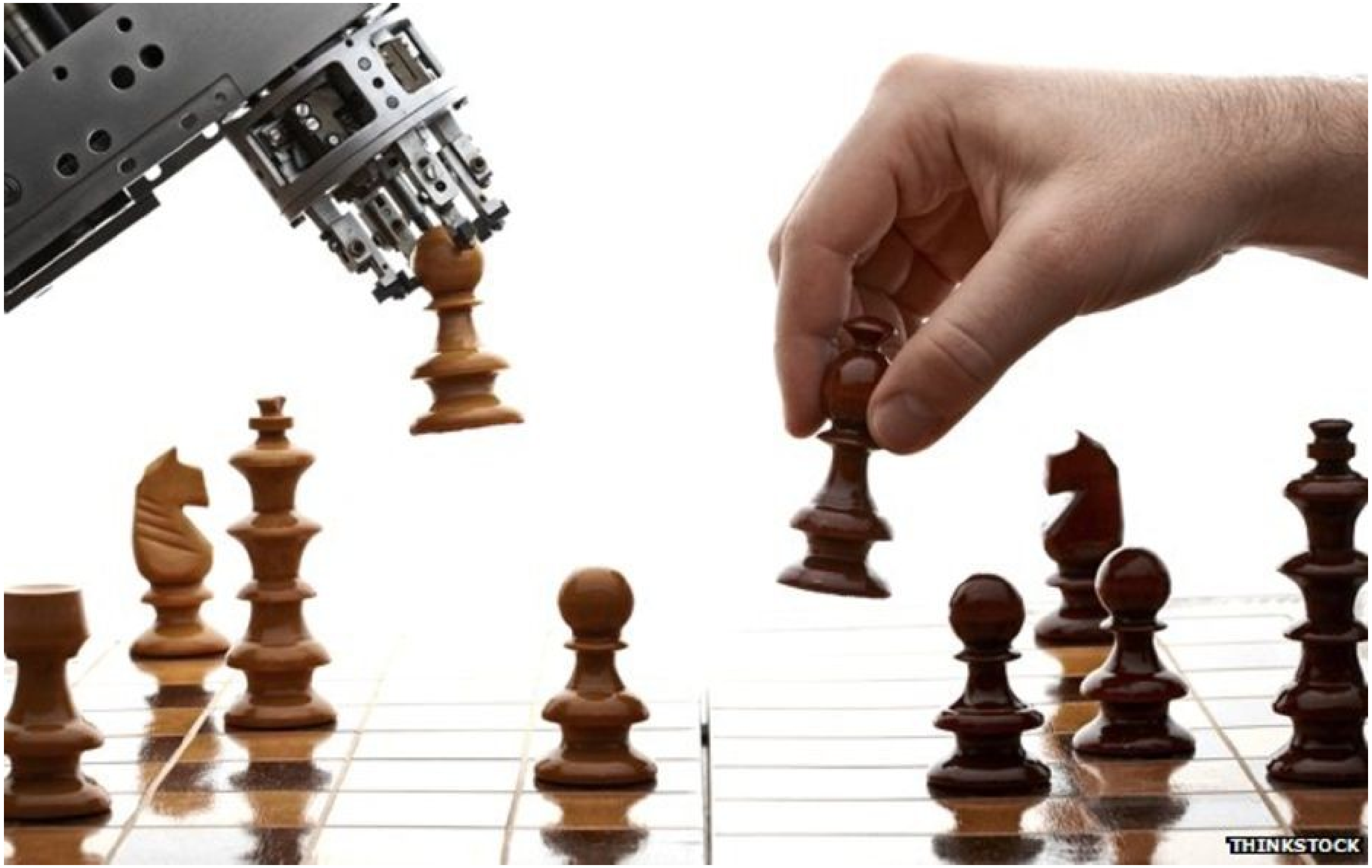


Introduction to Artificial Intelligence

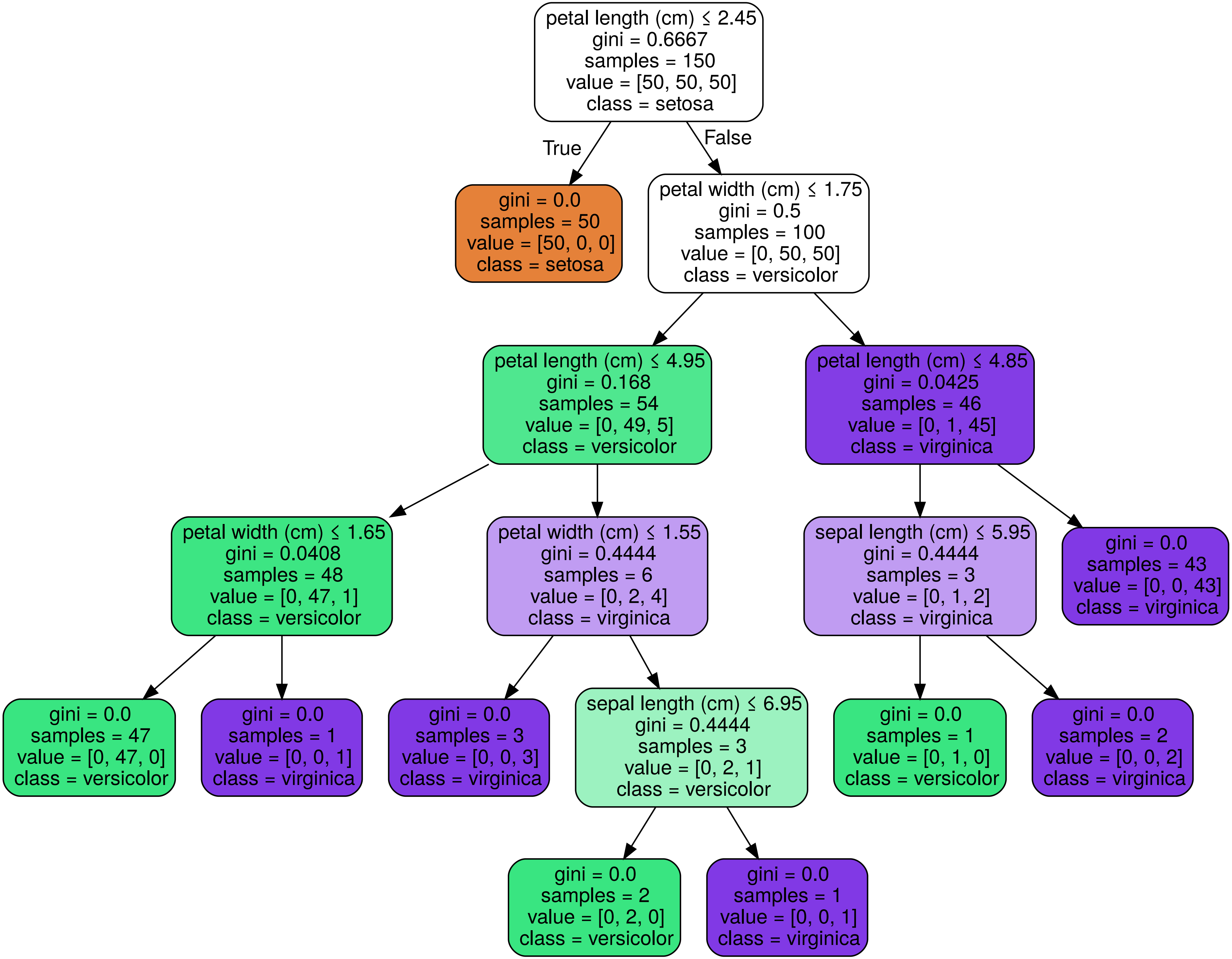
COSC 4550 / COSC 5550

Professor Cheney

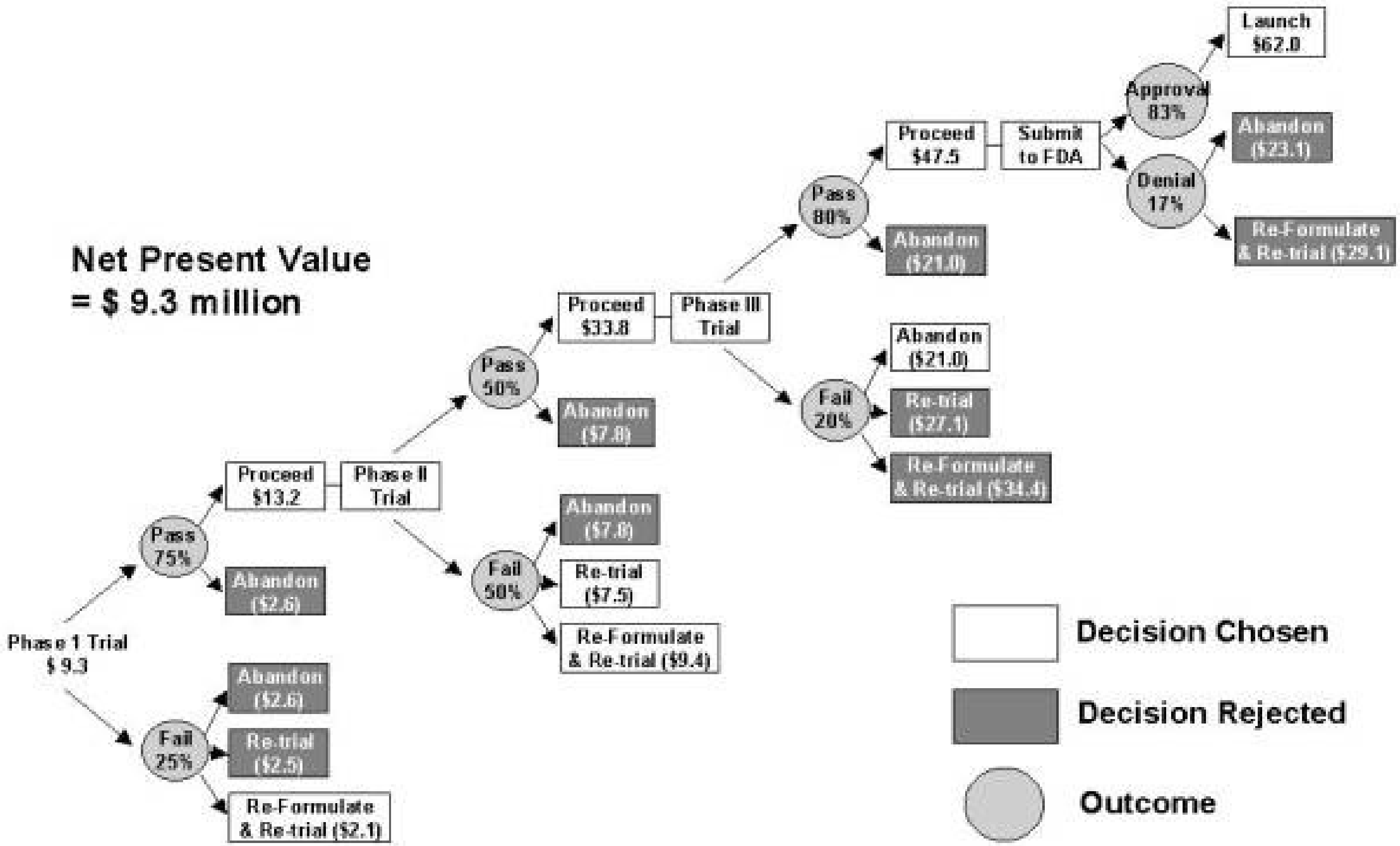
10/13/17

what about real valued data?

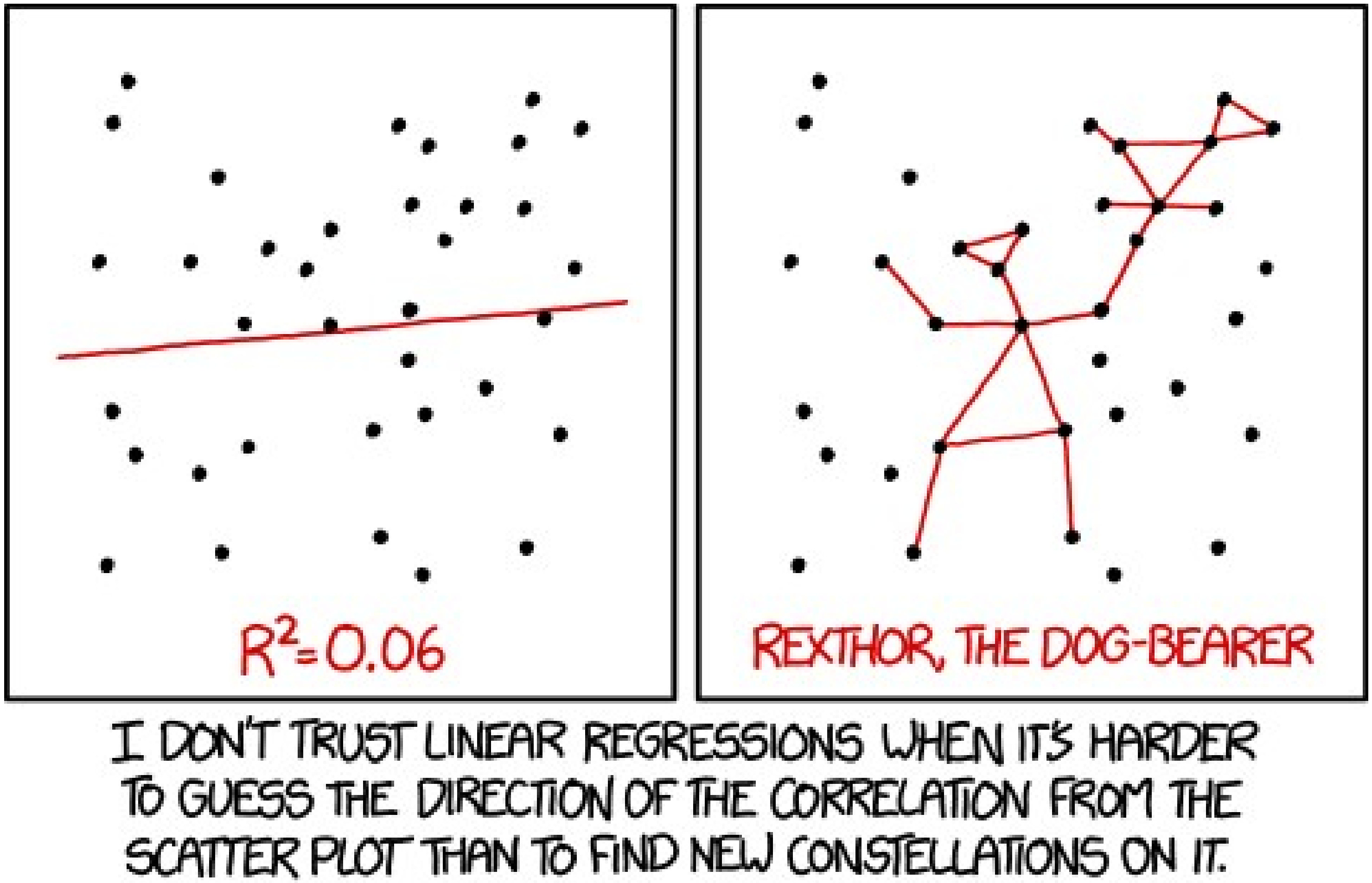
Day	Sky	Humidity	Wind	Play
D1	Sunny	63	Weak	No
D2	Sunny	61	Strong	No
D3	Overcast	59	Weak	Yes
D4	Rain	69	Weak	Yes
D5	Rain	51	Weak	Yes
D6	Rain	49	Strong	No
D7	Overcast	53	Strong	Yes
D8	Sunny	66	Weak	No
D9	Sunny	56	Weak	Yes
D10	Rain	52	Weak	Yes
D11	Sunny	44	Strong	Yes
D12	Overcast	71	Strong	Yes
D13	Overcast	45	Weak	Yes
D14	Rain	68	Strong	No

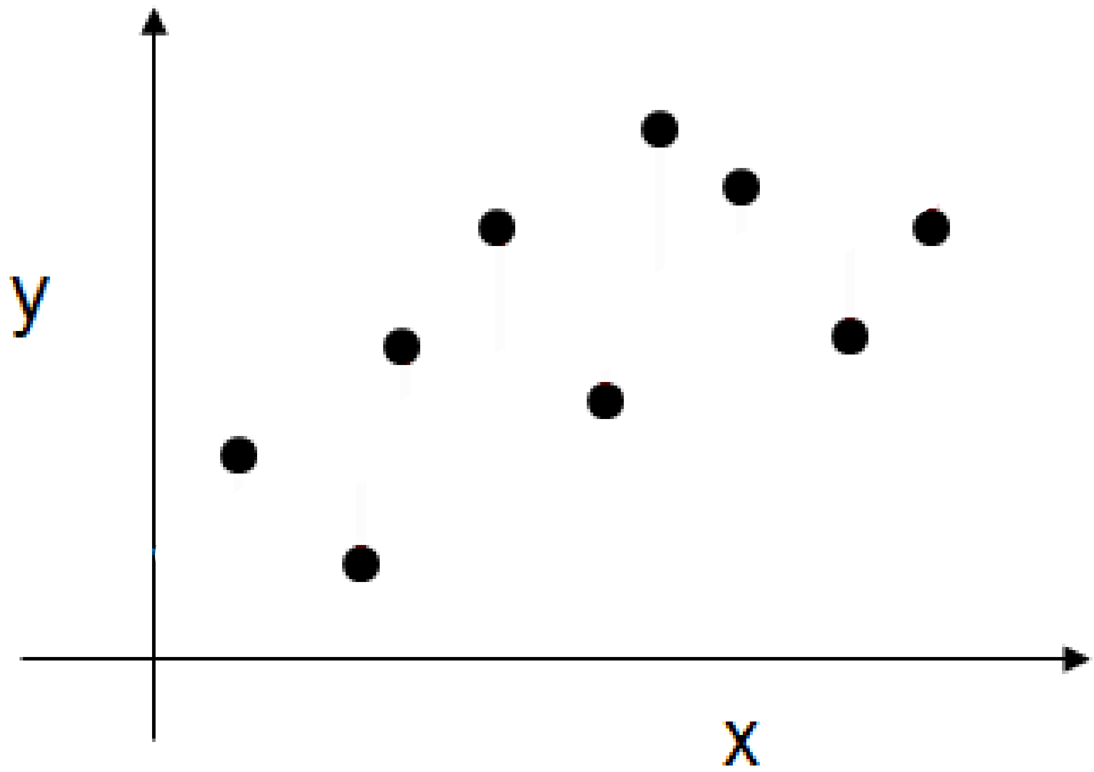

petal length (cm) ≤ 2.45
gini = 0.6667
samples = 150
value = [50, 50, 50]
class = setosa
True
False
gini = 0.0
samples = 50
value = [50, 0, 0]
class = setosa
petal width (cm) ≤ 1.75
gini = 0.5
samples = 100
value = [0, 50, 50]
class = versicolor
petal length (cm) ≤ 4.95
gini = 0.168
samples = 54
value = [0, 49, 5]
class = versicolor
petal length (cm) ≤ 4.85
gini = 0.0425
samples = 46
value = [0, 1, 45]
class = virginica
petal width (cm) ≤ 1.65
gini = 0.0408
samples = 48
value = [0, 47, 1]
class = versicolor
petal width (cm) ≤ 1.55
gini = 0.4444
samples = 6
value = [0, 2, 4]
class = virginica
sepal length (cm) ≤ 5.95
gini = 0.4444
samples = 3
value = [0, 1, 2]
class = virginica
gini = 0.0
samples = 43
value = [0, 0, 43]
class = virginica
gini = 0.0
samples = 47
value = [0, 47, 0]
class = versicolor
gini = 0.0
samples = 1
value = [0, 0, 1]
class = virginica
gini = 0.0
samples = 3
value = [0, 0, 3]
class = virginica
sepal length (cm) ≤ 6.95
gini = 0.4444
samples = 3
value = [0, 2, 1]
class = versicolor
gini = 0.0
samples = 1
value = [0, 1, 0]
class = versicolor
gini = 0.0
samples = 2
value = [0, 0, 2]
class = virginica
gini = 0.0
samples = 2
value = [0, 2, 0]
class = versicolor
gini = 0.0
samples = 1
value = [0, 0, 1]
class = virginica

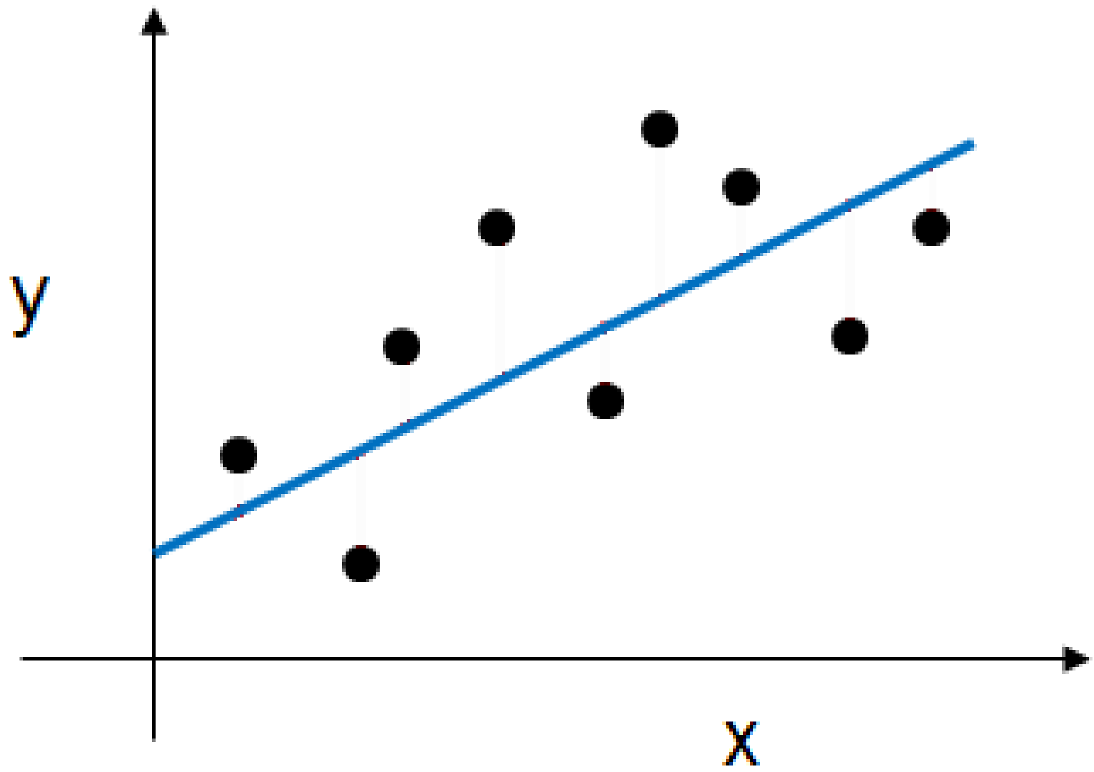
non-machine-learning uses of decision trees



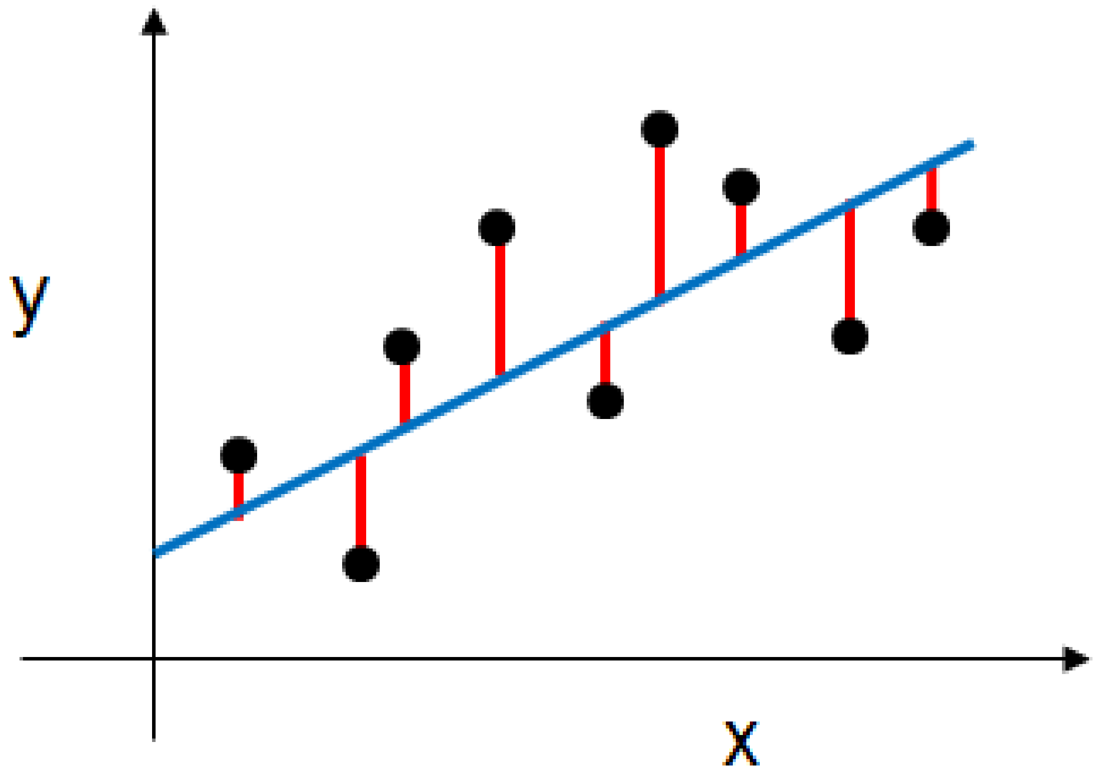
linear regression


$R^2 = 0.06$
REXTHOR, THE DOG-BEARER
I DON'T TRUST LINEAR REGRESSIONS WHEN IT'S HARDER TO GUESS THE DIRECTION OF THE CORRELATION FROM THE SCATTER PLOT THAN TO FIND NEW CONSTELLATIONS ON IT.


y
x


y
x

loss function



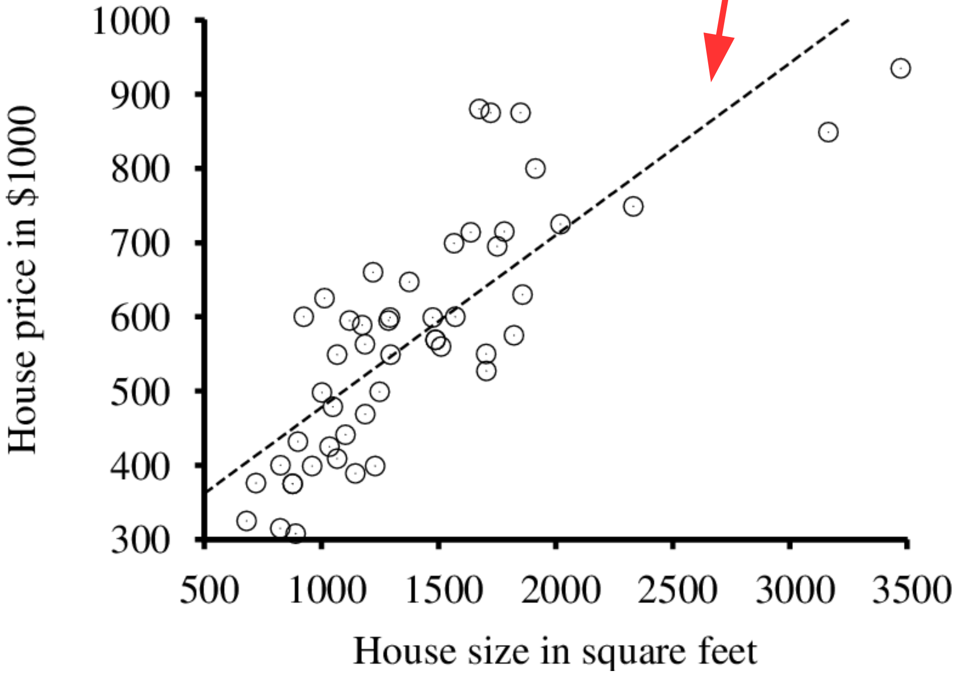
Model Prediction

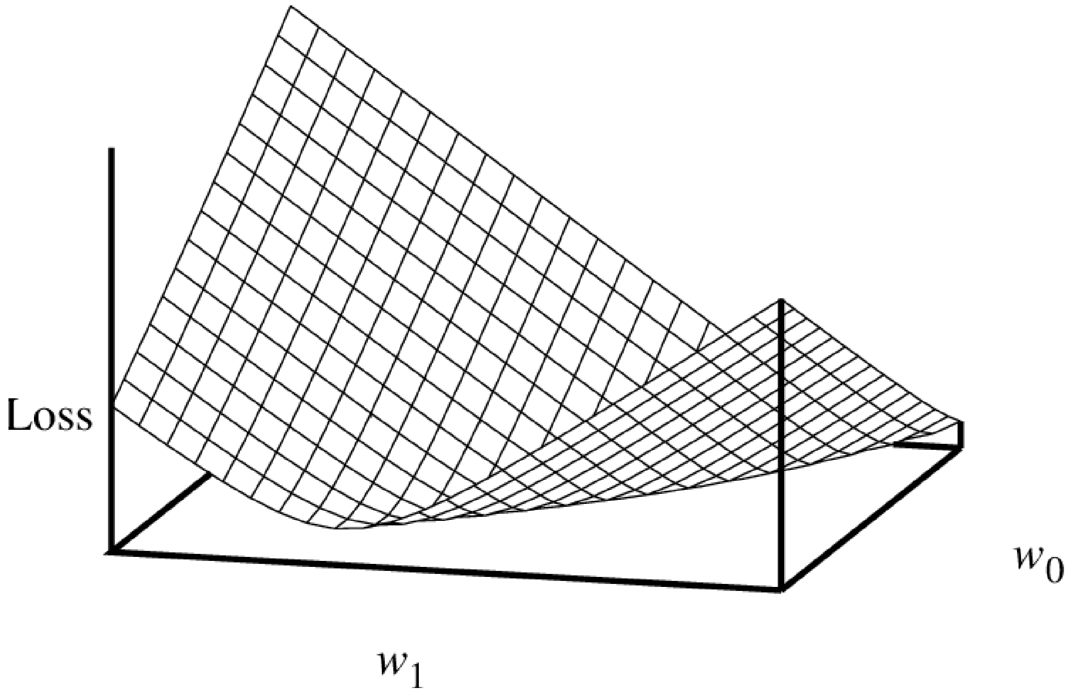
$$SS_{residuals} = \sum_{i=1}^{N} (\hat{y}_i - y_i)^2$$

Observed Result

linear model

$$h(x) = w_0 + w_1 * x$$





we want to find the minimum of this loss function

to do so, we'll move in the downhill direction of of the parameter space (along the gradients of w_0 and w_1)

Calculus review (chain rule):

$$f(x) = g(x)^n$$

$$\frac{\delta}{\delta x} f(x) = \frac{\delta}{\delta x} g(x)^n$$

$$\delta/\delta x\ f(x) = n * g(x)^{n-1} \frac{\delta}{\delta x} g(x)$$

for each (x, y):

$$Loss = (y - h(x))^2$$

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

$$\delta/\delta w_0\ Loss = \delta/\delta w_0 (y - h(x))^2$$

$$\delta/\delta w_0\ Loss = 2*(y - h(x)) * \delta/\delta w_0 (y - h(x))$$

$$\delta/\delta w_0\ Loss = 2*(y - h(x)) * \delta/\delta w_0 (y - (w_0 + w_1 * x))$$

$$\delta/\delta w_0\ Loss = 2*(y - h(x))$$

for each (x, y):

$$Loss = (y - h(x))^2$$

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

$$\delta/\delta w_1 \, Loss = \delta/\delta w_1 (y - h(x))^2$$

$$\delta/\delta w_1 \, Loss = 2*(y - h(x)) * \delta/\delta w_1 (y - h(x))$$

$$\delta/\delta w_1 \, Loss = 2*(y - h(x)) * \delta/\delta w_1 (y - (w_0 + w_1 * x))$$

$$\delta/\delta w_1 \, Loss = 2*(y - h(x)) * x$$

for each (x, y):

$$w_0 \leftarrow w_0 + \alpha * (y - h(x))$$
$$w_1 \leftarrow w_1 + \alpha * (y - h(x)) * x$$

“stochastic gradient descent”
(when you randomly choose a (x,y)

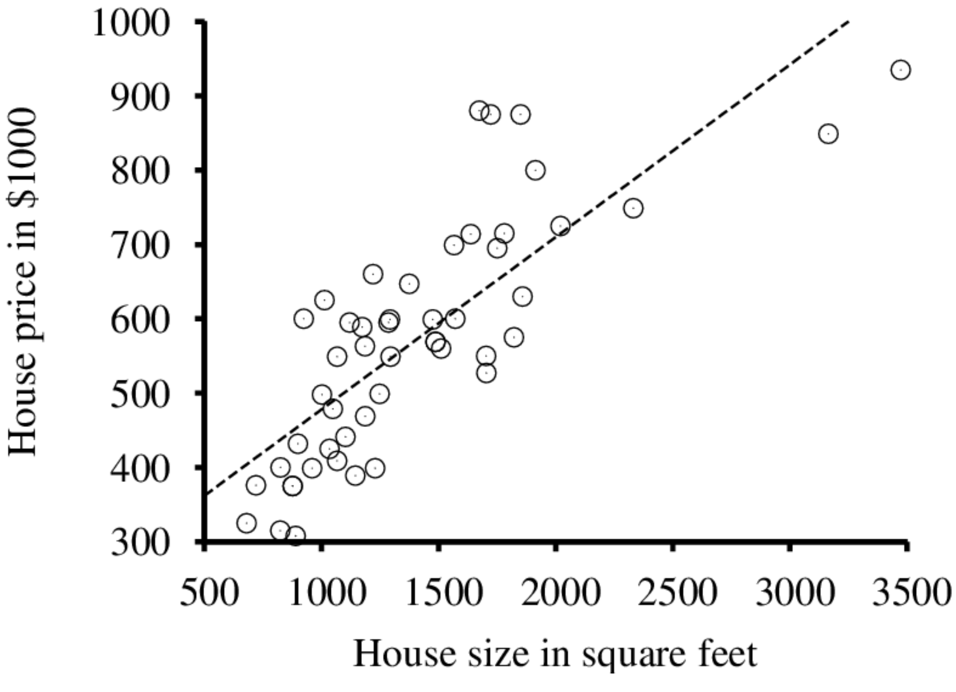
$$w_0 \leftarrow w_0 + \alpha * \Sigma_{j=1:n} (y - h(x_j))$$
$$w_1 \leftarrow w_1 + \alpha * \Sigma_{j=1:n} (y - h(x_j)) * x_j$$

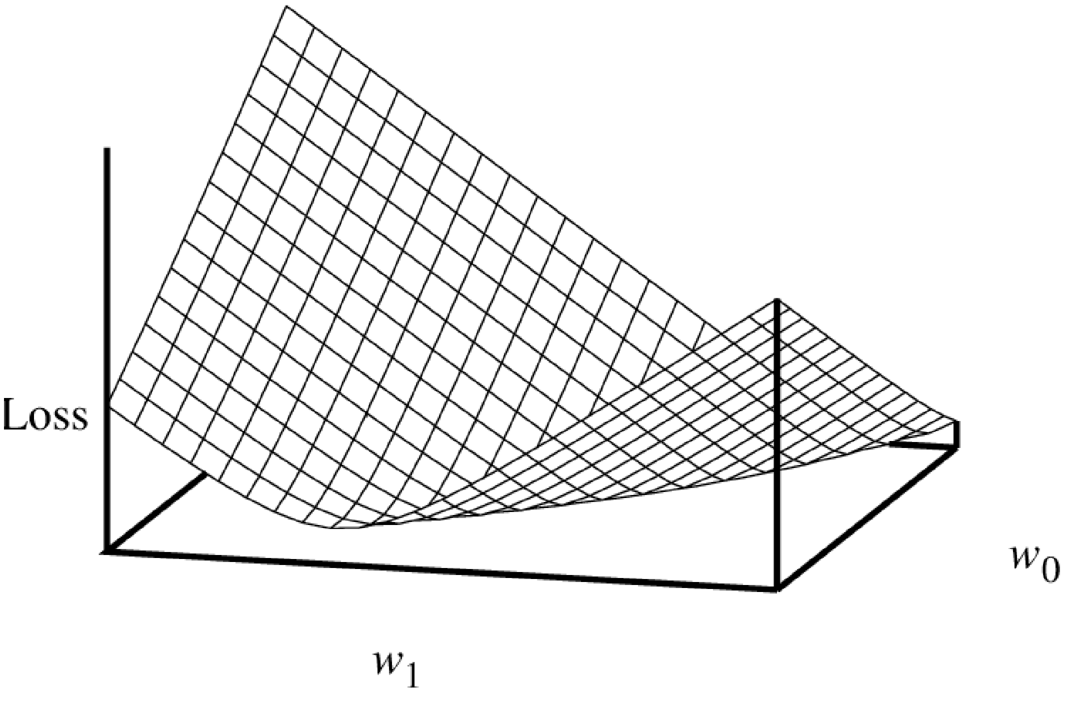
"batch gradient descent"
(when you use all (x,y) pairs for one update)

the derivative of a sum is the sum of the derivatives batch gradient descent prevent overfitting to any data point

stochastic gradient descent runs more efficiently

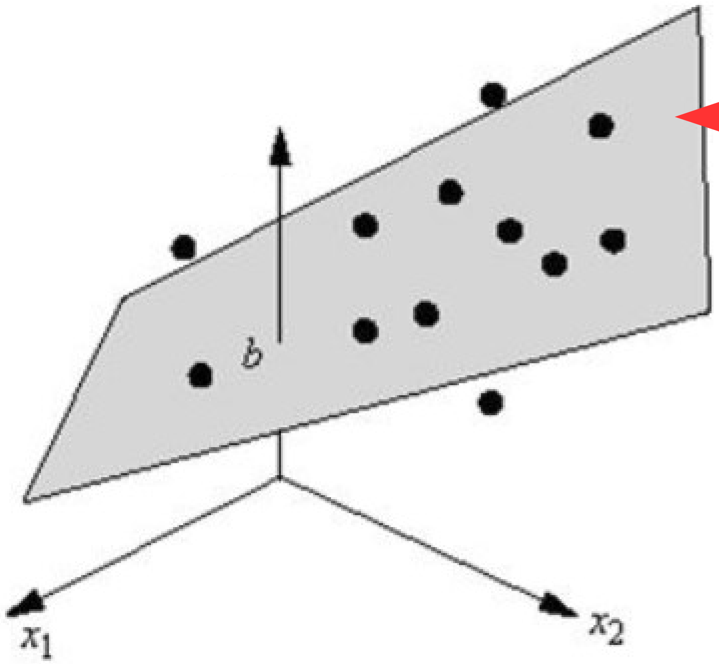
stochastic GD or “mini batches” used in practice





python example!

multivariate linear regression



$$h(x) = w_0 + w_1 * x_1 + w_2 * x_2 + \ldots$$

$$h(x_j) = \Sigma_{i=1:n}(w_i * x_j) = \overbrace{w^T x_j}^{\text{dot product}}$$

$$h(x) = 4.5x_0 + 5.1x_1 + 8.3x_2 + 7.9x_3 + 3.6x_4$$

$$h(x) = [\ 4.5\ \ 5.1\ \ 8.3\ \ 7.9\ \ 3.6\ ] \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

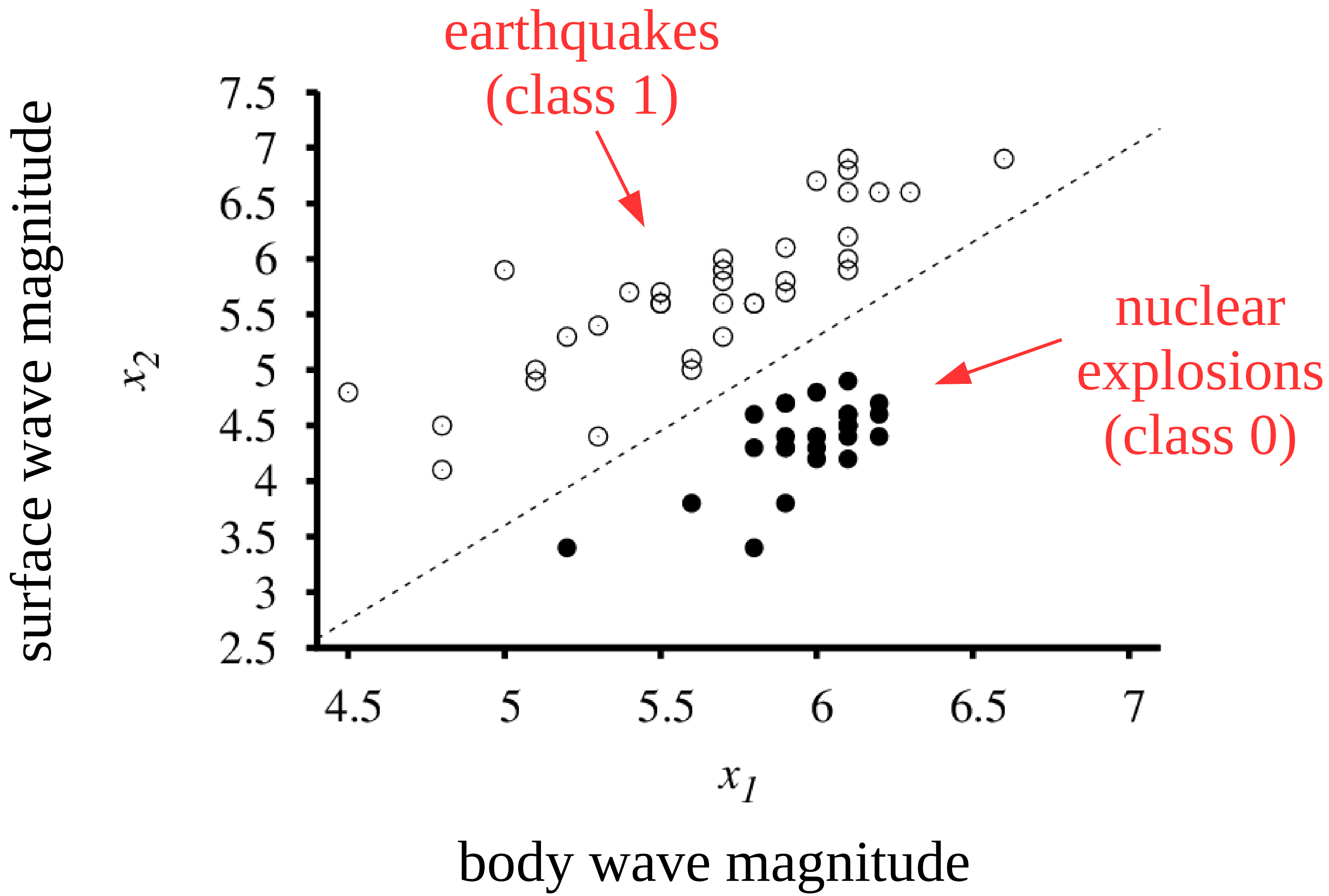
$$w_i \leftarrow w_i + \alpha * \Sigma_{j=1:n} (y_j - h(x_j)) * x_{ji}$$

(assuming all x_0 entries are 1)

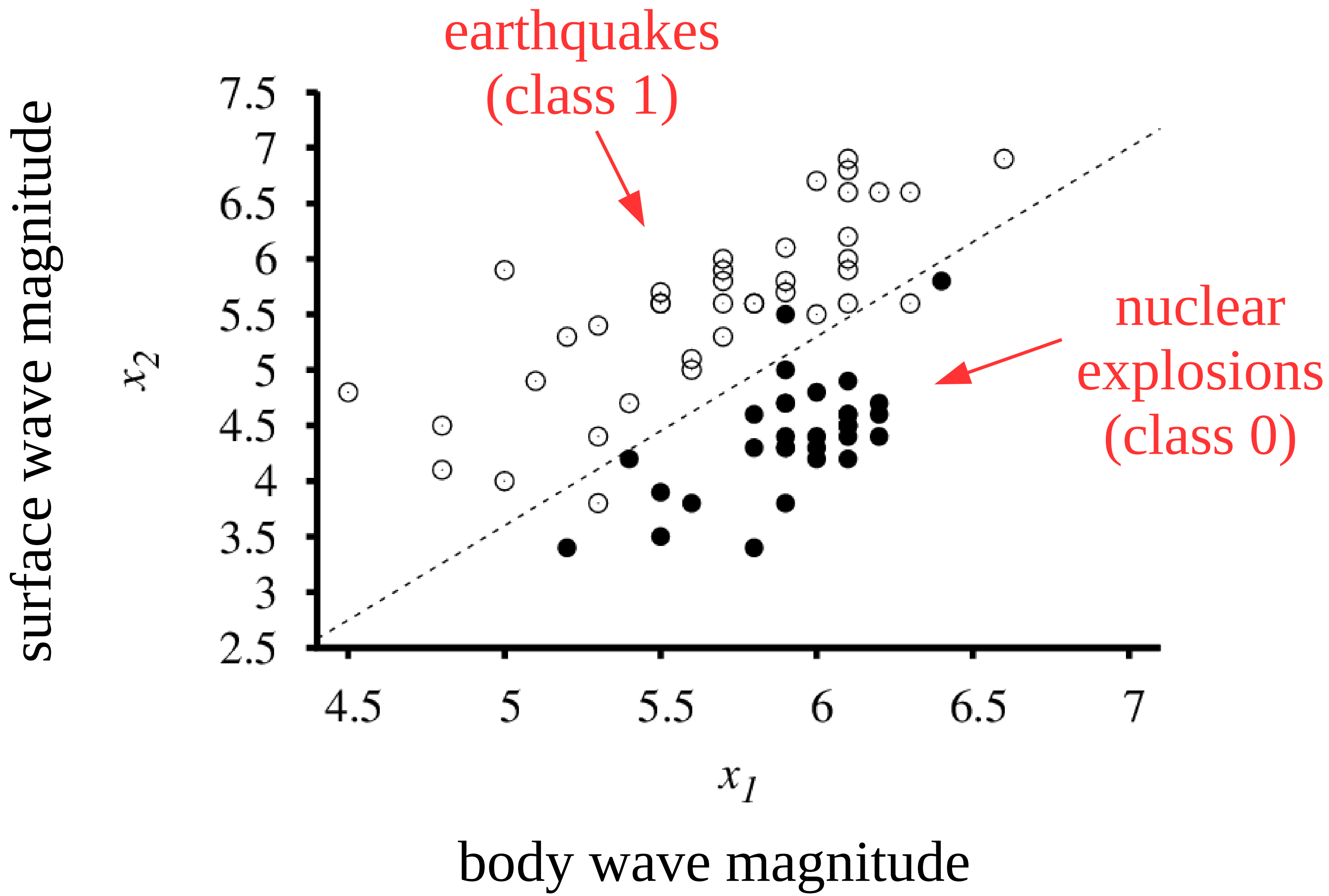
$$w_i \leftarrow w_i + \alpha (y - h(x)) * x_i$$

linear classifiers

Seismic events from 1982 to 1990 in Asia and the Middle East



Seismic events from 1982 to 1990 in Asia and the Middle East



just like before, we are still trying to find the parameters for a straight line that best “fits” the data

now “fit” is the line that best separates the classes, rather than tracks the data points (“linear separator”)

more generally called a “decision boundary” we need a function to turn the outputs of our linear function into class labels

$$\text{Threshold}(z) = \begin{cases} 1 \text{ if } z >= 0 \\ 0 \text{ if } z < 0 \end{cases}$$

$$w_i \leftarrow w_i + \alpha\ (y - h(x)) * x_i$$

where $h(x) = \text{Threshold}(w \circ x)$