

Kernel methods

Shikui Tu

**Department of Computer Science and
Engineering, Shanghai Jiao Tong University**

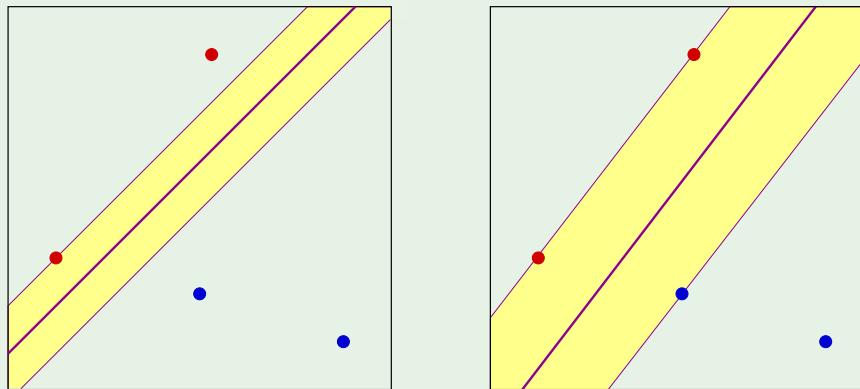
2018-05-03

Outline

- **Review of SVM in the previous lecture**
- A bit of history of kernel methods and SVM
- Kernel methods
- Kernel PCA
- Designing kernels

Maximize margin

- The margin

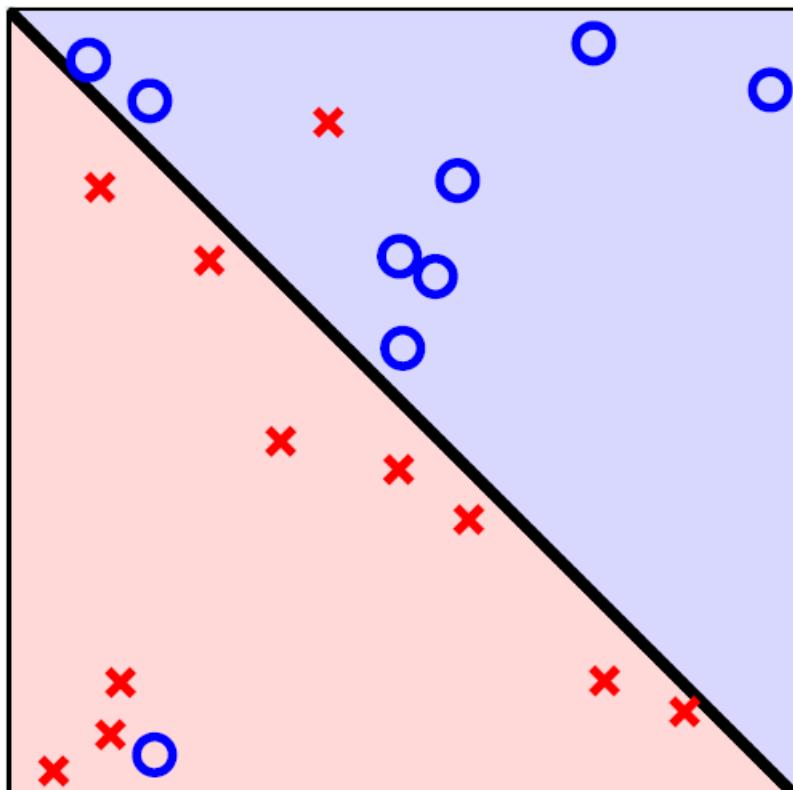


Maximizing the margin \implies dual problem:

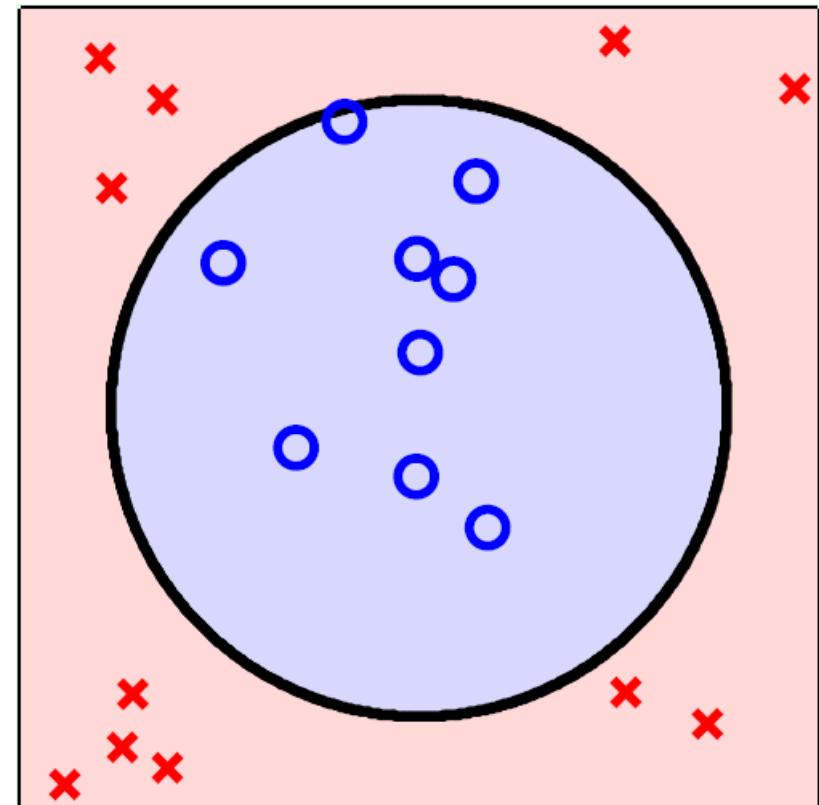
$$\mathcal{L}(\boldsymbol{\alpha}) = \sum_{n=1}^N \alpha_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N y_n y_m \alpha_n \alpha_m \mathbf{x}_n^\top \mathbf{x}_m$$

quadratic programming

Non-separable Data



Slightly
non-separable

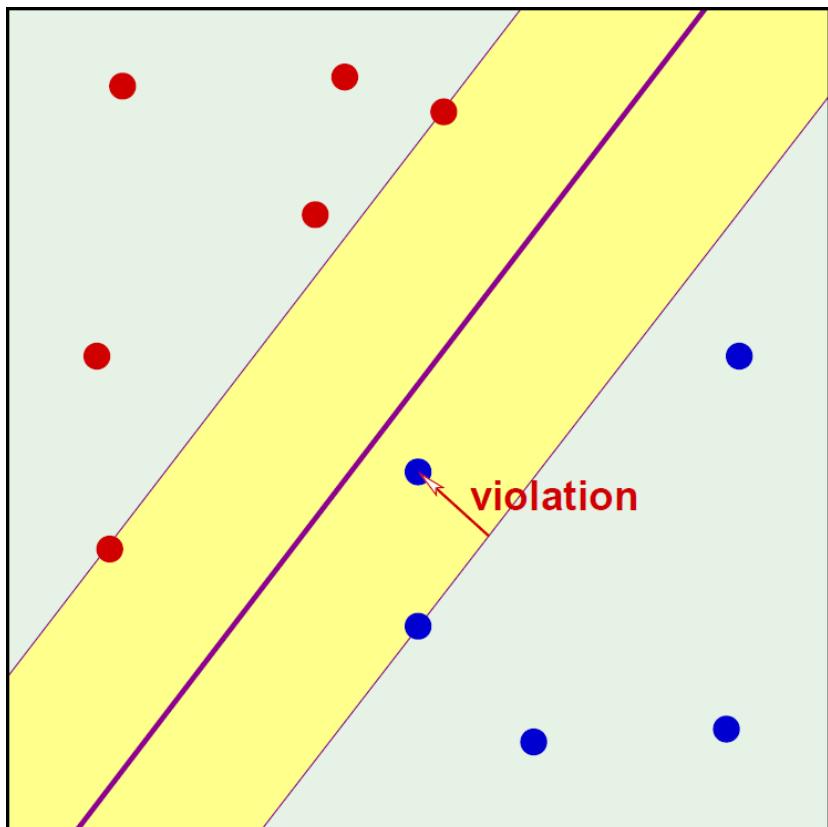


Seriously
non-separable

Soft-margin SVM

Margin violation: $y_n (\mathbf{w}^\top \mathbf{x}_n + b) \geq 1$ fails

Quantify: $y_n (\mathbf{w}^\top \mathbf{x}_n + b) \geq 1 - \xi_n$



$$\text{Total violation} = \sum_{n=1}^N \xi_n$$

ξ_n = ksi = “amount” of margin violation

$$\boxed{\xi_n \geq 0}$$

Soft-margin SVM optimization

- Similar to hard-margin optimization, but get compromise between maximizing the margin and allowing for violations

Still get large margin after term minimization

Allow for small violations by minimizing

Minimize

$$\frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{n=1}^N \xi_n$$

subject to

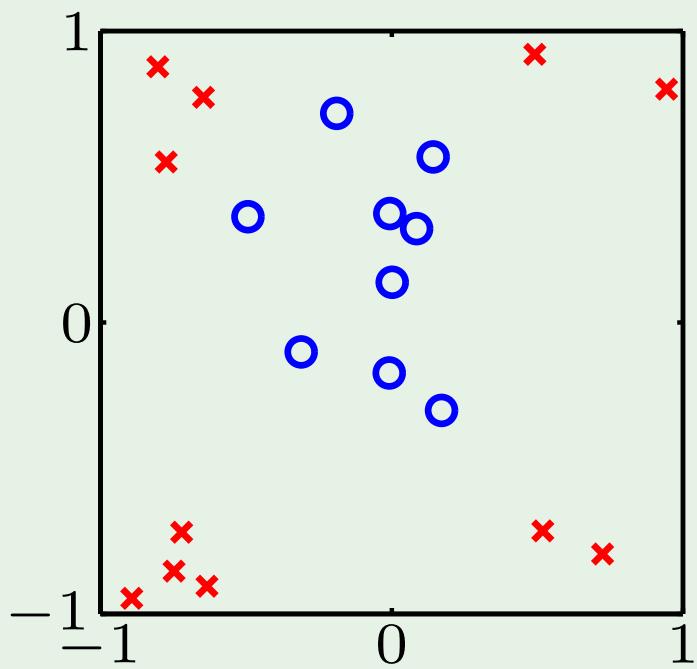
$$y_n (\mathbf{w}^T \mathbf{x}_n + b) \geq 1 - \xi_n$$

for $n = 1, \dots, N$

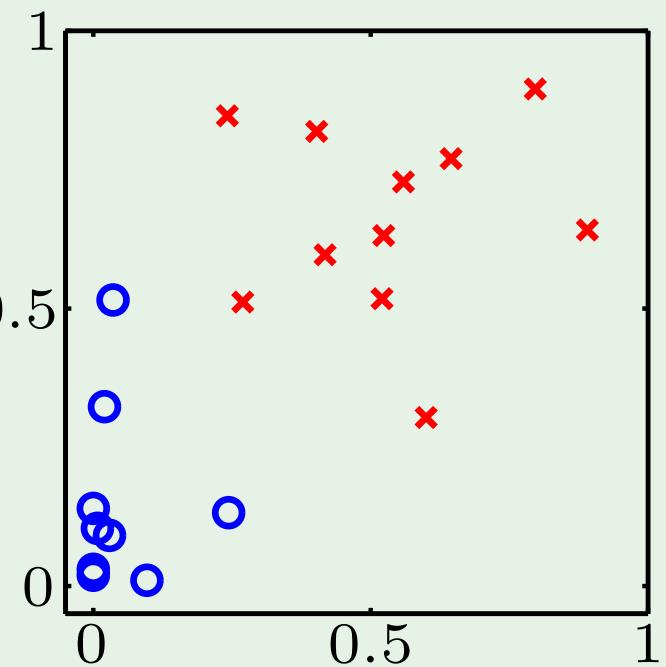
and $\xi_n \geq 0$ for $n = 1, \dots, N$

Kernel transform

$$\mathcal{L}(\boldsymbol{\alpha}) = \sum_{n=1}^N \alpha_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N y_n y_m \alpha_n \alpha_m \mathbf{z}_n^\top \mathbf{z}_m$$



$\mathcal{X} \longrightarrow \mathcal{Z}$



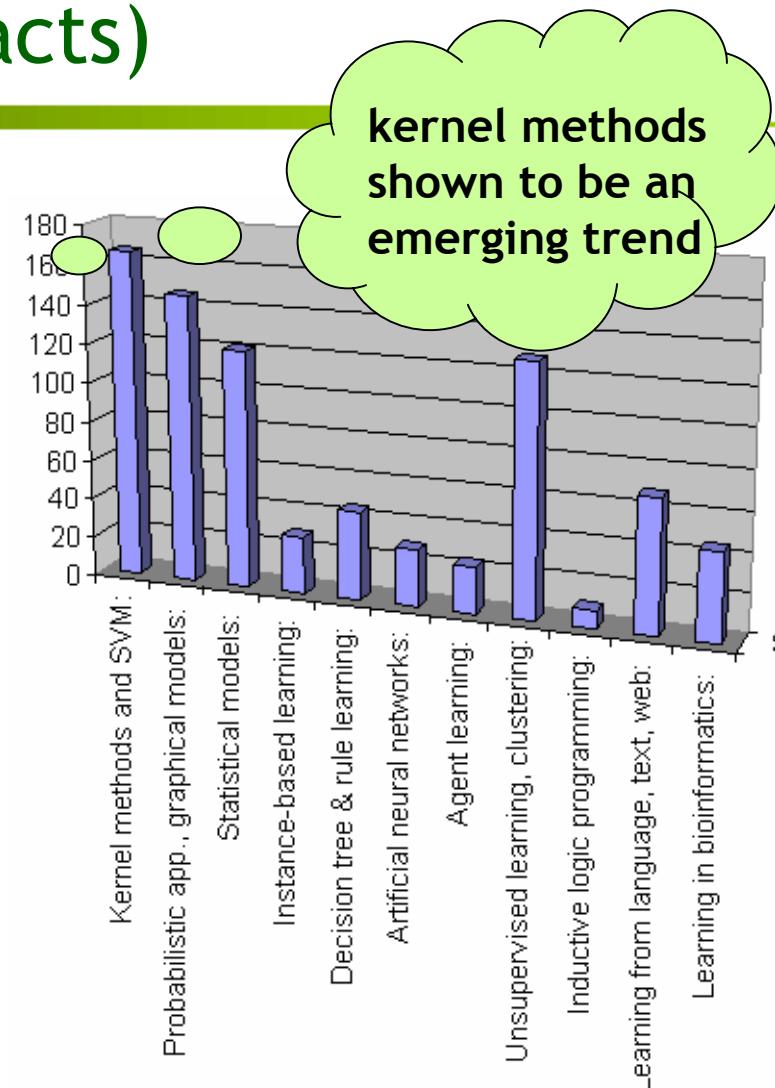
Outline

- Review of SVM in the previous lecture
- **A bit of history of kernel methods and SVM**
- Kernel methods
- Kernel PCA
- Designing kernels

Very popular around 2006

ICML 2006 (720 abstracts)

Kernel methods & SVM	166
Probabilistic, graphical models	146
Unsupervised learning, clustering	128
Statistical models	121
Language, Text & web	68
Learning in bioinformatics	45
ANN	29
ILP	9
CRF	13



A bit of history

- Aronszajn (1950) and Parzen (1962) were some of the first to employ positive definite **kernels** in statistics.
- Aizerman et al. (1964) used positive definite kernels in a way closer to the **kernel trick**.
- Boser et al. (1992) constructed the SVMs, a generalization of optimal hyperplane algorithm worked for **vectorial data**.
- Scholkopf (1997): kernels can work with **nonvectorial data** by providing a vectorial representation of the data in the feature space.
- Scholkopf et al. (1998): kernels can be used to build generalizations of any algorithm that can be carried out in terms of **dot products**.
- A large number of “**kernelizations**” of various algorithms since last 5 years.

Computation from the Z space

$$\mathcal{L}(\alpha) = \sum_{n=1}^N \alpha_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N y_n y_m \alpha_n \alpha_m \mathbf{z}_n^\top \mathbf{z}_m$$

Constraints: $\alpha_n \geq 0$ for $n = 1, \dots, N$ and $\sum_{n=1}^N \alpha_n y_n = 0$



$$g(\mathbf{x}) = \text{sign}(\mathbf{w}^\top \mathbf{z} + b)$$

need $\mathbf{z}_n^\top \mathbf{z}$

where $\mathbf{w} = \sum_{\mathbf{z}_n \text{ is SV}} \alpha_n y_n \mathbf{z}_n$

and b : $y_m (\mathbf{w}^\top \mathbf{z}_m + b) = 1$ need $\mathbf{z}_n^\top \mathbf{z}_m$

Generalized inner product

Given two points \mathbf{x} and $\mathbf{x}' \in \mathcal{X}$, we need $\mathbf{z}^\top \mathbf{z}'$

Let $\mathbf{z}^\top \mathbf{z}' = K(\mathbf{x}, \mathbf{x}')$ (the kernel) “inner product” of \mathbf{x} and \mathbf{x}'

Second order transform:

Example: $\mathbf{x} = (x_1, x_2) \longrightarrow$ 2nd-order Φ

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

$$K(\mathbf{x}, \mathbf{x}') = \mathbf{z}^\top \mathbf{z}' = 1 + x_1 x'_1 + x_2 x'_2 +$$

$$x_1^2 x'^2_1 + x_2^2 x'^2_2 + x_1 x'_1 x_2 x'_2$$

Without transforming explicitly

- Can we compute $K(\mathbf{x}, \mathbf{x}')$ without transforming \mathbf{x} and \mathbf{x}' ?

Example:

$$\begin{aligned} K(\mathbf{x}, \mathbf{x}') &= (1 + \mathbf{x}^\top \mathbf{x}')^2 = (1 + x_1 x'_1 + x_2 x'_2)^2 \\ &= 1 + x_1^2 x'^2_1 + x_2^2 x'^2_2 + 2x_1 x'_1 + 2x_2 x'_2 + 2x_1 x'_1 x_2 x'_2 \end{aligned}$$



Inner product

$$(1, x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2)$$

$$(1, x'^2_1, x'^2_2, \sqrt{2}x'_1, \sqrt{2}x'_2, \sqrt{2}x'_1x'_2)$$

The polynomial kernel

$\mathcal{X} = \mathbb{R}^d$ and $\Phi : \mathcal{X} \rightarrow \mathcal{Z}$ is polynomial of order Q

The “equivalent” kernel $K(\mathbf{x}, \mathbf{x}') = (1 + \mathbf{x}^\top \mathbf{x}')^Q$

$$= (1 + x_1 x'_1 + x_2 x'_2 + \cdots + x_d x'_d)^Q$$

Adjust the scales:

$$K(\mathbf{x}, \mathbf{x}') = (a \mathbf{x}^\top \mathbf{x}' + b)^Q$$

We only need Z to exist

- If $K(x, x')$ is an inner product in some space Z, that is ok.
- For example,

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

Infinite-dimensional \mathcal{Z} : take simple case

$$\begin{aligned} K(x, x') &= \exp(-(x - x')^2) \\ &= \exp(-x^2) \exp(-x'^2) \underbrace{\sum_{k=0}^{\infty} \frac{2^k (x)^k (x')^k}{k!}}_{\exp(2xx')} \end{aligned}$$

- How do we know whether Z exists or not?
 1. By construction
 2. Math properties (Mercer's condition)
 3. Who cares?

Kernel SVM

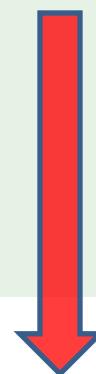
$$\min_{\alpha} \frac{1}{2} \alpha^\top \begin{bmatrix} y_1y_1 \mathbf{x}_1^\top \mathbf{x}_1 & y_1y_2 \mathbf{x}_1^\top \mathbf{x}_2 & \dots & y_1y_N \mathbf{x}_1^\top \mathbf{x}_N \\ y_2y_1 \mathbf{x}_2^\top \mathbf{x}_1 & y_2y_2 \mathbf{x}_2^\top \mathbf{x}_2 & \dots & y_2y_N \mathbf{x}_2^\top \mathbf{x}_N \\ \dots & \dots & \dots & \dots \\ y_Ny_1 \mathbf{x}_N^\top \mathbf{x}_1 & y_Ny_2 \mathbf{x}_N^\top \mathbf{x}_2 & \dots & y_Ny_N \mathbf{x}_N^\top \mathbf{x}_N \end{bmatrix} \alpha + \underbrace{(-1^\top) \alpha}_{\text{linear}}$$

quadratic coefficients

subject to $\underbrace{\mathbf{y}^\top \alpha = 0}_{\text{linear constraint}}$

$\underbrace{0}_{\text{lower bounds}} \leq \alpha \leq \underbrace{\infty}_{\text{upper bounds}}$

Kernel formulation



$$\begin{bmatrix} y_1y_1 K(\mathbf{x}_1, \mathbf{x}_1) & y_1y_2 K(\mathbf{x}_1, \mathbf{x}_2) & \dots & y_1y_N K(\mathbf{x}_1, \mathbf{x}_N) \\ y_2y_1 K(\mathbf{x}_2, \mathbf{x}_1) & y_2y_2 K(\mathbf{x}_2, \mathbf{x}_2) & \dots & y_2y_N K(\mathbf{x}_2, \mathbf{x}_N) \\ \dots & \dots & \dots & \dots \\ y_Ny_1 K(\mathbf{x}_N, \mathbf{x}_1) & y_Ny_2 K(\mathbf{x}_N, \mathbf{x}_2) & \dots & y_Ny_N K(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix}$$

quadratic coefficients

Solution to kernel SVM

Express $g(\mathbf{x}) = \text{sign}(\mathbf{w}^\top \mathbf{z} + b)$ in terms of $K(-, -)$

$$\mathbf{w} = \sum_{\mathbf{z}_n \text{ is SV}} \alpha_n y_n \mathbf{z}_n \implies g(\mathbf{x}) = \text{sign} \left(\sum_{\alpha_n > 0} \alpha_n y_n K(\mathbf{x}_n, \mathbf{x}) + b \right)$$

$$\text{where } b = y_m - \sum_{\alpha_n > 0} \alpha_n y_n K(\mathbf{x}_n, \mathbf{x}_m)$$

for any support vector ($\alpha_m > 0$)

Mercer's condition

$K(\mathbf{x}, \mathbf{x}')$ is a valid kernel iff

1. It is symmetric and 2. The matrix:

$$\begin{bmatrix} K(\mathbf{x}_1, \mathbf{x}_1) & K(\mathbf{x}_1, \mathbf{x}_2) & \dots & K(\mathbf{x}_1, \mathbf{x}_N) \\ K(\mathbf{x}_2, \mathbf{x}_1) & K(\mathbf{x}_2, \mathbf{x}_2) & \dots & K(\mathbf{x}_2, \mathbf{x}_N) \\ \dots & \dots & \dots & \dots \\ K(\mathbf{x}_N, \mathbf{x}_1) & K(\mathbf{x}_N, \mathbf{x}_2) & \dots & K(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix}$$

is **positive semi-definite**

for any $\mathbf{x}_1, \dots, \mathbf{x}_N$ (Mercer's condition)

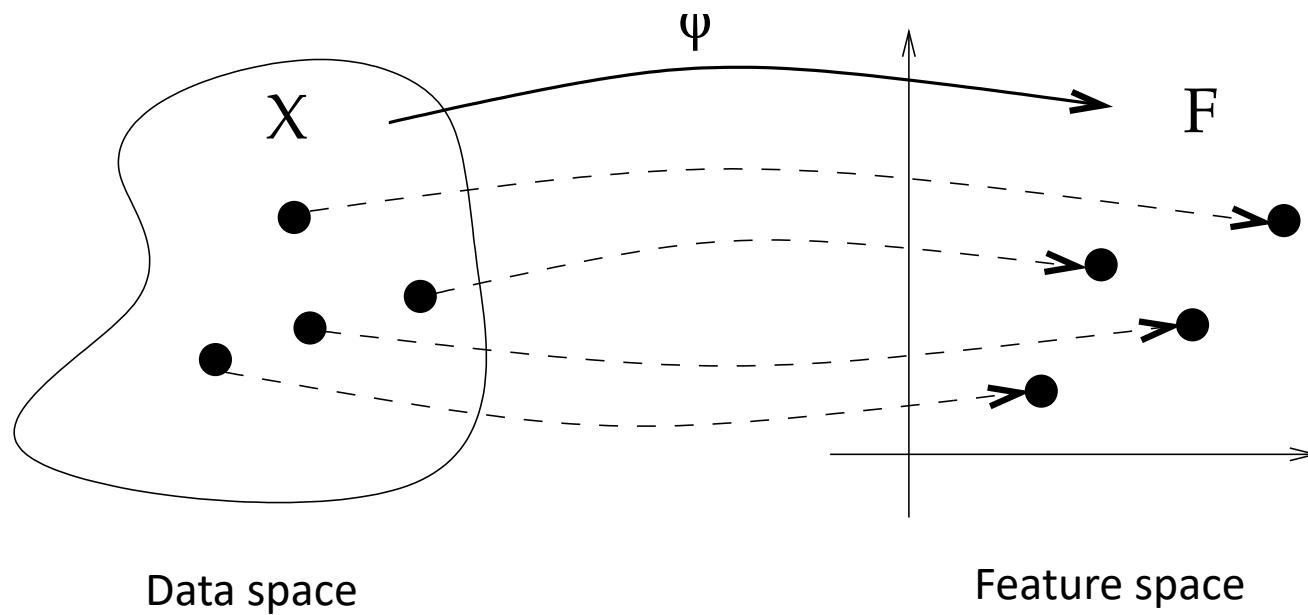
Then, you can design your own kernel as above.

Outline

- Review of SVM in the previous lecture
- A bit of history of kernel methods and SVM
- **Kernel methods**
- Kernel PCA
- Designing kernels

Kernels

- a general framework to represent data and must satisfy some math conditions that give them properties useful to understand kernel methods and kernel design



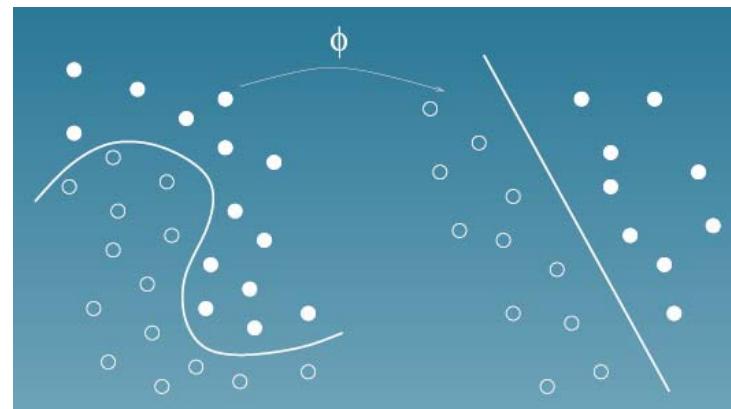
The issue of data representation

- Let $\mathcal{S} = (x_1, \dots, x_n)$ a set of n objects to be analyzed.
- Suppose that each object x_i is an element of a set \mathcal{X} , which may be images, molecules, texts, etc.
- Majority of data analysis methods represent \mathcal{S} by:
 - defining a representation $\phi(x) \in \mathcal{F}$ for each object $x \in \mathcal{X}$, where $\phi(x)$ can be a real-valued vector ($\mathcal{F} = \mathbb{R}^p$) or a finite-length string, or more complex representation.
 - representing \mathcal{S} by a set of representations of the objects

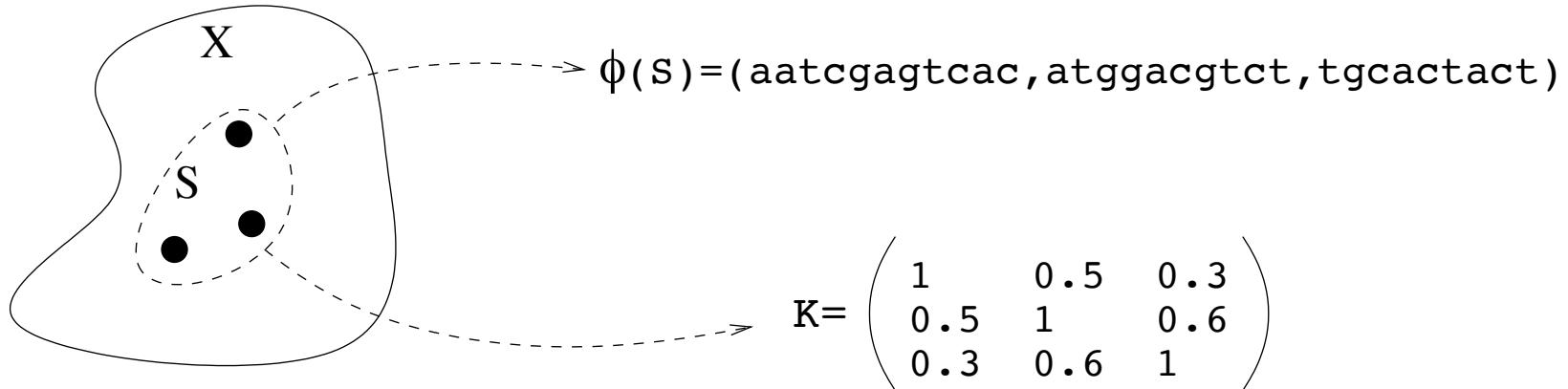
$$\phi(\mathcal{S}) = (\phi(x_1), \dots, \phi(x_n))$$

The idea of kernel representation

- Data are not represented individually anymore, but only through a set of pairwise comparisons.
- Instead of using a mapping $\phi: \mathcal{X} \rightarrow \mathcal{F}$ to represent each object $\mathbf{x} \in \mathcal{X}$ by $\phi(\mathbf{x}) \in \mathcal{F}$, a real-valued “comparison function” (called **kernel**) $k: \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ is used, and the data set \mathcal{S} is represented by the $n \times n$ matrix (Gram matrix) of pairwise comparisons $k_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$.
- A main question is how to find a kernel k such that, in the new space, problem solving is easier (e.g. linear).
- All **kernel methods** have two parts:
 - (1) find such a kernel k , and
 - (2) process such Gram matrices.

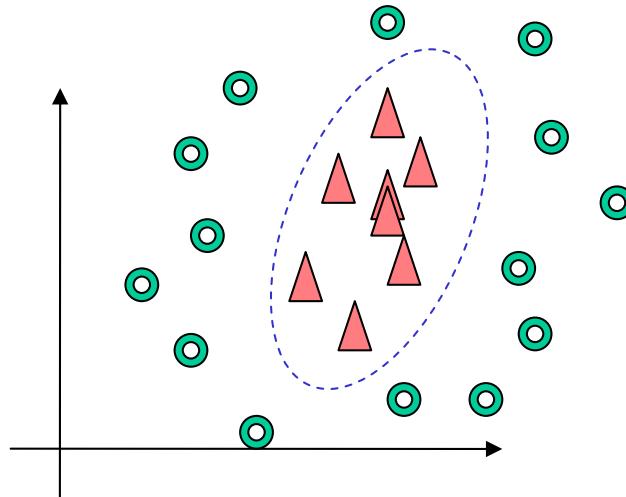


Kernel representation: example

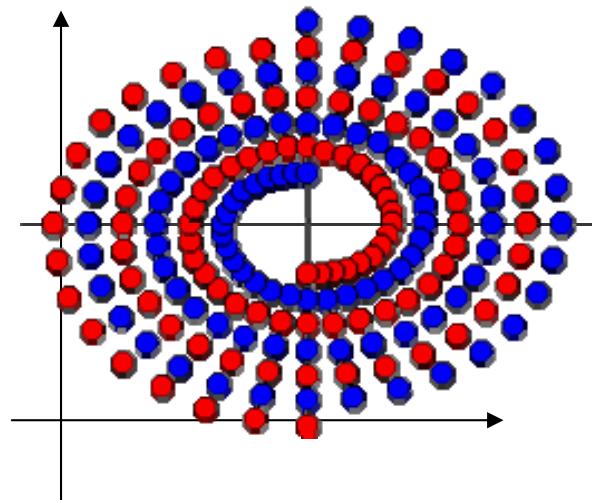


- X is the set of all oligonucleotides, S consists of three oligonucleotides.
- Traditionally, each oligonucleotide is represented by a sequence of letters.
- In kernel methods, S is represented as a matrix of pairwise similarity between its elements.

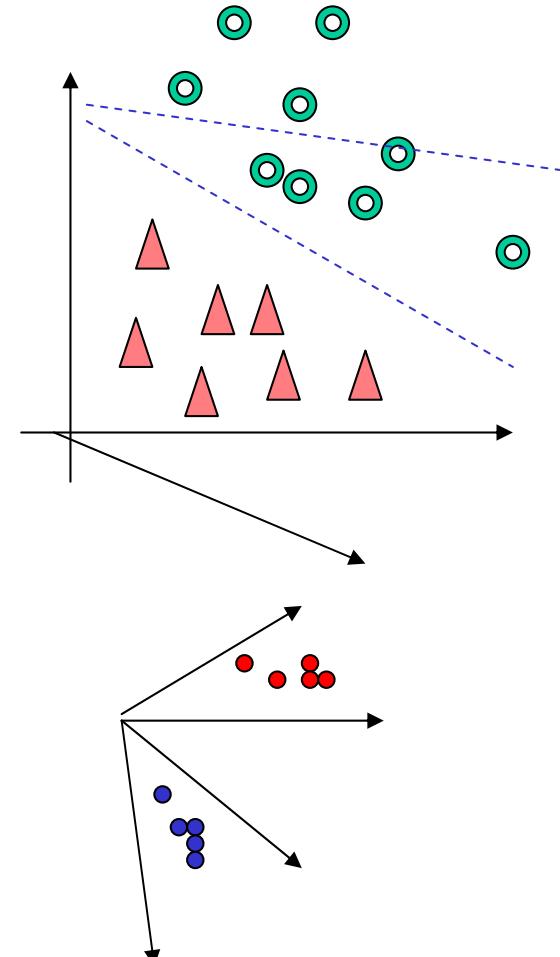
Examples of kernels



by polynomial kernel ($n=2$)
 $\phi:(x_1, x_2) \mapsto (x_1^2, \sqrt{2}x_1x_2, x_2^2)$



by RBF kernel
($n=2$)
 $\phi:(x_1, x_2) \mapsto \exp\left(-\frac{d(x_1, x_2)^2}{2\sigma^2}\right)$



(Jean-Michel Renders, ICFT'04)

Dot product

- A **vector space** (linear space) \mathcal{K} of “vectors” over the reals \mathbb{R} if two operations,
 - **vector addition:** $\mathcal{K} \times \mathcal{K} \rightarrow \mathcal{K}$ denoted $\mathbf{v} + \mathbf{w}$, where $\mathbf{v}, \mathbf{w} \in \mathcal{K}$, and
 - **scalar multiplication:** $\mathbb{R} \times \mathcal{K} \rightarrow \mathcal{K}$ denoted $\lambda\mathbf{v}$, where $\lambda \in \mathbb{R}$ and $\mathbf{v} \in \mathcal{K}$such that some axioms are satisfied (<http://www.answers.com/topic/vector-space>).
- **Dot product** (inner, scalar product) on \mathcal{K} is a symmetric bilinear form $\langle \cdot, \cdot \rangle: \mathcal{K} \times \mathcal{K} \rightarrow \mathbb{R}$, $(\mathbf{x}, \mathbf{x}') \mapsto \langle \mathbf{x}, \mathbf{x}' \rangle$ that is strictly positive definite, i.e., $\forall \mathbf{x} \in \mathcal{K}, \langle \mathbf{x}, \mathbf{x} \rangle \geq 0$ with equality only for $\mathbf{x} = 0$.
- **Norm** $\|\mathbf{x}\| := \langle \mathbf{x}, \mathbf{x} \rangle^{1/2}$ ($\|\cdot\|: \mathcal{K} \rightarrow \mathbb{R}^+$), p-norm: $\|\mathbf{x}\|_p := \left(\sum_{i=1}^n |x_i|^p \right)^{\frac{1}{p}}$
- An **dot product space** (inner product space) \mathcal{K} is a vector space endowed with a dot product. The dot product space allows us to introduce geometrical notions such as **angles** and **lengths** of vectors:

$$\cos \theta = \frac{\langle \mathbf{x}, \mathbf{x}' \rangle}{\|\mathbf{x}\| \|\mathbf{x}'\|}$$

Hilbert space

- A **Hilbert space** \mathcal{H} is a dot product space with the additional properties that it is *separable* and *complete*.

Completeness means every Cauchy sequence $\{h_n\}_{n \geq 1}$ of elements of \mathcal{H} converges to an element $h \in \mathcal{H}$, where a Cauchy sequence is one satisfying

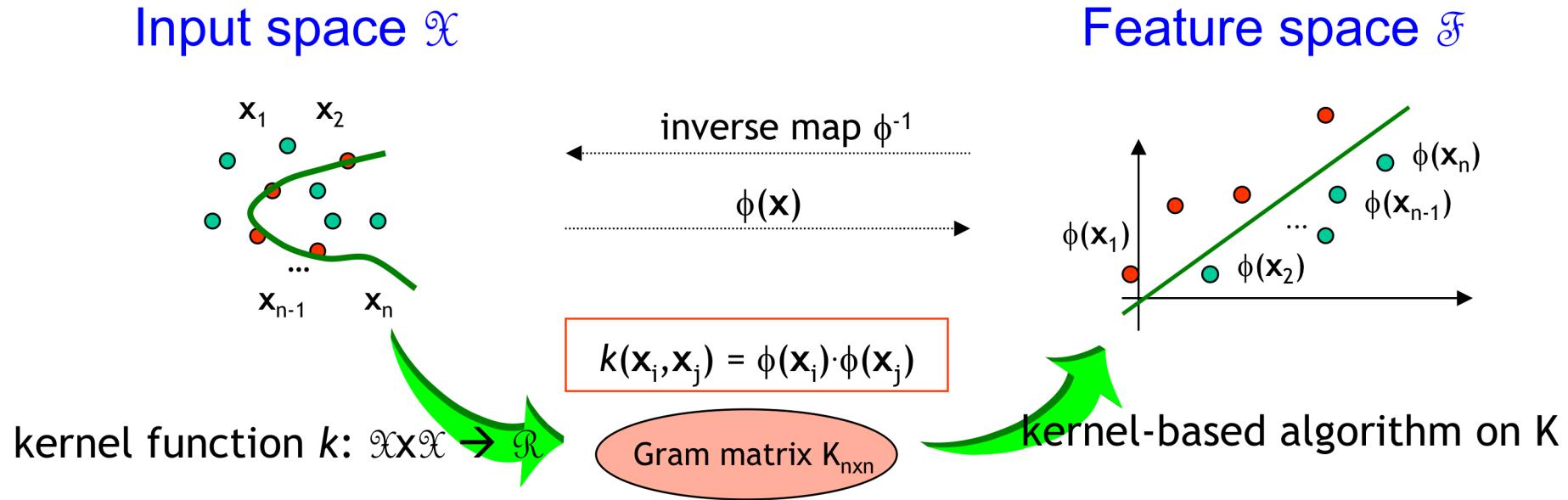
$$\sup_{m > n} \|h_n - h_m\| \rightarrow 0, \text{ as } n \rightarrow \infty$$

\mathcal{H} is separable if for any $\varepsilon > 0$ there is a finite set of elements h_1, \dots, h_N of \mathcal{H} such that for all $h \in \mathcal{H}$,

$$\min_i \|h_i - h\| < \varepsilon$$

-
- **Importance:** As all elements of a Hilbert space \mathcal{H} are linear functions in \mathcal{H} via the dot product: for a point z the corresponding function is $f_z(x) = \langle z, x \rangle$. Our target is to learn linear functions represented by a weight vector in the feature space. Finding the weight vector is therefore equivalent to identifying an appropriate point of the feature space.

General scheme of kernel methods



- Map the data from \mathcal{X} into a (high-dimensional) vector space, the feature space \mathcal{F} , by applying the **feature map** ϕ on the data points \mathbf{x} .
- Find a **linear** (or other easy) **pattern** in \mathcal{F} using a well-known algorithm (that works on the Gram matrix).
- By applying the **inverse map**, the linear pattern in \mathcal{F} can be found to correspond to a complex pattern in \mathcal{X} .
- This implicitly by only making use of inner products in \mathcal{F} (**kernel trick**)

Valid kernels

- Most kernel methods can only process **symmetric positive definite** matrix: $k_{i,j} = k_{j,i}$ for all i, j and $\mathbf{c}^T \mathbf{k} \mathbf{c} \geq 0$, for any $\mathbf{c} \in \mathbb{R}^p$.
- **Definition 1:** A function $k: \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ is called a **positive definite kernel** if it is symmetric (that is, $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}', \mathbf{x}) \quad \forall \mathbf{x}, \mathbf{x}' \in \mathcal{X}$), and positive definite, that is for any $n > 0$, any choice of n objects $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathcal{X}$, and any choice of real number $c_1, \dots, c_n \in \mathbb{R}$

$$\sum_{i=1}^n \sum_{j=1}^n c_i c_j k(\mathbf{x}_i, \mathbf{x}_j) \geq 0 \quad (1)$$

- A positive definite kernel k is called a **valid kernel** (or simply kernel).
- **Mercer's theorem:** Any positive definite function can be written as an inner product in some feature space.

Kernels as inner product (1/2)

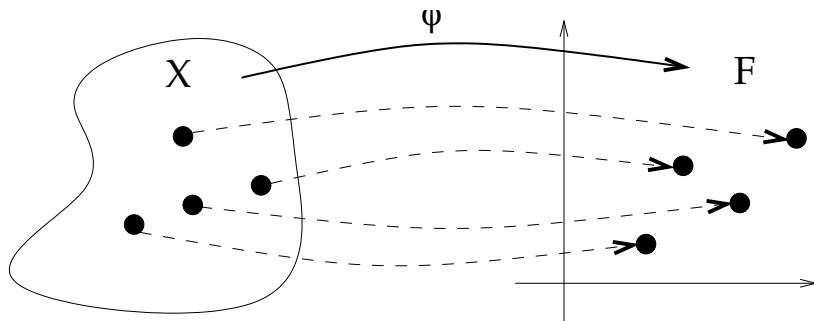
- Suppose $\mathcal{X} = \mathbb{R}^p$ and $\mathbf{x} = (x_1, \dots, x_p)^T$. Comparing such vectors by inner product, for any $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^p$,

$$k_L(\mathbf{x}, \mathbf{x}') := \mathbf{x}^T \mathbf{x}' = \sum_{i=1}^p x_i x'_i = \langle \mathbf{x}, \mathbf{x}' \rangle$$

- This function is a kernel as it is symmetric and positive definite, usually called *linear kernel*.
- One systematic way to define kernels for general objects:
 - representing each object $\mathbf{x} \in \mathcal{X}$ as a vector $\phi(\mathbf{x}) \in \mathbb{R}^p$
 - defining a kernel for any $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ as inner product $k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T \phi(\mathbf{x}')$

Can define kernel without requiring \mathcal{X} to be a vector space.

Kernels as inner product (2/2)



Any kernel on a space \mathcal{X} can be represented as an inner product after the space \mathcal{X} is mapped to a Hilbert space \mathcal{F} , called feature space.

- Kernels can all be thought of as dot products in some space \mathcal{F} , called the *feature space*. Using a kernel boils down to representing each object $x \in \mathcal{X}$ as a vector $\phi(x) \in \mathcal{F}$, and computing dot products.
- We don't need to compute explicitly $\phi(x)$ for each point in \mathcal{S} , but only the pairwise dot products. \mathcal{F} usually is infinite-dimensional, and $\phi(x)$ is tricky to represent though the kernel is simple to compute.
- Most kernel methods possess such an interpretation when the points $x \in \mathcal{X}$ are viewed as points $\phi(x)$ in the feature space.

The representer theorem

- **Theorem** (Kimeldorf, Wahba, 1971): *If the problem*

$$\min_{f \in \mathcal{H}} \{C(f, \{x_i, y_i\}) + \Omega(\|f\|_{\mathcal{H}})\} \quad (5)$$

satisfies (1) C is pointwise; i.e., $C(f, \{x_i, y_i\}) = C(\{x_i, y_i, f(x_i)\})$ which only depends on $\{f(x_i)\}$ and (2) $\Omega(\cdot)$ is monotonically increasing, then every minimizer of the problem admits a representation of the form

$$f(\cdot) = \sum_{i=1}^m \alpha_i k(x_i, \cdot) \quad (6)$$

- The quantity $\|\cdot\|_{\mathcal{H}}$ included in the function to optimize is a penalization that forces the solution to be smooth and usually a powerful protection against over-fitting of the data.
- Meaning: Instead of finding optimal solution in infinite-dimensional space, (5) can be reformulated as a n -dimensional optimization problem, by plugging (6) into (5) and optimizing over $(\alpha_1, \dots, \alpha_n) \in \mathbb{R}^p$.

Outline

- Review of SVM in the previous lecture
- A bit of history of kernel methods and SVM
- Kernel methods
- **Kernel PCA**
- Designing kernels

Principal component analysis (PCA)

- **PCA:** Given $\mathbf{x}_1, \dots, \mathbf{x}_m \in \mathbb{R}^p$. Find: $\mathbf{x}'_1, \dots, \mathbf{x}'_m \in \mathbb{R}^d$, $d \ll p$ so that $\mathbf{x}'_1, \dots, \mathbf{x}'_m$ preserve most information.
- PCA finds the principal axes by diagonalizing the covariance matrix C with singular value decomposition

$$\lambda v = Cv$$

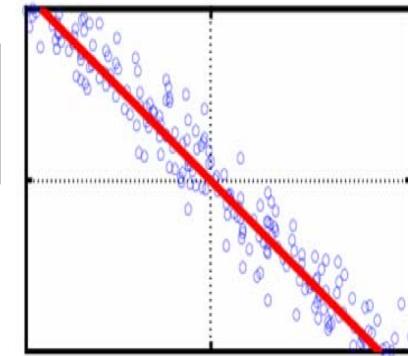
$$C = \frac{1}{m} \sum_{j=1}^m x_j x_j^T$$

$$\lambda v = Cv = \frac{1}{m} \sum x_j x_j^T v$$

$$v = \frac{1}{m\lambda} \sum x_j x_j^T v = \frac{1}{m\lambda} \sum (x_j \cdot v) x_j$$

- All solutions v with $\lambda \neq 0$ lie in the span of x_1, \dots, x_m , i.e.

$$v = \sum_i \alpha_i x_i$$



1. Find eigenvectors, order them in decreasing.

2. Projects points onto eigenvectors.

3. Use new coordinates to do things

Kernel trick

- Do PCA in feature space? The covariance matrix

$$\bar{C} = \frac{1}{m} \sum_{j=1}^m \phi(x_j) \phi(x_j)^T$$

- can be diagonalized with nonnegative eigenvalues satisfying

$$\lambda V = \bar{C} V$$

- As V lie in the span of $\phi(x_i)$, so we have

$$\begin{aligned}\lambda \sum \alpha_i \phi(x_i) &= \frac{1}{m} \sum \phi(x_j) \phi(x_j)^T \cdot \sum \alpha_i \phi(x_i) = \frac{1}{m} \sum \sum \alpha_i \phi(x_j) \phi(x_j)^T \phi(x_i) \\ &= \frac{1}{m} \sum_j \sum_i \alpha_i \phi(x_i) \langle \phi(x_i), \phi(x_j) \rangle\end{aligned}$$

Kernel PCA

- Apply kernel trick, we have

$$K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$$

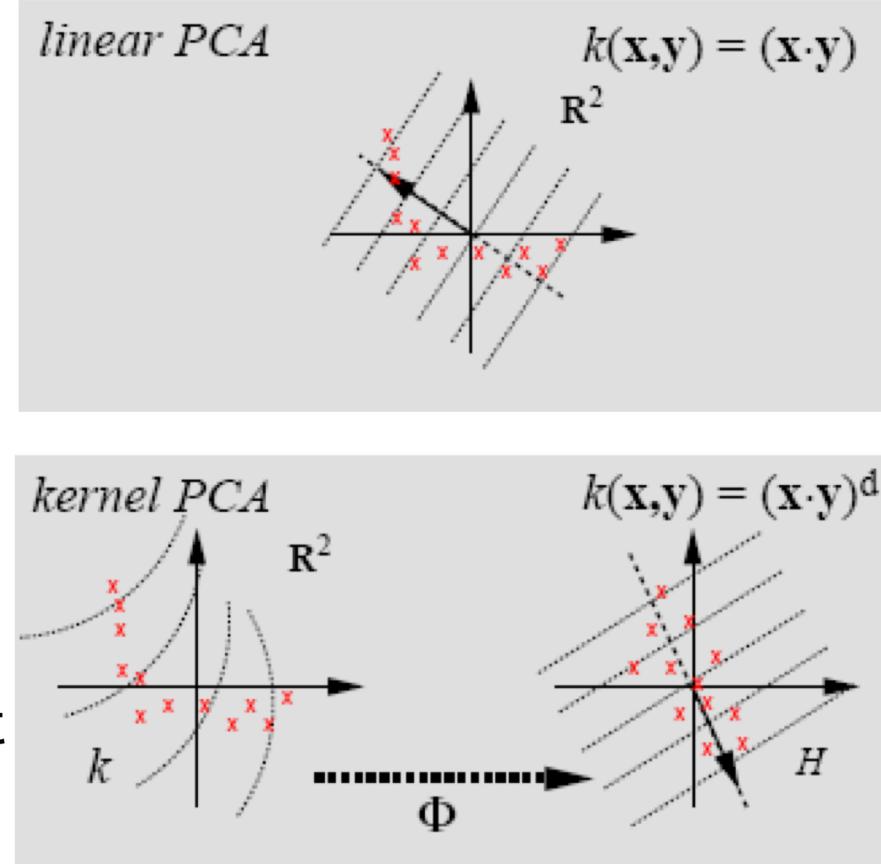
$$m\lambda \sum_i \alpha_i \phi(x_i) = \sum_i \sum_j \alpha_i \phi(x_i) K(x_i, x_j)$$

- And we can finally write the expression as the eigenvalue problem

$$\mathbf{K}\boldsymbol{\alpha} = \lambda \boldsymbol{\alpha}$$

- For a test pattern x , we extract a nonlinear component via

$$\langle V_k, x \rangle = \sum_{i=1}^m \alpha_i^k K(x_i, x)$$



Outline

- Review of SVM in the previous lecture
- A bit of history of kernel methods and SVM
- Kernel methods
- Kernel PCA
- **Designing kernels**

How to chose kernels

- There is no absolute rules for choosing the right kernel, adapted to a particular problem
- Kernel design can start from the desired feature space, from combination or from data
- Some considerations are important:
 - Use kernel to introduce a priori (domain) knowledge
 - Be sure to keep some information structure in the feature space
 - Experimentally, there is some “robustness” in the choice, if the chosen kernels provide an acceptable trade-off between
 - simpler and more efficient structure (e.g. linear separability), which requires some “explosion”
 - Information structure preserving, which requires that the “explosion” is not too strong.

Kernels built from data

- In general, this mode of kernel design can use both labeled and unlabeled data of the training set! Very useful for semi-supervised learning
- Basic ideas:
 - Intuitively, kernels define clusters in the feature space, and we want to find interesting clusters, i.e. cluster components that can be associated with labels.
 - Convex linear combination of kernels in a given family: find the best coefficient of eigen-components of the (complete) kernel matrix by maximizing the alignment on the labeled training data.
 - Build a generative model of the data, then use the Fischer Kernel.

Some operations on kernels

- The class of kernel functions on \mathcal{X} has several useful closure properties. It is a convex cone, which means that if k_1 and k_2 are two kernels, then any **linear combination**, $\lambda_1 k_1 + \lambda_2 k_2$, with $\lambda_1, \lambda_2 \geq 0$ is a kernel.
 - If k_1 and k_2 are two kernels, then $k(\mathbf{x}, \mathbf{x}') := k_1(\mathbf{x}, \mathbf{x}') k_2(\mathbf{x}, \mathbf{x}')$ is also a kernel. Several other operations are possible.
 - Some other operations are forbidden: if k is a kernel, then $\log(k)$ is not positive definite in general, and neither is k^β for $0 < \beta < 1$. However, these two operations can be linked.
-

Combining kernels

-
- Multiple kernel matrices k_1, k_2, \dots, k_c for the same set of objects are available.
 - We may design a single kernel k from several basic kernels k_1, k_2, \dots, k_c . A simple way to achieve this is to take the **sum of the kernels**:

$$k = \sum_{i=1}^c k_i$$

- Multiple kernel matrices k_1, k_2, \dots, k_c for the same set of objects are available.

$$k = \sum_{i=1}^c \mu_i k_i$$

From similarity scores to kernels

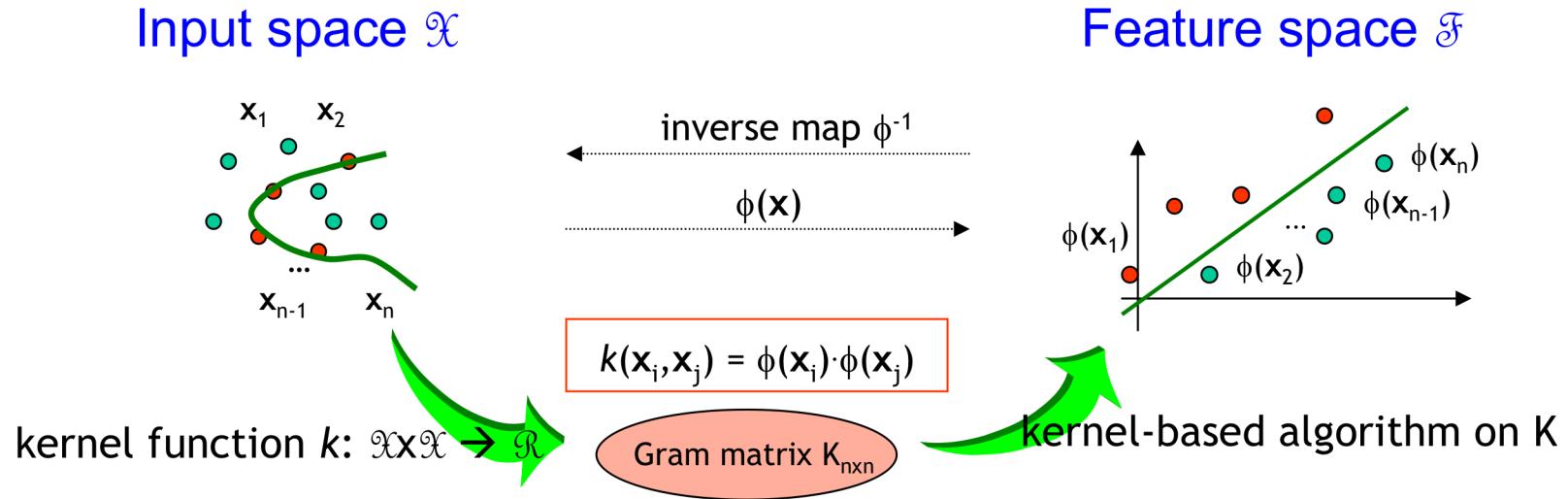
- \mathcal{X} be a set and $s: \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ a measure of similarity.
- Empirical kernel map (Tsuda, 1999): (1) Choosing a finite set of objects $t_1, \dots, t_r \in \mathcal{X}$ called **templates**. (2) $x \in \mathcal{X}$ is represented by a vector of similarity

$$x \in \mathcal{X} \rightarrow \phi(x) = (s(x, t_1), \dots, s(x, t_r))^T \in \mathbb{R}^p$$

- The kernel is defined as the dot product between two similarity vectors

$$\forall x, x' \in \mathcal{X}, k(x, x') = \phi(x)^T \phi(x) = \sum_{i=1}^r s(x, t_i) s(x', t_i)$$

Summary



- Map the data from \mathcal{X} into a (high-dimensional) vector space, the feature space \mathcal{F} , by applying the **feature map** ϕ on the data points x .
- Find a **linear** (or other easy) **pattern in \mathcal{F}** using a well-known algorithm (that works on the Gram matrix).
- By applying the **inverse map**, the linear pattern in \mathcal{F} can be found to correspond to a complex pattern in \mathcal{X} .
- This implicitly by only making use of inner products in \mathcal{F} (**kernel trick**)

Thank you!