Support Vector Machines (SVM)

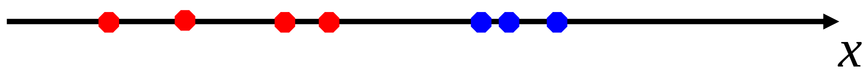
Data Intelligence and Learning (DIAL) Lab

Prof. Jongwuk Lee

Non-Linear SVM

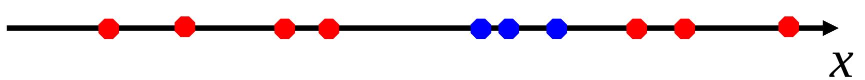
Motivation: Non-Linear SVM

- **Data that are linearly separable**



- **Data with noise**
  - linearly separable considering errors

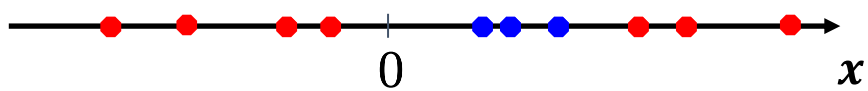
- **What about this?**



- **We need a non-linear boundary! But, how??**

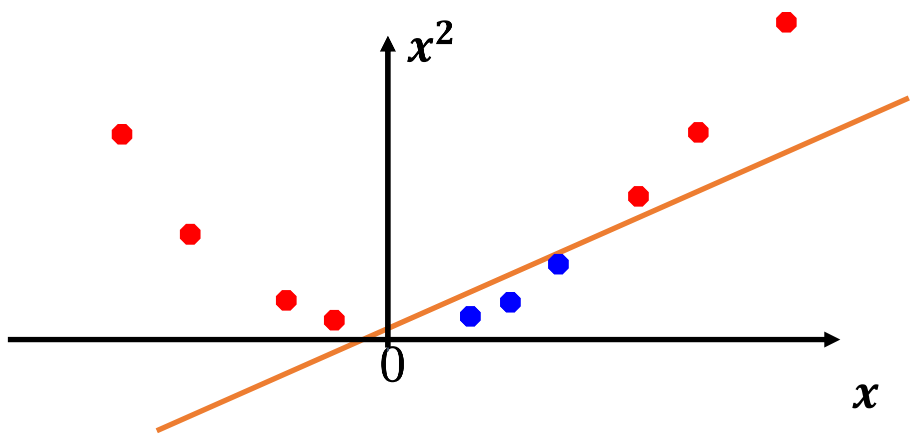
Motivation: Non-Linear SVM

- **Map data to a higher-dimensional space.**
- **Find a linear boundary in the higher-dimensional space.**



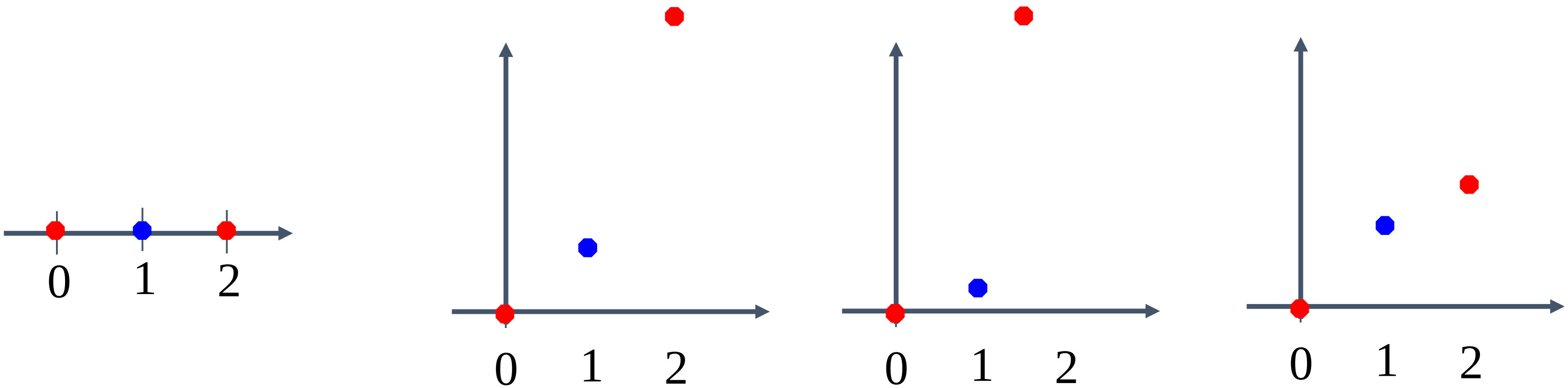
How can we find this mapping?

$$\Phi : x \rightarrow (x, x^2)$$



Non-Linear Mapping

- Most of the non-linear mapping functions does this!!

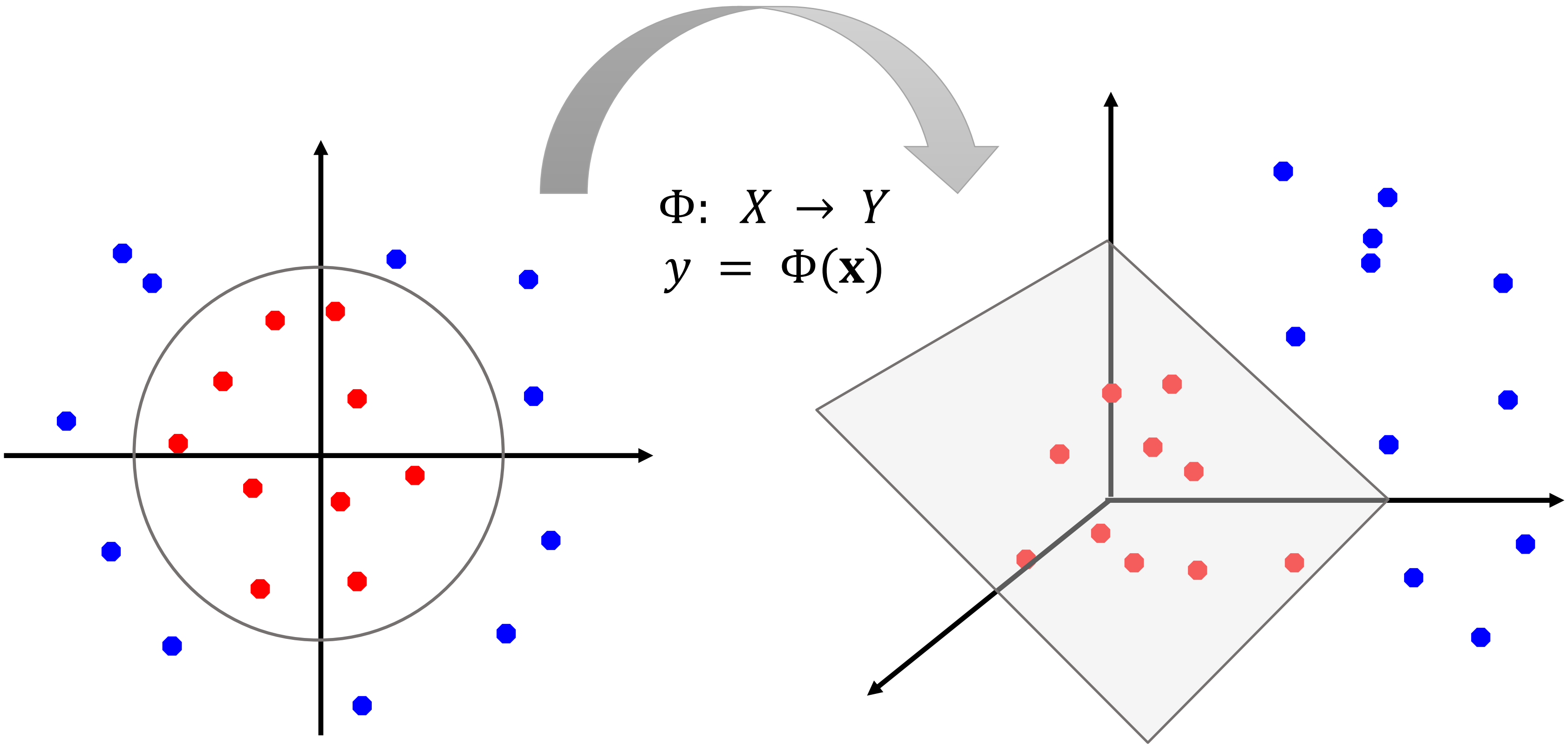


$\Phi: x \to (x, x^2)$ $\Phi: x \to (\sqrt{x}, e^x)$ $\Phi: x \to (x, \sqrt{x})$

- Then, how about higher dimensions?
- The higher dimension, the better.

Non-Linear Mapping

- The original input space can always be mapped to a higher-dimensional feature space in which classes are separable.



Recap: Formulating Soft Margin SVM

- **Given** $\mathcal{D} = \left\{\left(\mathbf{x}^{(i)}, y^{(i)}\right): 1 \leq i \leq n\right\}$**, where** $y^{(i)} \in \{-1, +1\}$**,**

$$\max_{\alpha_1,\cdots,\alpha_n} \left(\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y^{(i)} y^{(j)} \left(\mathbf{x}^{(i)} \cdot \mathbf{x}^{(j)}\right) \right)$$

$$\text{subject to} \begin{cases} \displaystyle\sum_{i=1}^{n} \alpha_i y^{(i)} = 0 \\ 0 \leq \alpha_i \leq C \quad\quad i = 1, \cdots, n \end{cases}$$

- **Solution**

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y^{(i)} \mathbf{x}^{(i)}$$

$$b = y^{(i)} - \mathbf{w}^{\mathrm{T}} \mathbf{x}^{(i)} \quad \text{for any } x^{(i)} \text{ such that } 0 < \alpha_i < C$$

Formulating Non-linear SVM

- **To consider a non-linear boundary,**
  - Given $\mathcal{D} = \left\{\left(\mathbf{x}^{(i)}, y^{(i)}\right): 1 \le i \le n\right\}$, where $y^{(i)} \in \{-1, +1\}$,
  - Define $\Phi: \mathbf{x} \rightarrow \Phi(\mathbf{x})$.
  - Convert data using $\mathcal{D} = \left\{\left(\Phi(\mathbf{x}^{(i)}), y^{(i)}\right): 1 \le i \le n\right\}$.

- **Formulation for the non-linear boundary**

$$\max_{\alpha_1,\cdots,\alpha_n} \left(\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y^{(i)} y^{(j)} \left(\Phi\left(\mathbf{x}^{(i)}\right) \cdot \Phi\left(\mathbf{x}^{(j)}\right) \right) \right)$$

$$\text{subject to} \begin{cases} \sum_{i=1}^{n} \alpha_i y^{(i)} = 0 \\ 0 \le \alpha_i \le C \quad i = 1, \cdots, n \end{cases}$$

Introduce non-linear mapping function.

Formulating Non-linear SVM

- **Given** $\mathcal{D} = \left\{\left(\mathbf{x}^{(i)}, y^{(i)}\right) : 1 \leq i \leq n\right\}$, **where** $y^{(i)} \in \{-1, +1\}$,

$$\max_{\alpha_1,\cdots,\alpha_n} \left(\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y^{(i)} y^{(j)} \left(\Phi\left(\mathbf{x}^{(i)}\right) \cdot \Phi\left(\mathbf{x}^{(j)}\right) \right) \right)$$

$$\text{subject to} \begin{cases} \displaystyle\sum_{i=1}^{n} \alpha_i y^{(i)} = 0 \\ 0 \leq \alpha_i \leq C \qquad i = 1, \cdots, n \end{cases}$$

- **Solution**

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y^{(i)} \Phi\left(\mathbf{x}^{(i)}\right)$$

$$b = y^{(i)} - \mathbf{w}^{\mathbf{T}} \Phi\left(\mathbf{x}^{(i)}\right) \text{ for any } \Phi\left(\mathbf{x}^{(i)}\right) \text{ such that } 0 < \alpha_i < C$$

Prediction for Test Samples

- The solution of SVM is as follows.

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y^{(i)} \Phi\left(\mathbf{x}^{(i)}\right)$$

$$b = y^{(i)} - \mathbf{w}^{\mathrm{T}} \Phi\left(\mathbf{x}^{(i)}\right) \text{ for any } \Phi\left(\mathbf{x}^{(i)}\right) \text{ such that } \alpha_i > 0$$

- Given a new sample $\mathbf{x}_{new}$,

$$\hat{y} = \mathrm{sign}(\mathbf{w}^{\mathrm{T}} \Phi(\mathbf{x}_{new}) + b)$$

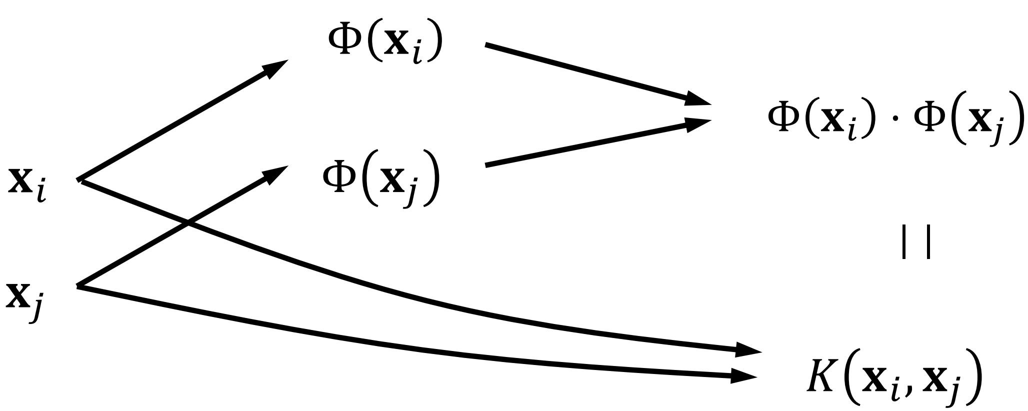
Kernel Trick for Non-linear SVM

What is a Kernel Function?

- It is the function that corresponds to **the dot product of two feature vectors in some expanded feature space.**

- We have two functions $\boldsymbol{\Phi}(\mathbf{x})$ and $K(\mathbf{x}_i, \mathbf{x}_j)$ and it happens that $\boldsymbol{\Phi}(\mathbf{x}_i)\boldsymbol{\Phi}(\mathbf{x}_j) = K(\mathbf{x}_i, \mathbf{x}_j)$.

- Then, $K(\cdot,\cdot)$ is called a kernel function.



What is Kernel Trick?

- **One possible transformation**

$$\Phi: (x_1, x_2) \to (x_1, x_2, x_1^2, x_2^2, x_1^3, x_2^3, x_1 x_2, x_1 x_2^2, x_1^2 x_2)$$

- **What about this?**

$$\Phi: (x_1, x_2) \to \left(1, \sqrt{3}x_1, \sqrt{3}x_2, \sqrt{3}x_1^2, \sqrt{3}x_2^2, x_1^3, x_2^3, \sqrt{6}x_1 x_2, \sqrt{3}x_1 x_2^2, \sqrt{3}x_1^2 x_2\right)$$

- **Evaluate $\boldsymbol{\Phi}\left(\mathbf{x}^{(i)}\right)\boldsymbol{\Phi}\left(\mathbf{x}^{(j)}\right)$.**

What is Kernel Trick?

- **Given two points $\mathbf{x}_1 = (x_{11}, x_{12})$ and $\mathbf{x}_2 = (x_{21}, x_{22})$**

$$\Phi(\mathbf{x}_1) = \left(1, \sqrt{3}x_{11}, \sqrt{3}x_{12}, \sqrt{3}x_{11}^2, \sqrt{3}x_{12}^2, x_{11}^3, x_{12}^3, \sqrt{6}x_{11}x_{12}, \sqrt{3}x_{11}x_{12}^2, \sqrt{3}x_{11}^2x_{12}\right)$$

$$\Phi(\mathbf{x}_2) = \left(1, \sqrt{3}x_{21}, \sqrt{3}x_{22}, \sqrt{3}x_{21}^2, \sqrt{3}x_{22}^2, x_{21}^3, x_{22}^3, \sqrt{6}x_{21}x_{22}, \sqrt{3}x_{21}x_{22}^2, \sqrt{3}x_{21}^2x_{22}\right)$$

$$\begin{aligned}
&\Phi(\mathbf{x}_2) \cdot \Phi(\mathbf{x}_2) \\
&= 1 + 3x_{11}x_{21} + 3x_{12}x_{22} + 3x_{11}^2x_{21}^2 + 3x_{12}^2x_{22}^2 + x_{11}^3x_{21}^3 + x_{12}^3x_{22}^3 \\
&\qquad + 6x_{11}x_{12}x_{21}x_{22} + 3x_{11}x_{12}^2x_{21}x_{22}^2 + 3x_{11}^2x_{12}x_{21}^2x_{22} \\
&= (x_{11}x_{21} + x_{12}x_{22})^3 + 3(x_{11}x_{21} + x_{12}x_{22})^2 + 3(x_{11}x_{21} + x_{12}x_{22})^1 + 1 \\
&\qquad = ((x_{11}x_{21} + x_{12}x_{22}) + 1)^3 = (\mathbf{x}_1 \cdot \mathbf{x}_2 + 1)^3
\end{aligned}$$

If the transform function is well-designed, we can easily evaluate the inner product!

Example: Kernel Trick

- Given two points $\mathbf{x_1} = (1,1)$ and $\mathbf{x_2} = (2,2)$

$$\Phi(\mathbf{x}_1) = \left(1, \sqrt{3}x_{11}, \sqrt{3}x_{12}, \sqrt{3}x_{11}^2, \sqrt{3}x_{12}^2, x_{11}^3, x_{12}^3, \sqrt{6}x_{11}x_{12}, \sqrt{3}x_{11}x_{12}^2, \sqrt{3}x_{11}^2x_{12}\right)$$

$$\Phi(\mathbf{x}_2) = \left(1, \sqrt{3}x_{21}, \sqrt{3}x_{22}, \sqrt{3}x_{21}^2, \sqrt{3}x_{22}^2, x_{21}^3, x_{22}^3, \sqrt{6}x_{21}x_{22}, \sqrt{3}x_{21}x_{22}^2, \sqrt{3}x_{21}^2x_{22}\right)$$

- In the transformed space, two points are

$$\Phi(\mathbf{x}_1) = \left(1,\ \sqrt{3},\ \sqrt{3},\ \sqrt{3},\ \sqrt{3},\ 1,\ 1,\ \sqrt{6},\ \sqrt{3},\ \sqrt{3}\right)$$
$$\Phi(\mathbf{x}_2) = \left(1,\ 2\sqrt{3},\ 2\sqrt{3},\ 4\sqrt{3},\ 4\sqrt{3},\ 8,\ 8,\ 4\sqrt{6},\ 8\sqrt{3},\ 8\sqrt{3}\right)$$

Example: Kernel Trick

$$\Phi(\mathbf{x}_1) = \left(1,\ \sqrt{3},\ \sqrt{3},\ \sqrt{3},\ \sqrt{3},\ 1,\ 1,\ \sqrt{6},\ \sqrt{3},\ \sqrt{3}\right)$$
$$\Phi(\mathbf{x}_2) = \left(1,\ 2\sqrt{3},\ 2\sqrt{3},\ 4\sqrt{3},\ 4\sqrt{3},\ 8,\ 8,\ 4\sqrt{6},\ 8\sqrt{3},\ 8\sqrt{3}\right)$$

$$\mathbf{x}_1 = (\ 1,\ 1\ )$$
$$\mathbf{x}_2 = (\ 2,\ 2\ )$$

$$\begin{aligned}\Phi(\mathbf{x}_2)\cdot\Phi(\mathbf{x}_2) &= 1+6+6+12+12+8+8 \\ &\quad +24+24+24 \\ &= 125\end{aligned}$$

$$K(\mathbf{x}_1,\mathbf{x}_2) = (\mathbf{x}_1\cdot\mathbf{x}_2+1)^3 = (4+1)^3 = 125$$

We can easily evaluate the inner product!

Common Kernels

- Polynomials of degree exactly d

$$K(\mathbf{u},\mathbf{v}) = (\mathbf{u}\cdot\mathbf{v})^d$$

- Polynomials of degree up to d,

$$K(\mathbf{u},\mathbf{v}) = (\mathbf{u}\cdot\mathbf{v}+\mathbf{1})^d$$

- Radial basis function (RBF) kernel

$$K(\mathbf{u},\mathbf{v}) = \exp\left(-\frac{\|\mathbf{u}-\mathbf{v}\|_2^2}{2\sigma^2}\right)$$

Polynomial Kernel Functions

- When $d = 1$,

$$\phi(\mathbf{u}) \cdot \phi(\mathbf{v}) = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \cdot \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = u_1 v_1 + u_2 v_2 = \mathbf{u} \cdot \mathbf{v}$$

- When $d = 2$,

$$\phi(\mathbf{u}) \cdot \phi(\mathbf{v}) = \begin{pmatrix} u_1^2 \\ u_1 u_2 \\ u_2 u_1 \\ u_2^2 \end{pmatrix} \cdot \begin{pmatrix} v_1^2 \\ v_1 v_2 \\ v_2 v_1 \\ v_2^2 \end{pmatrix}$$
$$= u_1^2 v_1^2 + 2 u_1 v_1 u_2 v_2 + u_2^2 v_2^2 = (u_1 v_1 + u_2 v_2)^2 = (\mathbf{u} \cdot \mathbf{v})^2$$

- For any d,

$$\phi(\mathbf{u}) \cdot \phi(\mathbf{v}) = (\mathbf{u} \cdot \mathbf{v})^d$$

Why is the RBF Kernel Effective?

- **Let $\sigma^2 = 1$.**

$$K(\mathbf{x},\mathbf{x}') = \exp\left(-\frac{1}{2}\|\mathbf{x}-\mathbf{x}'\|^2\right) = \exp\left(-\frac{1}{2}\langle \mathbf{x}-\mathbf{x}', \mathbf{x}-\mathbf{x}'\rangle\right)$$

$$= \exp\left(-\frac{1}{2}(\|\mathbf{x}\|^2 + \|\mathbf{x}'\|^2 - 2\langle \mathbf{x},\mathbf{x}'\rangle)\right) = \exp\left(-\frac{1}{2}(\|\mathbf{x}\|^2 + \|\mathbf{x}'\|^2)\right)\exp(\langle \mathbf{x},\mathbf{x}'\rangle)$$

Let $C \coloneqq \exp\left(-\frac{1}{2}(\|\mathbf{x}\|^2 + \|\mathbf{x}'\|^2)\right)$ be a constant.

$$= C\exp(\langle \mathbf{x},\mathbf{x}'\rangle)$$

By the Taylor extension of e^x

$$= C\sum_{n=0}^{\infty}\frac{\langle \mathbf{x},\mathbf{x}'\rangle^n}{n!}$$

The RBF kernel is formed by taking **an infinite sum over polynomial kernels**.

Example: Formulating Non-linear SVM

- **Given** $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)}) : 1 \le i \le n\}$**, where** $y^{(i)} \in \{-1, +1\}$**,**
  - $\Phi: (\mathrm{x}_1, \mathrm{x}_2) \to$ $\left(1, \sqrt{3}x_1, \sqrt{3}x_2, \sqrt{3}x_1^2, \sqrt{3}x_2^2, x_1^3, x_2^3, \sqrt{6}x_1x_2, \sqrt{3}x_1x_2^2, \sqrt{3}x_1^2x_2\right)$

$$\max_{\alpha_1,\cdots,\alpha_n} \left(\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y^{(i)} y^{(j)} \left(\Phi\left(\mathbf{x}^{(i)}\right) \cdot \Phi\left(\mathbf{x}^{(j)}\right) \right) \right)$$

$$\text{subject to} \begin{cases} \displaystyle\sum_{i=1}^{n} \alpha_i y^{(i)} = 0 \\ 0 \le \alpha_i \le C \qquad i = 1, \cdots, n \end{cases}$$

$$\max_{\alpha_1,\cdots,\alpha_n} \left(\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y^{(i)} y^{(j)} \left(\left(\mathbf{x}^{(i)} + \mathbf{x}^{(j)}\right)^3 \right) \right)$$

Formulation with Kernel Tricks

- **Given** $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)}): 1 \le i \le n\}$**, where** $y^{(i)} \in \{-1, +1\}$**,**
- **Choose** K **and** C**.**

$$\max_{\alpha_1,\cdots,\alpha_n} \left(\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y^{(i)} y^{(j)} K\left(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}\right) \right)$$

$$\text{subject to} \begin{cases} \displaystyle\sum_{i=1}^{n} \alpha_i y^{(i)} = 0 \\ 0 \le \alpha_i \le C \qquad i = 1, \cdots, n \end{cases}$$

- **For high-dimensional mapping, we can easily compute the inner product using the kernel trick.**

Formulation with Kernel Tricks

➢ Solution for the decision boundary

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y^{(i)} \Phi\left(\mathbf{x}^{(i)}\right)$$

$$b = y^{(k)} - \mathbf{w}^{\mathrm{T}} \cdot \Phi\left(\mathbf{x}^{(i)}\right)$$

$$= y^{(k)} - \sum_{i=1}^{n} \alpha_i y^{(i)} K\left(\mathbf{x}^{(i)}, \mathbf{x}^{(k)}\right) \text{ for any } k \text{ such that } 0 < \alpha_k < C$$

By applying $\mathbf{w} = \sum_{i=1}^{n} \alpha_i y^{(i)} \Phi\left(\mathbf{x}^{(i)}\right)$,

We can get $\mathbf{w}^{\mathrm{T}} \cdot \Phi\left(\mathbf{x}^{(i)}\right) = \sum_{i=1}^{n} \alpha_i y^{(i)} K\left(\mathbf{x}^{(i)}, \mathbf{x}^{(k)}\right)$.

Prediction for Test Samples

- The solution of SVM is as follows.

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y^{(i)} \Phi\left(\mathbf{x}^{(i)}\right)$$

$$b = y^{(k)} - \sum_{i=1}^{n} \alpha_i y^{(i)} K\left(\mathbf{x}^{(i)}, \mathbf{x}^{(k)}\right) \text{ for any } k \text{ such that } 0 < \alpha_k < C$$

- Given a new sample $\mathbf{x}_{new}$,

$$\hat{y} = \operatorname{sign}(\mathbf{w}^{\mathrm{T}} \Phi(\mathbf{x}_{new}) + b)$$

- **Do we consider the computation on the transformed space?**

Prediction for Test Samples

- Given a new sample $\mathbf{x}_{new}$,

$$\hat{y} = \mathrm{sign}(\mathbf{w}^{\mathrm{T}}\Phi(\mathbf{x}_{new}) + b)$$

By applying $\mathbf{w} = \sum_{i=1}^{n} \alpha_i y^{(i)} \Phi\left(\mathbf{x}^{(i)}\right)$,
We can get $\mathbf{w}^{\mathrm{T}} \cdot \Phi(\mathbf{x}_{new}) = \sum_{i=1}^{n} \alpha_i y^{(i)} K\left(\mathbf{x}^{(i)}, \mathbf{x}_{new}\right)$.

- Finally, our prediction is

$$\hat{y} = \mathrm{sign}\left(\sum_{i=1}^{n} \alpha_i y^{(i)} K\left(\mathbf{x}^{(i)}, \mathbf{x}_{new}\right) + b\right)$$

- **Still, we do not have to consider a transformed space.**

Summary: Kernel SVM

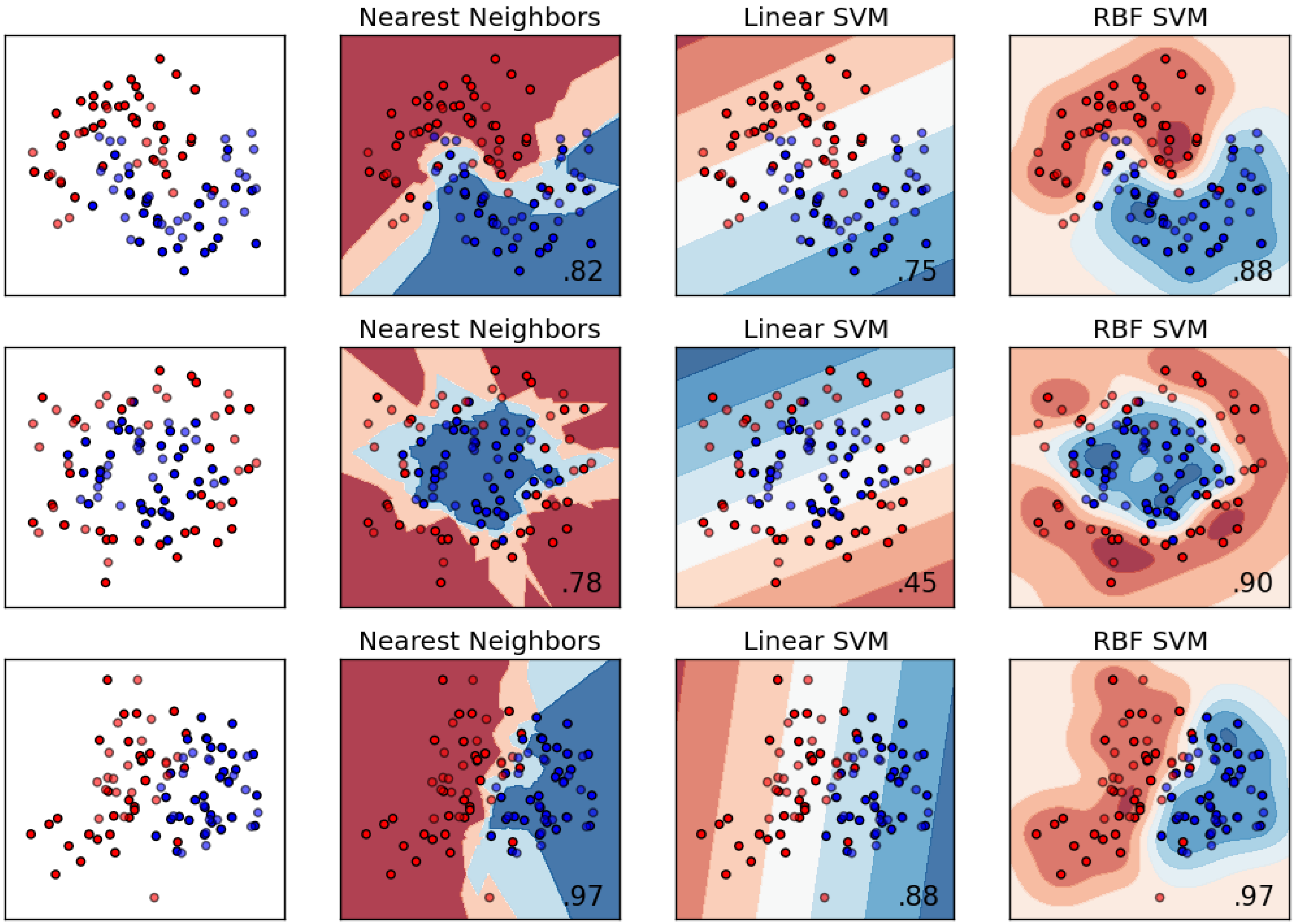
- **Choose a kernel function.**
  - RBF Kernels are mostly used.
  - To choose proper parameters, use k-fold validation.

- **Choose a value for C.**
  - To choose a proper value, use k-fold validation.

- **Solve the quadratic programming problem (many software packages available).**

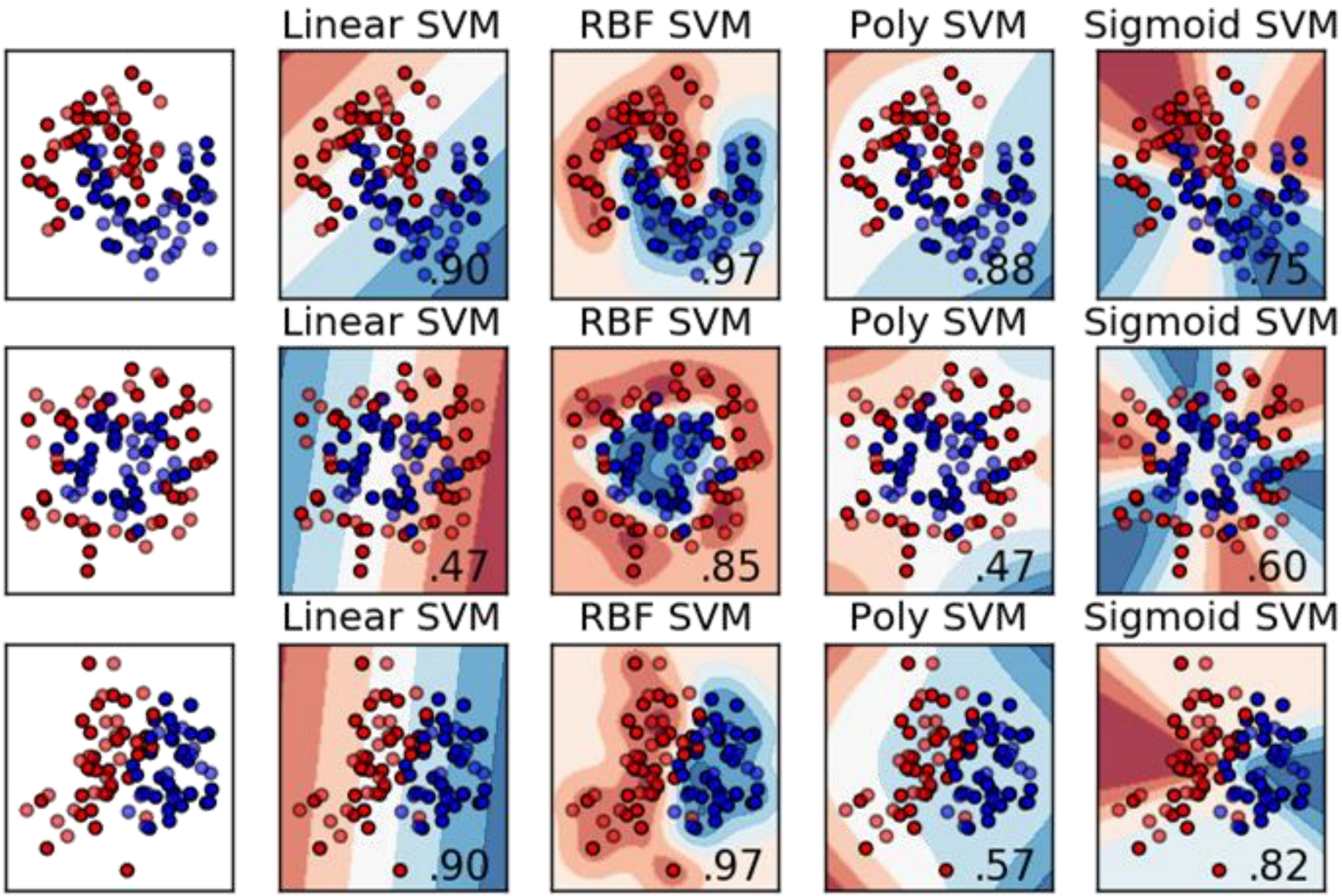
Q&A

Liner SVM vs. RBF SVM



https://martin-thoma.com/comparing-classifiers/

SVM with Various Kernel Functions



https://www.kaggle.com/code/residentmario/kernels-and-support-vector-machine-regularization/notebook