CS 461: Machine Learning Principles

Class 11: Oct. 10

Perceptron and Logistic Regression

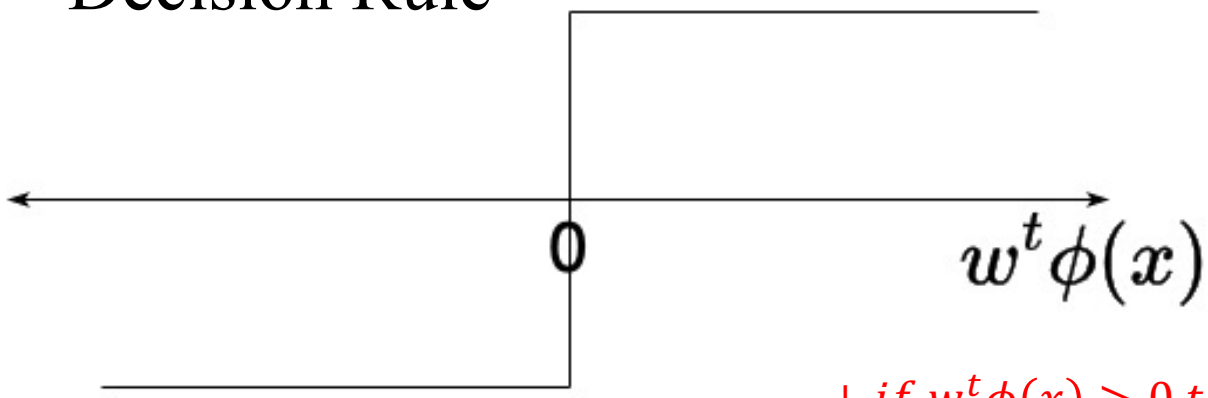
Kernel Methods

Instructor: Diana Kim

In the last class,
we studied perceptron algorithm.

- It learns the decision boundary directly.
  $\overrightarrow{W}\phi(\mathrm{x}) = 0$

- Decision Rule



$+ if\ w^t\phi(x) > 0\ then\ assign + 1, otherwise\ assign\ -1$

$if\ x$ and y are right classification samples, then $\overrightarrow{W}\phi(\mathrm{x})\cdot \mathrm{y} > 0$

In the last class,
we studied perceptron algorithm.

- The Objective Function

$$E(\vec{w}) = -\sum_{n \in \mathcal{M}} \vec{w}^t \phi(x_n) \cdot t_n - \sum_{n \in \mathcal{M}^c} \cdot 0$$

- Update Rule

$$w(t+1) = w(t) - \eta \nabla E(w) = w(t) + \eta \cdot \phi(\vec{x_n}) t_n$$

$(x_n , t_n)\ misclassification\ samples$

Perceptron Convergence theorem

If training data set is linearly separable,
then the perceptron algorithm is guaranteed
to find a solution in finite number of iterations.
(initial W and step size does not matter)

Perceptron Convergence theorem

If training data set is linearly separable,

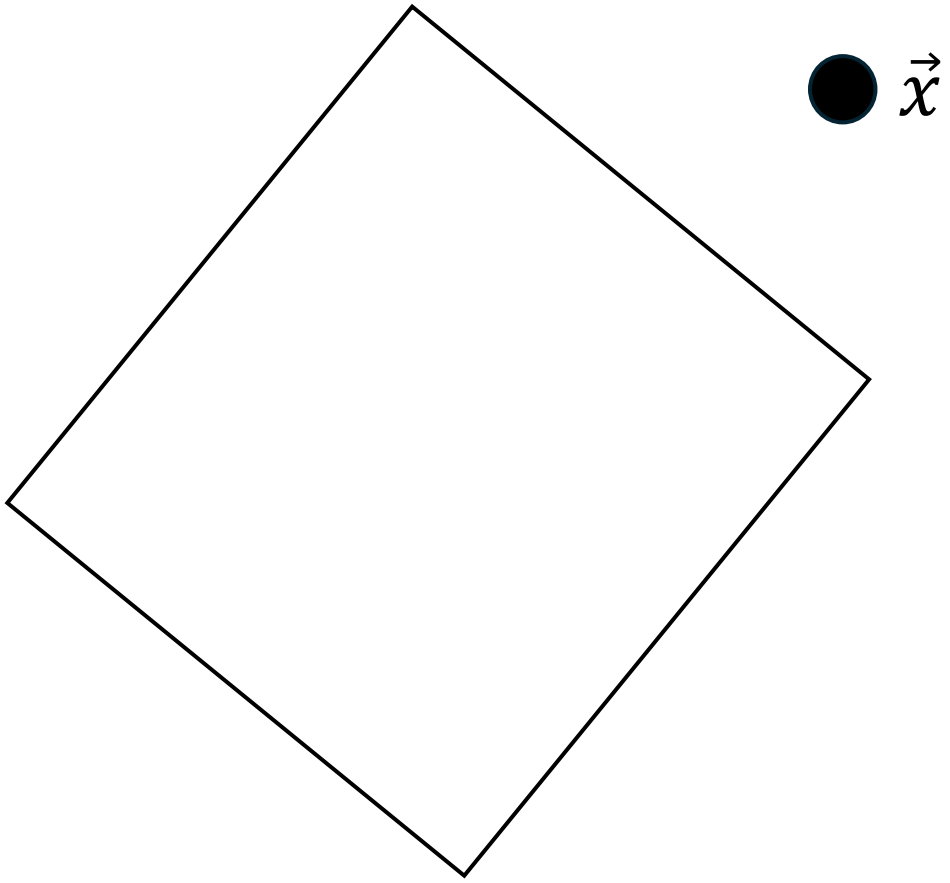
$$\exists w_* \quad \text{such that } w_*^t x_i \cdot y_i > 0 \quad \forall i$$

then the perceptron algorithm is guaranteed
to find a solution in finite number of iterations.

"# iterations $\leq C$" we need to show this.

Preliminary (1)

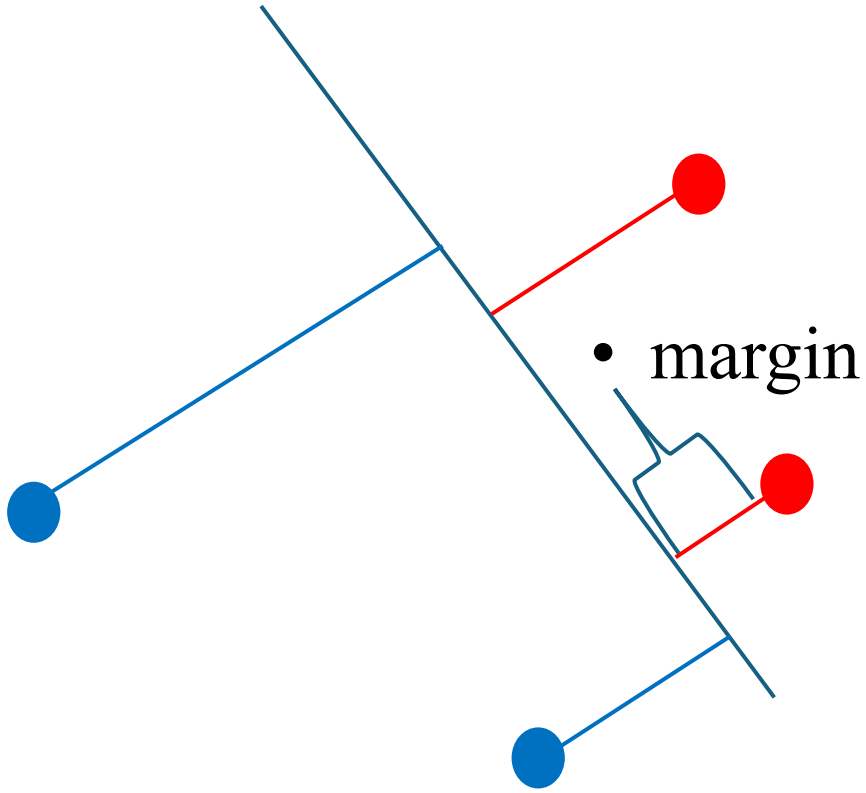
Distance from a Point ($\vec{x}$) to a Hyperplane $\vec{w}^t \cdot \vec{x} = 0$



+ Distance = $\dfrac{|w^t x|}{||w||^2}$

Preliminary (2) the Concept of Margin

Margin: the smallest distance
between the decision boundary and any of the samples



Suppose,

- $\exists w_* \quad \text{such that } w_*^t x_i \cdot y_i > 0 \quad \forall i$
- $||w*|| = 1$
- $||x*|| \leq 1$
- $w_0 = \vec{0}$
- margin: γ

One update

$$w_1 = w_0 + \eta \cdot x \cdot y$$

- (x, y) is a misclassification sample.

$$w_1^t w_* = (w_0 + \eta \cdot x \cdot y)^t w_*$$
$$= w_0 w_* + \eta y \cdot x^t w_* \geq w_0 w_* + \eta\gamma$$

$$w_1^t w_1 = (w_0 + \eta \cdot x \cdot y)^t \cdot (w_0 + \eta \cdot x \cdot y)$$
$$= w_1^t w_1 + 2 \cdot \eta \cdot y \cdot w_0^t \cdot x + \eta^2 y^2 x^t x$$
$$\leq w_1^t w_1 + \eta^2$$

- M times updates

$$w_1^t w_* \geq w_0 w_* + M\eta\gamma$$
$$w_1^t w_1 \leq w_0 w_* + M\eta^2$$

$$\begin{aligned} M\eta\gamma &\leq w_1^t w_* \\ &\leq ||w_1|| \cos\theta \\ &\leq ||w_1|| \leq \sqrt{M}\eta \\ M\gamma &\leq \sqrt{M} \\ M &\leq \frac{1}{\gamma^2} \end{aligned}$$

- # number of updates cannot beyond $\frac{1}{\gamma^2}$

Quiz1) Based on Perceptron Convergence Theorem, what can you say about the margin of a classifier? (is small margin or **large margin desirable**)?

Comparison between Binary Logistic Regression vs. Perceptron

- True/ False about Logistic Regression & Perceptron

1. Both are the algorithm for binary classification. (True)

2. When data is not linearly separable, then both does not converge. (False)

3. When data is linearly separable, logistic regression curve would resemble the Perceptron's sign function. (True)

4. Only Perceptron defines decision boundary. (False)

5. Perceptron learns a large margin classifier? (False)

Loss Comparison for One Data Point (Objective Function)

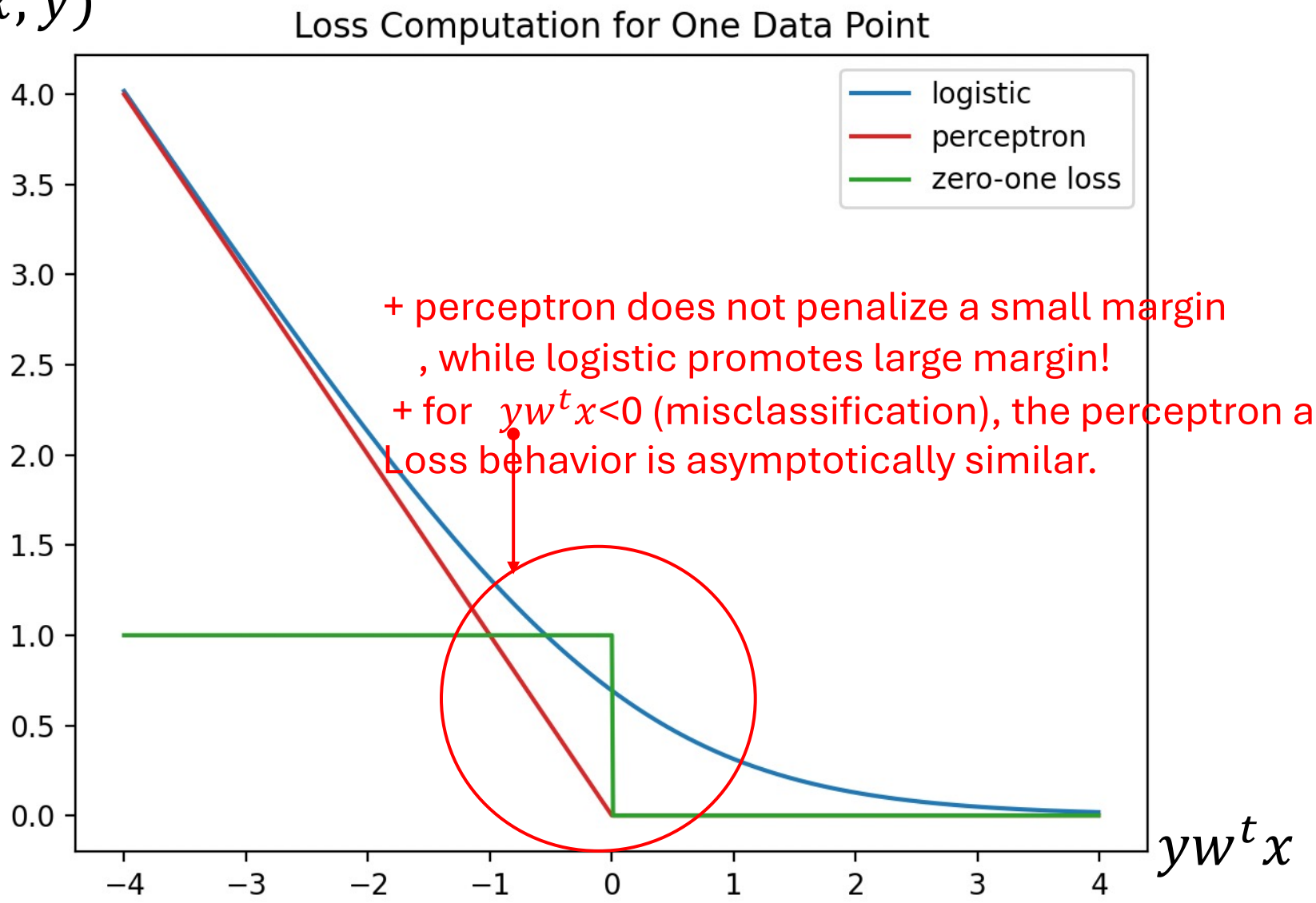
$$\text{Perceptron Loss } (x,y) = \begin{cases} -w^t xy & \text{if } (x,y) \text{ is misclassification sample} \\ 0 & \text{if } (x,y) \text{ is right classification sample} \end{cases}$$

$$\text{Logistic Loss } (x,y) = \begin{cases} -\ln \dfrac{1}{1+\exp(-w^t x)} & \text{if } y = +1 \\ -\ln \dfrac{\exp(-w^t x)}{1+\exp(-w^t x)} & \text{if } y = -1 \end{cases}$$

$$\text{Logistic Loss } (x,y) = \begin{cases} -\ln \dfrac{1}{1+\exp(-w^t x)} = -\ln \dfrac{\exp(w^t x)}{1+\exp(w^t x)} \\ -\ln \dfrac{\exp(-w^t x)}{1+\exp(-w^t x)} \end{cases}$$

Loss Comparison for One Data Point (Objective Function)

$L(x, y)$

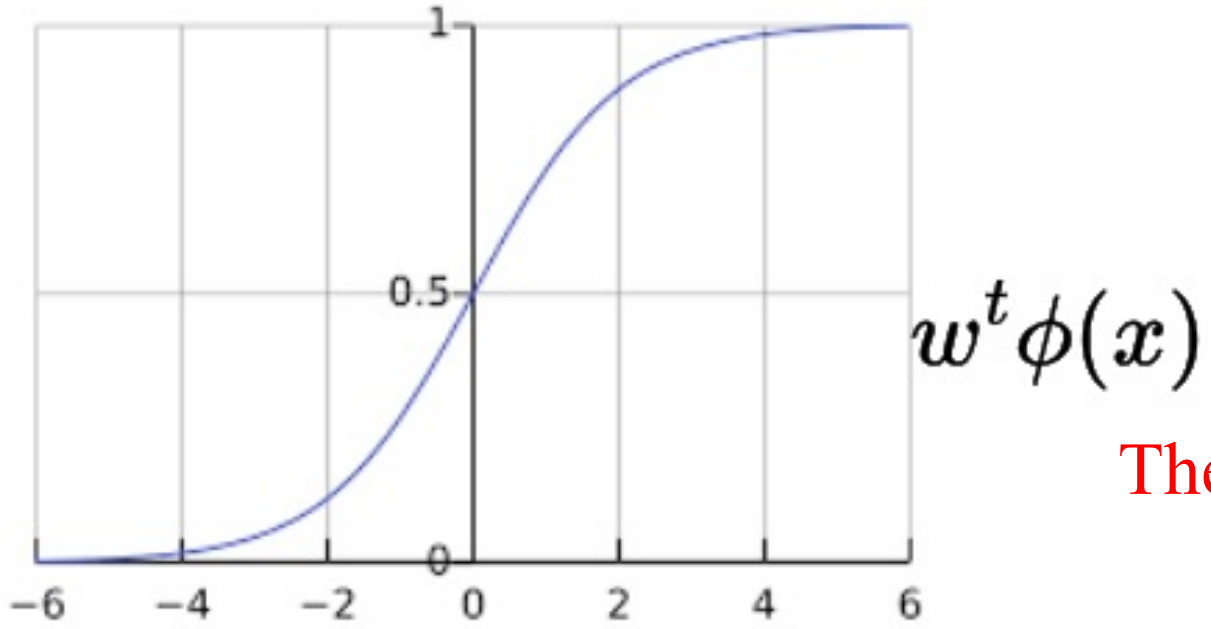


+ perceptron does not penalize a small margin , while logistic promotes large margin!

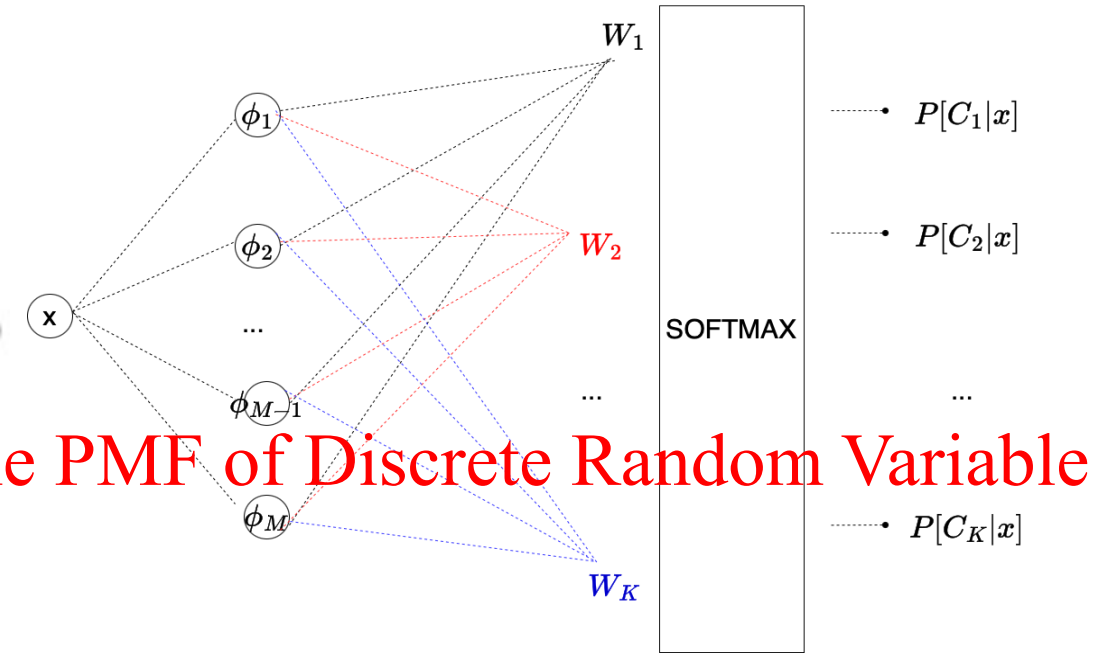
+ for $yw^t x$<0 (misclassification), the perceptron and logistic Loss behavior is asymptotically similar.

The Last Question Before Moving to the topic of Kernel Methods
: output shape of binary vs. multiclass logistic regression.

P[C0|x] = $\sigma(w^t\phi(x))$



Binary Logistic Classification



The PMF of Discrete Random Variable

Multiple Logistic Classification

Kernel Methods (Constructing Kernels)

Kernel function: it is a measure of the distance between two feature points.

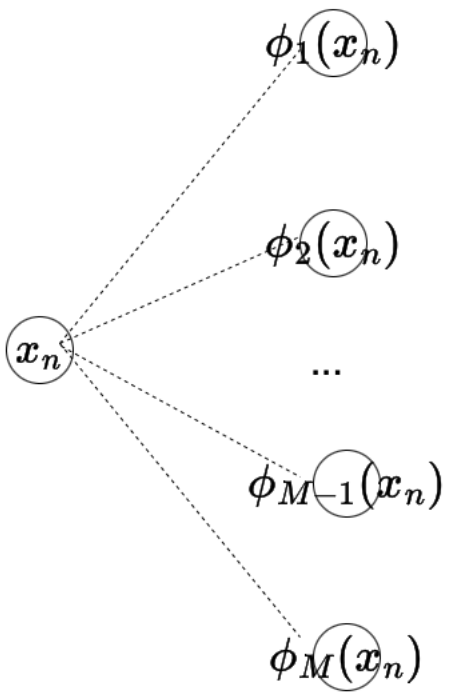
Once we have kernel functions, we don't need to know the feature mapping explicitly for computing the distance.

The function will be used for non-parametric modeling: linear combination of kernel functions evaluated at training data points.

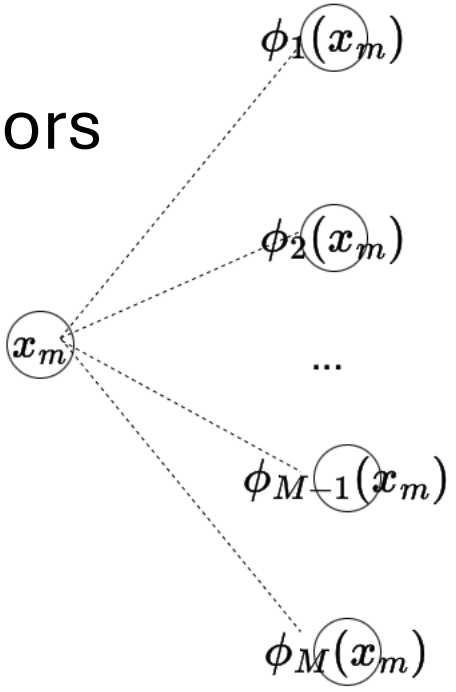
Kernel Function $\kappa(x_1, x_2)$: $\chi \times \chi \rightarrow \mathbb{R}$

- [The Definition of Kernel Function]

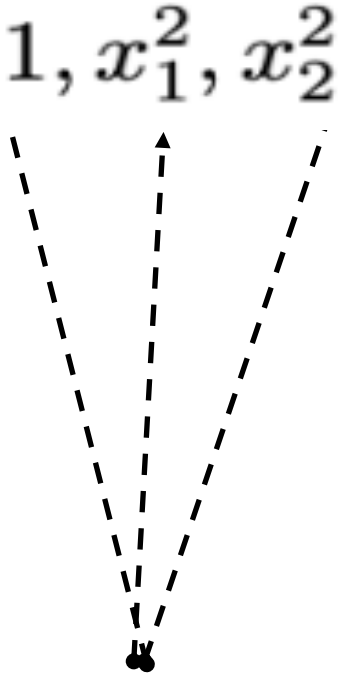
$$\kappa(x_n, x_m) = \phi(x_n)^t \phi(x_m)$$



Inner product
between the two feature vectors



Example1) Different feature design produces kernel functions.



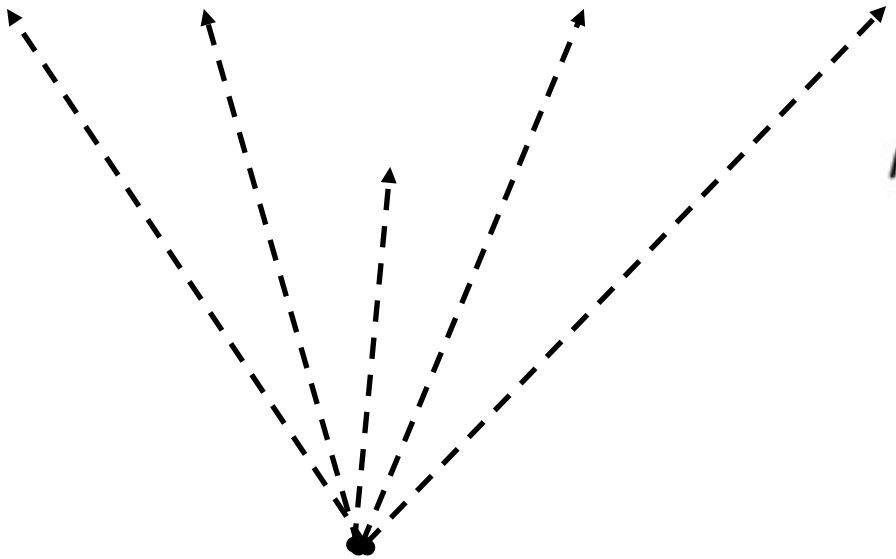
- Feature MAP

Q: $\kappa(x, x')$?

$$\kappa(x, x') = (1, x_1^2, x_2^2)^t \cdot (1, x_1'^2, x_2'^2)$$
$$= 1 + x_1^2 x_1'^2 + x_2^2 x_2'^2$$

Example2): The inner product of the two points on the space of features map.

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



$(x1, x2)$

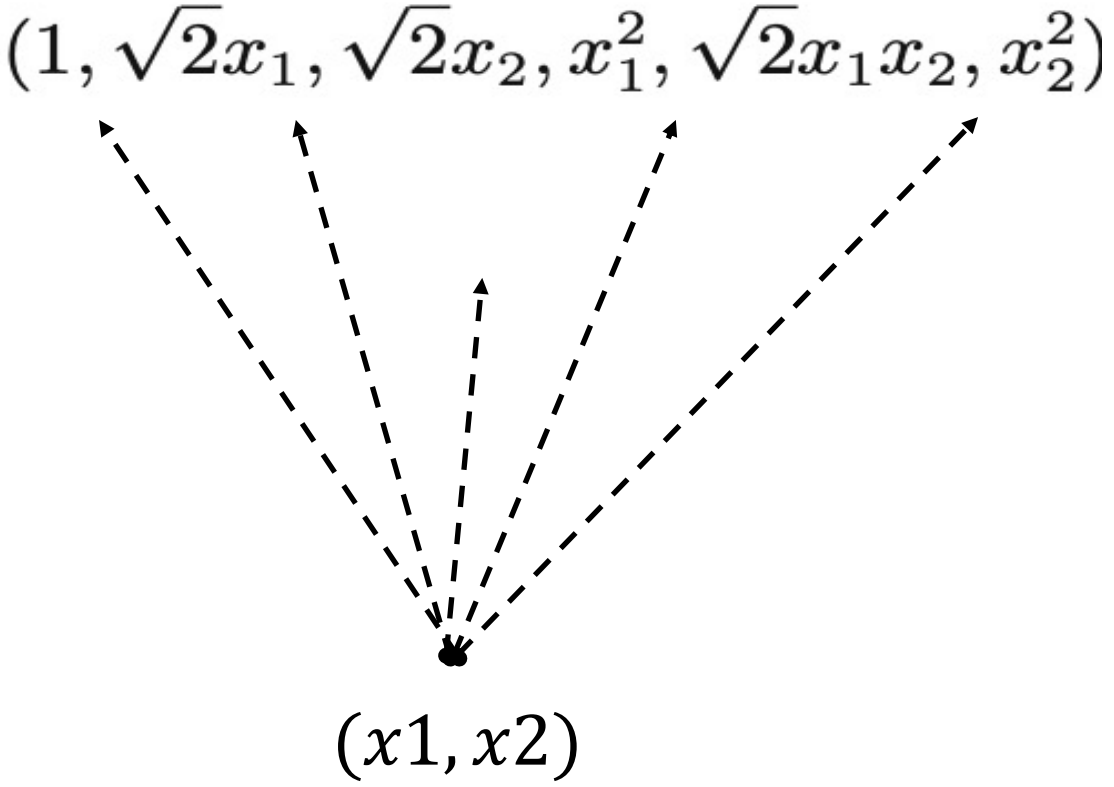
Q: $\kappa([1,1], [\sqrt{2}, \sqrt{2}])$?

$$\kappa(x, x') = (1, \sqrt{2}, \sqrt{2}, 1, \sqrt{2}, 1)^t (1, 2, 2, 2, 2\sqrt{2}, 2)$$

- Feature MAP

Once we have a kernel function,
we don't need to know the explicit feature map to compute the inner product.
$\kappa(x, x')$: $\chi \times \chi \rightarrow \mathbb{R}$

Example2): with kernel functions, no need to compute feature map.



- Feature MAP

Q: $\kappa([1,1], [\sqrt{2}, \sqrt{2}])$?

$$\kappa(x, x') = (x_1x_1' + x_2x_2' + 1)^2$$

With kernel function, we can generate a gram matrix for data samples.

Gram Matrix K = $\Phi\Phi^t$ for data points $x_1, x_2, \ldots, x_n$

$$K(i,j) = \phi(x_i)^t \phi(x_j)$$

$$K = \begin{bmatrix} \kappa(x_1,x_1) & \kappa(x_1,x_2) & \ldots & \kappa(x_1,x_n) \\ \kappa(x_2,x_1) & \kappa(x_2,x_2) & \ldots & \kappa(x_2,x_n) \\ \ldots & \ldots & \ldots & \ldots \\ \kappa(x_{n-1},x_1) & \kappa(x_{n-1},x_2) & \ldots & \kappa(x_{n-1},x_n) \\ \kappa(x_n,x_1) & \kappa(x_n,x_2) & \ldots & \kappa(x_n,x_n) \end{bmatrix}$$

$$= \begin{bmatrix} \phi(x_1)^t\phi(x_1) & \phi(x_1)^t\phi(x_1) & \ldots & \phi(x_1)^t\phi(x_n) \\ \phi(x_2)^t\phi(x_1) & \phi(x_2)^t\phi(x_2) & \ldots & \phi(x_2)^t\phi(x_n) \\ \ldots & \ldots & \ldots & \ldots \\ \phi(x_{n-1})^t\phi(x_1) & \phi(x_{n-1})^t\phi(x_2) & \ldots & \phi(x_{n-1})^t\phi(x_n) \\ \phi(x_n)^t\phi(x_1) & \phi(x_n)^t\phi(x_2) & \ldots & \phi(x_n)^t\phi(x_n) \end{bmatrix} = \Phi \cdot \Phi^t$$

$\kappa(x_1, x_2)\ is\ a\ kernel\ function\ \leftrightarrow (iff)$

$Gram\ matrix\ K (K = \Phi\Phi^t)$

$whose\ elements\ are\ given\ by\ \kappa(x_1,x_2)$ is semi − positive defnite.

- Positive semi-definite

$$x^t\Phi\Phi^t x = (\Phi^t x)^t(\Phi^t x) = ||\Phi^t x||^2 \geq 0 \quad \forall x$$

Constructing new kernels
out of simpler kernels as building blocks

- 1) $\kappa(x, x') = x^t A x'$ where A is positive semi-definite

$$K = X_N^t A X_N$$

$$y^t K y = y^t X_N^t A X_N y$$

$$= (X_N \cdot y)^t A (X_N \cdot y) \geq 0$$ by A is positive semi-definite

Gram matrix K is positive semi-definite

Constructing new kernels
out of simpler kernels as building blocks

- 2) $\kappa(x, x') = c \cdot \kappa_1(x, x')$ where c is positive const and $\kappa_1(x, x')$ *is kernel*.

$$K = \begin{bmatrix} c \cdot \kappa_1(x_1, x_1) & c \cdot \kappa_1(x_1, x_2) & \dots & c \cdot \kappa_1(x_1, x_n) \\ c \cdot \kappa_1(x_2, x_1) & c \cdot \kappa_1(x_2, x_2) & \dots & c \cdot \kappa_1(x_2, x_n) \\ \dots & \dots & \dots & \dots \\ c \cdot \kappa_1(x_{n-1}, x_1) & c \cdot \kappa_1(x_{n-1}, x_2) & \dots & c \cdot \kappa_1(x_{n-1}, x_n) \\ c \cdot \kappa_1(x_n, x_1) & c \cdot \kappa_1(x_n, x_2) & \dots & c \cdot \kappa_1(x_n, x_n) \end{bmatrix}$$

$x^t K x = c x^t K_1 x \geq 0$ by c is positive const and $\kappa_1(x, x')$ *is kernel*.

Constructing new kernels
out of simpler kernels as building blocks

- 3) $\kappa(x, x') = f(x)\kappa_1(x, x')f(x')$

$$K = \begin{bmatrix} f(x_1) \cdot \kappa_1(x_1, x_1) f(x_1) & f(x_1) \cdot \kappa_1(x_1, x_2) f(x_2) & \dots & f(x_1) \cdot \kappa_1(x_1, x_n) f(x_n) \\ f(x_2) \cdot \kappa_1(x_2, x_1) f(x_1) & f(x_2) \cdot \kappa_1(x_2, x_2) f(x_2) & \dots & f(x_2) \cdot \kappa_1(x_2, x_n) f(x_n) \\ \dots & \dots & \dots & \dots \\ f(x_n) \cdot \kappa_1(x_2, x_1) f(x_1) & f(x_2) \cdot \kappa_1(x_2, x_2) f(x_2) & \dots & f(x_n) \cdot \kappa_1(x_2, x_n) f(x_n) \end{bmatrix}$$

Constructing new kernels
out of simpler kernels as building blocks

- 3) $\kappa(x, x') = f(x)\kappa_1(x, x')f(x')$

$$K = \begin{bmatrix} f(x_1)\cdot\kappa_1(x_1,x_1)f(x_1) & f(x_1)\cdot\kappa_1(x_1,x_2)f(x_2) & \dots & f(x_1)\cdot\kappa_1(x_1,x_n)f(x_n) \\ f(x_2)\cdot\kappa_1(x_2,x_1)f(x_1) & f(x_2)\cdot\kappa_1(x_2,x_2)f(x_2) & \dots & f(x_2)\cdot\kappa_1(x_2,x_n)f(x_n) \\ \dots & \dots & \dots & \dots \\ f(x_n)\cdot\kappa_1(x_2,x_1)f(x_1) & f(x_2)\cdot\kappa_1(x_2,x_2)f(x_2) & \dots & f(x_n)\cdot\kappa_1(x_2,x_n)f(x_n) \end{bmatrix}$$

$$K = \begin{bmatrix} f(x_1) & 0 & \dots & 0 \\ 0 & f(x_1) & \dots & 0 \\ 0 & \dots & \dots & 0 \\ 0 & 0 & \dots & f(x_1) \end{bmatrix} \cdot K_1 \cdot \begin{bmatrix} f(x_1) & 0 & \dots & 0 \\ 0 & f(x_1) & \dots & 0 \\ 0 & \dots & \dots & 0 \\ 0 & 0 & \dots & f(x_1) \end{bmatrix}$$

$$y^t K y = y^t \begin{bmatrix} f(x_1) & 0 & \dots & 0 \\ 0 & f(x_1) & \dots & 0 \\ 0 & \dots & \dots & 0 \\ 0 & 0 & \dots & f(x_1) \end{bmatrix} K_1 \begin{bmatrix} f(x_1) & 0 & \dots & 0 \\ 0 & f(x_1) & \dots & 0 \\ 0 & \dots & \dots & 0 \\ 0 & 0 & \dots & f(x_1) \end{bmatrix} y$$

$$= y'^t K_1 y' \geq 0 \quad \text{by } \kappa_1(x, x') \textit{ is kernel.}$$

Technique for Constructing New Kernels from Old Ones.

$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = c\mathcal{K}_1(\boldsymbol{x}, \boldsymbol{x}'), \text{ for any constant } c > 0$$
$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = f(\boldsymbol{x})\mathcal{K}_1(\boldsymbol{x}, \boldsymbol{x}')f(\boldsymbol{x}'), \text{ for any function } f$$
$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = q(\mathcal{K}_1(\boldsymbol{x}, \boldsymbol{x}')) \text{ for any function polynomial } q \text{ with nonneg. coef.}$$
$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \exp(\mathcal{K}_1(\boldsymbol{x}, \boldsymbol{x}'))$$
$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^\mathsf{T}\mathbf{A}\boldsymbol{x}', \text{ for any psd matrix } \mathbf{A}$$

$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \mathcal{K}_1(\boldsymbol{x}, \boldsymbol{x}') + \mathcal{K}_2(\boldsymbol{x}, \boldsymbol{x}')$$
$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \mathcal{K}_1(\boldsymbol{x}, \boldsymbol{x}') \times \mathcal{K}_2(\boldsymbol{x}, \boldsymbol{x}')$$

All functional operations become element-wise operation in Gram Matrix generation.

Gaussian Kernel

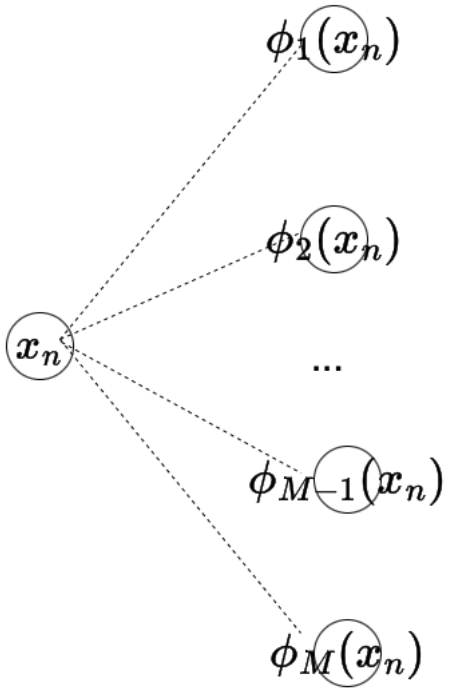
$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = c\mathcal{K}_1(\boldsymbol{x}, \boldsymbol{x}'), \text{ for any constant } c > 0$$
$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = f(\boldsymbol{x})\mathcal{K}_1(\boldsymbol{x}, \boldsymbol{x}')f(\boldsymbol{x}'), \text{ for any function } f$$
$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = q(\mathcal{K}_1(\boldsymbol{x}, \boldsymbol{x}')) \text{ for any function polynomial } q \text{ with nonneg. coef.}$$
$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \exp(\mathcal{K}_1(\boldsymbol{x}, \boldsymbol{x}'))$$
$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^\mathsf{T}\mathbf{A}\boldsymbol{x}', \text{ for any psd matrix } \mathbf{A}$$

$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \mathcal{K}_1(\boldsymbol{x}, \boldsymbol{x}') + \mathcal{K}_2(\boldsymbol{x}, \boldsymbol{x}')$$
$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \mathcal{K}_1(\boldsymbol{x}, \boldsymbol{x}') \times \mathcal{K}_2(\boldsymbol{x}, \boldsymbol{x}')$$

Q: how Gaussian is a valid kernel?

$$\kappa(x, x') = \exp \frac{-||x - x'||^2}{2\sigma^2}$$
$$= \exp \frac{-||x||^2 - ||x'||^2 + 2x^t x'}{2\sigma^2}$$

The Feature Vector of Gaussian Kernel has infinite dimensionality.



$$\kappa(x, x') = \exp \frac{-||x - x'||^2}{2\sigma^2}$$

$$= \exp \frac{-||x||^2 - ||x'||^2 + 2x^t x'}{2\sigma^2}$$

$$= \exp -||x||^2/2\sigma^2 \cdot \exp x^t x' \cdot \exp -||x'||^2/2$$

$$= \exp -||x||^2/2\sigma^2 \left(\sum_{k=0}^{\infty} \frac{(x^t x')^k}{k!}\right) \cdot \exp -||x'||^2/2$$

$$\exp^x = \sum_{k=1}^{\infty} \frac{x^k}{k!}$$ + Taylor Series!