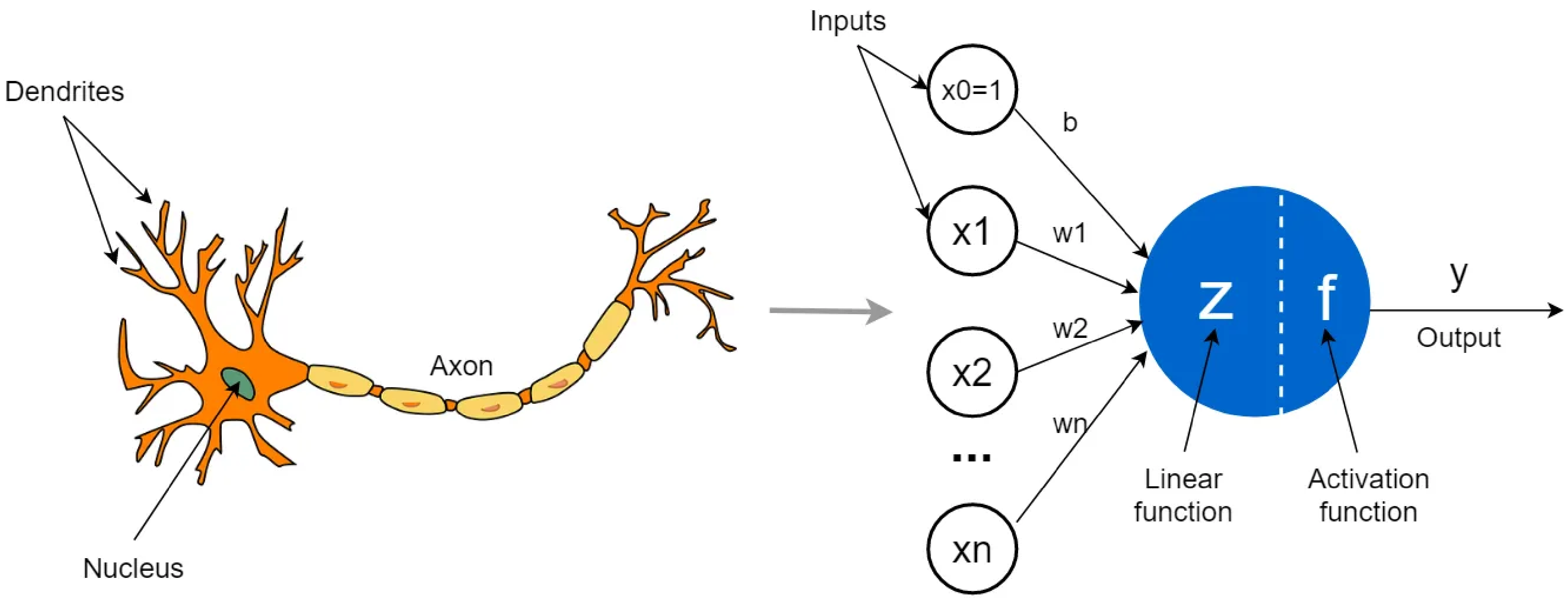
Introduction to Neural Networks

8th ICoMSE

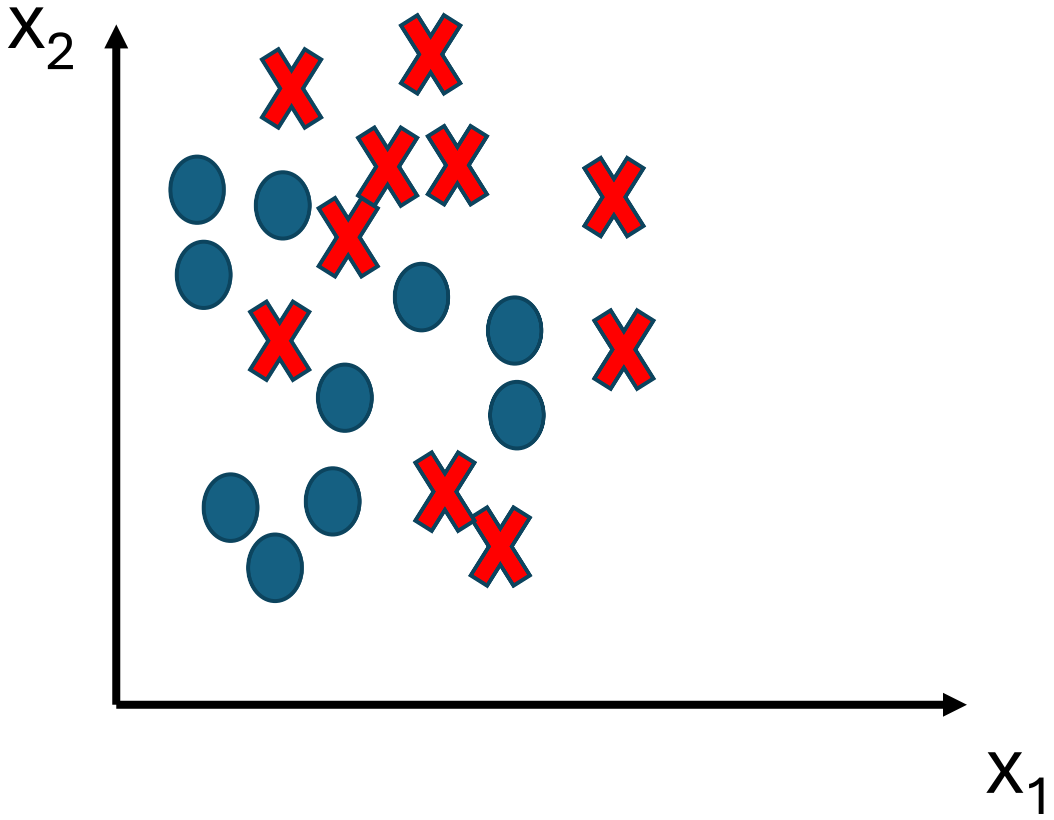
Yamil J. Colón

Neurons



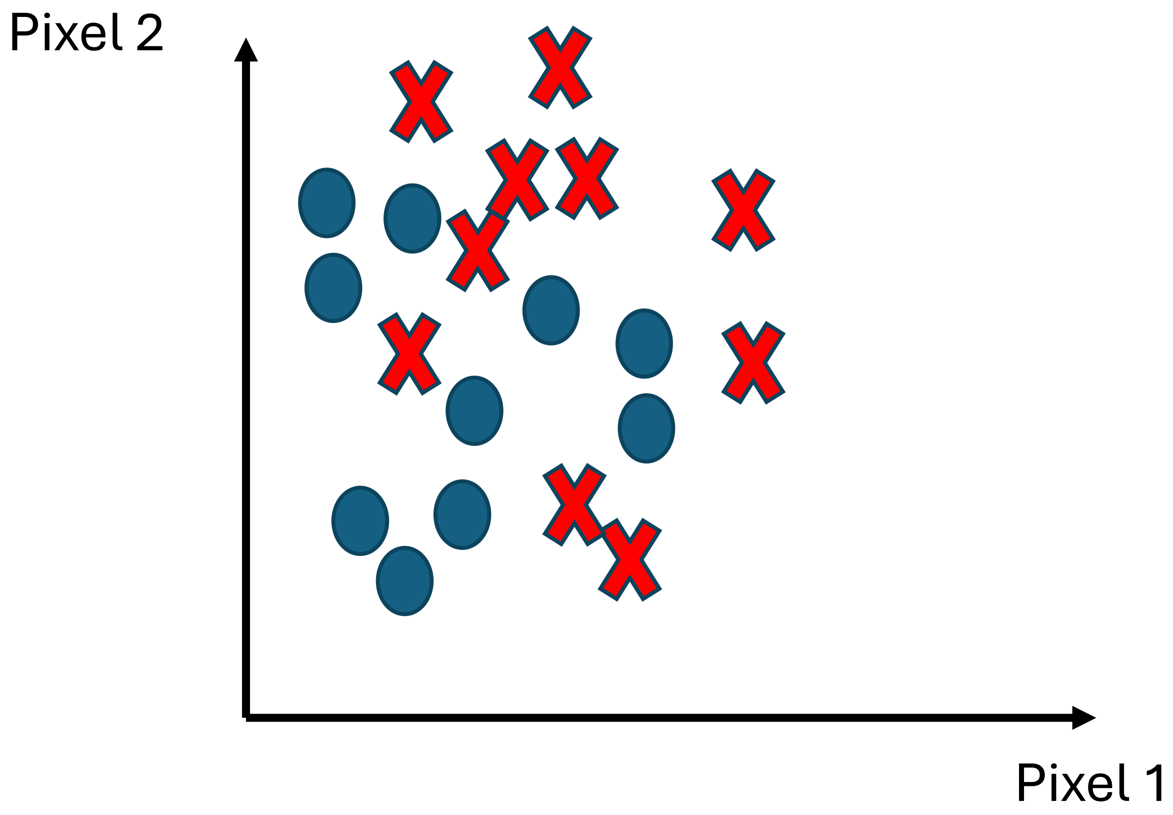
https://towardsdatascience.com/the-concept-of-artificial-neurons-perceptrons-in-neural-networks-fab22249cbfc

Non-linear classification

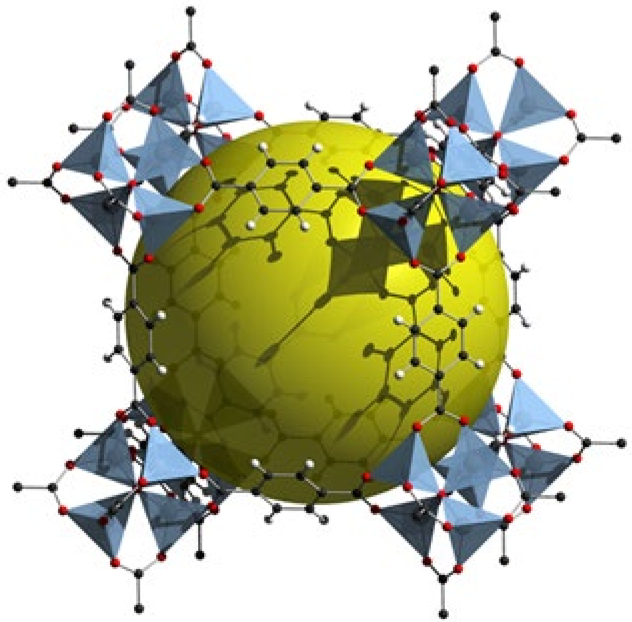


$$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 x_2 + \ \dots)$$

Think about images and their features...



- 50 x 50 pixel image in gray scale (0-255) n = 2500
- Quadratic features ~3,000,000
- RGB, n = 7500

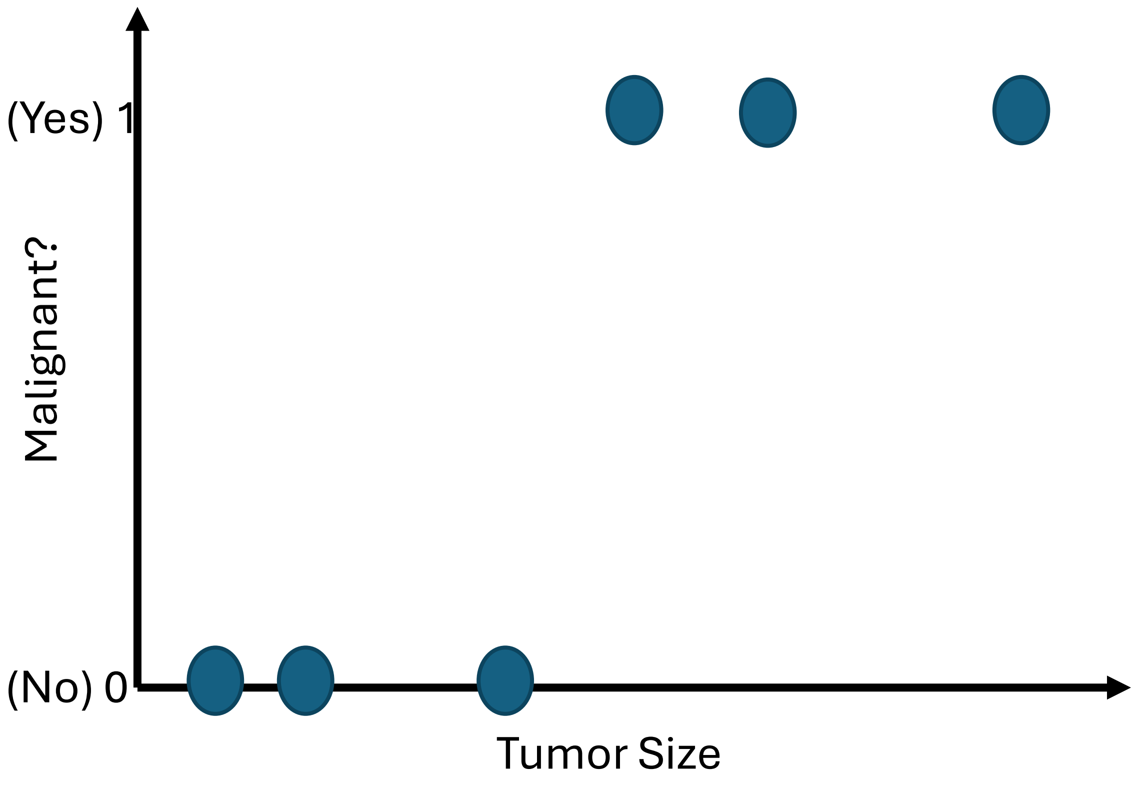


https://www.chemistryworld.com/podcasts/mofs-metal-organic-frameworks/3007204.article

Classic Example: Number Identification

Classification Example

- Threshold classifier.
- Need hypothesis that if ≥ 0.5, predicts y =1, and y =0 otherwise.



Linear vs. Logistic

- Classification: y = 0 or y = 1
  - Linear hypothesis can be >1 or <0.

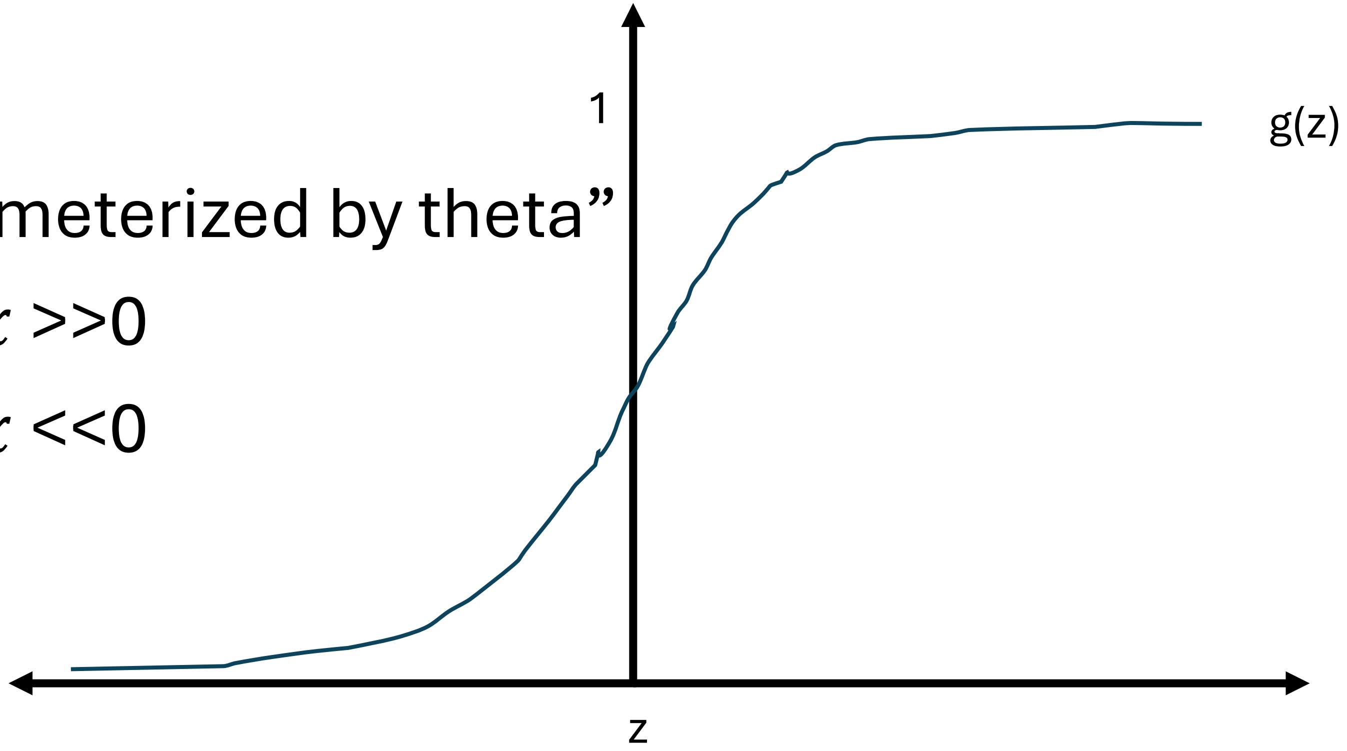
- Logistic Regression:
  - Hypothesis stays in [0,1]

Hypothesis Representation

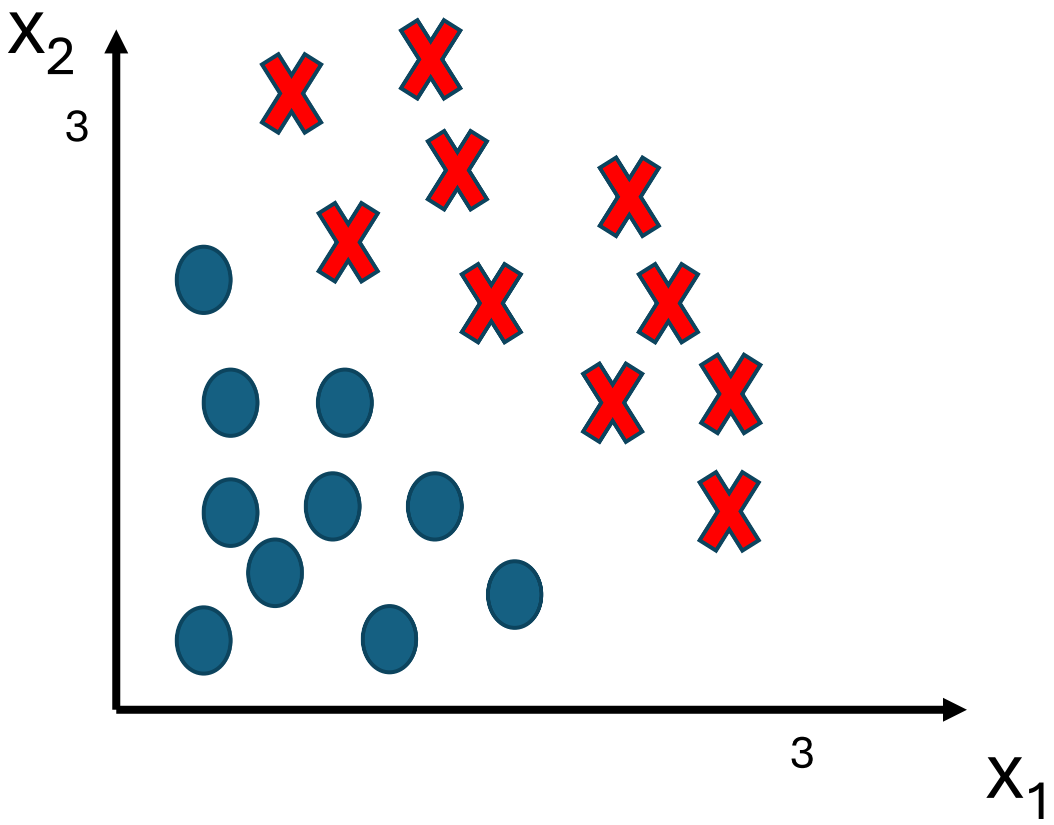
- Want $0 \leq h_\theta \leq 1$
- $h_\theta(x) = g(\theta^T x)$
- Sigmoid or logistic function $g(z) = \frac{1}{1+e^{-z}}$
- $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$

Hypothesis Interpretation

- Estimated probability that y =1 on input x
- Example: $h_\theta(x) = 0.7$
- 70% chance of y =1 condition
- $h_\theta(x) = P(y = 1|x;\theta)$
- "Probability y =1, given x, parameterized by theta"
- Predict y = 1 if $h_\theta(x)$ ≥ 0.5; $\theta^T x$ >>0
- Predict y = 0 if $h_\theta(x)$ < 0.5; $\theta^T x$ <<0

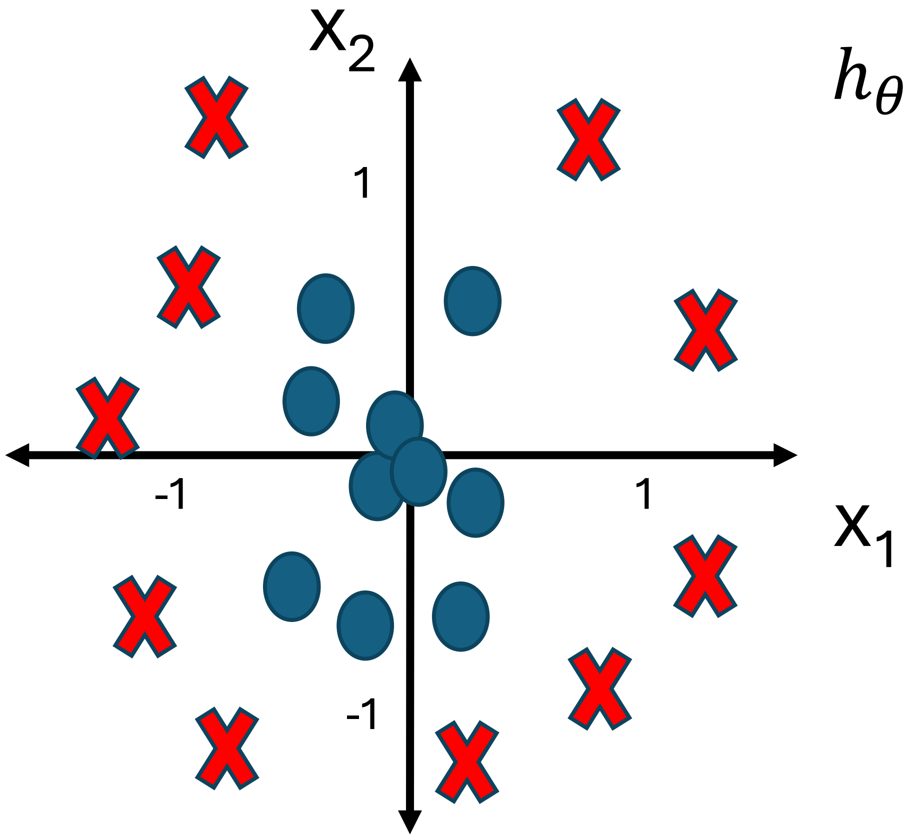


Decision Boundary



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

Nonlinear boundaries



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_1^2)$$

Cost Function

- Training set of bivariate data with m examples, $x_0 = 1$, $y \in [0,1]$
- Hypothesis: $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$
- How to choose thetas?

$$J(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}(h_\theta(\boldsymbol{x}) - \boldsymbol{y})^2 = \frac{1}{m}\sum_{i=1}^{m} Cost(h_\theta(\boldsymbol{x}), \boldsymbol{y})$$

Logistic regression

$$Cost(h_\theta(\boldsymbol{x}), \boldsymbol{y}) = \begin{cases} -\log h_\theta(\boldsymbol{x}) & \text{if } y = 1 \\ -\log\big(1 - h_\theta(\boldsymbol{x})\big) & \text{if } y = 0 \end{cases}$$

Summary

- y=0 or 1 always

$$J(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} Cost(h_\theta(\boldsymbol{x}), \boldsymbol{y})$$

$$Cost(h_\theta(\boldsymbol{x}), \boldsymbol{y}) = \begin{cases} -\log h_\theta(\boldsymbol{x}) \ \text{ if } y = 1 \\ -\log\big(1 - h_\theta(\boldsymbol{x})\big) \text{ if } y = 0 \end{cases}$$

$$Cost(h_\theta(\boldsymbol{x}), \boldsymbol{y}) = -y \log h_\theta(\boldsymbol{x}) \ - (1-y)\log(1 - h_\theta(\boldsymbol{x}))$$

Logistic regression cost function

$$J(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} Cost(h_\theta(\boldsymbol{x}), \boldsymbol{y})$$

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

- Want to fit parameters theta, to predict hypothesis given a new x.
- Probability y =1 given x, parameterized by theta.

Gradient Descent

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

Repeat until convergence {

$$\theta_j \coloneqq \theta_j - \alpha \frac{dJ(\boldsymbol{\theta})}{d\theta_j}$$

$$\theta_j \coloneqq \theta_j - \alpha \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) x_j^{(i)}$$

}

Advanced optimization algorithms

- We have Gradient Descent
- Other algorithms include:
  - Conjugate gradient
  - BFGS
  - L-BFGS
- They are usually faster and don’t need to specify learning rate
- But they’re more complex

- Code in notebook

The problem of overfitting

- If we have too many features, the learned hypothesis may fit the training set well, but fail to generalize to new examples.

Addressing overfitting

- Reduce number of features
  - Manually select which to keep
  - Model selection algorithm
- Regularization
  - Keep all the features, but reduce magnitude/values of thetas
  - Works well when we have a lot of features, each of which contributes a bit to predicting y.

Build intuition for regularization

- Linear regression
- What about lambda it it's too large??

$$J(\boldsymbol{\theta}) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

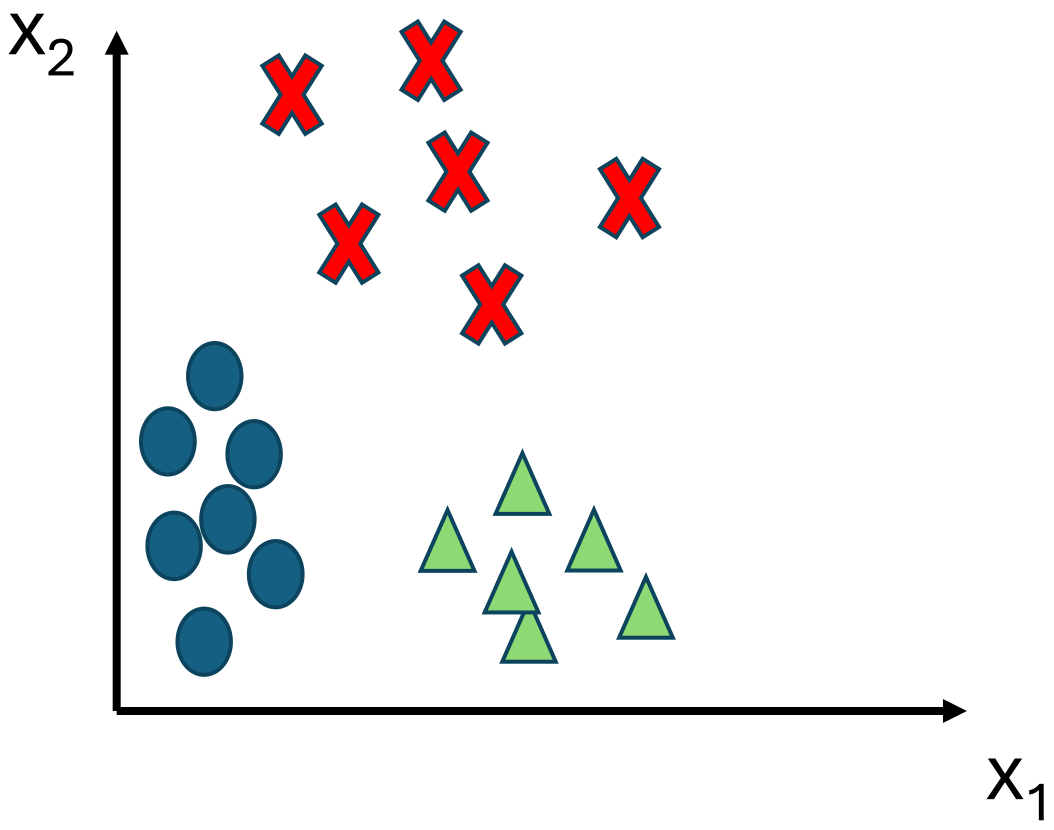
Regularized logistic regression

$$J(\boldsymbol{\theta}) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log h_\theta\left(x^{(i)}\right) + (1 - y^{(i)}) \log\left(1 - h_\theta\left(x^{(i)}\right)\right)\right] + \frac{\lambda}{2m}\sum_{j=1}^{n} \theta_j^2$$

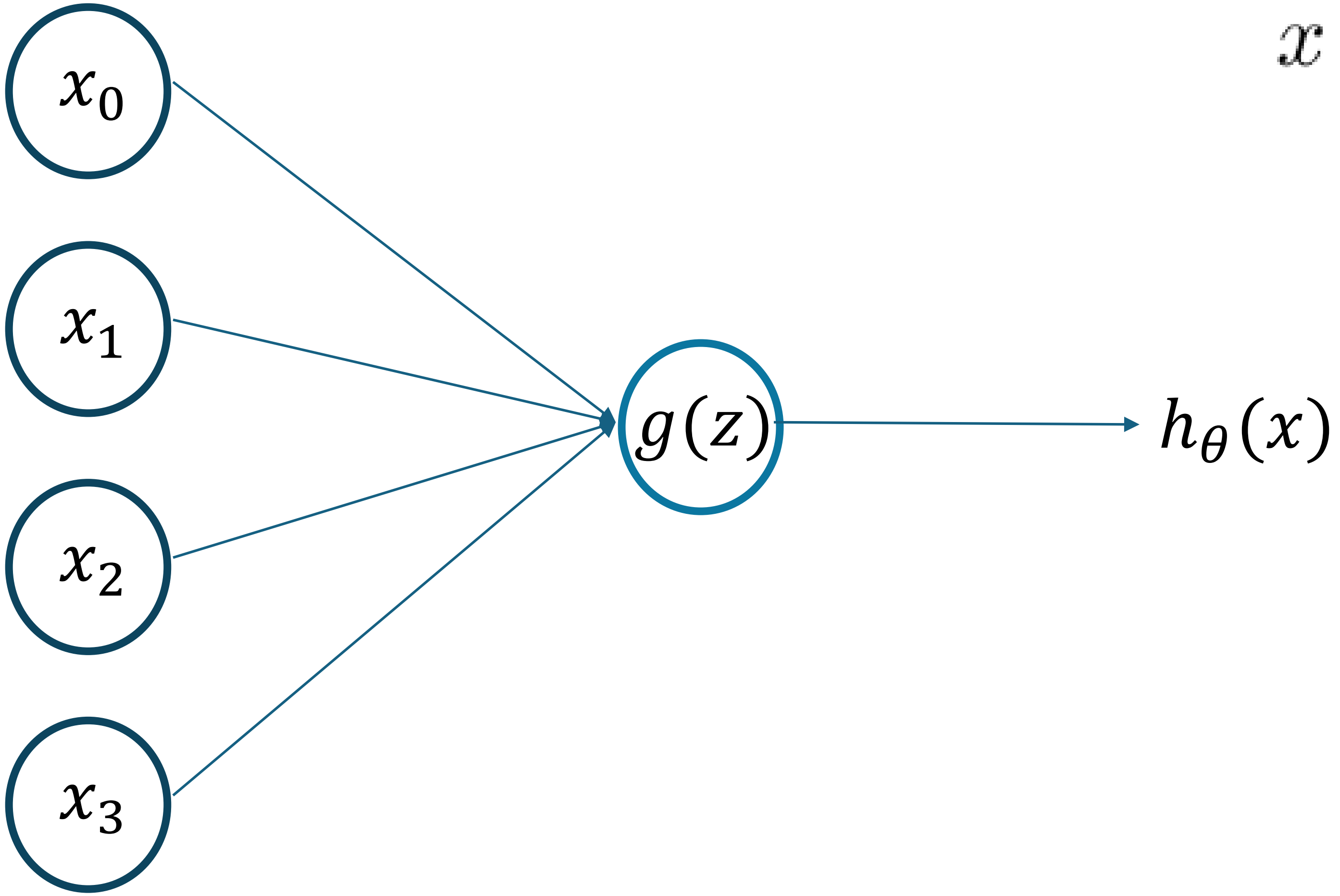
$$\theta_0 \coloneqq \theta_0 - \alpha \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) x_0^{(i)}$$

$$\theta_j \coloneqq \theta_j (1 - \alpha \frac{\lambda}{m}) - \alpha \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) x_j^{(i)}$$

Multiclass Classification



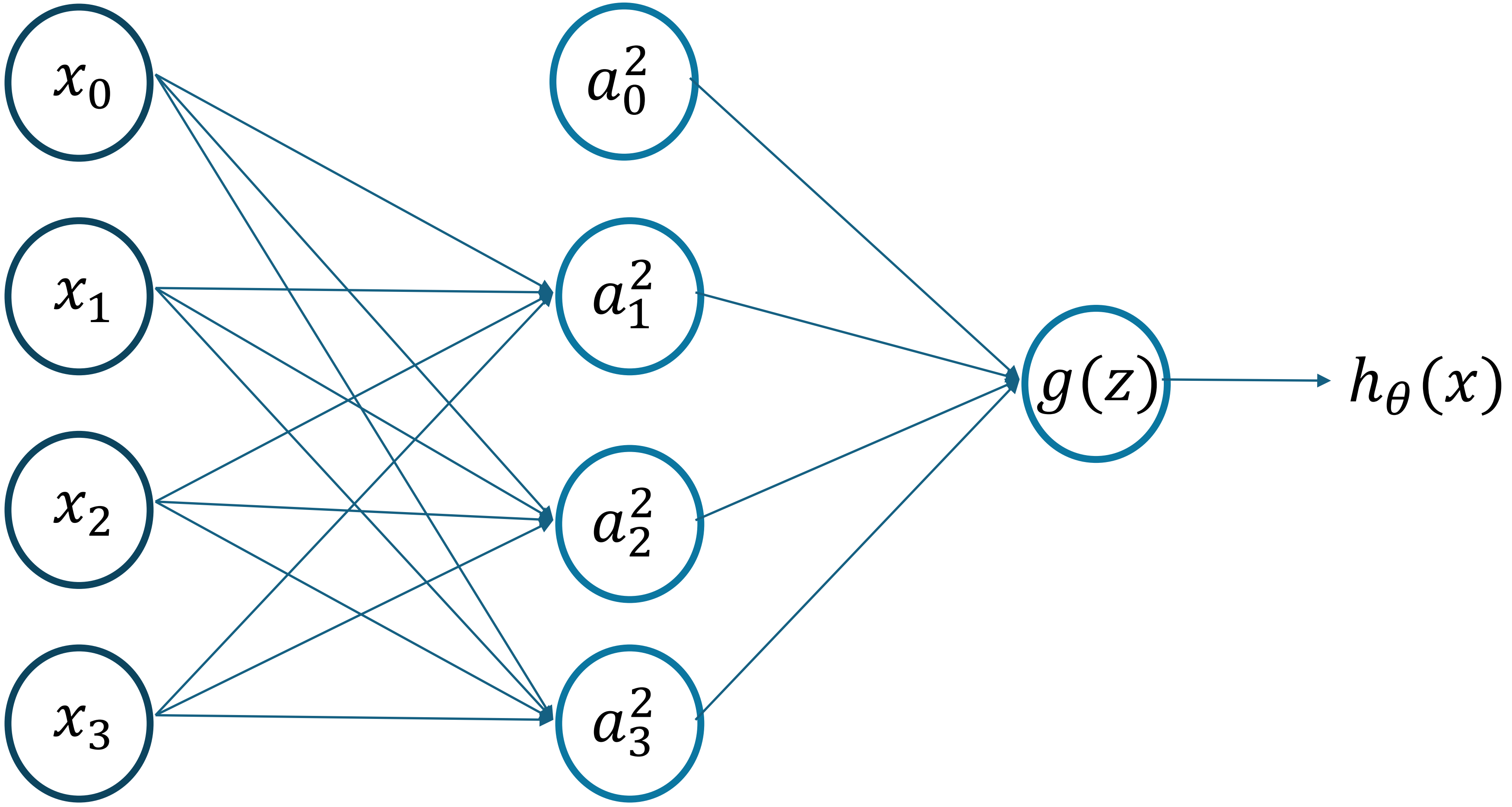
Logistic unit



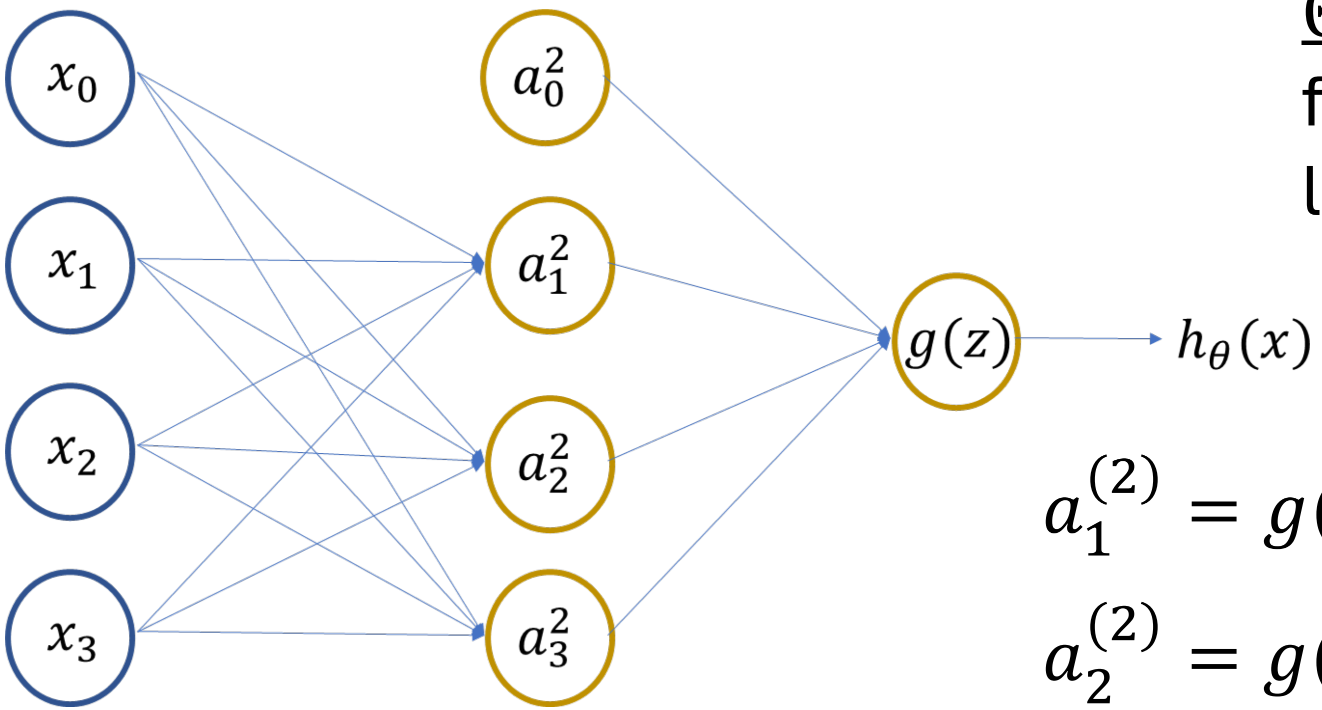
$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

Neural Network



Neural Network



$a_i^{(j)}$ = "activation" of unit i in layer j

$\underline{\Theta^{(j)}}$ = matrix of weights controlling function mapping from layer j to layer $j+1$

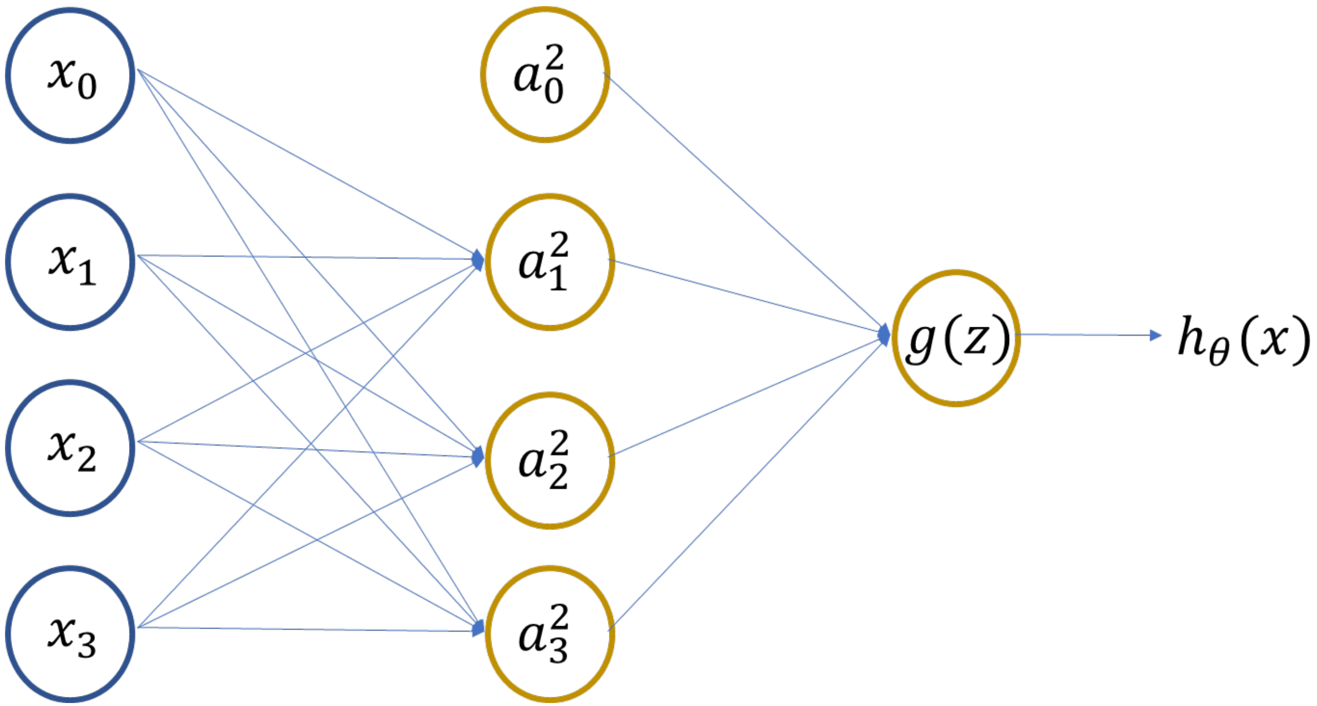
$$a_1^{(2)} = g(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3)$$

$$a_2^{(2)} = g(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3)$$

$$a_3^{(2)} = g(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3)$$

$$h_\theta(x) = a_1^{(3)} = g(\Theta_{10}^{(2)}a_0^{(2)} + \Theta_{11}^{(2)}a_1^{(2)} + \Theta_{12}^{(2)}a_2^{(2)} + \Theta_{13}^{(2)}a_3^{(2)})$$

Neural Network



$$a_1^{(2)} = g(z_1^{(2)})$$

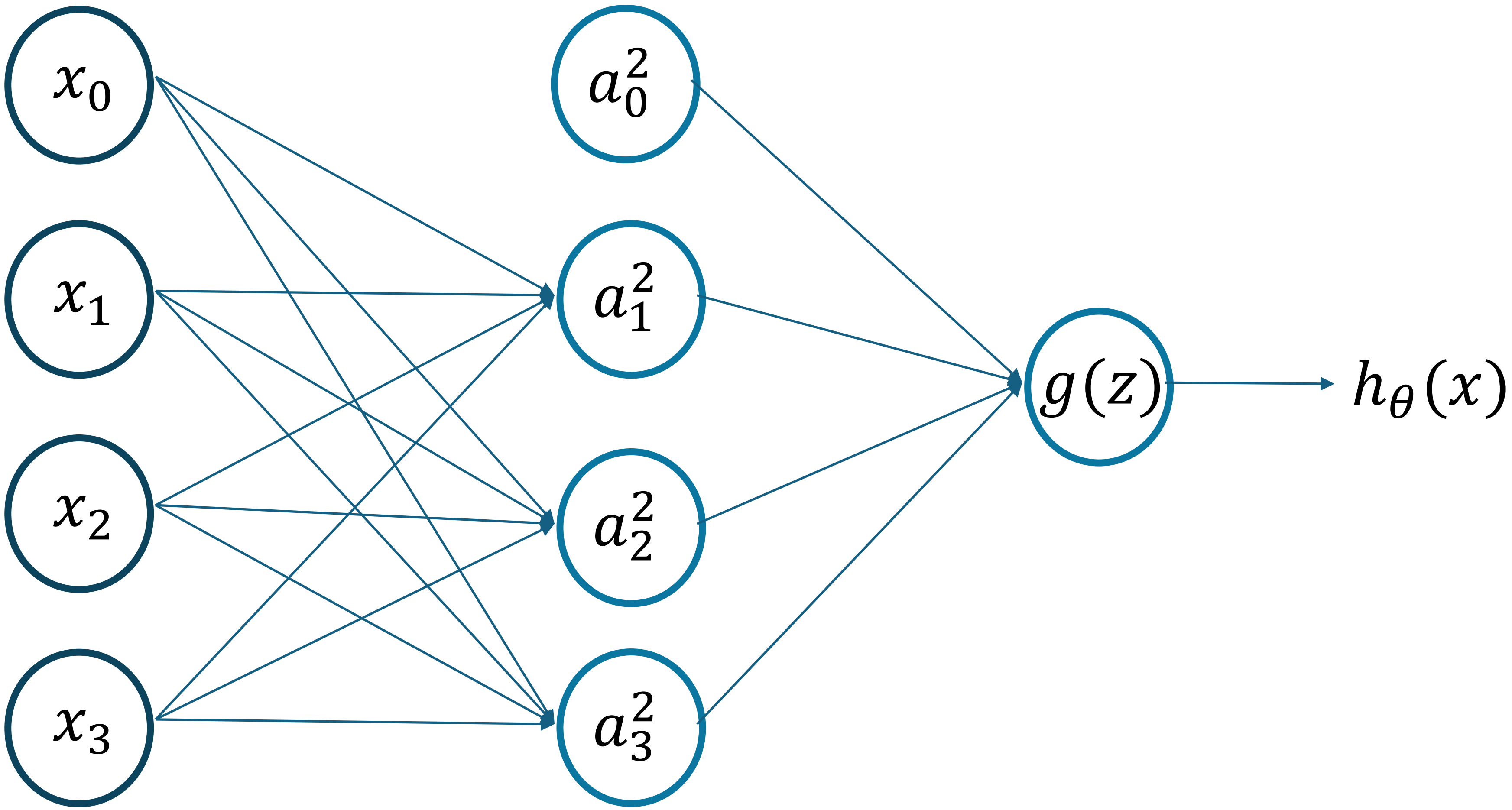
$$a_2^{(2)} = g(z_2^{(2)})$$

$$a_3^{(2)} = g(z_3^{(2)})$$

$$a^{(2)} = g(z^{(2)})$$

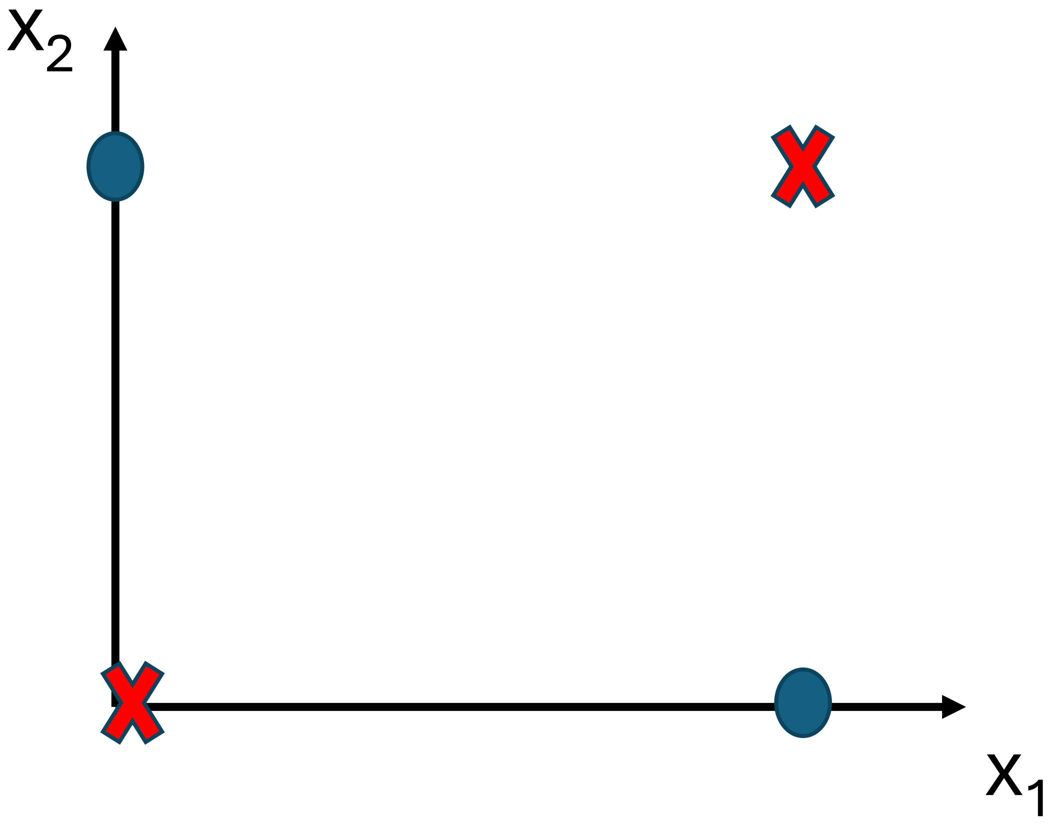
$$h_\theta(x) = a^{(3)} = g(z^{(3)})$$

Learning its own features



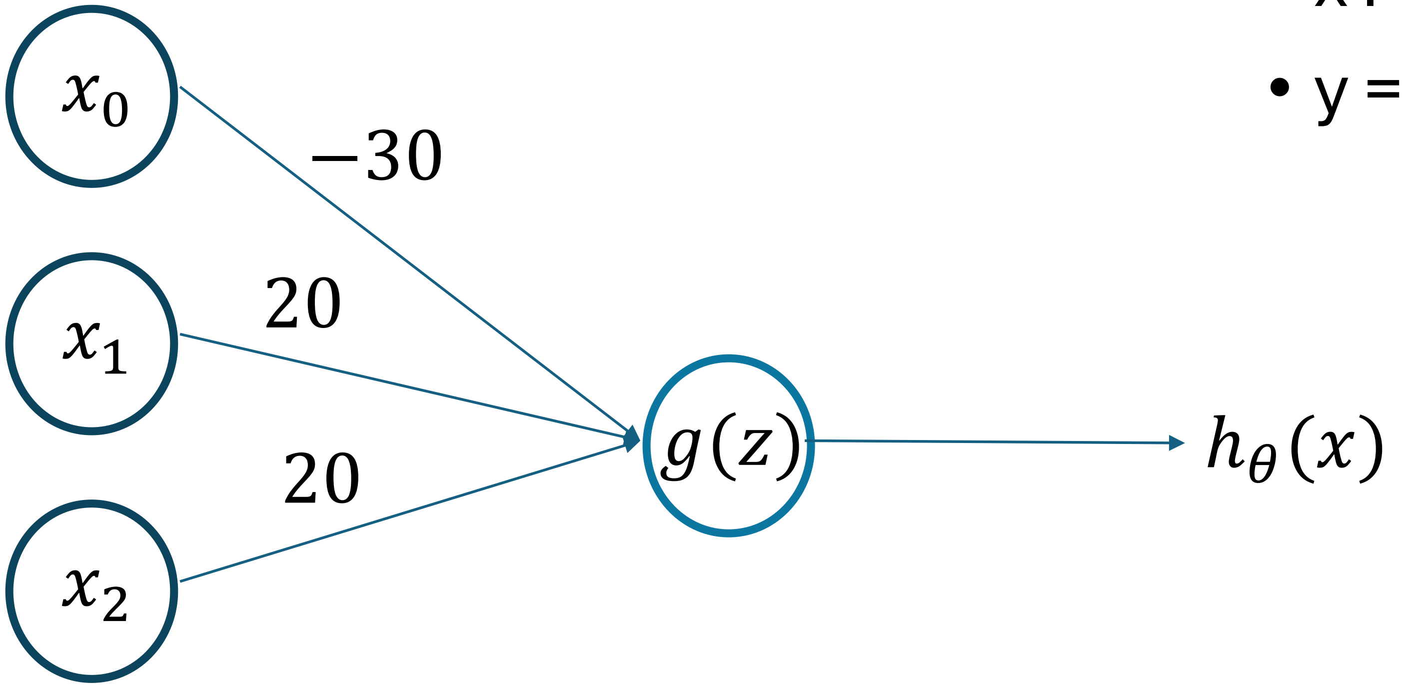
$$h_\theta(x) = a_1^{(3)} = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$$

Examples and intuitions for classification



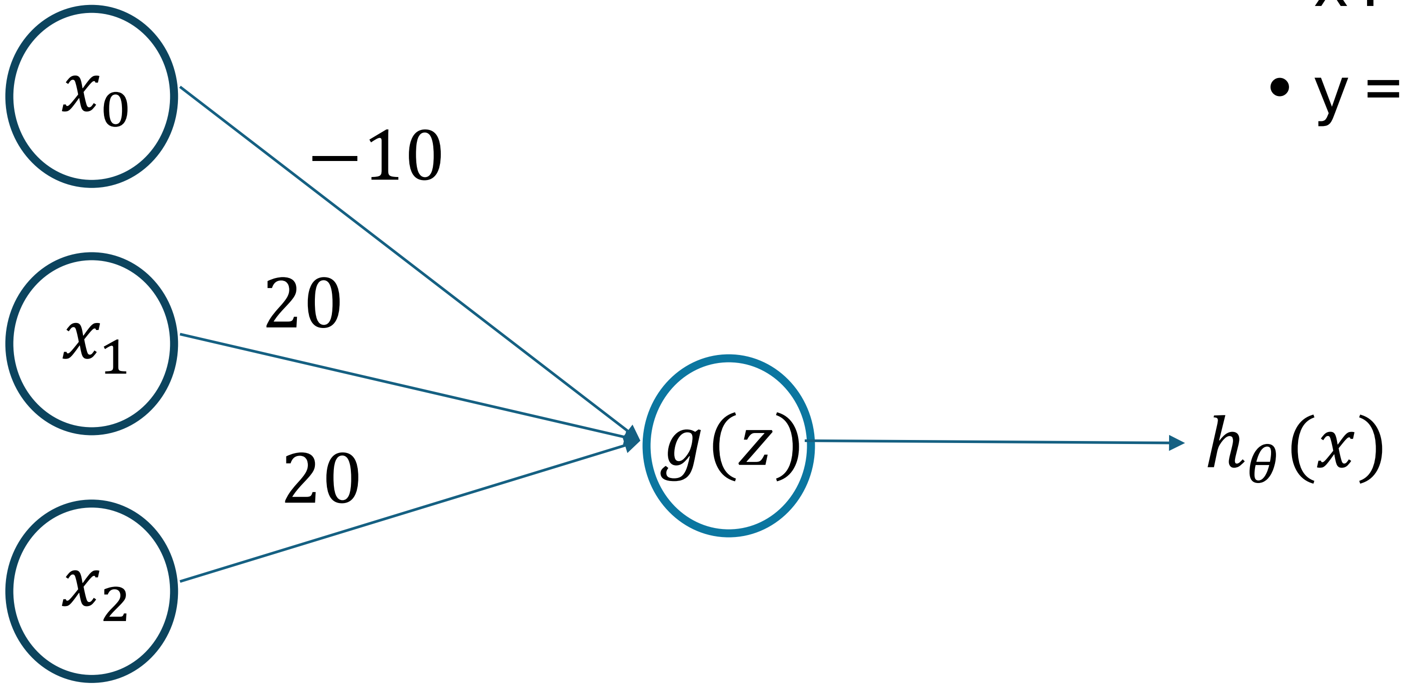
- x1 and x2 are binary
- y = 1: x , y = 0 : O

AND Example



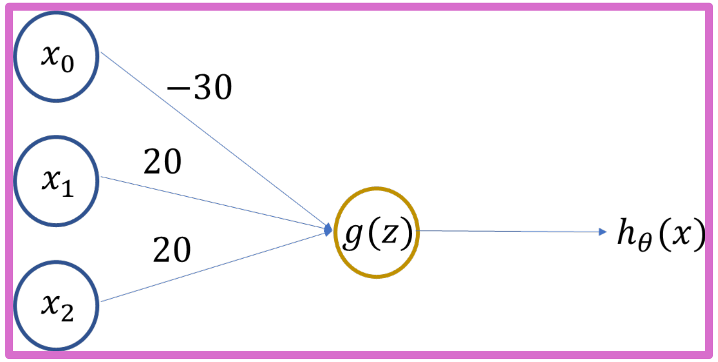
- x1 and x2 are binary
- y = 1 when x1 AND x2 =1

OR Example

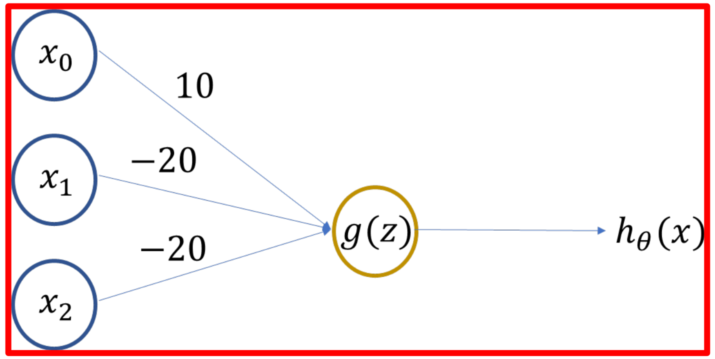


- x1 and x2 are binary
- y = 1 when x1 OR x2 =1

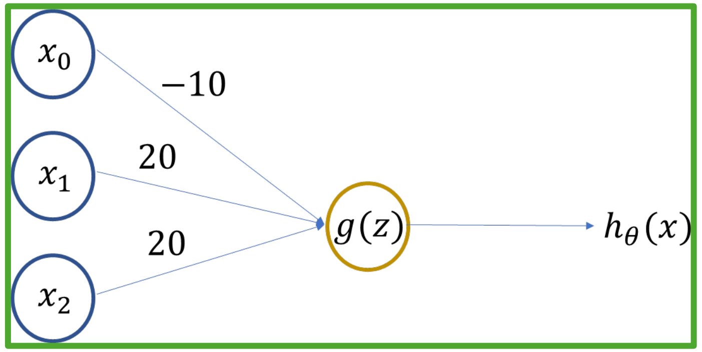
Back to the original situation



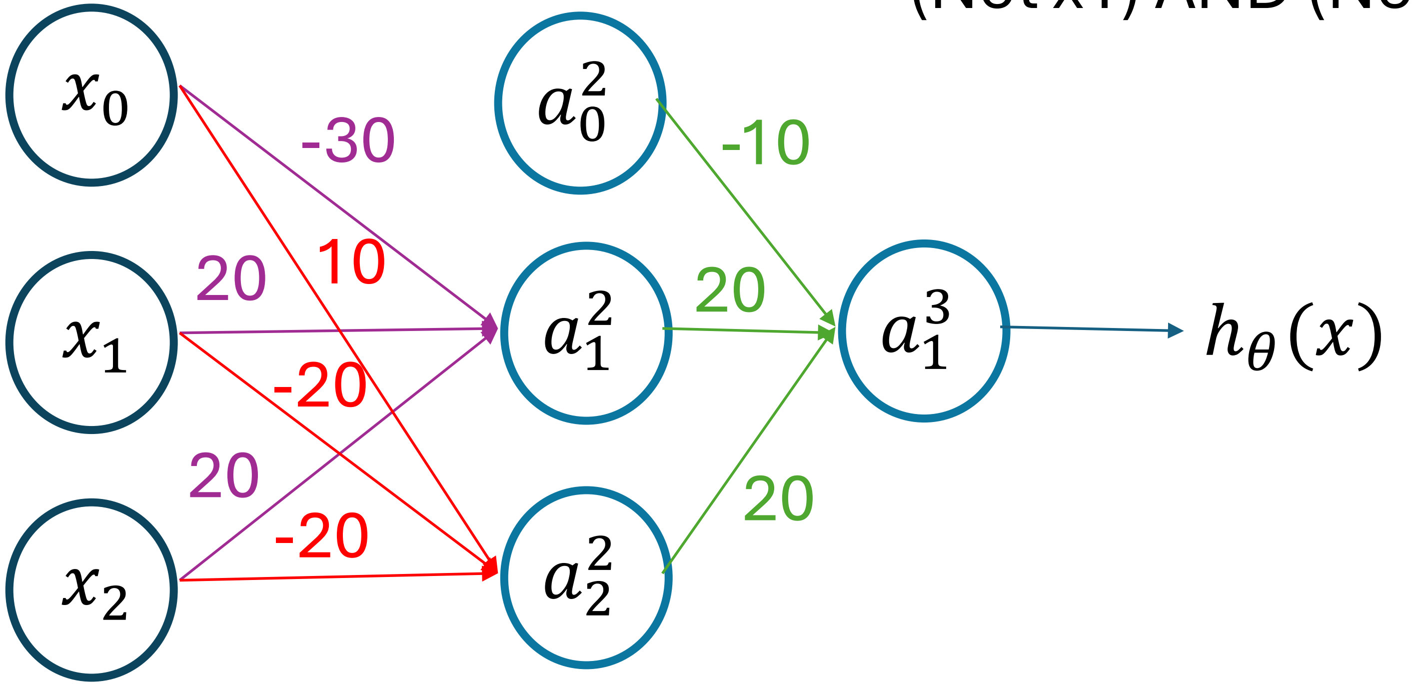
x1 AND x2

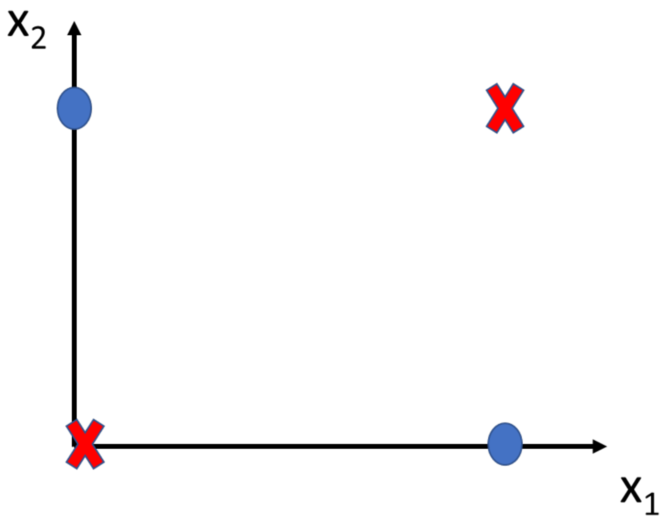


(Not x1) AND (Not x2)

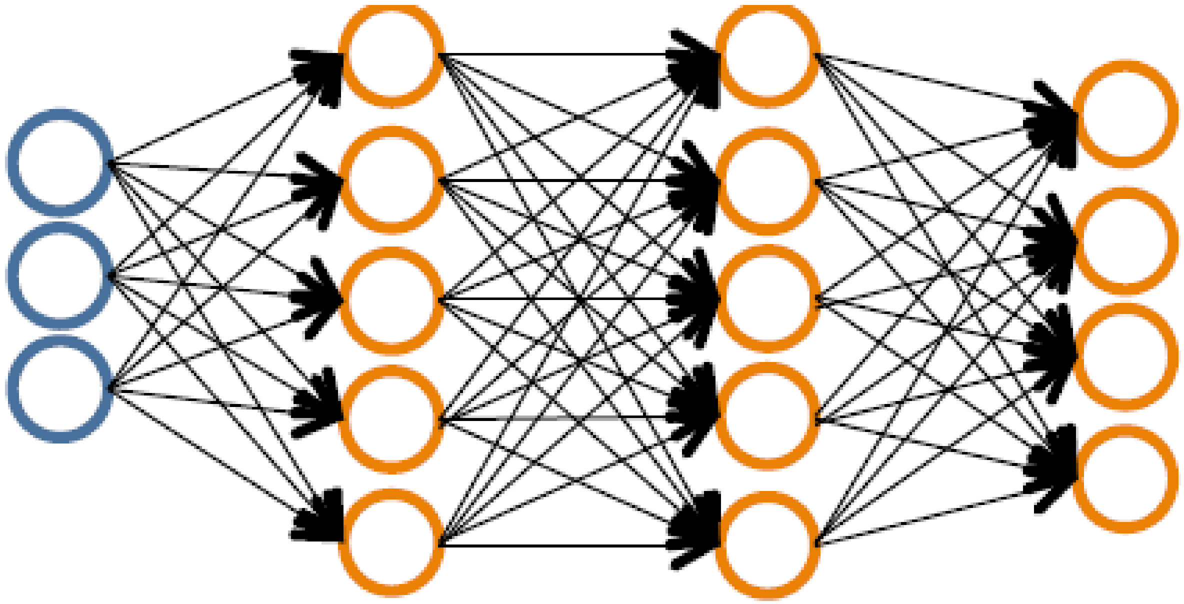


x1 OR x2





Neural Network for Classification



L = total number of layers in network

s_l = number of units in layer l, not counting the bias unit.

K = number of output units

Forward Propagation (Code)

- Given one training example (x,y):

$a^{(1)} = x$

$z^{(2)} = \Theta^{(1)} a^{(1)}$

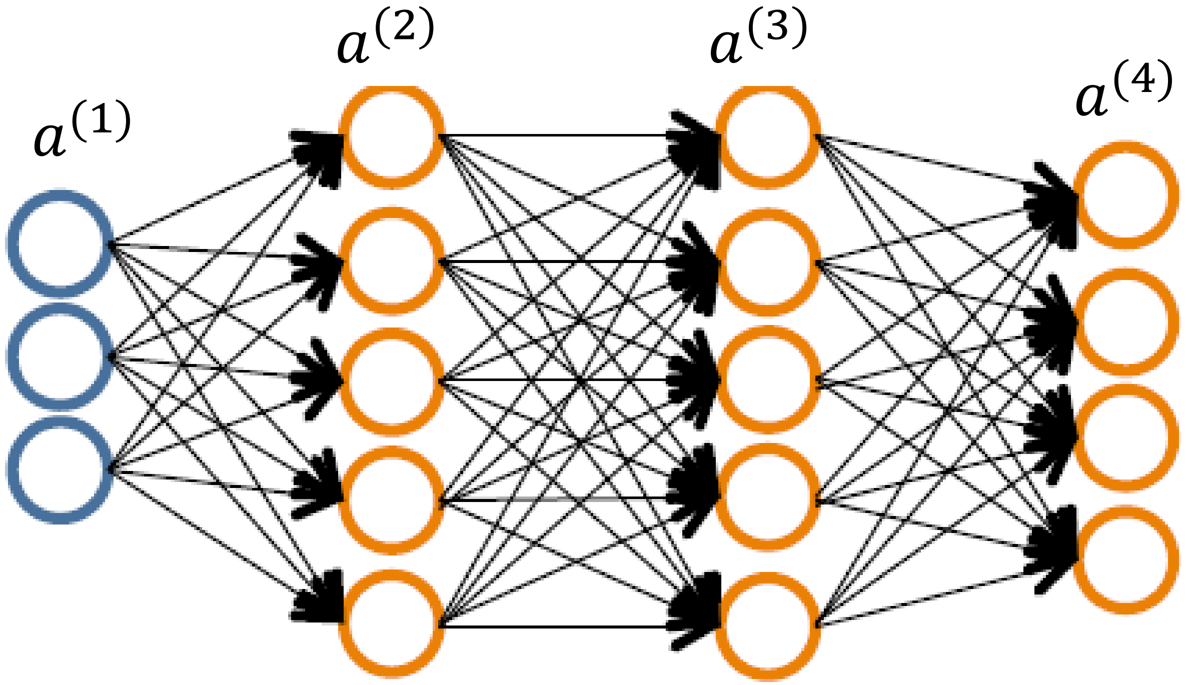
$a^{(2)} = g(z^{(2)})$ Add bias

$z^{(3)} = \Theta^{(2)} a^{(2)}$

$a^{(3)} = g(z^{(3)})$ Add bias

$z^{(4)} = \Theta^{(3)} a^{(3)}$

$a^{(4)} = h_\theta(x) = g(z^{(4)})$



Cost Function

$$J(\boldsymbol{\theta}) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log h_\theta\big(x^{(i)}\big) + (1 - y^{(i)}) \log\Big(1 - h_\theta\big(x^{(i)}\big)\Big)\right] + \frac{\lambda}{2m}\sum_{j=1}^{n} \theta_j^2$$

$$J(\boldsymbol{\theta}) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log h_\theta\big(x^{(i)}\big)_k + \Big(1 - y_k^{(i)}\Big) \log\Big(1 - h_\theta\big(x^{(i)}\big)\Big)_k\right]$$

$$+ \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}} (\Theta_{ji}^{(l)})^2$$

Cost Function

- Coding exercise with given thetas regularized and non-regularized cost functions

Backpropagation Algorithm

$$J(\boldsymbol{\theta}) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log h_\theta\left(x^{(i)}\right)_k + \left(1 - y_k^{(i)}\right)\log\left(1 - h_\theta\left(x^{(i)}\right)\right)_k\right]$$
$$+\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(\Theta_{ji}^{(l)})^2$$

- Need to find thetas that minimize the cost function.

- As we've done before, we need to calculate the cost function and the partial derivatives of the cost function w.r.t the thetas.

Forward Propagation Review

- Given one training example (x,y):

$a^{(1)} = x$

$z^{(2)} = \Theta^{(1)} a^{(1)}$

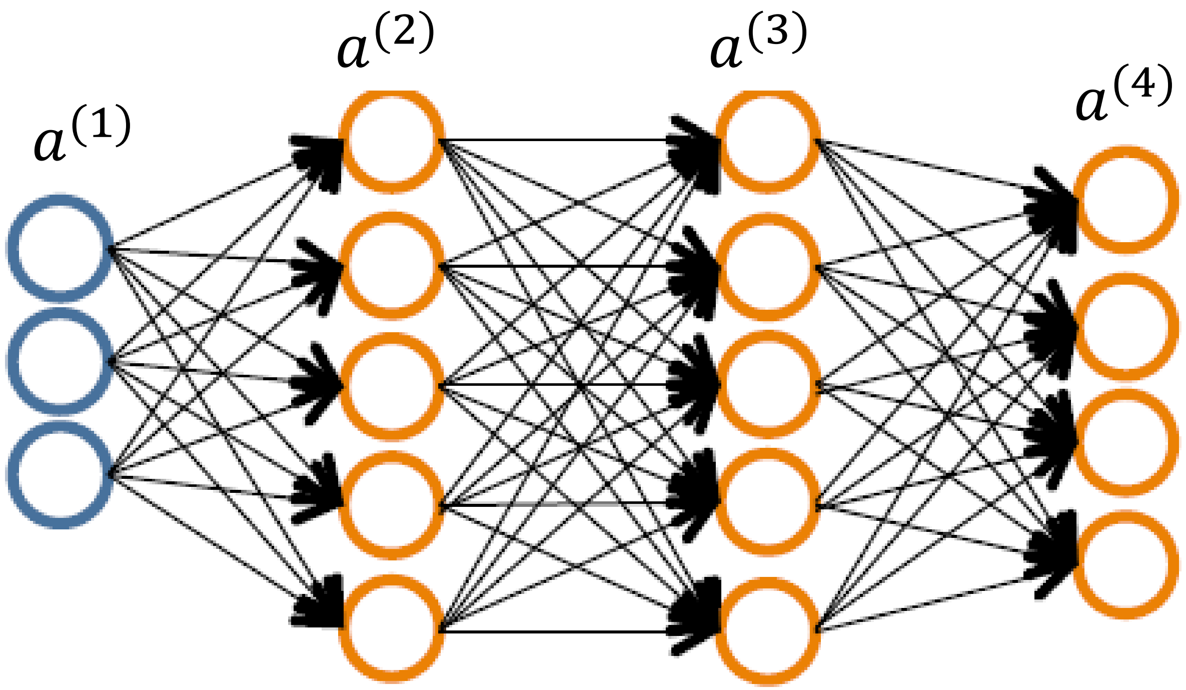
$a^{(2)} = g(z^{(2)})$ Add bias

$z^{(3)} = \Theta^{(2)} a^{(2)}$

$a^{(3)} = g(z^{(3)})$ Add bias

$z^{(4)} = \Theta^{(3)} a^{(3)}$

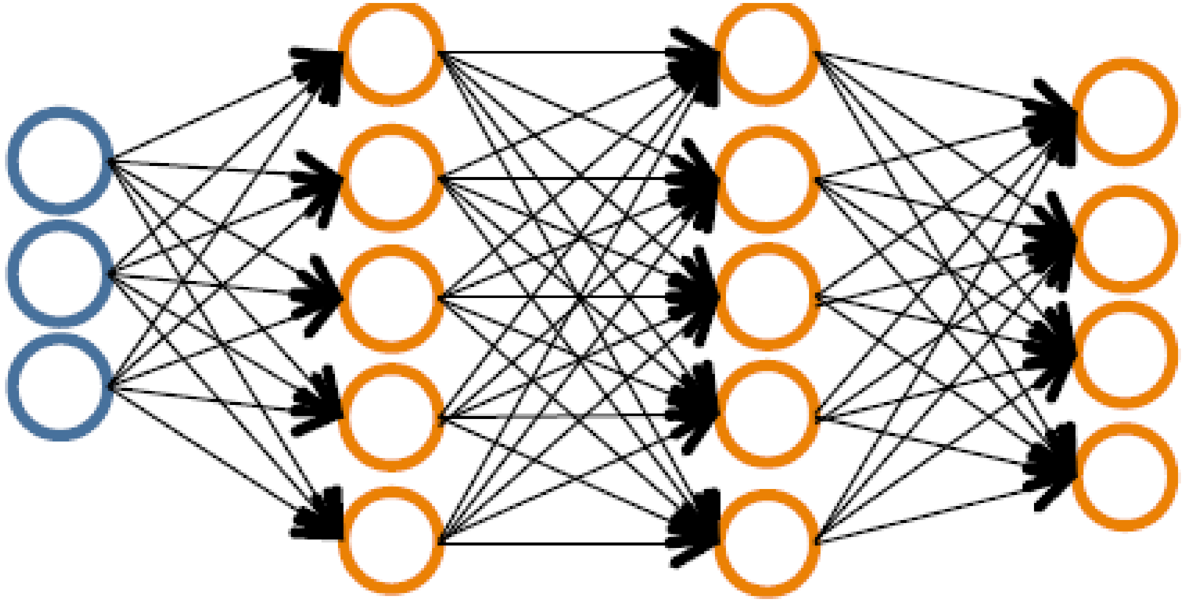
$a^{(4)} = h_\theta(x) = g(z^{(4)})$



Gradient Computation: Backprop

- To calculate gradient of cost function, need to calculate error and propagate through the nodes.

$\delta_j^{(l)}$ = "error" of node j in layer l



$$\delta^{(4)} = a^{(4)} - y$$

$$\delta^{(3)} = (\Theta^{(3)})^T \delta^{(4)} g'(z^{(3)})$$

$$\delta^{(2)} = (\Theta^{(2)})^T \delta^{(3)} g'(z^{(2)})$$

$$\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta) = a_j^{(l)} \delta_i^{(l+1)}$$

Backprop Pseudocode

$\Delta_{ij}^{(l)}$ contains deltas

$$\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta) = D_{ij}^{(l)}$$

- Given a training set

Set $\Delta_{ij}^{(l)} = 0$ for all l,i,j

For loop over data (t to m)

Set $a^{(1)} = x^{(t)}$

Fwd Prop to calculate $a^{(l)}$ for all layers

Calculate $\delta^{(L)} = a^{(L)} - y^{(t)}$

Compute the rest of deltas: $\delta^{(L-1)}$ … $\delta^{(2)}$

$\Delta_{ij}^{(l)} \coloneqq \Delta_{ij}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$

$D_{ij}^{(l)} \coloneqq \frac{1}{m} \Delta_{ij}^{(l)} + \lambda \Theta_{ij}^{(l)}$ if j $\neq 0$

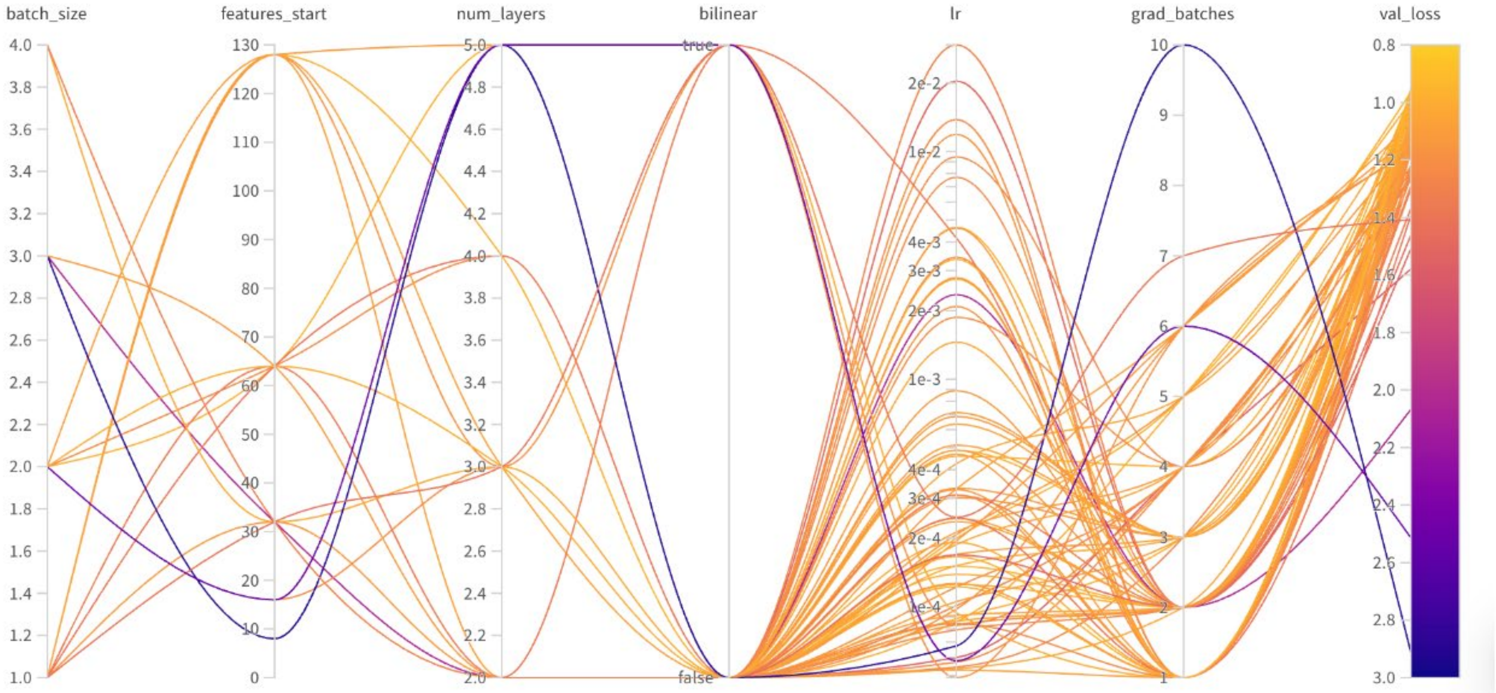
$D_{ij}^{(l)} \coloneqq \frac{1}{m} \Delta_{ij}^{(l)}$ if j $= 0$

Implementation Notes

- Review flattening matrices and rebuilding them (gradient, thetas, Ds)
- Random initialization: make sure initial guesses from theta are not all equal to zero, instead assign them randomly.

Putting it Together

- Coding exercises
- Backprop coding exercise: regularized gradient
- Regularized NN
- Learn parameters
- Hyperparameter optimization: https://wandb.ai/site



Debugging a learning algorithm

- We fit a model but find the error on new predictions is poor.
- What should we try?

Evaluating your hypothesis

- What are some ideas to help with generalization and evaluate generalization?

Train/Test procedure

- Learn parameters from training data (minimizing cost function)
- Compute the error based on the test set.

Model Selection

- Overfitting
- Try different models (say different degree polynomials)
- How to choose best model?

Diagnosing bias vs. variance

- High bias (underfit)
- High variance (overfit)

Diagnosing bias vs. variance

- Training error
- Cross validation error
- Plot error vs. degree of polynomial
- Based on the plots, how can you tell bias vs. variance?

Bias and variance with regularization

- Based on what we have discussed so far in this lecture and our knowledge of regularization, what is a large lambda prone to? small lambda?

Choosing the regularization parameter

- Strategies?
- Plot the training and the cross validation error versus lambda. Where is high bias? Where is high variance?

Learning Curves

- What do J_train and J_CV look like as a function of the number of data points increases?
- What would those curves look like if there is high bias? High variance?

Debugging a learning algorithm

- What do the following fix?
  - More training examples
  - Smaller set of features
  - Getting additional features
  - Adding polynomial features
  - Decreasing lambda
  - Increasing lambda
- How is this related to neural networks?
- Coding exercise