

Chapter 2

Learning Linear Separators, SVMs and Kernels

Contents

2	Learning Linear Separators, SVMs and Kernels	1
2.1	Linear Separator	2
2.2	Perceptron Algorithm	3
2.2.1	Example	4
2.3	Geometric Margin	4
2.4	Support Vector Machine	5
2.5	Optimal Linear Separator	5
2.6	Classification Margin	5
2.7	Maximizing the Margin	6
2.8	Linear SVM	6
2.9	Lagrangian Duality	7
2.10	SVM Solution	8
2.11	Soft Margin Classification	8
2.12	Soft Margin SVM Solution	9
2.13	Kernel Method	9
2.13.1	Local Kernels	10
2.13.2	Global Kernels	10
2.14	Kernel Functions	10
2.15	Kernel Trick	11
2.16	Determining Kernels	11
2.16.1	Mercer's theorem	12
2.17	Non-linear SVMs	12
2.18	Kernel Closure Properties	12
2.19	Kernel Benefits	13
2.20	Conclusion	13

2.1 Linear Separator

Assuming that red and blue datasets represents points X_1 and X_2 , then the two sets X_1 and X_2 are linearly separable if there exists $(n + 1)$ real numbers w_1, w_2, \dots, w_n, k

- such that every point in X_1 satisfies $\sum_{i=1}^n w_i x_i < k$
- such that every point in X_2 satisfies $\sum_{i=1}^n w_i x_i > k$

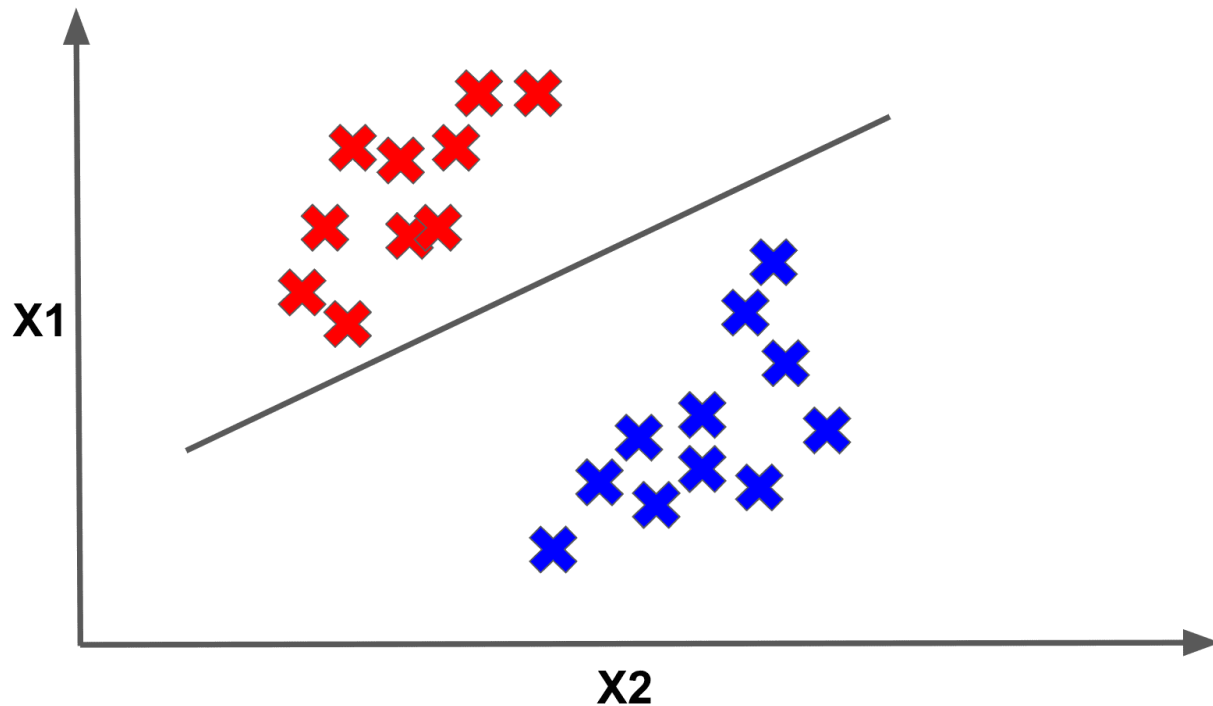
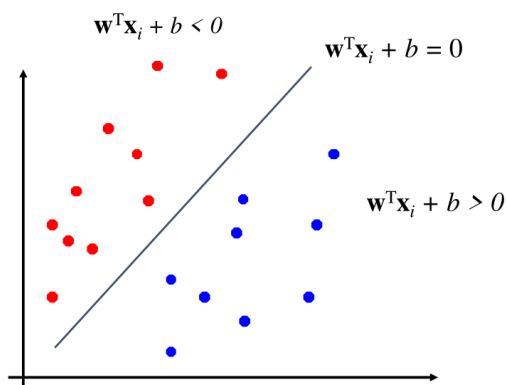


Figure 2.1: Linear separator is a vector-threshold pair (w, k) which can satisfy these 2 relations.

Binary classification $y_i \in \{-1, 1\}$ can be viewed as the task of separating classes in feature space.



- Hypothesis class of linear decision surfaces is $f(\mathbf{x}_i) = \text{sign}(\mathbf{w}^T \mathbf{x}_i + b)$.
- Without loss of generality, we assume that $b = 0$. Thus, we get the simplified $f(x_i) = \text{sign}(\mathbf{w}^T \mathbf{x}_i)$.
- $y_i(\mathbf{w}^T \mathbf{x}_i) > 0 \Leftrightarrow$ data point x_i is correctly classified.
 - Remember, y_i is counting as 1 or -1.

Figure 2.2: Vectorized linear separator.

2.2 Perceptron Algorithm

- Set time $t = 1$, start with vector $\mathbf{w}_1 = \vec{0}$.
- Given example \mathbf{x} , predict positive iff (if and only if) $\mathbf{w}_1 \cdot \mathbf{x} \geq 0$.

- On a mistake, update as follows:
 - Mistake on **positive**, then update $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \mathbf{x}$.
 - Mistake on **negative**, then update $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \mathbf{x}$.

2.2.1 Example

$$\begin{array}{llll} \mathbf{x}_i: & (1, 2) & (2, 3) & (2, 1) & (3, 0) \\ \mathbf{y}_i: & + & + & - & - \end{array}$$

$$\begin{aligned} \mathbf{w}_1 &= [0 \ 0]^T \\ \mathbf{w}_1 \cdot \mathbf{x} &= [0 \ 0]^T \cdot [1 \ 2] = 0 \rightarrow \text{predict } (+) \text{ correct} \\ \mathbf{w}_1 \cdot \mathbf{x} &= [0 \ 0]^T \cdot [2 \ 3] = 0 \rightarrow \text{predict } (+) \text{ correct} \\ \mathbf{w}_1 \cdot \mathbf{x} &= [0 \ 0]^T \cdot [2 \ 1] = 0 \rightarrow \text{predict } (+) \text{ wrong} \\ \mathbf{w}_2 &= \mathbf{w}_1 - [2 \ 1] = [-2 \ -1] \\ \mathbf{w}_2 \cdot \mathbf{x} &= [-2 \ -1]^T \cdot [3 \ 0] < 0 \rightarrow \text{predict } (-) \text{ correct} \\ \mathbf{w}_2 \cdot \mathbf{x} &= [-2 \ -1]^T \cdot [1 \ 2] < 0 \rightarrow \text{predict } (-) \text{ wrong} \\ \mathbf{w}_3 &= \mathbf{w}_2 + [1 \ 2] = [-1 \ 1] \\ \mathbf{w}_3 \cdot \mathbf{x} &= [-1 \ 1]^T \cdot [2 \ 3] > 0 \rightarrow \text{predict } (+) \text{ correct} \\ \mathbf{w}_3 \cdot \mathbf{x} &= [-1 \ 1]^T \cdot [2 \ 1] < 0 \rightarrow \text{predict } (-) \text{ correct} \\ \mathbf{w}_3 \cdot \mathbf{x} &= [-1 \ 1]^T \cdot [3 \ 0] < 0 \rightarrow \text{predict } (-) \text{ correct} \\ \mathbf{w}_3 \cdot \mathbf{x} &= [-1 \ 1]^T \cdot [1 \ 2] > 0 \rightarrow \text{predict } (+) \text{ correct} \end{aligned}$$

$$\mathbf{w}_3 \cdot \mathbf{x} = [-1 \ 1]^T \cdot [x_1 \ x_2]$$

$-1x_1 + 1x_2 = 0$ is the linear separator.

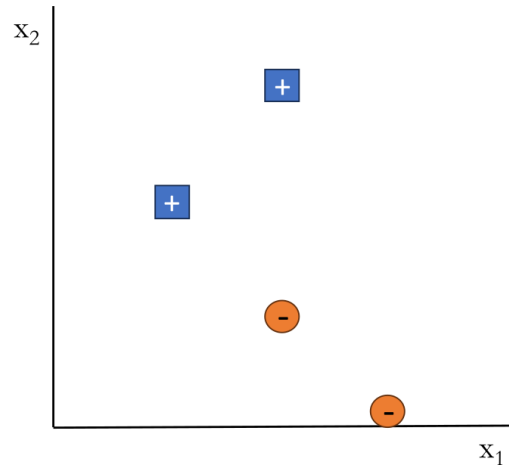


Figure 2.3

2.3 Geometric Margin

The **margin** of example \mathbf{x} w.r.t (with respect to) a **linear separator** \mathbf{w} is the distance from \mathbf{x} to the plane $\mathbf{w}^T \cdot \mathbf{x} = 0$.

The **margin** γ of a set of examples S w.r.t a **linear separator** \mathbf{w} is the largest margin over points $\mathbf{x} \in S$. Theorem: If the data has a **margin** γ and all points lie inside a ball of **radius** R , then the Perceptron algorithm makes $\leq \frac{R}{\gamma^2}$ **mistakes**.

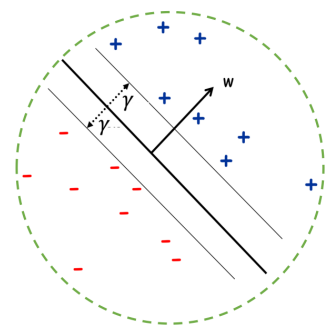


Figure 2.4

2.4 Support Vector Machine

Support vector machines (SVMs) are supervised max-margin models with associated learning algorithms.

- Good generalization in theory.
- Good generalization in practice.
- Work well with few training instances.
- Find globally best model.
- Efficient algorithms.
- Amenable to the kernel trick.

2.5 Optimal Linear Separator

Which of the linear separators is optimal?

2.6 Classification Margin

Examples closest to the hyperplane are support vectors. Margin ρ of the separator is the distance between support vectors.

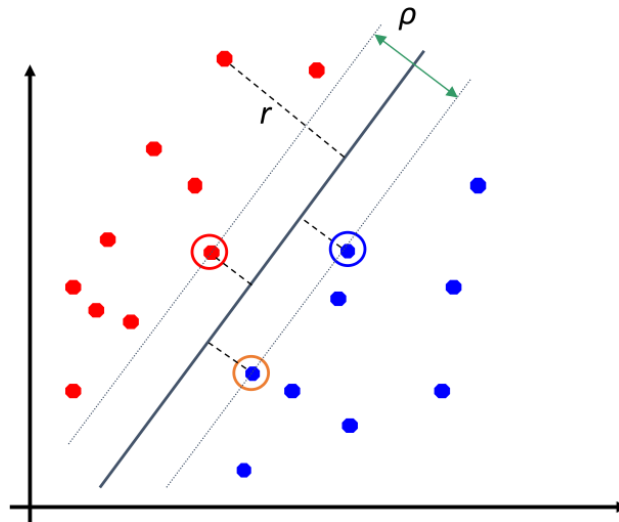


Figure 2.5

2.7 Maximizing the Margin

- Better **Generalization** – A larger margin allows the SVM to **better generalize** to new, unseen data, leading to **higher predictive accuracy**.
- Improved **Robustness** – A larger margin can lead to improved robustness **against noise and outliers** in the training data, as it allows for **greater tolerance** of misclassified examples.
- Reducing **Overfitting** – A larger margin can help **reduce overfitting** by creating a **simpler decision boundary** that is less sensitive to small fluctuations in the training data.
- Enhanced **Interpretability** – A larger margin can lead to clearer and more **interpretable decision boundaries**, making it easier to understand the model's behavior.

2.8 Linear SVM

Let training set $\{(\mathbf{x}_i, y_i)_{i=1 \dots n}, \mathbf{x}_i \in \mathbb{R}^d, y_i \in \{-1, 1\}\}$ be separated by a hyperplane with margin ρ . Then for each training example (\mathbf{x}_i, y_i)

$$\begin{aligned} \mathbf{w}^T \mathbf{x}_i + b &\geq 1 && \text{if } y_i = 1 \\ \mathbf{w}^T \mathbf{x}_i + b &\leq -1 && \text{if } y_i = -1 \end{aligned} \quad \Leftrightarrow y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$$

Geometrically, the **distance** between the **2 hyperplanes** can be expressed as:

$$\rho = \frac{2}{\|\mathbf{w}\|} \tag{2.1}$$

Then we can formulate the **quadratic optimization problem**:

Find \mathbf{w} and b such that

$$\rho = \frac{2}{\|\mathbf{w}\|}$$

is **maximized** and for all $(\mathbf{x}_i, y_i), i = 1 \dots n : y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$

Which can be reformulated as:

$\forall (\mathbf{x}_i, y_i)$, find \mathbf{w} and b such that

$$\text{Minimize } Q(\mathbf{w}) = \frac{1}{2} \|\mathbf{w}\|^2 = \frac{1}{2} \mathbf{w}^T \mathbf{w}$$

subject to $y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1, \forall i \in [1, n]$

2.9 Lagrangian Duality

- Need to **optimize** a **quadratic function** subject to **linear constraints**.
- Quadratic optimization problems are a well-known class of mathematical programming problems for which several (non-trivial) algorithms exist.
- Solution involves **constructing dual problem** where **Lagrange multipliers** α_i is associated with all inequality constraint in primal (original) problem:

$\forall i$, find $\alpha_1, \alpha_2, \dots, \alpha_n$ such that

$$\text{Minimize } Q(\alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

subject to $\alpha_i \geq 0$

Minimize $\frac{1}{2} \mathbf{w}^T \mathbf{w}$ s.t. (subject to) $1 - y_i(\mathbf{w}^T \mathbf{x}_i + b) \leq 0, \quad \forall i \in [1, n]$

The Lagrangian $\mathcal{L} = \frac{1}{2} \mathbf{w}^T \mathbf{w} + \sum_i \alpha_i (1 - y_i(\mathbf{w}^T \mathbf{x}_i + b))$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = 0$$

$$\mathbf{w} + \sum_i \alpha_i (-y_i) \mathbf{x}_i = 0$$

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$$

$$\frac{\partial \mathcal{L}}{\partial b} = 0$$

$$\sum_i \alpha_i y_i = 0$$

Substituting $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$ into \mathcal{L}

$$\begin{aligned} \mathcal{L} &= \frac{1}{2} \left(\sum_i \alpha_i y_i \mathbf{x}_i \right)^T \left(\sum_i \alpha_i y_i \mathbf{x}_i \right) + \sum_i \alpha_i \left(1 - y_i \left(\sum_j \alpha_j y_j \mathbf{x}_j^T \mathbf{x}_i + b \right) \right) \\ &= \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j + \sum_i \alpha_i - \sum_i \sum_j \alpha_i \alpha_j y_i y_j \mathbf{x}_j^T \mathbf{x}_i + b \left(\sum_i \alpha_i y_i \right) \\ &= -\frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j + \sum_i \alpha_i \end{aligned}$$

- The **new objective function** is in terms of α_i only.
- The original problem is known as the primal problem.
- The objective function of the **dual** problem needs to be **maximized**.

2.10 SVM Solution

- Given a solution $a_1 \dots a_n$ to the dual problem, solution to the primal is:

$$\mathbf{w} = \sum a_i y_i \mathbf{x} \quad b = y_k - \sum a_i y_i \mathbf{x}_i^T \mathbf{x}_k \text{ for any } a_k > 0$$

- Each non-zero a_i indicates that corresponding \mathbf{x}_i is a support vector.
- Then the classifying function is:

$$f(x) = \sum a_i y_i \mathbf{x}_i^T \mathbf{x} + b$$

- Notice that it relies on the inner product between the test point \mathbf{x} and the support vectors \mathbf{x}_i .
- Solving the optimization problem involves computing the inner products $\mathbf{x}_i^T \mathbf{x}_j$ between all training points.

2.11 Soft Margin Classification

- What if the training set is not linearly separable?
- Slack variables ξ_i can be added to allow misclassification of difficult or noisy examples; thus, the result margin is called soft.

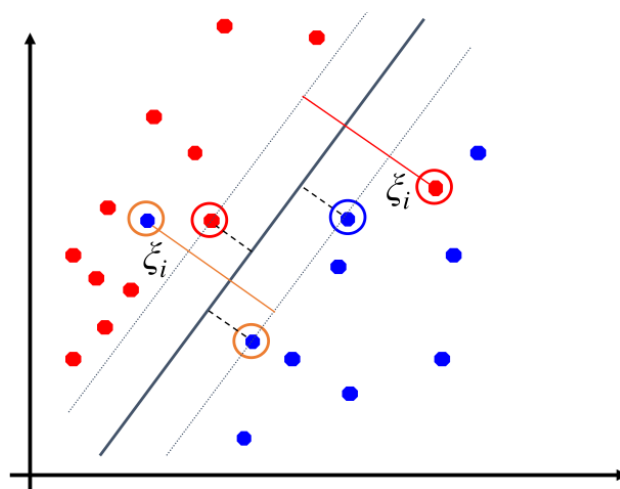


Figure 2.6

$\forall (\mathbf{x}_i, y_i)$, find \mathbf{w} and b such that

$$\text{Minimize } Q(\mathbf{w}, \xi_i) = \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^n \xi_i$$

subject to $\xi_i \geq 0$ and $y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \forall i \in [1, n]$

The **tradeoff** between the **maximization** of the **margin** and **minimization** of the **classification error** is determined by the margin **parameter** C .

2.12 Soft Margin SVM Solution

$\forall i$, find $\alpha_1 \dots \alpha_n$ such that

$$\text{Maximize } Q(a) = \sum_i \alpha_i = \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

subject to $0 \leq \alpha_i \leq C$ and $\sum_i \alpha_i y_i = 0$

The solution is:

$$\mathbf{w} = \sum \alpha_i y_i \mathbf{x}_i \quad b = y_k(1 - \xi_i) - \sum \alpha_i y_i \mathbf{x}_i^T \mathbf{x}_k \text{ for any } \alpha_k > 0$$

2.13 Kernel Method

- If we **map** the **input vectors** into a very **high-dimensional feature space**, the task of **finding** the **maximum-margin separator** can become **computationally intractable**.
- All of the **computations** that we need to do to find the **maximum-margin separator** (SVM optimization problem) can be expressed in terms of **inner products** between **pairs of data points** (in the high-dimensional feature space).
- These **inner products** are the only part of the computation that depends on the dimensionality of the high-dimensional space. So, if we had a **fast way** to do the dot products, we would **not have to pay a price** for solving the learning problem in the high-dimensional space.
- The **kernel trick** is just a way of **doing inner products** a whole lot faster than is usually possible. It relies on **choosing a way of mapping** to the **high-dimensional feature space** that allows fast scalar products.

- By using a **nonlinear vector function** $\phi(x) = \langle \phi(\mathbf{x}_1), \dots, \phi(\mathbf{x}_n) \rangle$, the n -dimensional input vector \mathbf{x} can be mapped into high-dimensional feature space. The **decision function** in the feature space is expressed as:

$$f(x) = \mathbf{w}^T \phi(\mathbf{x}) + b$$

- In terms of solving the **quadratic optimization problem of SVM**, each training data point is in the form of dot products. A **kernel function** K simplifies the calculation of dot product terms $\langle \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j) \rangle$.

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$$

Table 2.1: Kernel Functions

Kernel Name	Category	Kernel Function
Polynomial	Global	$K(\mu, v) = (\mu \cdot v + 1)^d$
Radial Basis Kernel (RBF)	Local	$K(\mu, v) = \exp(-\gamma \ \mu - v\ ^2)$

2.13.1 Local Kernels

- Only **nearby data points** affect SVM model.
- Has **higher learning** ability, but the **generalization** ability is **lower**.
- Used when there is **no prior knowledge** about the training dataset.

2.13.2 Global Kernels

- Data points** from **greater distance** can affect SVM.
- Has a **higher generalization** ability, but the **learning** ability is **lower**.

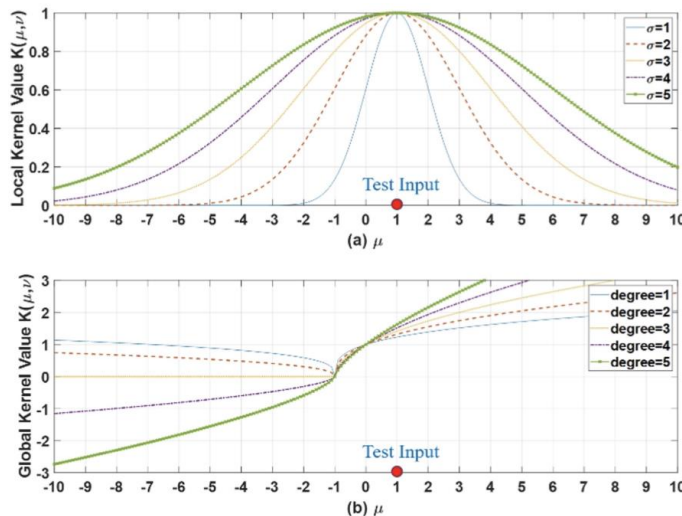


Figure 2.7

2.14 Kernel Functions

Linear:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j \quad (2.2)$$

Polynomial:

$$K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i \cdot \mathbf{x}_j)^p \quad (2.3)$$

Gaussian (Radial Basis Function):

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{\left(\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)} \quad (2.4)$$

Laplace:

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{\left(\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|}{2\sigma^2}\right)} \quad (2.5)$$

Sigmoid:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\kappa(\mathbf{x}_i \cdot \mathbf{x}_j) + \sigma) \quad (2.6)$$

2.15 Kernel Trick

- The linear classifier (2.2) relies on the **inner product** between vectors.

$$K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$$

- If **every data point** is **mapped** into **high-dimensional space** via some transformation $\phi: x \rightarrow \phi(x)$, the inner product becomes –

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$$

- A **kernel function** is a function that is equivalent to an inner product in some feature space. As long as we can **calculate the inner product in the feature space**, we do **not** need to compute each $\phi(x)$ explicitly.

Example: Take this 2-dimensional vector: $\mathbf{x} = [x_1, x_2]$

$$\begin{aligned} K(\mathbf{x}_i, \mathbf{x}_j) &= (1 + \mathbf{x}_i^T \mathbf{x}_j)^2 \\ &= 1 + x_{i1}^2 x_{j1}^2 + 2x_{i1}x_{j1}x_{i2}x_{j2} + x_{i2}^2 x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2} \\ &= \begin{bmatrix} 1 & x_{i1}^2 & \sqrt{2}x_{i1}x_{i2} & x_{i2}^2 & \sqrt{2}x_{i1} & \sqrt{2}x_{i2} \end{bmatrix}^T \begin{bmatrix} 1 & x_{j1}^2 & \sqrt{2}x_{j1}x_{j2} & x_{j2}^2 & \sqrt{2}x_{j1} & \sqrt{2}x_{j2} \end{bmatrix} \\ &= \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) \end{aligned}$$

where $\phi(\mathbf{x}) = [1 \ x_1^2 \ \sqrt{2}x_1x_2 \ x_2^2 \ \sqrt{2}x_1 \ \sqrt{2}x_2]^T$

This use of the **kernel function** to **avoid** computing $\phi(x)$ explicitly is known as the **kernel trick**.

2.16 Determining Kernels

For some **functions** $K(\mathbf{x}_i, \mathbf{x}_j)$ checking that $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$ can be cumbersome.

2.16.1 Mercer's theorem

- K is a **kernel** iff:
 - K is **symmetric** i.e., $K = K^T$.
 - K is **positive semi-definite** i.e., $c^T K c \geq 0$, where $c \in \mathbb{R}$ is a vector.

2.17 Non-linear SVMs

$\forall i$, find $\alpha_1 \dots \alpha_n$ such that

$$\text{Maximize } Q(a) = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

subject to $\alpha_i \geq 0$ and $\sum_i \alpha_i y_i = 0$.

The solution is:

$$f(\mathbf{x}) = \sum \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_j) + b \quad (2.7)$$

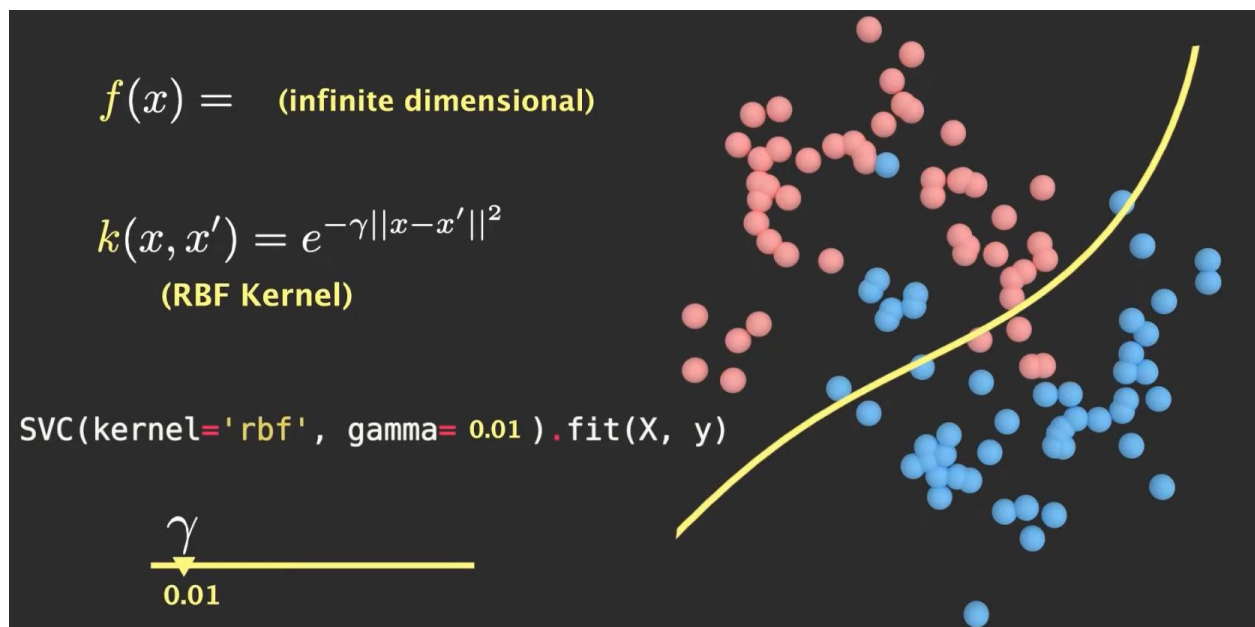


Figure 2.8

2.18 Kernel Closure Properties

- Easily **create new kernels** using basic ones.
- If $K_1(\cdot, \cdot)$ and $K_2(\cdot, \cdot)$ are kernels and $c_1 \geq 0$, $c_2 \geq 0$, then –

- $K = c_1 K_1 + c_2 K_2$ is a kernel,
- $K = K_1 K_2$ is a kernel.

2.19 Kernel Benefits

- Offers great modularity.
- No need to change the underlying learning algorithm to accommodate a particular choice of kernel function.
- Can substitute a different algorithm while maintaining the same kernel.

2.20 Conclusion

Strengths:

- Fast.
- Training is relatively easy.
- It scales relatively well high dimensional data.
- Tradeoff between classifier complexity and error can be controlled explicitly.
- Non-traditional data like strings and trees can be used as input to SVM, instead of feature vectors.

Weaknesses:

- Need to choose a “good” kernel function.