Happy Thursday!

- Quiz 10, Friday, Oct 30th 6am until Nov 1st 11:59pm (midnight)
  - SVM and Kernel SVM

Coming up soon

- Touch-point 2: deliverables due Nov 1st, live-event Mon, Nov 2nd
  - Single-slide presentation outlining progress highlights and current challenges
  - Three-minute pre-recorded presentation with your progress and current challenges
- Project midpoint report due Nov 6th 11:59pm (midnight)
  - GitHub page with the results you have achieved utilizing unsupervised learning

CS4641B Machine Learning

Lecture 20: Kernel SVM

Rodrigo Borela ▸ rborelav@gatech.edu

These slides are based on slides from Yaser Mostafa, Le Song and Eric Eaton and Mahdi Roozbahani

Outline

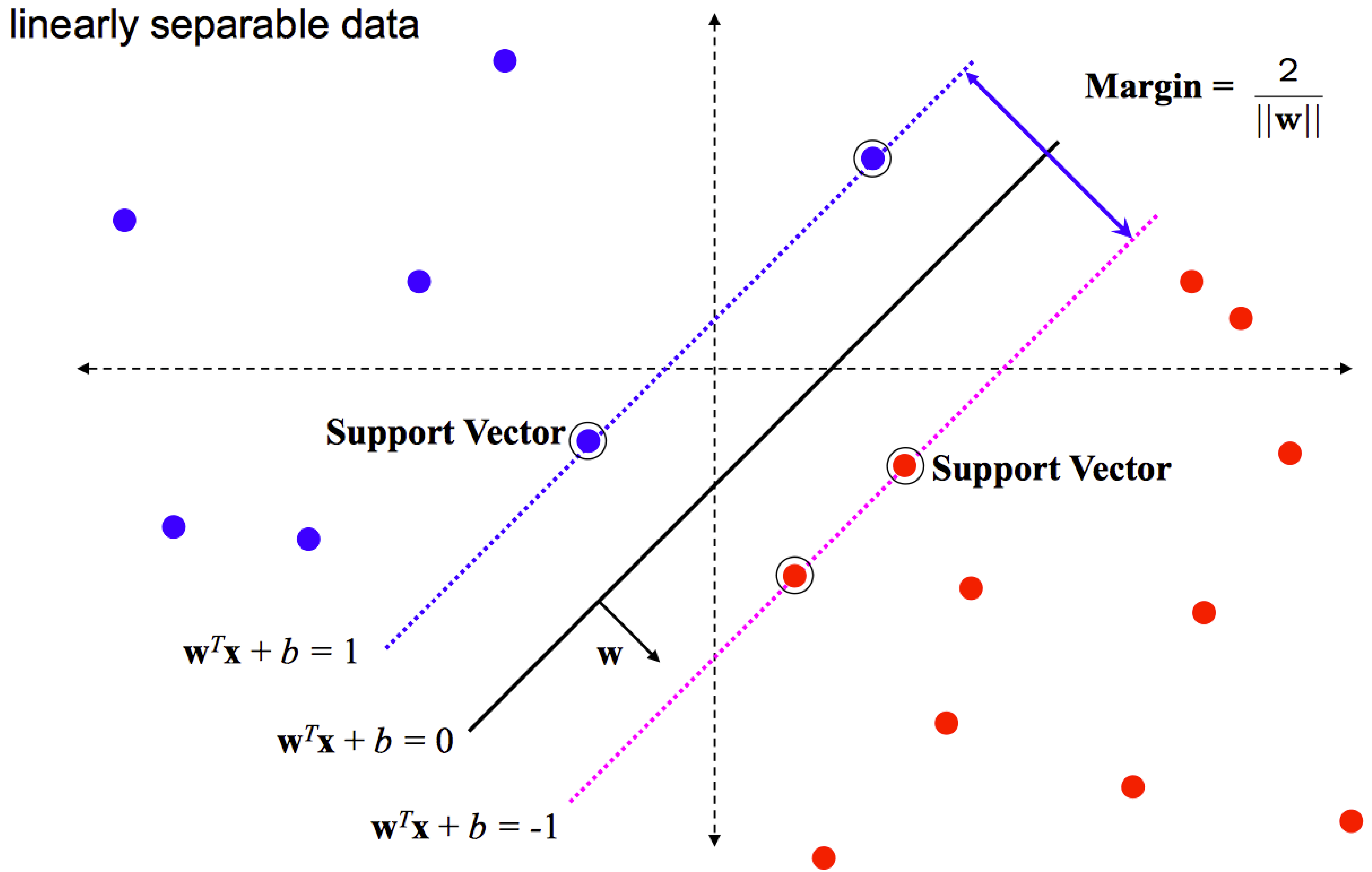
- Kernel method
- Soft SVM

- *Complementary reading: Bishop PRML – Chapter 7, Section 7.1.1 to 7.1.3*

Outline

- Kernel method
- Soft SVM

Recap: SVM


linearly separable data
Margin = $\frac{2}{||\mathbf{w}||}$
Support Vector
Support Vector
$\mathbf{w}^T\mathbf{x} + b = 1$
$\mathbf{w}$
$\mathbf{w}^T\mathbf{x} + b = 0$
$\mathbf{w}^T\mathbf{x} + b = -1$

Training

$$\mathbf{w} = \sum_{n=1}^{N} a_n t_n \mathbf{x}_n$$

Since $a_n = 0$ if $\mathbf{x}_n$ is **not** a support vector, and $a_n > 0$if it is a support vector:

$$\mathbf{w} = \sum_{x_n \in SV} a_n t_n \mathbf{x}_n$$

and for b pick any support vector and calculate: $t_n(\mathbf{w}^T\mathbf{x} + b) = 1$

Testing

For a new test point $\mathbf{x}$, compute:

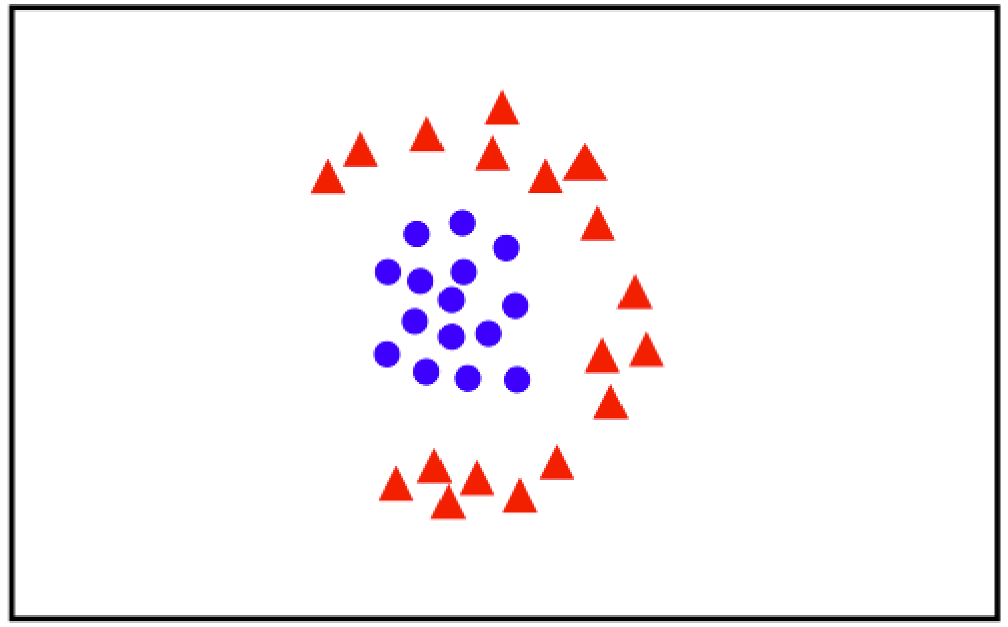
$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = \sum_{n=1}^{N} a_n t_n \mathbf{x}_n^T\mathbf{x} + b$$

Since $a_n = 0$ if $\mathbf{x}_n$ is **not** a support vector, and $a_n > 0$ if it is a support vector:

$$f(\mathbf{x}) = \sum_{x_n \in SV} a_n t_n \mathbf{x}_n^T\mathbf{x} + b$$

Classify $\mathbf{x}$ as class 1 if the result is positive, and class 2 otherwise

Handling non-linearly separable data



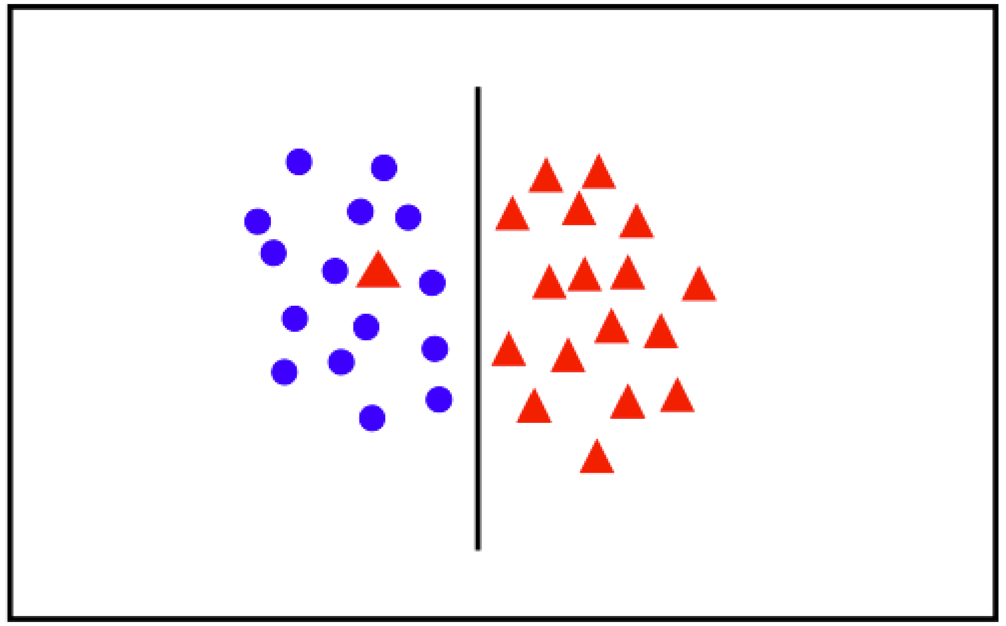
Linear classifier on original feature space

$$\tilde{\mathcal{L}}(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} t_n t_m a_n a_m k(\mathbf{x}_n, \mathbf{x}_m)$$

subject to

$$a_n \geq 0$$
$$\sum_{m=1}^{N} a_n t_n = 0$$
, for $n = 1, \ldots, N$

Kernel trick



Introduce slack variables

$$\min_{\mathbf{w},\boldsymbol{\xi}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{n=1}^{N} \xi_n$$

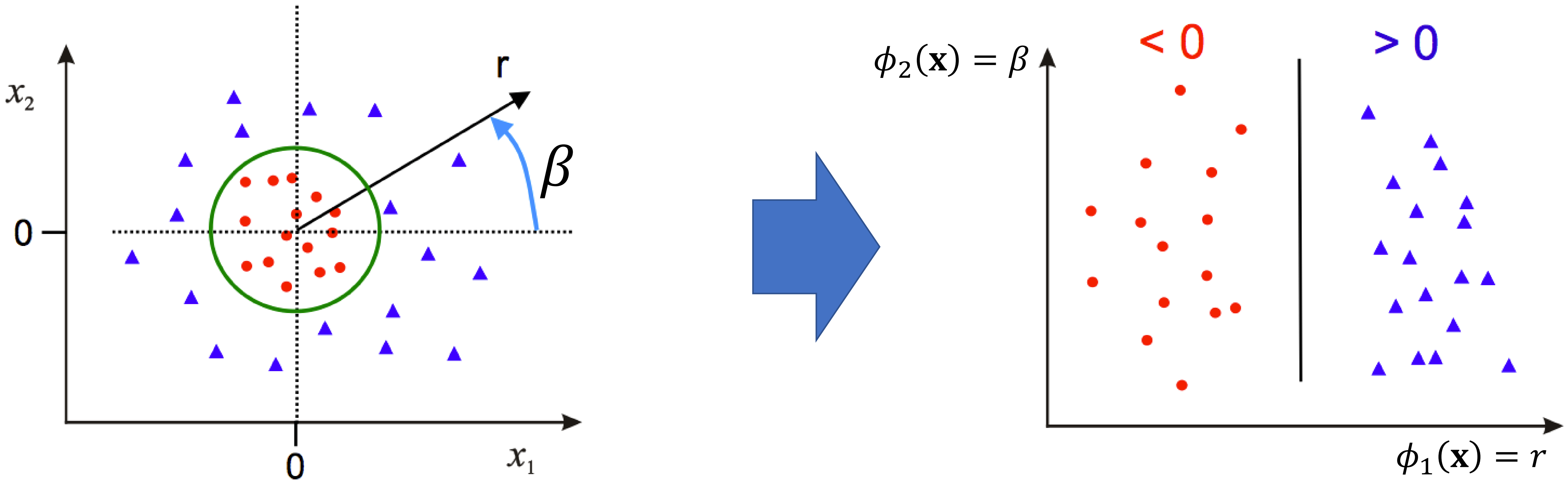
subject to

$$t_n(w^T x_n + b) \geq 1 - \xi_n$$
$$\xi_n \geq 0$$
, for $n = 1, \ldots, N$

Soft Margin SVM

(allowing ourselves to make errors)

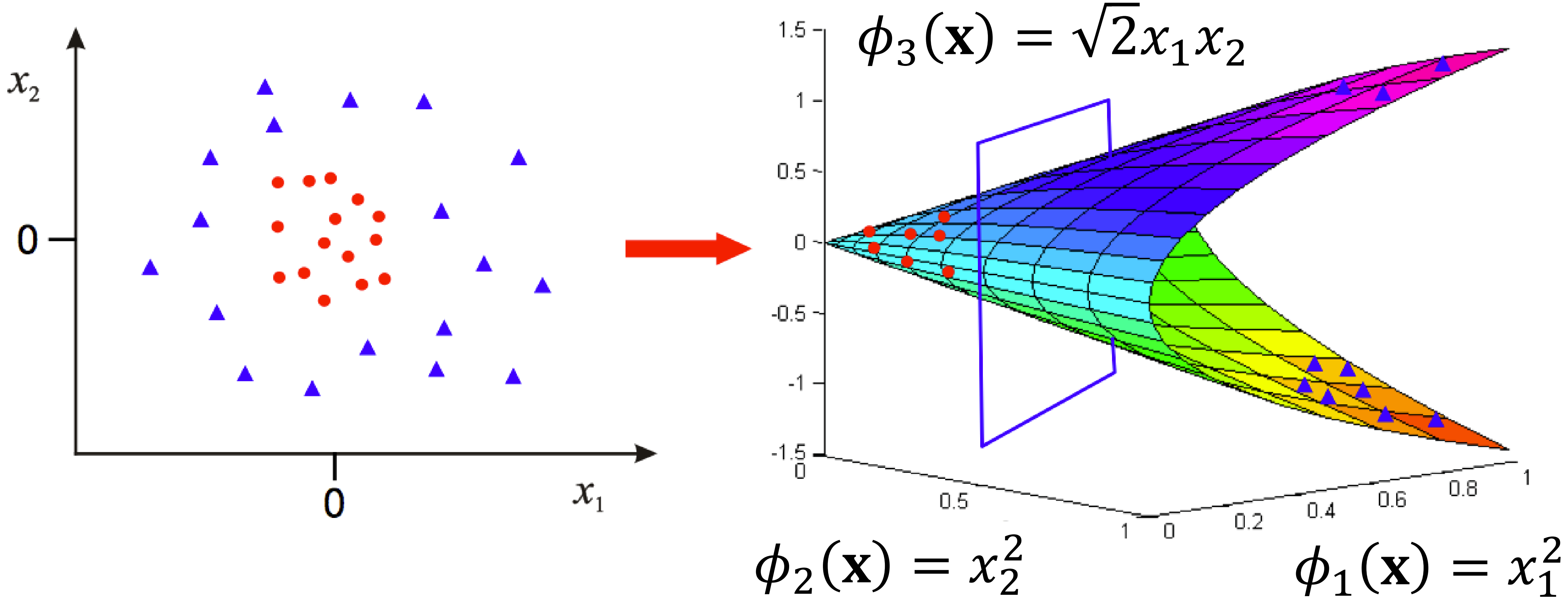
Idea 1: Use polar coordinates to go to $\boldsymbol{\phi}(\mathbf{x})$-space



- Data is linearly separable in polar coordinates
- Acts non-linearly in original space
- $\boldsymbol{\phi}: \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} r \\ \beta \end{pmatrix}, \quad \mathbb{R}^2 \rightarrow \mathbb{R}^2$

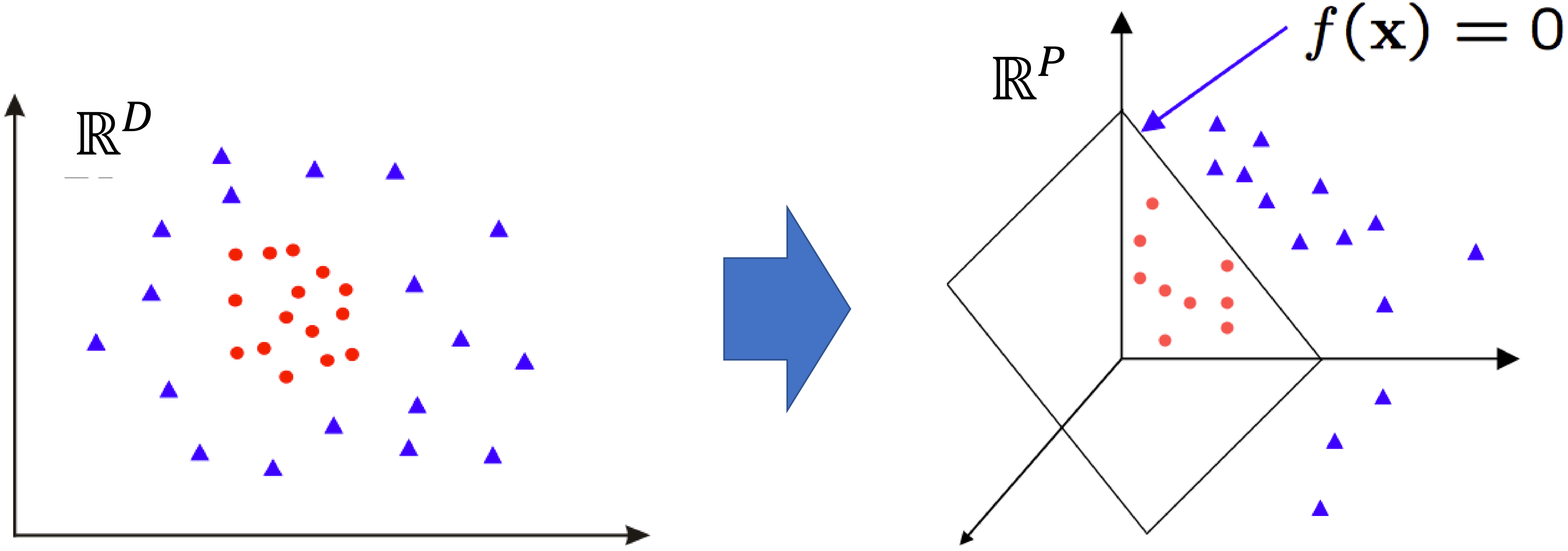
Idea 1: Map data to higher dimension $\boldsymbol{\phi}(\mathbf{x})$-space

- $\boldsymbol{\phi}: \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{pmatrix}, \quad \mathbb{R}^2 \rightarrow \mathbb{R}^3$



- Data is linearly separable in 3D
- This means that the problem can still be solved by a linear classifier

SVM in a transformed feature space



- $\boldsymbol{\phi}: \mathbf{x} \to \boldsymbol{\phi}(\mathbf{x}), \quad \mathbb{R}^D \to \mathbb{R}^P$
- Learn classifier linear in $\mathbf{w}$ for $\mathbb{R}^P$:

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

- $\boldsymbol{\phi}(\mathbf{x})$ is a basis function (or feature map)

Kernel trick – what do we need from $\boldsymbol{\phi}(\mathbf{x})$-space?

$$\max_{\mathbf{a}} \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{M} t_n t_m a_n a_m \boldsymbol{\phi}(\mathbf{x}_n)^T \boldsymbol{\phi}(\mathbf{x}_m)$$

subject to: $\begin{matrix} a_n \geq 0 \\ \sum_{n=1}^{N} a_n t_n = 0 \end{matrix}$, for $n = 1, \dots, N$

- We already have this:

$$\begin{bmatrix} t_1 t_1 \boldsymbol{\phi}(\mathbf{x}_1)^T \boldsymbol{\phi}(\mathbf{x}_1) & t_1 t_2 \boldsymbol{\phi}(\mathbf{x}_1)^T \boldsymbol{\phi}(\mathbf{x}_2) & \dots & t_1 t_N \boldsymbol{\phi}(\mathbf{x}_1)^T \boldsymbol{\phi}(\mathbf{x}_N) \\ t_2 t_1 \boldsymbol{\phi}(\mathbf{x}_2)^T \boldsymbol{\phi}(\mathbf{x}_1) & t_2 t_2 \boldsymbol{\phi}(\mathbf{x}_2)^T \boldsymbol{\phi}(\mathbf{x}_2) & \cdots & t_2 t_N \boldsymbol{\phi}(\mathbf{x}_2)^T \boldsymbol{\phi}(\mathbf{x}_N) \\ \dots & \dots & \cdots & \dots \\ t_N t_1 \boldsymbol{\phi}(\mathbf{x}_N)^T \boldsymbol{\phi}(\mathbf{x}_1) & t_N t_2 \boldsymbol{\phi}(\mathbf{x}_N)^T \boldsymbol{\phi}(\mathbf{x}_2) & \cdots & t_N t_N \boldsymbol{\phi}(\mathbf{x}_N)^T \boldsymbol{\phi}(\mathbf{x}_N) \end{bmatrix}$$

- **Same result as hard SVM:**
  - Solve a_n using quadratic programming and predict a test data point in $\boldsymbol{\phi}(\mathbf{x})$-space

$$f(\mathbf{x}) = \sum_{x_n \in SV} a_n t_n \boldsymbol{\phi}(\mathbf{x}_n)^T \boldsymbol{\phi}(\mathbf{x}) + b$$

Generalized inner product

- Given two points $\mathbf{x_1}$ and $\mathbf{x}_2$, we need $\boldsymbol{\phi}(\mathbf{x})^T\boldsymbol{\phi}(\mathbf{y})$

$$\begin{bmatrix} t_{\mathbf{x}}^2 k(\mathbf{x},\mathbf{x}) & t_{\mathbf{x}} t_{\mathbf{y}} k(\mathbf{x},\mathbf{y}) \\ t_{\mathbf{y}} t_{\mathbf{x}} k(\mathbf{y},\mathbf{x}) & t_{\mathbf{y}}^2 k(\mathbf{y},\mathbf{y}) \end{bmatrix}$$

- Let $\boldsymbol{\phi}(\mathbf{x})^T\boldsymbol{\phi}(\mathbf{y}) = k(\mathbf{x},\mathbf{y})$
- Example:
  - Consider $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \to \mathbb{R}^2$

$$k(\mathbf{x},\mathbf{y}) = (1+\mathbf{x}^T\mathbf{y})^2 = \left(1 + \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}\right)^2 = (1 + x_1 y_1 + x_2 y_2)^2$$

$$= 1 + 2x_1 y_1 + 2x_2 y_2 + x_1^2 y_1^2 + 2x_1 y_1 x_2 y_2 + x_2^2 y_2^2$$

$$= \left[1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, \sqrt{2}x_1 x_2, x_2^2\right]\left[1, \sqrt{2}y_1, \sqrt{2}y_2, y_1^2, \sqrt{2}y_1 y_2, y_2^2\right]^T = \boldsymbol{\phi}(\mathbf{x})^T\boldsymbol{\phi}(\mathbf{y})$$

Polynomial kernel

- $\mathbf{x} \in \mathbb{R}^D$ and $\boldsymbol{\phi}: \mathbb{R}^D \rightarrow \mathbb{R}^Q$ is polynomial of order Q
- The equivalent kernel $= k(\mathbf{x}, \mathbf{y}) = (1 + \mathbf{x}^T \mathbf{y})^Q = (1 + x_1 y_1 + x_2 y_2 + \cdots + x_D y_D)^Q$
- (Inhomogeneous kernel)

- Does it matter if Q is 2 or 1000?

- What will happen if we have $D = 10$ and $Q = 100$ and we want to compute the inner product explicitly?

- We need to calculate the inner product of two big huge ugly vectors

We only need $\boldsymbol{\phi}$-space to exist

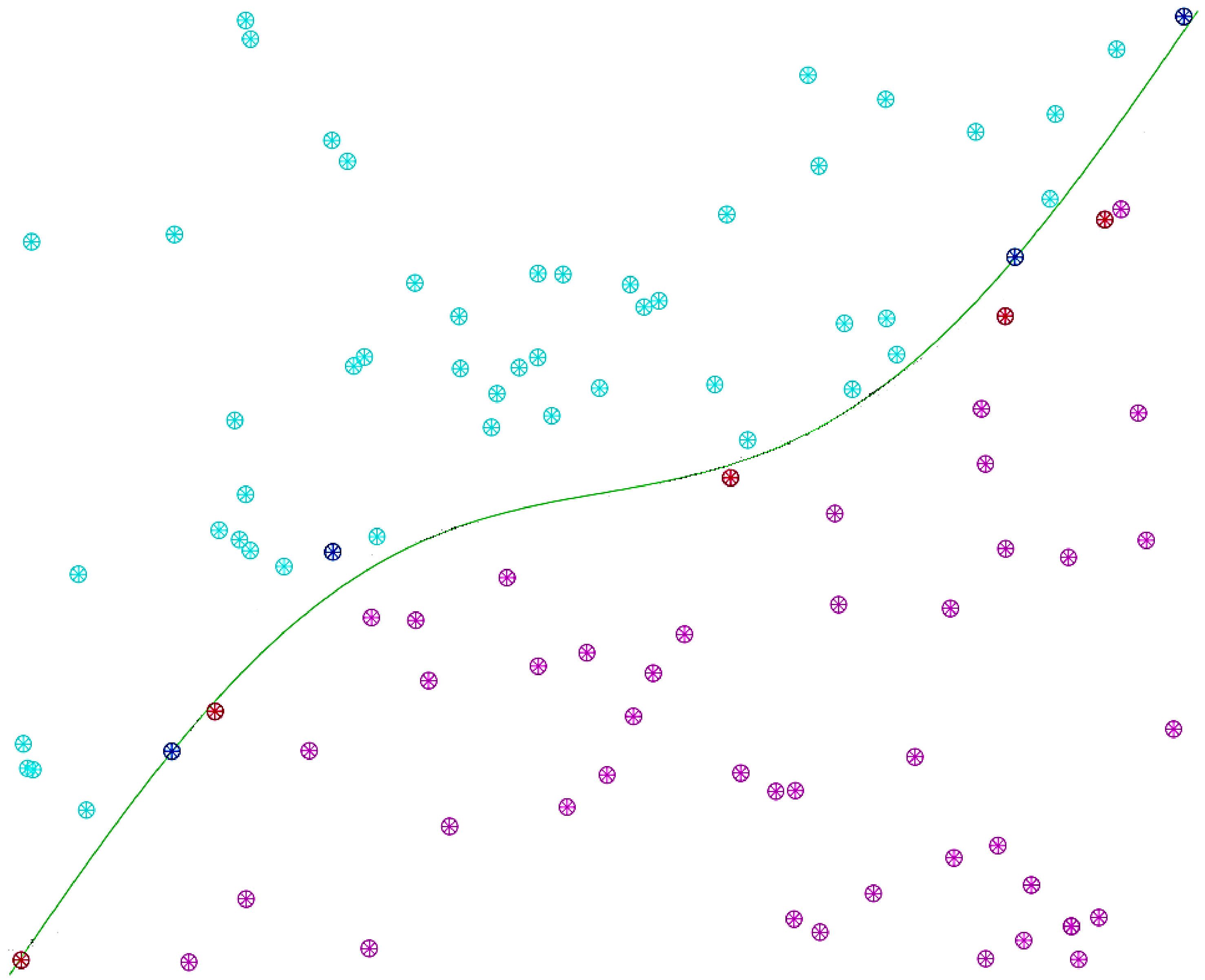
- If $k(\mathbf{x}, \mathbf{y})$is an inner product in some $\boldsymbol{\phi}$-space, we are doing good

- Example:
  - $k(\mathbf{x}, \mathbf{y}) = \exp(-\gamma\|\mathbf{x} - \mathbf{y}\|^2)$ Radial basis kernel
  - First thing first, this is a function of $\mathbf{x}$ and $\mathbf{y}$
  - This function will take us to infinite-dimensional feature space → PARTY!
  - For D and $\gamma = 1$

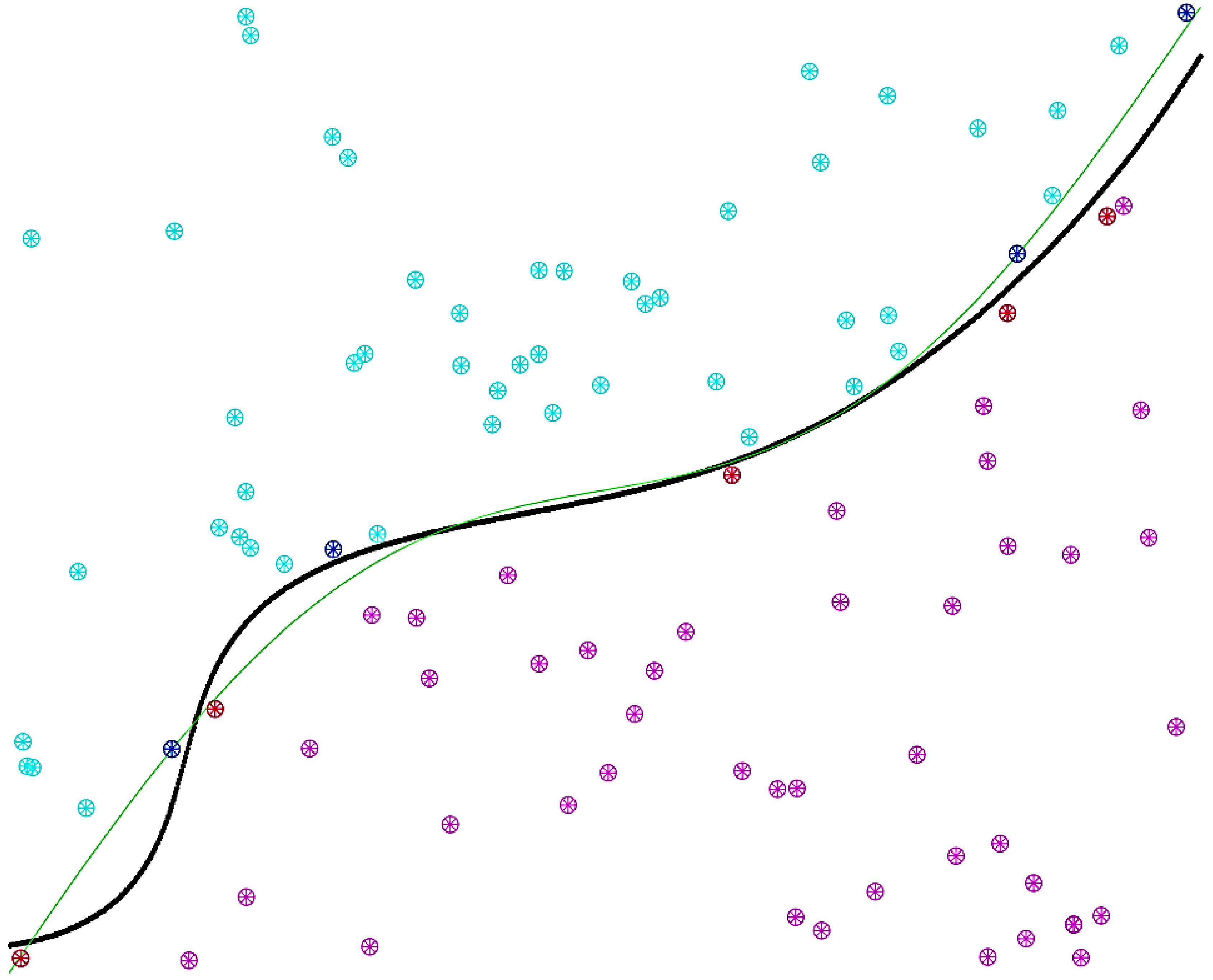
$$k(\mathbf{x}, \mathbf{y}) = \exp(-(\mathbf{x} - \mathbf{y})^2) = \exp(-\mathbf{x}^T\mathbf{x}) \exp\,(-\mathbf{y}^T\mathbf{y}) \sum_{k=0}^{\infty} \frac{2^k \mathbf{x}^k \mathbf{y}^k}{k!}$$

Radial basis kernel in action

Slightly non-linearly separable case for 100 datapoints:

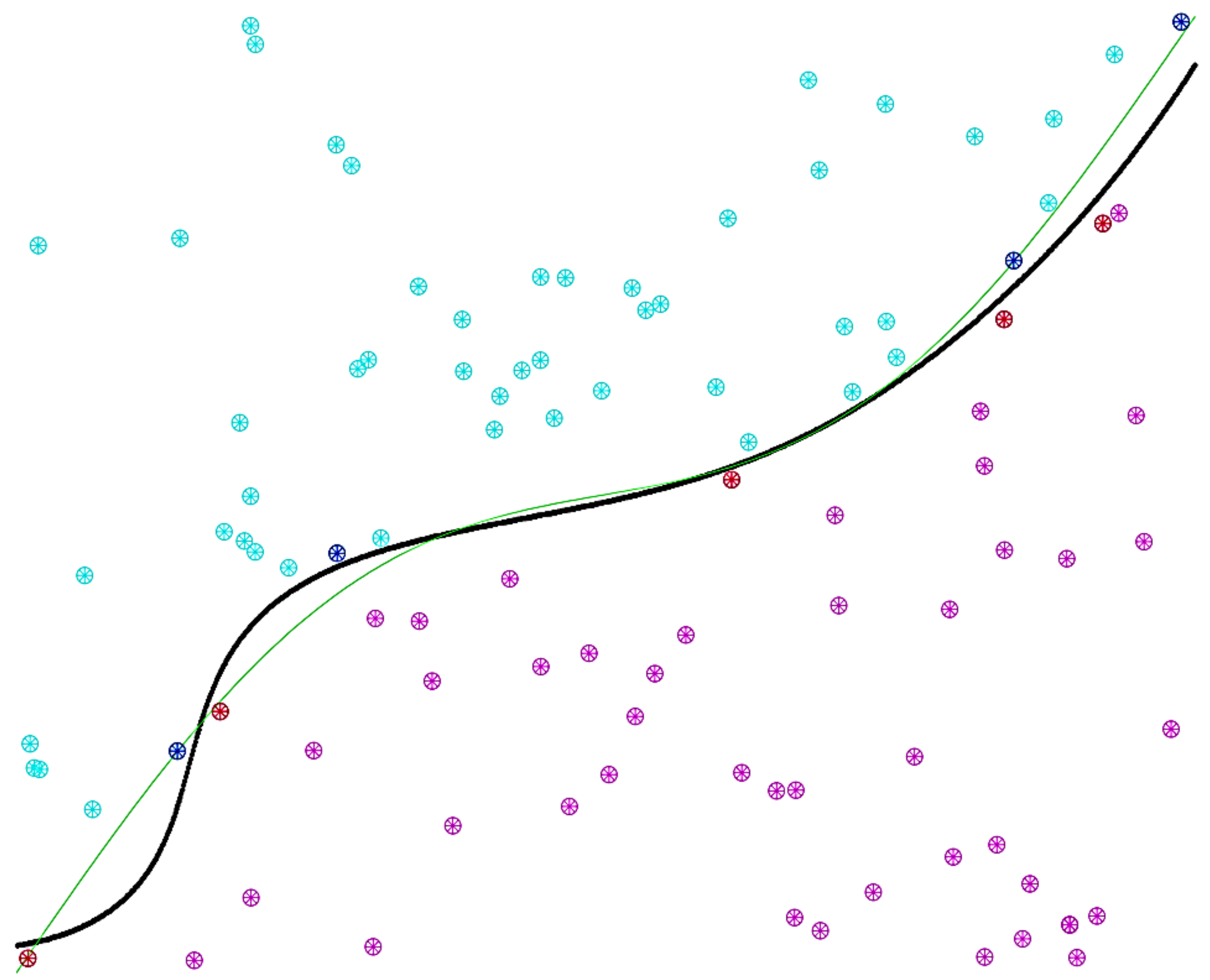


Transforming $\mathbf{x}$ into an ∞-dimensional space



Generalization

- Are we killing the generalization by going to infinite-dimension? (overfitting)
- What will happen if we have many support vectors?
- The decision boundary line (plane) will be super wiggly → overfitting alarm
- $\mathrm{E}[E_{\text{out}}] \leq \frac{\mathrm{E}[Number\ of\ support\ vectors]}{N-1}$
- N is number of datapoints



Kernel formulation of SVM

- Remember quadratic programming?

$$\underbrace{\begin{bmatrix} t_1 t_1 \mathbf{x}_1^T \mathbf{x}_1 & t_1 t_2 \mathbf{x}_1^T \mathbf{x}_2 & \dots & t_1 t_N \mathbf{x}_1^T \mathbf{x}_N \\ t_2 t_1 \mathbf{x}_2^T \mathbf{x}_1 & t_2 t_2 \mathbf{x}_2^T \mathbf{x}_2 & \cdots & t_2 t_N \mathbf{x}_2^T \mathbf{x}_N \\ \dots & \dots & \cdots & \dots \\ t_N t_1 \mathbf{x}_N^T \mathbf{x}_1 & t_N t_2 \mathbf{x}_N^T \mathbf{x}_2 & \cdots & t_N t_N \mathbf{x}_N^T \mathbf{x}_N \end{bmatrix}}_{\text{Quadratic coefficients}}$$

- In $\boldsymbol{\phi}(\mathbf{x})$-space, the only thing you need:

$$\begin{bmatrix} t_1 t_1 k(\mathbf{x}_1, \mathbf{x}_1) & t_1 t_2 k(\mathbf{x}_1, \mathbf{x}_2) & \dots & t_1 t_N k(\mathbf{x}_1, \mathbf{x}_N) \\ t_2 t_1 k(\mathbf{x}_2, \mathbf{x}_1) & t_2 t_2 k(\mathbf{x}_2, \mathbf{x}_2) & \cdots & t_2 t_N k(\mathbf{x}_2, \mathbf{x}_N) \\ \dots & \dots & \cdots & \dots \\ t_N t_1 k(\mathbf{x}_N, \mathbf{x}_1) & t_N t_2 k(\mathbf{x}_N, \mathbf{x}_2) & \cdots & t_N t_N k(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix}$$

Final stage

$f(\mathbf{x}) = \text{sign}(\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}) + b)$, with $\mathbf{w} = \sum_{\mathbf{x}_n \in SV} a_n t_n \phi(\mathbf{x}_\text{n})$

Equivalent to:

$$f(\mathbf{x}) = \text{sign}\left(\sum_{\mathbf{x}_n \in SV} a_n t_n \phi(\mathbf{x}_n)^T \phi(\mathbf{x}) + b \right)$$

In terms of $k(-,-)$

$$f(\mathbf{x}) = \text{sign}\left(\sum_{\mathbf{x}_n \in SV} a_n t_n k(\mathbf{x}_n, \mathbf{x}) + b \right)$$

$$b = t_i - \sum_{\mathbf{x}_i, \mathbf{x}_j \in SV} a_j t_j k(\mathbf{x}_i, \mathbf{x}_j)$$

How do we know that the kernel is valid?

- For a given $k(\mathbf{x}, \mathbf{y}) \rightarrow$ We can check the validity
- Three approaches:
    1. By construction (Polynomial one)
    2. Math properties (Mercer's condition)
    3. Who cares? ☺

Design your kernel

- $k(\mathbf{x}, \mathbf{y})$ is valid $\mathbf{iff}$

  1. It is symmetric → $k(\mathbf{x}, \mathbf{y}) = k(\mathbf{y}, \mathbf{x})$

  2. The matrix: $\begin{bmatrix} t_1 t_1 k(\mathbf{x}_1, \mathbf{x}_1) & t_1 t_2 k(\mathbf{x}_1, \mathbf{x}_2) & \dots & t_1 t_N k(\mathbf{x}_1, \mathbf{x}_N) \\ t_2 t_1 k(\mathbf{x}_2, \mathbf{x}_1) & t_2 t_2 k(\mathbf{x}_2, \mathbf{x}_2) & \cdots & t_2 t_N k(\mathbf{x}_2, \mathbf{x}_N) \\ \dots & \dots & \cdots & \dots \\ t_N t_1 k(\mathbf{x}_N, \mathbf{x}_1) & t_N t_2 k(\mathbf{x}_N, \mathbf{x}_2) & \cdots & t_N t_N k(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix}$

  is <u>positive-semi definite</u>, for any $\mathbf{x}_1, \dots, \mathbf{x}_N$

  (Mercer's condition)

Common kernels

- **Linear** kernels $k(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T \mathbf{y}$

- **Polynomial** kernels $k(\mathbf{x}, \mathbf{y}) = (1 + \mathbf{x}^T \mathbf{y})^Q$ for any $Q > 0$
  - Contains all polynomial terms up to degree Q

- **Gaussian** kernels $k(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|_2^2}{2\sigma^2}\right)$ for $\sigma > 0$
  - Infinite dimensional features space

Generalized inner product

- Primal version of classifier

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

- Dual version of classifier

$$f(\mathbf{x}_{\text{test}}) = \sum_{x_n \in SV} a_n t_n \boldsymbol{\phi}(\mathbf{x}_n)^T \boldsymbol{\phi}(\mathbf{x}_{\text{test}}) + b$$

Kernel SVM: summary

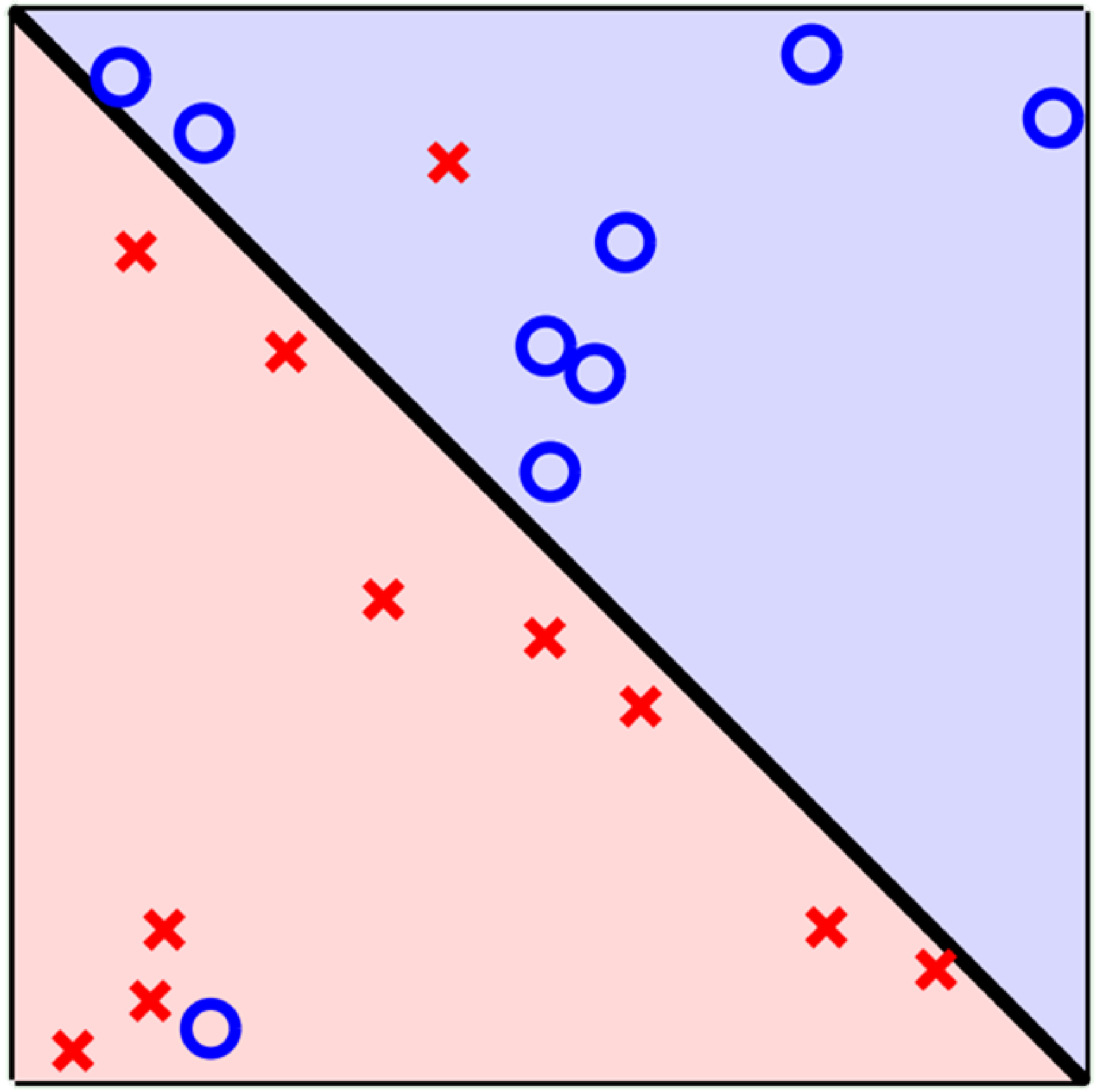
- Classifiers can be learnt for high dimensional feature spaces, without actually having to map the points into the high dimensional space
- Data may be linearly separable in the high dimensional space, but not linearly separable in the original feature space
- Kernels can be used for an SVM because of the scalar product in the dual form, but can also be used elsewhere – they are not tied to the SVM formalism

Outline

- Kernel method
- Soft SVM

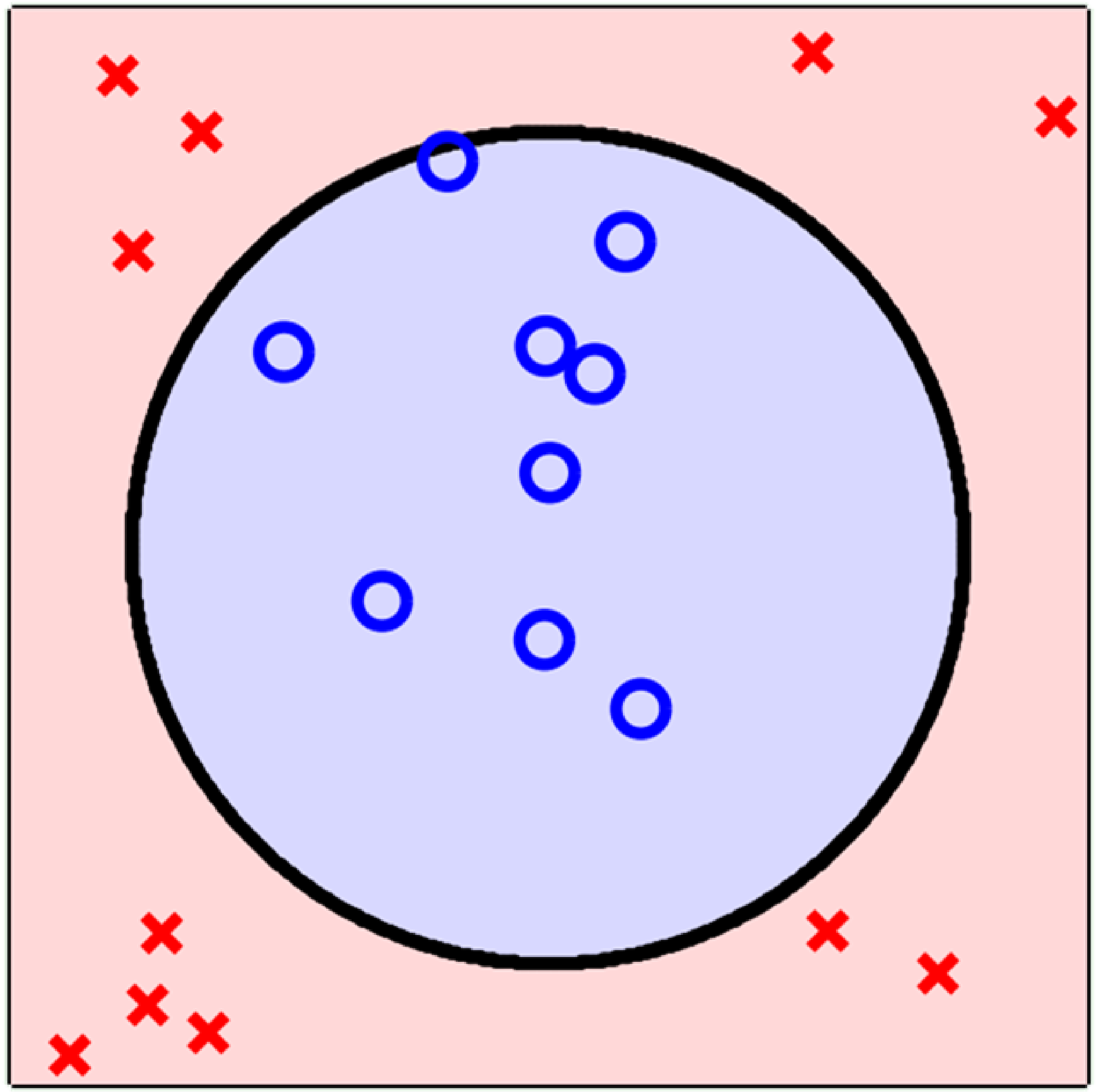
Soft SVM – two types of non-separability

slight



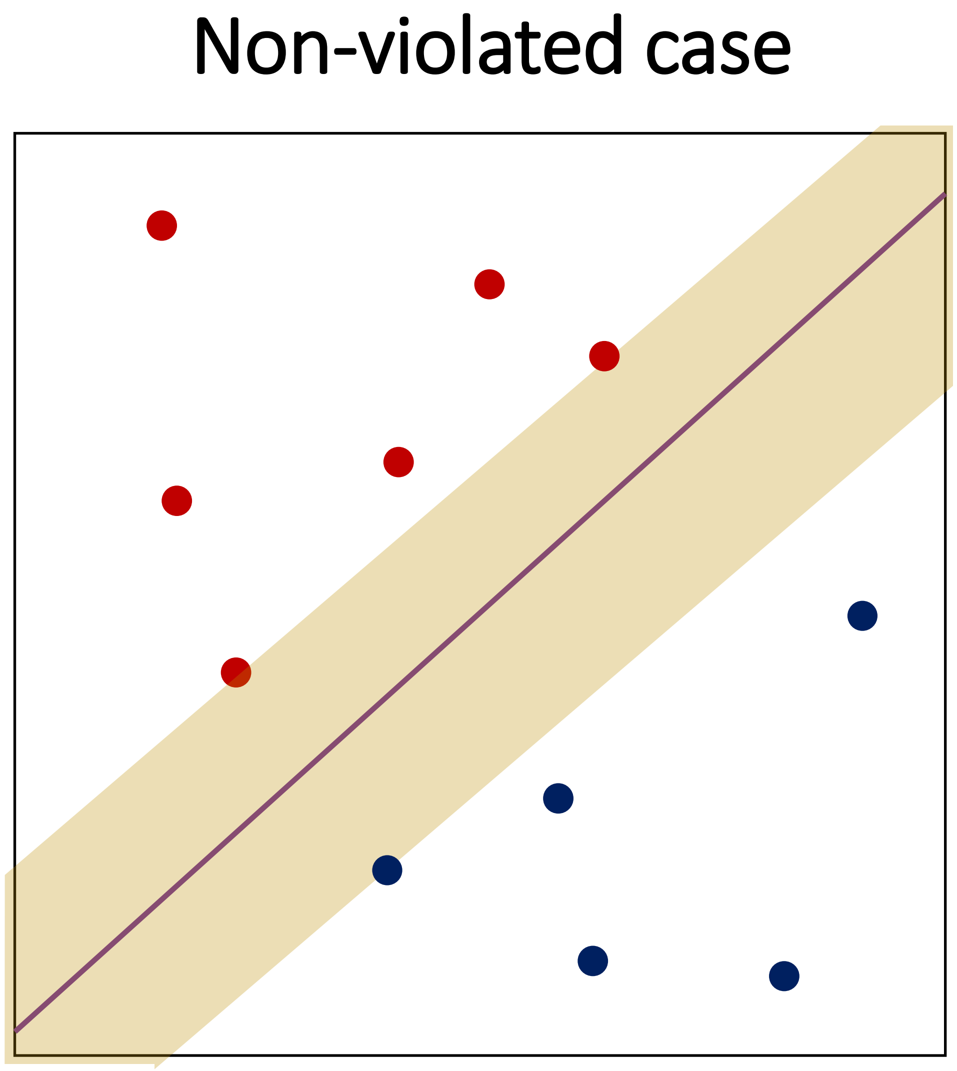
Soft SVM will deal with this

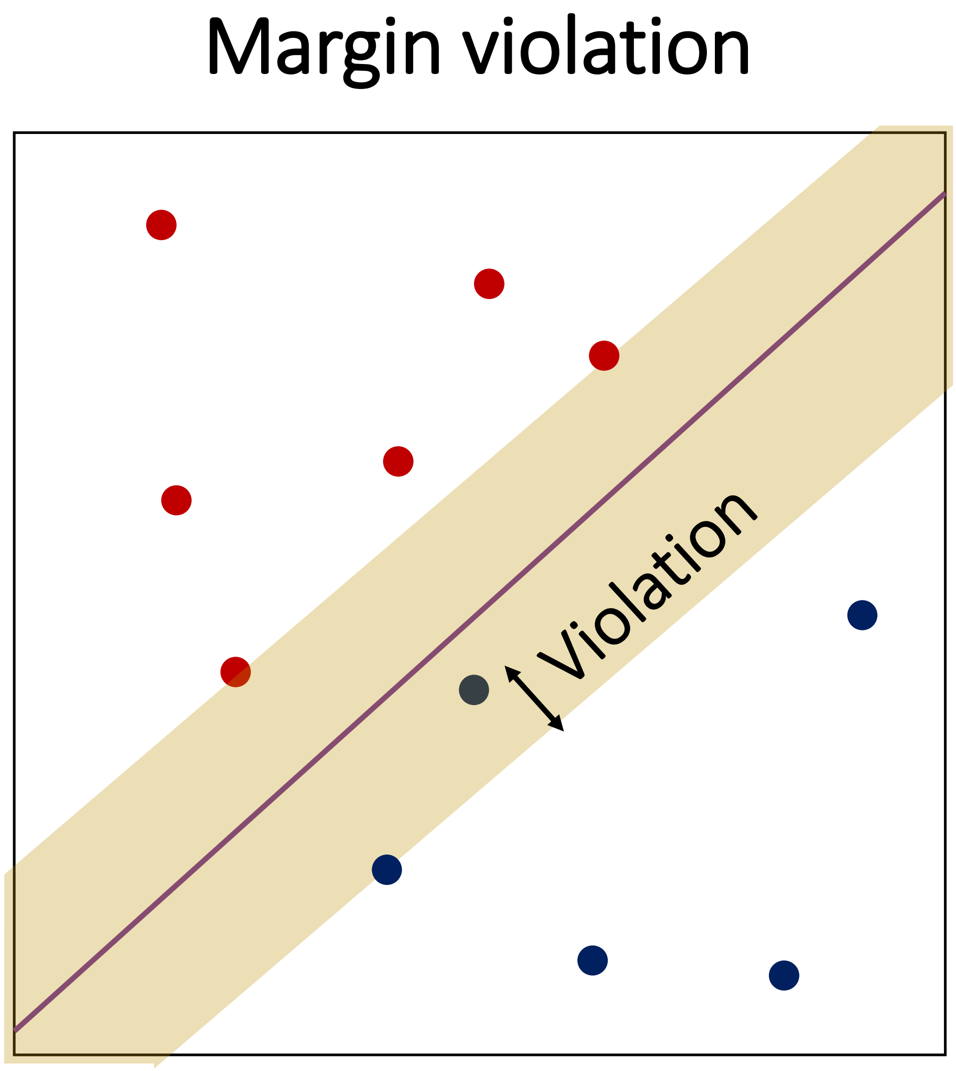
serious



Kernel will deal with this

Error measure





- if $t_n(\mathbf{w}^T\mathbf{x}_n + b) > 1 \rightarrow$ Non SV
- Let's introduce a slack variable: $t_n(\mathbf{w}^T\mathbf{x}_n + b) \geq 1 - \xi_n, \quad \xi_n \geq 0$
- Total violation = $\sum_{n=1}^{N} \xi_n$

The new optimization

$$\min_{\mathbf{w},\boldsymbol{\xi}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{n=1}^{N} \xi_n$$

subject to

$$t_n(w^T x_n + b) \geq 1 - \xi_n, \text{ for } n = 1, \dots, N$$
$$\xi_n \geq 0$$

Lagrange formulation

- Hard SVM:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2$$

$$s.t. \quad t_n(\mathbf{w}^T\mathbf{x}_n + b) \geq 1$$

$$\mathcal{L}(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{n=1}^{N} a_n\{t_n(\mathbf{w}^T\mathbf{x}_n + b) - 1\}$$

- Soft SVM:

$$\min_{\mathbf{w},\boldsymbol{\xi}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{n=1}^{N} \xi_n$$

$$s.t. \quad t_n(\mathbf{w}^T\mathbf{x}_n + b) \geq 1 - \xi_n \text{ and } \xi_n \geq 0$$

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\xi}, \mathbf{a}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{n=1}^{N} \xi_n - \sum_{n=1}^{N} a_n\{t_n(\mathbf{w}^T\mathbf{x}_n + b) - 1 + \xi_n\} - \sum_{n=1}^{N} \beta_n\xi_n$$

Lagrange formulation

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\xi}, \mathbf{a}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{n=1}^{N}\xi_n - \sum_{n=1}^{N} a_n\{t_n(\mathbf{w}^T\mathbf{x}_n + b) - 1 + \xi_n\} - \sum_{n=1}^{N}\beta_n\xi_n$$

- Minimize w.r.t $\mathbf{w}, b, and\ \boldsymbol{\xi}$ and maximize w.r.t $a_n \geq 0\ and\ \beta_n \geq 0$
- Let's do the minimization:

$$\nabla_{\mathbf{w}}\,\mathcal{L}(\mathbf{w}, b, \boldsymbol{\xi}, \mathbf{a}) = \mathbf{w} - \sum_{n=1}^{N} a_n t_n \mathbf{x}_n = 0$$

$$\nabla_{b}\,\mathcal{L}(\mathbf{w}, b, \boldsymbol{\xi}, \mathbf{a}) = -\sum_{n=1}^{N} a_n t_n = 0$$

$$\nabla_{\boldsymbol{\xi}}\,\mathcal{L}(\mathbf{w}, b, \boldsymbol{\xi}, \mathbf{a}) = C - a_n - \beta_n$$

- If we substitute β_n up there, the whole formulation will get back to hard SVM

Solution

- $\beta_n = C - a_n$
- $\beta_n \geq 0 \rightarrow C - a_n \geq 0 \rightarrow 0 \leq a_n \leq C$, for $n = 1, \ldots, N$

$$\max_{\mathbf{a}} \mathcal{L}(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} t_n t_m a_n a_m \mathbf{x}_n^T \mathbf{x}_m$$

subject to

$$\begin{matrix} 0 \leq a_n \leq C \\ \sum_{m=1}^{N} a_n t_n = 0 \end{matrix} \text{, for } n = 1, \ldots, N$$

- Minimize: $\frac{1}{2} \mathbf{w}^T \mathbf{w} + \mathrm{C} \sum_{n=1}^{N} \xi_n \rightarrow \mathbf{w} = \sum_{n=1}^{N} a_n t_n \mathbf{x}_n$

Types of support vectors

- We call the three points as **margin** support vectors

$$0 < a_n < C$$

$$\beta_n = C - a_n$$

$$t_n(\mathbf{w}^T\mathbf{x}_n + b) = 1 \rightarrow \beta_n > 0 \rightarrow \xi_n = 0$$

(KKT condition)

- **Non-margin** support vectors ($a_n = C$)

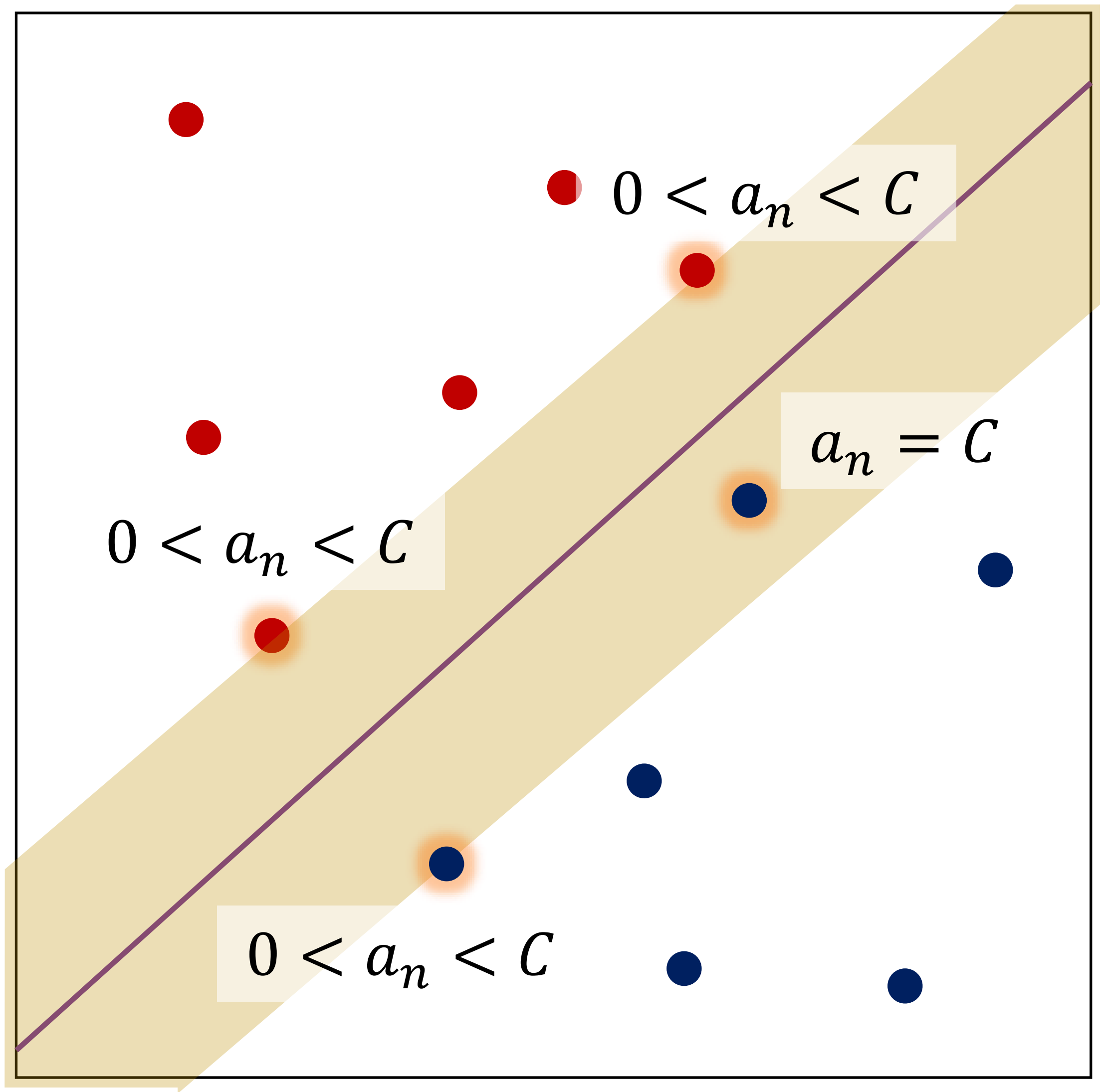
$$\beta_n = 0 \rightarrow \xi_n > 0$$

(KKT condition)

$$t_n(\mathbf{w}^T\mathbf{x}_n + b) > 1 - \xi_n \quad \text{if} \quad \xi_n > 0$$

$$t_n(\mathbf{w}^T\mathbf{x}_n + b) < 1$$

- Any violating points become support vectors



$$a_n = 0 \rightarrow t_n(\mathbf{w}^T\mathbf{x}_n + b) > 1$$

Non SV

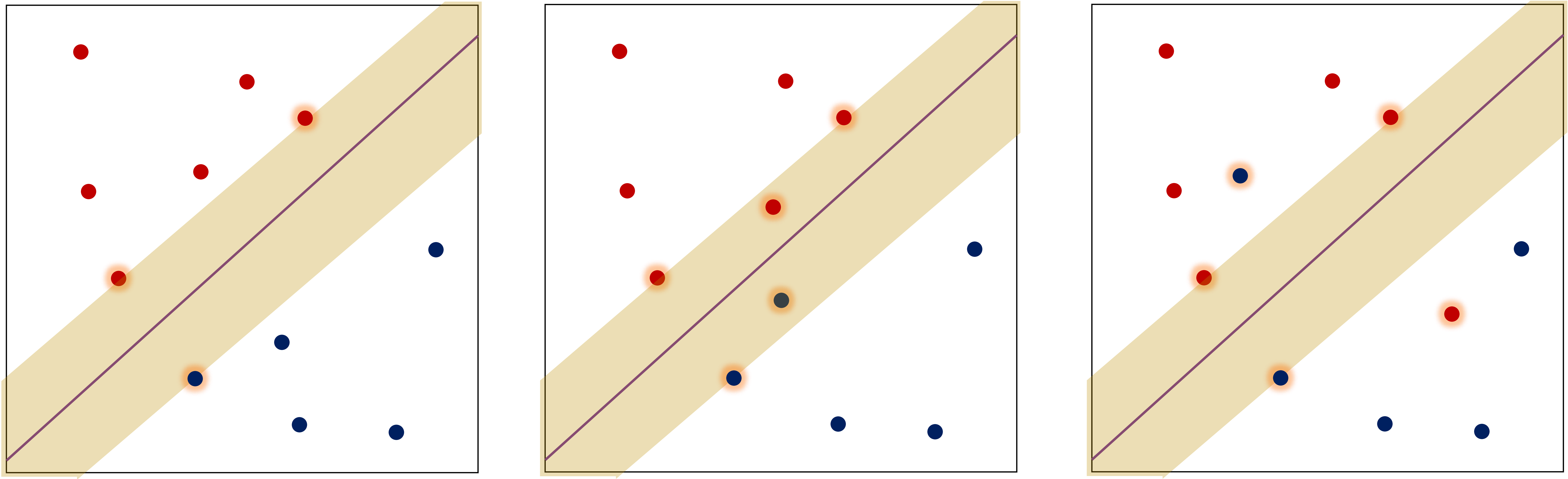
$$a_n = C \rightarrow t_n(\mathbf{w}^T\mathbf{x}_n + b) < 1$$

SV on the wrong side

$$0 < a_n < C \rightarrow t_n(\mathbf{w}^T\mathbf{x}_n + b) = 1$$

SV on the margin

How to choose C?



violating points become support vectors

How to define the hyper-parameter C → cross-validation