# Lecture 9 Neural Networks

http://deeplearning.stanford.edu/wiki/index.php/Neural_Networks
http://adventuresinmachinelearning.com/neural-networks-tutorial/

EE-UY 4563/EL-GY 9123: INTRODUCTION TO MACHINE LEARNING

PROF. LINDA SELLIE

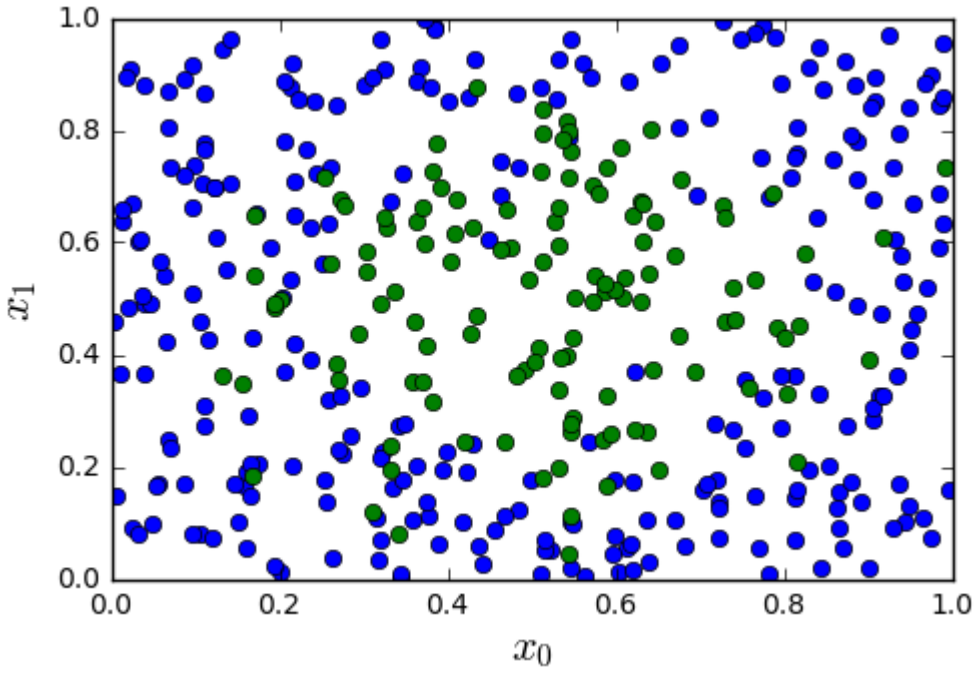
# Learning Objectives

- Mathematically describe a neural network with a single hidden layer
  - Describe mappings for the hidden and output units
- Select the loss function based on the problem type
- Build and train a simple neural network in Keras
- Write the formulas for gradients using backpropagation
- Describe mini-batches in stochastic gradient descent

# Outline

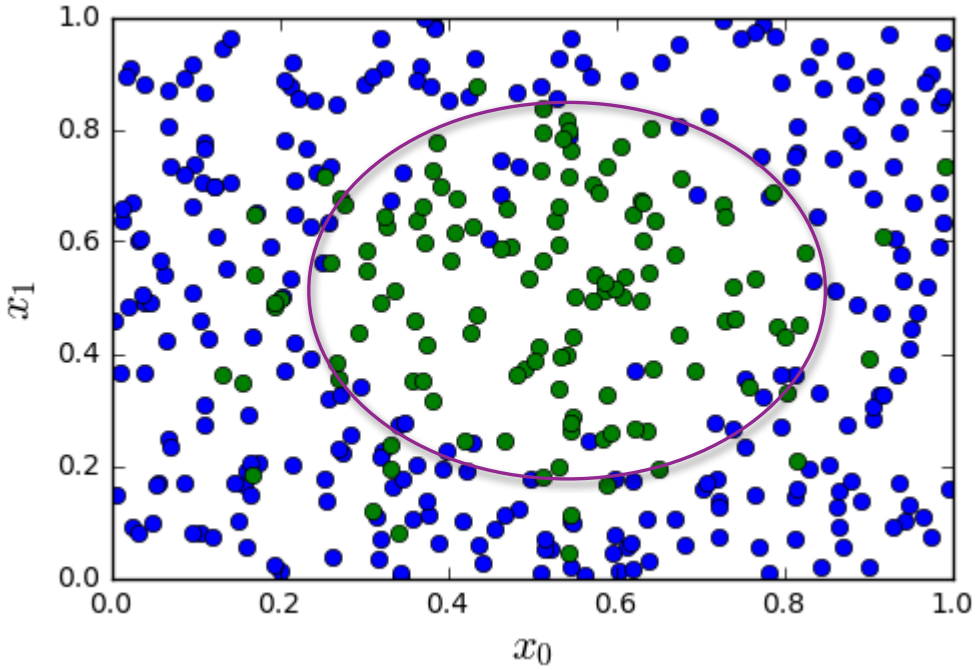
- Motivating Idea:  Nonlinear classifiers from linear features
- Neural Networks
- Backpropagation & Stochastic Gradient Descent
- Neural Network Loss Function
- Building and Training a Network in Keras
  - Synthetic data
  - MNIST

# Most Datasets are not Linearly Separable



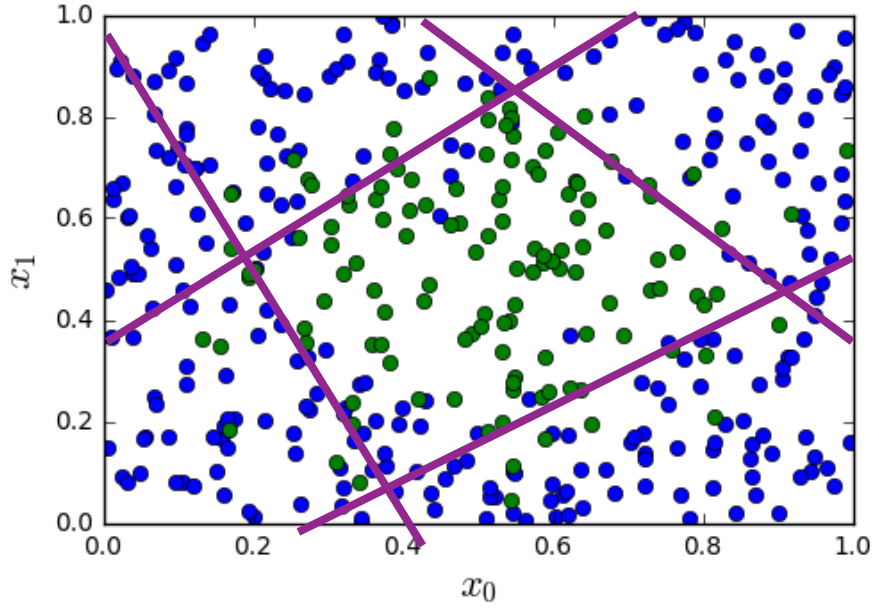

- Consider simple synthetic data
  - See figure to the left
  - 2D features
  - Binary class label
- Not separated linearly
- We can use SVM or Logistic Regression with non-linear features

# From Linear to Nonlinear




- Idea: Build nonlinear region from linear decisions

# From Linear to Nonlinear

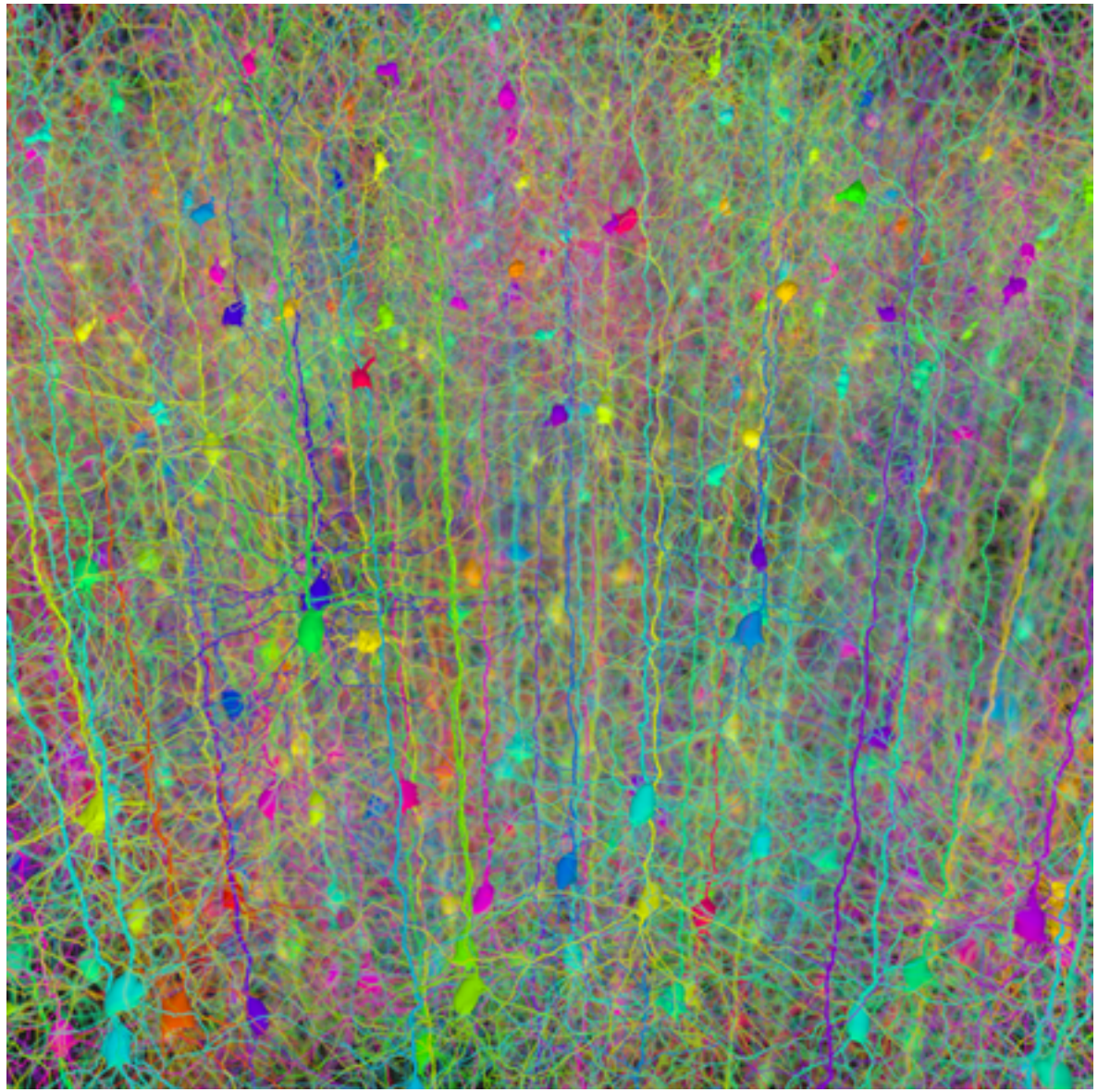



- Idea: Build nonlinear region from linear decisions

# How can we learn the right feature transformations (aka ***functions*** to transform our features)?

This is what neural networks do! We input the original features and the network learns different function of the features

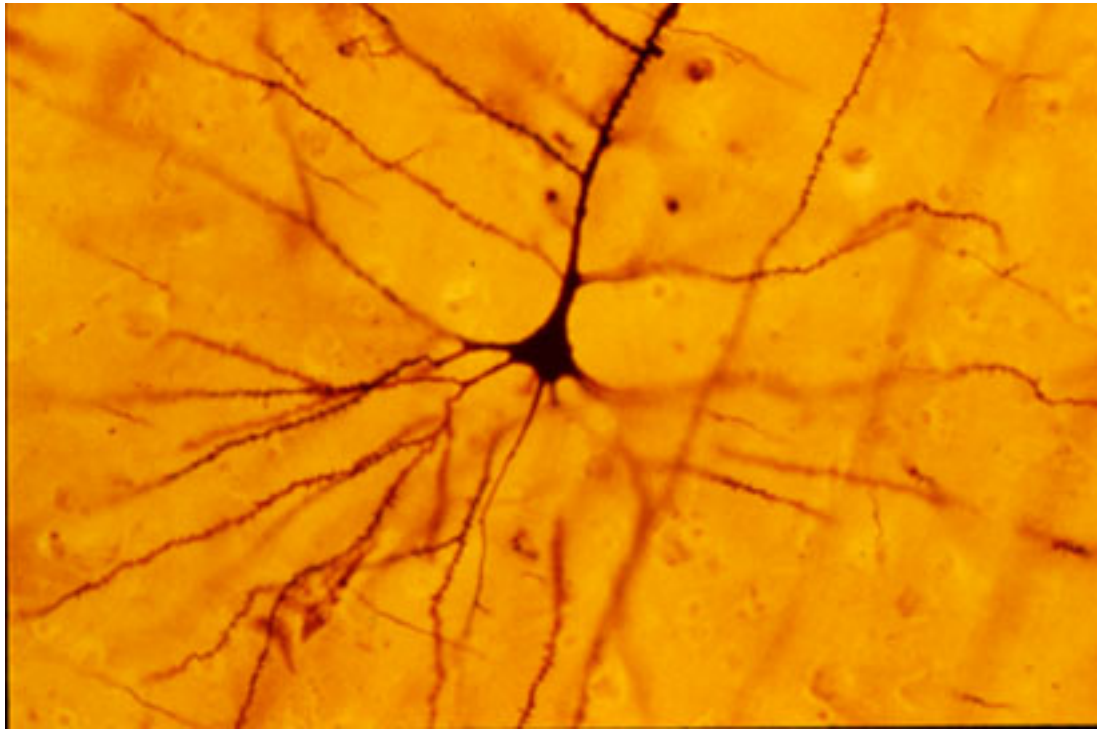
THAT IS THE MAIN TOPIC OF THIS LECTURE



"Computer simulation of the branching architecture of the dendritesof pyramidal neurons."

https://en.wikipedia.org/wiki/Neural_network

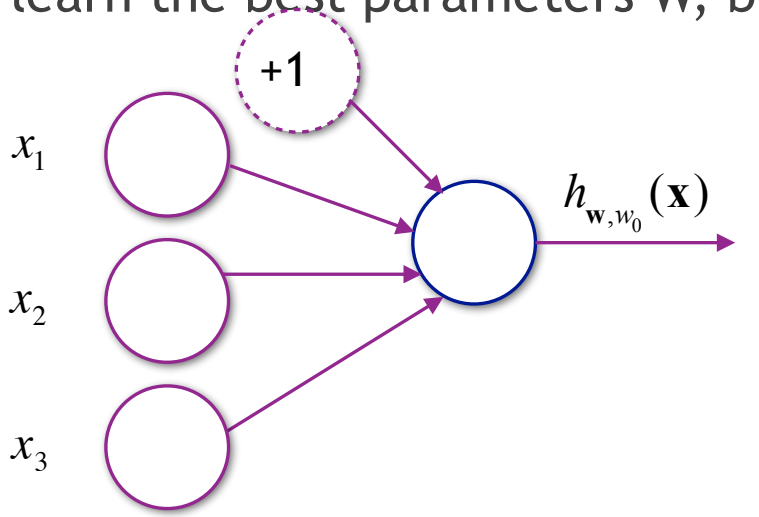
"**Pyramidal cells**, or **pyramidal neurons**, are a type of multipolar neuronfound in areas of the brain including the cerebral cortex, the hippocampus, and the amygdala"



https://en.wikipedia.org/wiki/Pyramidal_cell

# Overview

- Neural networks approximate a complex, non-linear function f* with parameters W,b
- $y = f(W^T x + b)$
- The goal is to learn the best parameters W, b that give the best approximation




approximating single "neuron" using the sigmoid function

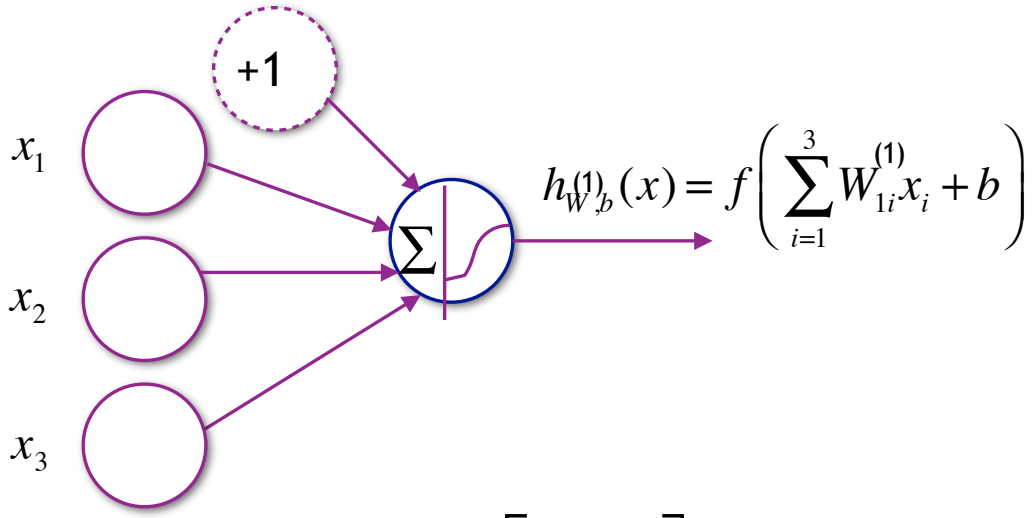
$$h_{\mathbf{w},w_0}(\mathbf{x}) = f\left(\sum_{i=1}^{3} w_i x_i + w_0\right) \qquad f(z) = \frac{1}{1+\exp(-z)}$$

Or.. We could have used the tanh (hyperbolic target function) $z_i = \sum_{i=1}^{3} w_i x_i + b$ $f(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$ or ReLu (i.e. $\max(0, z)$ ) or …

- Wait!!!! This is the logistic regression function….
- Neural network are called a network, because they typically compose many functions. The functions are composed in an order determined by an acyclic graph.

# New Notation

Unfortunately, there will be quite a bit of notation…..
We will follow the notation from http://deeplearning.stanford.edu/wiki/index.php/Neural_Networks



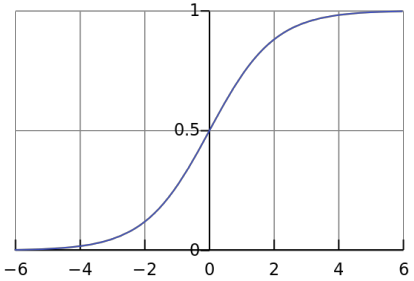

$$h_{W,b}^{(1)}(x) = f\left(\sum_{i=1}^{3} W_{1i}^{(1)} x_i + b\right)$$

Change of notation
$x_1$ is the first feature

$$W = \begin{bmatrix} W_{11}^{(1)} \\ W_{12}^{(1)} \\ W_{13}^{(1)} \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$z_i = [W_{11}^{(1)}, W_{12}^{(1)}, W_{13}^{(1)}] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + b$$

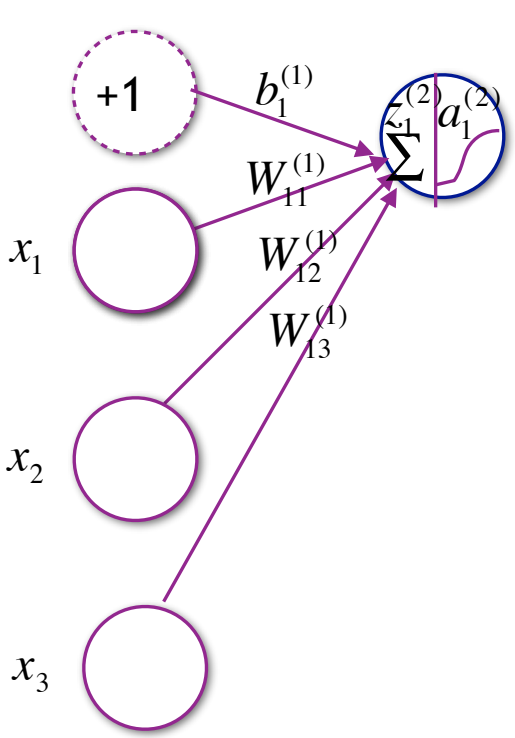
$$f(z) = \frac{1}{1+\exp(-z)}$$



# Neural Network Model & Notation

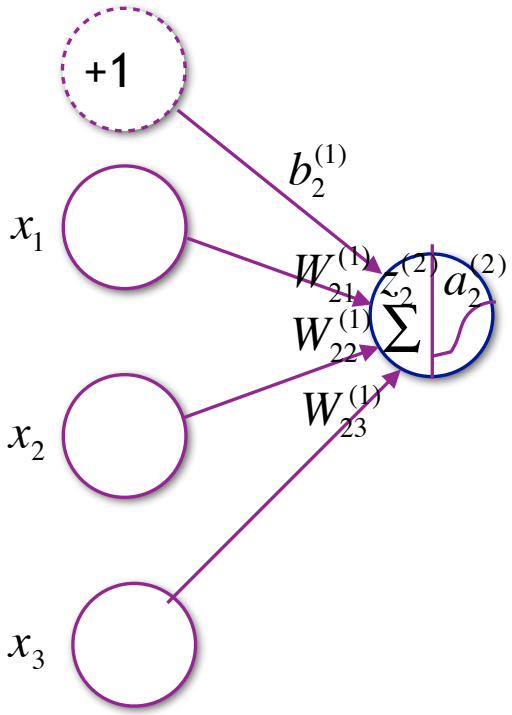
$$z_i^{(2)} = \sum_{j=1}^{3} W_{ij}^{(1)} x_j + b_i^{(1)} \qquad a_i^{(2)} = f(z_i^{(2)})$$

$$f(z) = \frac{1}{1+\exp(-z)}$$



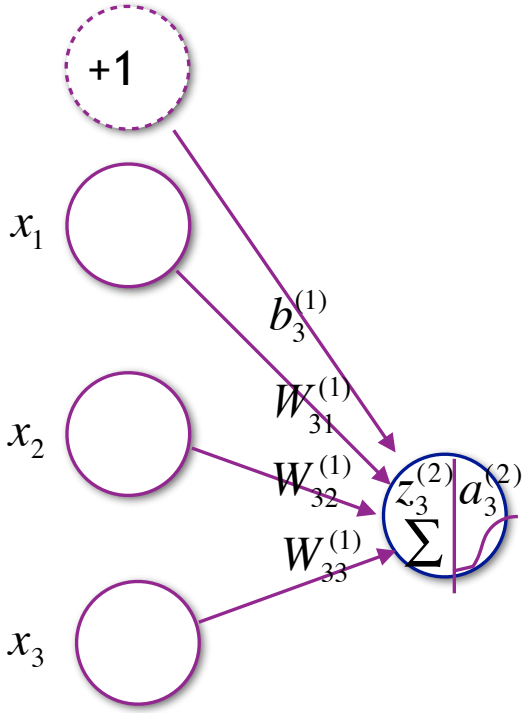

$$z_1^{(2)} = \sum_{j=1}^{3} W_{1j}^{(1)} x_j + b_1^{(1)}$$

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



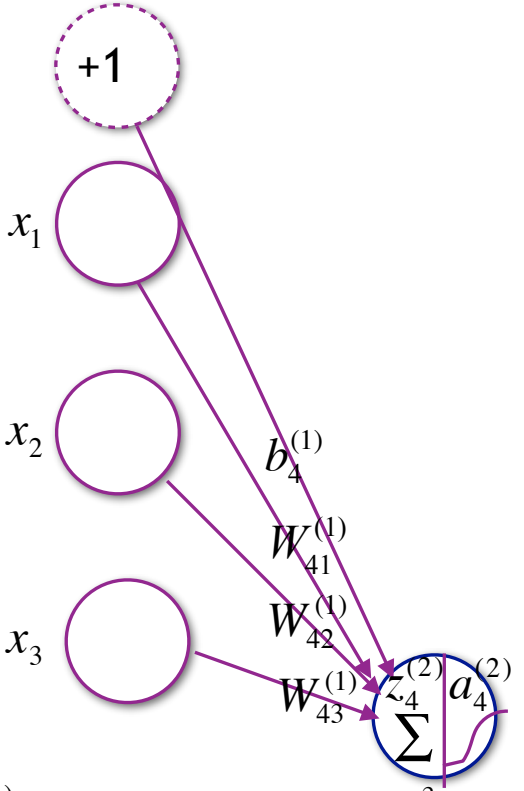

$$z_2^{(2)} = \sum_{j=1}^{3} W_{2j}^{(1)} x_j + b_2^{(1)}$$

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




$$z_3^{(2)} = \sum_{j=1}^{3} W_{3j}^{(1)} x_j + b_3^{(1)}$$

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



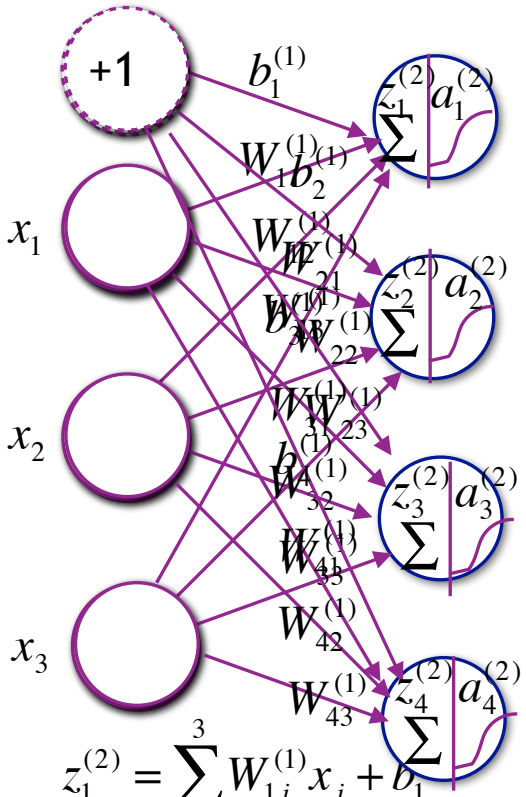

$$z_4^{(2)} = \sum_{j=1}^{3} W_{4j}^{(1)} x_j + b_4^{(1)}$$

$$a_4^{(2)} = f(z_4^{(2)})$$

# Neural Network Model & Notation

$$z_i^{(2)} = \sum_{j=1}^{3} W_j^{(1)} x_j + b_i^{(1)} \qquad a_i^{(2)} = f(z_i^{(2)})$$

$$f(z) = \frac{1}{1+\exp(-z)}$$




$$z_1^{(2)} = \sum_{j=1}^{3} W_{1j}^{(1)} x_j + b_1$$

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

$$z_2^{(2)} = \sum_{j=1}^{3} W_{2j}^{(1)} x_j + b_2^{(1)}$$

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

$$z_3^{(2)} = \sum_{j=1}^{3} W_{3j}^{(1)} x_j + b_3^{(1)}$$

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

$$z_4^{(2)} = \sum_{j=1}^{3} W_{4j}^{(1)} x_j + b_4^{(1)}$$

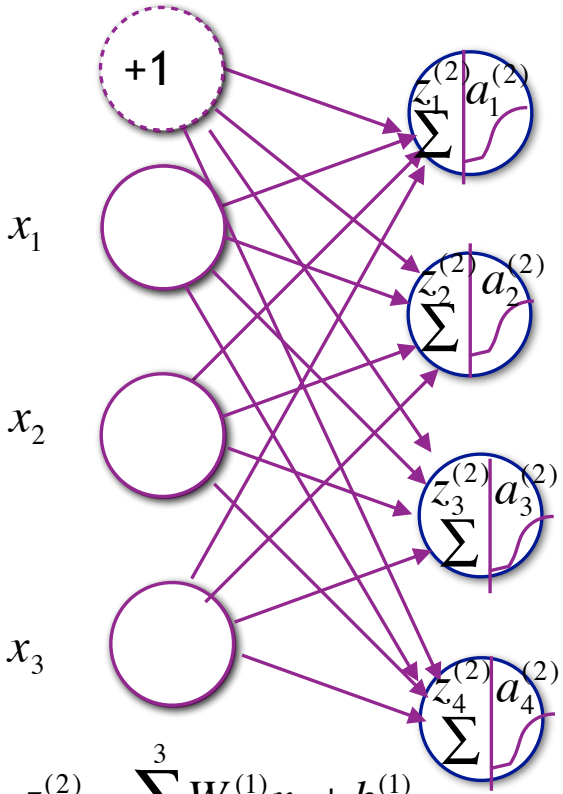
$$a_4^{(2)} = f(z_4^{(2)})$$

# Neural Network Model & Notation

$$z_i^{(2)} = \sum_{j=1}^{3} W_{ij}^{(1)} x_j + b_i^{(1)} \qquad a_i^{(2)} = f(z_i^{(2)})$$

$$z^{(2)} = W^{(1)} x + b^{(1)}$$
$$a^{(2)} = f(z^{(2)}) = f(W^{(1)} x + b^{(1)})$$

$$f(z) = \frac{1}{1+\exp(-z)}$$




$$\begin{bmatrix} W_{11}^{(1)} & W_{12}^{(1)} & W_{13}^{(1)} \\ W_{21}^{(1)} & W_{22}^{(1)} & W_{23}^{(1)} \\ W_{31}^{(1)} & W_{32}^{(1)} & W_{33}^{(1)} \\ W_{41}^{(1)} & W_{42}^{(1)} & W_{43}^{(1)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1^{(1)} \\ b_2^{(1)} \\ b_3^{(1)} \\ b_4^{(1)} \end{bmatrix} = \begin{bmatrix} W_{11}^{(1)} x_1 + W_{12}^{(1)} x_2 + W_{13}^{(1)} x_3 \\ W_{21}^{(1)} x_1 + W_{22}^{(1)} x_2 + W_{23}^{(1)} x_3 \\ W_{31}^{(1)} x_1 + W_{32}^{(1)} x_2 + W_{33}^{(1)} x_3 \\ W_{41}^{(1)} x_1 + W_{42}^{(1)} x_2 + W_{43}^{(1)} x_3 \end{bmatrix} + \begin{bmatrix} b_1^{(1)} \\ b_2^{(1)} \\ b_3^{(1)} \\ b_4^{(1)} \end{bmatrix}$$

$$= \begin{bmatrix} W_{11}^{(1)} x_1 W + W_{12}^{(1)} x_2 + W_{13}^{(1)} x_3 + b_1^{(1)} \\ W_{21}^{(1)} x_1 W + W_{22}^{(1)} x_2 + W_{23}^{(1)} x_3 + b_2^{(1)} \\ W_{31}^{(1)} x_1 W + W_{32}^{(1)} x_2 + W_{33}^{(1)} x_3 + b_3^{(1)} \\ W_{41}^{(1)} x_1 W + W_{42}^{(1)} x_2 + W_{43}^{(1)} x_3 + b_4^{(1)} \end{bmatrix} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \\ z_4^{(2)} \end{bmatrix}$$

$$a^{(2)} = \begin{bmatrix} f(z_1^{(2)}) \\ f(z_2^{(2)}) \\ f(z_3^{(2)}) \\ f(z_4^{(2)}) \end{bmatrix} = \begin{bmatrix} a_1^{(2)} \\ a_2^{(2)} \\ a_3^{(2)} \\ a_4^{(2)} \end{bmatrix}$$

$$z_1^{(2)} = \sum_{j=1}^{3} W_{1j}^{(1)} x_j + b_1^{(1)} \qquad a_1^{(2)} = f(z_1^{(2)})$$

$$z_2^{(2)} = \sum_{j=1}^{3} W_{2j}^{(1)} x_j + b_2^{(1)} \qquad a_2^{(2)} = f(z_2^{(2)})$$

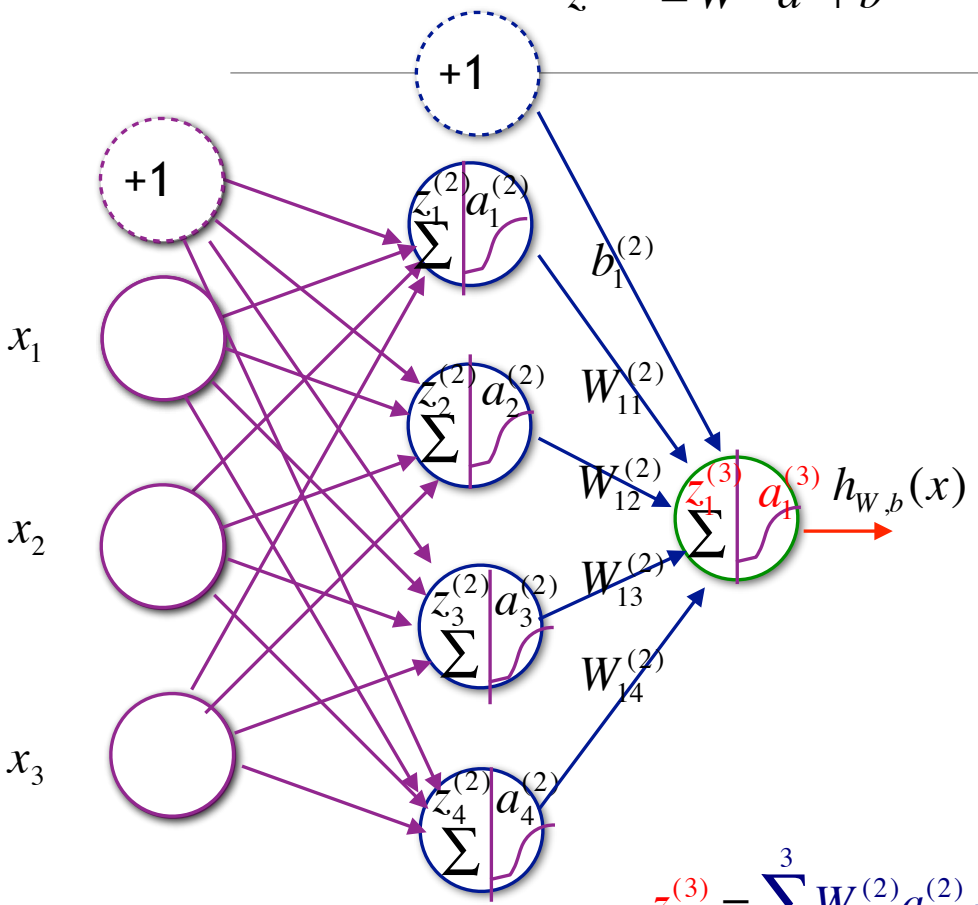
$$z_3^{(2)} = \sum_{j=1}^{3} W_{3j}^{(1)} x_j + b_3^{(1)} \qquad a_3^{(2)} = f(z_3^{(2)})$$

$$z_4^{(2)} = \sum_{j=1}^{3} W_{4j}^{(1)} x_j + b_4^{(1)} \qquad a_4^{(2)} = f(z_4^{(2)})$$

# Neural Network Model & Notation

$$z^{(l+1)} = W^{(l)} a^{l} + b^{(l)} \qquad a^{(l+1)} = f(z^{(l+1)}) = f(W^{(l)} a^{l} + b^{(l)})$$

$$f(z) = \frac{1}{1+\exp(-z)}$$




$$\begin{bmatrix} W_{11}^{(2)} & W_{12}^{(2)} & W_{13}^{(2)} & W_{14}^{(2)} \end{bmatrix} \begin{bmatrix} a_1^{(2)} \\ a_2^{(2)} \\ a_3^{(2)} \\ a_4^{(2)} \end{bmatrix} + \begin{bmatrix} b_1^{(2)} \end{bmatrix} = \begin{bmatrix} z_1^{(3)} \end{bmatrix}$$

$$a^{(3)} = \left[ f(z_1^{(2)}) \right] = \left[ a_1^{(3)} \right]$$

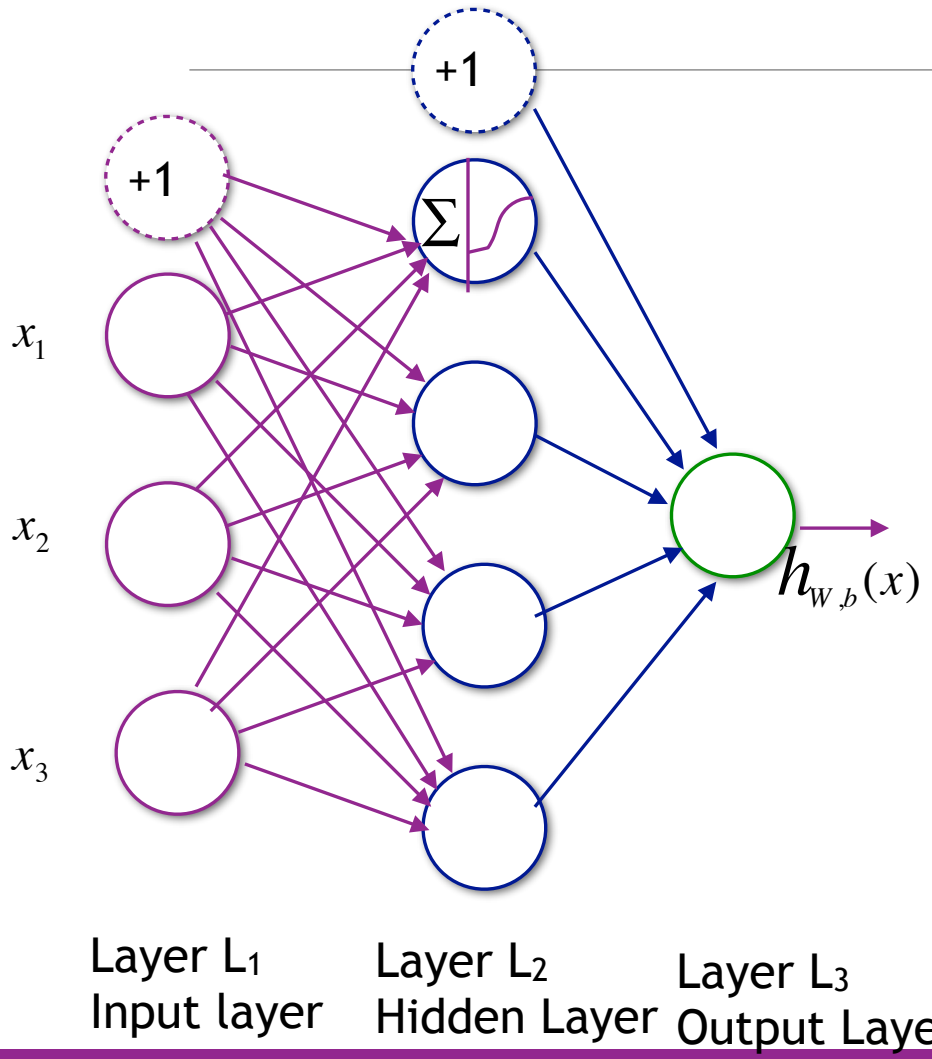
$$z^{(3)} = W^{(2)} a^{(2)} + b^{(2)}$$

$$a^{(2)} = f(z^{(3)}) = f(W^{(2)} x + b^{(2)})$$

$$z_1^{(3)} = \sum_{j=1}^{3} W_{1j}^{(2)} a_j^{(2)} + b_1^{(2)}$$

$$a_1^{(3)} = f(z_1^{(3)})$$

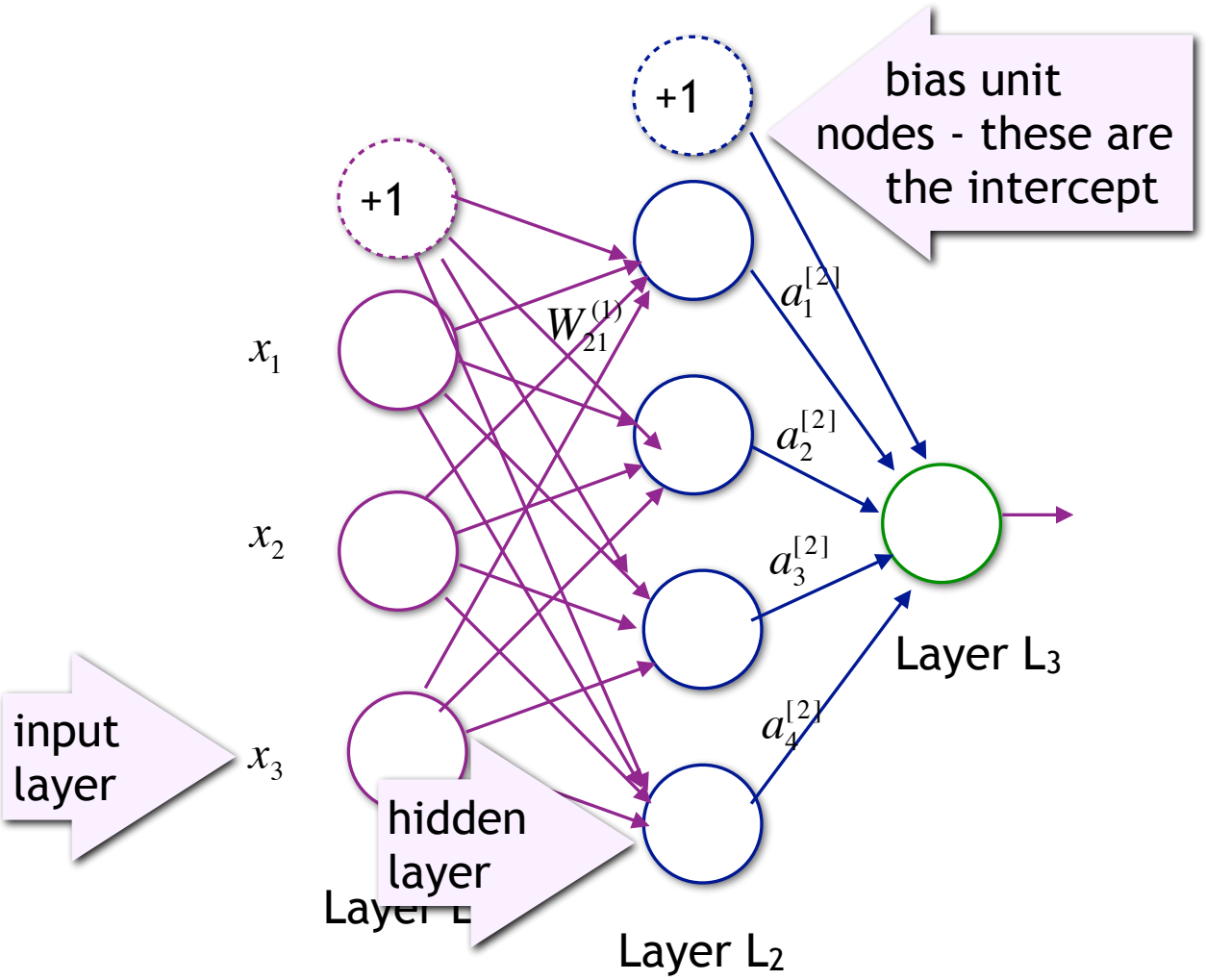
# Neural Network Model & Notation




- We can combine many of the simple "neurons" so the output of one is the input of another neuron
- Input layer units are set by **x**
- The input for all other nodes are typically the weighted sum of the activation on the links connected to this node
- The activation function is applied to the input to provide the value (we will use the sigmoid function

$$f(z) = \frac{1}{1+\exp(-z)}$$

# Neural Network Model & Notation




$$W_{ij}^{\ell} \quad \begin{cases} 1 \le \ell \le L & \text{layers} \\ 1 \le j \le s_{\ell-1} & \text{inputs} \\ 1 \le i \le s_{\ell} & \text{outputs} \end{cases}$$

- We can combine many of the simple "neurons" so the output of one is the input of another neuron
- The network has 3 input units (not 4), 4 hidden units (not 5) and 1 output unit
- $n_L$ is the number of layers in the network
- Each "neuron" has it own weights. We will store all the sigmoid weights for one level together $W^{(1)}$ such that $W_{ij}^{(1)}$ is the weights for leaving node j at level $L_1$ and entering node i at level $L_2$
- $b_i^{(1)}$ is the bias of unit i in level $L_2$

$$W^{(1)} \in \mathbb{R}^{4\times 3} \qquad W^{(2)} \in \mathbb{R}^{1\times 4}$$

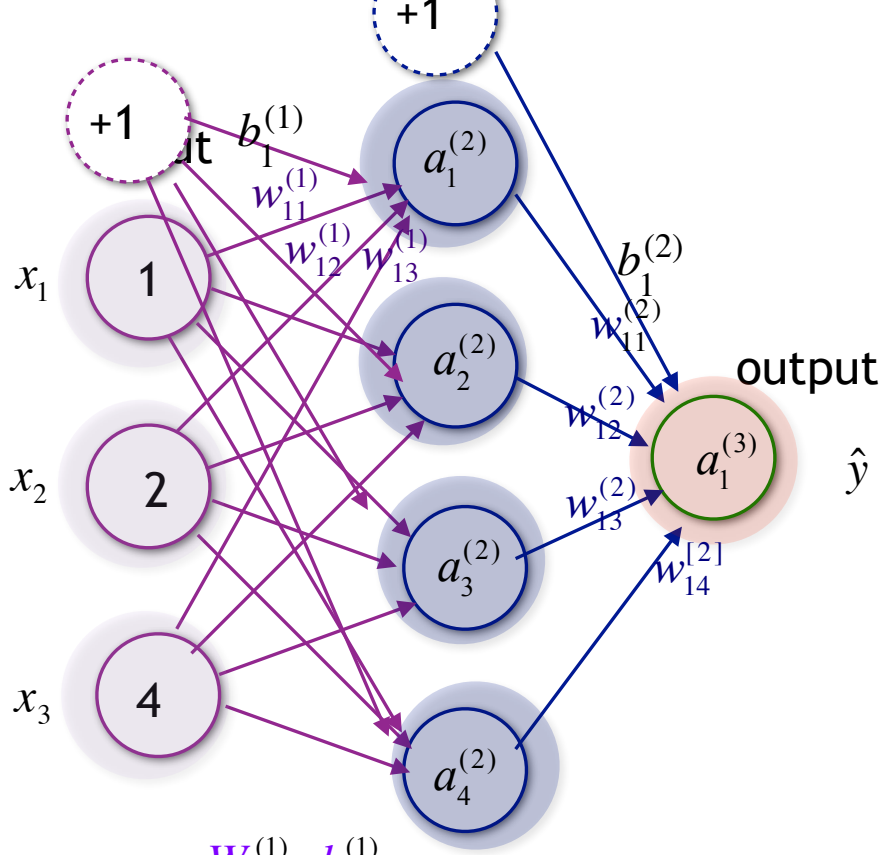
# Artificial Neural Networks

- Each node (we call the nodes *neurons*) represents a linear function of its input, similar to logistic regression

$$z_1^{(2)} = W_{11}^{(1)} x_1 + W_{12}^{(1)} x_2 + W_{13}^{(1)} x_3 + b_1^{(1)}$$
$$a_1^{(2)} = f(z_1^{(2)})$$

  - $W_{11}^{(1)}\ W_{12}^{(1)}\ W_{13}^{(1)}\ b_1^{(1)}$ are the ***weights*** (aka parameters)
  - $W^{(1)}$ is the matrix of weights that map values from input layer 1 to the first hidden layer (layer 2) ($W^{(\ell)}$ is the matrix that maps layer $\ell$ to layer $\ell$+1)
  - $b^{(1)}$ is the vector of bias values for the neurons on layer 1

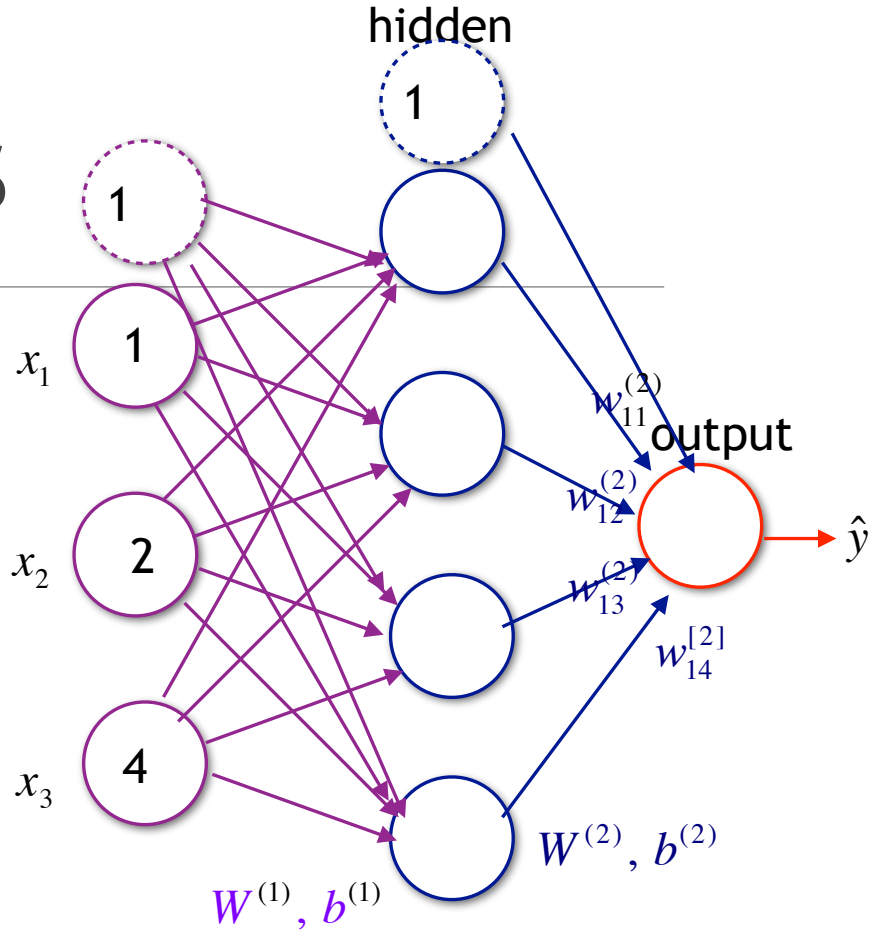
- $f^{(1)}$ is the "activation function" for layer 1. ($f^{(\ell)}$ is the activation function for layer $\ell$)
- $a_j^{(2)}$ is the activation value for the j$^{th}$ neuron of layer 2 ($a_j^{(\ell)}$ is the activation value for the j$^{th}$ neuron of layer the $\ell$




$$W^{(1)} = \begin{bmatrix} W_{11}^{(1)} & W_{12}^{(1)} & W_{13}^{(1)} \\ W_{21}^{(1)} & W_{22}^{(1)} & W_{23}^{(1)} \\ W_{31}^{(1)} & W_{32}^{(1)} & W_{33}^{(1)} \\ W_{41}^{(1)} & W_{42}^{(1)} & W_{43}^{(1)} \end{bmatrix} \quad b^{(1)} = \begin{bmatrix} b_1^{(1)} \\ b_2^{(1)} \\ b_3^{(1)} \\ b_4^{(1)} \end{bmatrix}$$

# Artificial Neural Networks

- Suppose we have a binary {0,1} classification problem with 3 input features
- Suppose we have been given a ***trained neural network*** (i.e. it is ready to predict)
- Suppose we are asked to predict the class of the following example $\mathbf{x} = (1, 2, 4)^T$
- To find the output, $\hat{y}$ , we use the information gained from:
  - input layer is x (layer $\ell$=1)
  - output layer is the last layer (layer $\ell$=3)
  - hidden layers are all the layers between input and output (in this case there is only one hidden layer $\ell$=2)
  - Notation: $s_\ell$ is the number of neurons in layer $\ell$

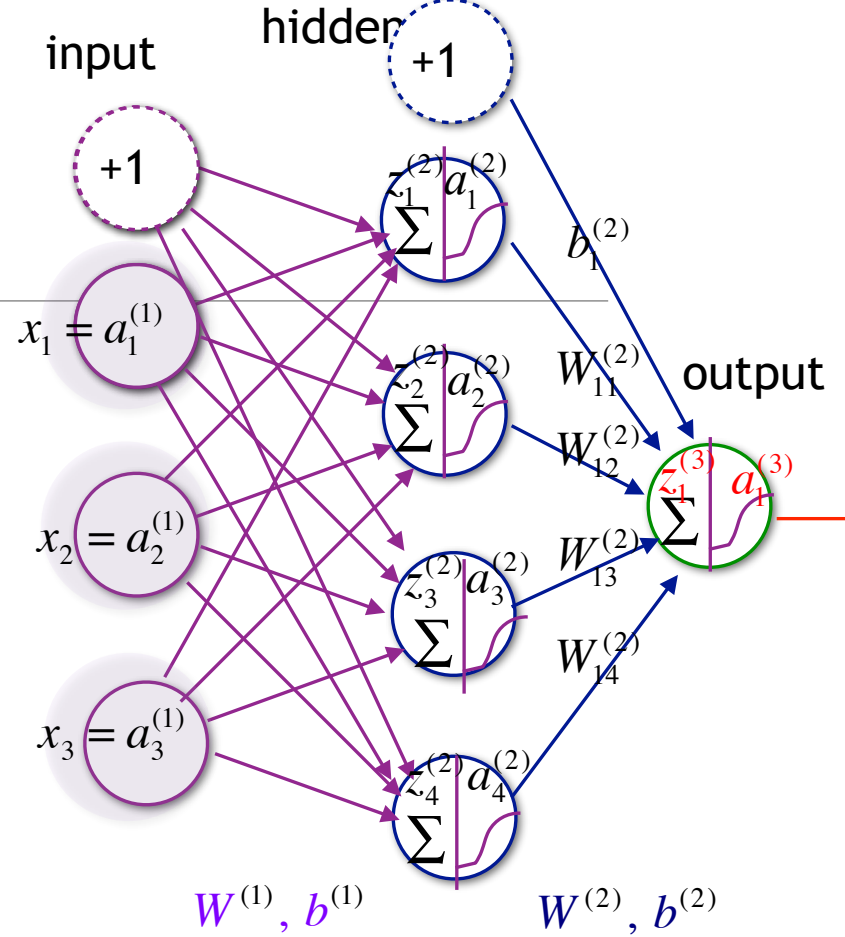


# Forward Propagation

$$z^{(l+1)} = W^{(l)} a^{(l)} + b^{(l)} \qquad a^{(l+1)} = f(z^{(l+1)}) = f(W^{(l)} a^{(l)} + b^{(l)})$$

- Forward propagation:

$$a^{(1)} = x$$
for $\ell$ = 1 to L do
$$z^{(\ell+1)} = W^{(\ell)} a^{(\ell)} + b^{(\ell)}$$
$$a^{(\ell+1)} = f(z^{(\ell+1)})$$
$$\hat{y} = a^{(n_L)}$$



# Forward Propagation

$$z^{(l+1)} = W^{(l)} a^{l} + b^{(l)} \qquad a^{(l+1)} = f(z^{(l+1)}) = f(W^{(l)} a^{l} + b^{(l)})$$

- Generalizing for an arbitrary neural network. Forward propagation:

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

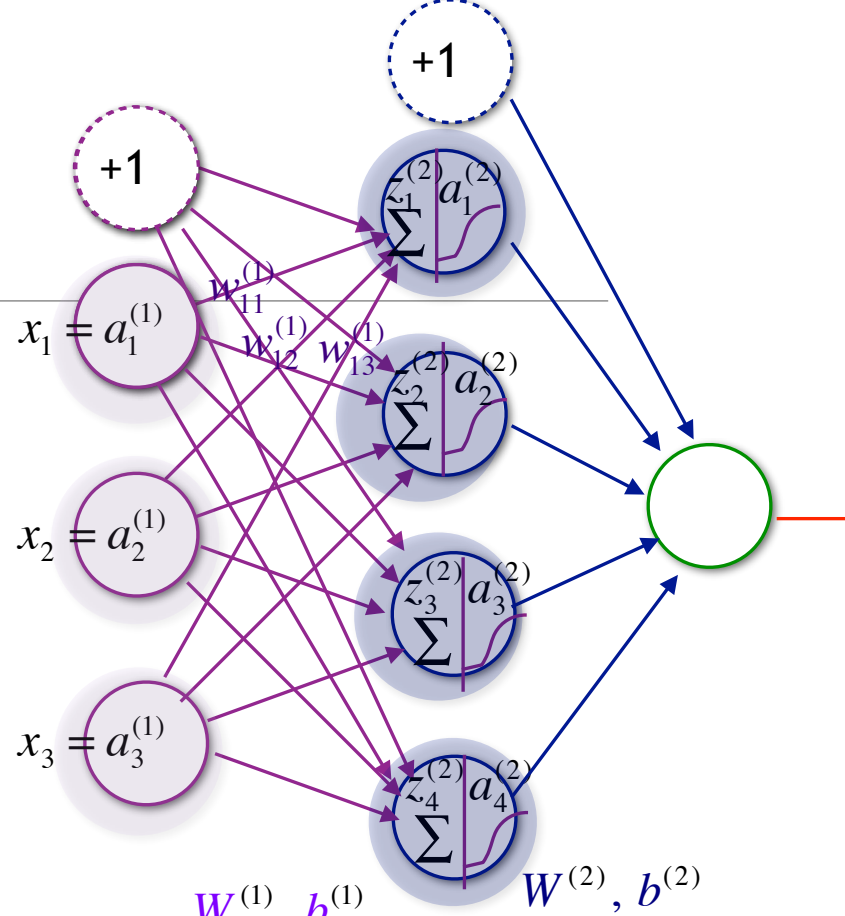
for $\ell$ = 1 to L do

$$z^{(\ell+1)} = W^{(\ell)} a^{(\ell)} + b^{\ell} = z^{(2)} = \begin{bmatrix} W_{11}^{(1)} & W_{12}^{(1)} & W_{13}^{(1)} \\ W_{21}^{(1)} & W_{22}^{(1)} & W_{23}^{(1)} \\ W_{31}^{(1)} & W_{32}^{(1)} & W_{33}^{(1)} \\ W_{41}^{(1)} & W_{42}^{(1)} & W_{43}^{(1)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1^{(1)} \\ b_2^{(1)} \\ b_3^{(1)} \\ b_4^{(1)} \end{bmatrix}$$

$$a^{(\ell+1)} = f(z^{(\ell+1)})$$

$$\hat{y} = a^{(n_L)}$$

$$a^{(2)} = \begin{bmatrix} f(z_1^{(2)}) \\ f(z_2^{(2)}) \\ f(z_3^{(2)}) \\ f(z_4^{(2)}) \end{bmatrix} = \begin{bmatrix} a_1^{(2)} \\ a_2^{(2)} \\ a_3^{(2)} \\ a_4^{(2)} \end{bmatrix} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \\ z_4^{(2)} \end{bmatrix}$$



# Forward Propagation

$$z^{(l+1)} = W^{(l)} a^{l} + b^{(l)} \qquad a^{(l+1)} = f(z^{(l+1)}) = f(W^{(l)} a^{l} + b^{(l)})$$

- Generalizing for an arbitrary neural network. Forward propagation:

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

for $\ell$ = 1 to L do

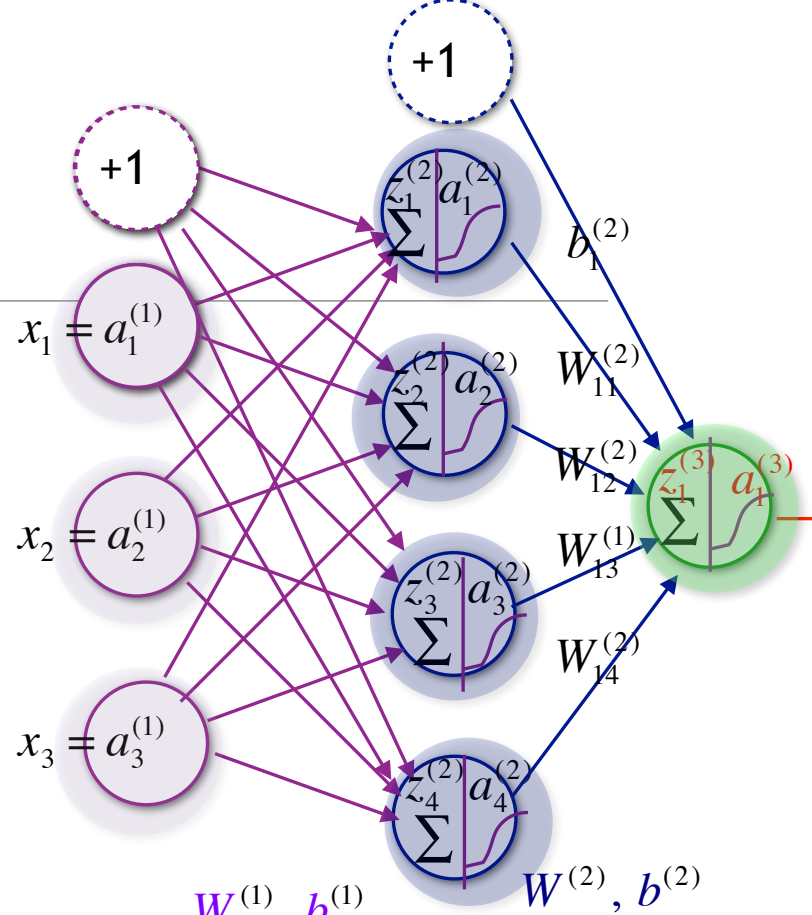
$$z^{(\ell+1)} = W^{(\ell)} a^{(\ell)} + b^{\ell} = z^{(2)}$$

$$a^{(\ell+1)} = f(z^{(\ell+1)})$$

$$\hat{y} = a^{(n_L)}$$

$$z^{(3)} = \begin{bmatrix} W_{11}^{(2)} & W_{12}^{(2)} & W_{13}^{(2)} & W_{14}^{(2)} \end{bmatrix} \begin{bmatrix} a_1^{(2)} \\ a_2^{(2)} \\ a_3^{(2)} \\ a_4^{(2)} \end{bmatrix} + \begin{bmatrix} b_1^{(2)} \end{bmatrix} = \begin{bmatrix} z_1^{(3)} \end{bmatrix}$$

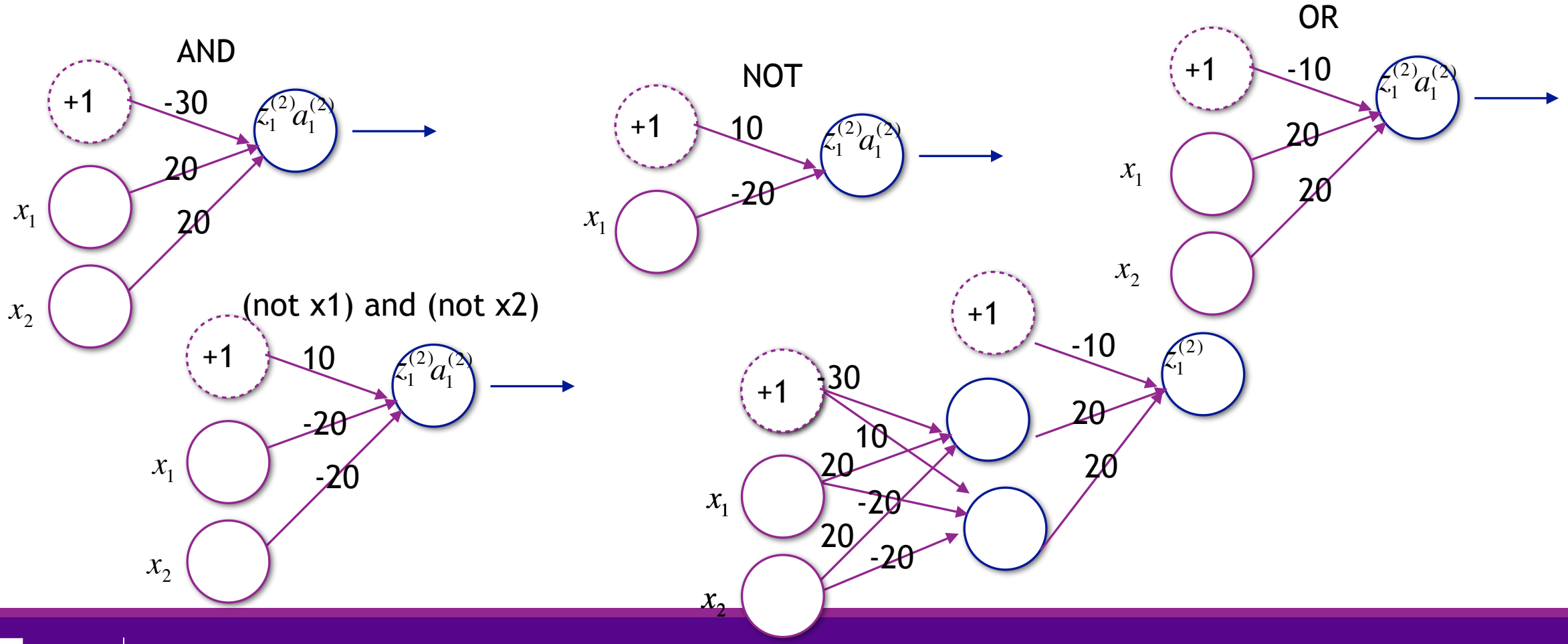
$$a^{(3)} = \begin{bmatrix} f(z_1^{(2)}) \end{bmatrix} = \begin{bmatrix} a_1^{(3)} \end{bmatrix}$$



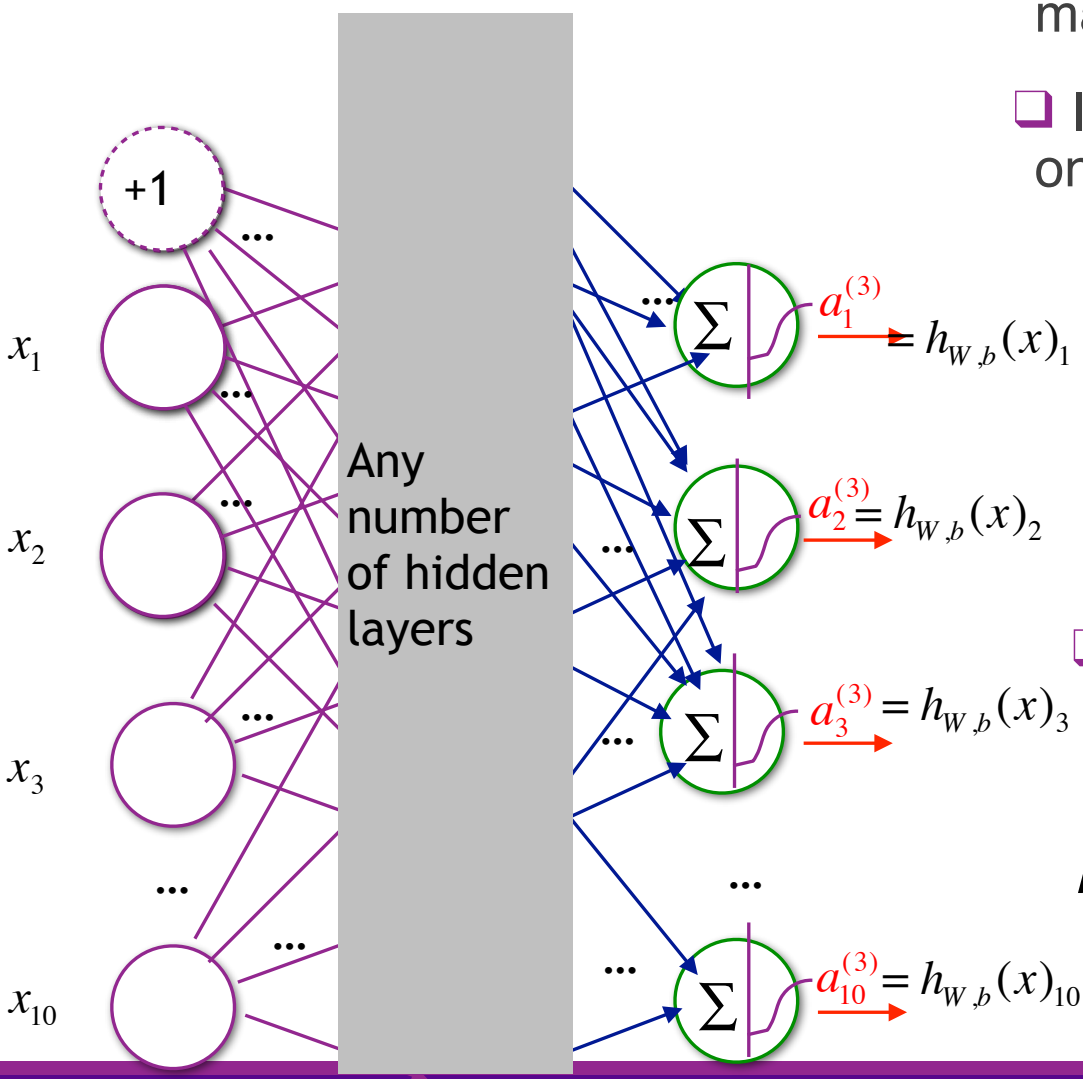
# Computing Boolean Functions

https://excel.ucf.edu/classes/2007/Spring/appsII/Chapter1.pdf

$$f(z) = \frac{1}{1+\exp(-z)}$$



# Multiple Output Units: One-vs-All




- Instead of just computing one output, we can compute as many as we would like
- If we were classifying digits, we could have ten outputs, one for each digit - using a 1 of k representation

$$y = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad y = \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix} \quad \cdots \quad y = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

- For our hypothesis $h_{W,b}(x) \in \mathbb{R}^K$ we want:

$$h_{W,b}(x) = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad h_{W,b}(x) = \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix} \quad \cdots \quad h_{W,b}(x) = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

# Extra Slides about SVM's

# Multi-Class SVMs

- Suppose there are $K$ classes
- One-vs-one:
  - Train $\binom{K}{2}$ SVMs for each pair of classes
  - Test sample assigned to class that wins "majority of votes"
  - Used in sklearn. Best results but very slow
- One-vs-all:
  - Train $K$ SVMs: train each class $k$ against all other classes
  - Pick class with highest $z_k$

For additional resources see:
https://www.youtube.com/watch?v=6kzvrq-MIO0

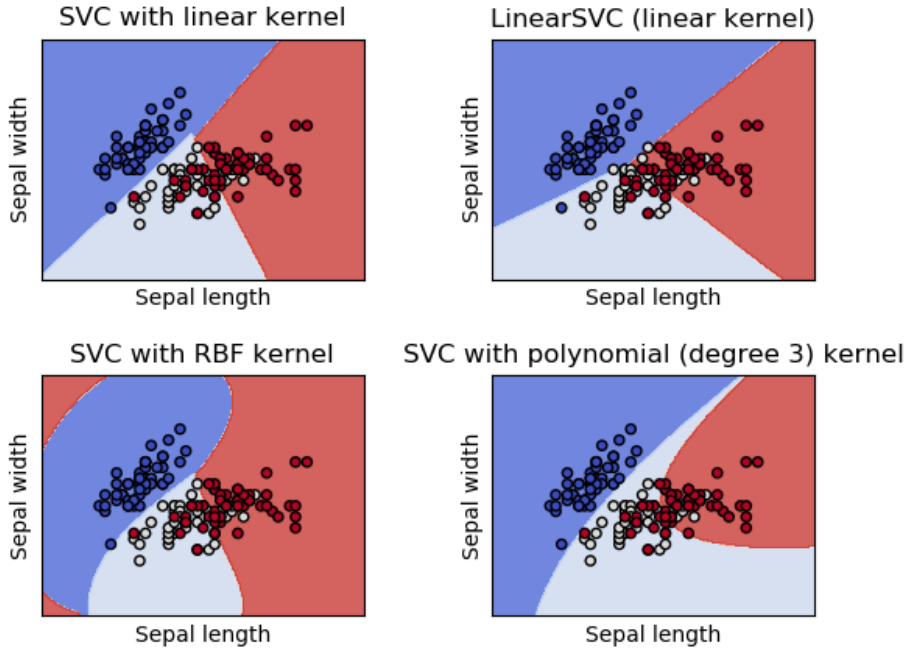
http://l2r.cs.uiuc.edu/Teaching/CS446-17/LectureNotesNew/multiclass/main.pdf

# Sklearn multi-class classification

Pictures and (modified) code from http://scikit-learn.org/stable/modules/svm.html

"**SVC** and **NuSVC** implement the "one-against-one" approach (Knerr et al., 1990) for multi- class classification. If `n_class` is the number of classes, then `n_class * (n_class - 1) / 2` classifiers are constructed and each one trains data from two classes. To provide a consistent interface with other classifiers, the `decision_function_shape` option allows to aggregate the results of the "one-against-one" classifiers to a decision function of shape `(n_samples, n_classes)`:"

"On the other hand, **LinearSVC** implements "one-vs-the-rest" multi-class strategy, thus training n_class models. If there are only two classes, only one model is trained"




**SVC** one-against-one (aka one-verses-one) example

```
>>> X = [[0], [1], [2], [3]]
>>> Y = [0, 1, 2, 3]
>>> clf = svm.SVC() # OVO
>>> clf.fit(X, Y)
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovo', degree=3, gamma='auto', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
>>> clf.predict(2.2)
    array([2])
```