Support Vector Machine

Outline
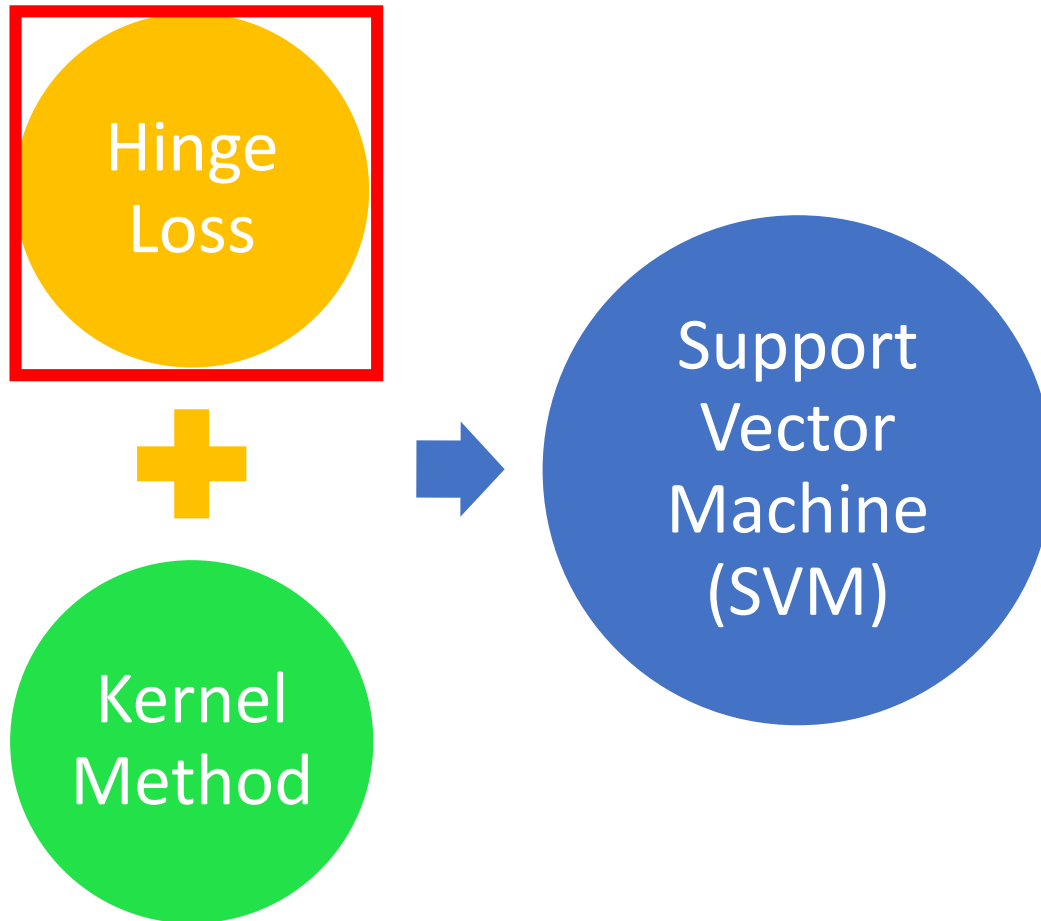
Binary Classification

$$x^1 \quad x^2 \quad x^3 \quad \cdots\cdots$$
$$\hat{y}^1 \quad \hat{y}^2 \quad \hat{y}^3$$

$$\hat{y}^n = +1, -1$$

- Step 1: Function set (Model)

$$g(x) = \begin{cases} f(x) > 0 & \text{Output} = +1 \\ f(x) < 0 & \text{Output} = -1 \end{cases}$$

- Step 2: Loss function:

$$L(f) = \sum_n \cancel{\delta(g(x^n) \neq \hat{y}^n)} \\ l(f(x^n), \hat{y}^n)$$

The number of times g get incorrect results on training data.

- Step 3: Training by gradient descent is difficult

Gradient descent is possible if $g(*)$ and $\delta(*)$ is differentiable
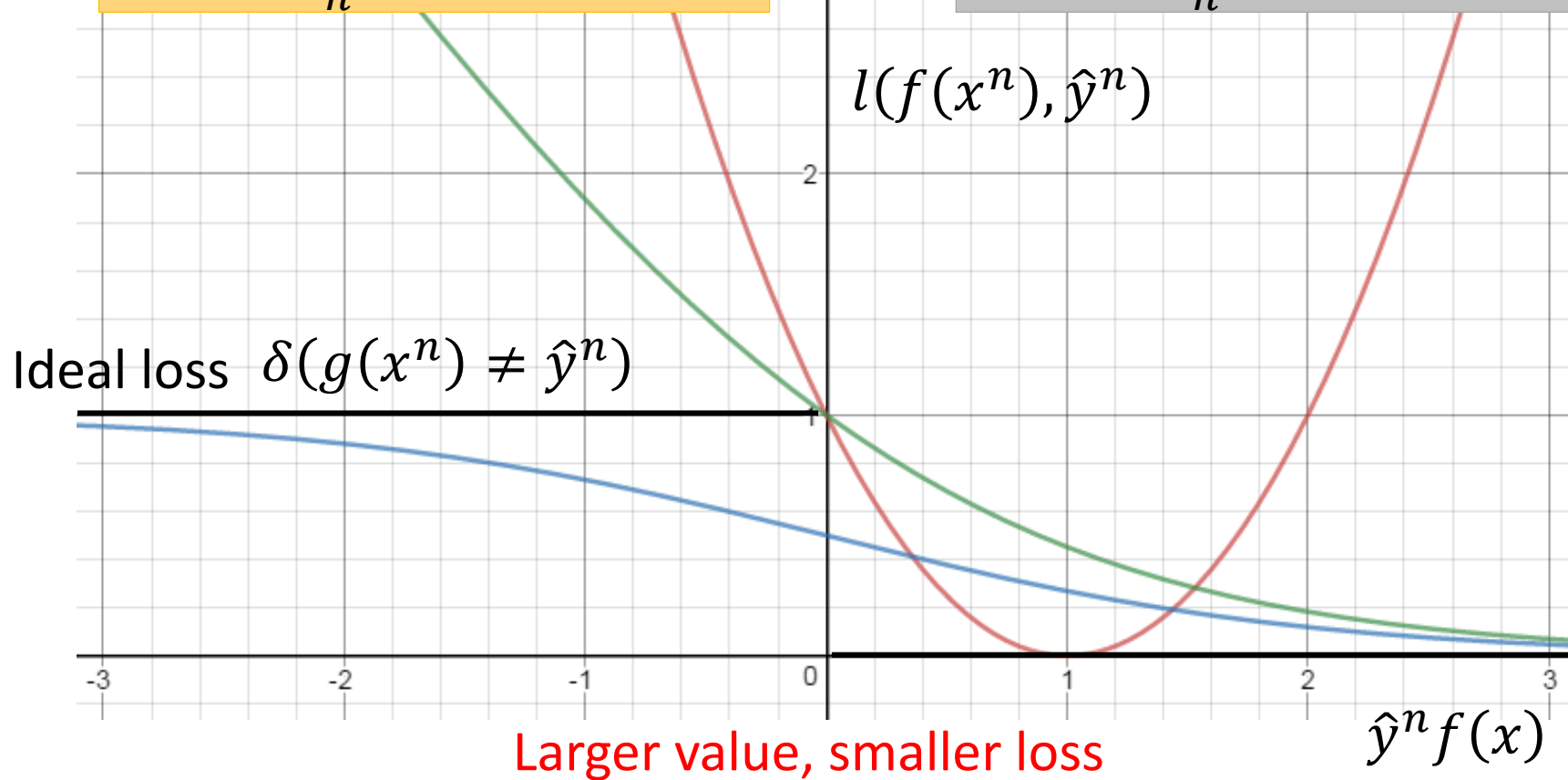
Step 2: Loss function

$$g(x) = \begin{cases} f(x) > 0 & \text{Output} = +1 \\ f(x) < 0 & \text{Output} = -1 \end{cases}$$

Ideal loss:

$$L(f) = \sum_n \delta(g(x^n) \neq \hat{y}^n)$$

Approximation:

$$L(f) = \sum_n l(f(x^n), \hat{y}^n)$$

$l(f(x^n), \hat{y}^n)$

Ideal loss $\delta(g(x^n) \neq \hat{y}^n)$

Larger value, smaller loss

$\hat{y}^n f(x)$

Step 2: Loss function

Square Loss:
If $\hat{y}^n = 1$, $f(x)$ close to 1
If $\hat{y}^n = -1$, $f(x)$ close to -1

$$(f(x^n) - 1)^2$$

$$l(f(x^n), \hat{y}^n) = (\hat{y}^n f(x^n) - 1)^2$$

$$(-f(x^n) - 1)^2 \qquad (f(x^n) + 1)^2$$
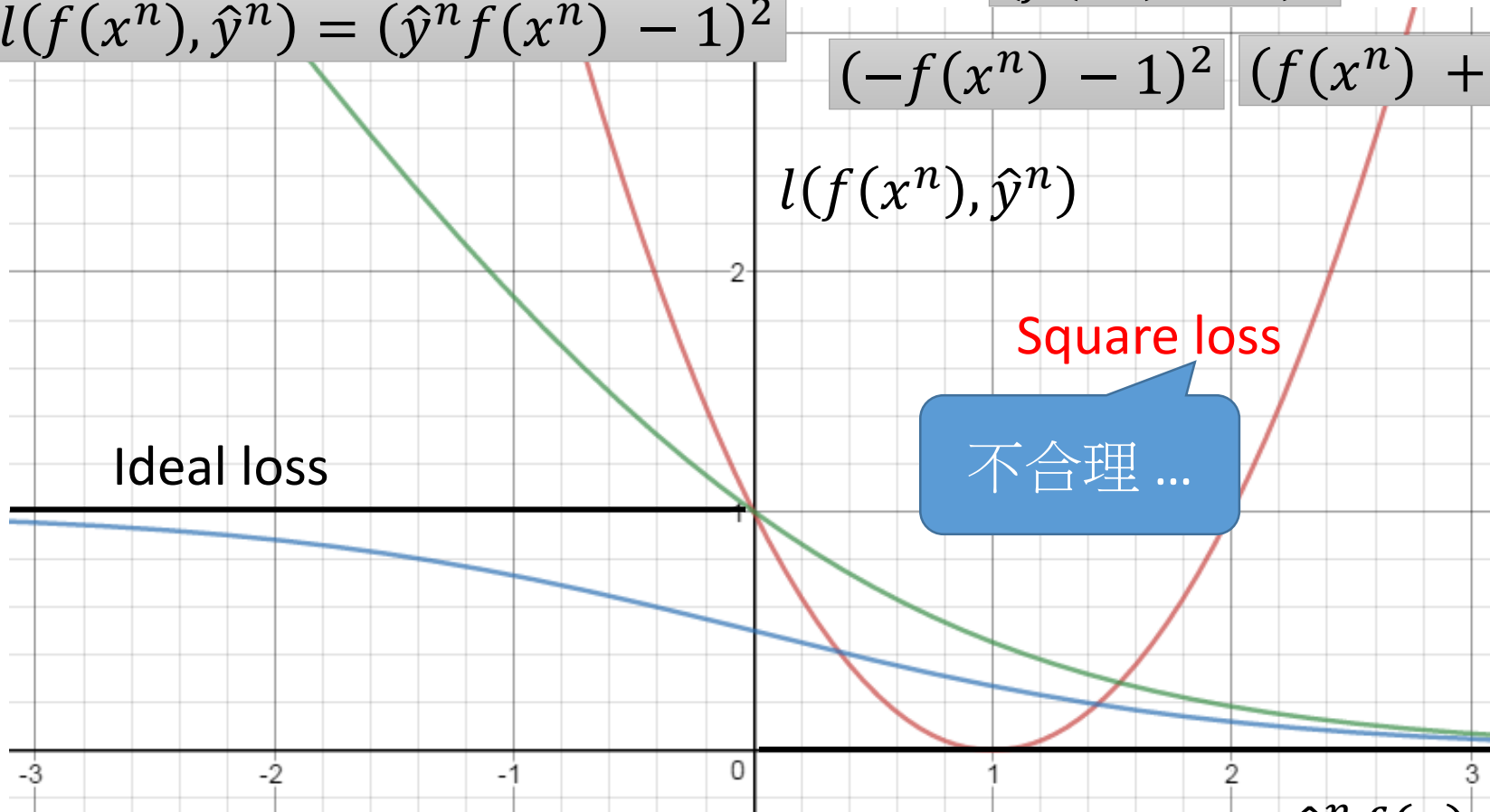
$$l(f(x^n), \hat{y}^n)$$

Square loss

不合理 …

Ideal loss

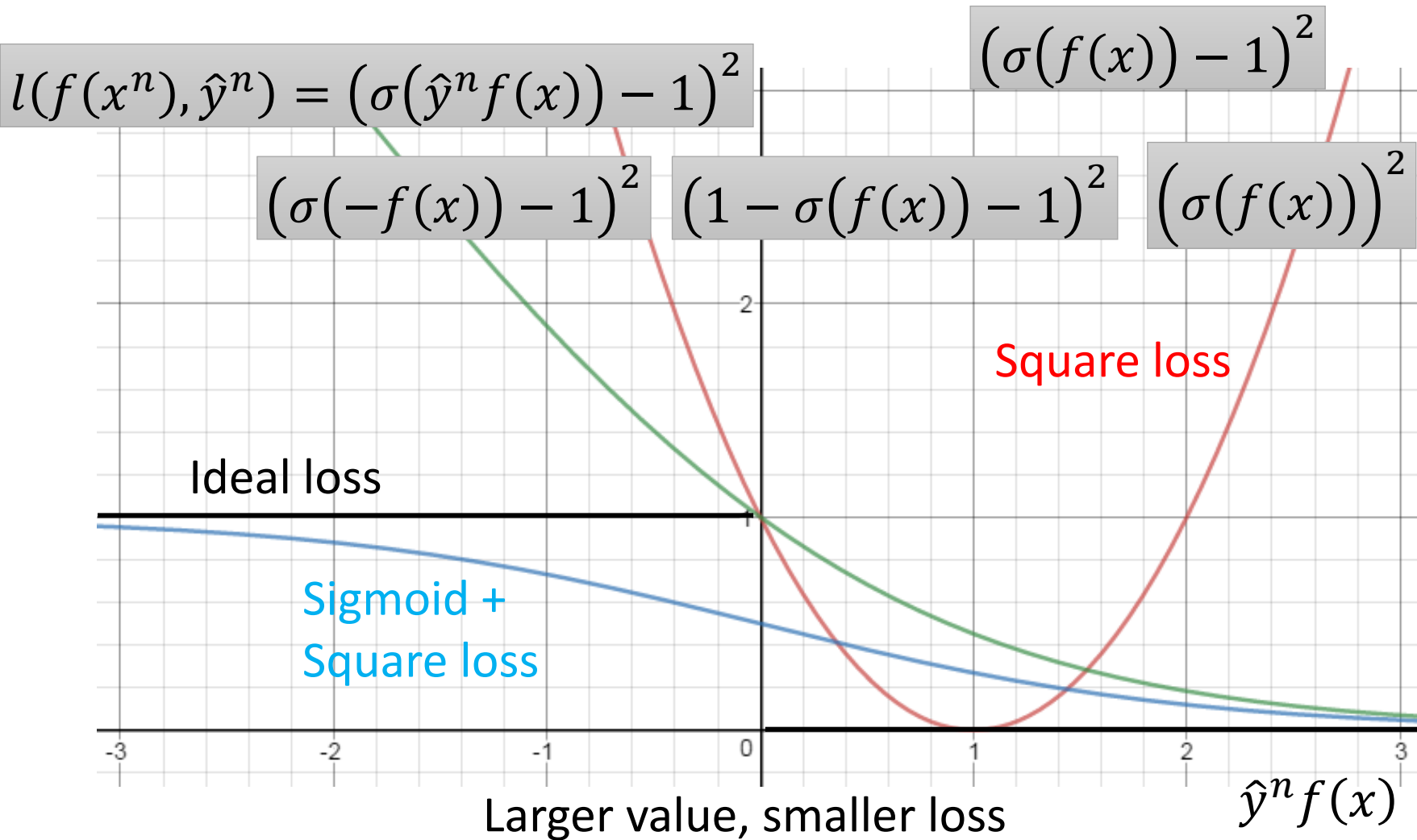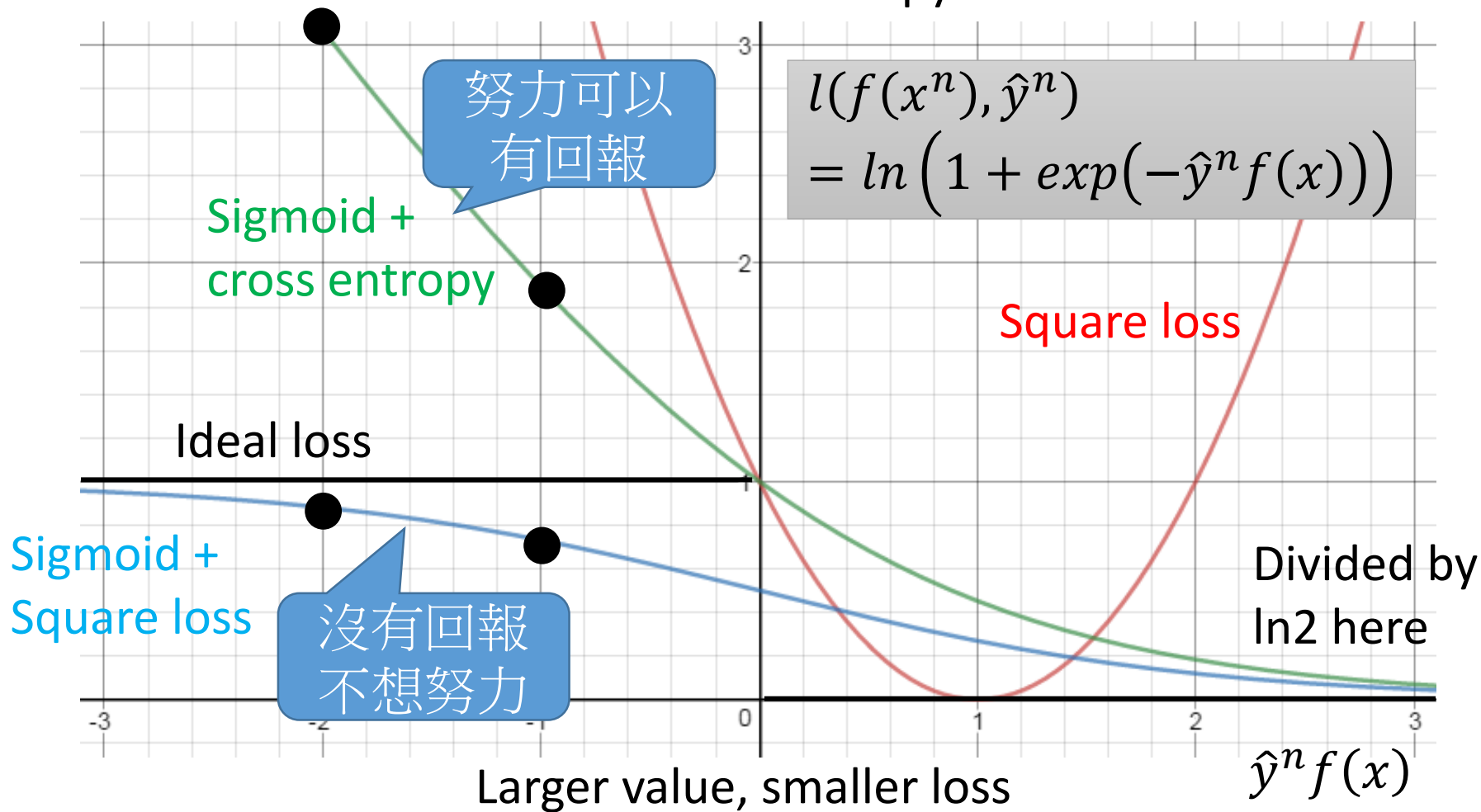Larger value, smaller loss

$\hat{y}^n f(x)$

Step 2: Loss function

Sigmoid + Square Loss:

If $\hat{y}^n = 1$, $\quad \sigma(f(x))$ close to 1

If $\hat{y}^n = -1$, $\quad \sigma(f(x))$ close to 0

$$l(f(x^n), \hat{y}^n) = (\sigma(\hat{y}^n f(x)) - 1)^2$$

$$(\sigma(f(x)) - 1)^2$$

$$(\sigma(-f(x)) - 1)^2 \qquad (1 - \sigma(f(x)) - 1)^2 \qquad (\sigma(f(x)))^2$$

Ideal loss

Square loss

Sigmoid +
Square loss

Larger value, smaller loss

$\hat{y}^n f(x)$

Step 2: Loss function Sigmoid + cross entropy (logistic regression)
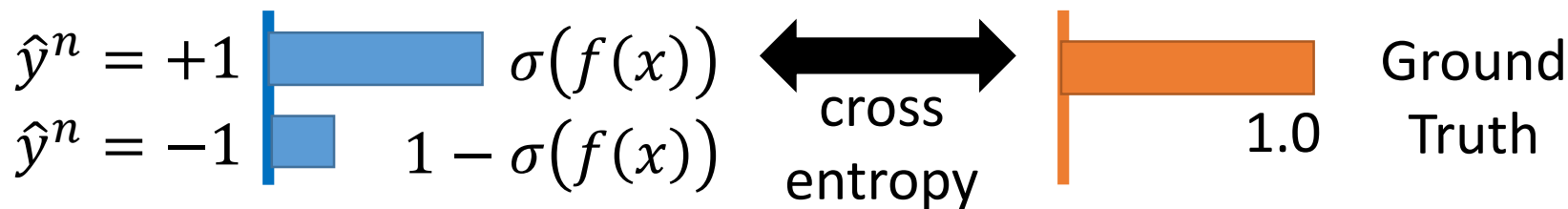
$\hat{y}^n = +1$ $\sigma(f(x))$ cross entropy   Ground

$\hat{y}^n = -1$ $1 - \sigma(f(x))$ 1.0   Truth

努力可以
有回報

Sigmoid +
cross entropy

$$l(f(x^n), \hat{y}^n) = ln\left(1 + exp(-\hat{y}^n f(x))\right)$$

Square loss

Ideal loss

Sigmoid +
Square loss

沒有回報
不想努力

Divided by
ln2 here

Larger value, smaller loss

$\hat{y}^n f(x)$

Step 2: Loss function

$$l(f(x^n), \hat{y}^n) = max(0, 1 - \hat{y}^n f(x))$$

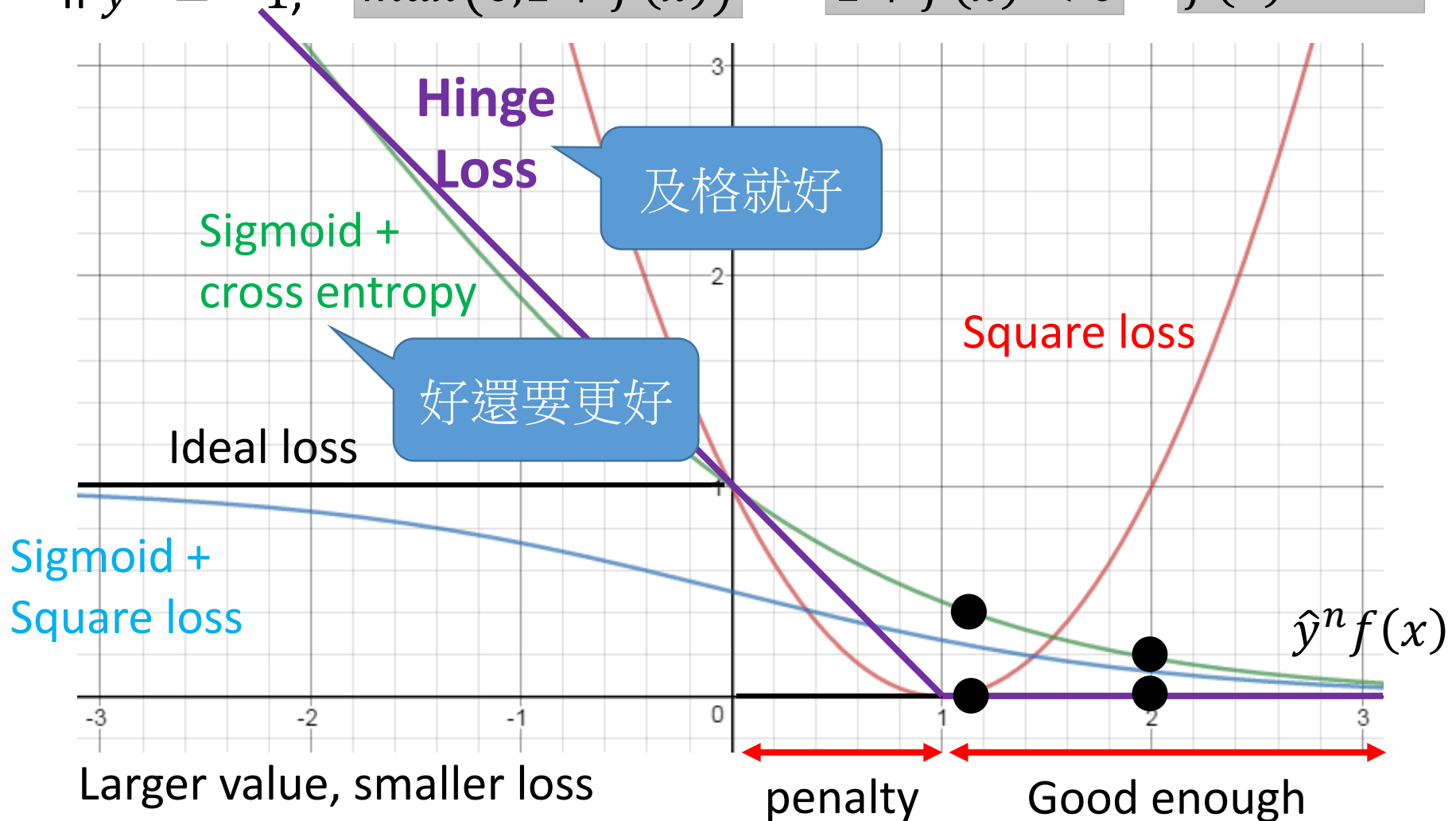If $\hat{y}^n = 1$, $max(0, 1 - f(x))$ $1 - f(x) < 0$ $f(x) > 1$

If $\hat{y}^n = -1$, $max(0, 1 + f(x))$ $1 + f(x) < 0$ $f(x) < -1$

Hinge Loss

及格就好

Sigmoid + cross entropy

好還要更好

Ideal loss

Square loss

Sigmoid + Square loss

$\hat{y}^n f(x)$

Larger value, smaller loss

penalty   Good enough

Linear SVM

Compared with logistic regression, linear SVM has different loss function

Deep version:

Yichuan Tang , "Deep Learning using Linear Support Vector Machines", ICML 2013 Challenges in Representation Learning Workshop

- Step 1: Function (Model)

New w

$$f(x) = \sum_i w_i x_i + b = \begin{bmatrix} w \\ b \end{bmatrix} \cdot \begin{bmatrix} x \\ 1 \end{bmatrix} = w^T x$$

New x

- Step 2: Loss function

$$L(f) = \sum_n l(f(x^n), \hat{y}^n) + \lambda \|w\|_2$$

convex

$$l(f(x^n), \hat{y}^n) = max(0, 1 - \hat{y}^n f(x))$$
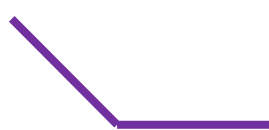
- Step 3: gradient descent?

Recall relu, maxout network

Linear SVM – gradient descent

Ignore regularization for simplicity

$$L(f) = \sum_n l(f(x^n), \hat{y}^n) \qquad l(f(x^n), \hat{y}^n) = max(0, 1 - \hat{y}^n f(x^n))$$

$$\frac{\partial l(f(x^n), \hat{y}^n)}{\partial w_i} = \frac{\partial l(f(x^n), \hat{y}^n)}{\partial f(x^n)} \boxed{\frac{\partial f(x^n)}{\partial w_i}} x_i^n \qquad \boxed{\begin{array}{c} f(x^n) \\ = w^T \cdot x^n \end{array}}$$

$$\frac{\partial max(0, 1 - \hat{y}^n f(x^n))}{\partial f(x^n)} = \begin{cases} -\hat{y}^n & \text{If } \hat{y}^n f(x^n) < 1 \\ \\ 0 & \text{otherwise} \end{cases}$$

$$\frac{\partial L(f)}{\partial w_i} = \sum_n \underbrace{-\delta(\hat{y}^n f(x^n) < 1)\hat{y}^n x_i}_{c^n(w)} \qquad w_i \leftarrow w_i - \eta \sum_n c^n(w) x_i^n$$

Linear SVM – another formulation

Minimizing loss function L:

$$L(f) = \sum_n \boxed{\varepsilon^n} + \lambda \|w\|_2$$

$$\varepsilon^n = max\left(0, 1 - \hat{y}^n f(x)\right)$$
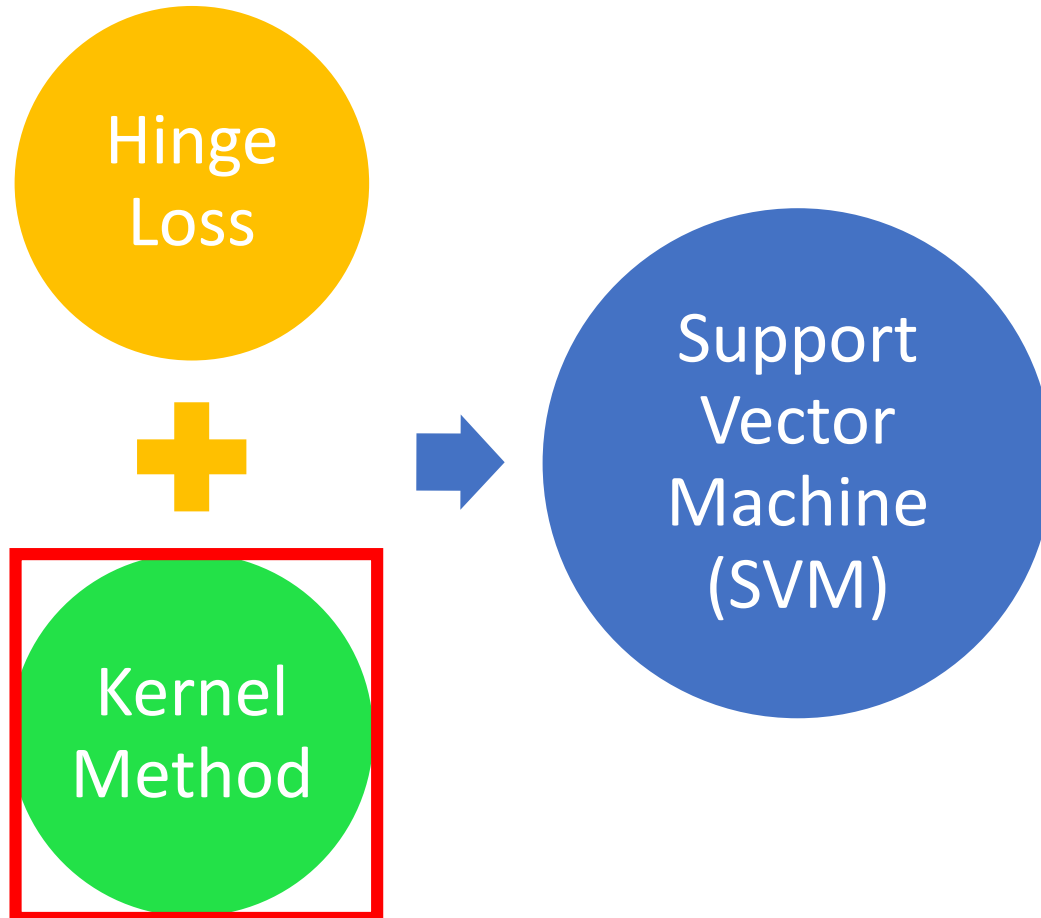
$=$

$$\varepsilon^n \geq 0$$

$$\varepsilon^n \geq 1 - \hat{y}^n f(x) \implies \hat{y}^n f(x) \geq 1 - \varepsilon^n$$

ε^n: slack variable

Quadradic programming problem

Outline

Dual Representation

$$w^* = \sum_n \alpha_n^* x^n$$

Linear combination of data points

α_n^* may be sparse ➡ x^n with non-zero α_n^* are support vectors

$$w_1 \leftarrow w_1 - \eta \sum_n c^n(w) x_1^n$$

$$\vdots$$

$$w_i \leftarrow w_i - \eta \sum_n c^n(w) x_i^n$$

$$\vdots$$

$$w_k \leftarrow w_k - \eta \sum_n c^n(w) x_k^n$$

If w initialized as **0**

$$w \leftarrow w - \eta \sum_n c^n(w) x^n$$

$$c^n(w)$$

$$= \frac{\partial l(f(x^n), \hat{y}^n)}{\partial f(x^n)}$$

Hinge loss: usually zero

c.f. for logistic regression, it is always non-zero

Dual Representation

$$w = \sum_n \alpha_n x^n = X\boldsymbol{\alpha} \qquad X = \boxed{\begin{array}{cccc} x^1 & x^2 & \cdots\cdots & x^N \end{array}} \qquad \boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_N \end{bmatrix}$$

$$w = X\boldsymbol{\alpha}$$

Step 1: $\quad f(x) = w^T x \qquad \Longrightarrow \qquad f(x) = \boldsymbol{\alpha}^T X^T x$

$$[\alpha_1 \quad \cdots \quad \alpha_N]$$

$$\boxed{\begin{array}{c} x^1 \\ x^2 \\ \vdots \\ x^N \end{array}} \qquad \boxed{x}$$

$$f(x) = \sum_n \alpha_n (x^n \cdot x)$$

$$= \sum_n \alpha_n K(x^n, x)$$

$$\begin{bmatrix} x^1 \cdot x \\ x^2 \cdot x \\ \vdots \\ x^N \cdot x \end{bmatrix}$$

Dual Representation

Step 1: $f(x) = \sum_n \alpha_n K(x^n, x)$

Step 2, 3: Find $\{\alpha_1^*, \cdots, \alpha_n^*, \cdots, \alpha_N^*\}$, minimizing loss function L

$$L(f) = \sum_n l(\underline{f(x^n)}, \hat{y}^n)$$

$$= \sum_n l\left(\underline{\sum_{n'} \alpha_{n'} K(x^{n'}, x^n)}, \hat{y}^n\right)$$

We don't really need to know vector x

We only need to know the inner project between a pair of vectors x and z

$$K(x, z)$$

Kernel Trick

Kernel Trick

Directly computing $K(x,z)$ can be faster than "feature transformation + inner product" sometimes.

Kernel trick is useful when we transform all x to $\phi(x)$

$$K(x,z) = \phi(x) \cdot \phi(z) = \begin{bmatrix} x_1{}^2 \\ \sqrt{2}x_1x_2 \\ x_2{}^2 \end{bmatrix} \cdot \begin{bmatrix} z_1{}^2 \\ \sqrt{2}z_1z_2 \\ z_2{}^2 \end{bmatrix}$$

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$= x_1{}^2 z_1{}^2 + 2x_1x_2z_1z_2 + x_2{}^2 z_2{}^2$$

$$\phi(x) = \begin{bmatrix} x_1{}^2 \\ \sqrt{2}x_1x_2 \\ x_2{}^2 \end{bmatrix}$$

$$= (x_1z_1 + x_2z_2)^2 = \left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \cdot \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \right)^2$$

$$= (x \cdot z)^2$$

Kernel Trick

Directly computing $K(x, z)$ can be faster than "feature transformation + inner product" sometimes.

$$K(x, z) = (x \cdot z)^2$$

$$= (x_1 z_1 + x_2 z_2 + \cdots + x_k z_k)^2$$

$$= x_1{}^2 z_1{}^2 + x_2{}^2 z_2{}^2 + \cdots + x_k{}^2 z_k{}^2$$

$$+ 2x_1 x_2 z_1 z_2 + 2x_1 x_3 z_1 z_3 + \cdots$$

$$+ 2x_2 x_3 z_2 z_3 + 2x_2 x_4 z_2 z_4 + \cdots$$

$$= \phi(x) \cdot \phi(z)$$

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_k \end{bmatrix} \quad z = \begin{bmatrix} z_1 \\ \vdots \\ z_k \end{bmatrix}$$

$$\phi(x) = \begin{bmatrix} x_1{}^2 \\ \vdots \\ x_k{}^2 \\ \sqrt{2} x_1 x_2 \\ \sqrt{2} x_1 x_3 \\ \vdots \\ \sqrt{2} x_2 x_3 \\ \vdots \end{bmatrix}$$

Radial Basis Function Kernel

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \end{bmatrix} \quad z = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \end{bmatrix}$$

$$K(x, z) = exp\left(-\frac{1}{2}\|x - z\|_2\right) = \phi(x) \cdot \phi(z)?$$

$\phi(*)$ has inf dim!!!

$$= exp\left(-\frac{1}{2}\|x\|_2 - \frac{1}{2}\|z\|_2 + x \cdot z\right)$$

$$= exp\left(-\frac{1}{2}\|x\|_2\right)exp\left(-\frac{1}{2}\|z\|_2\right)exp(x \cdot z) \quad = C_x C_z exp(x \cdot z)$$

$$= C_x C_z \sum_{i=0}^{\infty} \frac{(x \cdot z)^i}{i!} = C_x C_z + C_x C_z(x \cdot z) + C_x C_z \frac{1}{2}(x \cdot z)^2 \dots$$

$$[C_x] \cdot [C_z] \quad \begin{bmatrix} C_x x_1 \\ C_x x_2 \\ \vdots \end{bmatrix} \cdot \begin{bmatrix} C_z z_1 \\ C_z z_2 \\ \vdots \end{bmatrix} \quad \frac{1}{\sqrt{2}}\begin{bmatrix} C_x {x_1}^2 \\ \vdots \\ \sqrt{2}C_x x_1 x_2 \\ \vdots \end{bmatrix} \cdot \frac{1}{\sqrt{2}}\begin{bmatrix} C_z {z_1}^2 \\ \vdots \\ \sqrt{2}C_z z_1 z_2 \\ \vdots \end{bmatrix}$$
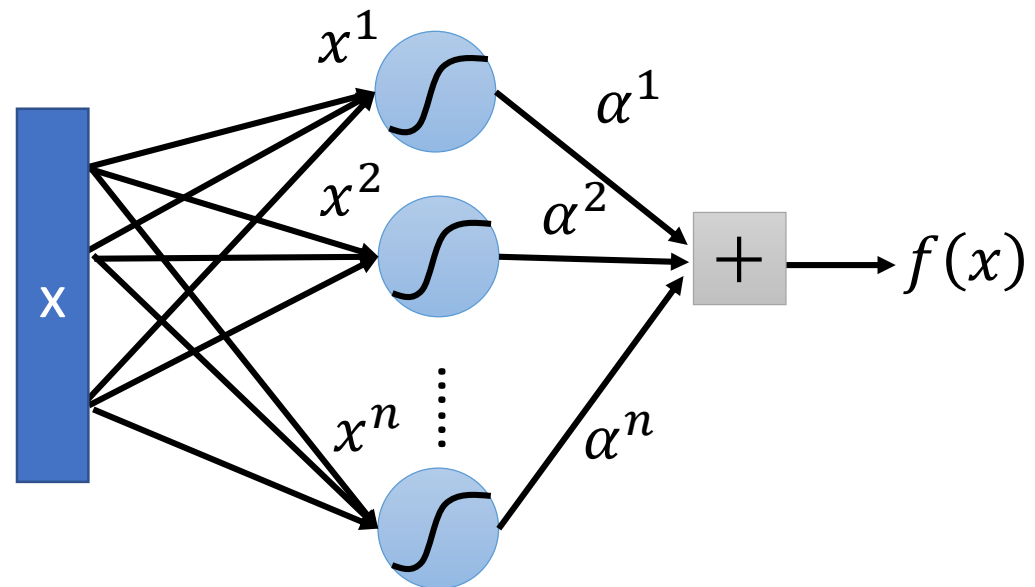
Sigmoid Kernel

$$K(x, z) = tanh(x \cdot z)$$

- When using sigmoid kernel, we have a 1 hidden layer network.

$$f(x) = \sum_n \alpha_n K(x^n, x) = \sum_n \alpha^n tanh(x^n \cdot x)$$

The weight of each neuron is a data point

The number of support vectors is the number of neurons.

You can directly design $K(x, z)$ instead of considering $\phi(x), \phi(z)$

When x is structured object like sequence, hard to design $\phi(x)$

$K(x, z)$ is something like similarity (Mercer's theory to check)

Evaluate the similarity between sequences as $K(x, z)$

Audio Segment
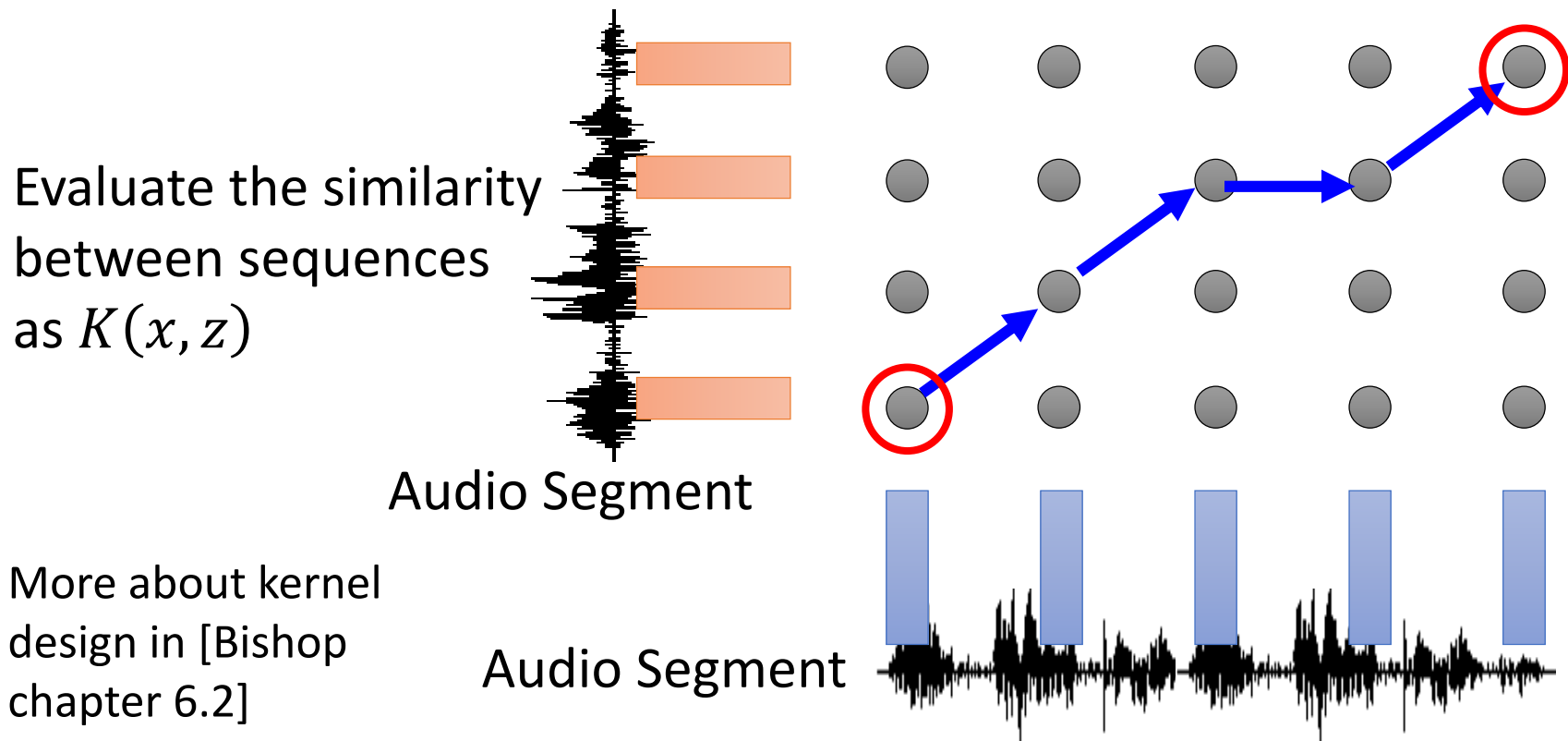
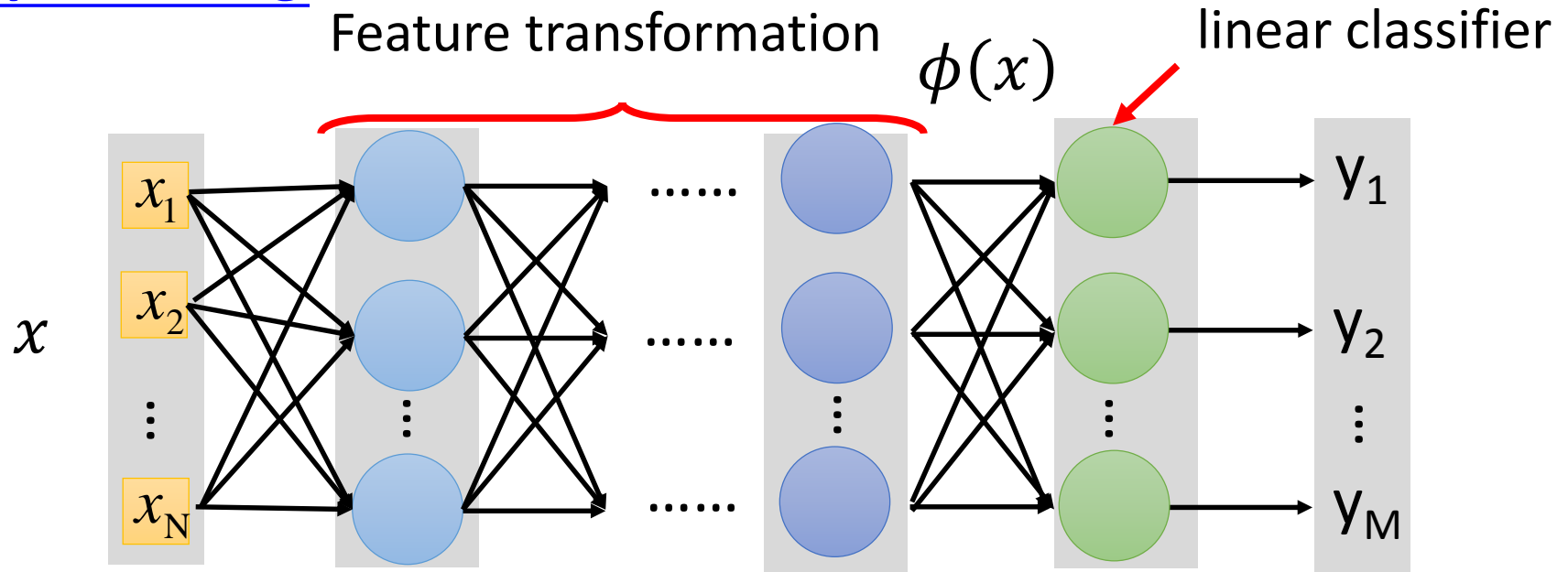More about kernel design in [Bishop chapter 6.2]

Audio Segment

Hiroshi Shimodaira, Ken-ichi Noma, Mitsuru Nakai, Shigeki Sagayama, "Dynamic Time-Alignment Kernel in Support Vector Machine", NIPS, 2002

Marco Cuturi, Jean-Philippe Vert, Oystein Birkenes, Tomoko Matsui, A kernel for time series based on global alignments, ICASSP, 2007

SVM related methods

- Support Vector Regression (SVR)
  - [Bishop chapter 7.1.4]

- Ranking SVM
  - [Alpaydin, Chapter 13.11]

- One-class SVM
  - [Alpaydin, Chapter 13.11]

Deep Learning

Feature transformation $\phi(x)$ linear classifier

x — x_1, x_2, $\vdots$, x_N

....

y_1, y_2, $\vdots$, y_M

SVM

Based on kernel function

$\phi(x)$

Multiple Kernel learning [Alpaydin, Chapter 13.8]

Input Space **Feature Space**

linear classifier