UVA CS 4774: Machine Learning

S4: Lecture 21: Support Vector Machine (nonlinear) Kernel Trick and in Practice

Module I

Dr. Yanjun Qi

University of Virginia

Department of Computer Science

What Left in SVM?

❑ Support Vector Machine (SVM)

- ✓ History of SVM
- ✓ Large Margin Linear Classifier
- ✓ Define Margin (M) in terms of model parameter
- ✓ Optimization to learn model parameters (w, b)
- ✓ Linearly Non-separable case (soft SVM)
- ✓ Optimization with dual form
- ✓ Nonlinear decision boundary
- ✓ Practical Guide

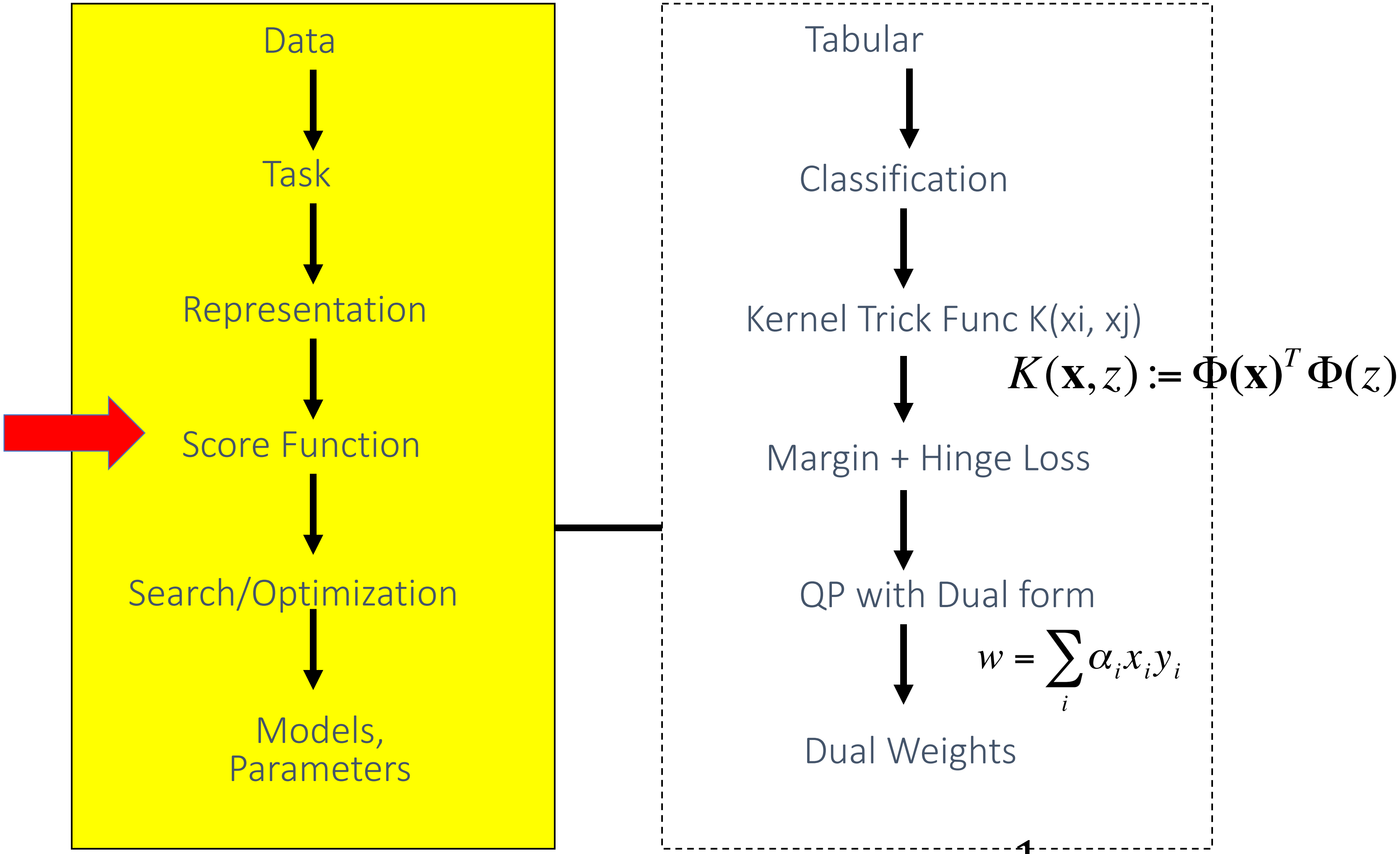
Today

- ❑ Support Vector Machine (SVM)
  - ✓ History of SVM
  - ✓ Large Margin Linear Classifier
  - ✓ Define Margin (M) in terms of model parameter
  - ✓ Optimization to learn model parameters (w, b)
  - ✓ Non linearly separable case (Extra)
  - ✓ Optimization with dual form (Extra)
  - ✓ Nonlinear decision boundary
  - ✓ Practical Guide

Extra

Today

Last: Basic Support Vector Machine



Data

Task

Representation

Score Function

Search/Optimization

Models, Parameters

Tabular

Classification

Kernel Trick Func K(xi, xj)

$$K(\mathbf{x}, z) := \Phi(\mathbf{x})^T \Phi(z)$$

Margin + Hinge Loss

QP with Dual form

$$w = \sum_i \alpha_i x_i y_i$$

Dual Weights

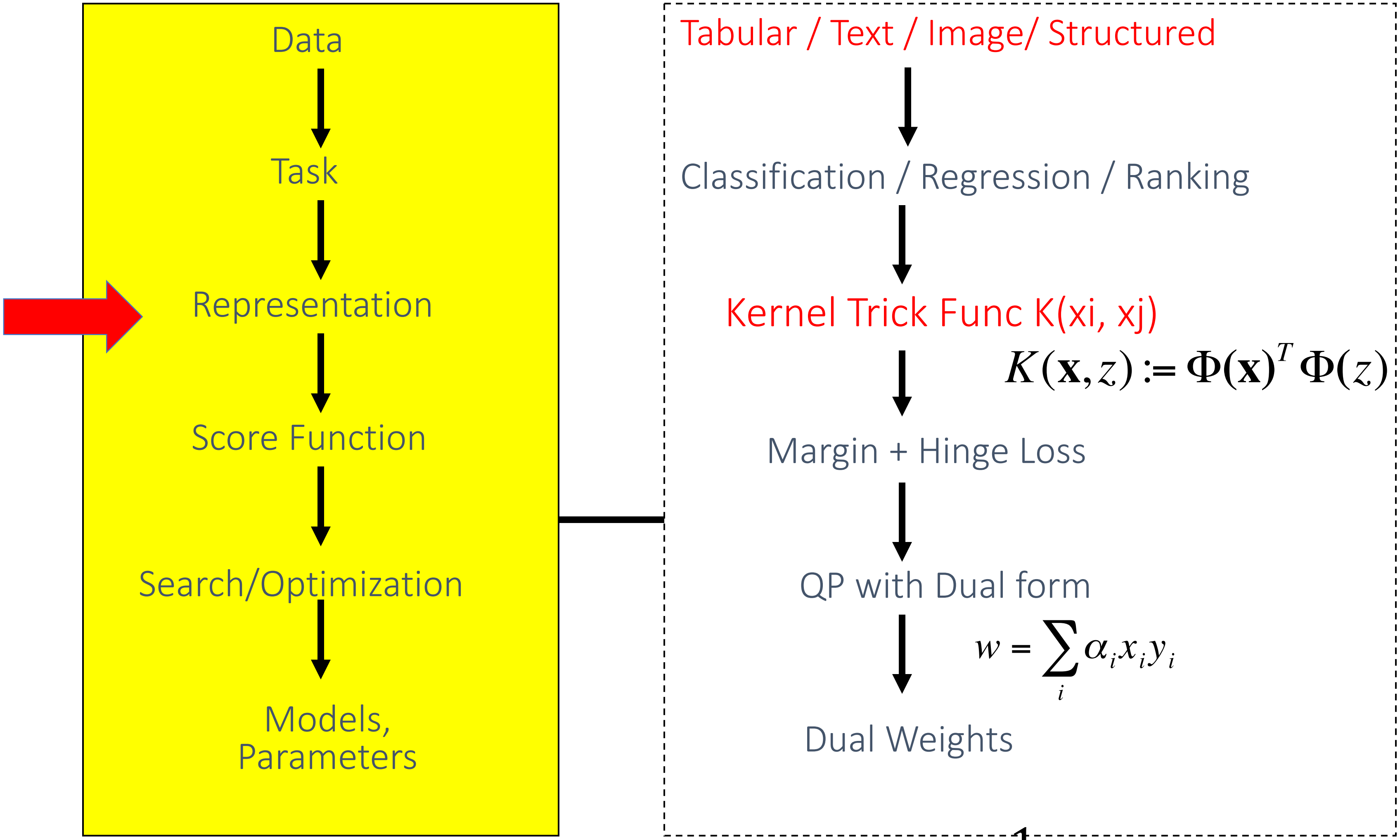
$$\underset{\mathbf{w},b}{\operatorname{argmin}} \sum_{i=1}^{p} w_i^2 + \mathrm{C}\sum_{\mathrm{i}=1}^{\mathrm{n}} \varepsilon_{\mathrm{i}}$$

subject to $\forall \mathbf{x}_i \in Dtrain : y_i\left(\mathbf{x}_i \cdot \mathbf{w} + b\right) \geq 1 - \varepsilon_{\mathrm{i}}$

$$\max_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_{\mathrm{i,j}} \alpha_i \alpha_j \mathrm{y}_{\mathrm{i}} y_j \mathbf{x}_{\mathbf{i}}^T \mathbf{x}_{\mathbf{j}}$$

$$\sum_i \alpha_i \mathrm{y}_{\mathrm{i}} = 0, \qquad \alpha_i \geq 0 \qquad \forall i$$

This: Kernel Support Vector Machine



Data → Task → Representation → Score Function → Search/Optimization → Models, Parameters

Tabular / Text / Image/ Structured

Classification / Regression / Ranking

Kernel Trick Func K(xi, xj)

$$K(\mathbf{x}, z) := \Phi(\mathbf{x})^T \Phi(z)$$

Margin + Hinge Loss

QP with Dual form

$$w = \sum_i \alpha_i x_i y_i$$

Dual Weights

$$\underset{\mathbf{w},b}{\text{argmin}} \sum_{i=1}^{p} w_i^2 + \mathrm{C}\sum_{\mathrm{i}=1}^{\mathrm{n}} \varepsilon_{\mathrm{i}}$$

subject to $\forall \mathbf{x}_i \in Dtrain : y_i\left(\mathbf{x}_i \cdot \mathbf{w} + b\right) \geq 1 - \varepsilon_{\mathrm{i}}$

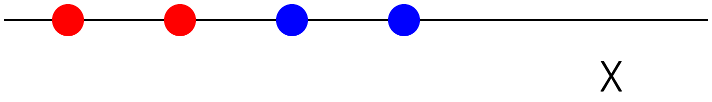
$$\max_\alpha \sum_i \alpha_i - \frac{1}{2}\sum_{\mathrm{i,j}} \alpha_i \alpha_j \mathrm{y}_{\mathrm{i}} y_j \mathbf{x}_{\mathbf{i}}^T \mathbf{x}_{\mathbf{j}}$$

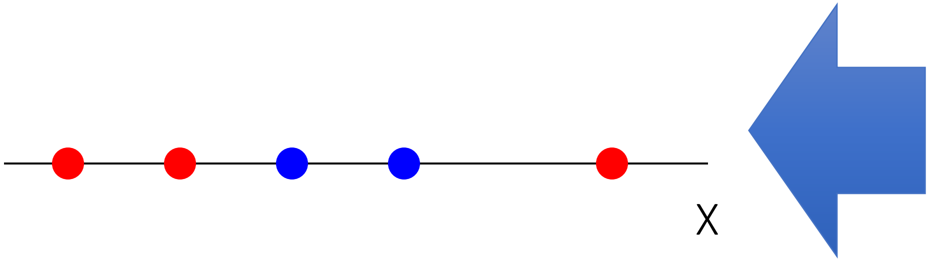
$$\sum_i \alpha_i \mathrm{y}_{\mathrm{i}} = 0, \qquad \alpha_i \geq 0 \qquad \forall i$$

Classifying in 1-d

Can an SVM correctly classify this data?

What about this?





Classifying in 1-d

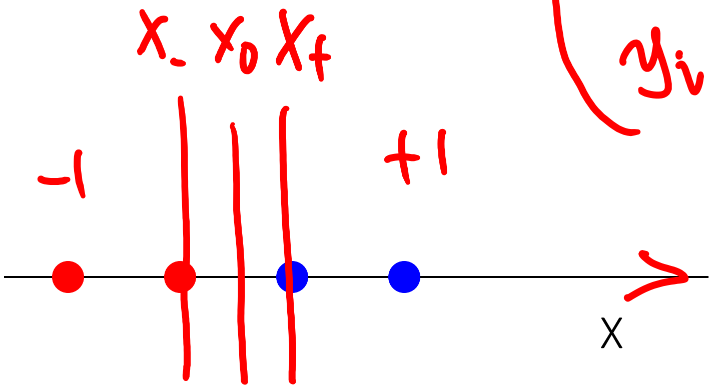
Can an SVM correctly classify this data?

$$M = \frac{2}{\sqrt{w^T w}} \Rightarrow \max M \Downarrow \min w^T w$$

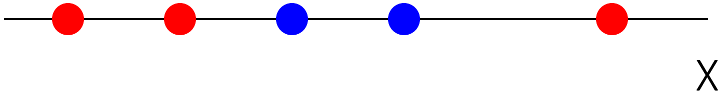
s.t.

$\forall i \in \text{train}$

$y_i(w^T x_i + b) \geq 1$



What about this?



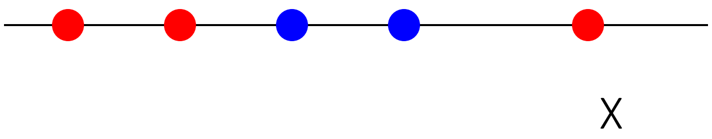
Classifying in 1-d

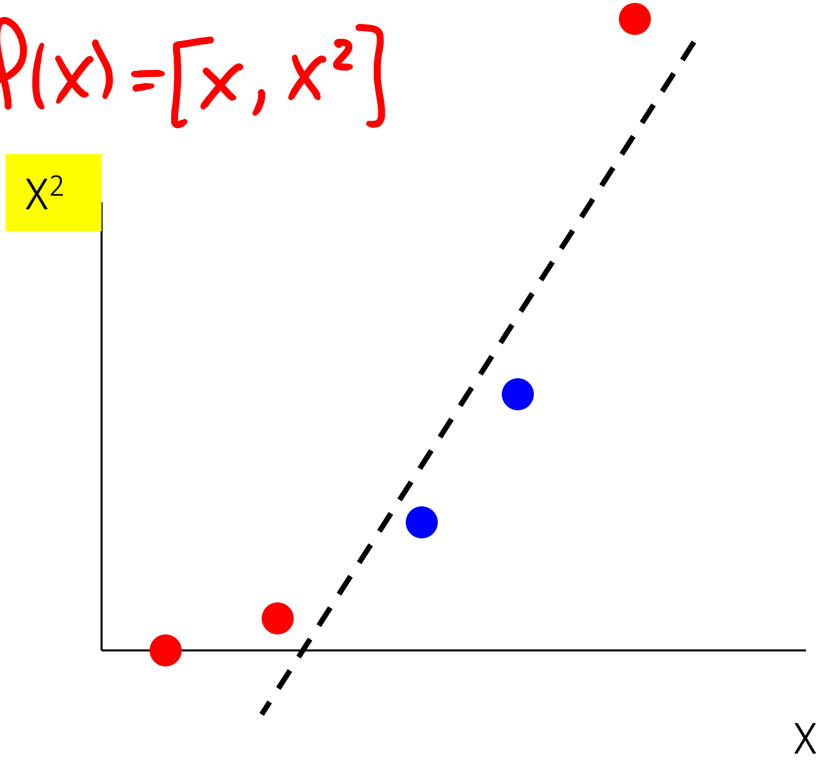
→ separable
→ nonlinear

Can an SVM correctly classify this data?

And now? (extend with polynomial basis)

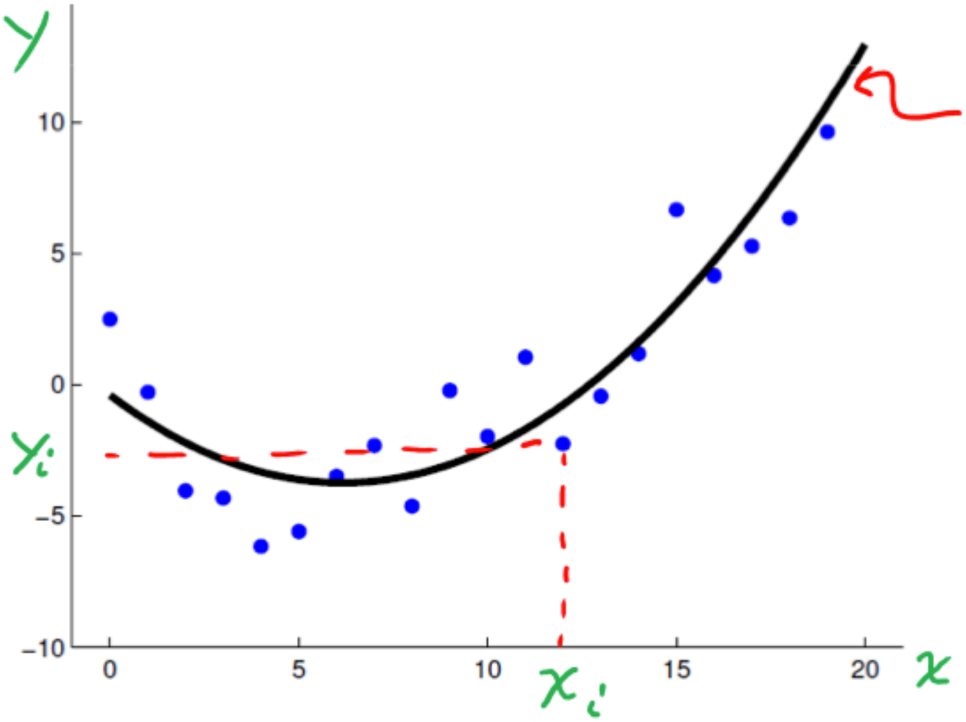
$(x) \longrightarrow \phi(x) = [x, x^2]$





RECAP: Polynomial regression

For example, $\phi(x) = [1, x, x^2]$

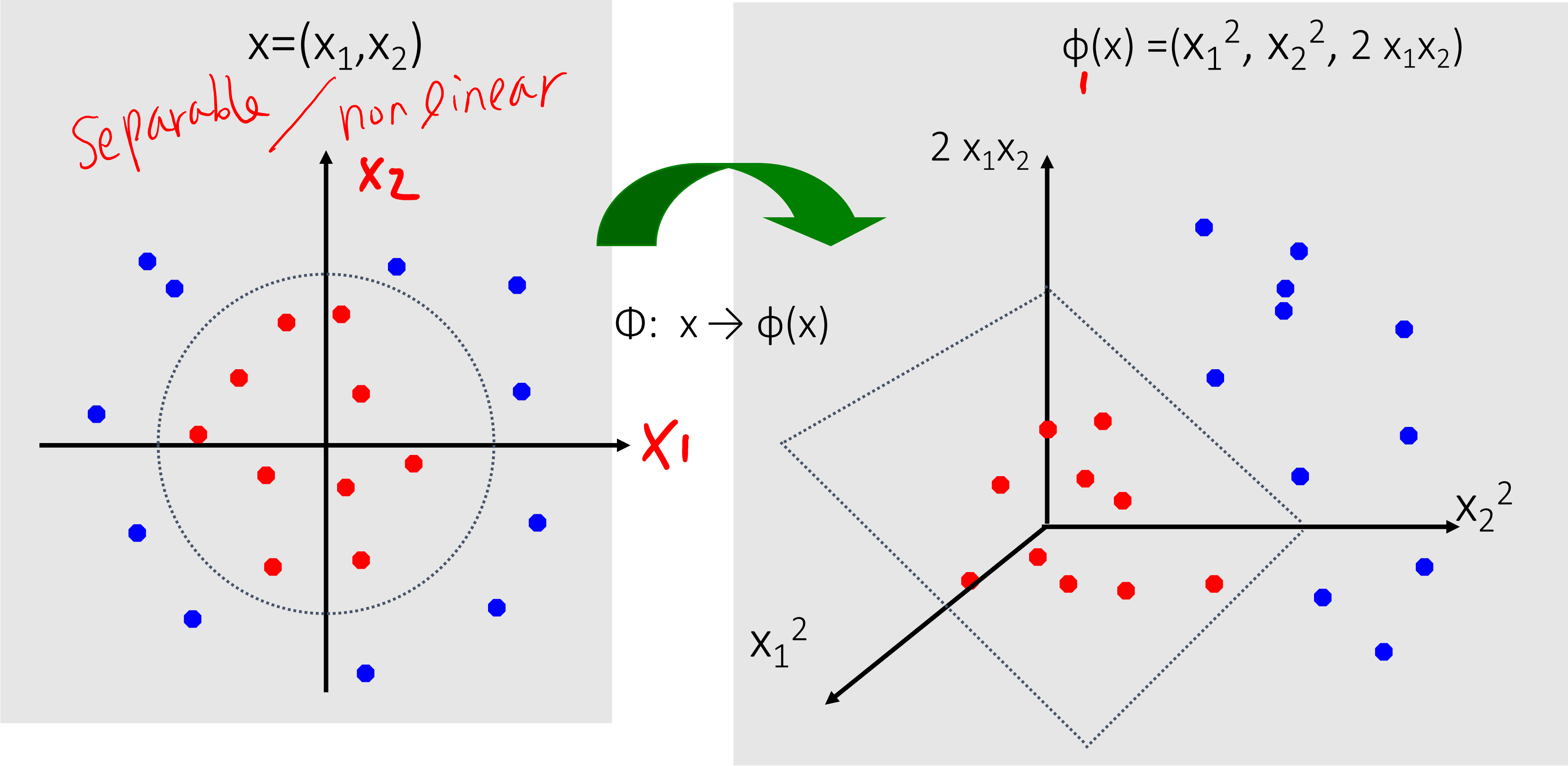


$$\hat{y} = \phi(x)\,\theta$$
$$= \theta_0 + x\theta_1 + x^2\theta_2$$

Dr. Nando de Freitas's tutorial slide

Non-linear SVMs: 2D

- The original input space (x) can be mapped to some higher-dimensional feature space ($\phi(x)$)where the training set is separable:



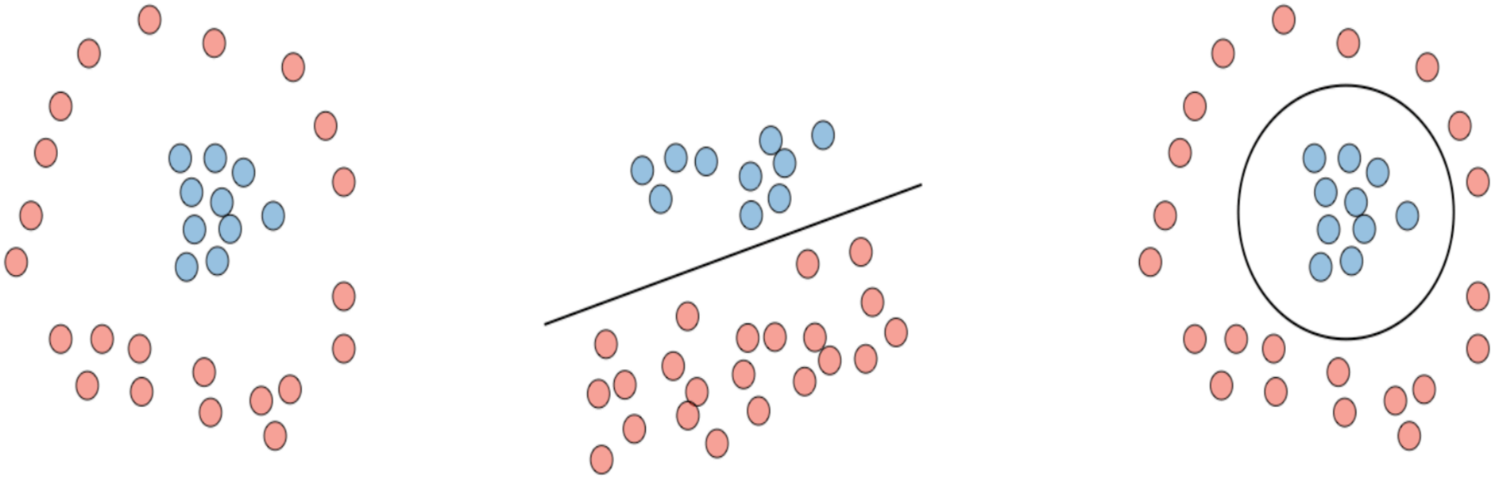
This slide is courtesy of www.iro.umontreal.ca/~pift6080/documents/papers/svm_tutorial.ppt

❒ **Kernel** – Given a feature mapping ϕ, we define the kernel K to be defined as:

$$\boxed{K(x,z)=\phi(x)^T\phi(z)}$$

In practice, the kernel K defined by $K(x,z) = \underbrace{\exp\left(-\frac{||x-z||^2}{2\sigma^2}\right)}_{\text{RBF}}$ is called the Gaussian kernel and is commonly used.

$\phi_2(x)^T\phi_2(z)$



Non-linear separability ➡ Use of a kernel mapping ϕ ➡ Decision boundary in the original space

~~Remark:~~ When *we say that we use the "kernel trick" to compute the cost function using the kernel ~~because~~ we actually don't need to know the explicit mapping ϕ, which is often very complicated. Instead, only the values $K(x,z)$ are needed.*

Credit: Stanford ML course

A little bit theory:

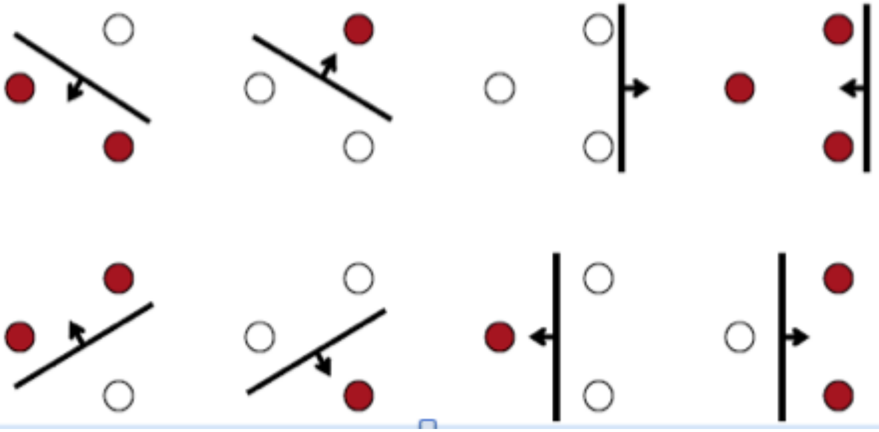
Vapnik-Chervonenkis (VC) dimension

$X \longrightarrow \phi(x)$

$\geq N-1$

If data is mapped into sufficiently high dimension, then samples will in general be linearly separable;
N data points are in general separable in a space of N-1 dimensions or more!!!

- VC dimension of the set of oriented lines in R^2 is 3
  - It can be shown that the VC dimension of the family of oriented separating hyperplanes in R^N is at least N+1



If data is mapped into sufficiently high dimension, then samples will in general be linearly separable;

N data points are in general separable in a space of N-1 dimensions or more!!!

$$X \longrightarrow \Phi(X)$$

linearly separated into two classes $\{+1, -1\}$

Thank You

Thank you

UVA CS 4774: Machine Learning

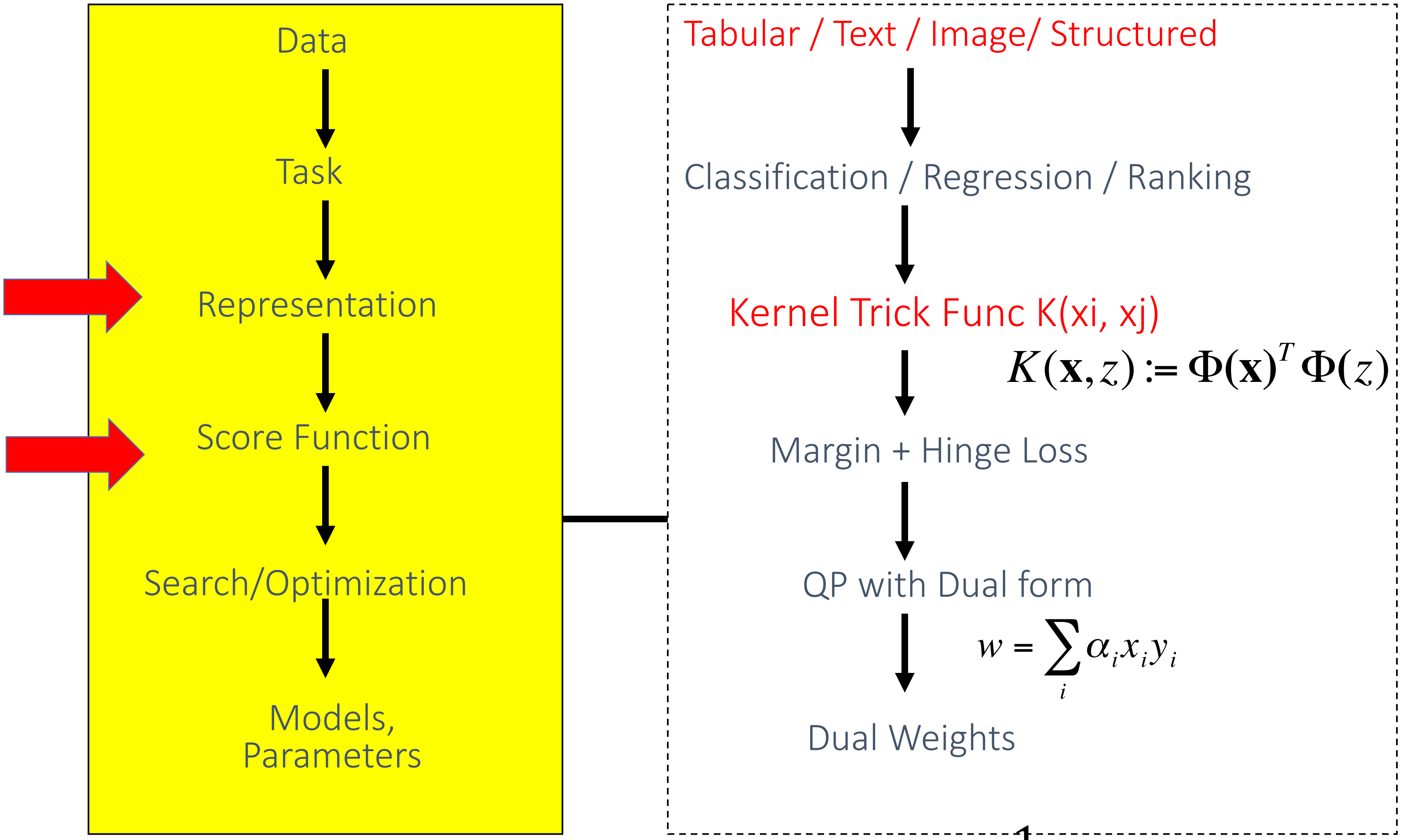
S4: Lecture 21: Support Vector Machine (nonlinear) Kernel Trick and in Practice

Dr. Yanjun Qi

University of Virginia
Department of Computer Science

Module II

Kernel Support Vector Machine



Data

Task

Representation

Score Function

Search/Optimization

Models, Parameters

Tabular / Text / Image/ Structured

Classification / Regression / Ranking

Kernel Trick Func K(xi, xj)

$$K(\mathbf{x}, z) := \Phi(\mathbf{x})^T \Phi(z)$$

Margin + Hinge Loss

QP with Dual form

$$w = \sum_i \alpha_i x_i y_i$$

Dual Weights

$$\underset{\mathbf{w},b}{\operatorname{argmin}} \sum_{i=1}^{p} w_i^2 + \mathrm{C}\sum_{i=1}^{n} \varepsilon_{\mathrm{i}}$$

$$\text{subject to } \forall \mathbf{x}_i \in Dtrain : y_i\left(\mathbf{x}_i \cdot \mathbf{w} + b\right) \geq 1 - \varepsilon_{\mathrm{i}}$$

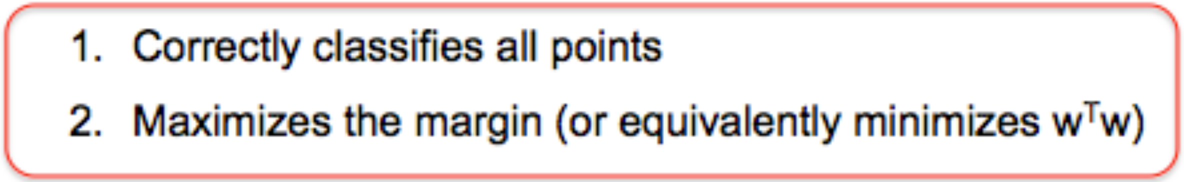
$$\max_\alpha \sum_i \alpha_i - \frac{1}{2}\sum_{\mathrm{i,j}} \alpha_i \alpha_j \mathrm{y}_{\mathrm{i}} y_j \mathbf{x}_{\mathbf{i}}^T \mathbf{x}_{\mathbf{j}}$$

$$\sum_i \alpha_i \mathrm{y}_{\mathrm{i}} = 0, \qquad \alpha_i \geq 0 \qquad \forall i$$

Optimization Reformulation (for linearly separable case)

$x_i \rightarrow \phi(x_i)$

f(x,w,b) = sign(w^{T}x + b)



1. Correctly classifies all points
2. Maximizes the margin (or equivalently minimizes w^{T}w)

$y_i \in \{+1, -1\}$

Min (w^{T}w)/2

subject to the following constraints:

For all x in class + 1

w^{T}x+b >= 1

For all x in class - 1

w^{T}x+b <= -1

A total of n constraints if we have n input samples

Quadratic Objective

$$\underset{\boxed{\mathbf{w},b}}{\operatorname{argmin}} \sum_{i=1}^{p} w_i^2 = \frac{1}{2} w^T W$$

$$\text{subject to } \forall \mathbf{x}_i \in Dtrain : \underbrace{y_i}_{1\times1}\left(\underbrace{\underbrace{\mathbf{w}^T}_{1\times P}\underbrace{\mathbf{x}_i}_{P\times1}}_{1\times1} + \underbrace{b}_{1\times1}\right) \geq 1$$

Quadratic programming i.e.,
- Quadratic objective
- Linear constraints

An alternative representation of the SVM QP

- Instead of encoding the correct classification rule and constraint we will use Lagrange multiplies to encode it as part of the our minimization problem

Min $(w^T w)/2$

s.t.

$(w^T x_i + b) y_i >= 1$

Recall that Lagrange multipliers can be applied to turn the following problem:

$\forall i, \alpha_i \geq 0$ every training

$$L_{primal}(w,b,\alpha)=\frac{1}{2}\mathbf{w}^{T}\cdot\mathbf{w}-\sum_{i=1}^{N}\underset{m}{\alpha_i}\left(\underbrace{y_i(\mathbf{w}^{T}\cdot\mathbf{x}_i+b)}_{\geq 1}-1\right)$$

The Dual Problem (Extra)

$$\max_{\alpha_i \geq 0} \underline{\min_{w,b} L(w,b,\alpha)}$$

Dual formulation

- We minimize L with respect to w and b first:

$$\nabla_w L(w,b,\alpha) = w - \sum_{i=1}^{train} \alpha_i y_i x_i = 0, \qquad (*)$$

$$\nabla_b L(w,b,\alpha) = \sum_{i=1}^{train} \alpha_i y_i = 0, \qquad (**)$$

$$f(x_t) = \text{sign}(\vec{w}^T \vec{x}_t + b)$$

Note that **(*)** implies:

$$\underline{w = \sum_{i=1}^{train} \alpha_i y_i x_i} \qquad (***)$$

- Plus (***) back to L , and using (**), we have:

$$L(w,b,\alpha) = \sum_{i=1} \alpha_i - \frac{1}{2} \sum_{i,j=1} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$$

SVM goal

Summary: Dual SVM for linearly separable case

Dual formulation

$$\max_\alpha \sum_i \alpha_i - \frac{1}{2}\sum_{\mathrm{i,j}} \alpha_i \alpha_j \mathrm{y}_\mathrm{i} y_j \mathbf{x}_\mathbf{i}^T \mathbf{x}_\mathbf{j}$$

$$\sum_i \alpha_i \mathrm{y}_\mathrm{i} = 0$$

$$\alpha_i \geq 0 \qquad \forall i$$

n α_i

Min $(w^Tw)/2$

subject to the following inequality constraints:

For all x in class + 1

$w^Tx+b >= 1$

For all x in class - 1

$w^Tx+b <= -1$

A total of n constraints if we have n input samples

Easier than original QP, more efficient algorithms exist to find $a_{i;}$ e.g. SMO (see extra slides)

Dual SVM for linearly separable case – Training / Testing

$$\{\alpha_1, \alpha_2, \ldots\ldots, \alpha_n\}$$

Our dual target function:

$$\max_{\alpha} \sum_{i} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \mathrm{y}_i y_j \mathbf{x}_{\mathbf{i}}^{T} \mathbf{x}_{\mathbf{j}}$$

$$\sum_{i} \alpha_i \mathrm{y}_{\mathrm{i}} = 0$$

$$\alpha_i \geq 0 \qquad \forall i$$

Dot product for all training samples

$$\begin{cases} \text{most } \alpha_i = 0 \\ \text{only support vectors } \alpha_i > 0 \end{cases}$$

Dual SVM for linearly separable case – Training / Testing

Our dual target function:

$$\max_{\alpha} \sum_{i} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1} \alpha_i \alpha_j \mathrm{y}_{\mathrm{i}} y_j \mathbf{x_i}^T \mathbf{x_j}$$

$$\sum_{i} \alpha_i \mathrm{y}_{\mathrm{i}} = 0$$

$$\alpha_i \geq 0 \qquad \forall i$$

Dot product for all training samples

Dot product with ("all" ??) training samples

To evaluate a new sample x_{ts} we need to compute:

$$\mathrm{w}^{\mathrm{T}} x_{ts} + b = \sum_{\mathrm{i}} \alpha_i \mathrm{y}_{\mathrm{i}} \mathbf{x_i}^T \mathbf{x}_{ts} + b$$

$$\xrightarrow{} K(x_i, x_{ts}) = \phi(x_i)^T \phi(x_{ts})$$

$$\widehat{y_{ts}} = \mathrm{sign}\left(\sum_{i \in SupportVectors} \alpha_i y_i \left(\mathbf{x}_i^T \mathbf{x}_{ts} \right) + b \right)$$

$$\max_\alpha \sum_i \alpha_i - \frac{1}{2}\sum_{\mathrm{i,j}} \alpha_i \alpha_j \mathrm{y}_\mathrm{i} y_j \mathbf{x}_\mathbf{i}{}^T \mathbf{x}_\mathbf{j}$$

$$\sum_i \alpha_i \mathrm{y}_\mathrm{i} = 0$$

$$C > \alpha_i \geq 0, \forall i$$

$x_i^T x_j = x_j^T x_i$

nonlinear $\Longrightarrow$

$$\max_\alpha \sum_i \alpha_i - \sum_{\mathrm{i,j}} \alpha_i \alpha_j \mathrm{y}_\mathrm{i} y_j \mathbf{\Phi}(\mathbf{x}_\mathbf{i})^T \mathbf{\Phi}(\mathbf{x}_\mathbf{j})$$

$$\sum_i \alpha_i \mathrm{y}_\mathrm{i} = 0$$

$$C > \alpha_i \geq 0, \forall i$$

$K(x_i, x_j)$

$$\begin{array}{c|ccccc} & 1 & 2 & \cdots & j & n \\ \hline 1 & & & & & \\ 2 & & & & & \\ i & & & & x_i^T x_j & \\ n & & & & & \end{array}$$

n×n

$O(p \cdot n^2/2)$

$$\begin{array}{c|ccccc} & 1 & 2 & \cdots & j & n \\ \hline 1 & & & & & \\ 2 & & & & & \\ i & & & & \Phi(x_i)^T \Phi(x_j) \rightarrow K(x_i, x_j) & \\ n & & & & & \end{array}$$

n×n

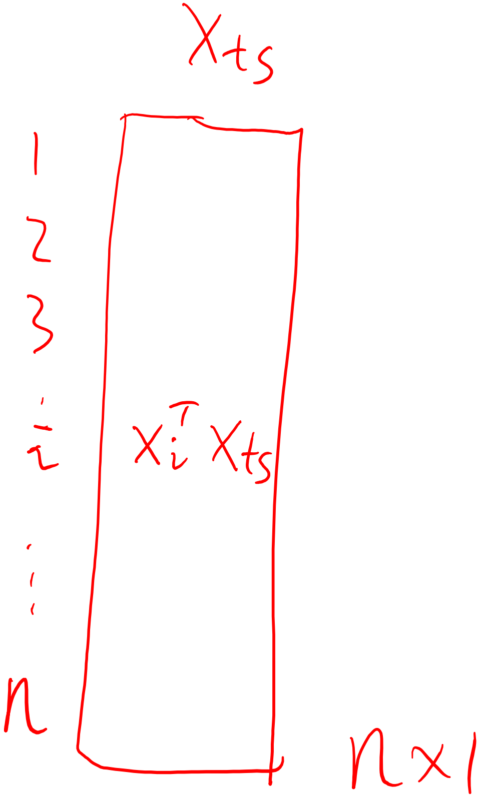
$O(p \cdot n^2/2)$

Training

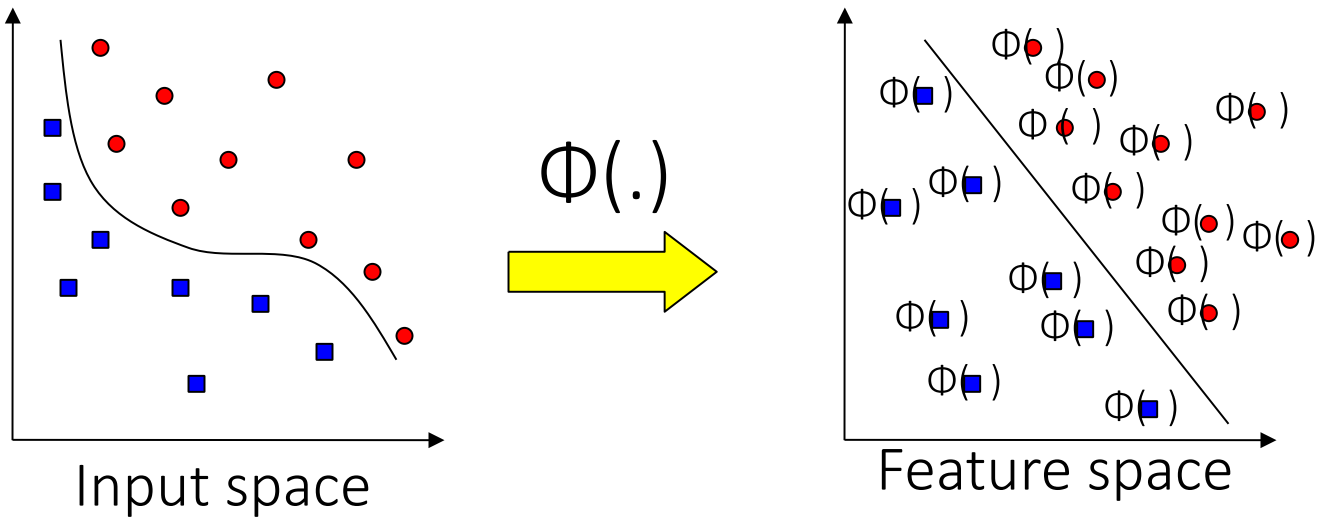
$$\mathrm{w}^{\mathrm{T}} x_{ts} + b = \sum_{\mathrm{i}} \alpha_i \mathrm{y}_{\mathrm{i}} \mathbf{x}_{\mathbf{i}}^{T} \mathbf{x}_{ts} + b$$

$$\widehat{y_{ts}} = \mathrm{sign}\left(\sum_{i \in SupportVectors} \alpha_i y_i \left(\mathbf{x}_i^T \mathbf{x}_{ts} \right) + b \right)$$

$$\Rightarrow \sum_{SV} \alpha_i y_i \underbrace{\Phi(x_i)\Phi(x_{ts})}_{} + b$$



Testing



SVM solves these two issues simultaneously

- “Kernel tricks” for efficient computation
- Dual formulation only assigns parameters to samples, not to features

- SVM solves these two issues simultaneously
  - “Kernel tricks” for efficient computation
  - Dual formulation only assigns parameters to samples, not features

(1). “Kernel tricks” for efficient computation

Never represent features explicitly
⍰ Compute dot products in closed form
Very interesting theory – Reproducing Kernel Hilbert Spaces
⍰ Not covered in detail here

$k(x, z)$

- SVM solves these two issues simultaneously
  - “Kernel tricks” for efficient computation
  - Dual formulation only assigns parameters to samples, not features

(1). “Kernel tricks” for efficient computation

~~$\Phi(x)$~~

Never represent features explicitly
⍰ Compute dot products in closed form
Very interesting theory – Reproducing Kernel Hilbert Spaces
⍰ Not covered in detail here

$k(x, z)$

$K(\mathbf{x}_i,\mathbf{x}_j) \equiv \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$ is called the kernel function.

- Linear kernel (we've seen it)

$$K(\mathbf{x},z)=\mathbf{x}^T z \qquad \begin{cases} x \in R^p \\ z \in R^p \end{cases}$$

- Polynomial kernel (we will see an example)

$$K(\mathbf{x},z)=\underbrace{\left(1+\mathbf{x}^T z\right)}_{O(p)}^d = \Phi_p(x)^T \underbrace{\Phi_p(z)}_{p_\Phi \to O(p^d)}$$

where d = 2, 3, … To get the feature vectors we concatenate all dth order polynomial terms of the components of x (weighted appropriately)

- Radial basis kernel

$$K(\mathbf{x},z)=\exp\left(-r\underbrace{\left\|\mathbf{x}-z\right\|^2}_{O(p)}\right) = \underbrace{\Phi_r(x)^T}_{} \Phi_r(z) \qquad p_\Phi = \infty$$

In this case., r is hyperpara. The feature space of the RBF kernel has an infinite number of dimensions

Never represent features explicitly
 Compute dot products with a closed form
Very interesting theory – Reproducing Kernel Hilbert Spaces
 Not covered in detail here

Example: Quadratic kernels

$$K(\mathbf{x},z)=\left(1+\mathbf{x}^{T}z\right)^{d}$$ ⇨ $(1+x^{T}z)^{2}$

$$\boxed{K(\mathbf{x},z) := \Phi(\mathbf{x})^{T}\Phi(z)}$$

- Consider all quadratic terms for x_1, x_2 … x_p

⇨ $$\Phi(x) = \begin{bmatrix} 1 \\ \sqrt{2}x_1 \\ \vdots \\ \sqrt{2}x_p \\ x_1^2 \\ \vdots \\ x_p^2 \\ \sqrt{2}x_1x_2 \\ \vdots \\ \sqrt{2}x_{p-1}x_p \end{bmatrix}$$

$$\max_{\alpha} \sum_{i} \alpha_i - \sum_{\mathrm{i,j}} \alpha_i \alpha_j \mathrm{y}_{\mathrm{i}} y_j \Phi(\mathbf{x_i})^{T}\Phi(\mathbf{x_j})$$

$$\sum_{i} \alpha_i \mathrm{y}_{\mathrm{i}} = 0$$

$$\alpha_i \geq 0 \qquad \forall i$$

$$K(\mathbf{x},z)=\left(1+\mathbf{x}^T z\right)^{\cancel{d}\,2} \quad [d=2], \quad [p=2] \quad \left(\begin{array}{l} x=(x_1,x_2) \\ z=(z_1,z_2) \end{array}\right)$$

$$\underbrace{K(x,z)} = \left(1+x_1 z_1 + x_2 z_2\right)^2 \Rightarrow \boxed{O(p)}$$

$$\boxed{O(p^2)} \left\{ \begin{array}{l} = \left(1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1 x_2\right)^T \\ \left(1, \sqrt{2}z_1, \sqrt{2}z_2, z_1^2, z_2^2, \sqrt{2}z_1 z_2\right) \end{array} \right.$$

$$= \underbrace{\Phi(x)^T \Phi(z)}$$

The kernel trick

$$\Phi(\mathbf{x})^T\Phi(z)$$

O(p^d) operations if using the basis function representations in building a poly-kernel matrix

So, if we define the kernel function as follows, there is no need to carry out basis function explicitly

$$K(\mathbf{x},z)=(1+x^Tz)^d$$

O(p) operations if building a poly-kernel matrix directly through the K(x,z) function ➔

This is because x^Tz gives a scalar, then its power of d only costs constant FLOPS.

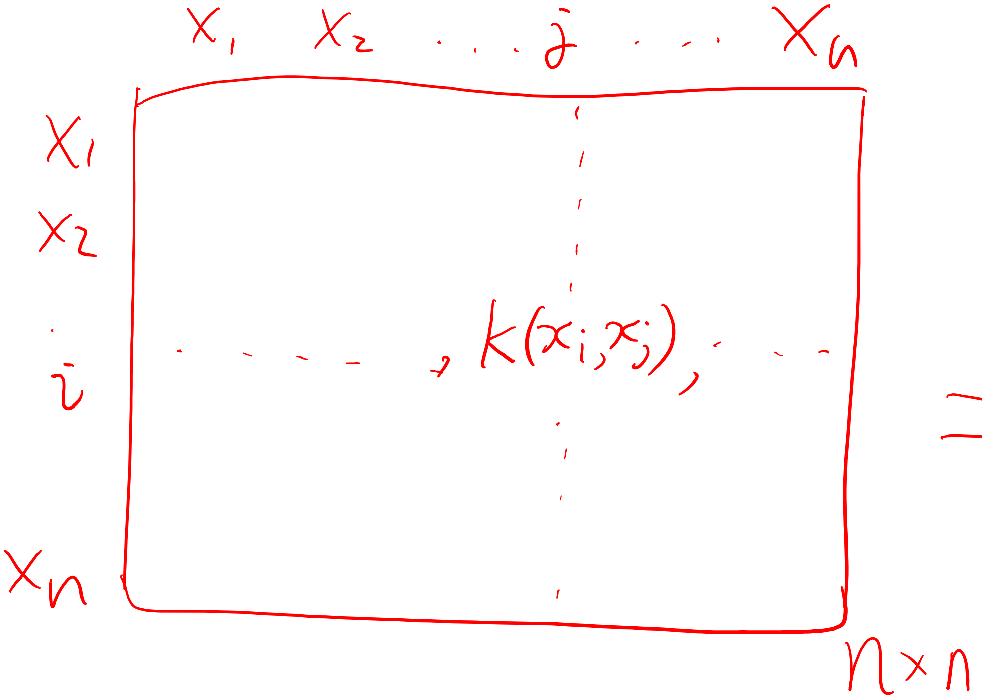
$$\max_\alpha \sum_i \alpha_i - \frac{1}{2}\sum_{\mathrm{i,j}} \alpha_i\alpha_j \mathrm{y}_\mathrm{i} y_j K(\mathbf{x_i},\mathbf{x_j})$$

$$\sum_i \alpha_i \mathrm{y}_\mathrm{i} = 0$$

$$C > \alpha_i \geq 0, \forall i \in train$$

Kernel Matrix

- Kernel function creates the kernel matrix, which summarize all the (train) data



$\Rightarrow K_{n \times n}$

[kernel matrix]

positive-semi-definite

Summary:
Modification Due to Kernel Trick

- Change all inner products to kernel functions
- For training,

Original Linear

$$\max_{\alpha} \sum_i \alpha_i - \frac{1}{2}\sum_{\mathrm{i,j}} \alpha_i \alpha_j \mathrm{y}_{\mathrm{i}} y_j \mathbf{x}_{\mathbf{i}}{}^T \mathbf{x}_{\mathbf{j}}$$

$$\sum_i \alpha_i \mathrm{y}_{\mathrm{i}} = 0$$

$$C > \alpha_i \geq 0, \forall i \in train$$

With kernel function - nonlinear

$$\max_{\alpha} \sum_i \alpha_i - \frac{1}{2}\sum_{\mathrm{i,j}} \alpha_i \alpha_j \mathrm{y}_{\mathrm{i}} y_j \underline{K(\mathbf{x}_{\mathbf{i}}, \mathbf{x}_{\mathbf{j}})}$$

VC: $\underline{\Phi(x)}$ large

$$\sum_i \alpha_i \mathrm{y}_{\mathrm{i}} = 0$$

$$C > \alpha_i \geq 0, \forall i \in train$$

Summary:
Modification Due to Kernel Trick

- For testing, the new data **x_ts**

Original Linear

$$\widehat{y_{ts}} = \text{sign}\left(\sum_{\mathrm{i} \in \mathrm{supportVectorn}} \alpha_i \mathrm{y}_\mathrm{i} \mathbf{x}_\mathbf{i}^T \mathbf{x}_{ts} + b \right)$$

With kernel function - nonlinear

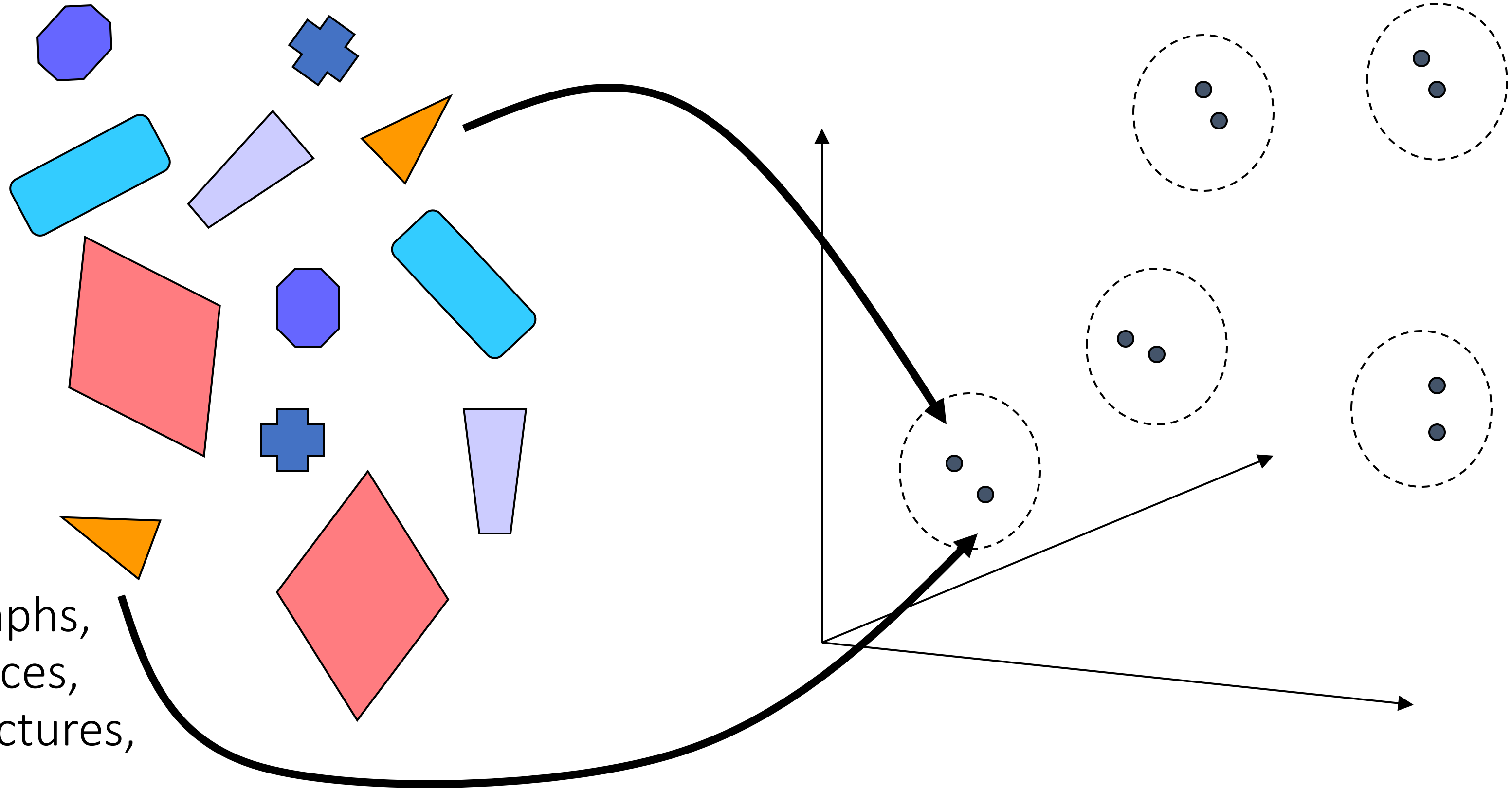
$$\widehat{y_{ts}} = \text{sign}\left(\sum_{\mathrm{i} \in \mathrm{supportVectors}} \alpha_i \mathrm{y}_\mathrm{i} K(\mathbf{x}_\mathbf{i}, \mathbf{x}_{ts}) + b \right)$$

Kernel trick has helped Non-traditional data like strings and trees able to be used as input to SVM, instead of feature vectors

When numerical x and z do not exist, we can calculate

$K(x,z)$

Vector vs. Relational data



e.g. Graphs,
Sequences,
3D structures,

Original Space

Feature Space

Thank You

Thank you

UVA CS 4774: Machine Learning

S4: Lecture 21: Support Vector Machine (nonlinear) Kernel Trick and in Practice

Module III

Dr. Yanjun Qi

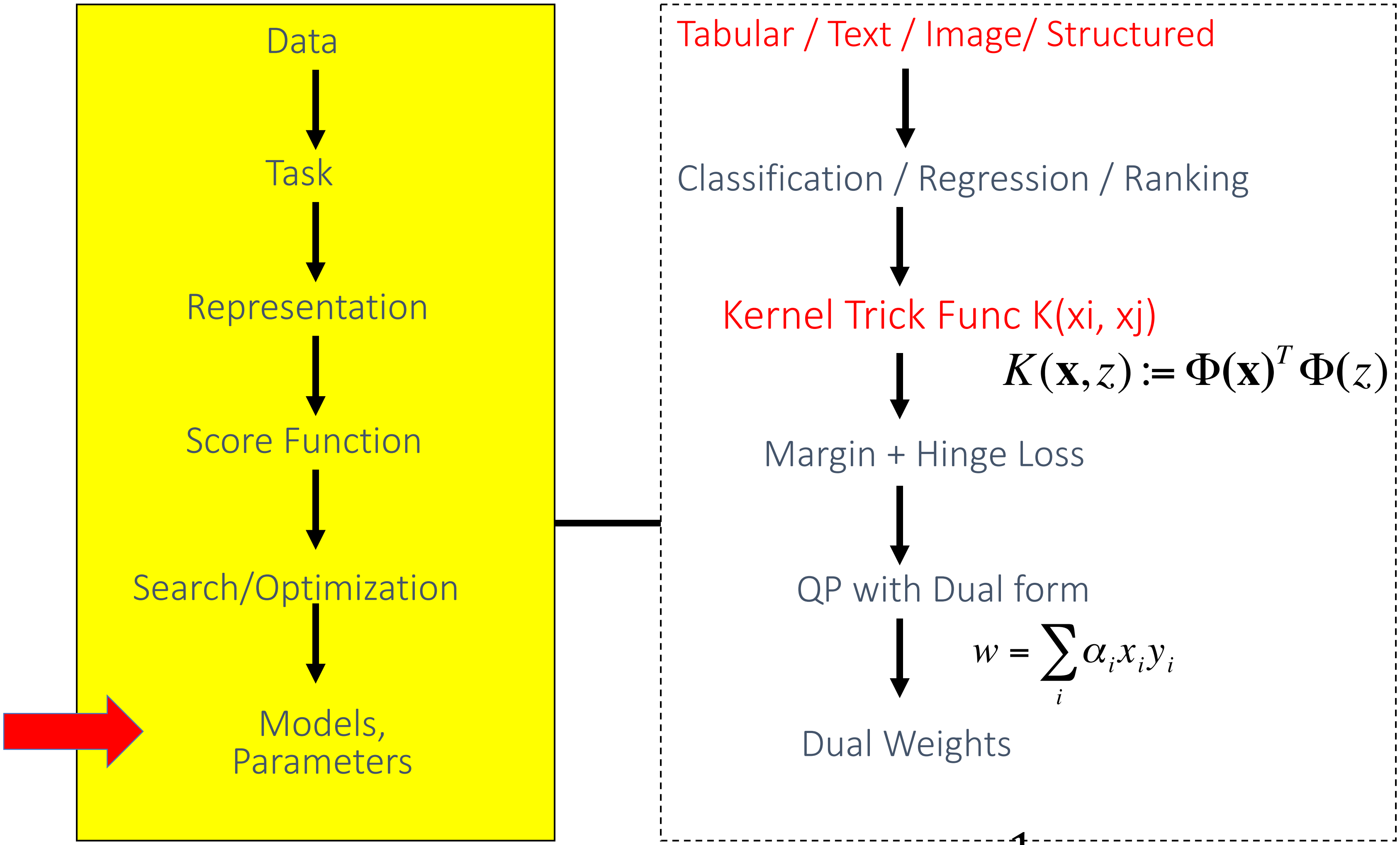
University of Virginia

Department of Computer Science

Today

- ❑ Support Vector Machine (SVM)
  - ✓ History of SVM
  - ✓ Large Margin Linear Classifier
  - ✓ Define Margin (M) in terms of model parameter
  - ✓ Optimization to learn model parameters (w, b)
  - ✓ Non linearly separable case
  - ✓ Optimization with dual form
  - ✓ Nonlinear decision boundary
  - ✓ Practical Guide

This: Kernel Support Vector Machine



Data

Task

Representation

Score Function

Search/Optimization

Models, Parameters

Tabular / Text / Image/ Structured

Classification / Regression / Ranking

Kernel Trick Func K(xi, xj)

$$K(\mathbf{x},z) := \Phi(\mathbf{x})^T \Phi(z)$$

Margin + Hinge Loss

QP with Dual form

$$w = \sum_i \alpha_i x_i y_i$$

Dual Weights

$$\underset{\mathbf{w},b}{\text{argmin}} \sum_{i=1}^{p} w_i^2 + \mathrm{C}\sum_{\mathrm{i}=1}^{\mathrm{n}} \varepsilon_{\mathrm{i}}$$

subject to $\forall \mathbf{x}_i \in Dtrain : y_i\left(\mathbf{x}_i \cdot \mathbf{w} + b\right) \geq 1 - \varepsilon_{\mathrm{i}}$

$$\max_\alpha \sum_i \alpha_i - \frac{1}{2}\sum_{\mathrm{i,j}} \alpha_i \alpha_j \mathrm{y}_{\mathrm{i}} y_j \mathbf{x}_{\mathbf{i}}^T \mathbf{x}_{\mathbf{j}}$$

$$\sum_i \alpha_i \mathrm{y}_{\mathrm{i}} = 0, \qquad \alpha_i \geq 0 \qquad \forall i$$

Software

- A list of SVM implementation can be found at
  - http://www.kernel-machines.org/software.html

- Some implementation (such as LIBSVM) can handle multi-class classification
- SVMLight is among one of the earliest implementation of SVM
- Several Matlab toolboxes for SVM are also available

Summary: Steps for Using SVM in HW

- Prepare the feature-data matrix
- Select the kernel function to use
- Select the parameter of the kernel function and the value of *C*
- Execute the training algorithm and obtain the \a$_i$
- Unseen data can be classified using the a_i and the support vectors

Practical Guide to SVM

- From authors of as LIBSVM:
  - A Practical Guide to Support Vector Classification Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin, 2003-2010
  - http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf

LIBSVM

- http://www.csie.ntu.edu.tw/~cjlin/libsvm/
    - ✓Developed by Chih-Jen Lin etc.
    - ✓Tools for Support Vector classification
    - ✓Also support multi-class classification
    - ✓C++/Java/Python/Matlab/Perl wrappers
    - ✓Linux/UNIX/Windows
    - ✓SMO implementation, fast!!!

A Practical Guide to Support Vector Classification

(a) Data file formats for LIBSVM

- Training.dat

```
+1 1:0.708333 2:1 3:1 4:-0.320755
-1 1:0.583333 2:-1  4:-0.603774 5:1
+1 1:0.166667 2:1 3:-0.333333 4:-0.433962
-1 1:0.458333 2:1 3:1 4:-0.358491 5:0.374429
…
```

- Testing.dat

(b) Feature Preprocessing

- (1) Categorical Feature
  - Recommend using m numbers to represent an m-category attribute.
  - Only one of the m numbers is one, and others are zero.

  - For example, a three-category attribute such as {red, green, blue} can be represented as (0,0,1), (0,1,0), and (1,0,0)

A Practical Guide to Support Vector Classification

A Practical Guide to Support Vector Classification

(b) Feature Preprocessing

- (2) Scaling before applying SVM is very important
  - to avoid attributes in greater numeric ranges dominating those in smaller numeric ranges.
  - to avoid numerical difficulties during the calculation
  - Recommend linearly scaling each attribute to the range [1, +1] or [0, 1].

① Normalization → $\begin{cases} \text{mean} & 0 \\ \text{std} & 1 \end{cases}$

② Scalig → linear ⇒ $[ax+b]$

e.g. $\left[\dfrac{X - X_{min}}{max - X_{min}}\right] \Leftarrow$

For i-th feature $\Rightarrow$ [Column operation on $X_{n \times p}$]

Centering : $X_i - \bar{X}_i \Rightarrow E(X_i) = 0$

Scaling : $aX_i + b \Rightarrow$ e.g. $\dfrac{X_i - \min(X_i)}{\max(X_i) - \min(X_i)}$

Normalization : $\Rightarrow \begin{cases} E(X_i) = 0 \\ var(X_i) = 1 \end{cases}$

[good practice : never touch test samples in any stage before testing]

Of course we have to use the same method to scale both training and testing data. For example, suppose that we scaled the first attribute of training data from $[-10, +10]$ to $[-1, +1]$. If the first attribute of testing data lies in the range $[-11, +8]$, we must scale the testing data to $[-1.1, +0.8]$. See Appendix B for some real examples.

If training and testing sets are separately scaled to $[0, 1]$, the resulting accuracy is lower than 70%.

```
$ ../svm-scale -l 0 svmguide4 > svmguide4.scale
$ ../svm-scale -l 0 svmguide4.t > svmguide4.t.scale
$ python easy.py svmguide4.scale svmguide4.t.scale
Accuracy = 69.2308% (216/312) (classification)
```

Using the same scaling factors for training and testing sets, we obtain much better accuracy.

```
$ ../svm-scale -l 0 -s range4 svmguide4 > svmguide4.scale
$ ../svm-scale -r range4 svmguide4.t > svmguide4.t.scale
$ python easy.py svmguide4.scale svmguide4.t.scale
Accuracy = 89.4231% (279/312) (classification)
```

(b) Feature Preprocessing

- (3) missing value
  - Very very tricky !
  - Easy way: to substitute the missing values by the mean value of the variable
  - A little bit harder way: imputation using nearest neighbors
  - Even more complex: e.g. EM based (beyond the scope)

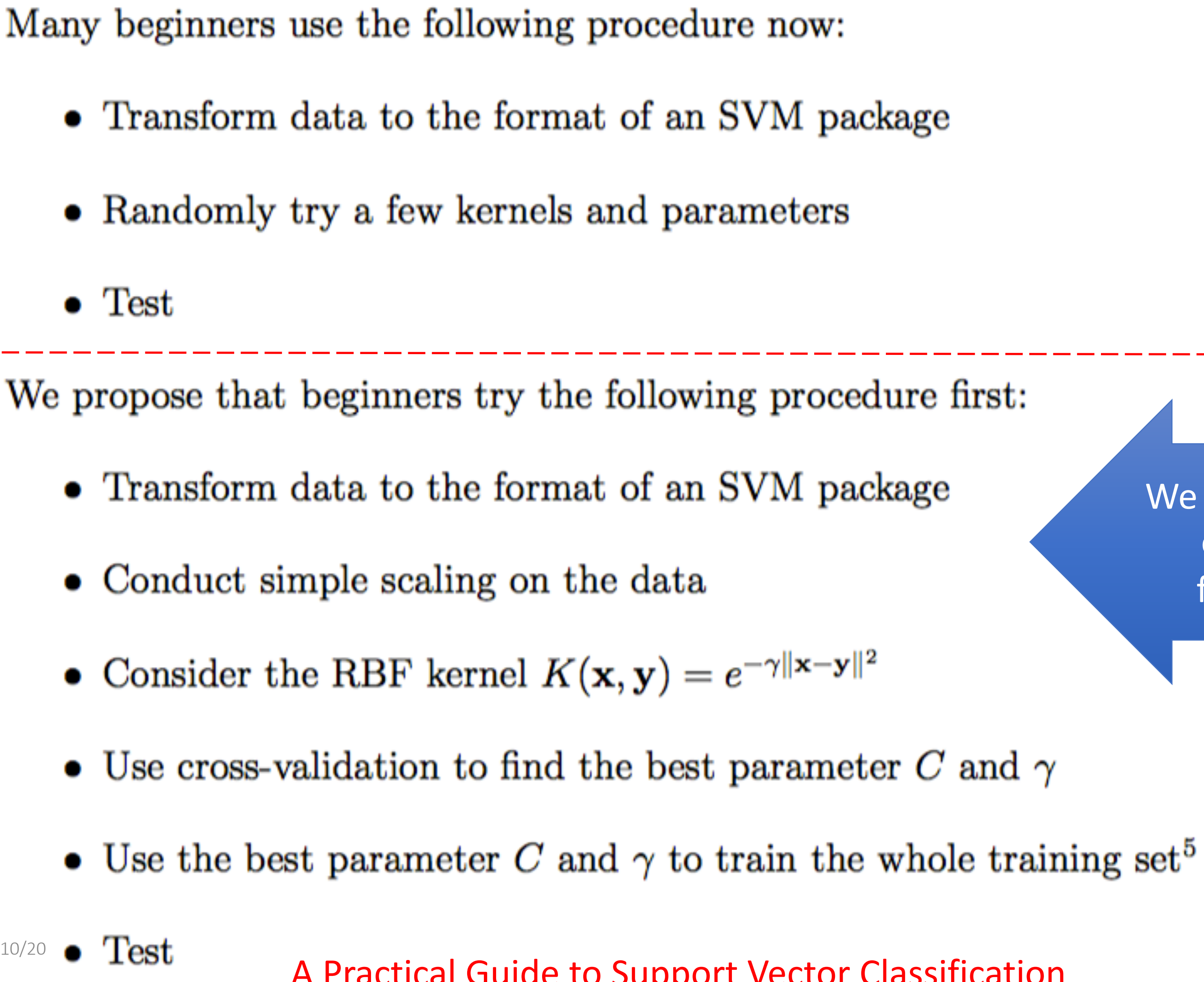
A Practical Guide to Support Vector Classification

(b) Feature Preprocessing

- (4) out of dictionary token issue
  - For discrete feature variable, very trick to handle
  - Easy way: to substitute the values by the most likely value (in train) of the variable
  - Easy way: to substitute the values by a random value (in train) of the variable
  - More solutions later in the NaiveBayes slides!

(C) Pipeline Procedures for model selection

- (I) train / test
- (II) k-folds cross validation
- (III) k-CV on train to choose hyperparameter / then test



Many beginners use the following procedure now:

- Transform data to the format of an SVM package
- Randomly try a few kernels and parameters
- Test

We propose that beginners try the following procedure first:

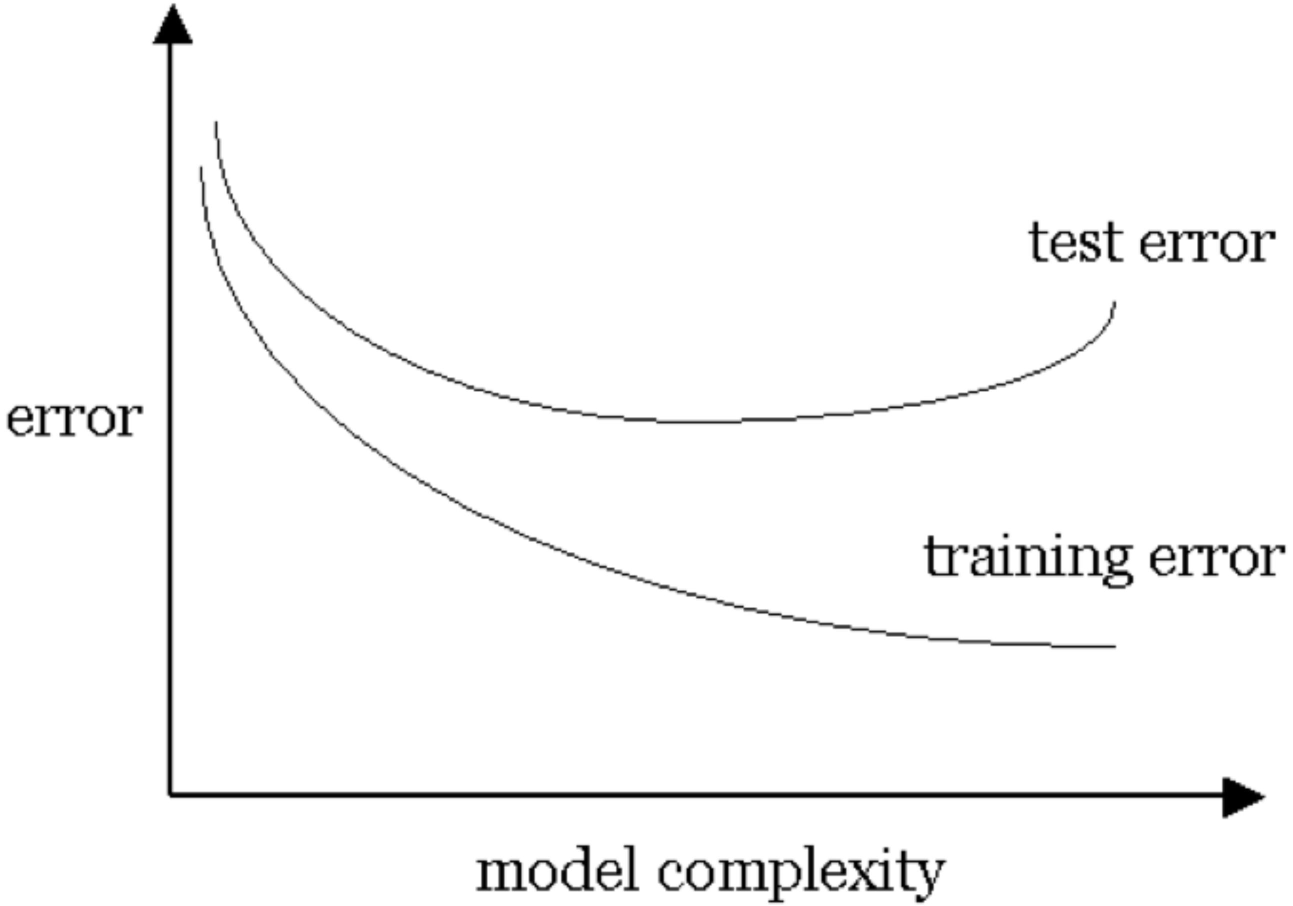
- Transform data to the format of an SVM package
- Conduct simple scaling on the data
- Consider the RBF kernel $K(\mathbf{x}, \mathbf{y}) = e^{-\gamma\|\mathbf{x}-\mathbf{y}\|^2}$
- Use cross-validation to find the best parameter C and γ
- Use the best parameter C and γ to train the whole training set[5]
- Test

We use lower option for HW

A Practical Guide to Support Vector Classification

(c) Model Selection

Our goal: find the model M which minimizes the test error:

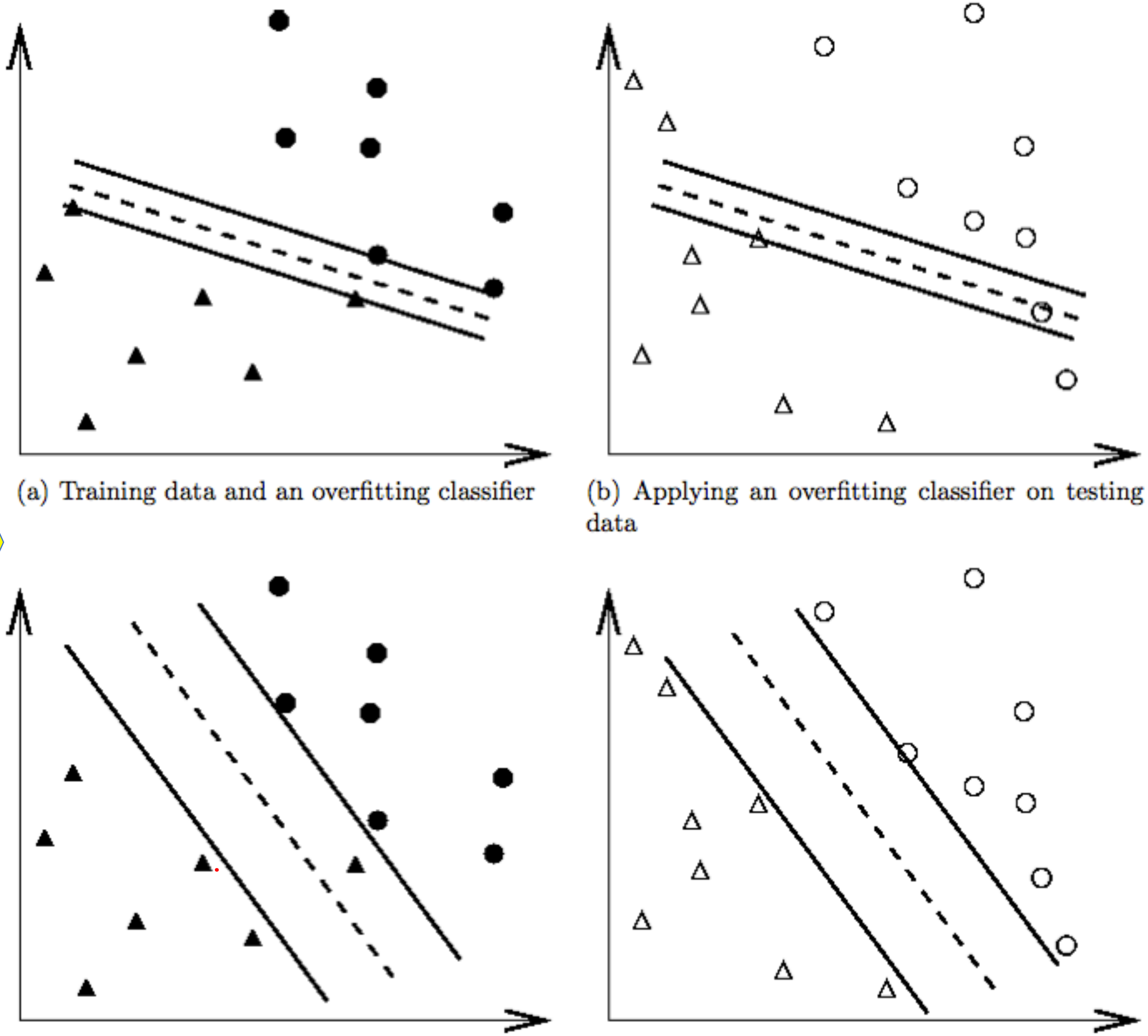


Model Selection, find right C

large C

Select the right penalty parameter C

small C



(a) Training data and an overfitting classifier

(b) Applying an overfitting classifier on testing data

(c) Training data and a better classifier

(d) Applying a better classifier on testing data

Model Selection, find right C

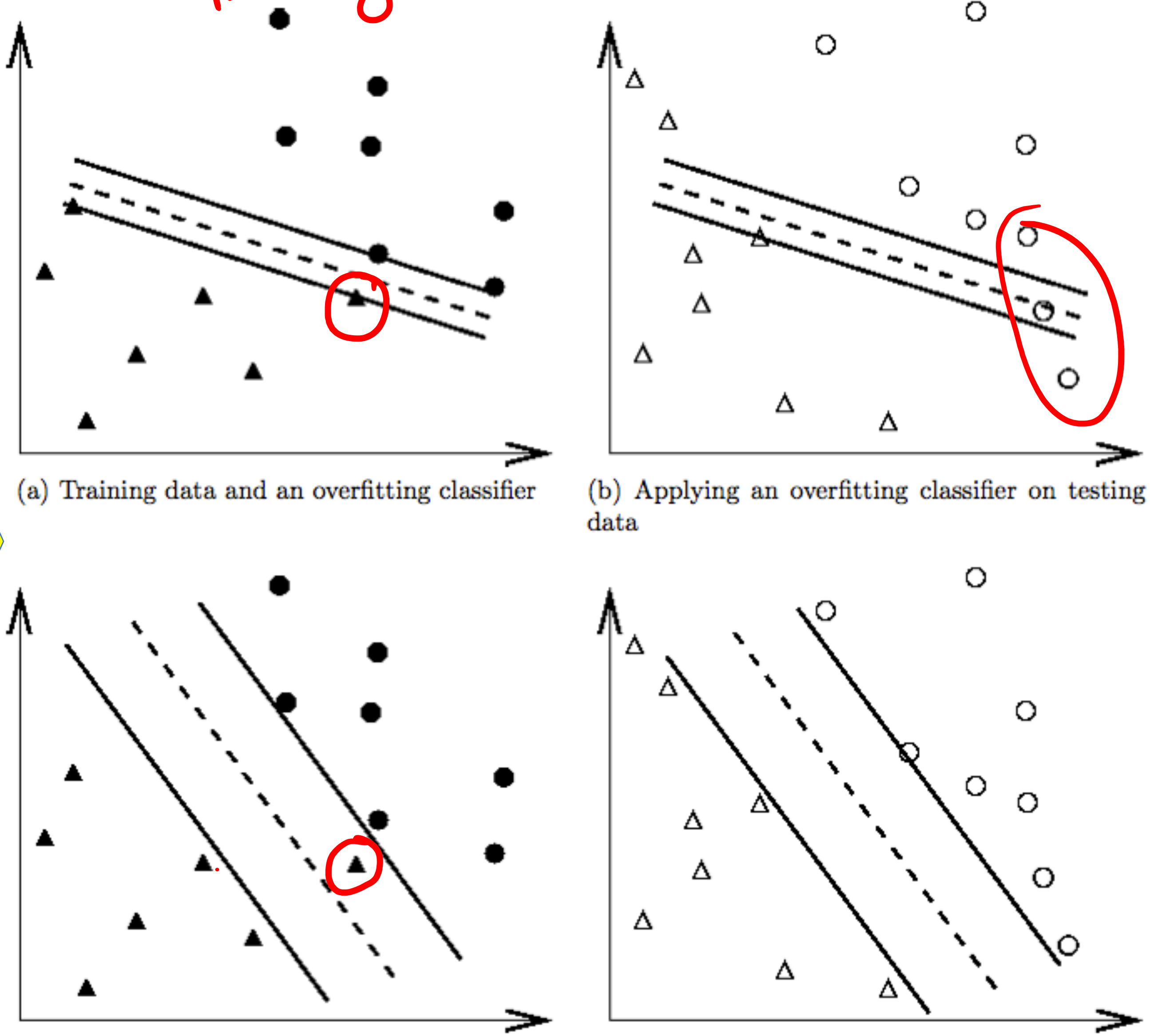
Training

Test

large C

Select the right penalty parameter C

small C



(a) Training data and an overfitting classifier

(b) Applying an overfitting classifier on testing data

(c) Training data and a better classifier

(d) Applying a better classifier on testing data

Model Selection, find right C

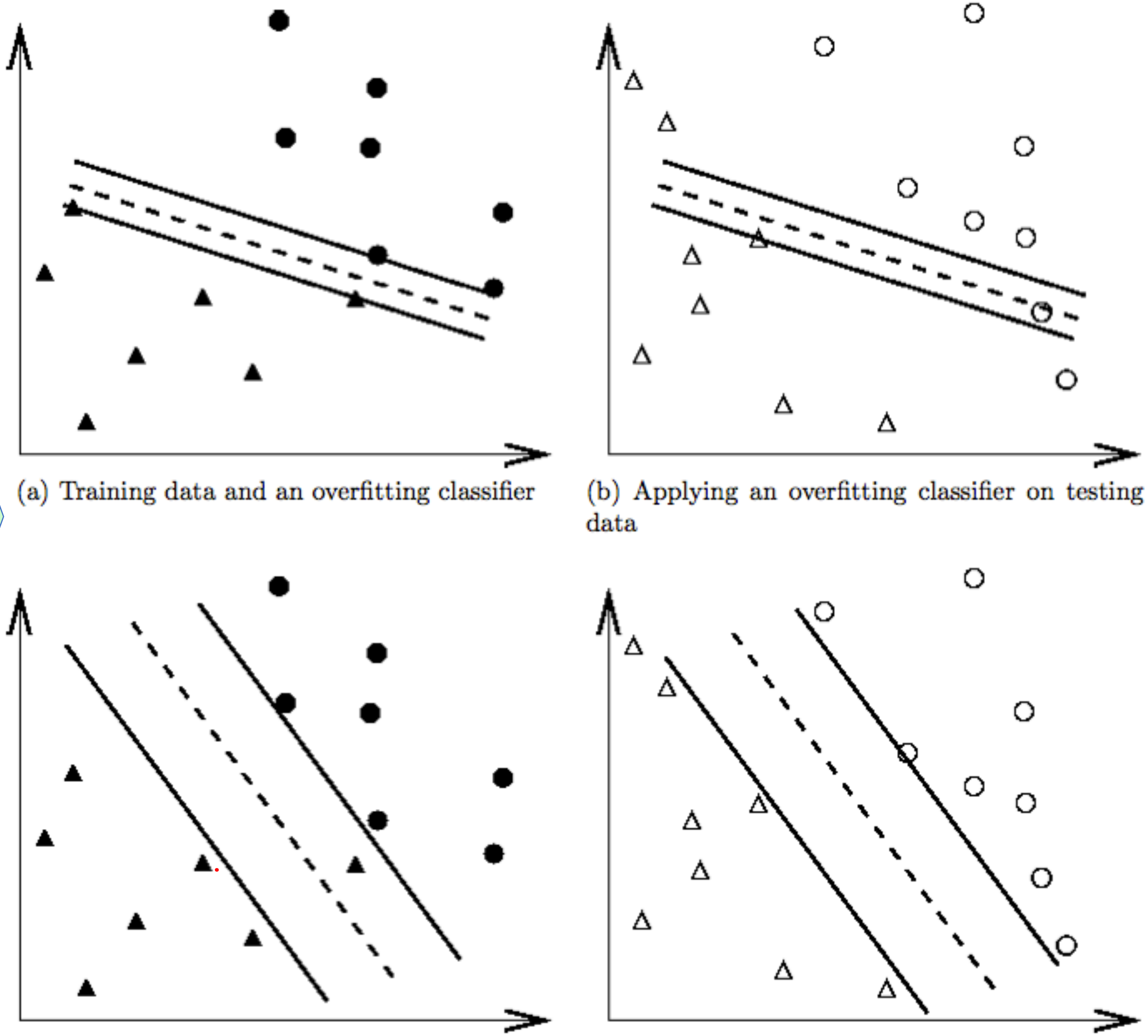
large C

A large value of C means that misclassifications are bad - resulting in smaller margins and less training error (but more expected true error).

may lead

A small C results in more training error, hopefully better true error.

small C



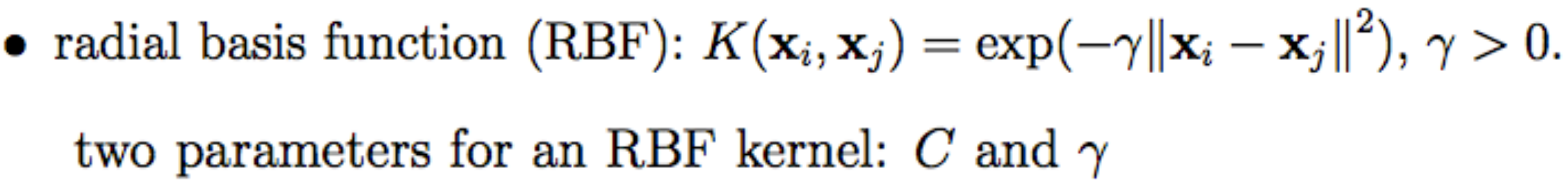
(a) Training data and an overfitting classifier

(b) Applying an overfitting classifier on testing data

(c) Training data and a better classifier

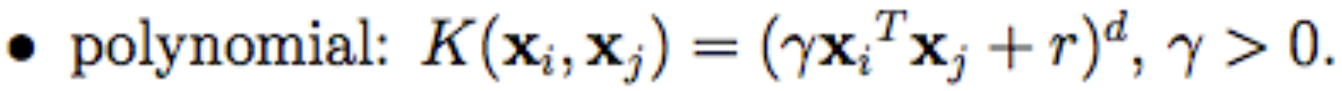
(d) Applying a better classifier on testing data

(c) Model Selection



- radial basis function (RBF): $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2)$, $\gamma > 0$.

  two parameters for an RBF kernel: C and γ



- polynomial: $K(\mathbf{x}_i, \mathbf{x}_j) = (\gamma \mathbf{x}_i{}^T \mathbf{x}_j + r)^d$, $\gamma > 0$.

  Three parameters for a polynomial kernel

A Practical Guide to Support Vector Classification

Choosing the Kernel Function

- Probably the most tricky part of using SVM.
- The kernel function is important because it creates the kernel matrix, which summarize all the data
- Many principles have been proposed (diffusion kernel, Fisher kernel, string kernel, tree kernel, graph kernel, ...)
  - Kernel trick has helped Non-traditional data like strings and trees able to be used as input to SVM, instead of feature vectors
- In practice, a low degree polynomial kernel or RBF kernel with a reasonable width is a good initial try for most applications.

Kernel Trick: Implicit Basis Representation

- For some kernels (e.g. RBF) the implicit transform basis form \phi(**x**) is infinite-dimensional!
  - But calculations with kernel are done in original space, so computational burden and curse of dimensionality aren't a problem.

$$\overset{O(p)}{\underline{K(\mathbf{x},z)}}=\exp\left(-r\left\|\mathbf{x}-z\right\|^{2}\right)$$

⇓

*O(p*n^2)* operations in building a RBF-kernel matrix for training

➔ Gaussian RBF Kernel corresponds to an infinite-dimensional vector space.

YouTube video of Caltech: Abu-Mostafa explaining this in more detailhttps://www.youtube.com/watch?v=XUj5JbQihlU&t=25m53s

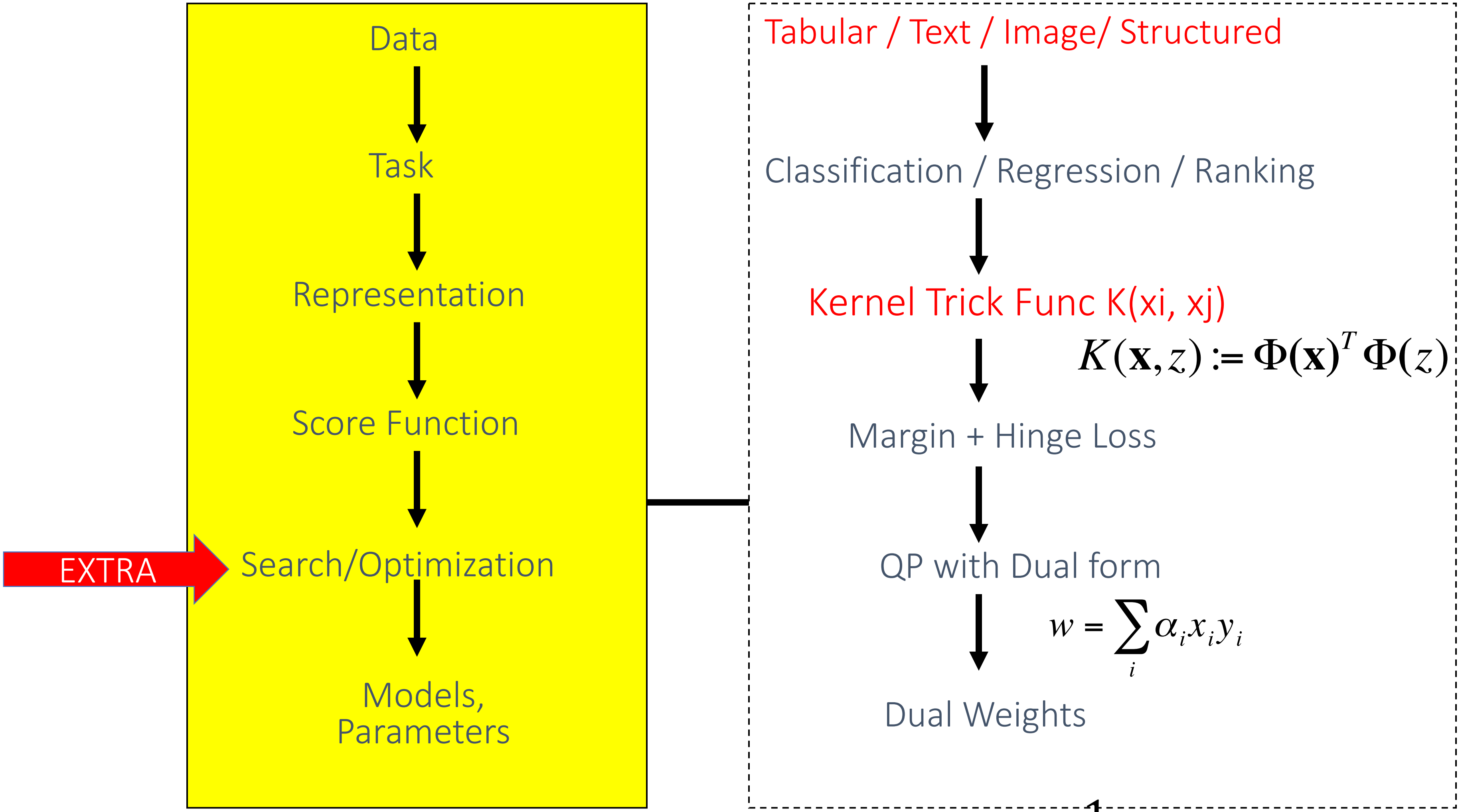
Kernel Functions (Extra)

- In practical use of SVM, only the kernel function (and not basis function) is specified
- Kernel function can be thought of as a similarity measure between the input objects
- Not all similarity measure can be used as kernel function, however Mercer's condition states that any positive semi-definite kernel *K*(*x*, *y*), i.e.

$$\sum_{i,j} K(x_i, x_j)c_i c_j \geq 0$$

can be expressed as a dot product in a high dimensional space.

This: Kernel Support Vector Machine



Data → Task → Representation → Score Function → Search/Optimization → Models, Parameters

EXTRA

Tabular / Text / Image/ Structured

Classification / Regression / Ranking

Kernel Trick Func K(xi, xj)

$$K(\mathbf{x}, z) := \Phi(\mathbf{x})^T \Phi(z)$$

Margin + Hinge Loss

QP with Dual form

$$w = \sum_i \alpha_i x_i y_i$$

Dual Weights

$$\underset{\mathbf{w},b}{\operatorname{argmin}} \sum_{i=1}^{p} w_i^2 + C\sum_{i=1}^{n} \varepsilon_i$$

$$\text{subject to } \forall \mathbf{x}_i \in Dtrain : y_i\left(\mathbf{x}_i \cdot \mathbf{w} + b\right) \geq 1 - \varepsilon_i$$

$$\max_\alpha \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\sum_i \alpha_i y_i = 0, \qquad \alpha_i \geq 0 \qquad \forall i$$

Why SVM Works? (Extra)

- Vapnik argues that the fundamental problem is not the number of parameters to be estimated. Rather, the problem is about the flexibility of a classifier

- Vapnik argues that the flexibility of a classifier should not be characterized by the number of parameters, but by the capacity of a classifier
  - This is formalized by the "VC-dimension" of a classifier

- The SVM objective can also be justified by structural risk minimization: the empirical risk (training error), plus a term related to the generalization ability of the classifier, is minimized

- Another view: the SVM loss function is analogous to ridge regression. The term $½||w||^2$ "shrinks" the parameters towards zero to avoid overfitting

Thank You

Thank you

References

- Big thanks to Prof. Ziv Bar-Joseph and Prof. Eric Xing @ CMU for allowing me to reuse some of his slides
- Elements of Statistical Learning, by Hastie, Tibshirani and Friedman
- Prof. Andrew Moore @ CMU's slides
- Tutorial slides from Dr. Tie-Yan Liu, MSR Asi
- A Practical Guide to Support Vector Classification Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin, 2003-2010
- Tutorial slides from Stanford "Convex Optimization I — Boyd & Vandenberghe

Mercer Kernel vs. Smoothing Kernel (Extra)

- The Kernels used in Support Vector Machines are different from the Kernels used in LocalWeighted /Kernel Regression.

- We can think
  - Support Vector Machines' kernels as **Mercer Kernels**
  - Local Weighted / Kernel Regression's kernels as **Smoothing Kernels**

KNN : $\hat{y}_{ts} = \frac{1}{k} \sum_{i \in kNeighbors\text{-}of\ X_{ts}} y_i$

find k neighbor of $X_{ts} \sim O(n\times)$

SVM : $\hat{y}_{ts} = \sum_{i \in SV} \alpha_i y_i K(\vec{X}_i, \vec{X}_{ts}) + b$

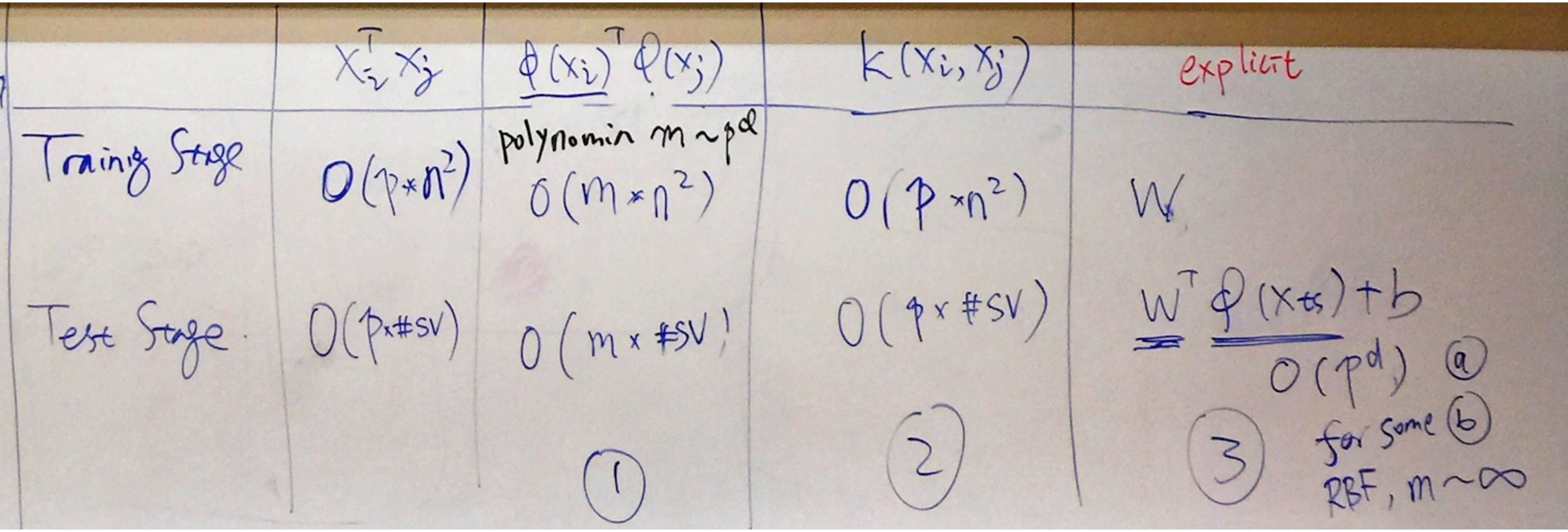
para $\sim O(n)$

Logistic Regression/ Linear Classifier

$\hat{y}_{ts} = \sigma(W^T X_{ts} + b)$

para $\sim O(p)$

Time Cost Comparisons



	$X_i^T X_j$	$\phi(x_i)^T \phi(x_j)$	$K(x_i, x_j)$	explicit
Training Stage	$O(p * n^2)$	polynomin $m \sim p^d$ $O(m * n^2)$	$O(p \times n^2)$	W
Test Stage	$O(p \times \#SV)$	$O(m \times \#SV)$	$O(p \times \#SV)$	$\underline{W}^T \underline{\phi(x_{ts})} + b$ $O(p^d)$ ⓐ for some ⓑ RBF, $m \sim \infty$
		①	②	③

Why do SVMs work?

$x \rightarrow \phi(x)$ e.g. RBF

- ❑ If we are using huge features spaces (e.g., with kernels), how come we are not overfitting the data?
  - ✓ Number of parameters remains the same (and most are set to 0) $O(n)$, α_i $i=1, \cdots, n$
  - ✓ While we have a lot of inputs, at the end we only care about the support vectors and these are usually a small group of samples
  - ✓ The maximizing of the margin acts as a sort of regularization term leading to reduced overfitting