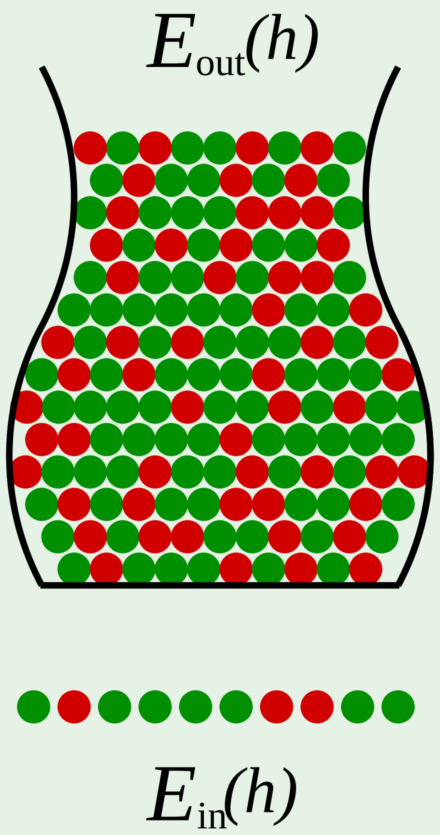
Review of Lecture 2

Is Learning feasible?

Yes, in a **probabilistic** sense.



$$\mathbb{P}\left[\,|E_{\text{in}}(h) - E_{\text{out}}(h)| > \epsilon\,\right] \leq 2e^{-2\epsilon^2 N}$$

Since g has to be one of $h_1, h_2, \cdots, h_M$, we conclude that

If:

$$|E_{\text{in}}(g) - E_{\text{out}}(g)| > \epsilon$$

Then:

$$|E_{\text{in}}(h_1) - E_{\text{out}}(h_1)| > \epsilon \quad \textbf{or}$$

$$|E_{\text{in}}(h_2) - E_{\text{out}}(h_2)| > \epsilon \quad \textbf{or}$$

$$\cdots$$

$$|E_{\text{in}}(h_M) - E_{\text{out}}(h_M)| > \epsilon$$

This gives us an added M factor.

Review of Lecture 6

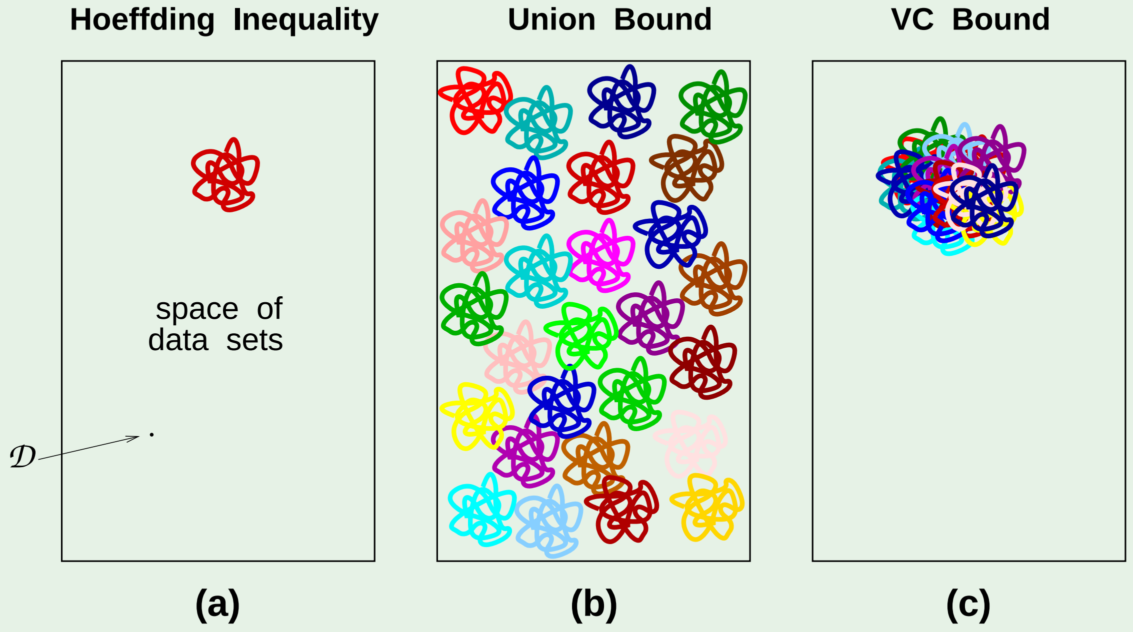
- $m_{\mathcal{H}}(N)$ is polynomial

if $\mathcal{H}$ has a break point k

$$
\begin{array}{cc|ccccccc}
 & & & & & k & & & \\
 & & 1 & 2 & 3 & 4 & 5 & 6 & .. \\
\hline
 & 1 & 1 & 2 & 2 & 2 & 2 & 2 & .. \\
 & 2 & 1 & & & & & & \\
 & 3 & 1 & & & & & & \\
N & 4 & 1 & & & & & & \\
 & 5 & 1 & & & & & & \\
 & 6 & 1 & & & & & & \\
 & \vdots & \vdots & & & & & &
\end{array}
$$

$$m_{\mathcal{H}}(N) \quad \leq \quad \underbrace{\sum_{i=0}^{k-1} \binom{N}{i}}_{\text{maximum power is } N^{k-1}}$$

- The VC Inequality



$$\mathbb{P}\left[\,|E_{\text{in}}(g) - E_{\text{out}}(g)| > \epsilon\,\right] \;\leq\; 2 \quad M \quad e^{-\,2\,\epsilon^2 N}$$

$$\downarrow \qquad \downarrow \qquad \downarrow$$

$$\downarrow \qquad \downarrow \qquad \downarrow$$

$$\mathbb{P}\left[\,|E_{\text{in}}(g) - E_{\text{out}}(g)| > \epsilon\,\right] \;\leq\; 4 \quad \boxed{m_{\mathcal{H}}(2N)} \quad e^{-\,\frac{1}{8}\,\epsilon^2 N}$$

Outline

- Input representation
- Linear Classification
- Linear Regression
- Nonlinear Transformation

A real data set

Input representation

'raw' input $\mathbf{x} = (x_0, x_1, x_2, \cdots, x_{256})$

linear model: $(w_0, w_1, w_2, \cdots, w_{256})$

Features: Extract useful information, e.g.,

intensity and symmetry $\mathbf{x} = (x_0, x_1, x_2)$

linear model: (w_0, w_1, w_2)

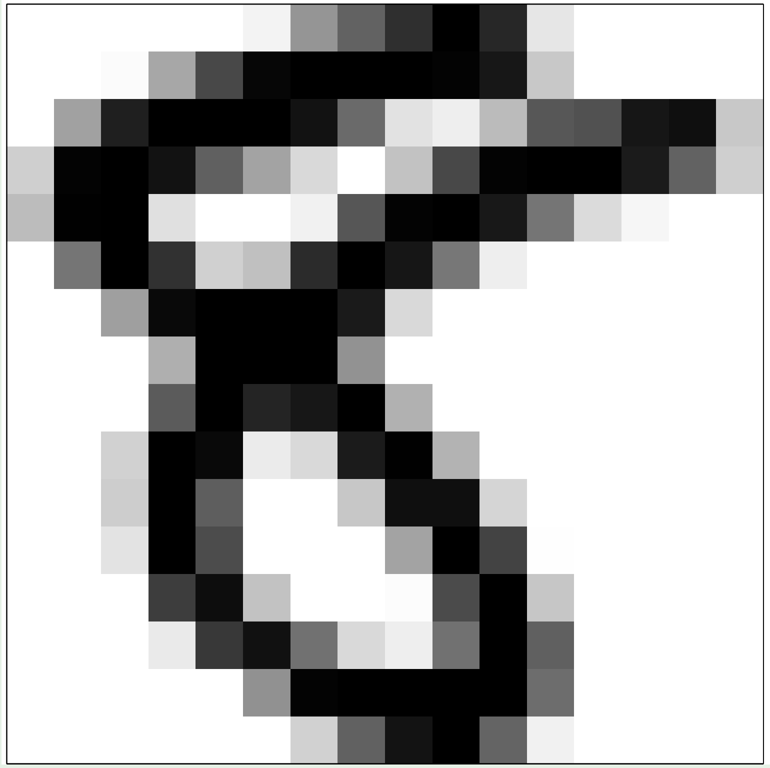
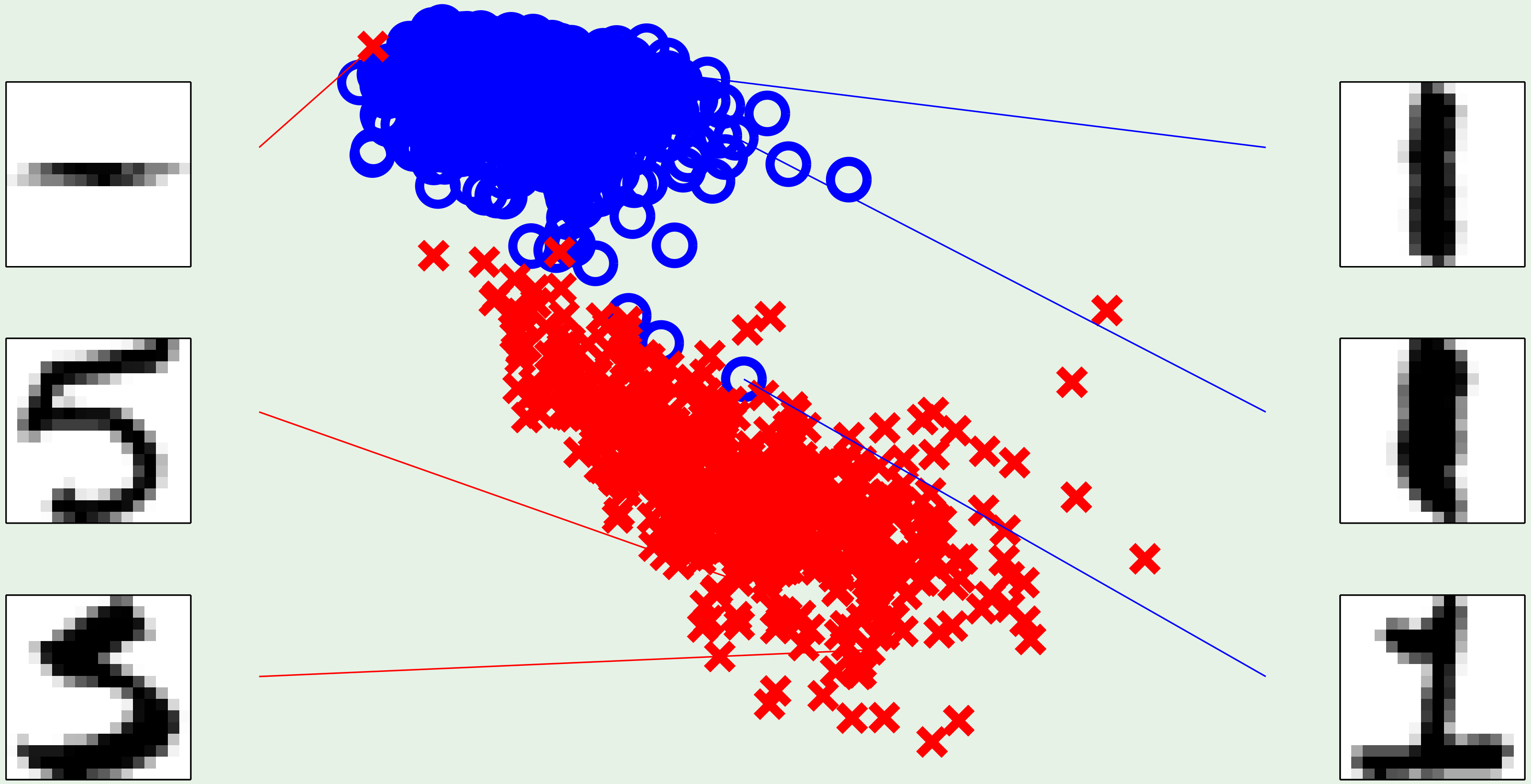


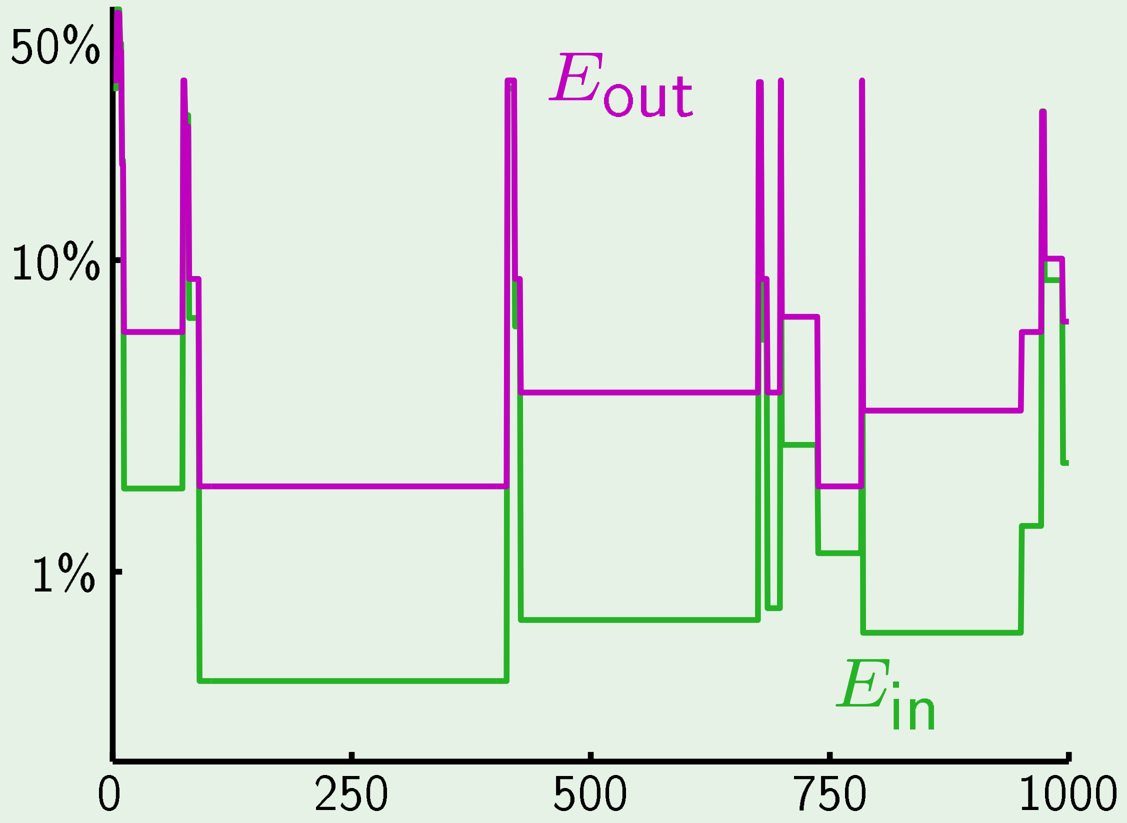
Illustration of features

$\mathbf{x} = (x_0, x_1, x_2)$ $\quad x_1$: intensity $\quad x_2$: symmetry

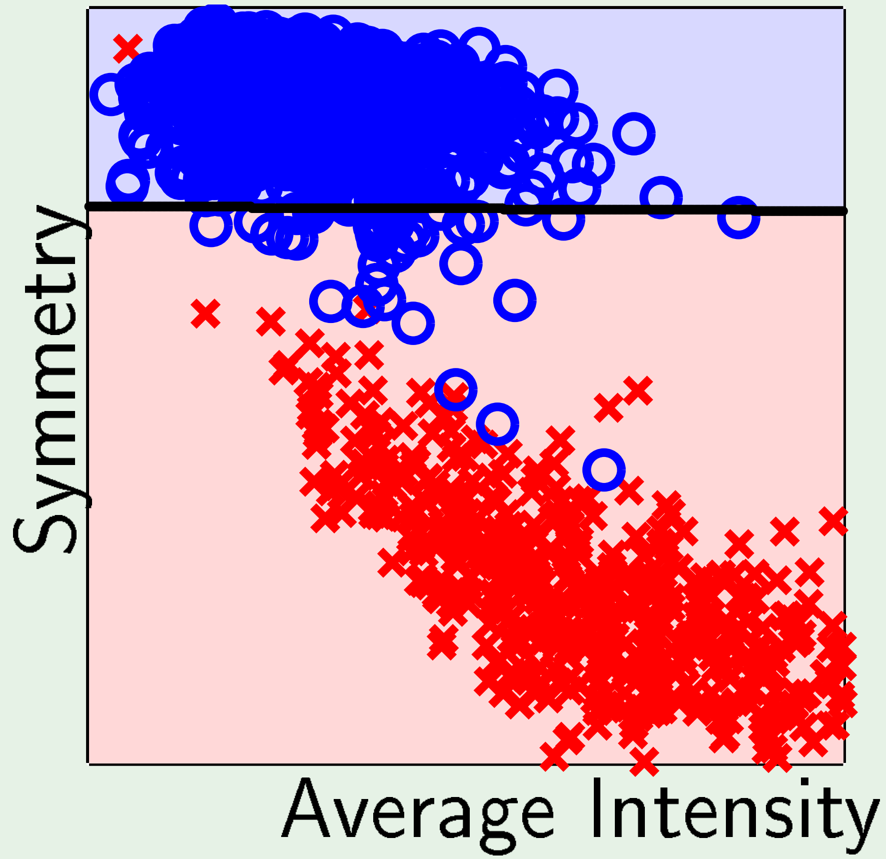


What PLA does

Evolution of E_{in} and E_{out}

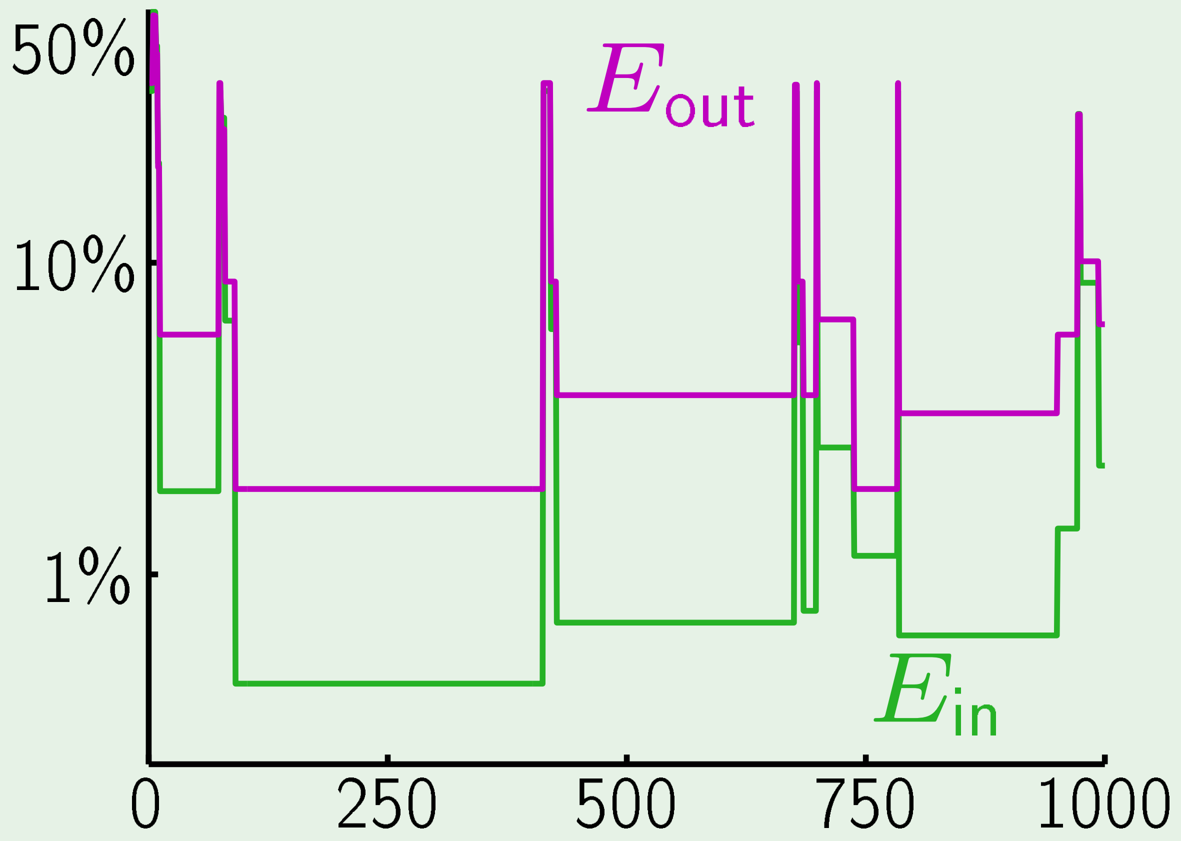


Final perceptron boundary

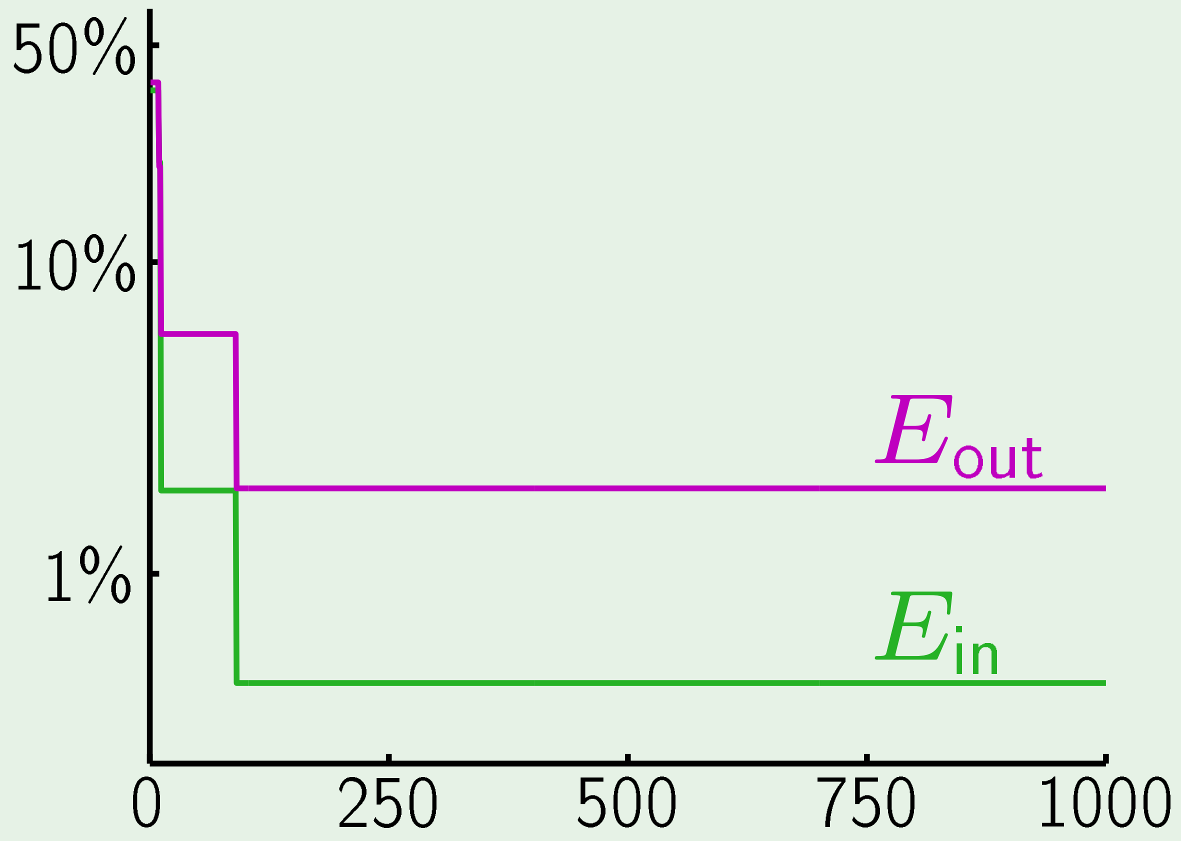


The ‘pocket’ algorithm

PLA:

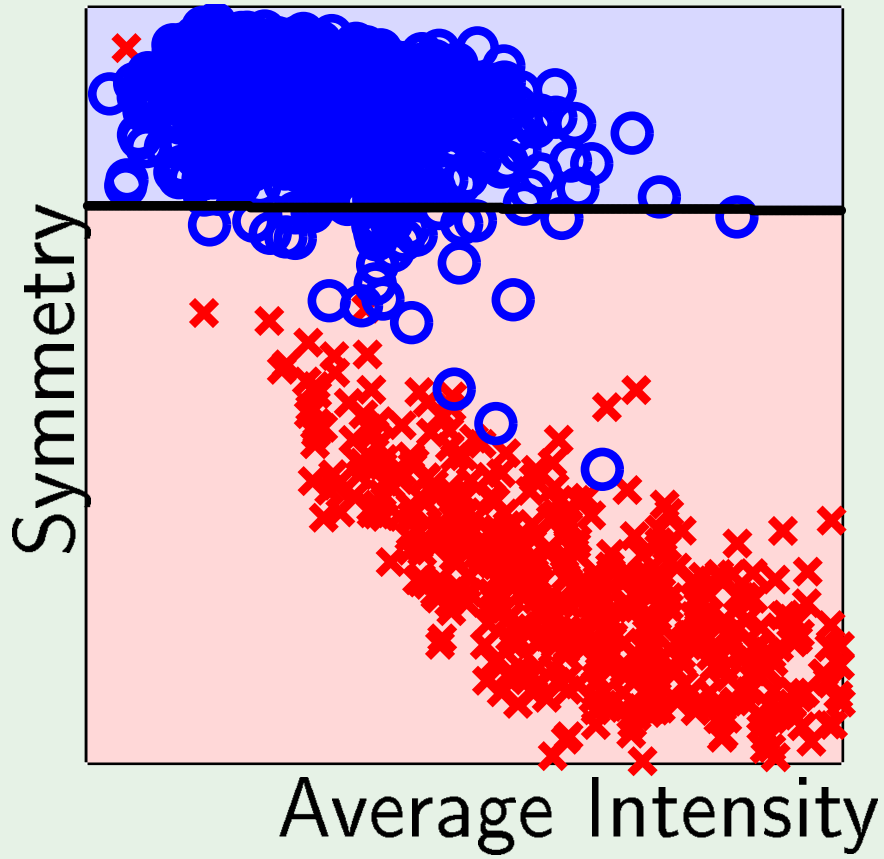


Pocket:

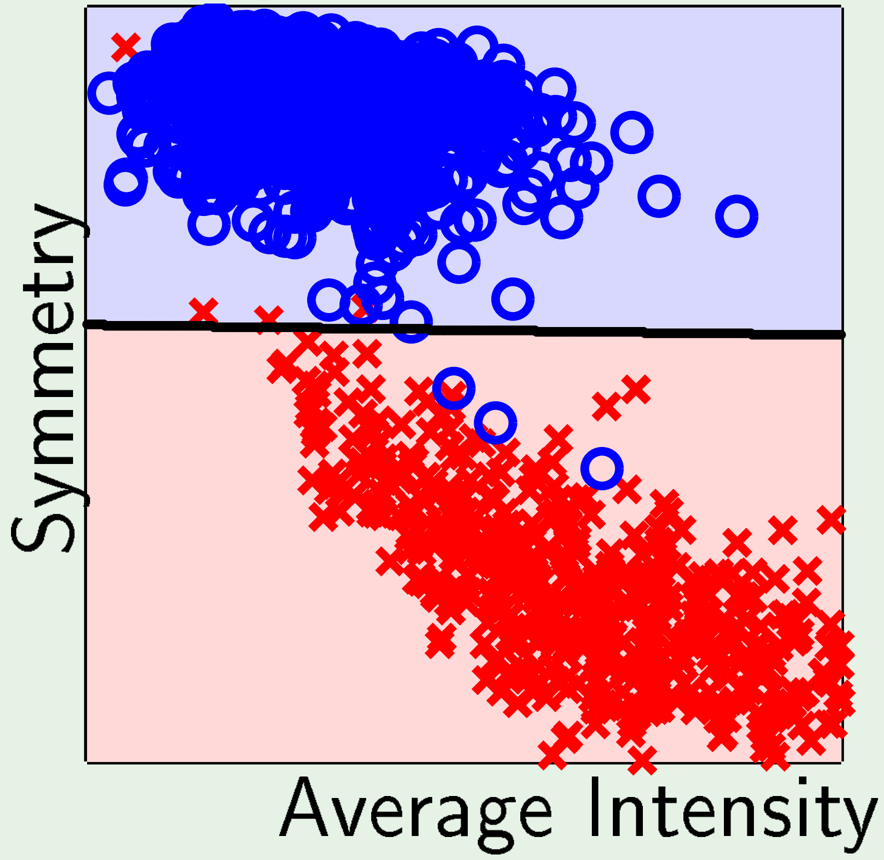


Classification boundary - PLA versus Pocket

PLA:



Pocket:



Outline

- Input representation

- Linear Classification

- Linear Regression **regression $\equiv$ real-valued output**

- Nonlinear Transformation

Credit again

Classification: Credit approval (yes/no)

Regression: Credit line (dollar amount)

Input: $\mathbf{x} =$

age	23 years
annual salary	\$30,000
years in residence	1 year
years in job	1 year
current debt	\$15,000
$\cdots$	$\cdots$

Linear regression output: $h(\mathbf{x}) = \sum_{i=0}^{d} w_i \, x_i = \mathbf{w}^{\intercal}\mathbf{x}$

The data set

Credit officers decide on credit lines:

$$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_N, y_N)$$

$y_n \in \mathbb{R}$ is the credit line for customer $\mathbf{x}_n$.

Linear regression tries to replicate that.

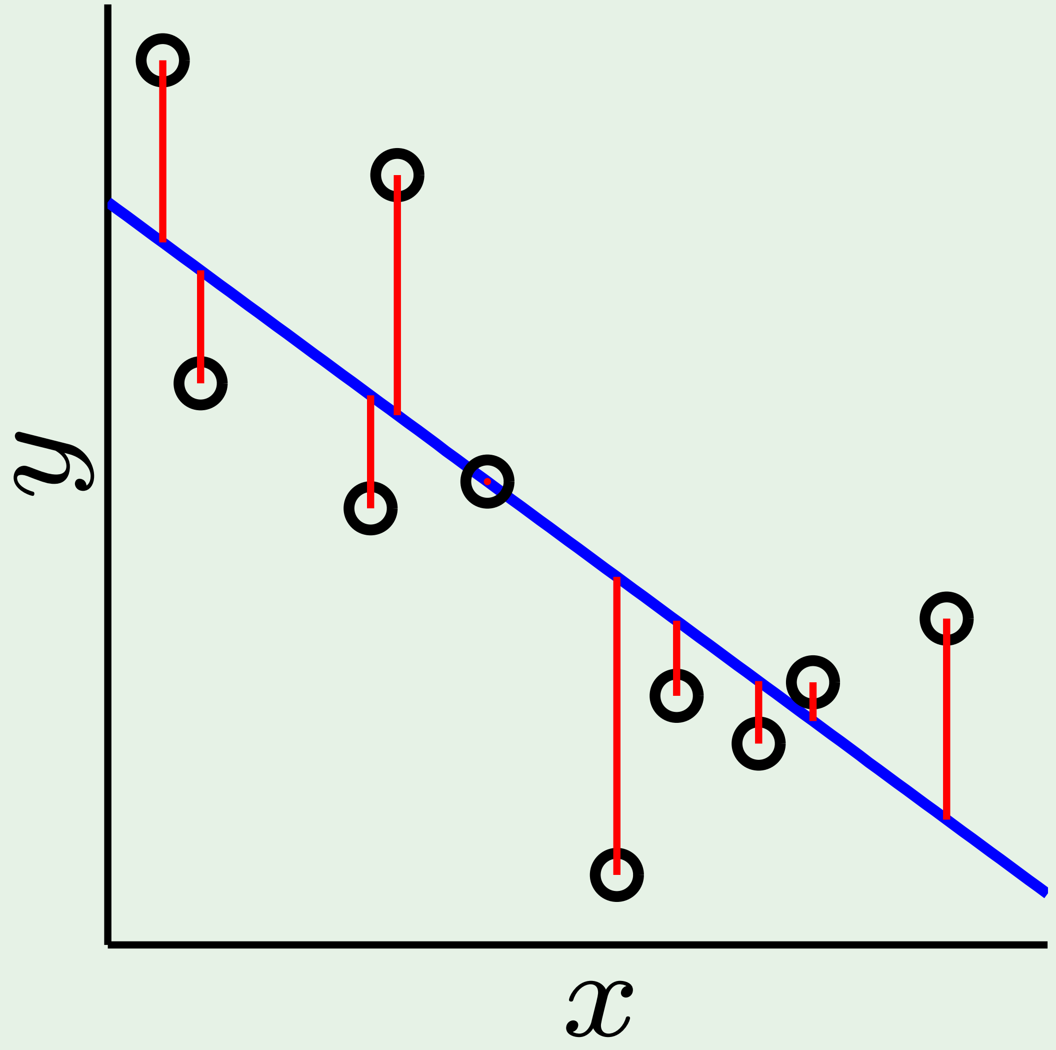
How to measure the error

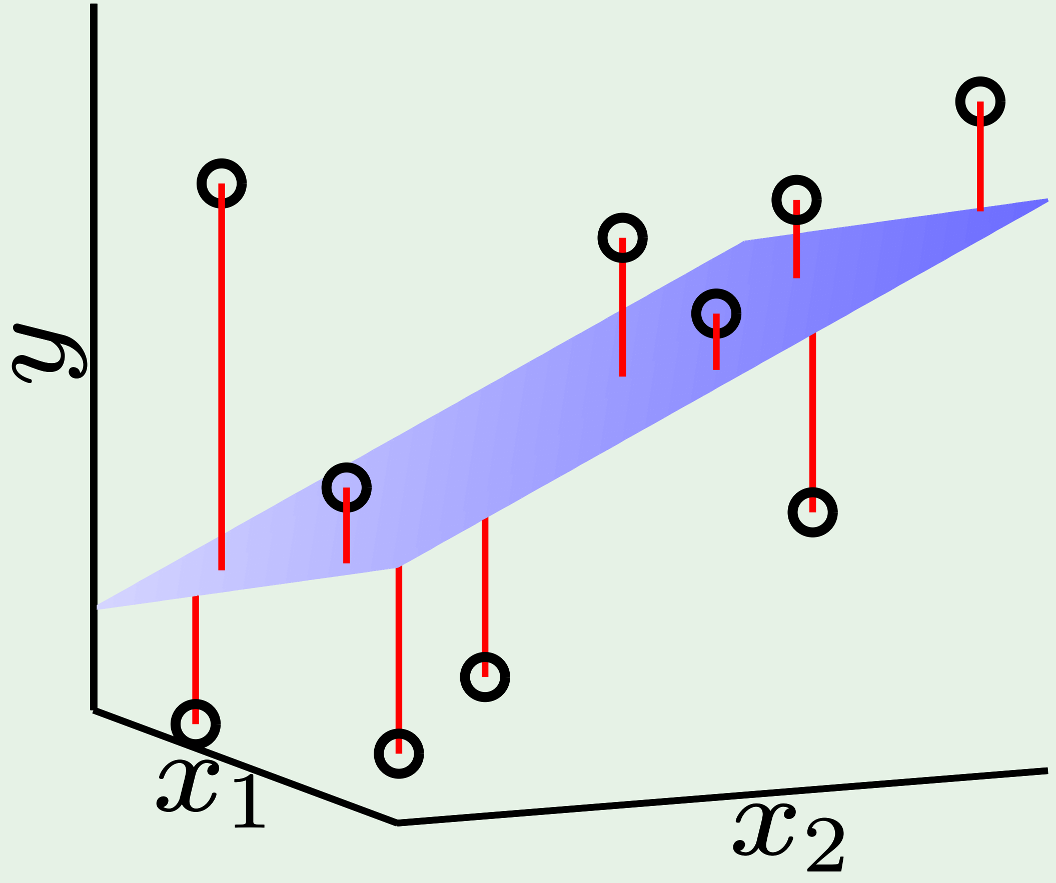
How well does $h(\mathbf{x}) = \mathbf{w}^{\intercal}\mathbf{x}$ approximate $f(\mathbf{x})$?

In linear regression, we use squared error $(h(\mathbf{x}) - f(\mathbf{x}))^2$

in-sample error: $E_{\text{in}}(h) = \dfrac{1}{N}\displaystyle\sum_{n=1}^{N}(h(\mathbf{x}_n) - y_n)^2$

Illustration of linear regression





The expression for E_{in}

$$
\begin{aligned}
E_{\text{in}}(\mathbf{w}) &= \frac{1}{N}\sum_{n=1}^{N}\left(\mathbf{w}^{\intercal}\mathbf{x}_n - y_n\right)^2 \\
&= \frac{1}{N}\|\mathrm{X}\mathbf{w} - \mathbf{y}\|^2
\end{aligned}
$$

where $\mathrm{X} = \begin{bmatrix} \text{—}\mathbf{x}_1{}^{\intercal}\text{—} \\ \text{—}\mathbf{x}_2{}^{\intercal}\text{—} \\ \vdots \\ \text{—}\mathbf{x}_N{}^{\intercal}\text{—} \end{bmatrix}, \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$

Minimizing E_{in}

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\|\mathrm{X}\mathbf{w} - \mathbf{y}\|^2$$

$$\nabla E_{\text{in}}(\mathbf{w}) = \frac{2}{N}\mathrm{X}^{\intercal}(\mathrm{X}\mathbf{w} - \mathbf{y}) = \mathbf{0}$$

$$\mathrm{X}^{\intercal}\mathrm{X}\mathbf{w} = \mathrm{X}^{\intercal}\mathbf{y}$$

$$\mathbf{w} = \mathrm{X}^{\dagger}\mathbf{y} \quad \text{where} \quad \mathrm{X}^{\dagger} = (\mathrm{X}^{\intercal}\mathrm{X})^{-1}\mathrm{X}^{\intercal}$$

$\mathrm{X}^{\dagger}$ is the '**pseudo-inverse**' of X

The pseudo-inverse

$$X^{\dagger} = (X^{\top}X)^{-1}X^{\top}$$

$$\underbrace{\left(\quad \underbrace{\left[\qquad \right]}_{d+1 \ \times \ d+1} \quad \right)^{-1} \underbrace{\left[\qquad\qquad \right]}_{d+1 \ \times \ N}}_{d+1 \ \times \ N}$$

The linear regression algorithm

1: Construct the matrix X and the vector $\mathbf{y}$ from the data set $(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_N, y_N)$ as follows

$$\underbrace{\mathrm{X} = \begin{bmatrix} —\mathbf{x}_1^\intercal— \\ —\mathbf{x}_2^\intercal— \\ \vdots \\ —\mathbf{x}_N^\intercal— \end{bmatrix}}_{\text{input data matrix}}, \qquad \underbrace{\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}}_{\text{target vector}}.$$

2: Compute the pseudo-inverse $\mathrm{X}^\dagger = (\mathrm{X}^\intercal \mathrm{X})^{-1} \mathrm{X}^\intercal$.

3: Return $\mathbf{w} = \mathrm{X}^\dagger \mathbf{y}$.

Linear regression for classification

Linear regression learns a real-valued function $y = f(\mathbf{x}) \in \mathbb{R}$

Binary-valued functions are also real-valued! $\pm 1 \in \mathbb{R}$

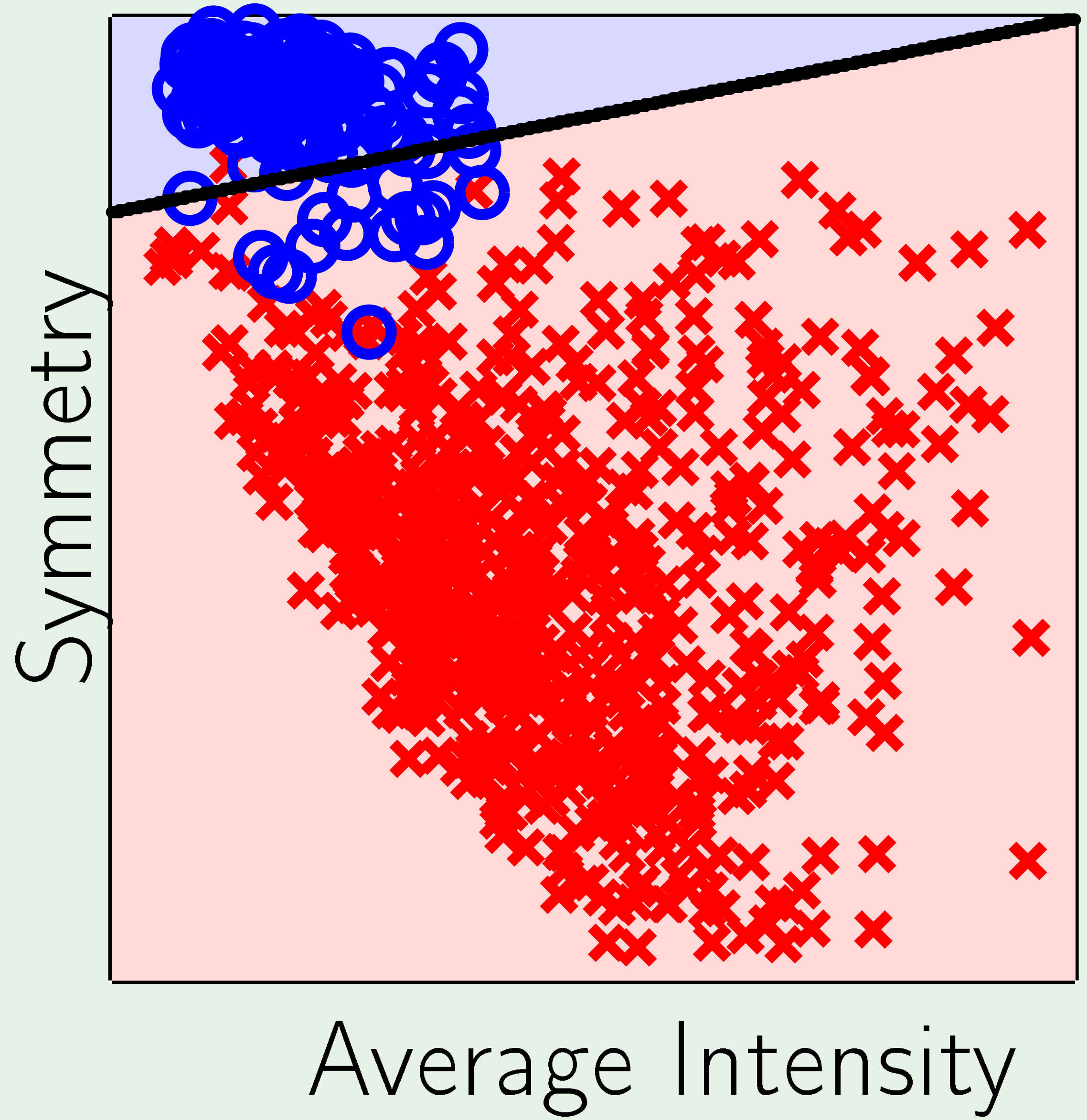
Use linear regression to get $\mathbf{w}$ where $\mathbf{w}^{\intercal}\mathbf{x}_n \approx y_n = \pm 1$

In this case, $\text{sign}(\mathbf{w}^{\intercal}\mathbf{x}_n)$ is likely to agree with $y_n = \pm 1$

Good initial weights for classification

© AML Creator: Yaser Abu-Mostafa - LFD Lecture 3

Linear regression boundary

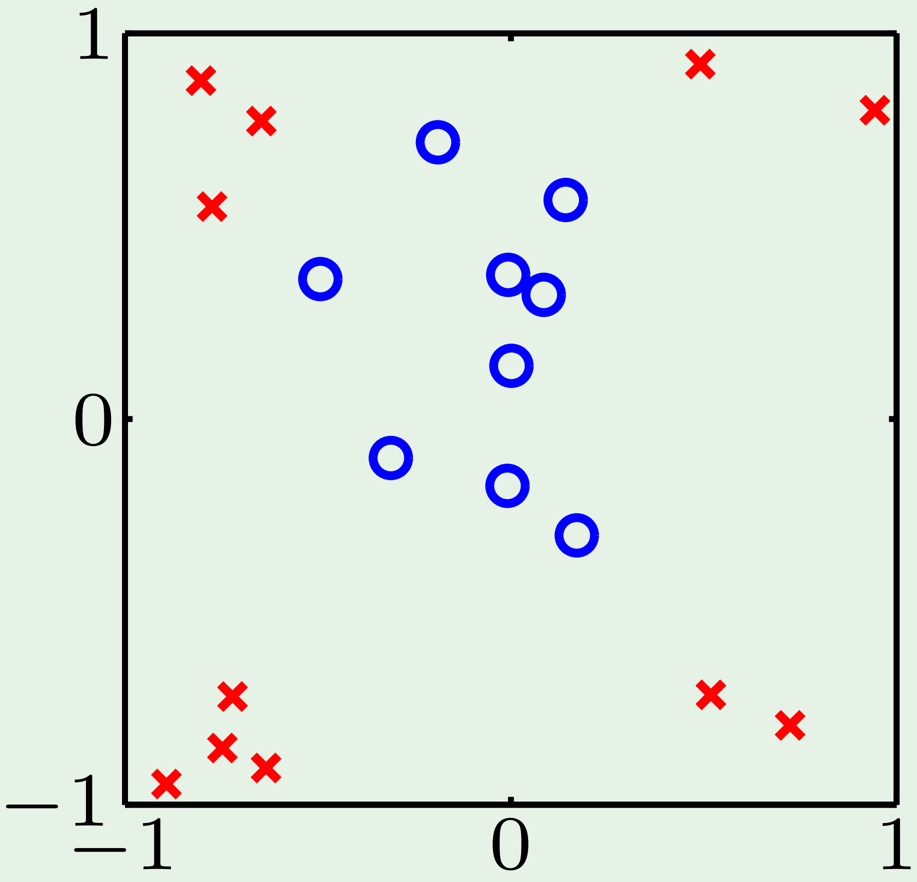


Outline

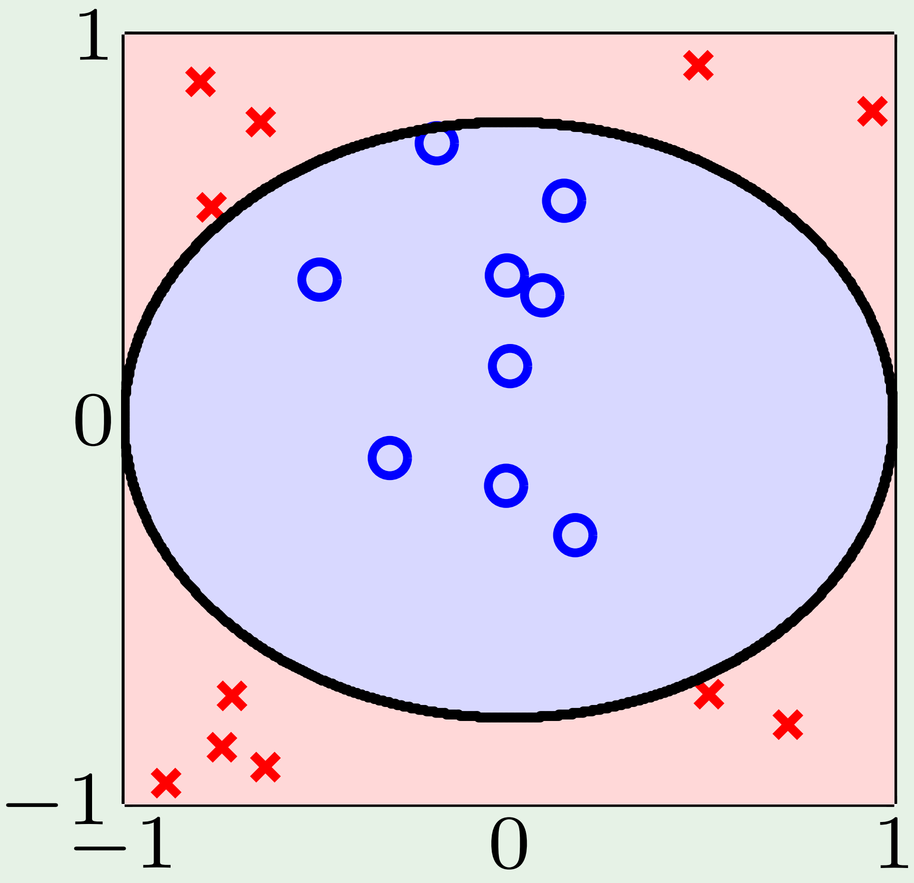
- Input representation
- Linear Classification
- Linear Regression
- Nonlinear Transformation

Linear is limited

Data:



Hypothesis:



Another example

Credit line is affected by 'years in residence'

but **not** in a linear way!

Nonlinear $[[x_i < 1]]$ and $[[x_i > 5]]$ are better.

Can we do that with linear models?

Linear in what?

Linear regression implements

$$\sum_{i=0}^{d} w_i \; x_i$$

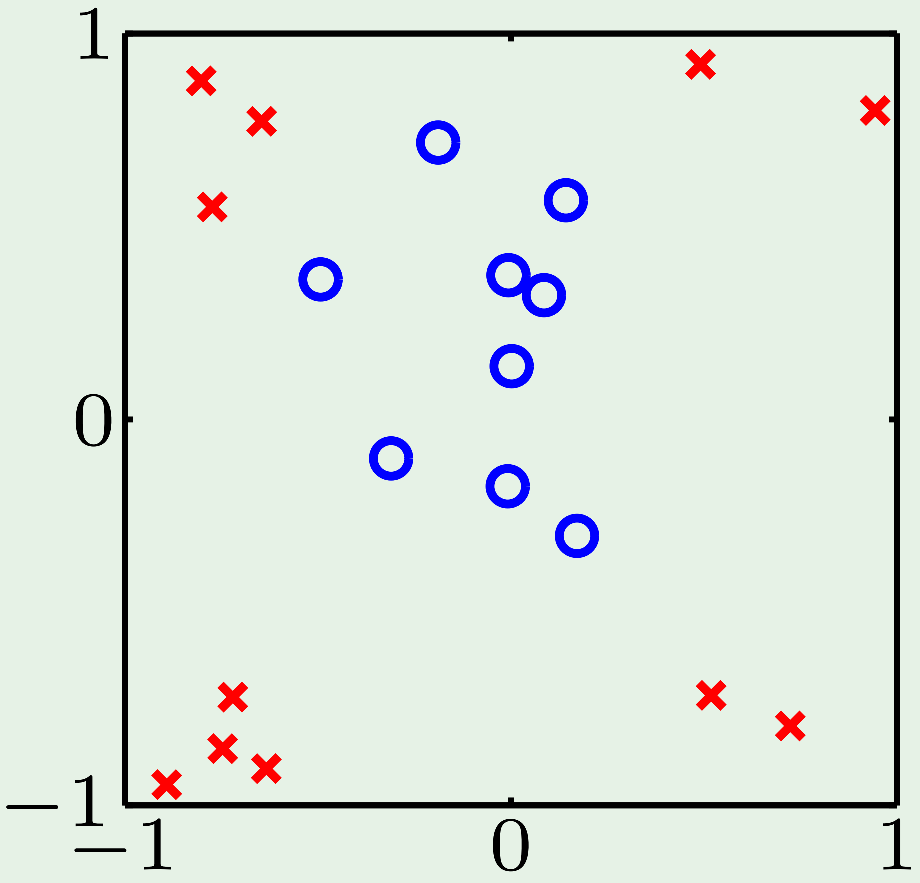
Linear classification implements

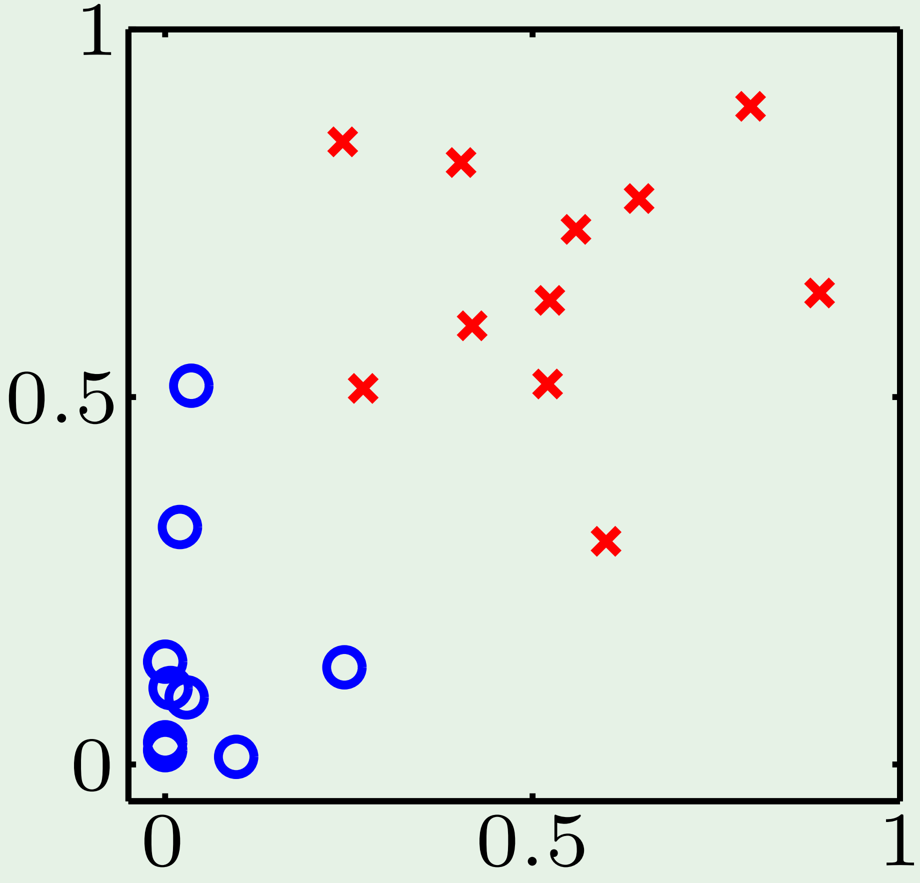
$$\text{sign}\left(\sum_{i=0}^{d} w_i \; x_i\right)$$

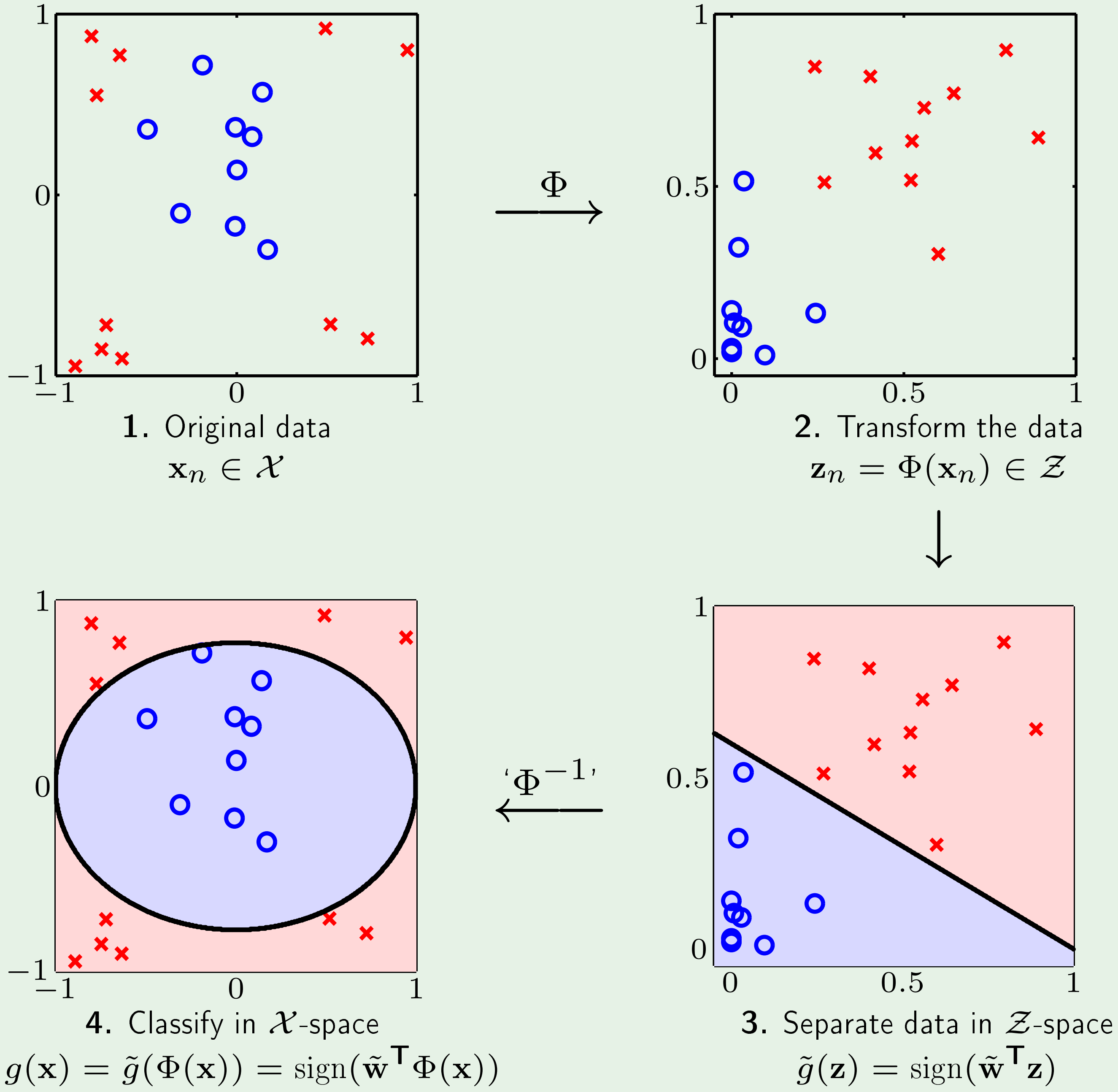
Algorithms work because of **linearity in the weights**

Transform the data nonlinearly

$$(x_1, x_2) \xrightarrow{\Phi} (x_1^2, x_2^2)$$







1. Original data
$\mathbf{x}_n \in \mathcal{X}$
Φ
2. Transform the data
$\mathbf{z}_n = \Phi(\mathbf{x}_n) \in \mathcal{Z}$
3. Separate data in $\mathcal{Z}$-space
$\tilde{g}(\mathbf{z}) = \text{sign}(\tilde{\mathbf{w}}^{\mathsf{T}}\mathbf{z})$
'Φ^{-1}'
4. Classify in $\mathcal{X}$-space
$g(\mathbf{x}) = \tilde{g}(\Phi(\mathbf{x})) = \text{sign}(\tilde{\mathbf{w}}^{\mathsf{T}}\Phi(\mathbf{x}))$

What transforms to what

$$\mathbf{x} = (x_0, x_1, \cdots, x_d) \quad \xrightarrow{\Phi} \quad \mathbf{z} = (z_0, z_1, \cdots\cdots\cdots, z_{\tilde{d}})$$

$$\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N \quad \xrightarrow{\Phi} \quad \mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_N$$

$$y_1, y_2, \cdots, y_N \quad \xrightarrow{\Phi} \quad y_1, y_2, \cdots, y_N$$

No weights in $\mathcal{X}$ $\qquad$ $\tilde{\mathbf{w}} = (w_0, w_1, \cdots\cdots\cdots, w_{\tilde{d}})$

$$g(\mathbf{x}) \quad = \quad \operatorname{sign}(\tilde{\mathbf{w}}^{\mathsf{T}}\Phi(\mathbf{x}))$$