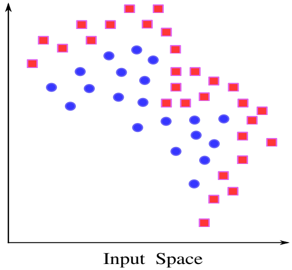
Kernel Function & Methods

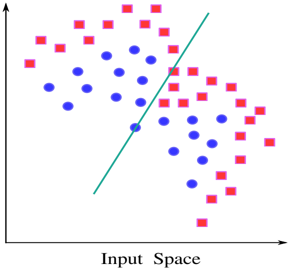
By Van Dinh Tran

February 16, 2025

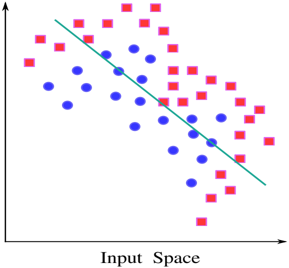
Kernel function



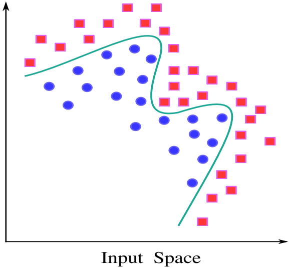
Kernel function



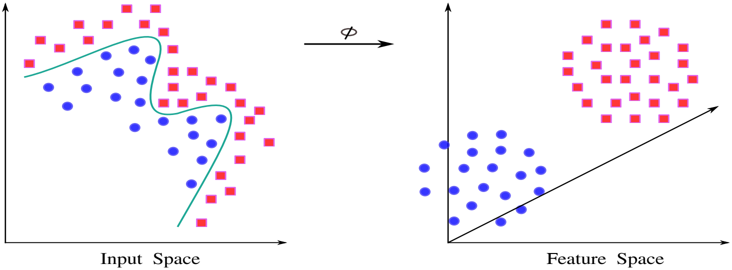
Kernel function



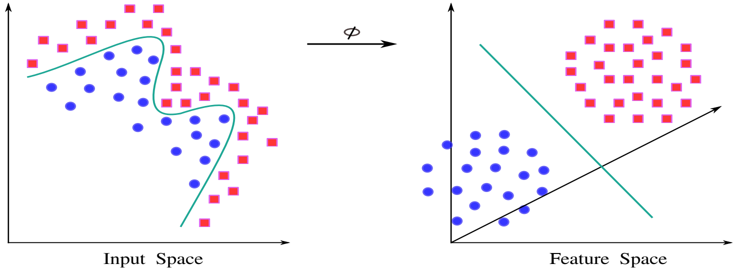
Kernel function



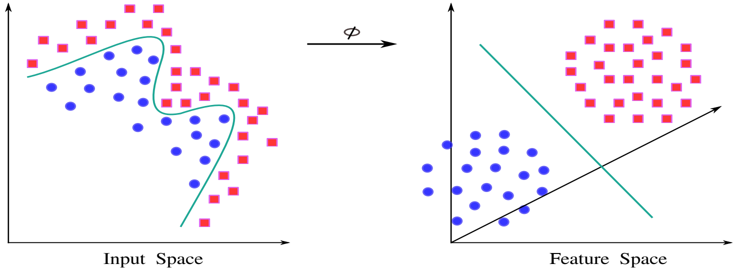
Kernel function



Kernel function



Kernel function



- In feature space, data are more likely to be linearly separable
- To build a model in feature space
  - Transform data to feature space ($\Phi\colon \mathbb{R}^n \mapsto \mathbb{R}^m$ $(n \gg m)$)
  - Train a model in feature space
- Problem: often leads to high computation

Kernel function

Question: What if the train and prediction of a model in feature space are involved in only pairwise similarities, but not individual representations?

$$\langle \Phi(\mathbf{x}_\mathrm{i}), \Phi(\mathbf{x}_\mathrm{j}) \rangle$$

Kernel function

Question: What if the train and prediction of a model in feature space are involved in only pairwise similarities but not individual representations?

$$\langle \Phi(\mathbf{x}_\mathrm{i}), \Phi(\mathbf{x}_\mathrm{j}) \rangle$$

Idea: Compute $\langle \Phi(\mathbf{x}_\mathrm{i}), \Phi(\mathbf{x}_\mathrm{j}) \rangle$ through operations in input space, so **no transformation** is needed.

Kernel function

Question: What if the train and prediction of a model in feature space are involved in only pairwise similarities, but not individual representations?

$$\langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle$$

Idea: Compute $\langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle$ through operations in input space, so **no transformation** is needed.

Solution: Defining a kernel function

$$\mathrm{k}\colon \mathbb{X} \times \mathbb{X} \longrightarrow \mathbb{R}$$
$$\text{such that } \mathrm{k}(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i)^\intercal \Phi(\mathbf{x}_j)$$

Mercer's condition: *If the function* $\mathrm{k}\colon \mathbb{X} \times \mathbb{X} \longrightarrow \mathbb{R}$ *is symmetric, positive semi-definite, there exists a space* $\mathbb{F}$ *and a mapping* $\Phi\colon \mathbb{X} \mapsto \mathbb{F}$ *such that* $\mathrm{k}(\mathbf{x}, \mathbf{y}) = \Phi^\intercal(\mathbf{x})\Phi(\mathbf{y})$.

- Using kernel to avoid explicit mapping is called *kernel trick*
- The matrix K, $\mathrm{K}_{ij} = \mathrm{k}(\mathbf{x_i}, \mathbf{x_j})$, is called *kernel matrix* or *gram-matrix*

Example of an explicit kernel

Problem: Given $\Phi\colon (\mathbf{x}_1, \mathbf{x}_2) \mapsto (\mathbf{x}_1^2, \sqrt{2}\mathbf{x}_1\mathbf{x}_2, \mathbf{x}_2^2)$ and $\mathrm{k}(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^\intercal \mathbf{y})^2$. Proof that k is a valid kernel. (Hint: $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^2 : \mathrm{k}(\mathbf{x}, \mathbf{y}) = \Phi^\intercal(\mathbf{x})\Phi(\mathbf{y})$).

Example of an explicit kernel

Problem: Given $\Phi \colon (\mathbf{x}_1, \mathbf{x}_2) \mapsto (\mathbf{x}_1^2, \sqrt{2}\mathbf{x}_1\mathbf{x}_2, \mathbf{x}_2^2)$ and $\mathrm{k}(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^\intercal \mathbf{y})^2$. Proof that k is a valid kernel. (Hint: $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^2 : \mathrm{k}(\mathbf{x}, \mathbf{y}) = \Phi^\intercal(\mathbf{x})\Phi(\mathbf{y})$).

Solution:

- Compute $\Phi^\intercal(\mathbf{x})\Phi(\mathbf{y})$
  - $\Phi(\mathbf{x}) = (\mathbf{x}_1^2, \sqrt{2}\mathbf{x}_1\mathbf{x}_2, \mathbf{x}_2^2)$
  - $\Phi(\mathbf{y}) = (\mathbf{y}_1^2, \sqrt{2}\mathbf{y}_1\mathbf{y}_2, \mathbf{y}_2^2)$

  $\implies \Phi^\intercal(\mathbf{x})\Phi(\mathbf{y}) = \mathbf{x}_1^2\mathbf{y}_1^2 + 2\mathbf{x}_1\mathbf{y}_1\mathbf{x}_2\mathbf{y}_2 + \mathbf{x}_2^2\mathbf{y}_2^2$
- Compute $\mathrm{k}(x, y)$

$$\begin{aligned} \mathrm{k}(\mathbf{x}, \mathbf{y}) &= (\mathbf{x}^\intercal \mathbf{y})^2 \\ &= (\mathbf{x}_1\mathbf{y}_1 + \mathbf{x}_2\mathbf{y}_2)^2 \\ &= \mathbf{x}_1^2\mathbf{y}_1^2 + 2\mathbf{x}_1\mathbf{y}_1\mathbf{x}_2\mathbf{y}_2 + \mathbf{x}_2^2\mathbf{y}_2^2 \end{aligned}$$

$\implies \mathrm{k}(\mathbf{x}, \mathbf{y}) = \Phi^\intercal(\mathbf{x})\Phi(\mathbf{y})$

Some existing kernels

- **Euclidean kernels**
  - Gaussian radial basis function (RBF): $\mathrm{k}(\mathbf{x}, \mathbf{y}) = \exp(-\gamma \|\mathbf{x} - \mathbf{y}\|^2), \ \gamma > 0$
  - Polynomial kernel: $\mathrm{k}(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^\intercal \mathbf{y} + 1)^{\mathrm{d}}$
  - Gaussian kernel: $\mathrm{k}(\mathbf{x}, \mathbf{y}) = \exp(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\delta^2})$
- **Graph kernels**
  - Diffusion-based kernels
  - Decomposition-based kernels

Constructing kernels

We can construct kernels from existing kernels. Given k_i as valid kernels, a valid kernel k can be constructed in the following ways.

- **Scalar multiplication:** $k(x, x') = \alpha k_1(x, x'), \quad \alpha \geq 0$
- **Adding a constant:** $k(x, x') = k_1(x, x') + \alpha, \quad \alpha \geq 0$
- **Linear combination:** $k(x, x') = \sum_{i=1}^{n} \alpha_i k_i(x, x'), \quad \alpha_i \geq 0$
- **Product:** $k(x, x') = k_1(x, x').k_2(x, x')$
- **Polynomial function of a kernel:** $k(x, x') = P(k_1(x, x'))$
- **Exponential function of a kernel:** $k(x, x') = \exp(k_1(x, x'))$

Kernel methods

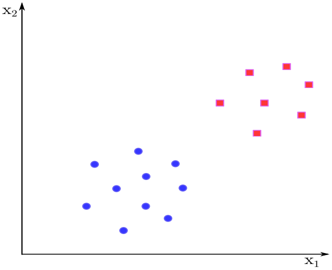
- Kernel methods are a class of learning algorithms that use *kernel functions*. Some examples are:
  - Kernel perceptron
  - Support Vector Machine (SVM)
  - Gaussian processes
- Kernel methods work on pairwise similarities between instances $\Longrightarrow$ The explicit representations of instances are not needed.
- Kernel methods with the use of kernel functions allow to operate in a high dimensional space without ever computing the coordinates of the data in that space, but rather in "low" dimensional space.

Kernel methods

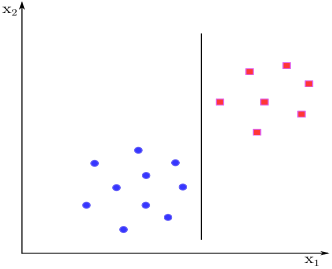
- Kernel methods are a class of learning algorithms that use *kernel functions*. Some examples are:
  - Kernel perceptron
  - Support Vector Machine (SVM)
  - Gaussian processes
- Kernel methods work on pairwise similarities between instances $\Longrightarrow$ The explicit representations of instances are not needed.
- Kernel methods with the use of kernel functions can operate in a "high" dimensional space without ever computing the coordinates of the data in that space, but rather in "low" dimensional space.

We focus on SVM [*Vapnik et al., 1997*], a linear method and the best known member of kernel methods.

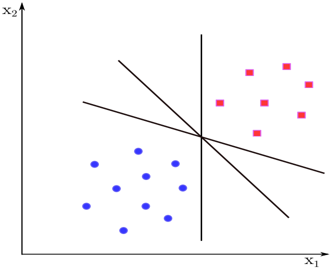
Hard-margin SVM



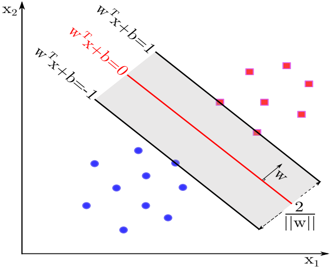
Hard-margin SVM



Hard-margin SVM



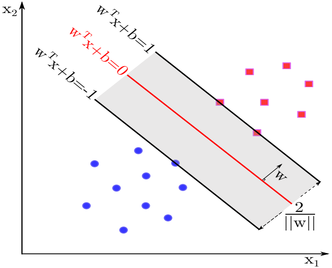
Hard-margin SVM



Hard-margin SVM

We define:

- h: $\mathbf{w}^\intercal \mathbf{x} + b = 0$ (decision boundary)
  If $(\mathbf{w}^\intercal \mathbf{x} + b > 0)$, Label "+1"
  If $(\mathbf{w}^\intercal \mathbf{x} + b < 0)$, Label "-1"
- $h_\oplus$: $\mathbf{w}^\intercal \mathbf{x}_\oplus + b = 1$ (positive hyperplane)
- $h_\ominus$: $\mathbf{w}^\intercal \mathbf{x}_\ominus + b = -1$ (negative hyperplane)



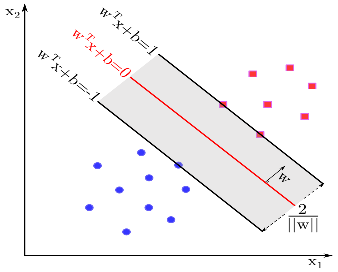
It can be seen that

- $\mathbf{w}^\intercal \mathbf{x}_+ + b \geq 1 \Rightarrow y_+(\mathbf{w}^\intercal \mathbf{x}_+ + b \geq 1)$
- $\mathbf{w}^\intercal \mathbf{x}_- + b \leq -1 \Rightarrow y_-(\mathbf{w}^\intercal \mathbf{x}_- + b \geq 1)$

$\Rightarrow y_i(\mathbf{w}^\intercal \mathbf{x}_i + b) \geq 1$

Margin

$$\begin{aligned}
\textbf{Margin} &= \mathrm{d}(\mathrm{h}_{\oplus}, \mathrm{h}_{\ominus}) \\
&= \mathrm{d}(\mathbf{x}_0 \in \mathrm{h}_{\oplus}, \mathrm{h}_{\ominus}) \\
&= \frac{|\mathbf{w}\mathbf{x}_0 + \mathrm{b} + 1|}{\|\mathbf{w}\|} \\
&= \frac{\left|(\mathbf{w}\mathbf{x}_0 + \mathrm{b} - 1) + 2\right|}{\|\mathbf{w}\|} \\
&= \frac{2}{\|\mathbf{w}\|}
\end{aligned}$$



Maximize margin

$$\underset{\mathbf{w}}{\mathrm{Max}}\ \frac{2}{\|\mathbf{w}\|}$$

Subject to $\mathrm{y_i}(\mathbf{w}^\intercal \mathbf{x}_\mathrm{i} + \mathrm{b}) \geq 1,\ \mathrm{i} = 1, \ldots, \mathrm{N}$

$$\underset{\mathbf{w}}{\mathrm{Min}}\ \frac{1}{2}\mathbf{w}^\intercal \mathbf{w}$$

Subject to $-(\mathrm{y_i}(\mathbf{w}^\intercal \mathbf{x}_\mathrm{i} + \mathrm{b}) - 1) \leq 0,\ \mathrm{i} = 1, \ldots, \mathrm{N}$

Note that optimization with constraints:

- Equalities $\rightarrow$ Lagrange multipliers
- Inequalities $\rightarrow$ KKT (Karush–Kuhn–Tucker)

Optimization

Lagrangian function:

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\mathbf{w}^{\intercal}\mathbf{w} - \sum_{i=1}^{N} \alpha_i (y_i(\mathbf{w}^{\intercal}\mathbf{x}_i + b) - 1)$$

$$\text{Subject to } \alpha_i \geq 0$$

Dual form:

$$\underset{\boldsymbol{\alpha}}{\text{Max}}\ \underset{\mathbf{w}, b}{\text{Min}}\ \mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha})$$

$$\text{Subject to } \alpha_i \geq 0$$

Solving optimization

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\mathbf{w}^\intercal \mathbf{w} - \sum_{i=1}^{N} \alpha_i (y_i (\mathbf{w}^\intercal \mathbf{x}_i + b) - 1)$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i = 0 \Longrightarrow \boxed{\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i}$$

$$\frac{\partial \mathcal{L}}{\partial b} = \boxed{\sum_{i=1}^{N} \alpha_i y_i = 0}$$

Substituting into Lagrangian function

$$\begin{aligned}\mathcal{L}(\boldsymbol{\alpha}) &= \frac{1}{2}\sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i \sum_{j=1}^{N} \alpha_j y_j \mathbf{x}_j - \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i \sum_{j=1}^{N} \alpha_j y_j \mathbf{x}_j - b \sum_{i=1}^{N} \alpha_i y_i + \sum_{i=1}^{N} \alpha_i \\ &= \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\intercal \mathbf{x}_j\end{aligned}$$

Solving optimization

$$\begin{aligned}&\underset{\boldsymbol{\alpha}}{\text{Max}}\,\mathcal{L}(\boldsymbol{\alpha})\\&\text{Subject to } \alpha_i \geq 0\end{aligned} \quad \Leftrightarrow \quad \begin{aligned}&\underset{\boldsymbol{\alpha}}{\text{Max}}\Big(\sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^\intercal \mathbf{x}_j\Big)\\&\text{Subject to } \alpha_i \geq 0\end{aligned}$$

Solving this optimization by using quadratic programming tools, we get

$$\begin{cases}\alpha_i > 0 & \text{if } \mathbf{x}_i \in h_\oplus \text{ or } \mathbf{x}_i \in h_\ominus\\ \alpha_i = 0 & \text{otherwise}\end{cases}$$

$$\mathbf{w} = \sum_{\mathbf{x}_i \text{ is SV}} \alpha_i y_i \mathbf{x}_i$$

- Find b: $y_j(\mathbf{w}^\intercal \mathbf{x}_j + b) = 1 \longrightarrow b = y_j - \sum_{\mathbf{x}_i \text{ is SV}} \alpha_i y_i \mathbf{x}_i^\intercal \mathbf{x}_j$, where $\mathbf{x}_j$ is any SV.

- Decision boundary: $\mathbf{w}^\intercal \mathbf{x} + b = 0 \Longleftrightarrow \sum_{\mathbf{x}_i \text{ is SV}} \alpha_i y_i \mathbf{x}_i^\intercal \mathbf{x} + b = 0$

- Decision Rule:

$$\boxed{y = \text{Sign}\Big(\sum_{\mathbf{x}_i \text{ is SV}} \alpha_i y_i \mathbf{x}_i^\intercal \mathbf{x} + b\Big)}$$

Solving optimization

$$\begin{aligned}&\underset{\boldsymbol{\alpha}}{\text{Max}}\,\mathcal{L}(\boldsymbol{\alpha})\\&\text{Subject to } \alpha_i \geq 0\end{aligned} \quad \Leftrightarrow \quad \begin{aligned}&\underset{\boldsymbol{\alpha}}{\text{Max}}\Big(\sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j)\Big)\\&\text{Subject to } \alpha_i \geq 0\end{aligned}$$

Solving this optimization by using quadratic programming tools, we get

$$\begin{cases}\alpha_i > 0 & \text{if } \mathbf{x}_i \in h_{\oplus} \text{ or } \mathbf{x}_i \in h_{\ominus}\\ \alpha_i = 0 & \text{otherwise}\end{cases}$$

$$\mathbf{w} = \sum_{\mathbf{x}_i \text{ is SV}} \alpha_i y_i \mathbf{x}_i$$

- Find b: $y_j(\mathbf{w}^\intercal \mathbf{x}_j + b) = 1 \longrightarrow b = y_j - \sum_{\mathbf{x}_i \text{ is SV}} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}_j)$, where $\mathbf{x}_j$ is any SV.

- Decision boundary: $\mathbf{w}^\intercal \mathbf{x} + b = 0 \Longleftrightarrow \sum_{\mathbf{x}_i \text{ is SV}} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) + b = 0$

- Decision Rule:

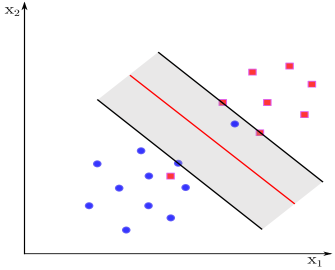
$$\boxed{y = \text{Sign}\Big(\sum_{\mathbf{x}_i \text{ is SV}} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) + b\Big)}$$

Soft-margin SVM

- Margin violation: $y_i(\mathbf{w}^\intercal \mathbf{x}_i + b) \ngeq 1$
- $\forall \mathbf{x}_i$, allowing error ξ_i, $\xi_i \geq 0$

$$y_i(\mathbf{w}^\intercal \mathbf{x}_i + b) \geq 1 - \xi_i$$

- Total error: $E = \sum_{i=1}^{N} \xi_i$



$$\operatorname*{Min}_{\mathbf{w}} \frac{1}{2}\mathbf{w}^\intercal \mathbf{w} + C \sum_{i=1}^{N} \xi_i$$

Subject to $y_i(\mathbf{w}^\intercal \mathbf{x}_i + b) \geq 1 - \xi_i;\ \xi_i \geq 0$

(C controls the cost of misclassification on the training data)

Soft-margin SVM

Lagrangian function:

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{2}\mathbf{w}^\intercal \mathbf{w} + C \sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} \alpha_i (y_i(\mathbf{w}\mathbf{x}_i + b) - 1 + \xi_i) - \sum_{i=1}^{N} \beta_i \xi_i$$

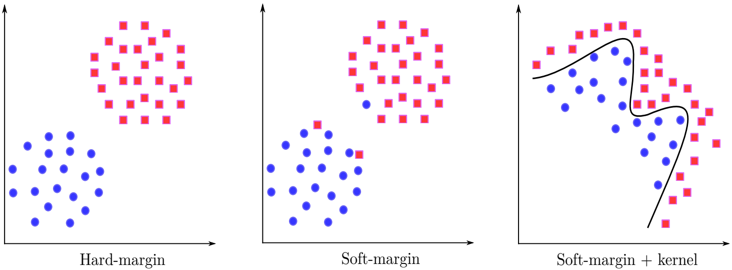
$$\text{Subject to } \alpha_i \geq 0;\ \beta_i \geq 0$$

Dual form:

$$\underset{\boldsymbol{\alpha},\boldsymbol{\beta}}{\text{Max}}\ \underset{\mathbf{w},b,\boldsymbol{\xi}}{\text{Min}}\ \mathcal{L}(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta})$$

$$\text{Subject to } \alpha_i \geq 0;\ \beta_i \geq 0$$

SVM in practice



In most cases, soft-margin SVM with kernel is used in practice.

SVM: advantages and disadvantages

- **Advantages:**
  - Scaling relatively well to high dimensional data
  - Working well with unstructured and semi-structured data
  - Generalizing well; the risk of over-fitting is less in SVM
  - Kernel trick is the real strength of SVM
  - Efficient in prediction
- **Disadvantages:**
  - Not easy to choose an appropriate kernel function
  - High time computation for large datasets
  - Difficult to understand and interpret