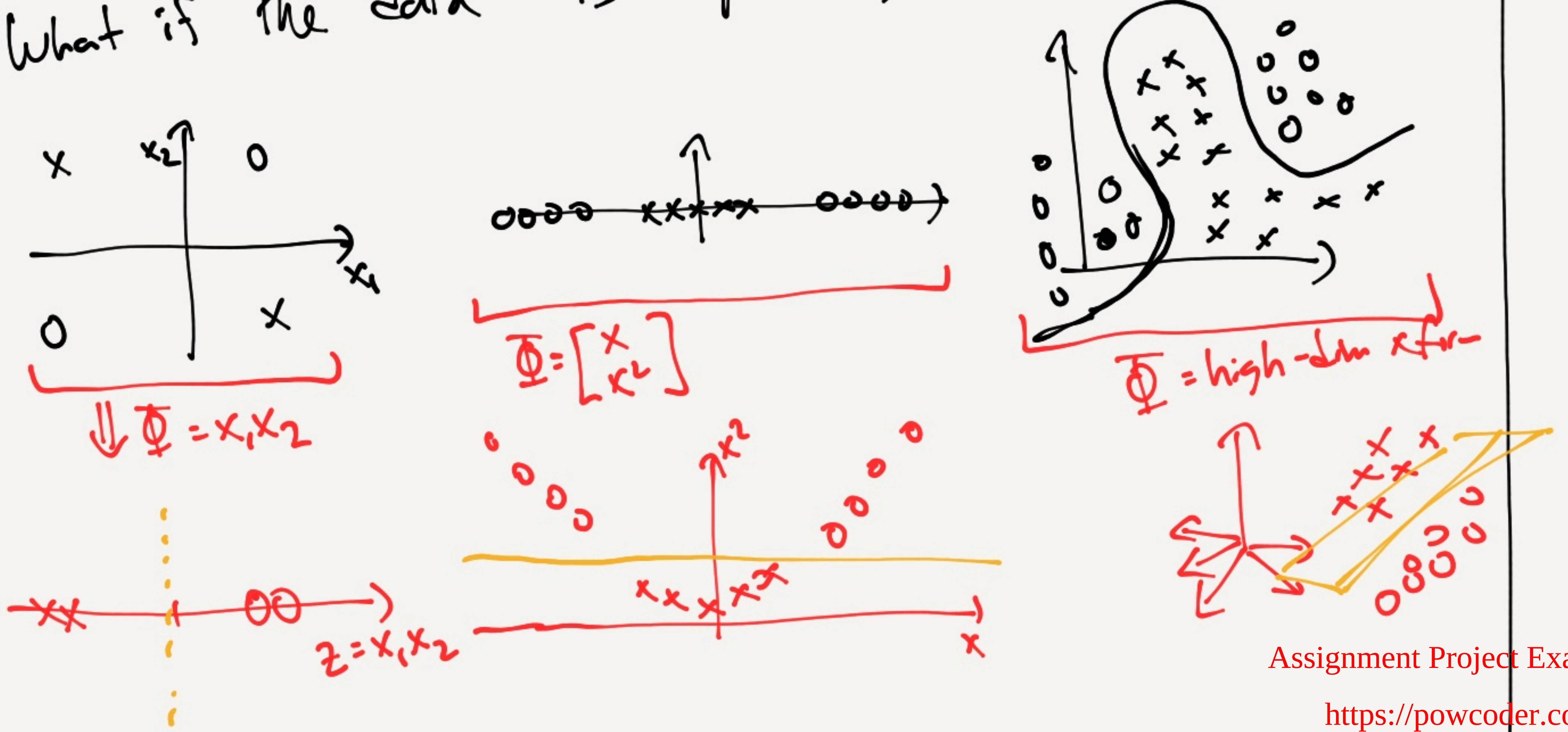
CS5487 Lecture Notes (2020)
Dr. Antoni B. Chan
Dept of Computer Science
City University of Hong Kong

Lecture 10 Nonlinear Classifiers & Kernels

- So far, we only saw linear discriminative classifiers (SVM, LR, perceptron, LSC, ...)
- What if the data is separable, but not linearly separable ?



- IDEA - transform the input space
  - mapping: $\Phi : \mathcal{X} \to \mathcal{Z}$
  - learn linear classifier in new space ($\mathcal{Z}$)
- if the new dimension is large enough, the data should be separable by a hyperplane.
  $\dim(\mathcal{Z}) > \dim(\mathcal{X})$
- in the limit, $\dim(\mathcal{Z}) \to \infty$
  - we are mapping a vector $x_i \to$ into a function

$$\Phi(x) = \begin{bmatrix} \phi_1(x) \\ \vdots \\ \phi_B(x) \end{bmatrix} \to \phi(x;t) \qquad x \to e^{-\frac{1}{2}(z-x)^2}$$

Kernel SVM

- consider the SVM dual problem $\to$ replace x_i with $\Phi(x_i)$

Training

$$\max_{\alpha} \sum_i \alpha_i - \frac{1}{2}\sum_i\sum_j \alpha_i\alpha_j y_i y_j \underbrace{\Phi(x_i)^T\Phi(x_j)}_{K(x_i,x_j)}$$

$$\text{s.t. } \sum_i \alpha_i y_i = 0, \quad \alpha_i \geq 0 \ \forall i$$

Decision boundary

$$w^* = \sum_i \alpha_i y_i \Phi(x_i)$$

w has same dimension as $\Phi(\cdot)$

Bias term

$$b^* = \frac{1}{|SV|}\sum_{i\in SV}(y_i - w^T\Phi(x_i))$$

$$= \frac{1}{|SV|}\sum_{i\in SV}\Big(y_i - \sum_j \alpha_j y_j \underbrace{\Phi(x_j)^T\Phi(x_i)}_{K(x_i,x_j)}\Big)$$

$SV = \{x_i \mid \alpha_i > 0\}$ (support vector set)

Decision function

$$y_* = \text{sign}(f(x_*))$$

$$f(x_*) = w^T\Phi(x_*) + b = \sum_i \alpha_i y_i \underbrace{\Phi(x_i)^T\Phi(x_*)}_{K(x_i,x_*)} + b$$

Note: The whole algorithm only depends on the data through $\Phi(x_i)^T\Phi(x_j)$!!!

- Define a kernel function: (dot-product kernel)

$$K(x_i,x_j) = \Phi(x_i)^T\Phi(x_j) = \langle \Phi(x_i), \Phi(x_j)\rangle$$

- The dual SVM can be written w/ this kernel function → nonlinear classifier.

- Actually, we only need the kernel function, not the xform $\Phi(x)$.
- Just define the kernel ⇒ called the "kernel trick"

Why is it good?

- save the calculation of $\Phi(x_i)$ (could be a large vector)
- Need to calculate $O(n^2)$ terms of $k(x_i, x_j)$

⇒ kernel matrix:

$$K = \begin{bmatrix} k(x_1,x_1) & \cdots & k(x_1,x_n) \\ \vdots & \ddots & \vdots \\ k(x_n,x_1) & \cdots & k(x_n,x_n) \end{bmatrix}$$

(Gram matrix)

Example: polynomial kernel

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix} \in \mathbb{R}^d$$

$$k(x,x') = (x^T x')^2 = \left(\sum_{i=1}^d x_i x_i'\right)^2 = \sum_{i=1}^d \sum_{j=1}^d x_i x_i' x_j x_j'$$

$$= \sum_i \sum_j (x_i x_j)(x_i' x_j')$$

$$= \underbrace{[x_1x_1 \quad x_1x_2 \quad \cdots \quad x_dx_d]}_{\Phi(x)} \underbrace{\begin{bmatrix} x_1'x_1' \\ x_1'x_2' \\ \vdots \\ x_d'x_d' \end{bmatrix}}_{\Phi(x')}$$

$$= \Phi(x)^T \Phi(x')$$

$\Phi: \mathbb{R}^d \to \mathbb{R}^{d^2} \to O(d^2)$

$k(x,x') = \underbrace{(x^T x')^2}_{d} \to O(d)$ ← more efficient than explicitly computing $\Phi(x)$

$\underbrace{}_{d+1}$

- what about the classifier?

$$f(x_*) = \sum_i \alpha_i y_i k(x_i, x_*) + b = \sum_i \alpha_i y_i (x_i^T x_*)^2 + b$$

$$= \sum_i \alpha_i y_i (x_*^T x_i x_i^T x_*) + b = x_*^T \overbrace{\left(\sum_i \alpha_i y_i x_i x_i^T\right)}^{A} x_* + b$$

$$= x_*^T A x_* + b$$ ← quadratic function in x_*.

* the kernel function specifies the class of non linear functions that are learned.

Kernel Functions

"kernel trick" depends on $k(x,x')$ being a dot-product kernel.

Define: mapping $k: \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a dot-product kernel iff

$$k(x,x') = \langle \Phi(x), \Phi(x') \rangle$$

where $\Phi: \mathcal{X} \to \mathcal{H}$ (vector space)

$\langle \cdot, \cdot \rangle$ is a dot product in $\mathcal{H}$.

How to check if $k(x,x')$ is a dot product kernel w/o knowing Φ, $\langle \cdot, \cdot \rangle$?

Define: $k(x,x')$ is a positive-definite kernel on $\mathcal{X} \times \mathcal{X}$ iff $\forall n$ and $\forall \{x_1, \dots, x_n\}$, $x_i \in \mathcal{X}$ the kernel matrix $K = [k(x_i,x_j)]_{ij}$ is a positive definite matrix. (for all possible datasets, the kernel matrix is pos def.)

K is posdef iff
1) $y^T K y > 0 \quad \forall y$
2) eigenvalues of K are positive
3) $K = AA^T$, A has independent columns.

✱✱✱ $\boxed{k(x,x') \text{ is a dotproduct kernel iff it is a posdef kernel.}}$

Given a posdef / dot product kernel, what is the high-dim xformation Φ?

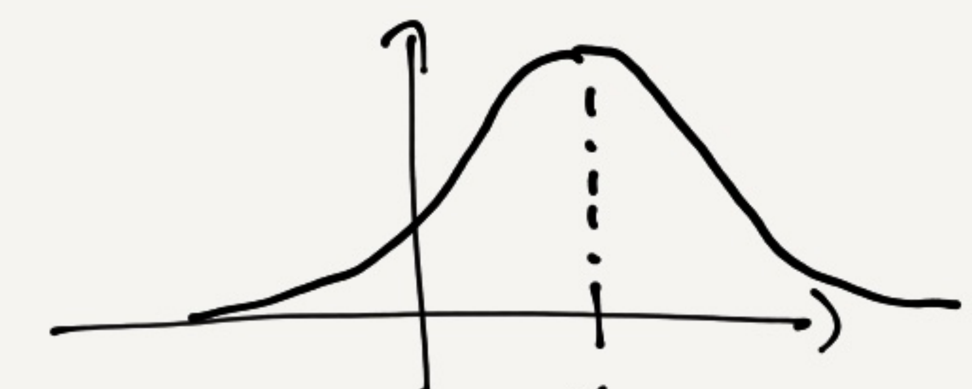
let $\mathcal{H}$ = space of all linear combinations of function $k(\cdot, x_i)$.

function of $\cdot$ x_i is fixed

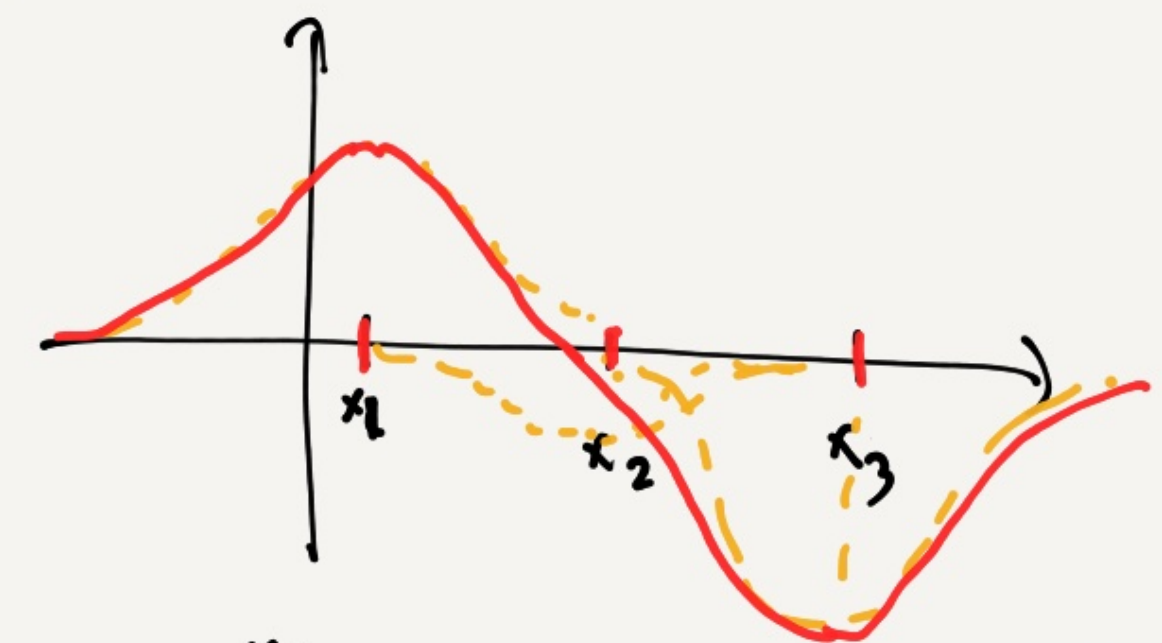
$$\mathcal{H} = \left\{ f(\cdot) \;\middle|\; f(\cdot) = \sum_{i=1}^{m} \alpha_i k(\cdot, x_i),\ \forall m,\ \forall x_i \in \mathcal{X} \right\}$$

e.g. use a Gaussian kernel:

$$k(\cdot, x_i) = e^{-\frac{1}{\sigma^2}\|\cdot - x_i\|^2}$$



$f(\cdot)$ = combinations of Gaussians



• Let $f(\cdot) = \sum_{i=1}^{m} \alpha_i k(\cdot, x_i)$, $\quad g(\cdot) = \sum_j \beta_j k(\cdot, x_j)$

We can show the dot-product btwn f, g is

$$\langle f, g \rangle = \sum_i \sum_j \alpha_i \beta_j k(x_i, x_j)$$

e.g. Gaussian kernel.

$$\langle f, g \rangle = \sum_i \sum_j \alpha_i \beta_j e^{-\frac{1}{\sigma^2}\|x_i - x_j\|^2}$$

Special case

$$\left.\begin{array}{l}\alpha_i = 1, \alpha_{\neg i} = 0 \\ \beta_j = 1, \beta_{\neg j} = 0\end{array}\right\} \Rightarrow \langle \underbrace{K(\cdot, x_i)}_{\Phi(x_i)}, \underbrace{K(\cdot, x_j)}_{\Phi(x_j)} \rangle = K(x_i, x_j)$$

Hence

$K(x, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$

where $\Phi: \mathcal{X} \to \mathcal{H}$

$x_i \to \Phi(x_i) = K(\cdot, x_i)$

transformation from vector x to a function (inf. dim space)

Final Note

given $f(\cdot) = \sum_i \alpha_i K(\cdot, x_i)$

then $\underbrace{\langle K(\cdot, x), f(\cdot) \rangle}$ $= \sum_i \alpha_i K(x_i, x) = \underbrace{f(x)}$

"reproducing property" - dot product of f w/ kernel gives back the f.

(similar to dirac delta & convolution)

$\mathcal{H}$ is called a "Reproducing Kernel Hilbert Space." (RKHS)

(Hilbert space = vector space & dot product & ....)

- RKHS uniquely specifies the kernel function & vice versa.
- Posdef kernels are also called Mercer kernels

Representer Theorem

Empirical Risk: $R_{emp} = \sum_{i=1}^{n} L(y_i, f(x_i))$

Regularizer: $\Omega(\|f\|_p)$, $\Omega \geq 0$ & strictly monotonically increasing.

$\mathcal{H}, K(x, x')$

$f^* = \underset{f}{\operatorname{argmin}}\ R_{emp}(f) + \lambda \Omega(\|f\|_p)$

then f^* has the form:

$f^* = \sum_i \alpha_i K(x, x_i)$, $f^* \in \mathcal{H}$ (RKHS)

(Many ML algorithms fit this framework
→ similar form of f
→ they are kernelizable (make it nonlinear)
→ inf. dim space of f → finite dim α.)

Kernel functions

Kernels on $\mathbb{R}^d$

linear: $k(x,x') = x^T x'$

$k(x,x') = x^T A x'$, A posdef

poly: $k(x,x') = (x^T x' + c)^d$

$k(x,x') = (x^T x')^d$

Gaussian: $k(x,x') = e^{-\frac{1}{\gamma}\|x-x'\|^2}$ ← Radial Basis Function (RBF)

Expo: $k(x,x') = e^{-\frac{1}{\gamma}\|x-x'\|}$

- can combine kernels

$$K(x,x') = \underbrace{x^T x'}_{\text{linear}} + \underbrace{e^{-\frac{1}{\gamma}\|x-x'\|^2}}_{\text{Gaussian (RBF)}}$$

see PS for the rules.

Kernel on histograms

x, x' are histograms

correlation kernel: $k(x,x') = x^T x' = \sum_{i=1}^{d} x_i x_i'$

Bhattacharyya kernel: $k(x,x') = \sum_{i=1}^{d} \sqrt{x_i}\sqrt{x_i'}$

χ-squared kernel: $k(x,x') = e^{-\frac{1}{\sigma^2}\chi^2(x,x')}$
RBF

$$\chi^2(x,x') = \sum_i \frac{(x_i - x_i')^2}{\frac{1}{2}(x_i + x_i')}$$

histogram intersection kernel: $k(x,x') = \sum_{i=1}^{d} \min(x_i, x_i')$

Kernels on Sets

$X = \{x_1, \dots, x_n\}$

$X' = \{x_1', \dots, x_m'\}$

$$\overbrace{|X \cap X'|}^{\#\text{ of common elements}}$$

intersection kernel: $k(X,X') = 2$

distance kernel: $k(X,X') = e^{-\frac{1}{\gamma}\sum_i \sum_j d(x_i, x_j')}$

pyramid match kernel:

Kernels on strings / trees / graphs

$K(x,x') = \sum_s w_s \phi_s(x) \phi_s(x')$

$w_s \geq 0$

# of times substring s appears in x'.

☆ Kernels allow us to learn classifiers directly on non-vector data.

⇒ need to define the appropriate PD kernel.