Learning From Data

Yaser S. Abu-Mostafa
California Institute of Technology

Lecture 3: Linear Models I

Sponsored by Caltech's Provost Office, E&AS Division, and IST • Tuesday, April 10, 2012

Outline

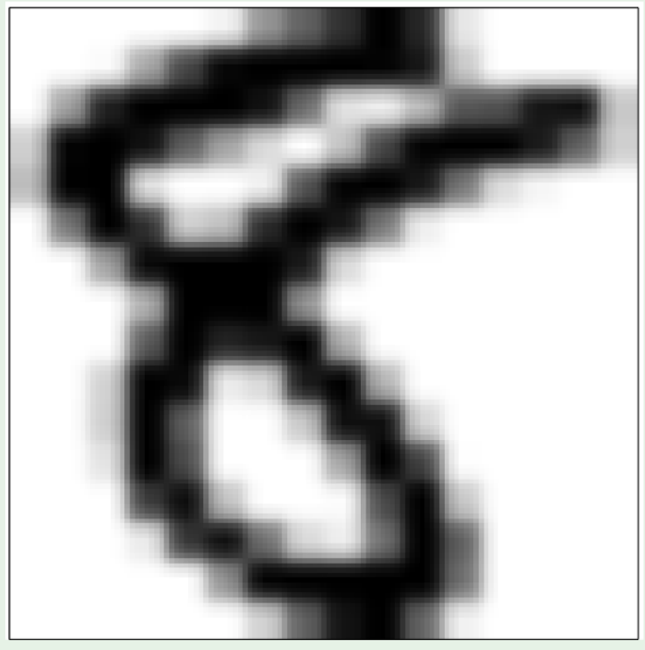
- Input representation
- Linear Classification
- Linear Regression
- Nonlinear Transformation

A real data set

Input representation

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

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



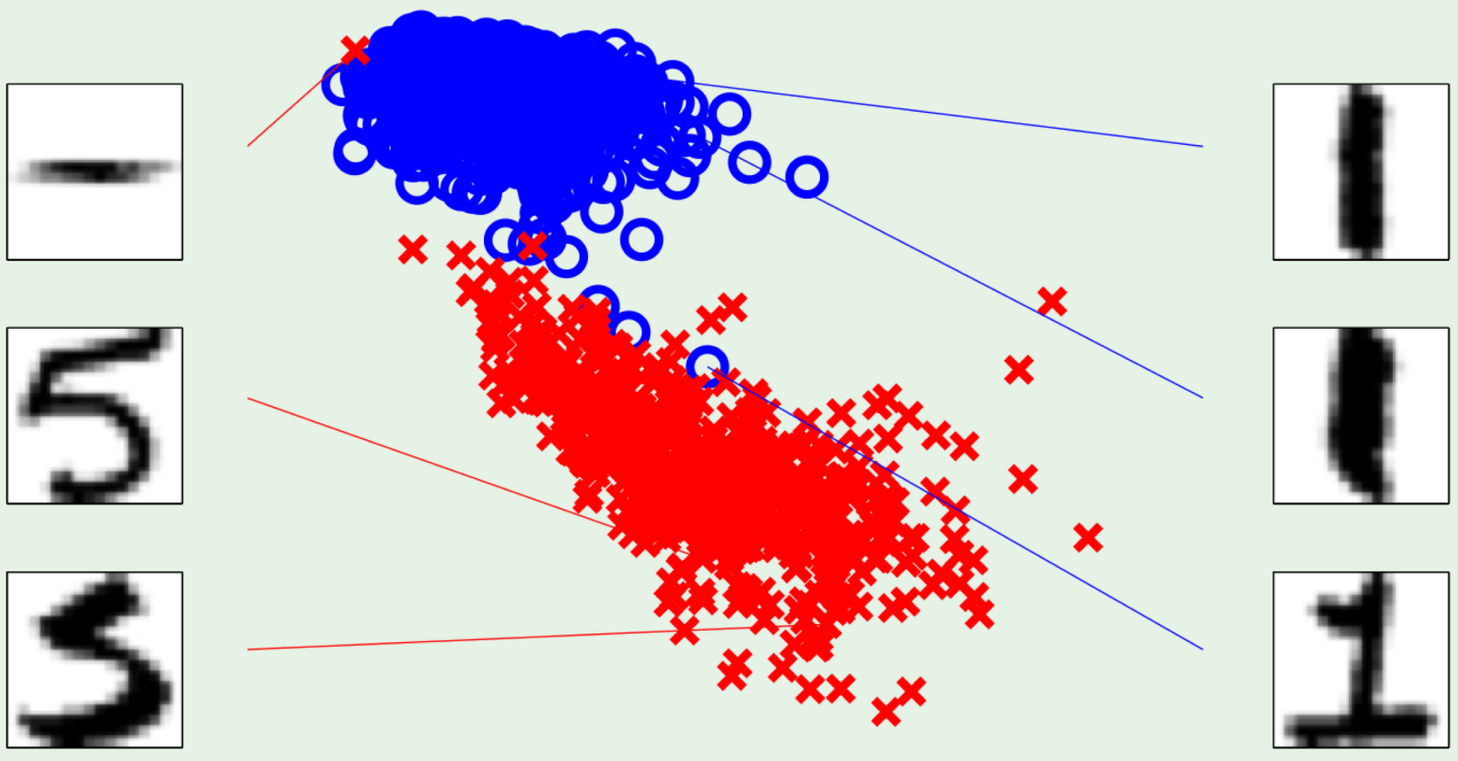
Features: Extract useful information, e.g.,

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

linear model: (w_0, w_1, w_2)

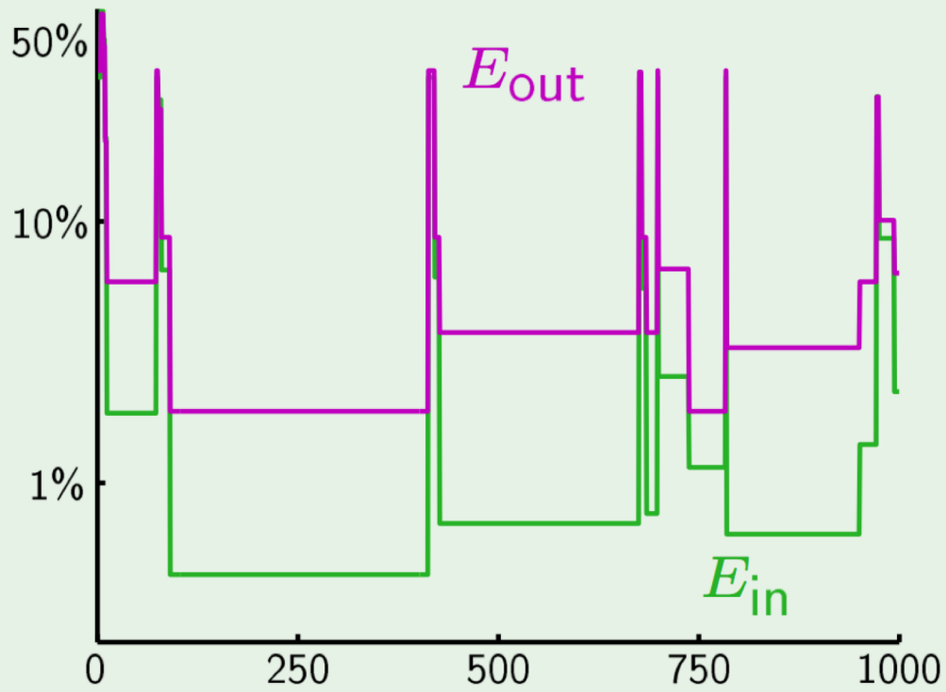
Illustration of features

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

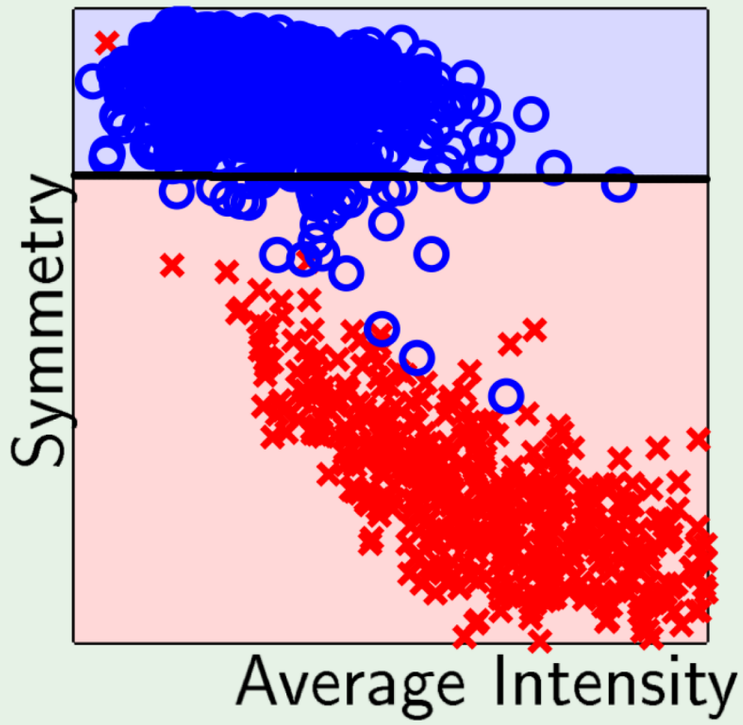


What PLA does

Evolution of E_{in} and E_{out}

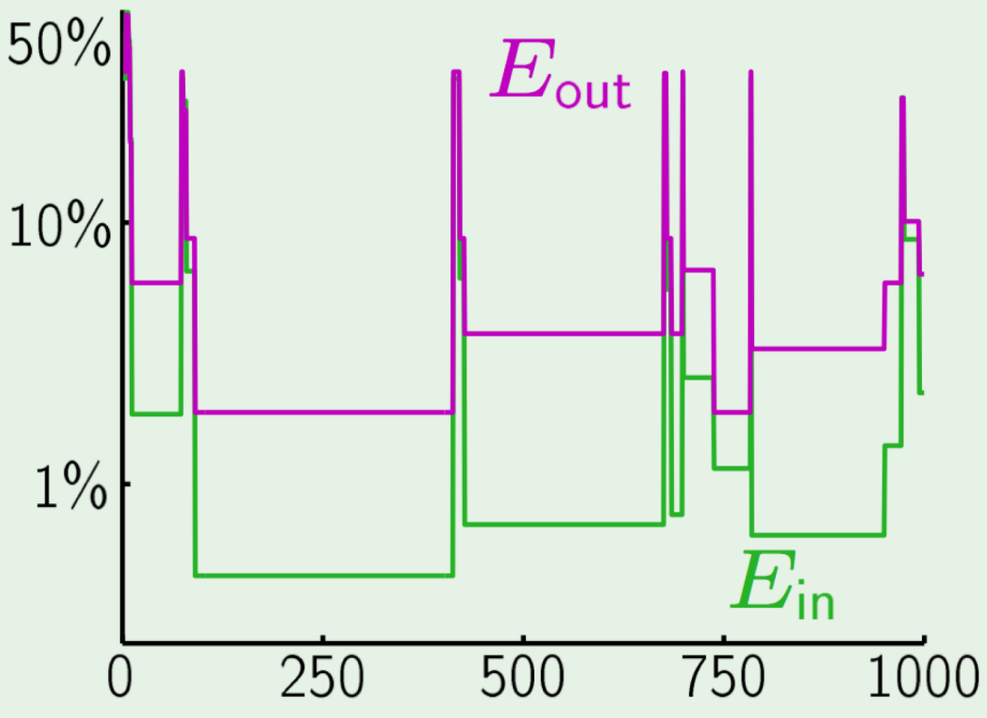


Final perceptron boundary

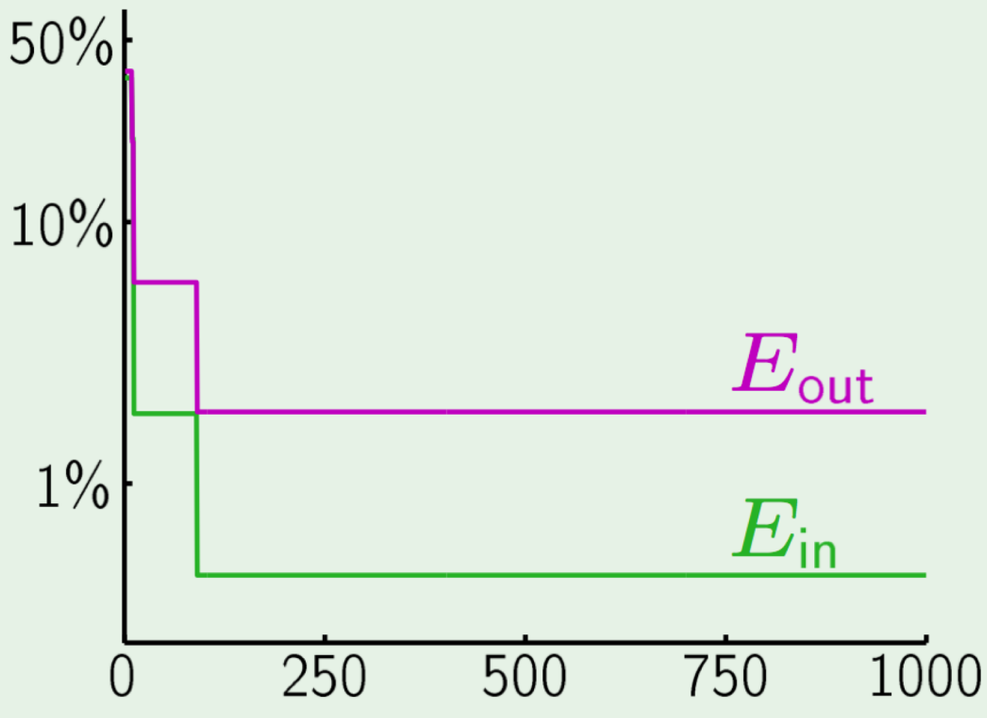


The 'pocket' algorithm

PLA:

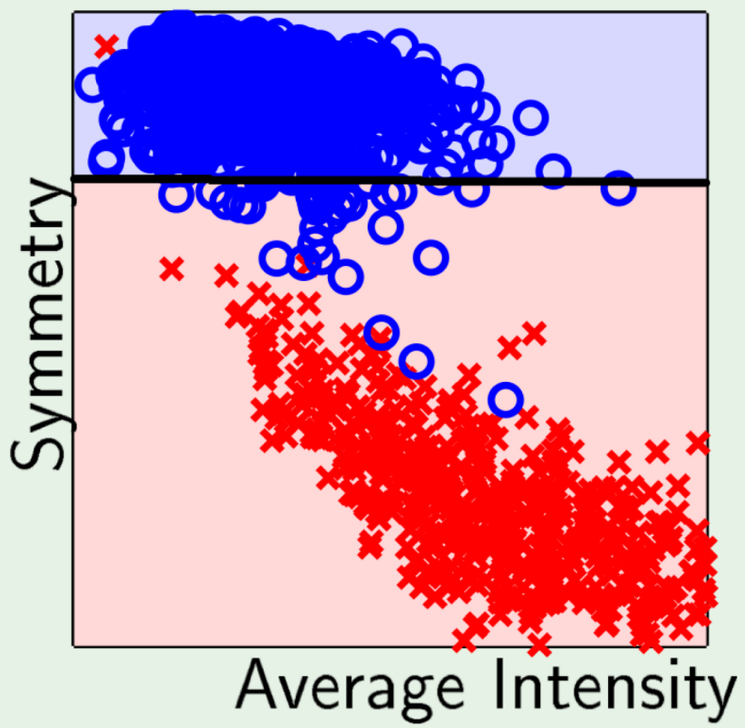


Pocket:

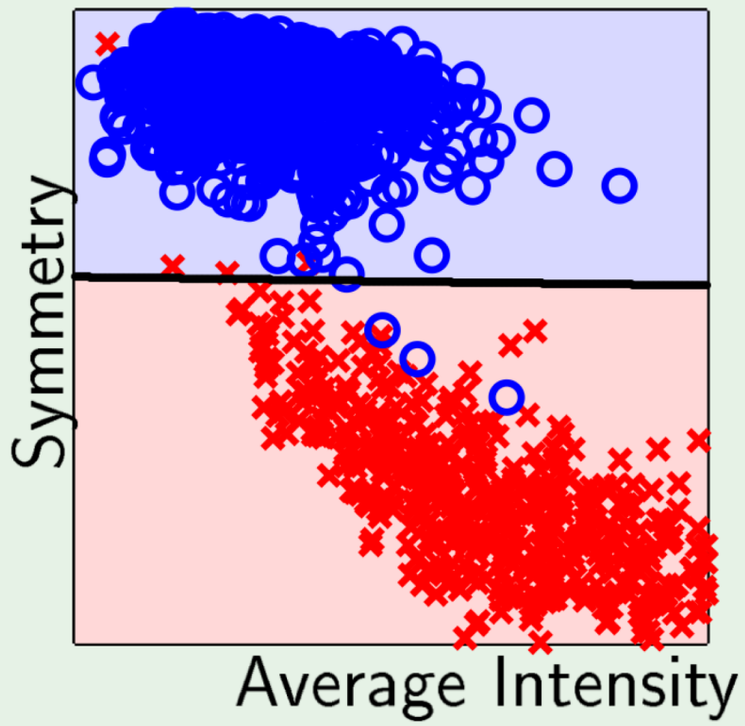


Classification boundary - PLA versus Pocket

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
...	...

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

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.

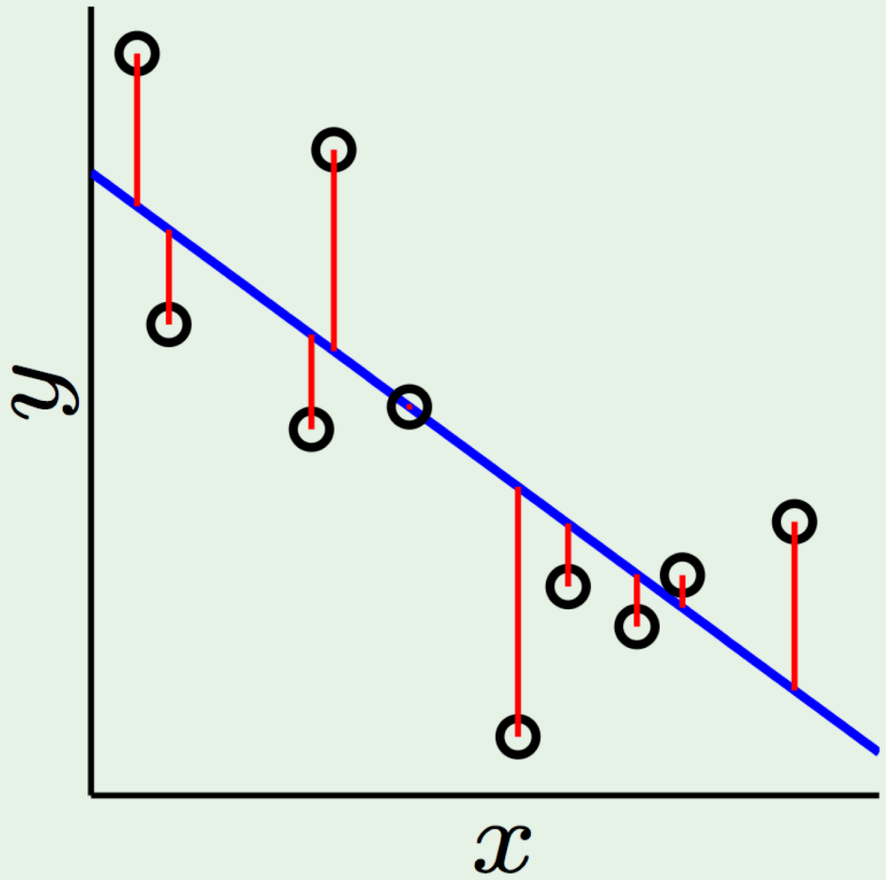
How to measure the error

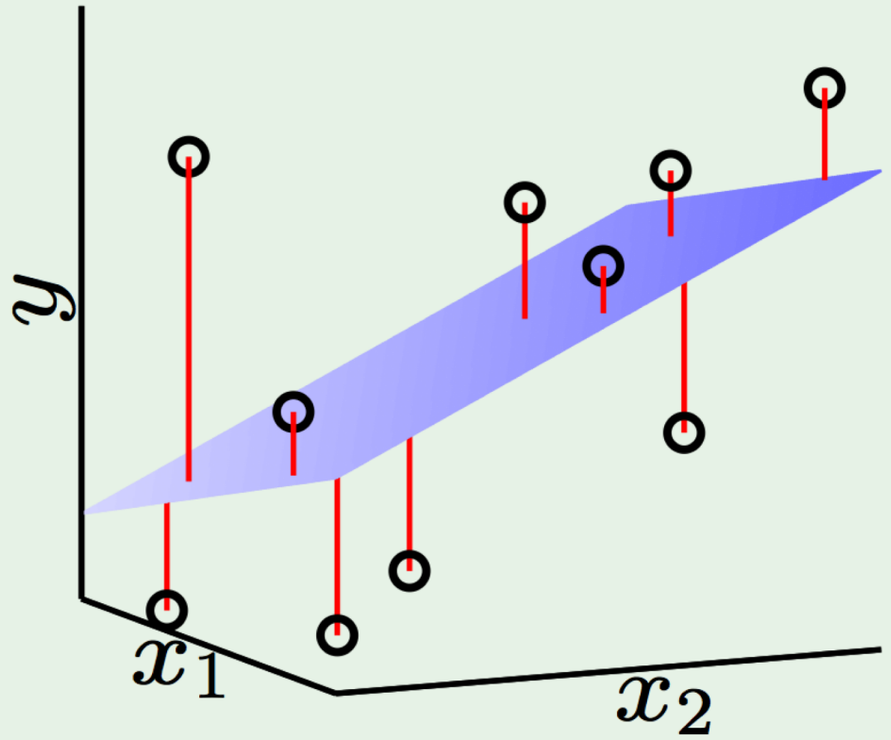
How well does $h(\mathbf{x}) = \mathbf{w}^{\mathsf{T}}\mathbf{x}$ approximate $f(\mathbf{x})$?

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

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

Illustration of linear regression





The expression for E_{in}

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

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

Minimizing E_{in}

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

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

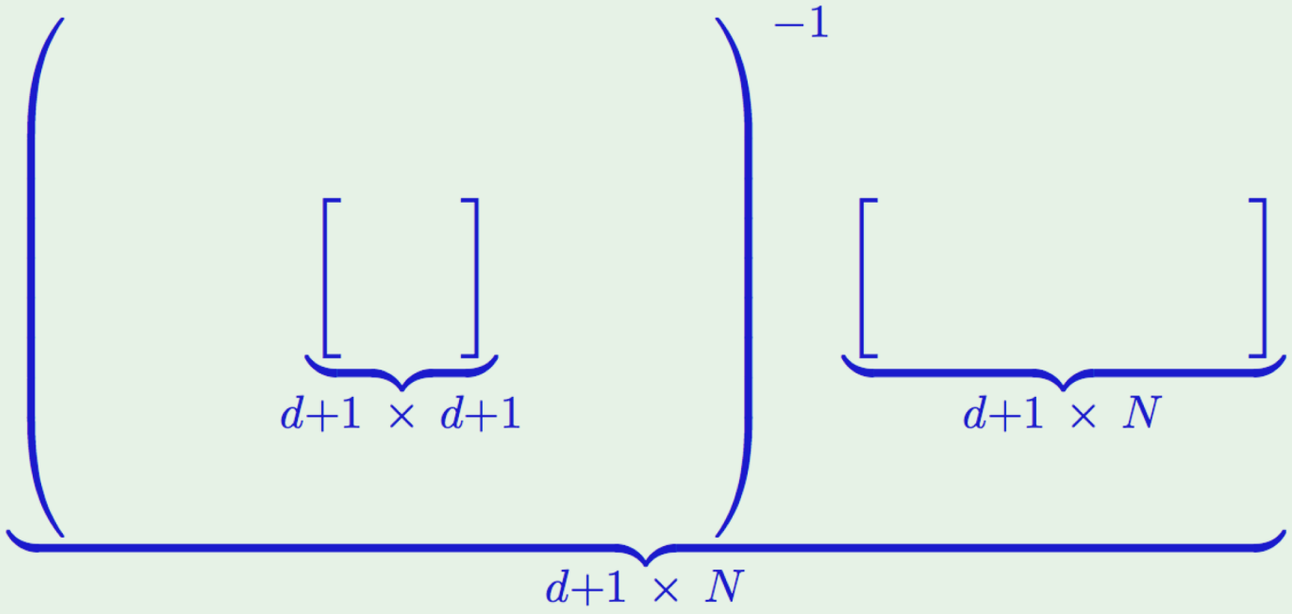
$$\mathrm{X}^\mathsf{T}\mathrm{X}\mathbf{w} = \mathrm{X}^\mathsf{T}\mathbf{y}$$

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

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

The pseudo-inverse

$$\mathrm{X}^\dagger = (\mathrm{X}^\mathsf{T}\mathrm{X})^{-1}\mathrm{X}^\mathsf{T}$$



The linear regression algorithm

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

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

2: Compute the pseudo-inverse $\mathbf{X}^\dagger = (\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}$.

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

Linear regression for classification

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

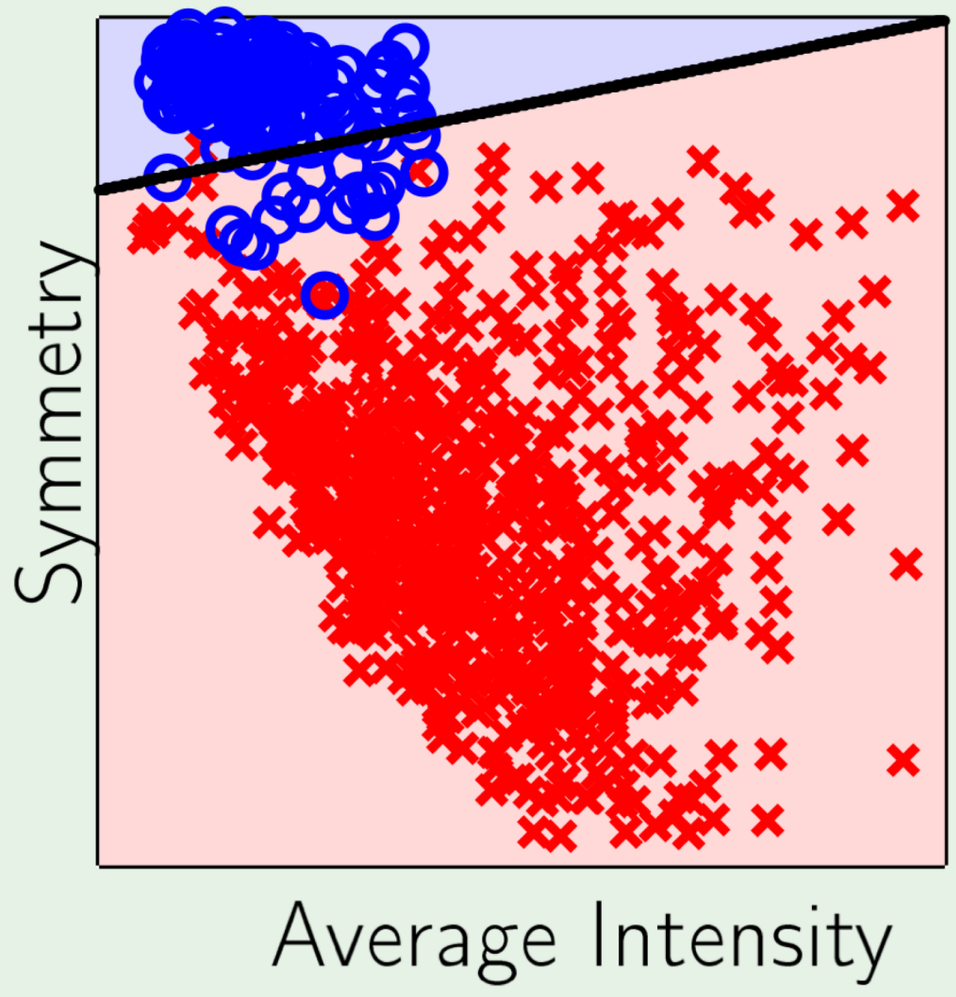
Binary-valued functions are also real-valued! $\pm 1 \in \mathbb{R}$

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

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

Good initial weights for classification

Linear regression boundary

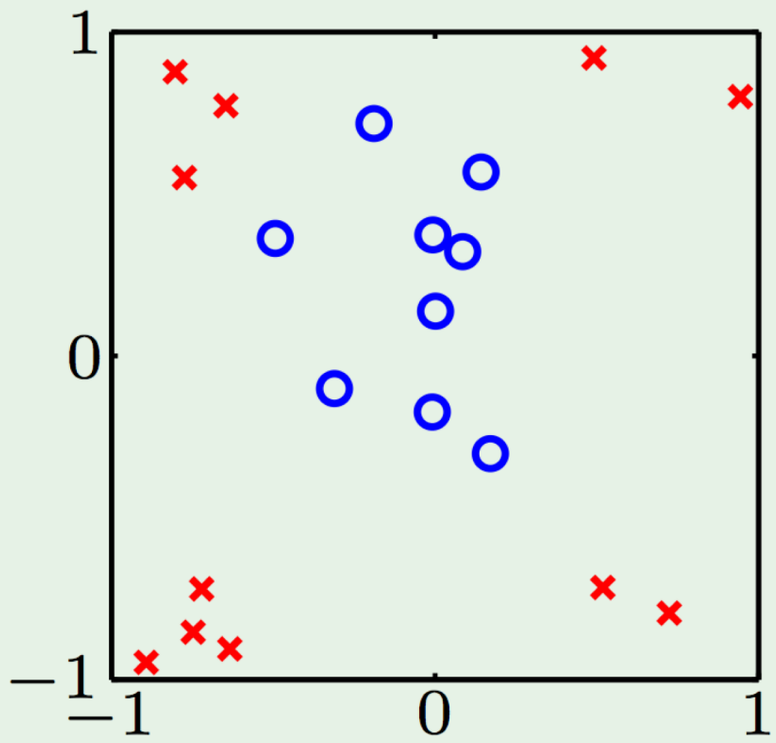


Outline

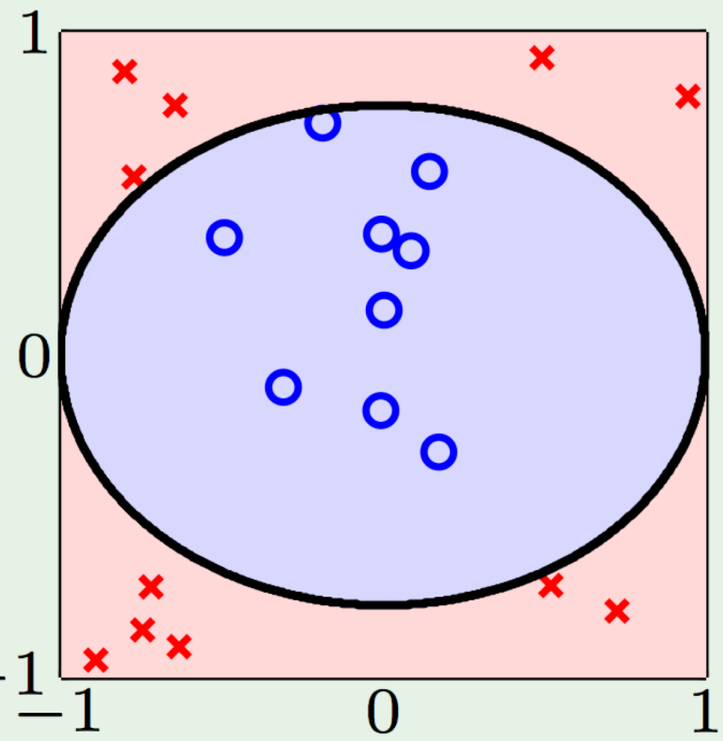
- Input representation
- Linear Classification
- Linear Regression
- Nonlinear Transformation

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

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

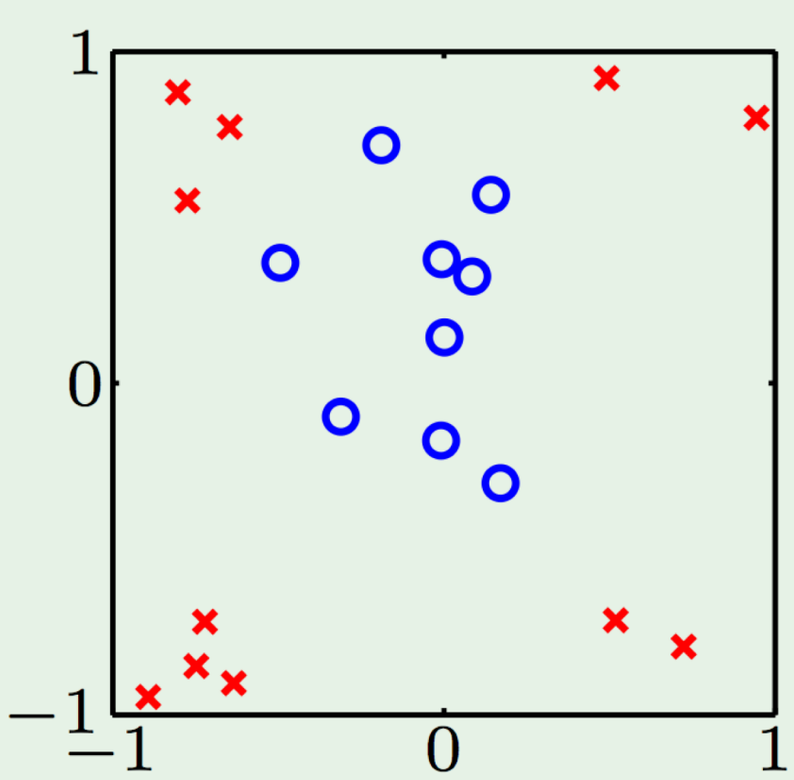
Linear classification implements

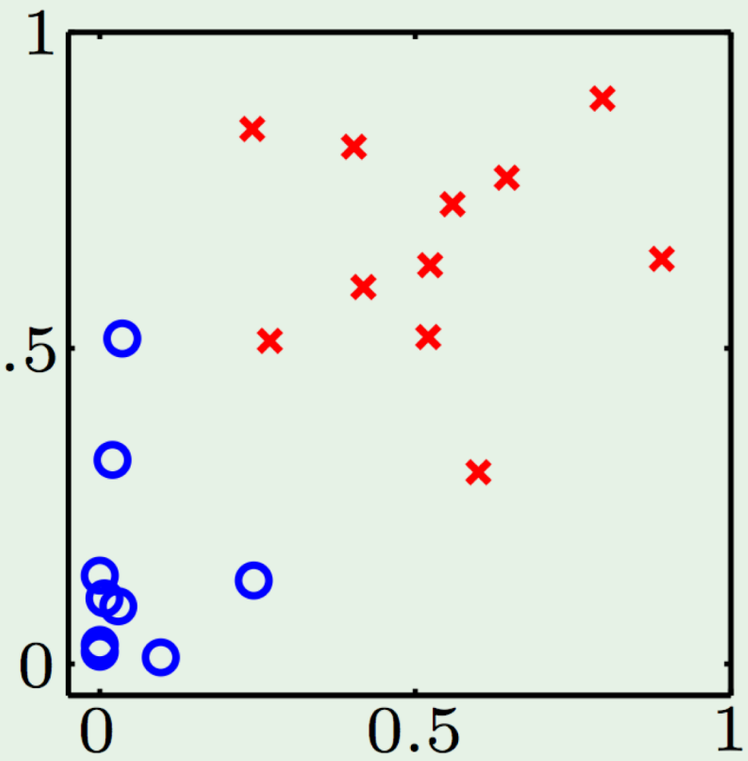
$$\text{sign}\left(\sum_{i=0}^{d} w_i \, x_i\right)$$

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

Transform the data nonlinearly

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



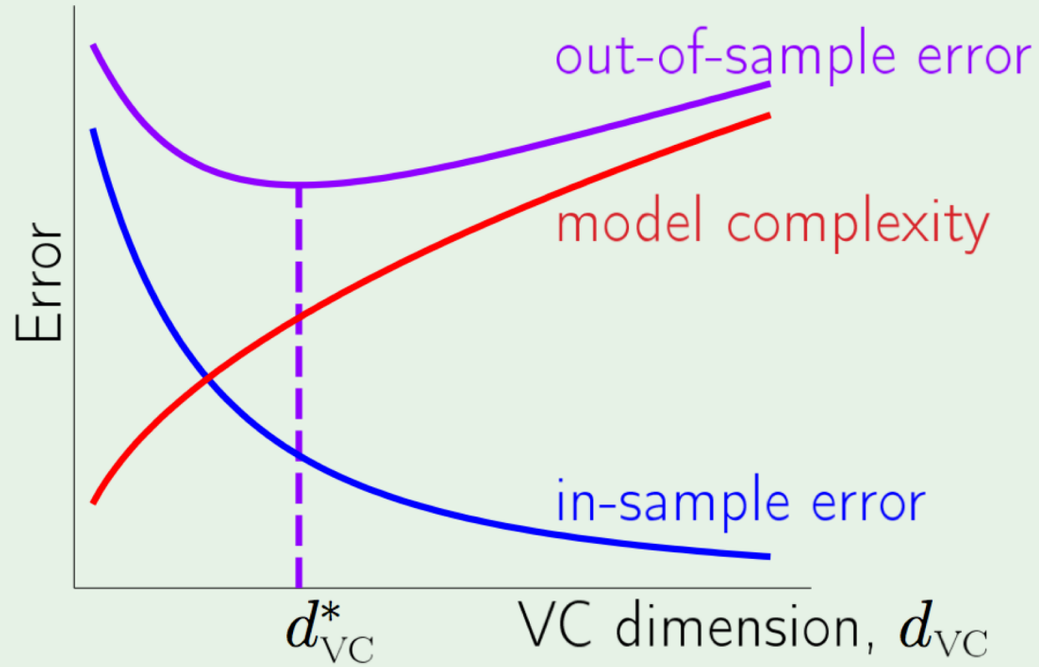


What the theory will achieve

Characterizing the feasibility of learning for infinite M

Characterizing the tradeoff:

Model complexity	$\uparrow$	E_{in}	$\downarrow$
Model complexity	$\uparrow$	$E_{\text{out}} - E_{\text{in}}$	$\uparrow$



Where we are

- Linear classification ✓
- Linear regression ✓
- Logistic regression
- Nonlinear transforms