

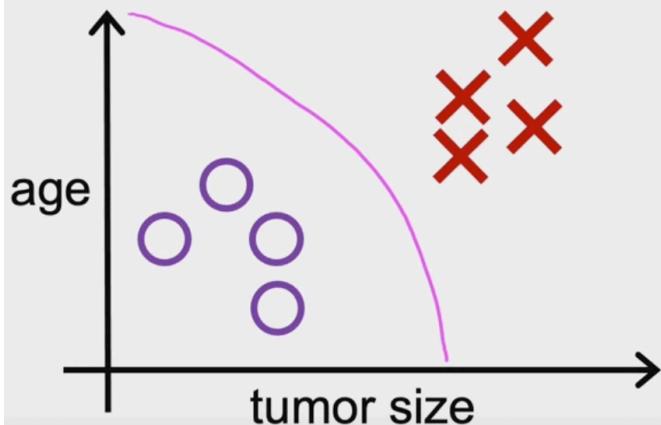
Supervised Machine Learning

Supervised Learning:

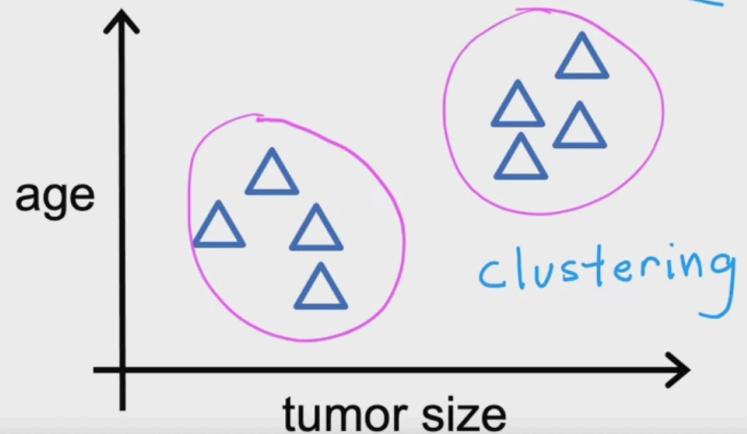
Regression (House Prices) - Classification (Breast Cancer Detection)

Unsupervised Learning:

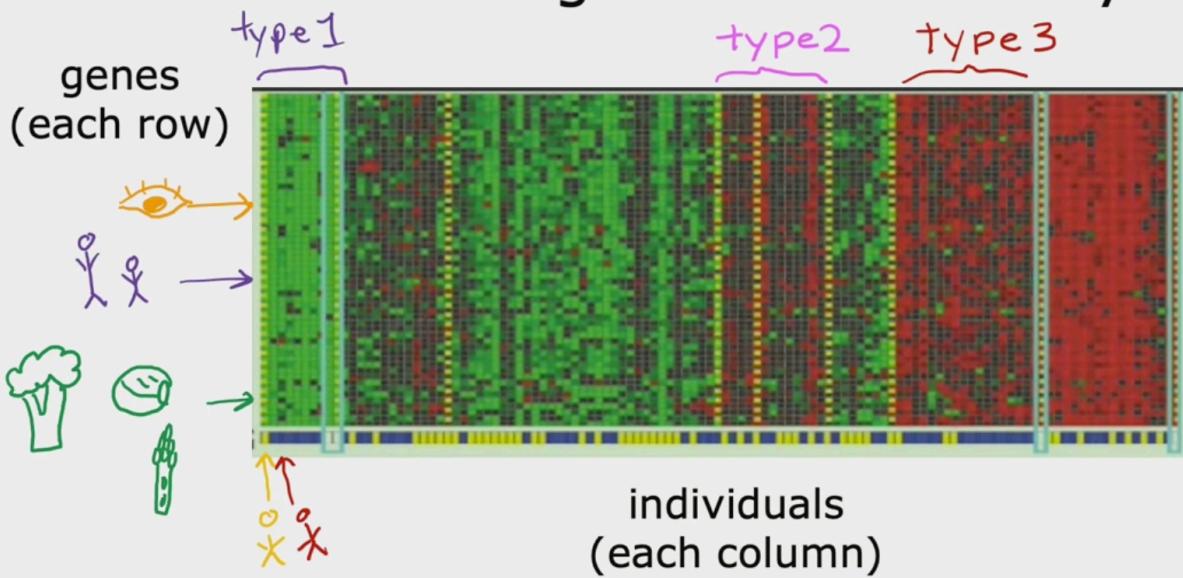
Supervised learning
Learn from data **labeled**
with the "**right answers**"



Unsupervised learning
Find something interesting
in **unlabeled** data.

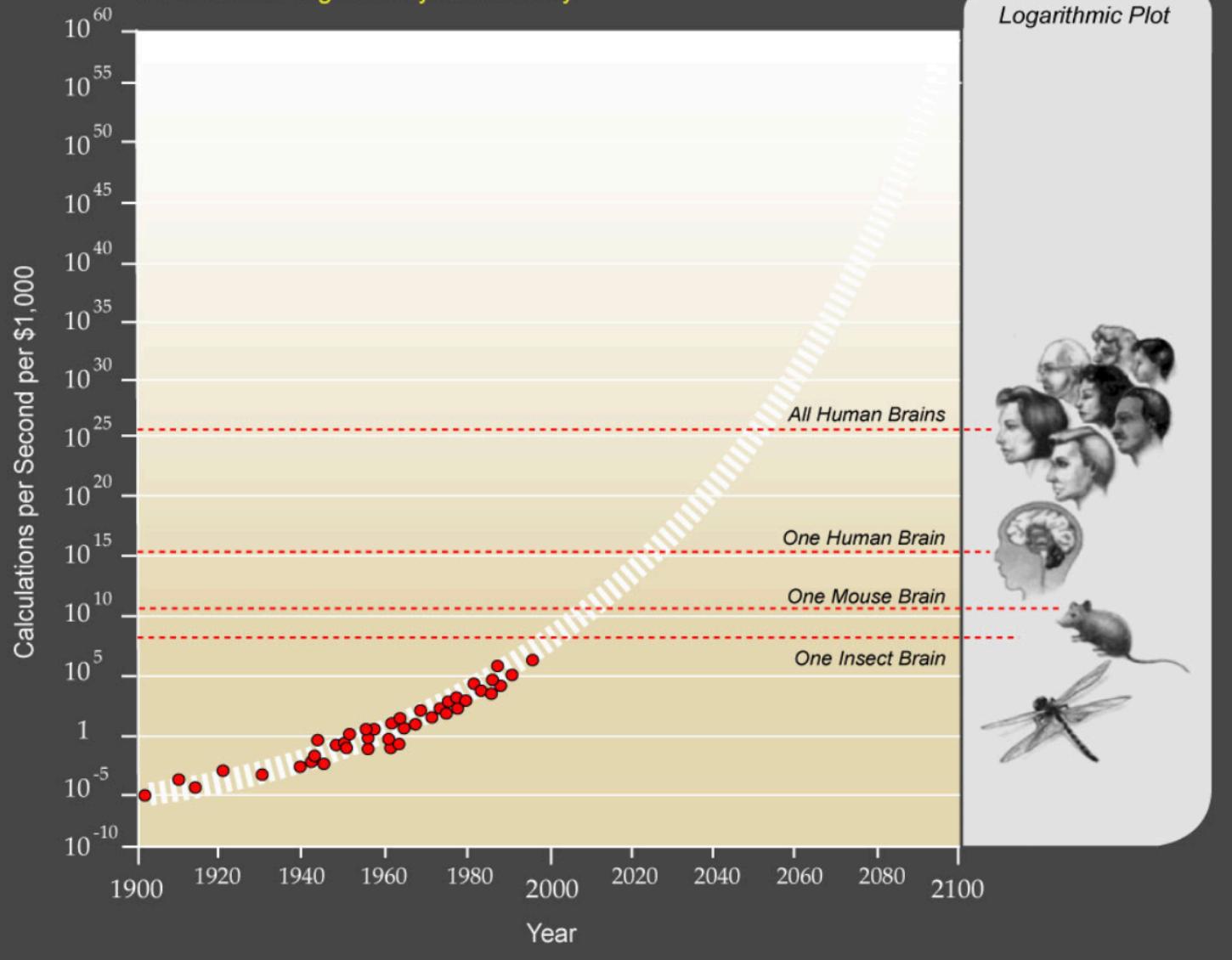


Clustering: DNA microarray



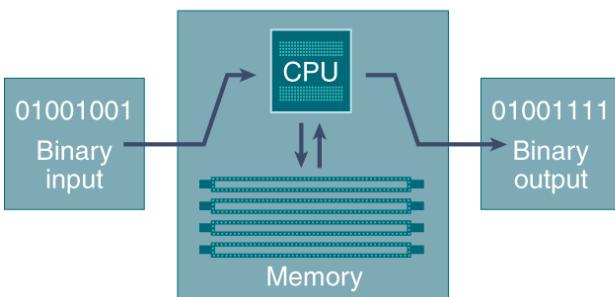
Exponential Growth of Computing

Twentieth through twenty first century



Neuromorphic Computing

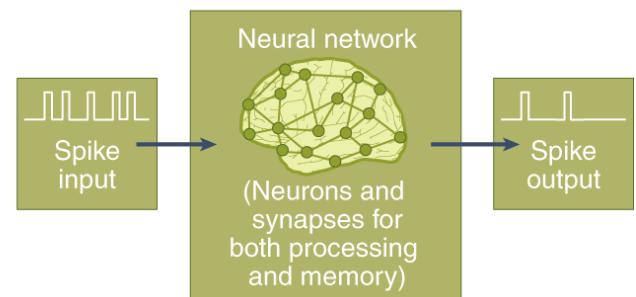
Von Neumann architecture



Sequential processing
 Separated computation and memory
 Code as binary instructions
 Binary data
 Synchronous (clock-driven)

versus

Neuromorphic architecture



Operation Organization
 Programming
 Communication
 Timing

Massively parallel processing
 Collocated processing and memory
 Spiking neural network
 Spikes
 Asynchronous (event-driven)

Architecture of AlexNet

$$f = \text{np.dot}(w, x) + b$$

Cost function:

$$j(w^*, b)$$

$$j(w_1, w_2, \dots, w_n, b)$$

Gradient Descent

```

repeat {
     $w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(\underbrace{w_1, \dots, w_n}_b, b)$ 
     $b = b - \alpha \frac{\partial}{\partial b} J(\underbrace{w_1, \dots, w_n}_b, b)$ 
}

```

```

repeat {
     $w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(\vec{w}, b)$ 
     $b = b - \alpha \frac{\partial}{\partial b} J(\vec{w}, b)$ 
}

```

gradient descent for multiple regression

Gradient descent

One feature

```

repeat {
     $w = w - \alpha \frac{1}{m} \sum_{i=1}^m (f_{w,b}(x^{(i)}) - y^{(i)}) x^{(i)}$ 
     $\hookrightarrow \frac{\partial}{\partial w} J(w, b)$ 
     $b = b - \alpha \frac{1}{m} \sum_{i=1}^m (f_{w,b}(x^{(i)}) - y^{(i)})$ 
    simultaneously update  $w, b$ 
}

```

n features ($n \geq 2$)

```

repeat {
     $j=1$   $w_1 = w_1 - \alpha \frac{1}{m} \sum_{i=1}^m (f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}) x_1^{(i)}$ 
    :
     $j=n$   $w_n = w_n - \alpha \frac{1}{m} \sum_{i=1}^m (f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}) x_n^{(i)}$ 
     $b = b - \alpha \frac{1}{m} \sum_{i=1}^m (f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)})$ 
    simultaneously update  $w_j$  (for  $j = 1, \dots, n$ ) and  $b$ 
}

```

gradient descent to find parameter values:

$$ax+by+cz+k$$

finds a,b,c.

upside down hill. finds the bottom.

rescaling the parameters

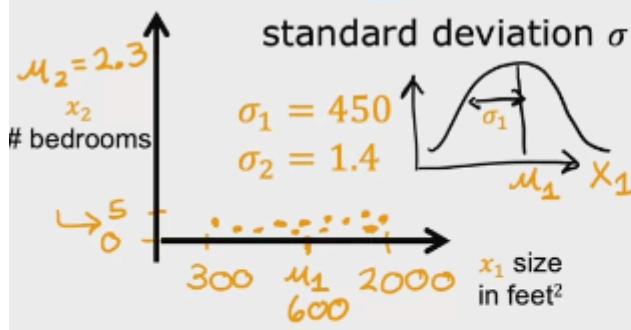
x between 300 and 2k, y between 0 and 5:

Option 1: x/2k, y/5

Option 2: Find mean (average), $(x - \text{mean})/(2000 - \text{mean}/2)$ and so on.

Option 3: Z score normalization

Z-score normalization



$$300 \leq x_1 \leq 2000 \quad 0 \leq x_2 \leq 5$$

$$x_1 = \frac{x_1 - \mu_1}{\sigma_1} \quad x_2 = \frac{x_2 - \mu_2}{\sigma_2}$$

$$-0.67 \leq x_1 \leq 3.1 \quad -1.6 \leq x_2 \leq 1.9$$

Best practice, between -1 and 1, or -n and n.

Do not put too much size differences between the parameters:

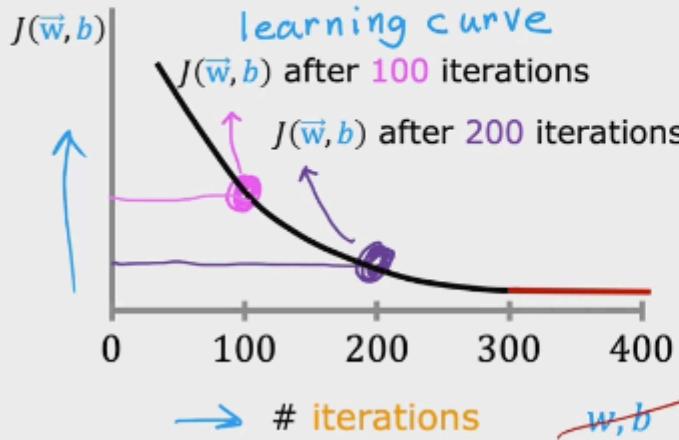
x1 0 and 3

x2 -0.5 and 2

x3 -100 and 100 Not good

Make sure gradient descent is working correctly

objective: $\min_{\vec{w}, b} J(\vec{w}, b)$ $J(\vec{w}, b)$ should decrease after every iteration

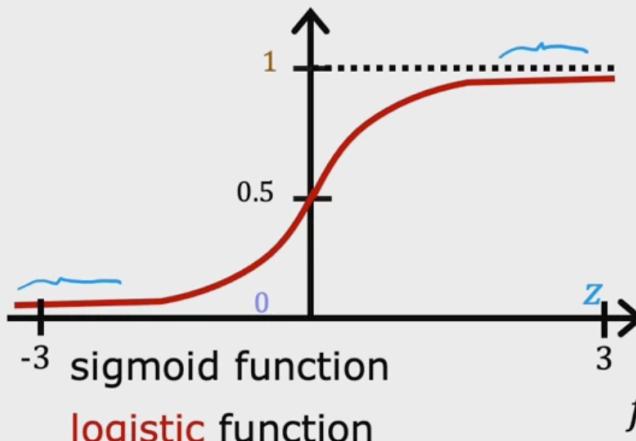


J - cost function should be decreasing

choose of learning rate, (powers of 10)

Logistic Regression Model:

Want outputs between 0 and 1



$f_{\vec{w}, b}(\vec{x})$

$$z = \vec{w} \cdot \vec{x} + b$$

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

1

$$f_{\vec{w}, b}(\vec{x}) = g(\underbrace{\vec{w} \cdot \vec{x} + b}_z) = \frac{1}{1+e^{-(\vec{w} \cdot \vec{x} + b)}}$$

"logistic regression"

outputs between 0 and 1

$$g(z) = \frac{1}{1+e^{-z}} \quad 0 < g(z) < 1$$

probability that the things is "1"

Linear regression

$$f_{\vec{w}, b}(\vec{x}) = \vec{w} \cdot \vec{x} + b$$

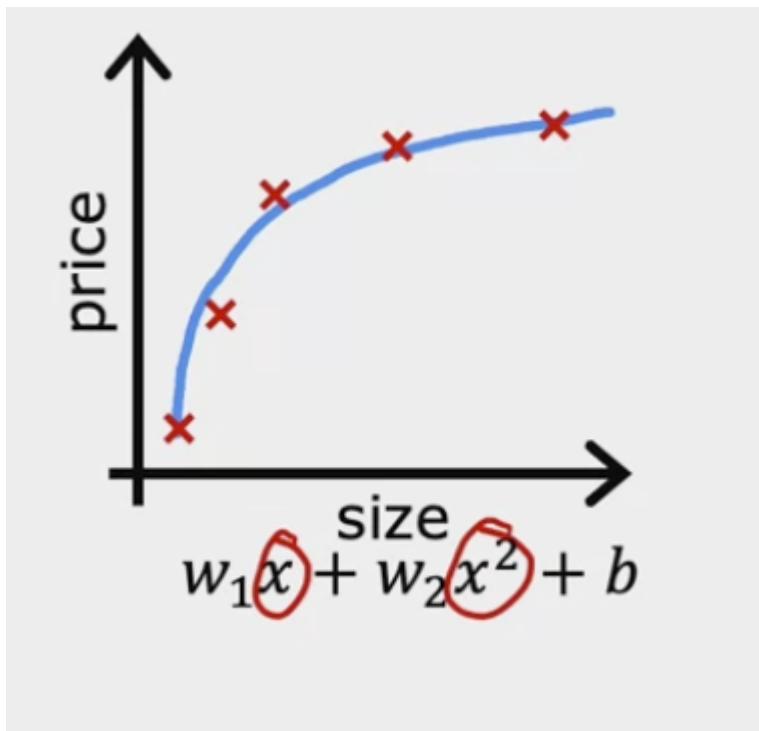
Logistic regression

$$f_{\vec{w}, b}(\vec{x}) = \frac{1}{1+e^{-(\vec{w} \cdot \vec{x} + b)}}$$

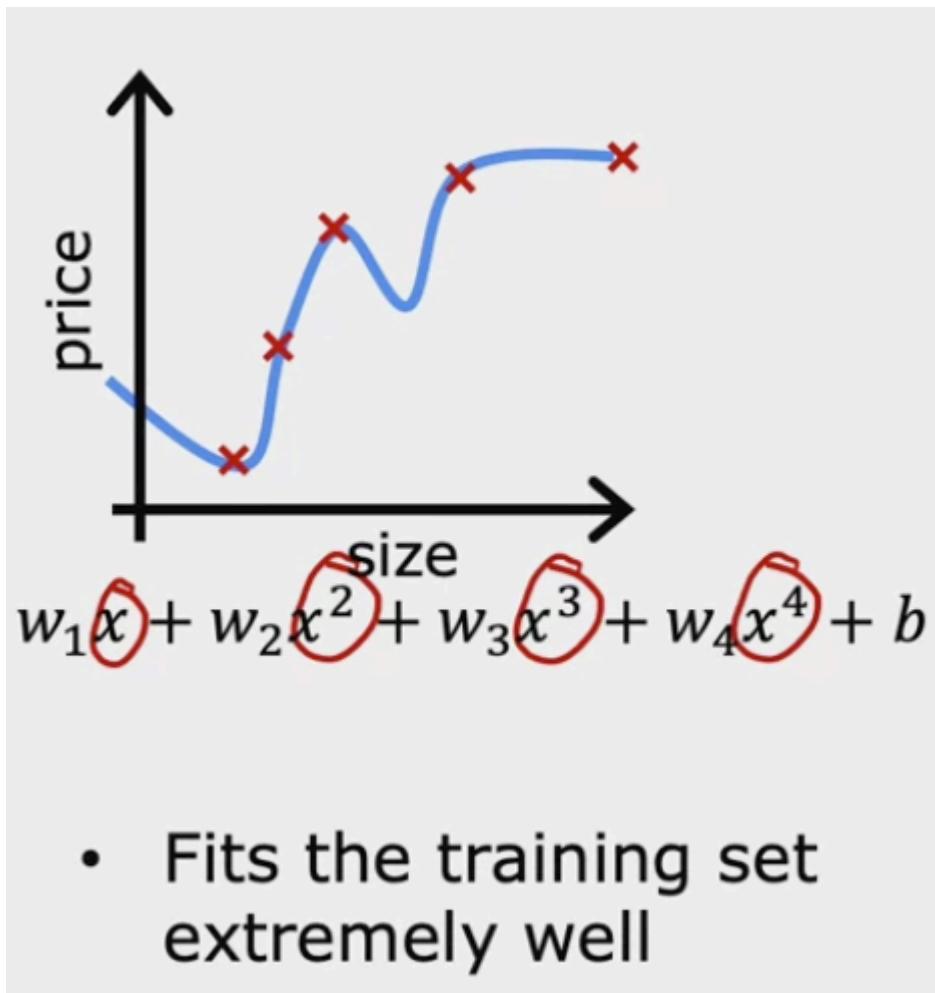
how they can help gradient

Regularization: helps minimize overfitting.

Generalization: fit to data not in train set. well trained.



Overfitting:



Underfitting: not able to fit to the training set.

Regression example



Too few parameteres: underfit, too many parameters: overfit.

Collect more data.

Feature selection - to prevent overfitting.

Regularization: Reduce the size of the parameters w_1, w_2, w_3, \dots

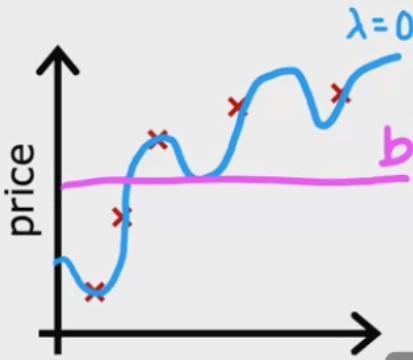
if you can multiply these parameters with large numbers, than they are negligible.

lambda is the value for heaviness of regularization.

Regularization

$$\min_{\vec{w}, b} J(\vec{w}, b) = \min_{\vec{w}, b} \left[\underbrace{\frac{1}{2m} \sum_{i=1}^m (f_{\vec{w}, b}(\vec{x}^{(i)}) - y^{(i)})^2}_{\text{mean squared error}} + \underbrace{\frac{\lambda}{2m} \sum_{j=1}^n w_j^2}_{\text{regularization term}} \right]$$

fit data ↗ Keep w_j small



choose $\lambda = 10^{10}$

$$f_{\vec{w}, b}(\vec{x}) = \underbrace{w_1 x}_{\approx 0} + \underbrace{w_2 x^2}_{\approx 0} + \underbrace{w_3 x^3}_{\approx 0} + \underbrace{w_4 x^4}_{\approx 0} + b$$

$$f(x) = b$$

choose λ

appropriately balances these first and second terms of trading off.