

CSE4088 Introduction to Machine Learning

Linear Models I

Slides are adopted from lecture notes of Yaser Abu-Mostafa

Outline

- Input representation
- Linear classification
- Linear regression
- Nonlinear transformation

A real data set

7	4	7	3	6	3	1	0	1
8	1	1	1	7	4	8	0	1
2	7	4	8	7	3	7	4	1
0	7	4	1	3	7	7	4	5
9	7	4	1	3	7	7	4	6
0	2	0	8	6	6	2	0	8

Input representation

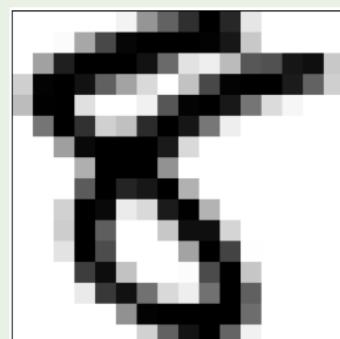
'raw' input $\mathbf{x} = (x_0, x_1, x_2, \dots, x_{256})$

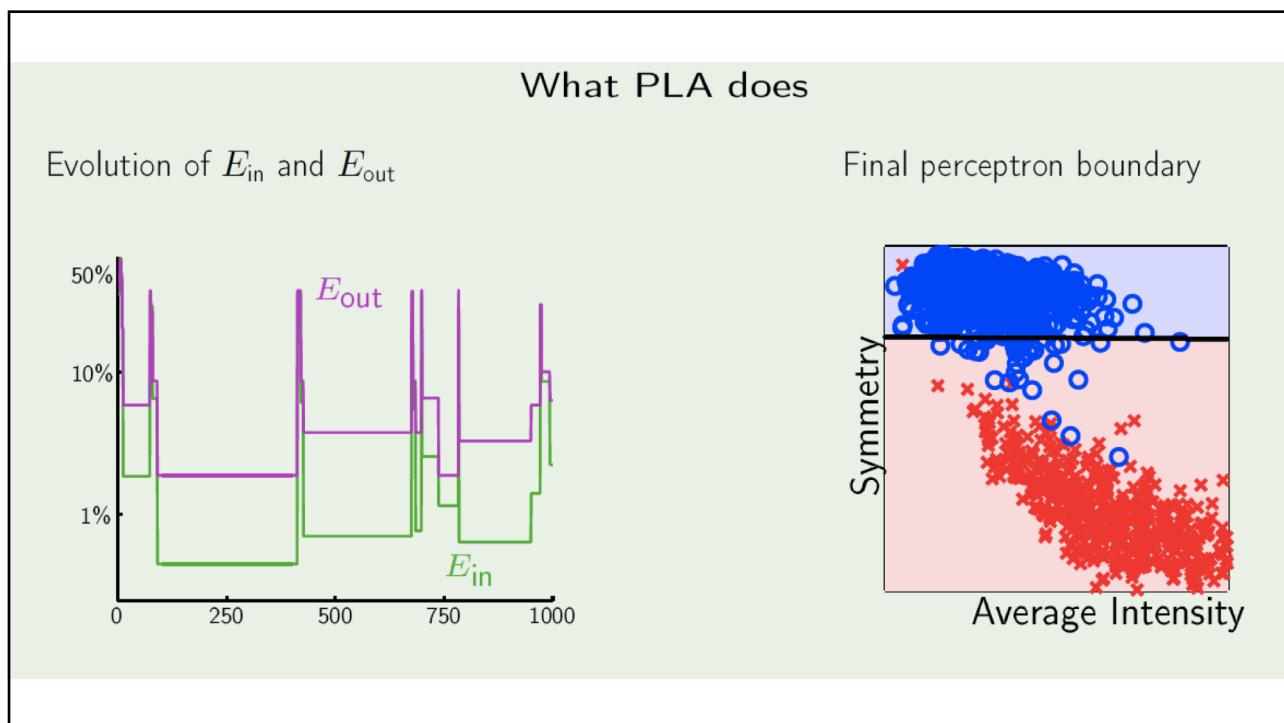
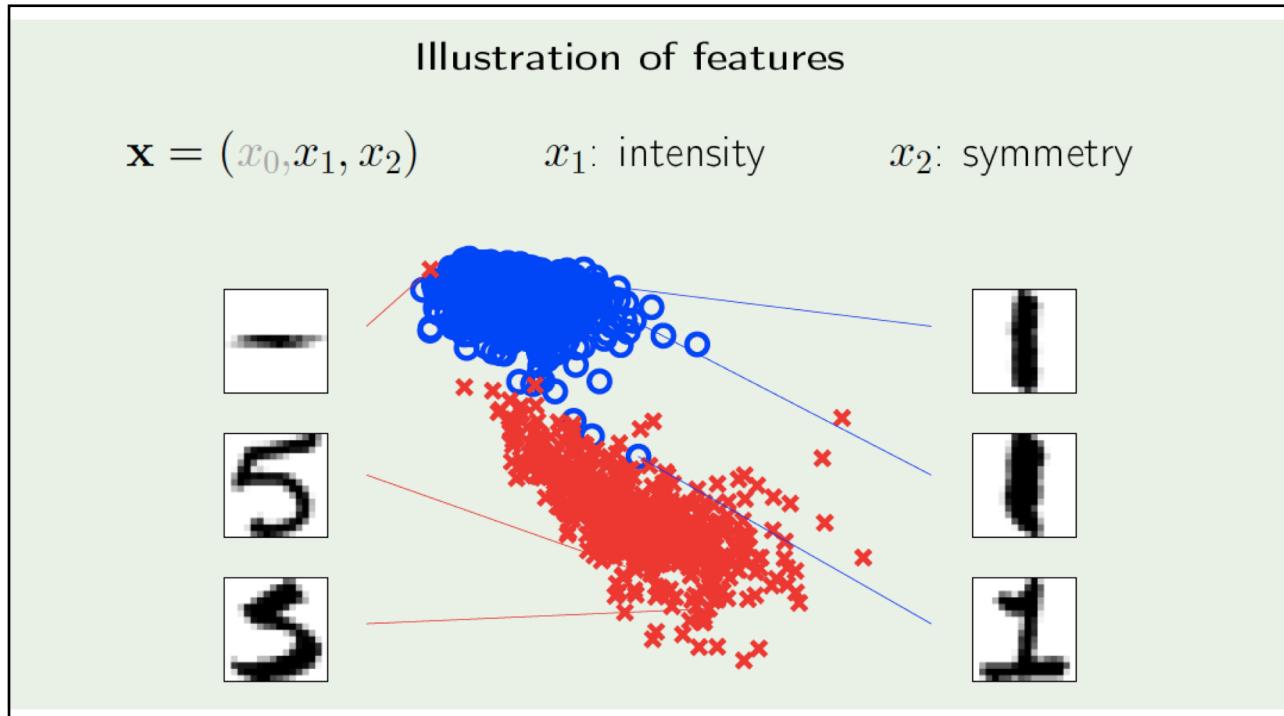
linear model: $(w_0, w_1, w_2, \dots, w_{256})$

Features: Extract useful information, e.g.,

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

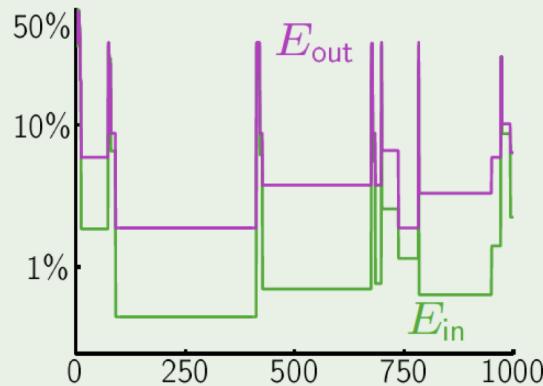
linear model: (w_0, w_1, w_2)



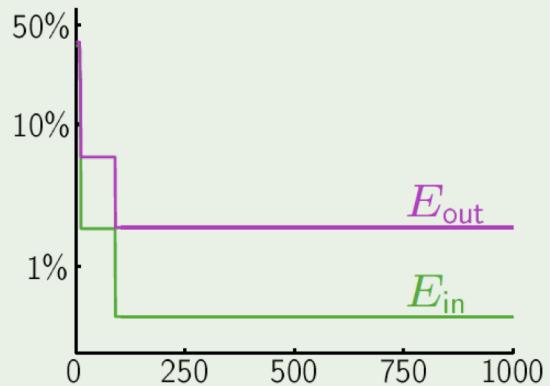


The ‘pocket’ algorithm

PLA:



Pocket:

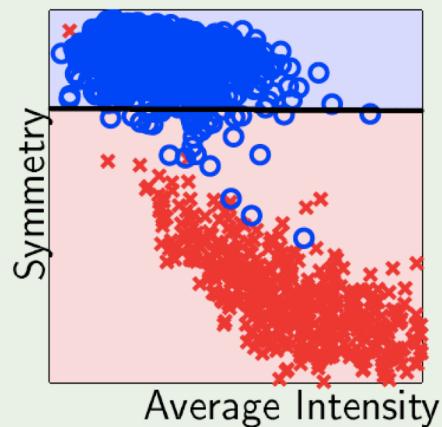


The pocket algorithm:

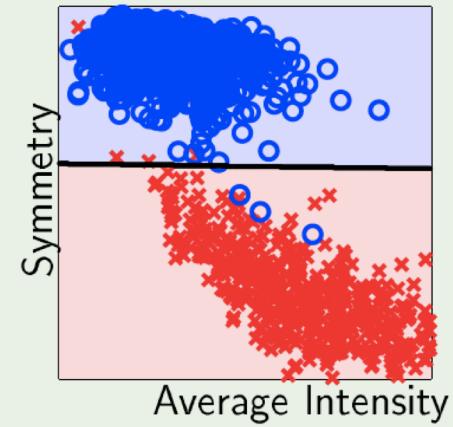
- 1: Set the pocket weight vector \hat{w} to $w(0)$ of PLA.
- 2: **for** $t = 0, \dots, T - 1$ **do**
- 3: Run PLA for one update to obtain $w(t + 1)$.
- 4: Evaluate $E_{in}(w(t + 1))$.
- 5: If $w(t + 1)$ is better than \hat{w} in terms of E_{in} , set \hat{w} to $w(t + 1)$.
- 6: Return \hat{w} .

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: $h(\mathbf{x}) = \sum_{i=0}^d w_i x_i = \mathbf{w}^\top \mathbf{x}$

The data set

Credit officers decide on credit lines:

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

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

Linear regression tries to replicate that.

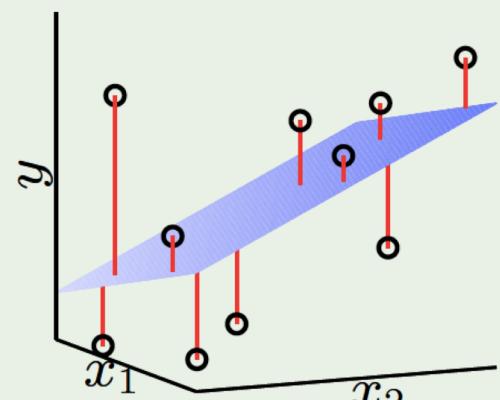
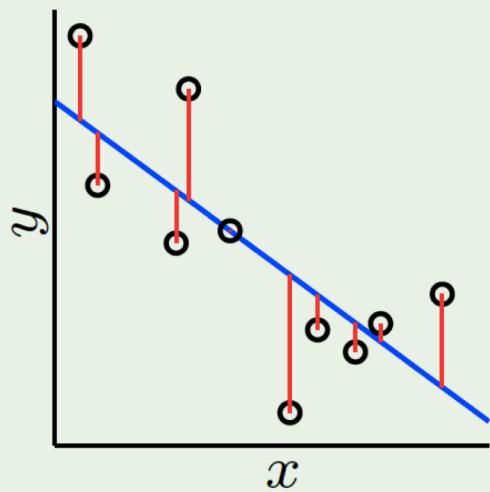
How to measure the error

How well does $h(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$ 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}

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

where $\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_N^\top \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$

Minimizing E_{in}

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

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

$$\mathbf{X}^\top \mathbf{X}\mathbf{w} = \mathbf{X}^\top \mathbf{y}$$

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

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

The Pseudo Inverse

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

$$\left(\begin{bmatrix} & \\ d+1 \times N & \end{bmatrix} \begin{bmatrix} & \\ N \times d+1 & \end{bmatrix}^{-1} \begin{bmatrix} & \\ d+1 \times N & \end{bmatrix} \right)$$

The pseudo-inverse

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

$$\left(\underbrace{\begin{bmatrix} & \\ d+1 \times d+1 & \end{bmatrix}}_{d+1 \times N} \right)^{-1} \begin{bmatrix} & \\ d+1 \times N & \end{bmatrix}$$

The linear regression algorithm

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

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

- 2: Compute the pseudo-inverse $\mathbf{X}^\dagger = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top$.
 3: Return $\mathbf{w} = \mathbf{X}^\dagger \mathbf{y}$.

Linear regression for classification

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

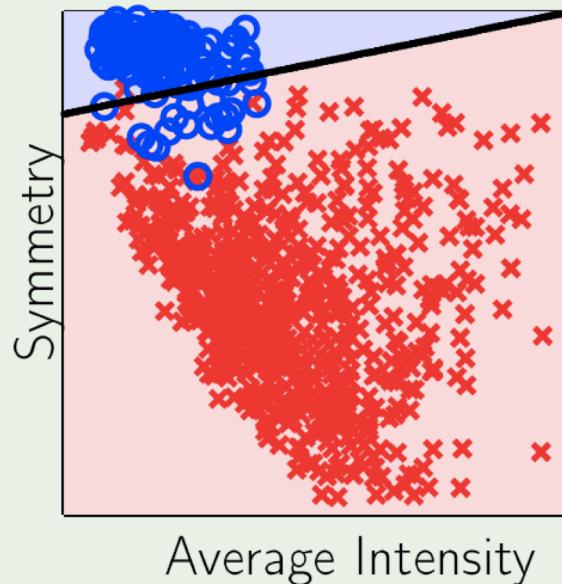
Binary-valued functions are also real-valued! $\pm 1 \in \mathbb{R}$

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

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

Good initial weights for classification

Linear regression boundary

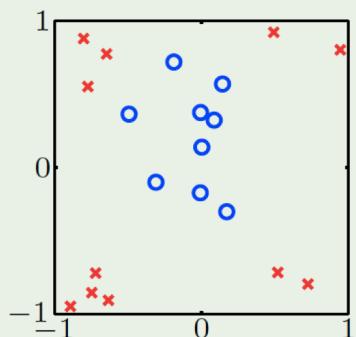


Outline

- Input representation
- Linear classification
- Linear regression
- Nonlinear transformation

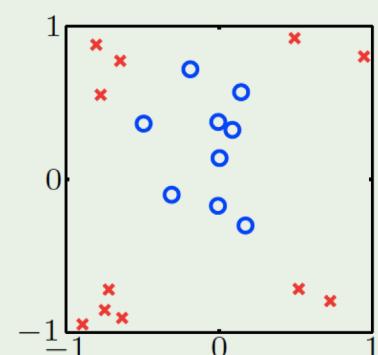
Linear is Limited

Data:

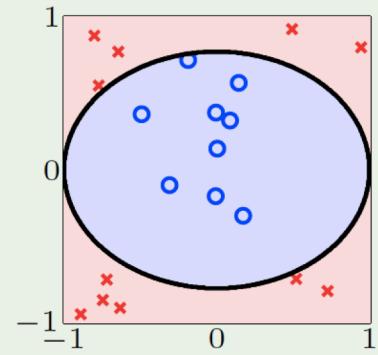


Linear is Limited

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

$$\sum_{i=0}^d \textcolor{red}{w}_i x_i$$

Linear classification implements

$$\text{sign} \left(\sum_{i=0}^d \textcolor{red}{w}_i x_i \right)$$

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

Transform the data nonlinearly

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

