mathbf{x}_N)^T \Phi(\mathbf{x}_N) \end{bmatrix} \\
 &= \begin{bmatrix} \mathbf{z}_1 & \mathbf{z}_2 & \dots & \mathbf{z}_N \end{bmatrix}^T \begin{bmatrix} \mathbf{z}_1 & \mathbf{z}_2 & \dots & \mathbf{z}_N \end{bmatrix} \\
 &= \mathbf{Z}\mathbf{Z}^T \text{ must always be positive semi-definite}
 \end{aligned}$$

define your own kernel: possible, **but hard**



# Fun Time

Which of the following is not a valid kernel? (*Hint: Consider two 1-dimensional vectors  $\mathbf{x}_1 = (1)$  and  $\mathbf{x}_2 = (-1)$  and check Mercer's condition.*)

①  $K(\mathbf{x}, \mathbf{x}') = (-1 + \mathbf{x}^T \mathbf{x}')^2$

②  $K(\mathbf{x}, \mathbf{x}') = (0 + \mathbf{x}^T \mathbf{x}')^2$

③  $K(\mathbf{x}, \mathbf{x}') = (1 + \mathbf{x}^T \mathbf{x}')^2$

④  $K(\mathbf{x}, \mathbf{x}') = (-1 - \mathbf{x}^T \mathbf{x}')^2$

## Fun Time

Which of the following is not a valid kernel? (*Hint: Consider two 1-dimensional vectors  $\mathbf{x}_1 = (1)$  and  $\mathbf{x}_2 = (-1)$  and check Mercer's condition.*)

①  $K(\mathbf{x}, \mathbf{x}') = (-1 + \mathbf{x}^T \mathbf{x}')^2$

②  $K(\mathbf{x}, \mathbf{x}') = (0 + \mathbf{x}^T \mathbf{x}')^2$

③  $K(\mathbf{x}, \mathbf{x}') = (1 + \mathbf{x}^T \mathbf{x}')^2$

④  $K(\mathbf{x}, \mathbf{x}') = (-1 - \mathbf{x}^T \mathbf{x}')^2$

Reference Answer: ①

The kernels in ② and ③ are just polynomial kernels. The kernel in ④ is equivalent to the kernel in ③. For ①, the matrix  $K$  formed from the kernel and the two examples is not positive semi-definite. Thus, the underlying kernel is not a valid one.

# Summary

## ① Embedding Numerous Features: Kernel Models

### Lecture 3: Kernel Support Vector Machine

- Kernel Trick

**kernel as shortcut of transform + inner product**

- Polynomial Kernel

**embeds specially-scaled polynomial transform**

- Gaussian Kernel

**embeds infinite dimensional transform**

- Comparison of Kernels

**linear for efficiency or Gaussian for power**

- **next: avoiding overfitting in Gaussian (and other kernels)**

## ② Combining Predictive Features: Aggregation Models

## ③ Distilling Implicit Features: Extraction Models