COMPUTER SCIENCE
UNIVERSITY OF MARYLAND

From Logistic Regression to Neural Networks

CMSC 470

Marine Carpuat

Logistic Regression
What you should know

How to make a prediction with logistic regression classifier

How to train a logistic regression classifier

Machine learning concepts:

Loss function

Gradient Descent Algorithm

function STOCHASTIC GRADIENT DESCENT($L()$, $f()$, x, y) **returns** θ

```
# where: L is the loss function
#       f is a function parameterized by θ
#       x is the set of training inputs x^(1), x^(2), ..., x^(n)
#       y is the set of training outputs (labels) y^(1), y^(2), ..., y^(n)
```

$\theta \leftarrow 0$

repeat til done # see caption

For each training tuple $(x^{(i)}, y^{(i)})$ (in random order)

1. Optional (for reporting): # How are we doing on this tuple?

   Compute $\hat{y}^{(i)} = f(x^{(i)};\theta)$ # What is our estimated output $\hat{y}$?

   Compute the loss $L(\hat{y}^{(i)}, y^{(i)})$ # How far off is $\hat{y}^{(i)}$) from the true output $y^{(i)}$?

2. $g \leftarrow \nabla_{\theta} L(f(x^{(i)};\theta), y^{(i)})$ # How should we move θ to maximize loss?

3. $\theta \leftarrow \theta - \eta\, g$ # Go the other way instead

return θ

Figure 5.5 The stochastic gradient descent algorithm. Step 1 (computing the loss) is used to report how well we are doing on the current tuple. The algorithm can terminate when it converges (or when the gradient $< \epsilon$), or when progress halts (for example when the loss starts going up on a held-out set).

SGD hyperparameter: the learning rate

- The hyperparameter η that control the size of the step down the gradient is called the learning rate
- If η is too large, training might not converge; if η is too small, training might be very slow.
- How to set the learning rate? Common strategies:
  - decay over time: $\eta = \frac{1}{C+t}$

    Constant hyperparameter set by user

    Number of samples
  - Use held-out test set, increase learning rate when likelihood increases

Multiclass Logistic Regression

Formalizing classification

Task definition

- *Given inputs*:
  - an example x

    often x is a D-dimensional vector of binary or real values
  - a fixed set of classes Y

    $Y = \{y_1, y_2, \ldots, y_J\}$

    e.g. word senses from WordNet
- *Output*: a predicted class $y \in Y$

So far we've only worked with binary classification problems i.e. J = 2

Classifier definition

A function $g: x \rightarrow g(x) = y$

Many different types of functions/classifiers can be defined

- We'll talk about perceptron, logistic regression, neural networks.

A multiclass logistic regression classifier

aka multinomial logistic regression, softmax logistic regression, maximum entropy (or maxent) classifier

Goal: predict probability $P(y=c|x)$, where c is one of k classes in set C

The softmax function

- A generalization of the sigmoid
- Input: a vector z of dimensionality k

$$z = [z_1, z_2, \ldots, z_k]$$

- Output: a vector of dimensionality k

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} \quad 1 \le i \le k$$

$$\text{softmax}(z) = \left[\frac{e^{z_1}}{\sum_{i=1}^{k} e^{z_i}}, \frac{e^{z_2}}{\sum_{i=1}^{k} e^{z_i}}, \ldots, \frac{e^{z_k}}{\sum_{i=1}^{k} e^{z_i}} \right]$$

Looks like a probability distribution!

The softmax function
Example

Thus for example given a vector:

$$z = [0.6, 1.1, -1.5, 1.2, 3.2, -1.1]$$

the result softmax(z) is

$$[0.055, 0.090, 0.0067, 0.10, 0.74, 0.010]$$

All values are in [0,1] and sum up to 1: they can be interpreted as probabilities!

A multiclass logistic regression classifier

aka multinomial logistic regression, softmax logistic regression, maximum entropy (or maxent) classifier

Goal: predict probability P(y=c|x), where c is one of k classes in set C

Model definition:

$$p(y=c|x) = \frac{e^{w_c \cdot x + b_c}}{\sum_{j=1}^{k} e^{w_j \cdot x + b_j}}$$

We now have one weight vector and one bias PER CLASS

Features in multiclass logistic regression

- Features are a function of the input example **and of a candidate output class c**
- $f_i(c, x)$ represents feature i for a particular class c for a given example x

Example: sentiment analysis with 3 classes {positive (+), negative (-), neutral (0)}

- Starting from the features for binary classification
- We create one copy of each feature per class

Var	Definition	Wt
$f_1(0,x)$	$\begin{cases} 1 & \text{if “!”} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	-4.5
$f_1(+,x)$	$\begin{cases} 1 & \text{if “!”} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	2.6
$f_1(-,x)$	$\begin{cases} 1 & \text{if “!”} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	1.3

Learning in Multiclass Logistic Regression

- Loss function for a single example

$$L_{CE}(\hat{y}, y) = -\sum_{k=1}^{K} 1\{y = k\} \log p(y = k|x)$$

1{ } is an indicator function that evaluates to 1 if the condition in the brackets is true, and to 0 otherwise

Learning in Multiclass Logistic Regression

- Loss function for a single example

$$
\begin{aligned}
L_{CE}(\hat{y}, y) &= -\sum_{k=1}^{K} 1\{y = k\} \log p(y = k|x) \\
&= -\sum_{k=1}^{K} 1\{y = k\} \log \frac{e^{w_k \cdot x + b_k}}{\sum_{j=1}^{K} e^{w_j \cdot x + b_j}}
\end{aligned}
$$

Learning in Multiclass Logistic Regression

$$
\begin{aligned}
L_{CE}(\hat{y}, y) &= -\sum_{k=1}^{K} 1\{y=k\} \log p(y=k|x) \\
&= -\sum_{k=1}^{K} 1\{y=k\} \log \frac{e^{w_k \cdot x + b_k}}{\sum_{j=1}^{K} e^{w_j \cdot x + b_j}}
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial L_{CE}}{\partial w_k} &= -(1\{y=k\} - p(y=k|x))x_k \\
&= -\left(1\{y=k\} - \frac{e^{w_k \cdot x + b_k}}{\sum_{j=1}^{K} e^{w_j \cdot x + b_j}}\right) x_k
\end{aligned}
$$

Logistic Regression
What you should know

How to make a prediction with logistic regression classifier

How to train a logistic regression classifier

For both binary and multiclass problems

Machine learning concepts:

Loss function

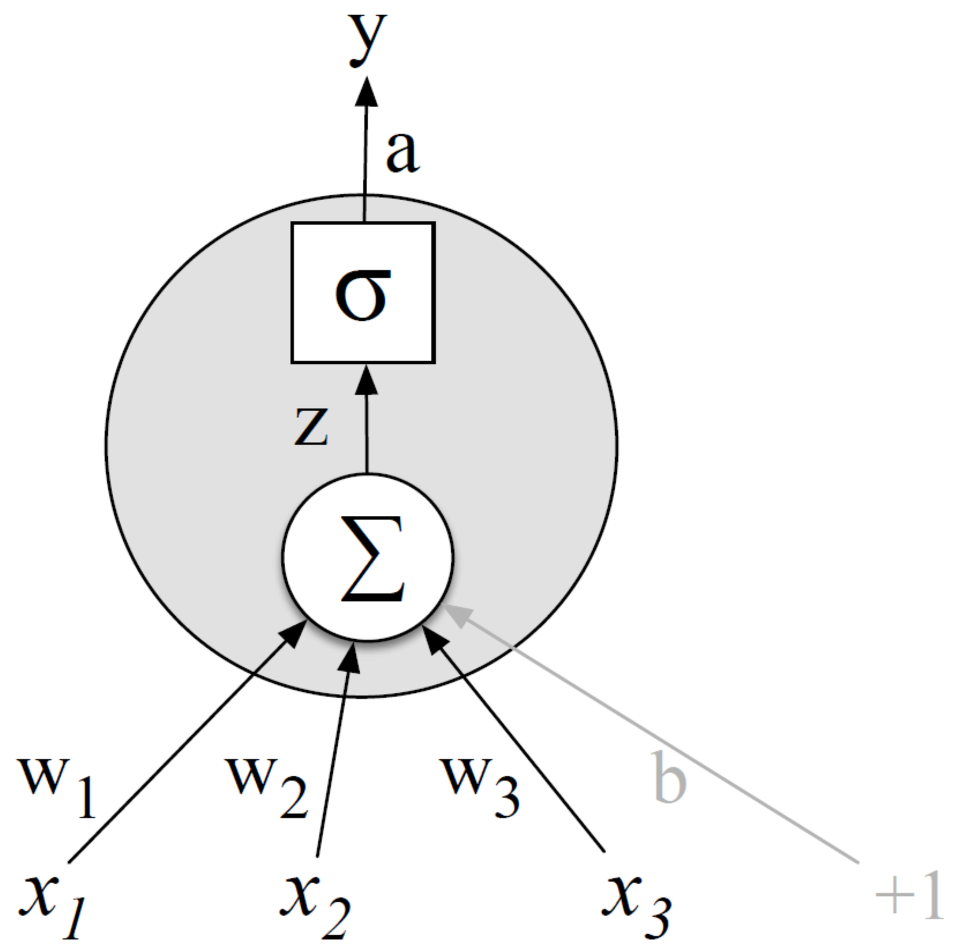
Gradient Descent Algorithm

Learning rate

Neural Networks

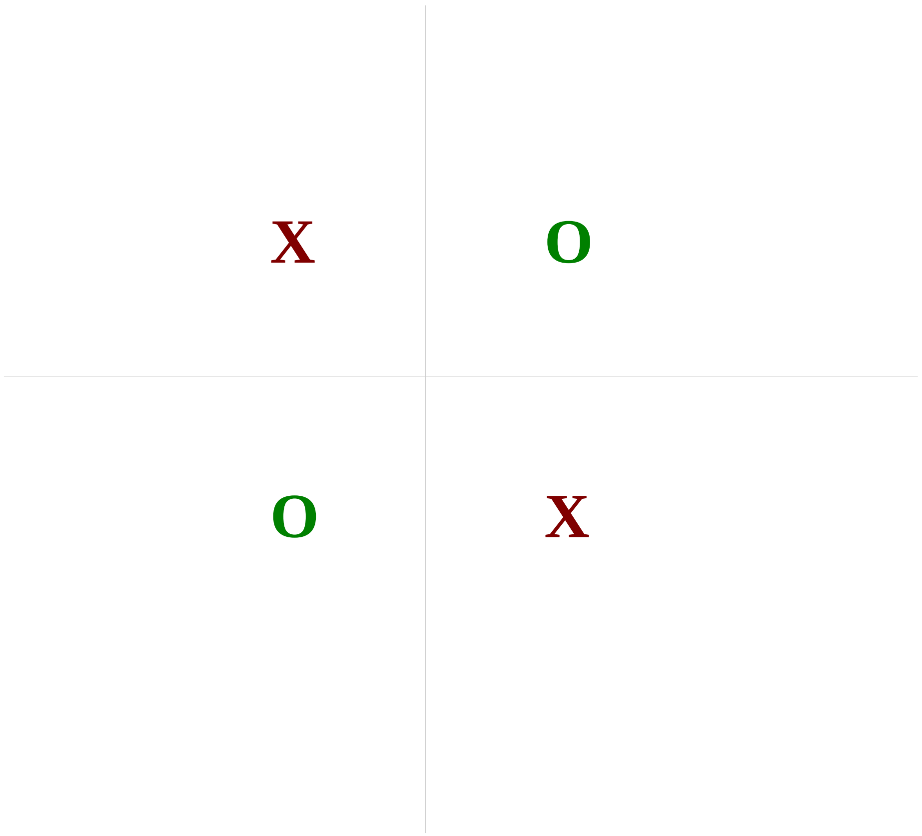
From logistic regression to a neural network unit

$$y = \sigma(w \cdot x + b) = \frac{1}{1 + \exp(-(w \cdot x + b))}$$



Limitation of perceptron

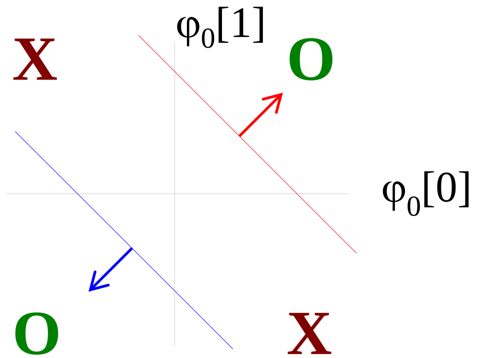
- can only find **linear separations** between positive and negative examples



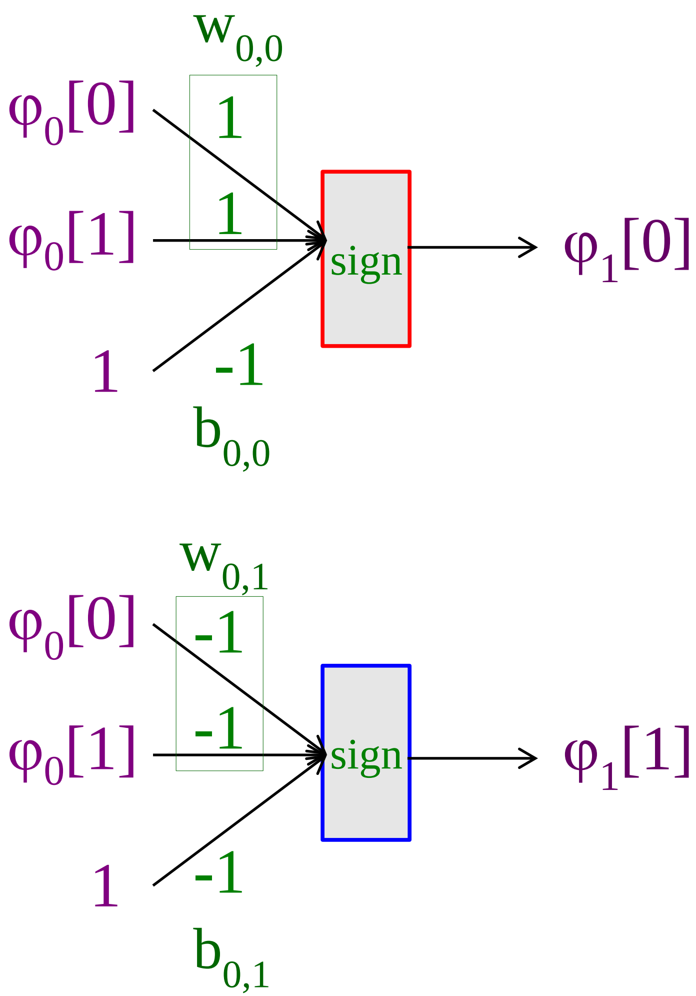
Example: binary classification with a neural network

- Create two classifiers

$\varphi_0(x_1) = \{-1, 1\}$ $\varphi_0(x_2) = \{1, 1\}$

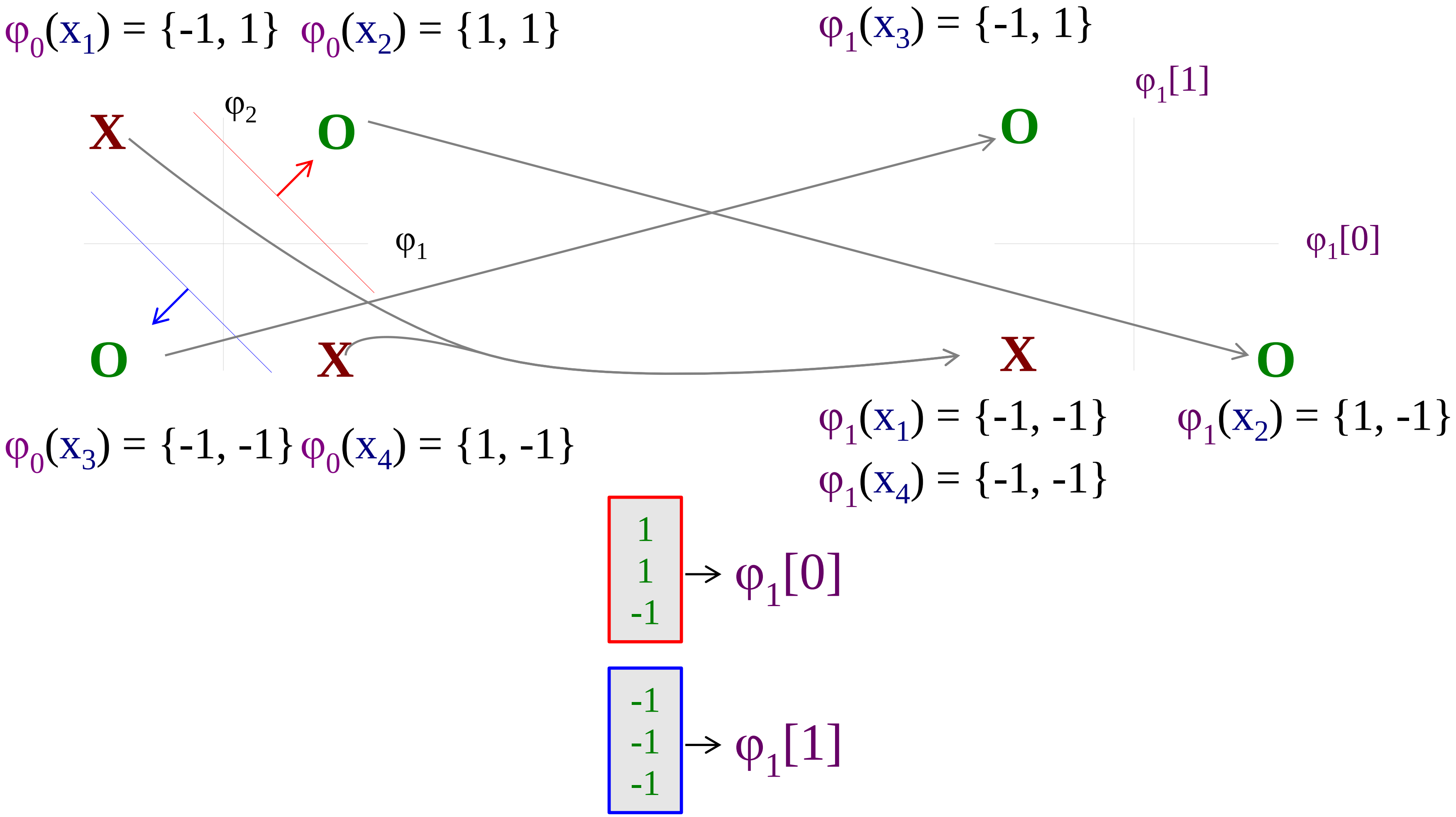


$\varphi_0(x_3) = \{-1, -1\}$ $\varphi_0(x_4) = \{1, -1\}$

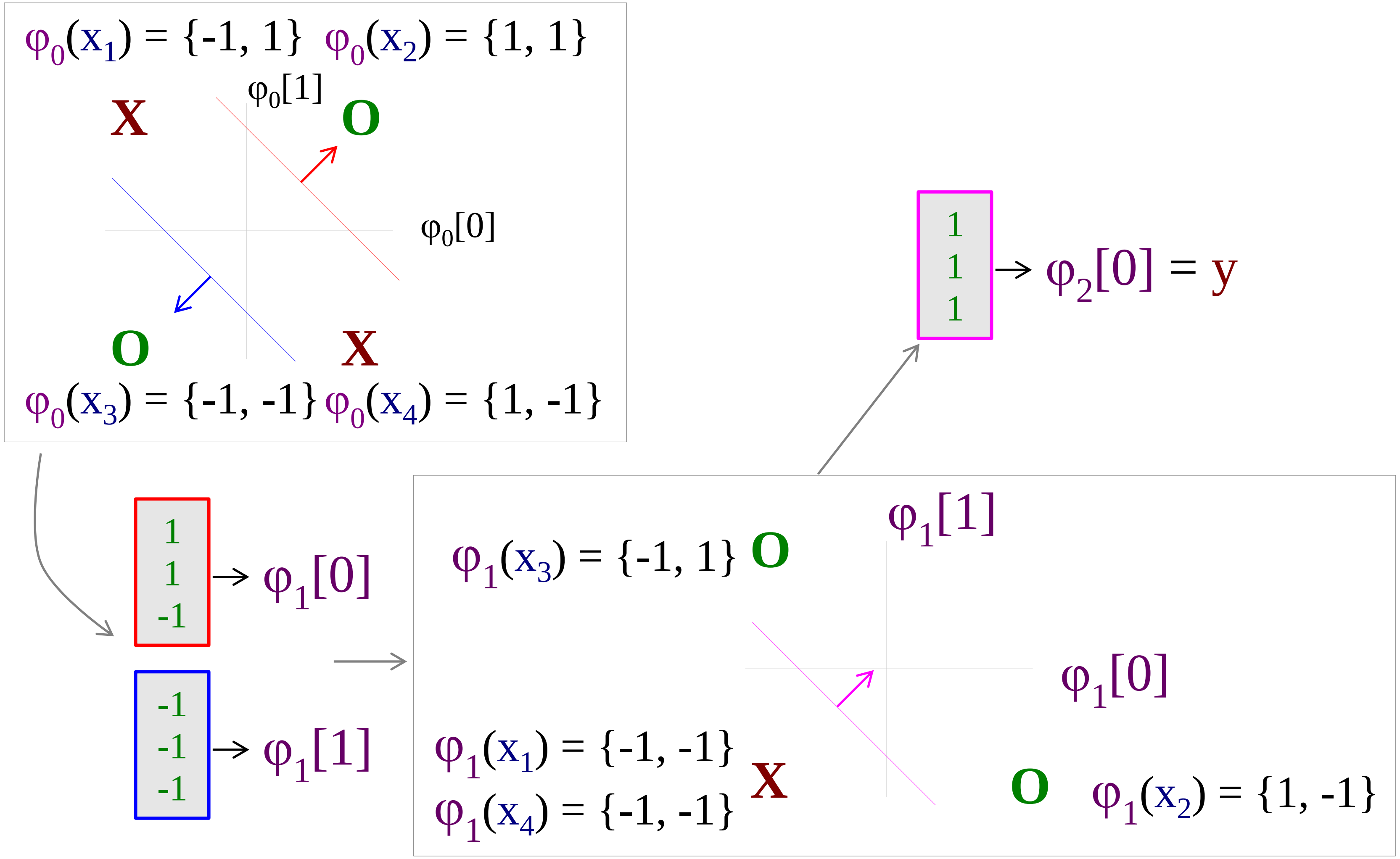


Example: binary classification with a neural network

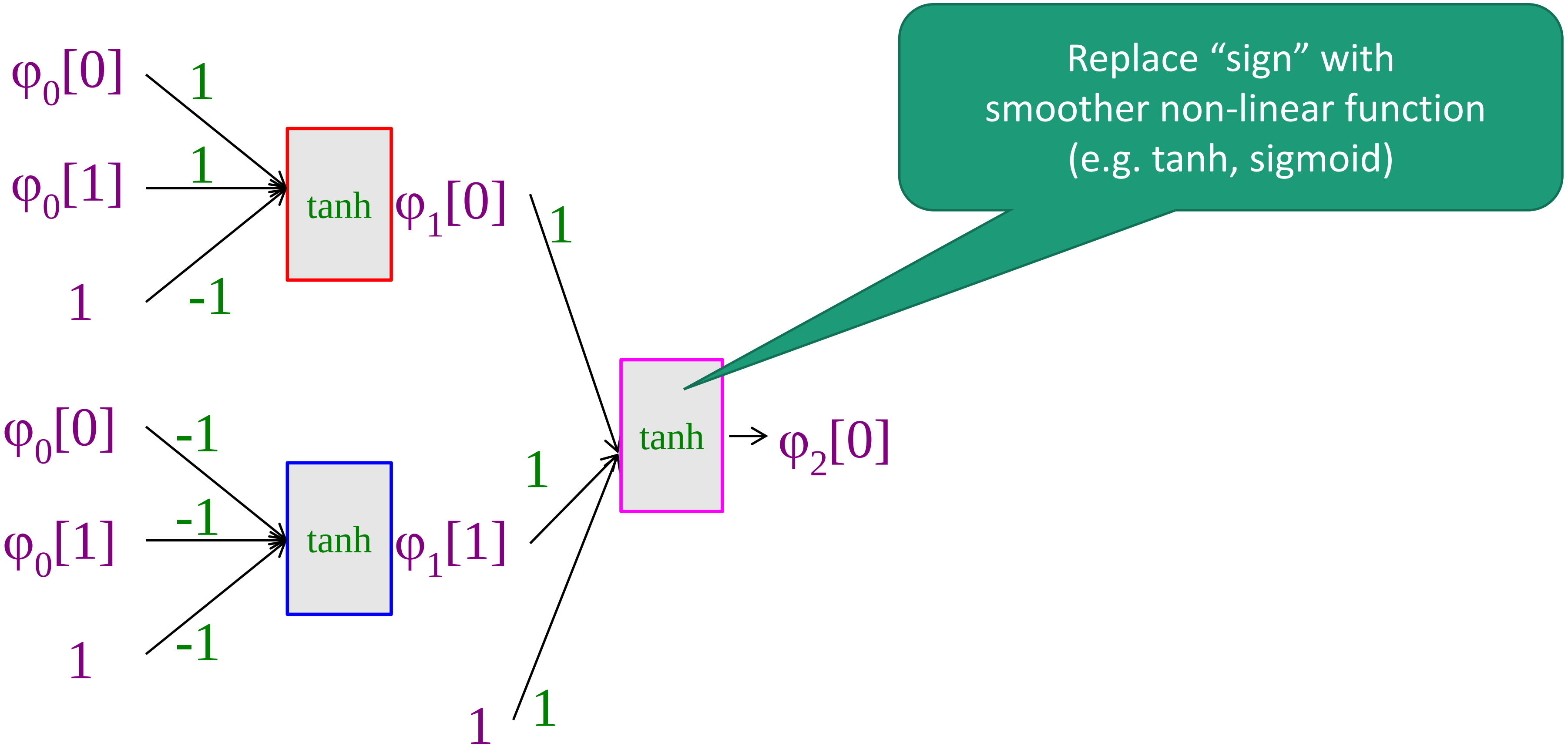
- These classifiers map to a new space



Example: binary classification with a neural network



Example: the final network can correctly classify the examples that the perceptron could not.



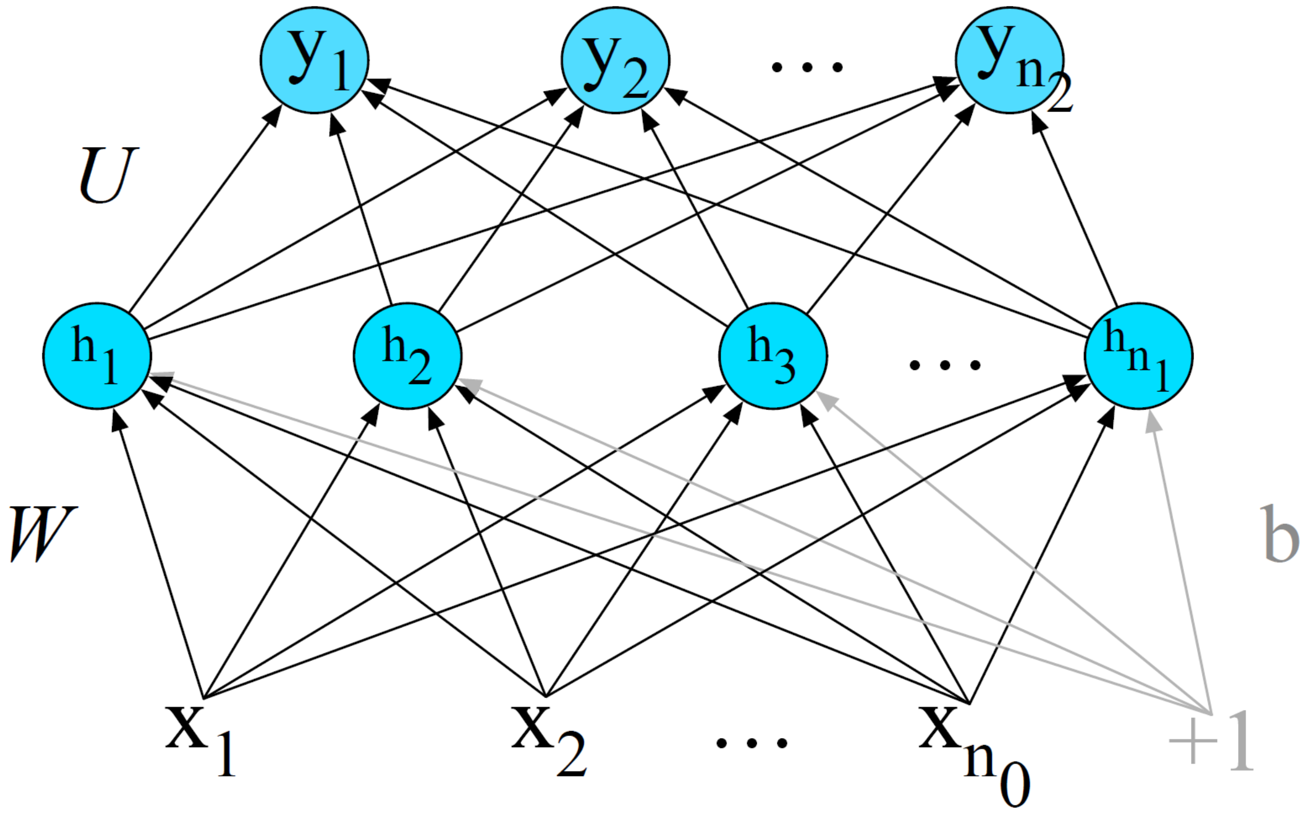
Feedforward Neural Networks

Components:

- an input layer
- an output layer
- one or more hidden layers

In a fully connected network: each hidden unit takes as input all the units in the previous laye

No loops!



A 2-layer feedforward neural network

Designing Neural Networks: Activation functions

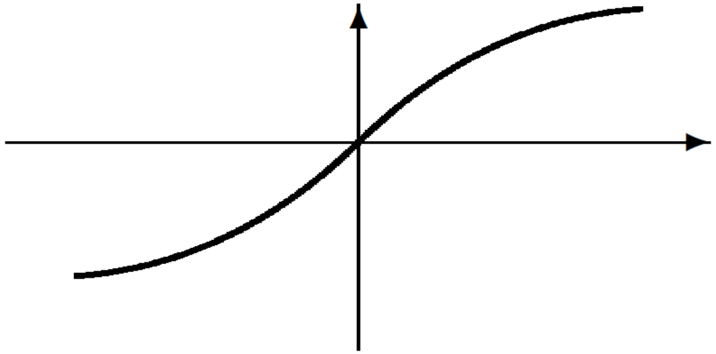
- Hidden layer can be viewed as set of hidden features
- The output of the hidden layer indicates the extent to which each hidden feature is “activated” by a given input
- The activation function is a non-linear function that determines range of hidden feature values

Designing Neural Networks: Activation functions

Hyperbolic tangent

$$\tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

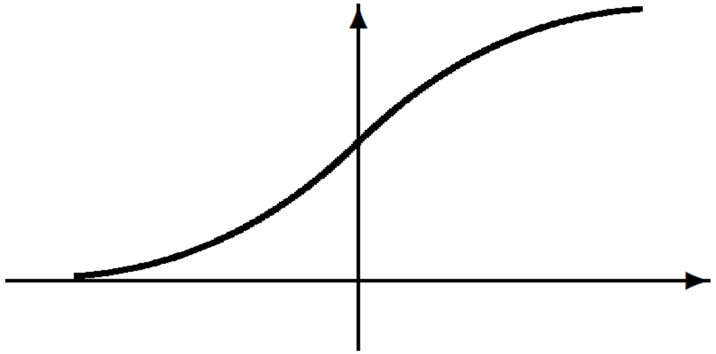
output ranges
from –1 to +1



Logistic function

$$\text{sigmoid}(x) = \frac{1}{1+e^{-x}}$$

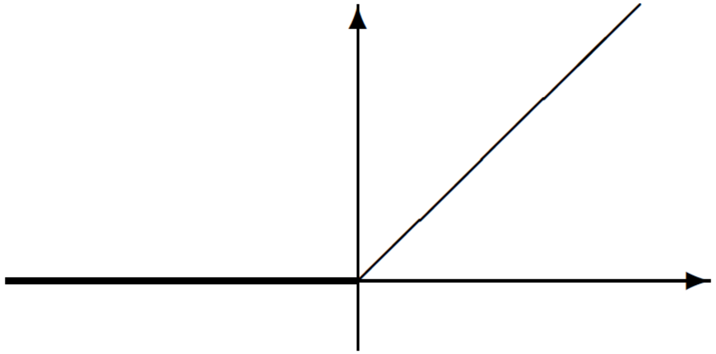
output ranges
from 0 to +1



Rectified linear unit

$$\text{relu}(x) = \max(0,x)$$

output ranges
from 0 to ∞

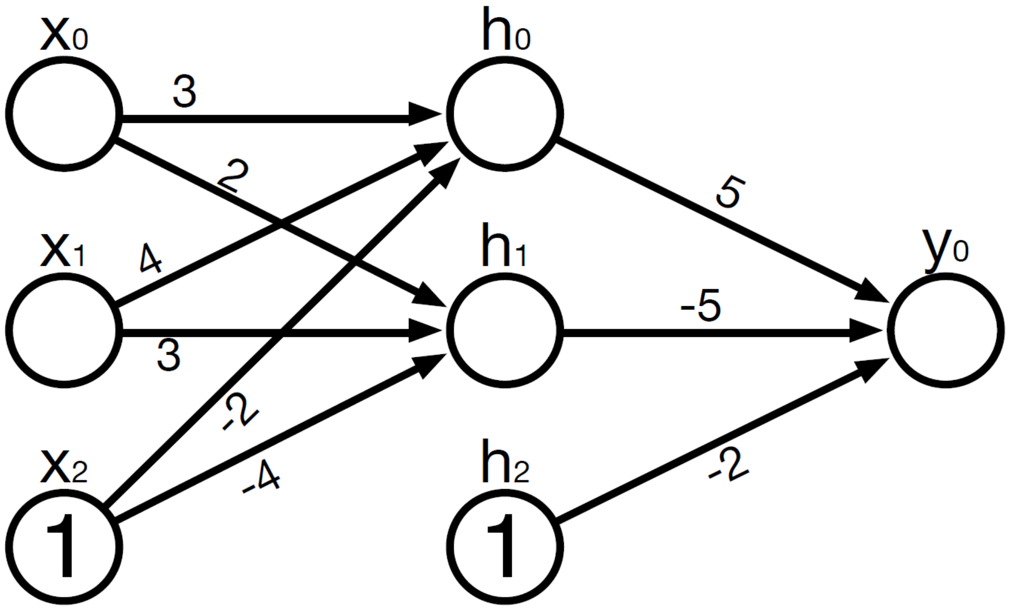


Designing Neural Networks: Network structure

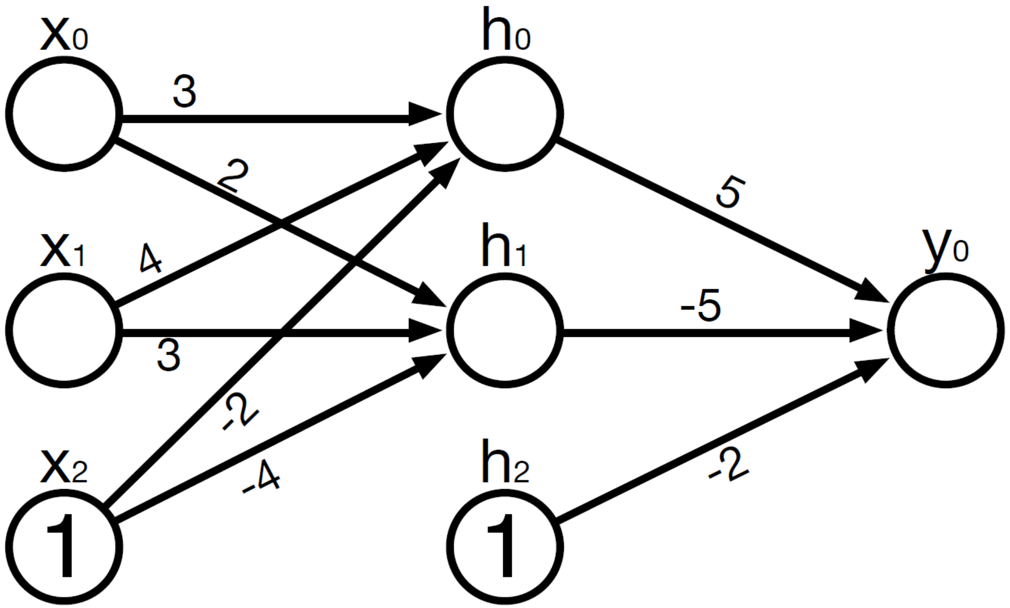
- 2 key decisions:
  - Width (number of nodes per layer)
  - Depth (number of hidden layers)

- More parameters means that the network can learn more complex functions of the input

Forward Propagation: For a given network, and some input values, compute output



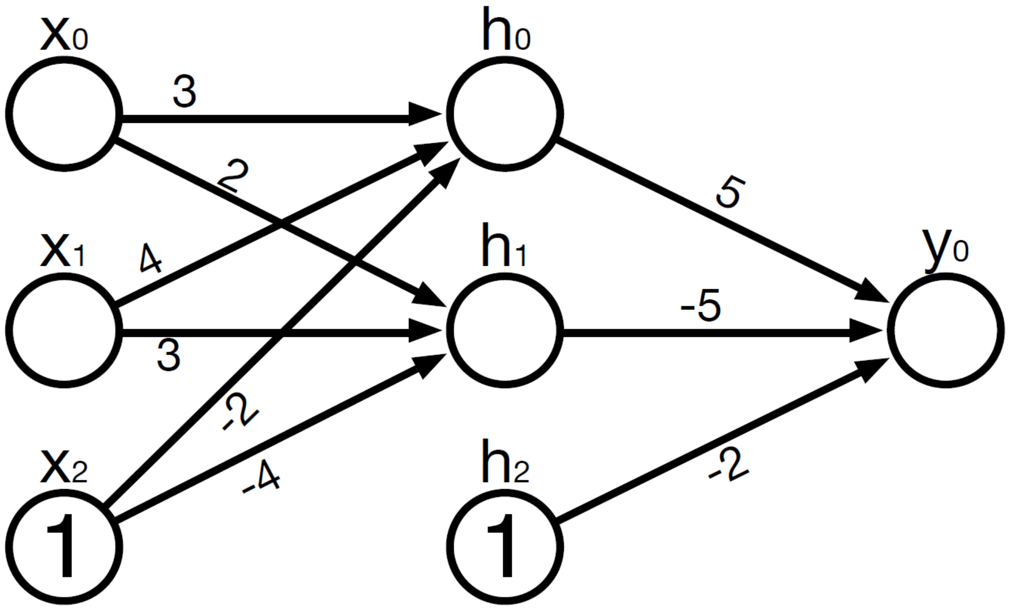
Forward Propagation: For a given network, and some input values, compute output



Given input (1,0) (and sigmoid non-linearities), we can calculate the output by processing one layer at a time:

Layer	Node	Summation	Activation
hidden	h_0	$1 \times 3 + 0 \times 4 + 1 \times -2 = 1$	0.731
hidden	h_1	$1 \times 2 + 0 \times 3 + 1 \times -4 = -2$	0.119
output	y_0	$0.731 \times 5 + 0.119 \times -5 + 1 \times -2 = 1.060$	0.743

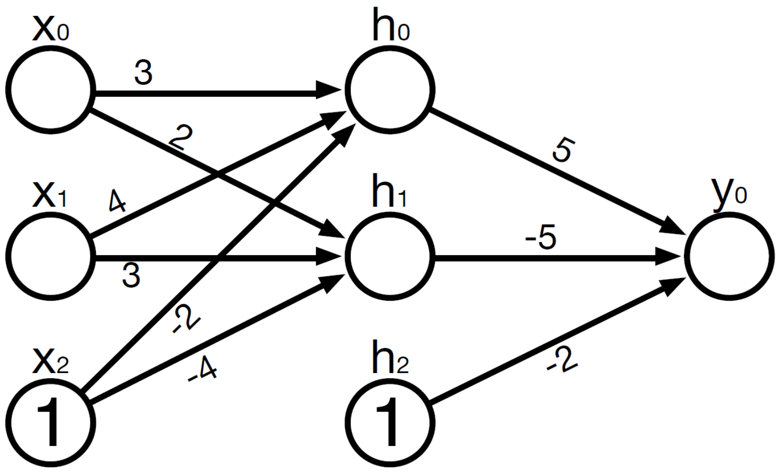
Forward Propagation: For a given network, and some input values, compute output

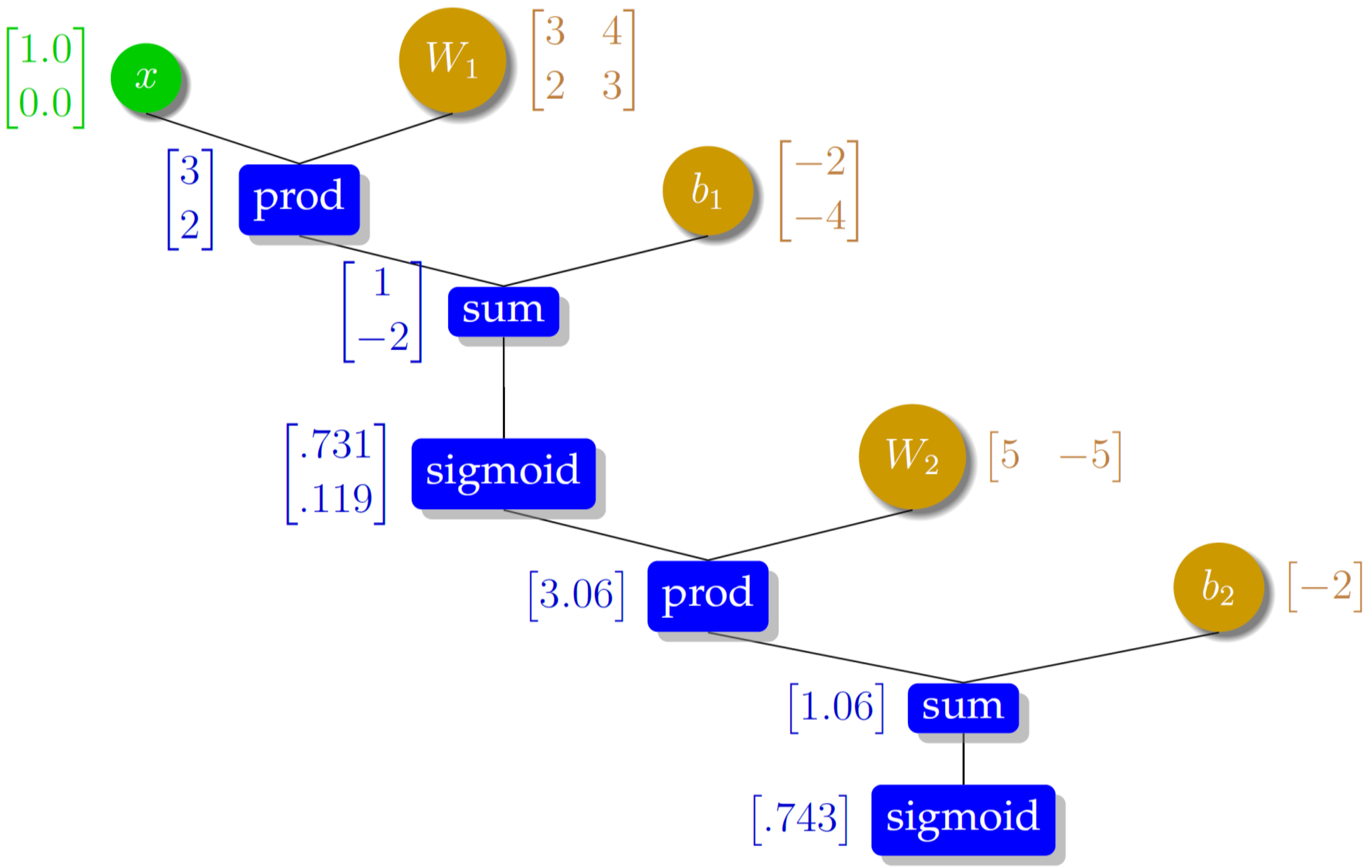


Output table for all possible inputs:

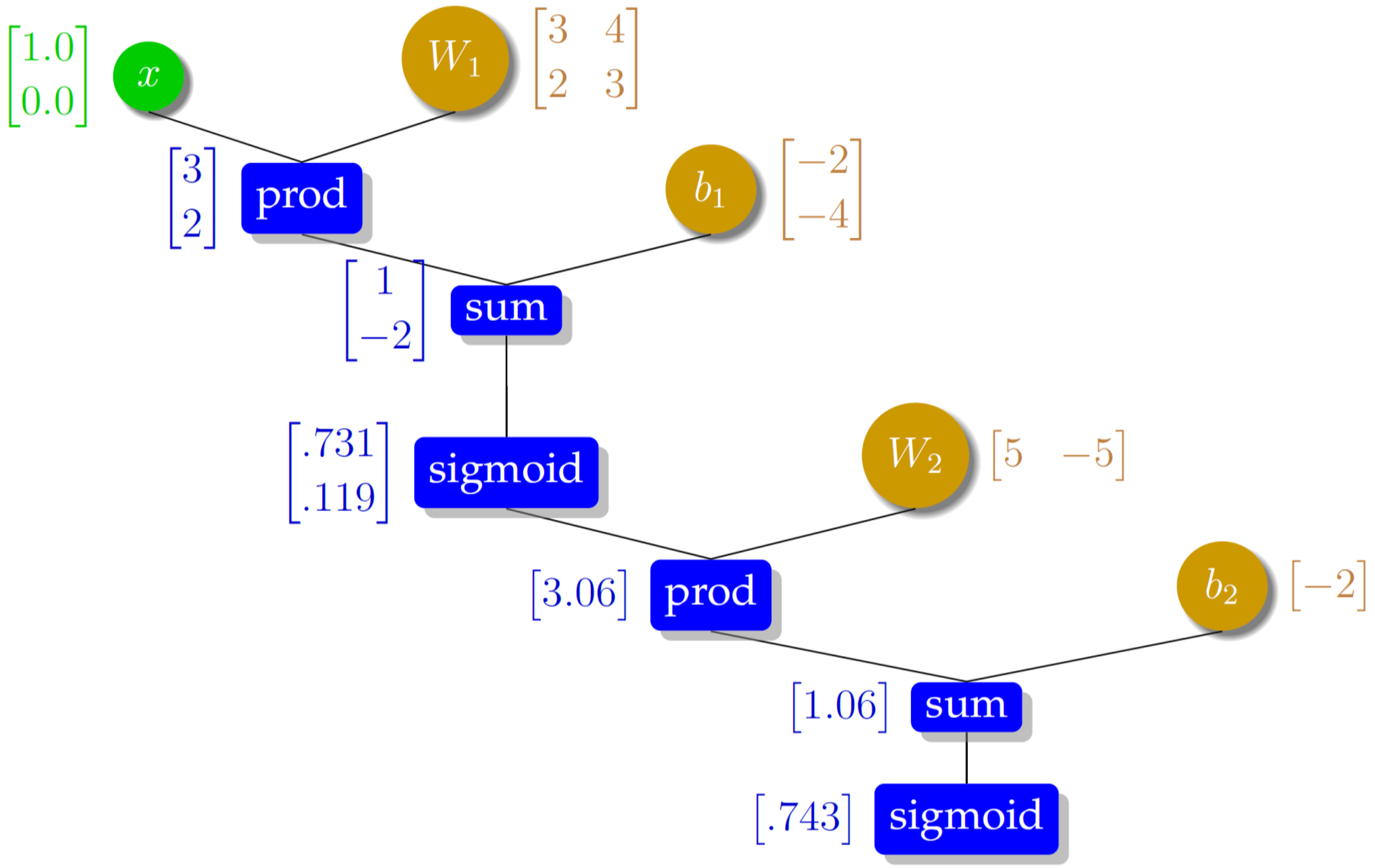
Input x_0	Input x_1	Hidden h_0	Hidden h_1	Output y_0
0	0	0.119	0.018	$0.183 \to 0$
0	1	0.881	0.269	$0.743 \to 1$
1	0	0.731	0.119	$0.743 \to 1$
1	1	0.993	0.731	$0.334 \to 0$

Neural Networks as Computation Graphs





Computation Graphs Make Prediction Easy: Forward Propagation consists in traversing graph in topological order



Neural Networks so far

- Powerful non-linear models for classification
- Predictions are made as a sequence of simple operations
  - matrix-vector operations
  - non-linear activation functions
- Choices in network structure
  - Width and depth
  - Choice of activation function
- Feedforward networks
  - no loop
- Next: how to train