Pattern Recognition & Machine Learning

模式识别与机器学习

Lecture 2: Linear Models

Gao Huang (黄高)
gaohuang@tsinghua.edu.cn
Department of Automation, Tsinghua University

Course ID：30250293

Recap: A typical machine learning model

Classifier

Prediction

Sample

$y = f(x)$

"Airplane"

Input x

Machine Learning Model

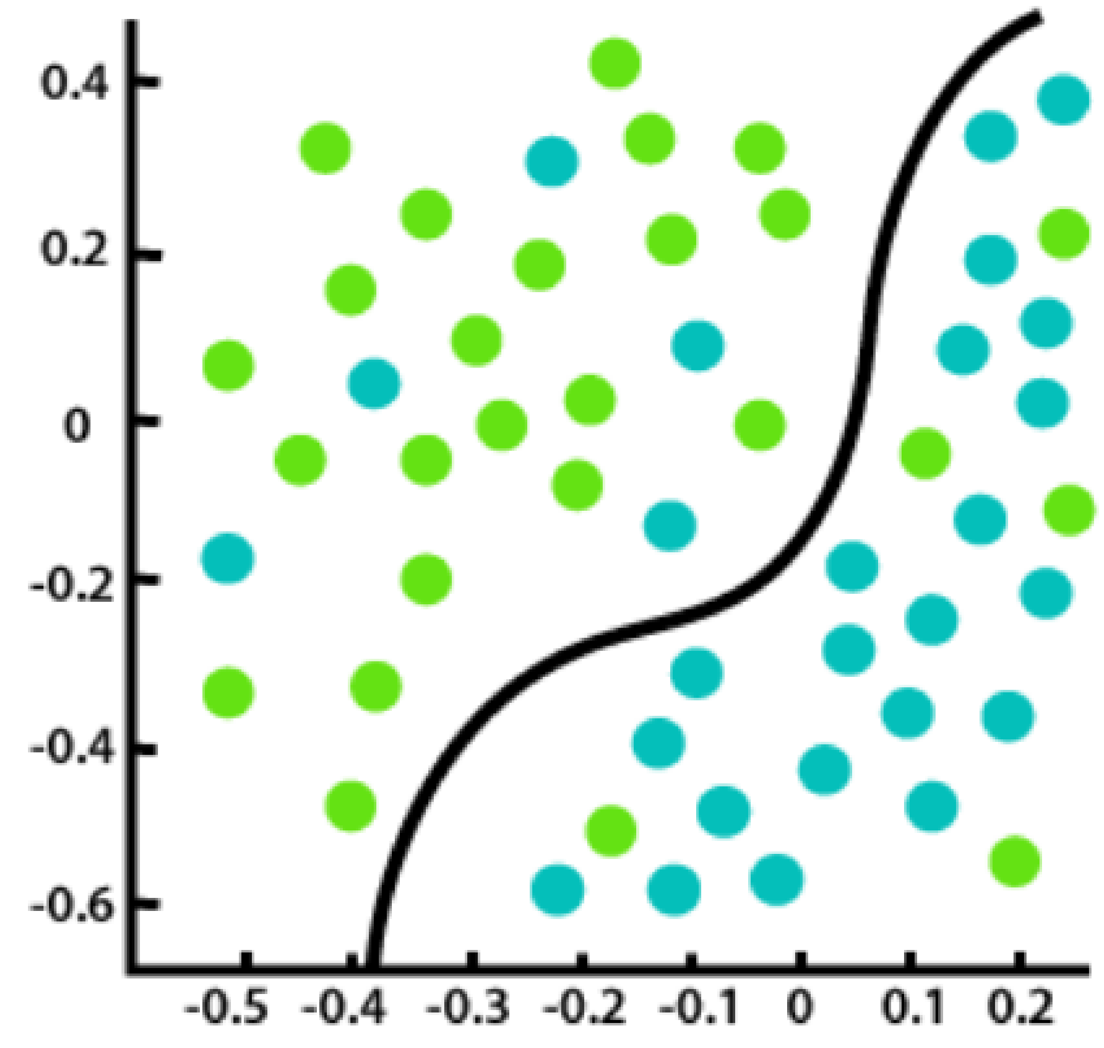
Output y

$x = [x_1, , \dots, x_d]$

Typical Learning Tasks

Classification (**Discrete** Label Space)

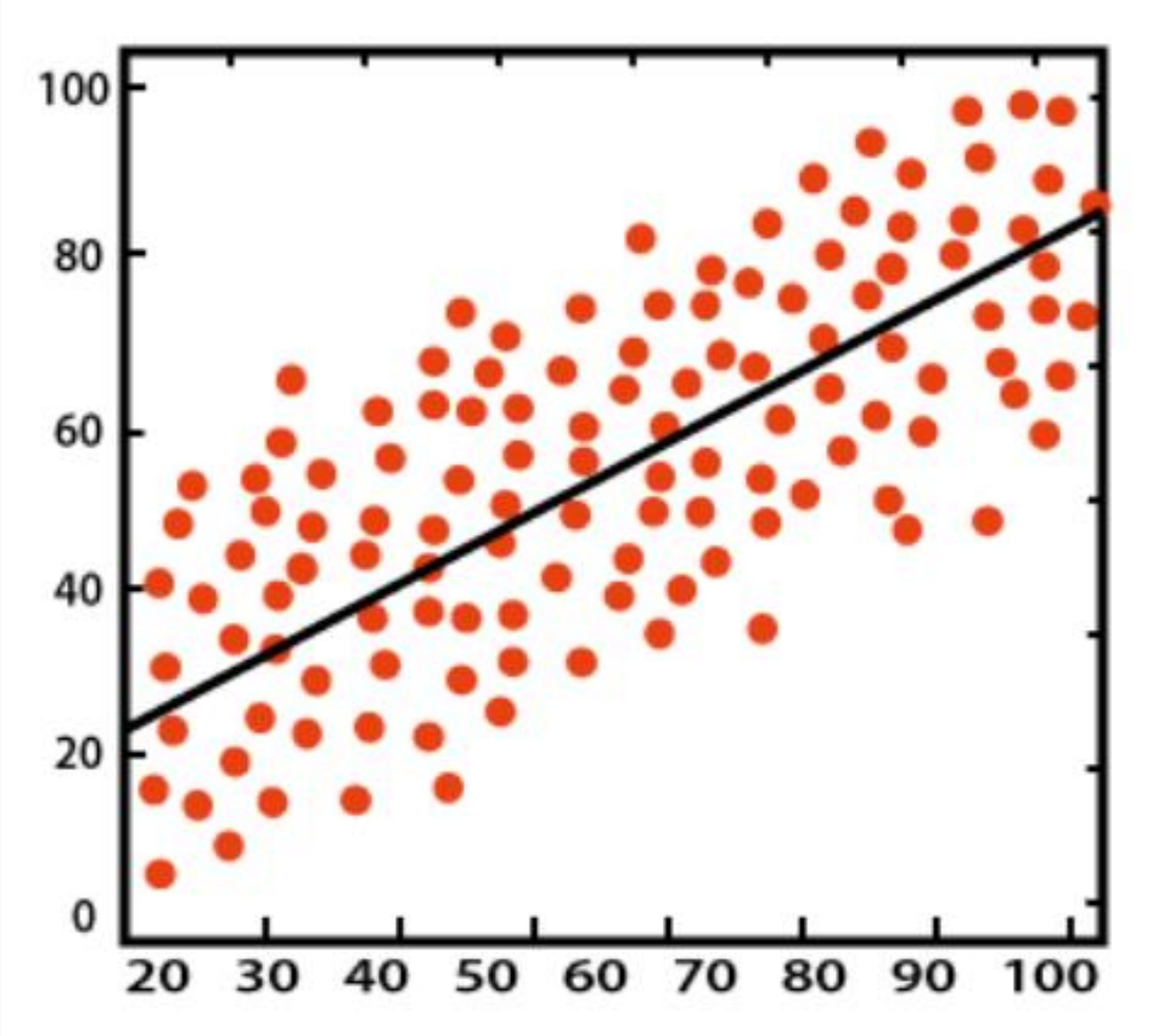
e.g., spam filtering, medical diagnosis



Classification algorithms:
Logistic Regression, K-Nearest Neighbors, Support Vector Machines, Naïve Bayes, Decision Tree Classification, Random Forest, Neural Networks

Regression (**Continuous** Label Space)

e.g., stock prediction, air quality prediction

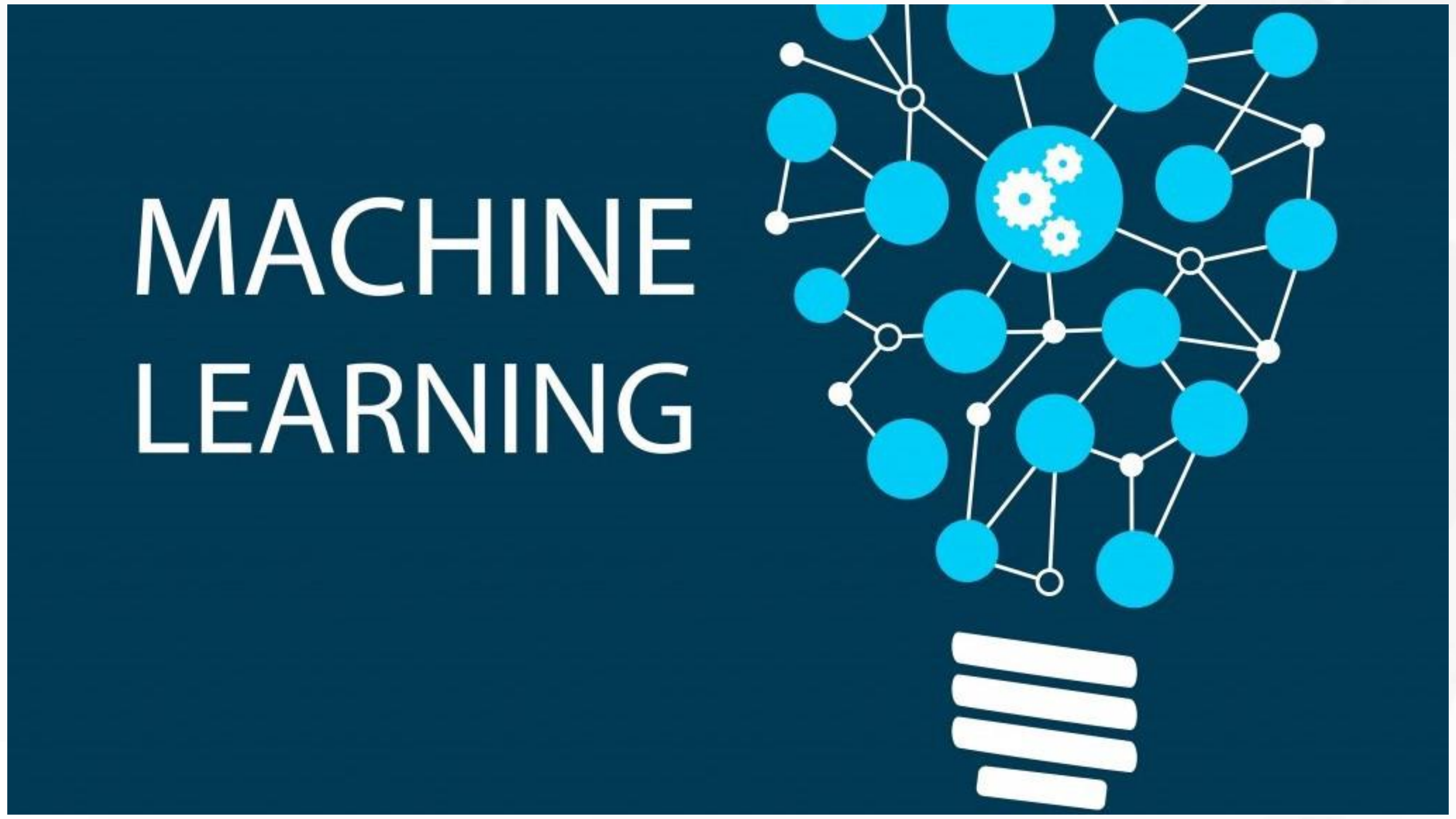


Regression algorithms:
Simple Linear Regression, Polynomial Regression, Support Vector Regression, Decision Tree Regression, Neural Networks

Basic Building Blocks of Machine Learning

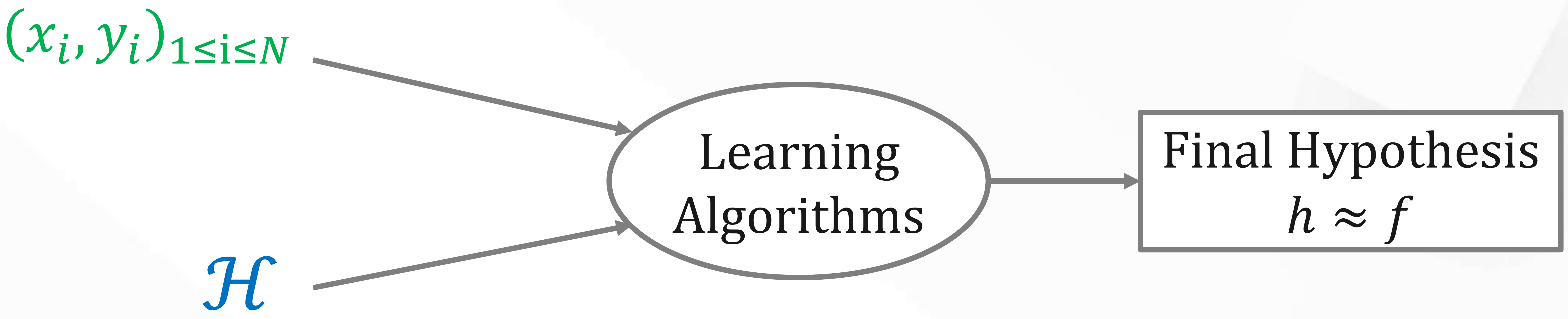
Tsinghua University

- How to create a learning machine?
  - It needs a teacher
    - We design it! (Features and Models)
  - It needs learning materials
    - Training Data
  - We need to set a learning target
    - Target Function or Learning Criterion
  - We need to tell it how to learn
    - Learning/Training Algorithms



Components of Learning

1. **Optimal Classifier (or Decision Function)** - the function that maps $f: \mathcal{X} \rightarrow \mathcal{Y}$
2. **Training Samples** - $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$
3. **Hypothesis Space $\mathcal{H}$** - collection of prediction functions we are choosing from



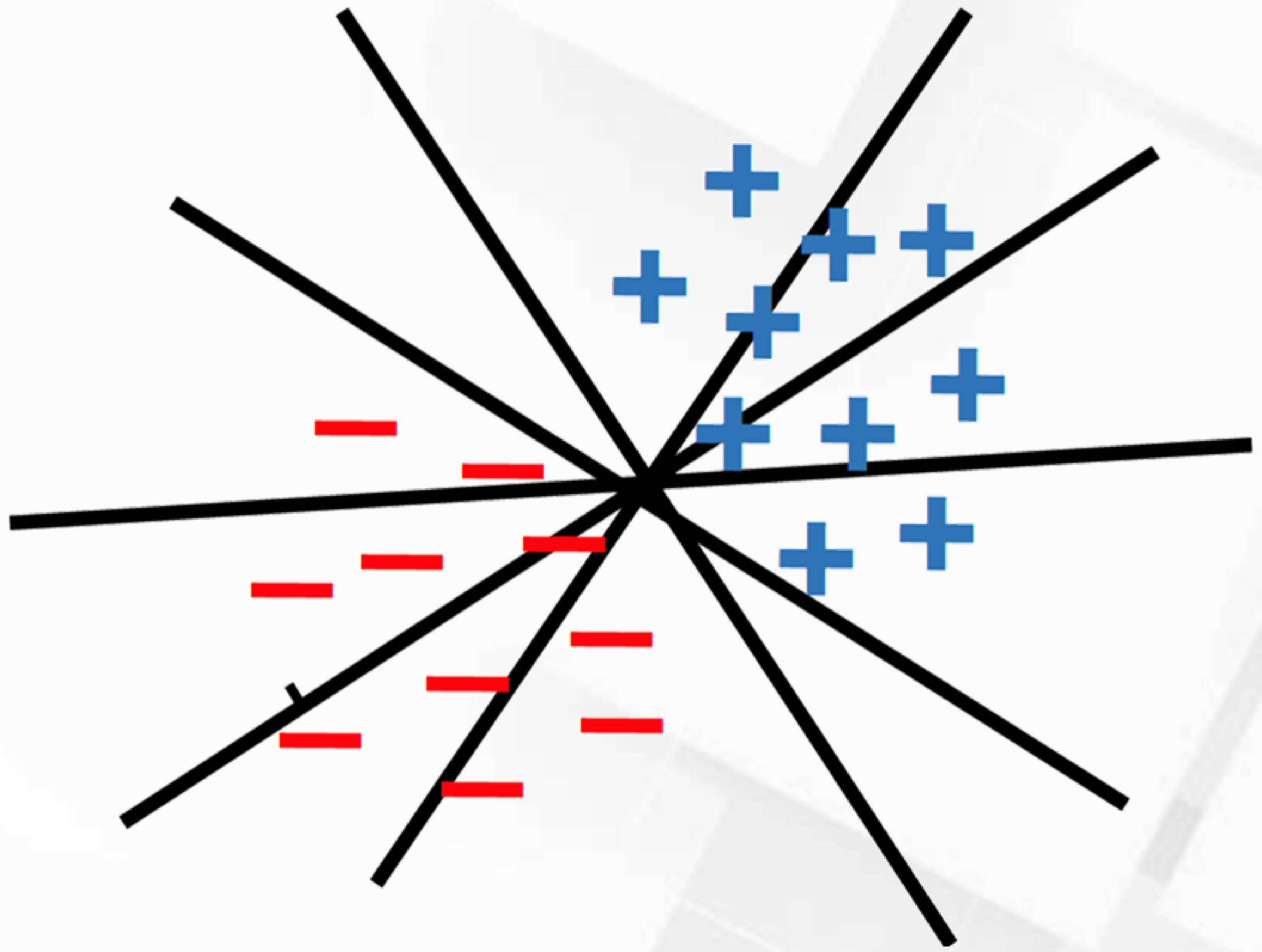
Find the **optimal classifier ($f: \mathcal{X} \rightarrow \mathcal{Y}$)**

from the **Hypothesis Space** that can fit our

training samples the best.

Hypothesis Space

- **Hypothesis space** is a set of functions that maps $\mathcal{X} \rightarrow \mathcal{Y}$.
  - It is the collection of prediction functions we are choosing from.
- We want hypothesis space that…
  - Includes only those functions that have desired regularity
    - Continuity
    - Smoothness
    - Simplicity
  - Easy to work with
- An example hypothesis space:
  - All linear hyperplanes for classification



Which hyperplane is the best?

Loss Function

- **Loss function:** $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ measures the difference between a prediction $h(\boldsymbol{x})$ and an actual output y :

$$\ell\big(y, h(\boldsymbol{x})\big) = \big(y - h(\boldsymbol{x})\big)^2 \quad \text{(}\textbf{Squared loss}\text{ for Regression)}$$

$$\ell\big(y, h(\boldsymbol{x})\big) = \mathbf{1}[y \neq h(\boldsymbol{x})] \quad \text{(}\textbf{0–1 loss}\text{ for Classification)}$$

In the logistic regression section, we will see why we cannot use squared loss for classification.

- The canonical training procedure of machine learning:

Fit dataset with best hypothesis

$$\min_{\boldsymbol{w}} \sum_{i=1}^{m} \ell\big(h_{\boldsymbol{w}}(\boldsymbol{x}_i), y_i\big)$$

w is parameters of the hypothesis h

Linear Model Outline

1. Linear Regression
2. Linear Discriminant Analysis
3. Logistic Regression
4. Perceptron

Linear Model Outline

Tsinghua University

1. Linear Regression

2. Linear Discriminant Analysis

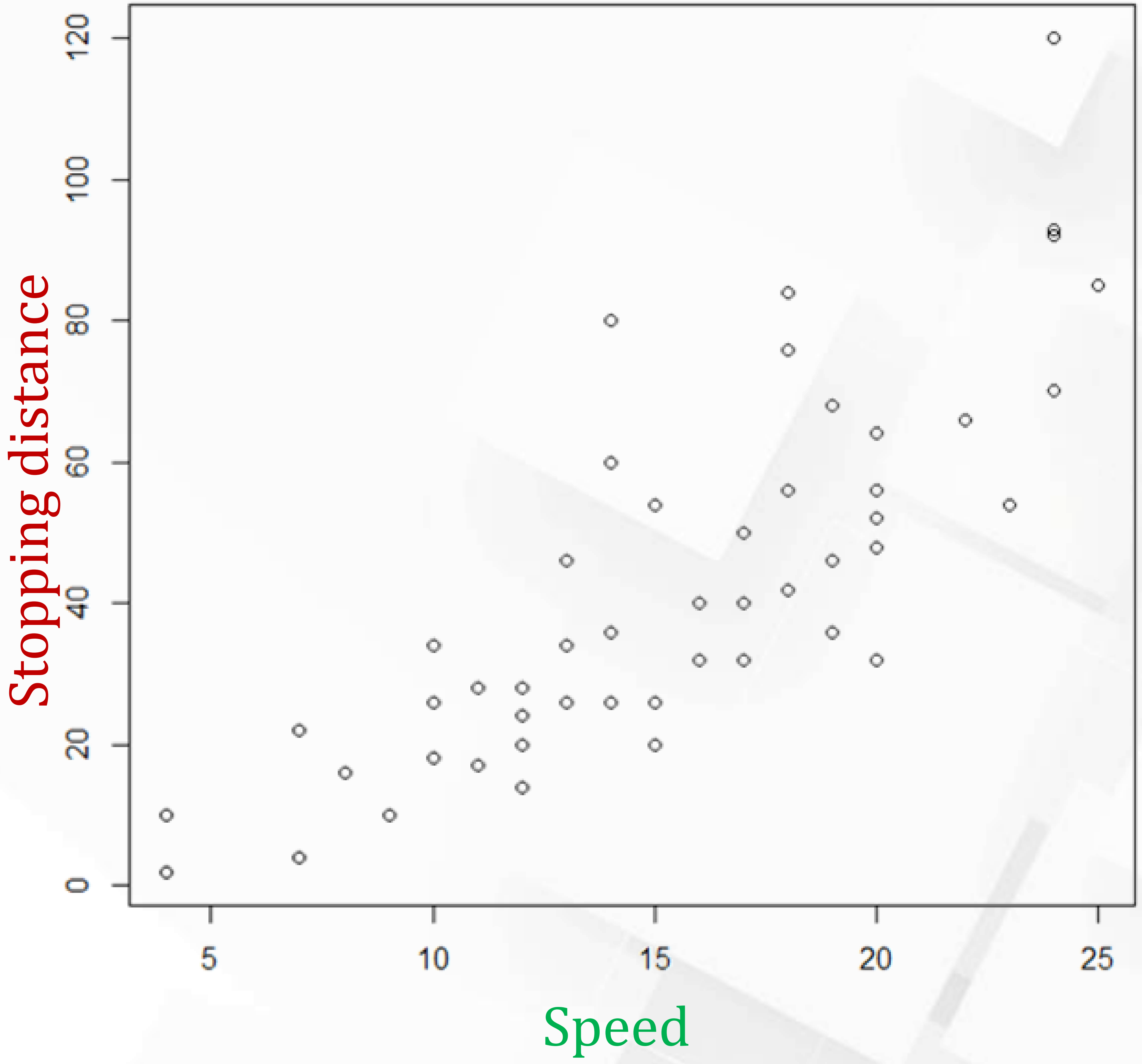
3. Logistic Regression

4. Perceptron

Introduction

- Given 50 pairs of data points for speed(miles per hour mph) vs stopping distance (ft) of cars, that was collected in 1920

- The figure shows the data of the **stopping distance** it takes for cars to fully stop from a certain **speed**

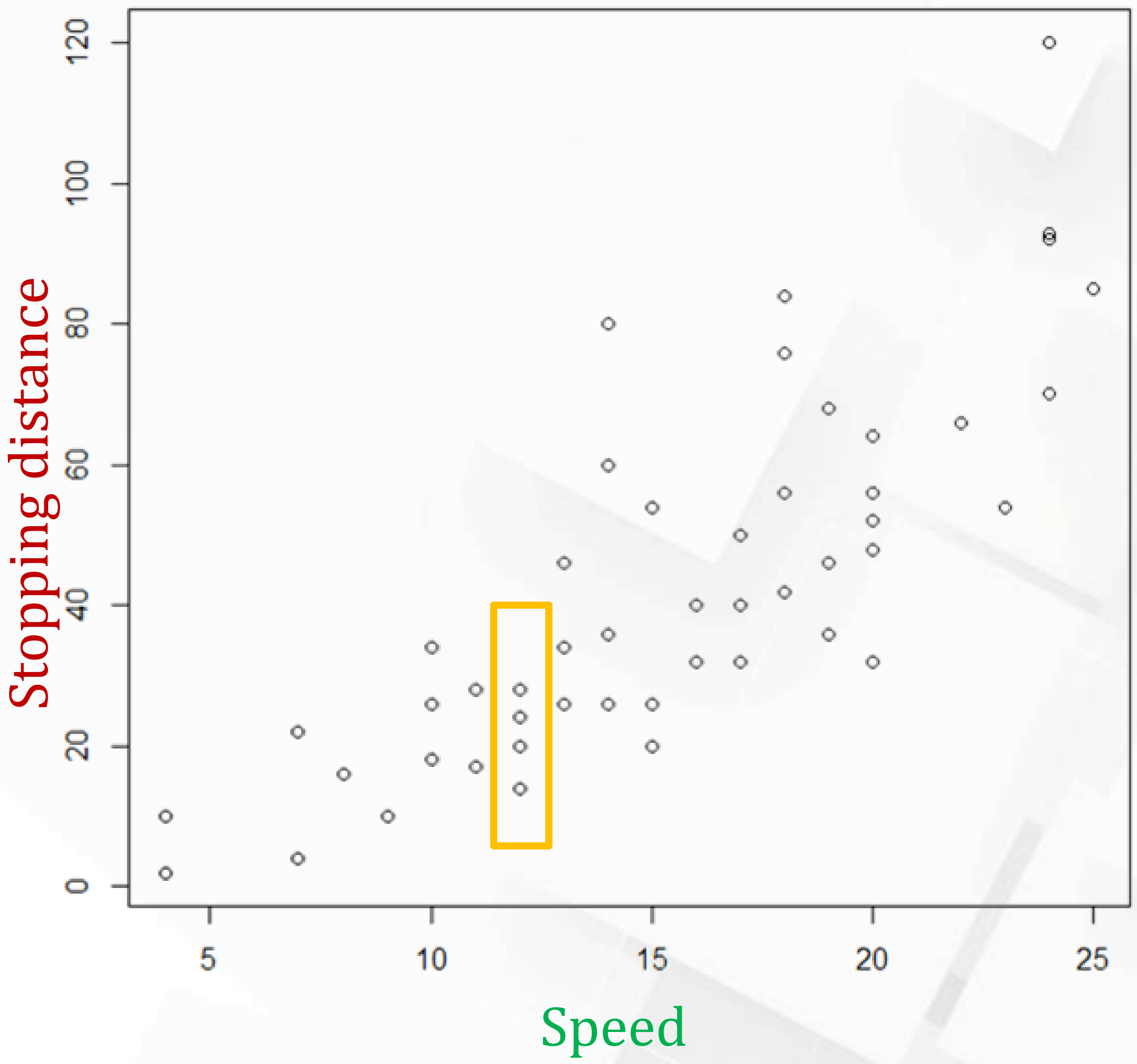


Simple Linear Regression

- Simple regression refers to a model which maps a **linear relationship** between a **singular output** and **input**

- In the orange box, there are multiple stopping distances for the same speed. This could be possible because of different cars, different roads, different weather condition etc.

- For discussion purposes, we only look at the relationship between speed and stopping distance.

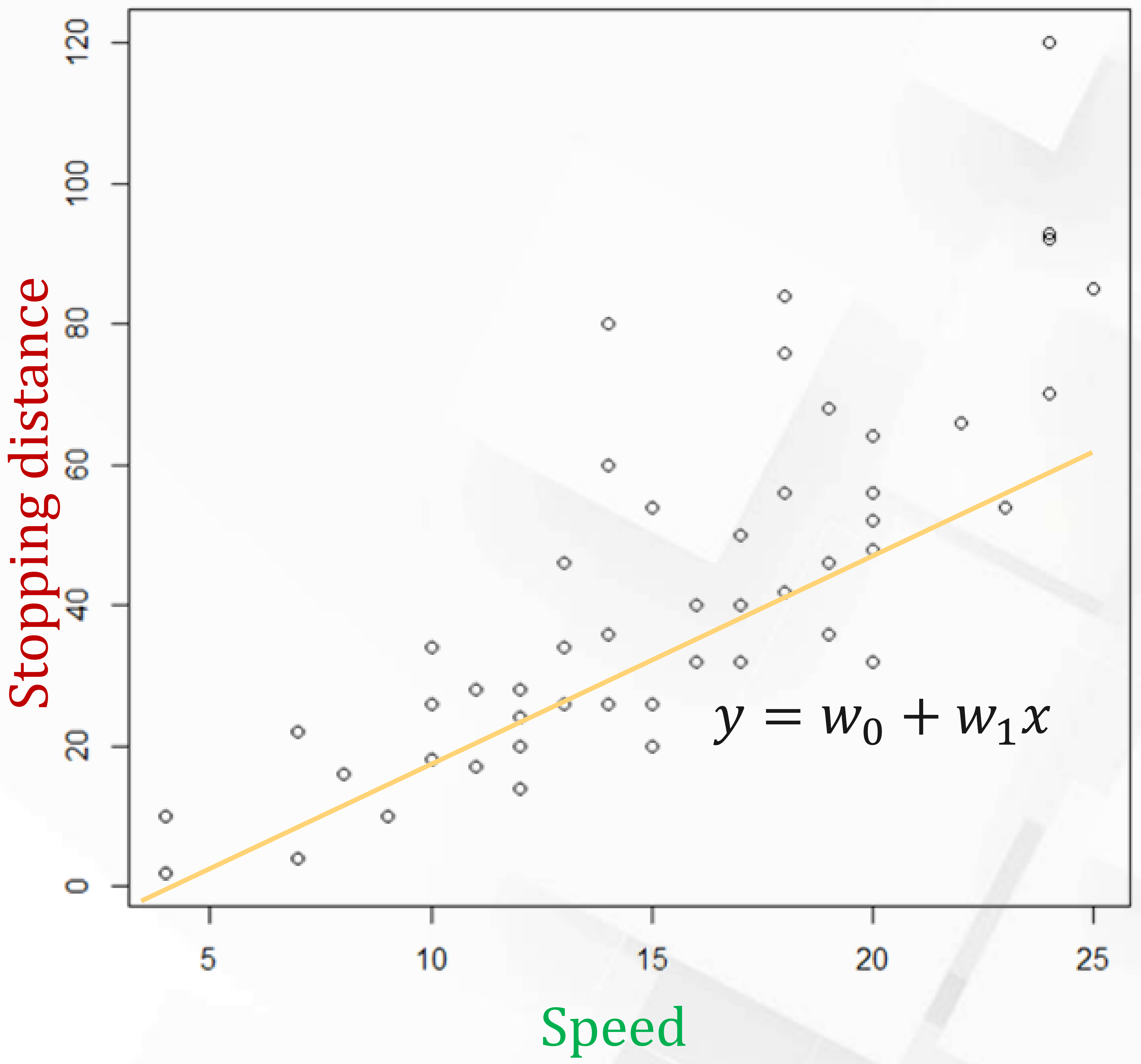


Simple Linear Regression

- Given N number of observed sample pairs, $(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)$
  - (Speed) $x \in \Re \rightarrow$ Predictor/Independent Variable
  - (Stopping distance) $y \in \Re \rightarrow$ Response/Dependent Variable

- **Goal**: find the equation of the line that best fit the dataset,

$$y = \underbrace{w_0}_{\text{Intercept}} + \underbrace{w_1}_{\text{Slope}} x$$



- ✓ In algebra, we call w_1 and w_0 slope and intercept, respectively.
- ✓ In ML, we call them weight and bias.

Multivariate Linear Regression

- Response variable y_i may rely upon various features $\boldsymbol{x}_i$ (*i.e.* As mentioned, different cars, different roads, different weather condition may also affect stopping distance).
- For multivariate linear regression, the function becomes
  - $y = w_0 + w_1 x^{(1)} + w_2 x^{(2)} + \cdots + w_d x^{(d)} = w_0 + \sum_{i=1}^{d} w_i x^{(i)} = \sum_{i=0}^{d} w_i x^{(i)} \qquad (x^{(0)} = 1)$
  - Which can be written in vector form,

$$y_i = \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i, \qquad i = 1, 2, \ldots, N$$

where,

$$\boldsymbol{x}_i = \begin{bmatrix} 1 \\ x_i^{(1)} \\ x_i^{(2)} \\ \vdots \\ x_i^{(d)} \end{bmatrix} \in \Re^{d+1} \qquad \boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_d \end{bmatrix} \in \Re^{d+1}$$

N = number of samples

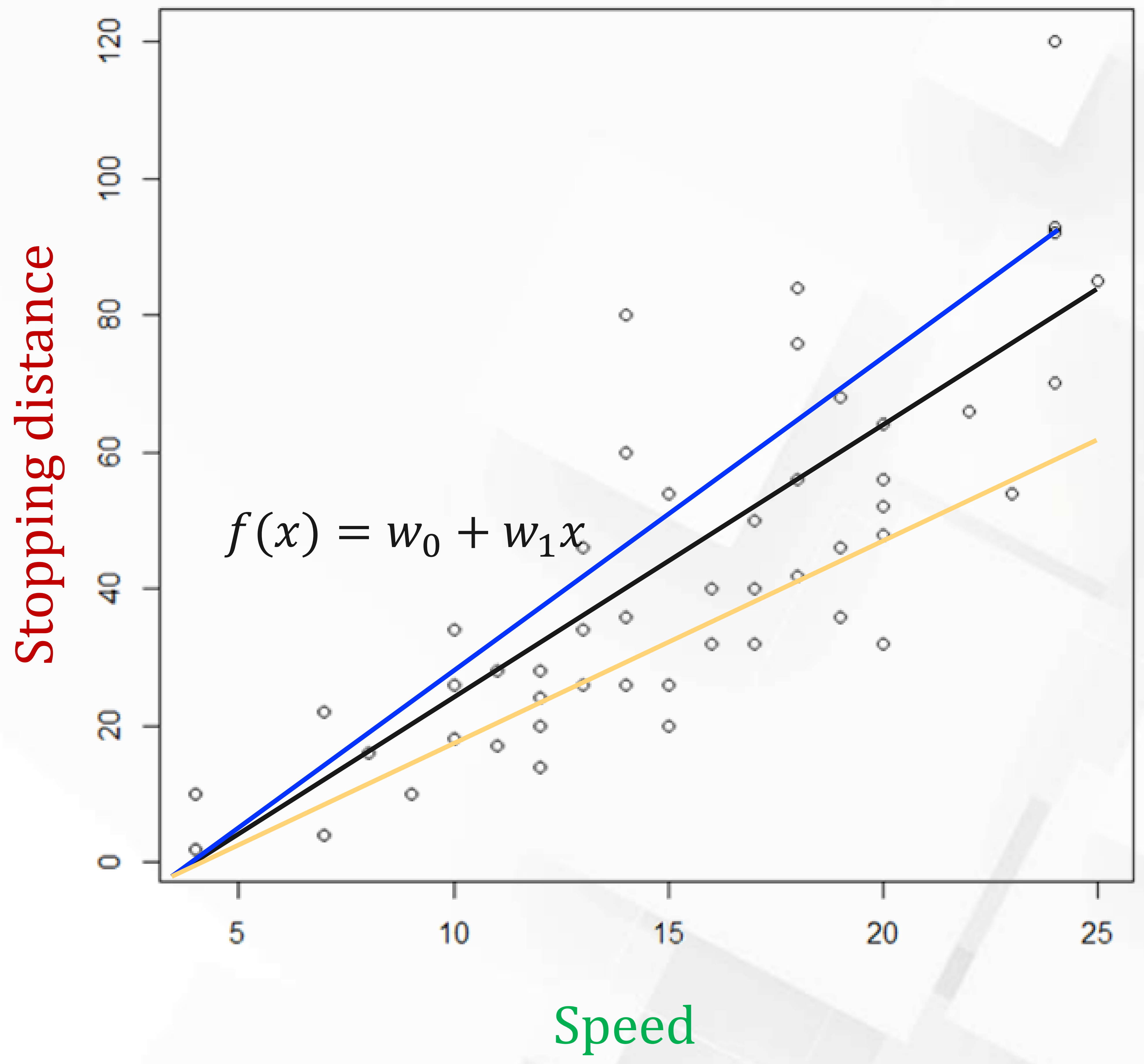
$x_i^{(j)}$

j–th feature

i–th sample

Back to the question

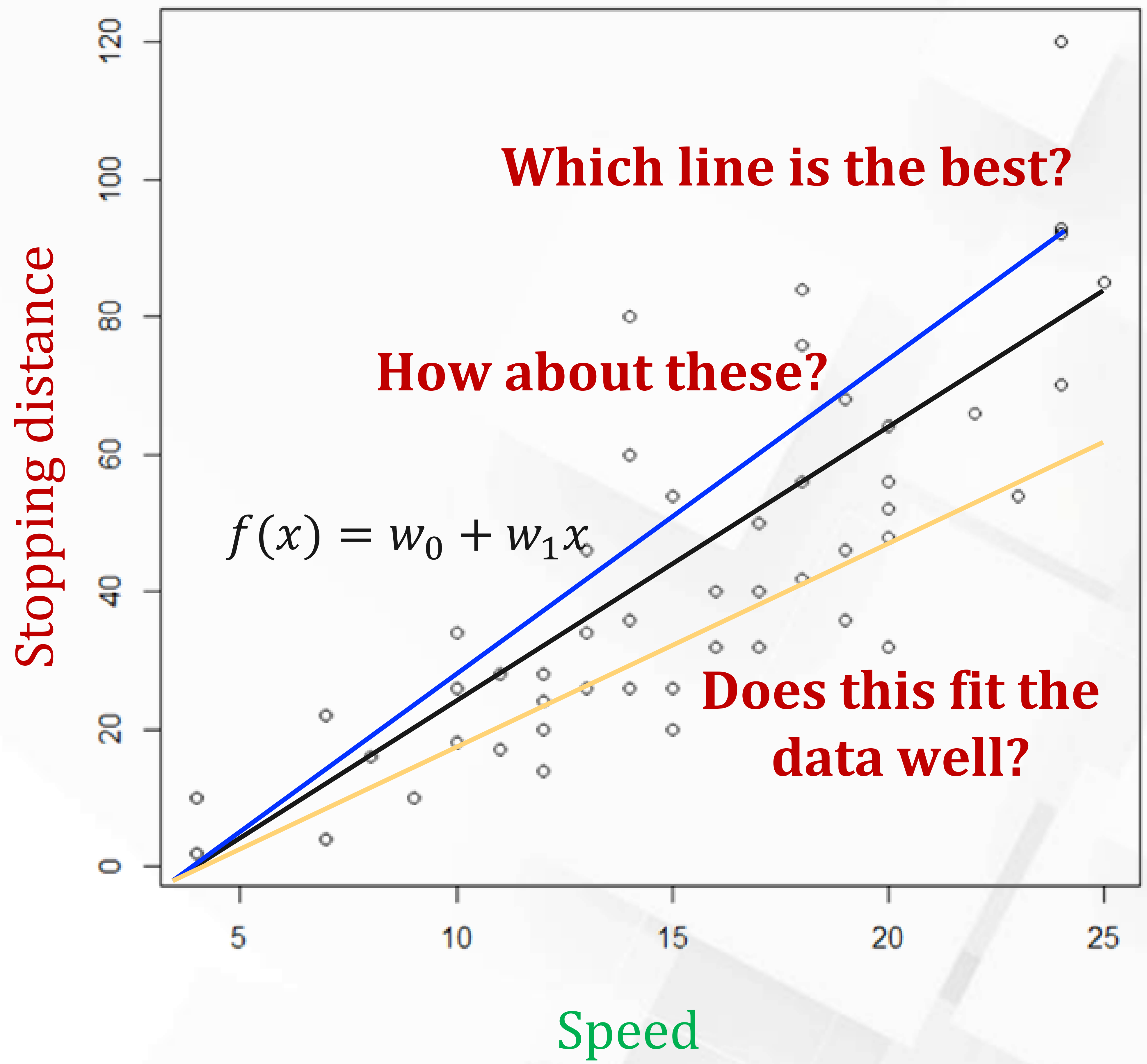
How do we find the parameters for the “best” fitting line?



Linear Regression

We can adjust the values of $\boldsymbol{w}$ to find the equation that gives the best fitting line:

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



Loss Function

- Remember our goal: find the values of $\boldsymbol{w}$ that gives the best fitting line
  - $y = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$
- Best fit (through least squares):
  - Given observations $(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_N, y_N)$

Estimated (or predicted) value for observation i

Estimate weight vector

Predictor vector $\boldsymbol{x}$ for observation i

$$f(\boldsymbol{x}_i) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i$$

  - We can find the best $\boldsymbol{w}^*$ using the Mean Squared Loss:

$$\ell(f(\boldsymbol{x}_i), y_i) = \min_{\boldsymbol{w}} \frac{1}{N} \sum_{j=1}^{N} (f(\boldsymbol{x}_i) - y_i)^2$$

Loss Function

$$\ell(f(\boldsymbol{x}_i), y_i) = \min_{\boldsymbol{w}} \frac{1}{N} \sum_{j=1}^{N} (f(\boldsymbol{x}_i) - y_i)^2$$

$$= \min_{\boldsymbol{w}} \frac{1}{N} \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|^2$$

$$\boxed{= \min_{\boldsymbol{w}} \frac{1}{N} (\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})^{\mathrm{T}} (\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})}$$

where $\boldsymbol{X} = \begin{bmatrix} - \boldsymbol{x}_1^{\mathrm{T}} - \\ - \boldsymbol{x}_2^{\mathrm{T}} - \\ \vdots \\ - \boldsymbol{x}_N^{\mathrm{T}} - \end{bmatrix}$ and $\boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$

Optimization - Minimum Mean Squared Error

Thus, find the values of $\boldsymbol{w}$ that minimizes the $\ell(f(\boldsymbol{x}_i), y_i)$

$$\ell(f(\boldsymbol{x}_i), y_i) = \min_{\boldsymbol{w}} \frac{1}{N} (\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})^{\mathrm{T}} (\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})$$

$\boldsymbol{w}^*$ must satisfy:

$$\frac{\partial\, \ell(f(\boldsymbol{x}_i), y_i)}{\partial \boldsymbol{w}} = \frac{2}{N} \boldsymbol{X}^{\mathrm{T}} (\boldsymbol{X}\boldsymbol{w}^* - \boldsymbol{y}) = 0$$

$$\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{w}^* = \boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}$$

If $(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})$ is inversible, we get

$$\boxed{\boldsymbol{w}^* = (\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})^{-1} \boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}}$$

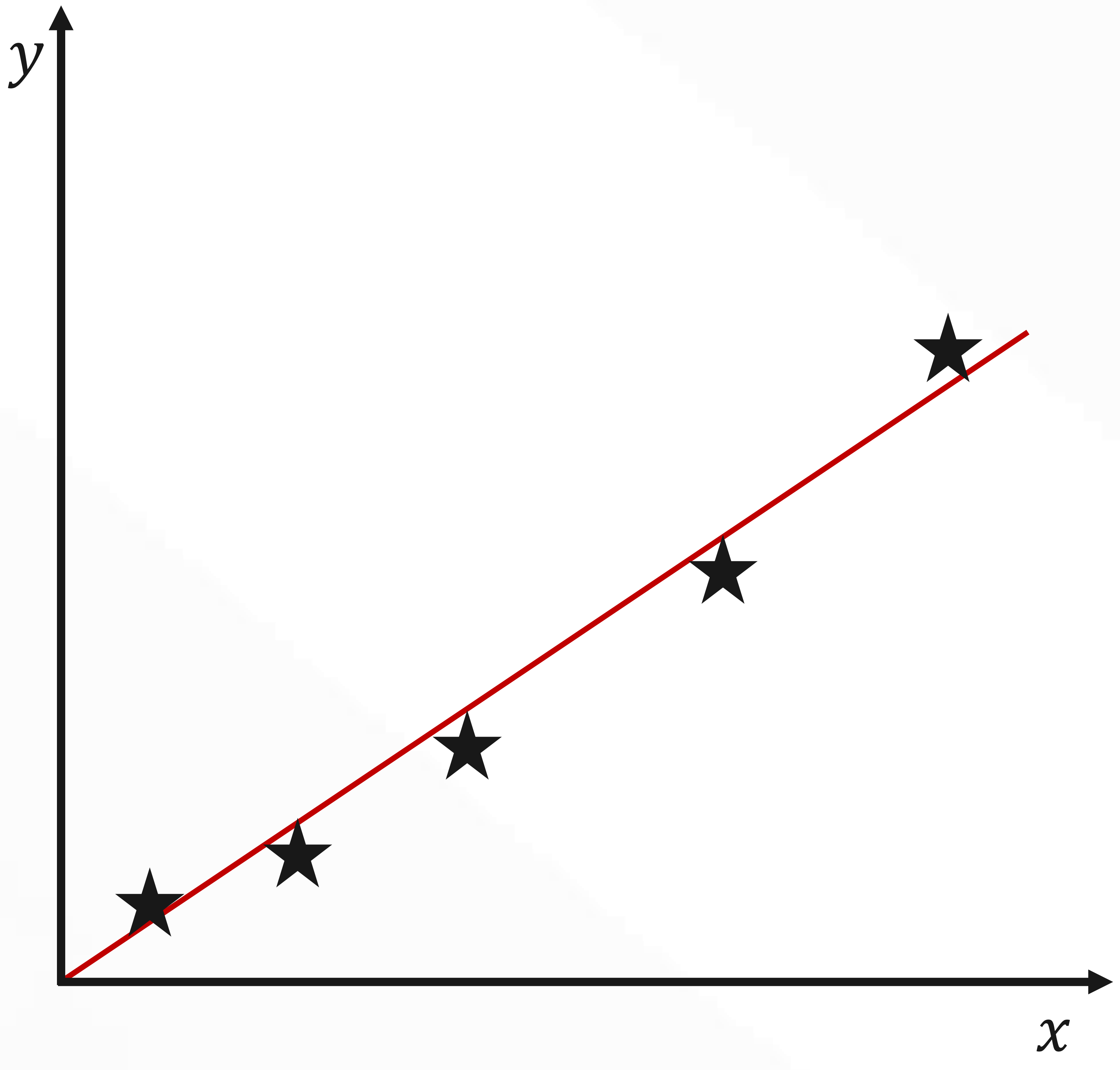
Ridge Regression

$$\boldsymbol{w}^* = (\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X} + \lambda \boldsymbol{I})^{-1} \boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}$$

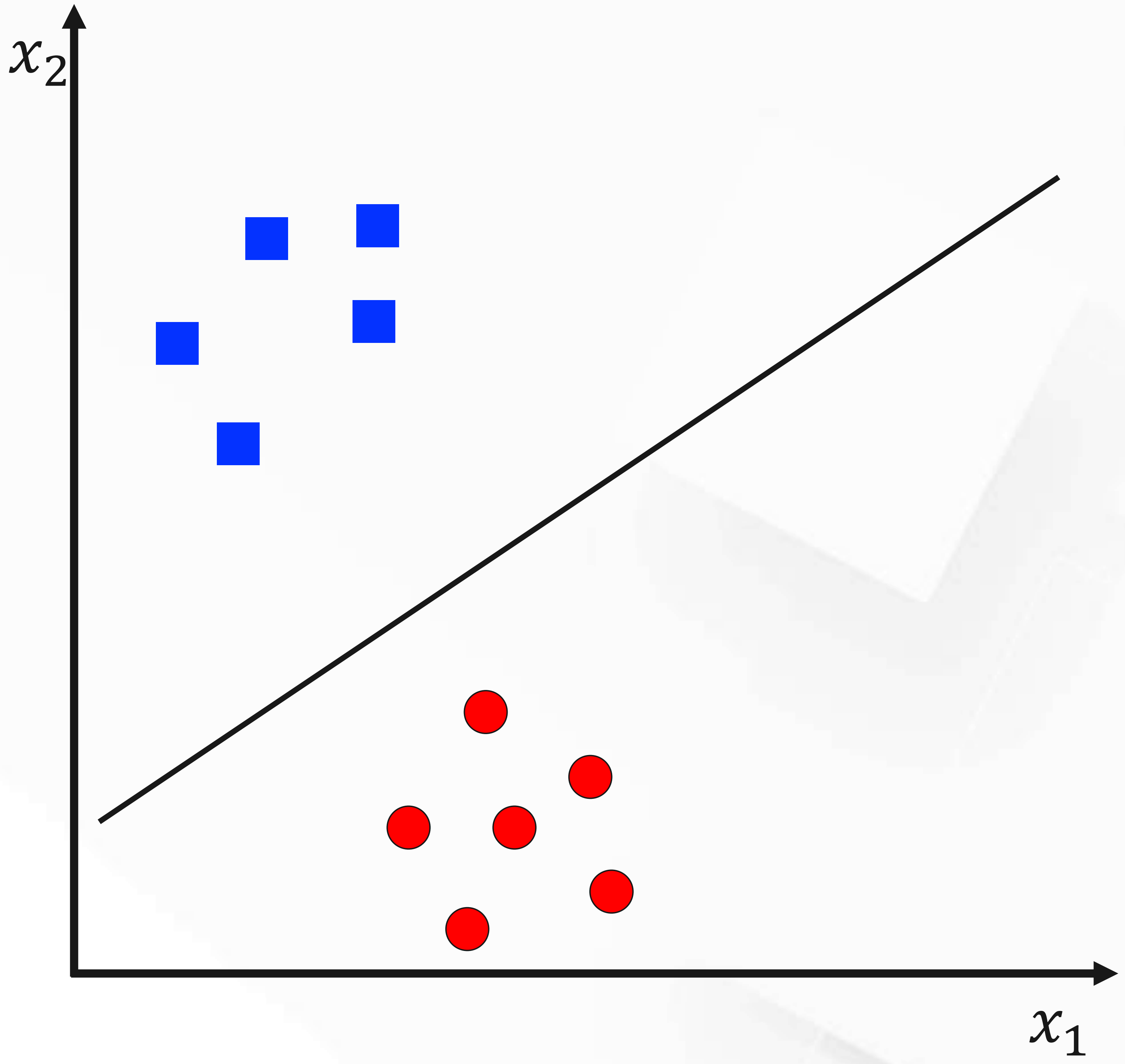
Linear Model Outline

1. Linear Regression
2. Linear Discriminant Analysis
3. Logistic Regression
4. Perceptron

Linear Regression vs Linear Classification



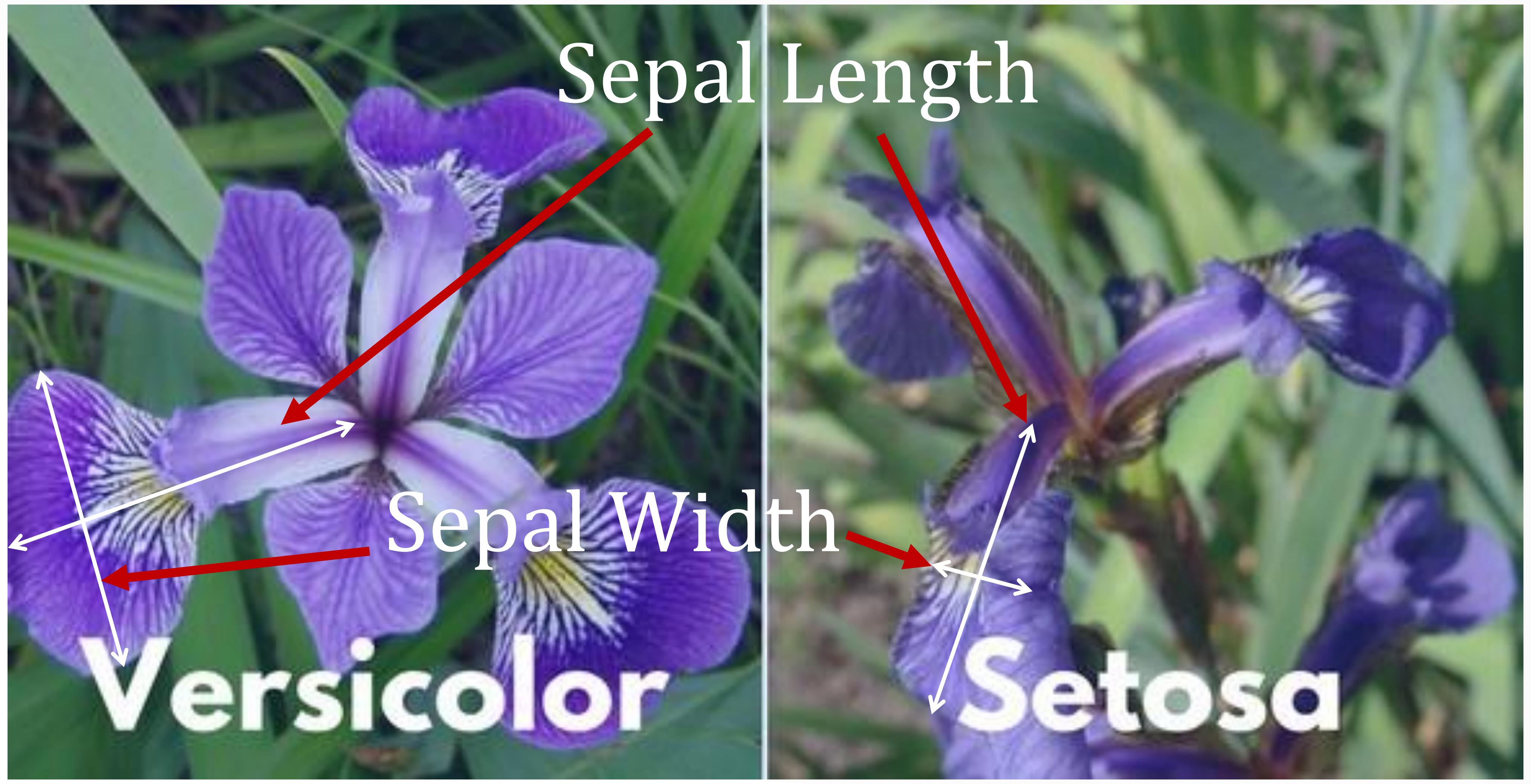
Linear Fit



Linear Decision Boundary

Linear Discriminant Analysis – The Problem

变色鸢尾

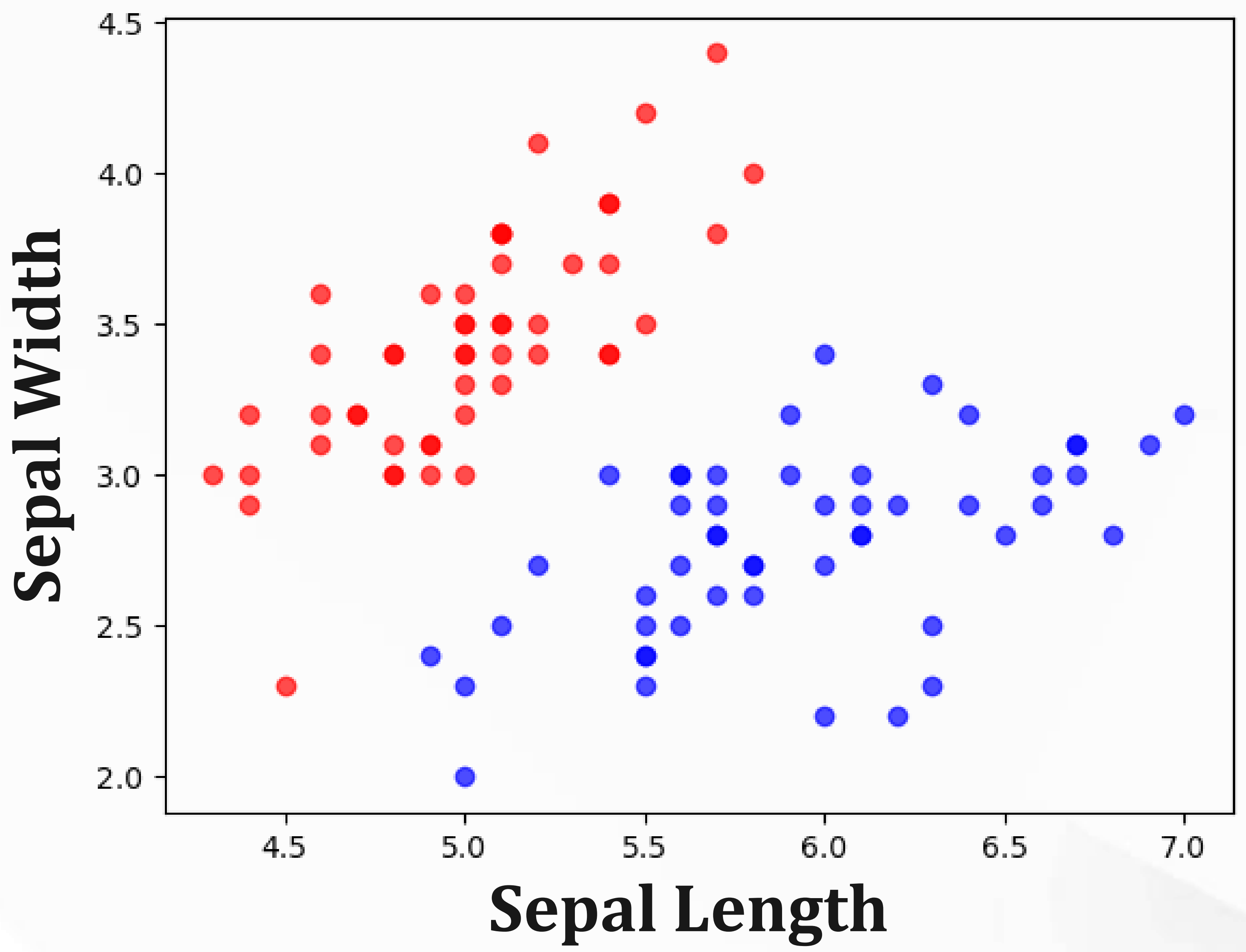


山鸢尾

- ☐ Let's say we got a set of Iris pictures…
  - ➢ We want to know which species does each of them belong.
  - ➢ **What do you observe?**
    - ❖ **The width and length of their sepals (萼片) seems different!**

Linear Discriminant Analysis – Formulation

Sepal Length	Sepal Width	Species
5.1	3.5	setosa
4.9	3.0	setosa
7.0	3.2	versicolor
6.4	3.2	versicolor
…	…	…

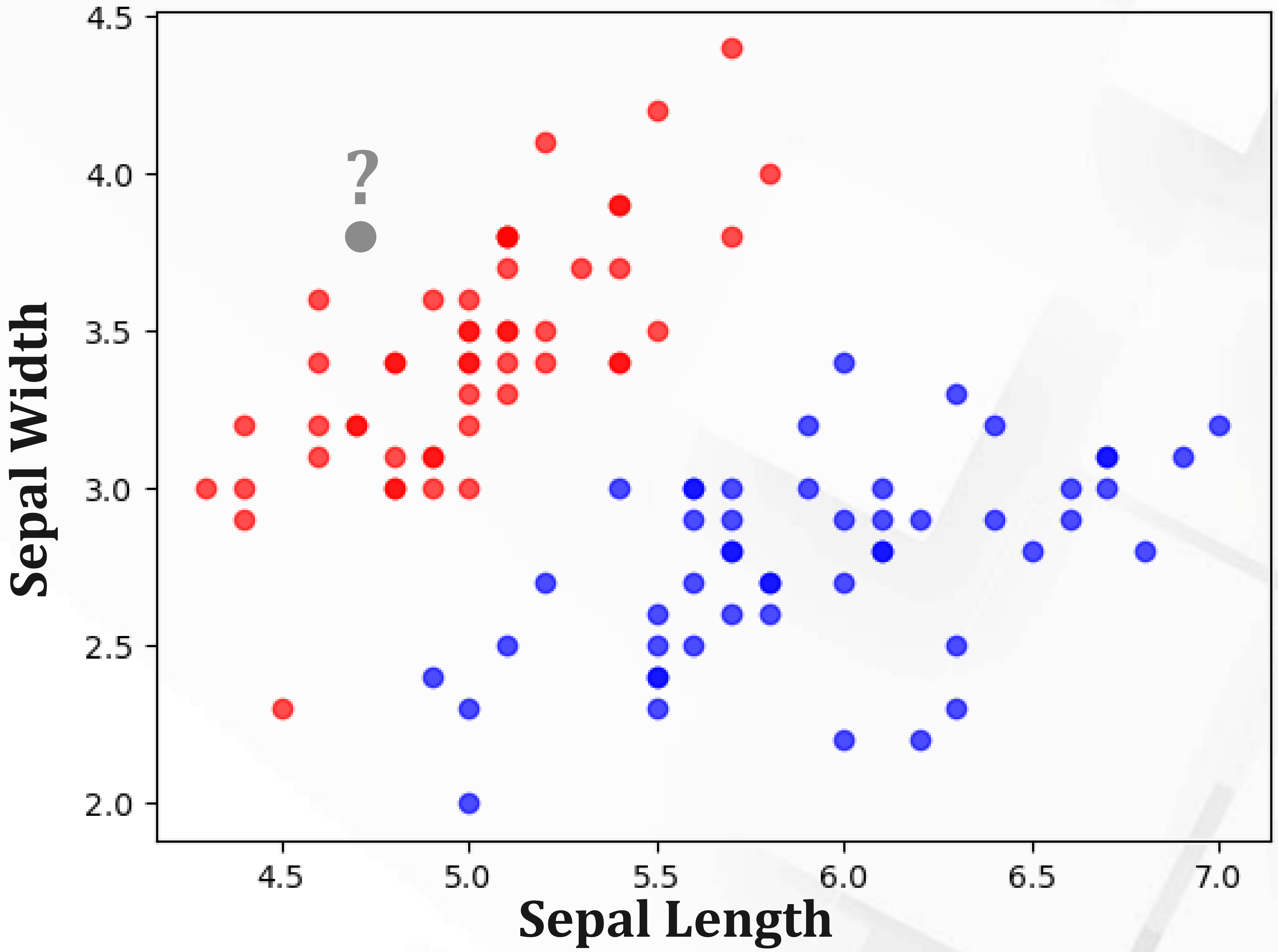


- Let's say we have measurements for 150 Iris flowers from 2 different species
- We can graph the data points with Sepal Length as horizontal axis and Sepal Width as vertical axis.
- It can be seen from the figure that the data is separated into 2 regions with red points representing **Setosa** and Blue points representing **Versicolor**

Linear Discriminant Analysis – Formulation

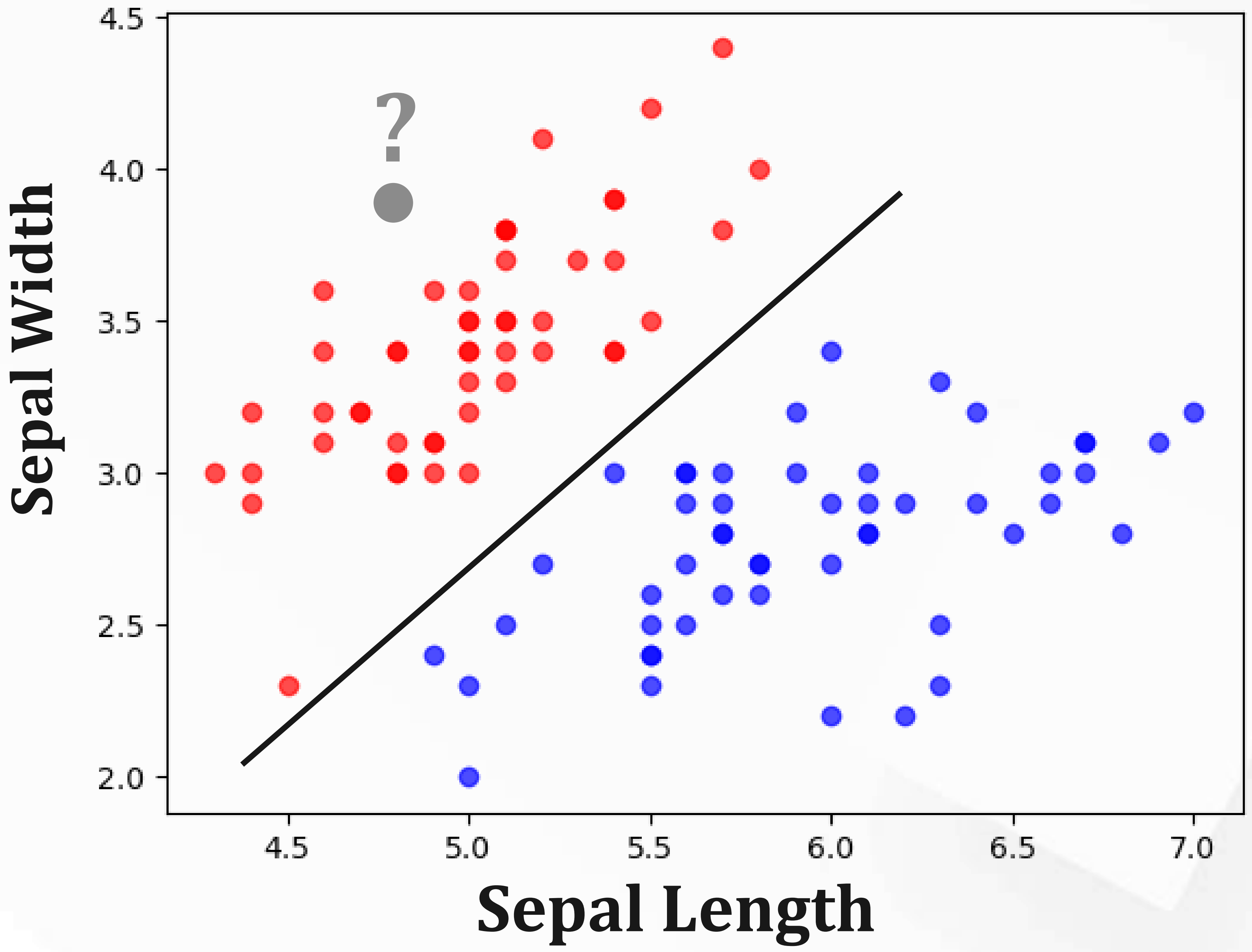
What if we are given new measurements for an unknown Iris flower?

Sepal Length	Sepal Width	Species
4.75	3.65	?



Graphical Approach

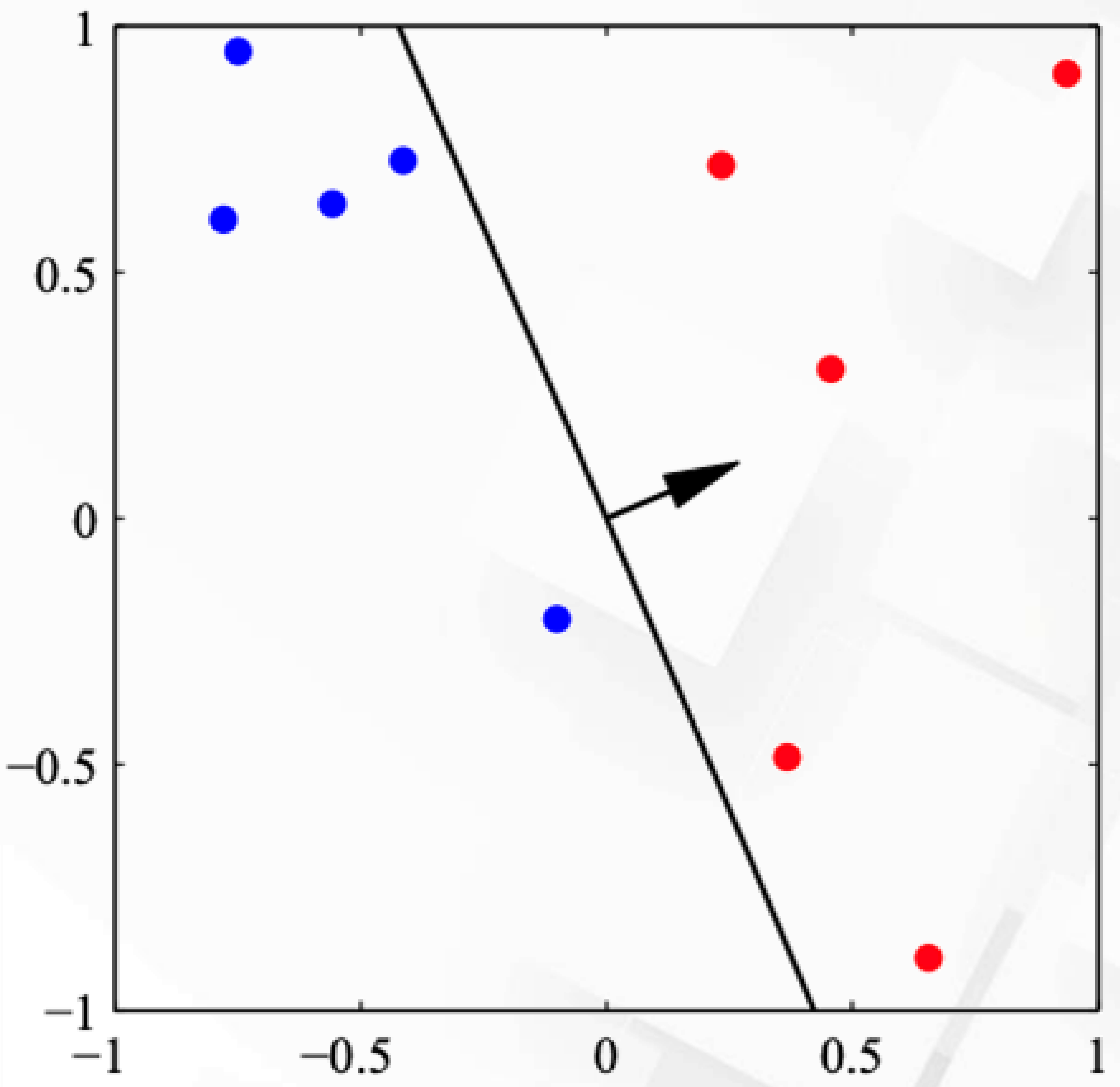
Sepal Length	Sepal Width	Species
4.75	3.65	?



- **Problem**: The species is unknown
- **Objective**: Given a new of these features, predict the Iris species.
- **Approach**:
  - Draw line that separates the 2 species
  - Graph the new data point on the figure
- It can be easily seen that the new data point belong to the Setosa species

Mathematical Approach

- Linear Classification Model – i.e., decision boundary is a linear function of $\boldsymbol{x}$
  - The data is called **linearly separable** if it can be separated exactly by linear decision boundary/surfaces



From PRML (Bishop, 2006)

Mathematical Approach

- With a linear discriminant function, we have

$$g(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + w_0$$

where w_0 is a constant

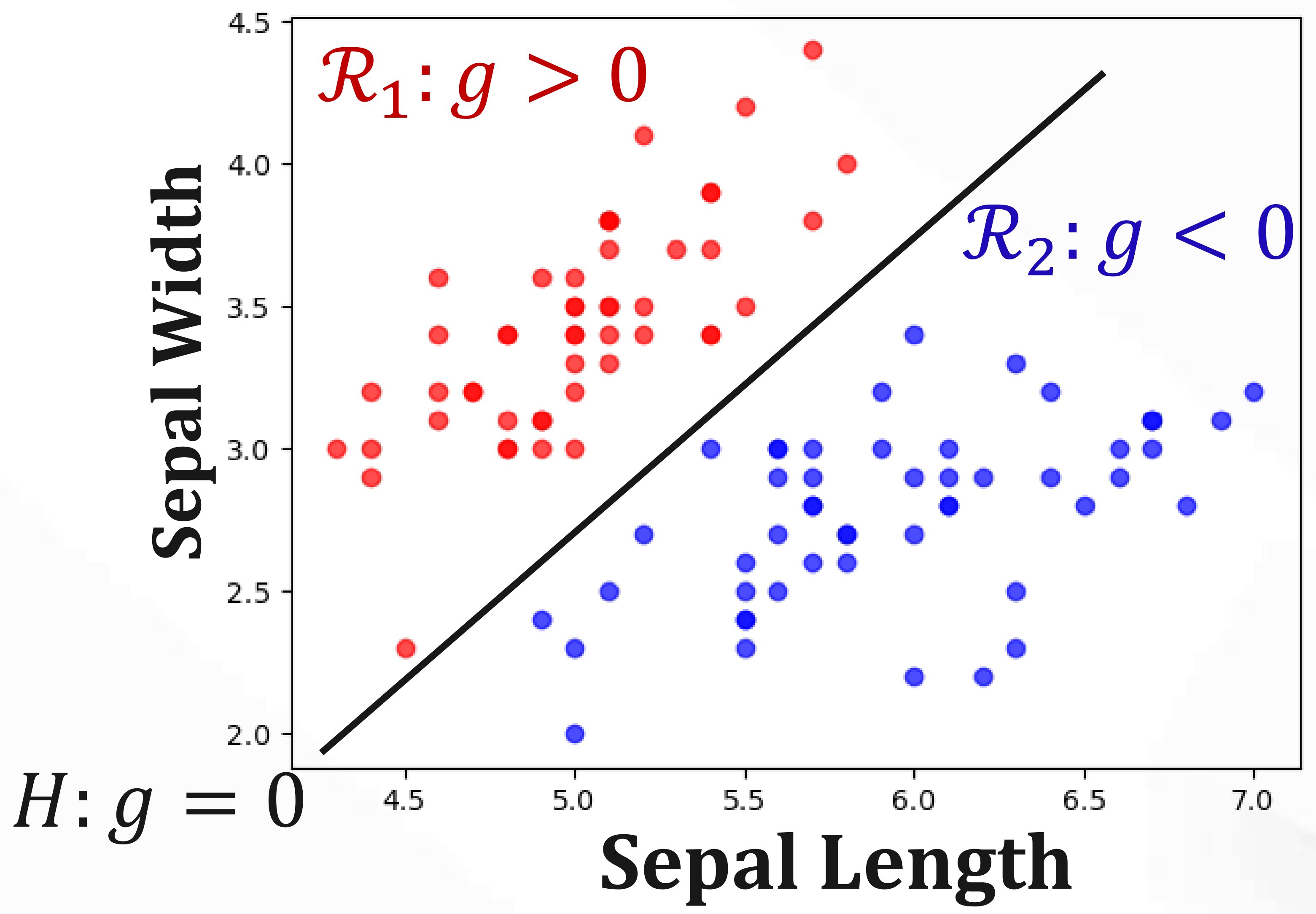
- For a binary (2-class) classification task, the decision function is given by

$$\begin{cases} g(\boldsymbol{x}) > 0, \boldsymbol{x} \in \omega_1 \\ g(\boldsymbol{x}) < 0, \boldsymbol{x} \in \omega_2 \\ g(\boldsymbol{x}) = 0, \boldsymbol{x} \text{ can be } \omega_1 \text{ or } \omega_2 \end{cases}$$

Decision Surface and Decision Region

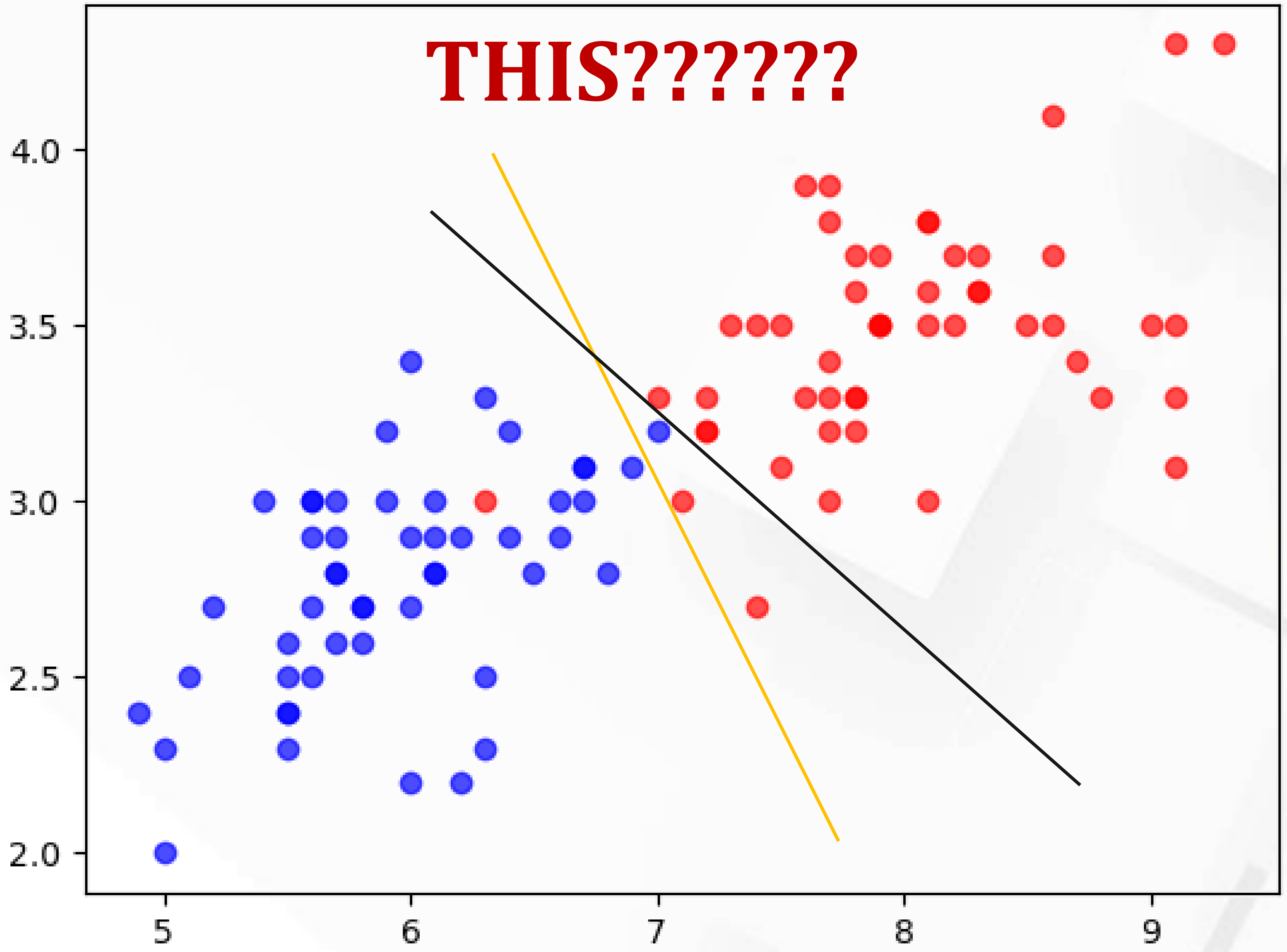
$$\begin{cases} g(\boldsymbol{x}) > 0, \boldsymbol{x} \in \omega_1 \\ g(\boldsymbol{x}) < 0, \boldsymbol{x} \in \omega_2 \\ g(\boldsymbol{x}) = 0, \boldsymbol{x} \text{ can be } \omega_1 \text{ or } \omega_2 \end{cases}$$

- $H: g(\boldsymbol{x}) = 0$ acts as a decision surface that separates samples from class ω_1 and ω_2
- $\mathcal{R}_1: g(\boldsymbol{x}) > 0$ is the decision region that contains samples from class ω_1
- $\mathcal{R}_2: g(\boldsymbol{x}) < 0$ is the decision region that contains samples from class ω_2

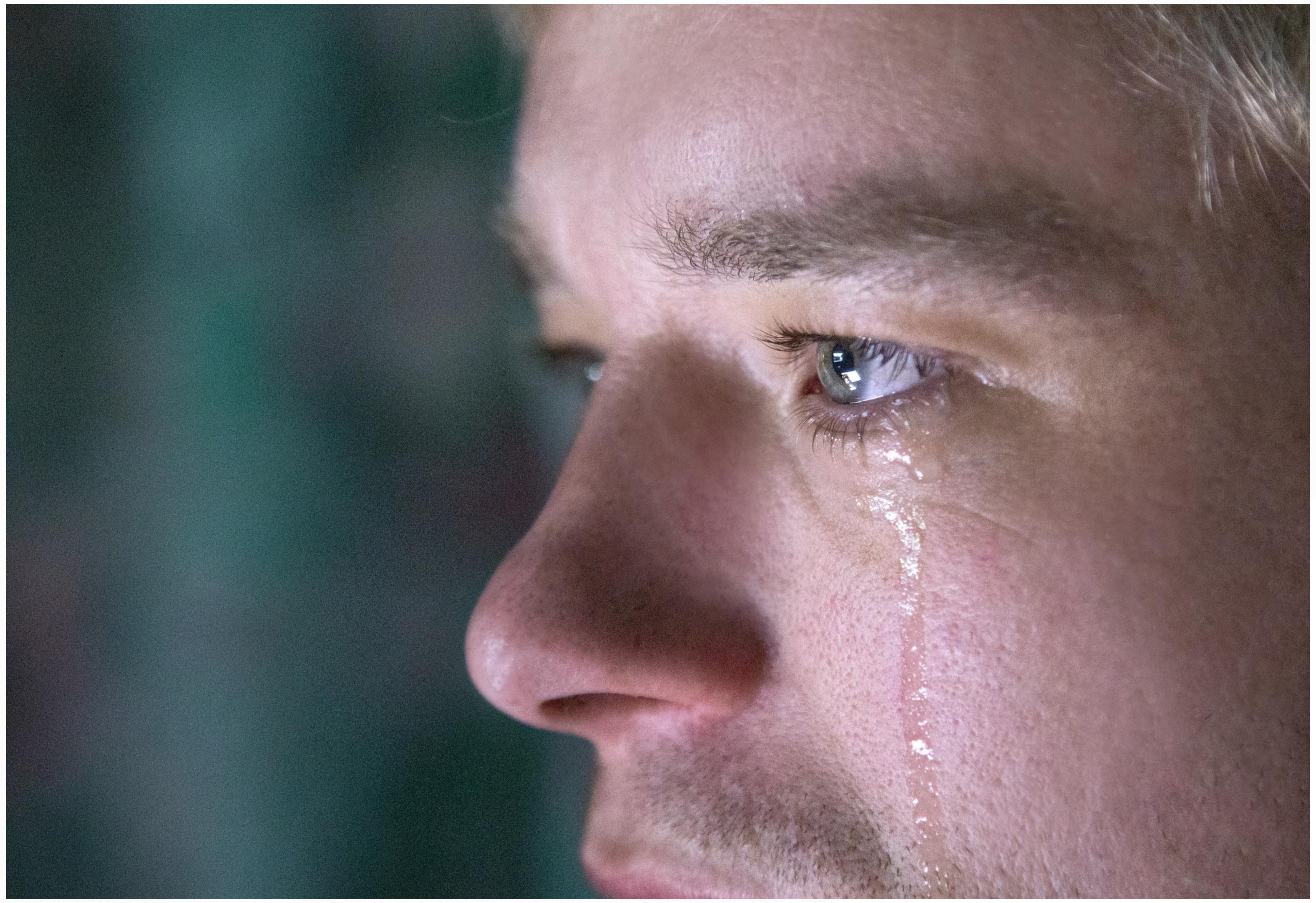


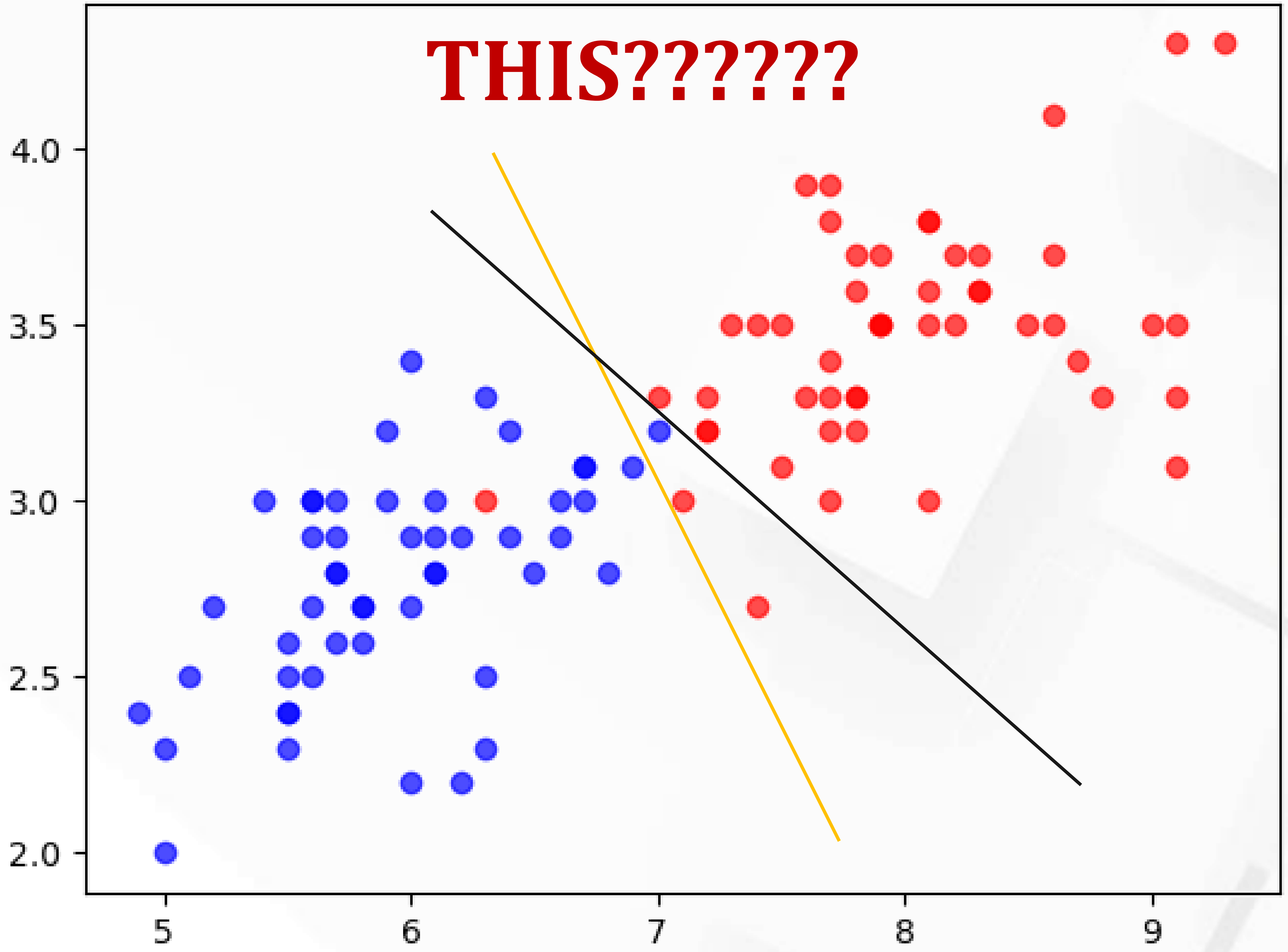
Not Linearly Separable Examples

What if the data is like…



Not Linearly Separable Examples

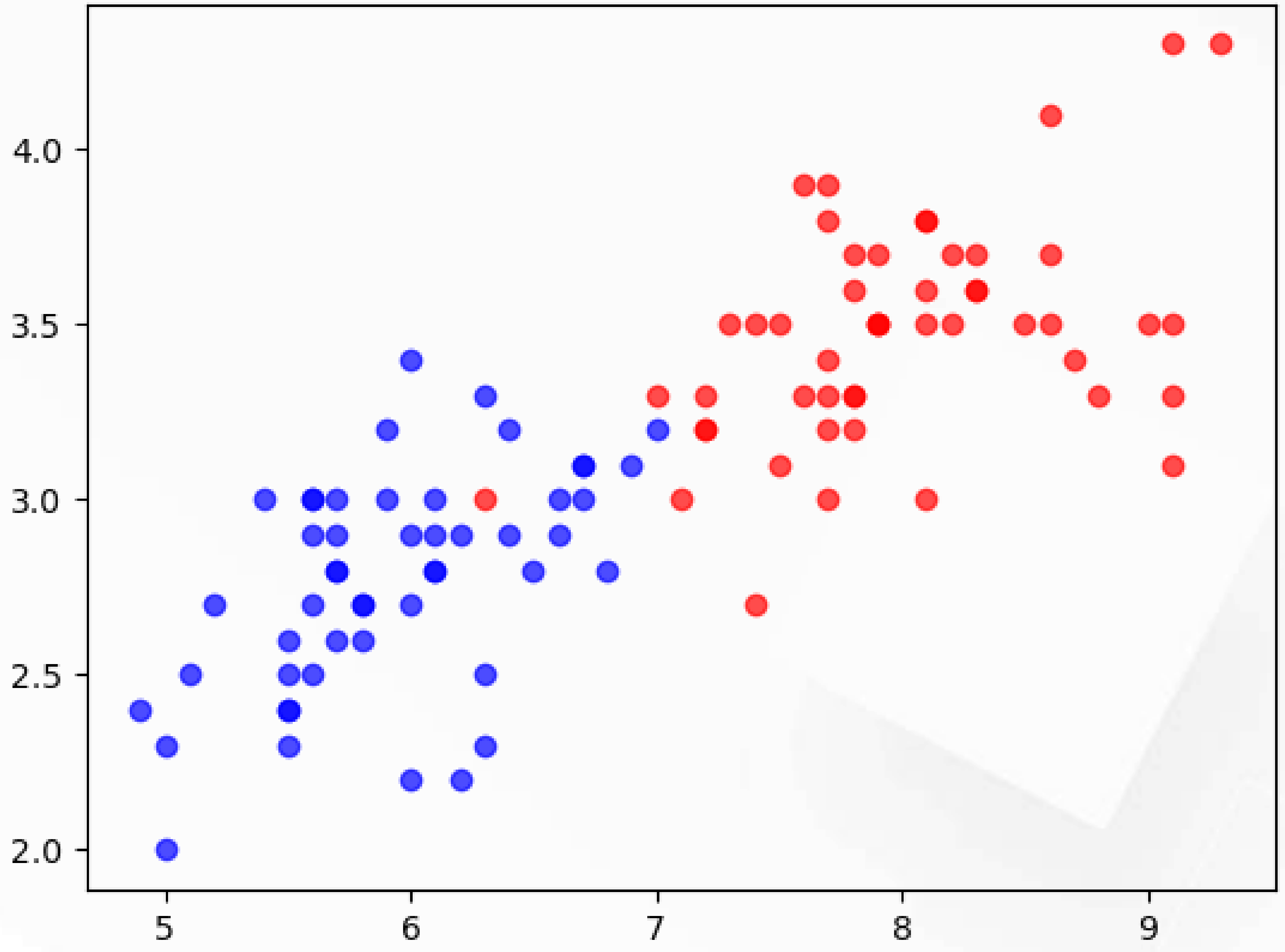




How do we decide?

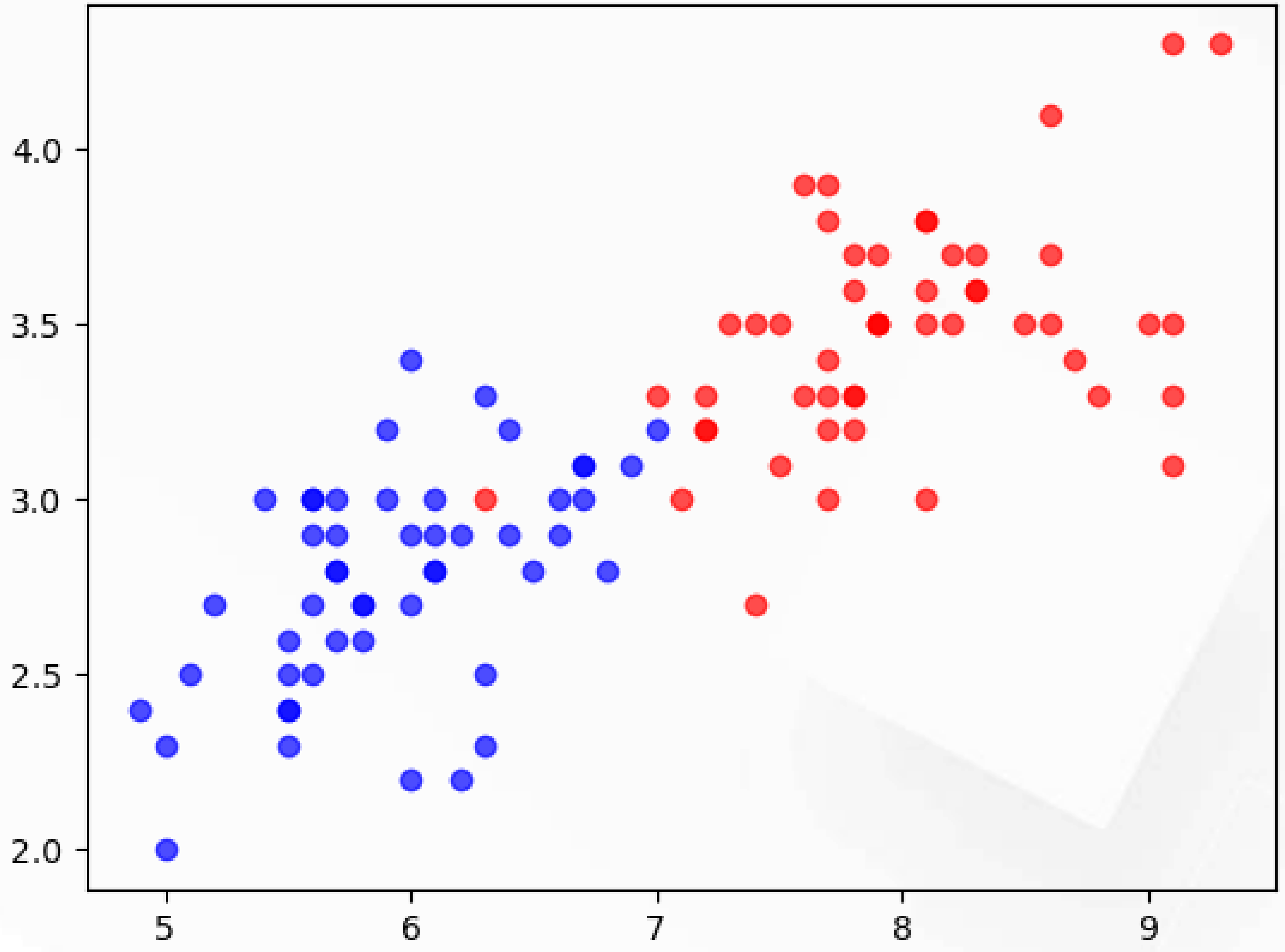
Not Linearly Separable Examples

维吉尼亚鸢尾



- Let's assume we have new Iris flowers species called virginica
- We graph the flower sepal length and width from versicolor and virginica

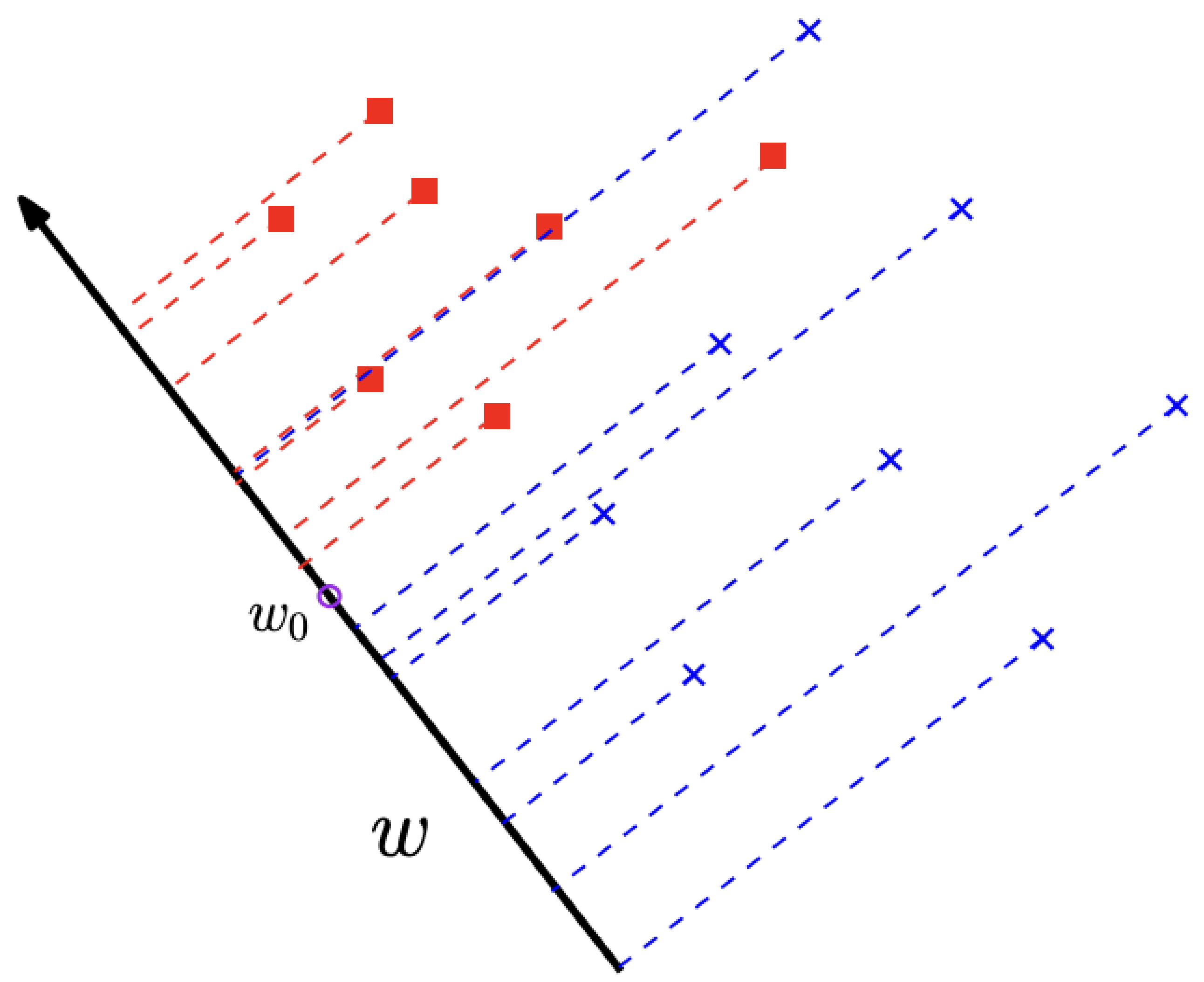
Not Linearly Separable Examples



- **Objective**: Given a new measurement of these features, predict the Iris species.
- **Problem**: The features are not linearly separable
- **Approach**: We may estimate the Iris species mathematically based on a projection onto a lower-dimensional space.

Linear Discriminant Analysis

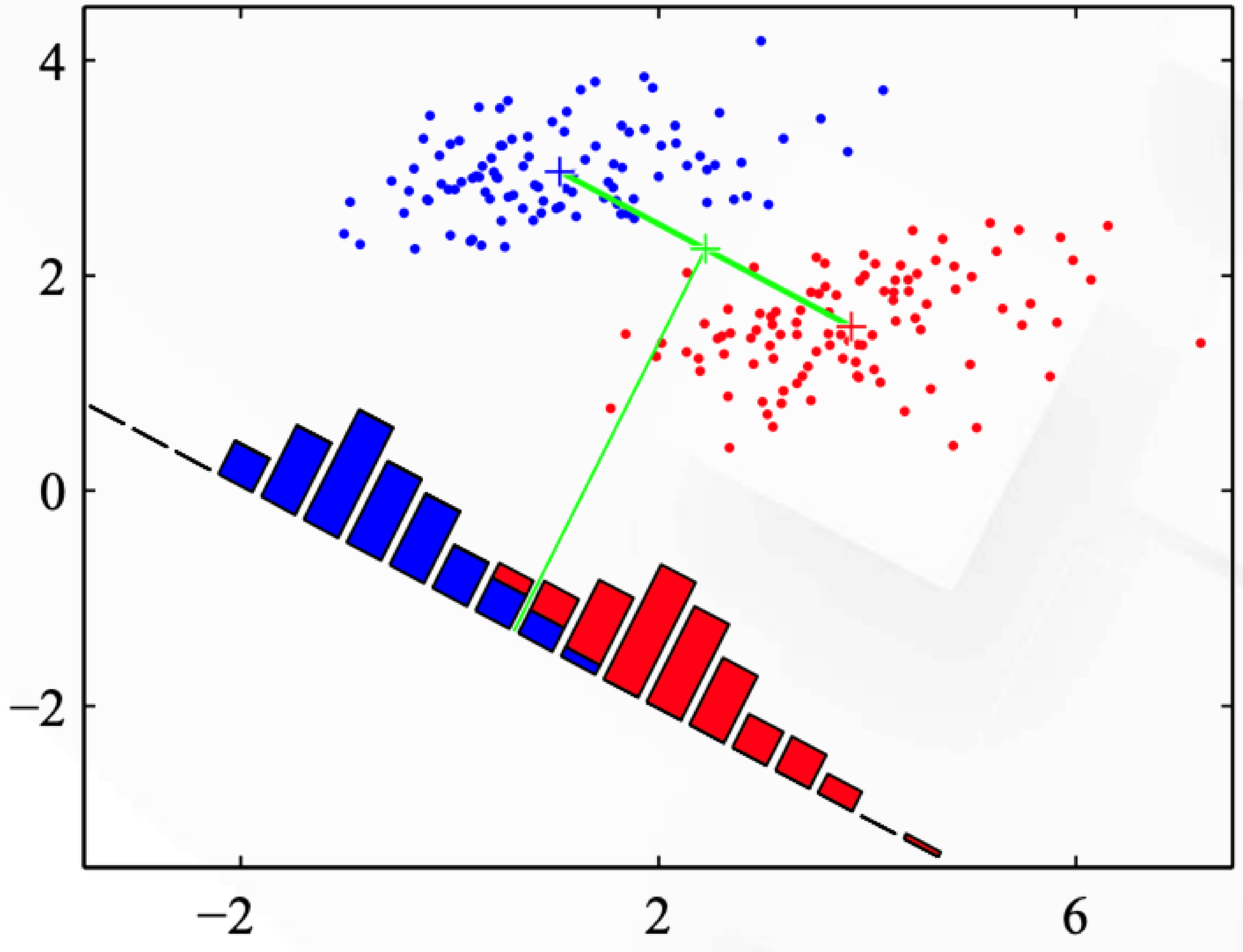
- Classification as Project



- Orthogonal Projections: project input vector $\boldsymbol{x} \in \Re^{d+1}$ down to a 1-dimensional subspace with projection vector $\boldsymbol{w}$
- Assume we know the projection vector $\boldsymbol{w}$, we can compute the projection of any point $\boldsymbol{x} \in \Re^{d+1}$ onto the one-dimensional subspace spanned by $\boldsymbol{w}$

Potential Problems

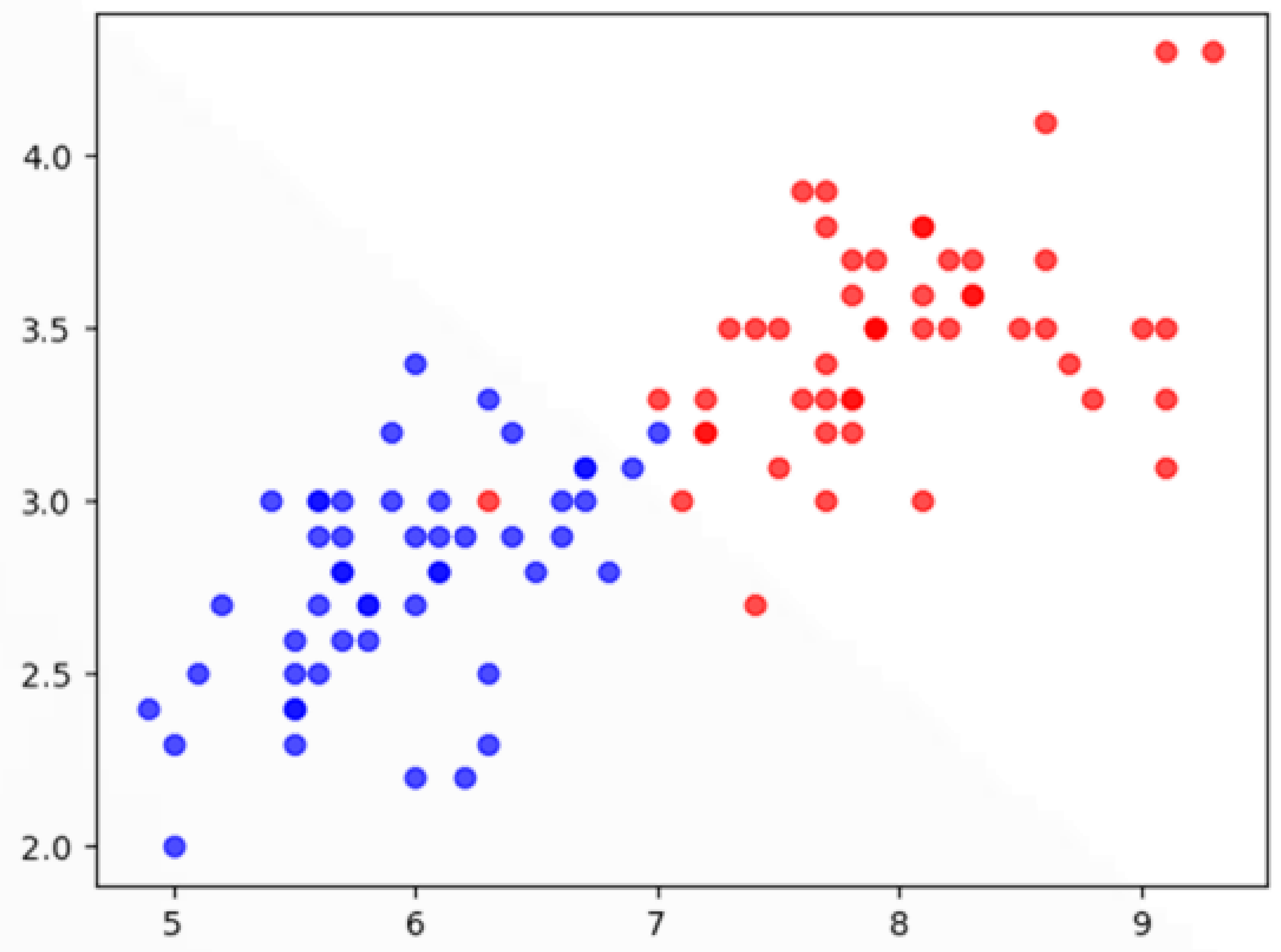
1. Considerable loss of information when projecting
2. Even if data was linearly separable in $\Re^{d+1}$, we may lose this separability
   - **Approach**: Find good projection vector $\boldsymbol{w}$ that spans the subspace we project into

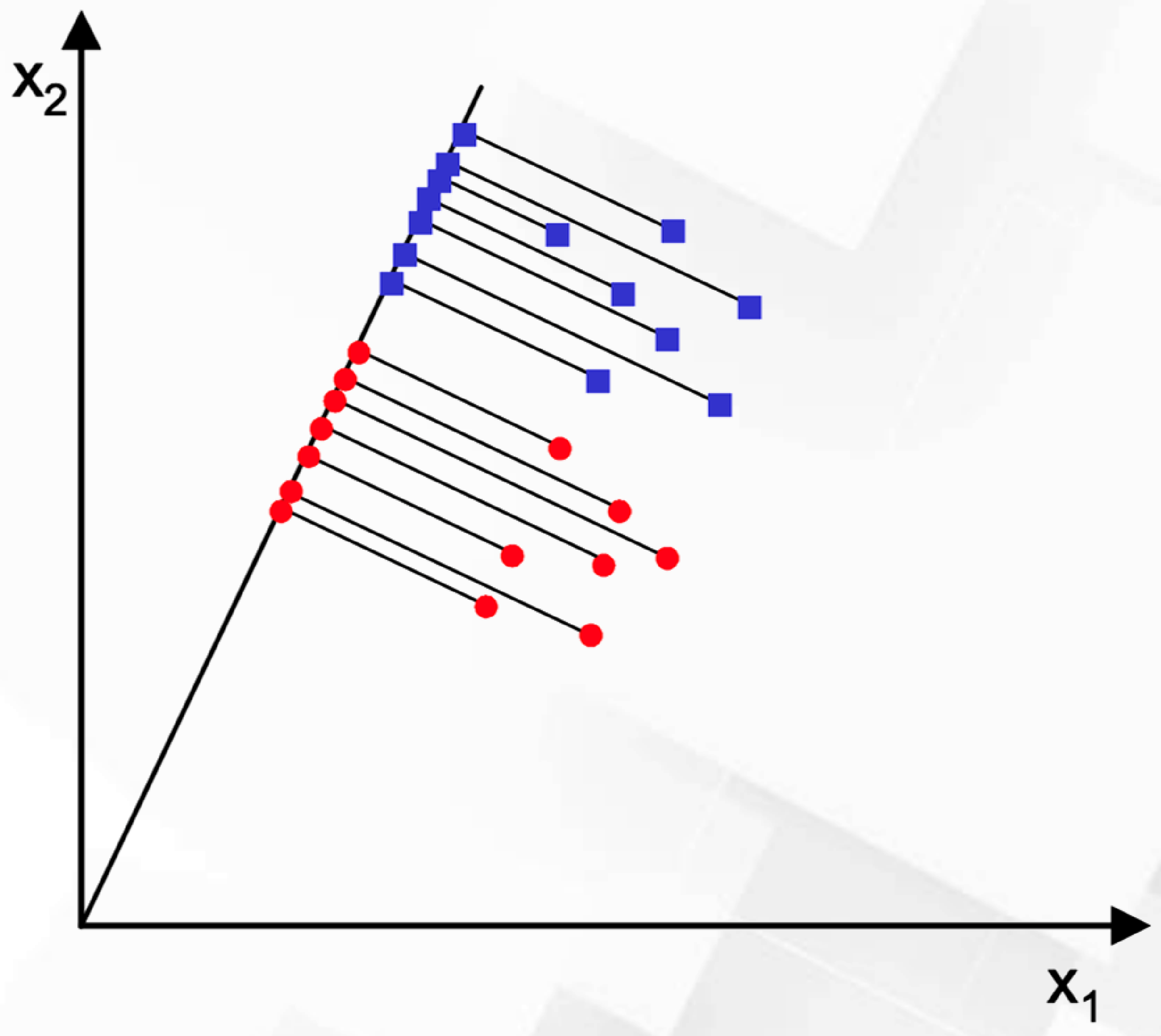


From PRML (Bishop, 2006)

Linear Discriminant Analysis

How do we find the good projection vector?



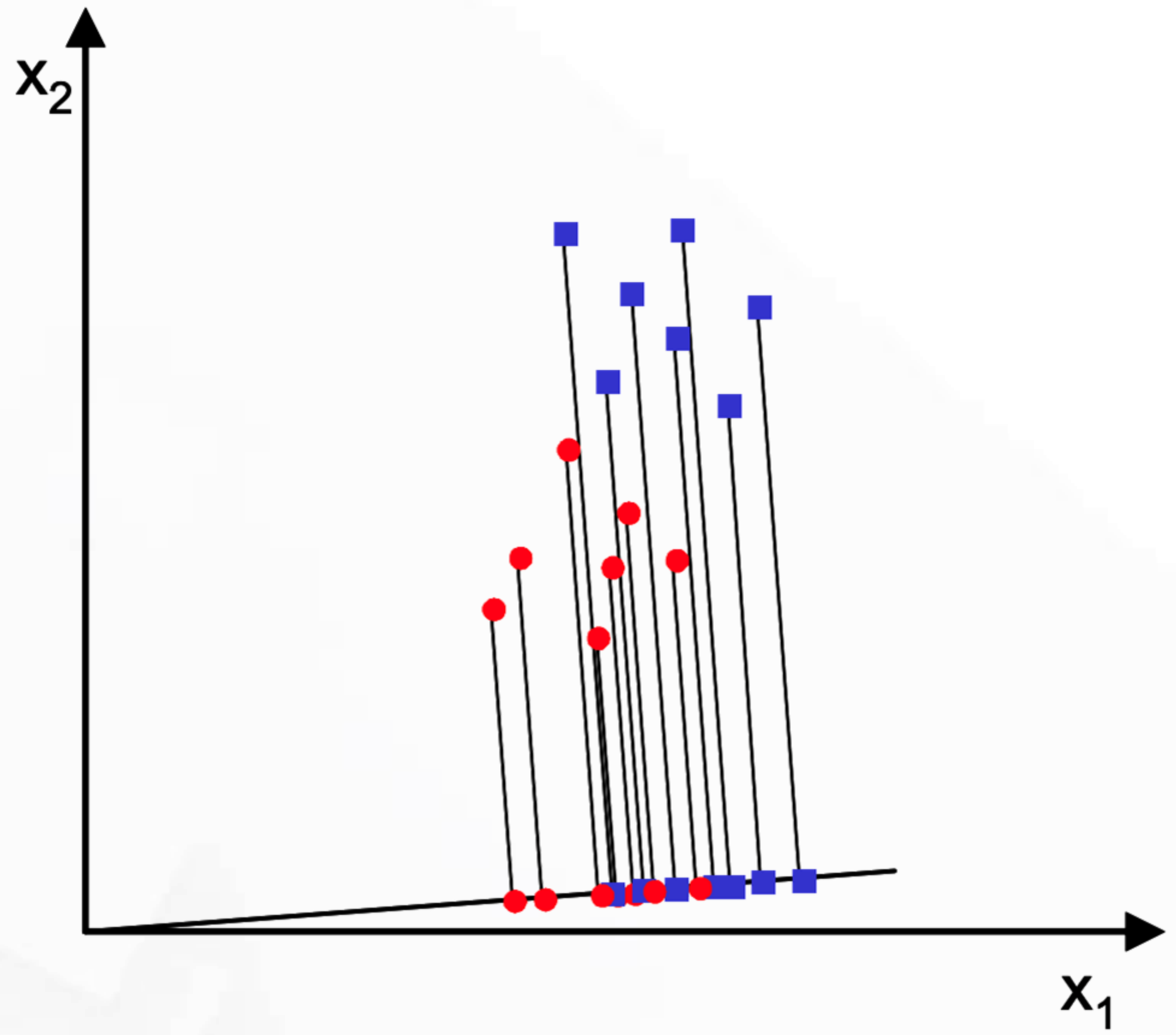


Approach: Maximize Class Separation

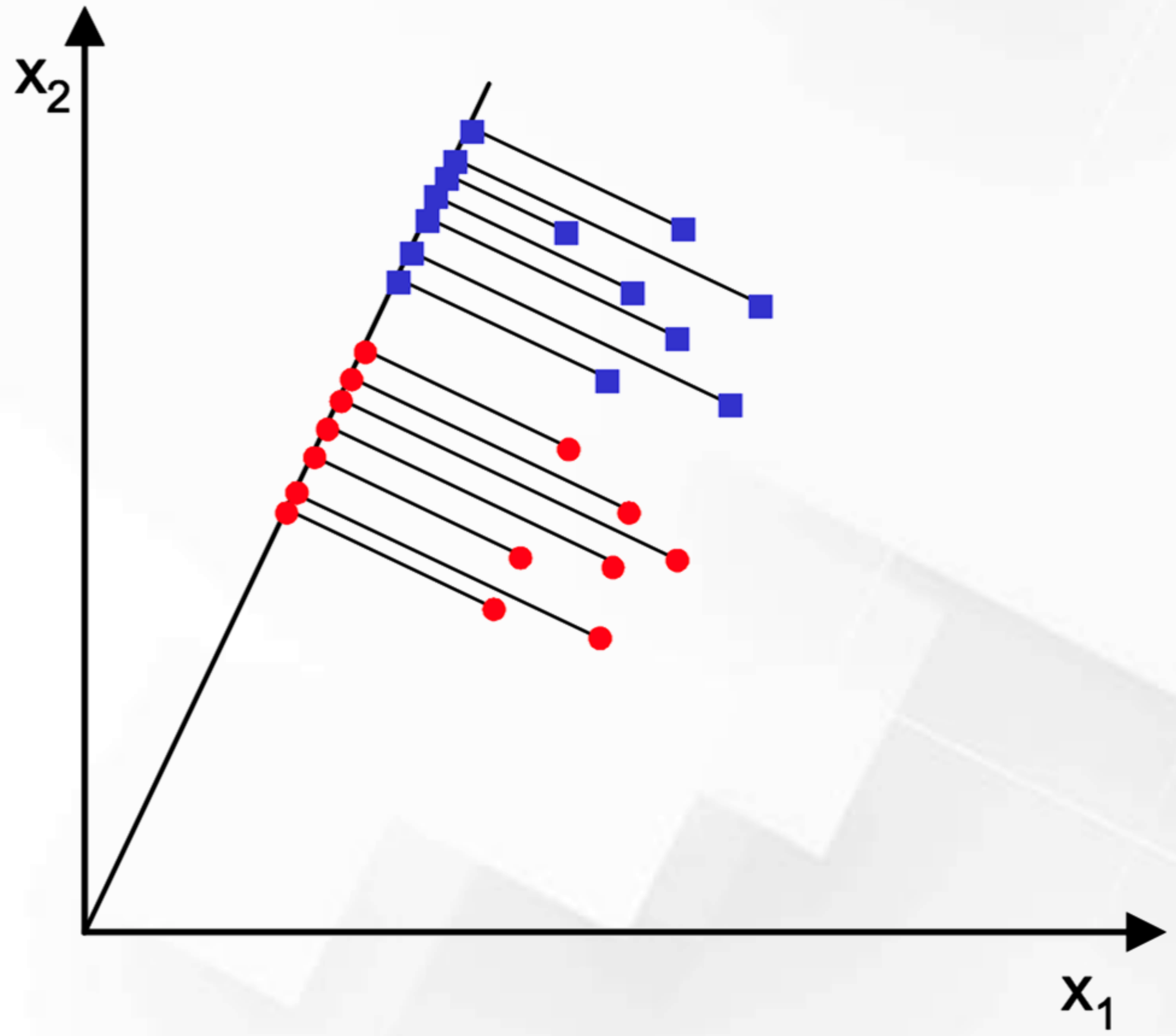
- Consider two classes: ω_1 with N_1 points and ω_2 with N_2 points
  - We seek to obtain a scalar y by projecting the samples $\boldsymbol{x}$ onto the line:

$$y = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$$

- We can adjust the values of $\boldsymbol{w}$ to obtain the projection that maximizes the separability of figures



Which one is a better projection?



Maximize Class Separation

- Mean vectors in $\boldsymbol{x}$ and y feature space is:

$$\boldsymbol{m}_i = \frac{1}{N_i} \sum_{\boldsymbol{x} \in \mathcal{X}_i} \boldsymbol{x}, \qquad i = 1, 2$$

Mean vector of input feature space

$$\tilde{m}_i = \frac{1}{N_i} \sum_{y_j \in \mathcal{Y}_i} y_j = \frac{1}{N_i} \sum_{\boldsymbol{x}_j \in \mathcal{X}_i} \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_j = \boldsymbol{w}^{\mathrm{T}} \boldsymbol{m}_i \qquad i = 1, 2$$

Mean value of projected feature space

- We could choose our objective function as the distance between the projected class means:

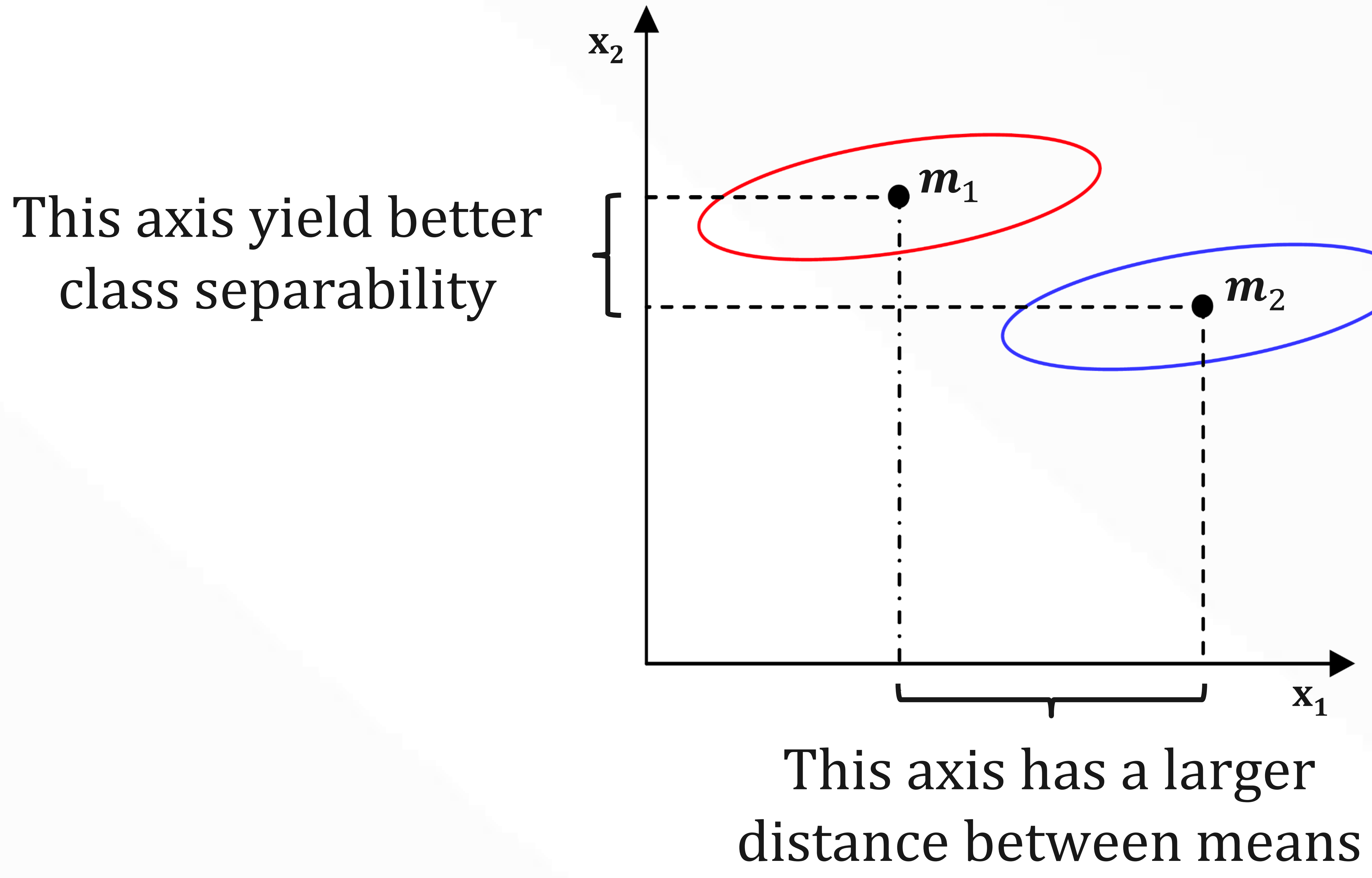
$$J(\boldsymbol{w}) = |\tilde{m}_1 - \tilde{m}_2| = \left|\boldsymbol{w}^{\mathrm{T}}(\boldsymbol{m}_1 - \boldsymbol{m}_2)\right|$$

  - then maximize $J(\boldsymbol{w})$ *w.r.t.* $\boldsymbol{w}$:

$$\max_{\boldsymbol{w}} J(\boldsymbol{w}) = \max_{\boldsymbol{w}} \left|\boldsymbol{w}^{\mathrm{T}}(\boldsymbol{m}_1 - \boldsymbol{m}_2)\right|$$

Maximize Class Separation

- Is this criterion $\max_{\boldsymbol{w}} J(\boldsymbol{w})$ is best?
- Let's look at a figure below:



➢ The distance between the projected means might **not always** be a good measure since it does not take into account the **standard deviation** within the classes

Sir Ronald Aylmer Fisher (R.A. Fisher)

- British statistician and geneticist
- Some of the stuff he invited or popularized:
  - ANOVA (analysis of variance)
  - Maximum likelihood
  - Fisher's z distribution (F distribution)
  - Fisher's method for data fusion (meta analysis)
  - The 0.05 cutoff of p value, the notion of hull hypothesis
  - Fisher's exact test
  - **Fisher's Discriminant Analysis (in 1936)**
  - ……
  - The Genetical Theory of Natural Selection (1930)
  - The Design of Experiments (1935)

R.A. Fisher

(17 February 1890 - 29 July 1962)

Solution: Fisher's Criterion

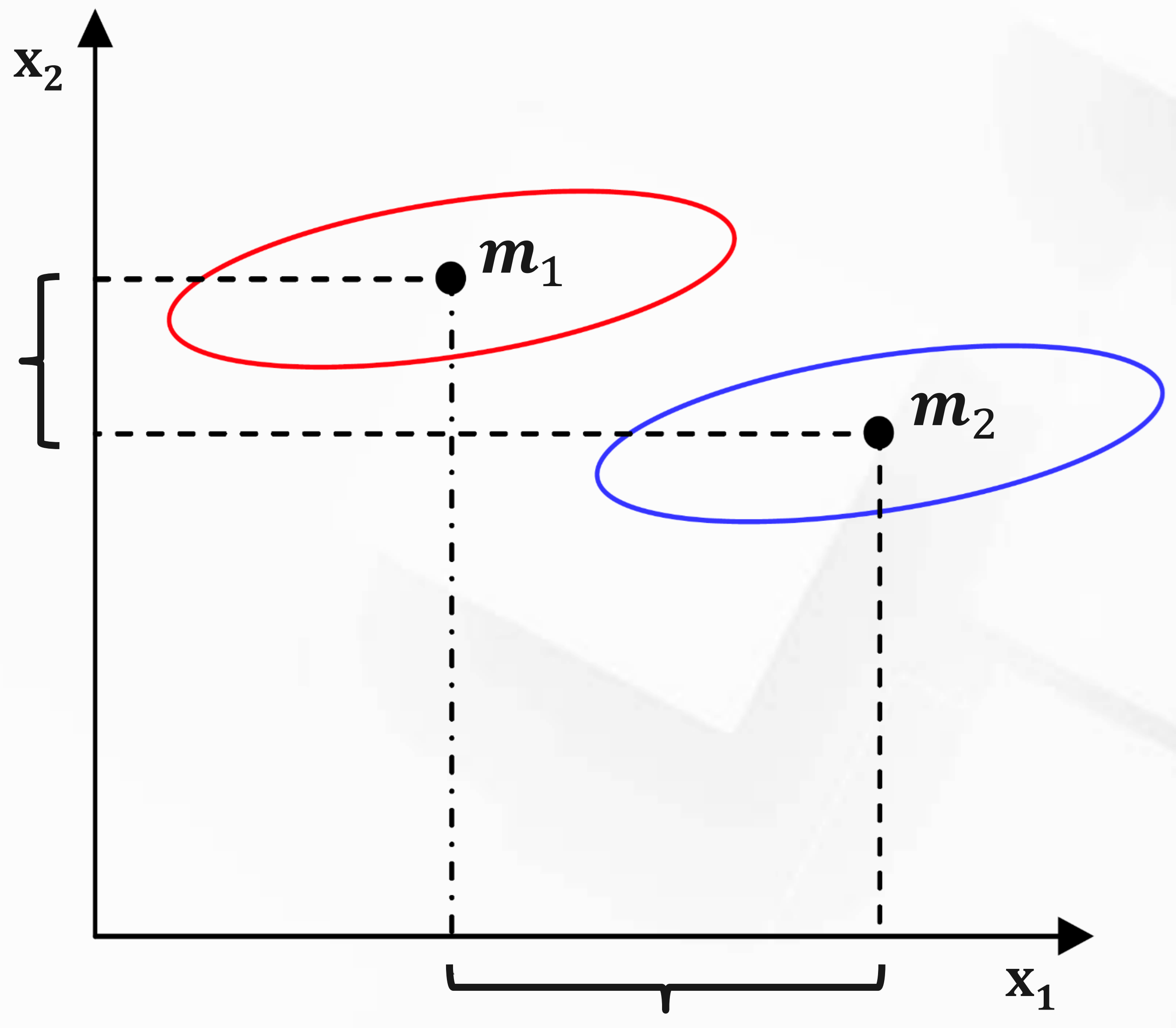
- Fisher proposed to **maximize** a function that represents the difference **between-class means**, which is **normalized** by a measure of the **within-class scatter.**

- Maximizing between-class means of the projection:
  - From previous discussion, we know that we need to maximize the following expression to obtain maximum separation of between-class means:

$$|\widetilde{m}_1 - \widetilde{m}_2|$$

  - By taking a square, we obtain an expression which we called **between-class scatter** of the projection:

$$\tilde{S}_{\mathrm{b}} = (\widetilde{m}_1 - \widetilde{m}_2)^2$$



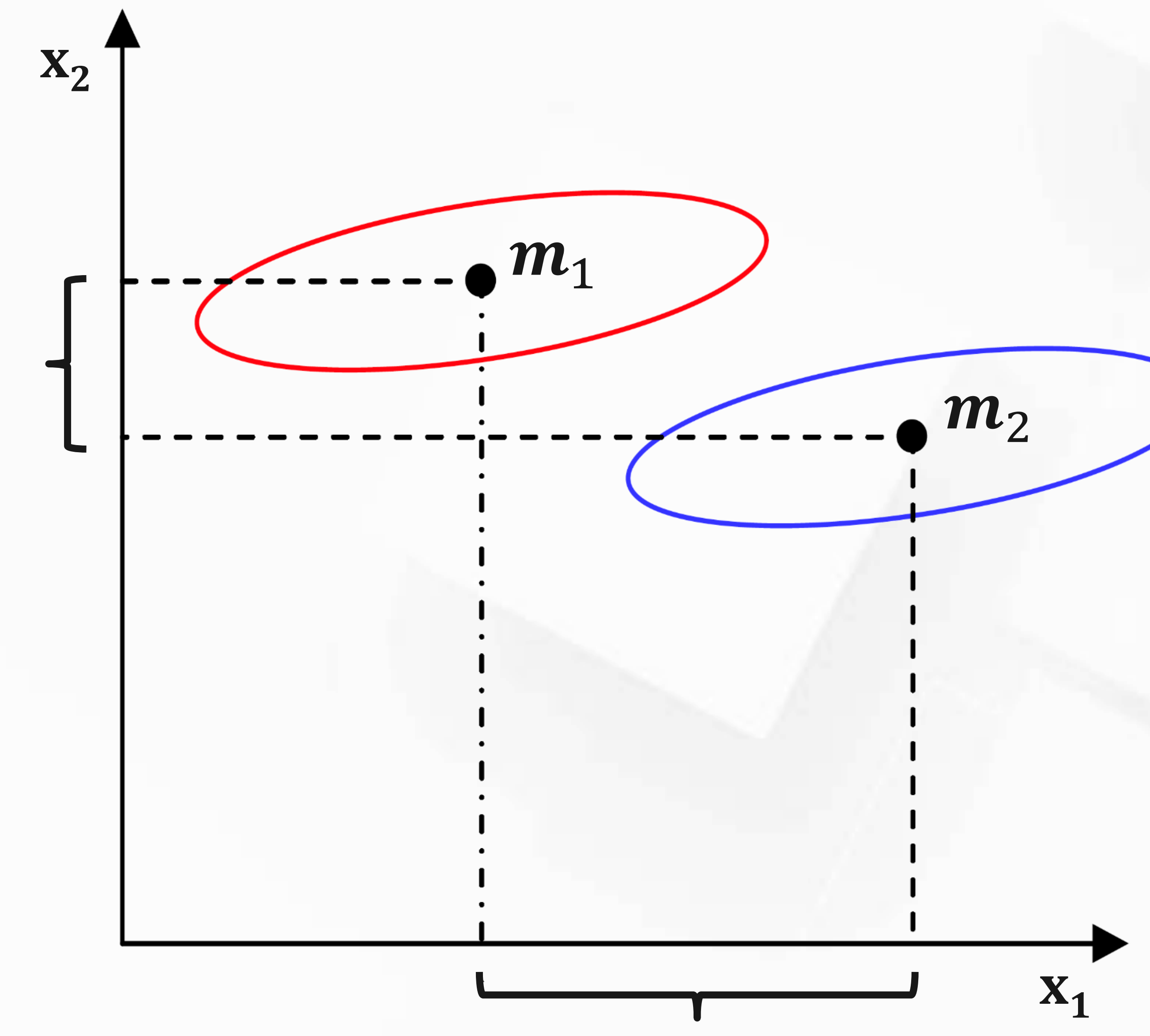
Solution: Fisher's Criterion

- Normalizing by a measure of the within-class scatter:
  - For each class ω_i, we define the **scatter**, an equivalent of the variance, of the projection as:

$$\tilde{S}_i^2 = \sum_{y_j \in \mathcal{Y}_i} (y_j - \tilde{m}_i)^2, \qquad i = 1, 2$$

  - Then the **total within-class scatter** of the projection y will be expressed as:

$$\tilde{S}_\mathrm{w} = \tilde{S}_1^2 + \tilde{S}_2^2$$

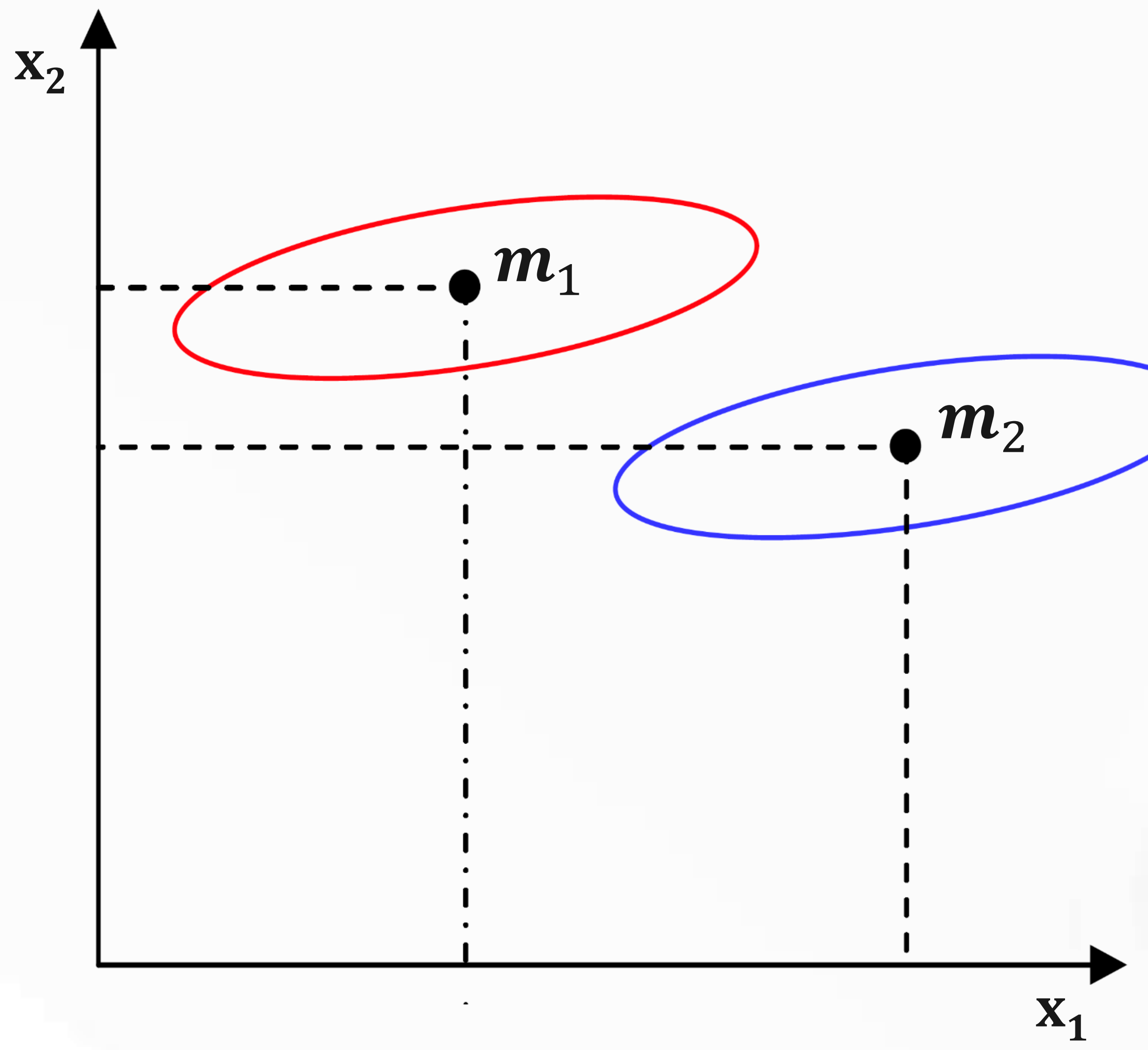


- Combining the 2 expressions:
  - The new objective function with now be:

$$\boxed{J_\mathrm{F}(\boldsymbol{w}) = \frac{\text{between−class scatter}}{\text{total within−class scatter}} = \frac{\tilde{S}_\mathrm{b}}{\tilde{S}_\mathrm{w}}}$$

Solution: Fisher's Criterion

$$J_F(\boldsymbol{w}) = \frac{\tilde{S}_b}{\tilde{S}_w}$$



Therefore, we will be looking for a projection where,

1. Examples from the same class are projected very close to each other
2. The projected means are as farther apart as possible

Optimization

- In order to find the parameters of $\boldsymbol{w}$ that optimizes $J_{\mathrm{F}}(\boldsymbol{w})$, we need to express it as a function of $\boldsymbol{w}$,

$$J_{\mathrm{F}}(\boldsymbol{w}) = \frac{\tilde{S}_{\mathrm{b}}}{\tilde{S}_{\mathrm{w}}} = \frac{|\tilde{m}_1 - \tilde{m}_2|^2}{\tilde{S}_1^2 + \tilde{S}_2^2}$$

  - From the previous discussion, we know the numerator can be express in original feature space ($\boldsymbol{x}$) as:

$$\begin{aligned}
\tilde{S}_{\mathrm{b}} &= (\tilde{m}_1 - \tilde{m}_2)^2 \\
&= \left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{m}_1 - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{m}_2\right)^2 \\
&= \boldsymbol{w}^{\mathrm{T}}\underbrace{(\boldsymbol{m}_1 - \boldsymbol{m}_2)(\boldsymbol{m}_1 - \boldsymbol{m}_2)^{\mathrm{T}}}_{\boldsymbol{S}_{\mathrm{b}}}\boldsymbol{w} \\
&= \boldsymbol{w}^{\mathrm{T}}\boldsymbol{S}_{\mathrm{b}}\boldsymbol{w}
\end{aligned}$$

$\boldsymbol{S}_{\mathrm{b}}$ is the between-class scatter of the original feature space

Optimization

- The denominator of J_{F}, $\tilde{S}_{\mathrm{w}}$ can be expressed as a function of scatter matrix in feature space ($\boldsymbol{x}$):

$$
\begin{aligned}
\tilde{S}_{\mathrm{w}} &= \tilde{S}_1^2 + \tilde{S}_2^2 \\
&= \sum_{y_j \in \mathcal{Y}_1} (y_j - \tilde{m}_1)^2 + \sum_{y_j \in \mathcal{Y}_2} (y_j - \tilde{m}_2)^2 \\
&= \sum_{\boldsymbol{x}_j \in \mathcal{X}_1} \boldsymbol{w}^{\mathrm{T}} (\boldsymbol{x} - \boldsymbol{m}_1)(\boldsymbol{x} - \boldsymbol{m}_1)^{\mathrm{T}} \boldsymbol{w} \\
&\quad + \sum_{\boldsymbol{x}_j \in \mathcal{X}_2} \boldsymbol{w}^{\mathrm{T}} \underbrace{(\boldsymbol{x} - \boldsymbol{m}_2)(\boldsymbol{x} - \boldsymbol{m}_2)^{\mathrm{T}}}_{\boldsymbol{S}_{\mathrm{i}}} \boldsymbol{w} \\
&= \boldsymbol{w}^{\mathrm{T}} \boldsymbol{S}_1 \boldsymbol{w} + \boldsymbol{w}^{\mathrm{T}} \boldsymbol{S}_2 \boldsymbol{w} \\
&= \boldsymbol{w}^{\mathrm{T}} \boldsymbol{S}_{\mathrm{w}} \boldsymbol{w}
\end{aligned}
$$

$\boldsymbol{S}_{\mathrm{i}}$ is the within-class scatter matrix of the original feature space

$\boldsymbol{S}_{\mathrm{w}} = \boldsymbol{S}_1 + \boldsymbol{S}_2$ is the **total** within-class scatter matrix of the original feature space

Optimization

- Finally, the Fisher Criterion can be expressed in terms of $\boldsymbol{S}_{\mathrm{w}}$ and $\boldsymbol{S}_{\mathrm{b}}$ of the feature space $(\boldsymbol{x})$ as:

$$J_{\mathrm{F}}(\boldsymbol{w}) = \frac{\tilde{S}_{\mathrm{b}}}{\tilde{S}_{\mathrm{w}}} = \frac{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{S}_{\mathrm{b}}\boldsymbol{w}}{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{S}_{\mathrm{w}}\boldsymbol{w}}$$

- We can now optimize the objective function J_{F} with respect to $\boldsymbol{w}$:

$$\boxed{\max_{\boldsymbol{w}} J_{\mathrm{F}}(\boldsymbol{w}) = \max_{\boldsymbol{w}} \frac{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{S}_{\mathrm{b}}\boldsymbol{w}}{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{S}_{\mathrm{w}}\boldsymbol{w}}}$$

Optimization

- It is worth noting that our goal is to obtain the maximum projection detection $\boldsymbol{w}$ of $J_{\mathrm{F}}(\boldsymbol{w})$. Since the change in amplitude of $\boldsymbol{w}$ will not affect its direction, for this reason, it will not affect the value of $J_{\mathrm{F}}(\boldsymbol{w})$.

  - Our optimization problem can be simplified to:

$$\max \quad \boldsymbol{w}^{\mathrm{T}} \boldsymbol{S}_{\mathrm{b}} \boldsymbol{w}$$

$$\text{s.t.} \quad \boldsymbol{w}^{\mathrm{T}} \boldsymbol{S}_{\mathrm{w}} \boldsymbol{w} = c \neq 0$$

  - Now we can use Lagrange Multiplier to solve the optimization problem:

$$\mathrm{L}(\boldsymbol{w}, \lambda) = \boldsymbol{w}^{\mathrm{T}} \boldsymbol{S}_{\mathrm{b}} \boldsymbol{w} - \lambda(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{S}_{\mathrm{w}} \boldsymbol{w} - c)$$

Optimization

- Optimizing using Lagrange Multiplier
  - First, we differentiate the Lagrangian and equate it to zero:

$$\frac{\partial \mathrm{L}(\boldsymbol{w}, \lambda)}{\partial \boldsymbol{w}} = 0$$

$$\Rightarrow \boldsymbol{S}_{\mathrm{b}} \boldsymbol{w}^* - \lambda \boldsymbol{S}_{\mathrm{w}} \boldsymbol{w}^* = 0$$

  - Assuming $\boldsymbol{S}_{\mathrm{w}}$ is nonsingular (which usually true when the number of samples is larger than the feature dimension):

$$\Rightarrow \boldsymbol{S}_{\mathrm{w}}^{-1} \boldsymbol{S}_{\mathrm{b}} \boldsymbol{w}^* = \lambda \boldsymbol{w}^*$$

  - Noting that $\boldsymbol{S}_{\mathrm{b}} = (\boldsymbol{m}_1 - \boldsymbol{m}_2)(\boldsymbol{m}_1 - \boldsymbol{m}_2)^{\mathrm{T}}$:

$$\Rightarrow \lambda \boldsymbol{w}^* = \boldsymbol{S}_{\mathrm{w}}^{-1} (\boldsymbol{m}_1 - \boldsymbol{m}_2)(\boldsymbol{m}_1 - \boldsymbol{m}_2)^{\mathrm{T}} \boldsymbol{w}^*$$

Optimization

- Since we only care about the direction of $\boldsymbol{w}^*$ and given that $(\boldsymbol{m}_1 - \boldsymbol{m}_2)^{\mathrm{T}}\boldsymbol{w}^*$ is scalar, the direction of $\boldsymbol{w}^*$ will only be affected by $\boldsymbol{S}_{\mathrm{w}}^{-1}(\boldsymbol{m}_1 - \boldsymbol{m}_2)$. Thus, we obtain:

$$\boxed{\boldsymbol{w}^* = \boldsymbol{S}_{\mathrm{w}}^{-1}(\boldsymbol{m}_1 - \boldsymbol{m}_2)}$$

- Which is the **best** projection direction under Fisher's Criterion

Classification

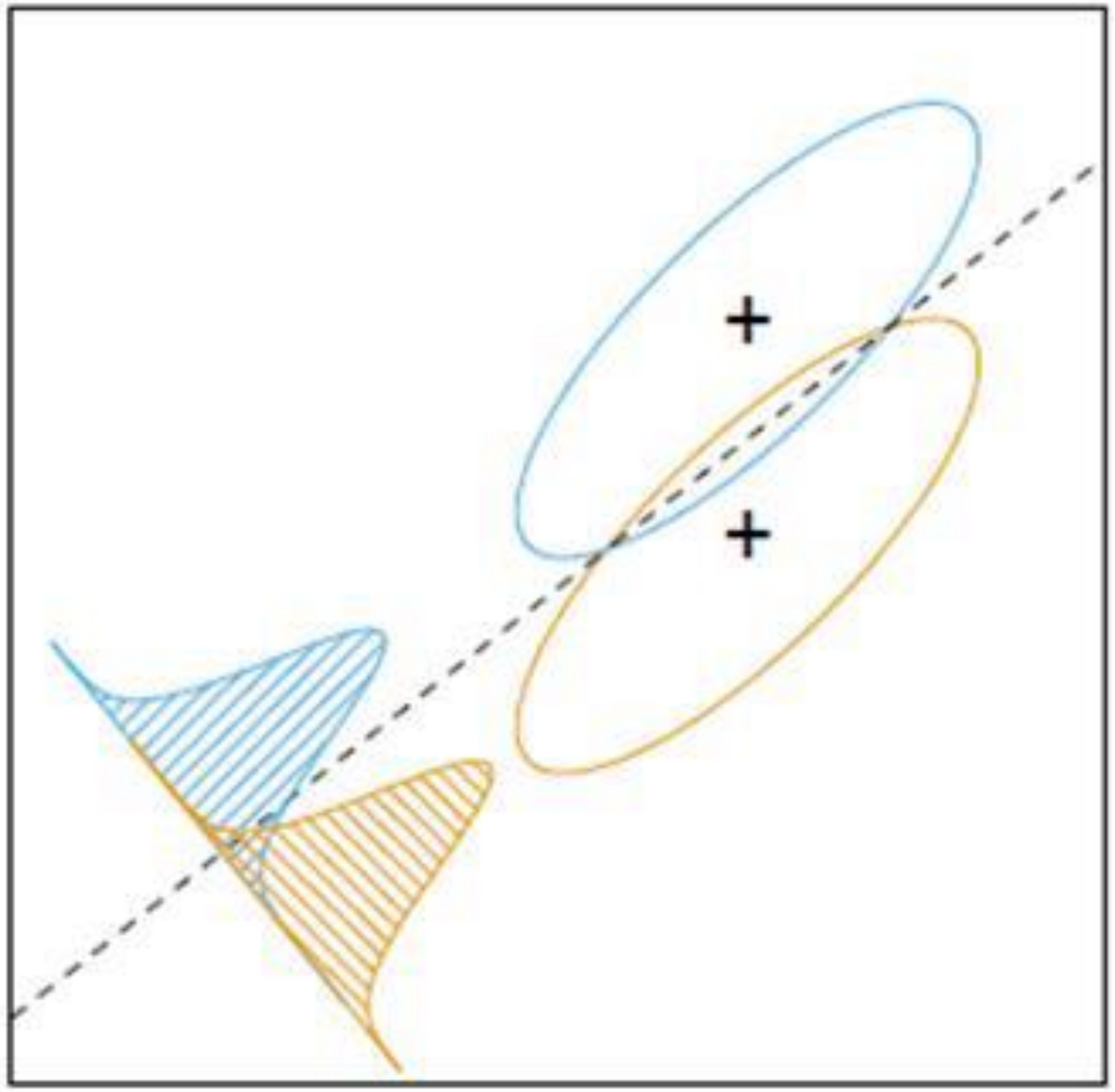
- With optimized the projection direction, now we can decide a decision boundary,

$$g(\boldsymbol{x}) = \text{sign}\left(\sum_{i=1}^{N} w_i\, x_i + w_0\right)$$

- How do we choose w_0?
  - Some commonly use thresholds:

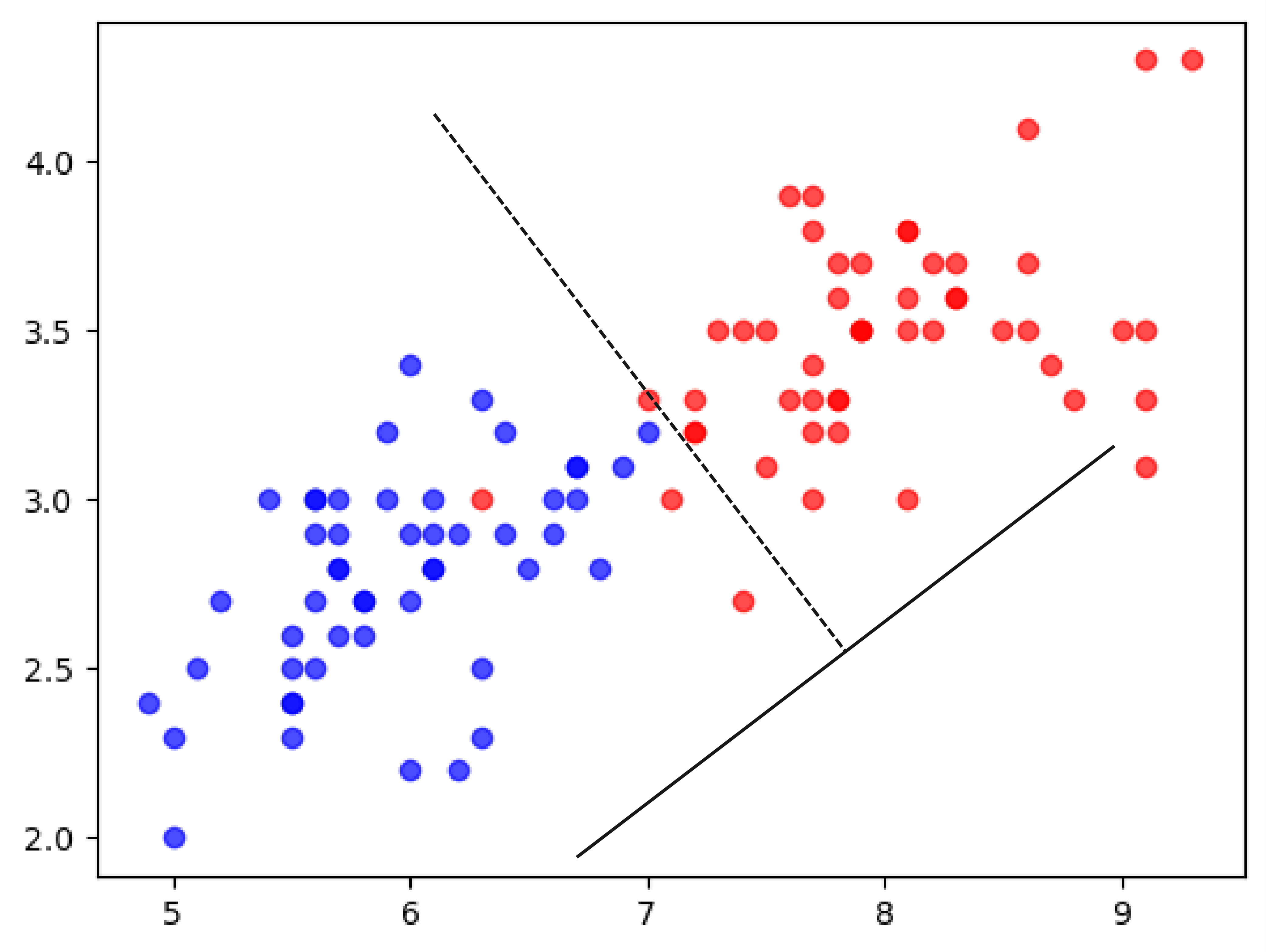
$$w_0 = -\frac{1}{2}(\tilde{m}_1 + \tilde{m}_2)$$

$$w_0 = -\frac{1}{2}(\tilde{m}_1 + \tilde{m}_2) + \frac{1}{N_1 + N_2 - 2}\ln\frac{P(\omega_1)}{P(\omega_2)}$$



Classification

- Even though Fisher's Criterion cannot fully separate the class of data, it can provide a good estimate of $\boldsymbol{w}$ such that there is not much overlap in the projected space.



Classification

What if we have more than 2 classes?

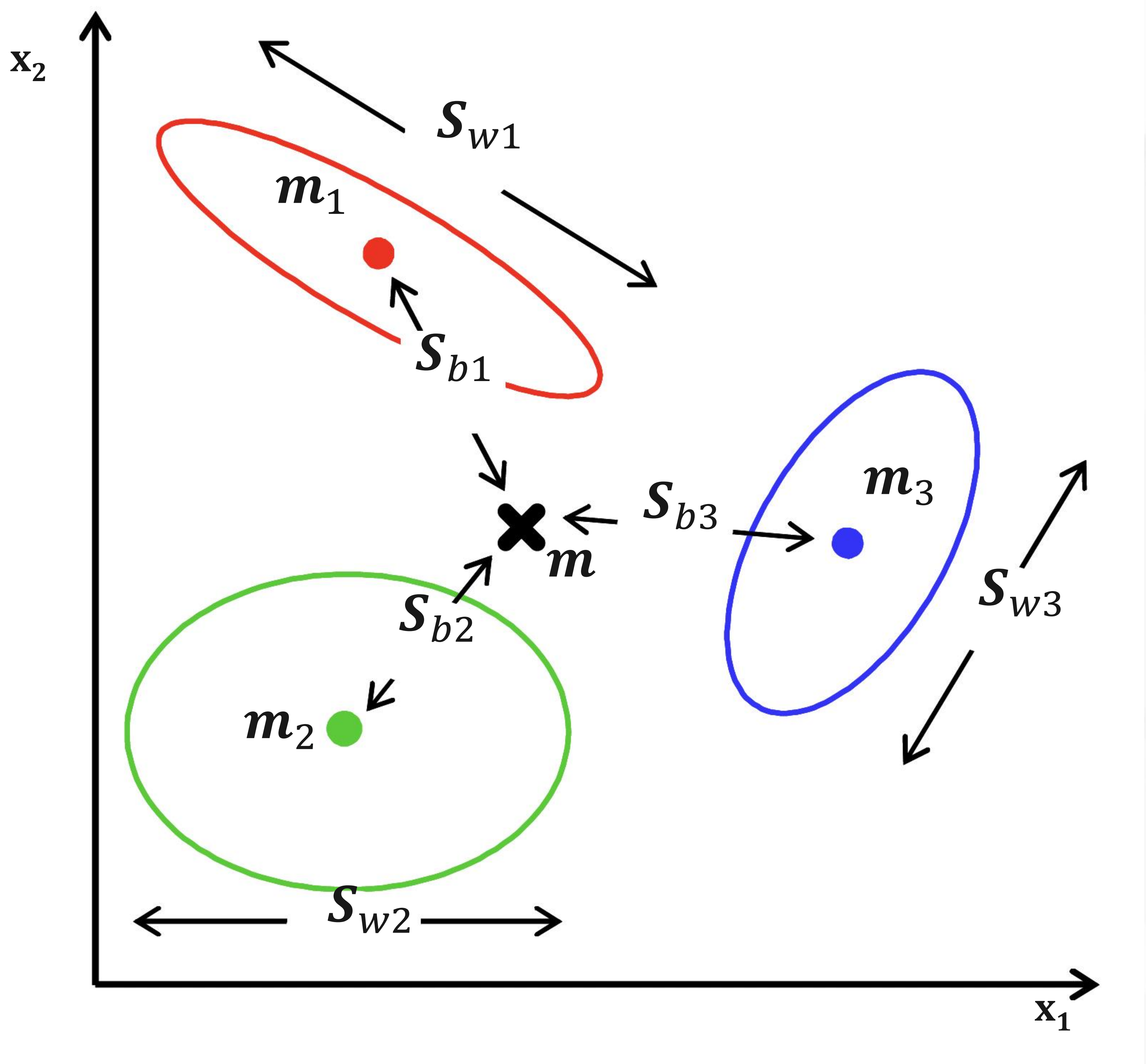
Generalization to K-Classes

- Fisher's LDA can be easily generalized to K-classes
  - Instead of one projection y, we will seek (K-1) projections $[y_1, y_2, \dots, y_N]$ by means of (K-1) projection vectors $\boldsymbol{w}_i$, which can be arranged by columns into a projection matrix containing

$$\boldsymbol{W} = \begin{bmatrix} | & \cdots & | \\ \boldsymbol{w}_1 & \ddots & \boldsymbol{w}_N \\ | & \cdots & | \end{bmatrix}:$$

$$y_i = \boldsymbol{w}_i^T \boldsymbol{x}$$

$$\Rightarrow \mathbf{y} = \boldsymbol{W}^{\mathrm{T}} \boldsymbol{x}$$



Generalization to K-Classes

- Then the derivation of everything else just follows:
  - The generalization of the **within-class scatter** of the **feature space** is

$$\boldsymbol{S}_w = \sum_{k=1}^{K} \sum_{\boldsymbol{x}_j \in \mathcal{X}_k} (\boldsymbol{x}_j - \boldsymbol{m}_k)(\boldsymbol{x}_j - \boldsymbol{m}_k)^{\mathrm{T}}$$

where $\boldsymbol{m}_k = \frac{1}{N_k} \sum_{\boldsymbol{x}_j \in \mathcal{X}_k} \boldsymbol{x}_j$ is the mean of the samples in class ω_i

  - The generalization of the **between-class scatter** of the **feature space** is

$$\boldsymbol{S}_b = \sum_{k=1}^{K} N_k (\boldsymbol{m}_k - \boldsymbol{m})(\boldsymbol{m}_k - \boldsymbol{m})^{\mathrm{T}}$$

where $\boldsymbol{m} = \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{x}_n$ is the global mean of all samples

Generalization to K-Classes

- The **within-class scatter** of the **projected space** is

$$\tilde{S}_w = \sum_{k=1}^{K} \sum_{\boldsymbol{y}_j \in \mathcal{Y}_k} (\boldsymbol{y}_j - \tilde{\boldsymbol{m}}_k)(\boldsymbol{y}_j - \tilde{\boldsymbol{m}}_k)^{\mathrm{T}}$$

where $\tilde{\boldsymbol{m}}_k = \frac{1}{N_k} \sum_{\boldsymbol{y}_j \in \mathcal{Y}_k} \boldsymbol{y}_j$ is the mean of the samples in class ω_i

- The **between-class scatter** of the **projected space** is

$$\tilde{S}_b = \sum_{k=1}^{K} N_k (\tilde{\boldsymbol{m}}_k - \tilde{\boldsymbol{m}})(\tilde{\boldsymbol{m}}_k - \tilde{\boldsymbol{m}})^{\mathrm{T}}$$

where $\tilde{\boldsymbol{m}} = \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{y}_n$ is the global mean of all projected samples

Generalization to K-Classes

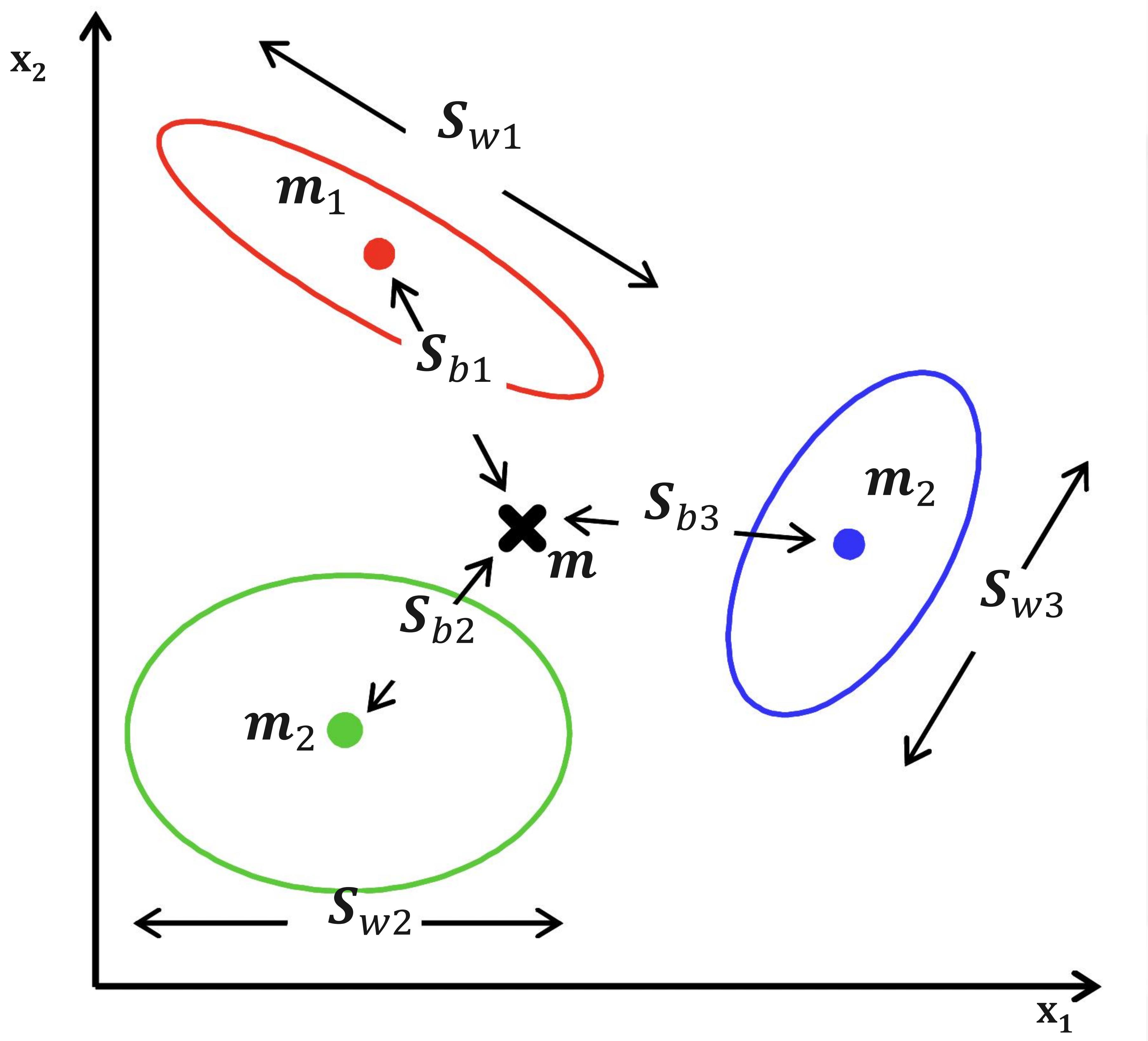
➢ From the derivation in 2 classes:

$$\widetilde{S}_\mathrm{b} = \boldsymbol{W}\boldsymbol{S}_\mathrm{b}\boldsymbol{W}^\mathrm{T}$$
$$\widetilde{S}_\mathrm{w} = \boldsymbol{W}\boldsymbol{S}_\mathrm{w}\boldsymbol{W}^\mathrm{T}$$

➢ The objective function for K-classes is:

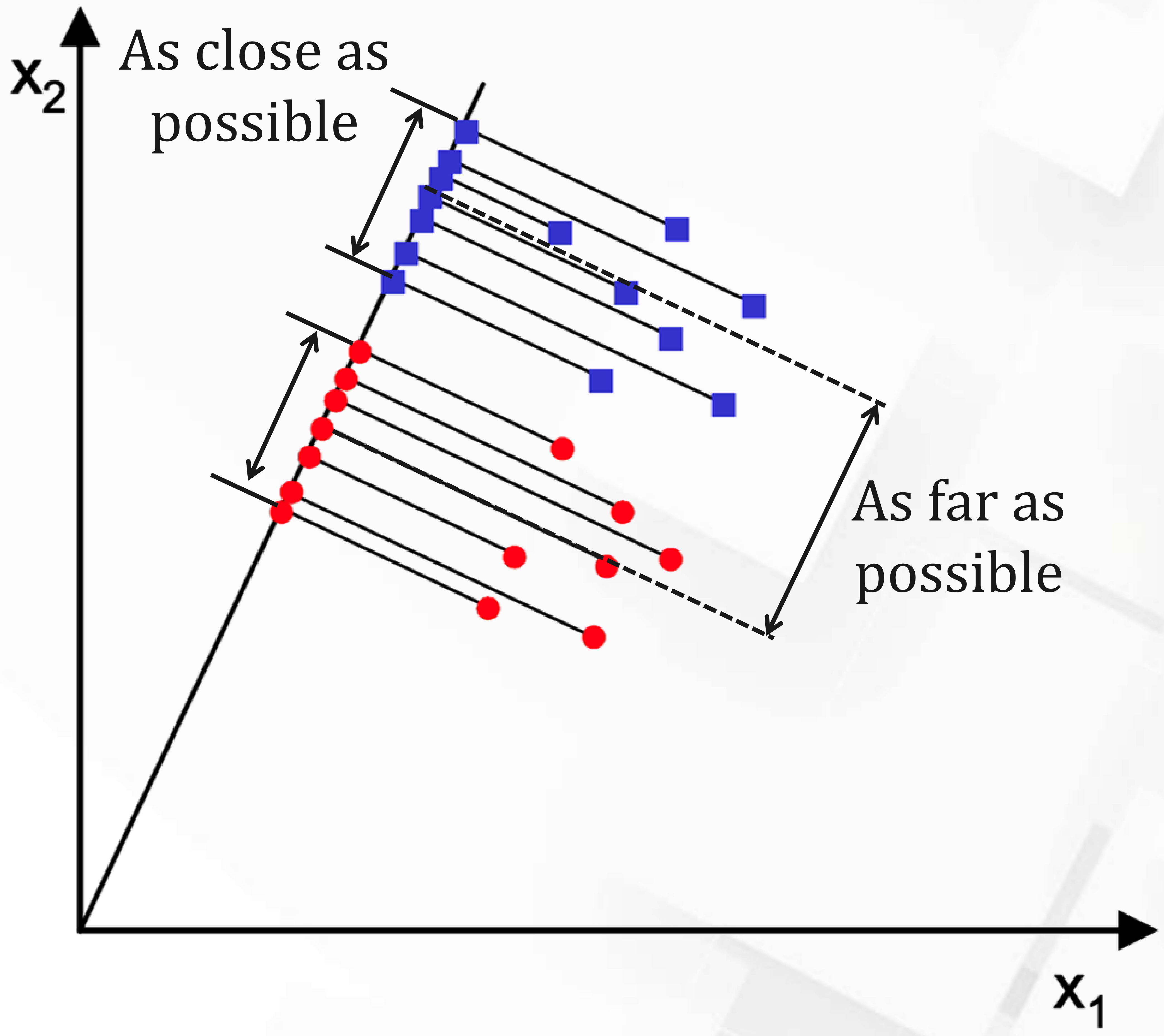
$$J_\mathrm{F}(\boldsymbol{W}) = \mathrm{Tr}\{\widetilde{\boldsymbol{S}}_\mathrm{w}^{-1}\widetilde{\boldsymbol{S}}_\mathrm{b}\}$$

$$\boxed{J_\mathrm{F}(\boldsymbol{W}) = \mathrm{Tr}\left\{\left(\boldsymbol{W}\boldsymbol{S}_\mathrm{w}\boldsymbol{W}^\mathrm{T}\right)^{-1}\left(\boldsymbol{W}^\mathrm{T}\boldsymbol{S}_\mathrm{b}\boldsymbol{W}^\mathrm{T}\right)\right\}}$$



Key Idea of LDA

- Separate samples of distinct groups by projecting them onto a space that
  - Maximizes their **between-class separability** while
  - Minimizing their **within-class variability**



Assumptions in LDA

- LDA assumes every density within each class is a Gaussian distribution.
- Performance of the standard LDA can be seriously degraded if there are only a limited number of total training observations N compared to the dimension D of the feature space - Shrinkage (Copas, 1983)

Limitations in LDA

- LDA will fail when the discriminatory information is not in the mean but rather in the variance of the data.
- If the distributions are significantly non-Gaussian, the LDA projections will not be able to preserve any complex structure of the data, which may be needed for classification

Linear Model Outline

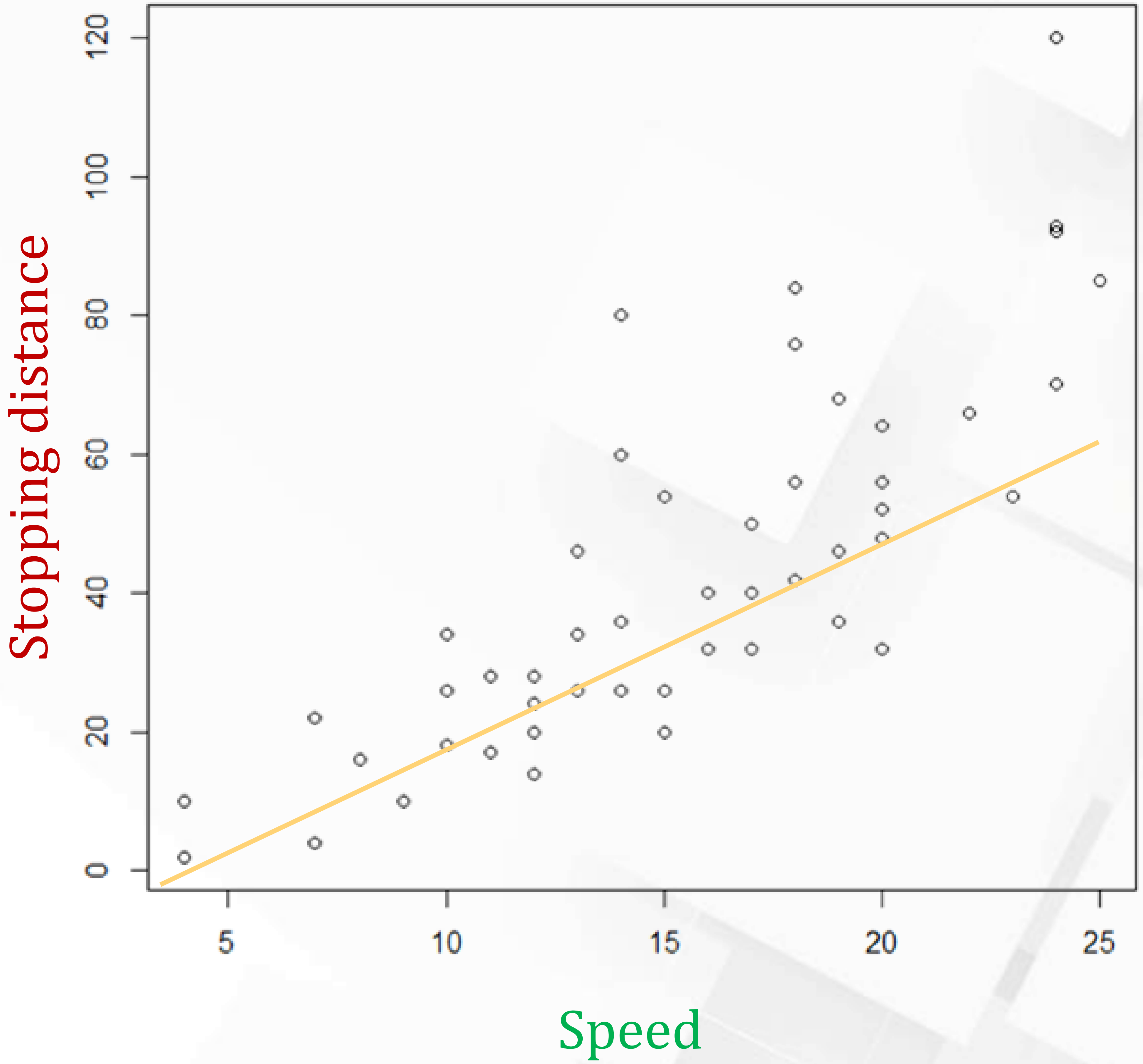
Tsinghua University

1. Linear Regression
2. Linear Discriminant Analysis
3. Logistic Regression
4. Perceptron

Recall Linear Regression

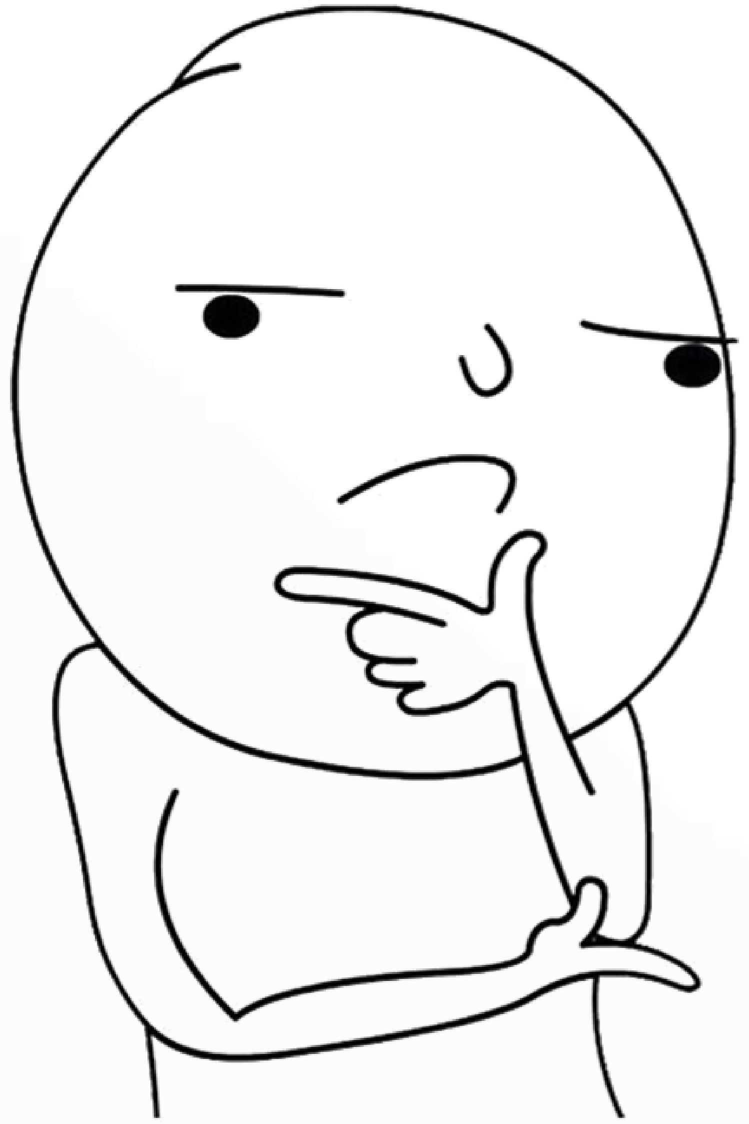
- Predicting values using linear regression
- Speed vs Stopping Distance
- The data points generally follows the $y = w_1 x + w_0$ trend.

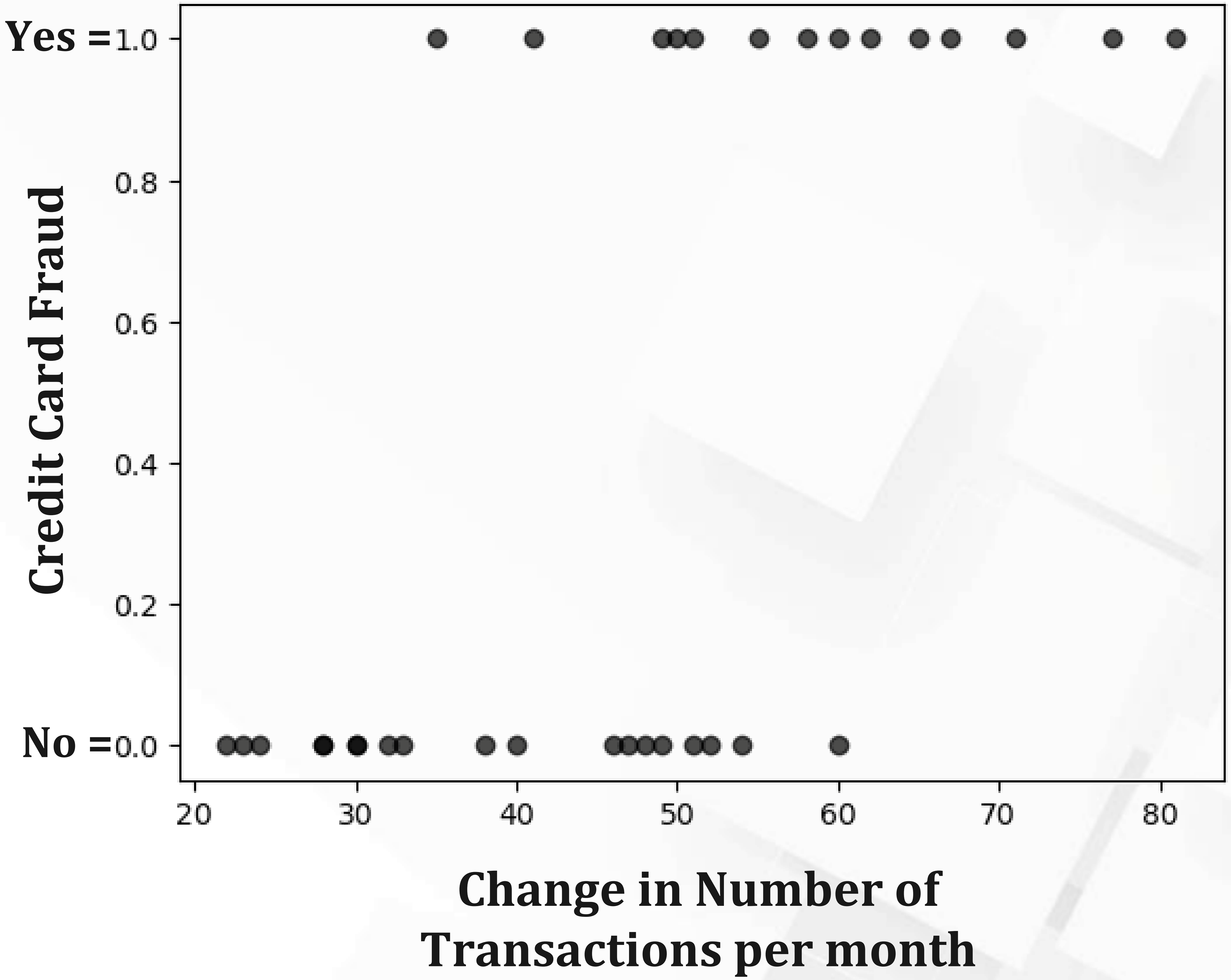
Obviously! Linear Regression!!!



Logistic Regression

What if the data looks like this?



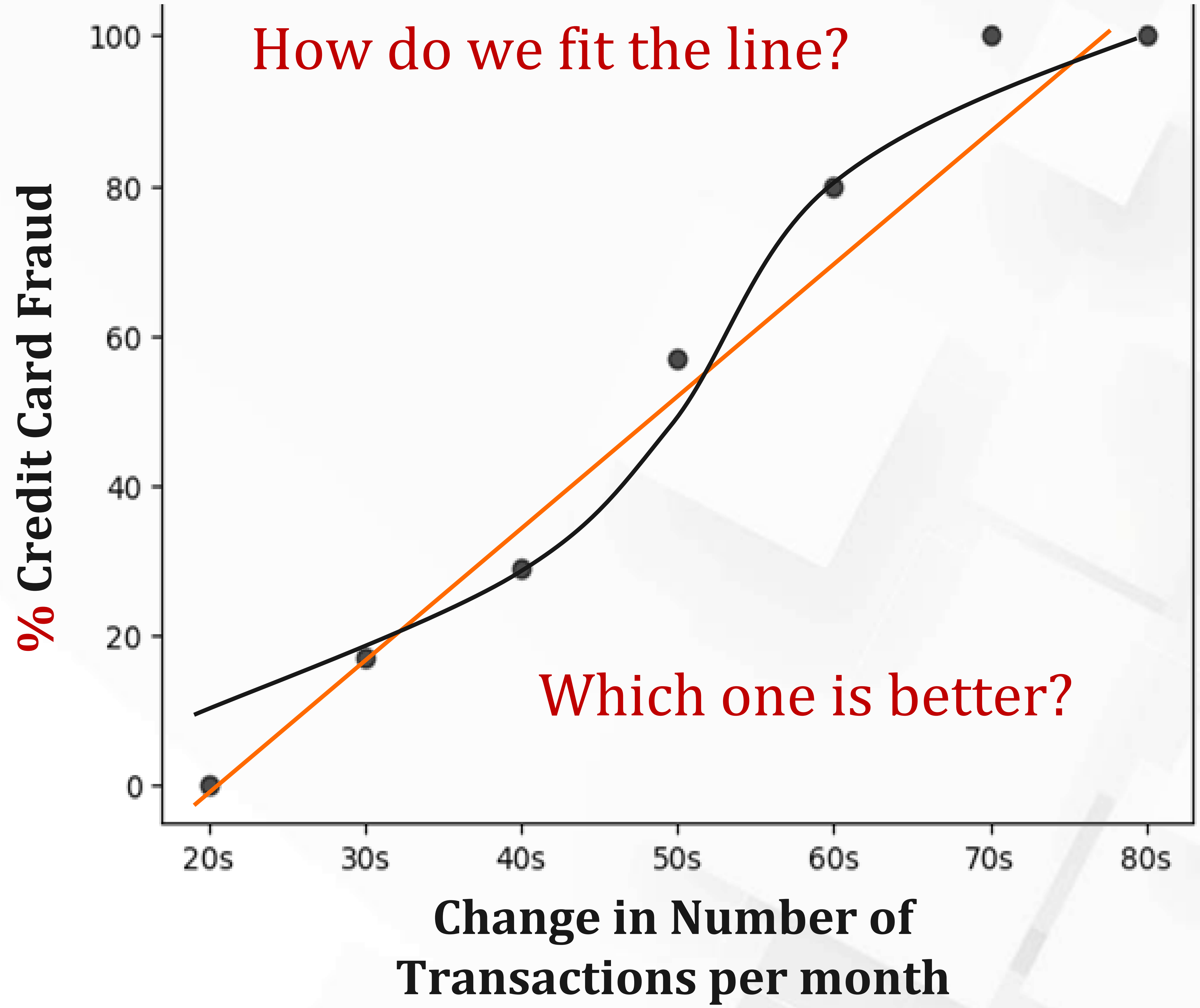


Logistic Regression

- Let's group the credit card data according to number of transactions per month and show the % of fraud in each group:

Change in Number of Transactions	% Credit Card Fraud
20-29	0
30-39	17
40-49	29
50-59	57
60-69	80
70-79	100
80-89	100

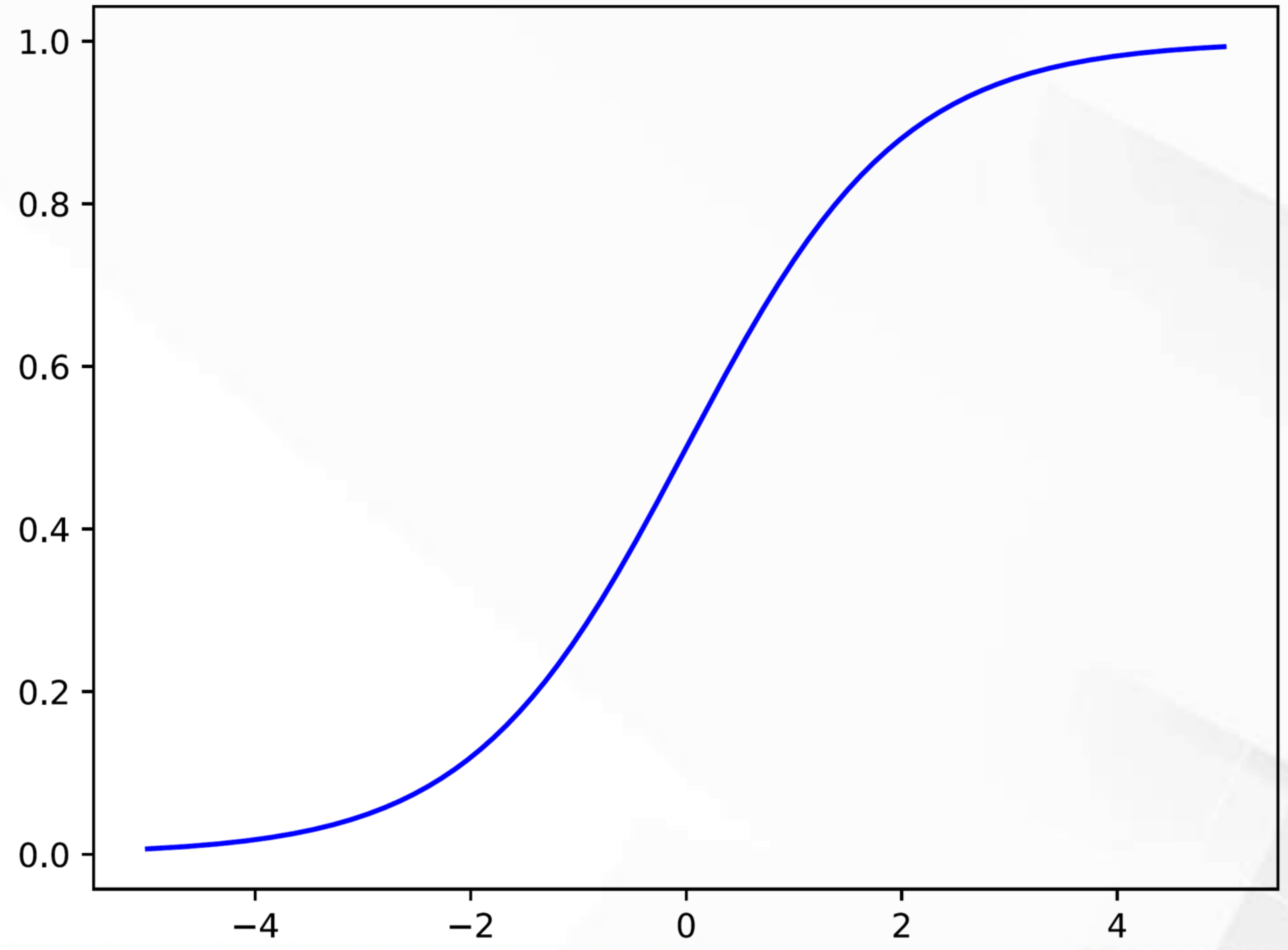
- Then we can graph the data...



Logistic Regression

- This S-shape line is called **Logistic Function**
  - a.k.a. **Sigmoid Function**
  - Logistic regression maps the input value x into values between 0 and 1

$$P(y|x) = \theta(s) = \frac{e^{w_0+w_1x}}{1+e^{w_0+w_1x}}$$



Logistic Regression

- **Probability** of the event

$$P(y|x) = \theta(w_0 + w_1 x) = \frac{e^{w_0+w_1x}}{1 + e^{w_0+w_1x}}$$

- Aside from probability, we also care about the **odds** of an event.

$$\frac{\text{Probability that the event will occur}}{\text{Probability that the event will not occur}} = \frac{P}{1 - P}$$

$$\frac{P}{1 - P} = \frac{\frac{e^{w_0+w_1x}}{1 + e^{w_0+w_1x}}}{1 - \frac{e^{w_0+w_1x}}{1 + e^{w_0+w_1x}}} = e^{w_0+wx}$$

- If we take the logarithmic of odds, we obtain **logits** of $P(y|x)$:

$$\text{logit} = \log(\text{odds}) = w_0 + w_1 x$$

Logistic Regression

- When having an input $\boldsymbol{x}$ with **more than one features**, we can also write it in **vector form**.
- **Probability** of the event becomes:

$$P(y|\boldsymbol{x}) = \theta(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}) = \frac{e^{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}}}{1 + e^{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}}}$$

- The **odds** of an event is:

$$\frac{P}{1 - P} = e^{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}}$$

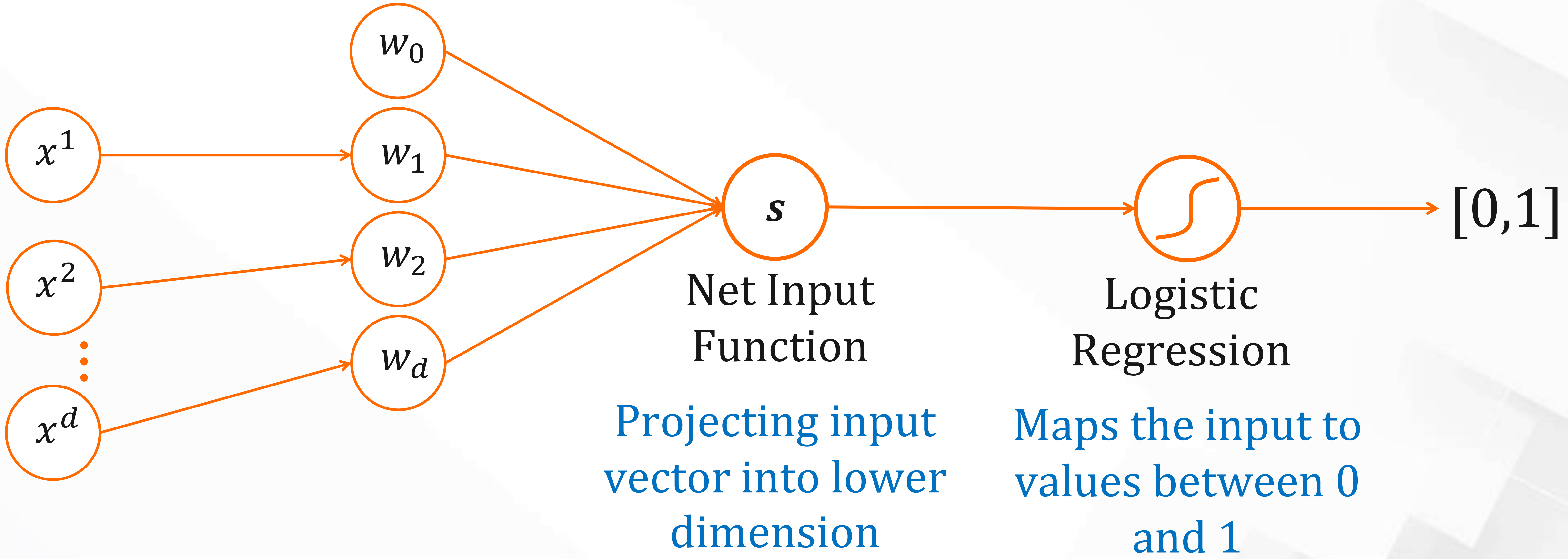
- Taking the logarithmic of odds, we obtain **logits** of $P(y|\boldsymbol{x})$:

$$\mathrm{logit} = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$$

Intuition

- We can think of logit $\mathbf{s} = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_d x_d$ as the overall information, which is also the weighted sum of the causes (or features) for a certain event

- Let $h(\boldsymbol{x}) = \theta(\boldsymbol{s}) = P(y = 1|x) = \frac{e^s}{1 + e^s}$, we say $h(\boldsymbol{x}) = \theta(\boldsymbol{s})$ is estimate of the probability of the event being $y = 1$



Classification

Can we use logistic regression for classification?

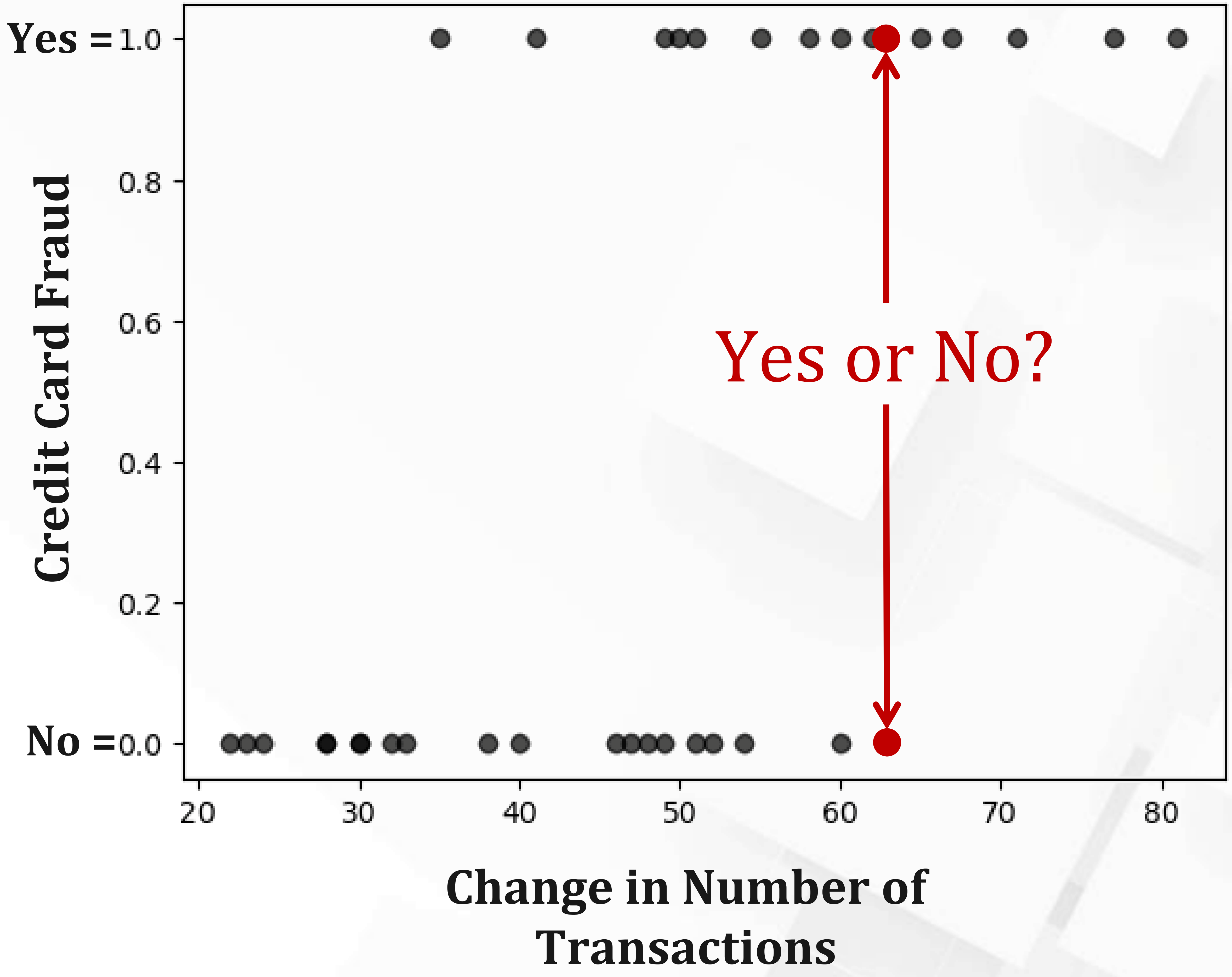
Of course!!!!!

Logistic Regression for Classification

- Let's say Card A is has 63 more transactions this month than the previous month, given the trend we got from the data, is this card involved in fraud?

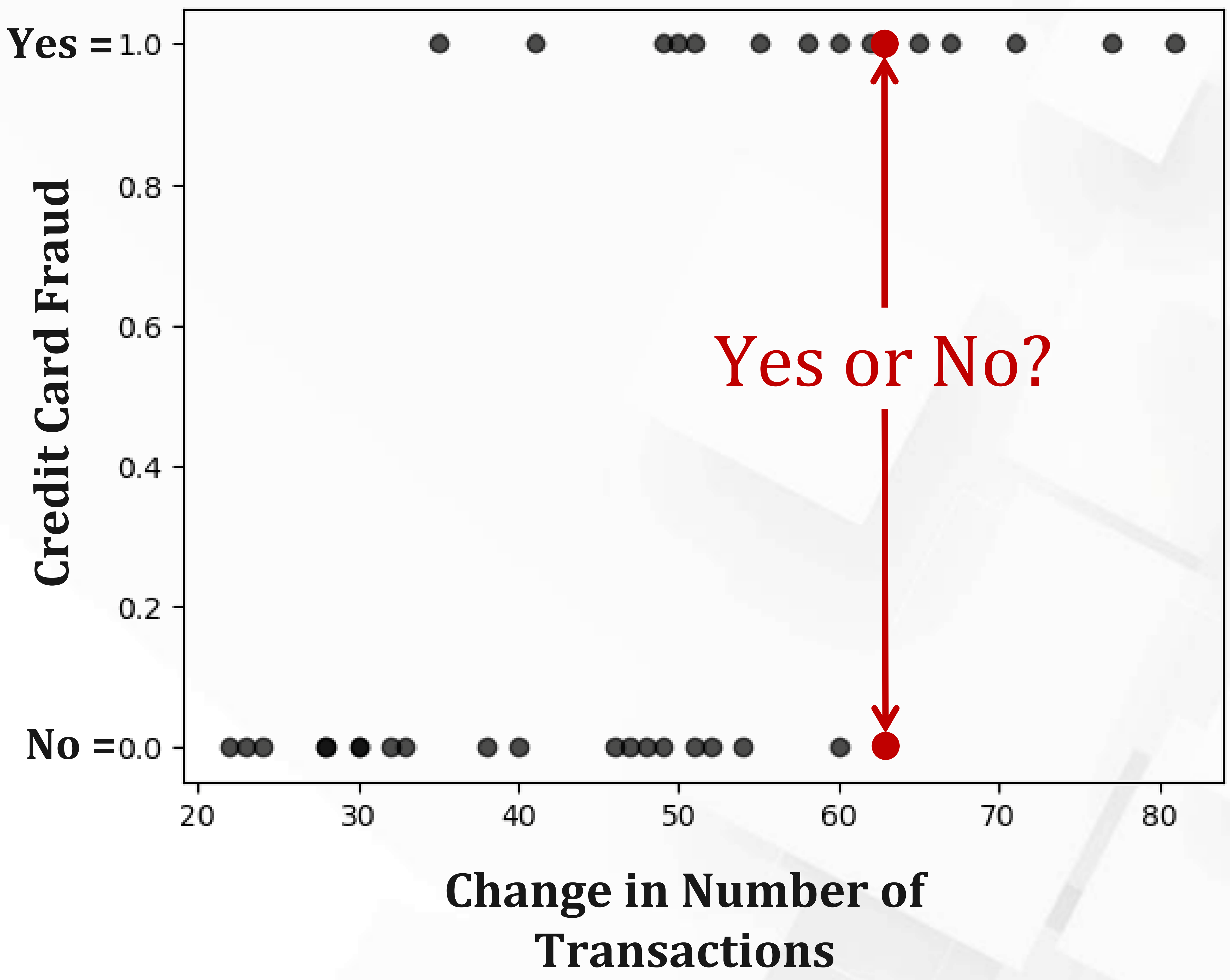
Change in Number of Transactions	Credit Card Fraud
63	Yes or No?

- Where should we graph it on the figure?



Logistic Regression for Classification

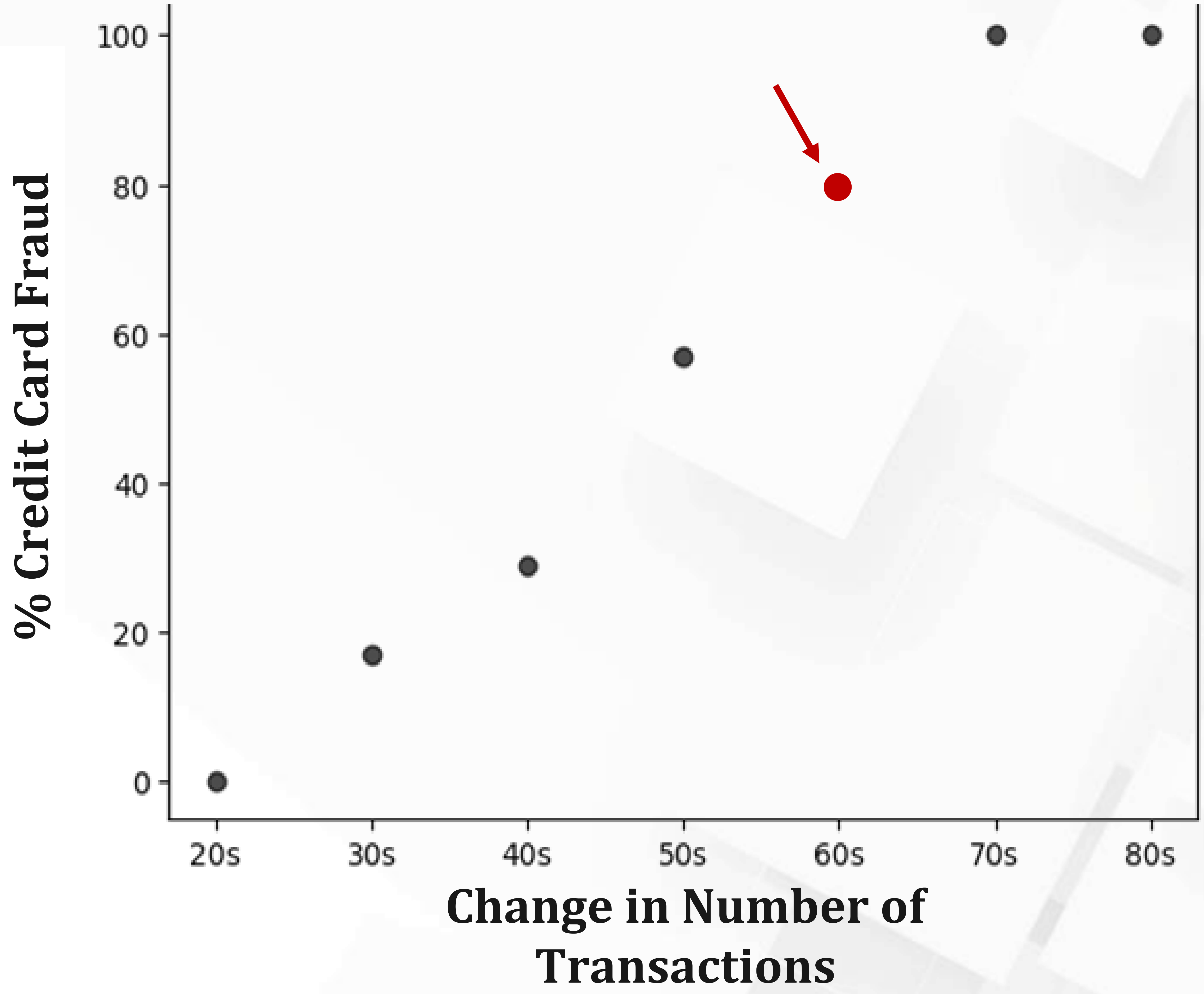
Hard to decide?



Logistic Regression for Classification

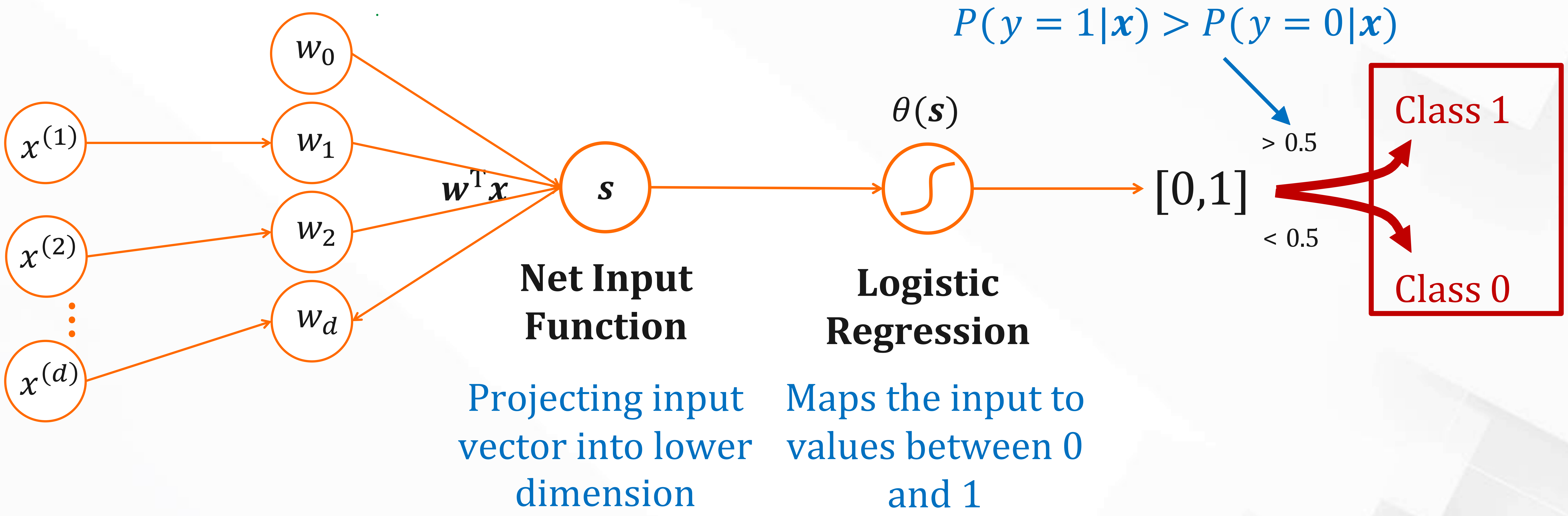
- How about we try plotting it on the grouped Number of Transactions figure with logistic regression trend?
  - Card A falls under the group 60-69!
  - The plot tells us this card has about 80% chance that it is involved in credit card fraud!

80% YES! > 20% No!



Logistic Regression for Classification

- Given an input vector $\boldsymbol{x}$, we set $h(\boldsymbol{x}) = p(y = 1|\boldsymbol{x})$ as the probability to label $\boldsymbol{x}$ as $y = 1$.
- For classification, the logistic regression maps values into values between 0 and 1.
- Then, we can set a threshold (e.g., 0.5) to decide!



Logistic Regression for Classification

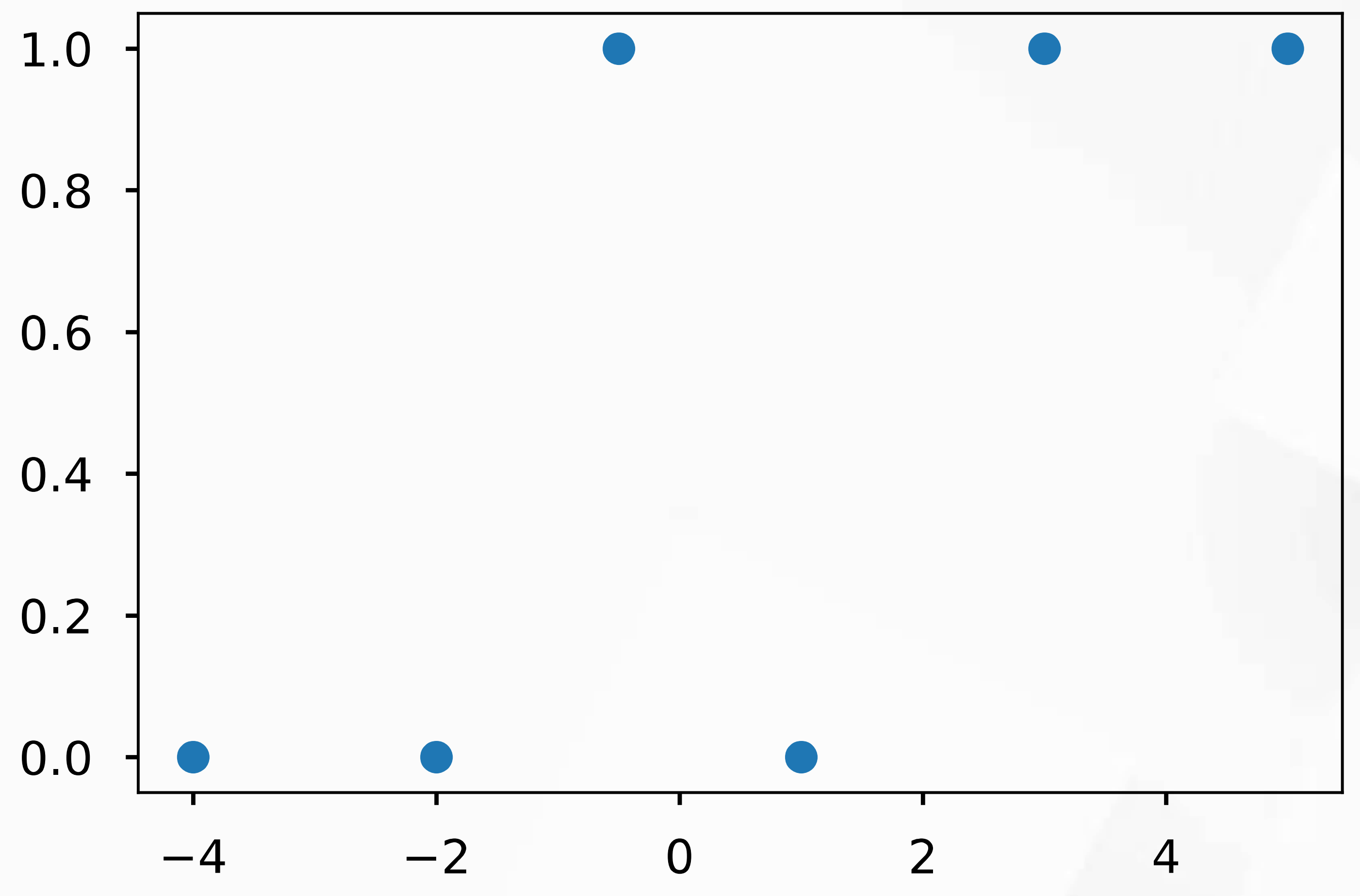
Same old... How do we find the w^{T} that projects the input vector?

Optimization – Maximum Likelihood Estimation

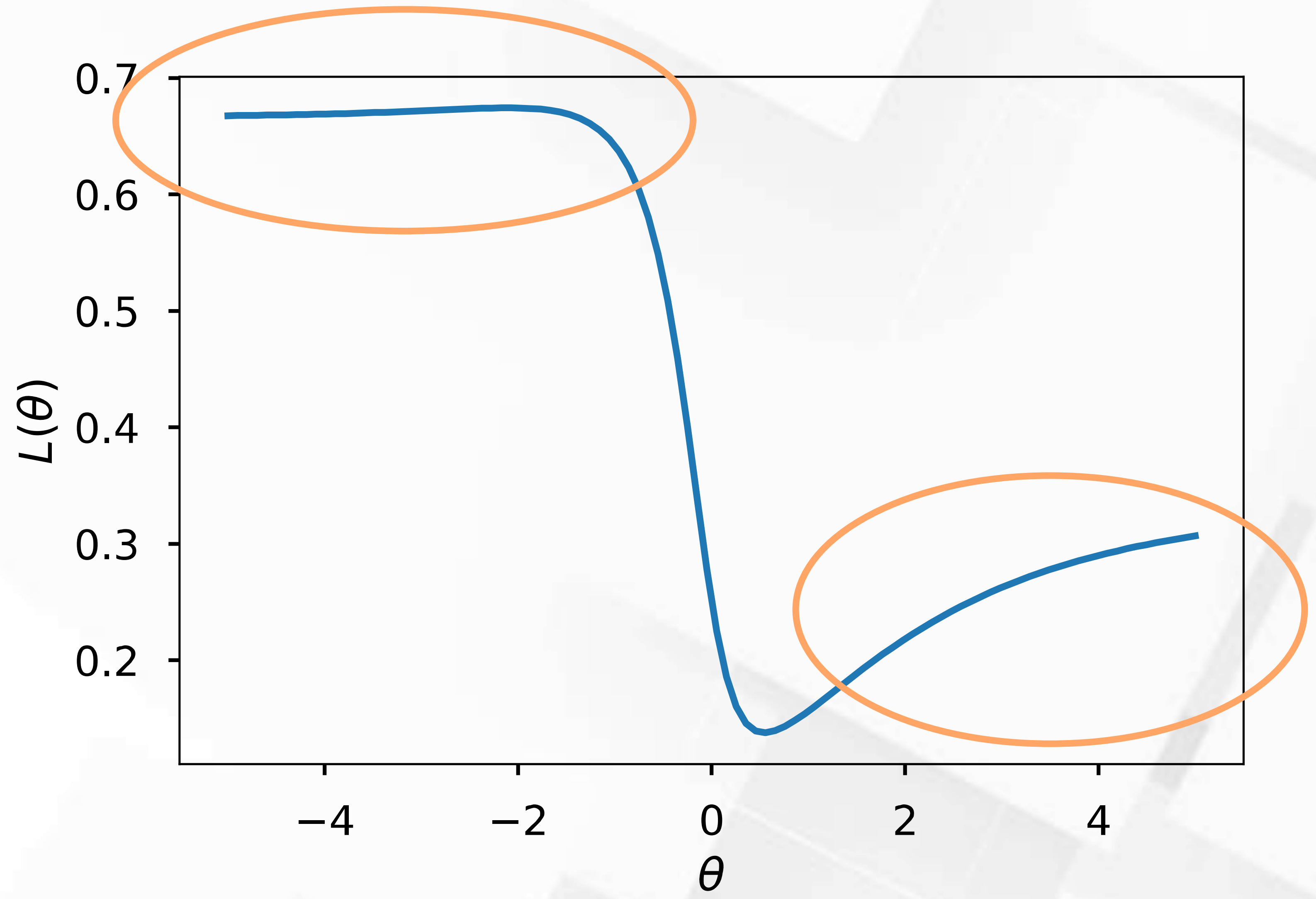
- Let's try **squared loss** for sigmoid function as $\theta(\boldsymbol{s}) \in [0,1]$ and y $\in \{0,1\}$

$$\ell(h(\boldsymbol{x}), y) = \sum_{i=1}^{N} (h(\boldsymbol{x}_i) - y_i)^2$$

- **See any problem?**
  - Tries to match continued probability with discrete 0/1 labels.
  - Non-Convex
  - Small loss when prediction is overly far in wrong side



Toy Data



Squared Loss Surface

Optimization – Maximum Likelihood Estimation

- Assuming we have *i.i.d* sample set $\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2) \dots (\boldsymbol{x}_N, y_N)\}$, where $\boldsymbol{x}_i \in \Re^{d+1}$ and $y_i \in \{0, 1\}$, we want to optimize the parameters $\boldsymbol{w}$ to obtain the maximum likelihood between y_i and $\boldsymbol{x}_i$.

- Maximum Likelihood Function:

$$\max_{\boldsymbol{w}} \prod_{i=1}^{N} \boxed{\prod_{k=0}^{1} P(y_i = c | \boldsymbol{x}_i; \boldsymbol{w})^{\mathbf{1}(y_i = k)}}$$

Bernoulli distribution

$$\max_{\boldsymbol{w}} \prod_{i=1}^{N} \left[\theta\left(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}\right)^{\mathbf{1}(y_i=1)} \times \left(1 - \theta\left(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}\right)\right)^{\mathbf{1}(y_i=0)}\right]$$

Optimization – Maximum Likelihood Estimation

- Applying negative log to the likelihood function, we obtain the log-likelihood for logistic regression:

$$\min_{\boldsymbol{w}} J(\boldsymbol{w}) = \min_{\boldsymbol{w}} \sum_{i=1}^{N} \left\{ \log \left[\theta(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x})^{\mathbf{1}(y_i=1)} \times \left(1 - \theta(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x})\right)^{\mathbf{1}(y_i=0)} \right] \right\}$$

$$\min_{\boldsymbol{w}} - \sum_{i=1}^{N} \left\{ y_i \log\left(\theta(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x})\right) + (1 - y_i) \log\left(1 - \theta(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x})\right) \right\}$$

$$\min_{\boldsymbol{w}} - \sum_{i=1}^{N} \left\{ y_i \log \frac{e^{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i}}{1 + e^{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i}} + (1 - y_i) \log\left(1 - \frac{e^{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i}}{1 + e^{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i}}\right) \right\}$$

Optimization – Maximum Likelihood Estimation

Tsinghua University

$$\min_{\boldsymbol{w}} -\sum_{i=1}^{N}\left\{y_i \log \frac{e^{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i}}{1+e^{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i}} + (1-y_i)\log\left(1-\frac{e^{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i}}{1+e^{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i}}\right)\right\}$$

- Another way of writing this is as follows: Suppose $\tilde{y}_i \in \{-1,+1\}$ instead of $y_i \in \{0,+1\}$, we have $P(\tilde{y}_i = 1|\boldsymbol{x}_i) = \frac{e^{-\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i}}{1+e^{-\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i}}$ and $P(\tilde{y}_i = -1|\boldsymbol{x}_i) = \frac{e^{+\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i}}{1+e^{+\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i}}$

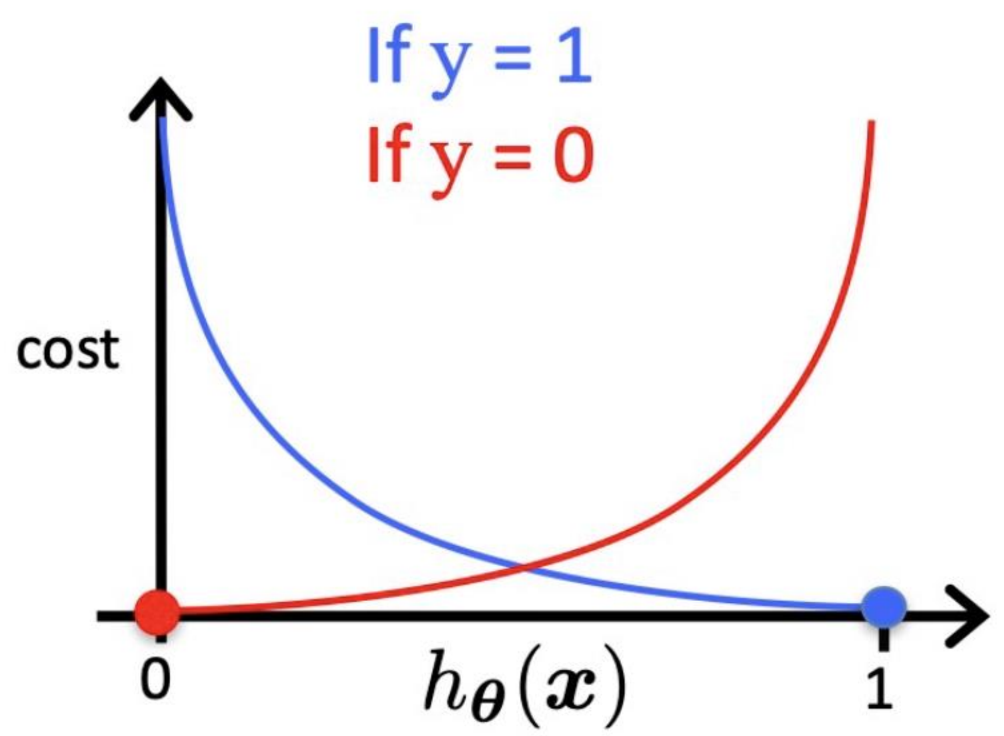
- Noting that, $\theta(-s) = 1-\theta(s)$, and by substituting y_i with $\tilde{y}_i$, we can simplify the previous expression as follows:

$$\boxed{\min_{\boldsymbol{w}} J(\boldsymbol{w}) = \min_{\boldsymbol{w}} \sum_{i=1}^{N}\left\{\log\left(1+e^{-\tilde{y}_i\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i}\right)\right\}}$$

Optimization – Cross Entropy Loss

- This function has a special name called Cross Entropy Loss:

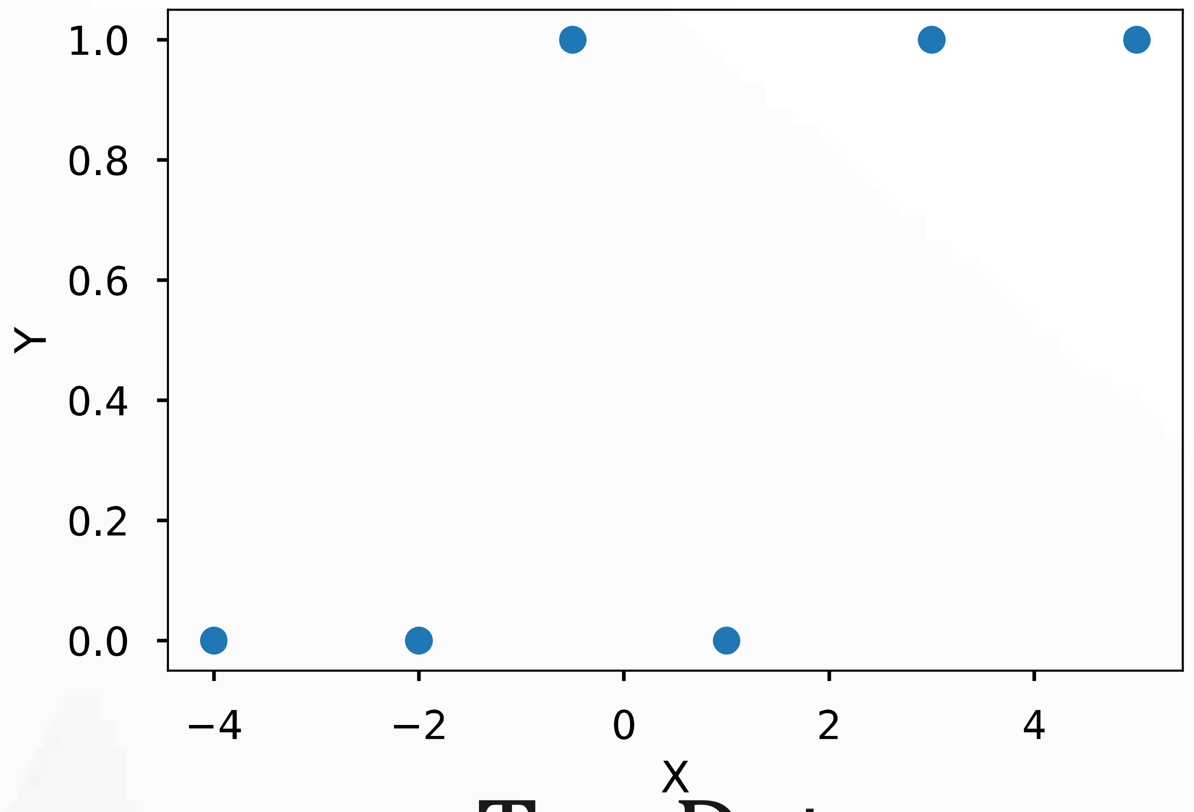
$$\ell(h(\boldsymbol{x}_i), y_i) = \begin{cases} -\log\left[\theta\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i\right)\right], & y_i = 1 \\ -\log\left[1-\theta\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i\right)\right], & y_i = 0 \end{cases}$$



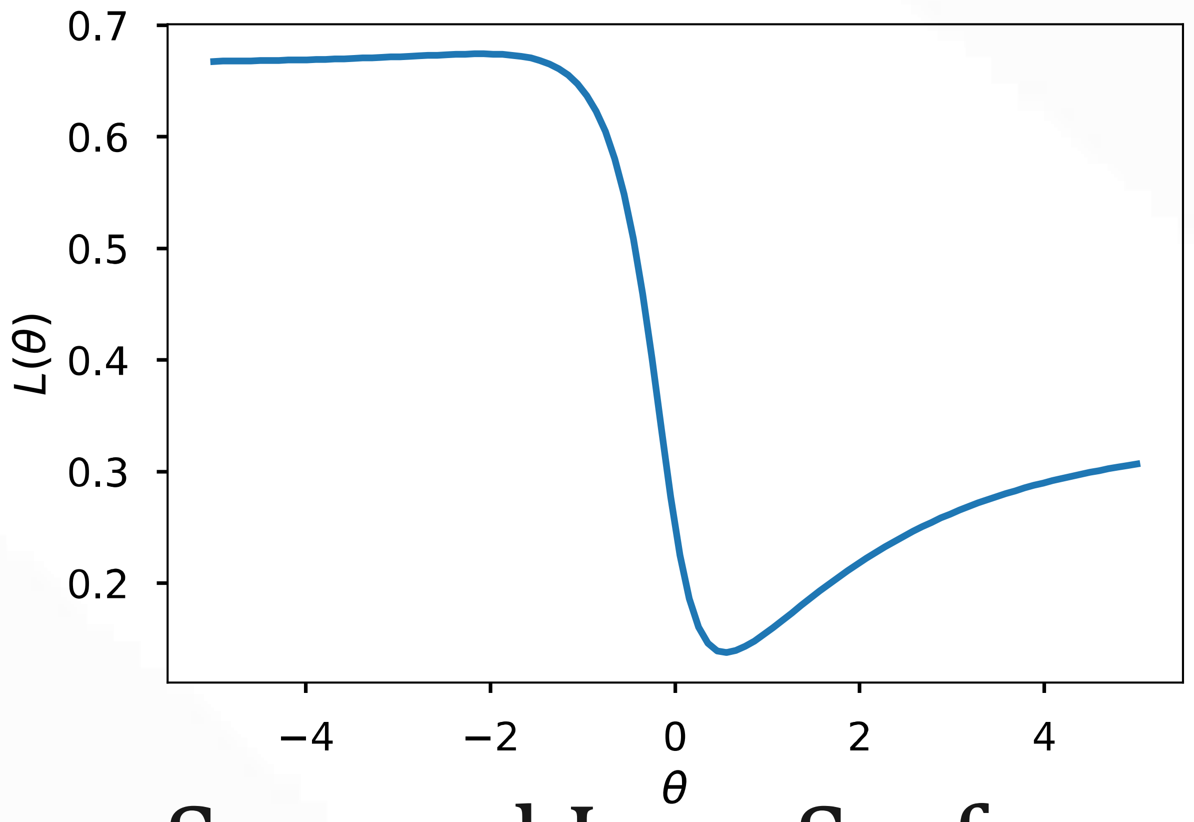
- If $y_i = 1$,

$$\theta\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i\right) \to 0, \text{loss} \to \infty$$

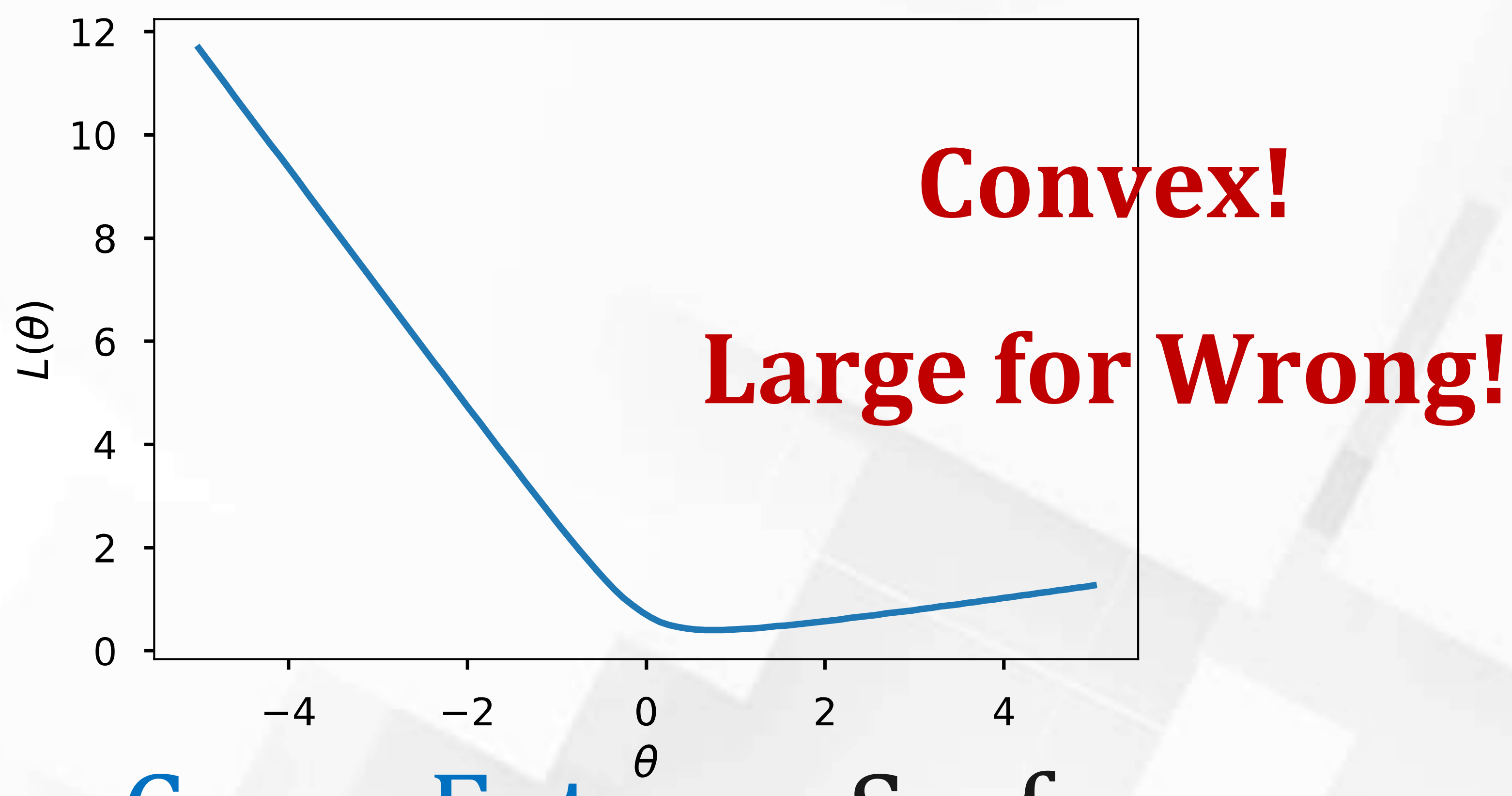
$$\theta\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i\right) \to 1, \text{loss} \to 0$$



Toy Data



Squared Loss Surface



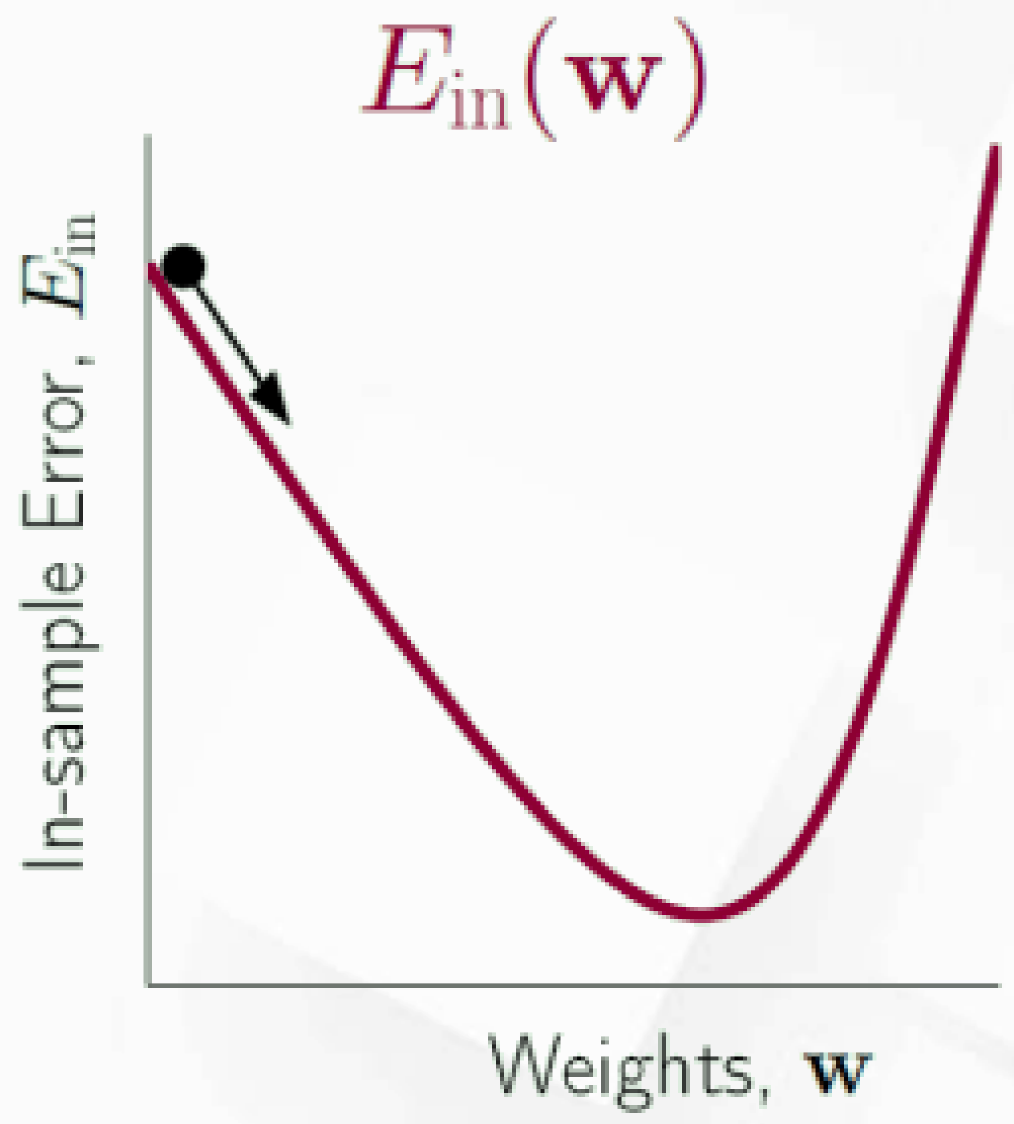
Cross-Entropy Surface

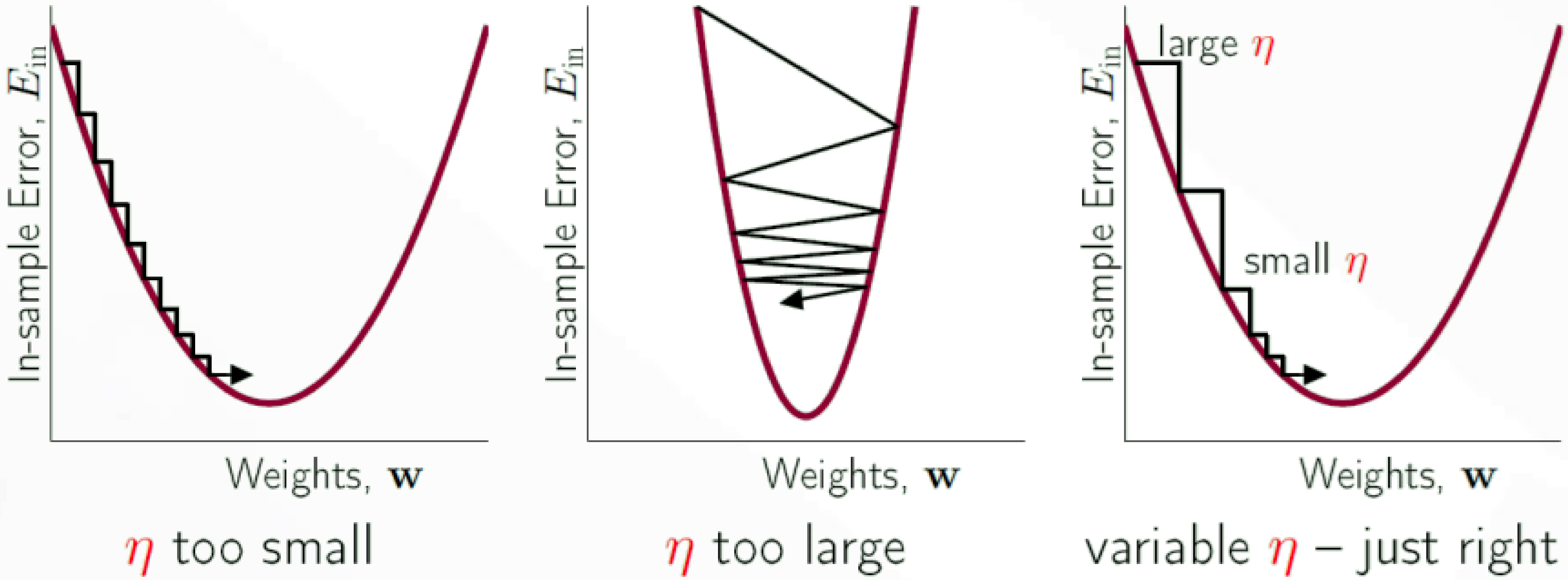
Optimization – Gradient Descent

- We can use gradient descent to find for the optimized parameter $\boldsymbol{w}$

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) + \eta\tilde{\boldsymbol{v}}$$

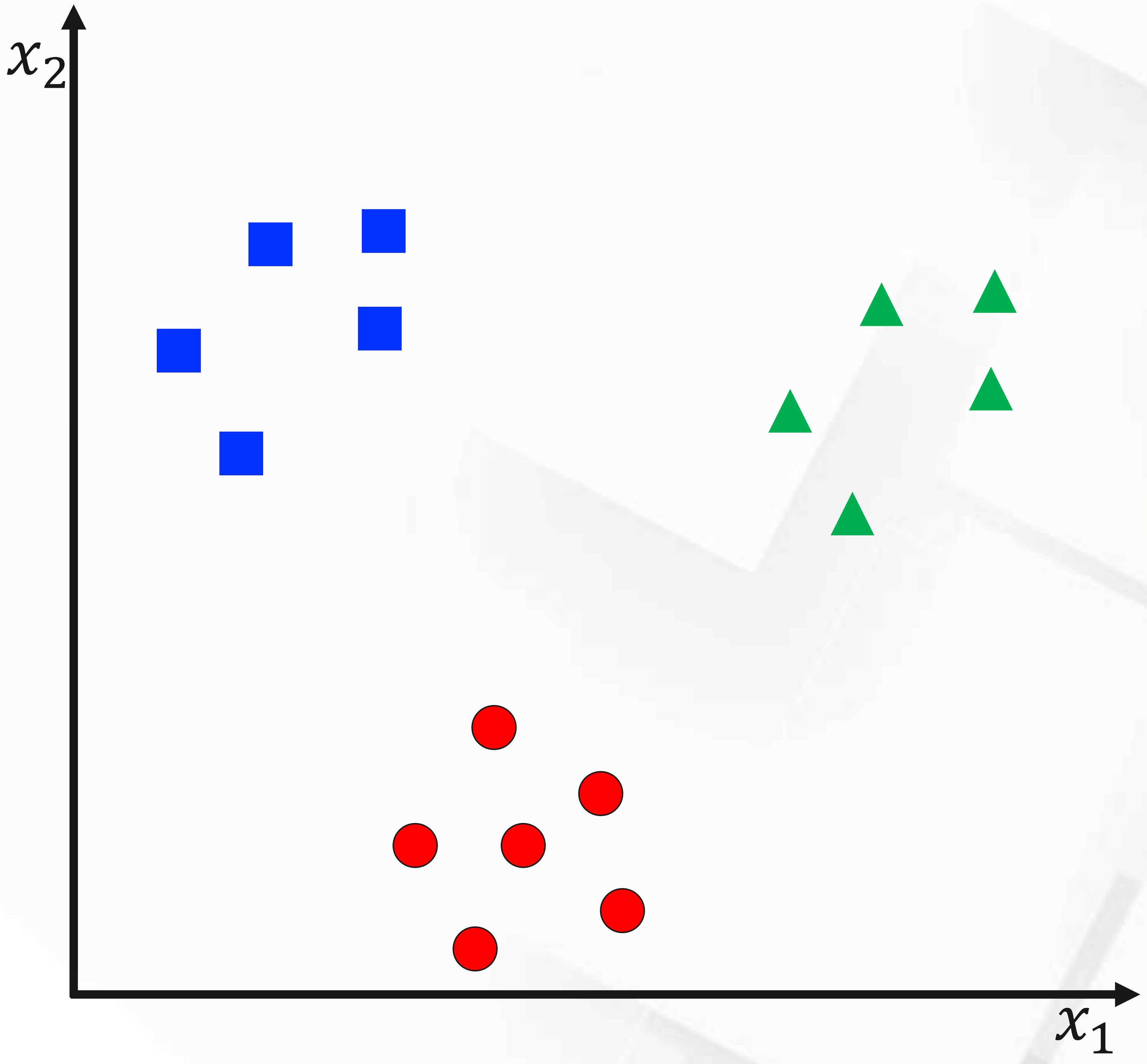
where η is the learning rate and $\tilde{\boldsymbol{v}} = -\nabla J(\boldsymbol{w}(k))$





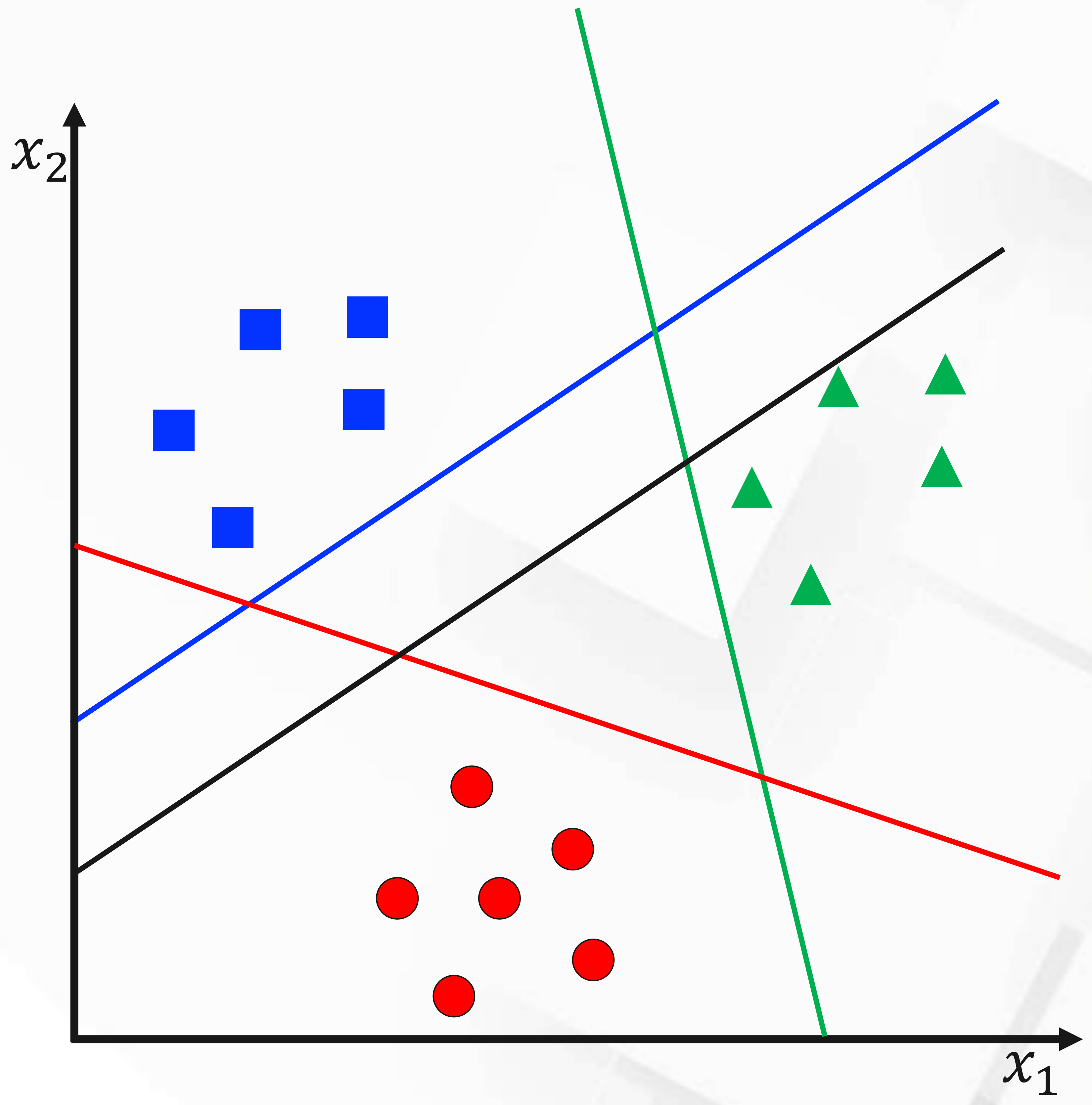
Logistic Regression for Classification

Here we go again… What if we have more than 2 classes?



Generalization to K-classes

- In logistic regression, we assumed that labels were binary (i.e., $y_i \in \{0, 1\}$).
- We can also generalize logistic regression to the case where we want to handle multiple classes.
  - This generalized version of logistic regression is called **Softmax Regression**
- In the softmax regression setting, we are interested in **multi-class classification** (as opposed to only binary classification), and so y can take on K different classes, rather than only two.



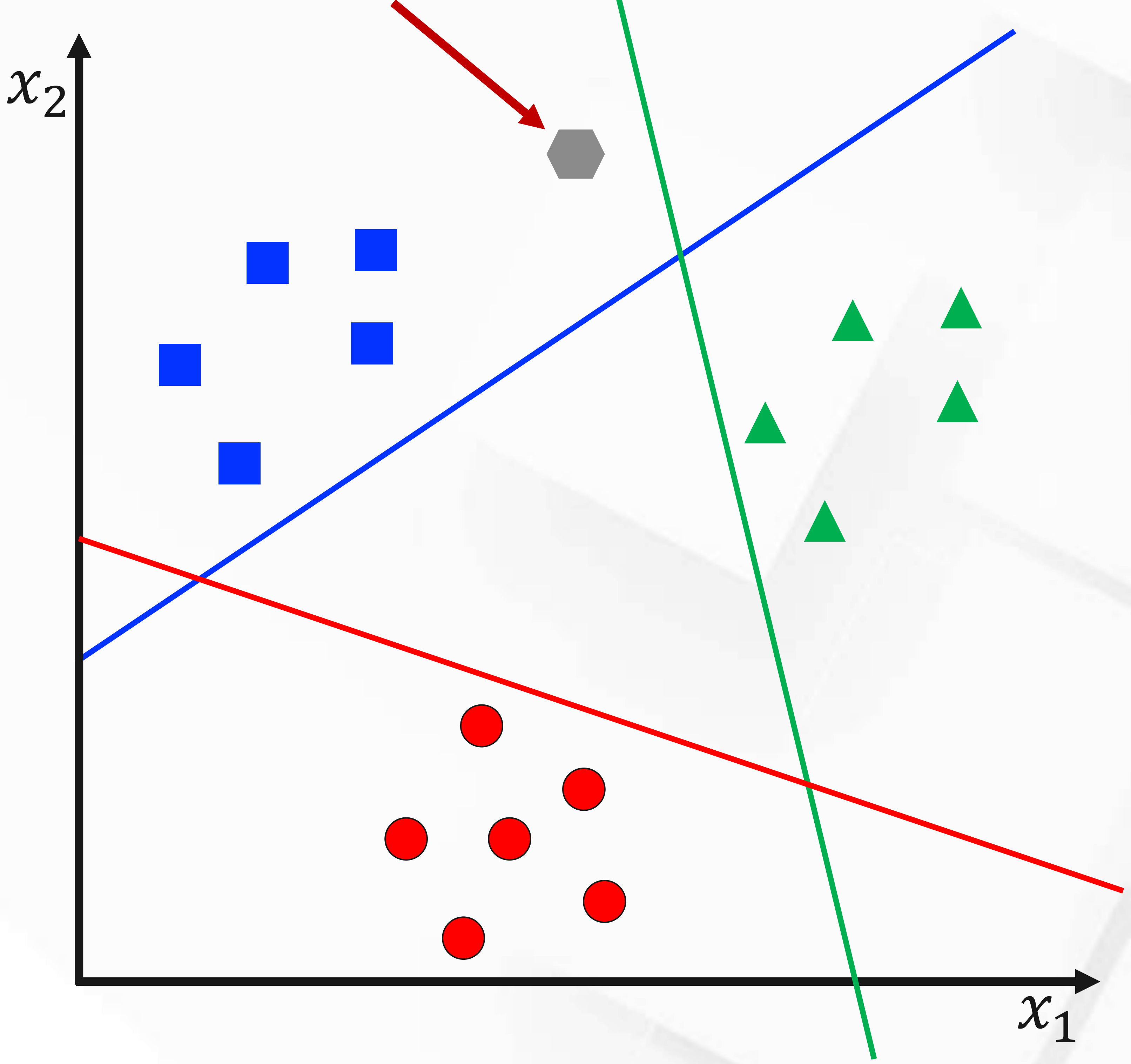
Generalization to K-classes

- Assuming we have a dataset for multiclass classification $\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2) \ldots (\boldsymbol{x}_N, y_N)\}$, where $\boldsymbol{x}_i \in \Re^{d+1}$ and $y_i \in \{0, 1, \ldots, K\}$, we want to optimize the parameters of the weight matrix $\boldsymbol{W}$ to obtain the class that has the highest probability.

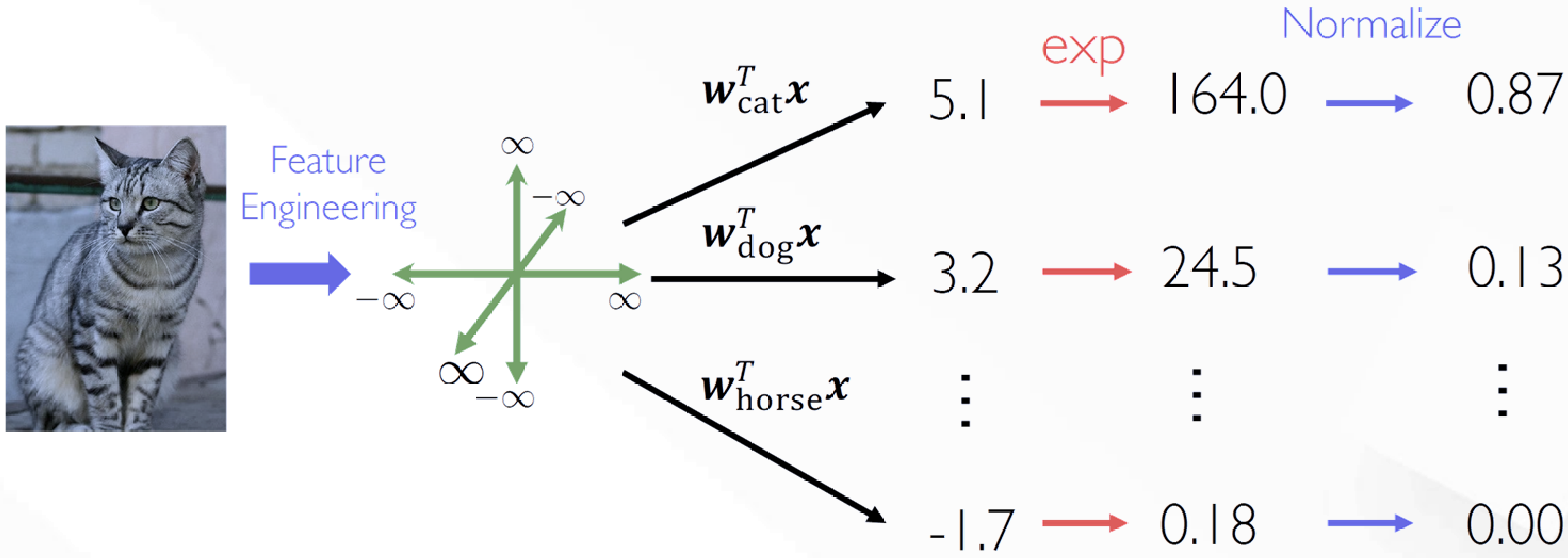
- The probability of an input $\boldsymbol{x}$ being class k is denoted as:

$$P(y = k|\boldsymbol{x}; \boldsymbol{W}) = \frac{e^{\boldsymbol{w}_k{}^{\mathrm{T}}\boldsymbol{x}}}{\underbrace{\sum_{i=1}^{K}}_{\text{Sum over all classes}} e^{\boldsymbol{w}_i{}^{\mathrm{T}}\boldsymbol{x}}}$$

Among all the shapes, what is the probability of the new sample being a square?



Generalization to K-classes



- Given a test input $\boldsymbol{x}$, we want our hypothesis to estimate the probability of each classes

$$h_{W}(\boldsymbol{x}) = \begin{bmatrix} P(y=1|\boldsymbol{x};\boldsymbol{W}) \\ P(y=2|\boldsymbol{x};\boldsymbol{W}) \\ \vdots \\ P(y=K|\boldsymbol{x};\boldsymbol{W}) \end{bmatrix} = \frac{1}{\sum_{k=1}^{K} e^{w_k^{\mathrm{T}} x}} \begin{bmatrix} e^{w_1{}^{\mathrm{T}} x} \\ e^{w_2{}^{\mathrm{T}} x} \\ \vdots \\ e^{w_k{}^{\mathrm{T}} x} \end{bmatrix}. \qquad \text{where } \boldsymbol{W} = \begin{bmatrix} | & & | \\ \boldsymbol{w}_1 & \cdots & \boldsymbol{w}_k \\ | & & | \end{bmatrix}$$

Generalization to K-classes

- Similar to 2-class, we can also use **maximum likelihood** to determine the parameters weight matrix $\boldsymbol{W} = \{\boldsymbol{w}_1, \boldsymbol{w}_2, \dots, \boldsymbol{w}_k\}$.

- In multiclass, the likelihood function can be written as:

$$\max_{\boldsymbol{w}_1, \boldsymbol{w}_2, \dots, \boldsymbol{w}_k} \prod_{i=1}^{N} \prod_{k=0}^{K} P(y_i = k | \boldsymbol{x}_i; \boldsymbol{W})^{\mathbf{1}(y_i = k)}$$

- We can use minimum negative log-likelihood estimation:

$$\boxed{\min_{\boldsymbol{W}} J(\boldsymbol{W}) = \min_{\boldsymbol{w}_1, \boldsymbol{w}_2, \dots, \boldsymbol{w}_k} -\frac{1}{N} \sum_{i=1}^{N} \sum_{k=0}^{K} \mathbf{1}(y_i = k) \cdot \log \frac{e^{\boldsymbol{w}_k{}^{\mathrm{T}} \boldsymbol{x}}}{\sum_{j=1}^{K} e^{\boldsymbol{w}_j^{\mathrm{T}} \boldsymbol{x}}}}$$

Generalization to K-classes

Loss for each data point $(\boldsymbol{x}_i, y_i)$:

$$\ell(h_{\boldsymbol{W}}(\boldsymbol{x}_i), y_i) = \begin{cases} -\log \dfrac{e^{\boldsymbol{w}_1{}^{\mathrm{T}} x}}{\sum_{j=1}^{K} e^{\boldsymbol{w}_j^{\mathrm{T}} \boldsymbol{x}}}, & y_i = 1 \\ -\log \dfrac{e^{\boldsymbol{w}_2{}^{\mathrm{T}} x}}{\sum_{j=1}^{K} e^{\boldsymbol{w}_j^{\mathrm{T}} \boldsymbol{x}}}, & y_i = 2 \\ \vdots & \\ \vdots & \\ -\log \dfrac{e^{\boldsymbol{w}_K{}^{\mathrm{T}} x}}{\sum_{j=1}^{K} e^{\boldsymbol{w}_j^{\mathrm{T}} \boldsymbol{x}}}, & y_i = K \end{cases}$$

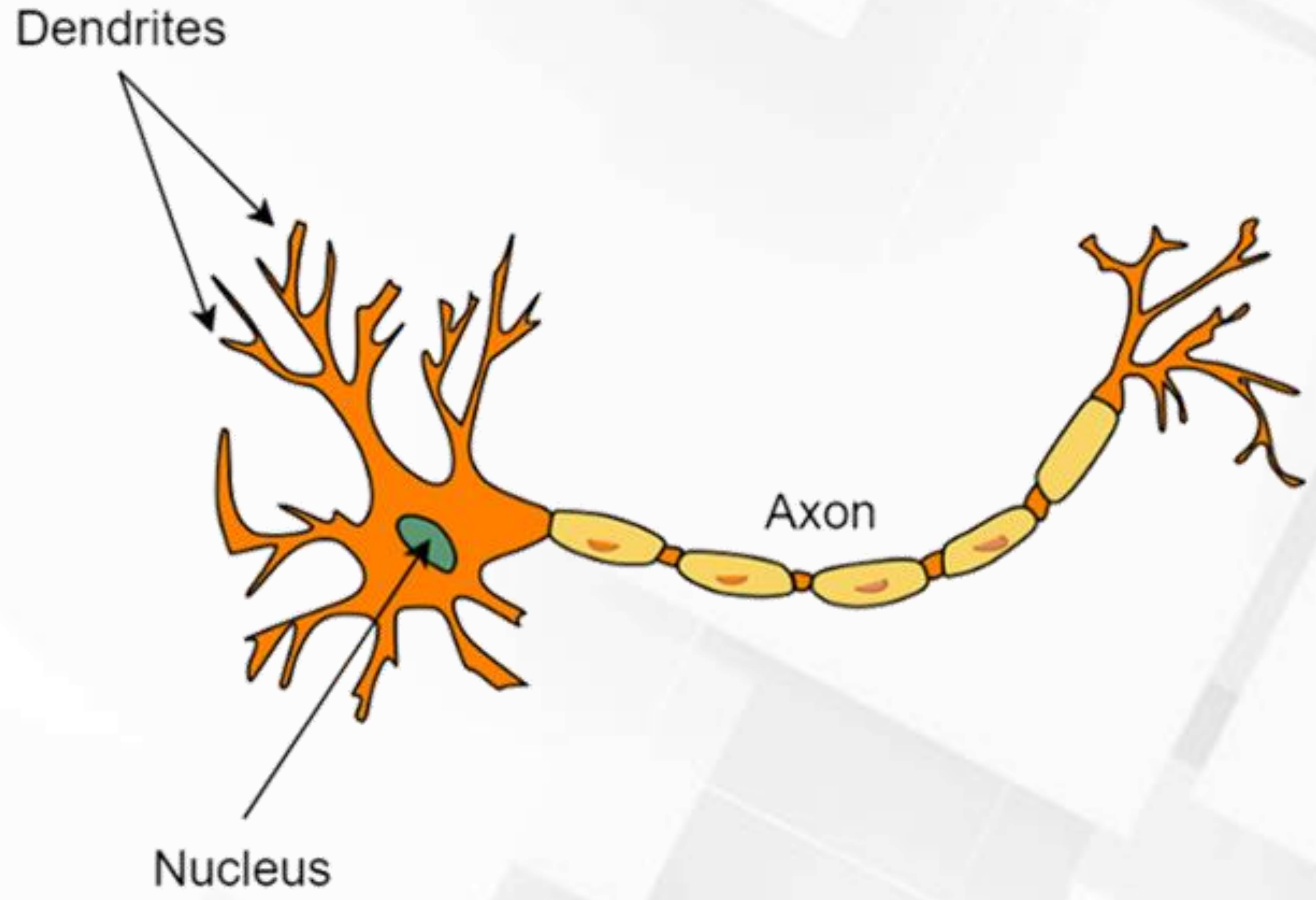
Linear Model Outline

Tsinghua University

1. Linear Regression
2. Linear Discriminant Analysis
3. Logistic Regression
4. Perceptron

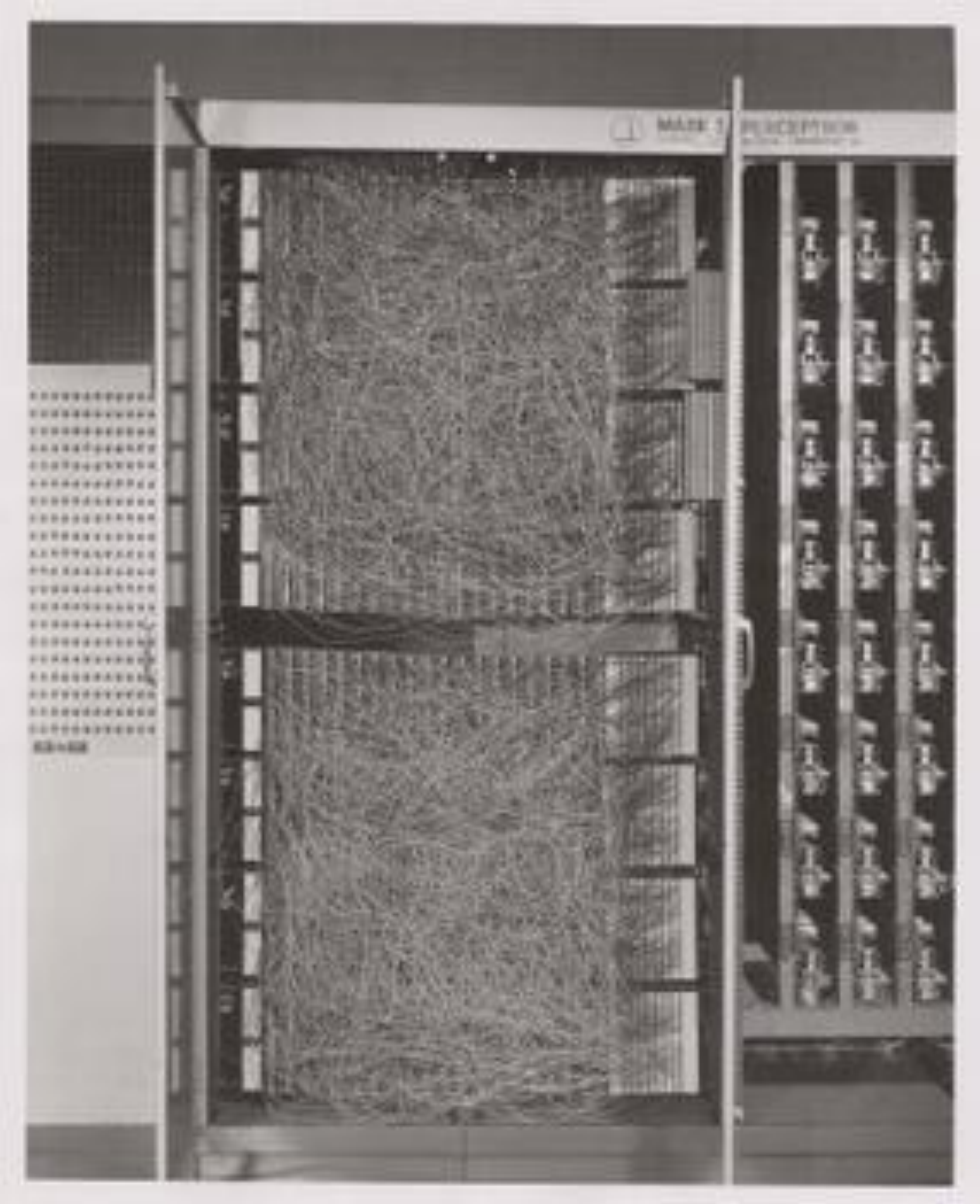
Perceptron

- Perceptron was a **learning machine** invented in 1943 by McCulloch and Pitt
- The first implementation was a machine built in 1958 at the Cornell Aeronautical Laboratory by Frank Rosenblatt, funded by the United States Office of Naval Research.
- This machine was designed for **image recognition**: it had an array of 400 photocells, randomly connected to the "neurons".



Perceptron

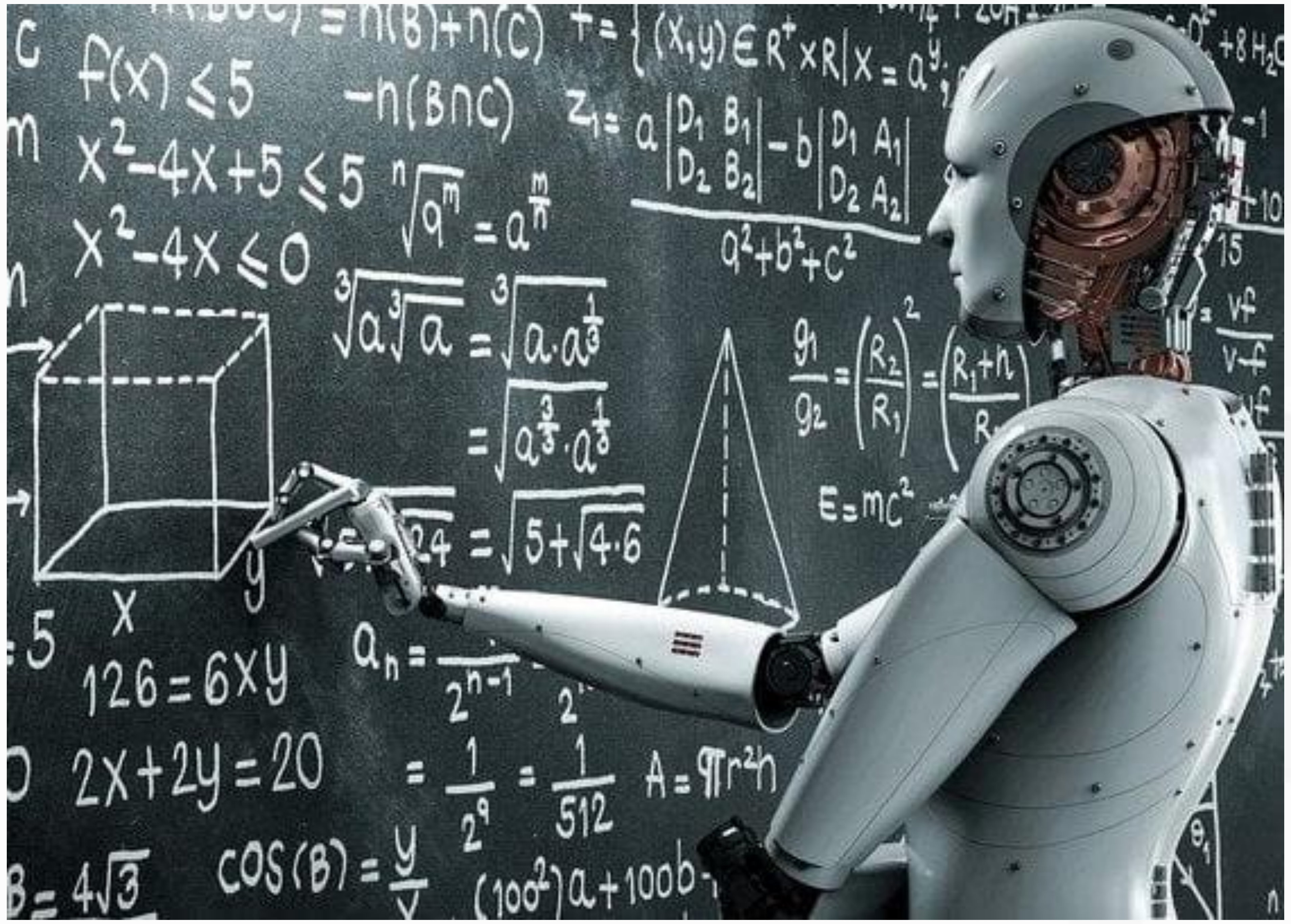
- The perceptron is a simplified model of a biological neuron. While the complexity of biological neuron models is often required to fully understand neural behavior, research suggests a perceptron-like linear model can produce some behavior seen in real neurons.
- Weights were encoded in potentiometers, and weight updates during learning were performed by electric motors.



Perceptron

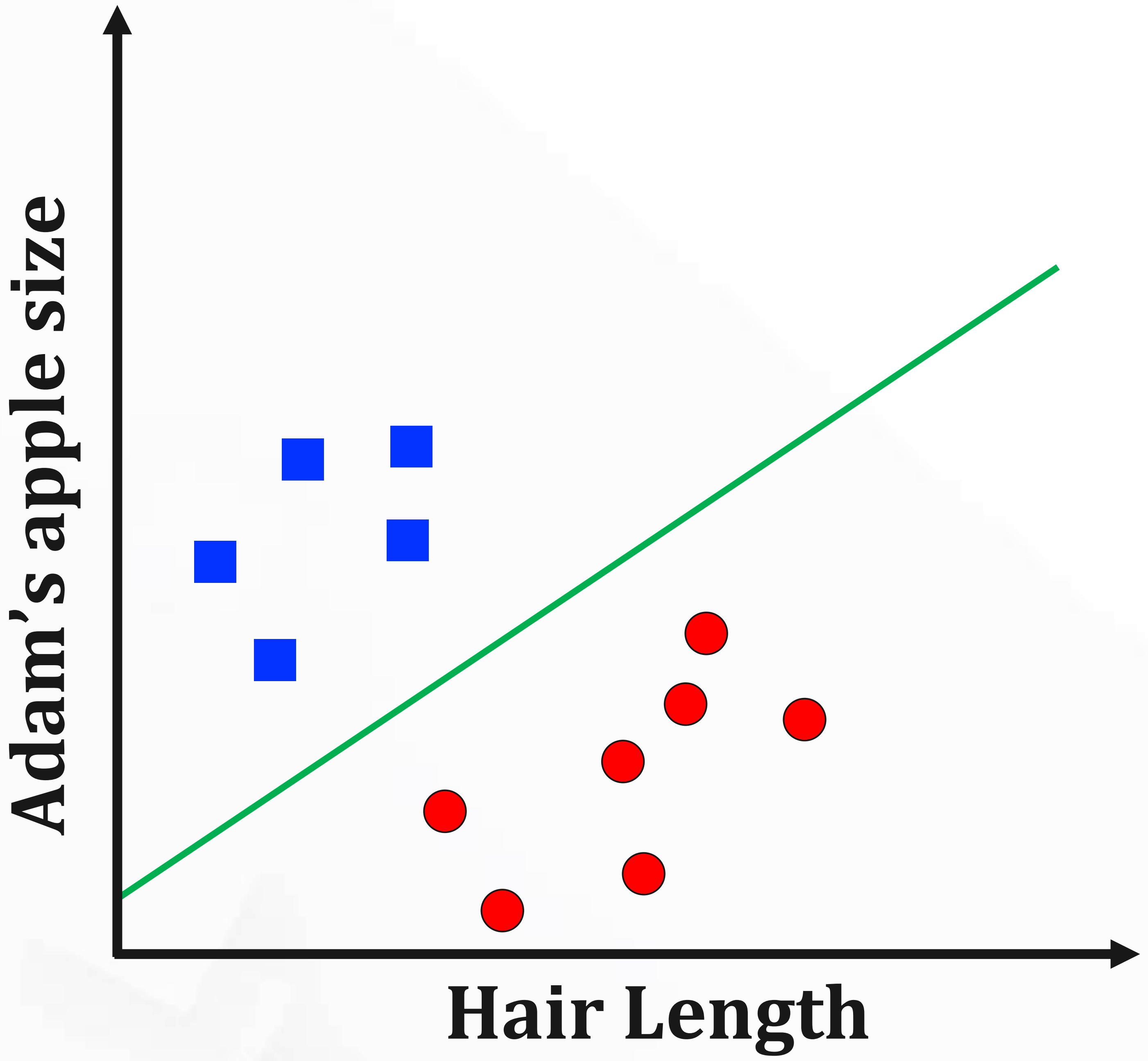
Why is it called learning machine?

- It was first intended to be a **machine**, rather than a program!
- Perceptron learns!
- Perceptron algorithm learns the parameters from the given samples during the learning(training) process of the neural network
- The parameters (weights $(w_1, \ldots, w_d)$ and bias w_0) do not have to be hard coded!



Perceptron

- How do we teach a kid to identify the gender?



For the green line:

$$w_0 + w_1 x^{(1)} + w_2 x^{(2)} = 0$$

For each blue dots:

$$w_0 + w_1 x^{(1)} + w_2 x^{(2)} > 0$$

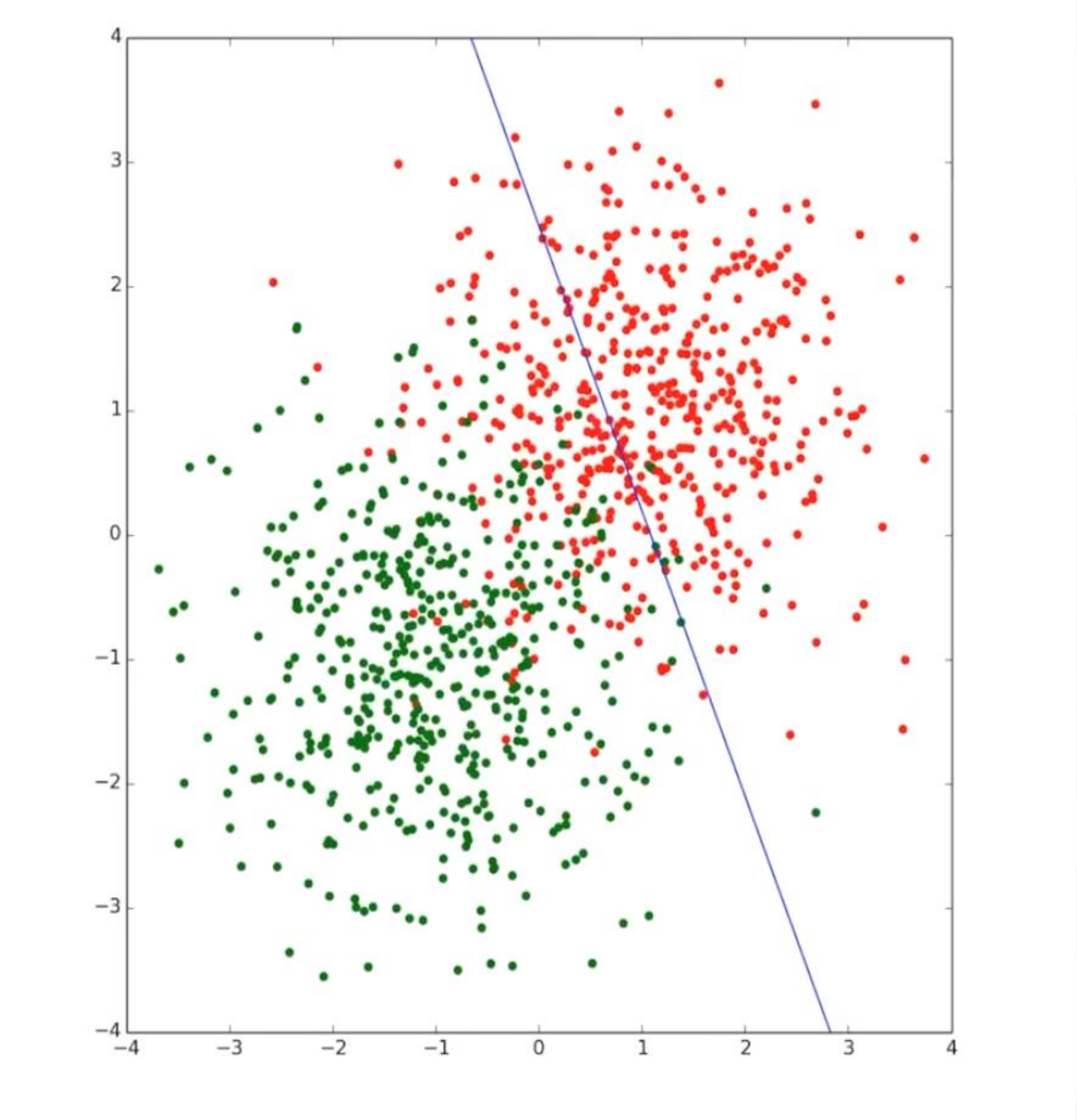
For each red dots:

$$w_0 + w_1 x^{(2)} + w_2 x^{(2)} < 0$$

Short Clip on How Perceptron Works

Tsinghua University

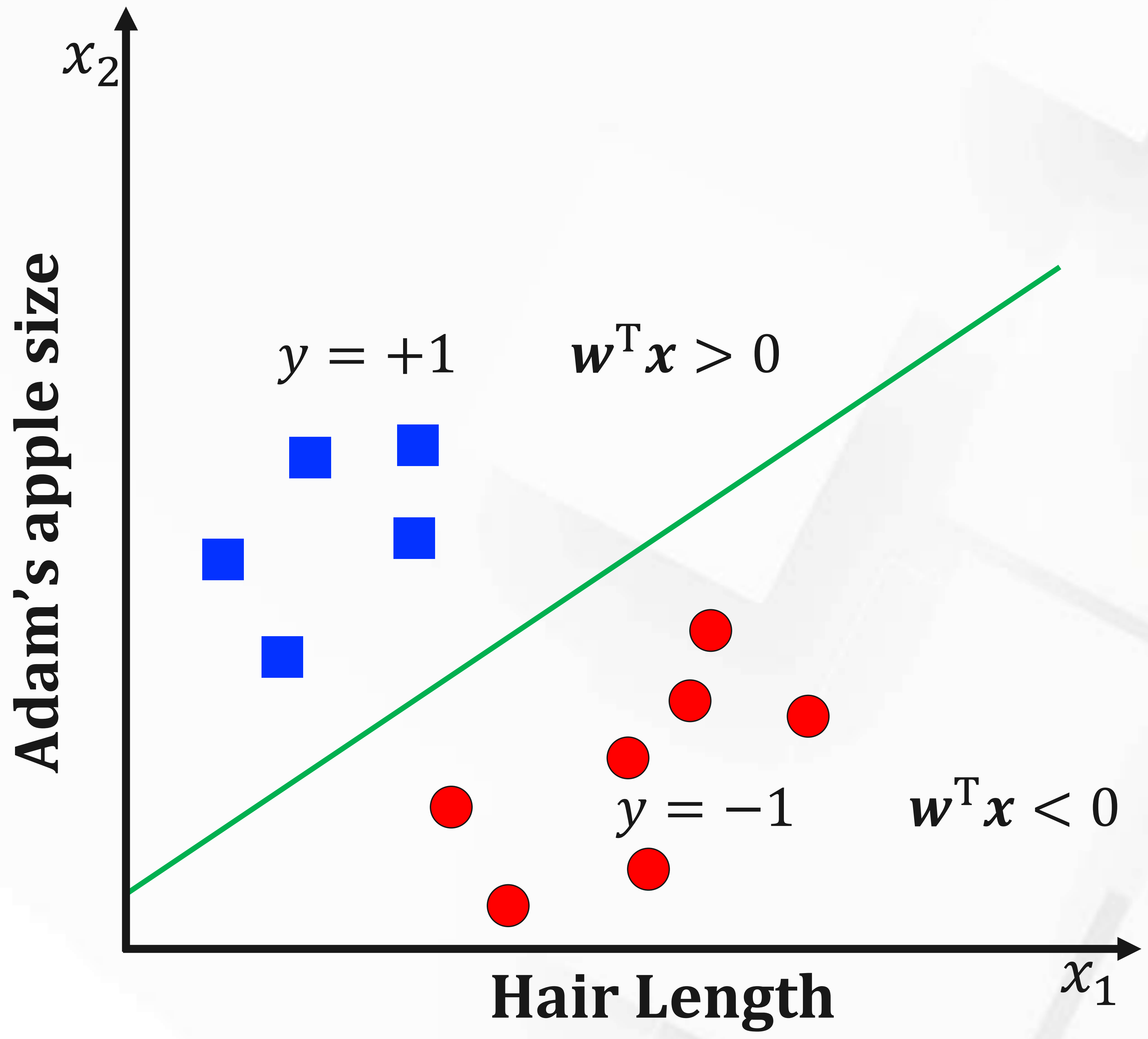
The decision boundary adjusts during training!



Credits: https://github.com/ayusek/Perceptron-Animation

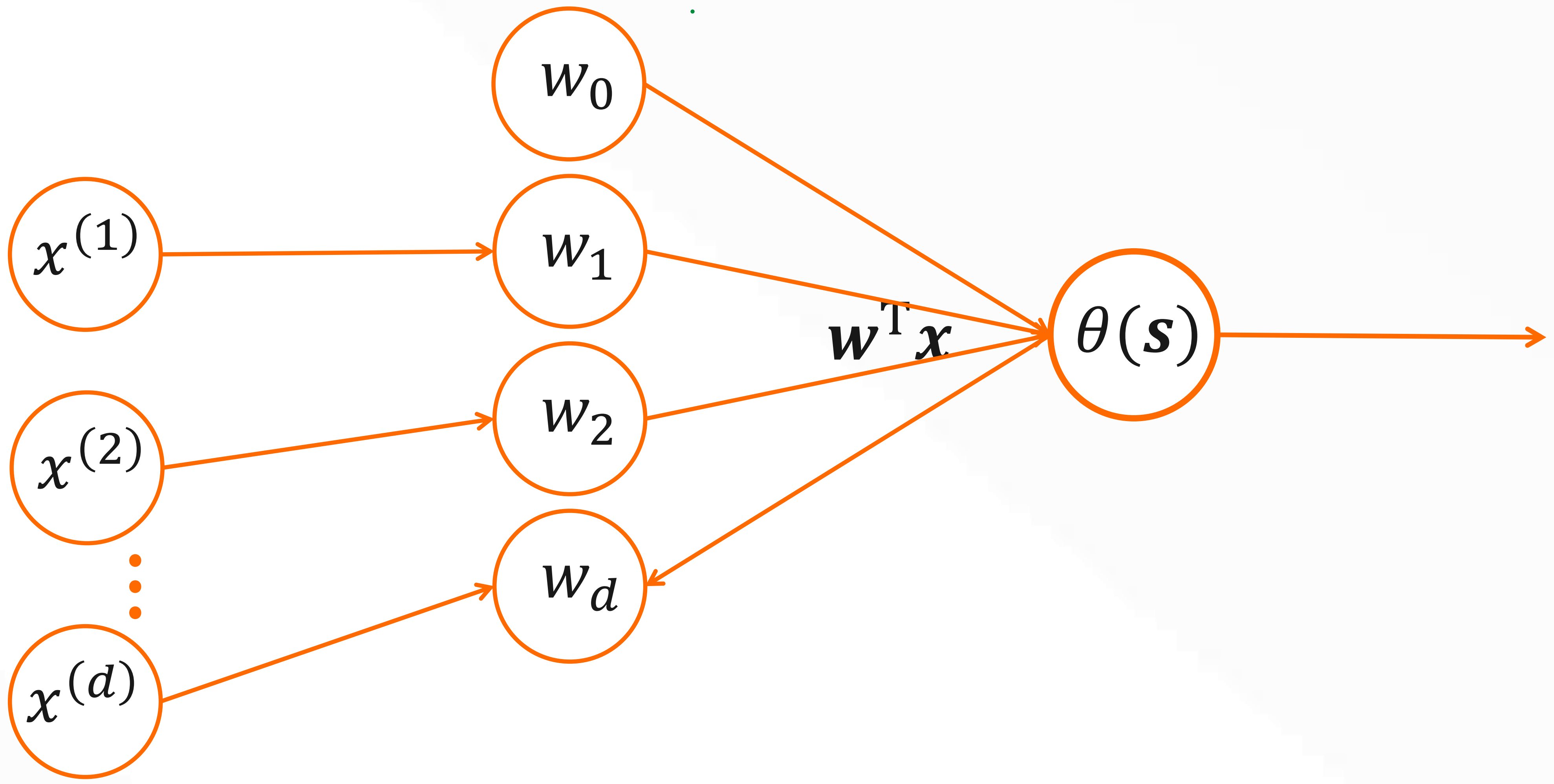
Perceptron - Concept

- Given training data $\{(x_i, y_i): 1 \leq i \leq n\}$ *i.i.d.* from distribution D
- **Hypothesis** $f_{\boldsymbol{w}}(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$
  - $y = +1$ if $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} > 0$
  - $y = -1$ if $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} < 0$
- Then we can **predict** based on the sign of y
  - $y = \mathrm{sign}(f_{\boldsymbol{w}}(\boldsymbol{x})) = \mathrm{sign}(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x})$



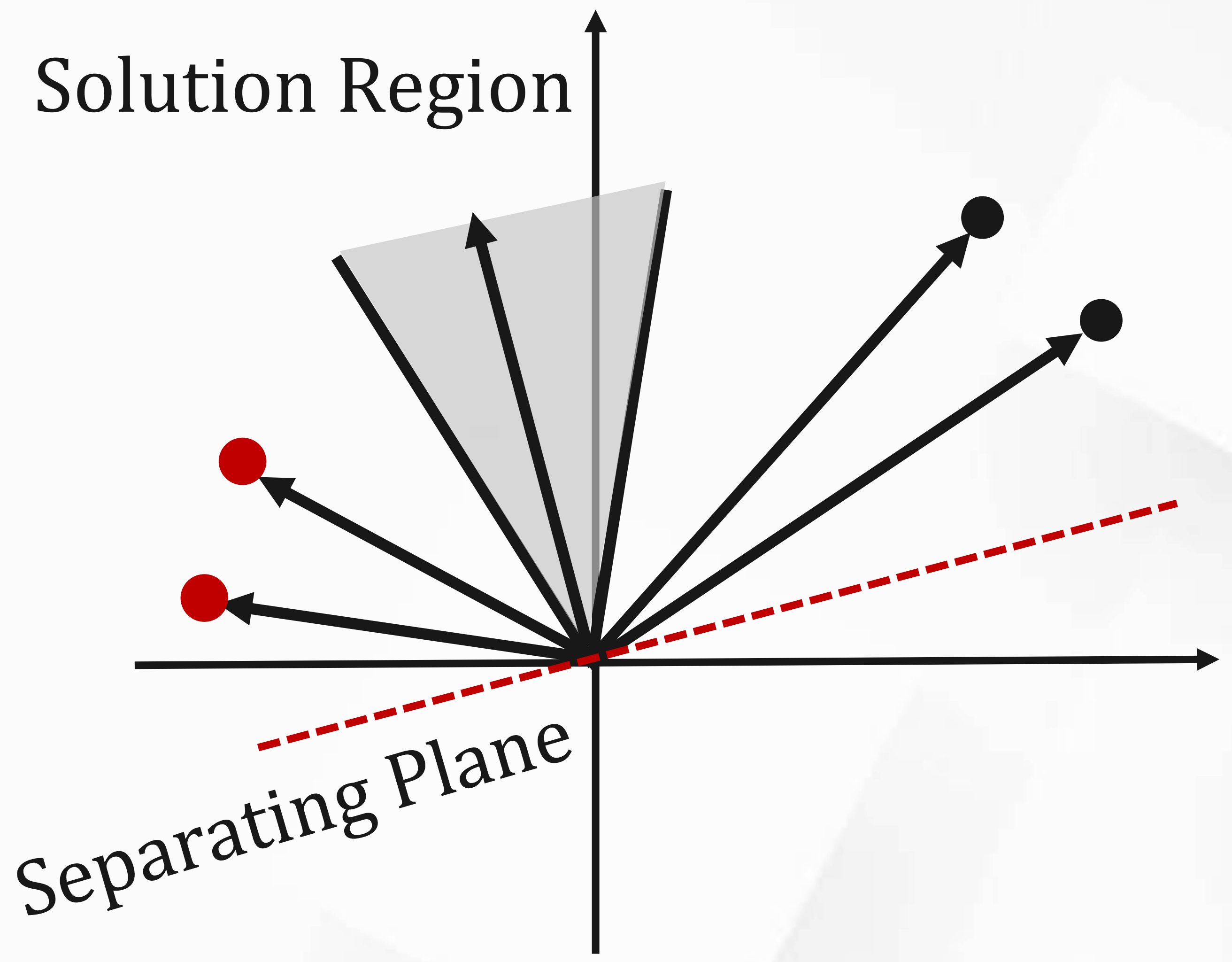
Perceptron - Concept

- Perceptron is a machine that can learn from the input data
  - $y = \text{sign}\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}\right) = \text{sign}\left(\sum_{i=0}^{d} w_i x^{(i)}\right)$
- The perceptron machine adjusts its parameter $\boldsymbol{w}$ by using the given data $(\boldsymbol{x}_i, y_i)$



- **Goal:** We want the perceptron to minimize the classification error $(J(\boldsymbol{w}))$ between our predicted (y) with parameter $(\boldsymbol{w})$ and the ground truth (y_i) of the given data

Perceptron - Concept



- We first let

$$\hat{\boldsymbol{x}}_i = \begin{cases} -\boldsymbol{x}_i, & \boldsymbol{x}_i \in \omega_1, \\ \boldsymbol{x}_i, & \boldsymbol{x}_i \in \omega_2, \end{cases} \qquad i = 1, \dots, N$$

  - Such that for every correctly classified sample *i*, we get $\boldsymbol{w}^{\mathrm{T}} \hat{\boldsymbol{x}}_i > 0$
  - Thus, every weight vector $\boldsymbol{w}^*$ that satisfies $\boldsymbol{w}^{\mathrm{T}} \hat{\boldsymbol{x}}_i > 0$ is called the **solution vector**
  - Then, the region intercepted by all solution vectors is called the **solution region**

Perceptron Loss Function

- When having a sample set $\mathcal{X}^k$, we can update the weights based on the **perceptron criterion**
- The perceptron algorithm attempts to minimize the **perceptron criterion**, for perceptron the objective loss function is defined as

$$J_{\mathrm{p}}(\boldsymbol{w}) = \sum_{\hat{\boldsymbol{x}}_j \in \mathcal{X}^k} \left(-\boldsymbol{w}^T \hat{\boldsymbol{x}}_j\right)$$

where $\mathcal{X}^k$ is the misclassified sample set at step k

- Perceptron Algorithm wants to minimize the errors as much as possible:

$$J_{\mathrm{p}}(\boldsymbol{w}^*) = \min J_{\mathrm{p}}(\boldsymbol{w})$$

Optimization: Gradient Descent

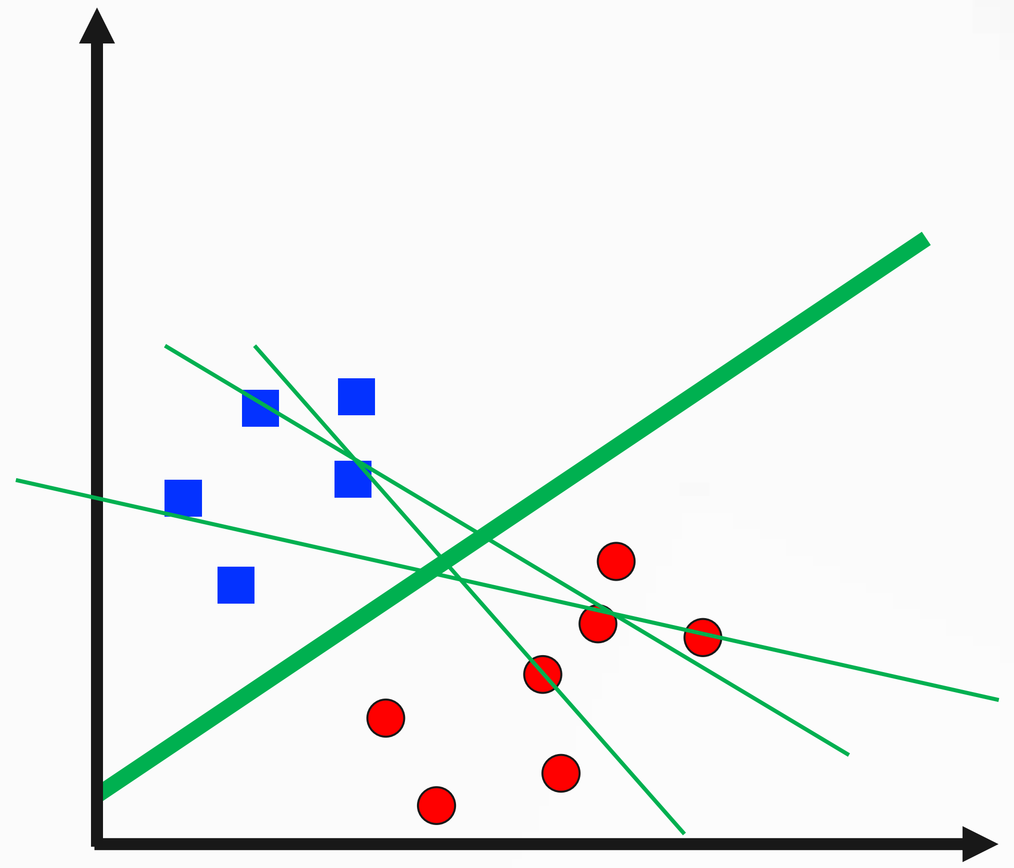
- Given $J_{\mathrm{p}}(\boldsymbol{w}^*) = \min J_{\mathrm{p}}(\boldsymbol{w}) = 0$
- We can use gradient descent to solve for $\boldsymbol{w}^*$:

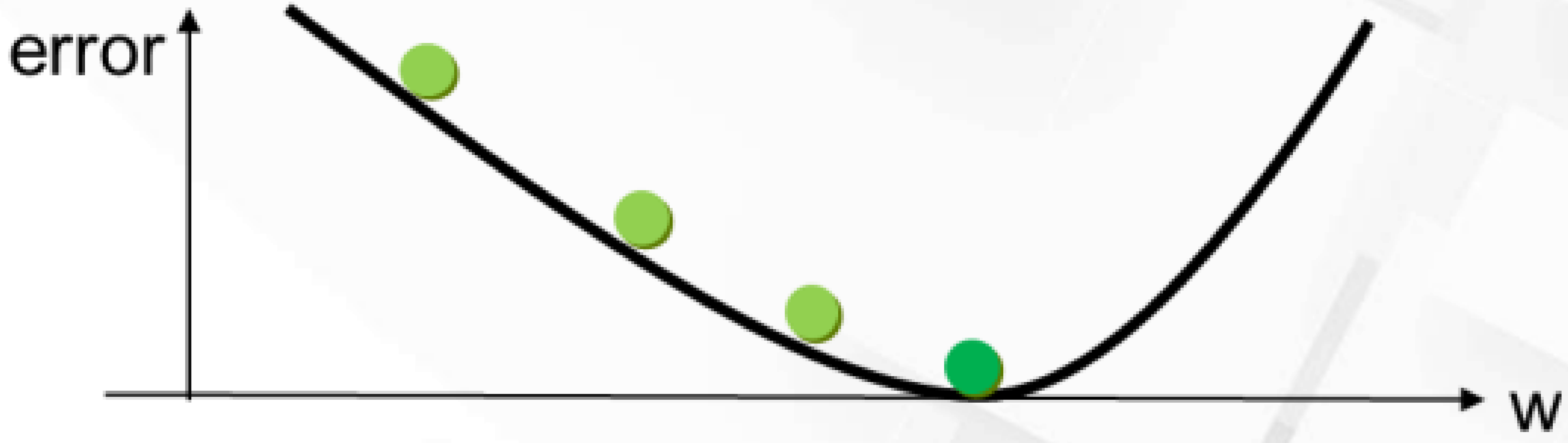
$$\boldsymbol{w}_{\mathrm{k+1}} = \boldsymbol{w}_k - \rho_k \nabla J_{\mathrm{p}}$$

$$\nabla J_{\mathrm{p}} = \frac{\partial J_{\mathrm{p}}(\boldsymbol{w})}{\partial \boldsymbol{w}} = \sum_{\boldsymbol{x}_j \in \mathcal{X}^k} (-\widehat{\boldsymbol{x}}_j)$$

$$\Longrightarrow \boldsymbol{w}_{\mathrm{k+1}} = \boldsymbol{w}_k + \rho_k \sum_{\boldsymbol{x}_j \in \mathcal{X}^k} (-\widehat{\boldsymbol{x}}_j)$$

ρ_k: Step size





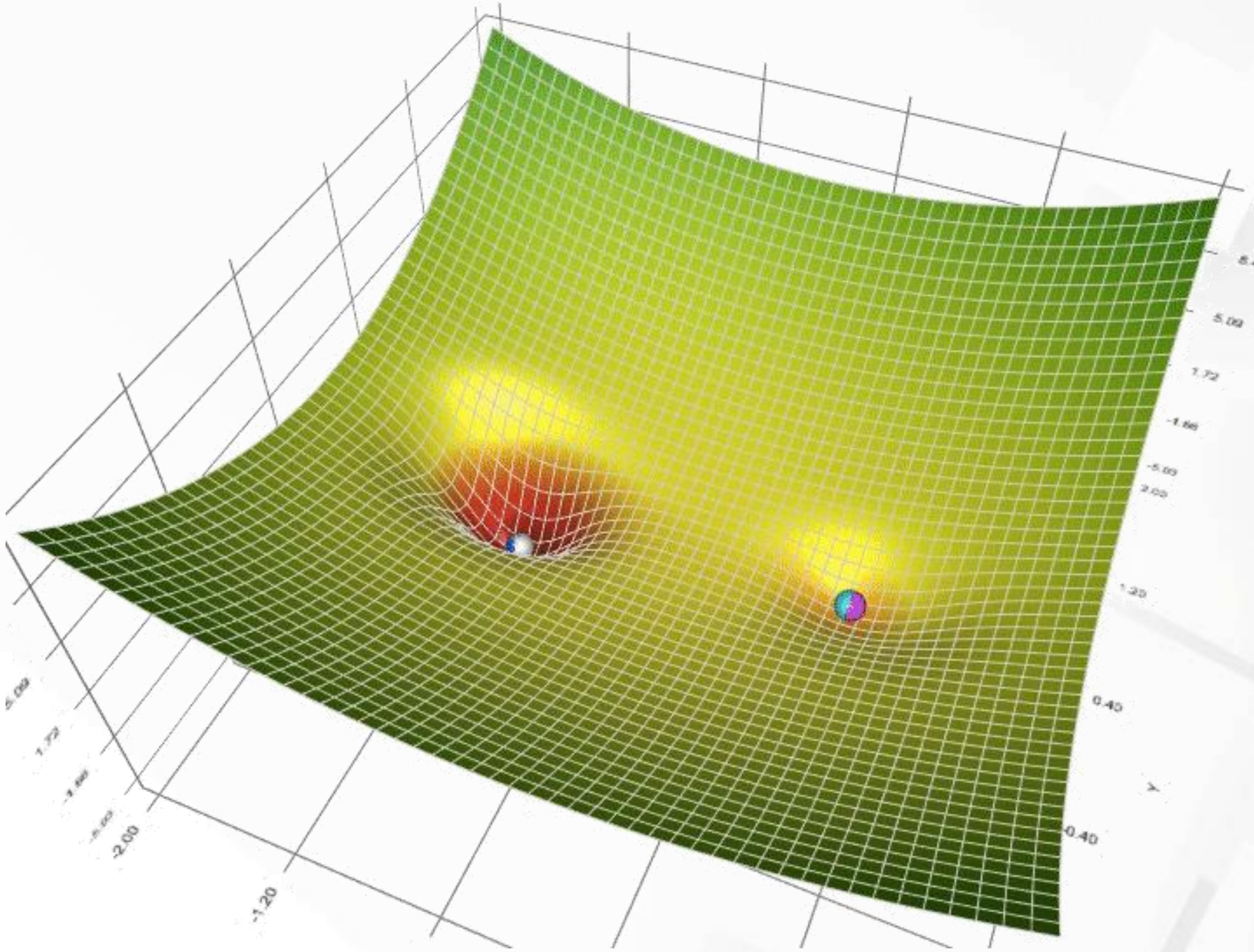
Optimization - Algorithm

- Assuming we have single sample at each step t, then the perceptron algorithm is:

Perceptron Algorithm
1. Start with the all-zeroes weight vector $\boldsymbol{w} = 0$ and initialize t to 1.
2. Given example $\boldsymbol{x}$, check if predicted correctly (*i.e.* $\boldsymbol{w}^{\mathrm{T}}\hat{\boldsymbol{x}} > 0$).
If Predicted Correctly: skip step (3)
Else: continue
3. On a mistake, update as follows:
Mistake on positive: $\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t + \boldsymbol{x}$
Mistake on negative: $\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t - \boldsymbol{x}$
$t \leftarrow t + 1$
Repeat Step (2) until $\boldsymbol{J}_{\mathrm{p}} = 0$

Perceptron Convergence

- The perceptron convergence theorem states that if there exists an exact solution (i.e. , if the training data set is linearly separable), then the perceptron learning algorithm is guaranteed to find an exact solution in a finite number of steps.



- We can also use variable step size methods, such as setting step size as

$$\rho_k = \frac{\left|w(k)^T \boldsymbol{x}_j\right|}{\left\|\boldsymbol{x}_j\right\|^2}$$

Limitations of Perceptron

Tsinghua University

Limitations

Possible Solutions?

Perceptron cannot handle nonlinearly separable samples

- Allow minimal errors
- Try other non-linear methods

Perceptron does not provide probabilistic outputs, nor does it generalize readily to K >2 classes.

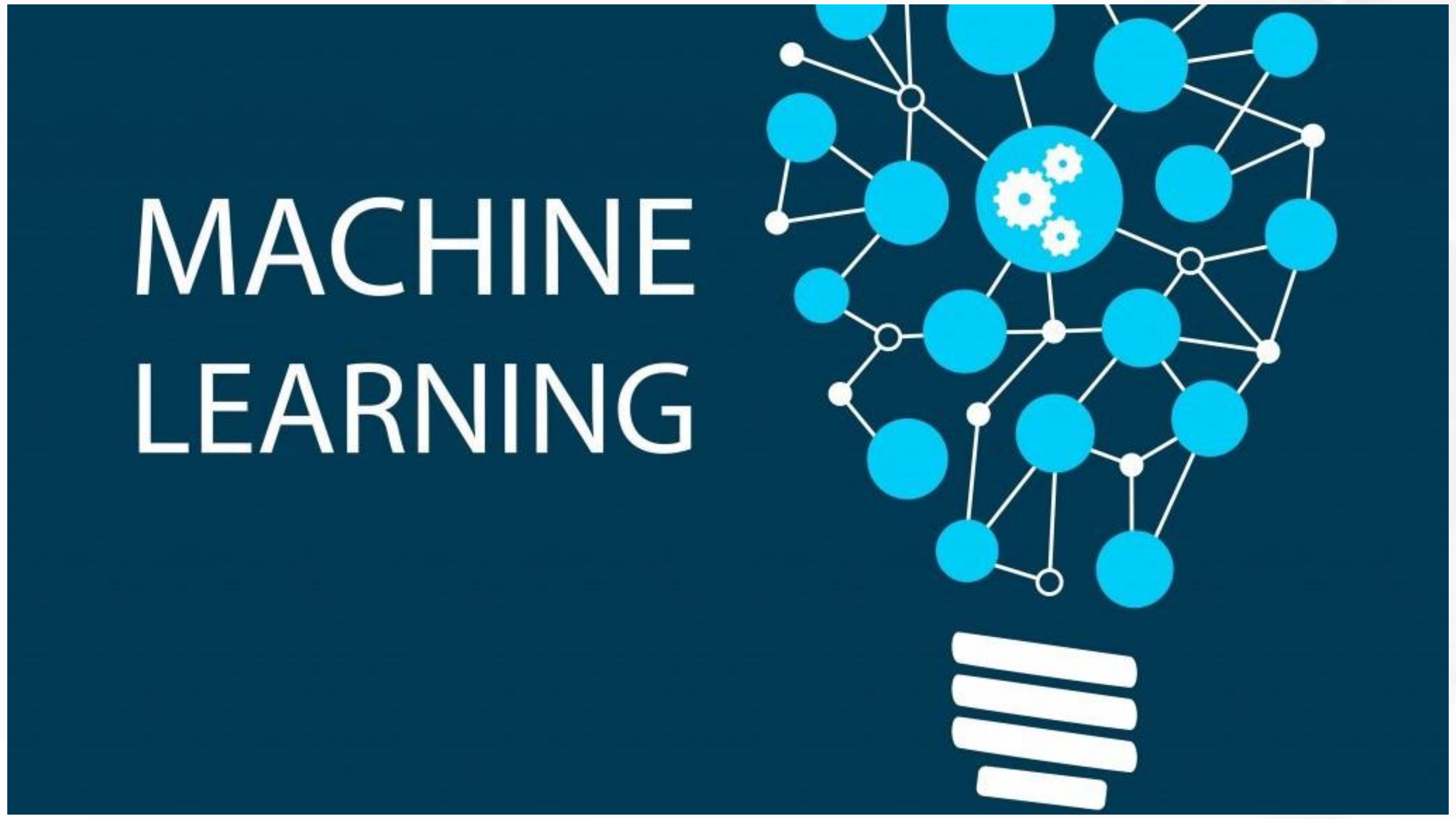
- Go for other multiclass classifiers
- Use 2-class classifier to perform multiclass classification

Linearly separable but have multiple solutions?

- Checkout "optimal classifiers" (e.g. **SVM – We save it for Next Meeting!)**

Basic Building Blocks of Machine Learning

- How to create a learning machine?
  - It needs a teacher
    - We design it! (Features and Models)
  - It needs learning materials
    - Training Data
  - We need to set a learning target
    - Target Function or Learning Criterion
  - We need to tell it how to learn
    - Learning/Training Algorithms



Summary

清華大學 Tsinghua University

How to create a learning machine?

Basic Building Block of Machine Learning	Linear Regression	Logistic Regression	Perceptron
Model	$f(x) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$	$h(x) = \dfrac{e^{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}}}{1 + e^{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}}}$	$y = \mathrm{sign}(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x})$
Training Data	$(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_N, y_N)$ $\boldsymbol{x}_i \in \Re^{d+1}, y_i \in \Re$	$(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_N, y_N)$ $\boldsymbol{x}_i \in \Re^{d+1}, y_i \in \{-1,1\}$	$(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_N, y_N)$ $\boldsymbol{x}_i \in \Re^{d+1}, y_i \in \{-1,1\}$
Target Function/Learning Criterion	$\min\limits_{\boldsymbol{w}} \frac{1}{N} \sum\limits_{i=1}^{N} (f(\boldsymbol{x}_i) - y_i)^2$	$\min\limits_{\boldsymbol{w}} \sum\limits_{i=1}^{N} \left\{\log\left(1 + e^{-\tilde{y}_i \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i}\right)\right\}$	$\min\limits_{\boldsymbol{w}} \sum\limits_{\hat{\boldsymbol{x}}_j \in \mathcal{X}^k} (-\boldsymbol{w}^{\mathrm{T}} \hat{\boldsymbol{x}}_j)$
Learning/Training Algorithms	$\boldsymbol{w}^* = (\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}$	$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) + \eta\tilde{\boldsymbol{v}}$	$\boldsymbol{w}_{\mathrm{k+1}} = \boldsymbol{w}_k + \rho_k \sum\limits_{\boldsymbol{x}_j \in \mathcal{X}^k} (-\hat{\boldsymbol{x}}_j)$

Thank You

Any Questions?

Gao Huang (黄高)
gaohuang@tsinghua.edu.cn

Course ID：30250293