CSC442: Introduction to Artificial Intelligence

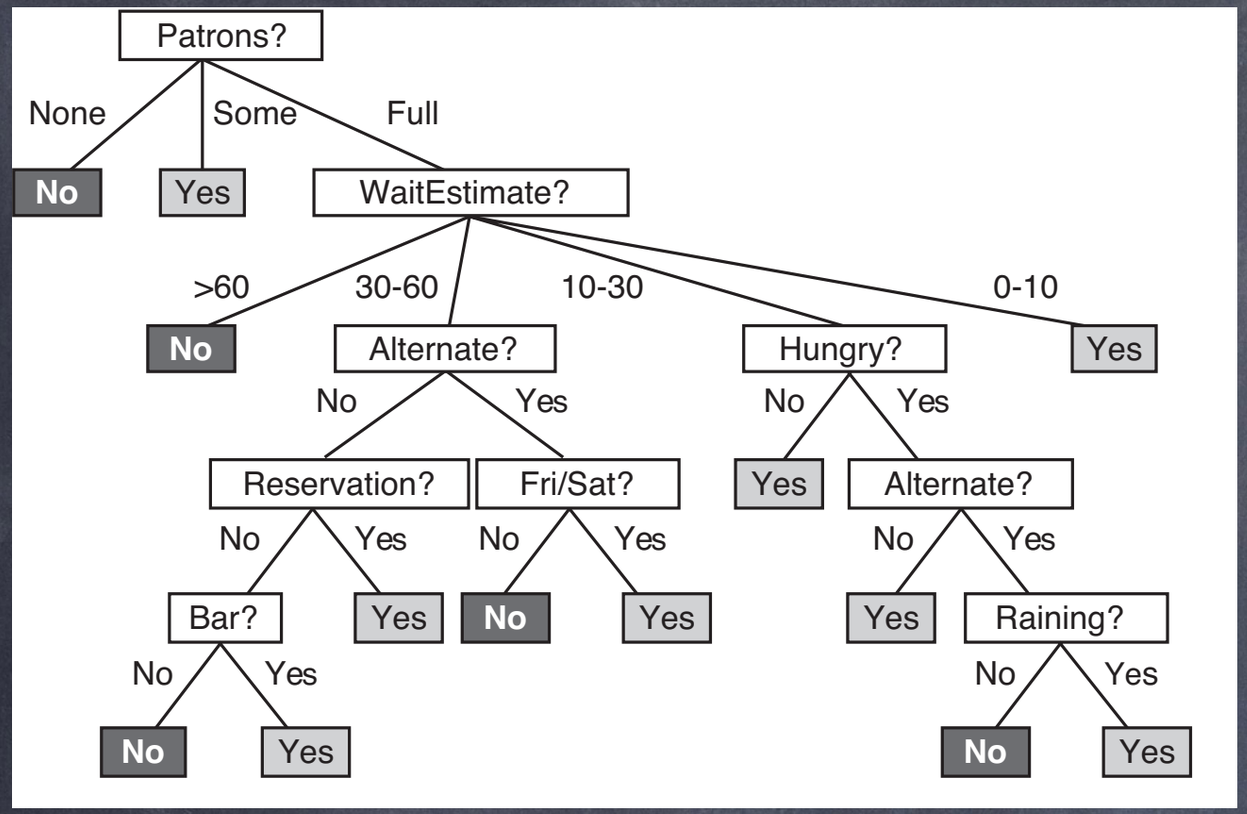
Linear Models for Regression and Classification

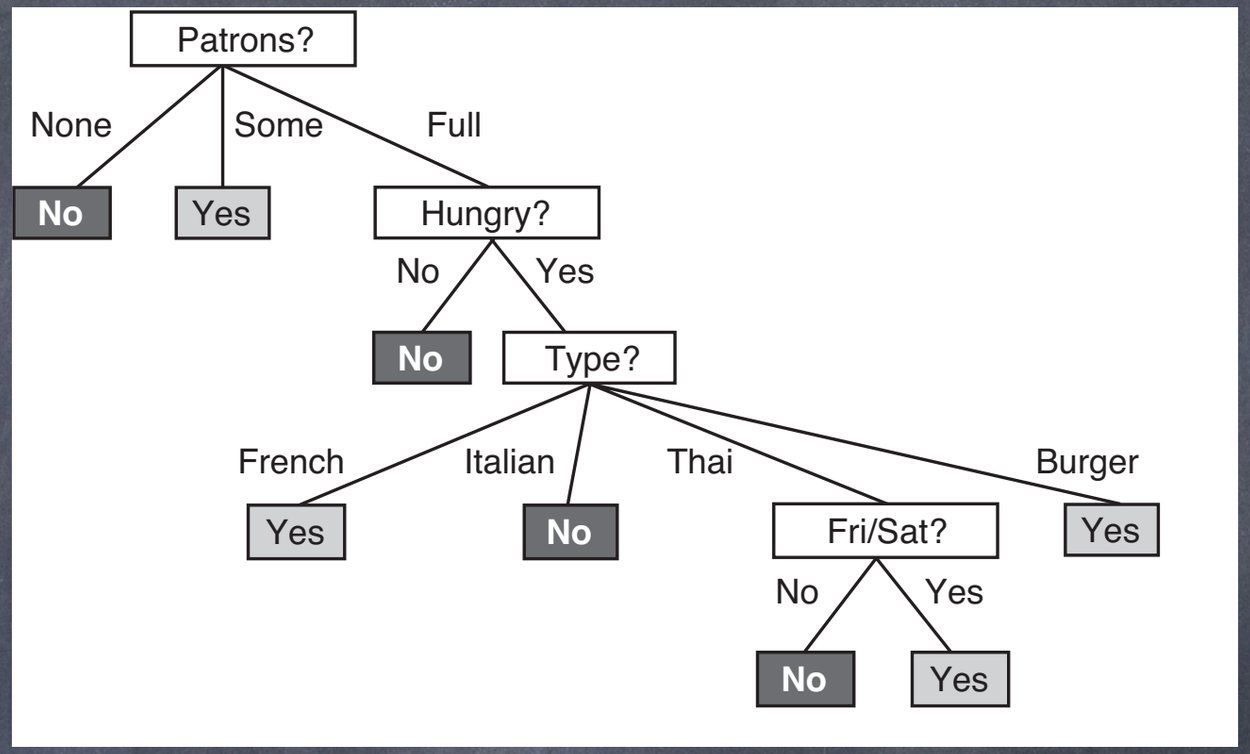
Learning

- Why?
  - Knowledge without programming
- What?
  - Probabilities, distributions, rules, ...
- How?
  - Unsupervised $\leftrightarrow$ Supervised

Decision Trees

- Represent sequence of tests that lead to a decision
- Compact representation of how to make the decision
- Can be learned from examples
- Decision trees have explanatory power!





Learning lets the data speak for itself

Supervised Learning

- Given a training set of N example input-output pairs:

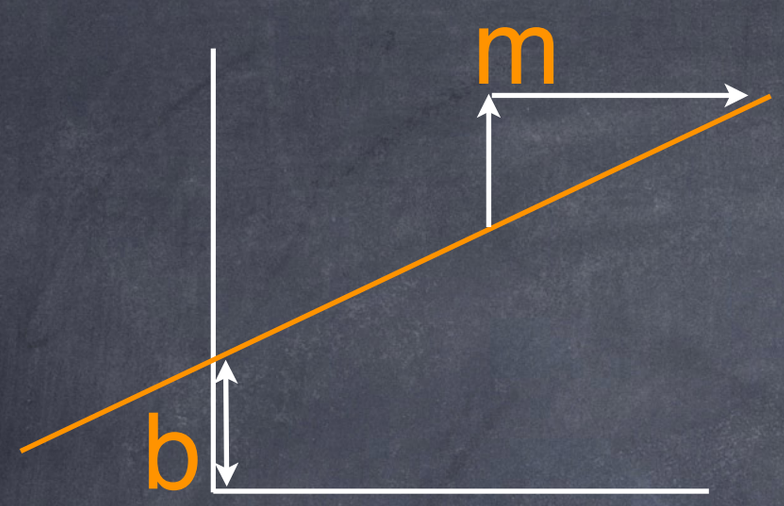
  $$(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$$

  where each $y_j = f(x_j)$
- Discover function h that approximates f
- Search through the space of possible hypotheses for one that will perform well

Regression and Classification with Linear Models

Outline

- Linear regression: Learning a linear function from examples
- Linear Classifiers: Using a line to separate data into classes
  - Hard and soft thresholds
- Neural Nets & Support Vector Machines
  - Powerful applications of linear classifiers



$$y = mx + b$$

$$= w_1 x + w_0$$

$$= w_0 + w_1 x$$

$$w = [w_0, w_1]$$

$$x = [1, x]$$

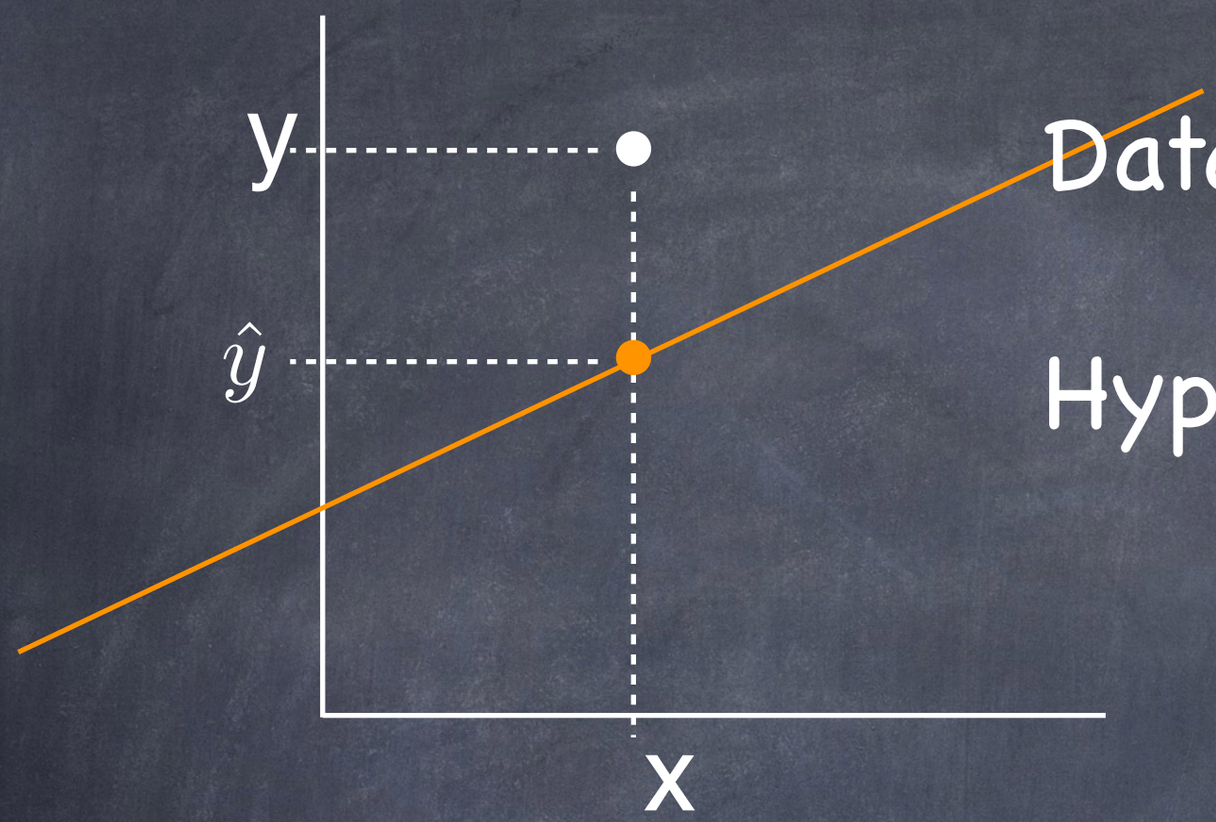
$$y = w \cdot x$$

Univariate Linear Regression

- Given a set of N data points (x,y)
- Find linear function (line)

  $$h_w(x) = w \cdot x = w_0 + w_1 x$$

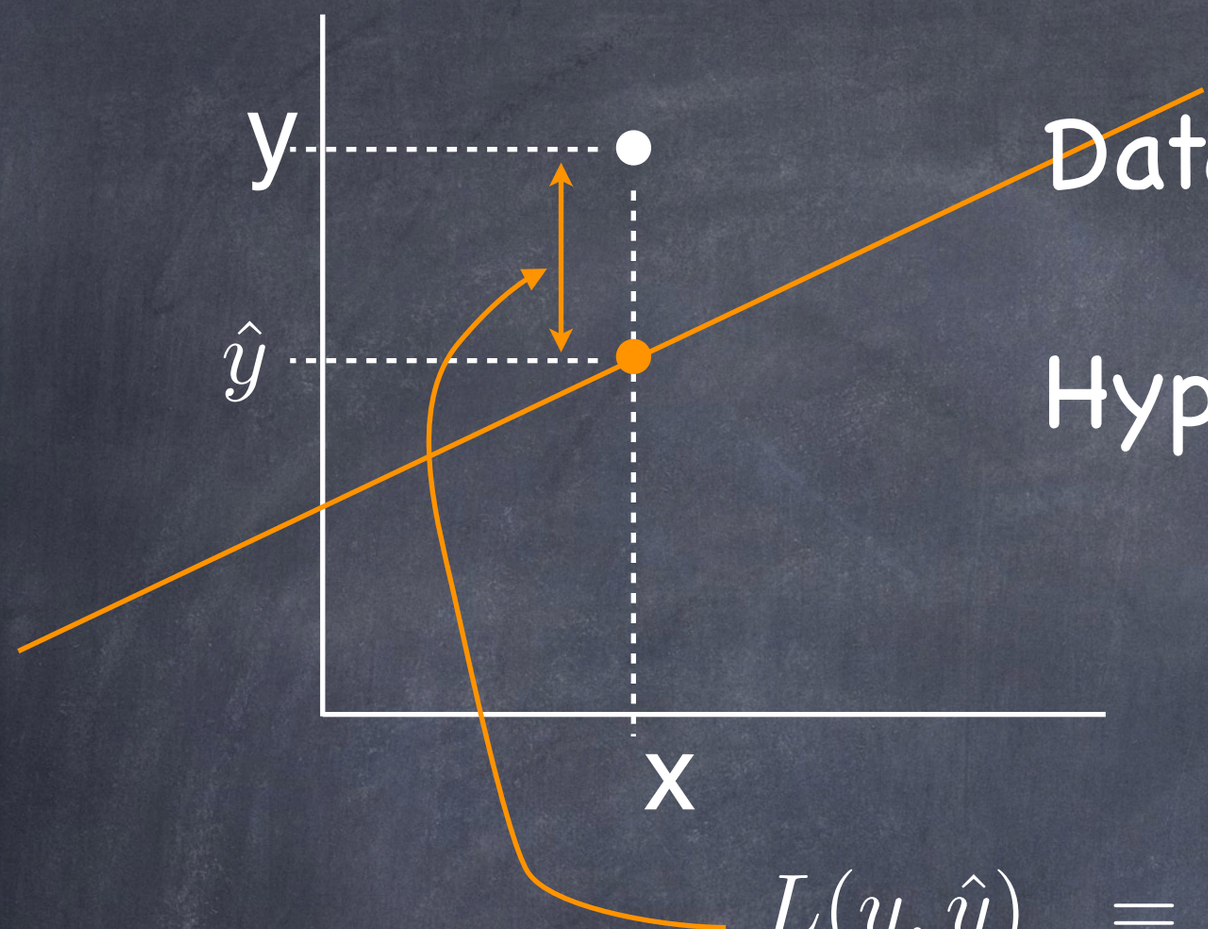
  that best fits the data



Data point (x,y) from y = f(x)

Hypothesis $h_w(x) = w_1 x + w_0$

$$\hat{y} = h_{\mathrm{w}}(x) = w_1 x + w_0$$



Data point (x,y) from y = f(x)

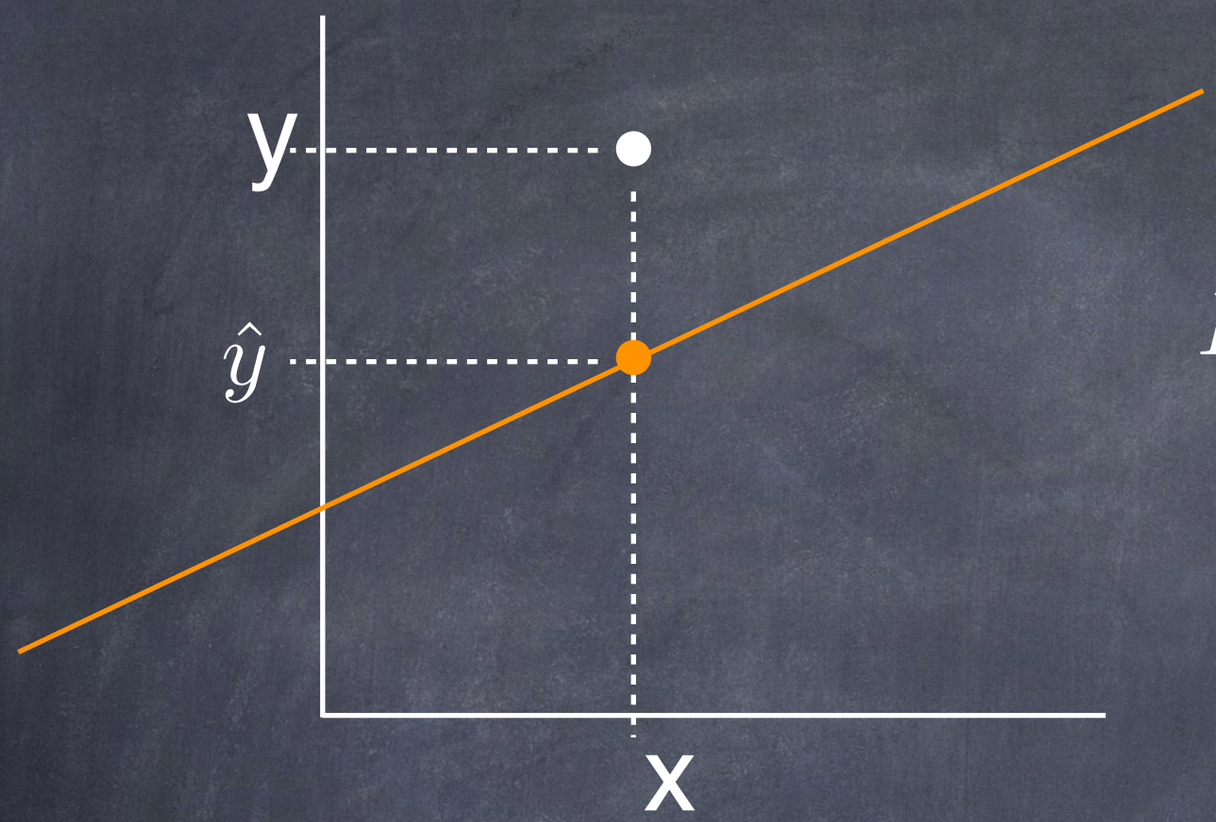
Hypothesis $h_w(x) = w_1 x + w_0$

$$\hat{y} = h_{\mathrm{w}}(x) = w_1 x + w_0$$

$$L(y, \hat{y}) = |y - \hat{y}| = L_1(y, \hat{y})$$

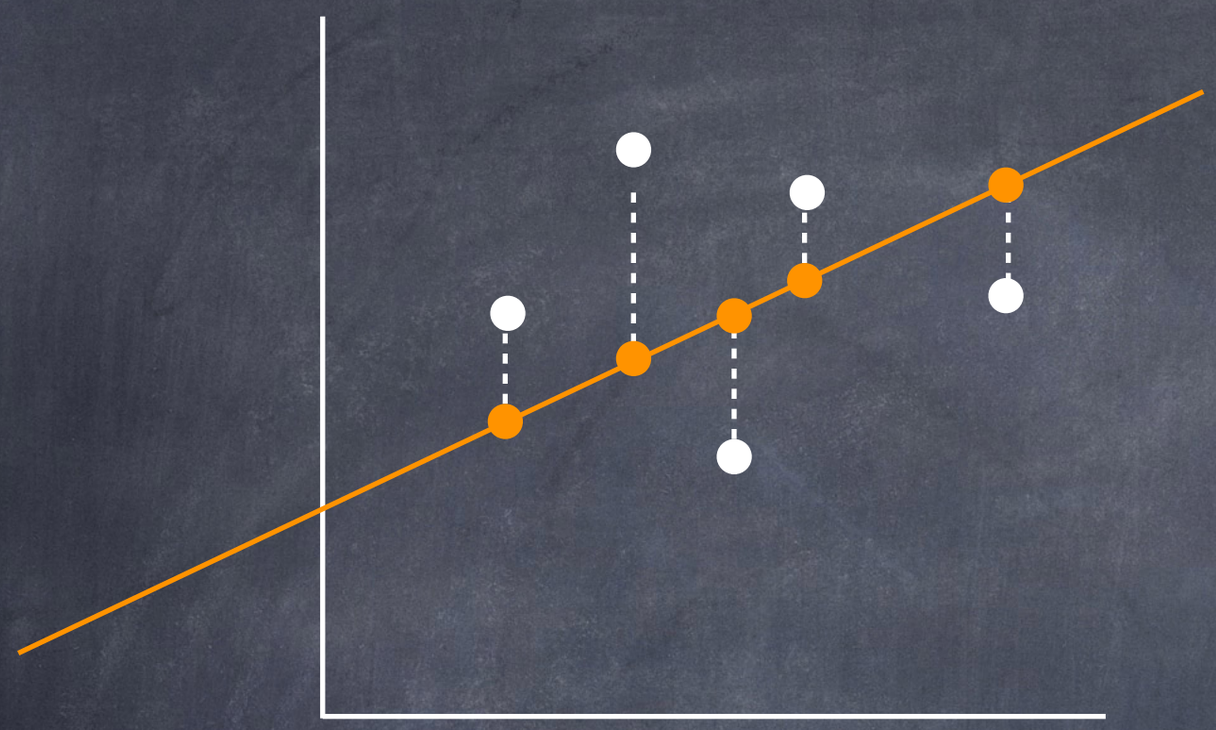
$$= (y - \hat{y})^2 = L_2(y, \hat{y})$$

$$= 0 \text{ if } y = \hat{y} \text{ else } 1 = L_{0/1}(y, \hat{y})$$



$$h_w(x) = w_1 x + w_0$$

$$\begin{aligned} L(y, h_{\mathbf{w}}(x)) &= L_2(y, h_{\mathbf{w}}(x)) \\ &= (y - h_{\mathbf{w}}(x))^2 \\ &= (y - (w_1 x + w_0))^2 \end{aligned}$$



$$h_w(x) = w_1 x + w_0$$

$$L(h_{\mathrm{w}}) = \sum_{j=1}^{N} L_2(y_j, h_{\mathbf{w}}(x_j))$$

$$= \sum_{j=1}^{N} (y_j - h_{\mathbf{w}}(x_j))^2$$

$$= \sum_{j=1}^{N} (y_j - (w_1 x_j + w_0))^2$$

Univariate Linear Regression

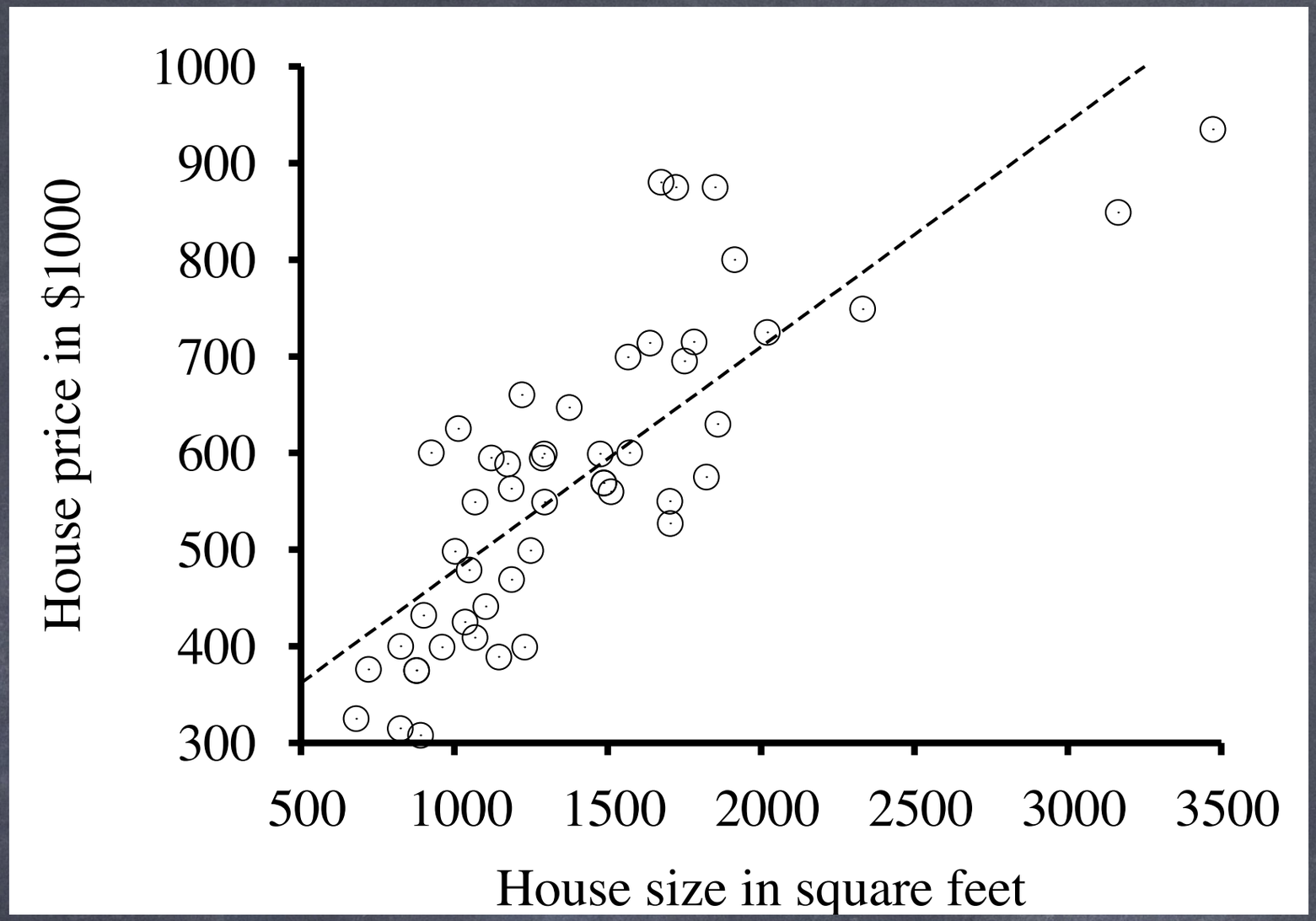
Find $w = [w_0, w_1]$ that minimizes $L(h_w)$

$$\mathrm{w}^* = \underset{\mathrm{w}}{\operatorname{argmin}}\, L(h_{\mathrm{w}})$$

$$= \underset{\mathrm{w}}{\operatorname{argmin}} \sum_{j=1}^{N} (y_j - (w_1 x_j + w_0))^2$$

$$w_1 = \frac{N(\sum x_j y_j) - (\sum x_j)(\sum y_j)}{N(\sum x_j^2) - (\sum x_j)^2}$$

$$w_0 = \frac{(\sum y_j - w_1(\sum x_j))}{N}$$


1000
900
800
700
600
500
400
300
House price in $1000
500
1000
1500
2000
2500
3000
3500
House size in square feet

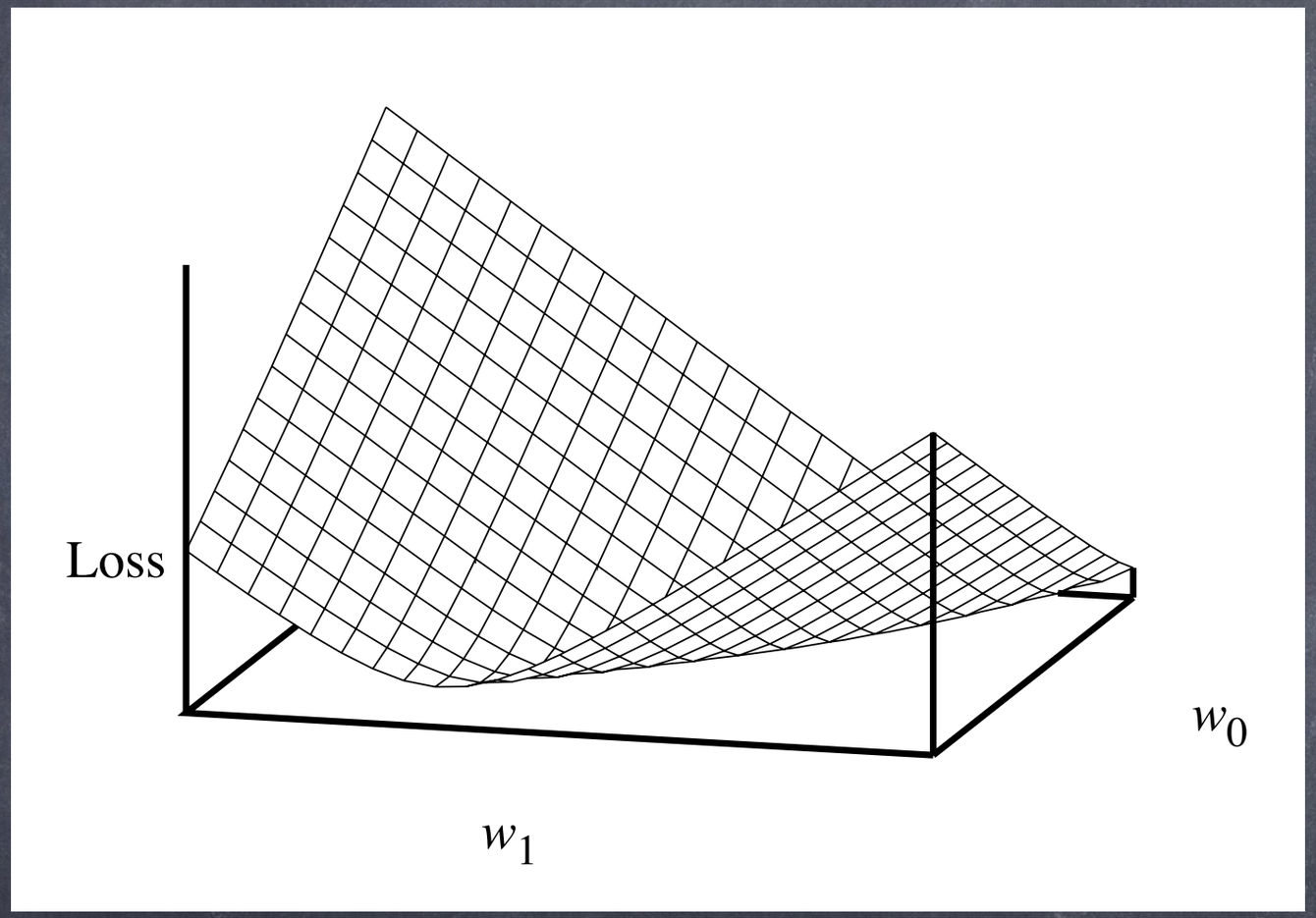
$$y = 0.232x + 246$$

General Learning

Find w that minimizes $L(h_w)$

$$\mathrm{w}^* = \underset{\mathrm{w}}{\operatorname{argmin}}\, L(h_{\mathrm{w}})$$

Weight Space



Gradient Descent

w ← any point in parameter space

`loop` until convergence do

`for each` w_i `in` w do

Gradient of loss function along w_i axis

Update rule

$$w_i \leftarrow w_i - \alpha \frac{\partial}{\partial w_i} L(\mathbf{w})$$

Learning rate

Gradient Descent for Univar. Linear Regression

$$w_0 \leftarrow w_0 + \alpha \sum_j (y_j - h_{\mathbf{w}}(x_j))$$

$$w_1 \leftarrow w_1 + \alpha \sum_j (y_j - h_{\mathbf{w}}(x_j)) \times x_j$$

Gradient Descent for Univar. Linear Regression

$$w_0 \leftarrow w_0 + \alpha \sum_j (y_j - h_{\mathbf{w}}(x_j))$$

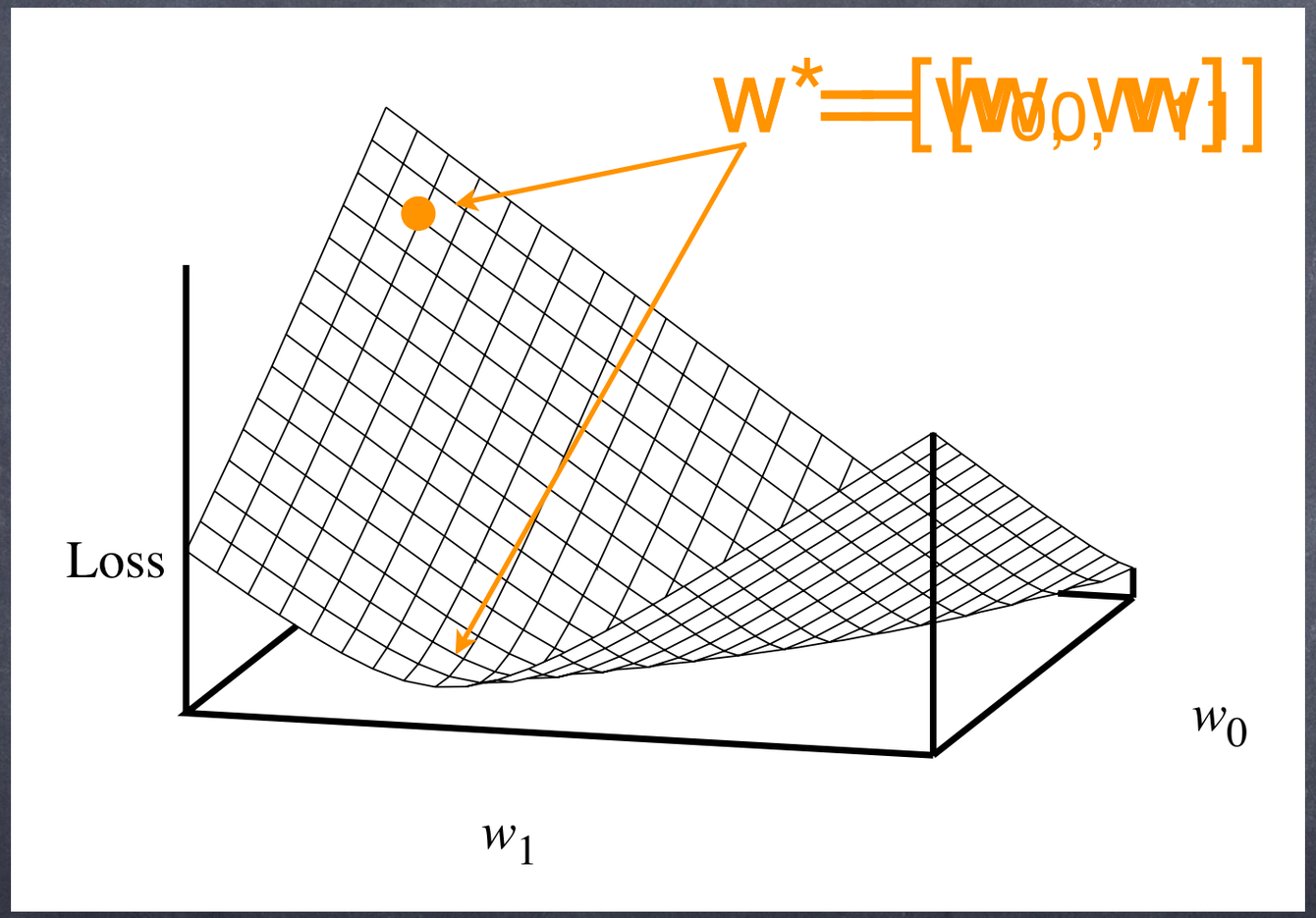
$$w_1 \leftarrow w_1 + \alpha \sum_j (y_j - h_{\mathbf{w}}(x_j)) \times x_j$$

Gradient Descent for Univar. Linear Regression

$$w_0 \leftarrow w_0 + \alpha \sum_j (y_j - h_{\mathbf{w}}(x_j))$$

$$w_1 \leftarrow w_1 + \alpha \sum_j (y_j - h_{\mathbf{w}}(x_j)) \times x_j$$

Gradient Descent In Weight Space



Batch Gradient Descent

- Use update rule over all training data every step
- Guaranteed to converge; but slow

Stochastic Gradient Descent

- Pick data points at random
- Apply single point update
- Not guaranteed; but much faster
  - Requires decreasing learning rate schedule (like simulated annealing)

Univariate Linear Regression

- Find $w = [w_0, w_1]$ that minimizes $L(h_w)$
- Closed form solution (in this case)
- In general: Gradient Descent
  - Update rule moves weights towards minimum loss over the training data
  - Batch vs. Stochastic

Multivariate Linear Regression

- Input x is not just a single value but a vector of n values $x = [x_1, x_2, \ldots, x_n]$
- Example:

  x_1 = weight of car

  x_2 = size of engine

  x_3 = color of car

  $f(x)$ = fuel economy of car

Multivariate Linear Regression

- Hypothesis space:

$$h_{\mathbf{w}}(x) = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_n x_n$$

$$= w_0 + \sum_i w_i x_i$$

$$\mathbf{w} = [w_0, w_1, w_2, \ldots, w_n]$$

$$\mathbf{x} = [1, x_1, x_2, \ldots, x_n]$$

$$h_{\mathbf{w}}(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} = \sum_i w_i x_i$$

Multivariate Linear Regression

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \sum_j L_2(y_j, \mathbf{w} \cdot \mathbf{x}_j)$$

Gradient Descent for Multivariate Linear Regression

$$w_i \leftarrow w_i + \alpha \sum_j x_{j,i}(y_j - h_{\mathbf{w}}(\mathbf{x}_j))$$

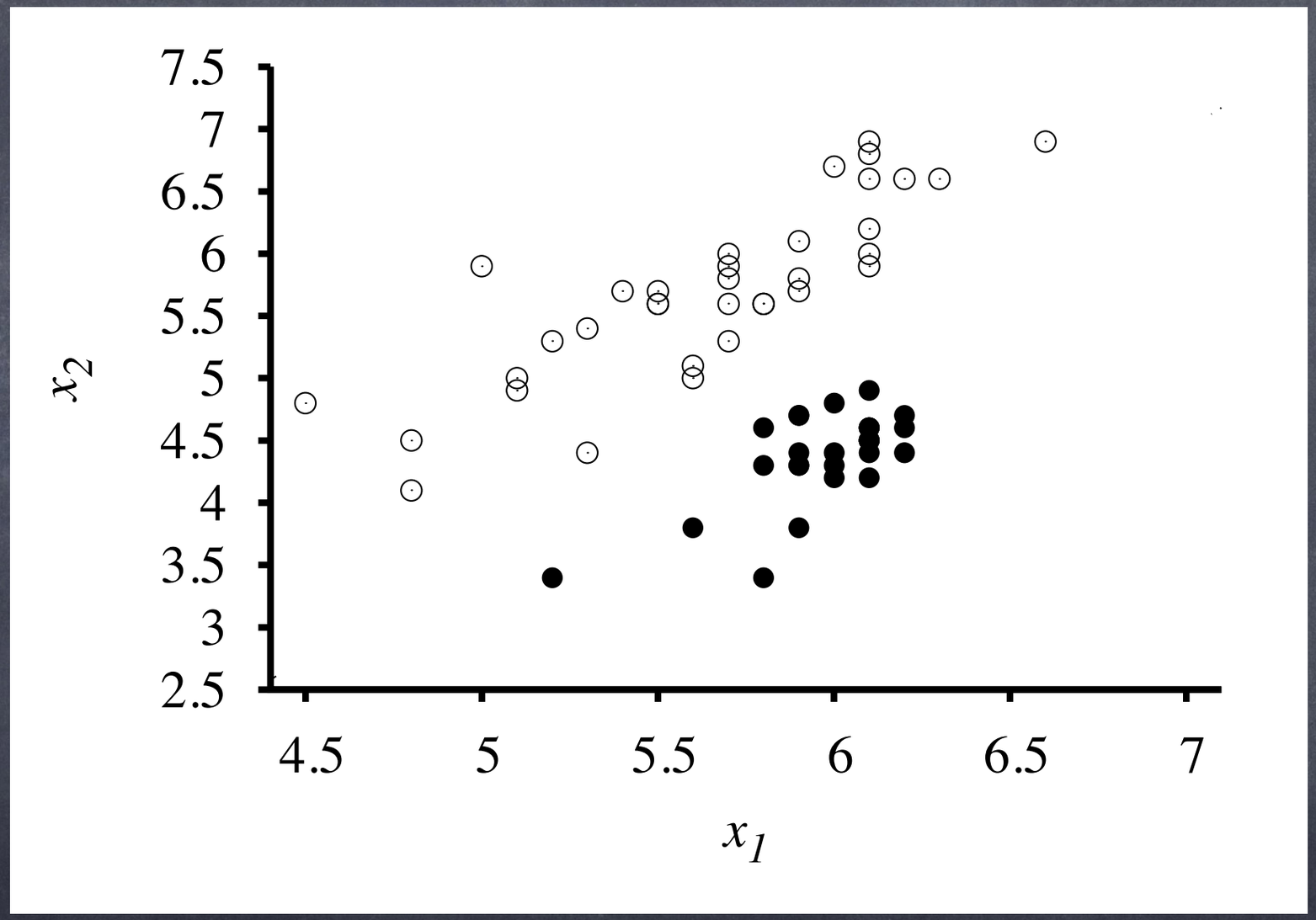
Linear Regression using Gradient Descent

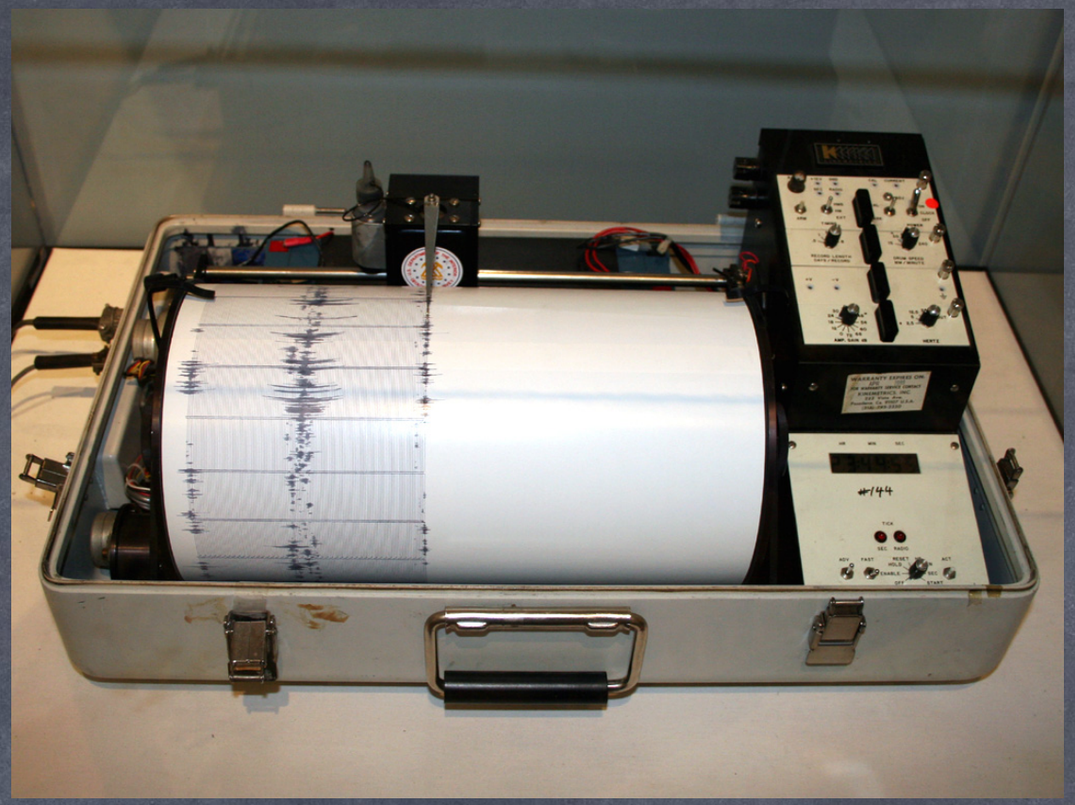
- Hypothesis space: $h_w(x) = w \cdot x$
- Goal: $\mathrm{w}^* = \underset{\mathrm{w}}{\operatorname{argmin}}\, L(h_{\mathrm{w}})$
- Gradient descent
  - Update rule:

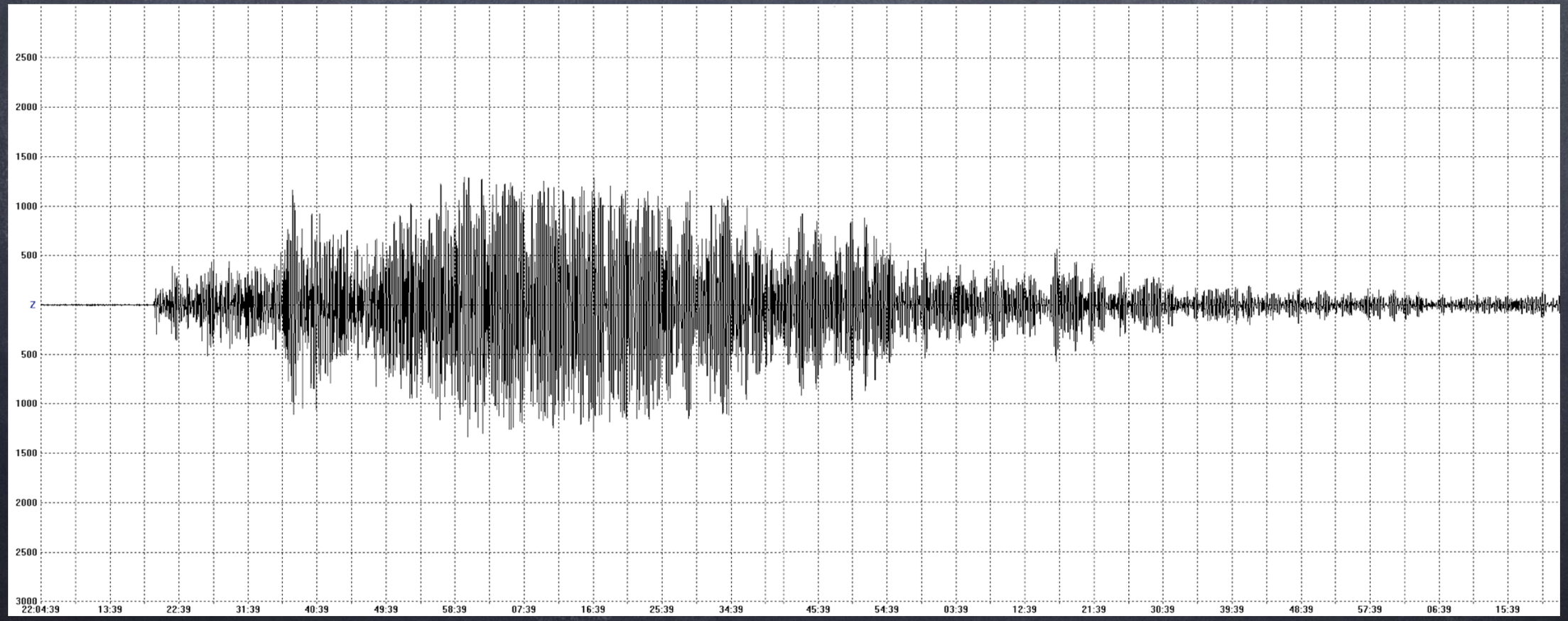
$$w_i \leftarrow w_i + \alpha \sum_j x_{j,i}(y_j - h_{\mathbf{w}}(\mathbf{x}_j))$$

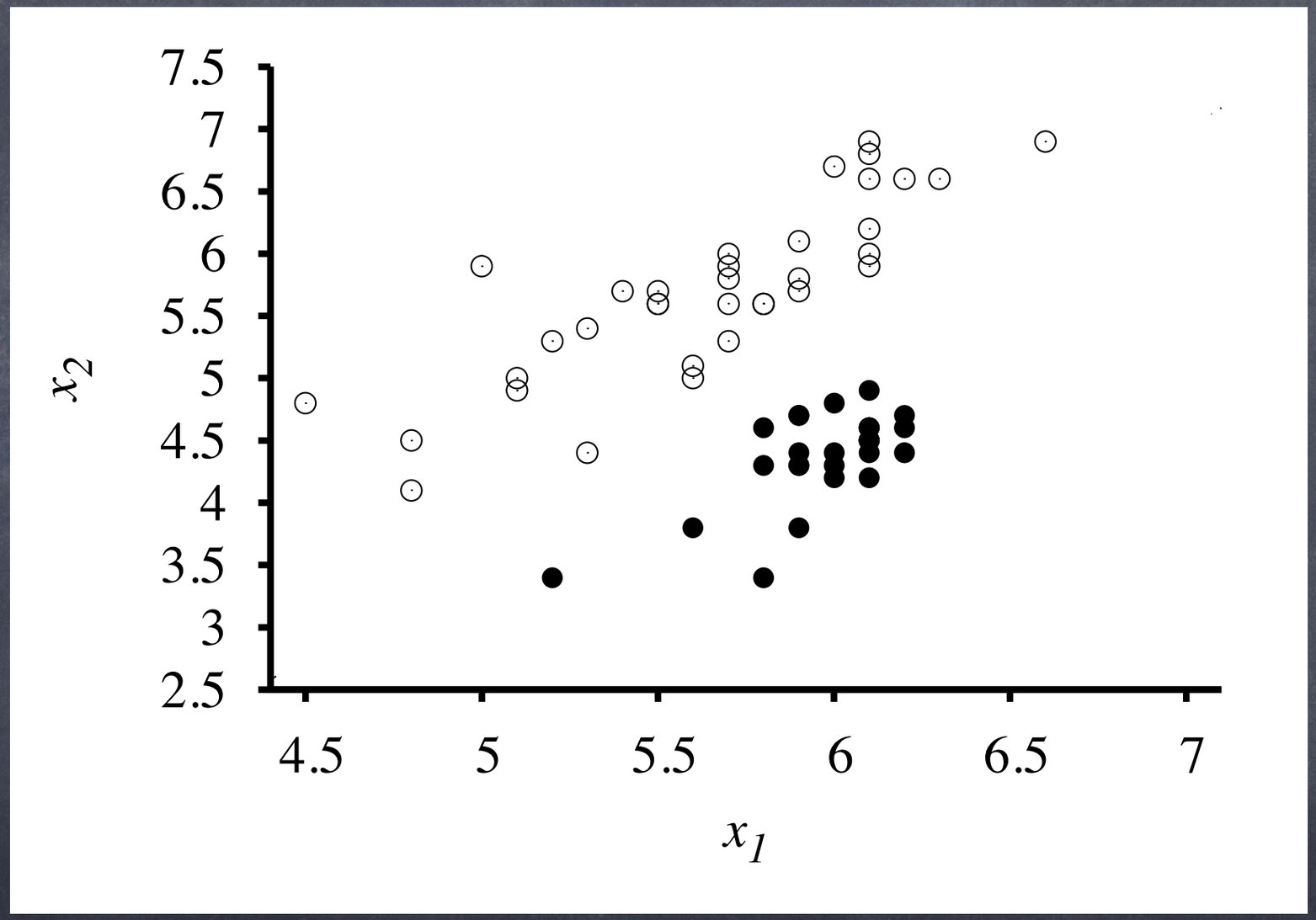
Linear Regression using Gradient Descent

- Learning (estimating) a function from examples: supervised learning
- Minimizing loss between hypothesis and actual values (typically L_2 loss)
- Solve using gradient descent (iterative, local search method) if no closed-form solution


7.5
7
6.5
6
5.5
5
4.5
4
3.5
3
2.5
x_2
4.5
5
5.5
6
6.5
7
x_1




2500
2000
1500
1000
500
Z
500
1000
1500
2000
2500
3000
22:04:39
13:39
22:39
31:39
40:39
49:39
58:39
07:39
16:39
25:39
34:39
45:39
54:39
03:39
12:39
21:39
30:39
39:39
48:39
57:39
06:39
15:39


7.5
7
6.5
6
5.5
5
4.5
4
3.5
3
2.5
x_2
4.5
5
5.5
6
6.5
7
x_1

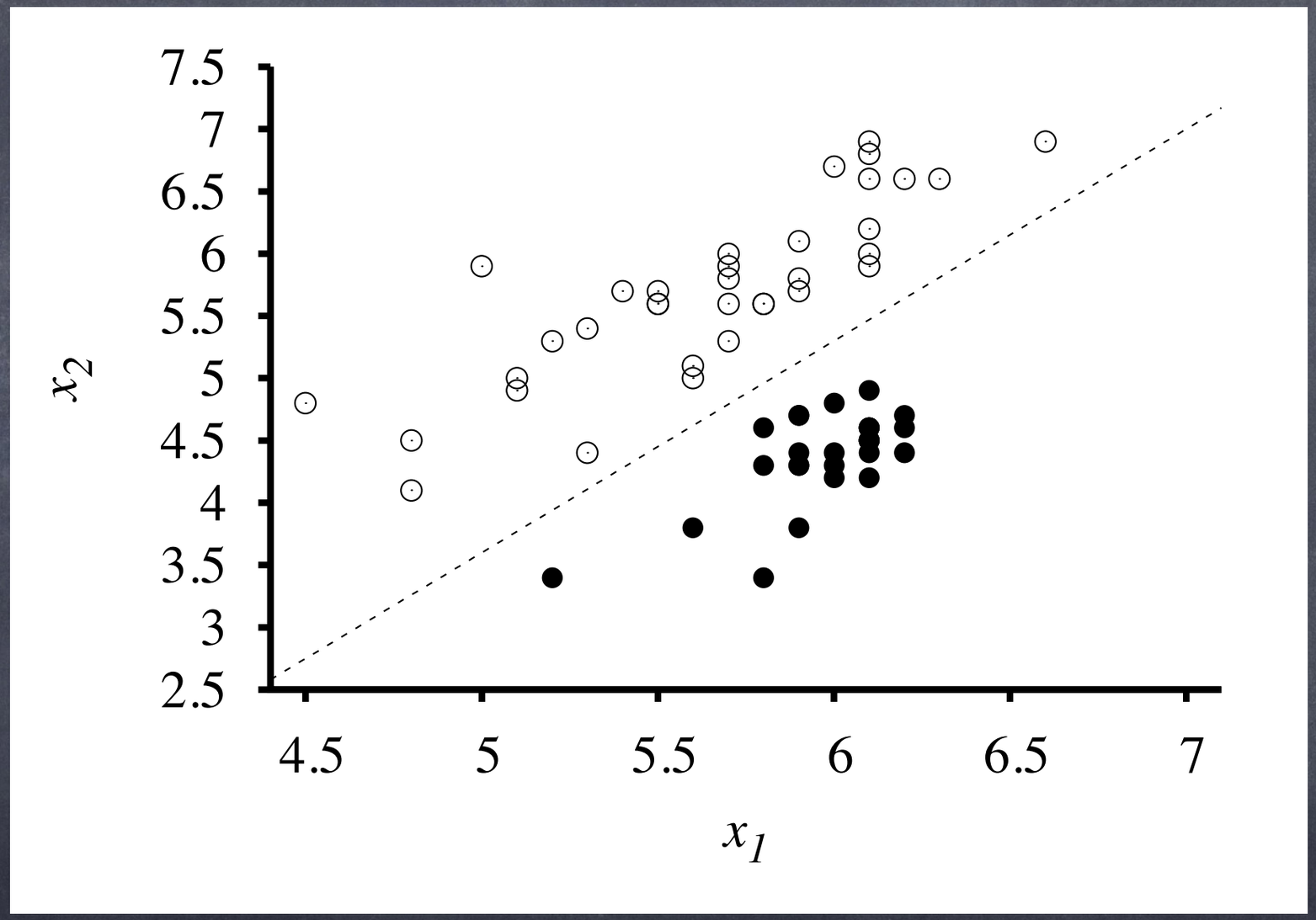
Classification

- Given training data $x = \langle x_1, x_2 \rangle$,
  Learn a hypothesis h such that:
  $h(x) = 0$ if x is from an earthquake
  $= 1$ if x is from an explosion

Decision Boundary

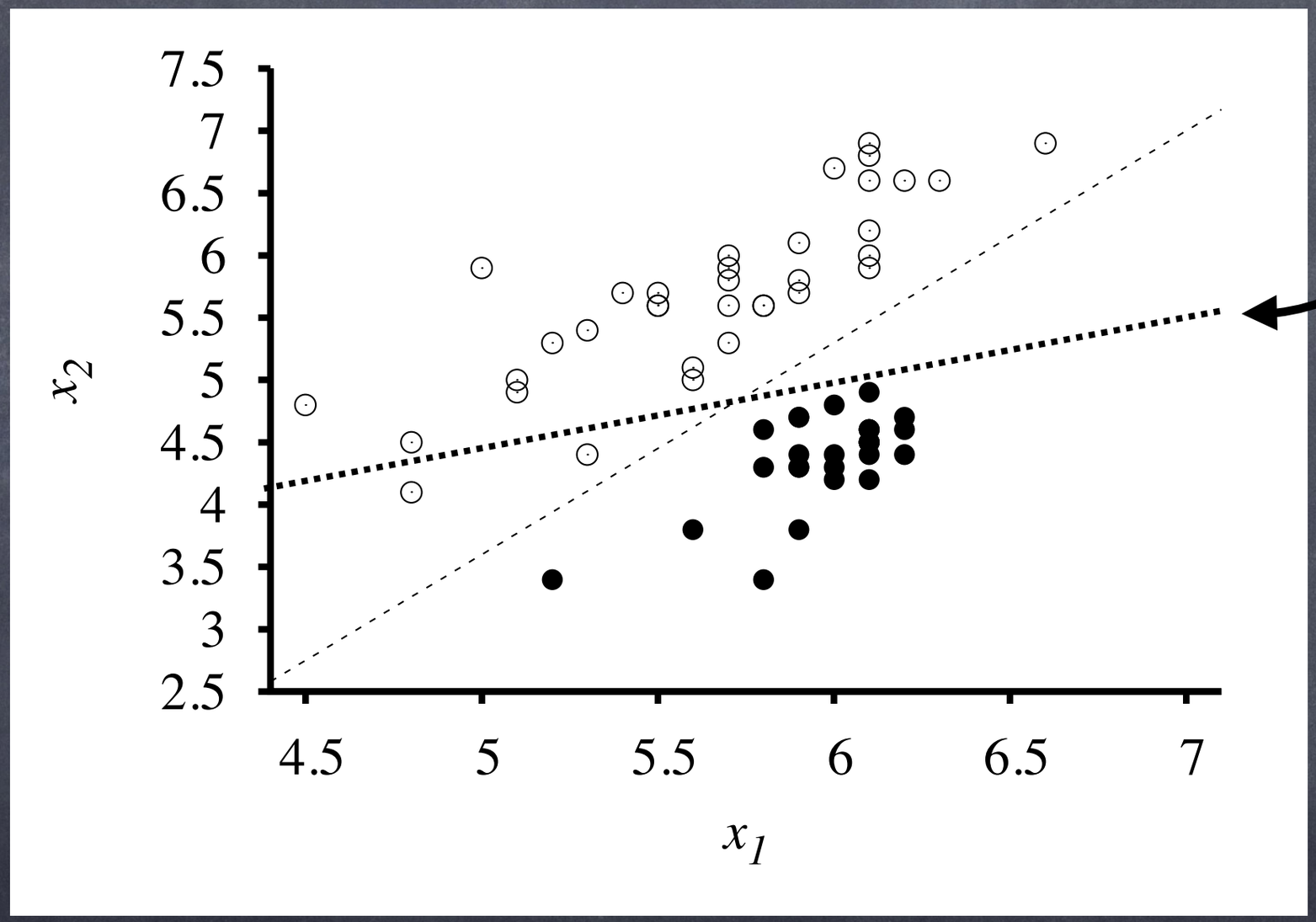
- Path (or surface in higher dimensions) that separates the two classes

$$h'(x) > 0 \text{ if } x \text{ is from an earthquake}$$
$$< 0 \text{ if } x \text{ is from an explosion}$$



$$x_2 = 1.7x_1 - 4.9$$

$x_2 = 0.52x_1 + 2.11$



$$x_2 = 1.7x_1 - 4.9$$

Linear Separator

- Decision boundary is a line in 2D, plane in 3D, ..., hyperplane in nD
- Data that admit a linear separator are said to be <u>linearly separable</u>

Linear Classifier

$$w_0 + w_1 x_1 + w_2 x_2 = 0$$

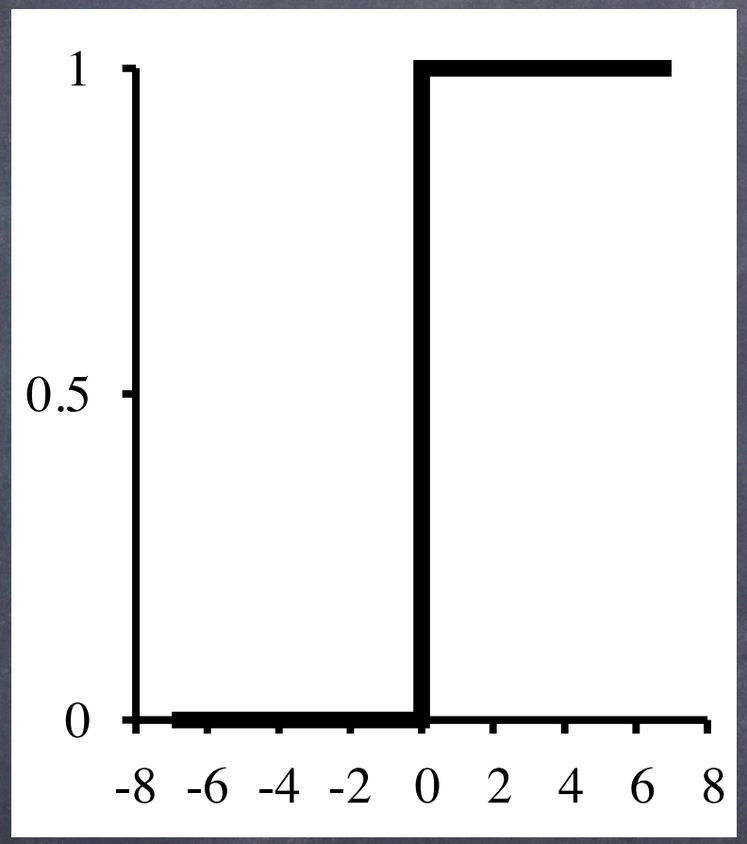
$$\mathbf{w} \cdot \mathbf{x} = 0$$

All instances of one class are above the line: $\mathbf{w} \cdot \mathbf{x} > 0$

All instances of one class are below the line: $\mathbf{w} \cdot \mathbf{x} < 0$

$$h_{\mathbf{w}}(\mathbf{x}) = Threshold(\mathbf{w} \cdot \mathbf{x})$$

Hard Threshold



$$Threshold(z) = 1 \text{ if } z \geq 0$$
$$= 0 \text{ otherwise}$$

Linear Classifier

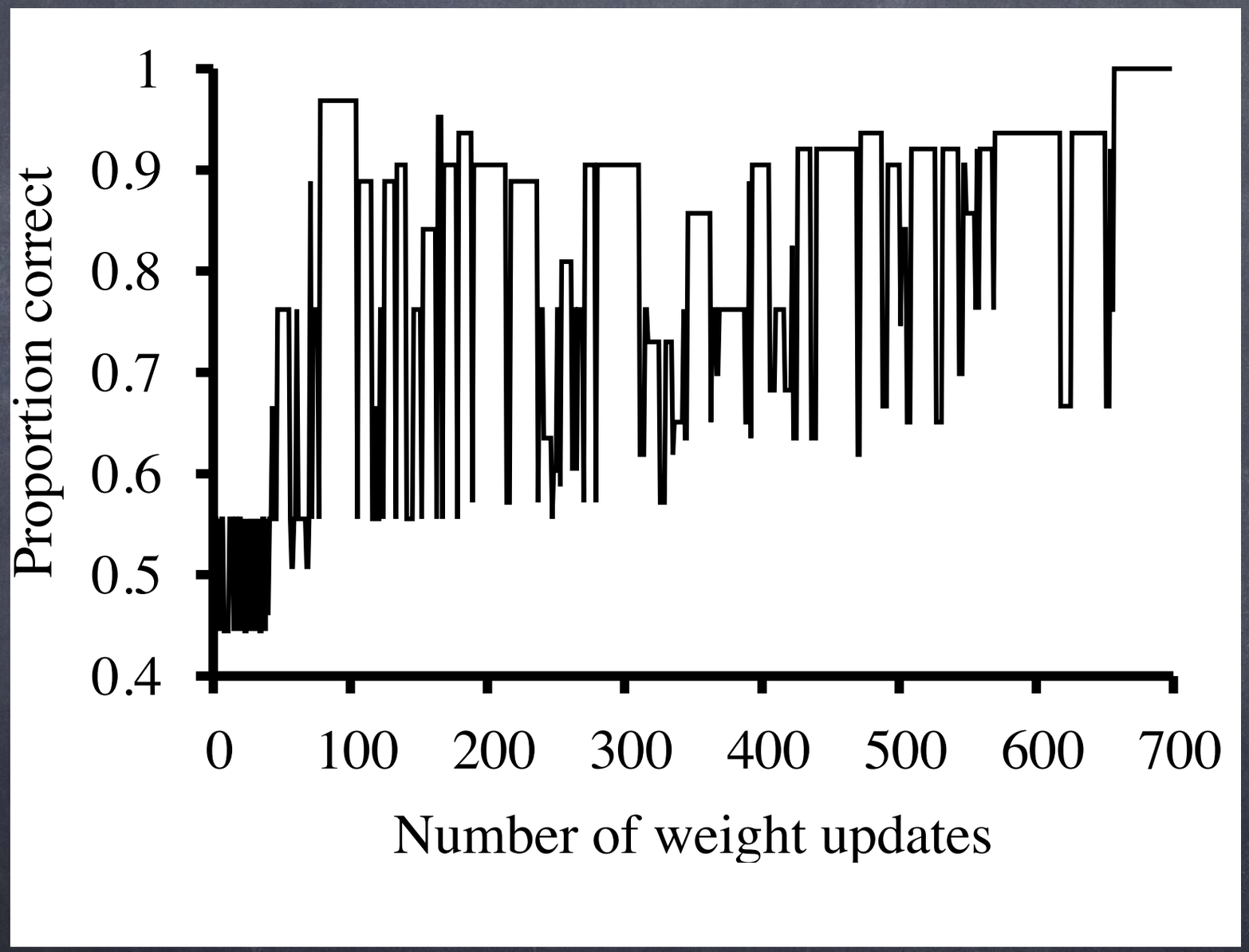
$$h_{\mathbf{w}}(\mathbf{x}) = \mathit{Threshold}(\mathbf{w} \cdot \mathbf{x})$$

$$\mathrm{w}^* = \underset{\mathrm{w}}{\operatorname{argmin}}\, L(h_{\mathrm{w}})$$

Perceptron Learning Rule

$$w_i \leftarrow w_i + \alpha(y - h_{\mathbf{w}}(\mathbf{x})) \times x_i$$

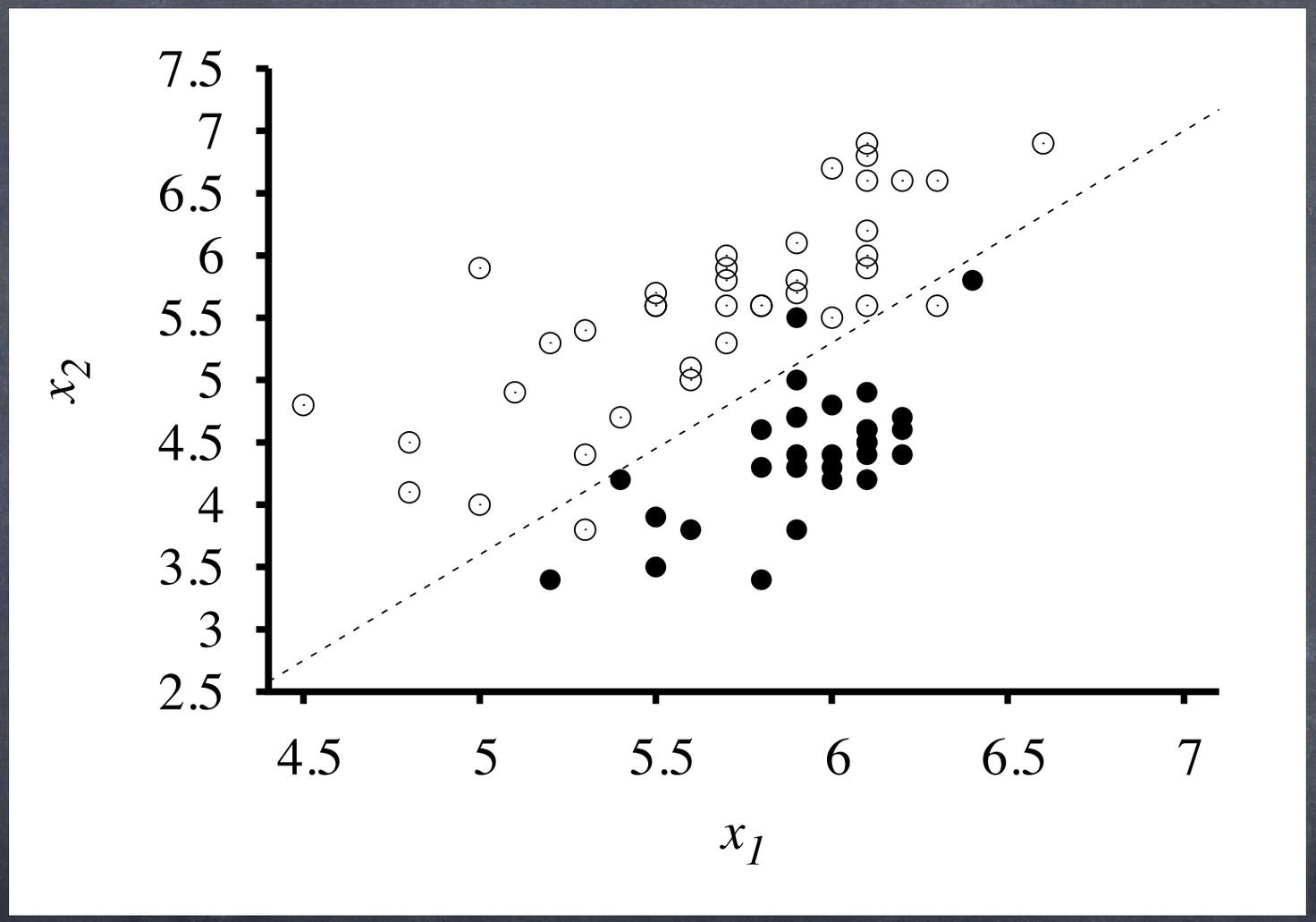
Learning Curve

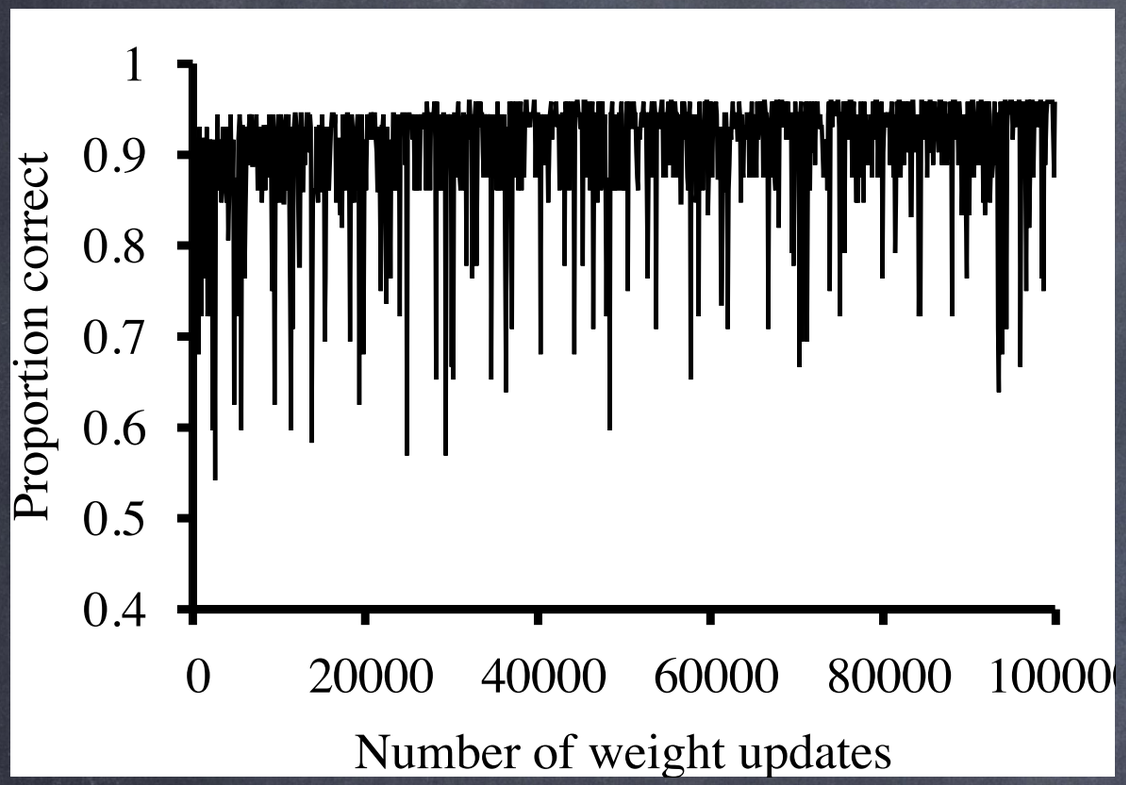


N=63

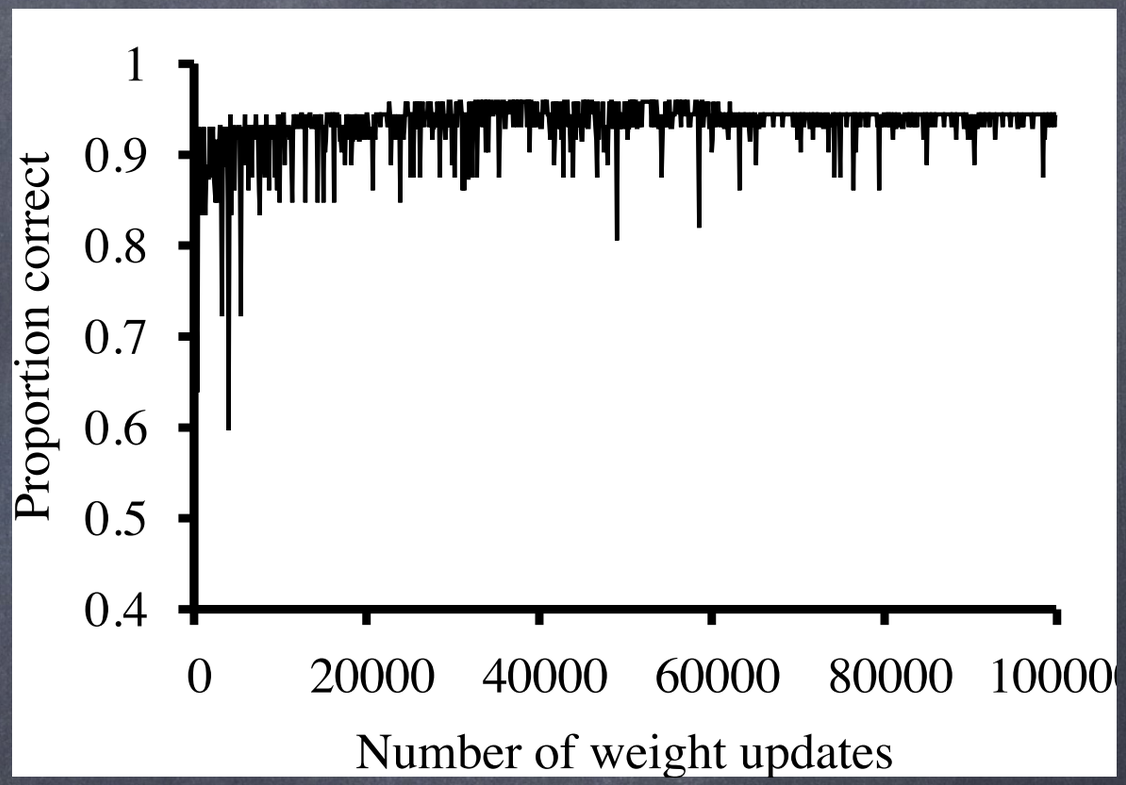
Linear Classifier with Hard Threshold

- Learn with gradient descent
  - Perceptron learning rule just like linear regression
- Convergence "isn't pretty but it works"

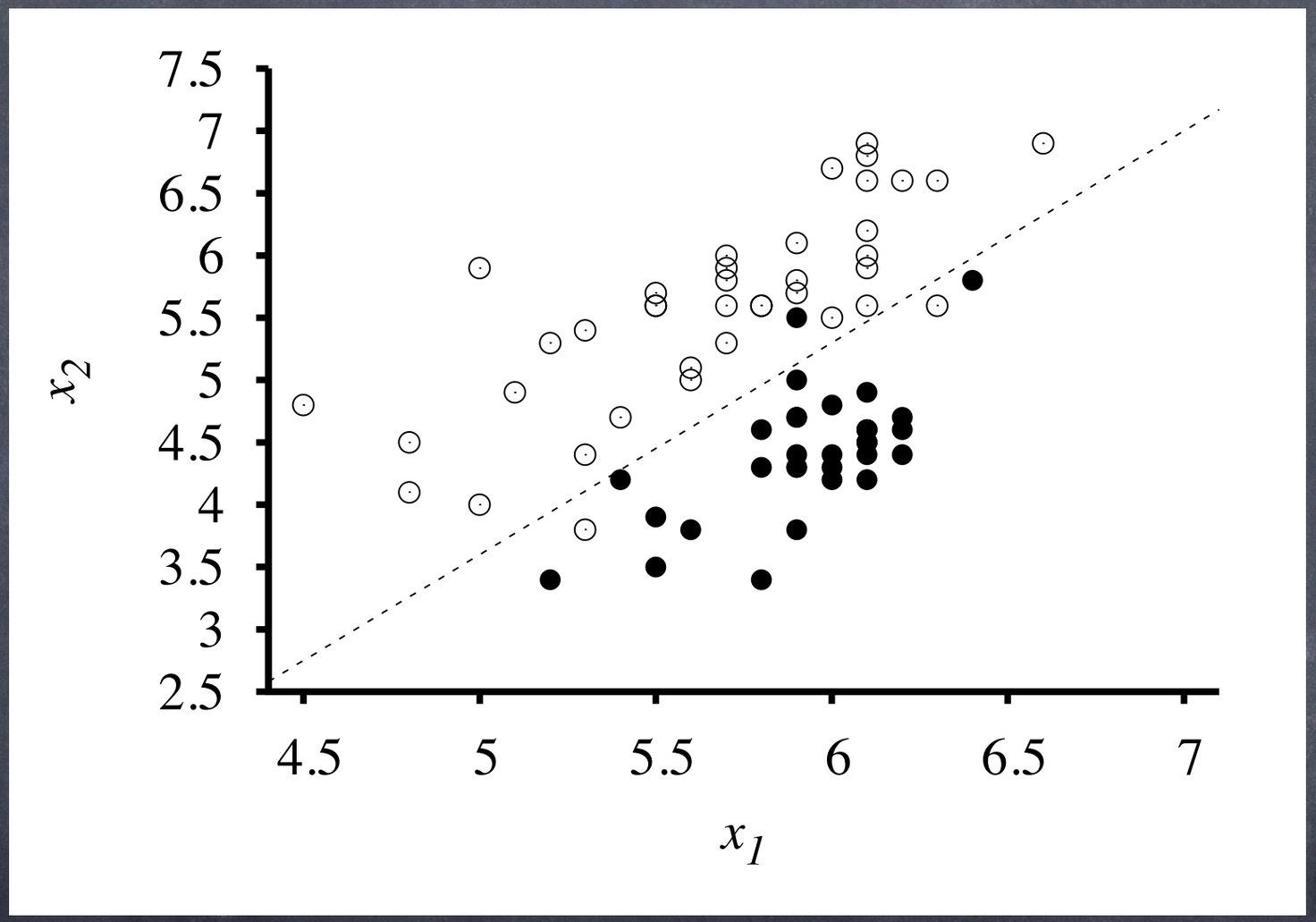

7.5
7
6.5
6
5.5
5
4.5
4
3.5
3
2.5
x_2
4.5
5
5.5
6
6.5
7
x_1



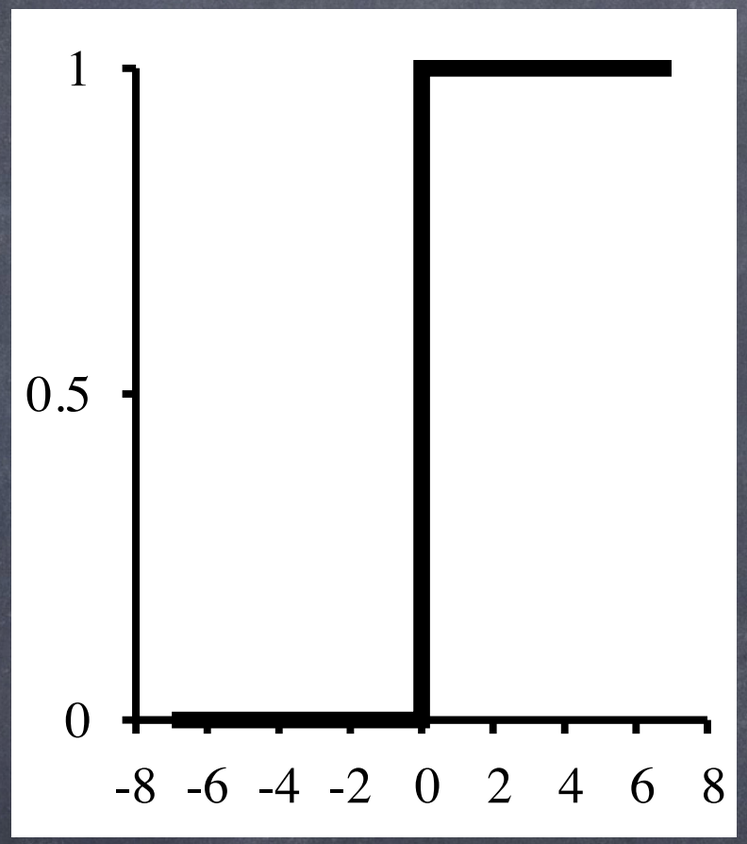
Fixed α



α decays as O(1/t)

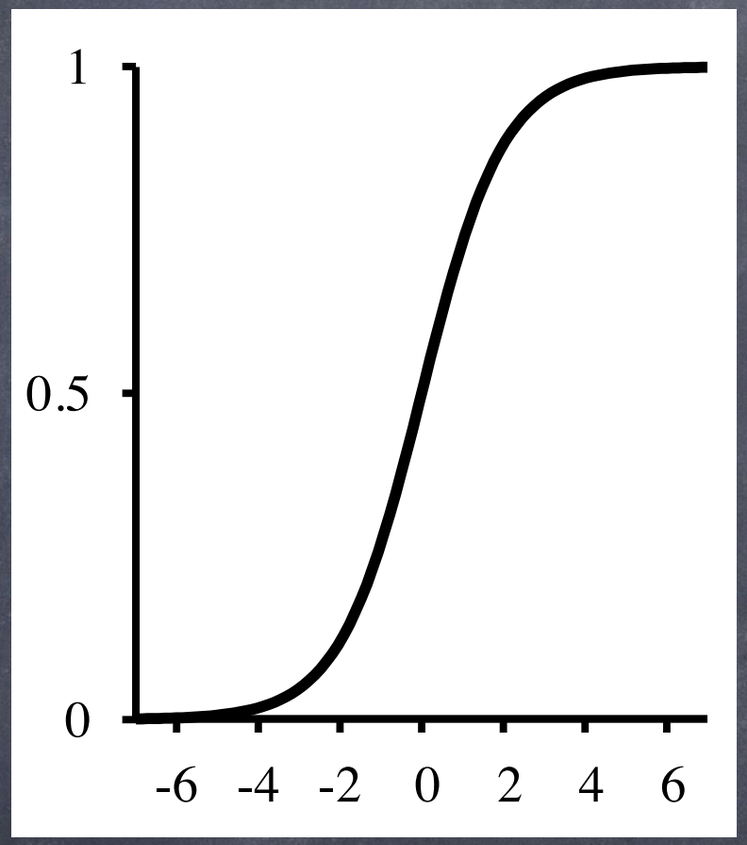

7.5
7
6.5
6
5.5
5
4.5
4
3.5
3
2.5
x_2
4.5
5
5.5
6
6.5
7
x_1

Hard Threshold



$$Threshold(z) = 1 \text{ if } z \geq 0$$
$$= 0 \text{ otherwise}$$

Soft Threshold

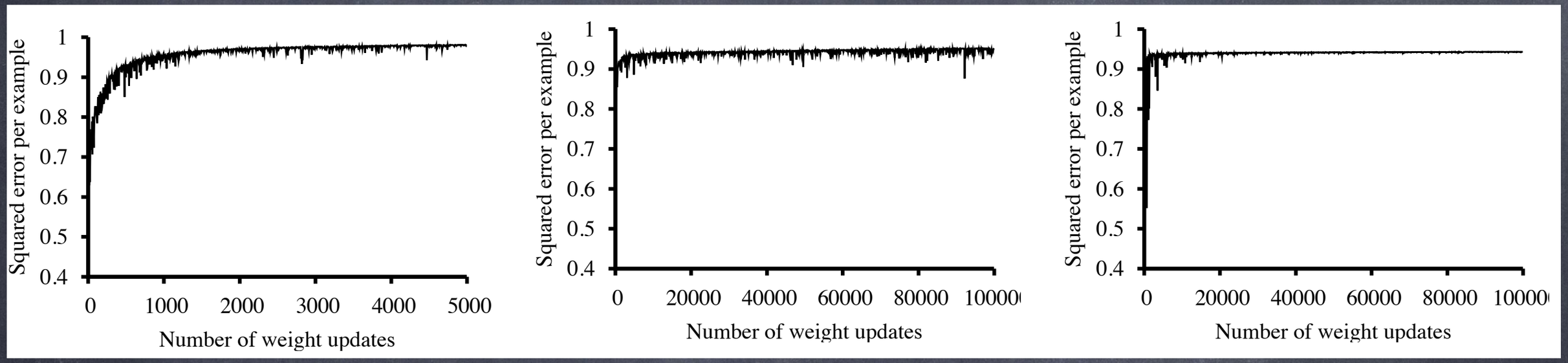


$$Logistic(z) = \frac{1}{1 + e^{-z}}$$

Logistic Regression

$$h_{\mathbf{w}}(\mathbf{x}) = \mathit{Logistic}(\mathbf{w} \cdot \mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w} \cdot \mathbf{x}}}$$

$$\mathrm{w}^* = \underset{\mathrm{w}}{\operatorname{argmin}}\, L(h_{\mathrm{w}})$$


Squared error per example
1
0.9
0.8
0.7
0.6
0.5
0.4
0
1000
2000
3000
4000
5000
Number of weight updates
Squared error per example
1
0.9
0.8
0.7
0.6
0.5
0.4
0
20000
40000
60000
80000
Number of weight updates
Squared error per example
1
0.9
0.8
0.7
0.6
0.5
0.4
0
20000
40000
60000
80000
Number of weight updates

Learning Linear Classifiers

- Can learn linear classifiers from data
  - Similar to linear regression
- Hard threshold
  - Perceptron learning rule
  - Unpredictable, not robust to noise
- Soft (sigmoid) threshold: logistic regression
  - Slower, but more predictable
  - Robust to noisy data

For Next Time:

AIMA 18.7