Neural Language Models

CS XXX: Introduction to Large Language Models

Neural Networks

- A model is a function that maps feature vectors to target values.

x_1	x_2	y
6	3300	2050
5	3100	2200

x_1	x_2
6	3300
5	3100

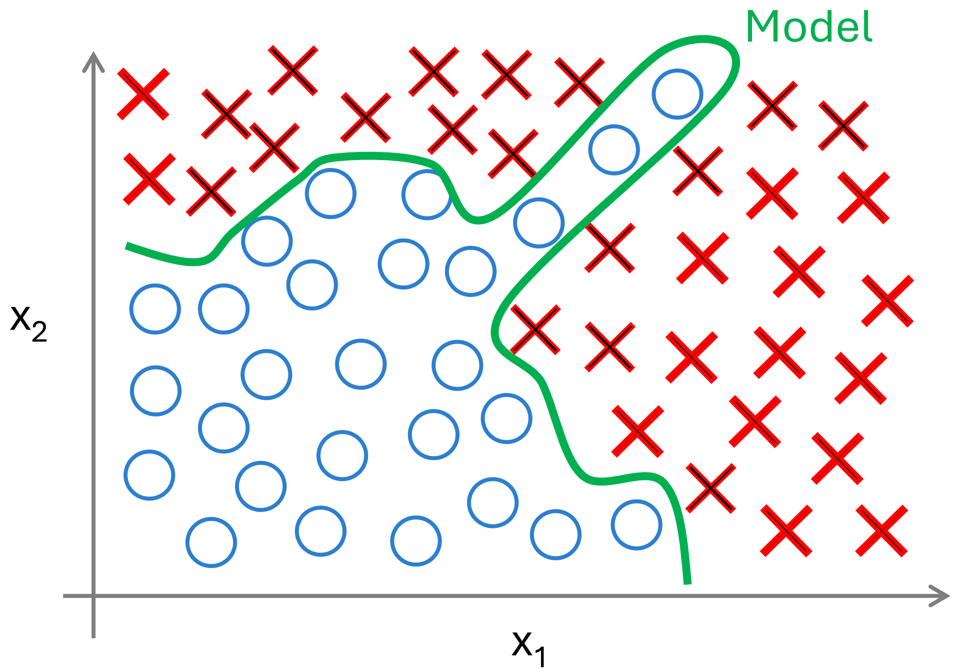
→ Model $h_\theta(x)$ →

y
2050
2200

Neural Networks

- Lots of problems that we want to solve (regression, classification) frequently center around creating functions that are non-linear

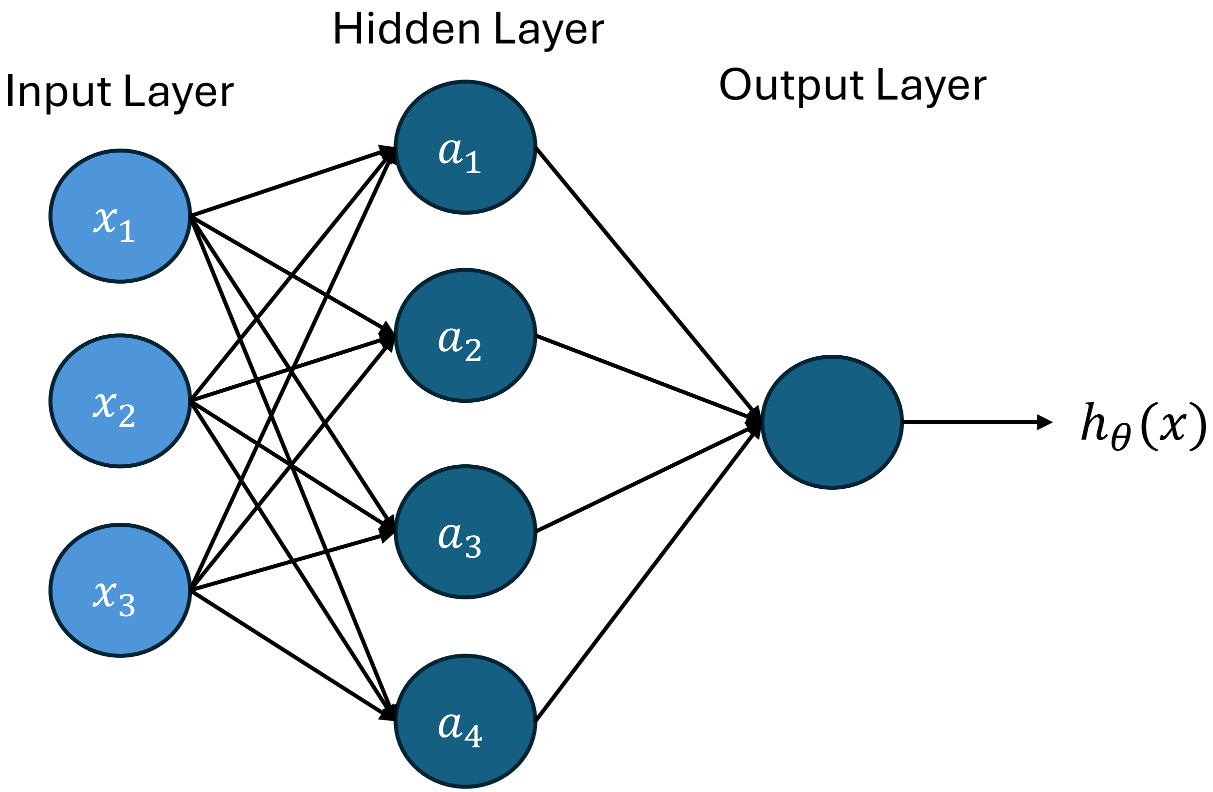
Non-linear Classification



- Is there a model class that allows approximating any function (model)?

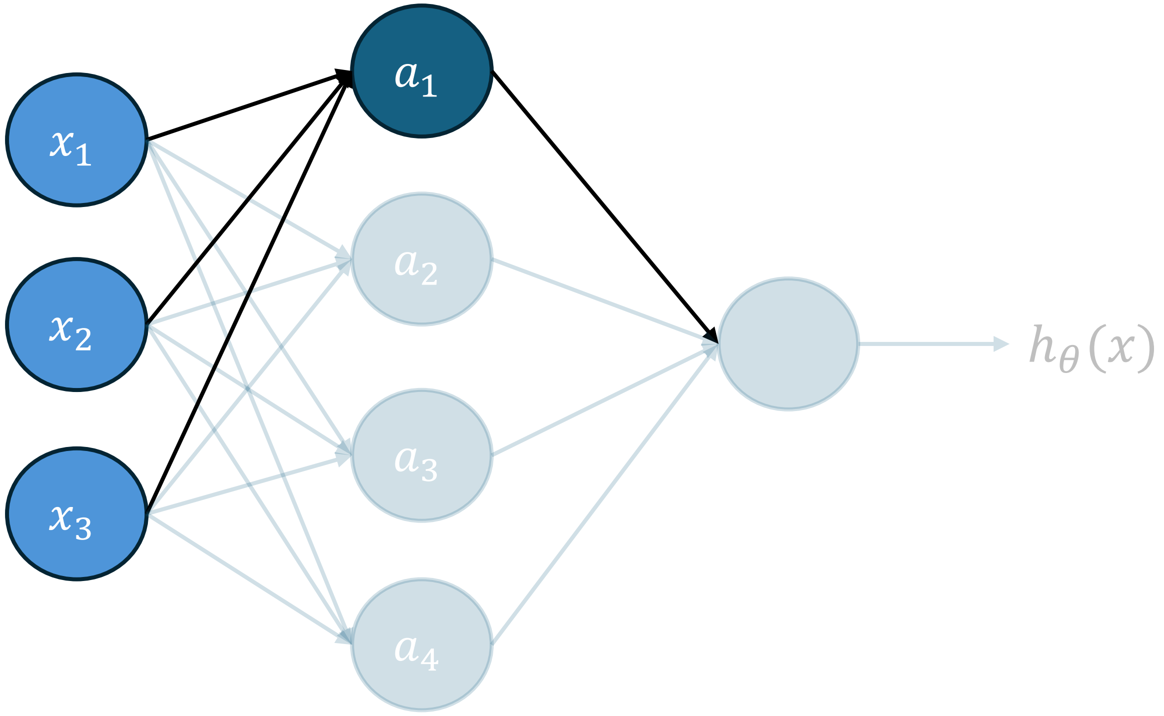
Neural Networks

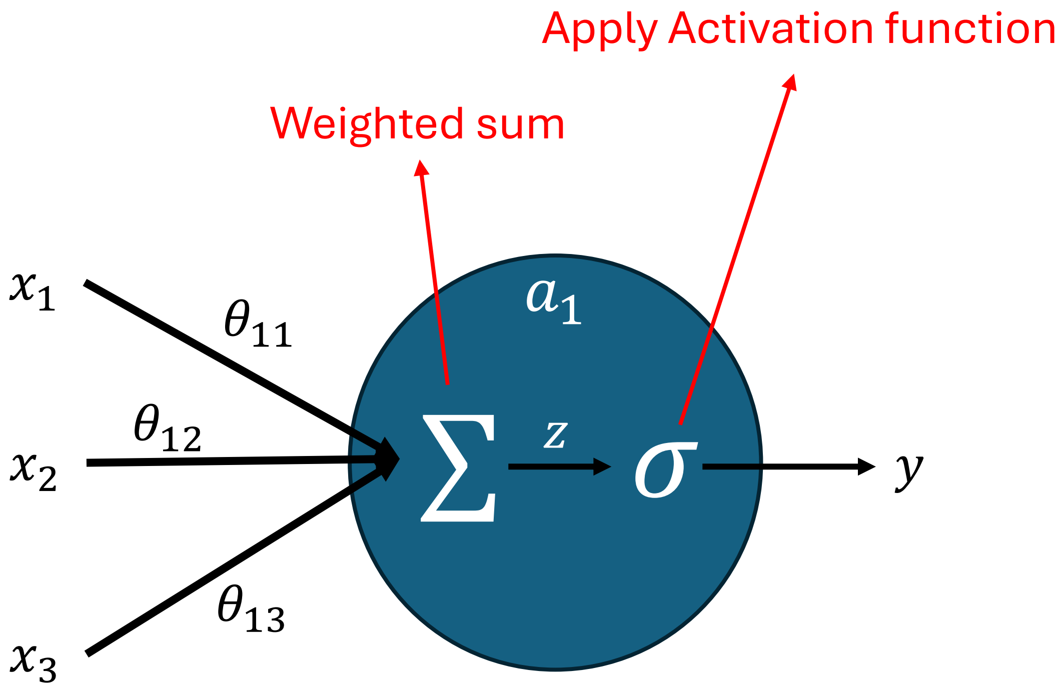
- A Neural Network (NN) with one hidden layer and a sufficient number of neurons, using a non-linear activation function (like sigmoid, ReLU, or tanh), can approximate any continuous function on a compact domain to any desired degree of accuracy.



Neural Networks

- Neuron Model



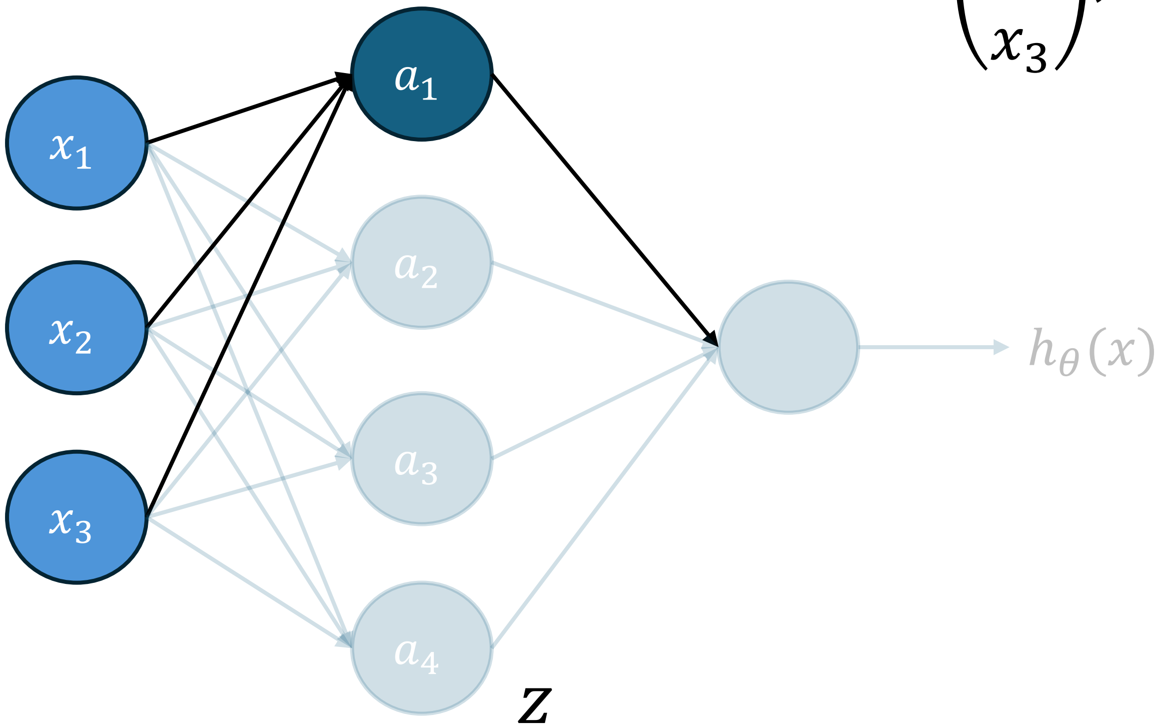


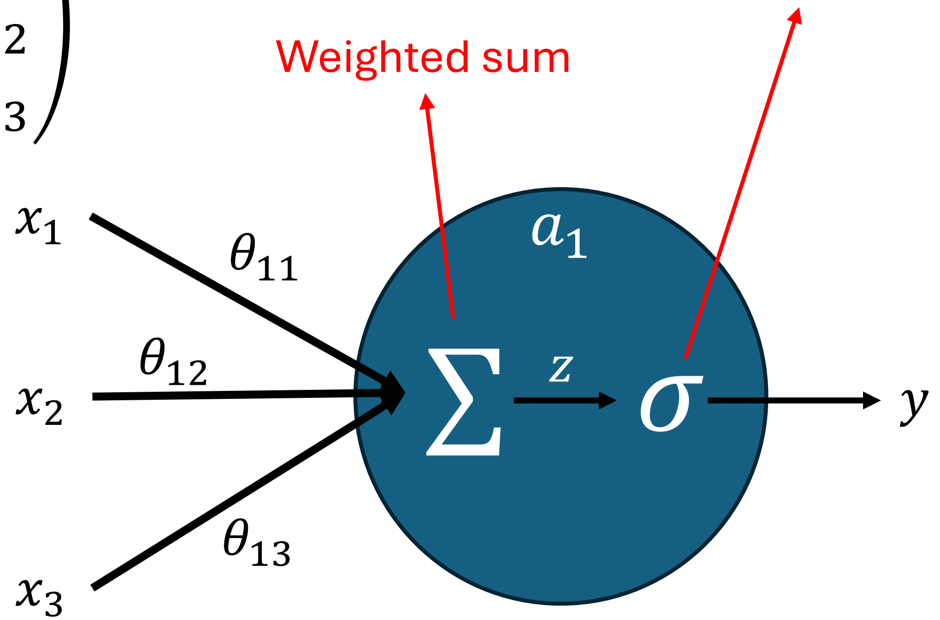
$$y = \boldsymbol{\sigma}(\overbrace{\theta_{11}x_1 + \theta_{12}x_2 + \theta_{13}x_3}^{z})$$

Neural Networks

- Neuron Model

$$\mathrm{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}, \quad \theta_1 = \begin{pmatrix} \theta_{11} \\ \theta_{12} \\ \theta_{13} \end{pmatrix}$$

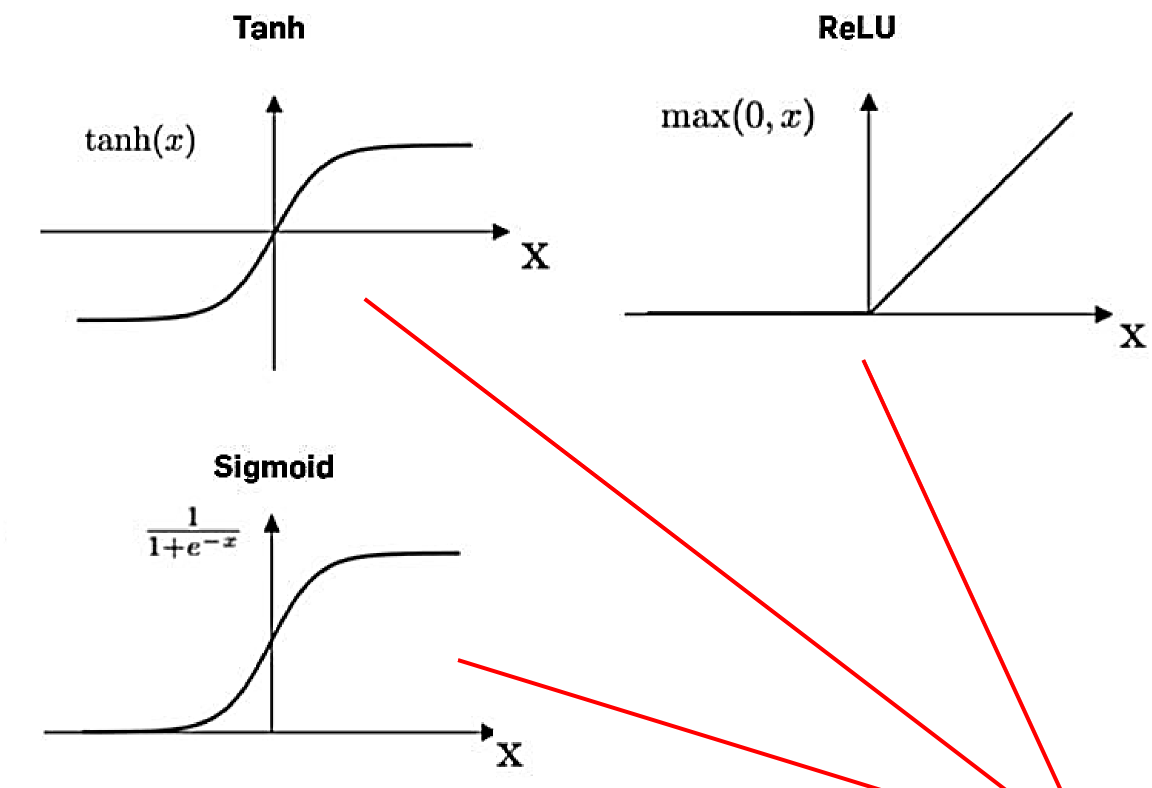


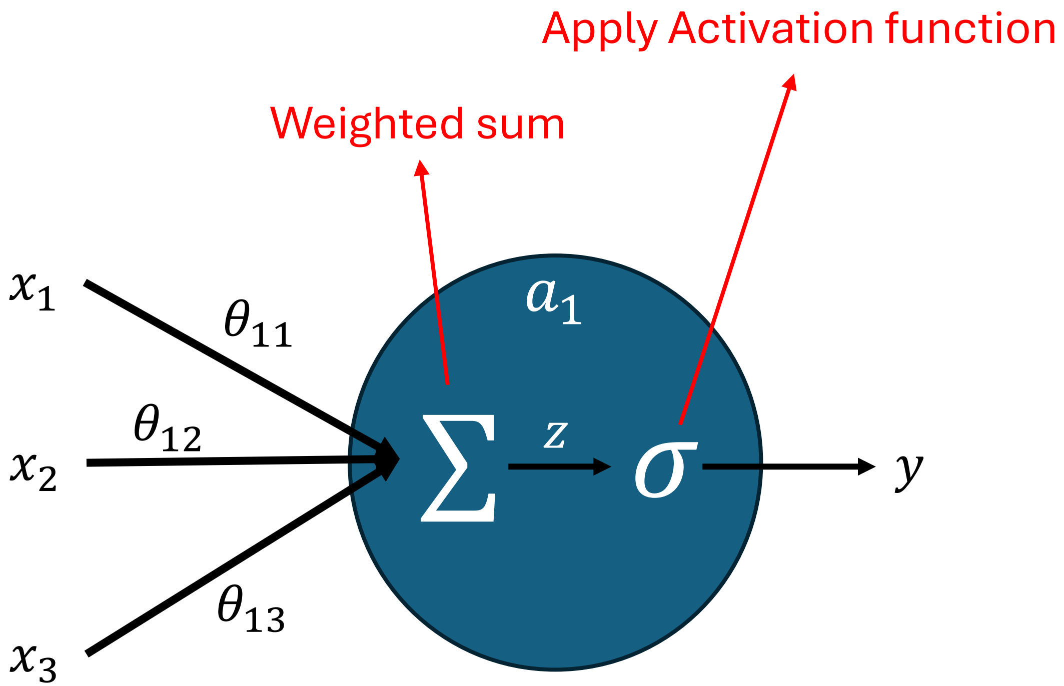


$$\boldsymbol{\sigma}(\underbrace{\theta_{11}x_1 + \theta_{12}x_2 + \theta_{13}x_3}_{z}) = \boldsymbol{\sigma}(\theta_1^{\mathbf{T}} \cdot \mathrm{x}) = \boldsymbol{\sigma}\left(\begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \right)$$

Neural Networks

- Neuron Model



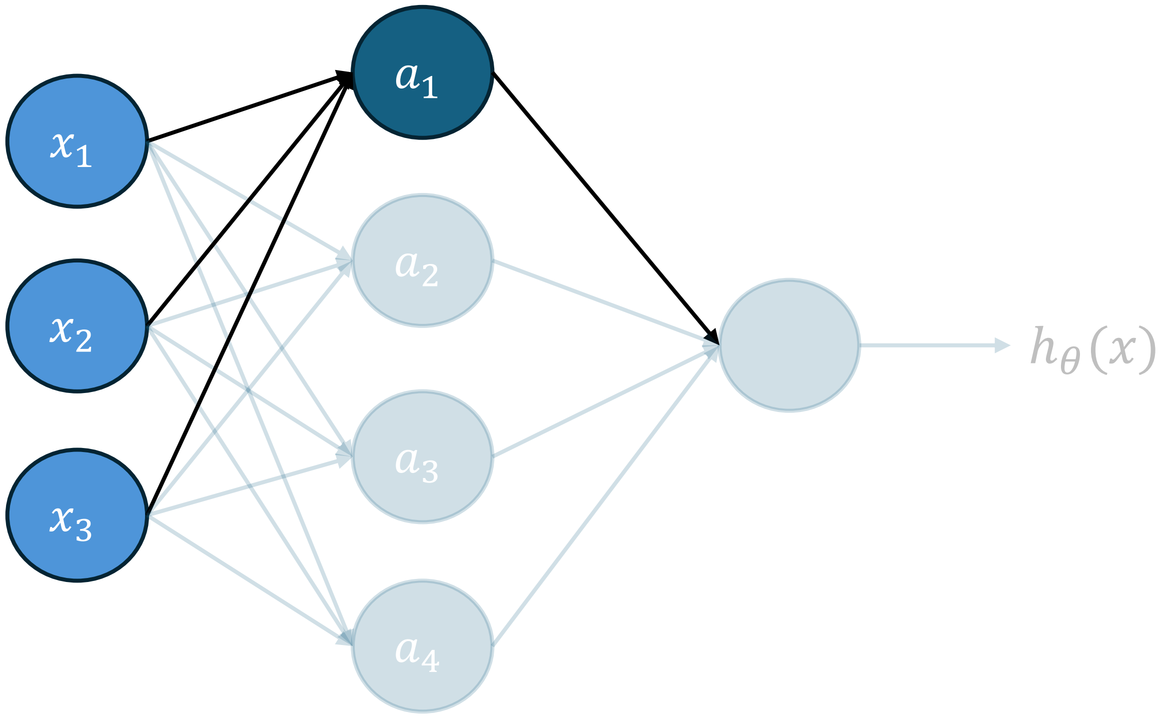


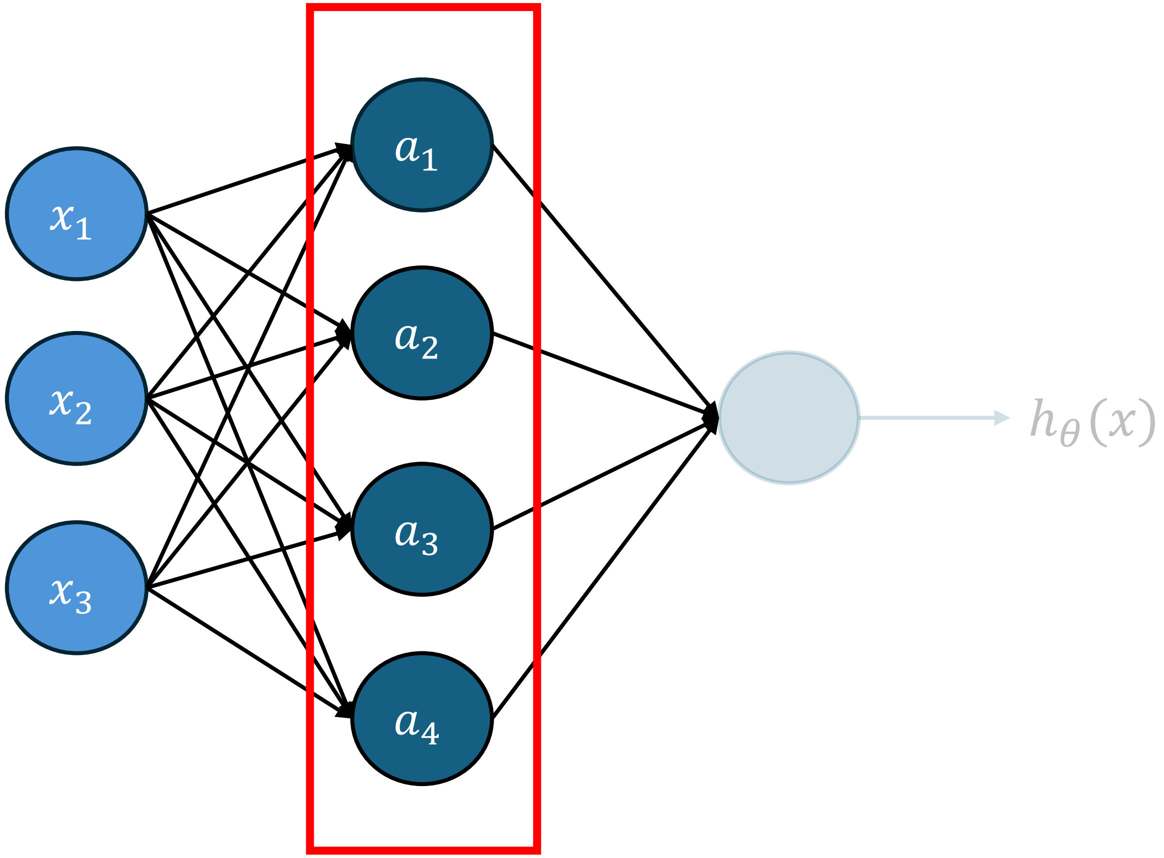
$$\boldsymbol{\sigma}(\theta_1^{\mathbf{T}} \cdot \mathrm{x})$$

e.g. $\mathrm{ReLU}(\theta_1^{\mathbf{T}} \cdot \mathrm{x}) = \max(0, \theta_1^{\mathbf{T}} \cdot \mathrm{x})$

Neural Networks

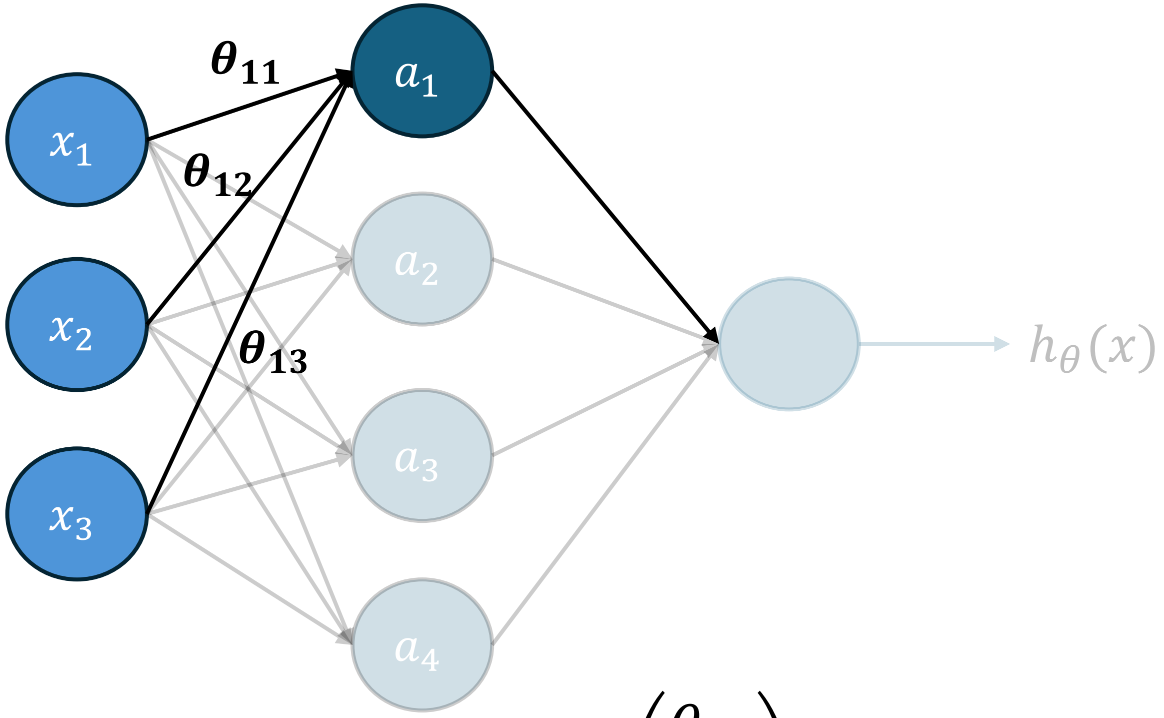
- Perceptron (Neuron) Layer consists of stacked neurons that share the same inputs



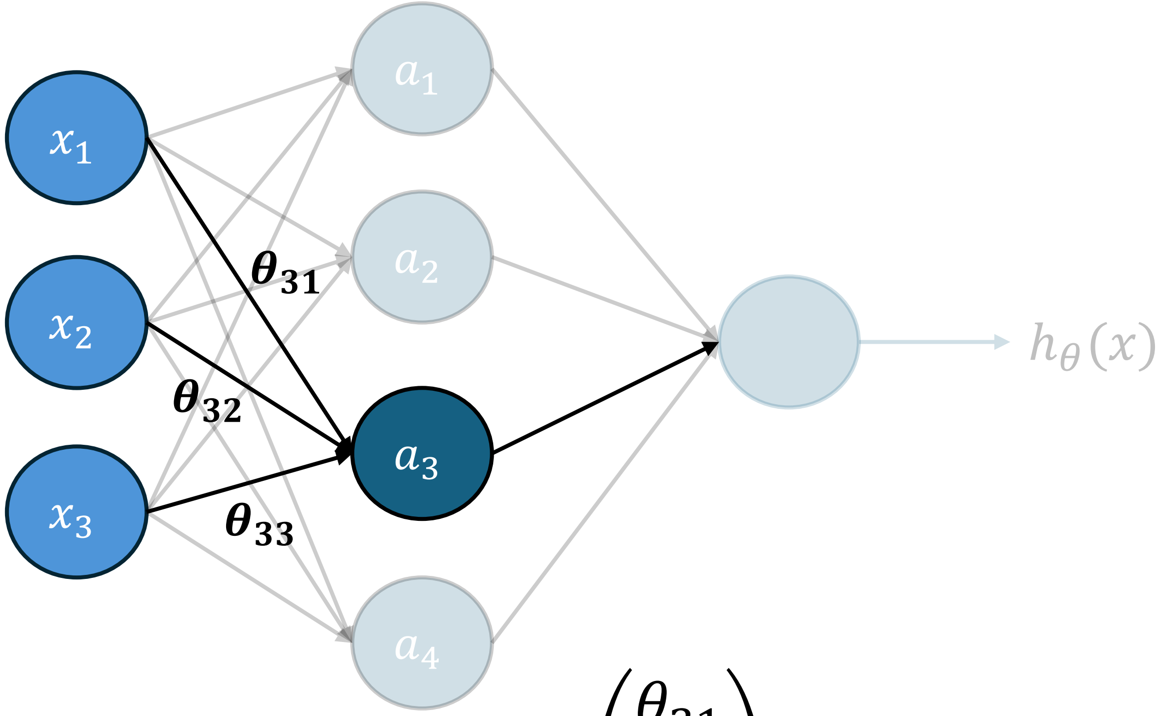


Neural Networks

- Perceptron (Neuron) Layer consists of stacked neurons that share the same inputs



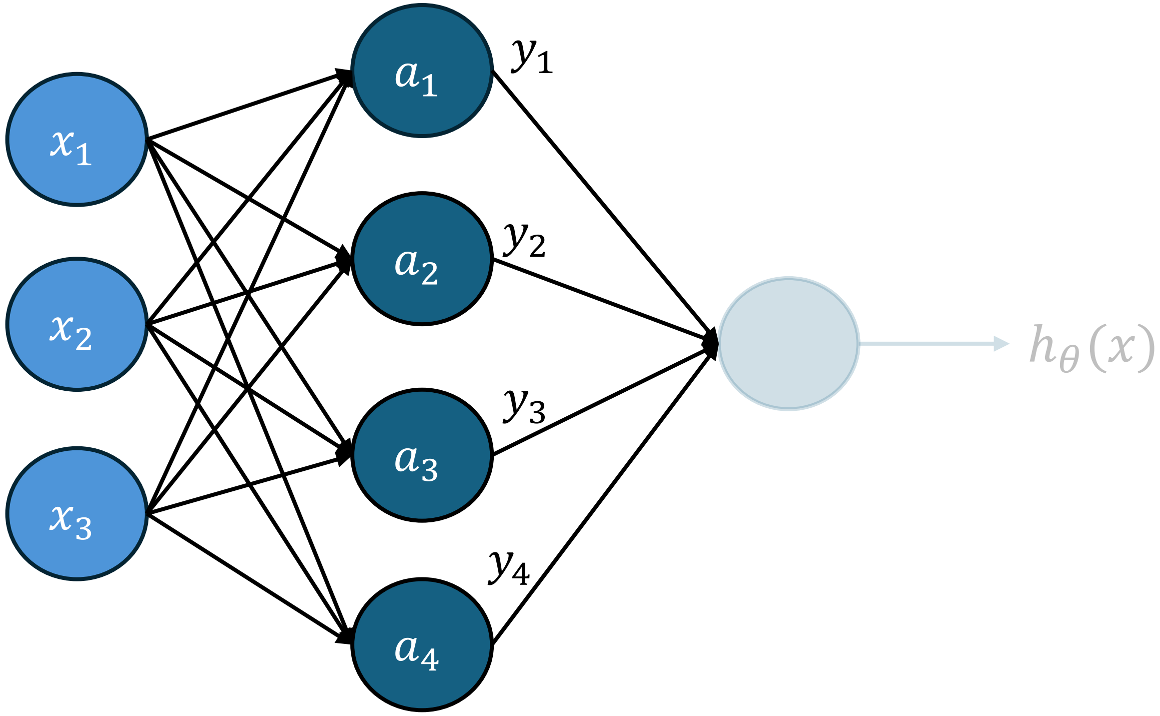
$$\theta_1 = \begin{pmatrix} \theta_{11} \\ \theta_{12} \\ \theta_{13} \end{pmatrix}$$



$$\theta_3 = \begin{pmatrix} \theta_{31} \\ \theta_{32} \\ \theta_{33} \end{pmatrix}$$

Neural Networks

- Perceptron (Neuron) Layer consists of stacked neurons that share the same inputs



$$\theta_1 = \begin{pmatrix} \theta_{11} \\ \theta_{12} \\ \theta_{13} \end{pmatrix},\ \theta_2 = \begin{pmatrix} \theta_{21} \\ \theta_{22} \\ \theta_{23} \end{pmatrix},\ \theta_3 = \begin{pmatrix} \theta_{31} \\ \theta_{32} \\ \theta_{33} \end{pmatrix},\ \theta_4 = \begin{pmatrix} \theta_{41} \\ \theta_{42} \\ \theta_{43} \end{pmatrix}$$

$$y_1 = \boldsymbol{\sigma}(\theta_1^{\mathbf{T}} \cdot \mathrm{x})$$

$$y_2 = \boldsymbol{\sigma}(\theta_2^{\mathbf{T}} \cdot \mathrm{x})$$

$$y_3 = \boldsymbol{\sigma}(\theta_3^{\mathbf{T}} \cdot \mathrm{x})$$

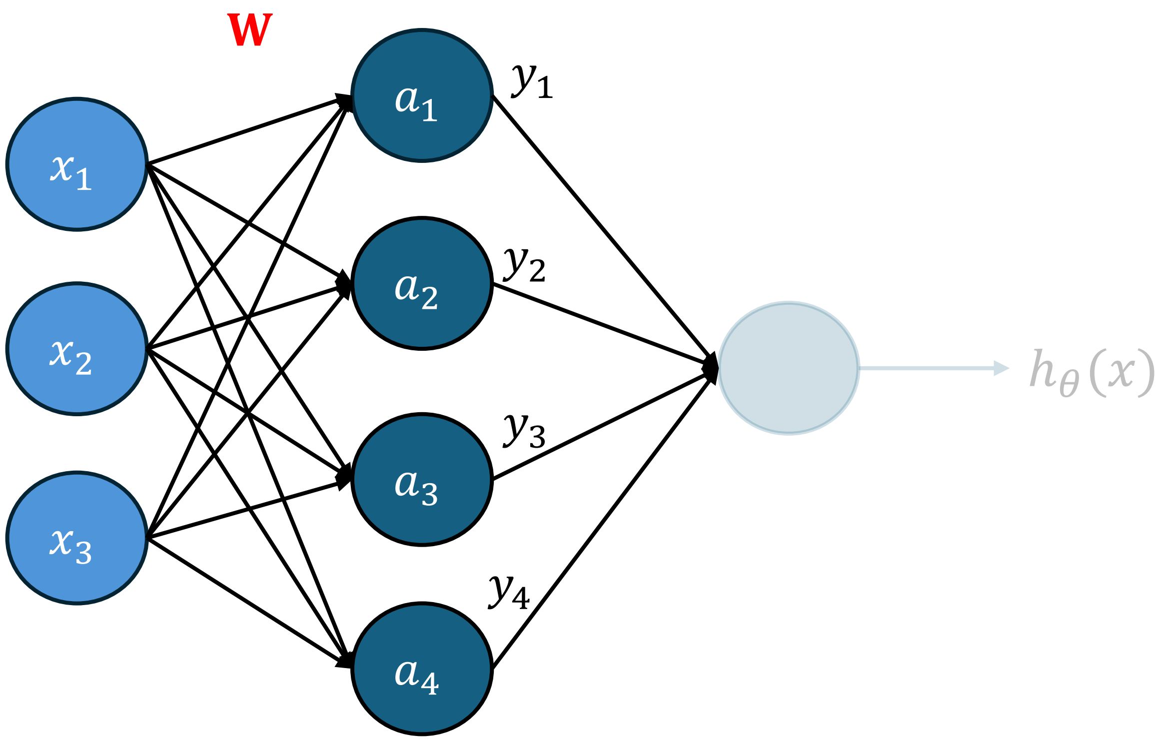
$$y_4 = \boldsymbol{\sigma}(\theta_4^{\mathbf{T}} \cdot \mathrm{x})$$

Use matrices instead

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \boldsymbol{\sigma}\left(\begin{pmatrix} \theta_1^{\mathbf{T}} \\ \theta_2^{\mathbf{T}} \\ \theta_3^{\mathbf{T}} \\ \theta_4^{\mathbf{T}} \end{pmatrix} \cdot \mathrm{x}\right)$$

Neural Networks

- Perceptron (Neuron) Layer consists of stacked neurons that share the same inputs



Use matrices instead

$$\underset{\mathrm{y}}{\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}} = \boldsymbol{\sigma}\left(\underset{\mathbf{W}}{\begin{pmatrix} \theta_1^{\mathbf{T}} \\ \theta_2^{\mathbf{T}} \\ \theta_3^{\mathbf{T}} \\ \theta_4^{\mathbf{T}} \end{pmatrix}} \cdot \mathrm{x} \right)$$

Weight Matrix: Matrix of parameters

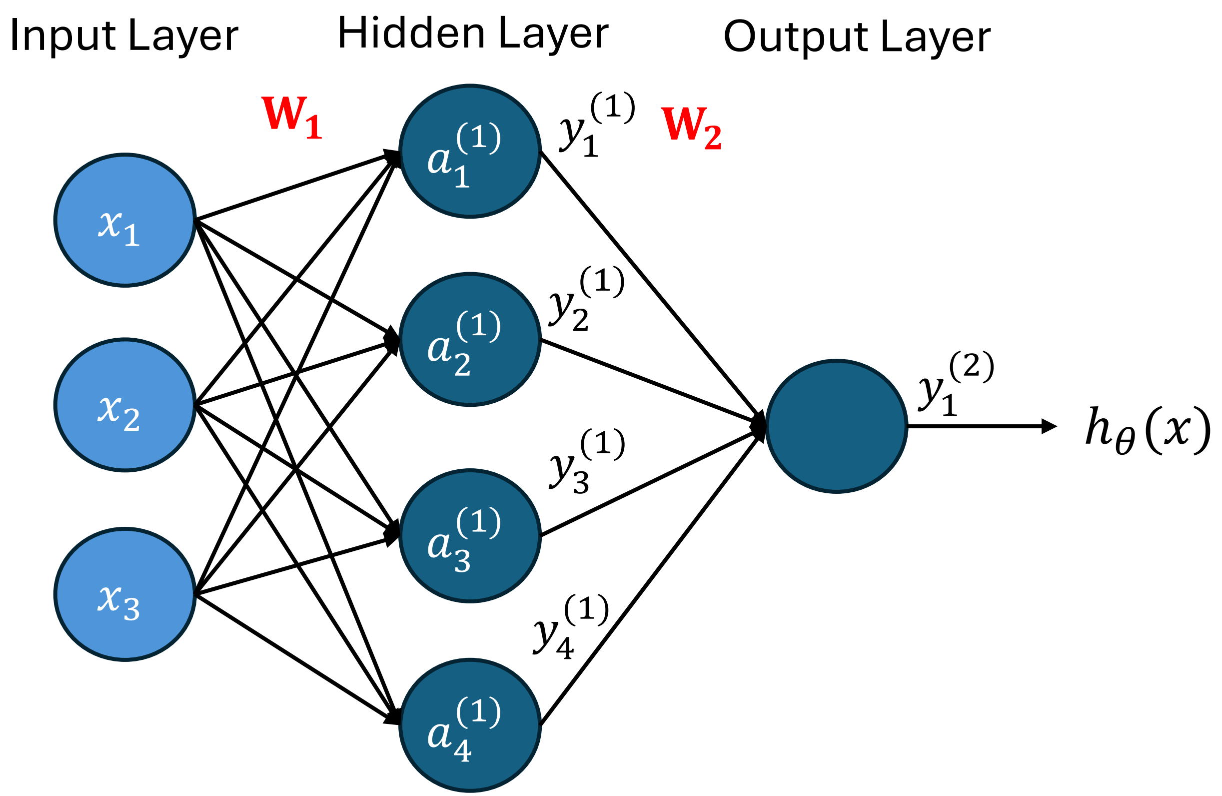
$$\mathbf{W} = \begin{pmatrix} \theta_1^{\mathbf{T}} \\ \theta_2^{\mathbf{T}} \\ \theta_3^{\mathbf{T}} \\ \theta_4^{\mathbf{T}} \end{pmatrix} = \begin{pmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \\ \theta_{31} & \theta_{32} & \theta_{33} \\ \theta_{41} & \theta_{42} & \theta_{43} \end{pmatrix}$$

Therefore,

$$\mathrm{y} = \boldsymbol{\sigma}(\mathbf{W}\mathrm{x}) = \boldsymbol{\sigma}\left(\begin{pmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \\ \theta_{31} & \theta_{32} & \theta_{33} \\ \theta_{41} & \theta_{42} & \theta_{43} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \right)$$

Neural Networks

- We can start to cascade these layers to form the full neural network



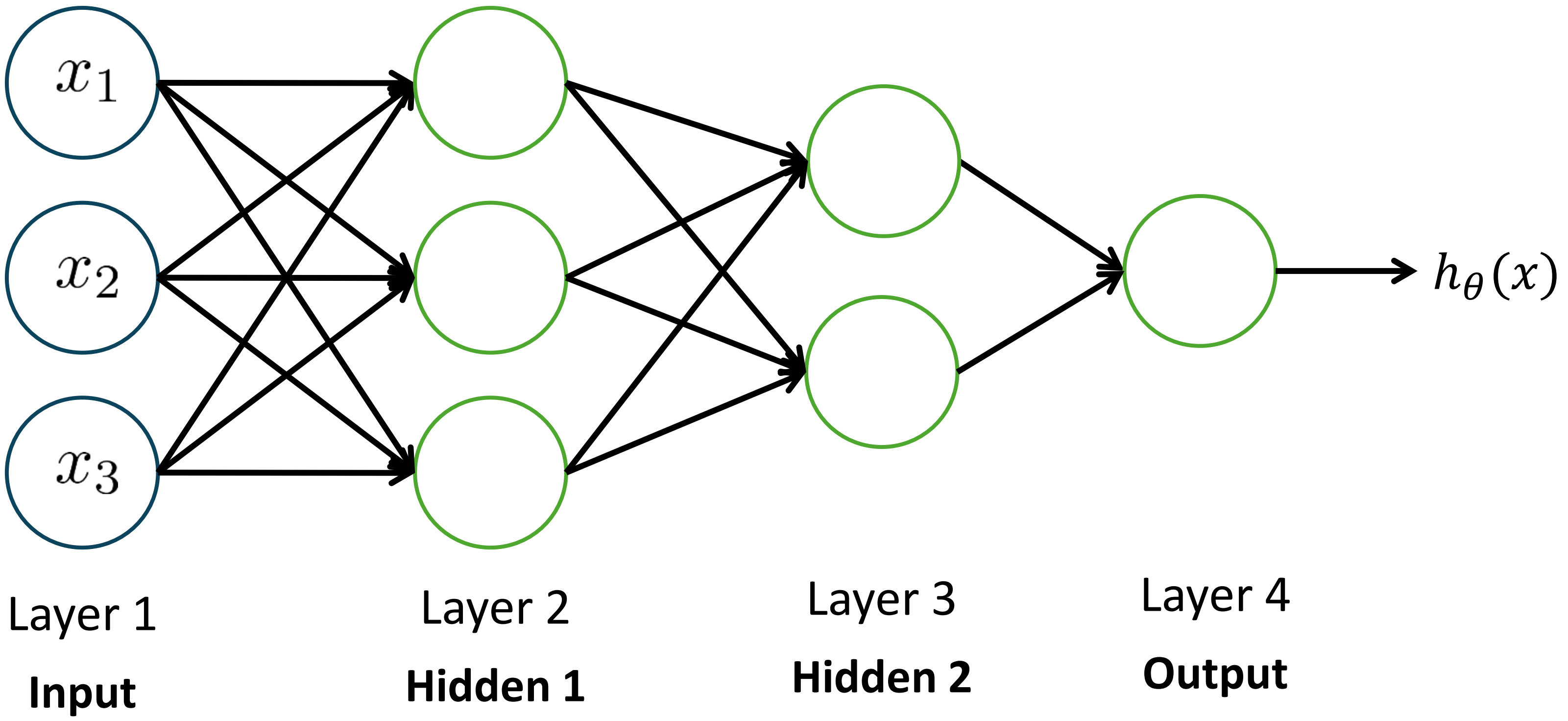
$$\mathrm{y}^{(1)} = \boldsymbol{\sigma_1}(\mathbf{W_1}\mathrm{x}) = \boldsymbol{\sigma_1}\left(\begin{pmatrix} \theta_{11}^{(1)} & \theta_{12}^{(1)} & \theta_{13}^{(1)} \\ \theta_{21}^{(1)} & \theta_{22}^{(1)} & \theta_{23}^{(1)} \\ \theta_{31}^{(1)} & \theta_{32}^{(1)} & \theta_{33}^{(1)} \\ \theta_{41}^{(1)} & \theta_{42}^{(1)} & \theta_{43}^{(1)} \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}\right)$$

$$\mathrm{y}^{(2)} = \boldsymbol{\sigma_2}(\mathbf{W_2}\mathrm{y}^{(1)})$$

$$\begin{aligned} h_\theta(x) &= \mathrm{y}^{(2)} \\ &= \boldsymbol{\sigma_2}(\mathbf{W_2}\mathrm{y}^{(1)}) \\ &= \boldsymbol{\sigma_2}(\mathbf{W_2}(\boldsymbol{\sigma_1}(\mathbf{W_1}\mathrm{x}))) \end{aligned}$$

Neural Networks

- Adding Layers: Deep Neural Networks



Neural Language Models

RECALL

- The classic definition of a language model (LM) is **a probability distribution over sequences of tokens**. Suppose we have a **vocabulary** V of a set of tokens. A language model p assigns each sequence of tokens $x_1, \ldots, x_L \in V$ a probability (a number between 0 and 1):

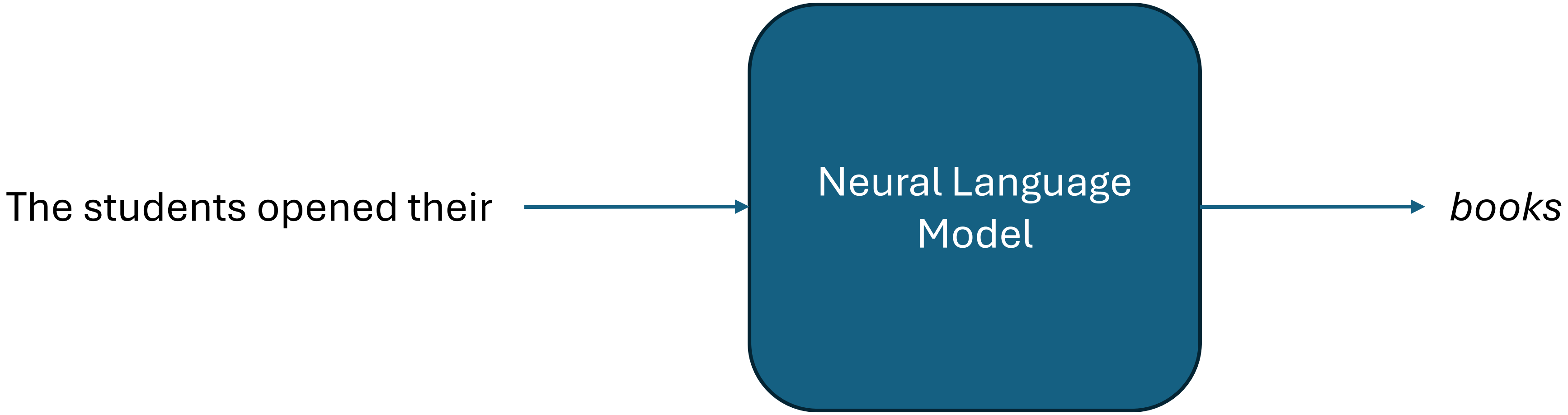
$$p(x_1, \ldots, x_L) = p(x_{1:L})$$

- A model that computes either of these

$p(x_{1:L})$ or $p(x_4 | x_1, x_2, x_3)$ is called a Language Model (LM)

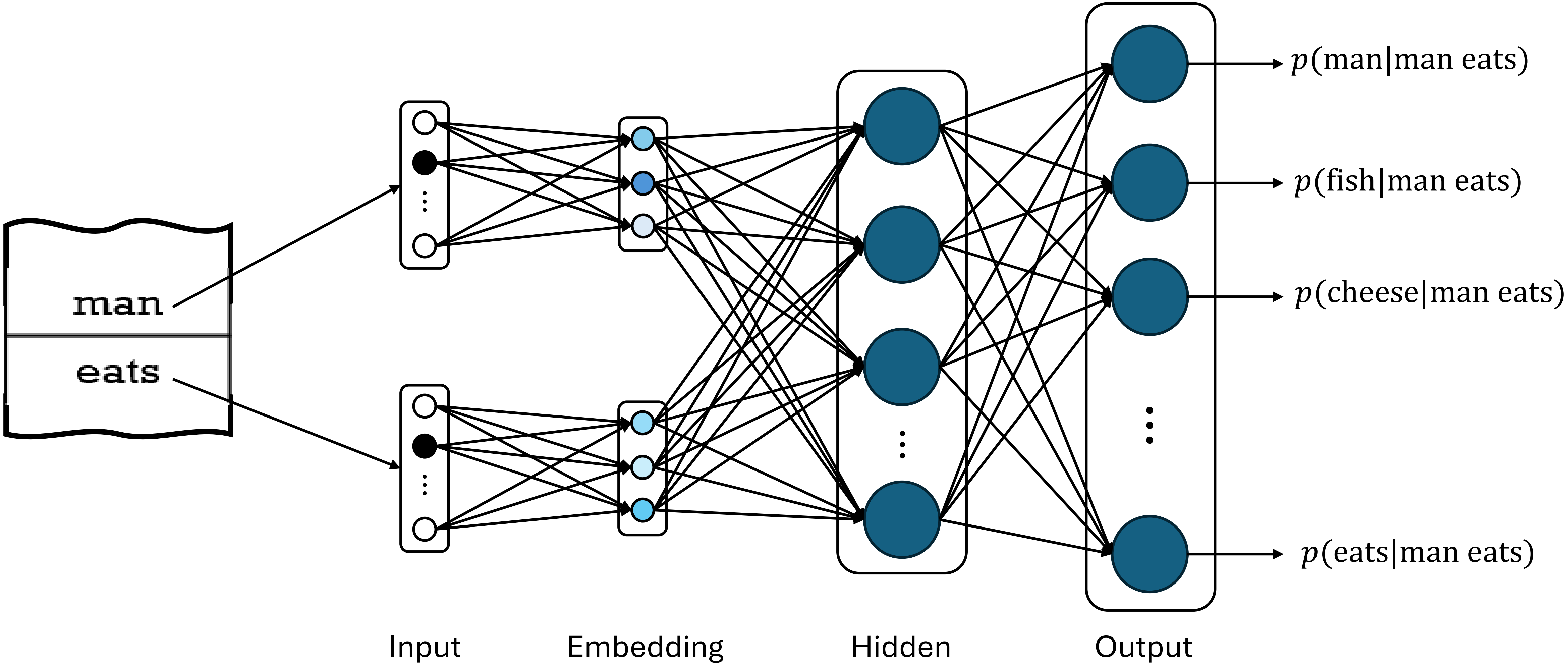
Neural Language Models

- Enter Neural Networks

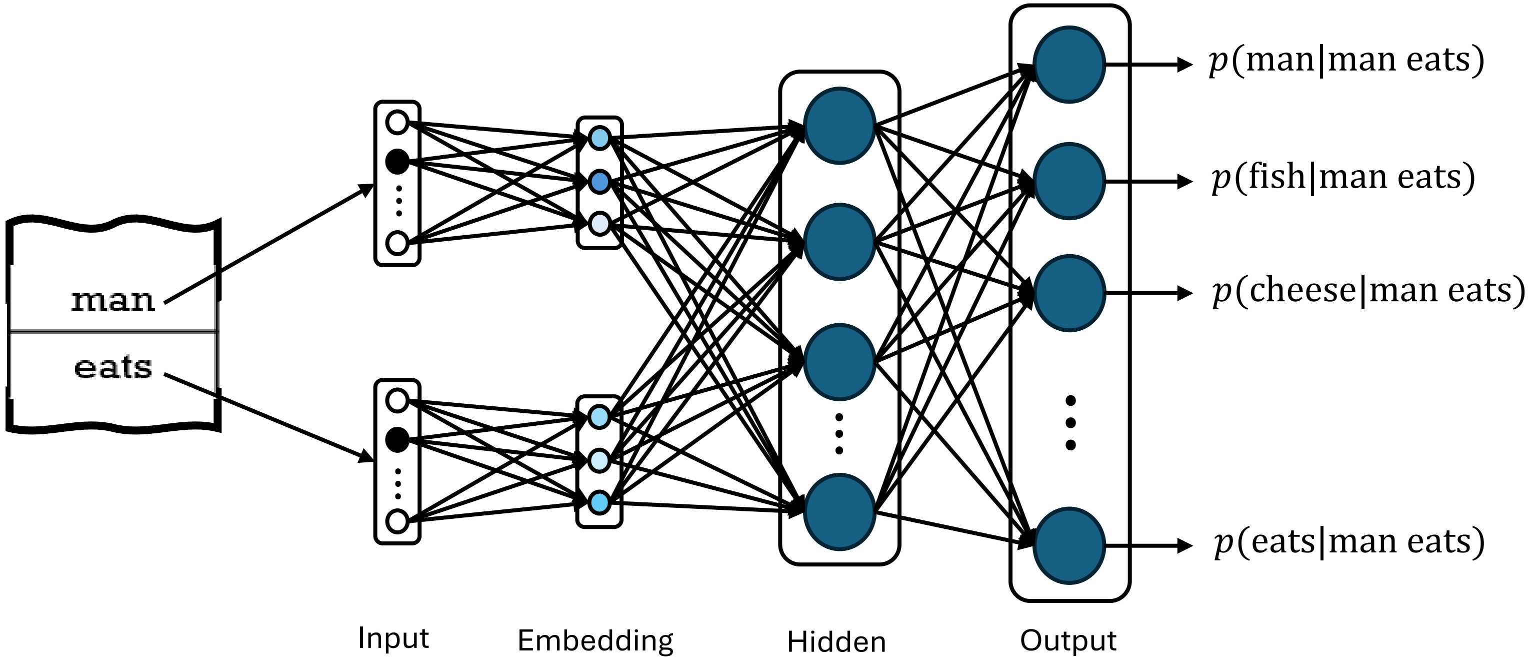


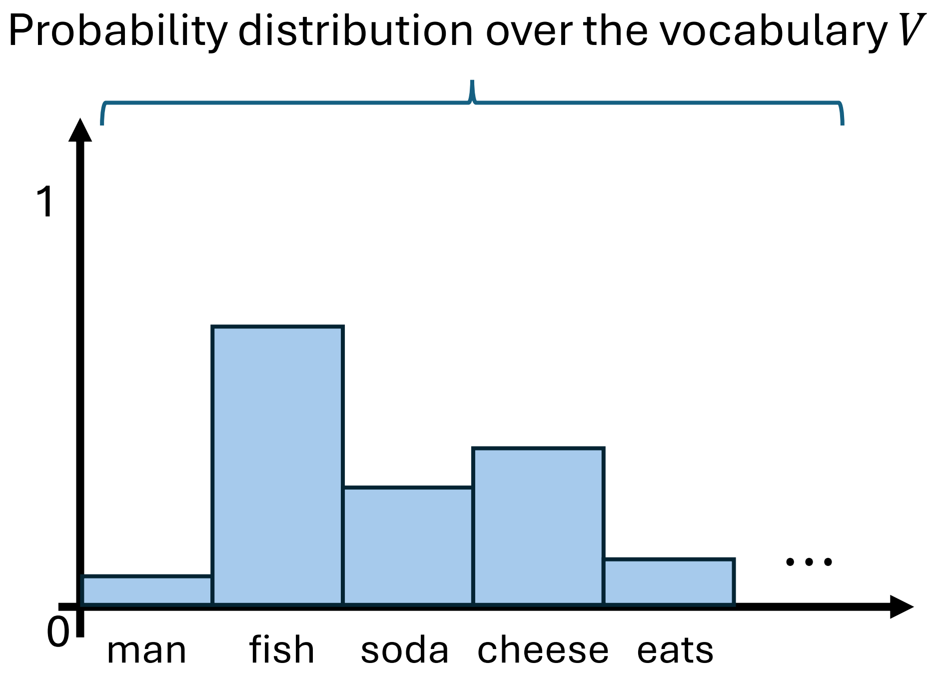
Neural Language Models

- Enter Neural Networks

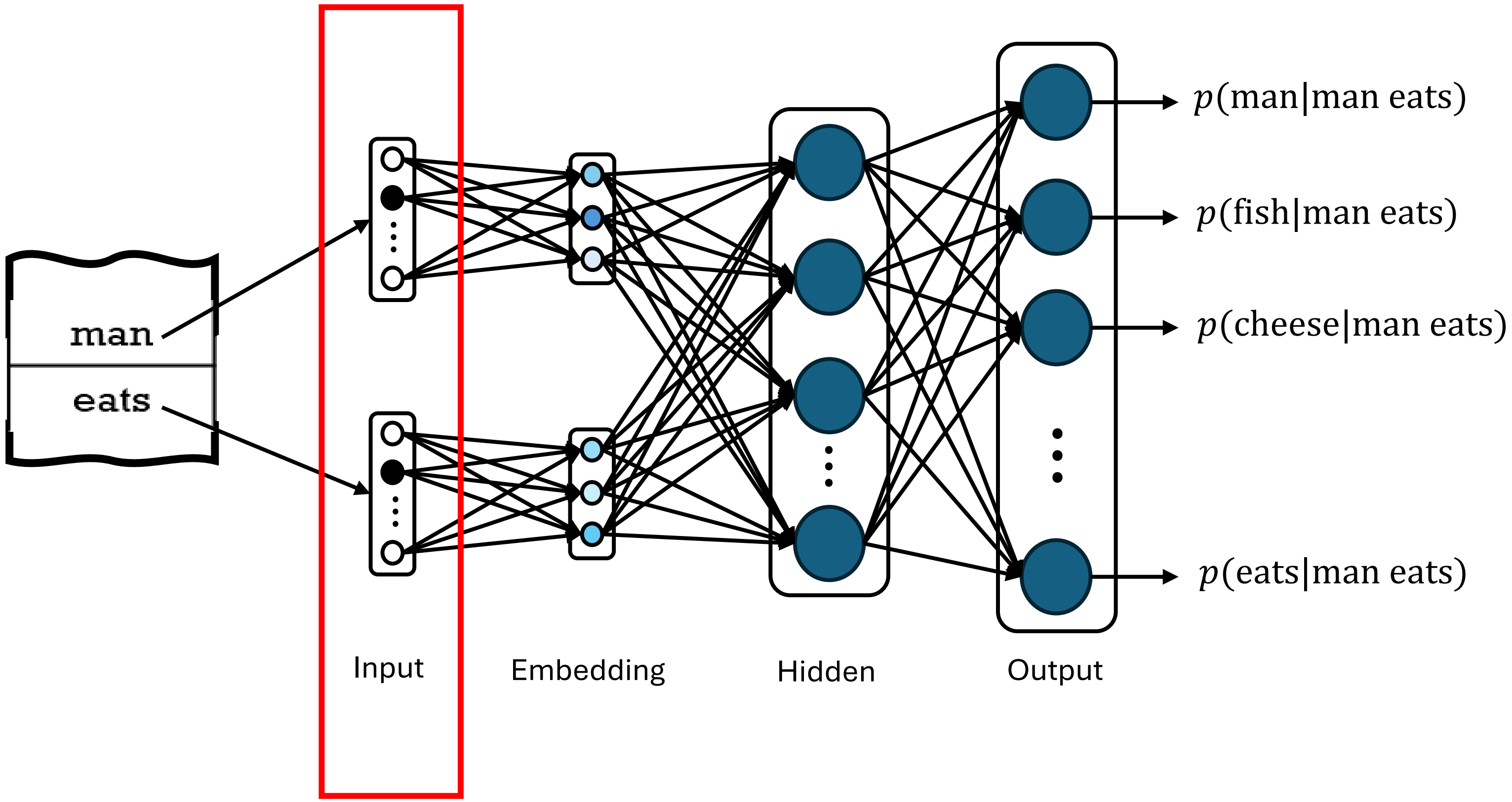


Neural Language Models





Neural Language Models

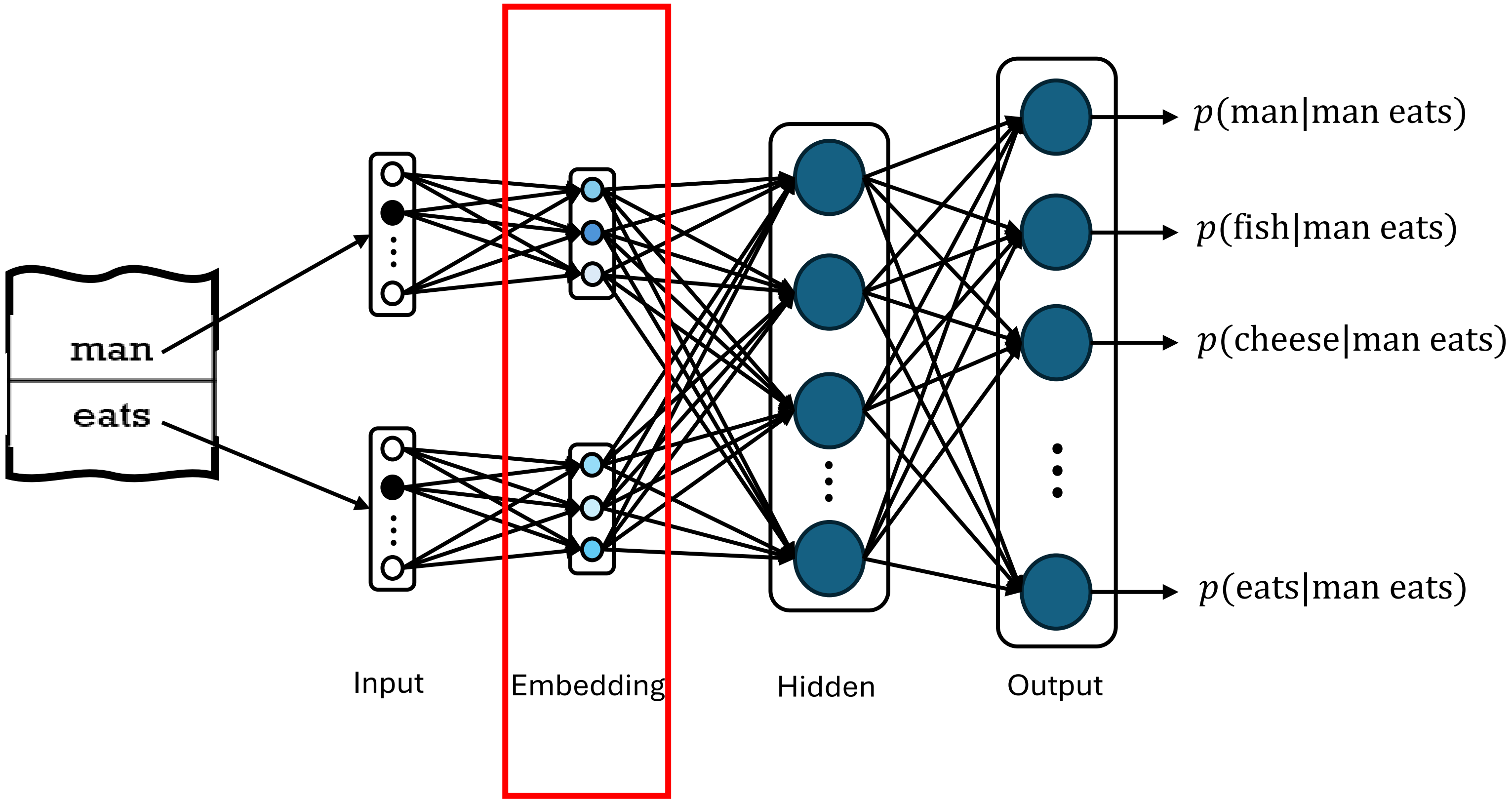


Every token needs to have a unique numeric representation.

- Example: One-hot Vector

Vocabulary	
man	<1,0,0,0>
eats	<0,1,0,0>
cheese	<0,0,1,0>
fish	<0,0,0,1>

Neural Language Models



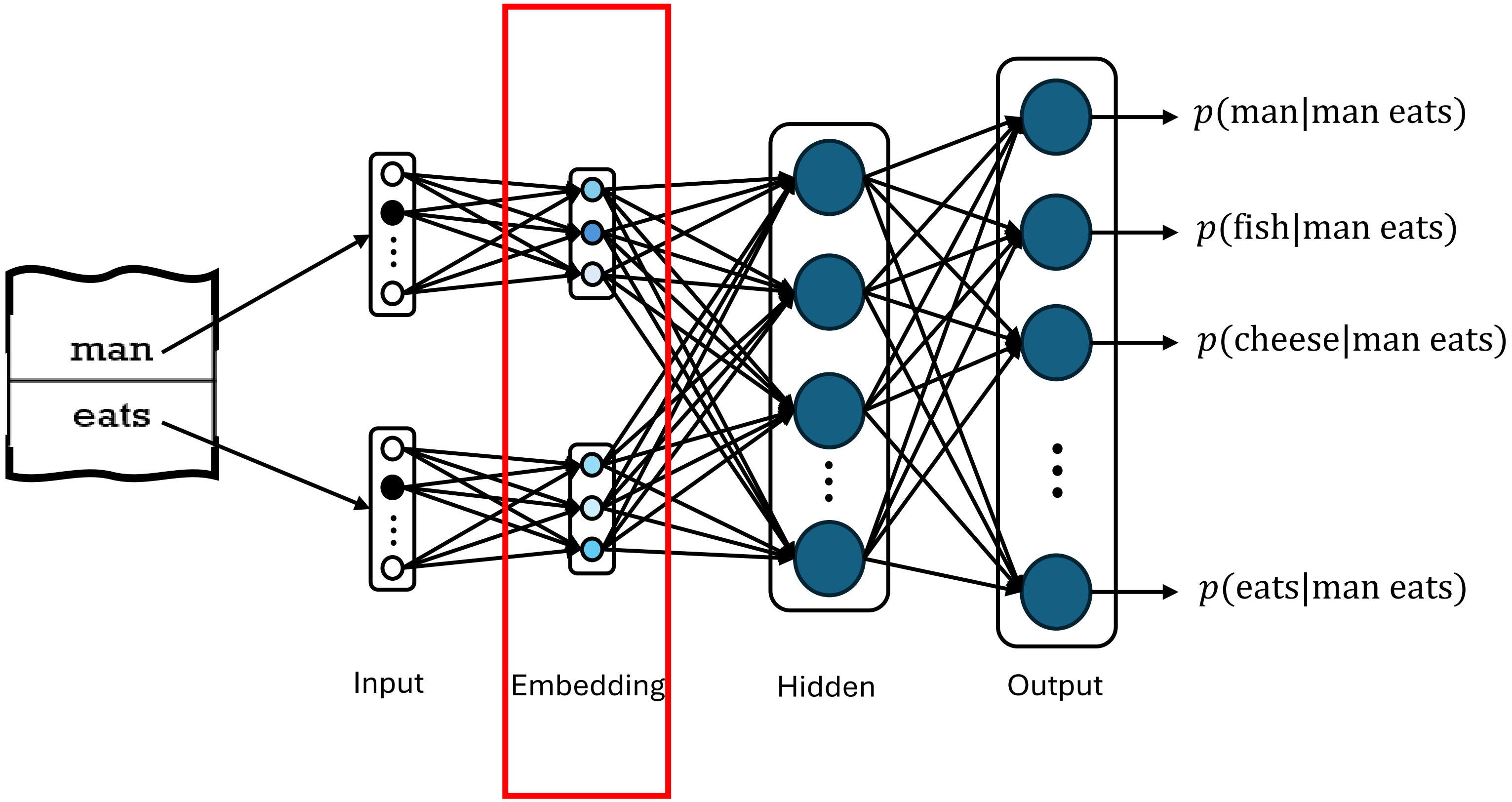
Represent tokens with low dimensional vectors called embeddings

$$\text{man} = \begin{bmatrix} 0.68 \\ 0.12 \\ 0.84 \end{bmatrix}, \quad \text{eats} = \begin{bmatrix} 0.27 \\ 0.91 \\ 0.43 \end{bmatrix}$$

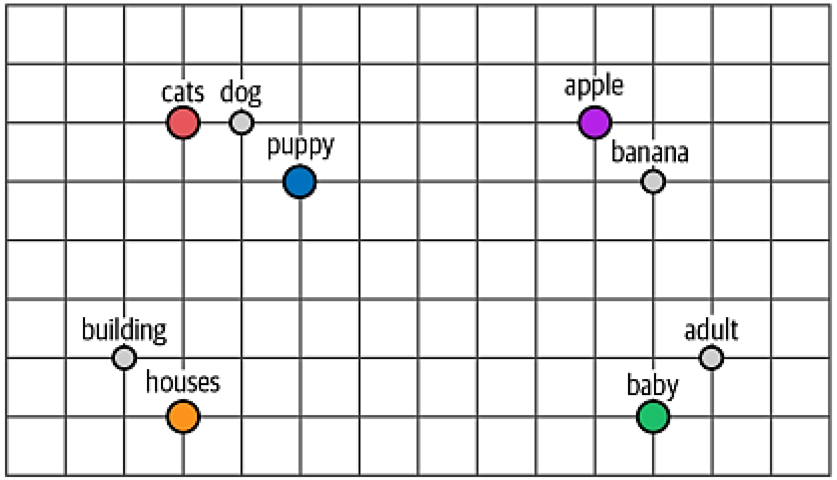
The values of embedding represent properties that are used to represent words.

	cat	puppy	houses	apple	baby
animal	.91	.93	-.56	-.67	.01
newborn	-.11	.71	-.32	-.1	.90
human	.19	.36	-.31	.29	.87
	⋮	⋮	⋮	⋮	⋮
plural	.94	-.82	-.94	-.51	-.11
fruit	-.51	-.91	-.5	.89	-.51

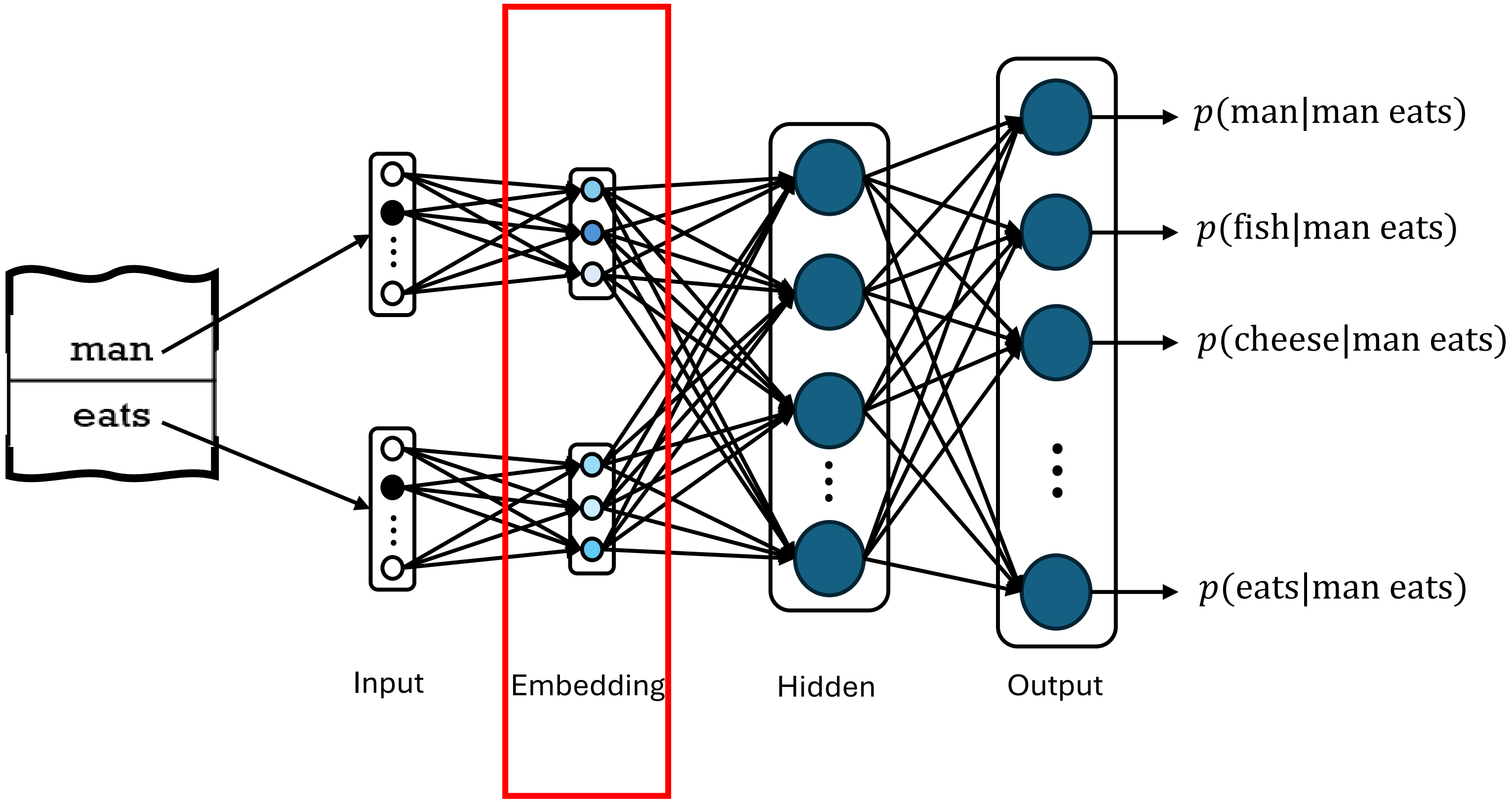
Neural Language Models



- Similar words will have embeddings close to each other in dimensional space



Neural Language Models



Represent tokens with low dimensional vectors called embeddings

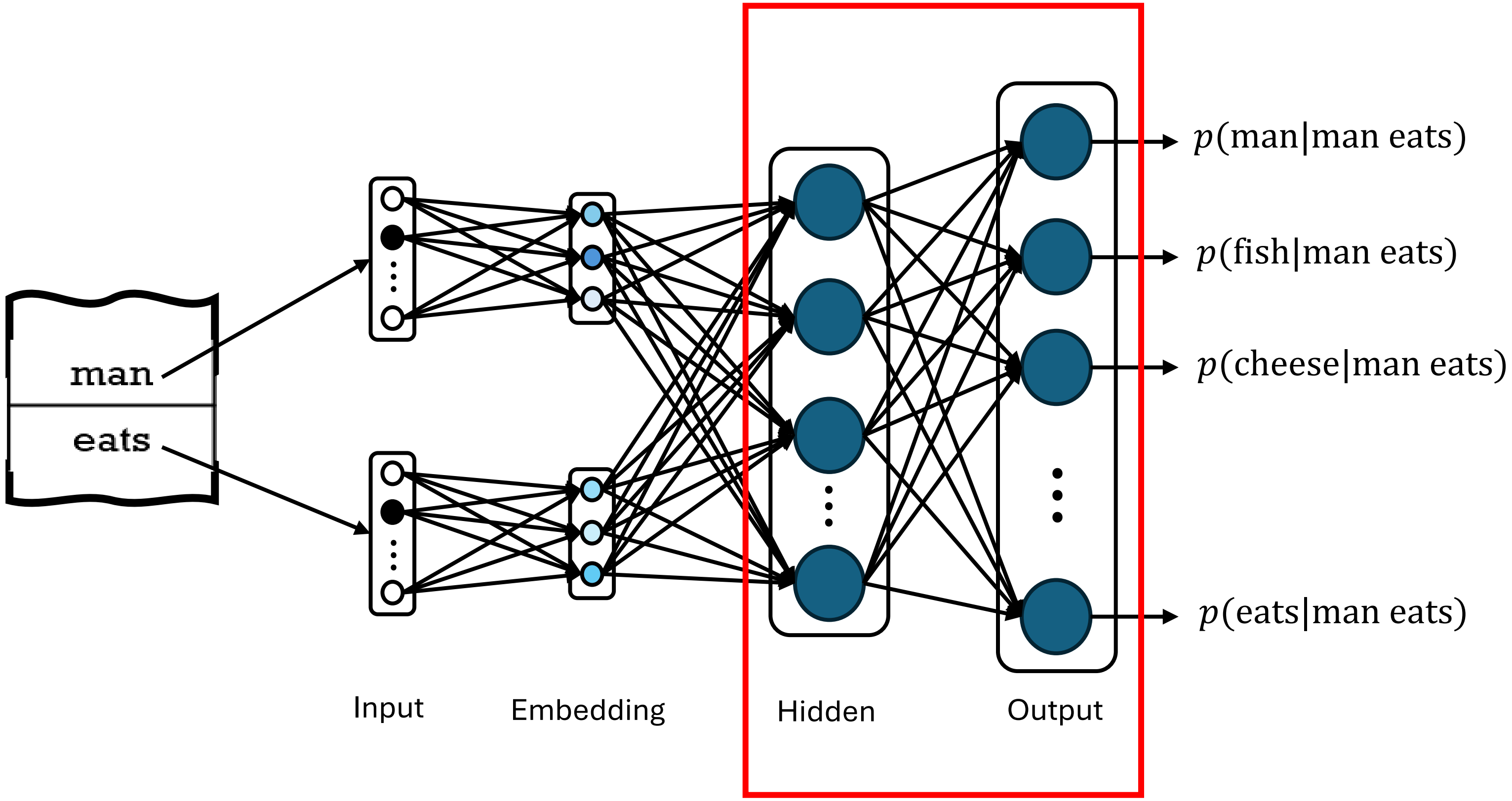
$$\text{man} = \begin{bmatrix} 0.68 \\ 0.12 \\ 0.84 \end{bmatrix}, \quad \text{eats} = \begin{bmatrix} 0.27 \\ 0.91 \\ -0.43 \end{bmatrix}$$

- These embeddings are composed into some low-dimensional vector representing a complete phrase, or sentence.

$$\text{Compose}\left(\begin{bmatrix} 0.68 \\ 0.12 \\ 0.84 \end{bmatrix}, \begin{bmatrix} 0.27 \\ 0.91 \\ -0.43 \end{bmatrix}\right) = \begin{bmatrix} -2.3 \\ 0.9 \\ 5.4 \end{bmatrix}$$

represents “man eats”

Neural Language Models



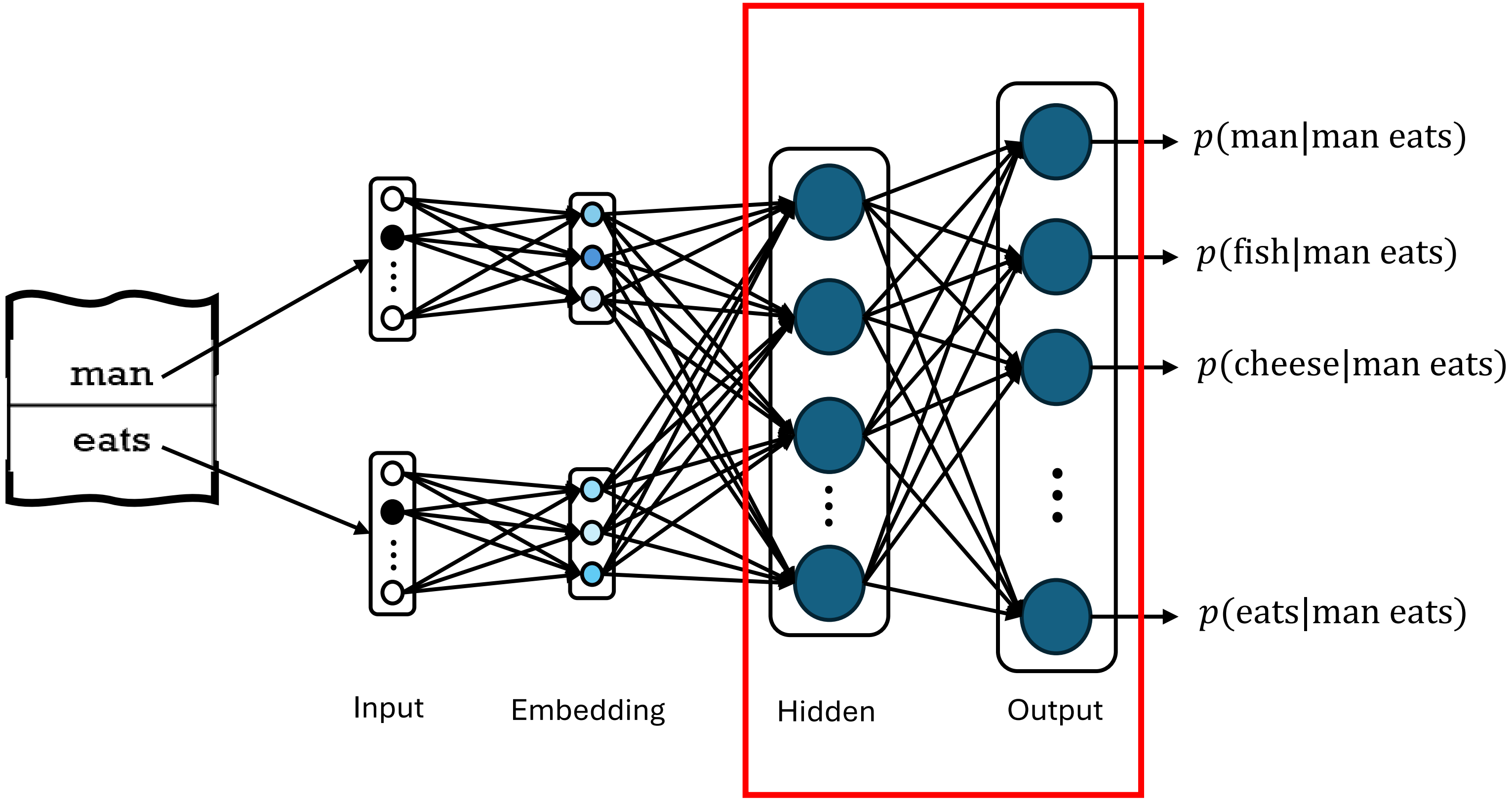
Finding a distribution over V

- Lets say our vocabulary consists of four words:

Vocabulary (V)
cheese
man
fish
eats

- $\text{"man eats"} = \begin{bmatrix} -2.3 \\ 0.9 \\ 5.4 \end{bmatrix}$

Neural Language Models



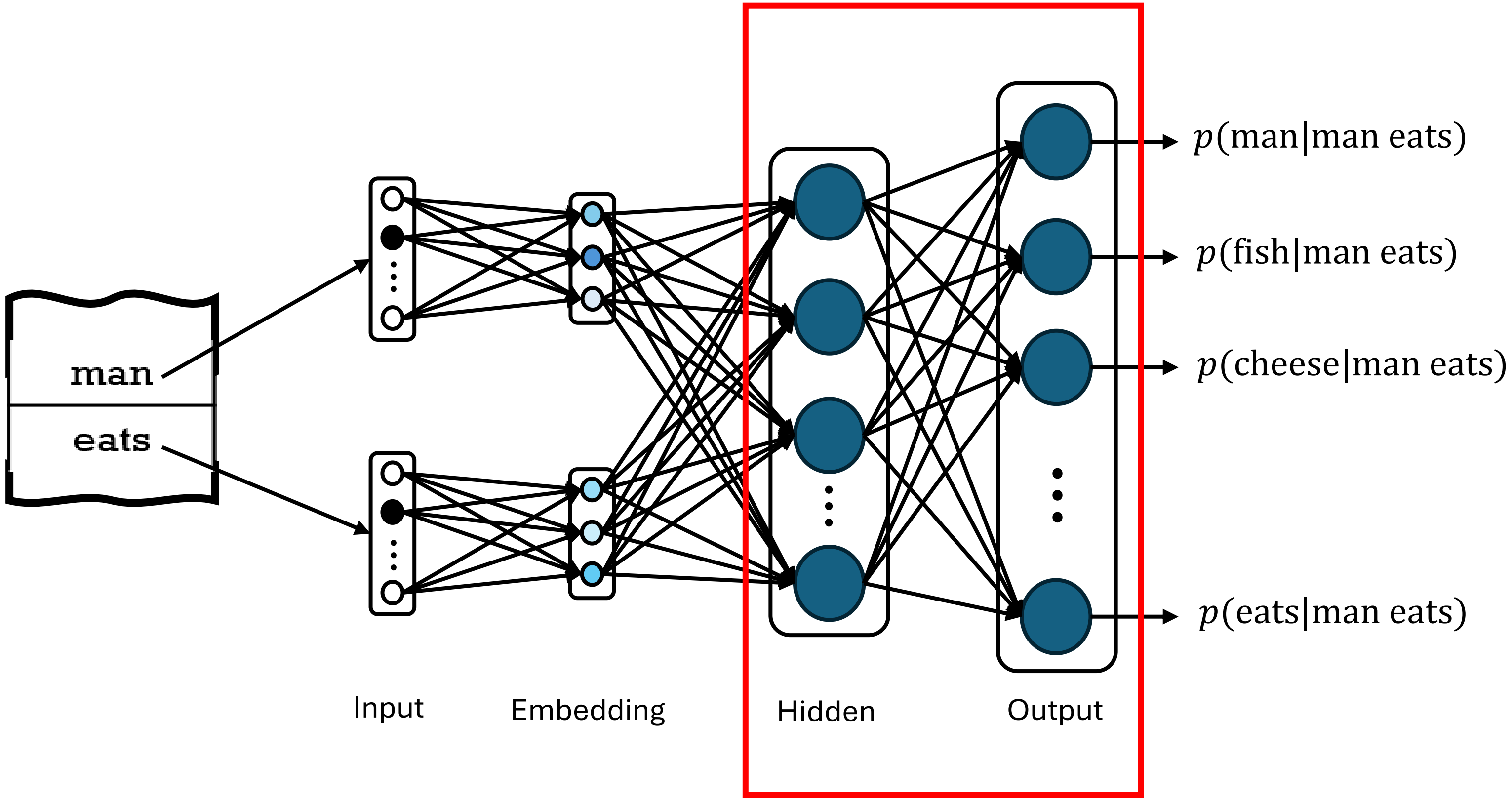
Finding a distribution over V

- W contains feature weights for a corresponding word in the vocabulary

$$W = \begin{array}{cl} \begin{pmatrix} 1.2 & -0.3 & 0.9 \\ 0.2 & 0.4 & -2.2 \\ 8.9 & -1.9 & 6.5 \\ 4.5 & 2.2 & -0.1 \end{pmatrix} & \begin{array}{l} \text{cheese} \\ \text{man} \\ \text{fish} \\ \text{eats} \end{array} \end{array}$$

- $\text{"man eats"} = \begin{bmatrix} -2.3 \\ 0.9 \\ 5.4 \end{bmatrix}$

Neural Language Models



Finding a distribution over V

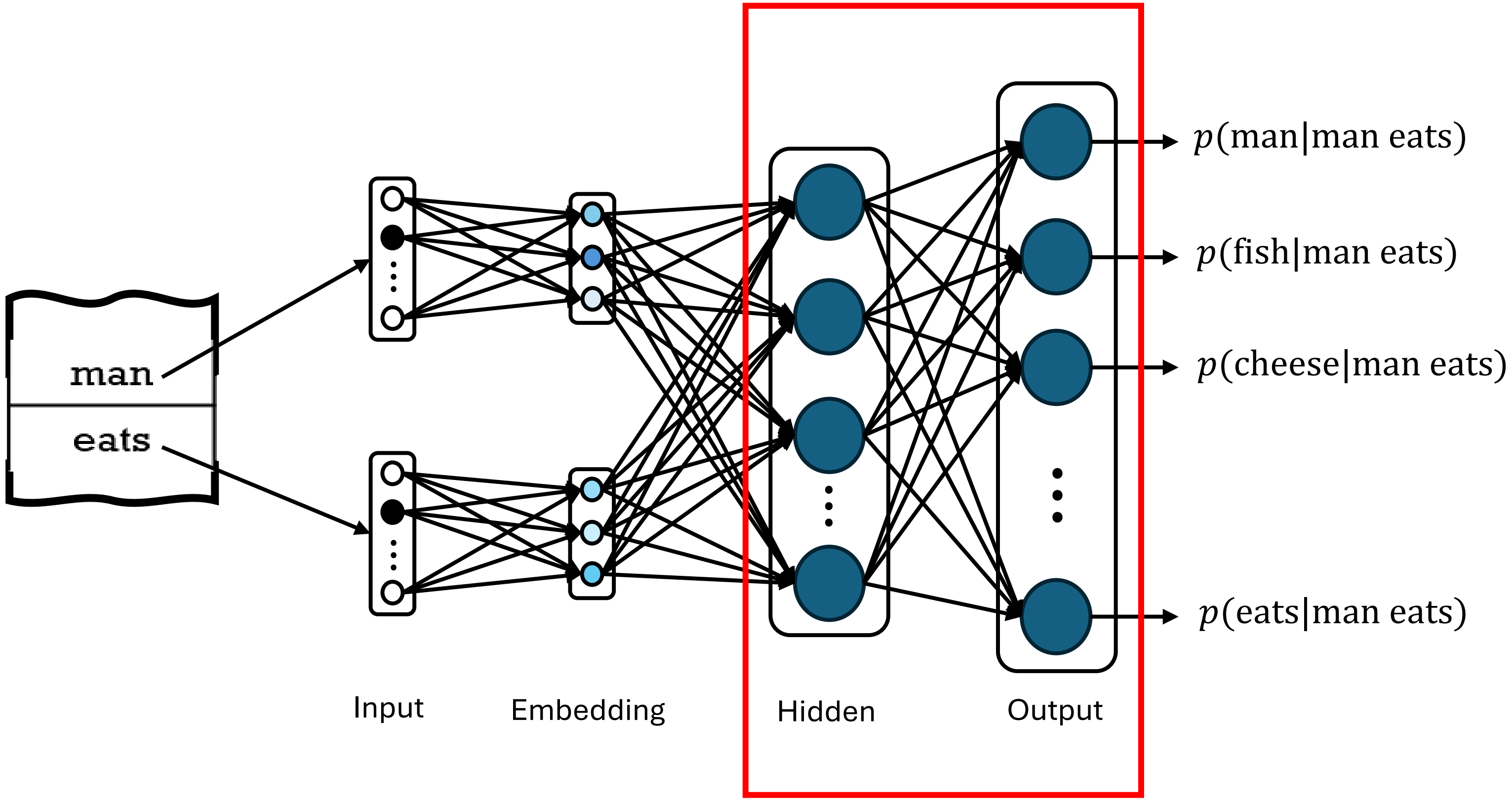
- $\mathbf{W}$ contains feature weights for a corresponding word in the vocabulary

$$\mathbf{W} = \begin{array}{c} \begin{pmatrix} 1.2 & -0.3 & 0.9 \\ 0.2 & 0.4 & -2.2 \\ 8.9 & -1.9 & 6.5 \\ 4.5 & 2.2 & -0.1 \end{pmatrix} \end{array} \begin{array}{l} \text{cheese} \\ \text{man} \\ \text{fish} \\ \text{eats} \end{array}$$

- $\text{"man eats"}(\mathrm{x}) = \begin{bmatrix} -2.3 \\ 0.9 \\ 5.4 \end{bmatrix}$

$$\mathbf{W}\mathrm{x} = \begin{pmatrix} 1.8 \\ -1.9 \\ 2.9 \\ -8.9 \end{pmatrix}$$

Neural Language Models



Finding a distribution over V

- Apply **Softmax** function to get probability distribution

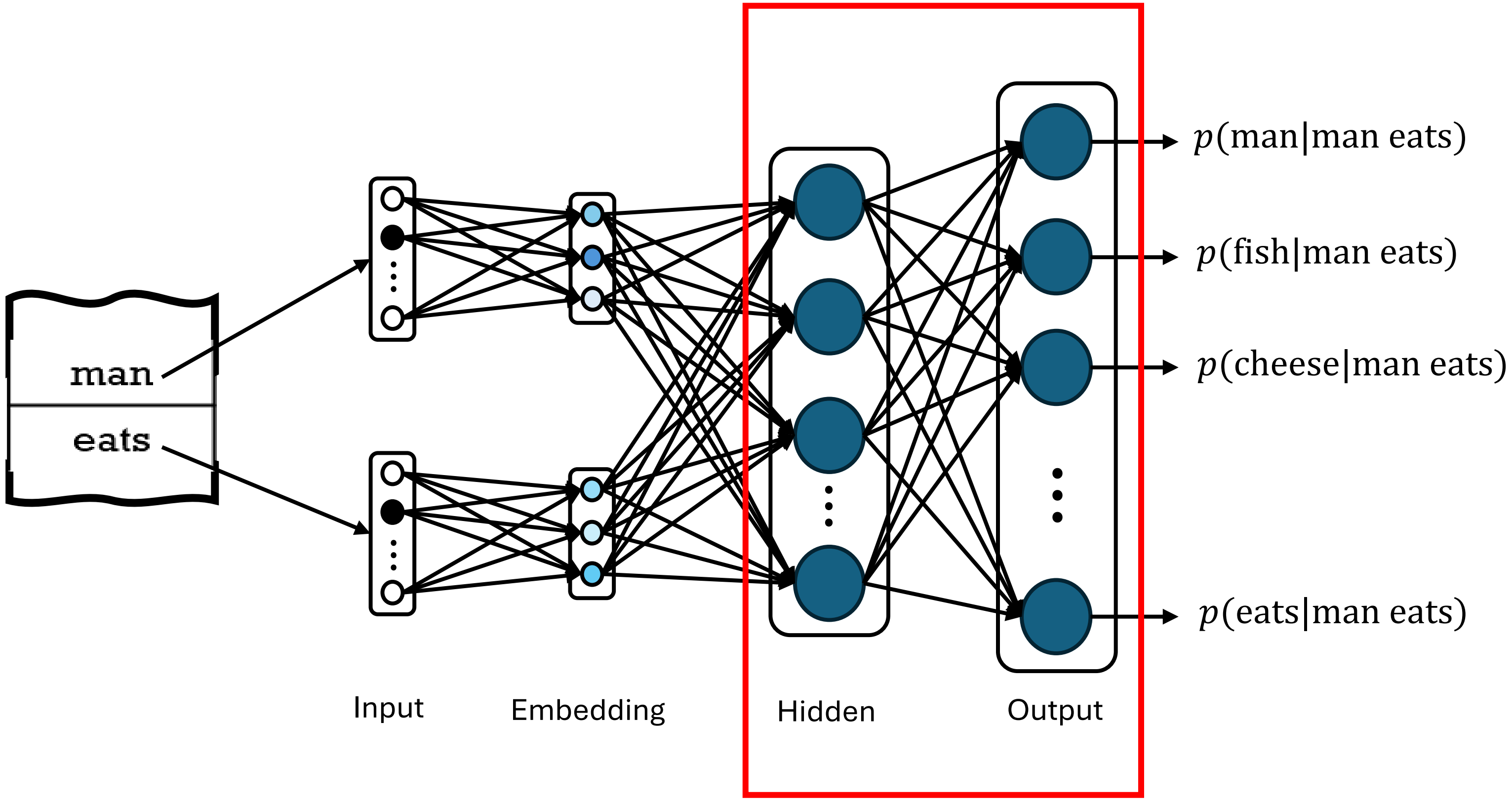
$$\text{softmax(x)} = \frac{e^{x}}{\sum_j e^{x_j}}$$

$$\mathbf{W}\text{x} = \begin{pmatrix} 1.8 \\ -1.9 \\ 2.9 \\ -8.9 \end{pmatrix}$$

$$\text{softmax}(\mathbf{W}\text{x}) = \frac{< e^{1.8}, e^{-1.9}, e^{2.9}, e^{-8.9} >}{e^{1.8} + e^{-1.9} + e^{2.9} + e^{-8.9}}$$

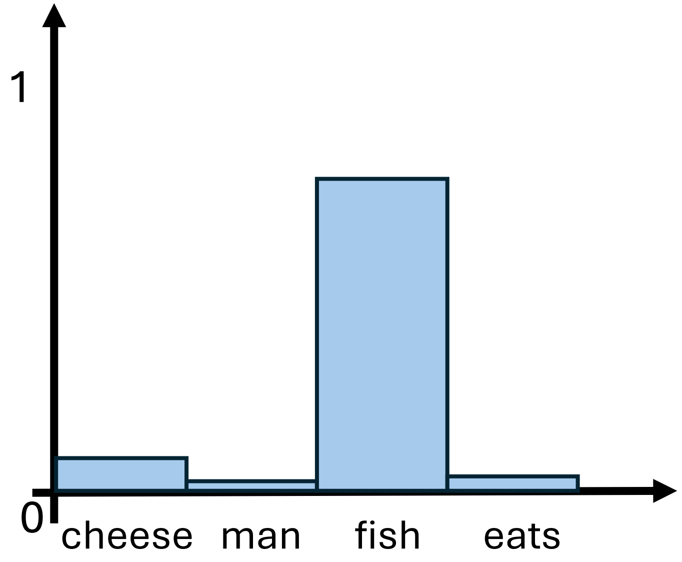
$$\text{softmax}(\mathbf{W}\text{x}) =< 0.24, 0.006, 0.73, 0.02 >$$

Neural Language Models



Finding a distribution over V

$$\text{softmax}(\mathbf{W}\text{x}) = < 0.24, 0.006, 0.73, 0.02 >$$

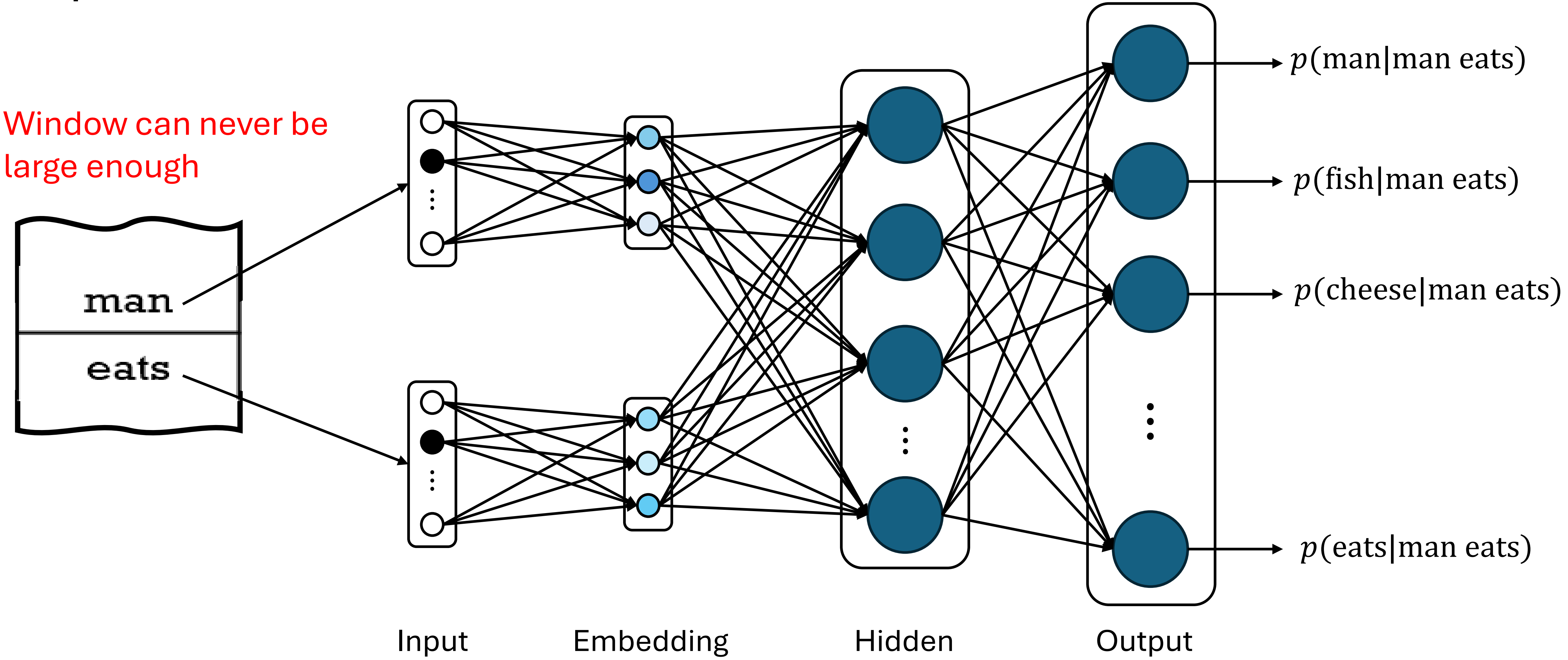


"man eats *fish*"

Neural Language Models

- A problem: Fixed Window

Window can never be large enough



Neural Language Models

- A problem: Fixed Window

  Stanford has a new course on large language models. It will be taught by _____

  taught by ???

- Solution: Attention (Next Lecture)
  - Full-Sequence Access: Attention computes dependencies between all pairs of tokens in the entire input sequence enabling direct modeling of long-range dependencies.