Kernel methods

Shikui Tu

Department of Computer Science and Engineering, Shanghai Jiao Tong University

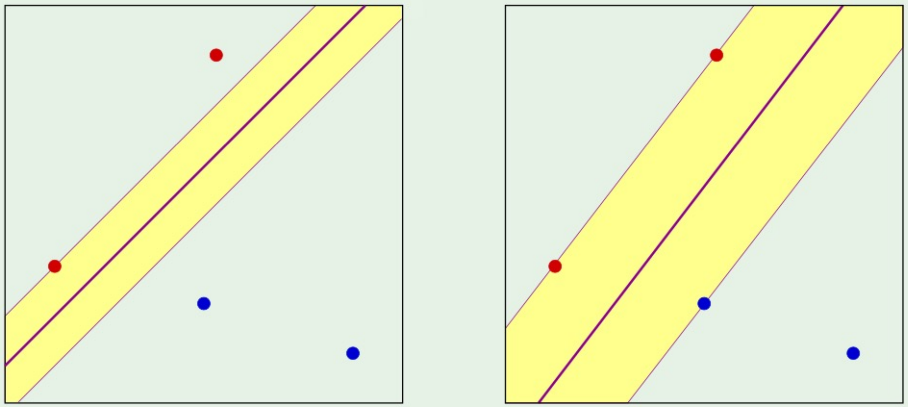
2021-05-28

Outline

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

Maximize margin

- The margin

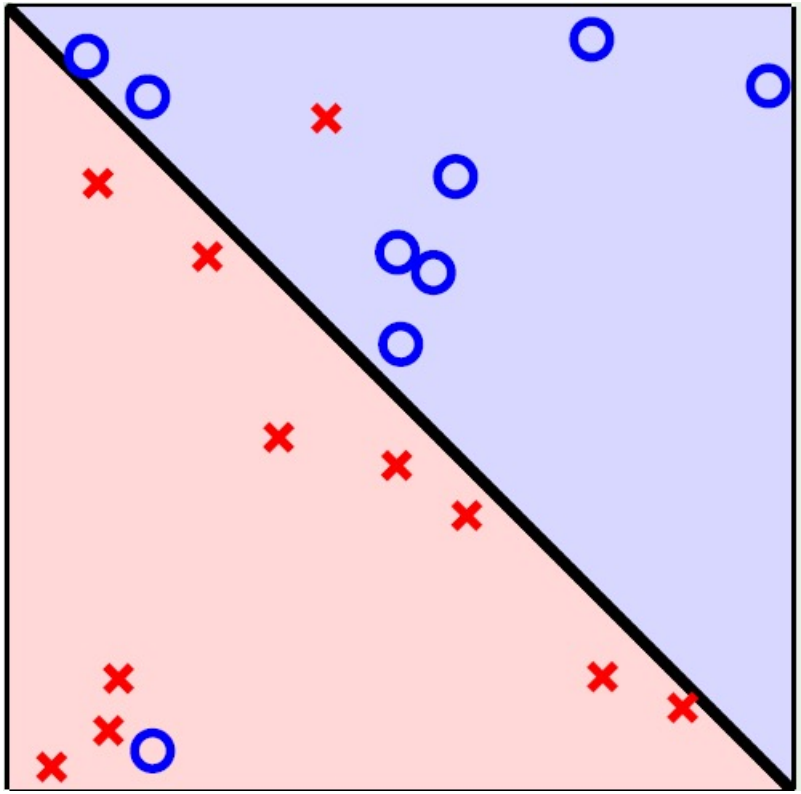


Maximizing the margin $\Longrightarrow$ dual problem:

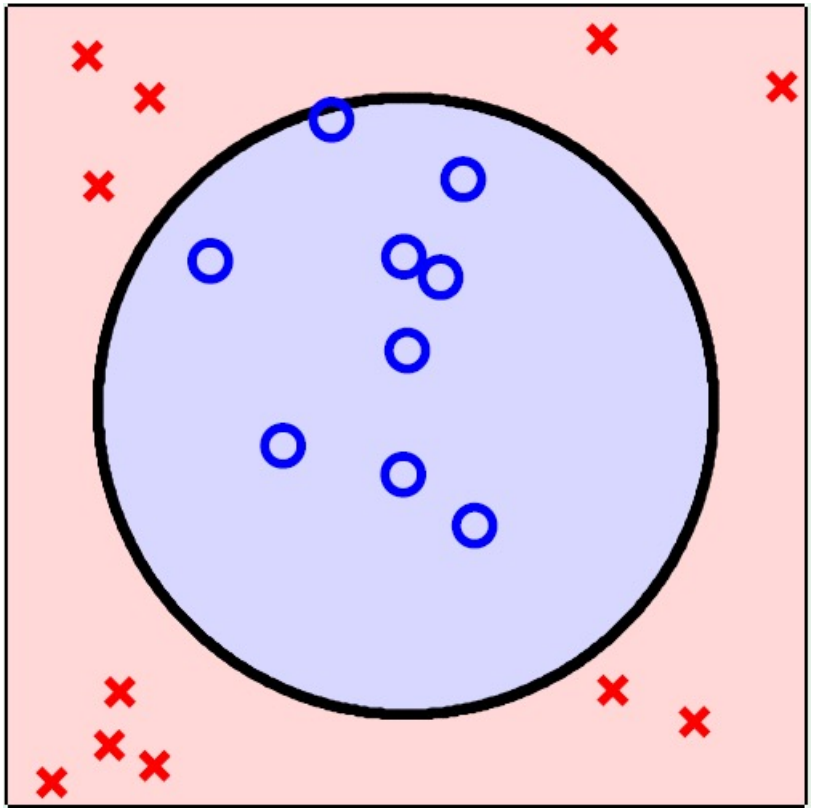
$$\mathcal{L}(\boldsymbol{\alpha}) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} y_n y_m \, \alpha_n \alpha_m \, \mathbf{x}_n^{\mathsf{T}} \mathbf{x}_m$$

quadratic programming

Non-separable Data



Slightly
non-separable

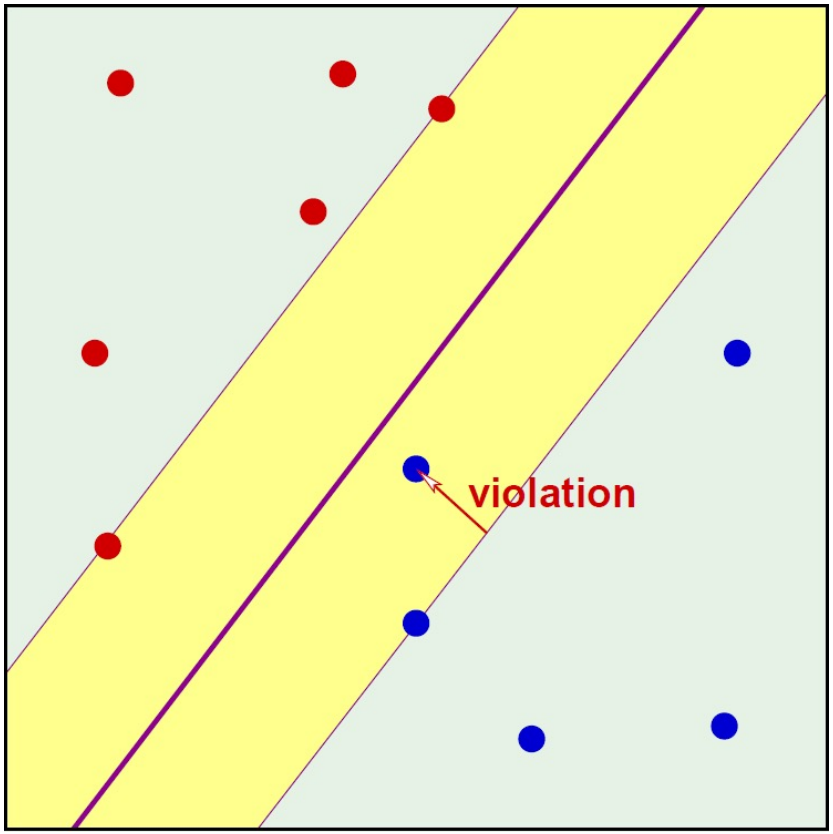


Seriously
non-separable

Soft-margin SVM

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

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



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

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

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

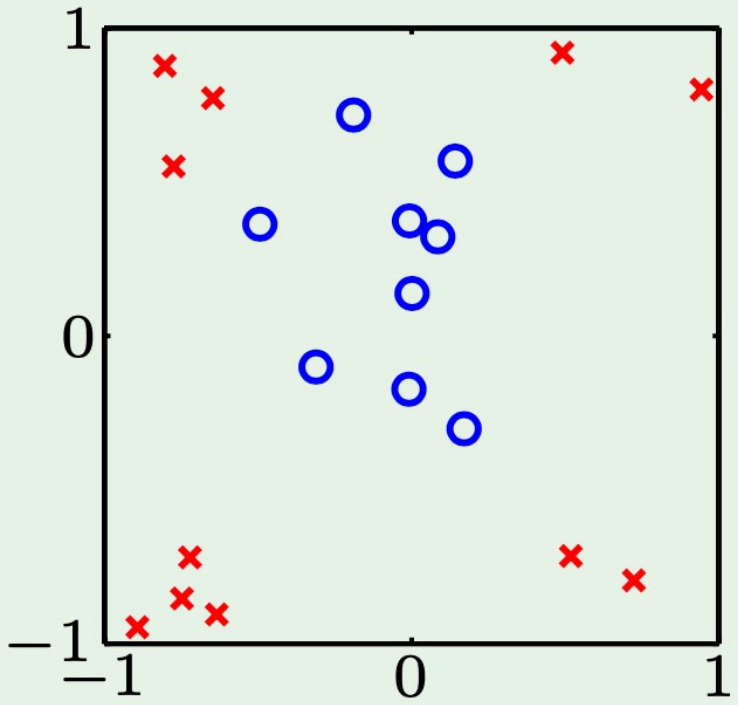
$$\text{subject to} \quad y_n\left(\mathbf{w}^{\mathsf{T}}\mathbf{x}_n + b\right) \geq 1 - \xi_n$$

$$\text{for} \quad n = 1, \ldots, N$$

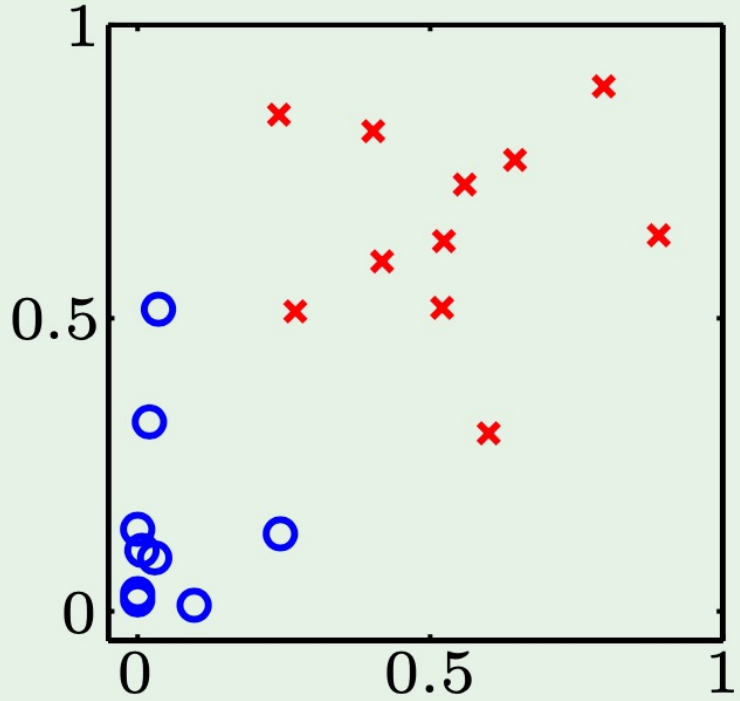
$$\text{and} \quad \xi_n \geq 0 \quad \text{for} \quad n = 1, \ldots, N$$

Kernel transform

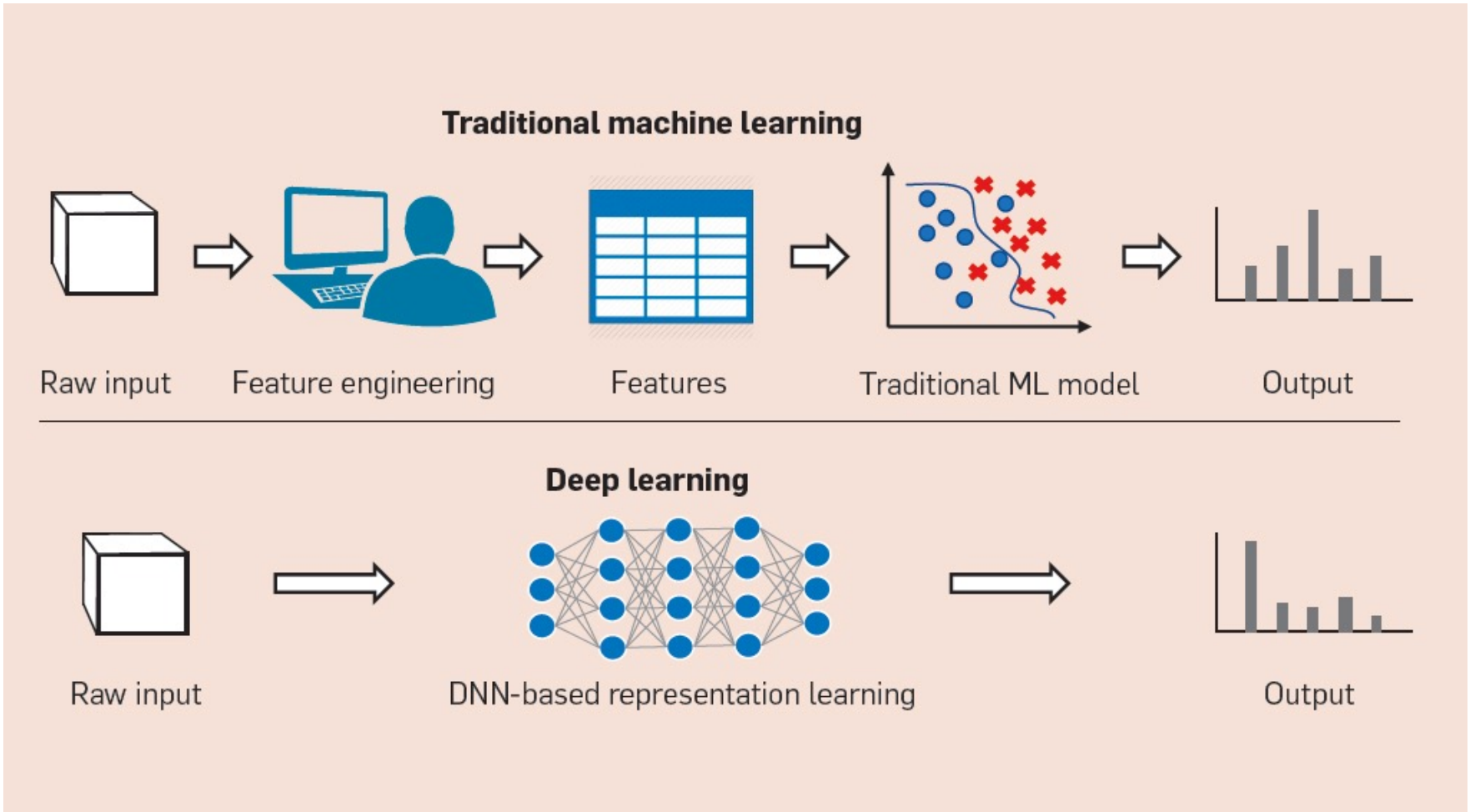
$$\mathcal{L}(\boldsymbol{\alpha}) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} y_n y_m \, \alpha_n \alpha_m \, \mathbf{z}_n^{\mathsf{T}} \mathbf{z}_m$$



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



Traditional machine learning



Outline

- Review of SVM in the previous lecture

- **A bit of history of kernel methods and SVM**

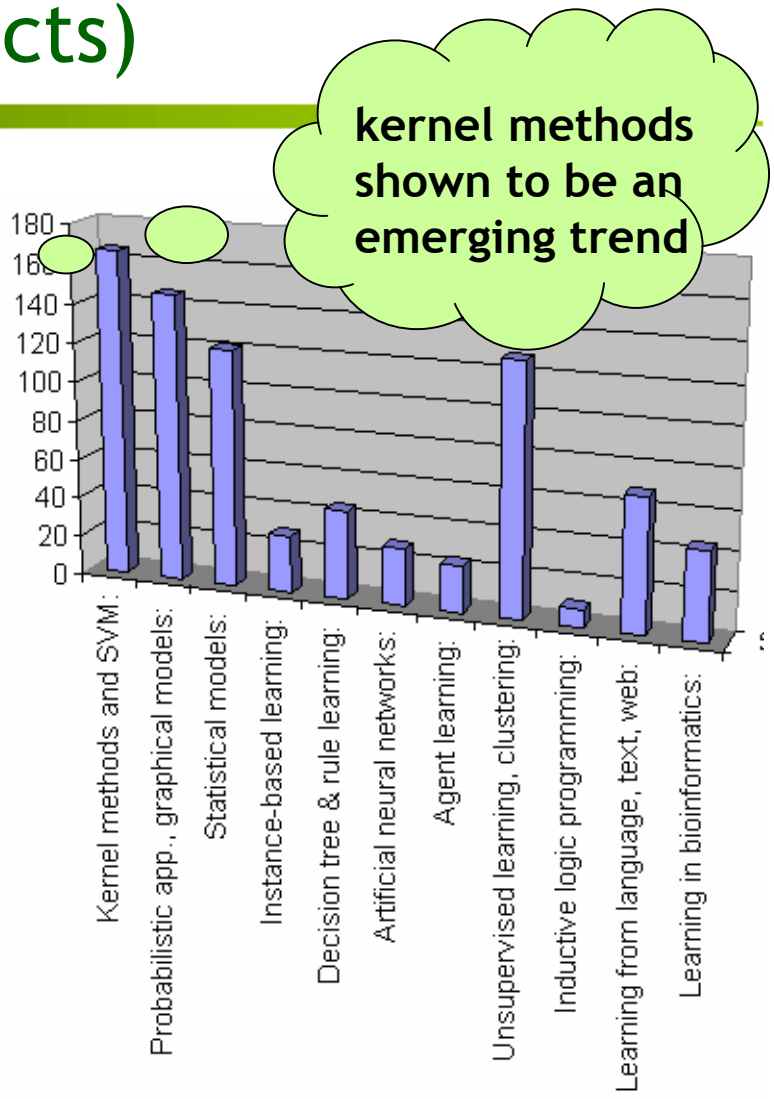
- Kernel methods

- Designing kernels

Very popular around 2006

ICML 2006 (720 abstracts)

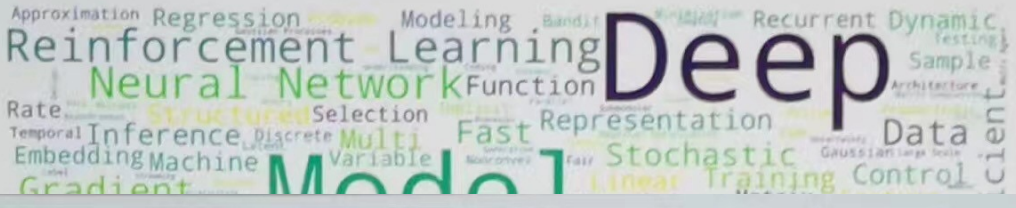
Kernel methods & SVM	**166**
Probabilistic, graphical models	146
Unsupervised learning, clustering	128
Statistical models	121
Language, Text & web	68
Learning in bioinformatics	45
ANN	29
ILP	9
CRF	13

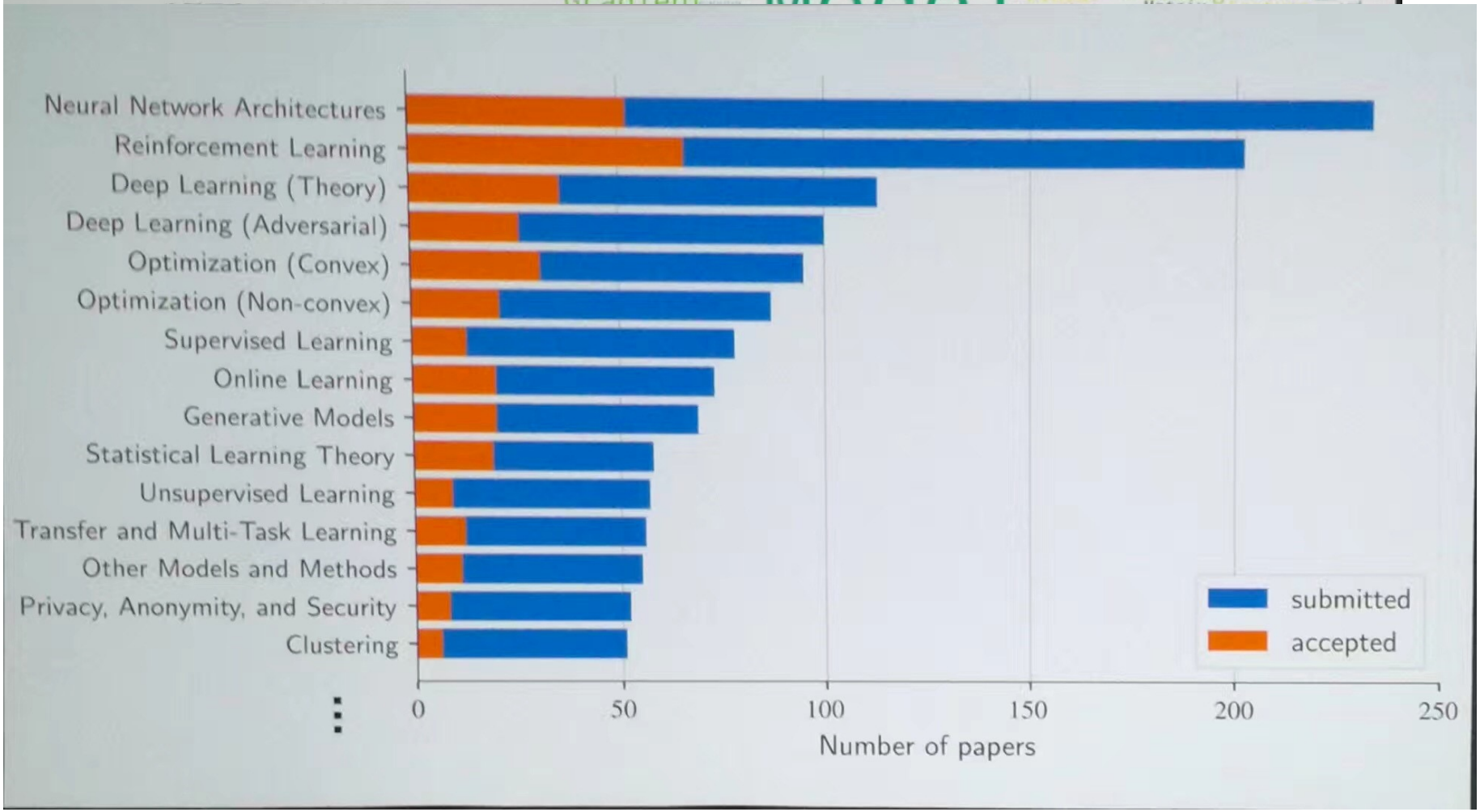


ICML 2018

Program Committee

160 Area Chairs





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

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

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

$$\boxed{g(\mathbf{x}) = \text{sign}\left(\mathbf{w}^{\mathsf{T}} \mathbf{z} + b\right)} \qquad \text{need } \mathbf{z}_n^{\mathsf{T}} \mathbf{z}$$

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

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

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

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

$$K(\mathbf{x}, \mathbf{x}') = \mathbf{z}^\top\mathbf{z}' = 1 + x_1x'_1 + x_2x'_2 + x_1^2x'^2_1 + x_2^2x'^2_2 + x_1x'_1x_2x'_2$$

Without transforming explicitly

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

Example:

$$K(\mathbf{x}, \mathbf{x}') = (1 + \mathbf{x}^\intercal \mathbf{x}')^2 = (1 + x_1 x'_1 + x_2 x'_2)^2$$

$$= 1 + x_1^2 x'^2_1 + x_2^2 x'^2_2 + 2x_1 x'_1 + 2x_2 x'_2 + 2x_1 x'_1 x_2 x'_2$$

Inner product

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

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

The polynomial kernel

$\mathcal{X} = \mathbb{R}^d$ and $\Phi : \mathcal{X} \to \mathcal{Z}$ 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 $K(\boldsymbol{x}, \boldsymbol{x}')$ is an inner product in some space Z, that is ok.
- For example,

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

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

$$K(x, x') = \exp\left(-(x - x')^2\right)$$

$$= \exp\left(-x^2\right) \exp\left(-x'^2\right) \underbrace{\sum_{k=0}^{\infty} \frac{2^k (x)^k (x')^k}{k!}}_{\exp(2xx')}$$

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

Kernel SVM

$$\min_{\boldsymbol{\alpha}} \quad \frac{1}{2}\,\boldsymbol{\alpha}^{\mathsf{T}} \underbrace{\begin{bmatrix} y_1 y_1\, \mathbf{x}_1^{\mathsf{T}}\mathbf{x}_1 & y_1 y_2\, \mathbf{x}_1^{\mathsf{T}}\mathbf{x}_2 & \dots & y_1 y_N\, \mathbf{x}_1^{\mathsf{T}}\mathbf{x}_N \\ y_2 y_1\, \mathbf{x}_2^{\mathsf{T}}\mathbf{x}_1 & y_2 y_2\, \mathbf{x}_2^{\mathsf{T}}\mathbf{x}_2 & \dots & y_2 y_N\, \mathbf{x}_2^{\mathsf{T}}\mathbf{x}_N \\ \dots & \dots & \dots & \dots \\ y_N y_1\, \mathbf{x}_N^{\mathsf{T}}\mathbf{x}_1 & y_N y_2\, \mathbf{x}_N^{\mathsf{T}}\mathbf{x}_2 & \dots & y_N y_N\, \mathbf{x}_N^{\mathsf{T}}\mathbf{x}_N \end{bmatrix}}_{\text{quadratic coefficients}} \boldsymbol{\alpha} + \underbrace{(-\mathbf{1}^{\mathsf{T}})}_{\text{linear}}\boldsymbol{\alpha}$$

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

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

Kernel formulation

$$\underbrace{\begin{bmatrix} y_1 y_1\, K(\mathbf{x}_1, \mathbf{x}_1) & y_1 y_2\, K(\mathbf{x}_1, \mathbf{x}_2) & \dots & y_1 y_N K(\mathbf{x}_1, \mathbf{x}_N) \\ y_2 y_1\, K(\mathbf{x}_2, \mathbf{x}_1) & y_2 y_2\, K(\mathbf{x}_2, \mathbf{x}_2) & \dots & y_2 y_N K(\mathbf{x}_2, \mathbf{x}_N) \\ \dots & \dots & \dots & \dots \\ y_N y_1 K(\mathbf{x}_N, \mathbf{x}_1) & y_N y_2 K(\mathbf{x}_N, \mathbf{x}_2) & \dots & y_N y_N K(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix}}_{\text{quadratic coefficients}}$$

Solution to kernel SVM

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

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

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 $\mathbf{x}_1, \cdots, \mathbf{x}_N$ (Mercer's condition)

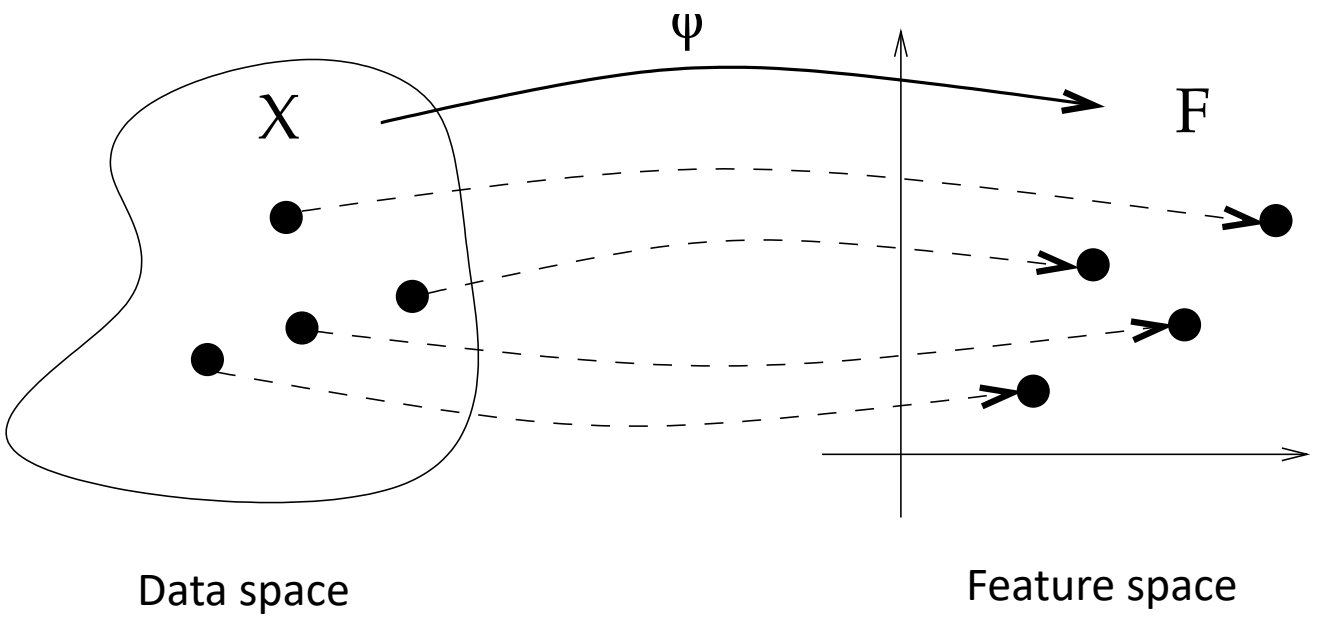
Then, you can design your own kernel as above.

Outline

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



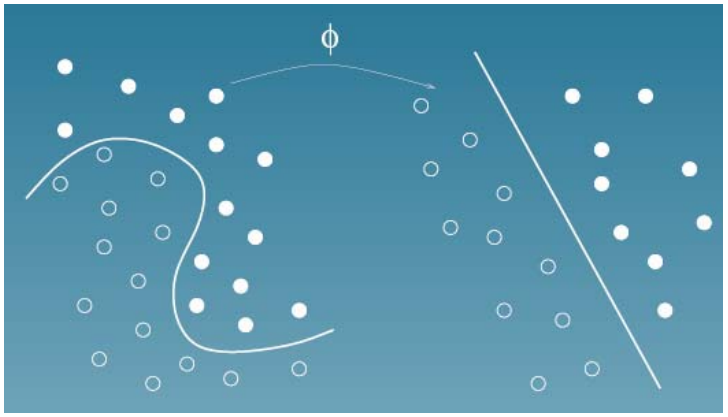
The issue of data representation

- Let $\mathbb{S} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ a set of *n* objects to be analyzed.
- Suppose that each object $\mathbf{x}_i$ is an element of a set $\mathbb{X}$, which may be images, molecules, texts, etc.
- Majority of data analysis methods represent $\mathbb{S}$ by:
  - defining a representation $\phi(\mathbf{x}) \in \mathbb{F}$ for each object $\mathbf{x} \in \mathbb{X}$, where $\phi(\mathbf{x})$ can be a real-valued vector ($\mathbb{F} = \Re^p$) or a finite-length string, or more complex representation.
  - representing $\mathbb{S}$ by a set of representations of the objects

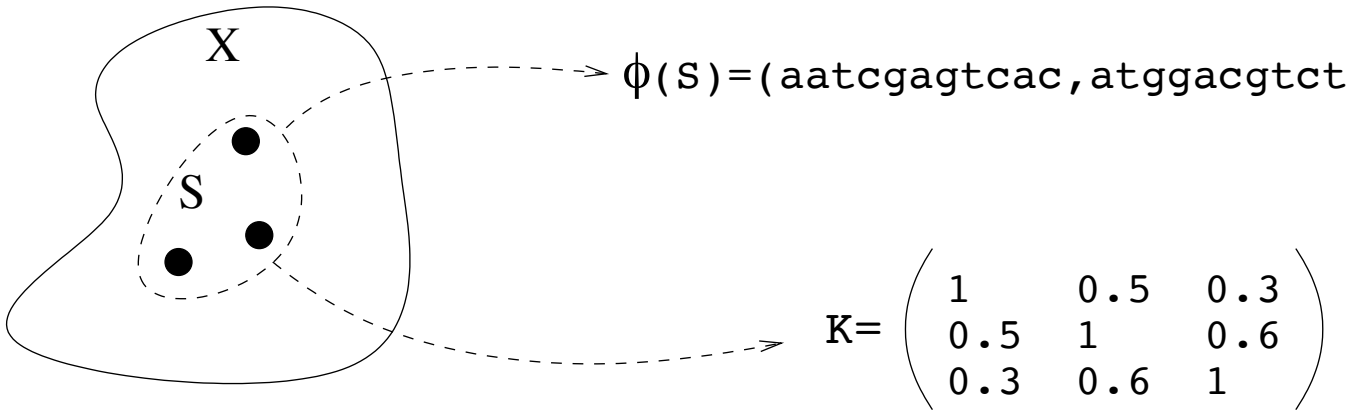
$$\phi(\mathbb{S}) = (\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_n))$$

The idea of kernel representation

- Data are not represented individually anymore, but only through a set of pairwise comparisons.
- Instead of using a mapping $\phi: \mathcal{X} \to \mathcal{F}$ to represent each object $\mathbf{x} \in \mathcal{X}$ by $\phi(\mathbf{x}) \in \mathcal{F}$, a real-valued "comparison function" (called kernel) $k: \mathcal{X} \times \mathcal{X} \to \Re$ is used, and the data set $\mathcal{S}$ is represented by the nxn matrix (Gram matrix) of pairwise comparisons $k_{i,j} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$.
- 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.

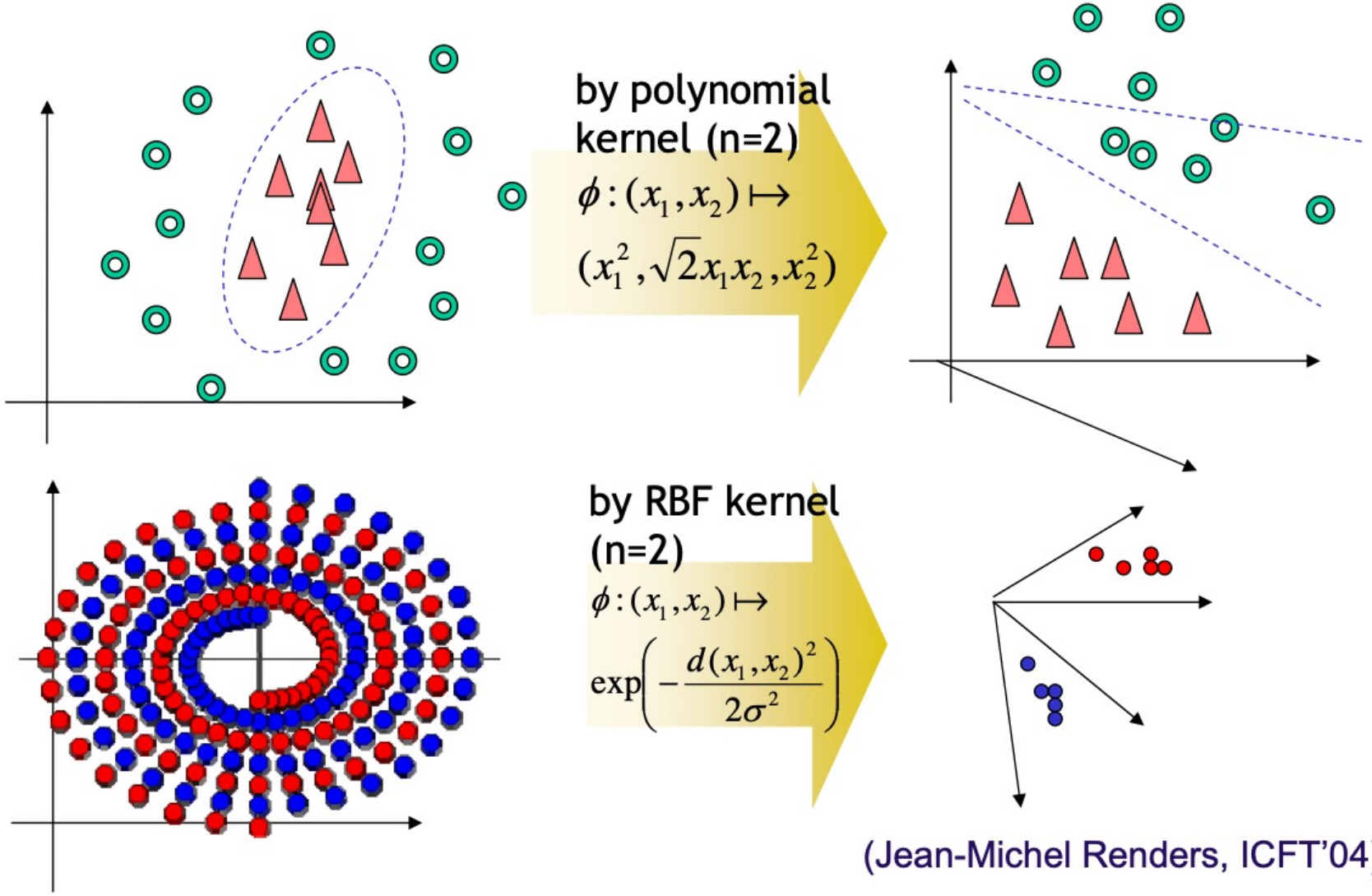


Kernel representation: example



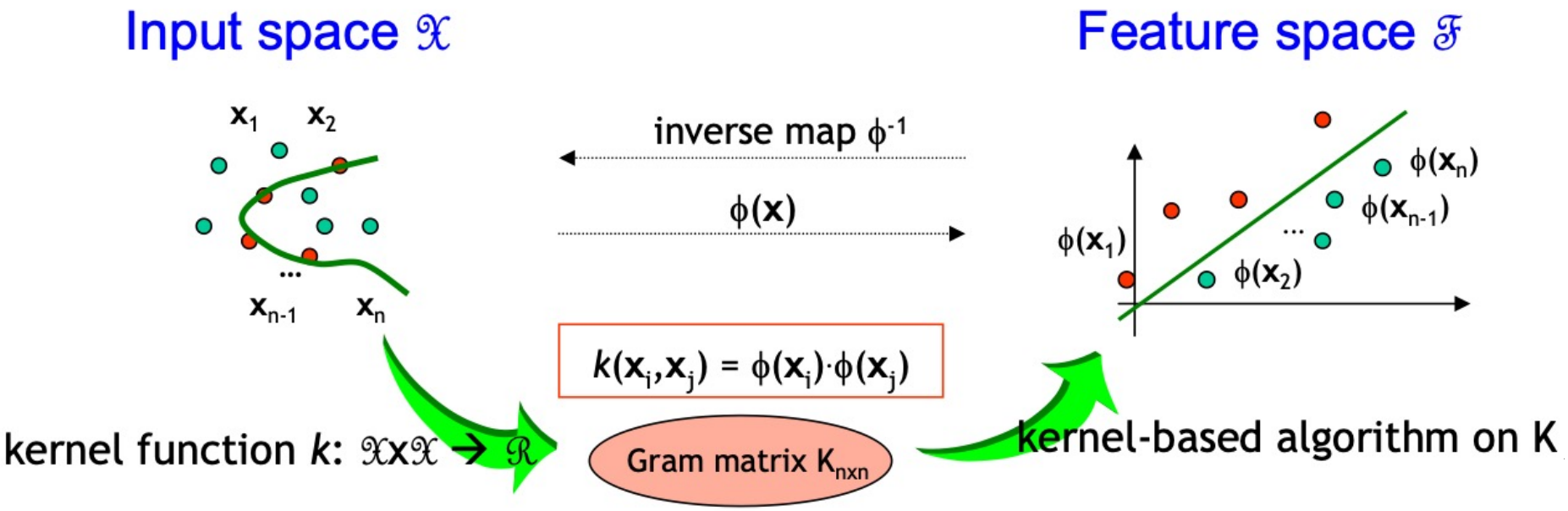
- X is the set of all oligonucleotides, S consists of three oligonucleotides.
- Traditionally, each oligonucleotide is represented by a sequence of letters.
- In kernel methods, S is represented as a matrix of pairwise similarity between its elements.

Examples of kernels



(Jean-Michel Renders, ICFT'04)

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 **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)

Outline

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

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

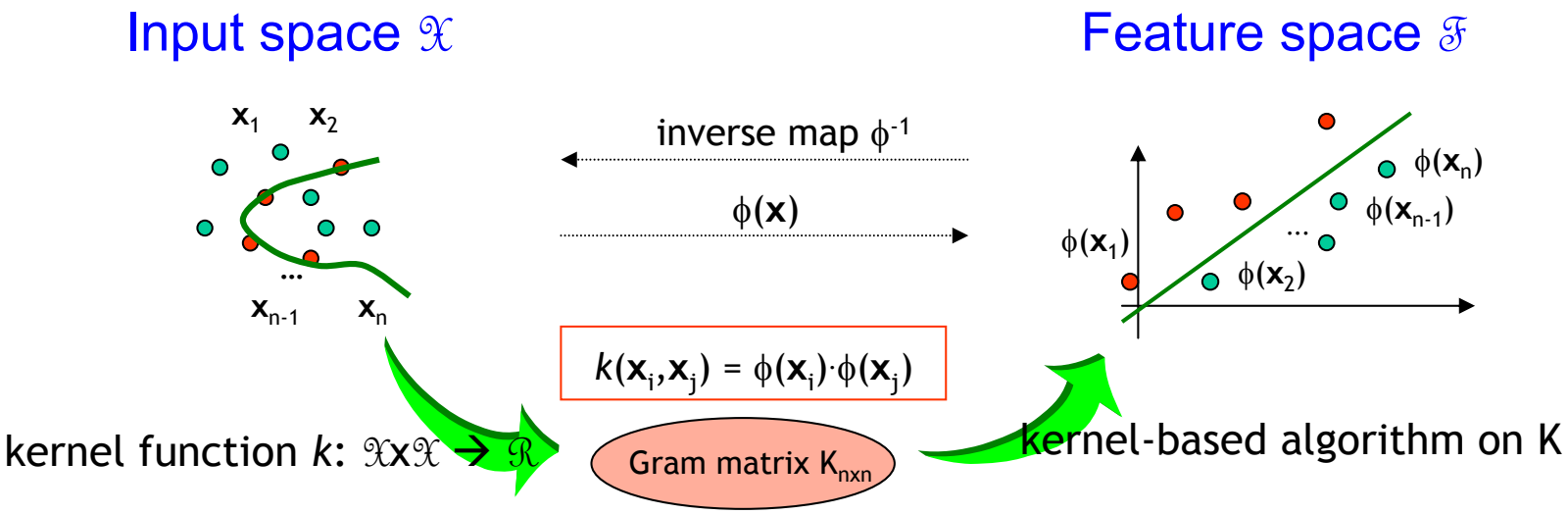
- Multiple kernel matrices k_1, k_2, …, k_c for the same set of objects are available.
- We may design a single kernel k from several basic kernels k_1, k_2, …, k_c. A simple way to achieve this is to take the sum of the kernels:

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

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

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

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!