The week ahead

- **Quiz 9:** mean is 84% and average completion time 5min 14sec
- **Assignment 3 due 11:59pm (midnight)** → 1 extra point (final grade) to everyone
  - Extra office hour (5- 6 pm) offered by Rodrigo tonight
- **Assignment 4** → releasing tonight
- **Quiz 10, Friday, Oct 30th 6am until Oct 31st 11:59am (noon)**
  - SVM and the kernel method

Coming up soon

- **Touch-point 2**: deliverables due Fri, Oct 30th, live-event Wed, 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 19: SVM

Rodrigo Borela ▸ rborelav@gatech.edu

These slides are based on slides from Andrew Zisserman and Mahdi Roozbahani

Outline

- Precursor: Linear classifier and perceptron
- Support vector machine
- Parameter learning

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

Outline

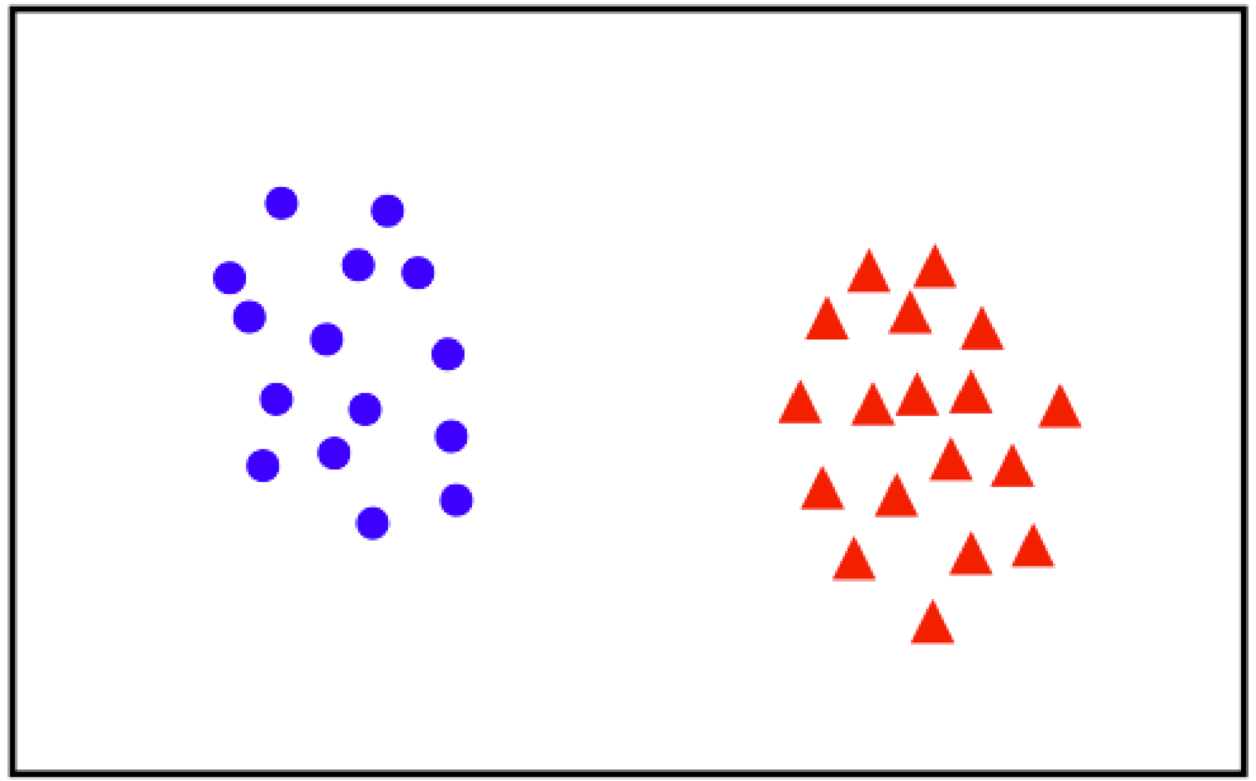
- Precursor: Linear classifier and perceptron
- Support vector machine
- Parameter learning

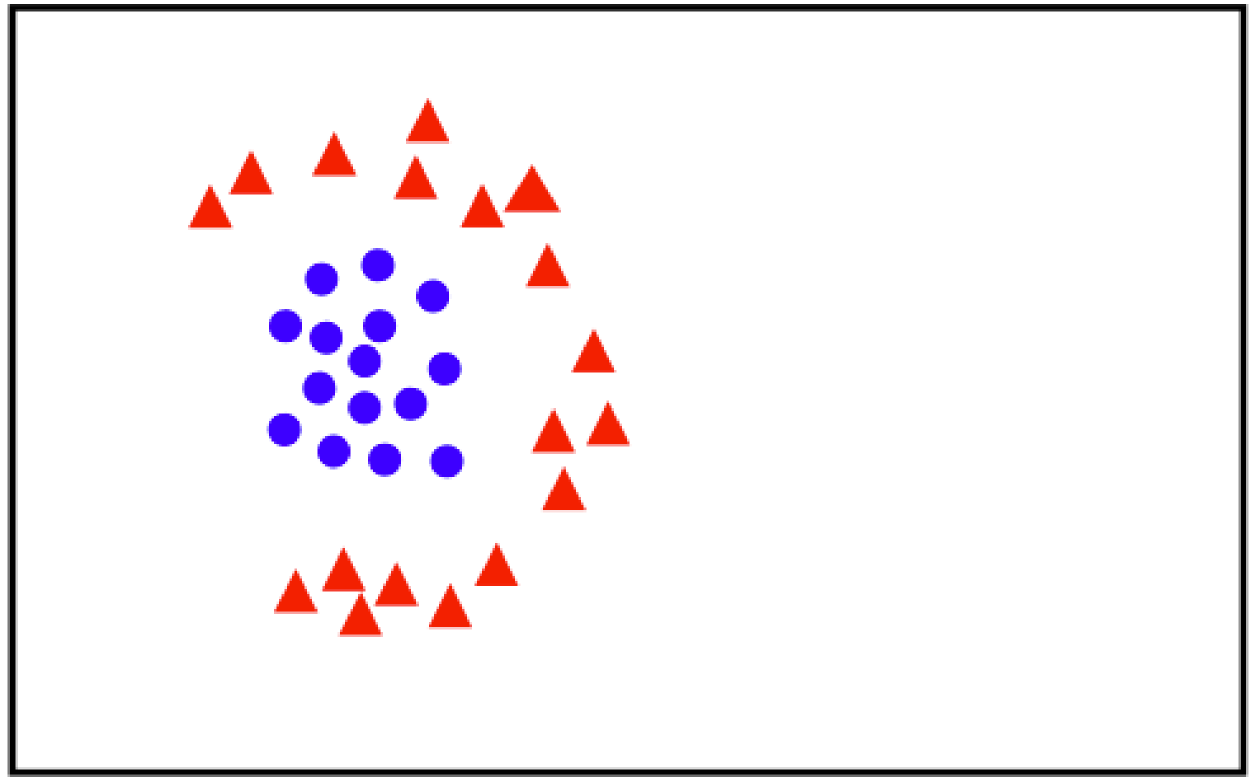
Binary Classification

- Given training data $(\mathbf{x}_n, t_n)$ for $n = 1, \ldots, N$, with $\mathbf{x}_n \in \mathbb{R}^D$ and $t_n \in \{-1, +1\}$, learn a classifier $f(\mathbf{x})$ such that

$$f(\mathbf{x}) = \begin{cases} \geq 0, t_n = +1 \\ < 0, t_n = -1 \end{cases}$$

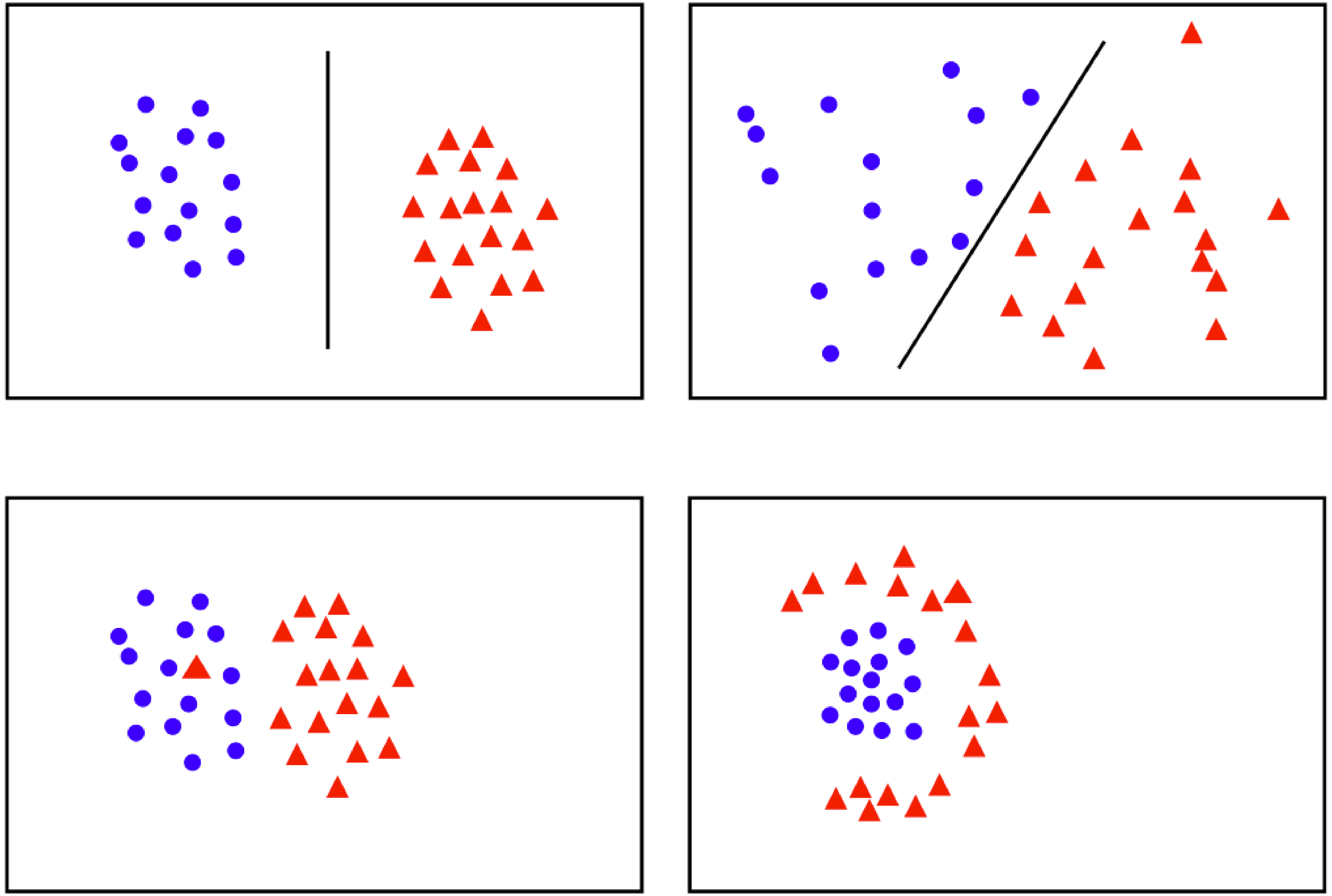
For a correct classification we should have $t_n f(x_n) > 0$.





Linear separability

linearly
separable



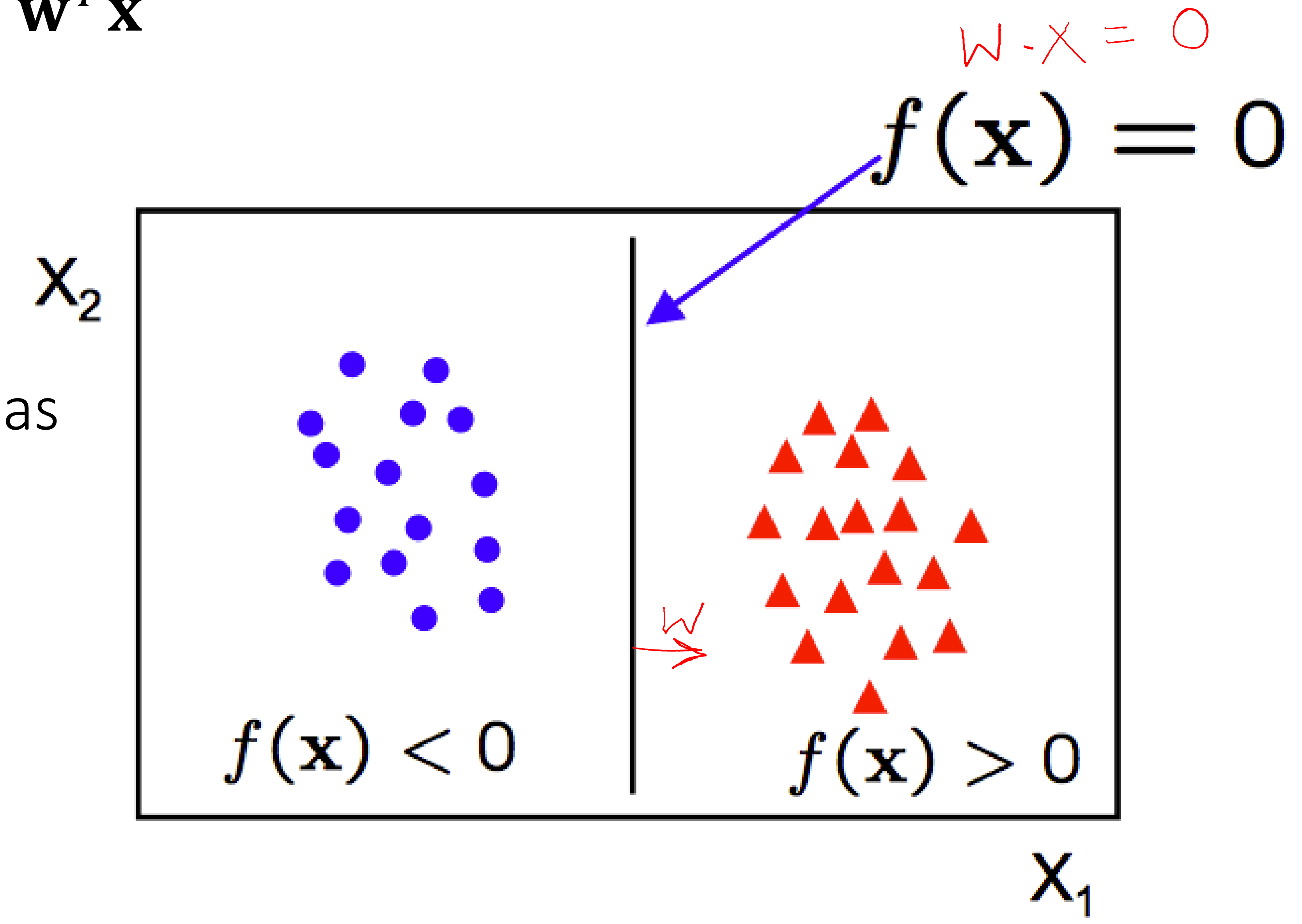
not
linearly
separable

Linear classifier

- A linear classifier has the form:

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

- In 2D the discriminant is a line
- $\mathbf{w}$ is known as the weight vector, and b the bias
- $\mathbf{w}$ is the normal to the line,



$$W = [b, w_1, w_2 \cdots\cdots w_d]_{d+1 \times 1}$$

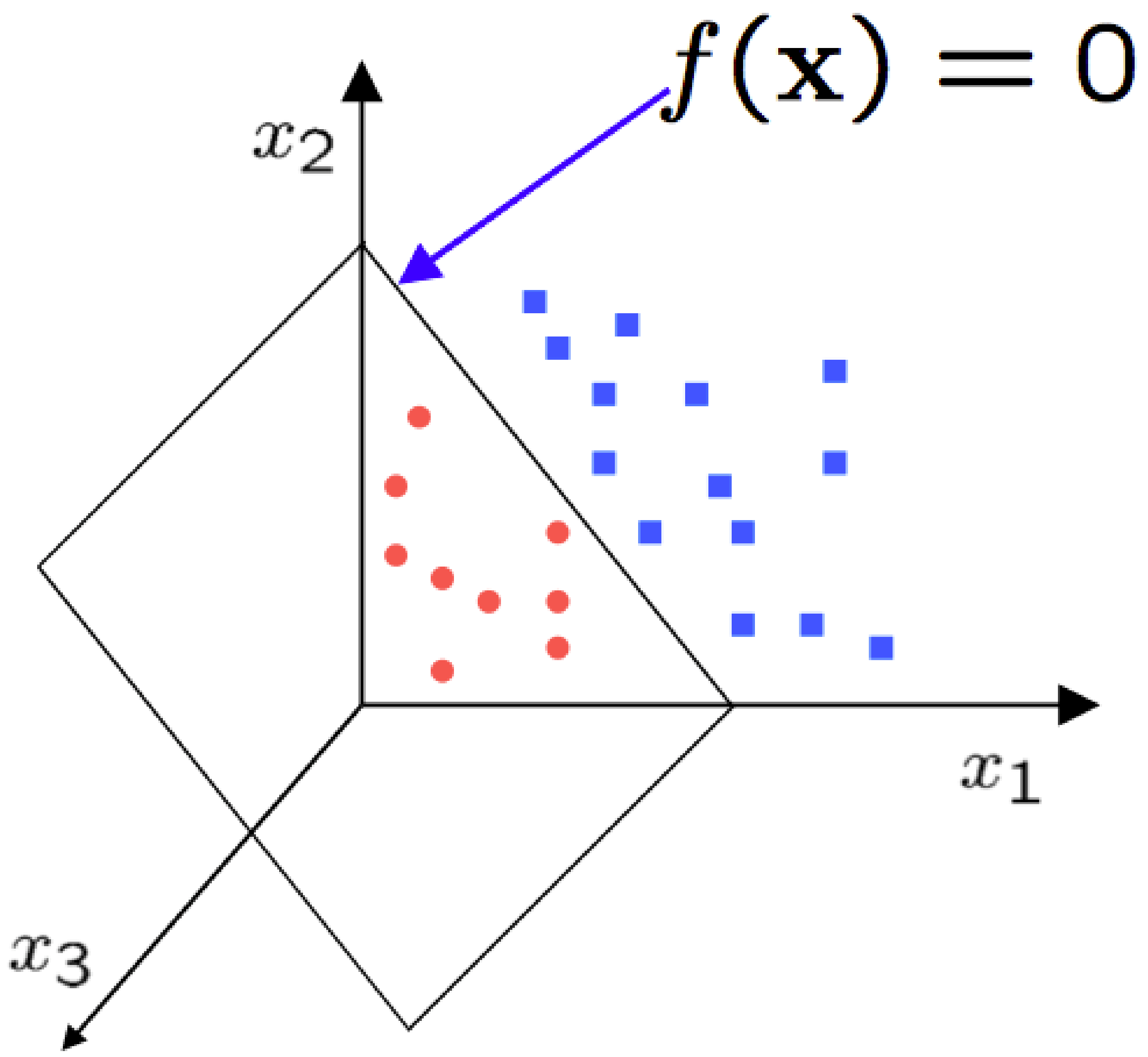
$$X = [1, x_1, x_2 \cdots\cdots x_d]_{N \times d+1}$$

Linear classifier (higher dimension)

- A linear classifier has the form:

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

- In 3D the discriminant is a plane and in D-dimensions it is a hyperplane



The perceptron classifier

- Considering $\mathbf{X}$ is linearly separable and $\mathbf{t}$ has two labels of {−1,1}

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

- How can we separate datapoints with label 1 from datapoints with label -1 using a line?
- Perceptron Algorithm:
  - Initialize $\mathbf{w} = 0$ (includes bias term, b)
  - Go through each datapoint $(\mathbf{x}_n, t_n)$
  - If $\mathbf{x}_n$ is misclassified, then $\mathbf{w}_{\tau+1} \leftarrow \mathbf{w}_\tau + \eta t_n \mathbf{x}_n$
  - Until all datapoints are correctly classified

Misclassified

ex. $t_n f(\mathbf{x}_n) < 0$

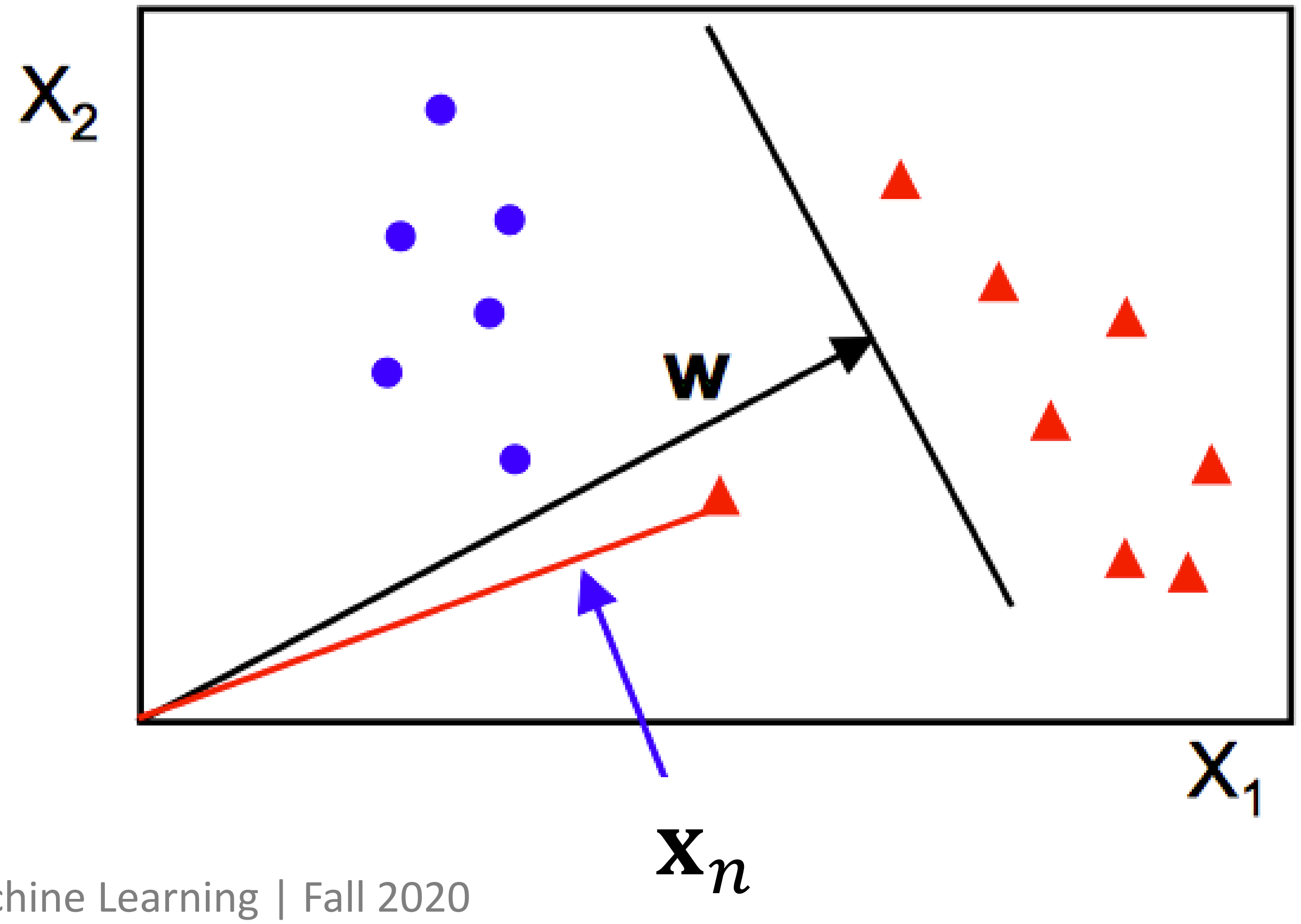
actual

predicted

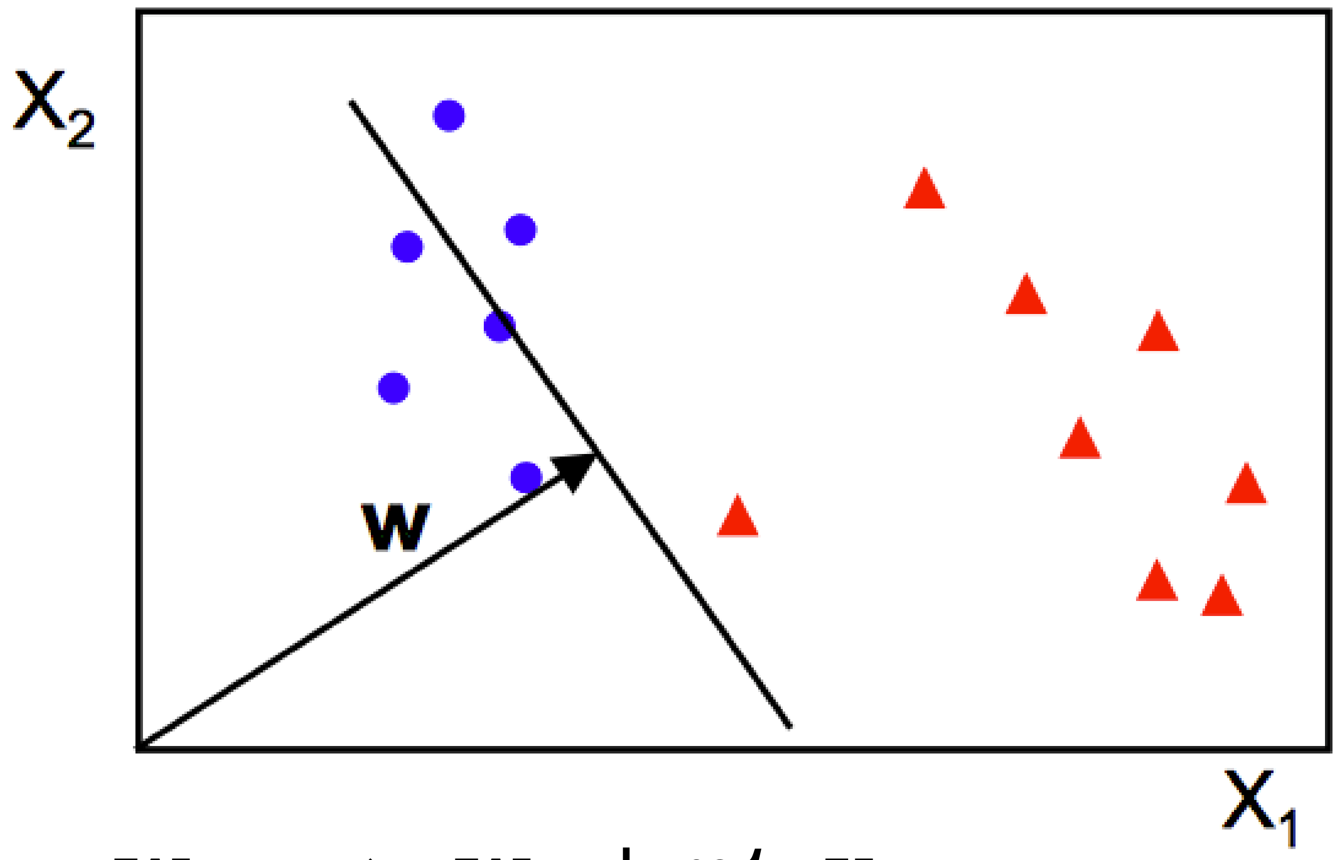
The perceptron classifier

- Perceptron Algorithm:
  - Initialize $\mathbf{w} = 0$ and $b = 0$
  - Go through each datapoint $(\mathbf{x}_n, t_n)$
  - If $\mathbf{x}_n$ is misclassified, then $\mathbf{w}_{\tau+1} \leftarrow \mathbf{w}_\tau + \eta t_n \mathbf{x}_n$
  - Until all datapoints are correctly classified

Before update



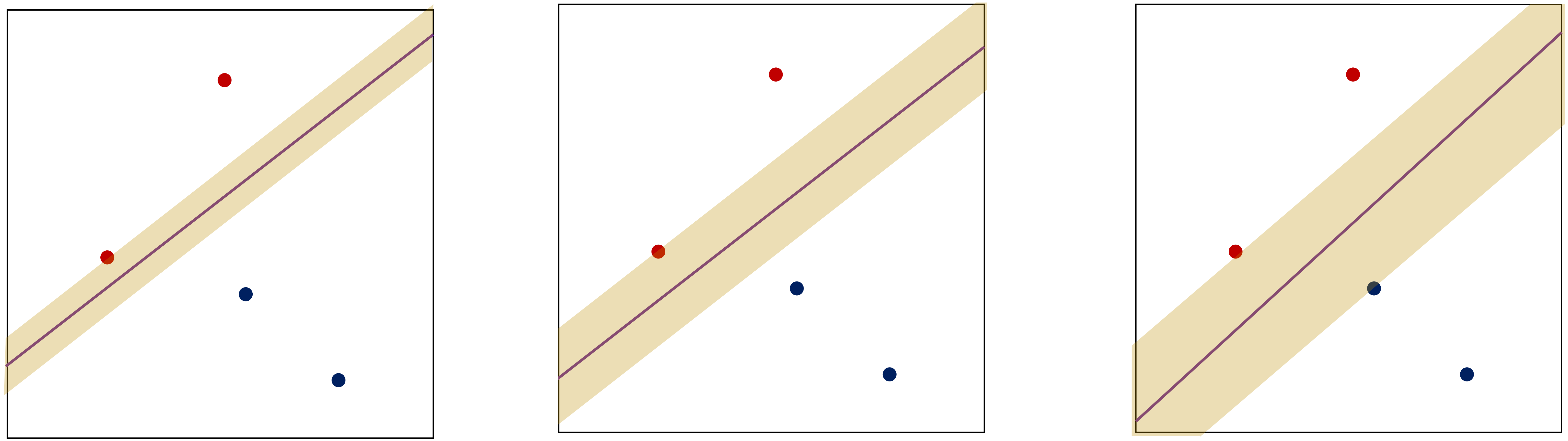
After update



$\mathbf{w}_{\tau+1} \leftarrow \mathbf{w}_\tau + \eta t_n \mathbf{x}_n$

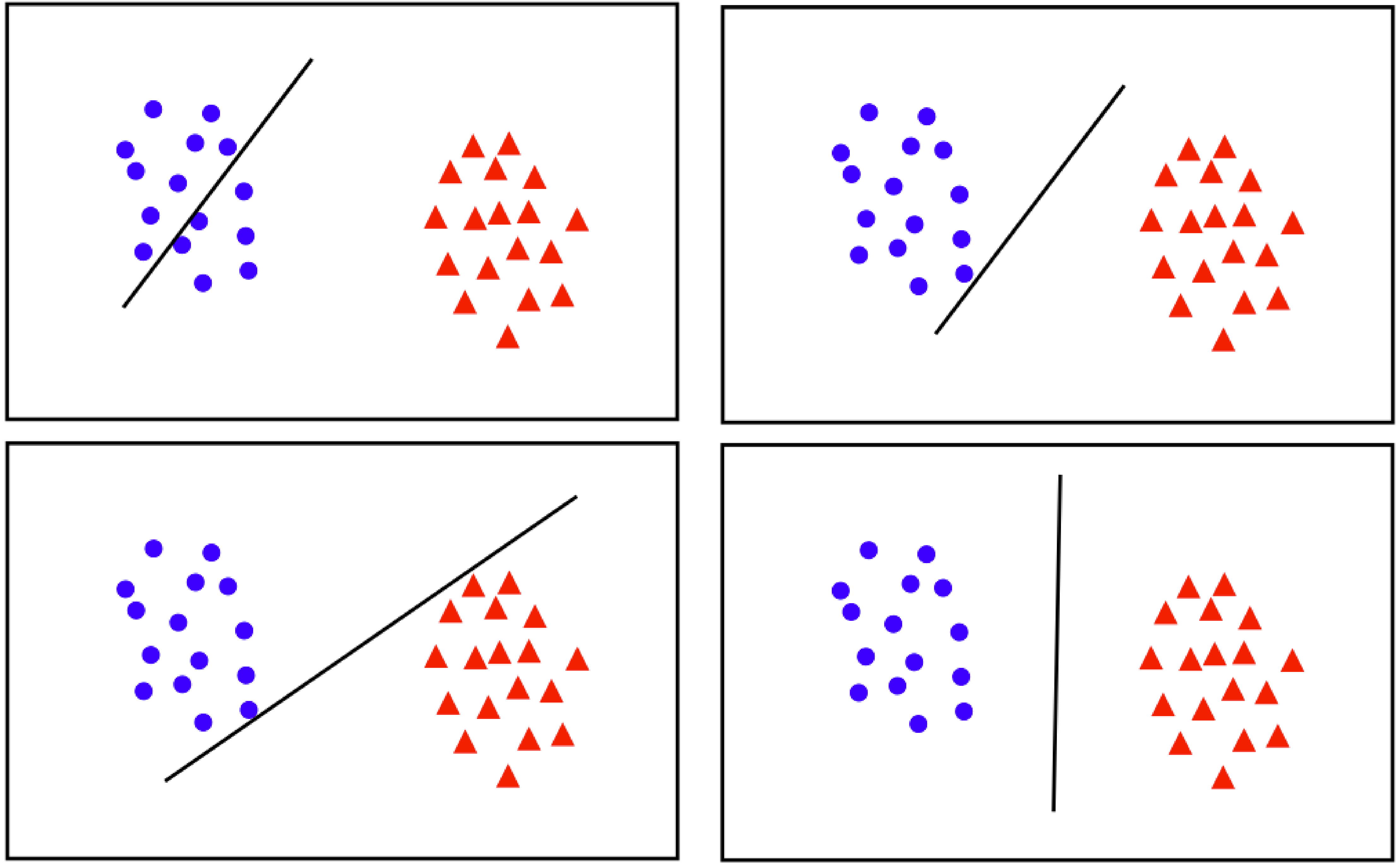
Linear separation

- We can have different separating lines:



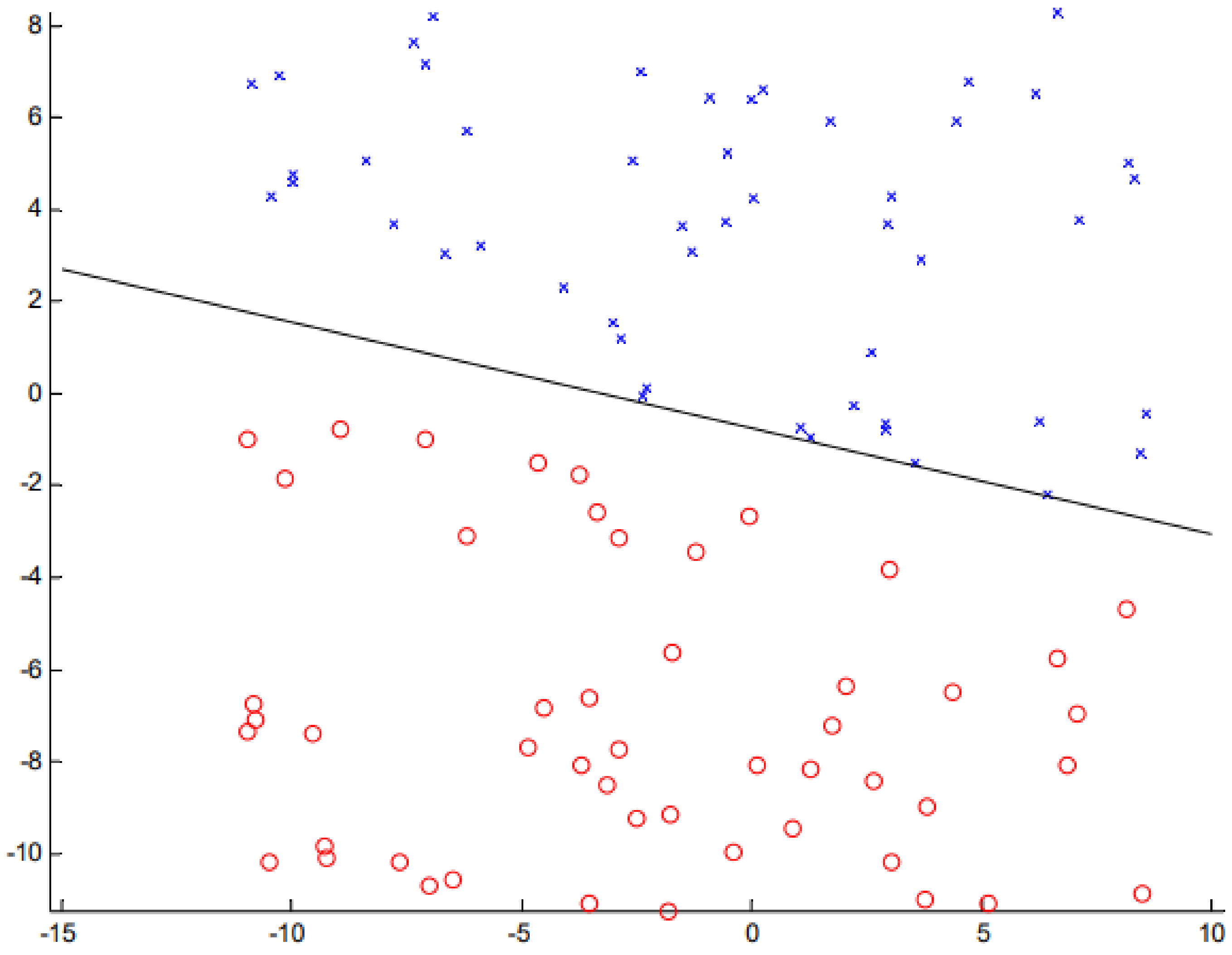
- **Which ones is the best?** Why is the bigger margin better? What $\mathbf{w}$ maximizes the margin?
- In all cases, error is zero and they are linear, so they are all good for generalization.

What is the best $\mathbf{w}$?



- **Maximum margin solution:** most stable under perturbations of the inputs

The perceptron classifier



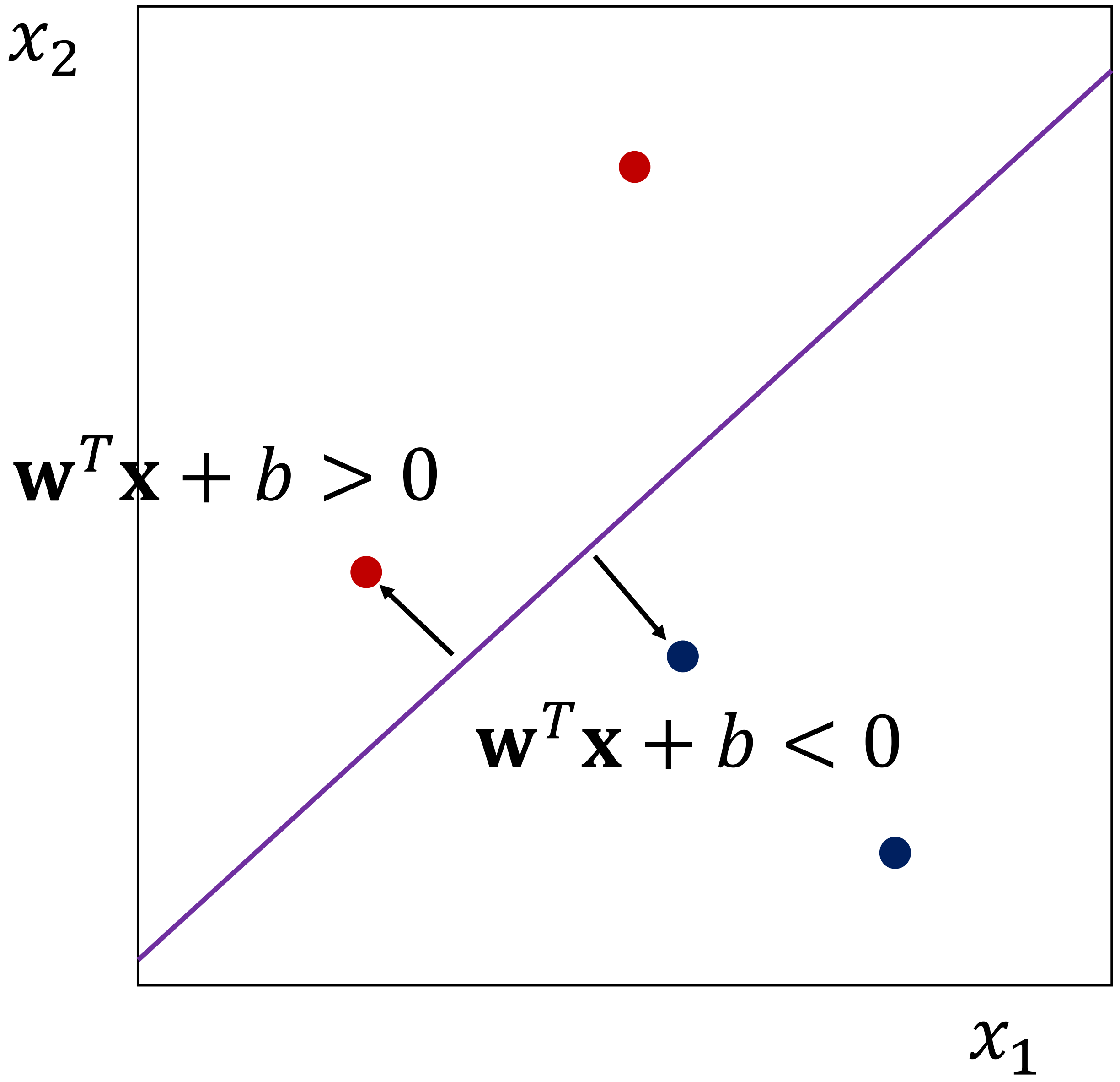
- If the data is linearly separable, then the algorithm will converge (test for linear separability)
- Convergence can be slow
- Separating line close to training data
- We would prefer a larger margin for generalization

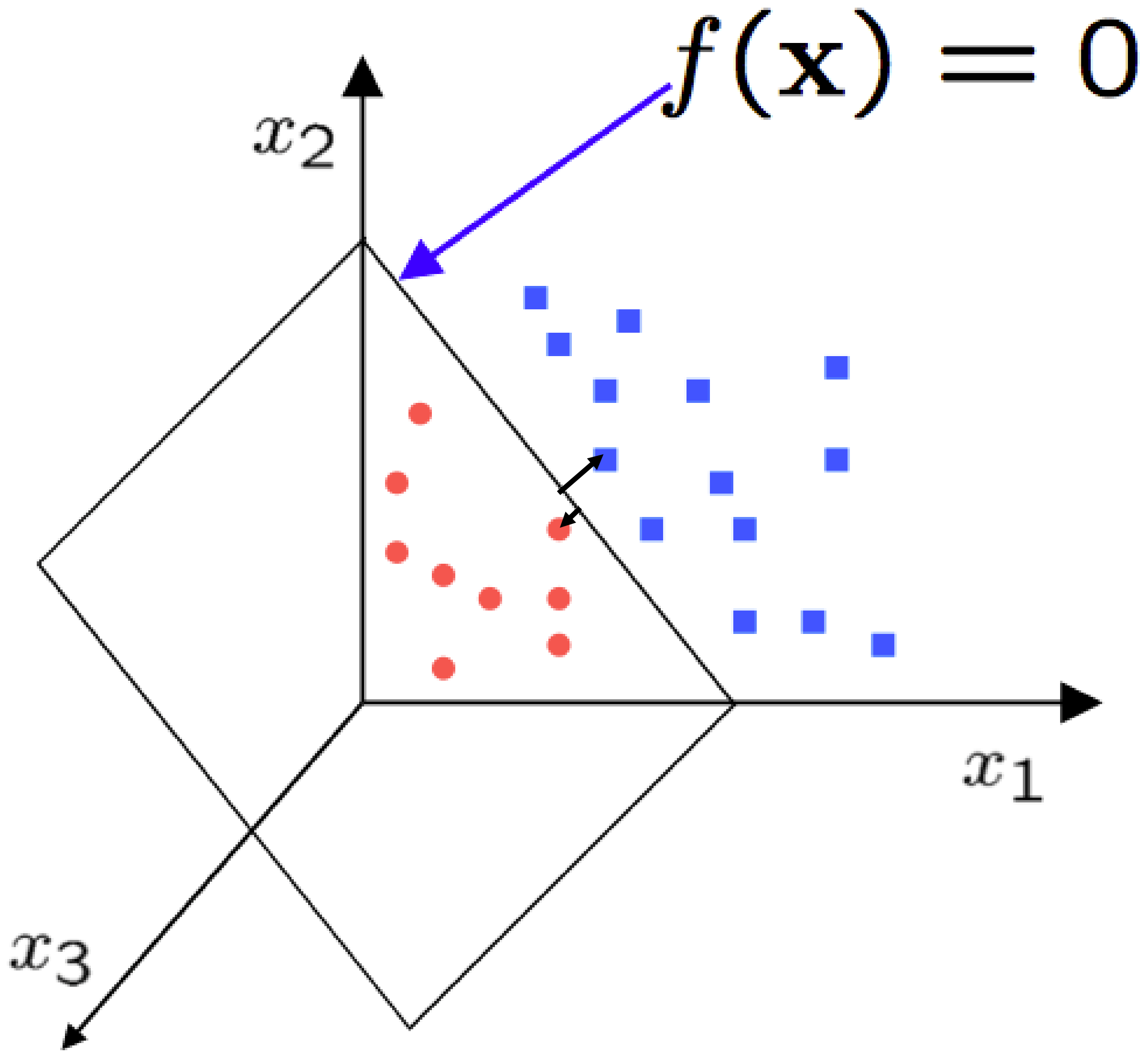
Outline

- Precursor: Linear classifier and perceptron
- Support vector machine
- Parameter learning

Finding $\mathbf{w}$ with a fat margin

- Solution for the decision boundary : $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = 0$





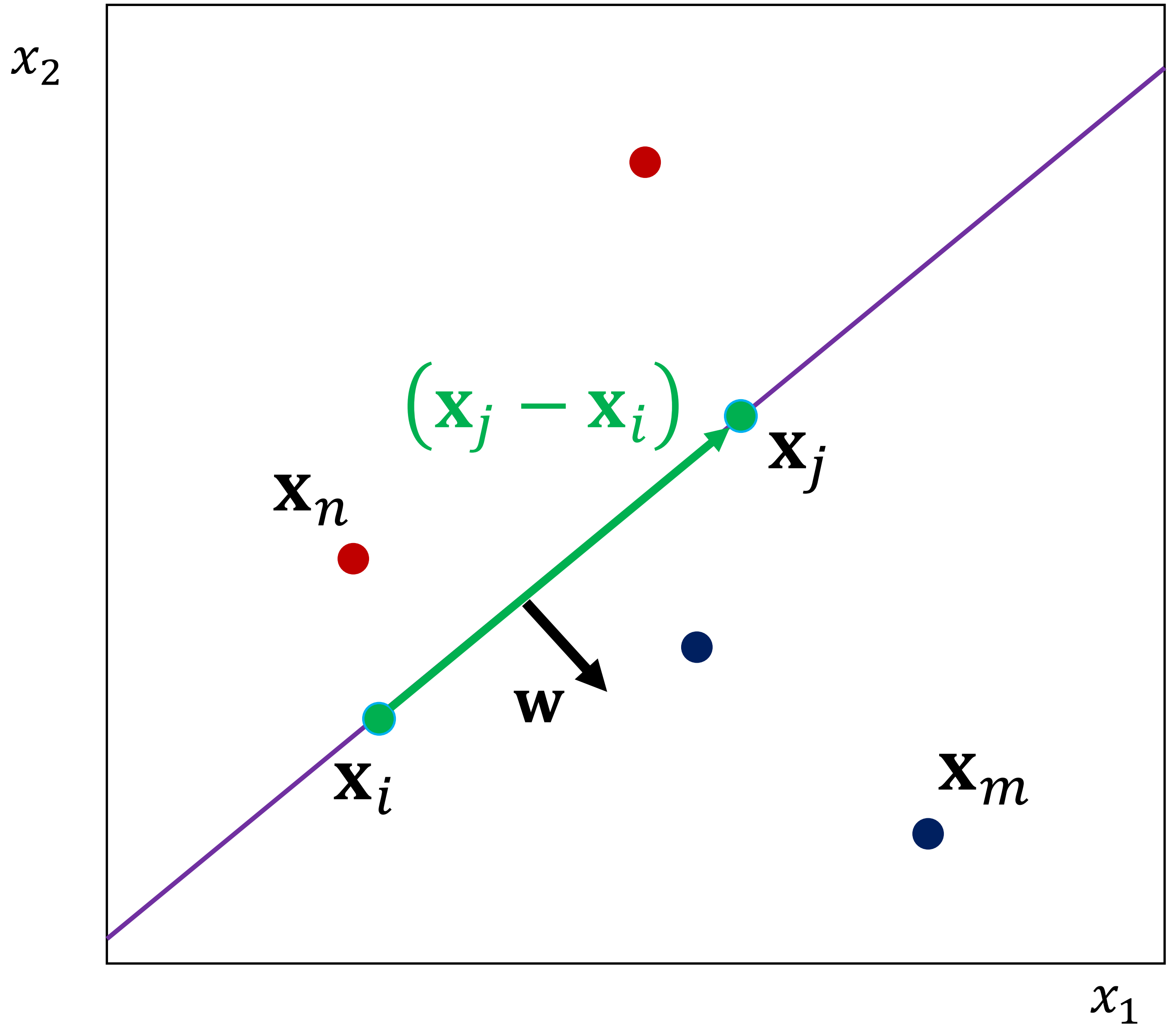
$\mathbf{w}$ perpendicular to decision line

- Consider $\mathbf{x}_j$ and $\mathbf{x}_i$ on the plane (line here):

$$\mathbf{w}^T\mathbf{x}_j + b = 0 \quad \text{and } \mathbf{w}^T\mathbf{x}_i + b = 0$$

$$\mathbf{w}^T\mathbf{x}_j = \mathbf{w}^T\mathbf{x}_i \rightarrow \mathbf{w}^T(\mathbf{x}_j - \mathbf{x}_i) = 0$$

- The vector $\mathbf{w}$ is therefore perpendicular to the decision line.



Computing the distance

- Decision line: $(\mathbf{w}^T\mathbf{x} + b)$ =0.
- Let $\mathbf{x}_n$ be the nearest data points to the decision line
- Does it matter if we scale **w**? It does not! Let's scale **w** such that $|(\mathbf{w}^T\mathbf{x}_n + b)|$ =1
- We can compute distance between $\mathbf{x}_n$ and decision line by projecting $(\mathbf{x}_n$ - $\mathbf{x}_i)$ on $\mathbf{w}$. $\mathbf{w}$ needs to be normalized to obtain the unit vector

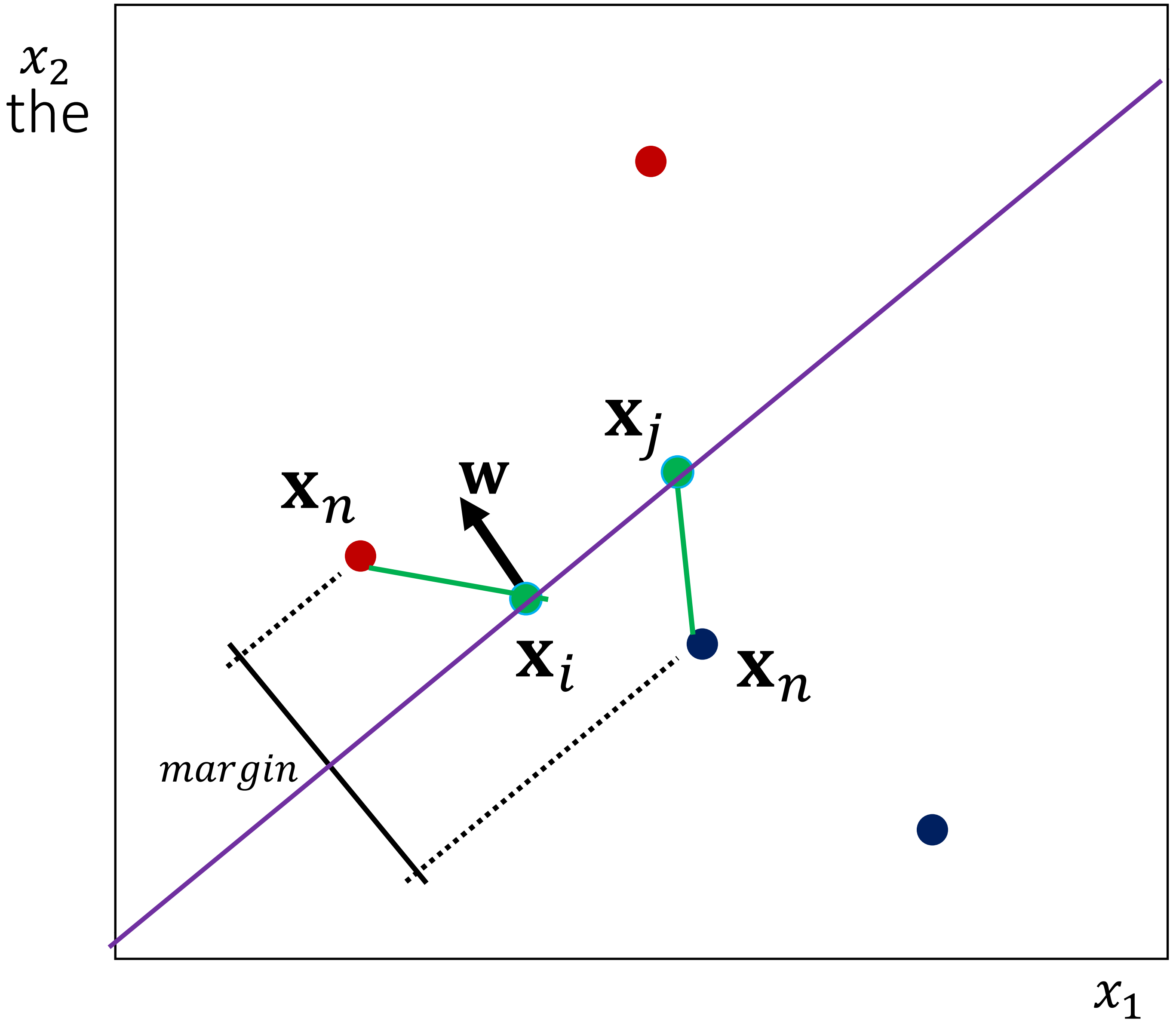
$$distance = \frac{\mathbf{w}^T}{\|\mathbf{w}\|_2}|(\mathbf{x}_n - \mathbf{x}_i)|$$

$$distance = \frac{\mathbf{1}}{\|\mathbf{w}\|_2}|(\underbrace{\mathbf{w}^T\mathbf{x}_n + \boldsymbol{b}}_{\substack{\text{Constraint} \\ |(\mathbf{w}^T\mathbf{x}_n + b)| = 1}} \underbrace{- \mathbf{w}^T\mathbf{x}_i - \boldsymbol{b}}_{\substack{\text{A point on the decision line} \\ |(\mathbf{w}^T\mathbf{x}_i + b)| = 0}})|$$

$$distance = \frac{\mathbf{1}}{\|\mathbf{w}\|_2}$$

- Equal *distance* on both sides of decision line

$$margin = 2 * distance = \frac{2}{\|\mathbf{w}\|_2}$$



Our goal is to maximize the margin

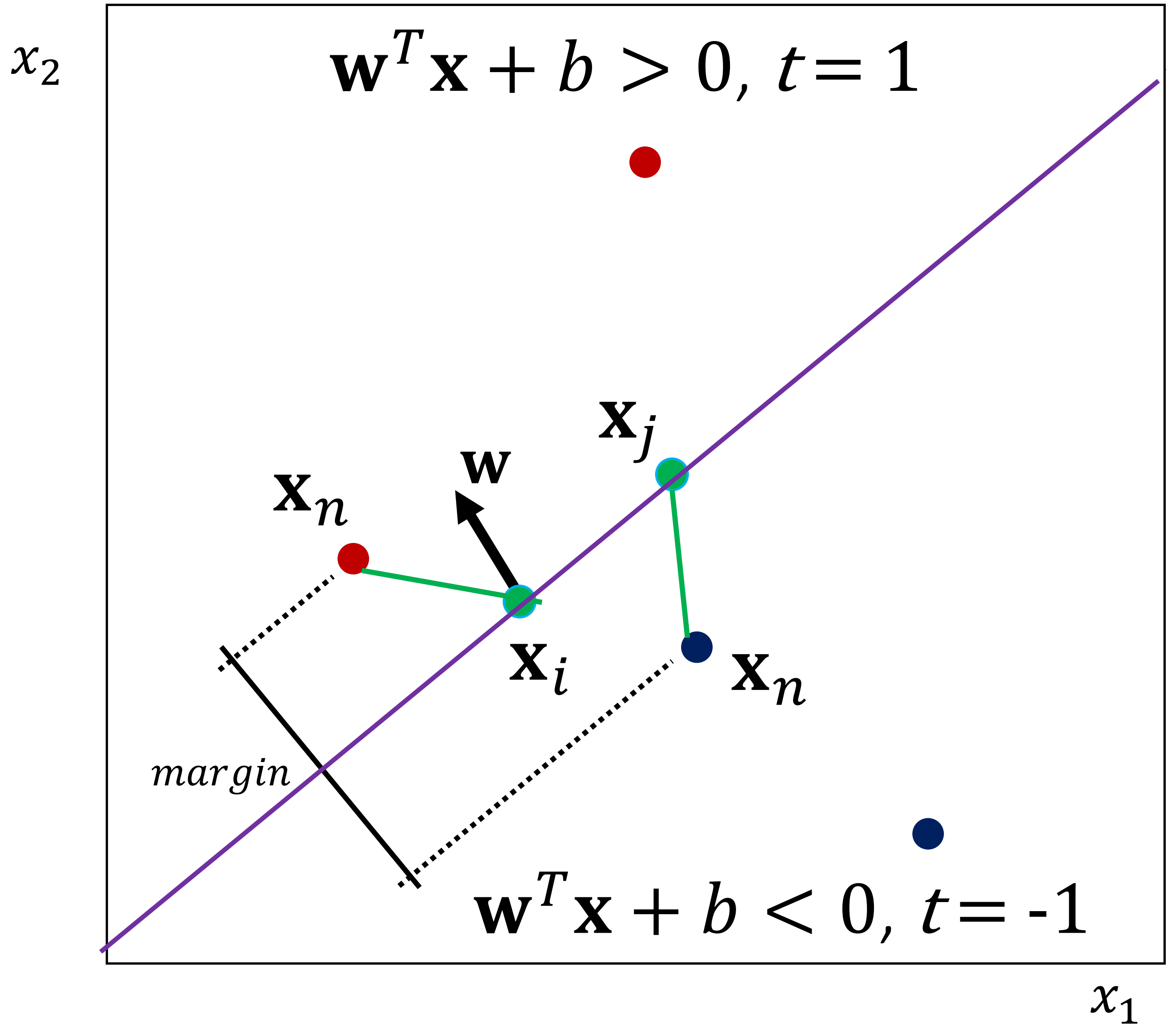
$$\max_{\mathbf{w},b} \frac{2}{||\mathbf{w}||}$$

$$s.t.\ |(\mathbf{w}^T\mathbf{x}_n + b)| = 1$$

for $n =$ nearest points to decision line

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

for $n = 1, \dots, N$



Our goal is to maximize the margin

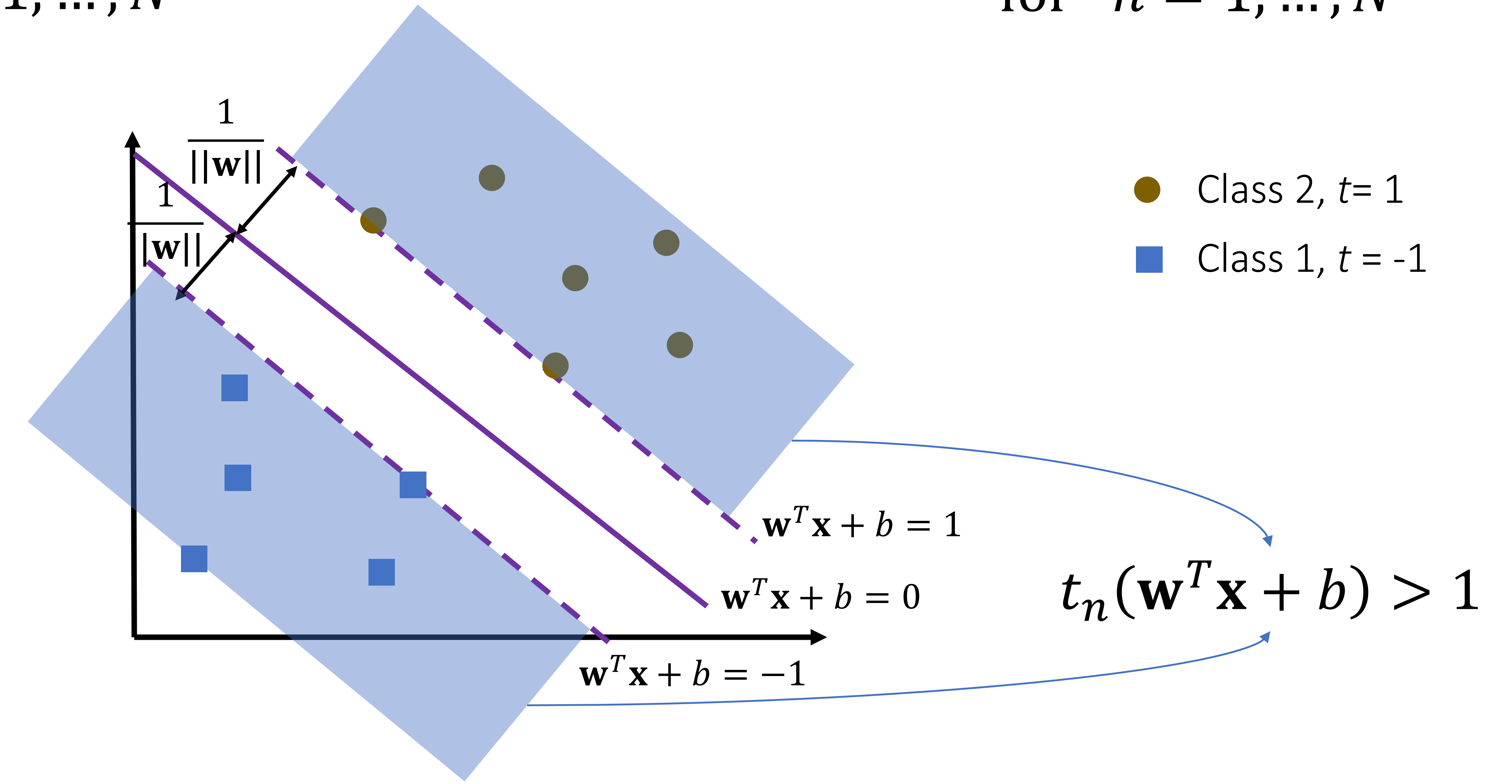
$$\max_{\mathbf{w},b} \frac{2}{||\mathbf{w}||}$$

$$s.t. \quad t_n(\mathbf{w}^T\mathbf{x}_n + b) \geq 1 \quad \text{for } n = 1, \dots, N$$

Equivalent

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

$$s.t. \quad t_n(\mathbf{w}^T\mathbf{x}_n + b) \geq 1 \quad \text{for } n = 1, \dots, N$$



Outline

- Precursor: Linear classifier and perceptron
- Support vector machine
- Parameter learning

Constrained optimization

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

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

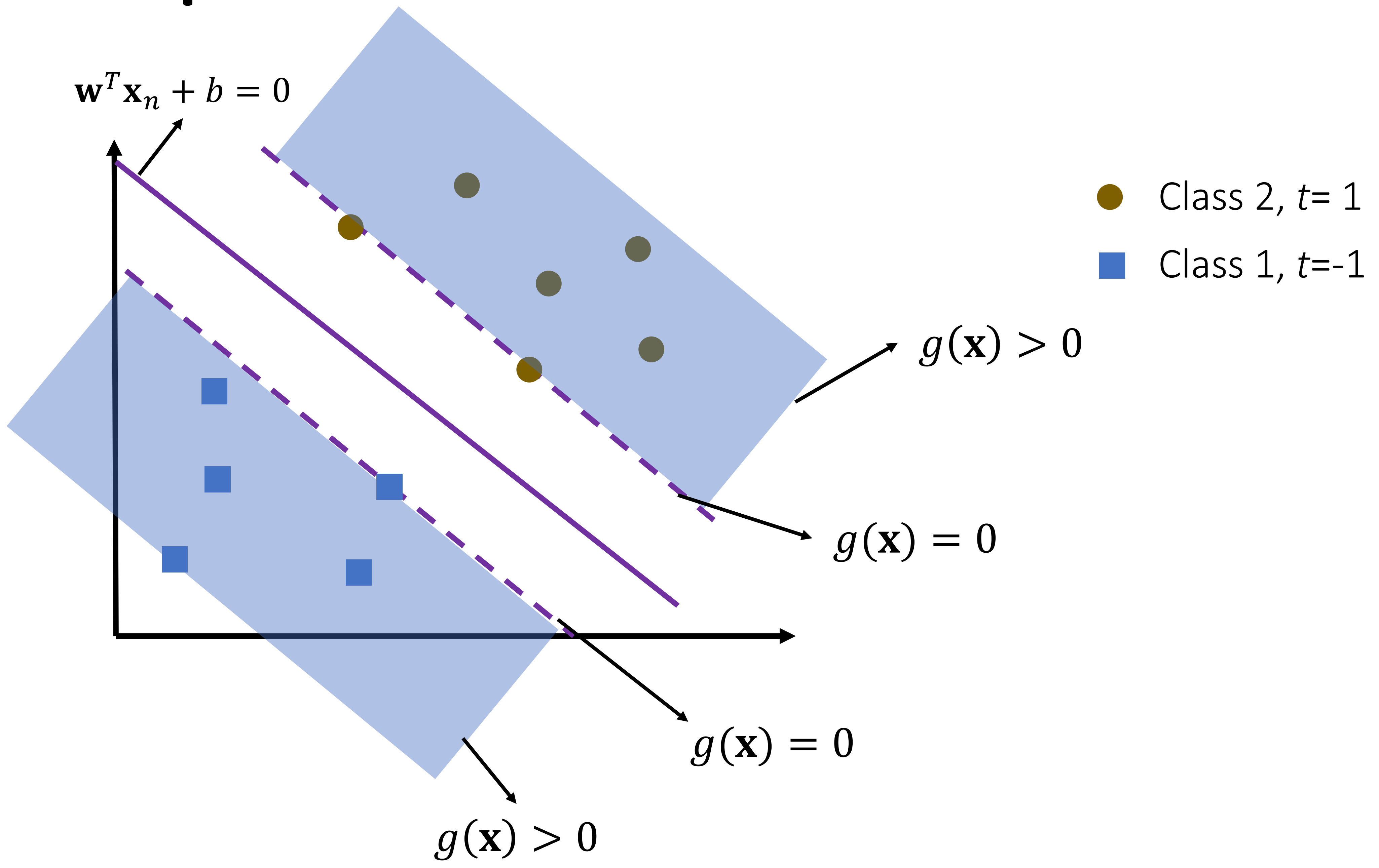
- Use the Lagrangian method with the Karush-Kuhn-Tucker (KKT) conditions:
  - Rewrite the inequality constraint as an equality constraint: $g(\mathbf{x}_n) = t_n(\mathbf{w}^T\mathbf{x}_n + b) - 1$
  - Write the Lagrangian with the equality constraint by introducing Lagrangian multipliers a_n:

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

- KKT conditions:
  1. $g(\mathbf{x}_n) \geq 0$ (Primal feasibility)
  2. $a_n \geq 0$ (Dual feasibility)
  3. $g(\mathbf{x}_n)a_n = 0$ (Complementary slackness)

$$\Rightarrow \begin{cases} g(\mathbf{x}_n) > 0, & a_n = 0 \\ a_n > 0, & g(\mathbf{x}_n) = 0 \end{cases}$$

Constrained optimization



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

$g(\mathbf{x}_n)a_n = 0$ (Complementary slackness) $\Rightarrow \begin{cases} g(\mathbf{x}_n) > 0, & a_n = 0 \\ a_n > 0, & g(\mathbf{x}_n) = 0 \end{cases}$

Lagrange formulation

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

- Minimize with respect to $\mathbf{w}$ and b and maximize with respect to $\mathbf{a}$:

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

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

- We can replace these expressions back into the Lagrangian so that it is now only a function of $\mathbf{a}$

Dual representation

- We can now maximize the Lagrangian w.r.t $\mathbf{a}$ (by substituting value of **w**):

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

Subject to

$$a_n \geq 0, \text{for } n = 1, \ldots, N$$

$$\sum_{m=1}^{N} a_n t_n = 0$$

The solution: quadratic programming

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

- Quadratic programming packages usually use **min,** so we multiply the expression by -1:

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

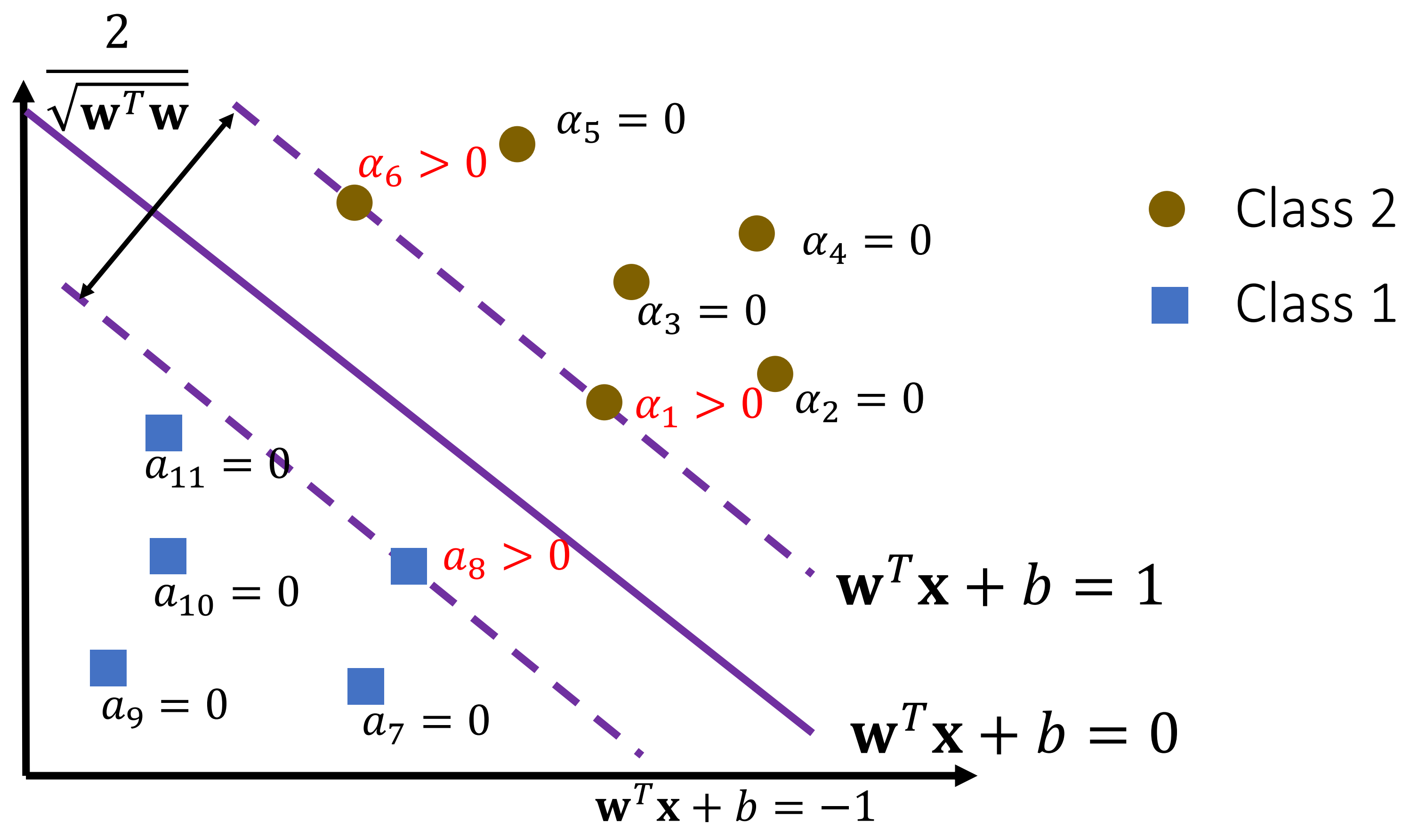
$$\min_{\mathbf{a}} \frac{1}{2} \mathbf{a}^T \begin{bmatrix} t_1 t_1 \mathbf{x}_1^T \mathbf{x}_1 & t_1 t_2 \mathbf{x}_1^T \mathbf{x}_1 & \dots & t_1 t_N \mathbf{x}_1^T \mathbf{x}_N \\ t_2 t_1 \mathbf{x}_2^T \mathbf{x}_1 & t_2 t_2 \mathbf{x}_2^T \mathbf{x}_2 & \cdots & t_2 t_N \mathbf{x}_2^T \mathbf{x}_N \\ \dots & \dots & \cdots & \dots \\ t_N t_N \mathbf{x}_1^T \mathbf{x}_N & t_N t_2 \mathbf{x}_N^T \mathbf{x}_2 & \cdots & t_N t_N \mathbf{x}_N^T \mathbf{x}_N \end{bmatrix} \mathbf{a} - \mathbf{Ia} = \frac{1}{2}\ \mathbf{a}^T \mathbf{Q} \mathbf{a} - \underbrace{\mathbf{Ia}}_{\text{Linear term}}$$

Quadratic coefficients

Subject to: $\mathbf{a}^T \mathbf{t} = 0$ and $a_n \geq 0$

The solution: quadratic programming

- A quadratic programming package will then give us $a = (a_1 \quad \cdots \quad a_N)^T$
- From our KKT condition $a_n g(\mathbf{x}_n) = 0$:
  - $t_n(\mathbf{w}^T\mathbf{x}_n + b) - 1 > 0 \rightarrow a_n = 0$
  - $t_n(\mathbf{w}^T\mathbf{x}_n + b) - 1 = 0 \rightarrow a_n > 0$, then $\mathbf{x}_n$ is a support vector



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:

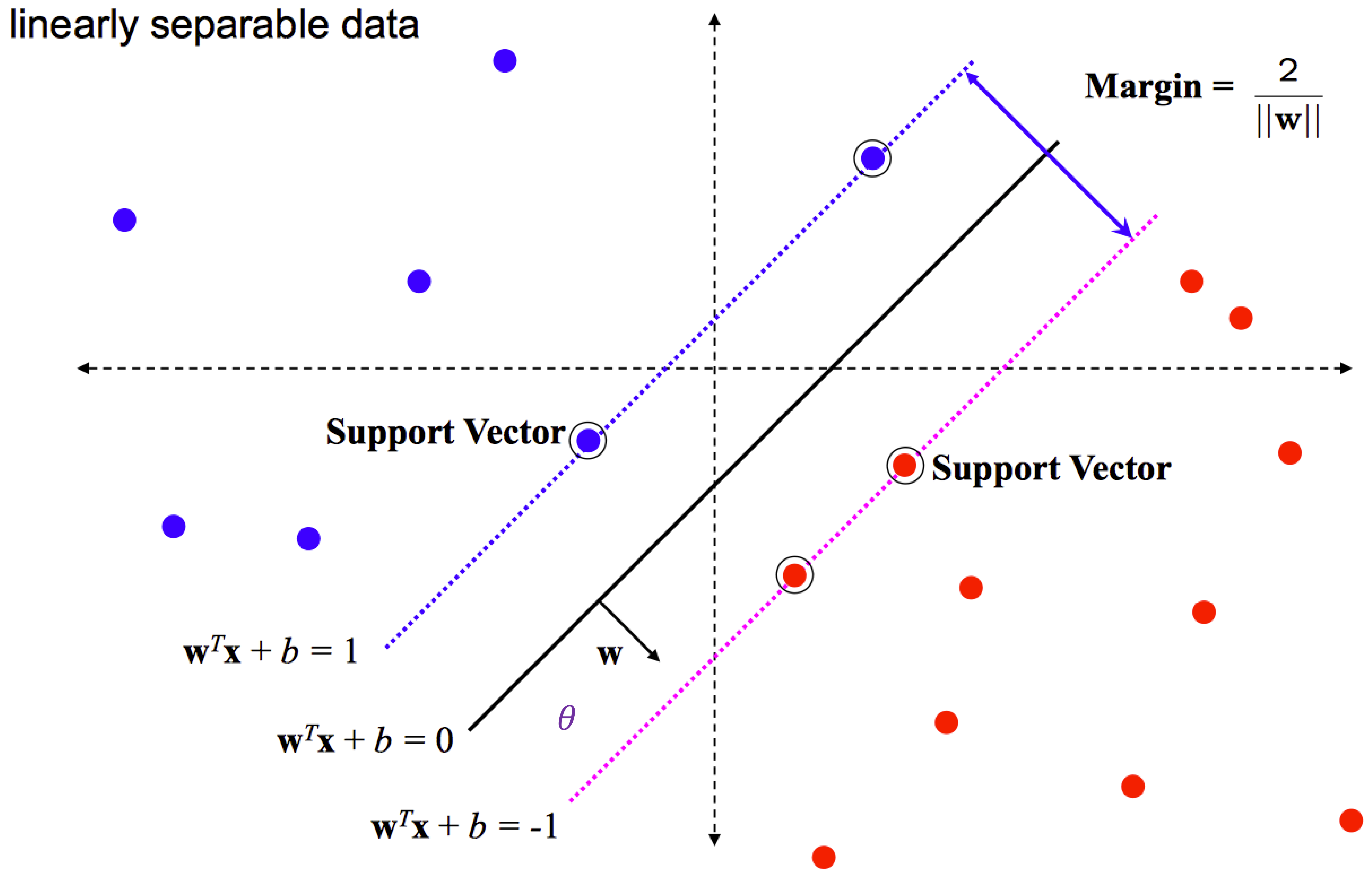
$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = \sum_{n=1}^{N} a_n t_n \mathbf{x}_n^T\mathbf{x} + b$$

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

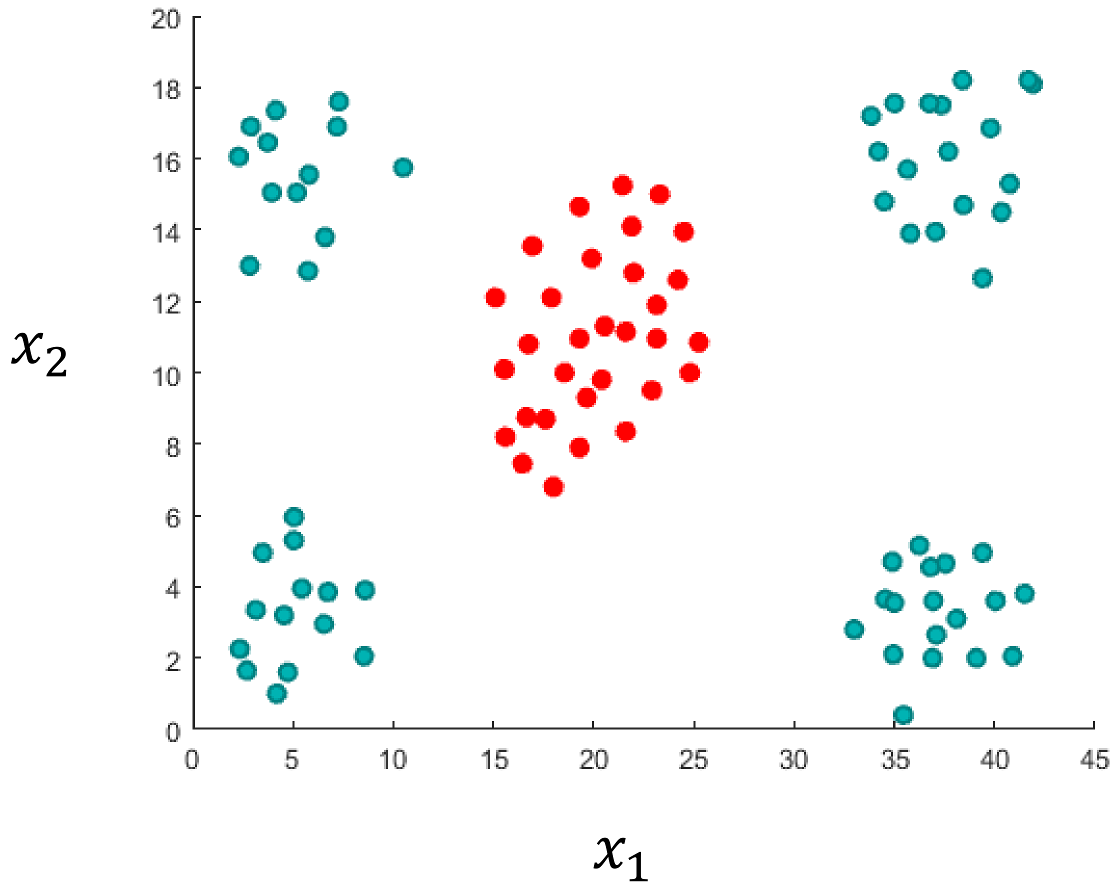
$$f(\mathbf{x}) = \sum_{x_n \in SV} a_n t_n \mathbf{x}_n^T\mathbf{x} + b$$

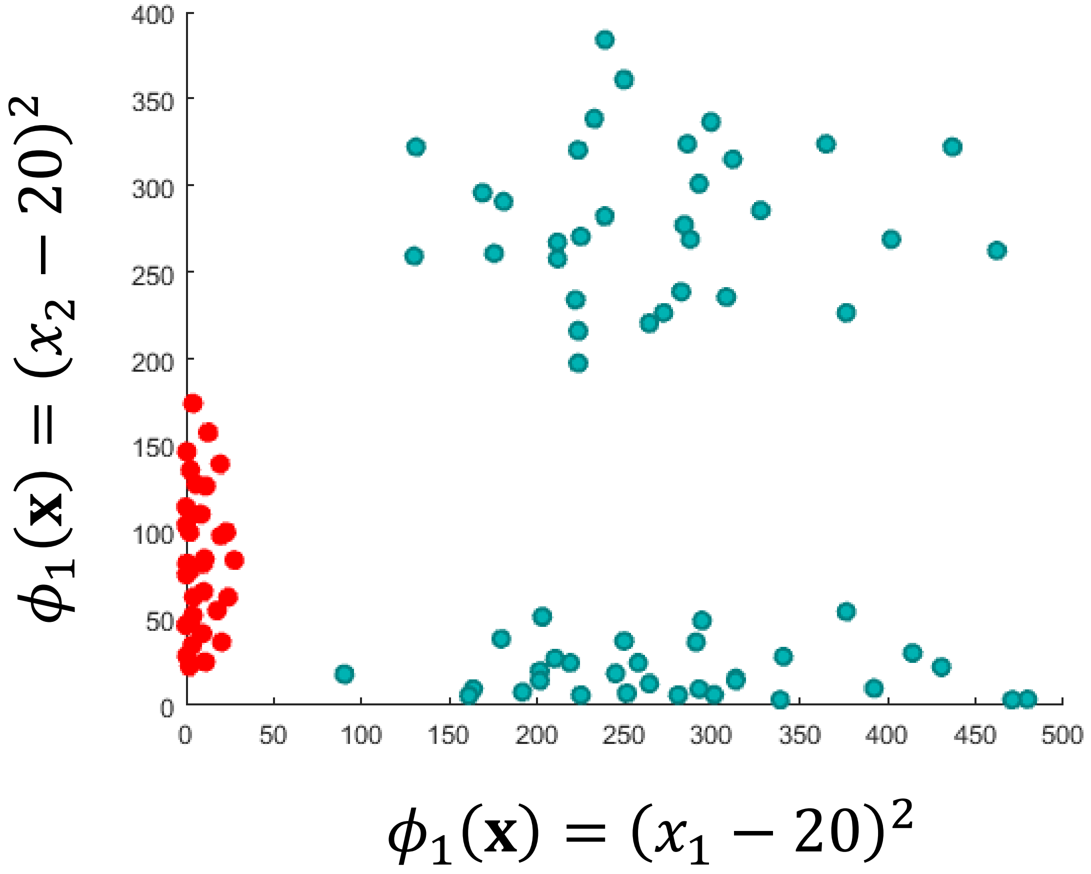
Classify $\mathbf{x}$ as class 1 if the result is positive, and class 2 otherwise

Geometric interpretation



From $\mathbf{x}$- to $\phi(\mathbf{x})$-space





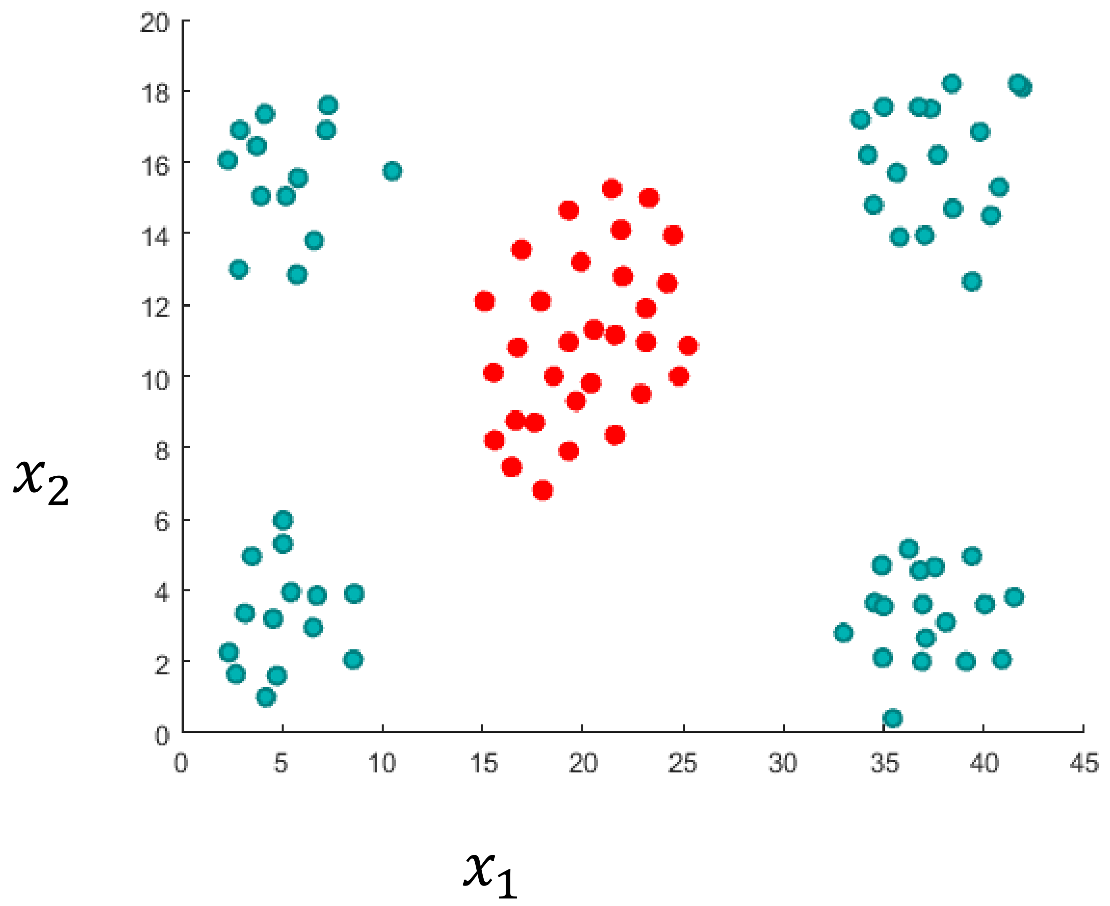
$$\mathbf{x} \longrightarrow \phi(\mathbf{x})$$

$$\phi(\mathbf{x}) = \begin{bmatrix} (x_1 - 20)^2 \\ (x_2 - 20)^2 \end{bmatrix}$$

Support vectors

- In $\mathbf{x}$-space:

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

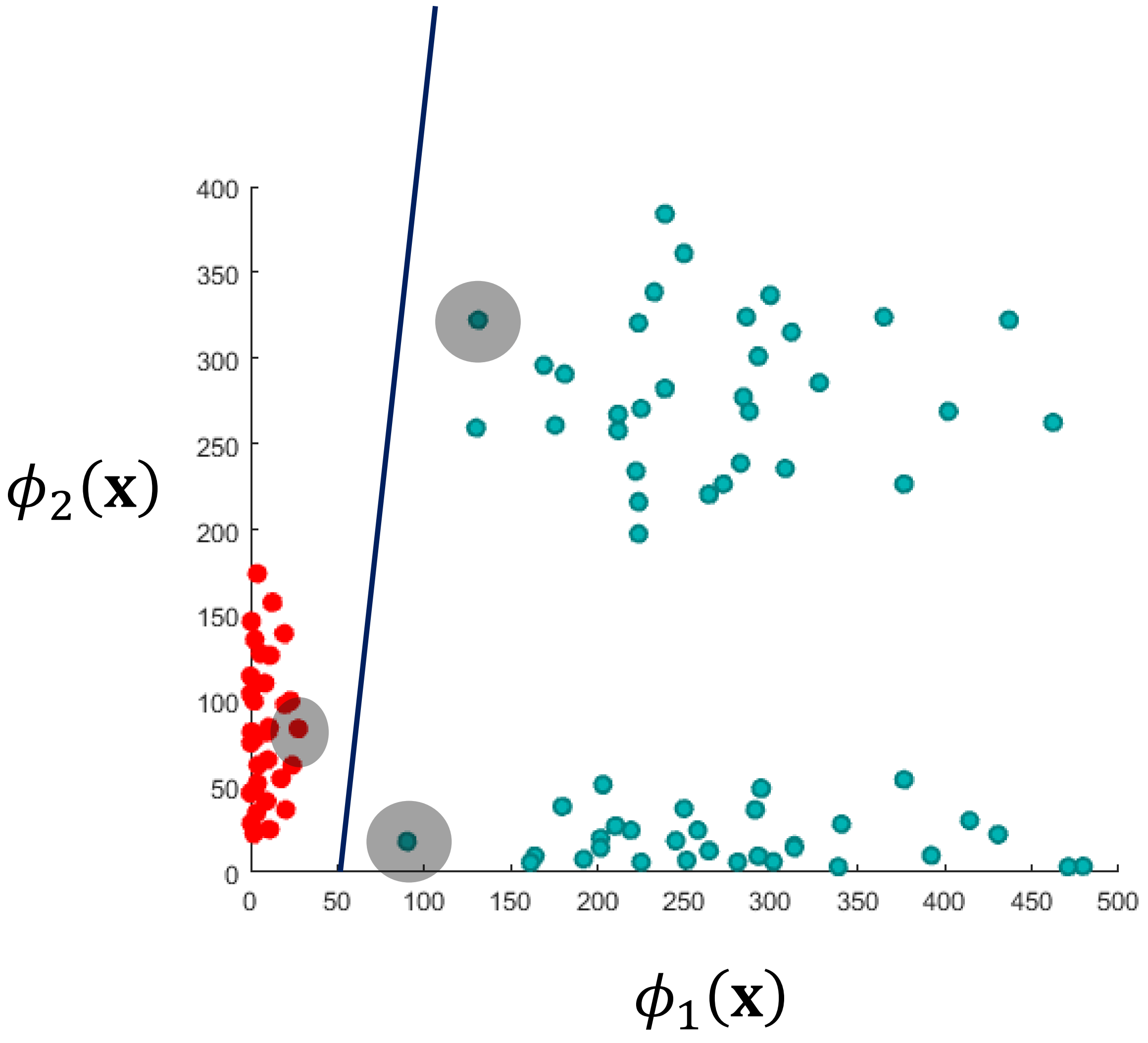


In this format for a dataset with N datapoints and D features, we need to learn N variables. Is this a good idea?

Support vectors

- In $\boldsymbol{\phi}(\mathbf{x})$-space:

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



Support vectors

- In $\mathbf{x}$-space, they are called pre-images of support vectors