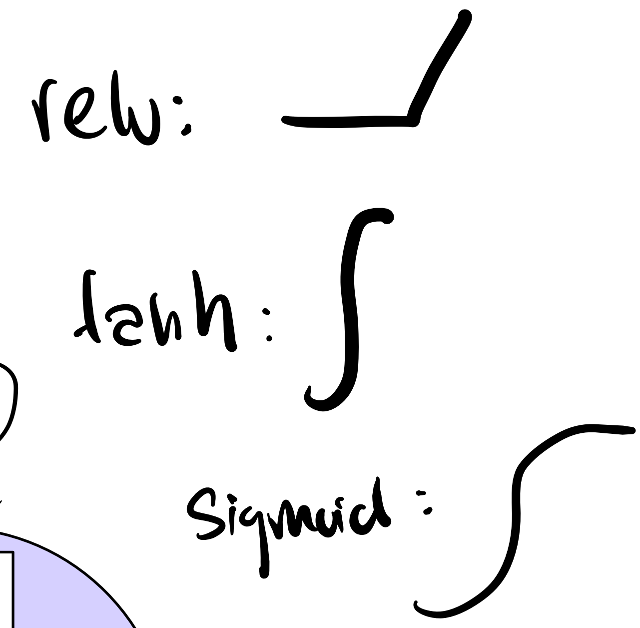
CS 333: NLP

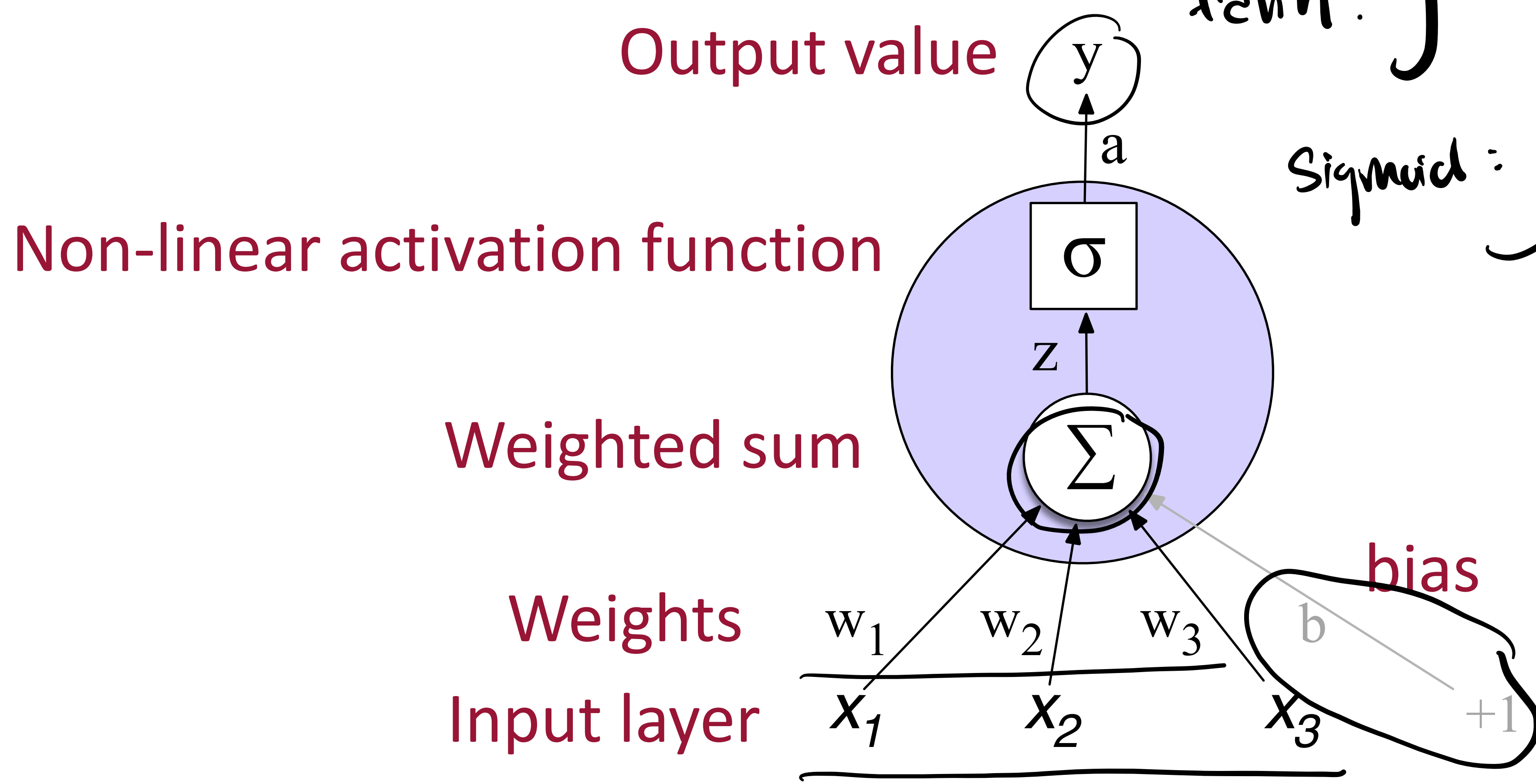
Fall 2023

Prof. Carolyn Anderson
Wellesley College

Neural Network Unit

This is not in your brain





Slide borrowed from Jurafsky & Martin Edition 3

Spot the differences

Neural Network Unit

$$z = b + \sum_i w_i x_i$$

$$y = \sigma(\mathbf{w} \cdot \mathbf{x} + b)$$

Logistic Regression

$$z = \left(\sum_{i=1}^{n} w_i x_i \right) + b$$

$$\underline{P(y=1)} = \sigma(w \cdot x + b)$$

Example: XOR

The XOR problem

Minsky and Papert (1969)

Can neural units compute simple functions of input?

AND			OR			XOR		
x1	x2	y	x1	x2	y	x1	x2	y
0	0	0	0	0	0	0	0	0
0	1	0	0	1	1	0	1	1
1	0	0	1	0	1	1	0	1
1	1	1	1	1	1	1	1	0

Perceptrons

A very simple neural unit

- Binary output (0 or 1)
- No non-linear activation function

$$y = \begin{cases} 0, & \text{if } \mathbf{w} \cdot \mathbf{x} + b \leq 0 \\ 1, & \text{if } \mathbf{w} \cdot \mathbf{x} + b > 0 \end{cases}$$

Solving AND

Slides borrowed from Jurafsky & Martin Edition 3

Deriving AND

$$y = \begin{cases} 0, & \text{if } \mathbf{w} \cdot \mathbf{x} + b \leq 0 \\ 1, & \text{if } \mathbf{w} \cdot \mathbf{x} + b > 0 \end{cases}$$

Goal: return 1 if x1 and x2 are 1

AND

x1	x2	y
0	0	0
0	1	0
1	0	0
1	1	1

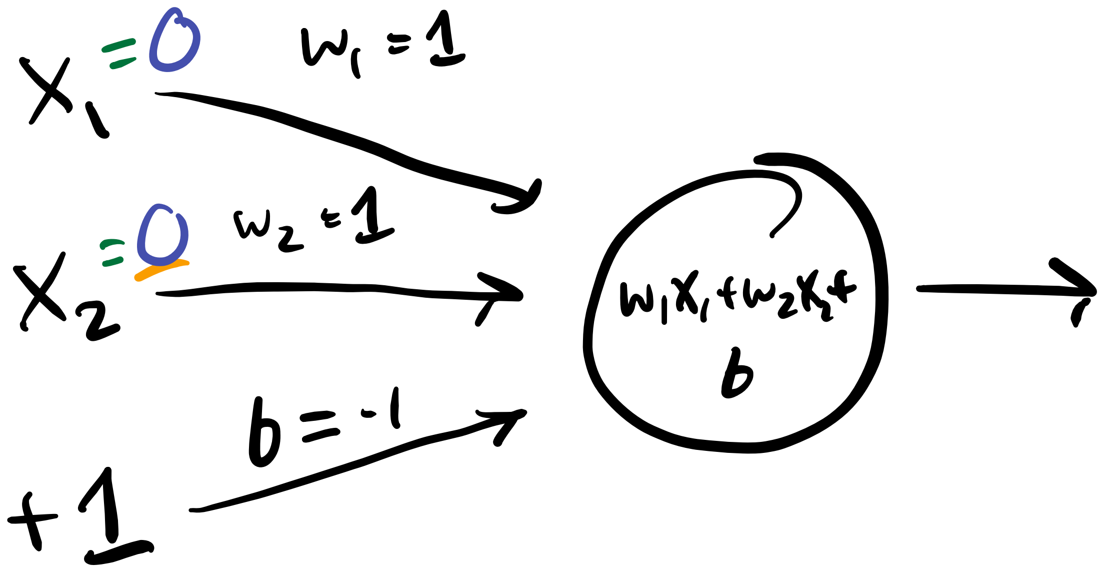
Deriving AND

Goal: return 1 if x1 and x2 are 1

$y = 0 + 0 + -1 = -1$

AND

x_1	x_2	y
1	1	1
1	0	0
0	1	0
0	0	0



Exercise: solving OR

OR

x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	1

Deriving OR

$$y = \begin{cases} 0, & \text{if } \mathbf{w} \cdot \mathbf{x} + b \leq 0 \\ 1, & \text{if } \mathbf{w} \cdot \mathbf{x} + b > 0 \end{cases}$$

Goal: return 1 if either input is 1

OR

x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	1

Deriving OR

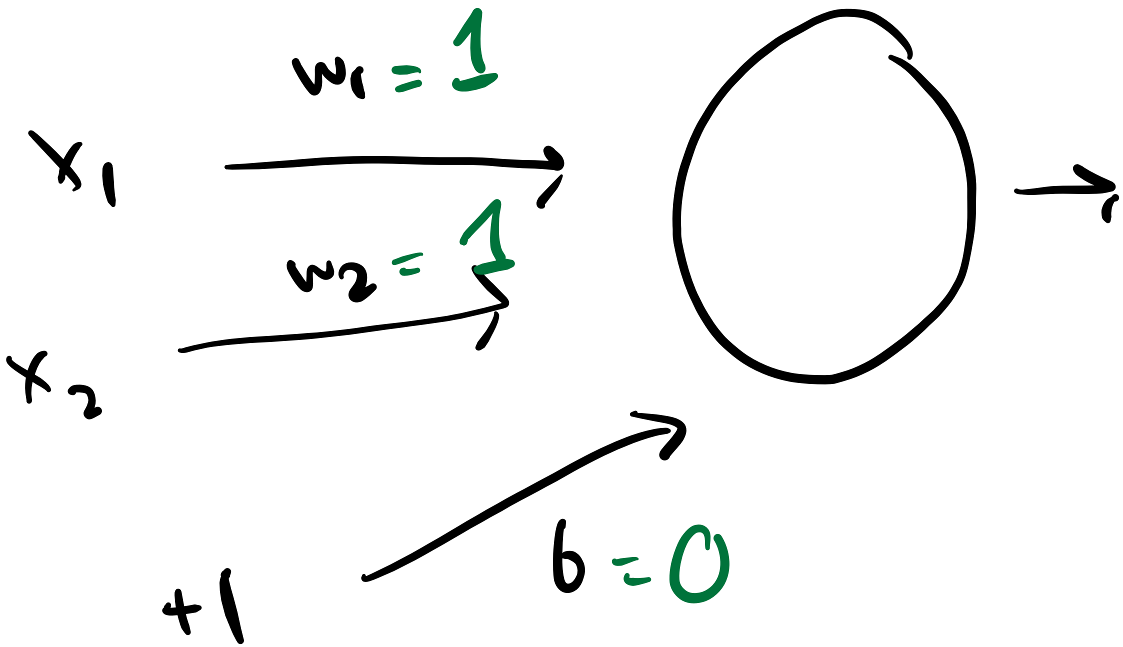
$y = 0 + 0 + 0$

$y = 1 + 0 + 0$

$y = 1 + 1 + 0$

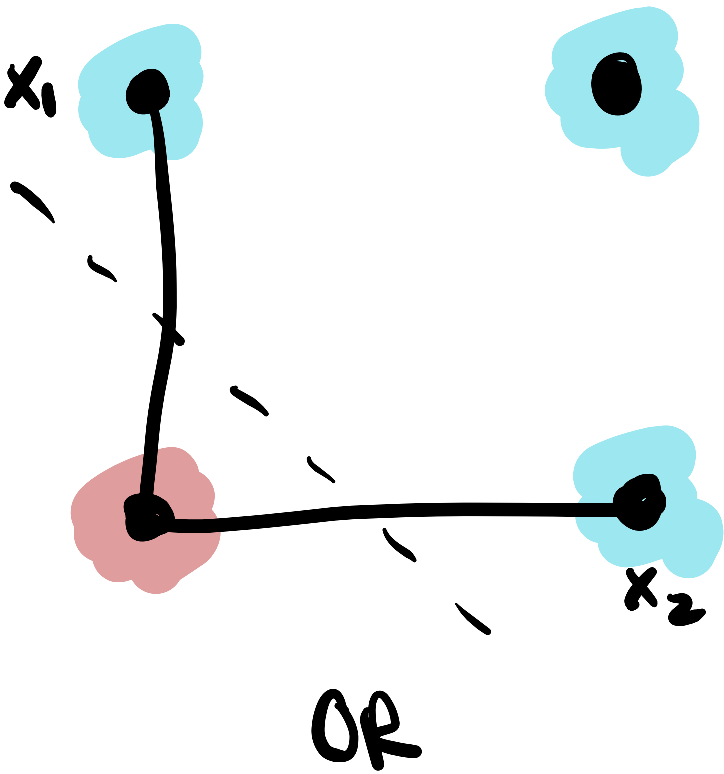
Goal: return 1 if either input is 1

OR



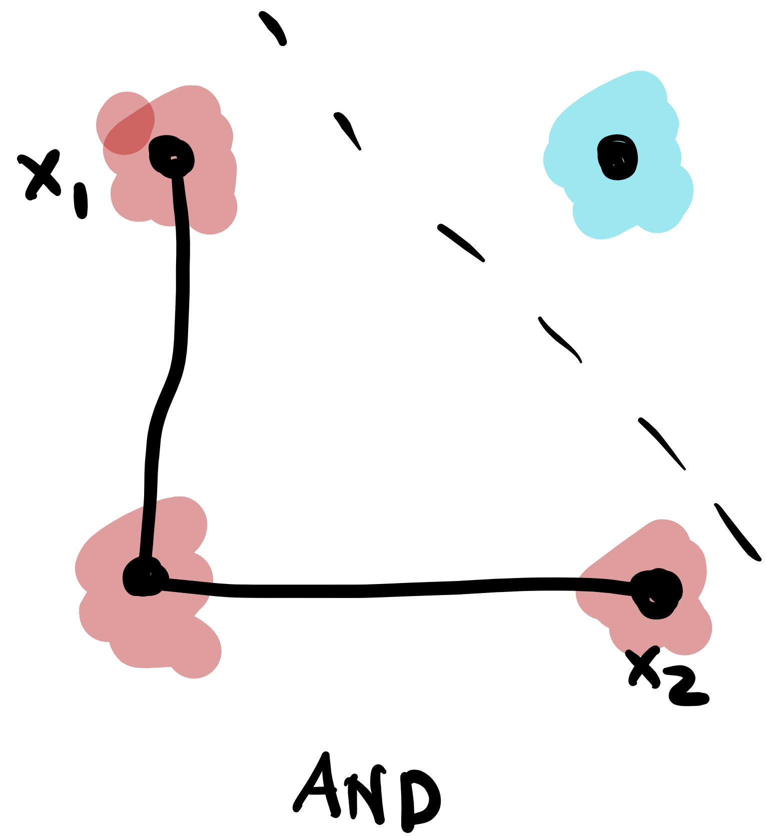
x_1	x_2	y
0	0	0
1	1	1
1	0	1
0	1	1

solving XOR

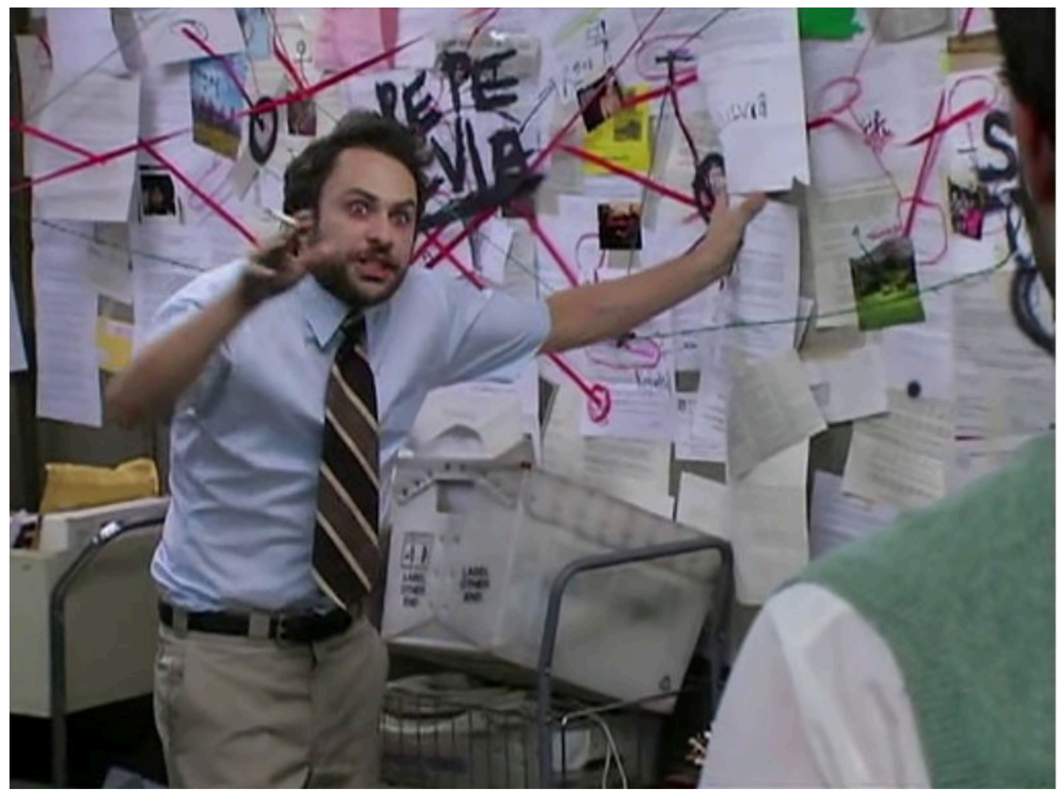


XOR

x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	0



Trick question!
It's not possible to capture XOR with perceptrons



Why? Perceptrons are linear classifiers

Perceptron equation is the equation of a line

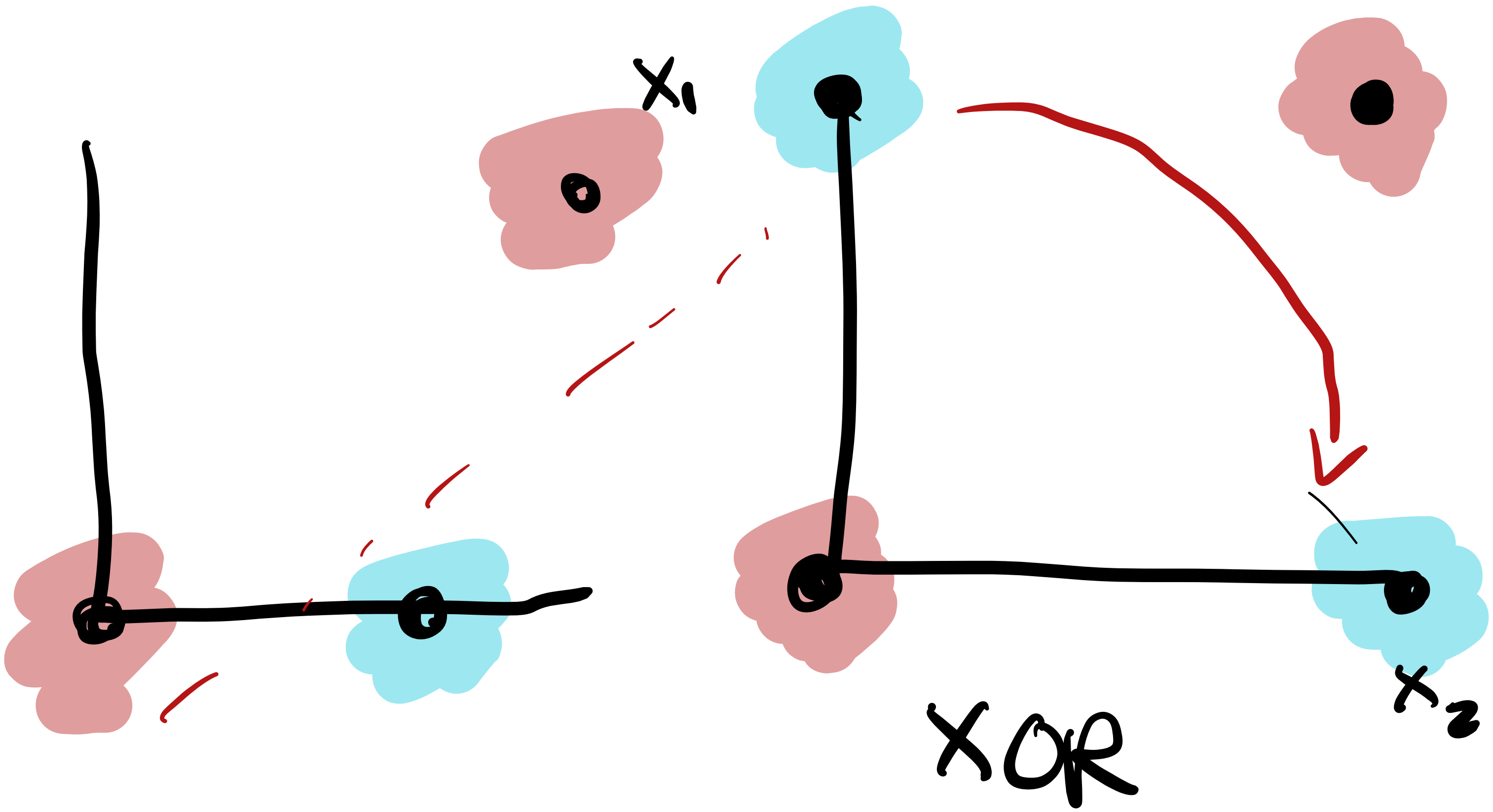
$w_1 x_1 + w_2 x_2 + b = 0$

(in standard linear format: $x_2 = (-w_1/w_2)x_1 + (-b/w_2)$)

This line acts as a **decision boundary**

- 0 if input is on one side of the line
- 1 if on the other side of the line

Decision boundaries



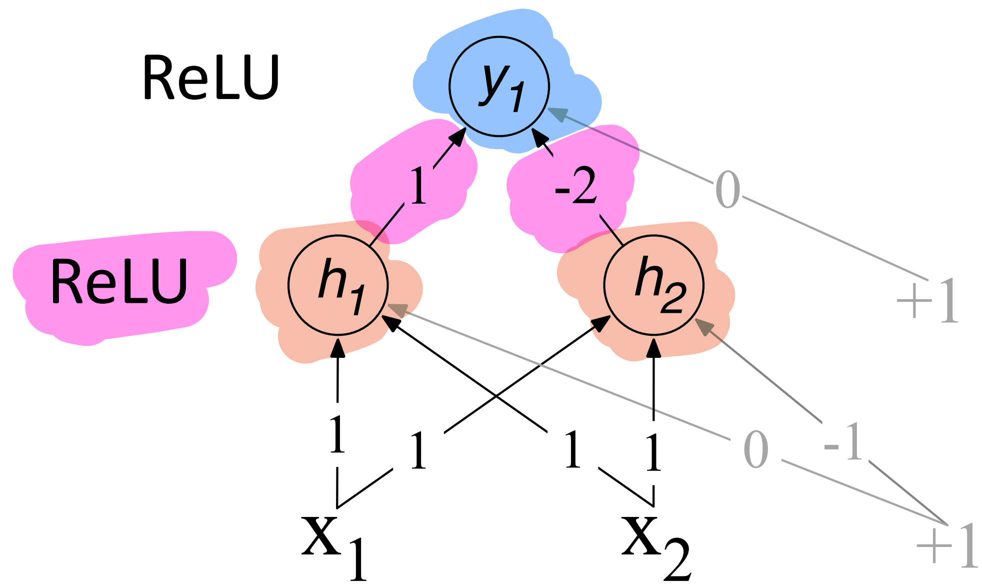
Solution to the XOR problem

XOR **can't** be calculated by a **single** perceptron

XOR **can** be calculated by a layered network of units.

XOR

x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	0



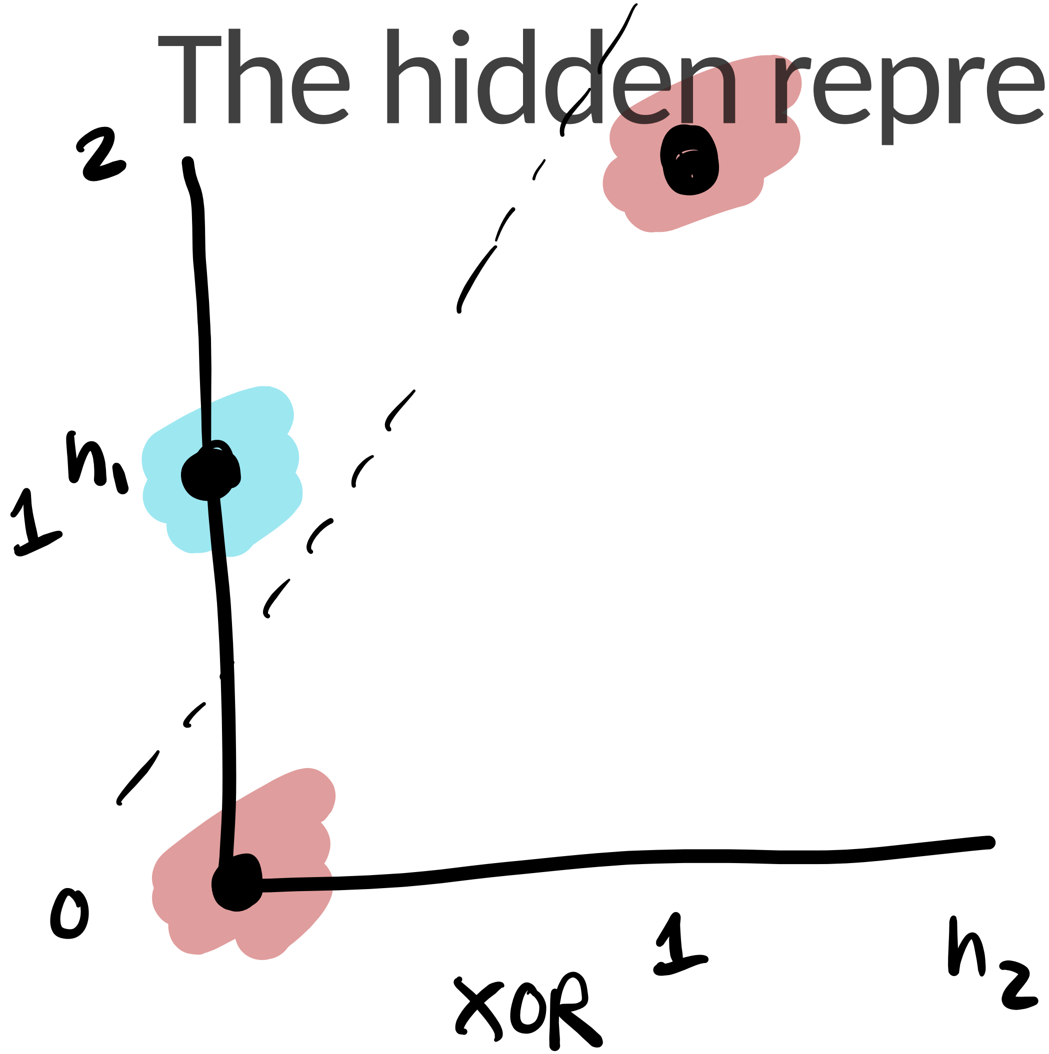
$h_1 = 2$

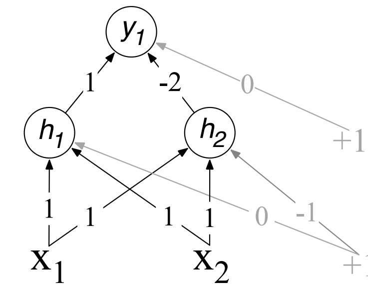
$h_2 = 1$

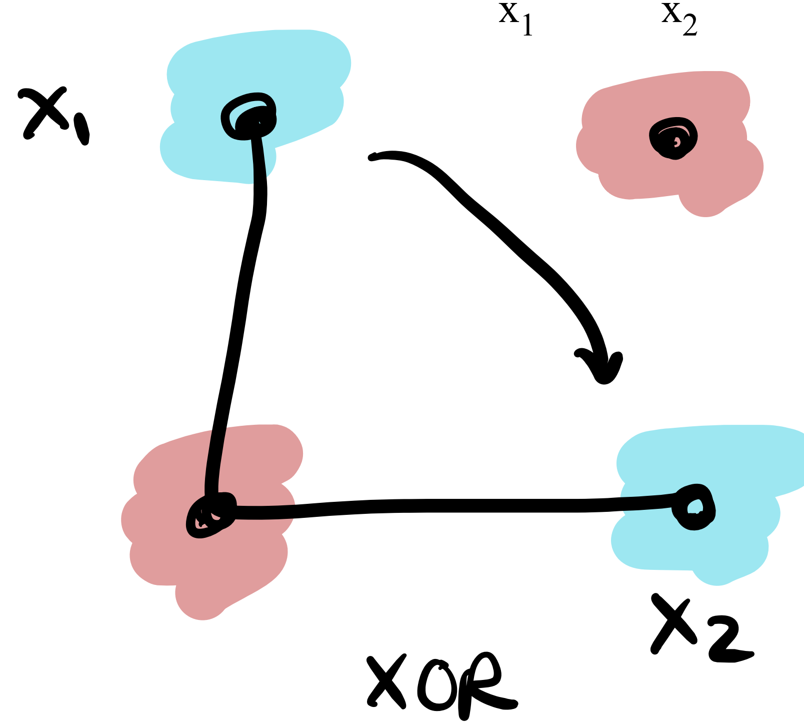
$y_1 = 1 \cdot 2 + -2 \cdot 1 + 0$

$= 0$

The hidden representation h







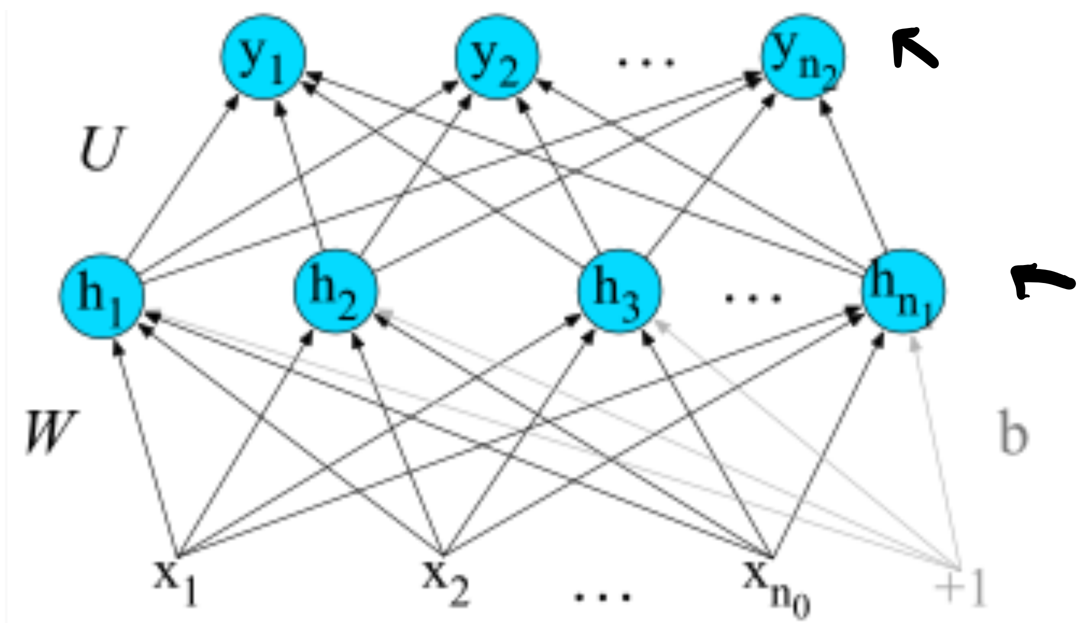
hidden layer: intermediate units
learn transformations of features

Feedforward Networks

Slides borrowed from Jurafsky & Martin Edition 3

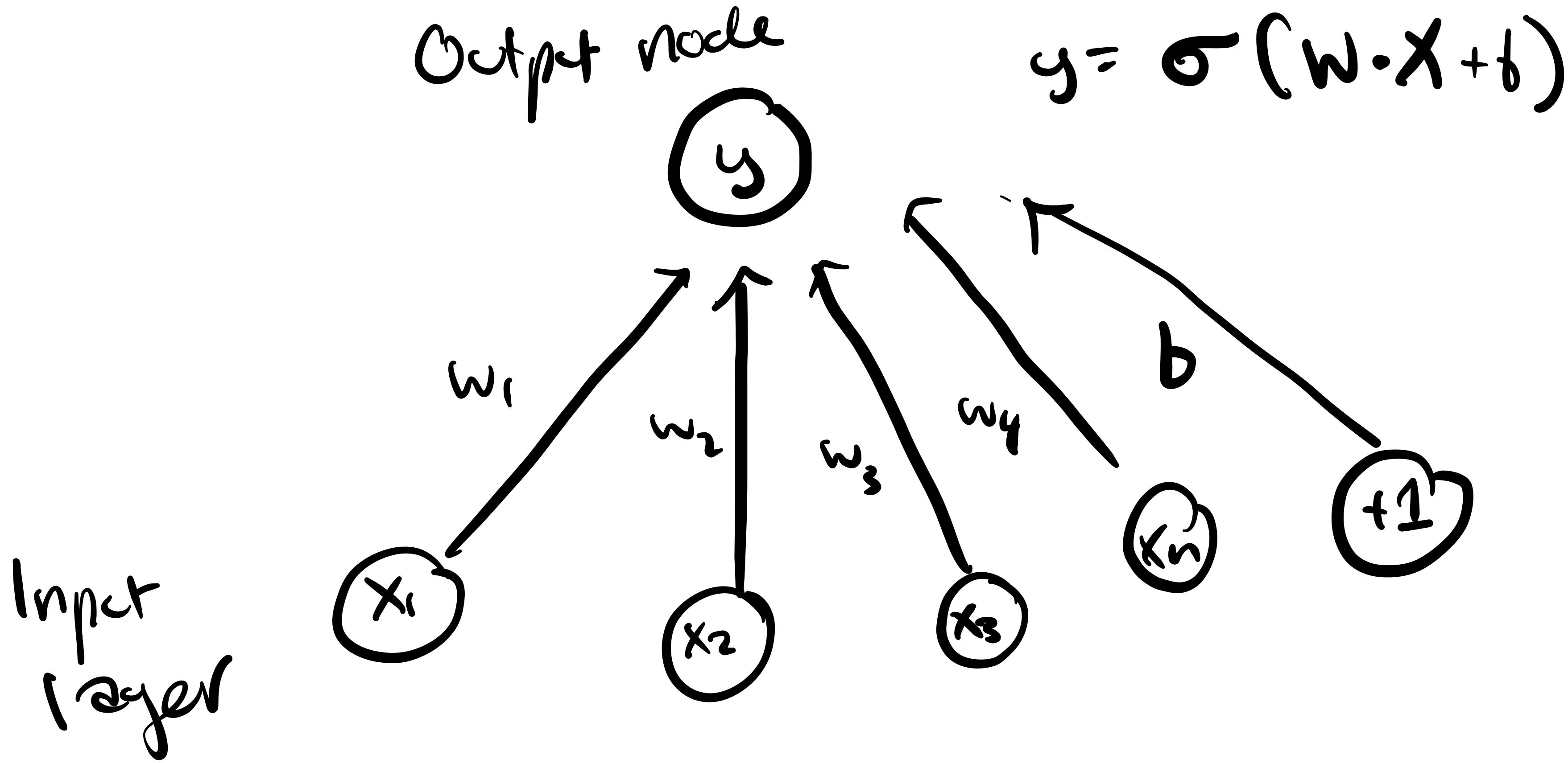
Feedforward Neural Networks

Can also be called **multi-layer perceptrons** (or **MLPs**) for historical reasons



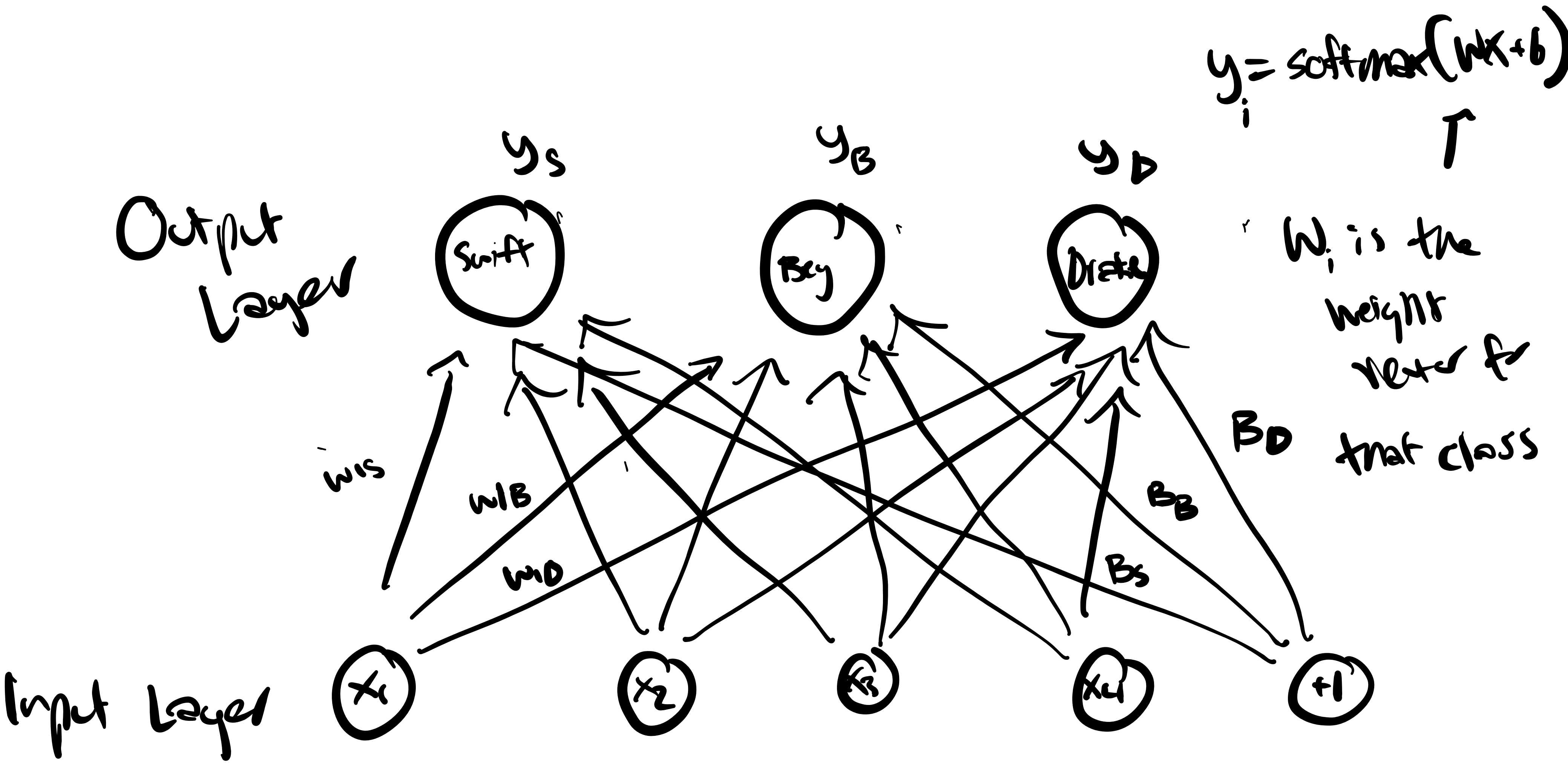
Binary Logistic Regression as a 1-layer Network

(we don't count the input layer in counting layers!)



Multinomial Logistic Regression as a 1-layer Network

Fully connected single layer network



Reminder: softmax: a generalization of sigmoid

For a vector *z* of dimensionality *k*, the softmax is:

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

$$\text{softmax}(\mathbf{z}_i) = \frac{\exp(\mathbf{z}_i)}{\sum_{j=1}^{d}\exp(\mathbf{z}_j)} \quad 1 \le i \le d$$

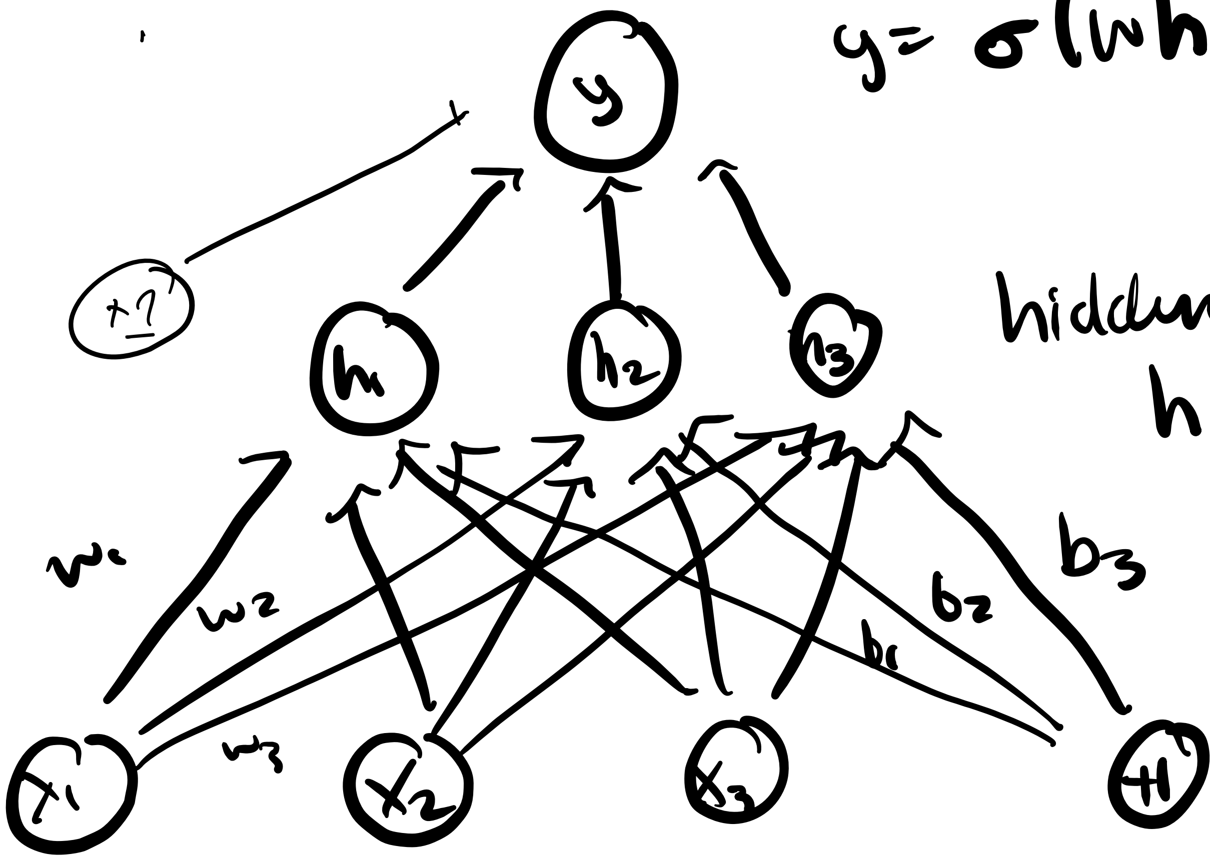
Example:

$$\mathbf{z} = [0.6, 1.1, -1.5, 1.2, 3.2, -1.1],$$

$$\text{softmax}(\mathbf{z}) = [0.055, 0.090, 0.0067, 0.10, 0.74, 0.010]$$

Binary Classification

Two-Layer Network with scalar output



$y = \sigma(wh)$

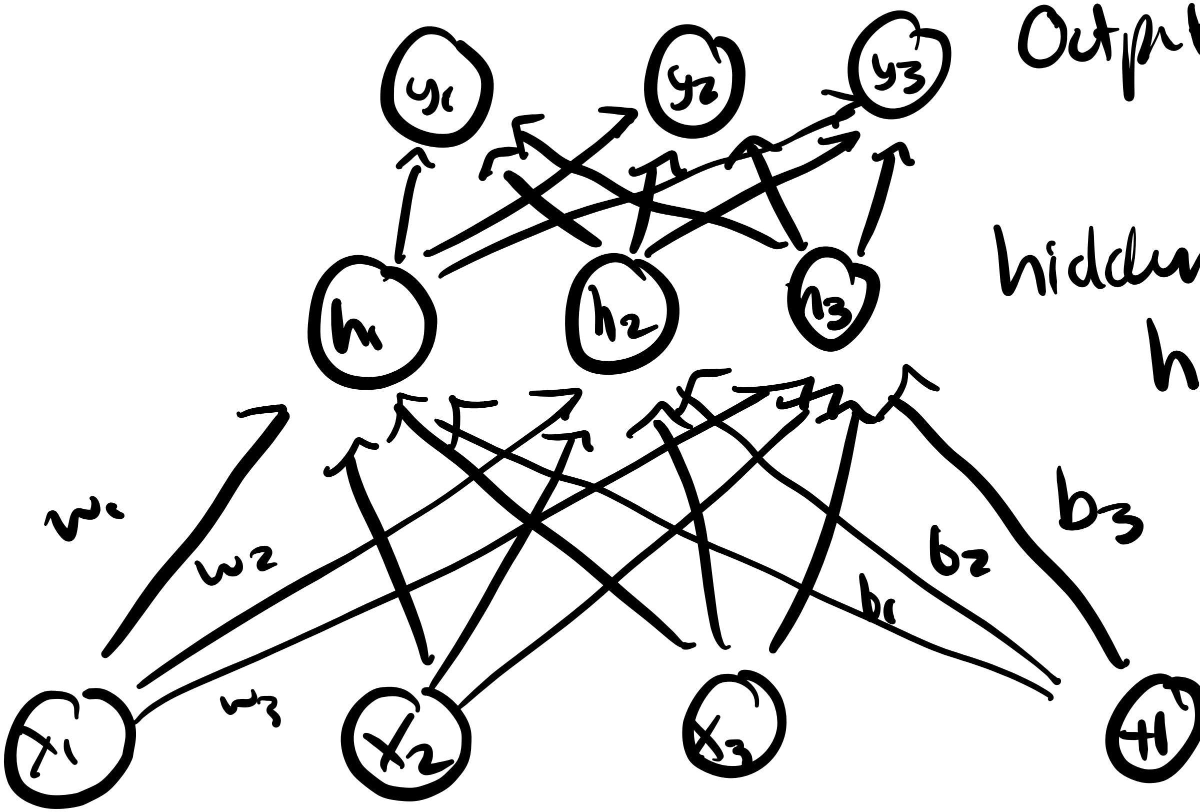
hidden layer

$h = \sigma(WX + b)$

could be tanh or relu

Two-Layer Network with softmax output

Multinomial Classification



Output layer

hidden layer

$h = \sigma(Wx + b)$

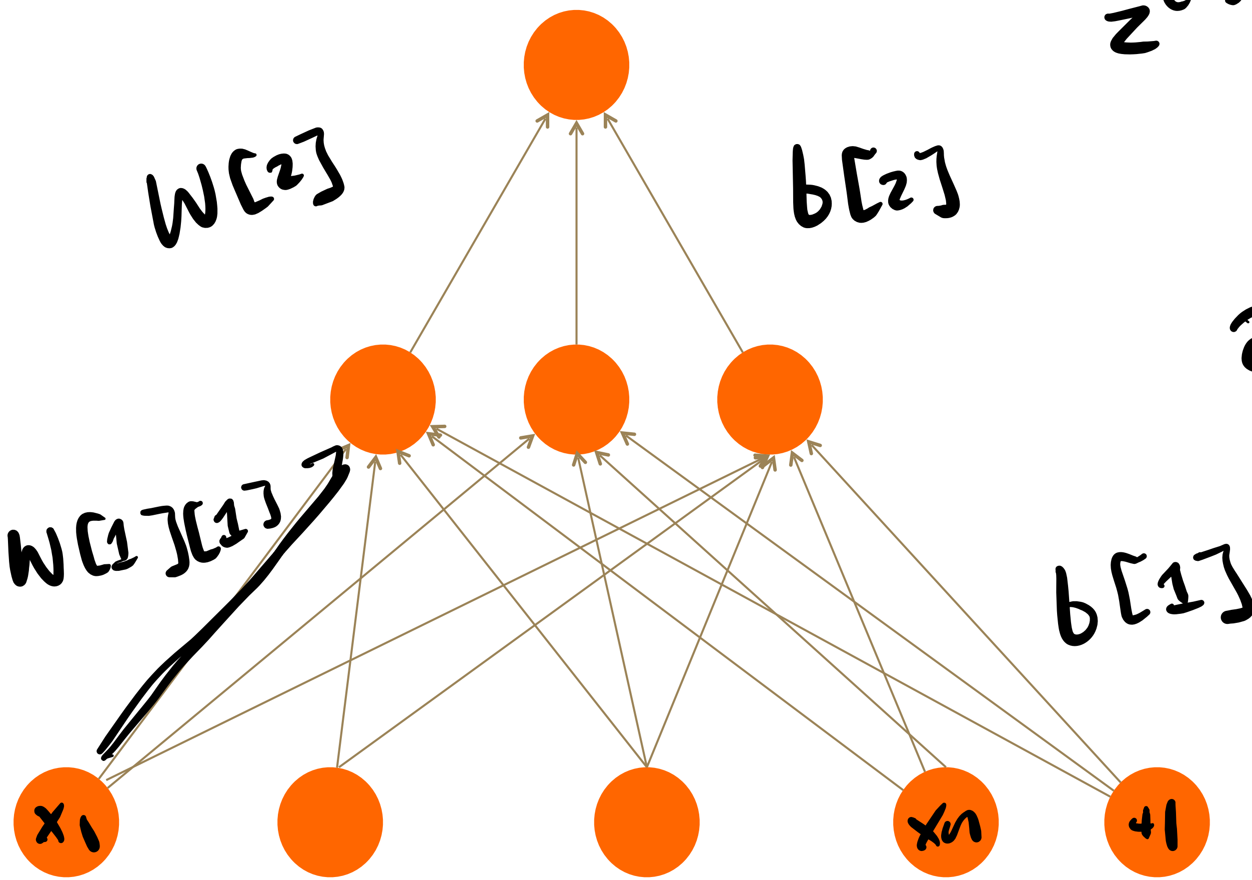
could be tanh or relu

Multi-layer Notation

$y = a^{[2]}$

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

$$z^{[2]} = W^{[2]} a^{[1]} + b^{[2]}$$



$$a^{[1]} = g^{[1]}(z^{[1]})$$

$$z^{[1]} = W^{[1]} x^{[1]} + b^{[1]}$$

Replacing the bias unit

Let's switch to a notation without the bias unit

Just a notational change

1. Add a dummy node $a_0=1$ to each layer
2. Its weight w_0 will be the bias
3. So input layer $a^{[0]}_0=1$,
   - And $a^{[1]}_0=1$, $a^{[2]}_0=1$,...

Replacing the bias unit

Instead of:

$x = x_1, x_2, \ldots, x_{n0}$

$$\mathbf{h} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$$

$$\mathbf{h}_j = \sigma\left(\sum_{i=1}^{n_0} \mathbf{W}_{ji}\mathbf{x}_i + \mathbf{b}_j\right)$$

We'll do this:

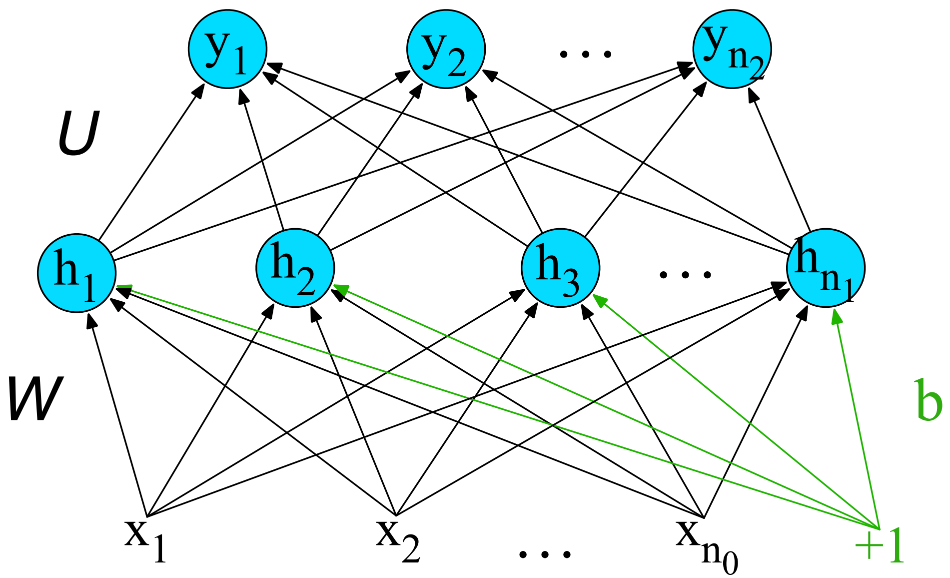
$x = x_0, \ldots x_{n0}$

$$h = \sigma(Wx)$$

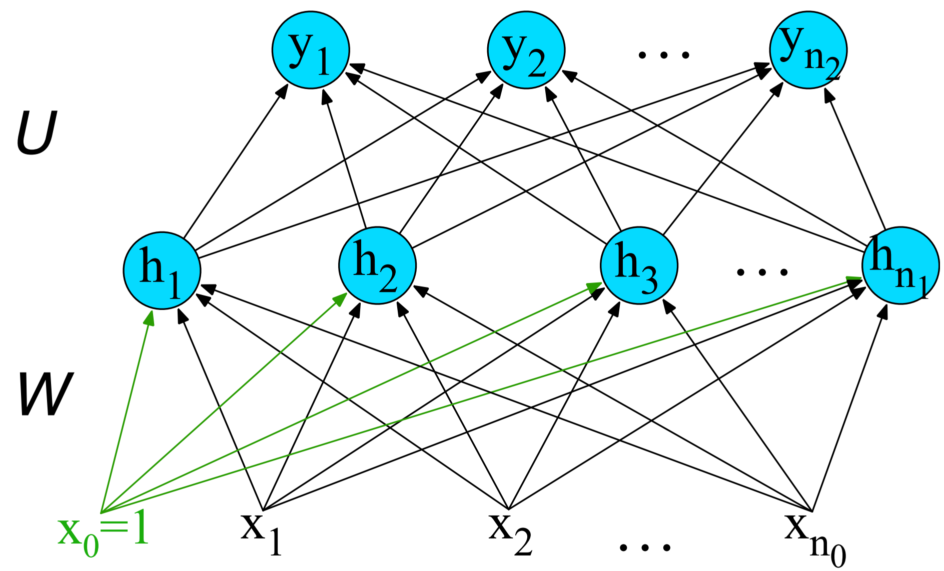
$$\sigma\left(\sum_{i=0}^{n_0} W_{ji} x_i\right)$$

Replacing the bias unit

Instead of:



We'll do this:



Using feedforward networks

Slides borrowed from Jurafsky & Martin Edition 3

Use cases for feedforward networks

Let's reconsider text classification

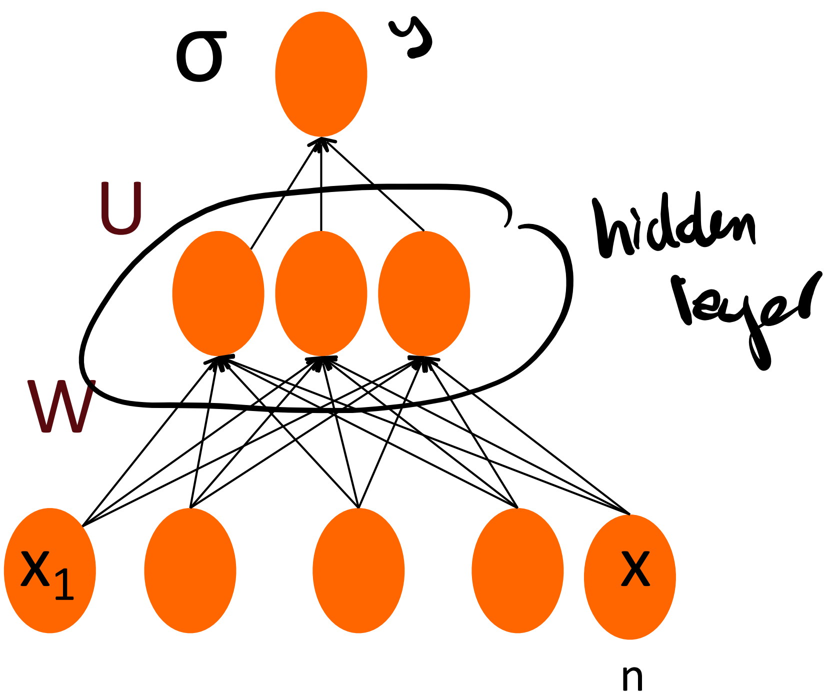
(State-of-the-art systems use more powerful architectures)

Classification: Sentiment Analysis

We could do exactly what we did with logistic regression

Input layer are binary features as before

Output layer is 0 or 1

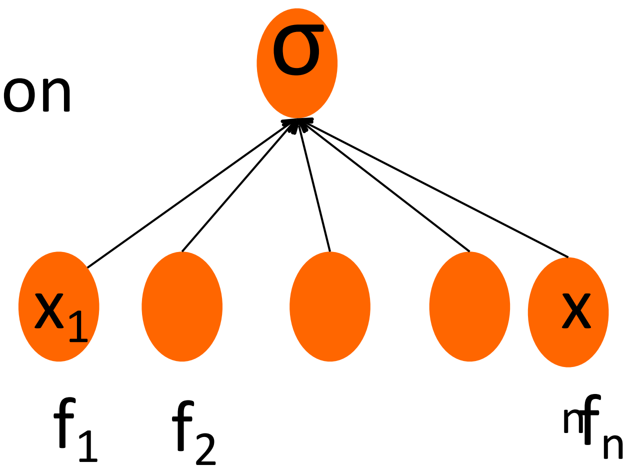


Sentiment Features

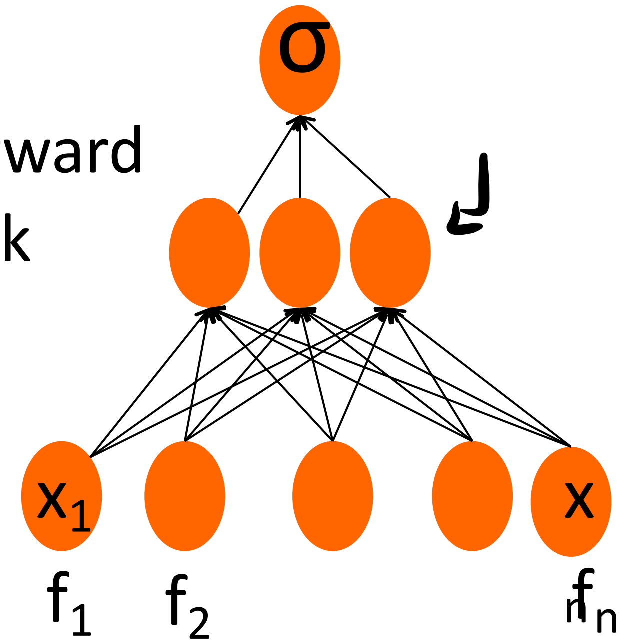
Var	Definition
x_1	count(positive lexicon words $\in$ doc)
x_2	count(negative lexicon words $\in$ doc)
x_3	$\begin{cases} 1 & \text{if ``no''} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$
x_4	count(1st and 2nd pronouns $\in$ doc)
x_5	$\begin{cases} 1 & \text{if ``!''} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$
x_6	log(word count of doc)

Feedforward nets for simple classification

Logistic Regression



2-layer feedforward network



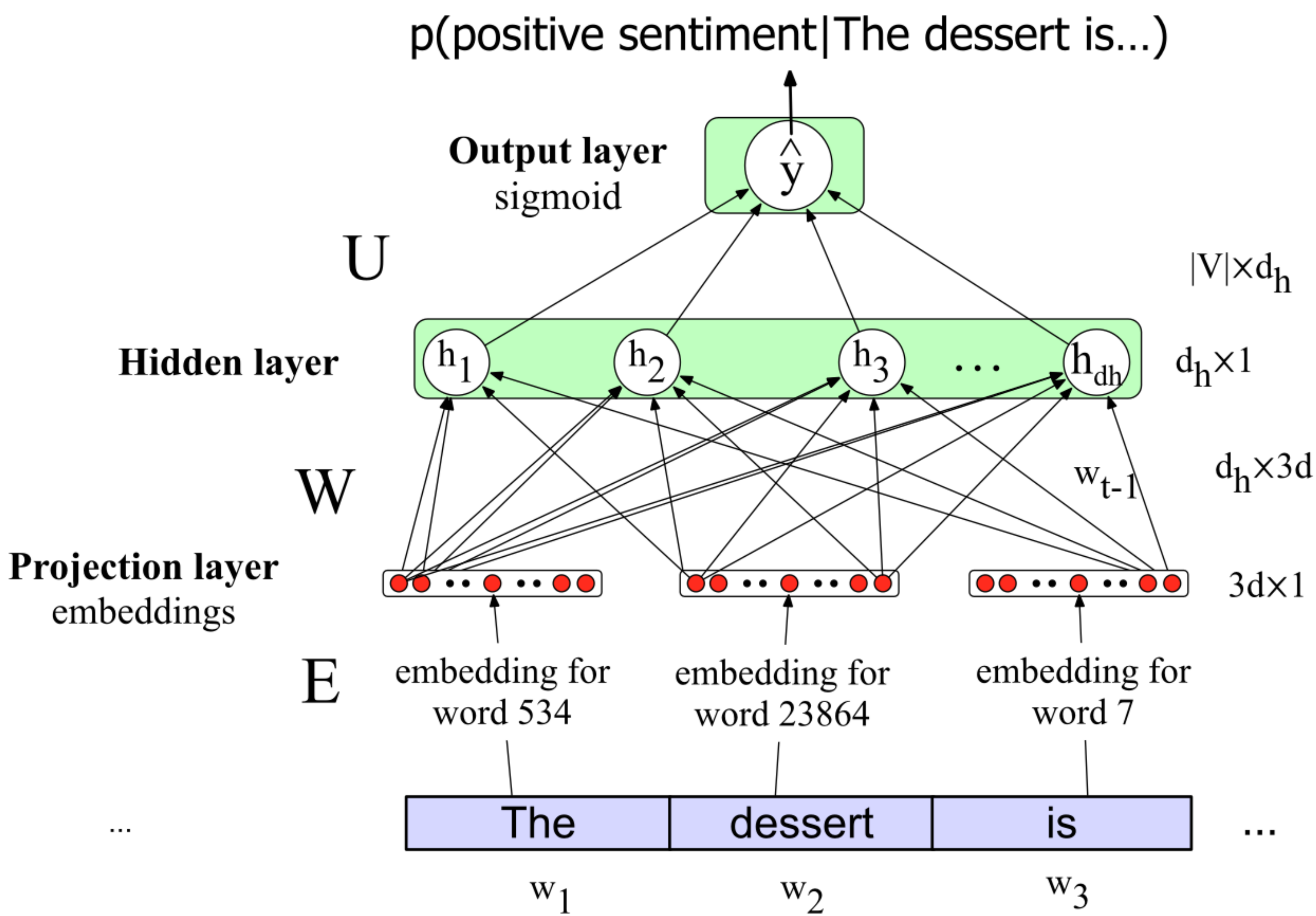
Just add a hidden layer to logistic regression

- allows the network to use non-linear interactions between features
- which may (or may not) improve performance.

Even better: representation learning

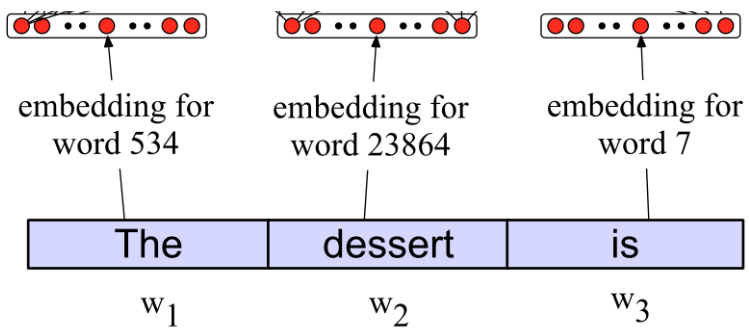
The real power of deep learning comes from the ability to **learn features** from the data, instead of using hand-built human-engineered features for classification.

Neural Net Classification with embeddings as input features!



Issue: texts come in different sizes

This assumes a fixed size length (3)!



Some simple solutions (more sophisticated solutions later)

1. Make the input the length of the longest review
   - If shorter then pad with zero embeddings
   - Truncate if you get longer reviews at test time
2. Create a single "sentence embedding" (the same dimensionality as a word) to represent all the words
   - Take the mean of all the word embeddings
   - Take the element-wise max of all the word embeddings
     - For each dimension, pick the max value from all words

Reminder: Multiclass Outputs

What if you have more than two output classes?

- Add more output units (one for each class)
- And use a “softmax layer”

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