Perceptron Learning Algorithm (1957, Rosenblatt)

Input: Training set D that is linearly separable

Output: $\underline{w} \in \mathbb{R}^{d+1}$ that achieves $E_{in}(\underline{w}) = 0$

Initialization

Initialize $\underline{w}$ in any arbitrary manner

$\underline{w} = (0, 0, 0, \ldots, 0) \in \mathbb{R}^{d+1}$

Step 1,

Check if $E_{in}(\underline{w}) = 0$

if yes, stop and output $\underline{w}$

Step 2,

$(\underline{x}_n, y_n)$ be any misclassified point

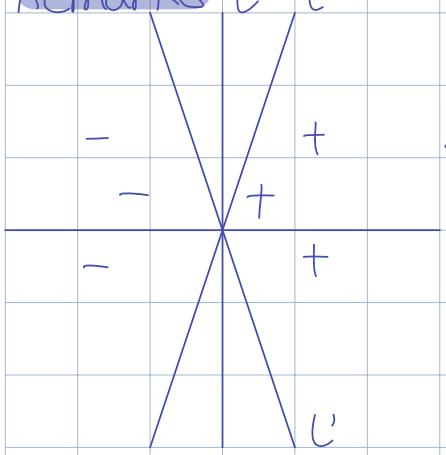
$\underline{w} \leftarrow \underline{w} + \underline{x}_n$ if $y_n = +1$

$\underline{w} \leftarrow \underline{w} - \underline{x}_n$ if $y_n = -1$

$\Rightarrow \underline{w} \leftarrow \underline{w} + y_n \underline{x}_n$

Go to step 1

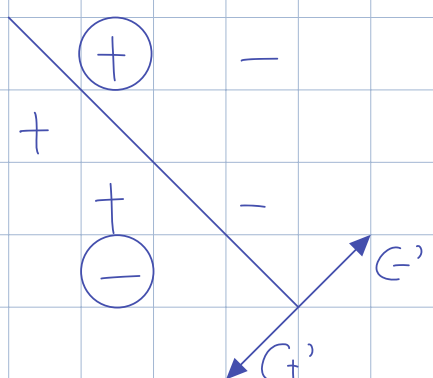
Remarks



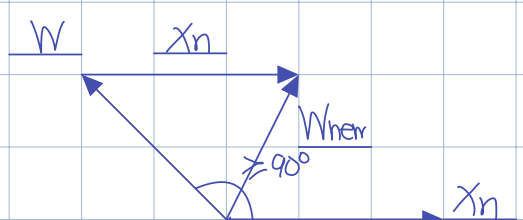
1) output is determined by the sequence of (x_n, y_n) in step 2
2) update rule is related gradient descent

Geometric Intuition for Update Rule

1) (x_n, y_n) is a misclassified point
$y_n = +1$ $\hat{y}_n = -1$ or x_n is on boundary

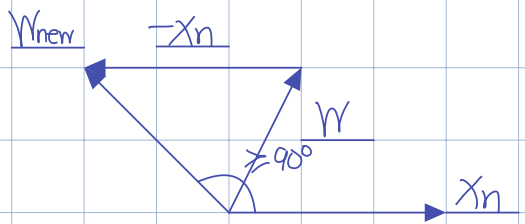


$\hat{y}_n = \text{sign}(w^T x_n)$
$w^T x_n \leq 0$



$W_{new} \leftarrow W + x_n$

$y_n = -1$ $\quad \hat{y}_n = +1$ $\quad w^T x_n \geq 0$



Problem 1.3 in textbook shows that PLA will converge in finitely many steps $E_{in}(w) = 0$

Pocket Algorithm

Idea: At each iteration, say t keep aside the "best weight vector" witnessed so far in iteration 1, 2, ..., t-1
Do the update in PLA and see if it yields a better weight vector (if so update accordingly)
Stop after sufficiently many iterations

Initialization

Initialize the "best weight vector" in some arbitrary manner

$\hat{w} = w(0) = (0, 0, 0, \ldots, 0) \in \mathbb{R}^{d+1}$

for $t = 0, 1, 2, \ldots, T-1$ do:

run PLA for one update to compute $w(t+1)$
evaluate $E_{in}(w(t+1))$
if $E_{in}(w(t+1)) < E_{in}(\hat{w})$:

$\hat{w} \leftarrow w(t+1)$

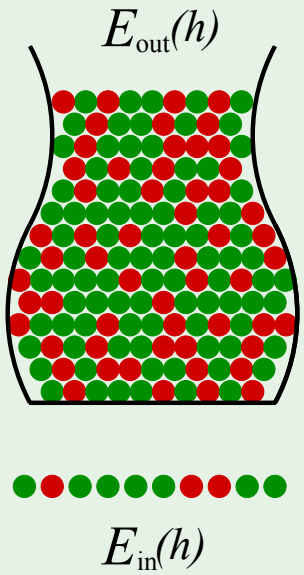
endif

output $\hat{w}$

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:

$$|E_{\text{in}}(h_1) - E_{\text{out}}(h_1)| > \epsilon \quad \textbf{or}$$
$$|E_{\text{in}}(h_2) - E_{\text{out}}(h_2)| > \epsilon \quad \textbf{or}$$
$$\cdots$$
$$|E_{\text{in}}(h_M) - E_{\text{out}}(h_M)| > \epsilon$$

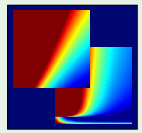
This gives us an added M factor.

Learning From Data

Yaser S. Abu-Mostafa
California Institute of Technology

Lecture 3: **Linear Models I**

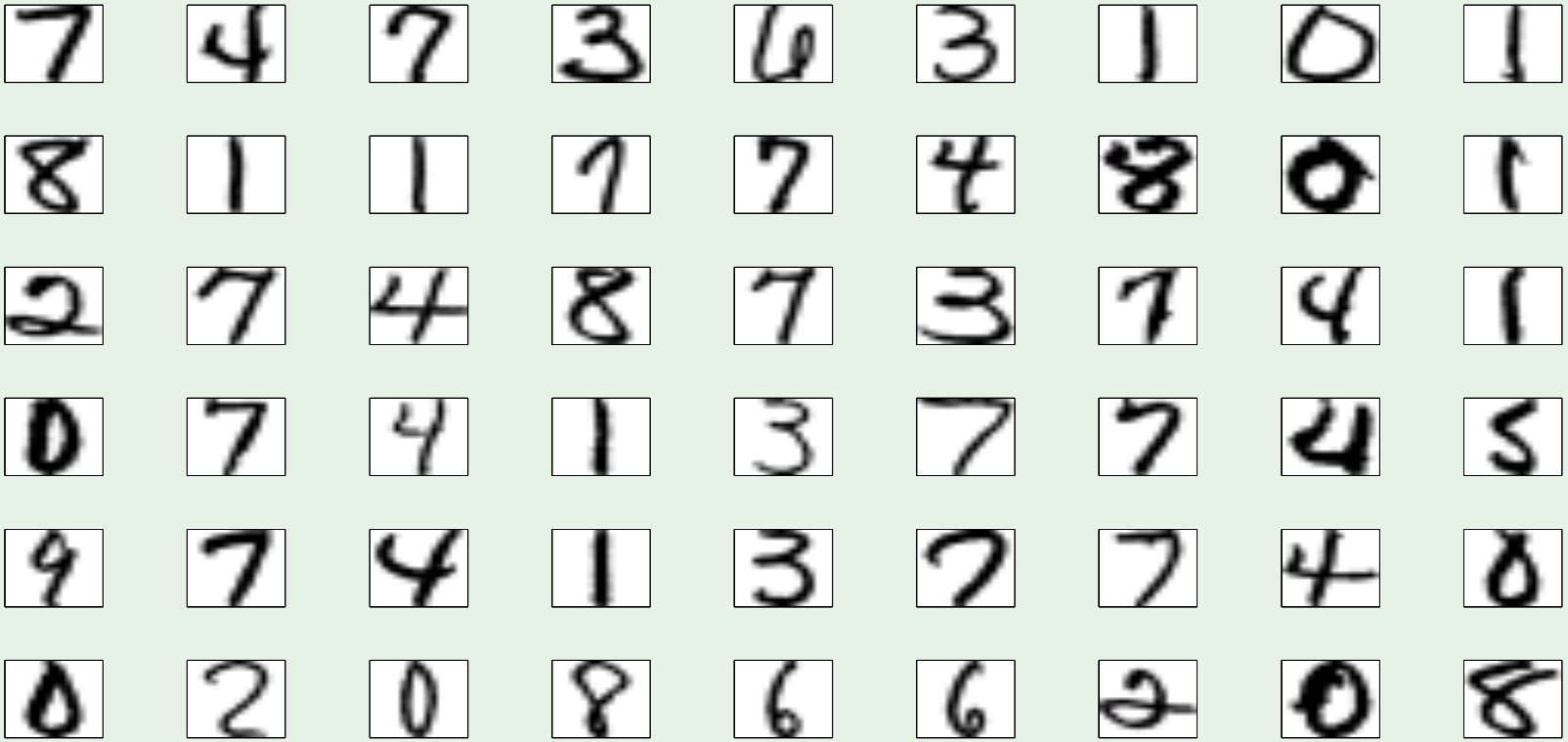
Sponsored by Caltech's Provost Office, E&AS Division, and IST • Tuesday, April 10, 2012



Outline

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

A real data set



16x16 Images

Input representation

'raw' input $\mathbf{x} = (x_0, x_1, x_2, \cdots, x_{256}) \in \mathbb{R}^{256+1}$

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

Features: Extract useful information, e.g.,

intensity and symmetry $\mathbf{x} = (x_0, x_1, x_2) \in \mathbb{R}^{2+1}$

linear model: (w_0, w_1, w_2)

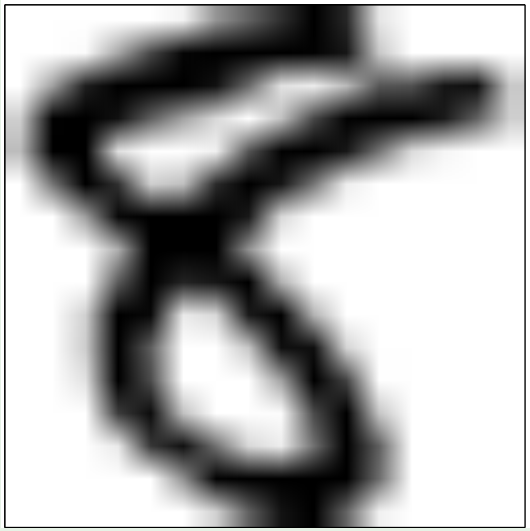
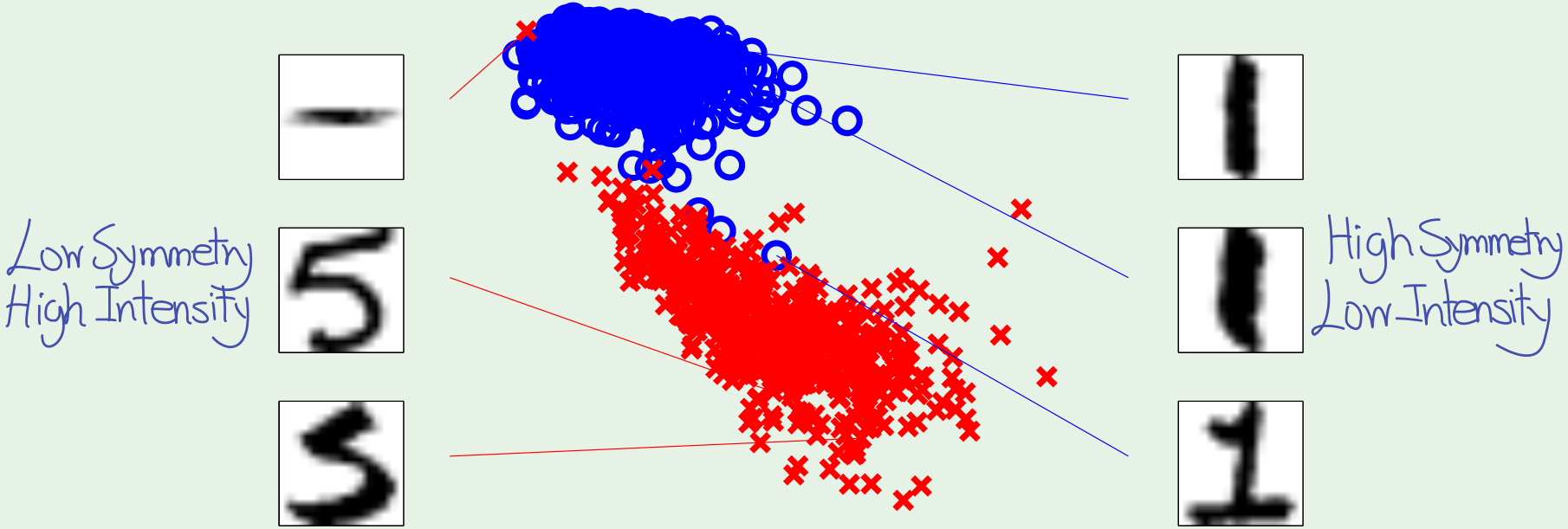


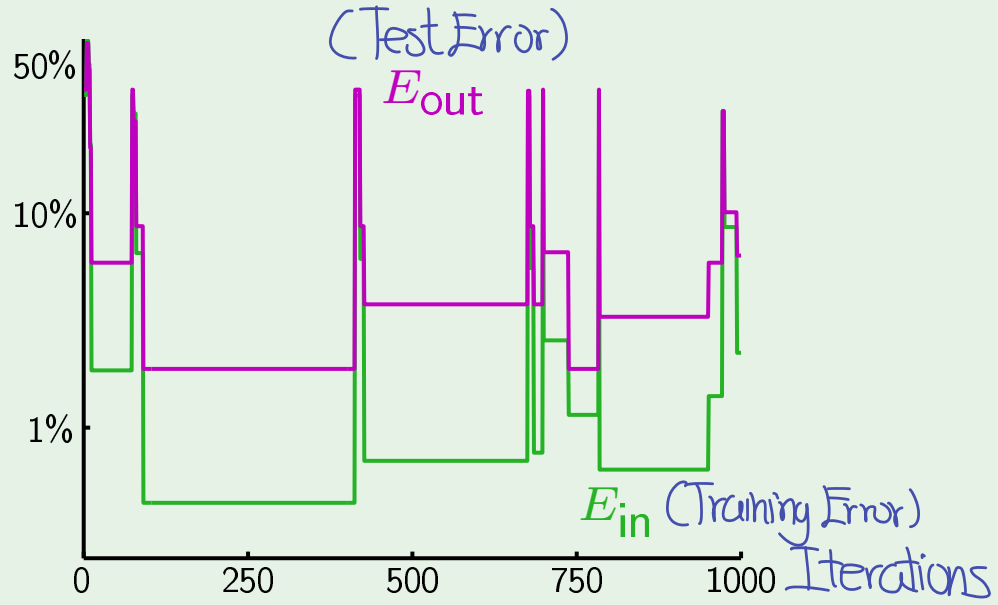
Illustration of features

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

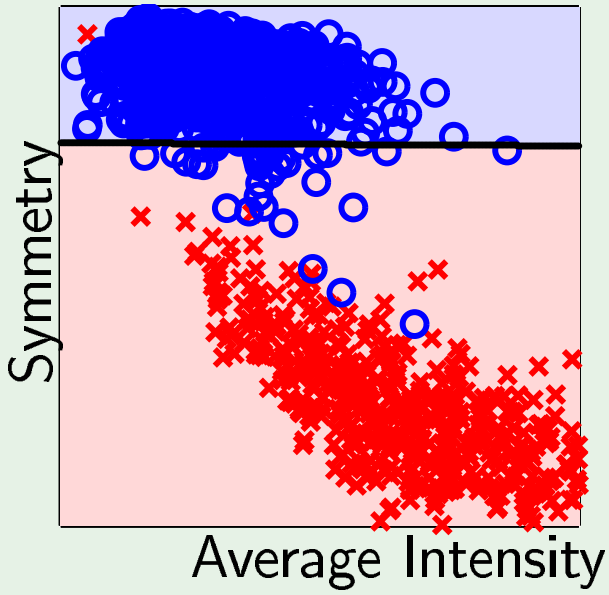


What PLA does

Evolution of E_{in} and E_{out}

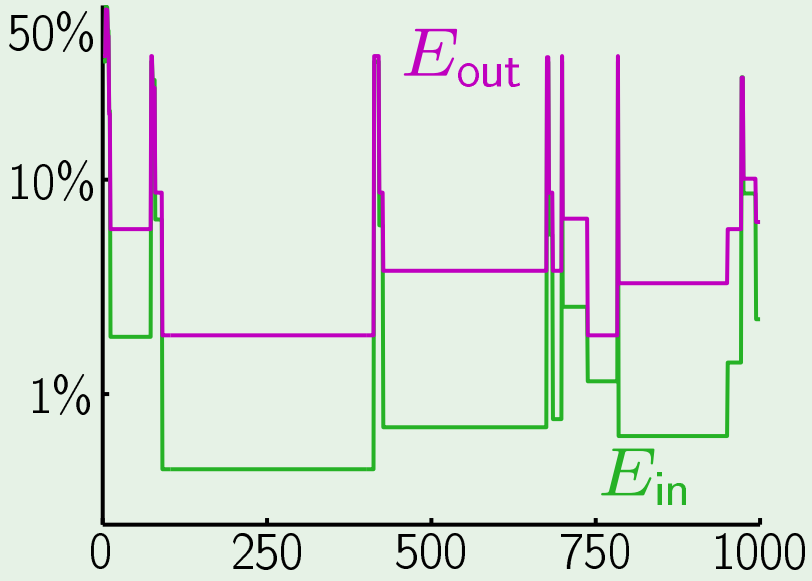


Final perceptron boundary

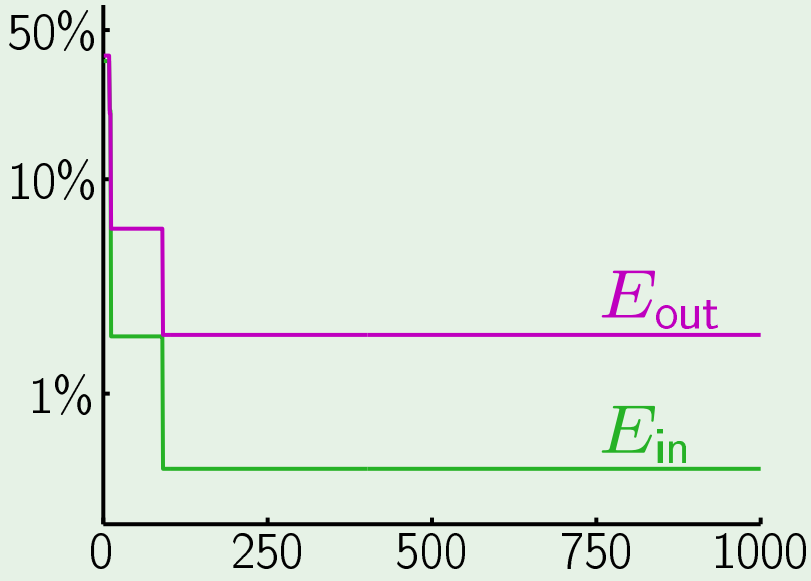


The 'pocket' algorithm

PLA:

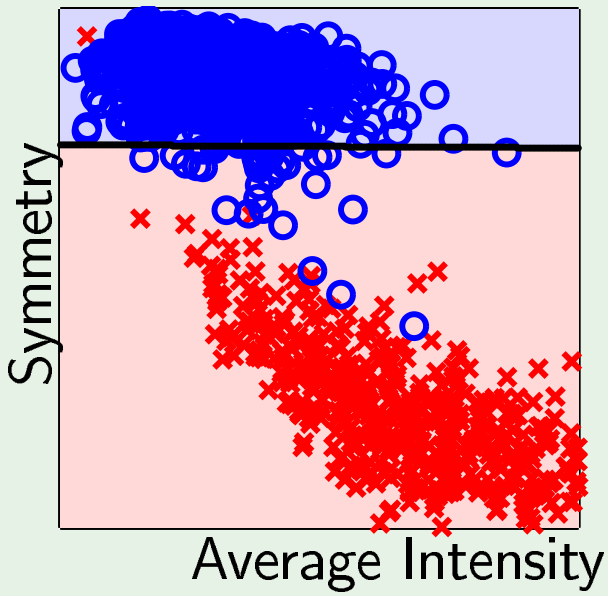


Pocket:

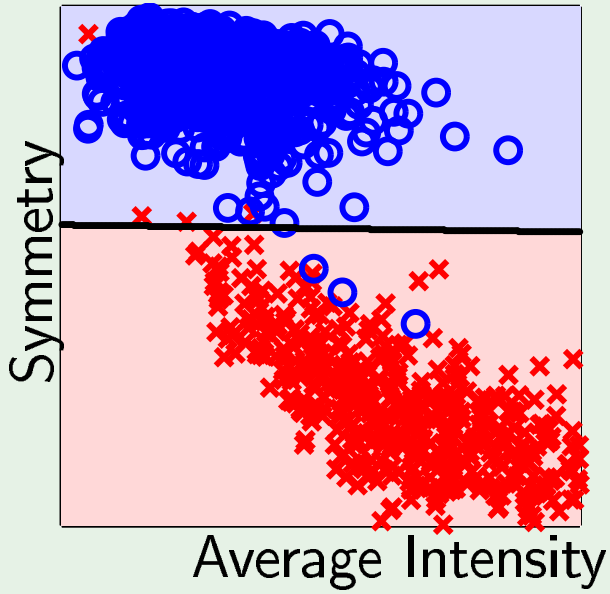


Classification boundary - PLA versus Pocket

PLA:



Pocket:



Outline

- Input representation
- Linear Classification
- Linear Regression **regression $\equiv$ real-valued output**
- Nonlinear Transformation

Linear Regression

Given $D = \{(\underline{x}_1, y_1), (\underline{x}_2, y_2), \ldots, (\underline{x}_d, y_d)\}$

$\underline{x}_i = (x_{i0}=1, x_{i1}, x_{i2}, \ldots, x_{id})$

$y_i \in \mathbb{R}$

Output a linear prediction function

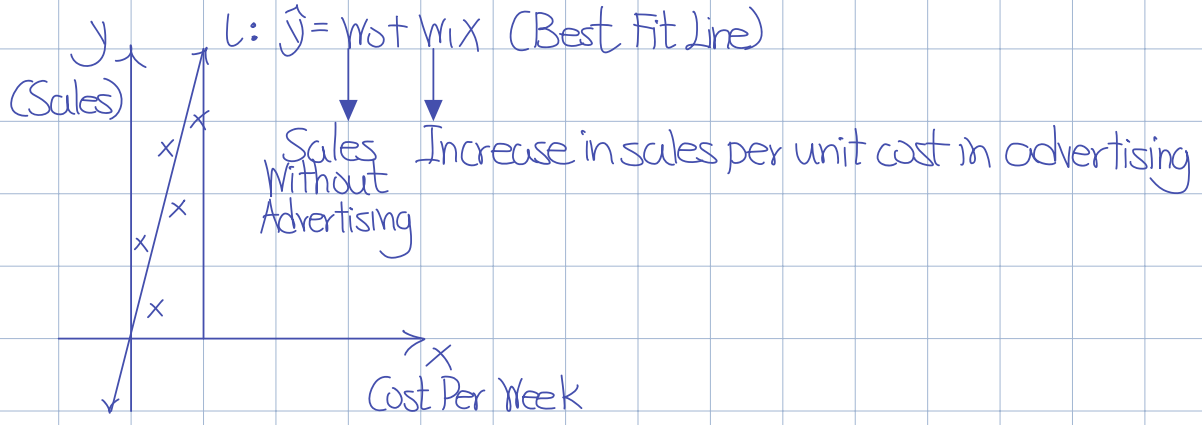
Given $\underline{x} = (x_0=1, x_1, x_2, \ldots, x_d)$

$\hat{y} = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_d x_d$

Example 1 ($d=1$)

x = advertising cost in 1 week

y = sales in that week



$d = 3$

$\underline{x} = (x_0, x_1, x_2, x_3)$

$x_0 = 1$

x_1 = cost per week in TV advertising

x_2 = cost per week in Radio advertising

X_3 = cost per week in Internet advertising

$\hat{y} = x_0 + w_1 x_1 + w_2 x_2 + w_3 x_3$

Notation

Data Vector

$$\underline{x_i} = \begin{bmatrix} x_{i0} = 1 \\ x_{i1} \\ \vdots \\ x_{id} \end{bmatrix} \in \mathbb{R}^{d+1}$$

$x_{i0} = 1 \rightarrow$ Arbitrary Scaling

Data Matrix

$$\mathcal{X} = \begin{bmatrix} \underline{x_1}^T \\ \underline{x_2}^T \\ \vdots \\ \underline{x_N}^T \end{bmatrix} \in \mathbb{R}^{N \times (d+1)}$$

Observation Vector

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \in \mathbb{R}^{N \times 1}$$

Weight Vector

$$w = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix} \in \mathbb{R}^{d+1}$$

Prediction Vector

Given x_i

$$\hat{y}_i = w_0 + w_1 x_{i1} + w_2 x_{i2} + \dots + w_d x_{id} = w^T x_i = x_i^T w$$

$$\hat{Y} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_N \end{bmatrix} \in \mathbb{R}^{N \times 1}$$

$$= \begin{bmatrix} x_1^T w \\ x_2^T w \\ \vdots \\ x_N^T w \end{bmatrix} = \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_N^T \end{bmatrix} w = \mathcal{X} w$$

Given $\mathcal{X}$, Y compute w that outputs $\hat{y} = \mathcal{X} w$

Loss Function

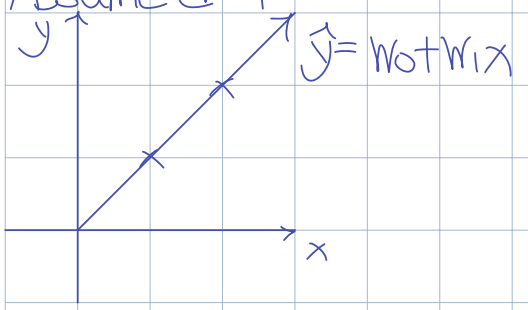
Ideal World

Find w such that all labels in training set are perfectly predicted

$y_1 = \hat{y}_1 = x_1^T w$ $\quad\quad$ $y_N = \hat{y}_N = x_N^T w$

$y_2 = \hat{y}_2 = x_2^T w$

Assume $d=1$



$N \leq d+1$

$$\left.\begin{aligned} y_1 &= w_0 + w_1 x_{11} + w_2 x_{12} + \dots + w_d x_{1d} \\ y_2 &= w_0 + w_1 x_{21} + w_2 x_{22} + \dots + w_d x_{2d} \\ &\vdots \\ y_N &= w_0 + w_1 x_{N1} + w_2 x_{N2} + \dots + w_d x_{Nd} \end{aligned}\right\} \begin{aligned} &\text{Equation 1} \\ &\text{Equation 2} \\ &\\ &\text{Equation } N \end{aligned}$$

Solve for $w_0, w_1, w_2, \dots, w_d$

# of variable $\geq$ # of constraints

$d+1 \geq N$

In practice, $N \gg d$

Typically, $N \approx 10d$

Loss Function

$$E_{in}(w) = \underbrace{\|y - \hat{y}\|^2}_{\substack{\text{observation} \\ \text{vector}}\ \ \substack{\text{prediction} \\ \text{vector}}} = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

$$= \|y - Xw\|^2$$

(Squared Error Loss)

$$\underline{w}^* = \underset{\underline{w} \in \mathbb{R}^{d+1}}{\operatorname{argmin}} \; E_{in}(\underline{w})$$

$$\underline{w}^* = \underbrace{(\underbrace{X^T}_{\mathbb{R}^{(d+1)\times N}} \underbrace{X}_{\mathbb{R}^{N\times(d+1)}})^{-1}}_{\mathbb{R}^{(d+1)\times(d+1)}} \underbrace{X^T}_{\mathbb{R}^{(d+1)\times N}} \underbrace{y}_{\mathbb{R}^{N\times 1}}$$

Optimal Weight Vector Solved Analytically

Special Case ($d = 1$)

$\underline{w} = (w_0, w_1)$

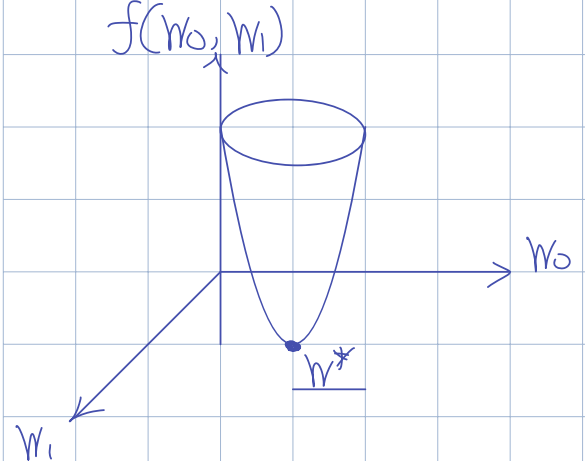
$\hat{y} = w_0 + w_1 x \qquad \hat{y}_i = w_0 + w_1 x_i$

$$E_{in}(\underline{w}) = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

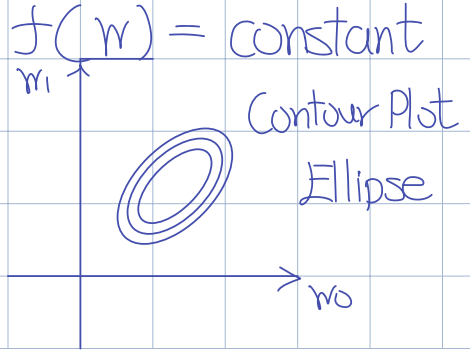
$$= f(\underline{w})$$

$$= \sum_{i=1}^{N} (y_i - w_0 - w_1 x_i)^2$$

minimize $E_{in}(\underline{w})$



Convex Function
Paraboloid



To minimize, $f(\underline{w}) \longrightarrow (w_0, w_1, w_2, \ldots, w_d)$

$$\nabla_{\underline{w}} f(\underline{w}) = 0$$

Gradient of $f(\cdot)$ wrt $\underline{w}$

$$\begin{vmatrix} \partial f/\partial w_0 \\ \partial f/\partial w_1 \\ \vdots \\ \partial f/\partial w_d \end{vmatrix} = 0$$

Gradient

1) Linear function

$$g(\underline{w}) = w_0 + w_1 v_1 + w_2 v_2 + \ldots + w_d v_d$$
$$= \underline{w^T v} = \underline{v^T w}$$

$\underline{v}$: fixed vector

$$\nabla_{\underline{w}} g(\underline{w}) = \begin{vmatrix} \partial g/\partial w_0 \\ \partial g/\partial w_1 \\ \vdots \\ \partial g/\partial w_d \end{vmatrix} = \begin{vmatrix} 1 \\ v_1 \\ \vdots \\ v_d \end{vmatrix} = \underline{v}$$

$g(\underline{w}) = \underline{w}^T A \underline{w}$

$A \in \mathbb{R}^{(d+1)\times(d+1)}$ matrix

$\nabla g(\underline{w}) = (A + A^T)\underline{w}$

$= 2A\underline{w}$ if $A = A^T$

$f(\underline{w}) = \|\underline{Y} - \hat{\underline{Y}}\|^2$

$= \|\underline{Y} - \underline{X}\underline{w}\|^2$

$= (\underline{Y} - \underline{X}\underline{w})^T(\underline{Y} - \underline{X}\underline{w})$

$= \underline{Y}^T\underline{Y} - \underline{Y}^T\underline{X}\underline{w} - (\underline{X}\underline{w})^T\underline{Y} + (\underline{X}\underline{w})^T(\underline{X}\underline{w})$

$= \underline{Y}^T\underline{Y} - 2\underline{w}^T\underline{X}^T\underline{Y} + \underline{w}^T\underline{X}^T\underline{X}\underline{w}$

$\nabla_{\underline{w}} f(\underline{w}) = \nabla_{\underline{w}}(\underline{Y}^T\underline{Y}) - 2\nabla_{\underline{w}}(\underline{w}^T\underline{X}^T\underline{Y}) + \nabla_{\underline{w}}(\underline{w}^T\underline{X}^T\underline{X}\underline{w}) = 0$

$= -2\underline{X}^T\underline{Y} + 2\underline{X}^T\underline{X}\underline{w} = 0$

$\underline{X}^T\underline{X}\underline{w} = \underline{X}^T\underline{Y}$

$\underline{w} = (\underline{X}^T\underline{X})^{-1}\underline{X}^T\underline{Y}$

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.

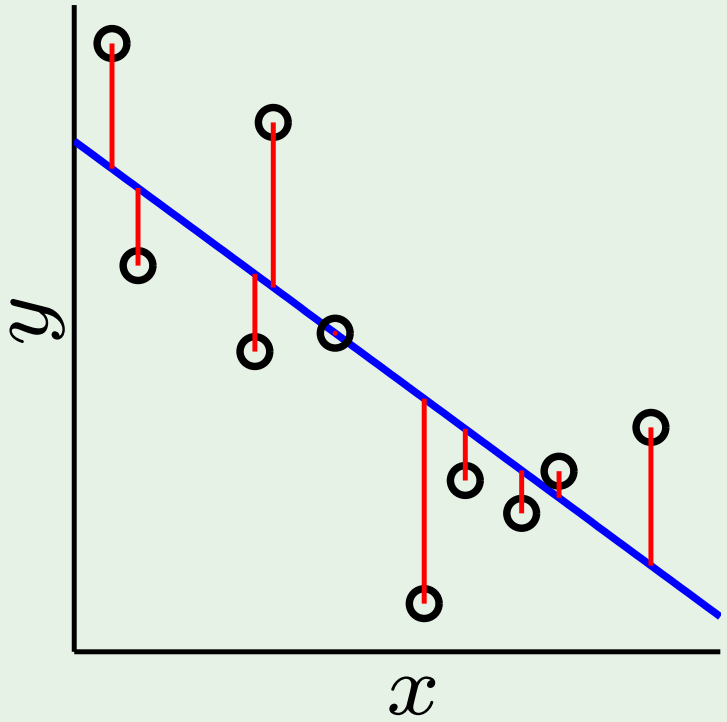
How to measure the error

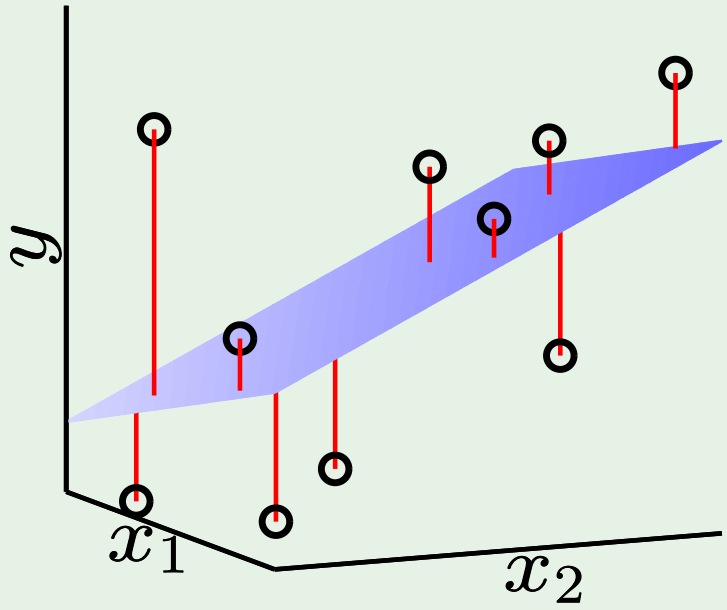
How well does $h(\mathbf{x}) = \mathbf{w}^{\mathsf{T}}\mathbf{x}$ approximate $f(\mathbf{x})$?

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

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

Illustration of linear regression





The expression for E_{in}

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}\left(\mathbf{w}^{\mathsf{T}}\mathbf{x}_n - y_n\right)^2$$

$$= \frac{1}{N}\|\mathrm{X}\mathbf{w} - \mathbf{y}\|^2$$

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}

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

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

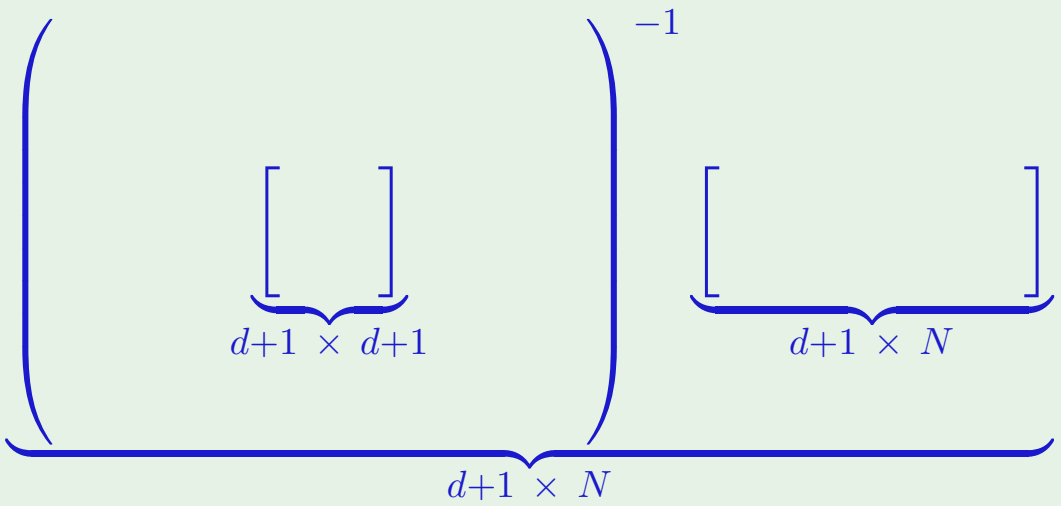
$$\mathrm{X}^{\mathsf{T}}\mathrm{X}\mathbf{w} = \mathrm{X}^{\mathsf{T}}\mathbf{y}$$

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

$\mathrm{X}^{\dagger}$ 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 X and the vector $\mathbf{y}$ from the data set $(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_N, y_N)$ as follows

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

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

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

Linear regression for classification

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

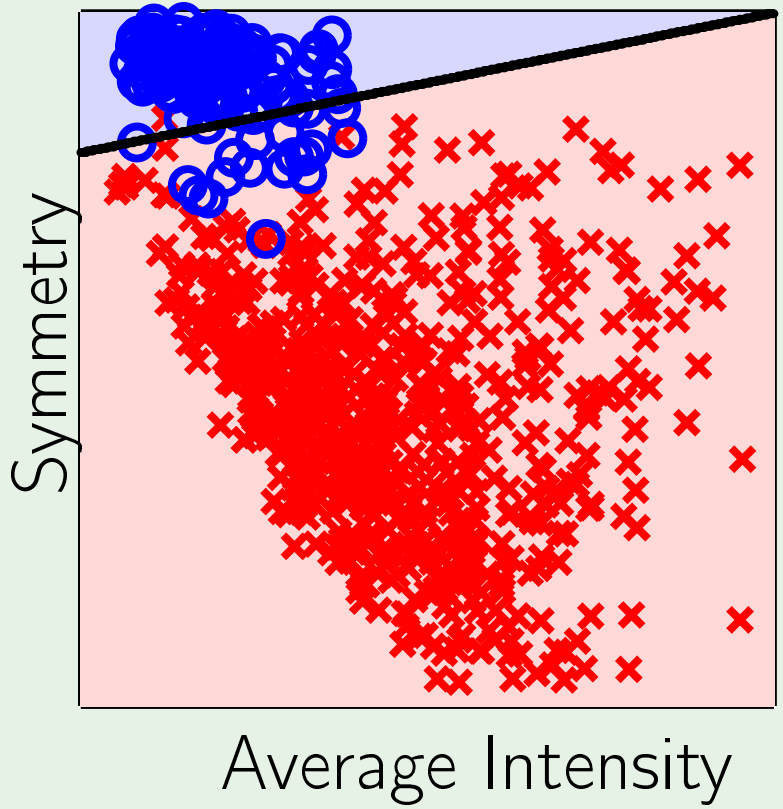
Binary-valued functions are also real-valued! $\pm 1 \in \mathbb{R}$

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

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

Good initial weights for classification

Linear regression boundary

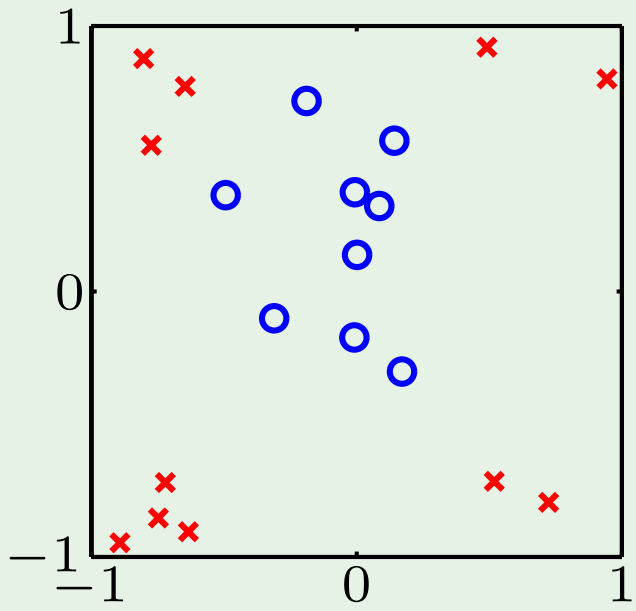


Outline

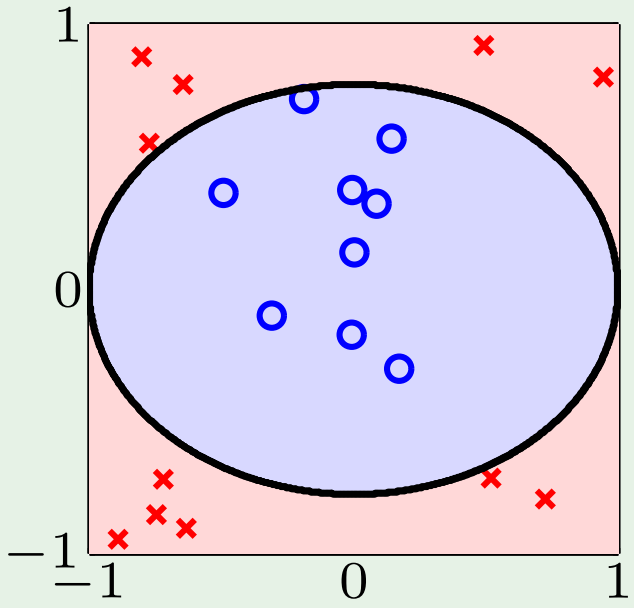
- Input representation
- Linear Classification
- Linear Regression
- Nonlinear Transformation

Linear is limited

Data:



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

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