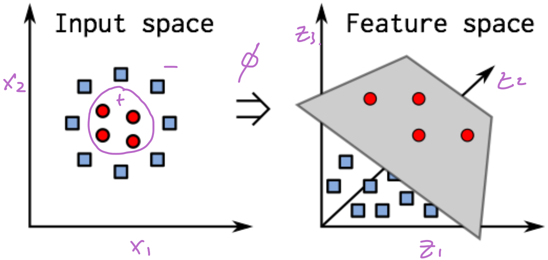
Kernel SVM

Non-linear SVM

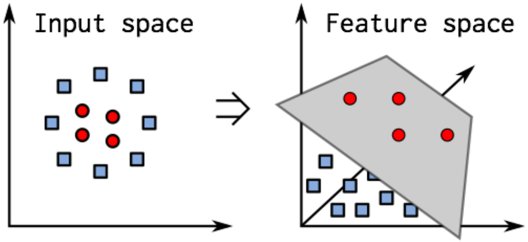
For non-separable data, we can use the **kernel trick**: Map input values $x \in \mathbb{R}^d$ to a higher dimension $\phi(x) \in \mathbb{R}^D$, such that the data becomes separable.



$\phi : \mathbb{R}^2 \to \mathbb{R}^3$

Non-linear SVM

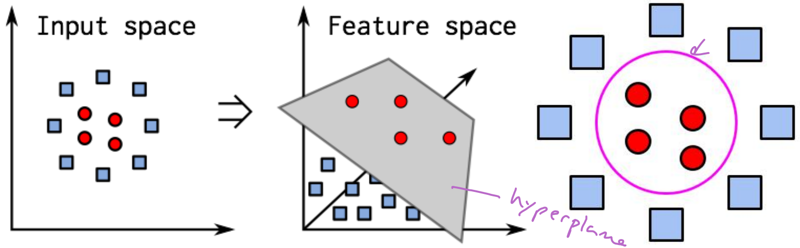
For non-separable data, we can use the **kernel trick**: Map input values $x \in \mathbb{R}^d$ to a higher dimension $\phi(x) \in \mathbb{R}^D$, such that the data becomes separable.



- ϕ is called a **feature mapping**.

Non-linear SVM

For non-separable data, we can use the **kernel trick**: Map input values $x \in \mathbb{R}^d$ to a higher dimension $\phi(x) \in \mathbb{R}^D$, such that the data becomes separable.



- ϕ is called a **feature mapping**.
- The classification function $w^T\boxed{x} + b$ becomes nonlinear: $w^T\boxed{\phi(x)} + b$

Kernel Function

Given a feature mapping ϕ, we define the **kernel function** to be

$$K(x, z) = \phi(x)^T \phi(z) \quad \in \mathbb{R}$$

Kernel Function

Given a feature mapping ϕ, we define the **kernel function** to be

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

Some kernel functions are easier to compute than $\phi(x)$, e.g.

Let's assume $x, z \in \mathbb{R}^2$

$$K(x, z) = \underline{(x^T z)^2}$$

$$\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^T \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \right)^2$$

$$= (x_1 z_1 + x_2 z_2)^2$$

$$= (x_1 z_1)^2 + (x_1 z_1)(x_2 z_2) + (x_2 z_2)(x_1 z_1) + (x_2 z_2)^2$$

$$= x_1^2 z_1^2 + (x_1 x_2)(z_1 z_2) + (x_2 x_1)(z_2 z_1) + (x_2^2 z_2^2)$$

$$= \begin{bmatrix} x_1^2 \\ x_1 x_2 \\ x_2 x_1 \\ x_2^2 \end{bmatrix}^T \begin{bmatrix} z_1^2 \\ z_1 z_2 \\ z_2 z_1 \\ z_2^2 \end{bmatrix}$$

$$= \langle \phi(x), \phi(z) \rangle \quad \text{where} \quad \phi(x) = \begin{bmatrix} x_1^2 \\ x_1 x_2 \\ x_2 x_1 \\ x_2^2 \end{bmatrix}$$

Kernel Function

Given a feature mapping ϕ, we define the **kernel function** to be

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

Some kernel functions are easier to compute than $\phi(x)$, e.g.

$$\begin{aligned} K(x, z) = (x^T z)^2 &= \left(\sum_{i=1}^{n} x_i z_i \right) \left(\sum_{j=1}^{n} x_j z_j \right) = \sum_{i=1}^{n} \sum_{j=1}^{n} x_i x_j z_i z_j \\ &= \phi(x)^T \phi(z) \end{aligned}$$

Kernel Function

Given a feature mapping ϕ, we define the **kernel function** to be

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

Some kernel functions are easier to compute than $\phi(x)$, e.g.

$$K(x, z) = (x^T z)^2 = \left(\sum_{i=1}^{n} x_i z_i\right)\left(\sum_{j=1}^{n} x_j z_j\right) = \sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j z_i z_j$$

$(x^T z)^2$

$$= \phi(x)^T \phi(z)$$

$x, z \in \mathbb{R}^n$

where $\phi(x) = \underbrace{\begin{bmatrix} x_1 x_1 \\ x_1 x_2 \\ \vdots \\ x_n x_{n-1} \\ x_n x_n \end{bmatrix}}_{n^2}$ takes $O(n^2)$ operations to compute, while

$(x^T z)^2$ only takes $O(n)$

Kernel SVM

$\downarrow$ ϕ is selected

In the dual problem, replace $\langle x_i, y_j \rangle$ with $\langle \phi(x_i), \phi(y_i) \rangle = K(x_i, x_j)$

$<x_i, x_j>$

$$\max_{\alpha} W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j K(x_i, x_j)$$

$$s.t.\ 0 \leq \alpha_i \leq C, i = 1, \ldots, m$$

$$\sum_{i=1}^{m} \alpha_i y^{(i)} = 0$$

Kernel SVM

In the dual problem, replace $\langle x_i, y_j \rangle$ with $\langle \phi(x_i), \phi(y_i) \rangle = K(x_i, x_j)$

$$\max_{\alpha} W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j K(x_i, x_j)$$

$$s.t.\ 0 \leq \alpha_i \leq C, i = 1, \ldots, m$$

$$\sum_{i=1}^{m} \alpha_i y^{(i)} = 0$$

No need to compute $\boxed{w^* = \sum_{i=1}^{m} \alpha_i^* y^{(i)} \phi(x^{(i)})}$ explicitly since

$w^T x + b$ training samples.

$$f(x) = w^T \phi(x) + b = \left(\boxed{\sum_{i=1}^{m} \alpha_i y^{(i)} \phi(x^{(i)})} \right)^T \phi(x) + b$$

new sample

$$= \sum_{i=1}^{m} \alpha_i y^{(i)} \langle \phi(x^{(i)}), \phi(x) \rangle + b$$

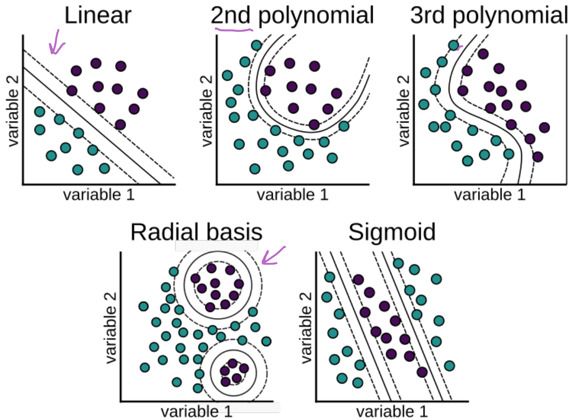
$$= \sum_{i=1}^{m} \alpha_i y^{(i)} K(x^{(i)}, x) + b$$

$\phi(x)$.

Kernel Matrix

kernel functions measure the similarity between samples x, z, e.g.

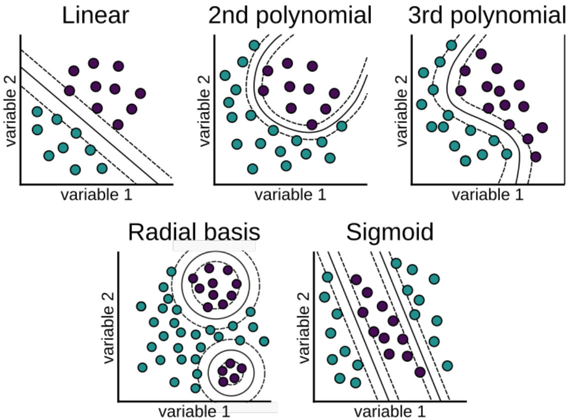
- Linear kernel: $K(x, z) = (x^T z)$ ϕ is identity function
- Polynomial kernel: $K(x, z) = (x^T z + 1)^p$ $(x^T z)^2$
- Gaussian / radial basis function (RBF) kernel:
  $K(x, z) = \exp\left(-\frac{||x-z||^2}{2\sigma^2}\right)$



Kernel Matrix

kernel functions measure the similarity between samples x, z, e.g.

- Linear kernel: $K(x,z) = (x^T z)$
- Polynomial kernel: $K(x,z) = (x^T z + 1)^p$
- Gaussian / radial basis function (RBF) kernel: $K(x,z) = \exp\left(-\frac{||x-z||^2}{2\sigma^2}\right)$



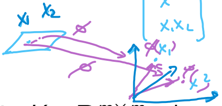
Can any function $K(x,y)$ be a kernel function?

$\phi(x)^T \phi(y)$

Kernel Matrix

$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \xrightarrow{\phi} \begin{bmatrix} x_1 \\ x_2 \\ x_1^2 \\ x^2 \\ x_1x_2 \end{bmatrix}$

$K(x^i, x^j) = \phi(x^i)^T\phi(x^i)$



$\phi(x_1)^T\phi(x_2) = K(x_1, x_2)$

matrix   function

Represent kernel function as a matrix $K \in \mathbb{R}^{m\times m}$ where $K_{i,j} = K(x_i, x_j) = \phi(x_i)^T\phi(x_j)$.

$K(x_1, \cdot)$

$x_1, \ldots, x_m$

$$K = \begin{array}{c} \\ m \\ \\ i \\ \\ \end{array} \begin{bmatrix} \phi(x_1)^T\phi(x_1) & \phi(x_1)^T\phi(x_2) & \cdots & \phi(x_1)^T\phi(x_m) \\ \phi(x_2^T)\phi(x_1) & & & \\ \vdots & & & \\ \vdots & \boxed{} & & \\ \phi(x_m)^T\phi(x_1) & \cdots & & \phi(x_m)^T\phi(x_m) \end{bmatrix}$$

j.

dig. $\rightarrow \|\phi(x_i)\|^2$

$\phi(x_i)^T\phi(x_j)$.

Kernel Matrix

Reduced Kernel Hilbert Space

Represent kernel function as a matrix $K \in \mathbb{R}^{m \times m}$ where $K_{i,j} = K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$.

Theorem (Mercer)

Let $K : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$ Then K is a valid (Mercer) kernel if and only if for any finite training set $\{x^{(i)}, \ldots, x^{(m)}\}$, K is symmetric positive semi-definite.

$(m \times m)$

i.e. $K_{i,j} = K_{j,i}$ and $\boxed{x^T} K x \geq 0$ for all $x \in \mathbb{R}^m$

$K = K^T$. $\quad (m \times 1)$ x

Two ways to show whether $K(x,z)$ is a valid kernel function:

① By definition: write $K(x,z) = \langle \phi(x), \phi(z) \rangle$

② Apply Mercer's theorem. show K matrix is SPSD

Kernel SVM Summary

RBF kernel

- Input: m training samples $(x^{(i)}, y^{(i)})$, $y^i \in \{-1, 1\}$, kernel function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, constant $C > 0$
- Output: non-linear decision function $f(x)$
- Step 1: solve the dual optimization problem for α^*

$\phi(x)$

kernel Trick

$$\max_{\alpha} W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j K(x^{(i)}, x^{(j)})$$

$$s.t.\ 0 \le \alpha_i \le C, \sum_{i=1}^{m} \alpha_i y^{(i)} = 0, i = 1, \ldots, m$$

- Step 2: compute the optimal decision function

$w^* =$

same (x^i, y^i) on the margin

$$\boxed{b^*} = y^{(j)} - \sum_{i=1}^{m} \alpha_i^* y^{(i)} K(x^{(i)}, x^{(j)}) \text{ for some } 0 < \alpha_j < C$$

$y^i = \sum_{j=1}^{m} \alpha_j^* y^j K(x^j, x^i) + b^*$

New x,

$$\boxed{f(x) = \sum_{i=1}^{m} \alpha_i y^{(i)} K(x^{(i)}, x) + b^*}$$

In practice, it's more efficient to compute kernel matrix K in advance.

SVM in Practice

(SMO)

Sequential Minimal Optimization: a fast algorithm for training soft margin kernel SVM

- Break a large SVM problem into smaller chunks, update two α_i's at a time
- Implemented by most SVM libraries.

SVM in Practice

Sequential Minimal Optimization: a fast algorithm for training soft margin kernel SVM

- Break a large SVM problem into smaller chunks, update two α_i's at a time
- Implemented by most SVM libraries.

Other related algorithms

- Support Vector Regression (SVR)
- Multi-class SVM (Koby Crammer and Yoram Singer. 2002. *On the algorithmic implementation of multiclass kernel-based vector machines*. J. Mach. Learn. Res. 2 (March 2002), 265-292.)