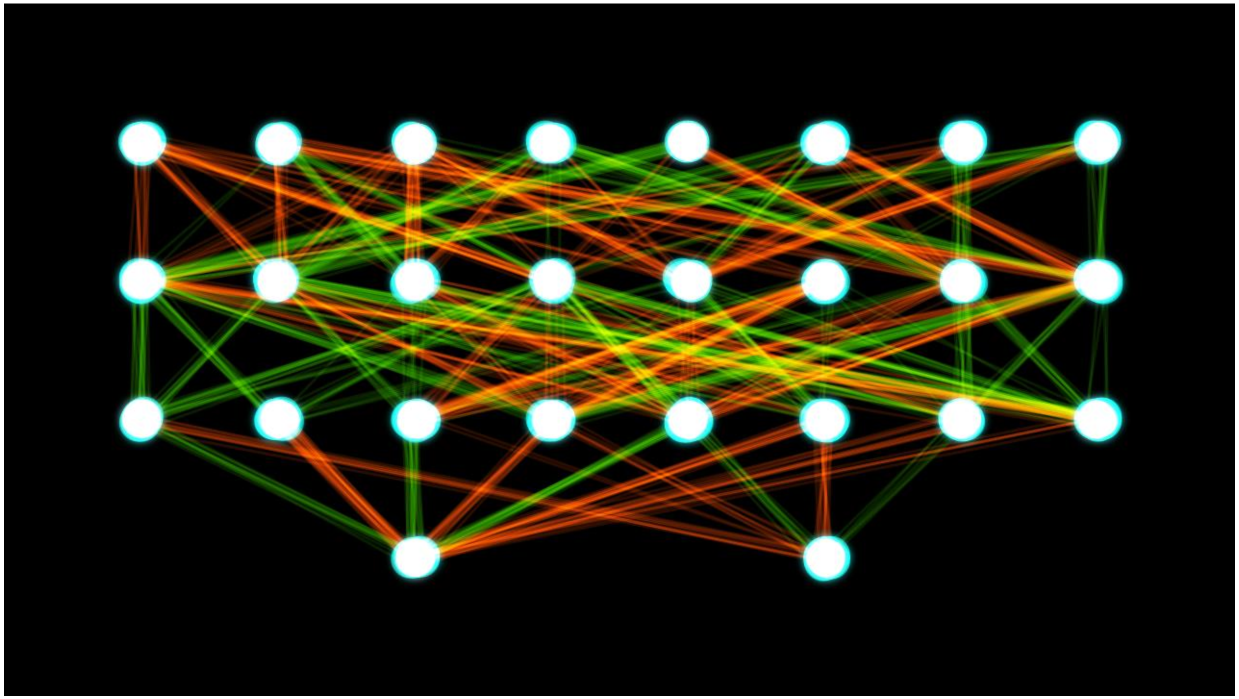
Artificial Neural Networks

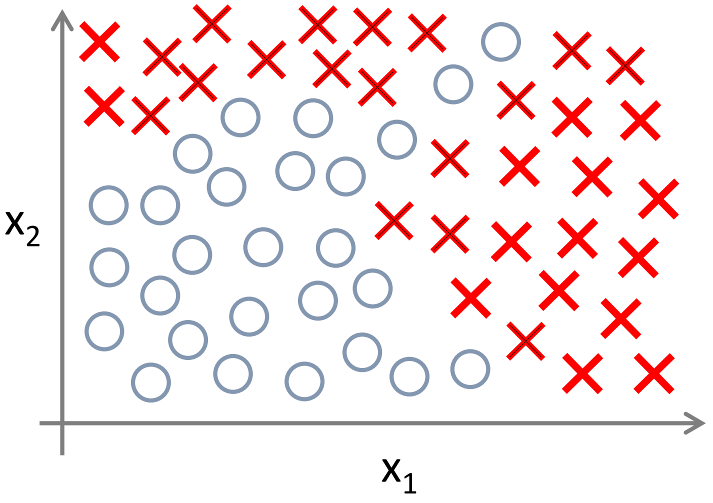


Slides from Andrew Ng, Geoffrey Hinton, and Tom Mitchell

CSC321: Intro to Machine Learning and Neural Networks, Winter 2016

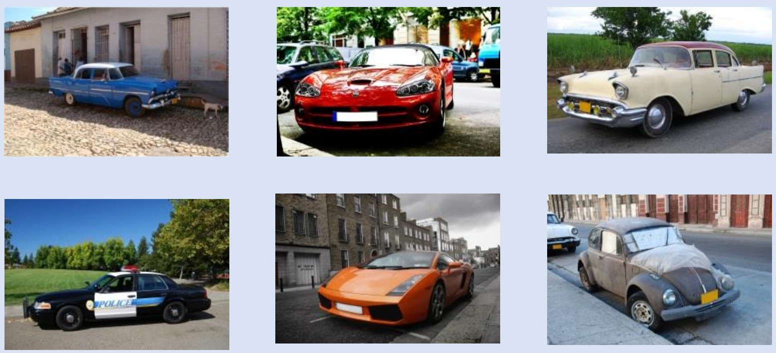
Michael Guerzhoy

Non-Linear Decision Surfaces

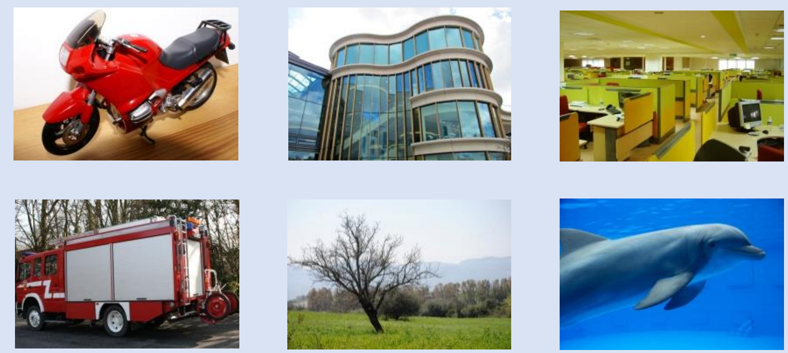


- There is no linear decision boundary

Car Classification



Cars

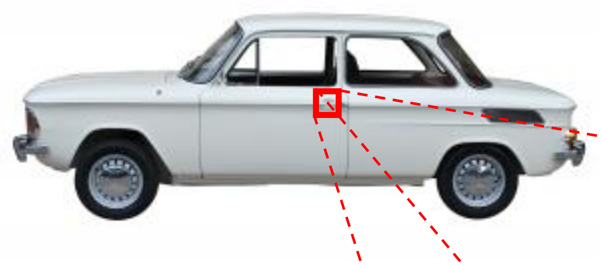


Not a car

Testing: 

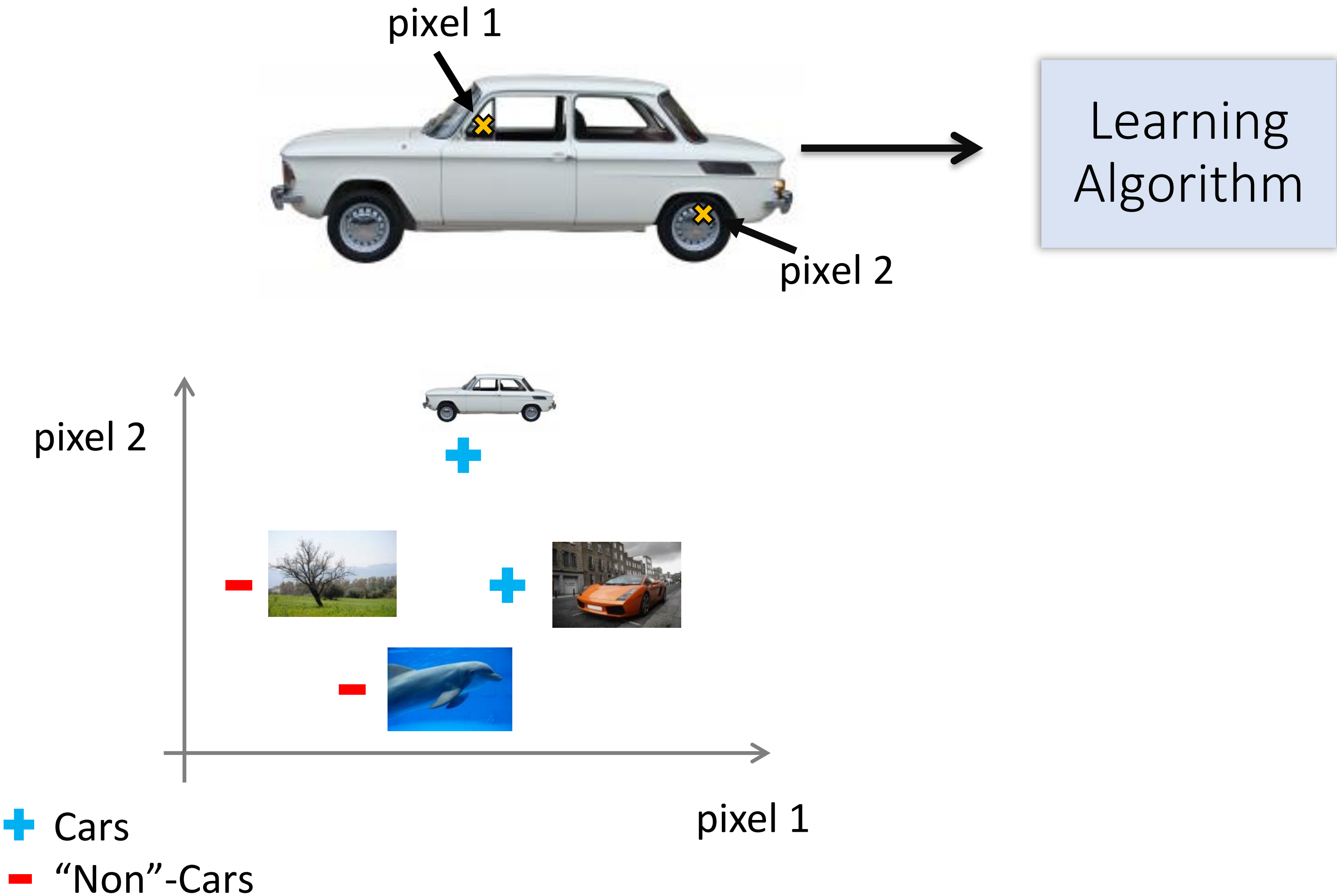
What is this?

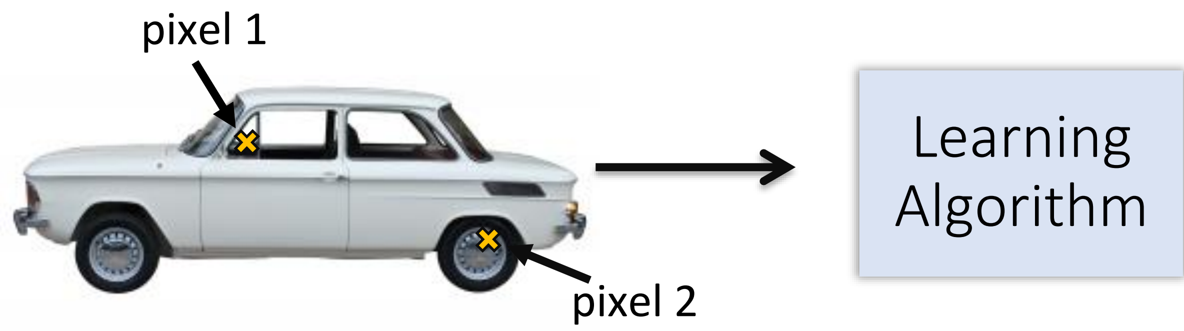
You see this:

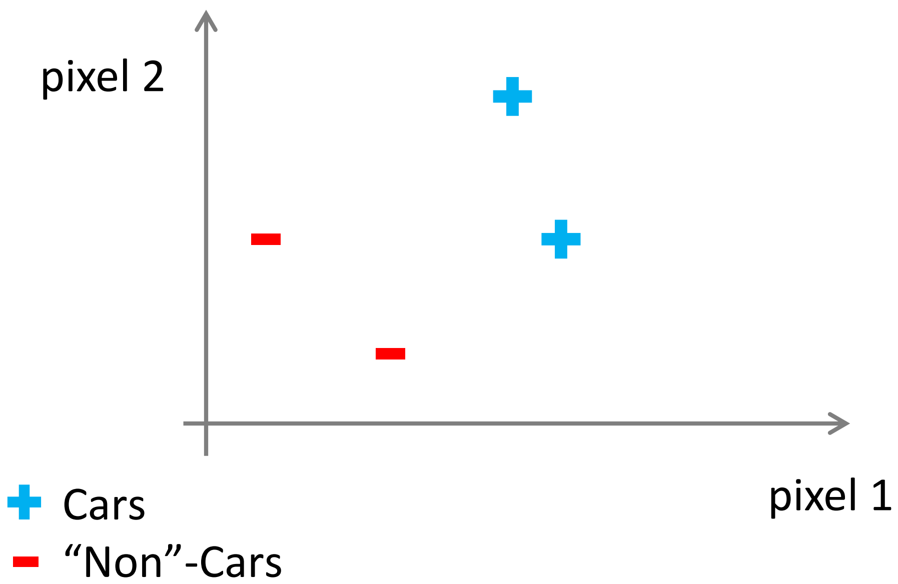


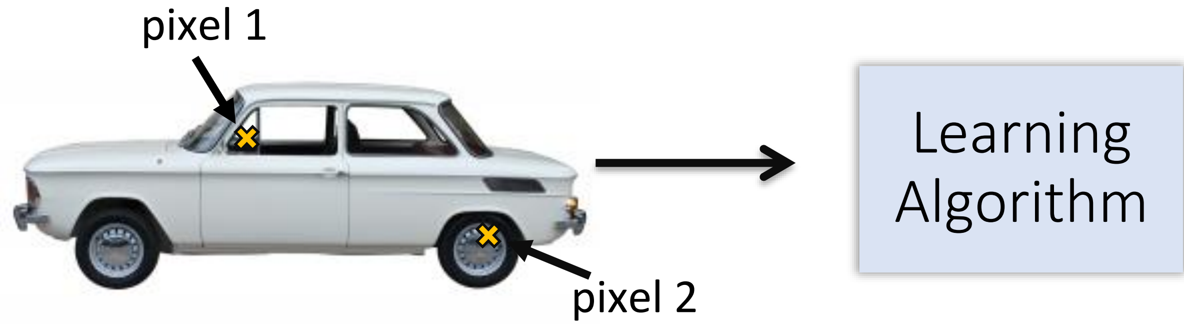
But the camera sees this:

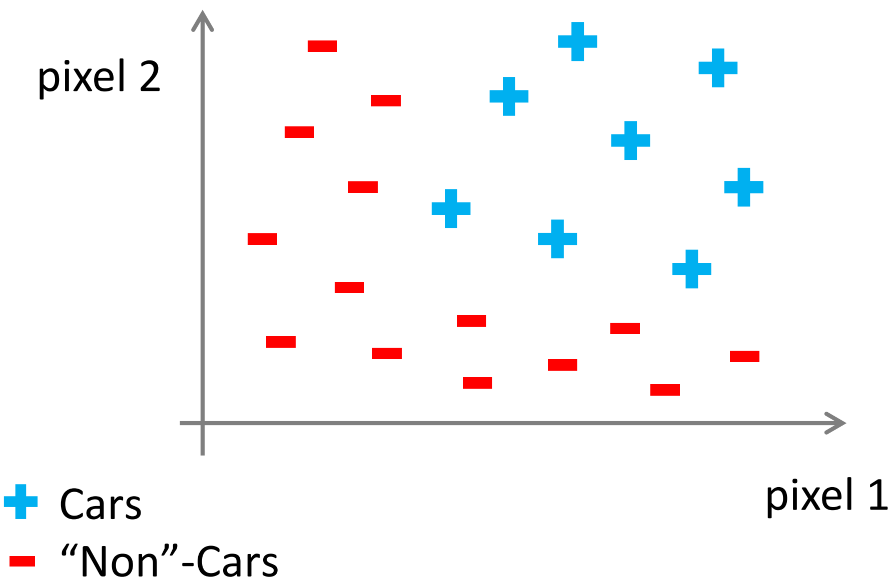
194	210	201	212	199	213	215	195	178	158	182	209
180	189	190	221	209	205	191	167	147	115	129	163
114	126	140	188	176	165	152	140	170	106	78	88
87	103	115	154	143	142	149	153	173	101	57	57
102	112	106	131	122	138	152	147	128	84	58	66
94	95	79	104	105	124	129	113	107	87	69	67
68	71	69	98	89	92	98	95	89	88	76	67
41	56	68	99	63	45	60	82	58	76	75	65
20	43	69	75	56	41	51	73	55	70	63	44
50	50	57	69	75	75	73	74	53	68	59	37
72	59	53	66	84	92	84	74	57	72	63	42
67	61	58	65	75	78	76	73	59	75	69	50


pixel 1
pixel 2
Learning Algorithm
pixel 2
pixel 1
Cars
"Non"-Cars


pixel 1
pixel 2
Learning
Algorithm


pixel 2
pixel 1
Cars
“Non”-Cars





50 x 50 pixel images→ 2500 pixels

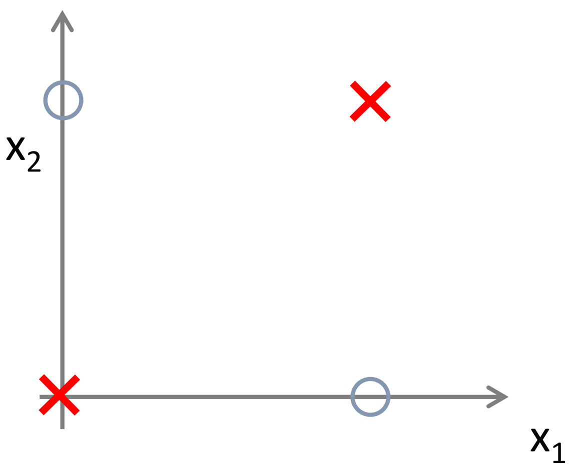
$n = 2500$ (7500 if RGB)

$$x = \begin{bmatrix} \text{pixel 1 intensity} \\ \text{pixel 2 intensity} \\ \vdots \\ \text{pixel 2500 intensity} \end{bmatrix}$$

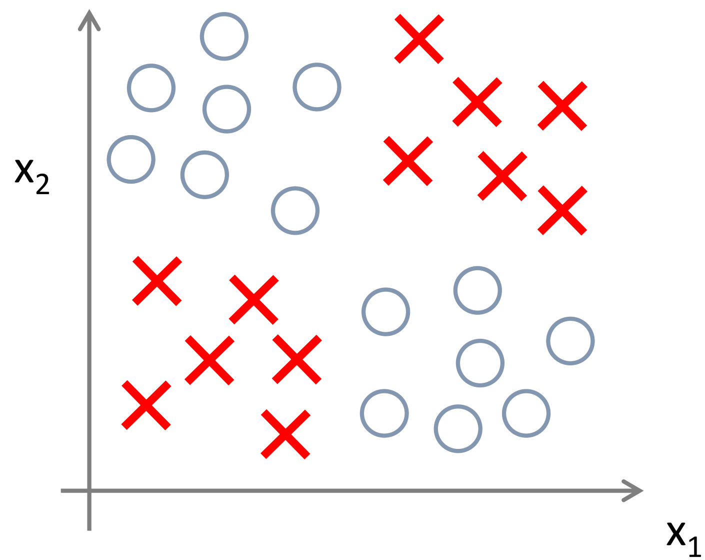
Quadratic features ($x_i \times x_j$): ≈3 million features

Simple Non-Linear Classification Example

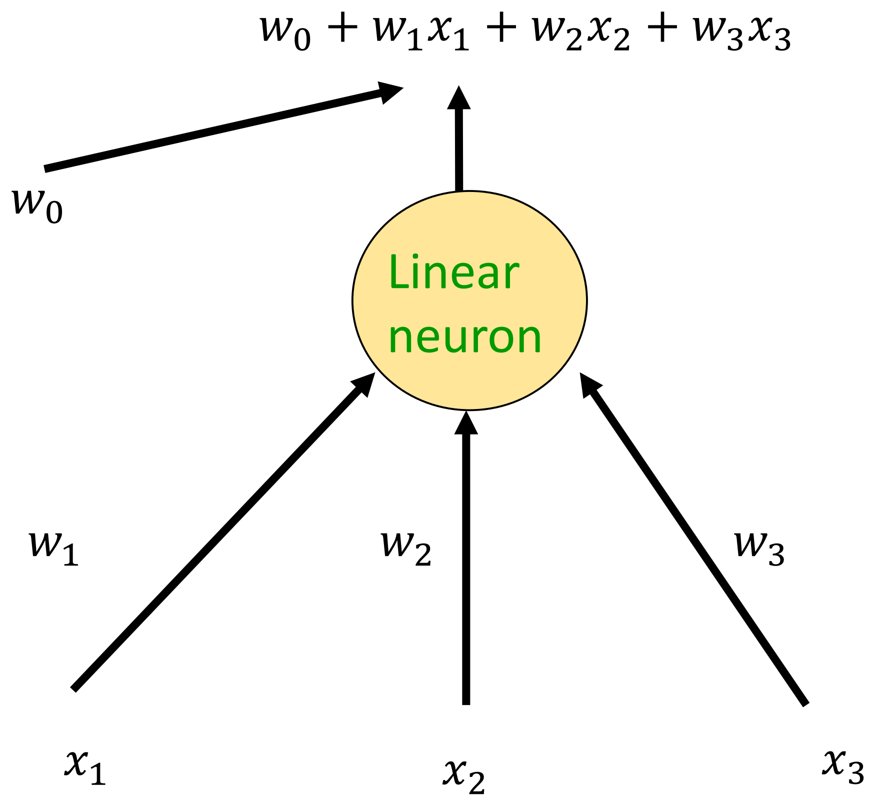
$x_1 \quad x_2$



$$y = x_1 \text{ XOR } x_2$$



Linear Neuron



Linear Neuron: Cost Function

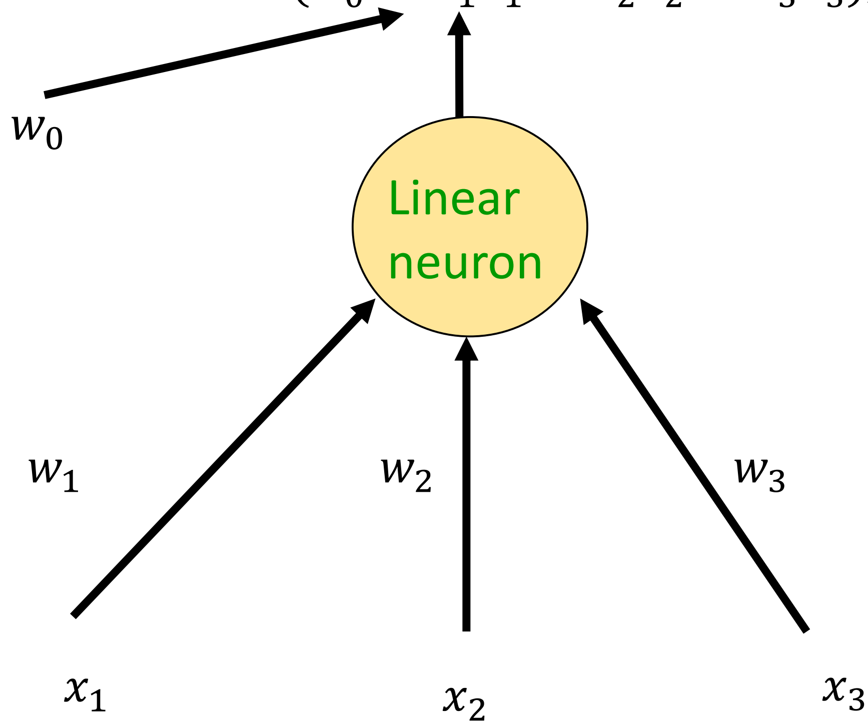
- Any number of choices. The one made for linear regression is

$$\sum_{i=1}^{m}\left(y^{(i)} - w^{T}x^{(i)}\right)^{2}$$

- Can minimize using gradient descent to obtain the best weights *w* for the training set

Logistic Neuron

$$\sigma(w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3), \sigma(t) = \frac{1}{1 + \exp(-t)}$$



Logistic Neuron: Cost Function

- Could use the quadratic cost function again
- Could use the “log-loss” function to make the neuron perform logistic regression

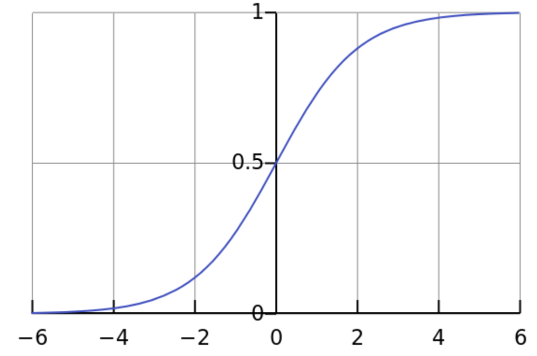
$$-\left(\sum_{i=1}^{m} y^{(i)} \log\left(\frac{1}{1+\exp(-\theta^T x^{(i)})}\right) + (1-y^{(i)}) \log\left(\frac{\exp\left(-\theta^T x^{(i)}\right)}{1+\exp(-\theta^T x^{(i)})}\right)\right)$$

(Note: we derived this cost function by saying we want to maximize the likelihood of the data under a certain model, but there’s nothing stopping us from just making up a loss function)

Logistic Regression Cost Function: Another Look

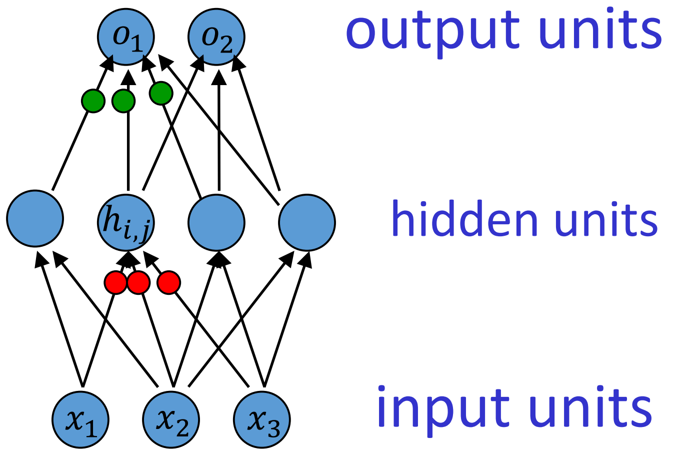
- $Cost(h_w(x), y) = \begin{cases} -\log(h_w(x)), y = 1 \\ -\log(1 - h_w(x)), y = 0 \end{cases}$
- If y = 1, want the cost to be small if $h_w(x)$ is close to 1 and large if $h_w(x)$ is close to 0
  - -log(t) is 0 for t=1 and infinity for t = 0
- If y = 0, want the cost to be small if $h_w(x)$ is close to 0 and large if $h_w(x)$ is close to 1
- Note:$0 < \sigma(t) < 1$

$$\sigma(t) = \frac{1}{1 + \exp(-t)}$$

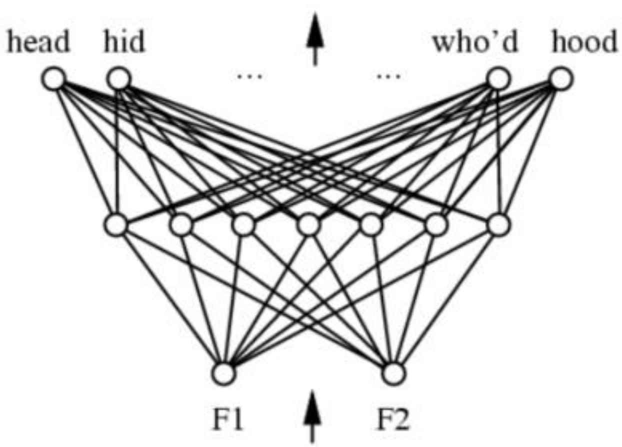


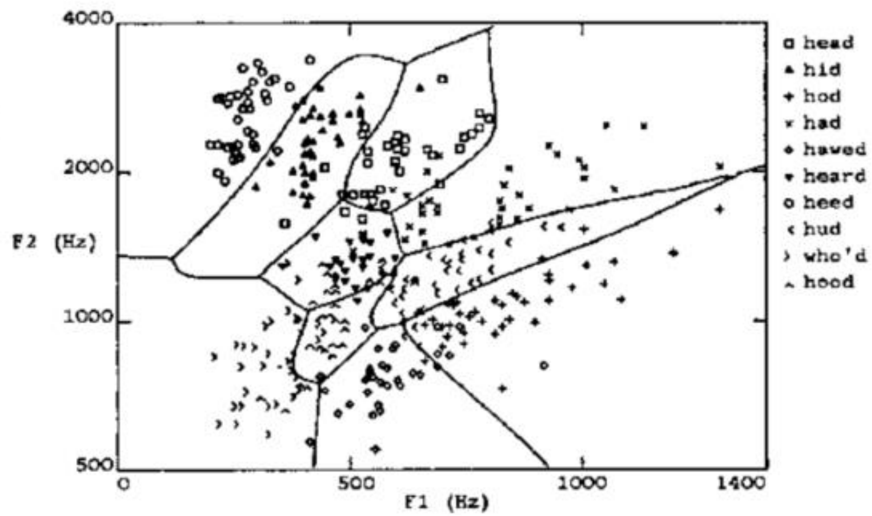
Multilayer Neural Networks

- $h_{i,j} = \mathrm{g}(\mathrm{W}_{i,j}x)$
  $= g(\sum_k W_{i,j,k}x_k)$
- $x_0 = 1$ always
- $W_{i,j,0}$ is the "bias"
- g is the **activation function**
  - Could be $g(t) = t$
  - Could be $g(t) = \sigma(t)$
    - Nobody uses those anymore…



Multilayer Neural Network: Speech Recognition Example





How to compute AND?

How to compute XOR?