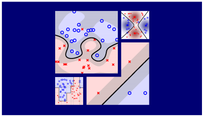
Machine Learning Techniques
(機器學習技法)



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:

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

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

can we do so?

Fast Inner Product for $\mathbf{\Phi}_2$

2nd order polynomial transform

$$\mathbf{\Phi}_2(\mathbf{x}) = (1, x_1, x_2, \ldots, x_d, x_1^2, x_1 x_2, \ldots, x_1 x_d, x_2 x_1, x_2^2, \ldots, x_2 x_d, \ldots, x_d^2)$$

—include both $x_1 x_2$ & $x_2 x_1$ for '**simplicity**' **:-)**

$$\begin{aligned} \mathbf{\Phi}_2(\mathbf{x})^T \mathbf{\Phi}_2(\mathbf{x}') &= 1 + \sum_{i=1}^{d} x_i x_i' + \sum_{i=1}^{d} \sum_{j=1}^{d} x_i x_j x_i' x_j' \\ &= 1 + \sum_{i=1}^{d} x_i x_i' + \sum_{i=1}^{d} x_i x_i' \sum_{j=1}^{d} x_j x_j' \\ &= 1 + \mathbf{x}^T \mathbf{x}' + (\mathbf{x}^T \mathbf{x}')(\mathbf{x}^T \mathbf{x}') \end{aligned}$$

for $\mathbf{\Phi}_2$, transform + inner product can be carefully done in $O(d)$ instead of $O(d^2)$

Kernel: Transform + Inner Product

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

$\boldsymbol{\Phi}_2 \Longleftrightarrow K_{\boldsymbol{\Phi}_2}(\mathbf{x}, \mathbf{x}') = 1 + (\mathbf{x}^T\mathbf{x}') + (\mathbf{x}^T\mathbf{x}')^2$

- 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 $(\mathbf{x}_s, y_s)$,

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

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

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

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

Kernel SVM with QP

Kernel Hard-Margin SVM Algorithm

1. $q_{n,m} = y_n y_m K(\mathbf{x}_n, \mathbf{x}_m)$; $\mathbf{p} = -\mathbf{1}_N$; $(\mathrm{A}, \mathbf{c})$ for equ./bound constraints
2. $\boldsymbol{\alpha} \leftarrow \mathrm{QP}(\mathrm{Q_D}, \mathbf{p}, \mathrm{A}, \mathbf{c})$
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 α_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 $O(N^2) \cdot$ (kernel evaluation)
- (2): QP with N variables and $N+1$ constraints
- (3) & (4): time complexity $O(\#\text{SV}) \cdot$ (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 $K_{\mathbf{\Phi}_2}(\mathbf{x}, \mathbf{x}')$?

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 $K_{\mathbf{\Phi}_2}(\mathbf{x}, \mathbf{x}')$?

1. 1
2. 11
3. 111
4. 1111

Reference Answer: 3

Using the derivation in previous slides,
$K_{\mathbf{\Phi}_2}(\mathbf{x}, \mathbf{x}') = 1 + \mathbf{x}^T\mathbf{x}' + (\mathbf{x}^T\mathbf{x}')^2$.

General Poly-2 Kernel

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

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

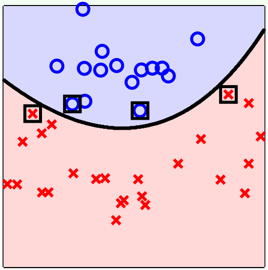
$$\Phi_2(\mathbf{x}) = (1, \sqrt{2\gamma}x_1, \ldots, \sqrt{2\gamma}x_d, \gamma x_1^2, \ldots, \gamma x_d^2)$$
$$\Leftrightarrow K_2(\mathbf{x}, \mathbf{x}') = 1 + 2\gamma\mathbf{x}^T\mathbf{x}' + \gamma^2(\mathbf{x}^T\mathbf{x}')^2$$

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

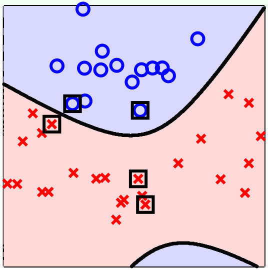
- K_2: somewhat '**easier**' to calculate than K_{Φ_2}
- Φ_2 and Φ_2: equivalent **power**,
  different inner product $\Longrightarrow$ different **geometry**

K_2 commonly used

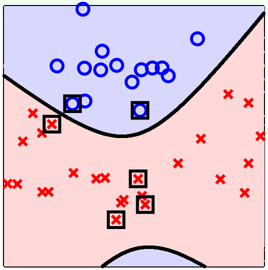
Poly-2 Kernels in Action



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



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



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

- g_{SVM} **different**, $\boxed{\text{SVs}}$ **different**
  —'hard' to say which is better before learning
- change of kernel $\Leftrightarrow$ change of **margin definition**

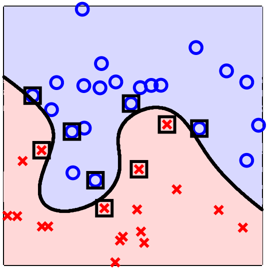
need selecting K, just like selecting $\mathbf{\Phi}$

General Polynomial Kernel

$$
\begin{aligned}
K_2(\mathbf{x}, \mathbf{x}') &= (\zeta + \gamma \mathbf{x}^T \mathbf{x}')^2 \text{ with } \gamma > 0, \zeta \geq 0 \\
K_3(\mathbf{x}, \mathbf{x}') &= (\zeta + \gamma \mathbf{x}^T \mathbf{x}')^3 \text{ with } \gamma > 0, \zeta \geq 0 \\
&\vdots \\
K_Q(\mathbf{x}, \mathbf{x}') &= (\zeta + \gamma \mathbf{x}^T \mathbf{x}')^Q \text{ with } \gamma > 0, \zeta \geq 0
\end{aligned}
$$

- embeds $\mathbf{\Phi}_Q$ specially with parameters (γ, ζ)
- allows computing large-margin polynomial classification without dependence on $\tilde{d}$

SVM + Polynomial Kernel: Polynomial SVM



10-th order polynomial with margin 0.1

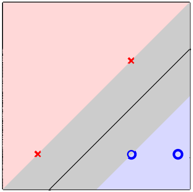
Special Case: Linear Kernel

$$K_1(\mathbf{x}, \mathbf{x}') = (0 + 1 \cdot \mathbf{x}^T \mathbf{x}')^1$$

$$\vdots$$

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

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

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

Reference Answer: 4

We need to have ζ^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.

Kernel of Infinite Dimensional Transform

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

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

with infinite dimensional $\mathbf{\Phi}(x) = \exp(-x^2) \cdot \left(1, \sqrt{\frac{2}{1!}}x, \sqrt{\frac{2^2}{2!}}x^2, \ldots\right)$

more generally, **Gaussian kernel**
$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\gamma\|\mathbf{x} - \mathbf{x}'\|^2\right)$ with $\gamma > 0$

Hypothesis of Gaussian SVM

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

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

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

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

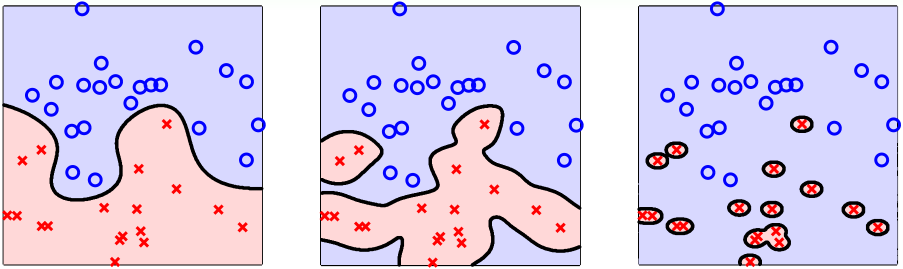
Support Vector Mechanism

	large-margin hyperplanes + higher-order transforms with **kernel trick**
#	**not many**
boundary	**sophisticated**

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

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

Gaussian SVM in Action



$\exp(-1\|\mathbf{x} - \mathbf{x}'\|^2)$ $\exp(-10\|\mathbf{x} - \mathbf{x}'\|^2)$ $\exp(-100\|\mathbf{x} - \mathbf{x}'\|^2)$

- large $\gamma \Longrightarrow$ sharp Gaussians $\Longrightarrow$ 'overfit'?
- **warning: SVM can still overfit :-(**

Gaussian SVM: need careful selection of γ

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 $\gamma \to \infty$?

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

Fun Time

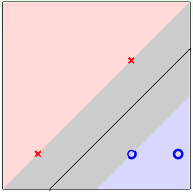
Consider the Gaussian kernel $K(\mathbf{x}, \mathbf{x}') = \exp(-\gamma\|\mathbf{x} - \mathbf{x}'\|^2)$. What function does the kernel converge to if $\gamma \to \infty$?

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

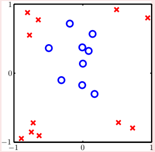
Linear Kernel: Cons and Pros



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

Cons

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

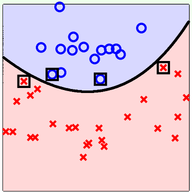


Pros

- 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

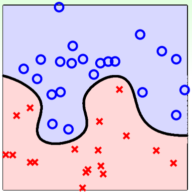
- **numerical difficulty** for large Q
  - $|\zeta + \gamma \mathbf{x}^T \mathbf{x}'| < 1$: $K \to 0$
  - $|\zeta + \gamma \mathbf{x}^T \mathbf{x}'| > 1$: $K \to$ big
- three parameters (γ, ζ, Q) —**more difficult to select**

Pros

- **less restricted** than linear
- strong physical control —'knows' **degree Q**

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

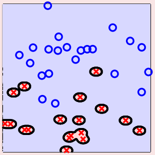
Gaussian Kernel: Cons and Pros



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

Cons

- **mysterious**—no **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.**

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

Other Valid Kernels

- kernel represents **special** similarity: $\mathbf{\Phi}(\mathbf{x})^T \mathbf{\Phi}(\mathbf{x}')$
- any similarity $\Longrightarrow$ 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 K

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

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