CSC242: Introduction to Artificial Intelligence

Lecture 4.2

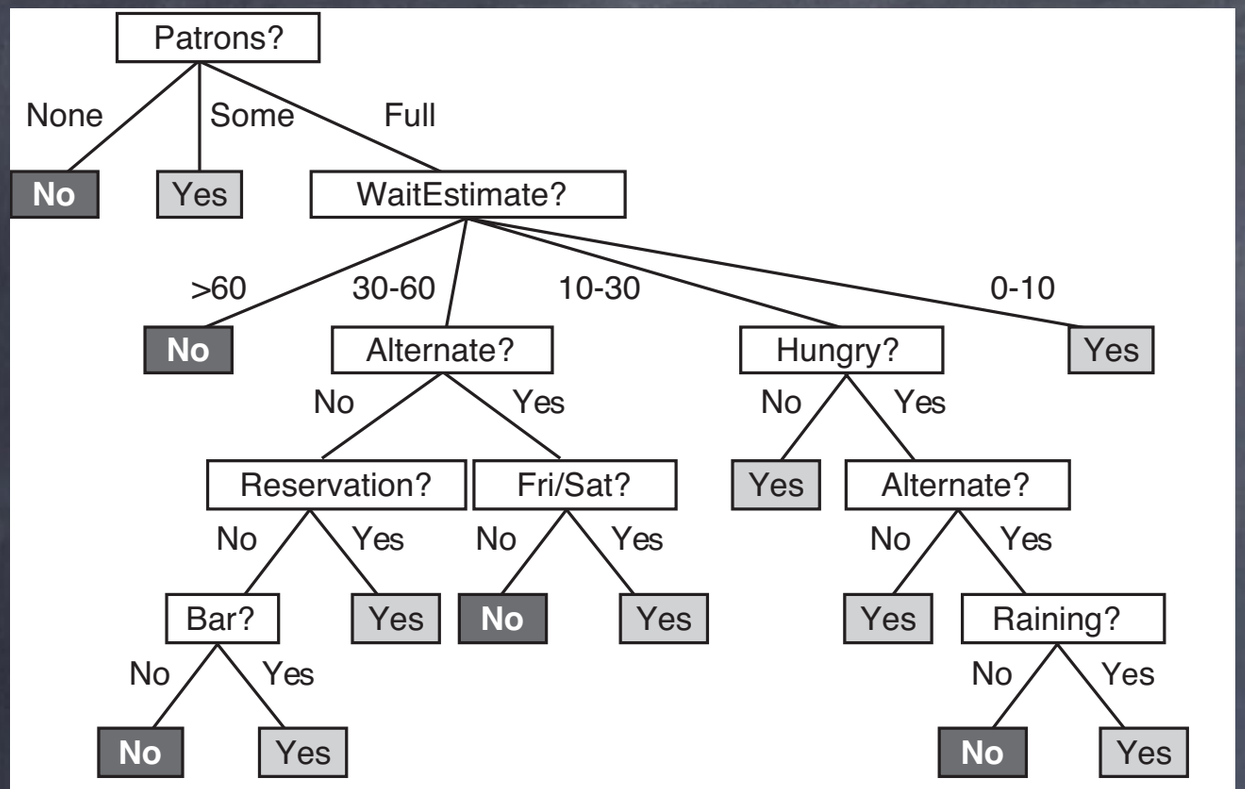
Please put away all electronic devices

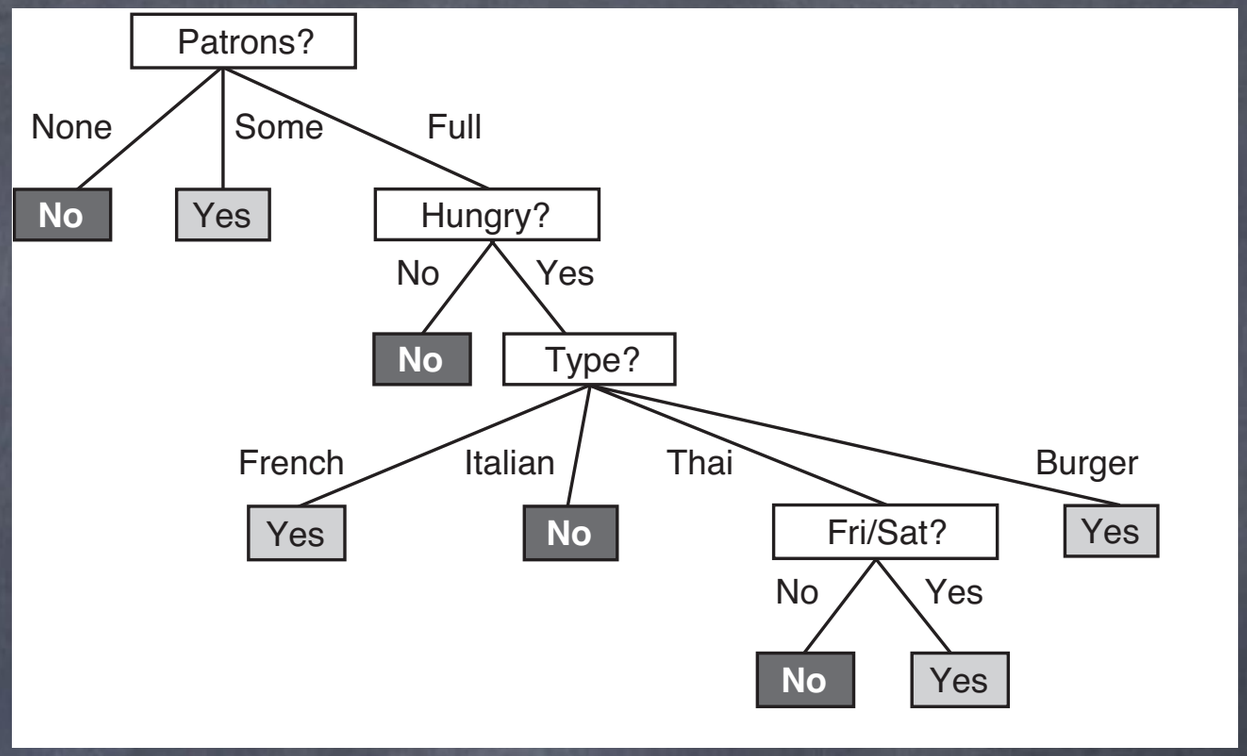
Learning

- Why?
  - Knowledge without programming
- What?
  - Probabilities, distributions, rules, ...
- How?
  - Unsupervised ↔ 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

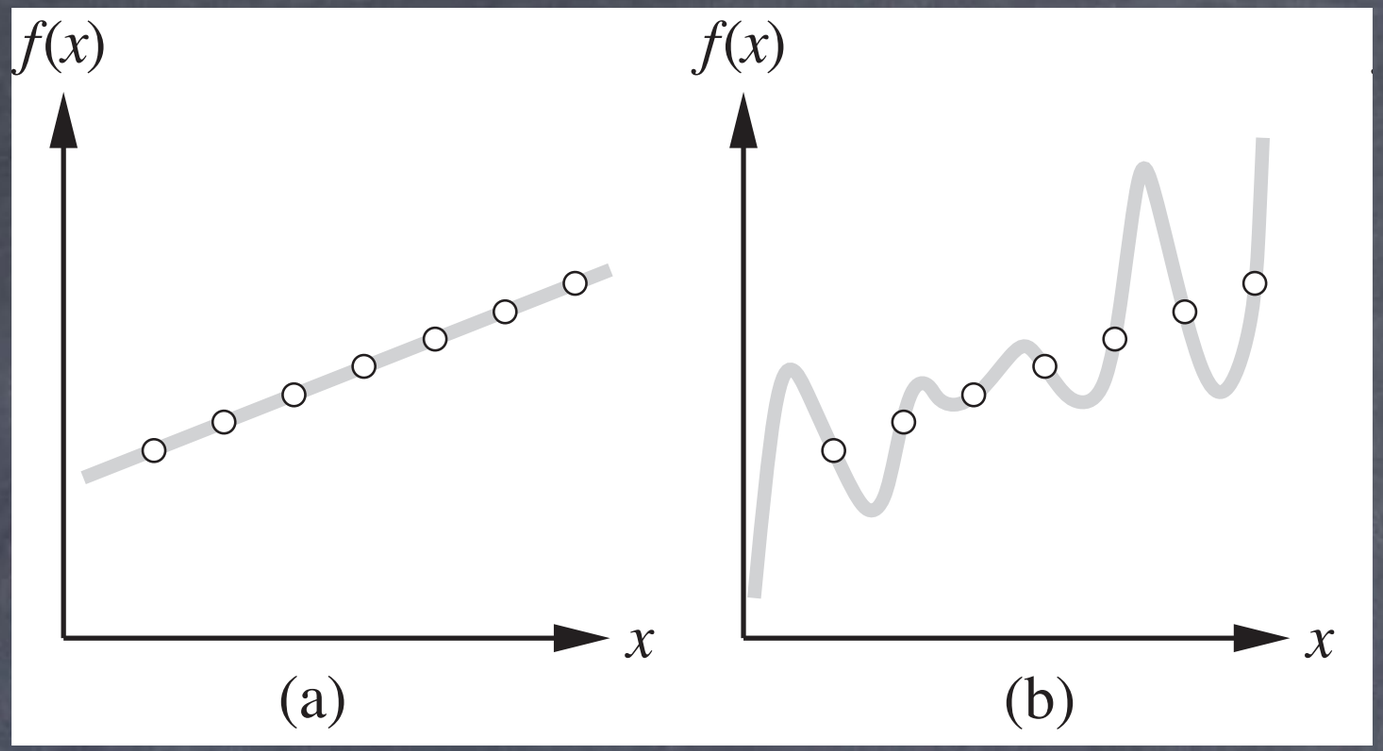
- Given a training set of N example input-output pairs:

$$(\mathbf{x}_1,\ y_1),\ (\mathbf{x}_2,\ y_2),\ \ldots,\ (\mathbf{x}_\mathrm{N},\ y_\mathrm{N})$$

  where each $y_j = f(\mathbf{x}_j)$
- Discover function h that approximates f
- Search through the space of possible hypotheses for one that will perform well

Hypothesis Space

- Decision trees (Boolean formulas)
- Linear functions $y = mx + b$
- Polynomials (of some degree)
- Java programs?
- Turing machines?


f(x)
x
(a)
f(x)
x
(b)
f(x)
x
(c)
f(x)
x
(d)

Evaluating Hypotheses

- Accuracy: fits the data
  - Test on some examples
  - Generalization: predicts outputs for unseen inputs
    - Test with cross-validation
- Simplicity and "searchability"

Accuracy

- Training set: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)$
- Test accuracy of h by comparing $h(\mathbf{x}_j)$ to y_j

Generalization

- A hypothesis (function) generalizes well if it correctly predicts the value of y for novel examples $\mathbf{x}$

Cross-Validation

- Randomly split data into training and testing (in some proportion)
  - Hold out test data during training
  - Doesn't use all data for training

k-Fold Cross-Validation

- Divide data into k equal subsets
- Perform k rounds of learning
  - Leave out 1 subset ($1/k$ of the data) each round; use for testing that round
- Average test scores over k rounds

Evaluating Hypotheses

- Accuracy: fits the data
- Generalization: predicts outputs for unseen inputs
- Simplicity and "searchability"

Simplicity

- Hypothesis space easier to search
- Less likely to memorize the data
- Easier to compute
  - During learning
  - During decision-making

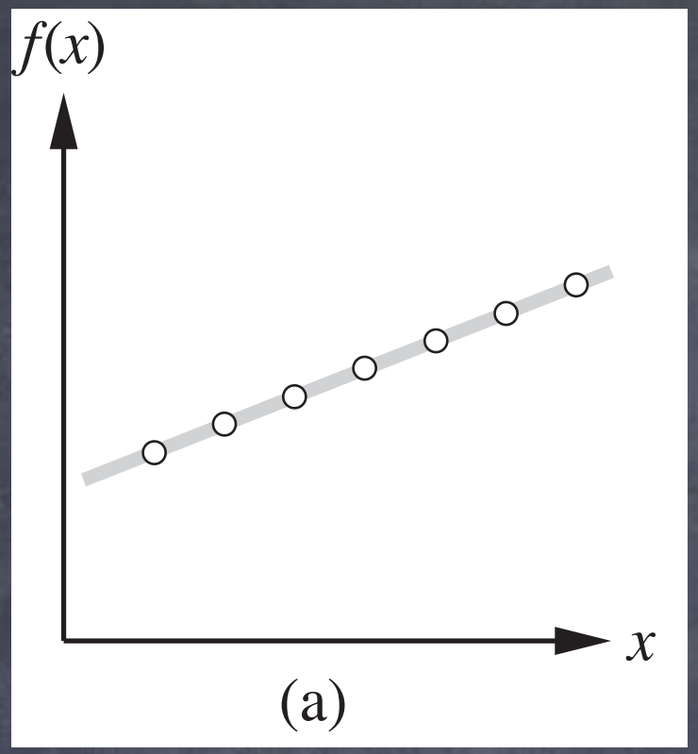
Occam's Razor

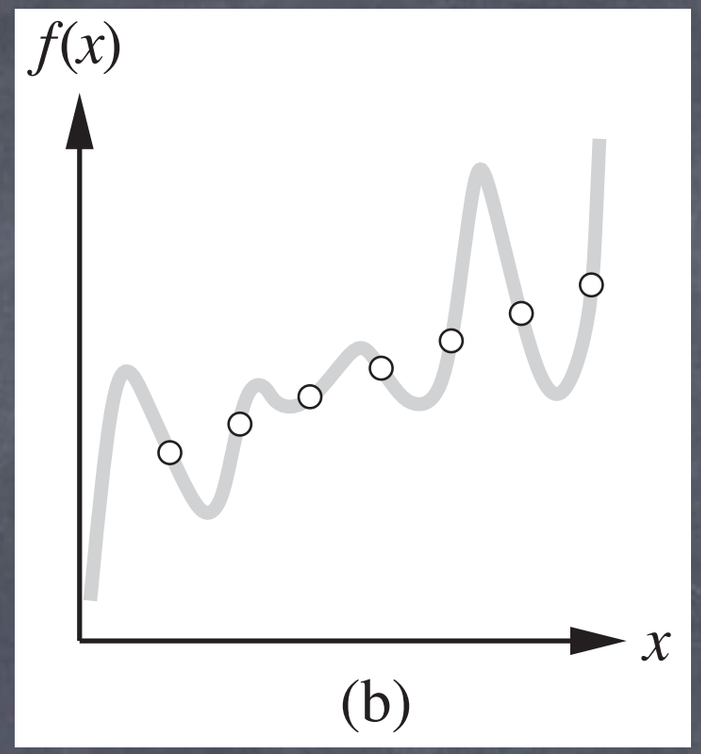
William of Occam (or Ockham)

14th c.

Evaluating Hypotheses

- Accuracy: fits the data
- Generalization: predicts outputs for unseen inputs
- Simplicity and "searchability"





Hypothesis space:

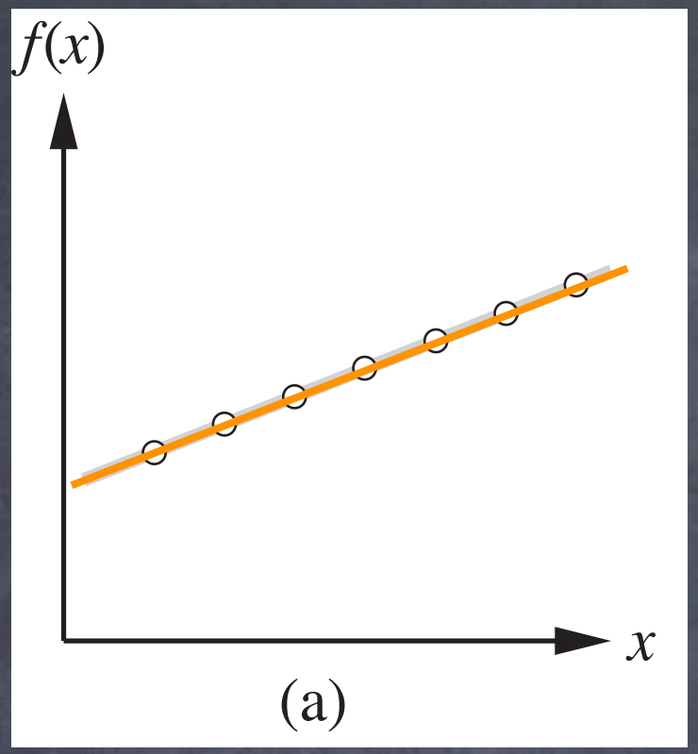
$$y = mx+b$$

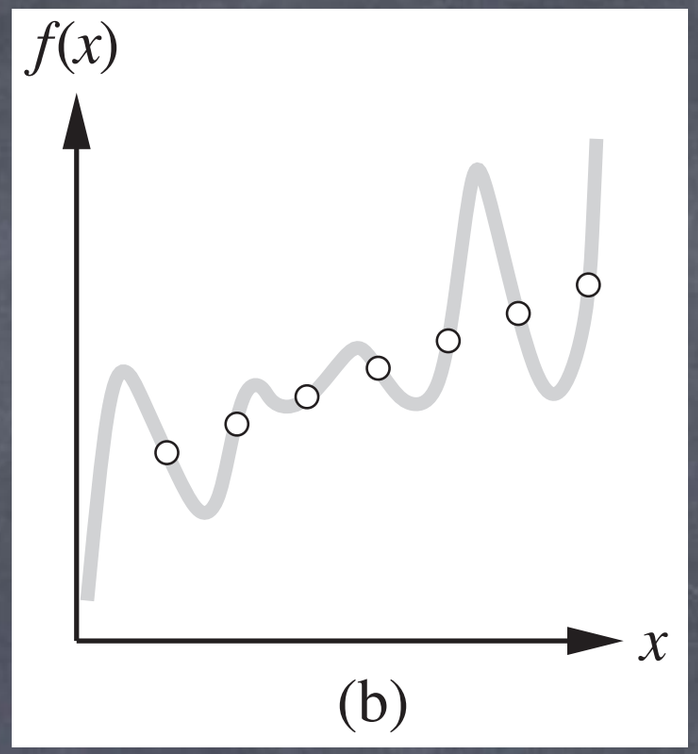
$$y = c_7x^7 + c_6x^6 + \ldots + c_1x + c_0$$

$$= \sum_{i=0}^{7} c_i x^i$$

Hypothesis:

$$y = -0.4x+3$$





Hypothesis space:

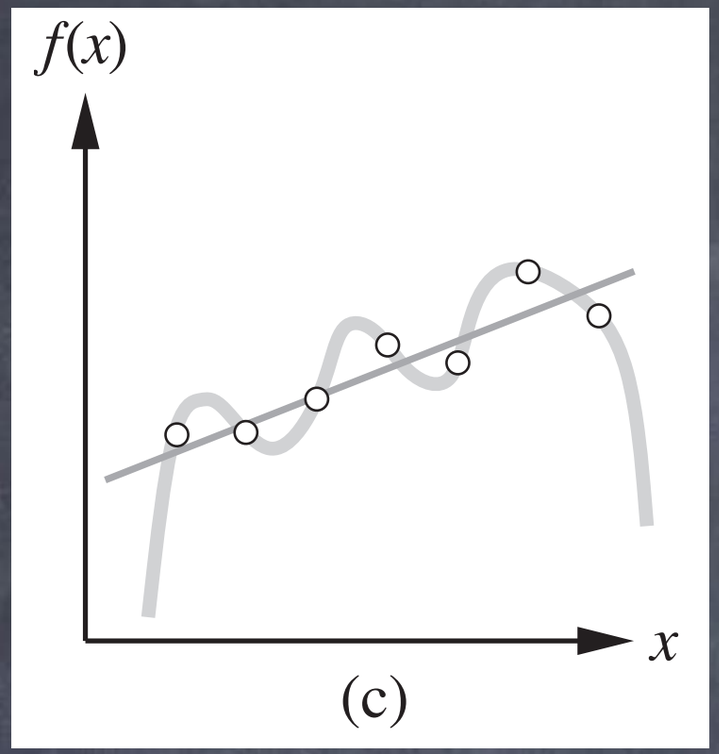
$$y = mx+b$$

$$y = c_7x^7 + c_6x^6 + \ldots + c_1x + c_0$$

$$= \sum_{i=0}^{7} c_i x^i$$

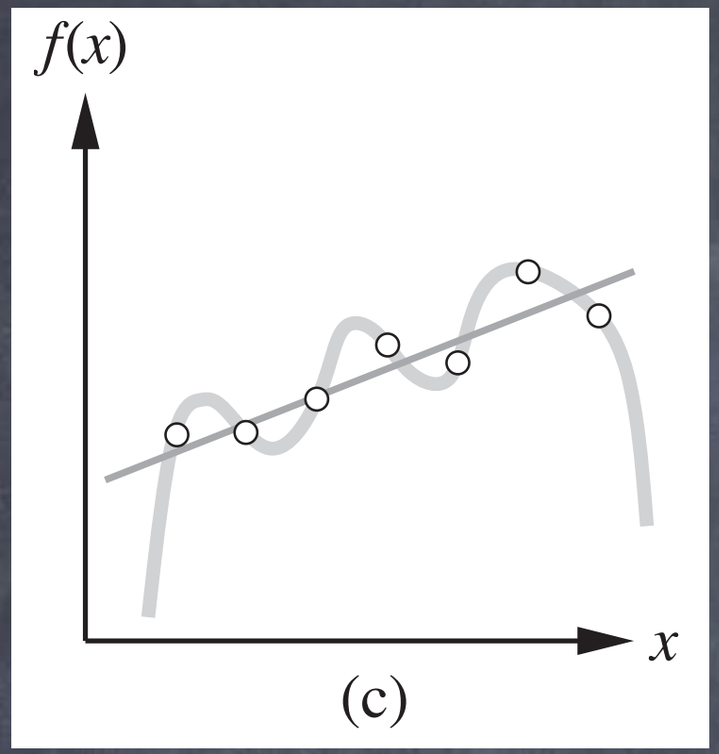
Hypothesis:

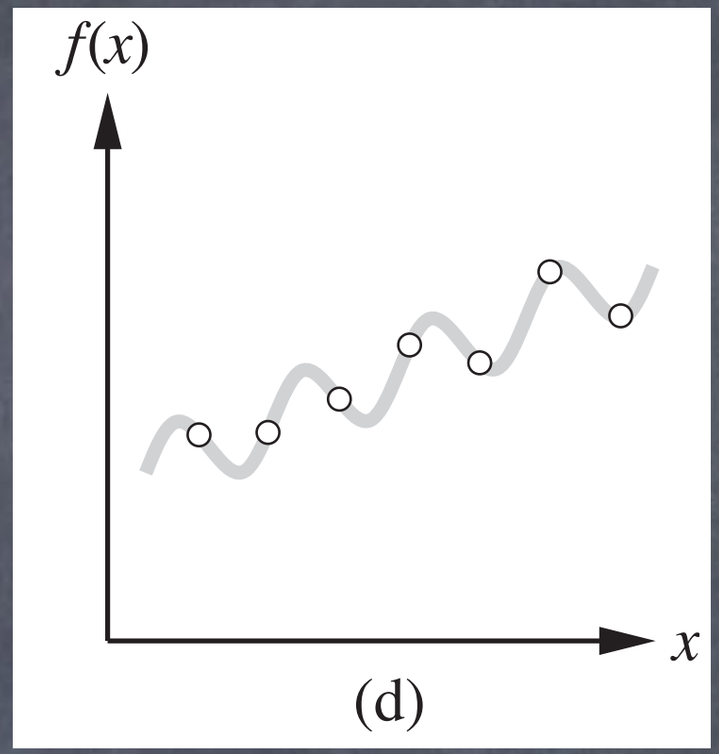
$$y = -0.4x+3$$



$$y = c_6x^6 + c_5x^5 \ldots + c_1x + c_0$$

$$y = mx + b$$





$y = c_6x^6 + c_5x^5 \ldots + c_1x + c_0$

$ax + b + c\sin(x)$

Evaluating Hypotheses

- Accuracy: fits the data
- Generalization: predicts outputs for unseen inputs
- Simplicity and "searchability"

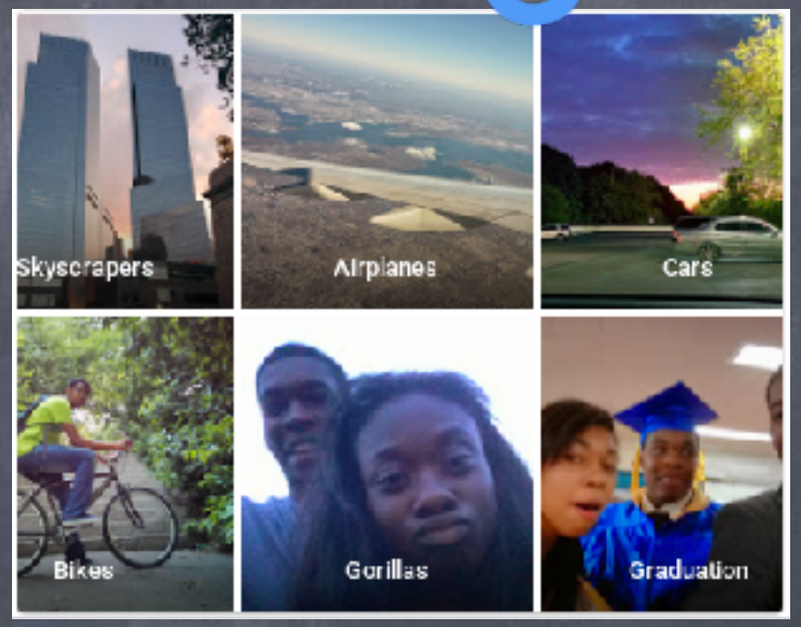
When Learning Goes Wrong

theguardian

Google says sorry for racist auto-tag in photo app

- Google Photos labelled a picture of two black people as ‘gorillas’
- Google Maps and Flickr have also suffered from race-related problems

Google



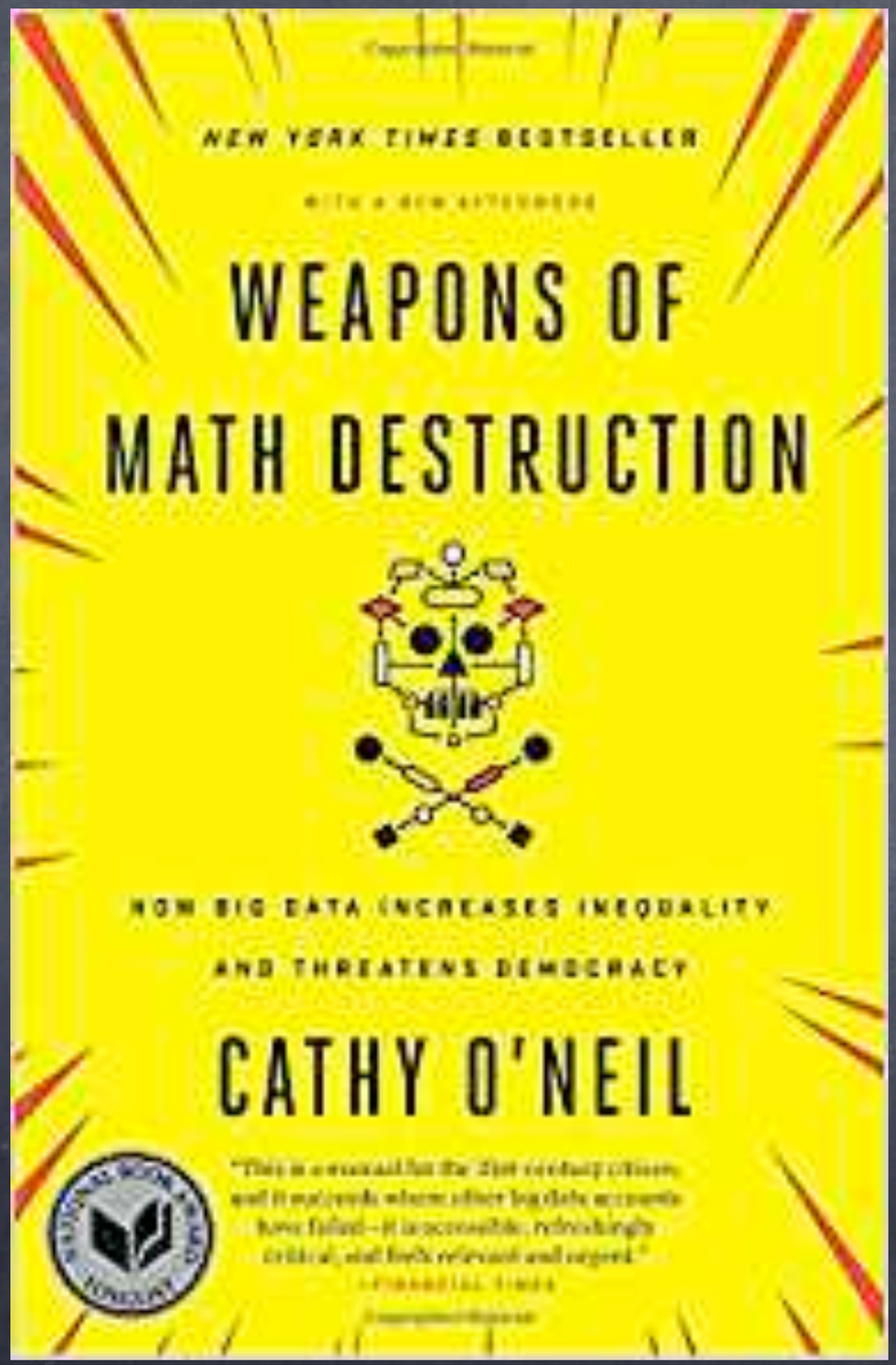
https://www.theguardian.com/technology/2015/jul/01/google-sorry-racist-auto-tag-photo-app

The Telegraph

Microsoft

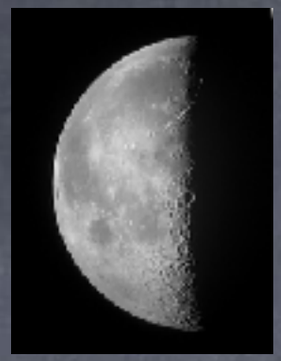
Microsoft deletes 'teen girl' AI after it became a Hitler-loving sex robot within 24 hours

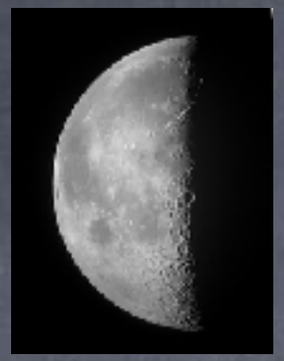
www.telegraph.co.uk/technology/2016/03/24/microsofts-teen-girl-ai-turns-into-a-hitler-loving-sex-robot-wit/


NEW YORK TIMES BESTSELLER
WEAPONS OF
MATH DESTRUCTION
HOW BIG DATA INCREASES INEQUALITY
AND THREATENS DEMOCRACY
CATHY O'NEIL

When Learning Goes Wrong

$$P(Die{=}i) = 1/6 \text{ for } i \text{ in } \{\ 1,\ ...,\ 6\ \}$$

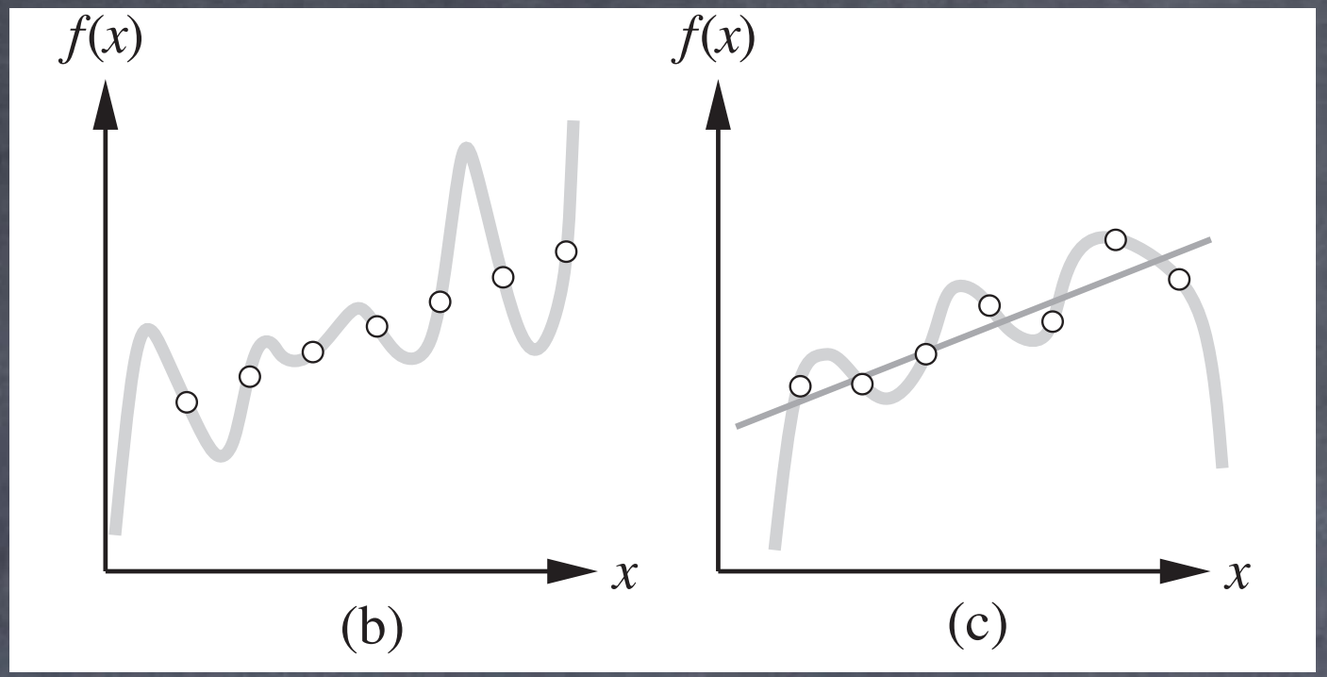




red	3:25	full	yes	6
red	17:31	3/4	yes	3
blue	21:09	full	no	6
red	11:15	1/8	no	1
green	14:28	1/4	yes	2

Overfitting

- A learned model adjusts to random features of the input that have no causal relation to the target function
- Usually happens when a model has too many parameters relative to the number of training examples

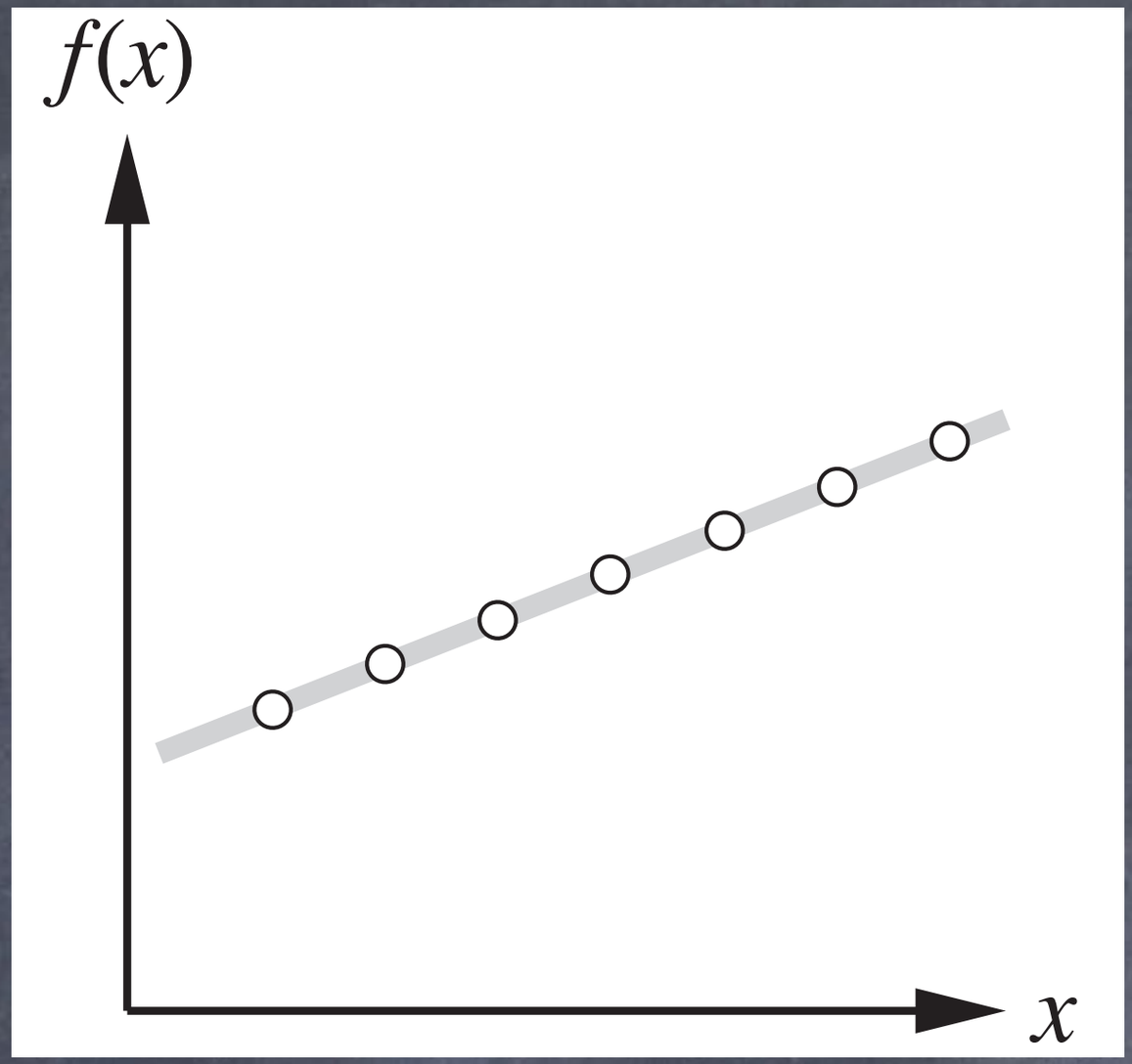

f(x)
x
(b)
f(x)
x
(c)

Overfitting

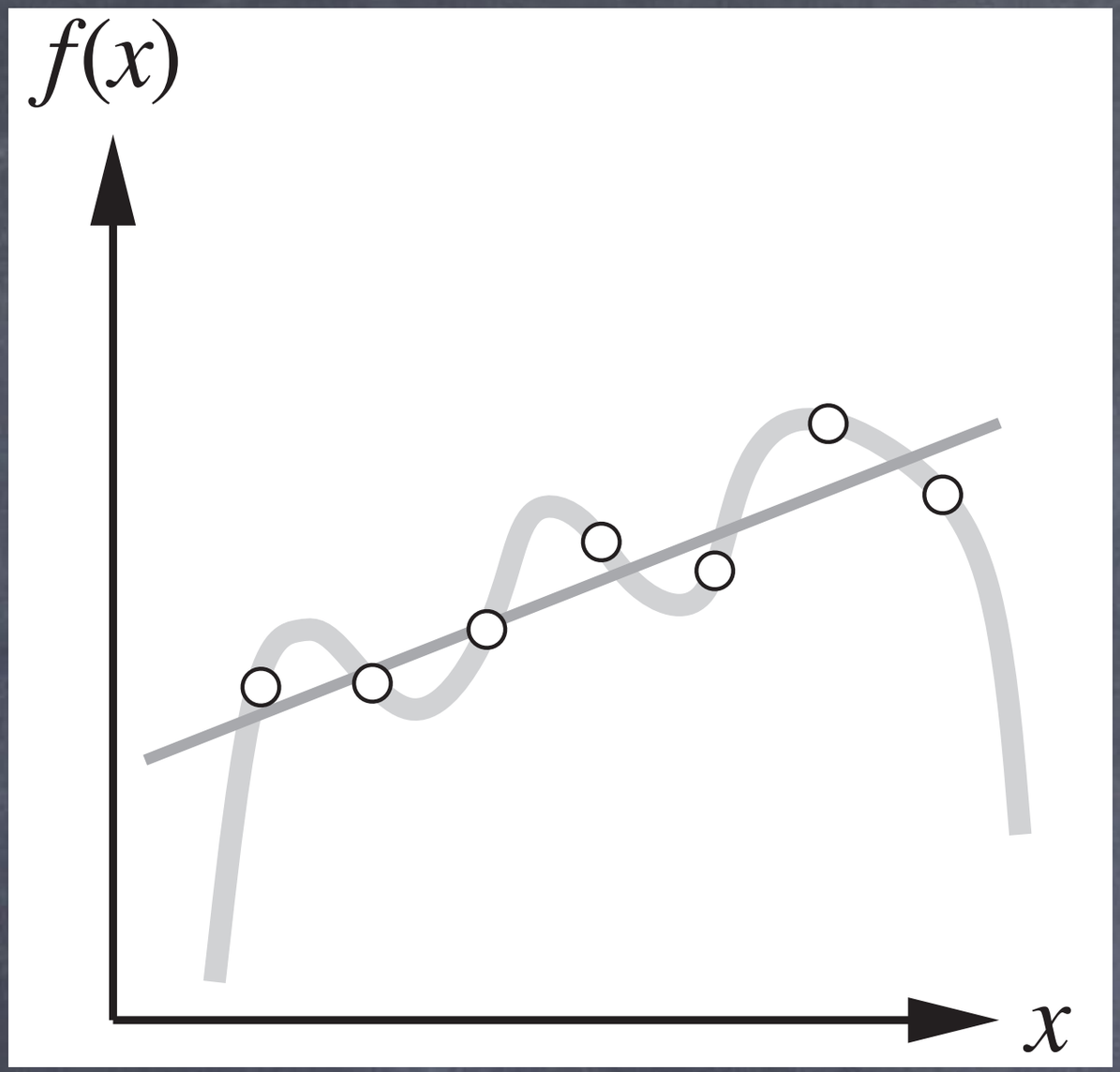
- Becomes more likely as the hypothesis space and number of input attributes grows
- Becomes less likely as the number of training examples increases
- Overfitting => Poor generalization

More Data More Data More Data
(or simpler hypotheses)

Regression and Classification with Linear Models

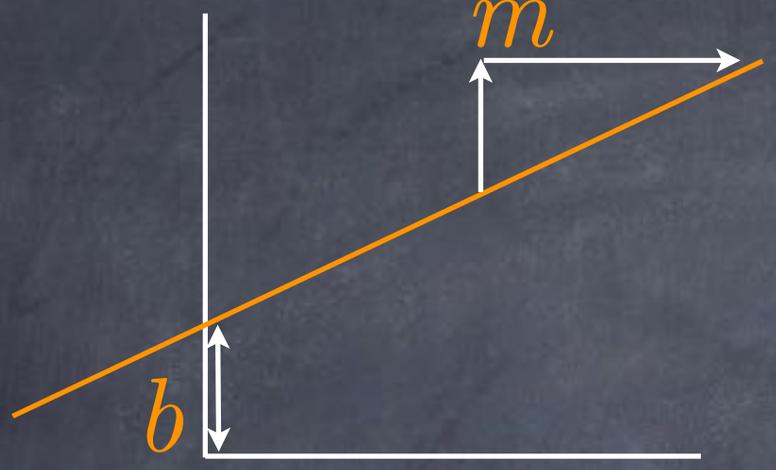


$$y = 0.4x+3$$


$f(x)$
x

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



$$y = mx + b$$

$$= w_1 x + w_0$$

$$= w_0 + w_1 x$$

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

$$\mathbf{x} = [1, x]$$

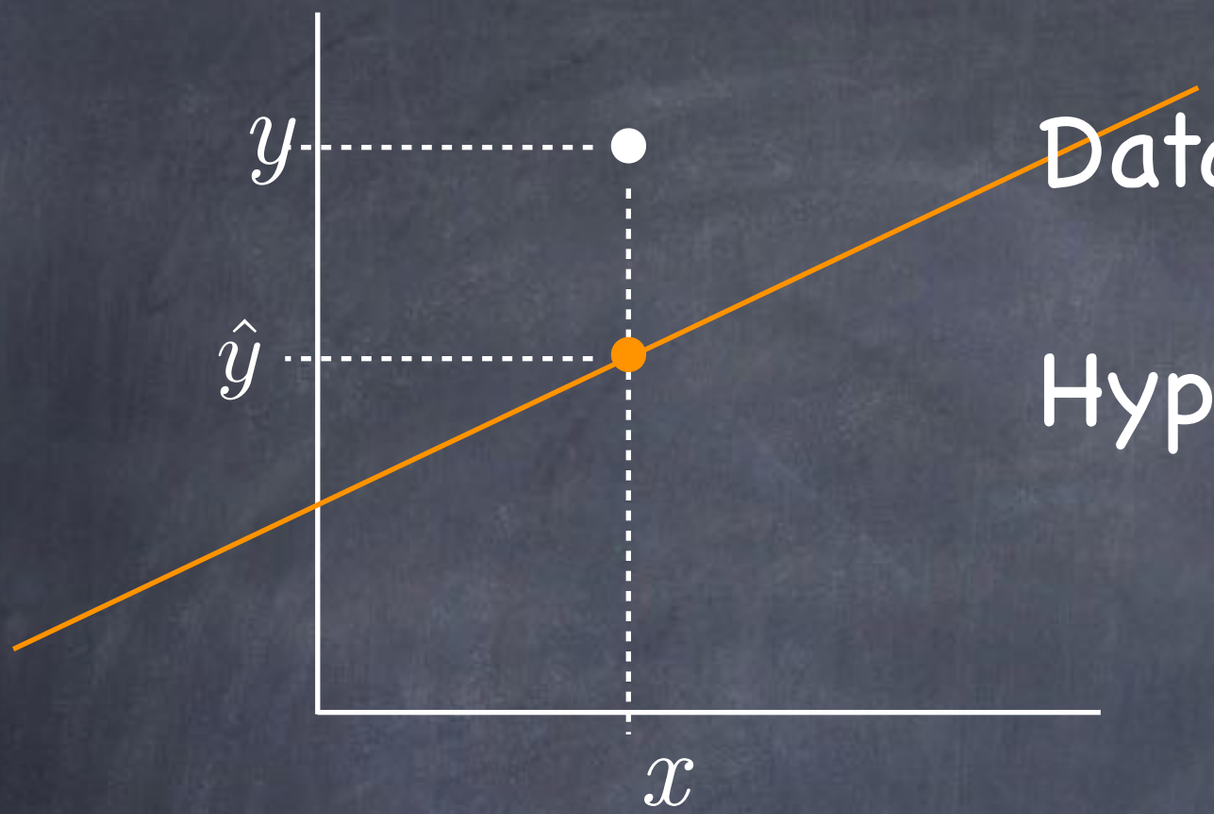
$$\mathrm{y} = \mathbf{w} \cdot \mathbf{x}$$

Univariate Linear Regression

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

$$h_{\mathrm{w}}(x) = \mathbf{w} \cdot \mathbf{x} = w_0 + w_1 x$$

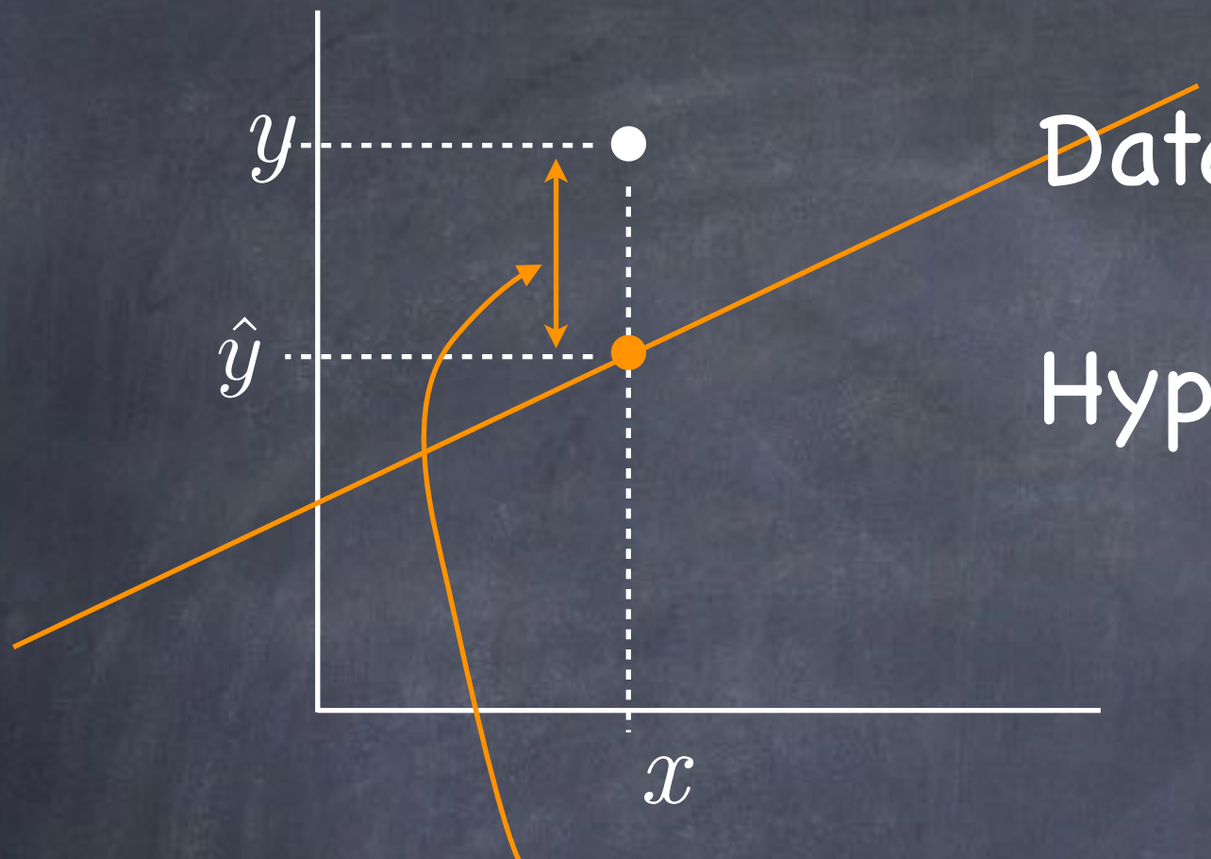
that best fits the data



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

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

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



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

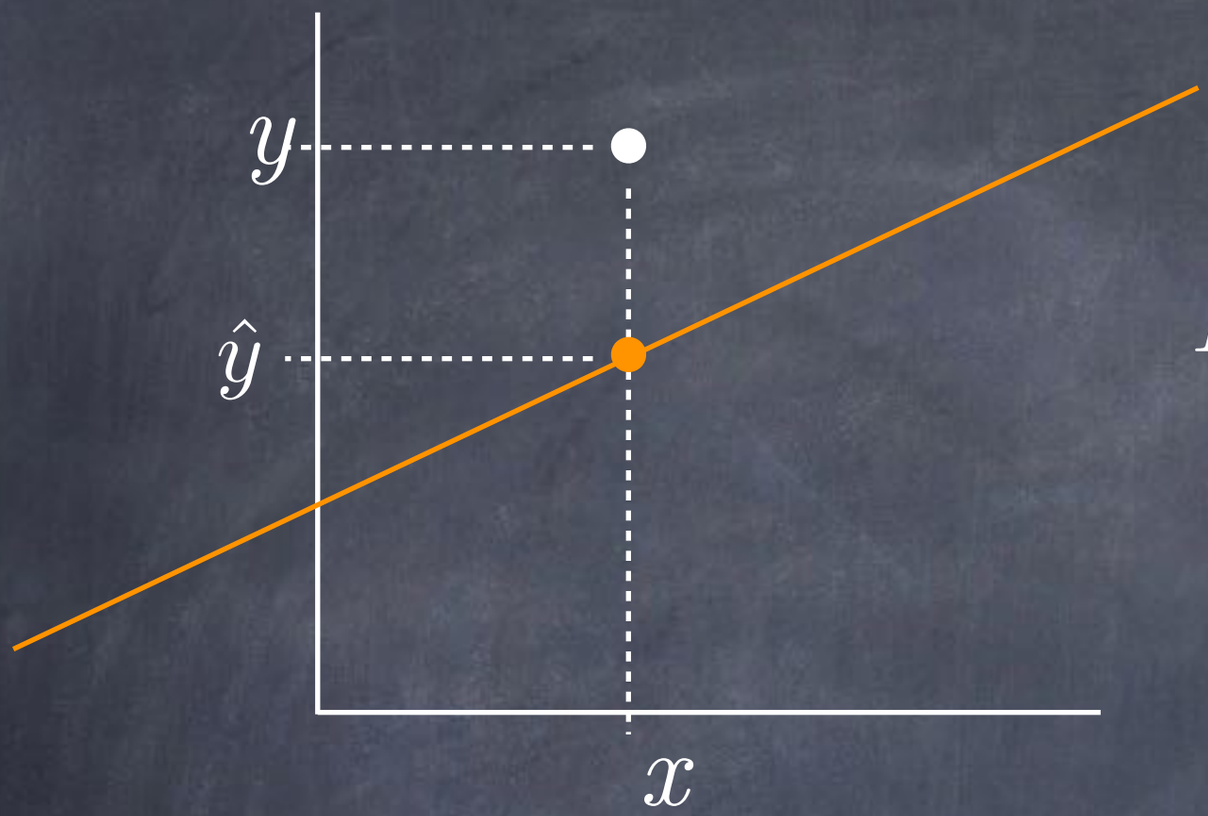
Hypothesis $h_{\mathrm{w}}(x) = w_1 x + w_0$

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

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

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

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

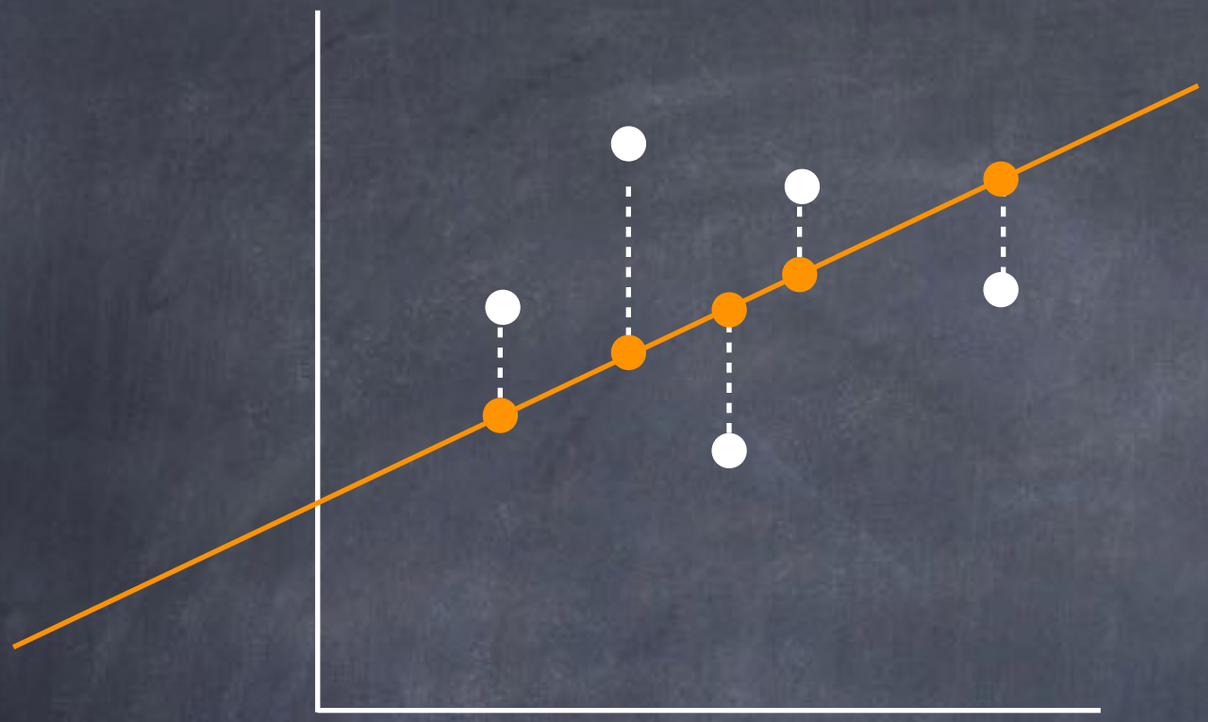


$$h_{\mathrm{w}}(x) = w_1 x + w_0$$

$$L(y, h_{\mathbf{w}}(x)) = L_2(y, h_{\mathbf{w}}(x))$$

$$= (y - h_{\mathbf{w}}(x))^2$$

$$= (y - (w_1 x + w_0))^2$$



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

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

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

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

Univariate Linear Regression

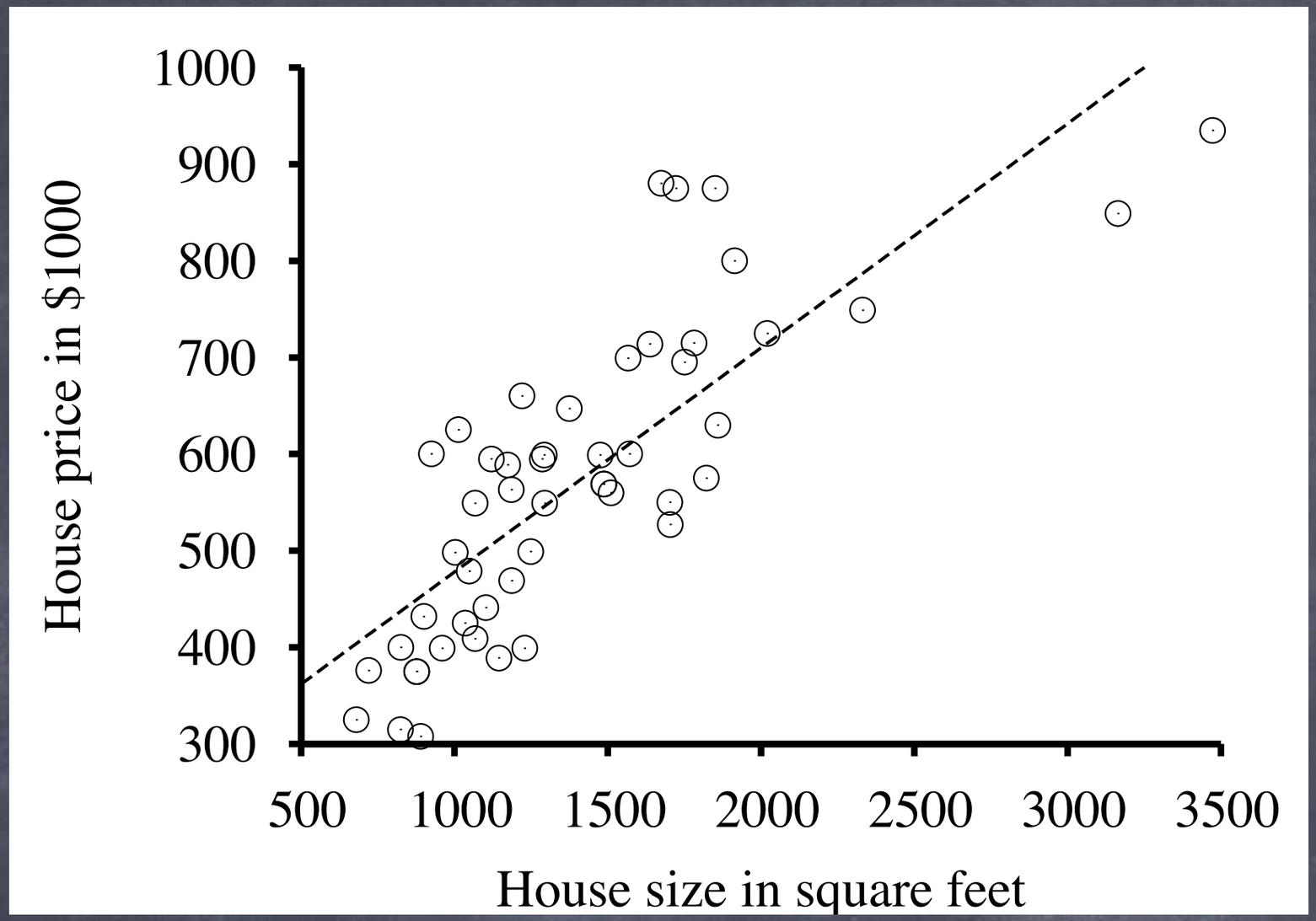
Find $\mathbf{w} = [w_0, w_1]$ that minimizes $L(h_\mathrm{w})$

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

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

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

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


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

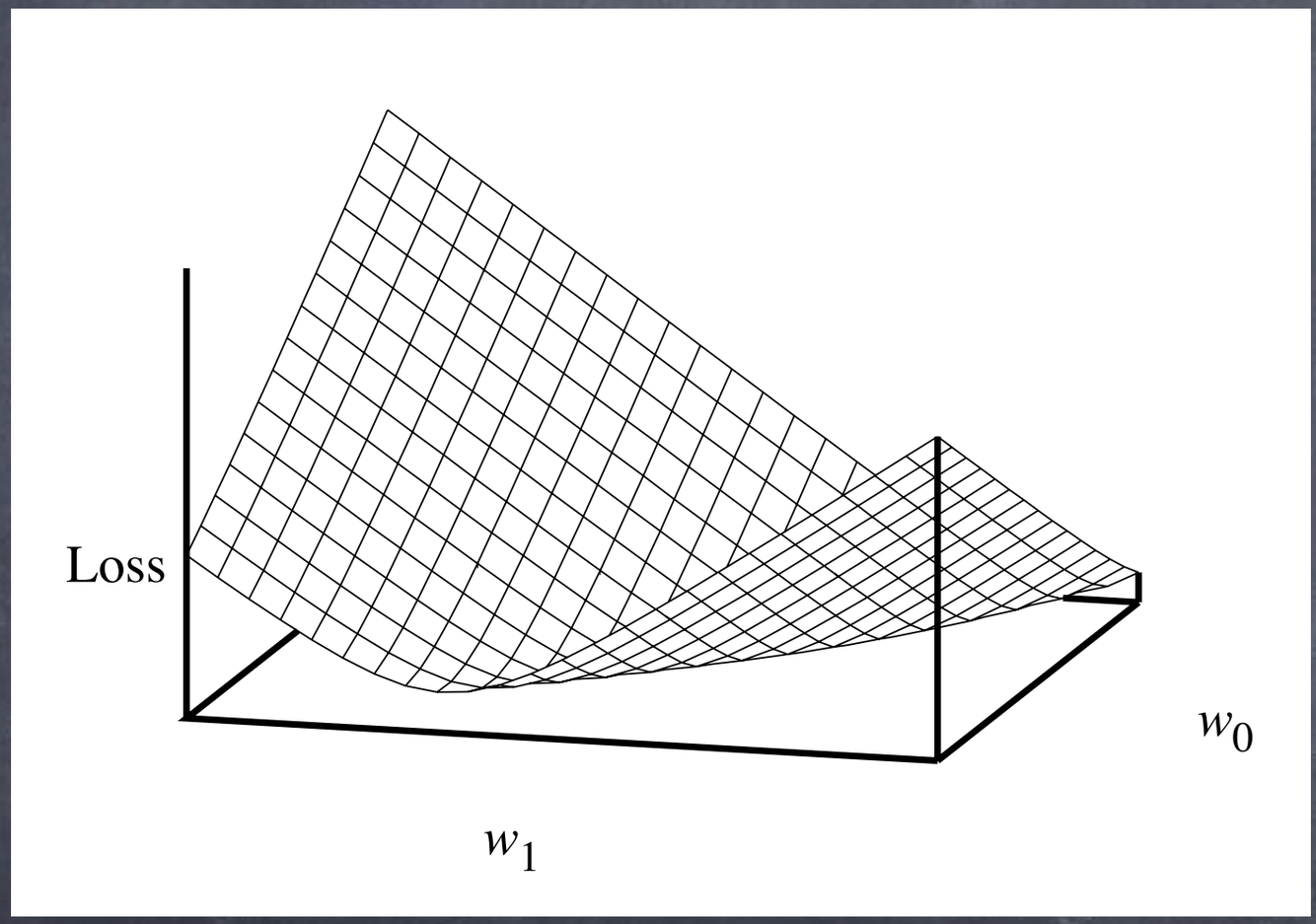
$$y = 0.232x + 246$$

General Regression

Find $\mathbf{w} = [w_0, w_1]$ that minimizes $L(h_\mathrm{w})$

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

Weight Space



Gradient Descent

$\mathbf{w} \leftarrow$ any point in parameter space

loop until convergence do

for each w_i in $\mathbf{w}$ 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

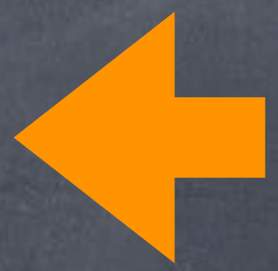
Gradient Descent for Univar. Linear Regression

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

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

Gradient Descent for Univar. Linear Regression

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

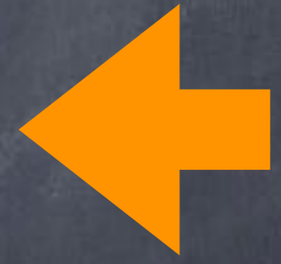


$$w_1 \leftarrow w_1 + \alpha \sum_j (y_j - h_{\mathbf{w}}(x_j)) \times x_j$$

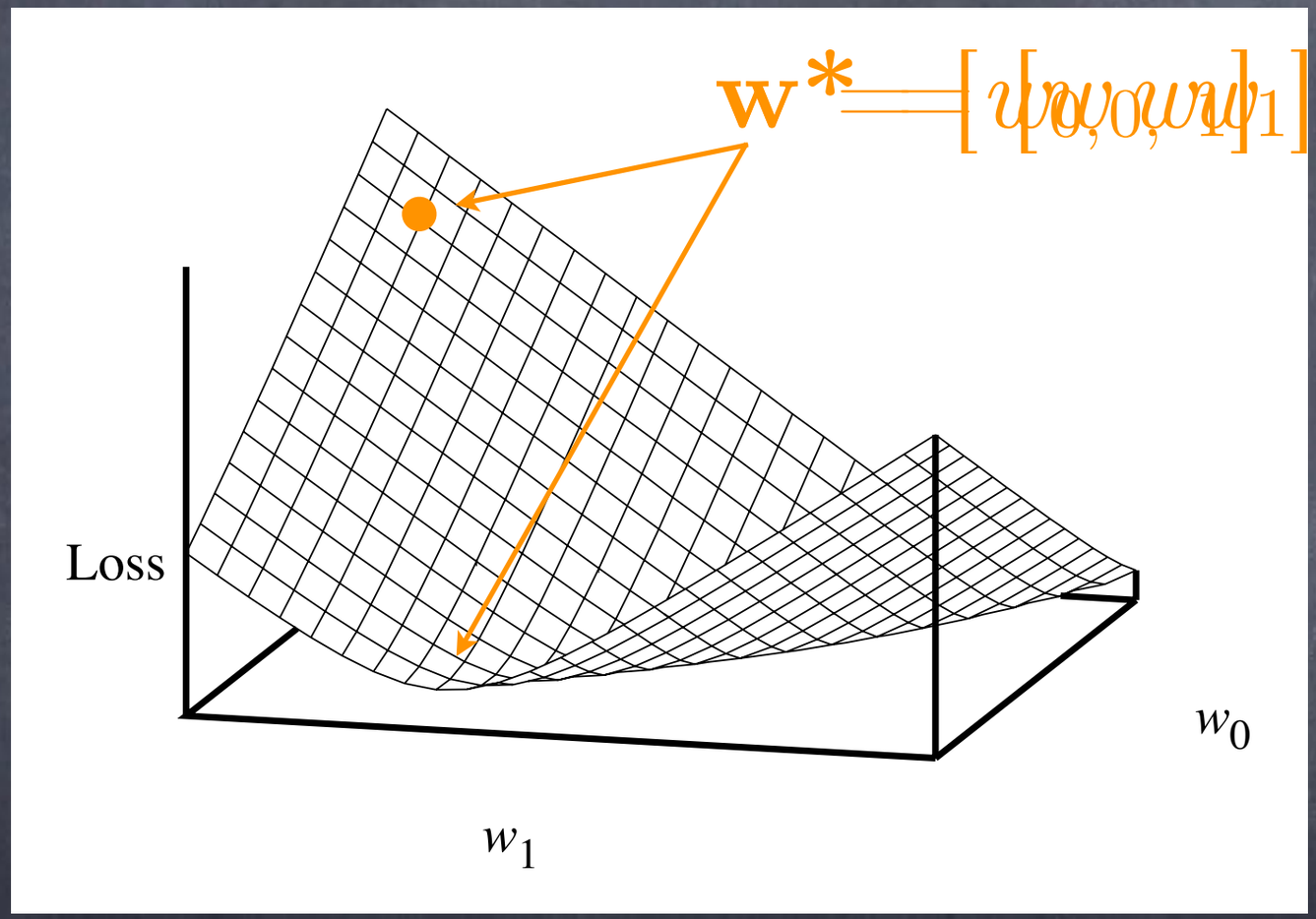
Gradient Descent for Univar. Linear Regression

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

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



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 $\mathbf{w} = [w_0, w_1]$ that minimizes $L(h_{\mathrm{w}})$
- 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 $\mathbf{x} = [x_1, x_2, \ldots, x_n]$
- Example:

x_1 = weight of car

x_2 = size of engine

x_3 = color of car

$f(\mathbf{x})$ = 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: $h_{\mathbf{w}}(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x}$
- Goal: $\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} L(h_{\mathbf{w}})$
- 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

THE OFFICIAL

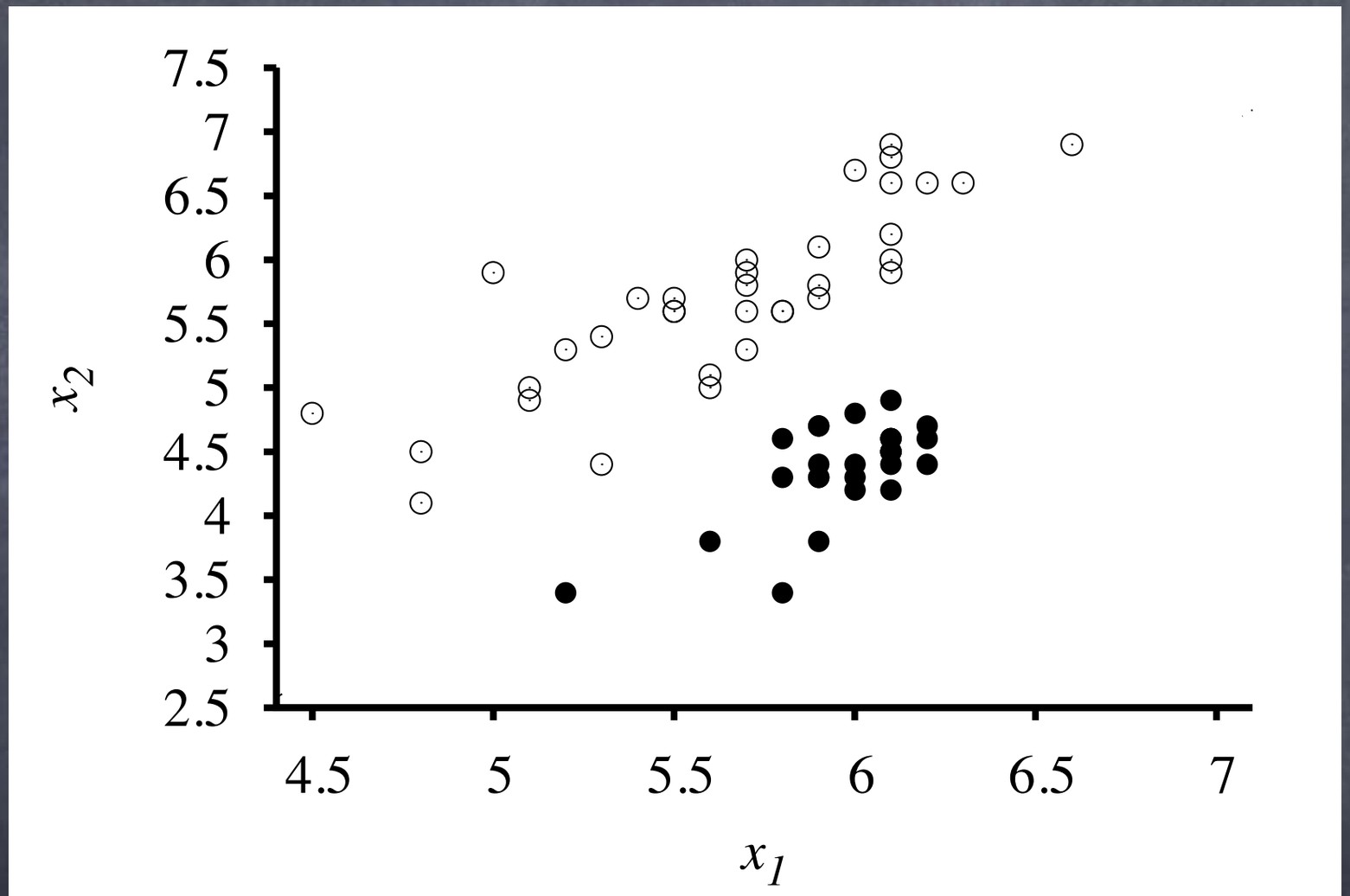
METALLICA

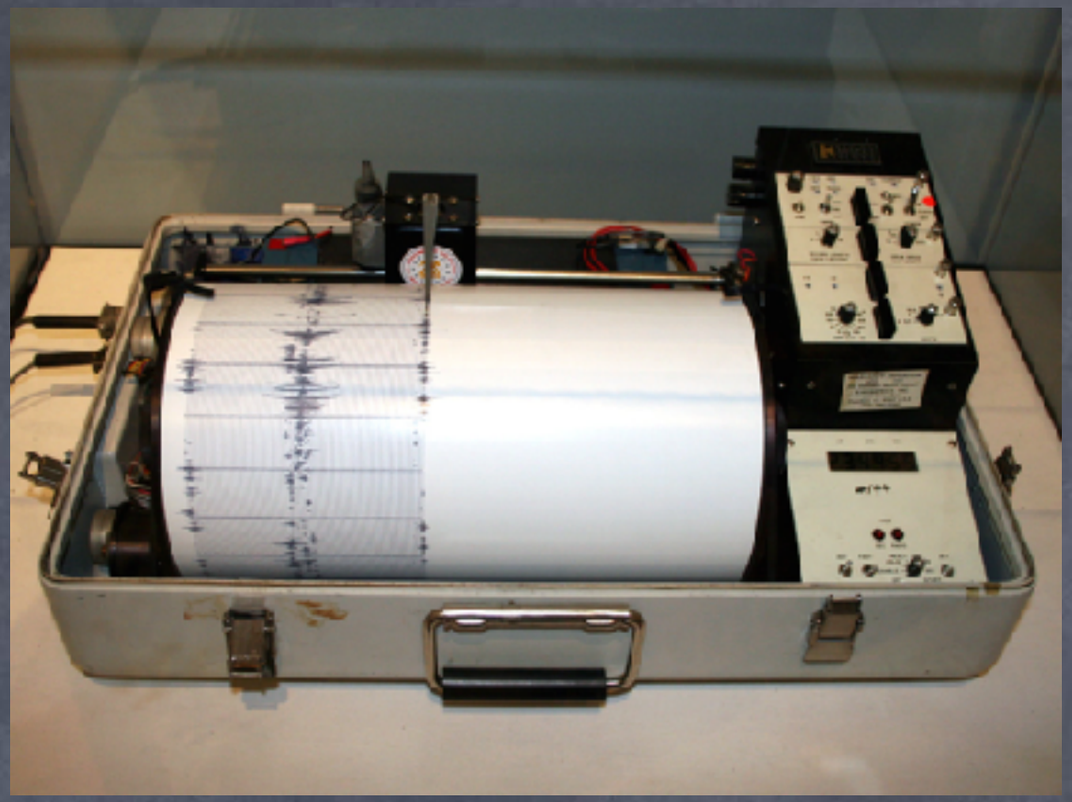
ILLUSTRATED CHRONICLES

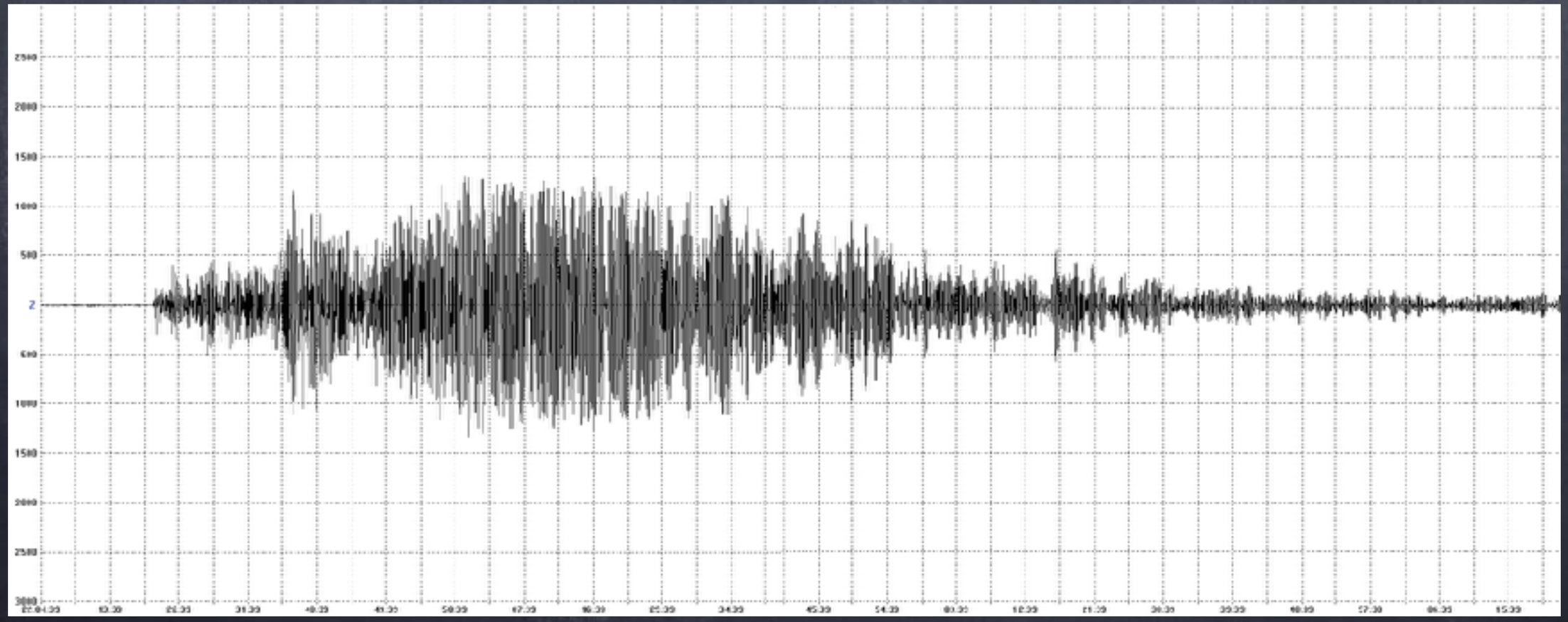
SO WHAT!

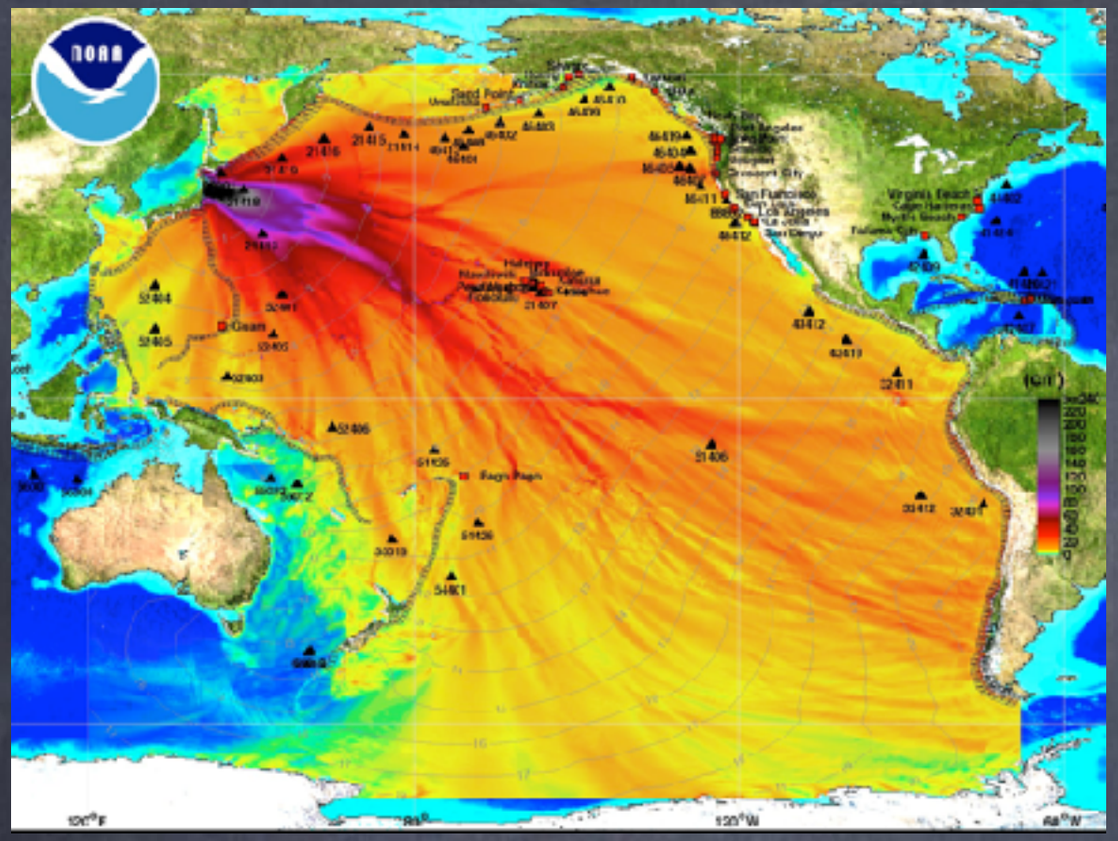
THE GOOD, THE MAD AND THE UGLY

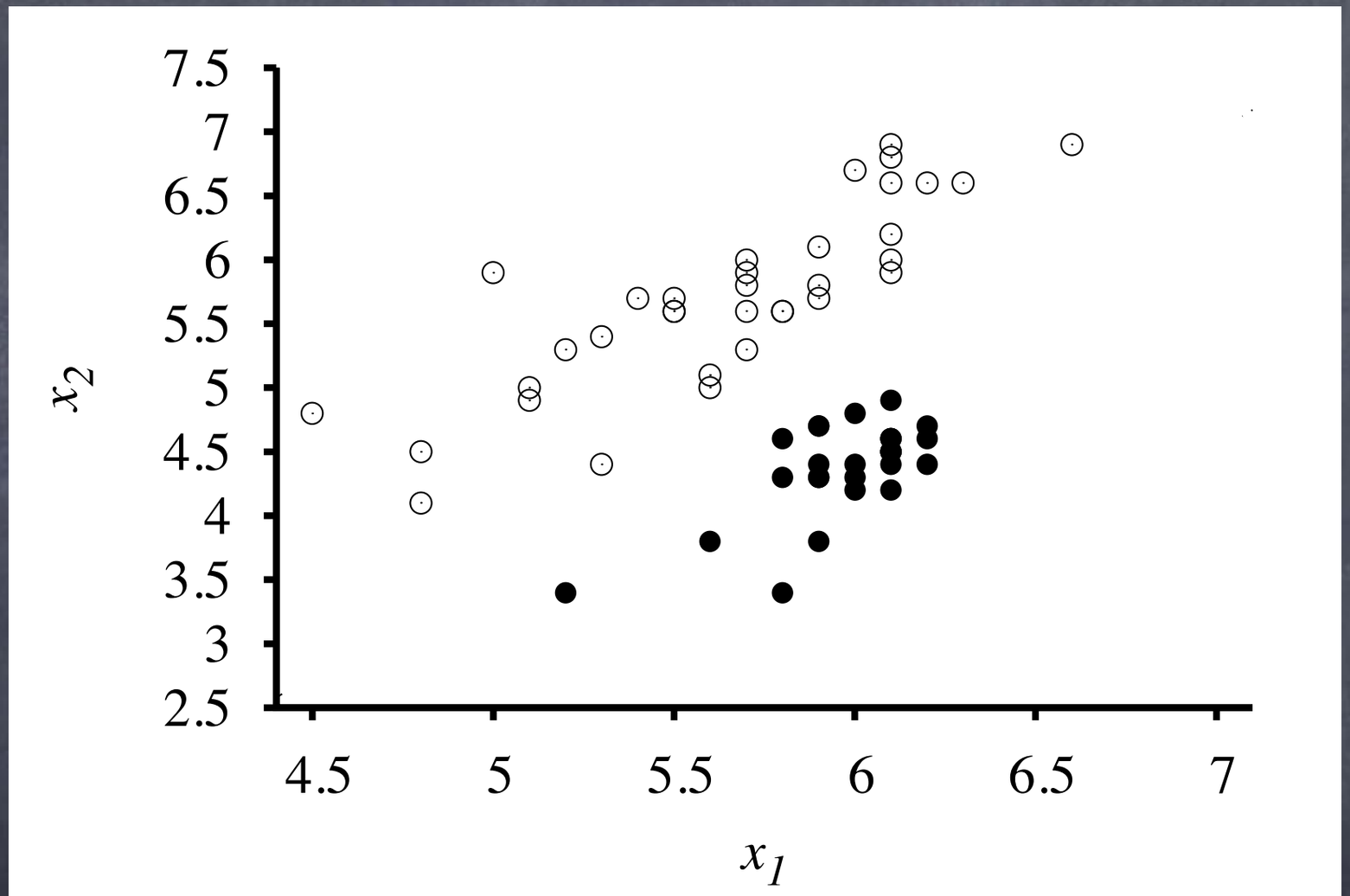
EDITED BY STEFFAN CHIRAZI


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

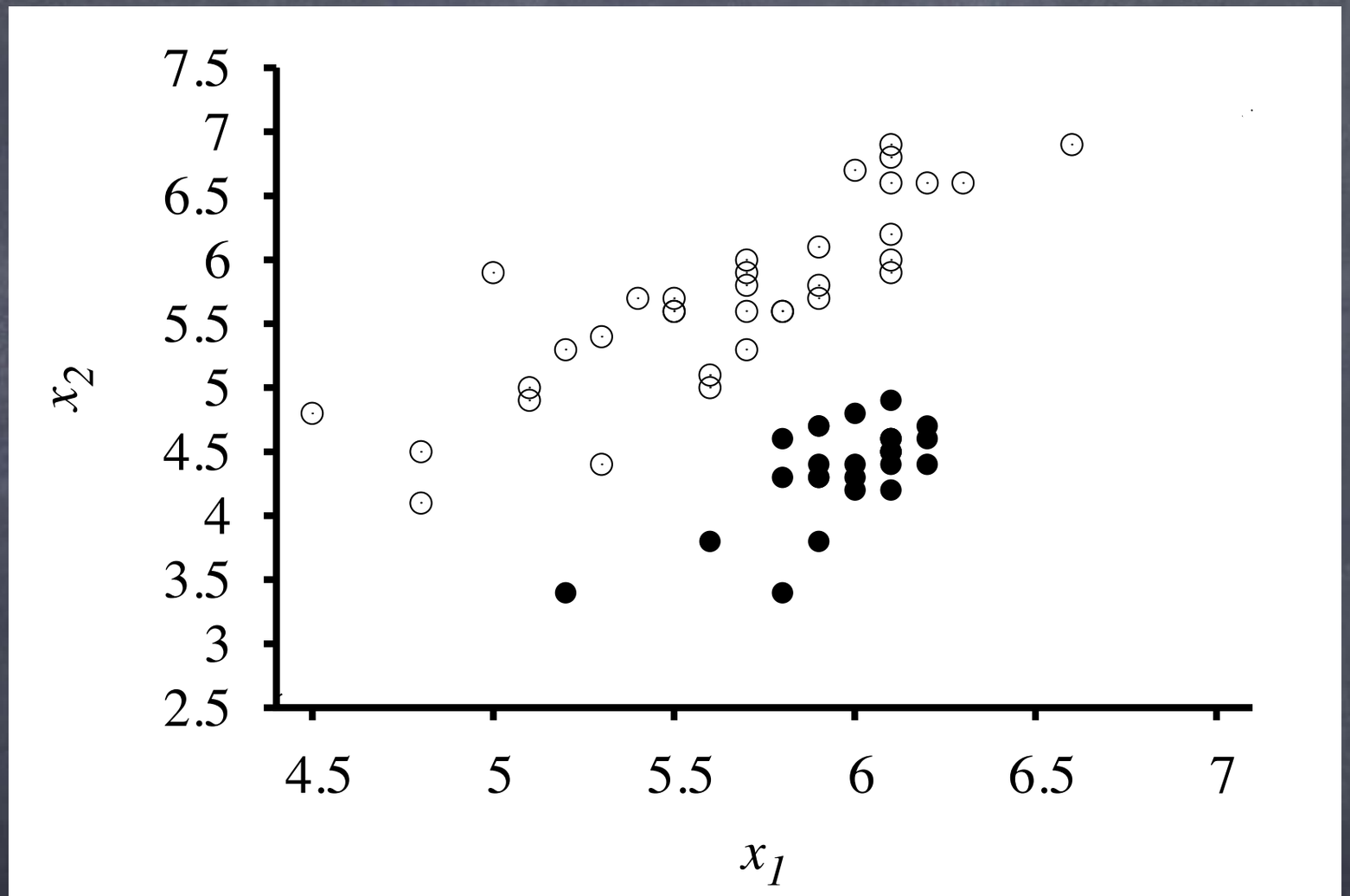





NOAA


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


WAR ALERT
ACTIONS BOOK


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 $\mathbf{x} = \langle x_1, x_2 \rangle$,

  Learn a hypothesis h such that:

  $h(\mathbf{x}) = 0$ if $\mathbf{x}$ is from an earthquake

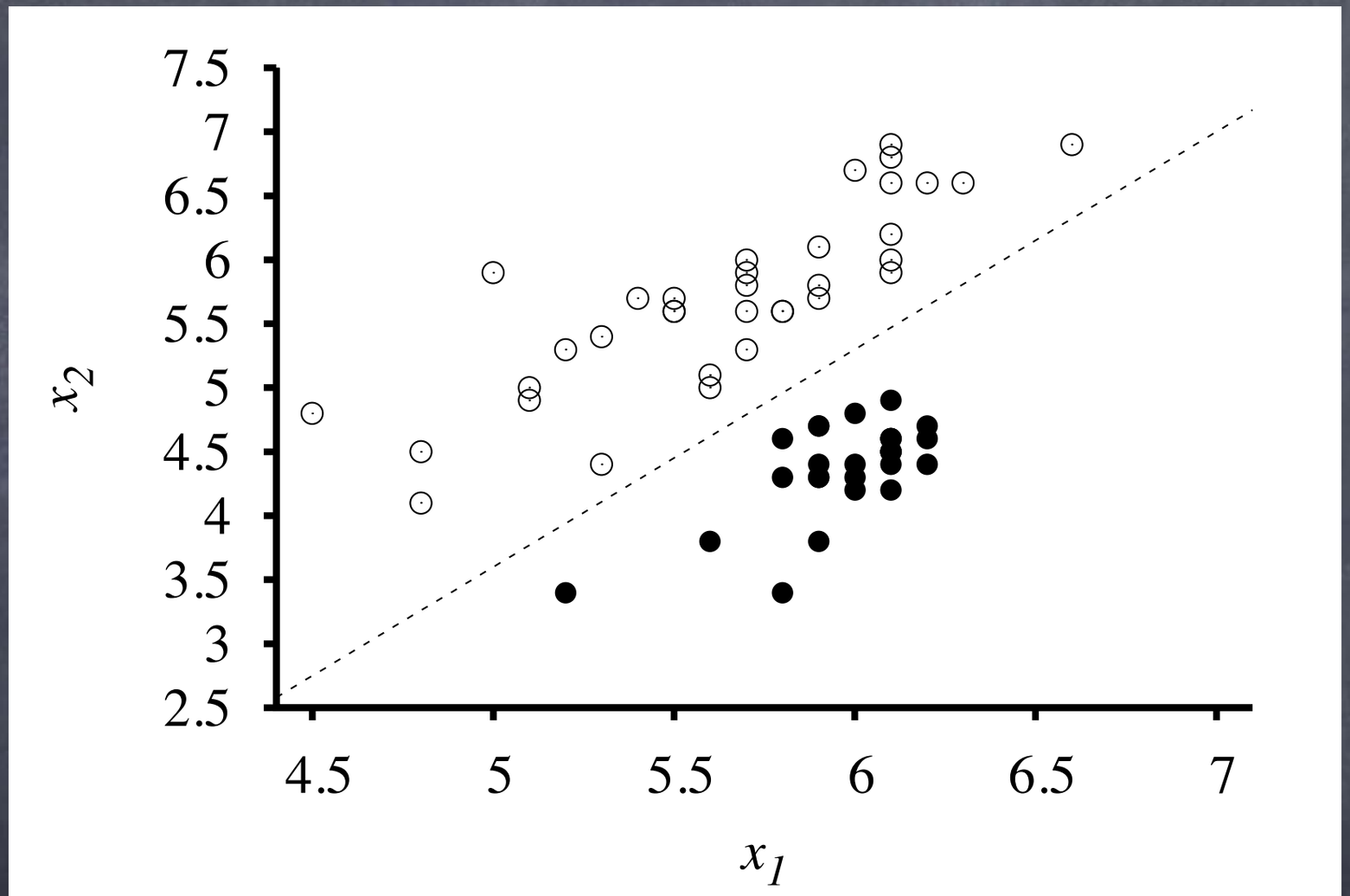
  $= 1$ if $\mathbf{x}$ is from an explosion

Decision Boundary

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

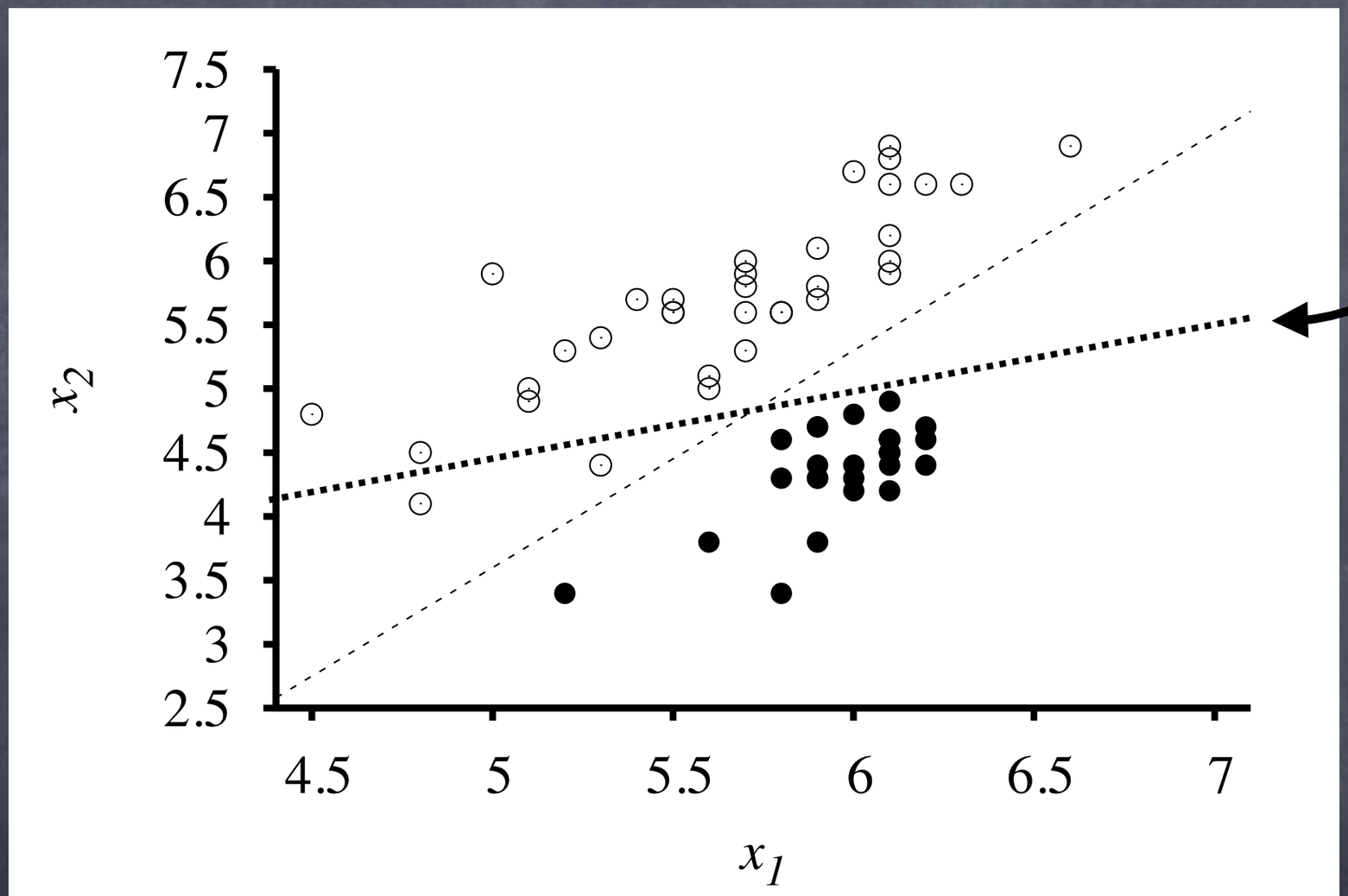
$h'(\mathbf{x}) > 0$ if $\mathbf{x}$ is from an earthquake

< 0 if $\mathbf{x}$ 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
- Line in 2D, plane in 3D, hyperplane in nD
- Data that admit a linear separator are said to be linearly separable

Linear Classifier

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

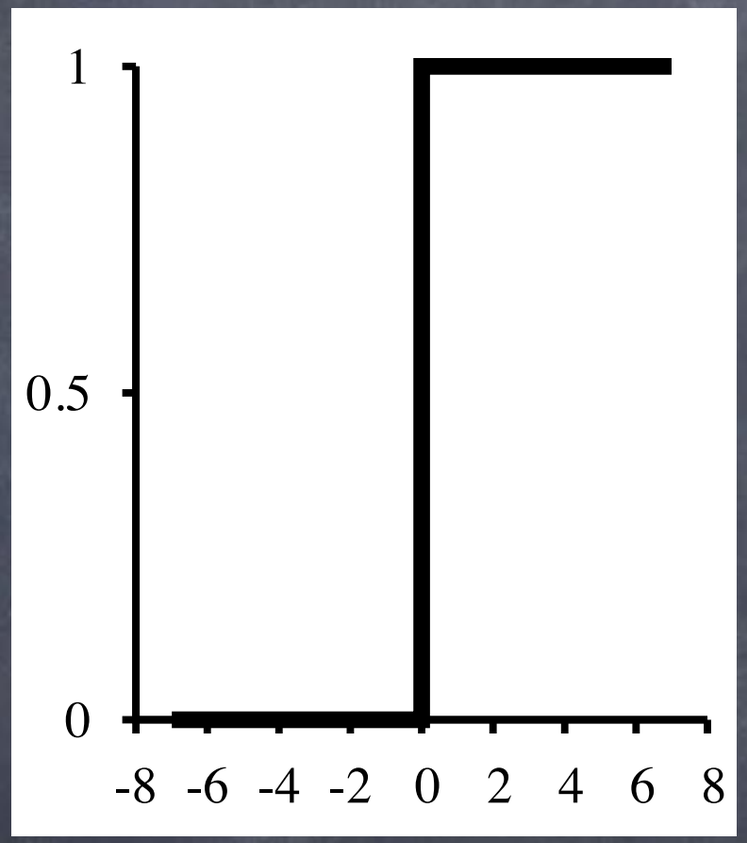
$$\mathbf{w} \cdot \mathbf{x} = 0$$

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

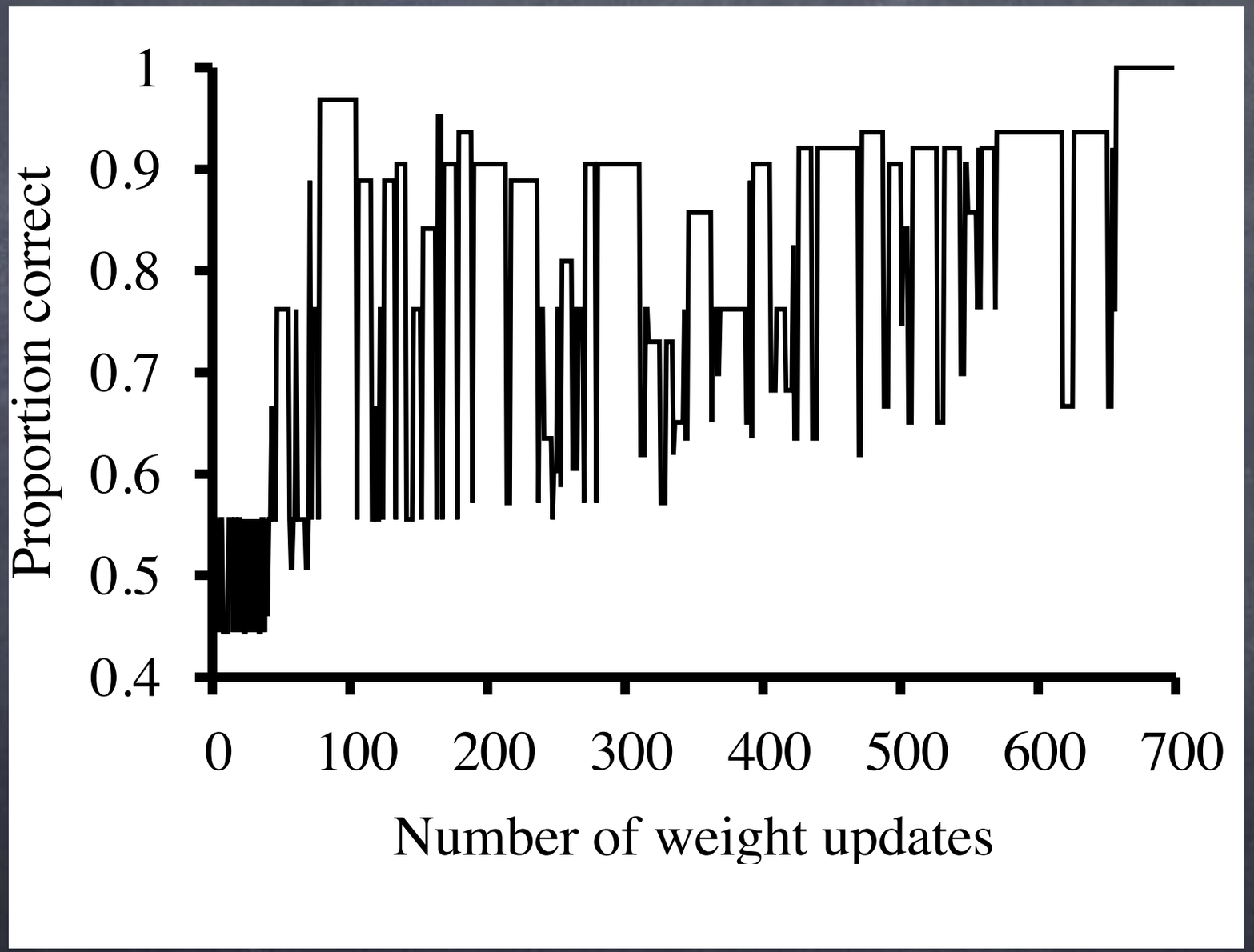
$$h_{\mathbf{w}}(\mathbf{x}) = \mathit{Threshold}(\mathbf{w} \cdot \mathbf{x})$$

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

Perceptron Learning Rule

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

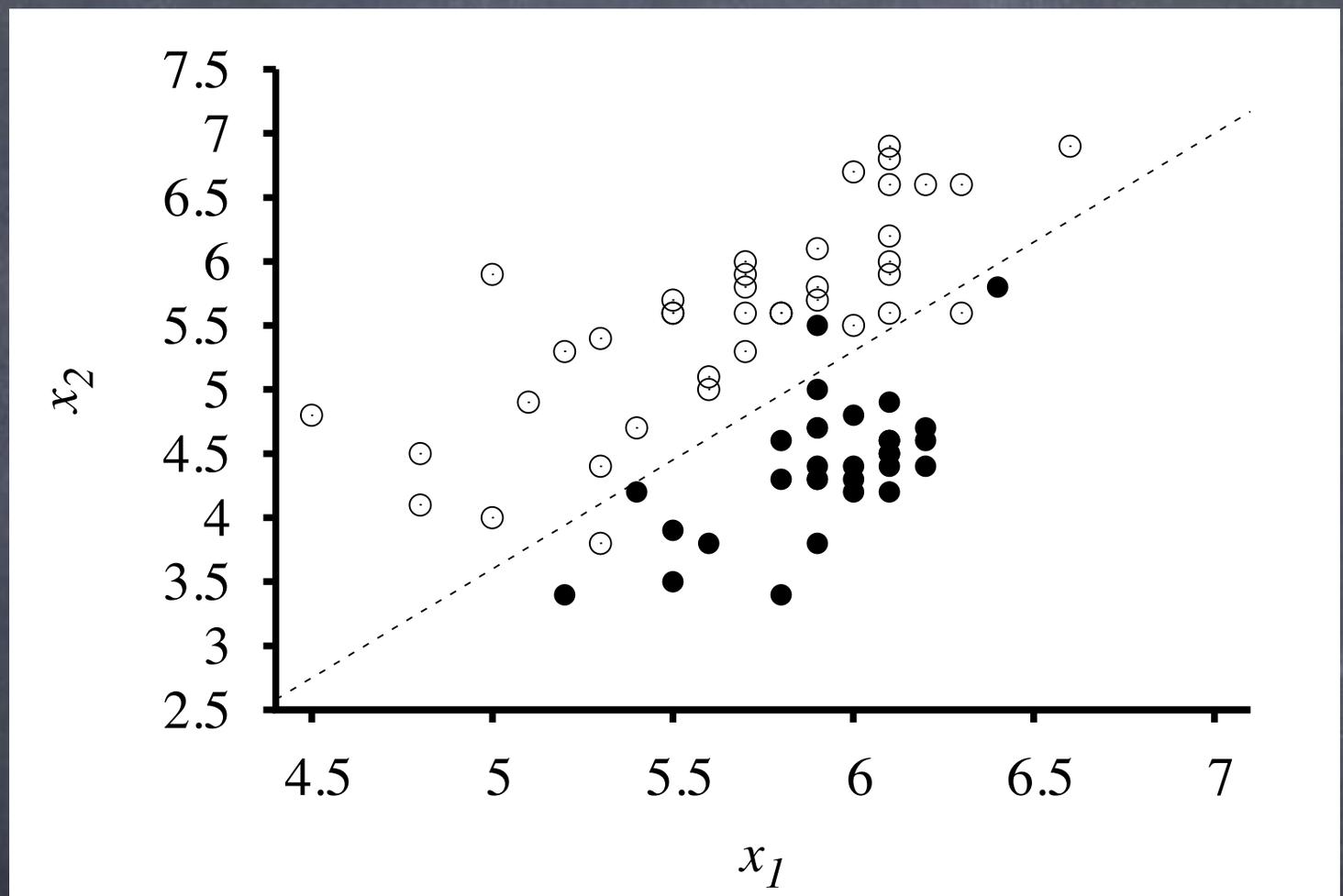
Learning Curve

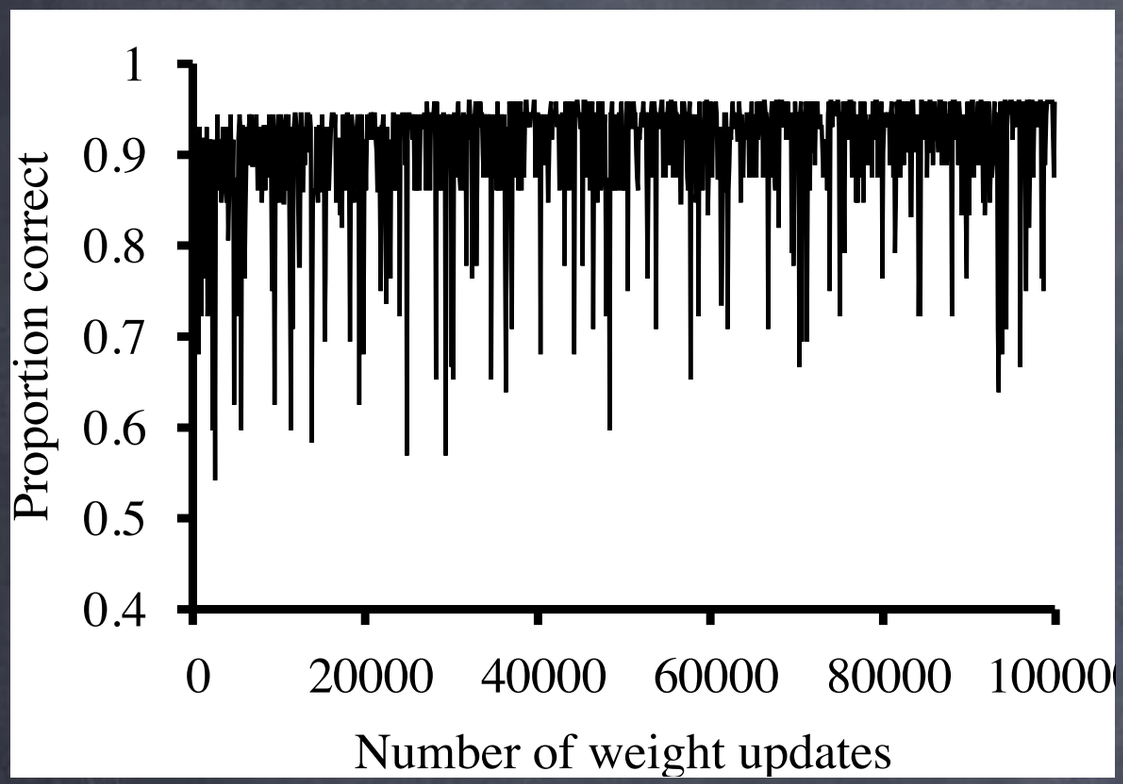


N=63

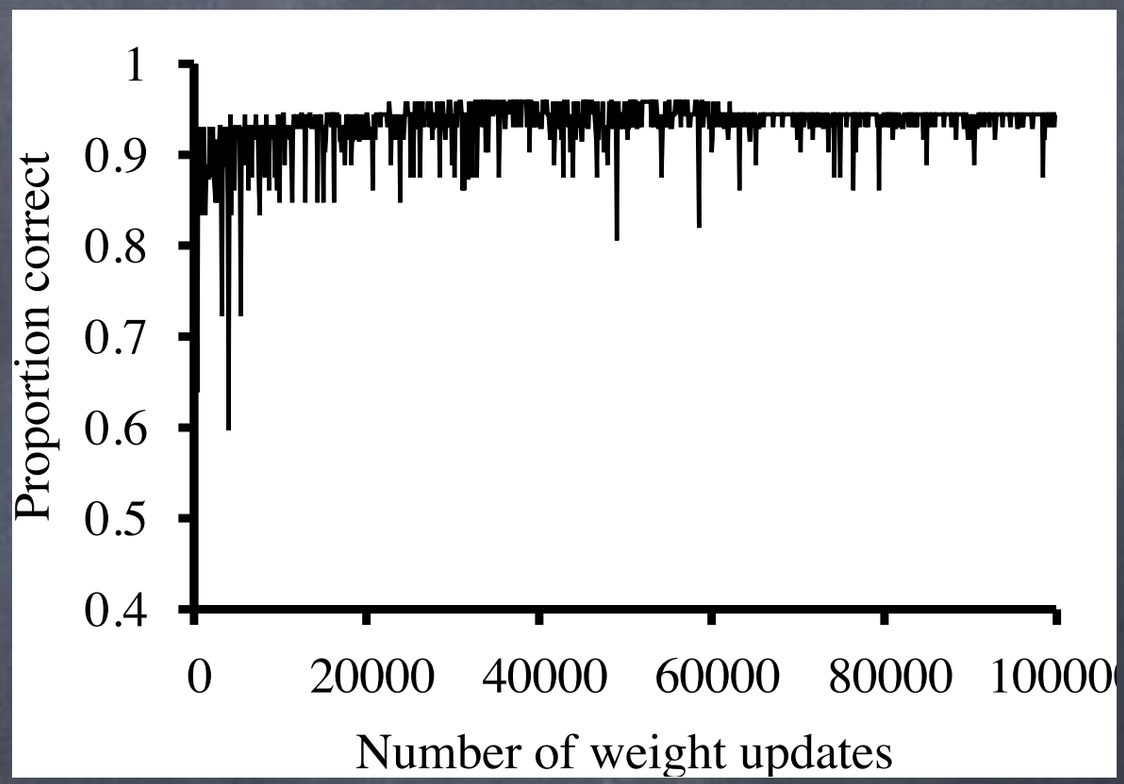
Linear Classifier with Hard Threshold

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

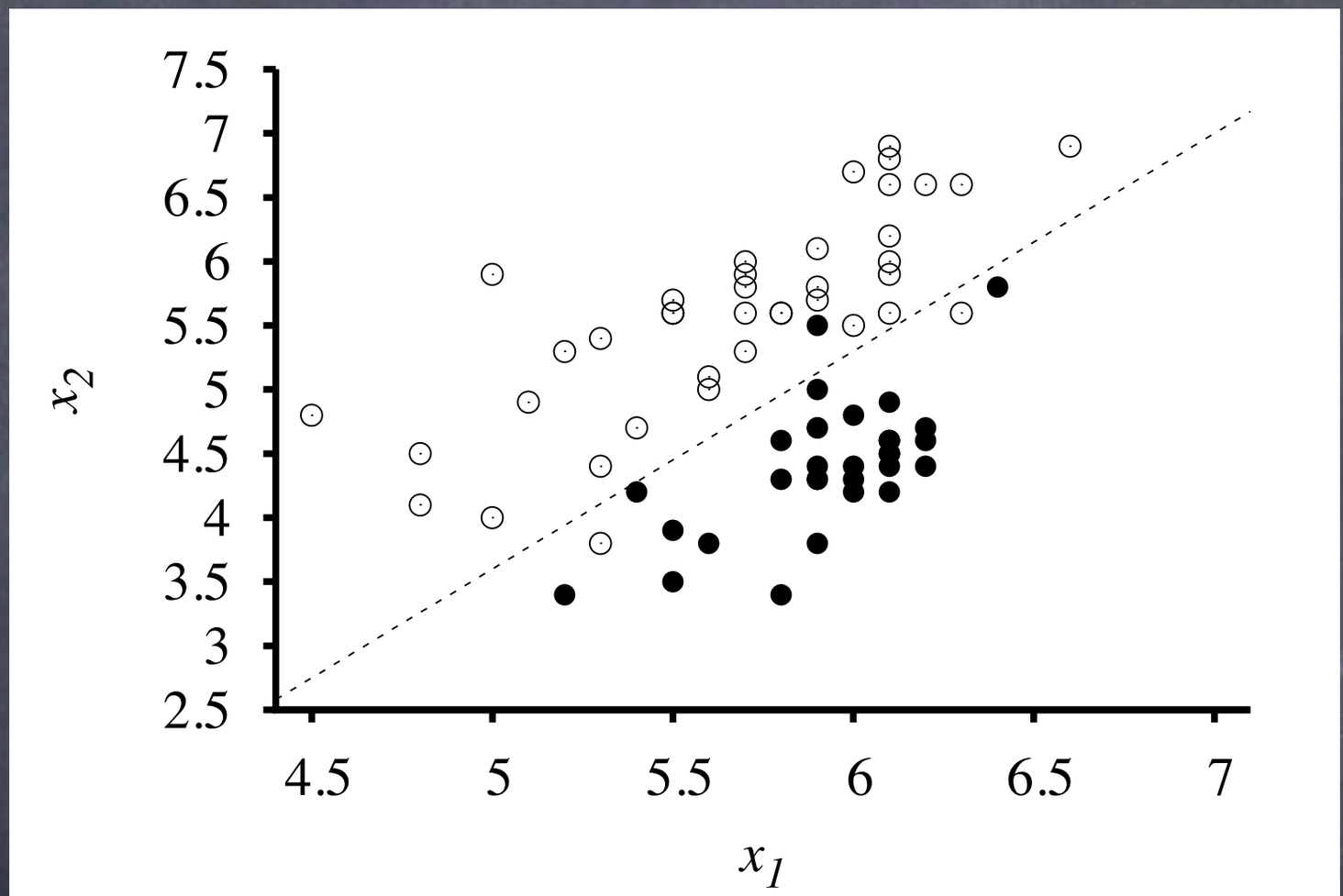

7.5
7
6.5
6
5.5
5
4.5
4
3.5
3
2.5
x_2
4.5
5
5.5
6
6.5
7
x_1



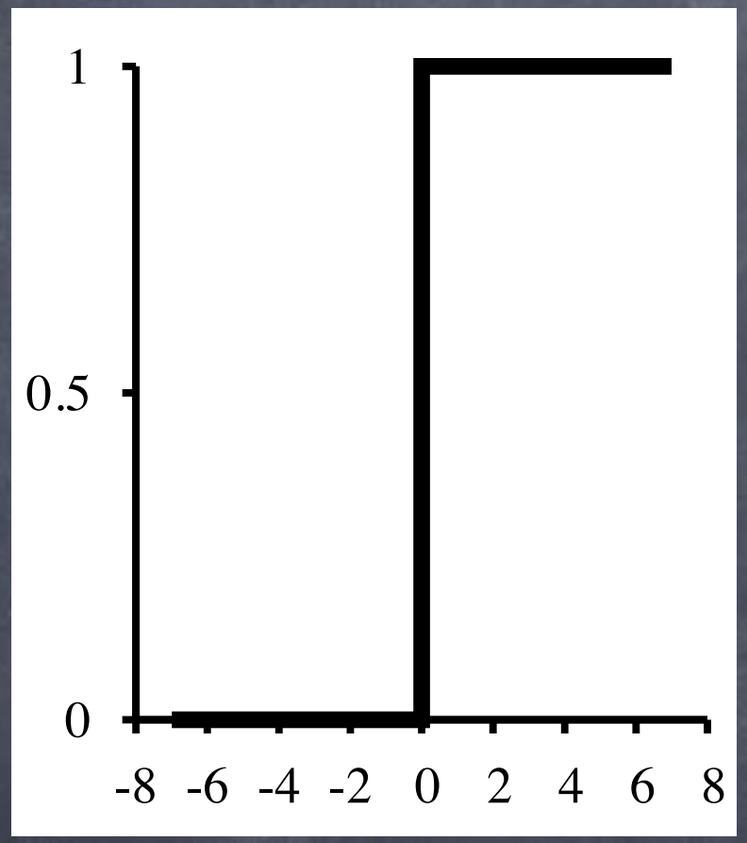
Fixed α



α decays as $O(1/t)$

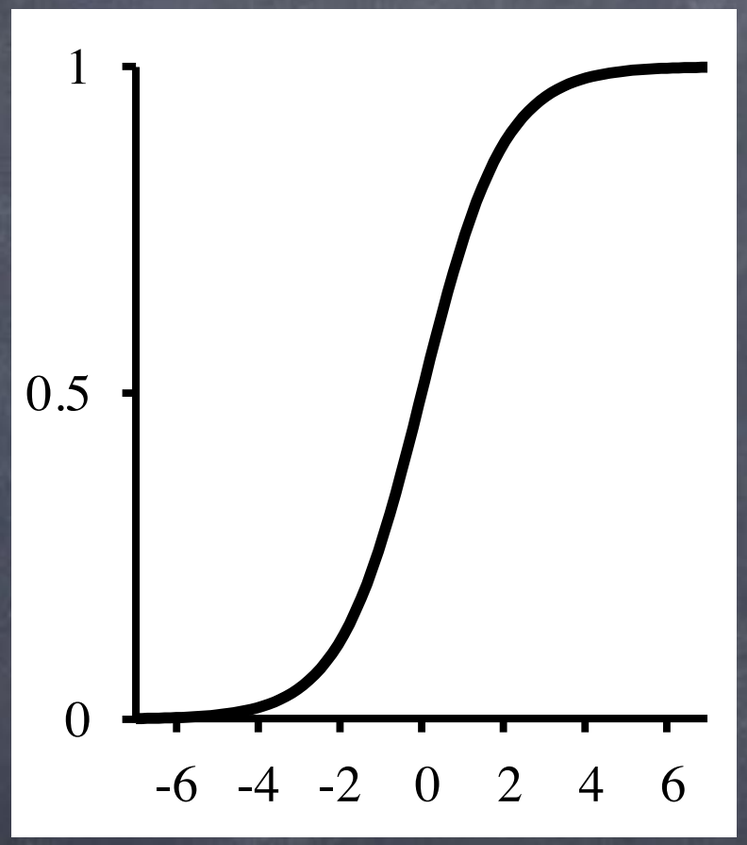

7.5
7
6.5
6
5.5
5
4.5
4
3.5
3
2.5
x_2
4.5
5
5.5
6
6.5
7
x_1

Hard Threshold



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

Soft Threshold

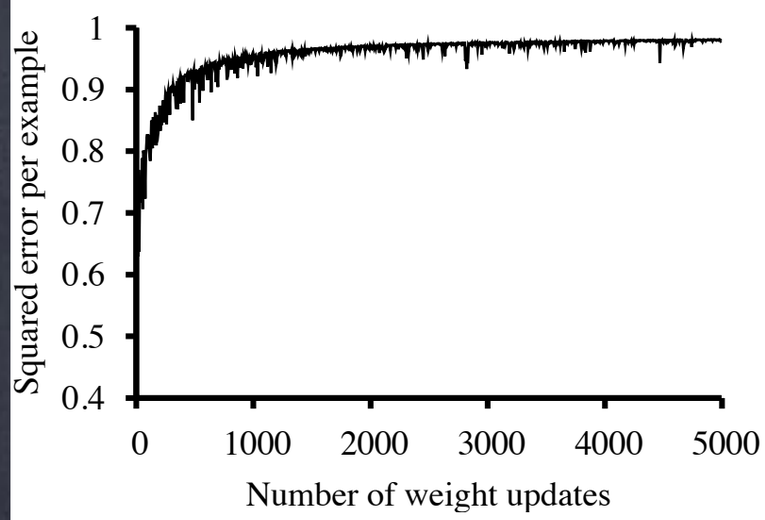


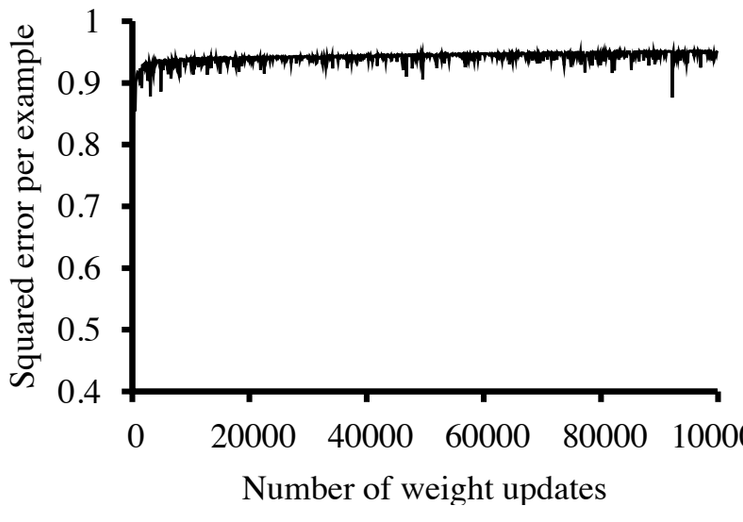
$$Logistic(z) = \frac{1}{1 + e^{-z}}$$

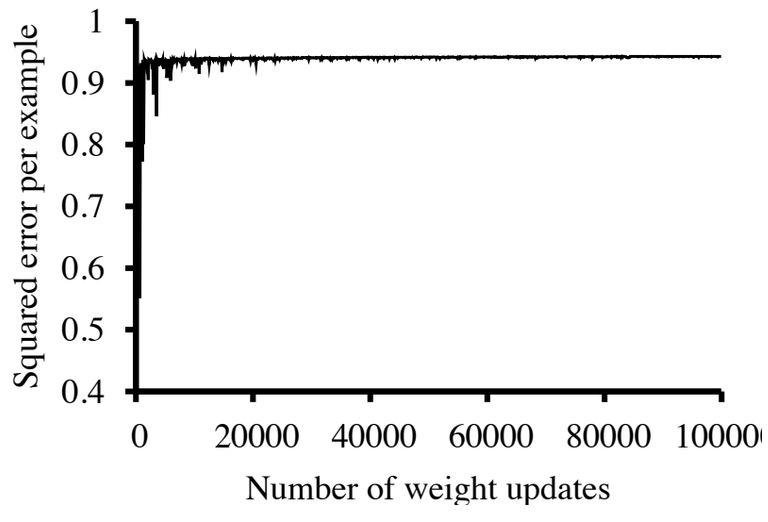
Logistic Regression

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

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


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
10000
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
10000
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