CS 2731
Introduction to Natural Language Processing

Session 11: Neural networks part 1

Michael Miller Yoder

October 1, 2025

University of Pittsburgh | School of Computing and Information

Quiz

- Go to **Quizzes > Quiz 10-01** on Canvas

- You have until **2:40pm** to complete it

- Allowed resources

  - Textbook

  - Your notes (on a computer or physical)

  - Course slides and website

- Resources not allowed

  - Generative AI

  - Internet searches

Course logistics: homework

- [Homework 2](#) is **due next Thu Oct 9**
  - The Kaggle competition has been posted

Course logistics: project

- Next project deliverable: project proposal due **Oct 16**

  - Will include plans for **task, data, methods, evaluation**

  - Include example input and output

  - Literature review of at least 3 related papers

  - Feel free to email or book office hours with Michael to discuss

- We have $150 total as a class to use on OpenAI LLM credits

- Access to open-source LLM set up on School of Computing and Information servers for API access is coming soon

  - Gemma, LLaMa, Deepseek

Midterm course evaluation (OMETs)

- https://go.blueja.io/Iq36newH2UeDZRnTEA4pDg

- All types of feedback are welcome (critical and positive)

- **Completely anonymous, will not affect grades**

- Let me know what's working and what to improve on while the course is still running!

- Please be as specific as possible

- Available until **next Mon Oct 6**

Structure of this course

MODULE 1	Introduction and text processing	text normalization, machine learning, NLP tasks	

	Approaches	How text is represented	NLP tasks
MODULE 2	statistical machine learning	n-grams	language modeling text classification
MODULE 3	neural networks	static word vectors	text classification
MODULE 4	transformers and LLMs	contextual word vectors	language modeling text classification

MODULE 5	Sequence labeling and parsing	

MODULE 6	NLP applications and ethics	

Lecture overview: neural networks part 1

- Neural network fundamentals

- Non-linear activation functions

- Feedforward neural networks as classifiers

- Feedforward neural networks with word embedding input

- Coding activity

Neural network fundamentals

This is in your brain

Cell body

Axon

Telodendria
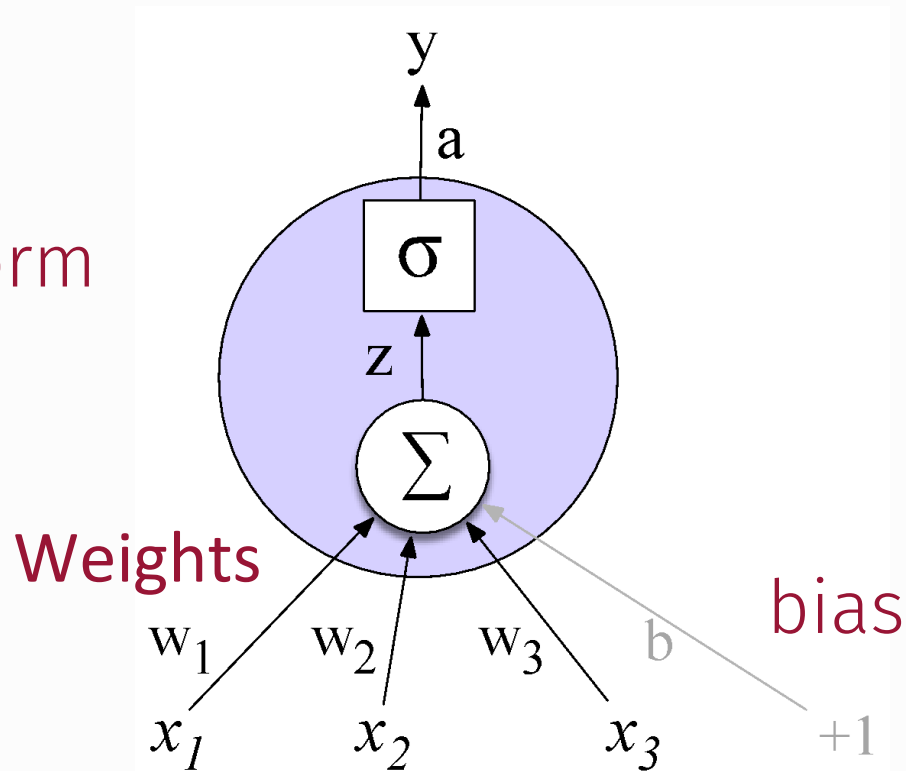
Nucleus

Axon hillock

Synaptic terminals

Golgi apparatus

Endoplasmic
reticulum

Mitochondrion

Dendrite

Dendritic branches

Slide adapted from Jurafsky & Martin
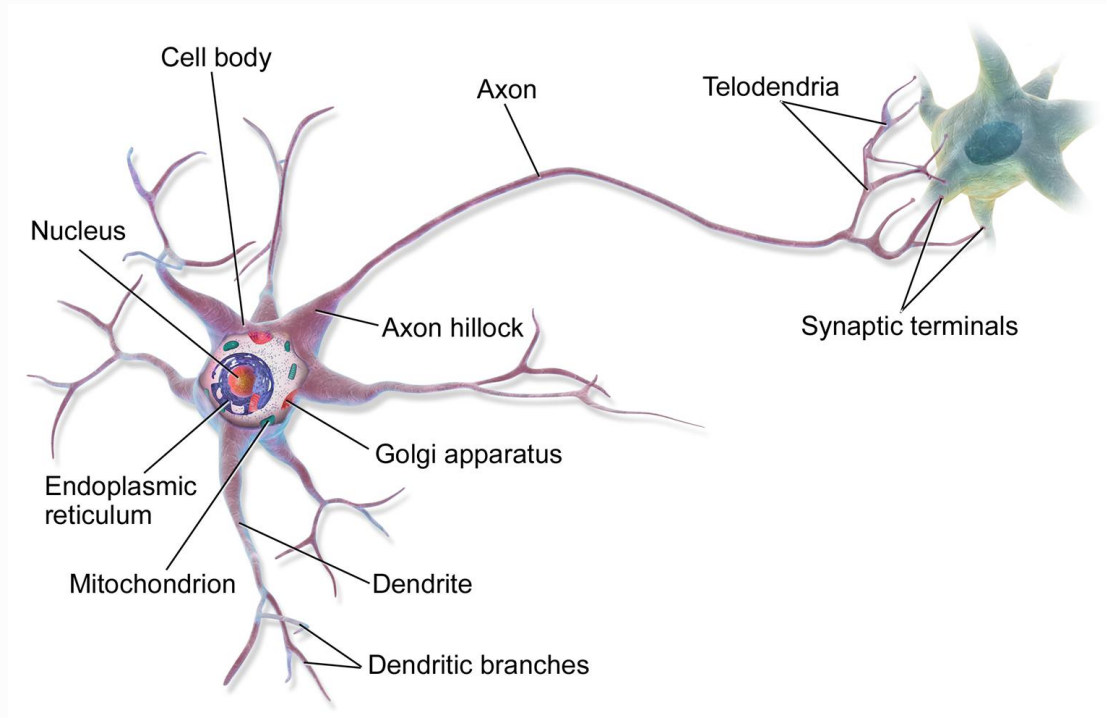
Neural network unit: This is not in your brain

Output value

Non-linear transform

Weighted sum

Weights

Input layer

$$y$$
$$a$$
$$\sigma$$
$$z$$
$$\Sigma$$
$$w_1 \quad w_2 \quad w_3 \quad b$$
$$x_1 \quad x_2 \quad x_3 \quad +1$$

The Variables in Our Very Important Formula

$\mathbf{x}$ A vector of features of n dimensions (like number of positive sentiment words, length of document, etc.)

$\mathbf{w}$ A vector of weights of n dimensions specifying how discriminative each feature is

b A scalar bias term that shifts z

z The raw score

y A random variable (e.g., $y = 1$ means positive sentiment and $y = 0$ means negative sentiment

Slide adapted from David Mortensen

The fundamental equation that describes a unit of a neural network should look very familiar:

$$z = b + \sum_i w_i x_i \tag{1}$$

Which we will represent as

$$z = \mathbf{w} \cdot \mathbf{x} + b \tag{2}$$

But we do not use z directly. Instead, we pass it through a non-linear function, like the sigmoid function:

$$y = \sigma(z) = \frac{1}{1 + e^{-z}} \tag{3}$$

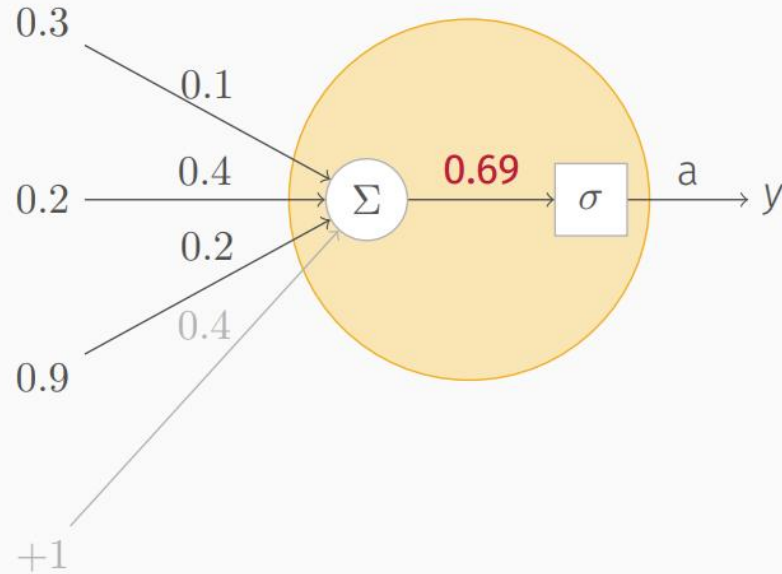(which has some nice properties even though, in practice, we will prefer other functions like tanh and ReLU).

Slide adapted from David Mortensen
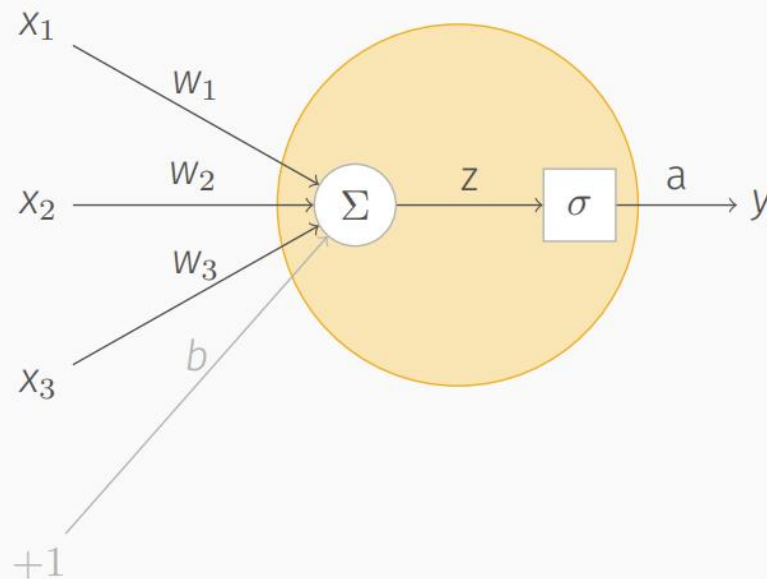
Take, for example, a scenario in which our unit has the weights [0.1, 0.4, 0.2] and the bias term 0.4 and the input vector x has the values [0.3, 0.2, 0.9].
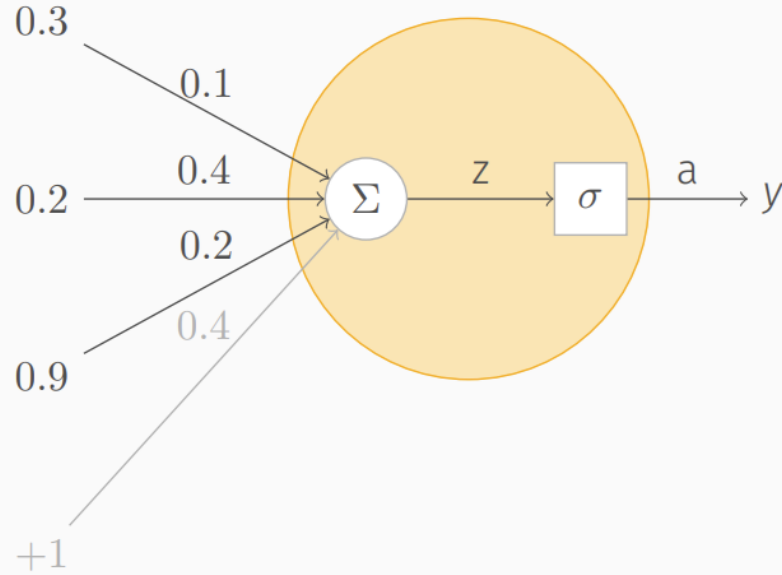
Slide adapted from David Mortensen

Slide adapted from David Mortensen

$$z = x_1w_1 + x_2w_2 + x_3w_3 + b = 0.1(0.3) + 0.4(0.2) + 0.2(0.9) + 0.4 = 0.69 \qquad (4)$$
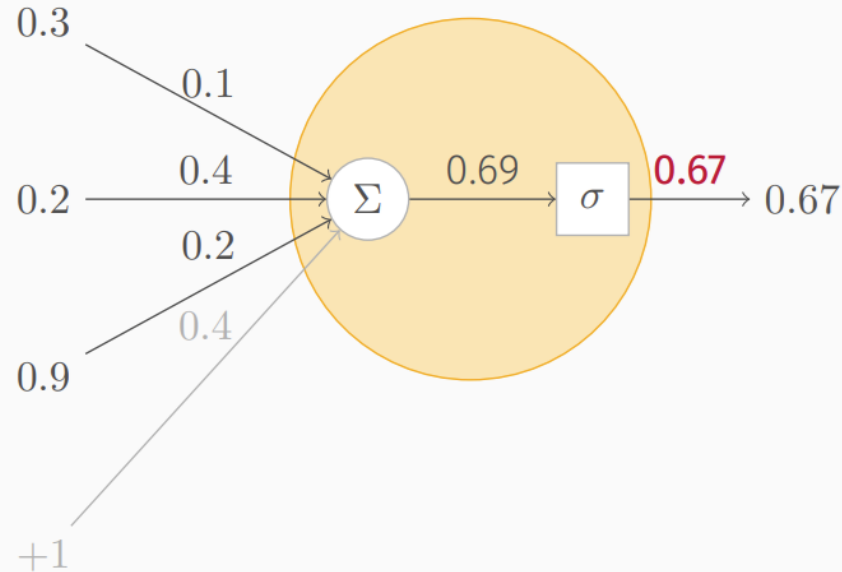
$$y = \sigma(0.69) = \frac{1}{1 + e^{-0.69}} = 0.67 \tag{5}$$

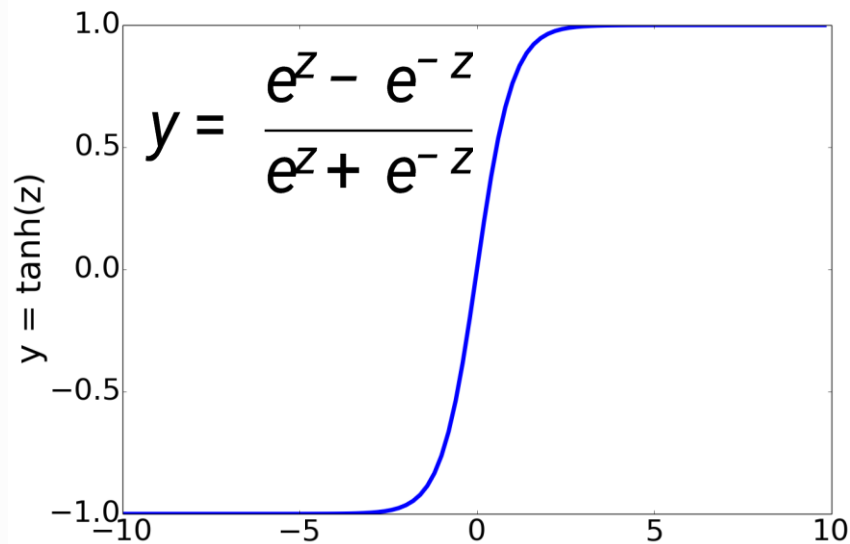# Non-linear activation functions

Non-Linear Activation Functions

We're already seen the sigmoid for logistic regression:

Sigmoid
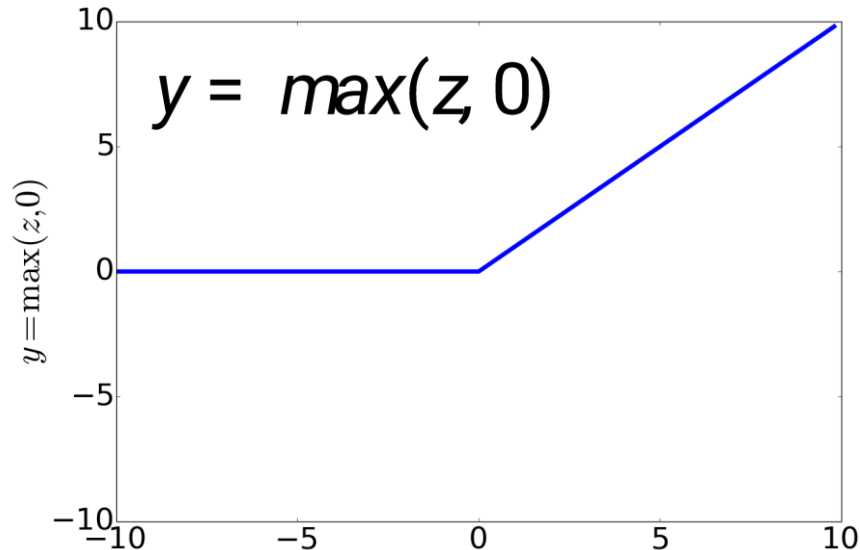
$$y = \sigma(z) = \frac{1}{1 + e^{-z}}$$

$$y = 1/(1 + e^{-z})$$

18
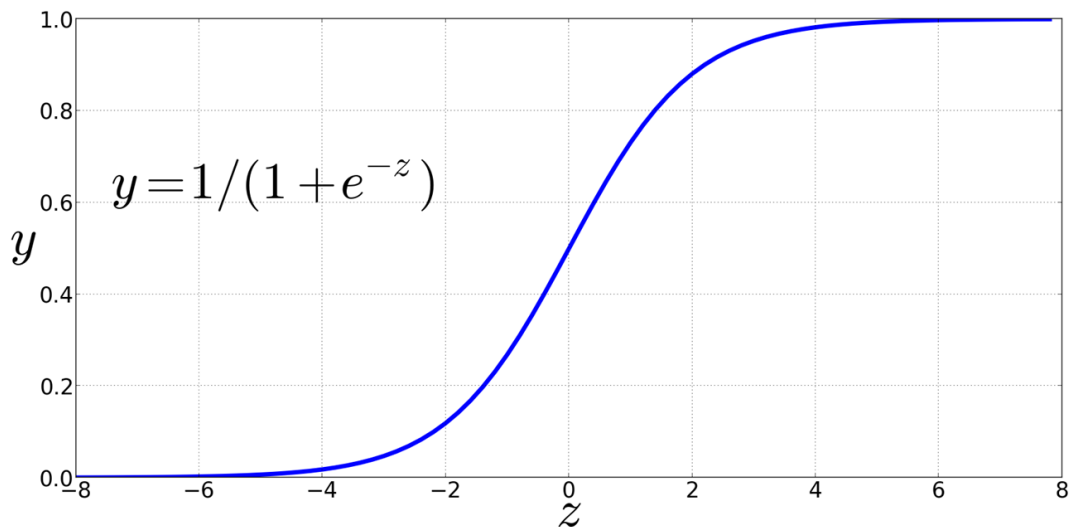
Nonlinear activation functions besides sigmoid
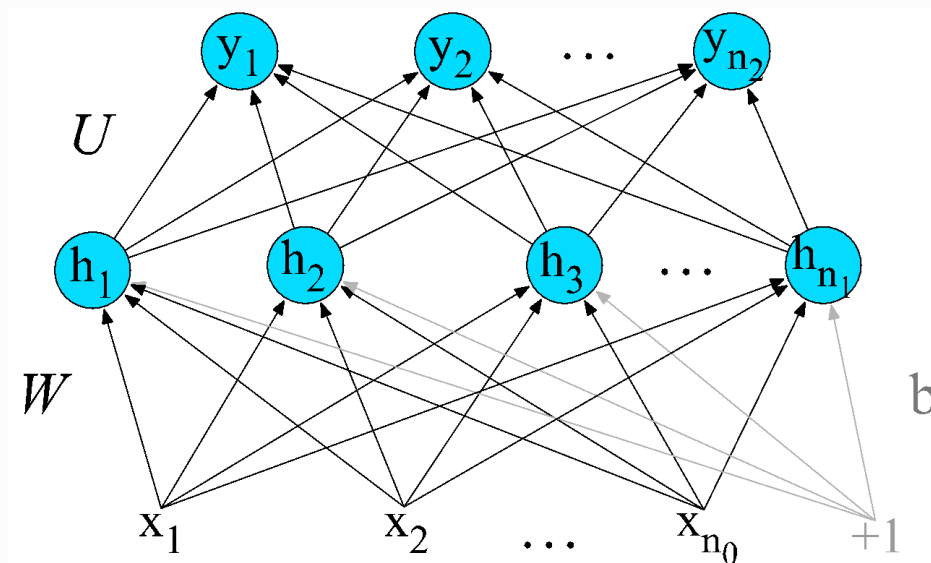
Most common:

$$y = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

$$y = max(z, 0)$$

tanh

ReLU
Rectified Linear

Feedforward neural networks

Adding multiple units to a neural network increases its power to learn patterns in data. **Feedforward Neural Nets (FFNNs or MLPs)**
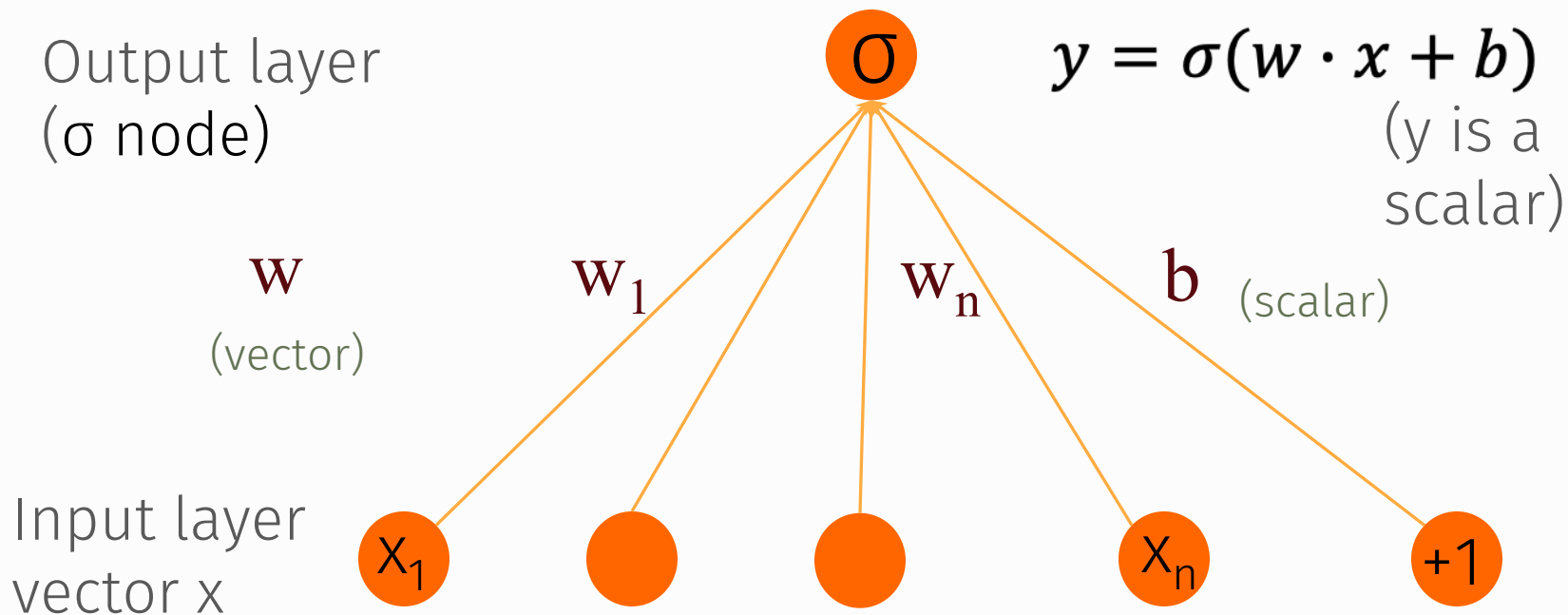
Feedforward Neural Networks

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

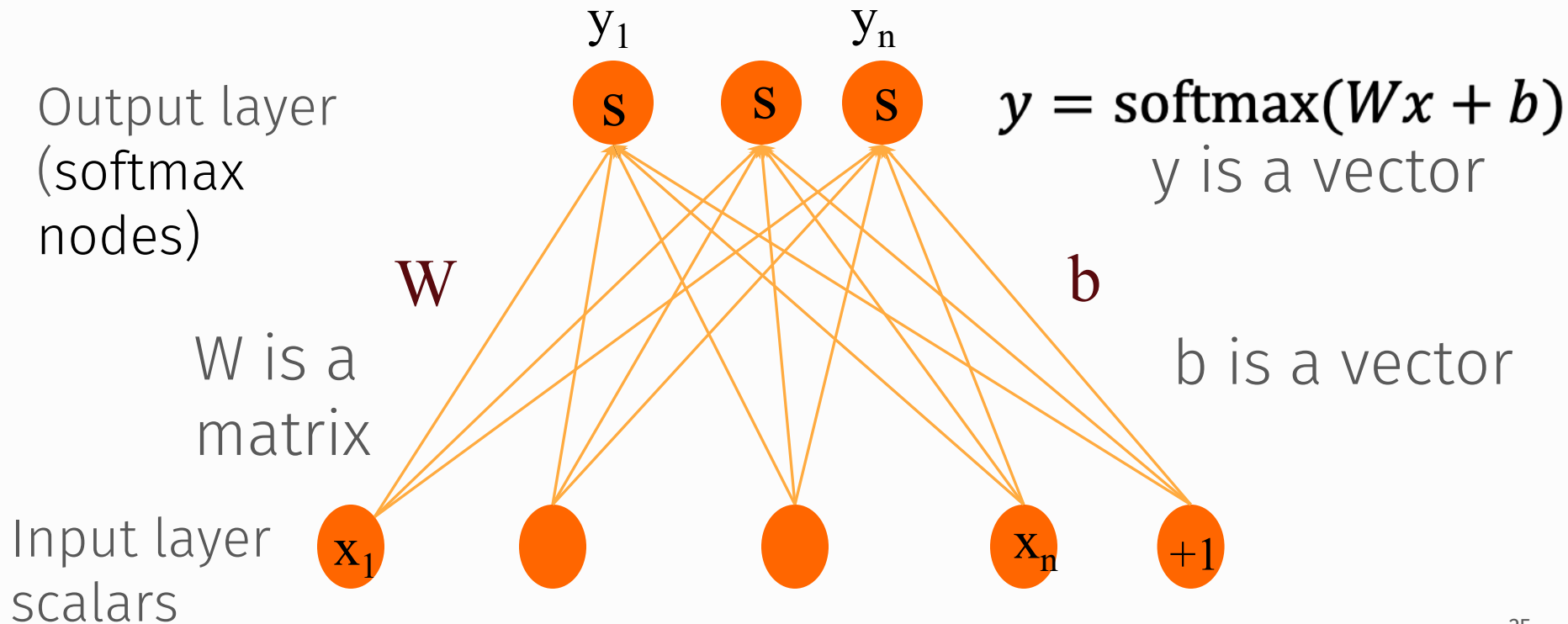The simplest FFNN is just binary logistic regression
(INPUT LAYER = feature vector)

Binary Logistic Regression as a 1-layer Network

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

Output layer
(σ node)

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

(y is a scalar)

w
(vector)

w_1

w_n

b (scalar)

Input layer
vector x

x_1 x_n +1

σ

Fully connected single layer network

Output layer (softmax nodes)

y_1

y_n

$y = \text{softmax}(Wx + b)$

y is a vector

W

b

W is a matrix

b is a vector

Input layer scalars

x_1

x_n

+1

The real power comes when multiple layers are added

Slide adapted from David Mortensen

Two-Layer Network with scalar output

Output layer
(σ node)

hidden units
(σ node)

Input layer
(vector)

U

W

b

$y = \sigma(z)$ y is a scalar
$z = Uh$

$h = \sigma(Wx + b)$

Could be
ReLU
or tanh

x_1 x_n +1

Slide adapted from Jurafsky & Martin

Output layer
(σ node)

U

$y = \sigma(z)$ y is a scalar

$z = Uh$

hidden units
(σ node)

W_{ji}

$h = \sigma(Wx + b)$

W

b vector

Input layer
(vector)

x_1 i x_n +1

Two-Layer Network with scalar output

Output layer
(σ node)

$y = \sigma(z)$ y is a scalar

$z = Uh$

U

hidden units
(σ node)

$h = \sigma(Wx + b)$

W

b

Could be
ReLU
Or tanh

Input layer
(vector)

x_1 x_n +1
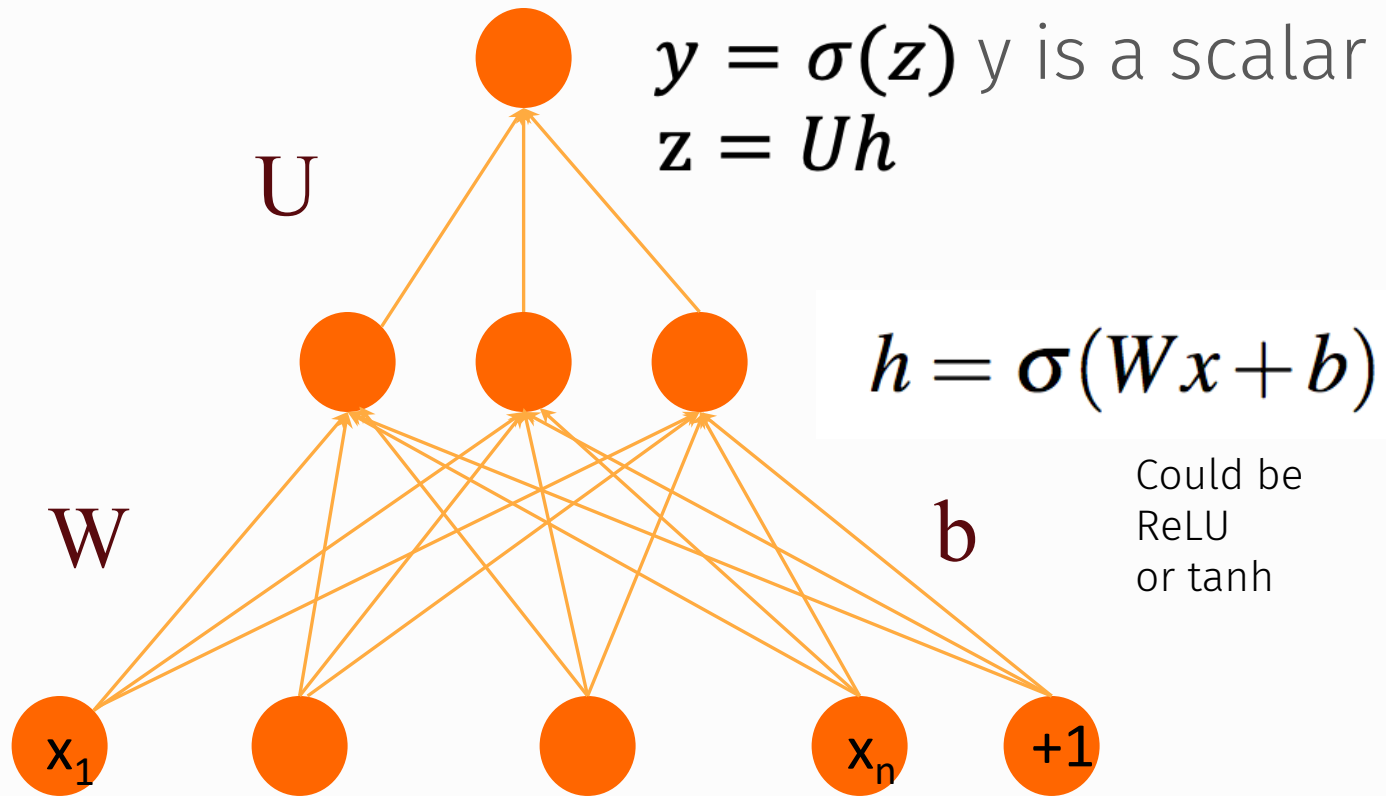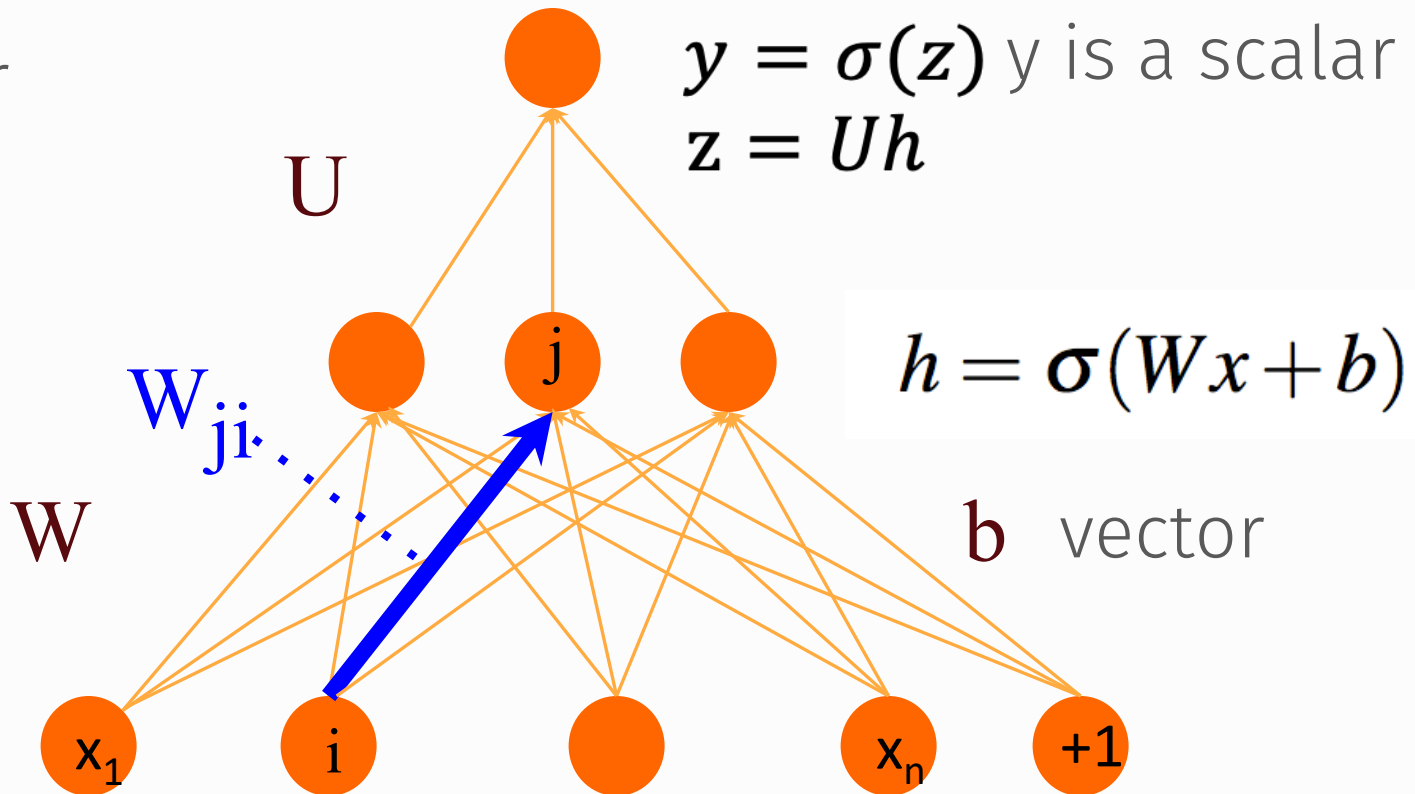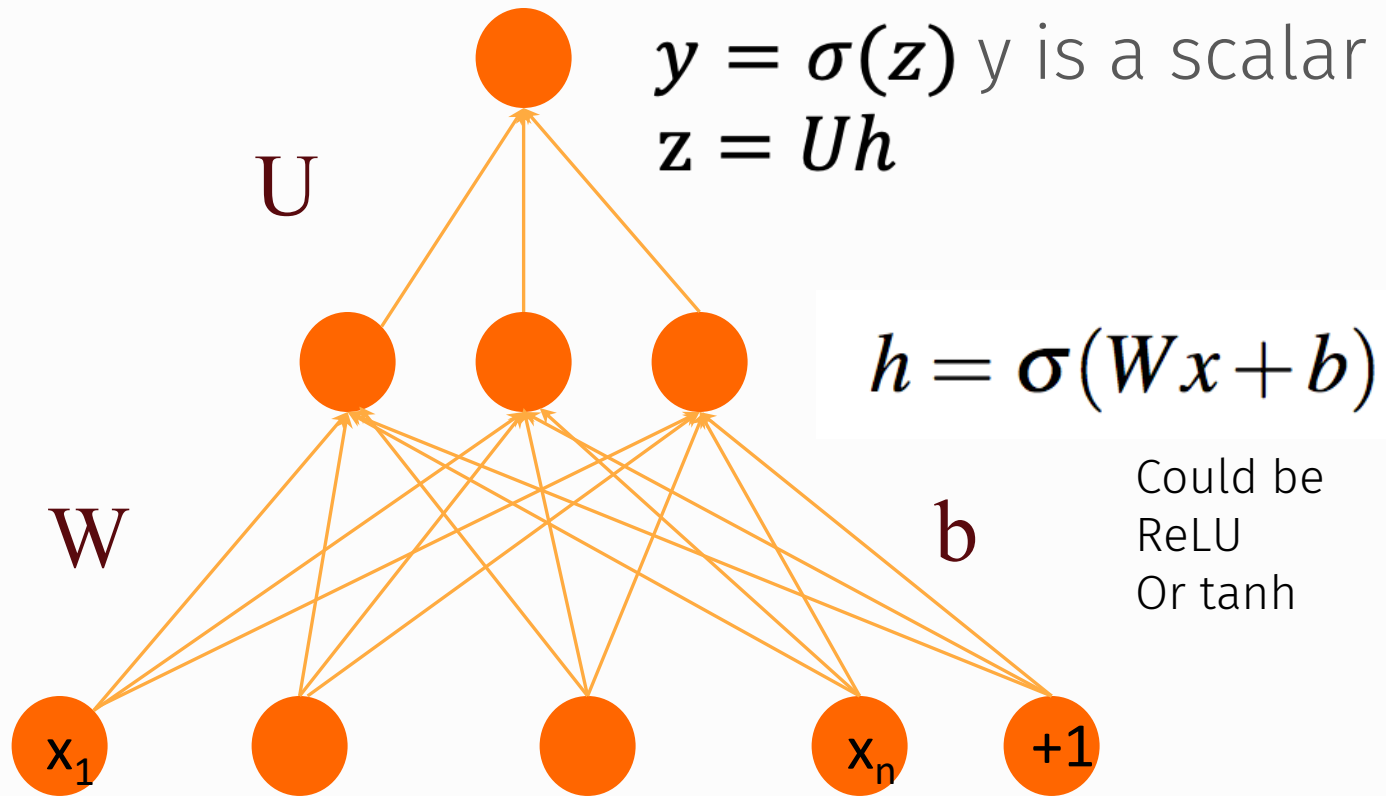
Two-Layer Network with softmax output

Output layer (σ node)

hidden units (σ node)

Input layer (vector)

U

W

b

$y = \text{softmax}(z)$
$z = Uh$

y is a vector

$h = \boldsymbol{\sigma}(Wx + b)$

Could be ReLU Or tanh
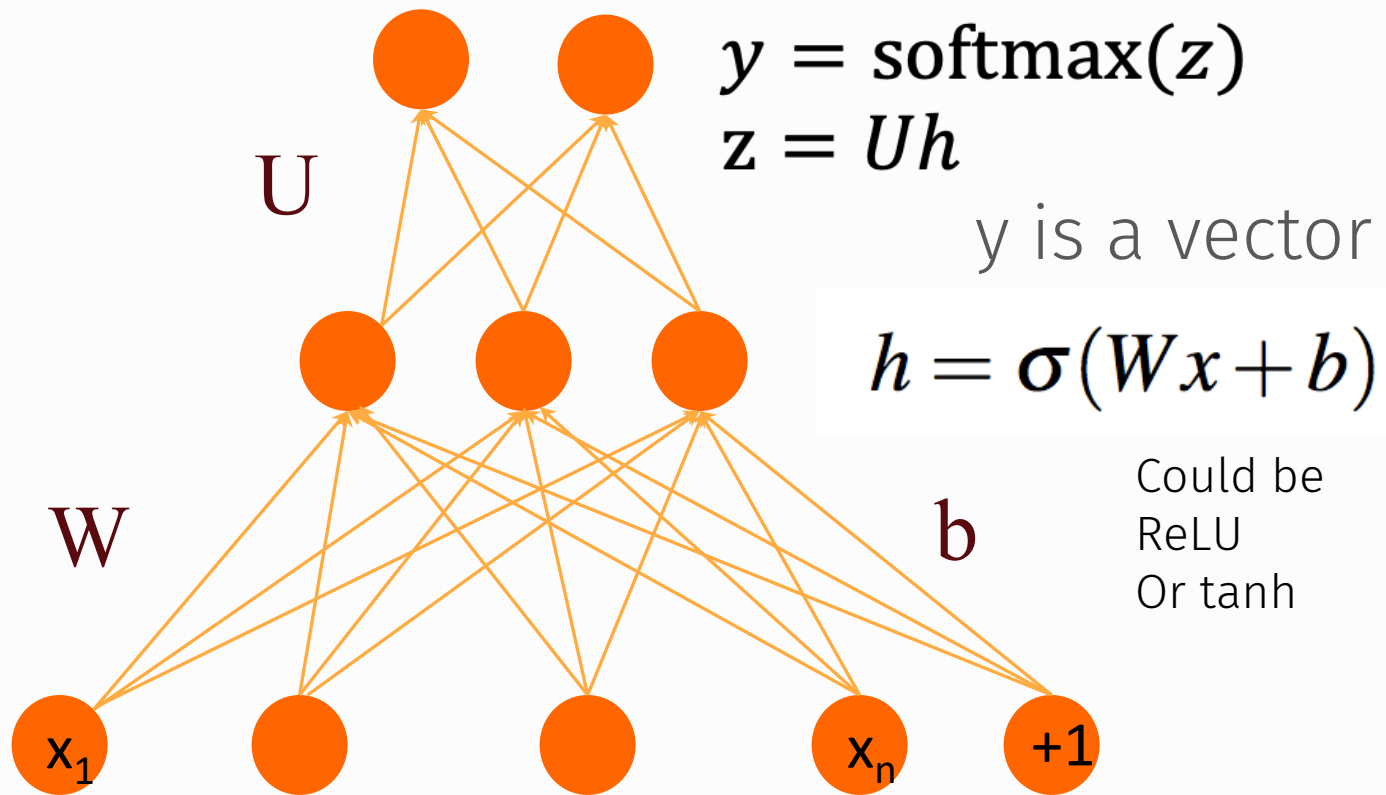
Multi-layer Notation
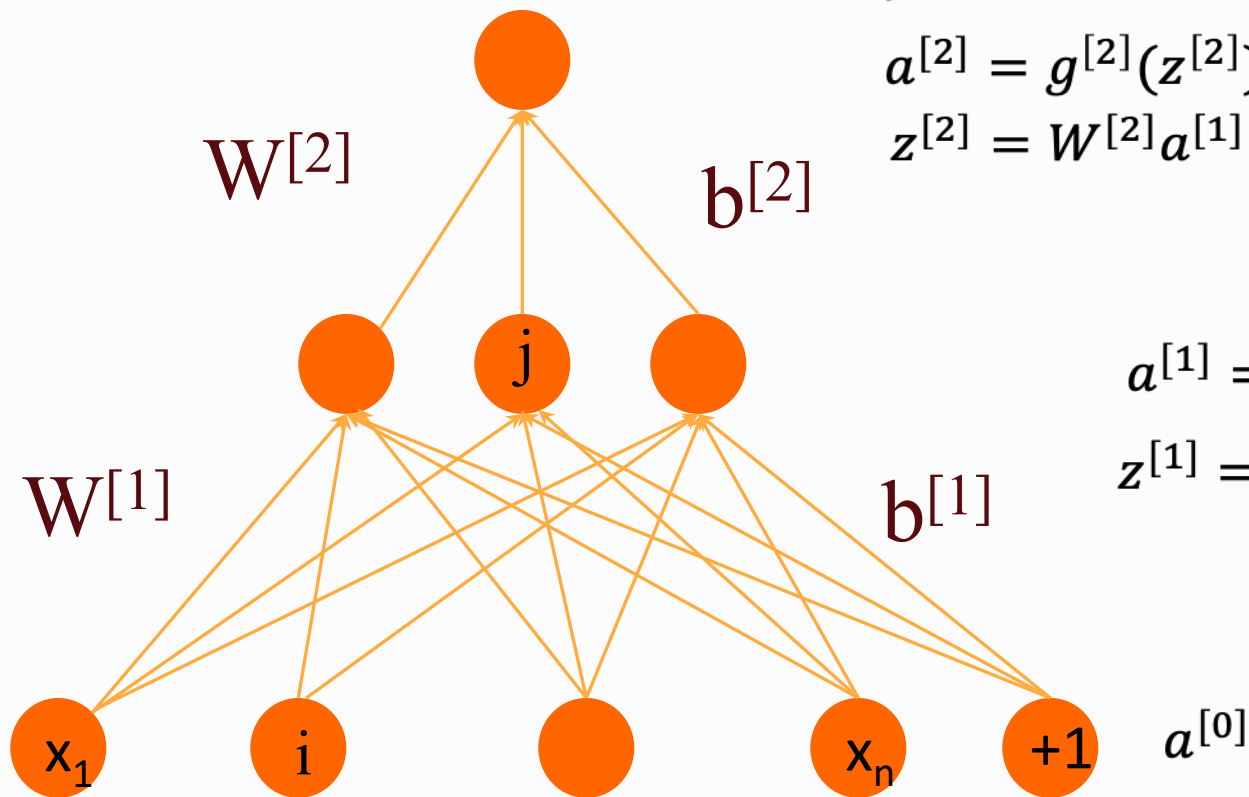
$$y = a^{[2]}$$

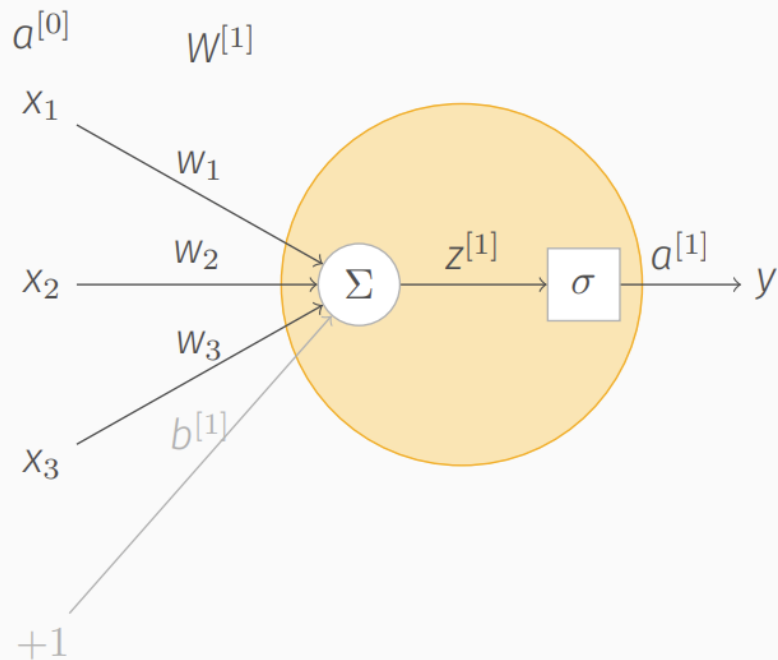$$a^{[2]} = g^{[2]}(z^{[2]}) \quad \text{sigmoid or softmax}$$

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

$$W^{[2]}$$

$$b^{[2]}$$

$$a^{[1]} = g^{[1]}(z^{[1]}) \quad \text{ReLU}$$

$$z^{[1]} = W^{[1]}a^{[0]} + b^{[1]}$$

$$W^{[1]}$$

$$b^{[1]}$$

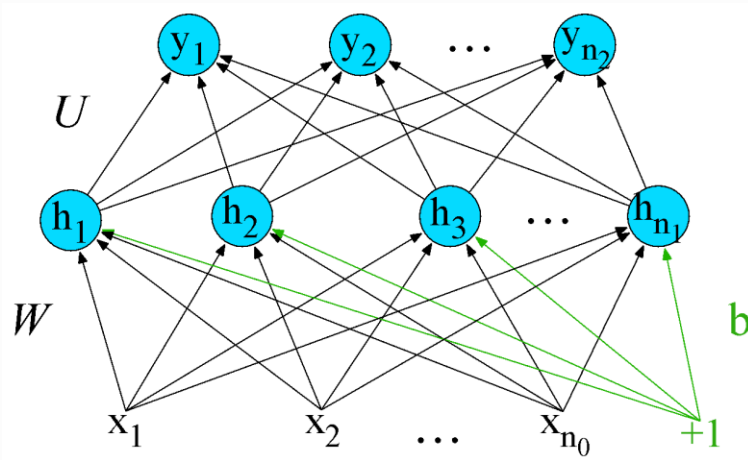$$a^{[0]}$$

$a^{[0]}$

$W^{[1]}$

x_1

w_1

w_2

x_2

w_3

$b^{[1]}$

x_3

$+1$
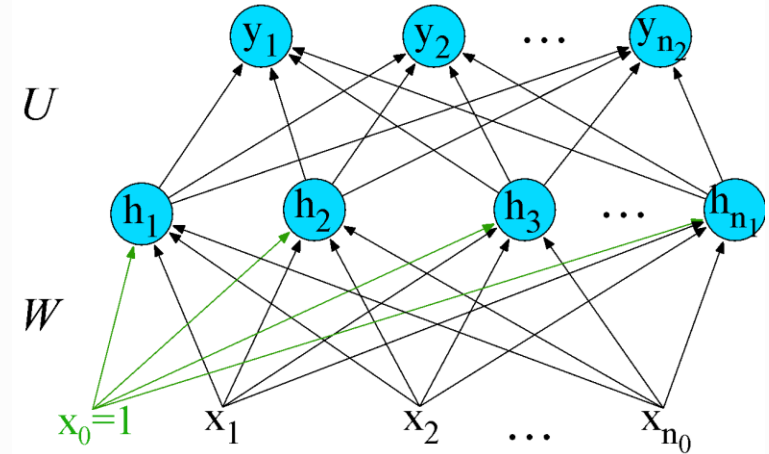
Σ

$z^{[1]}$

σ

$a^{[1]}$

y

for each $i \in 1..n$ **do**
    $z^{[i]} \leftarrow W^{[i]}a^{[i-1]} + b^{[i]}$
    $a^{[i]} \leftarrow g^{[i]}(z^{[i]})$
end for
$\hat{y} \leftarrow a^{[n]}$

Slide credit: David Mortensen

Replacing the bias unit
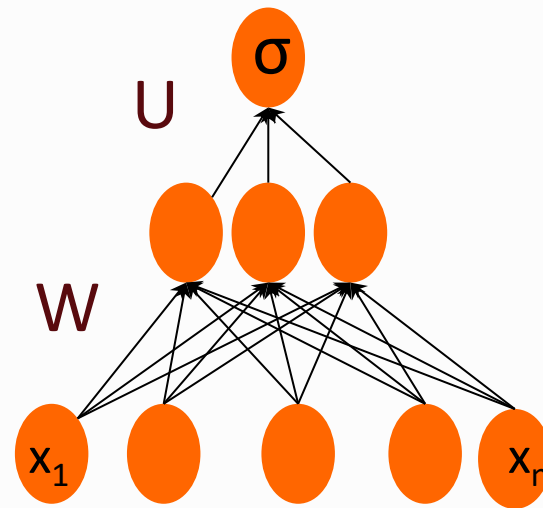
Instead of:

We'll do this:

Feedforward neural nets as classifiers

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) $\in$ doc)
x_2	count(negative lexicon) $\in$ 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

W

σ

x_1 x_2 x_n

f_1 f_2 f_n
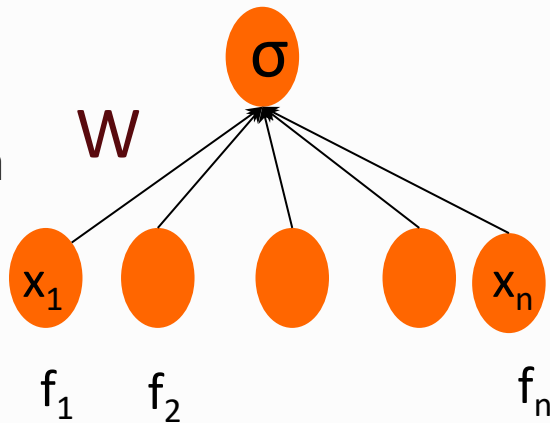
2-layer feedforward network

U

W

σ

x_1 x_n

f_1 f_2 f_n
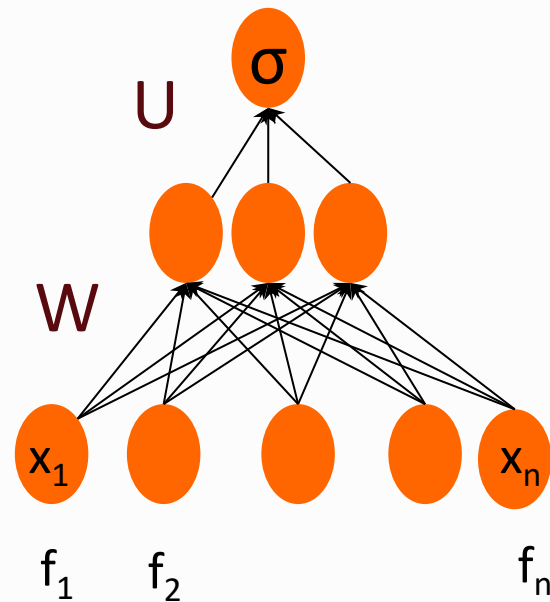
Just adding a hidden layer to logistic regression
- allows the network to use non-linear interactions between features
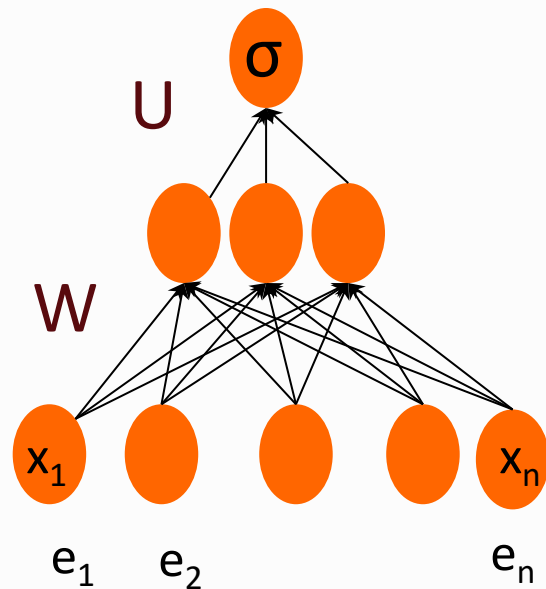- which may (or may not) improve performance.

Feedforward neural networks with word embedding input
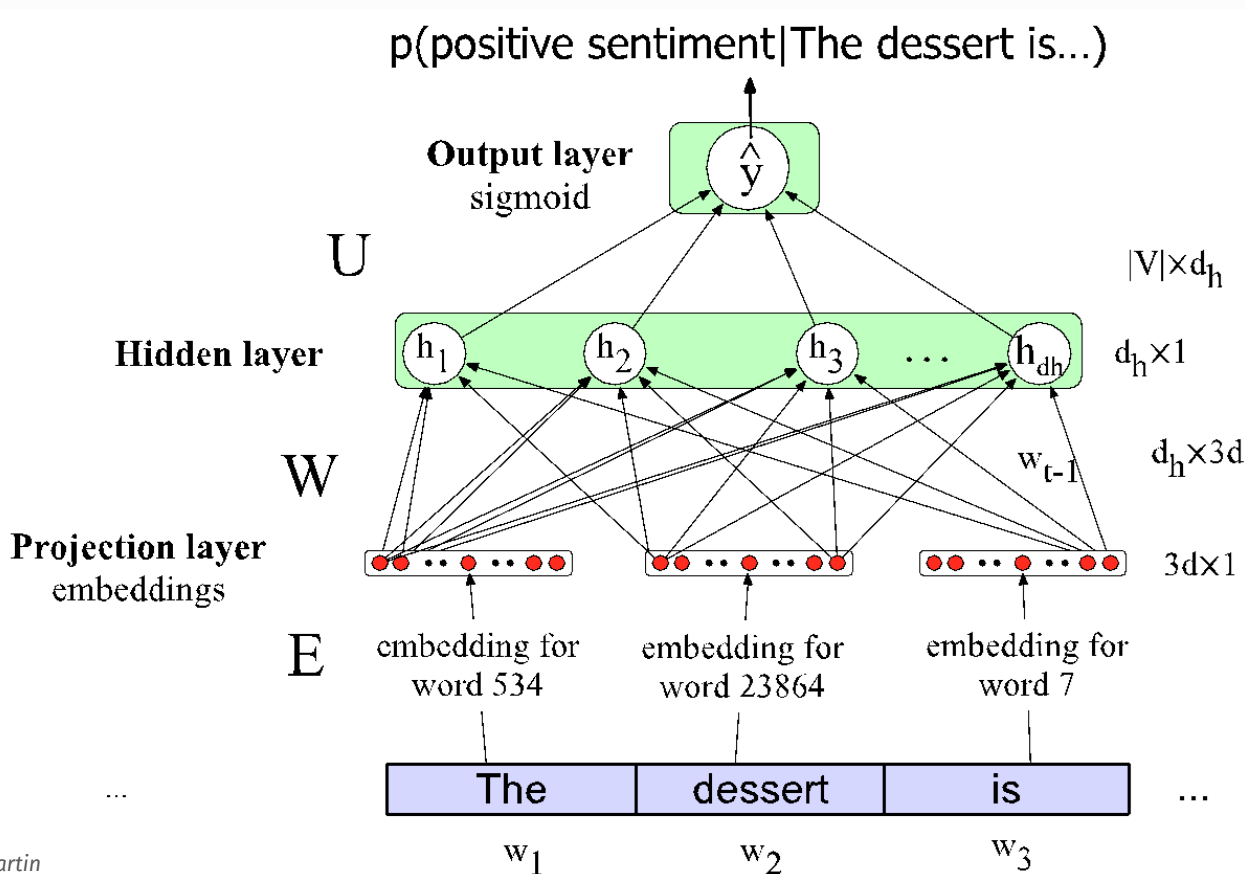
Even better: representation learning

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

Use learned representations like embeddings!

Neural net classification with embeddings as input features!

*Slide adapted from Jurafsky & Martin*

Issue: texts come in different sizes

This assumes a fixed size length (3)!
Kind of unrealistic.
Some simple solutions:

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

41
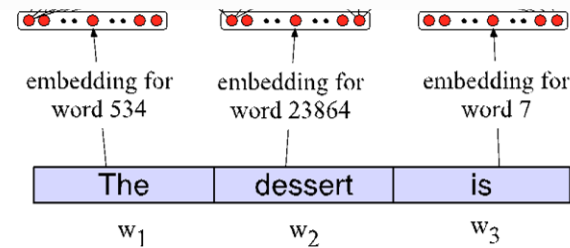
Coding activity

Notebook: feedforward neural network

- [Click on this nbgitpuller link](#)

  - Or find the link on the course website

- Open **session11_ffnn.ipynb**

Questions?