N-Gram Language Models & Neural Networks

Instructor: Lara J. Martin (she/they)

TA: Omkar Kulkarni (he)

https://laramartin.net/NLP-class/

Slides modified from Dr. Frank Ferraro

Learning Objectives

Create a LM using smoothed counts

Define the basic architecture of a neural network

Distinguish between count-based, logistic regression, and neural LMs

Review: Probability Chain Rule

$$p(x_1, x_2, \ldots, x_S) =$$
$$p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1, x_2) \cdots p(x_S \mid x_1, \ldots, x_{S-1}) =$$
$$\prod_{i}^{S} p(x_i \mid x_1, \ldots, x_{i-1})$$

Language modeling is about how to estimate each of these factors in {great, good, sufficient, …} ways

Review: MLE with Trigrams

p(Colorless green ideas sleep furiously) =
p(Colorless | *<BOS> <BOS>*) *
p(green | *<BOS>* Colorless) *
p(ideas | Colorless green) *
p(sleep | green ideas) *
p(furiously | ideas sleep) *
p(*<EOS>* | sleep furiously)

Consistent notation: Pad the left with <BOS> (beginning of sentence) symbols
Fully proper distribution: Pad the right with a single <EOS> symbol

Review: Count-Based N-Grams (Unigrams)

word type word type

$$p(\mathrm{z}) \propto count(\mathrm{z})$$

$$= \frac{count(\mathrm{z})}{\sum_v count(\mathrm{v})}$$

word type

Review: Count-Based N-Grams (Trigrams)

$$p(\mathrm{z}|\mathrm{x},\mathrm{y}) \propto count(\mathrm{x},\mathrm{y},\mathrm{z})$$

$$= \frac{count(\mathrm{x},\mathrm{y},\mathrm{z})}{\sum_{v} count(\mathrm{x},\mathrm{y},\mathrm{v})}$$

For more information, check out Section 3.3 of the SLP textbook
Other resources:
https://en.wikipedia.org/wiki/Perplexity#Perplexity_of_a_probability_model
https://www.geeksforgeeks.org/nlp/perplexity-for-llm-evaluation/

What is perplexity?

$$\text{perplexity} = \exp(\frac{-1}{M} \log p(w_1, \ldots, w_M))$$

$$= \exp(\frac{-1}{M} \sum_{i=1}^{M} \log p(w_i \mid h_i))$$

$$= \exp(\frac{-1}{M} \sum_{i=1}^{M} \log p(w_i \mid w_{i-1}, w_{i-2}, w_{i-3}, \ldots))$$

$$= 2^{H(p)}$$

H(p) is entropy of prediction *p*

Types of Early LMs

Maximum likelihood (MLE): simple counting

Other count-based models

- **Laplace smoothing, add- λ** ← Easy to implement
- Interpolation models
- Discounted backoff
- Interpolated (modified) Kneser-Ney
- Good-Turing
- Witten-Bell

(Interpolation models through Witten-Bell: Advanced/ out of scope)

Maxent n-gram models ← Featureful LMs

Neural n-gram models ← Feedforward LMs

Recurrent/autoregressive NNs ← Precursor to modern LMs

Add-λ estimation

Other names: Laplace smoothing, Lidstone smoothing

Pretend we saw each word λ more times than we did

$$p(\mathrm{z}) \propto count(\mathrm{z}) + \lambda$$

Add λ to all the counts

Add-λ estimation

Other names: Laplace smoothing, Lidstone smoothing

Pretend we saw each word λ more times than we did

Add λ to all the counts

$$p(\mathrm{z}) \propto count(\mathrm{z}) + \lambda$$

$$= \frac{count(\mathrm{z}) + \lambda}{\sum_v (count(\mathrm{v}) + \lambda)}$$

Add-λ estimation

Other names: Laplace smoothing, Lidstone smoothing

Pretend we saw each word λ more times than we did

Add λ to all the counts

$$p(\mathrm{z}) \propto count(\mathrm{z}) + \lambda$$

$$= \frac{count(\mathrm{z}) + \lambda}{W + V\lambda}$$

W: # tokens

V: # types

Add-λ N-Grams (Unigrams)

The film got a great opening and the film went on to become a hit .

Word (Type)	Raw Count	Norm	Prob.	Add-λ Count	Add-λ Norm.	Add-λ Prob.
The	1	16	1/16			
film	2		1/8			
got	1		1/16			
a	2		1/8			
great	1		1/16			
opening	1		1/16			
and	1		1/16			
the	1		1/16			
went	1		1/16			
on	1		1/16			
to	1		1/16			
become	1		1/16			
hit	1		1/16			
.	1		1/16			

Add-1 N-Grams (Unigrams)

The film got a great opening and the film went on to become a hit .

Word (Type)	Raw Count	Norm	Prob.	Add-1 Count	Add-1 Norm.	Add-1 Prob.
The	1	16	1/16	2		
film	2		1/8	3		
got	1		1/16	2		
a	2		1/8	3		
great	1		1/16	2		
opening	1		1/16	2		
and	1		1/16	2		
the	1		1/16	2		
went	1		1/16	2		
on	1		1/16	2		
to	1		1/16	2		
become	1		1/16	2		
hit	1		1/16	2		
.	1		1/16	2		

Add-1 N-Grams (Unigrams)

The film got a great opening and the film went on to become a hit .

Word (Type)	Raw Count	Norm	Prob.	Add-1 Count	Add-1 Norm.	Add-1 Prob.
The	1	16	1/16	2	16 + 14*1 = 30	
film	2		1/8	3		
got	1		1/16	2		
a	2		1/8	3		
great	1		1/16	2		
opening	1		1/16	2		
and	1		1/16	2		
the	1		1/16	2		
went	1		1/16	2		
on	1		1/16	2		
to	1		1/16	2		
become	1		1/16	2		
hit	1		1/16	2		
.	1		1/16	2		

Add-1 N-Grams (Unigrams)

The film got a great opening and the film went on to become a hit .

Word (Type)	Raw Count	Norm	Prob.	Add-1 Count	Add-1 Norm.	Add-1 Prob.
The	1	16	1/16	2	16 + 14*1 = 30	=1/15
film	2		1/8	3		=1/10
got	1		1/16	2		=1/15
a	2		1/8	3		=1/10
great	1		1/16	2		=1/15
opening	1		1/16	2		=1/15
and	1		1/16	2		=1/15
the	1		1/16	2		=1/15
went	1		1/16	2		=1/15
on	1		1/16	2		=1/15
to	1		1/16	2		=1/15
become	1		1/16	2		=1/15
hit	1		1/16	2		=1/15
.	1		1/16	2		=1/15

An Extended Trigram Example

The film got a great opening and the film went on to become a hit .

Q: With OOV, EOS, and BOS, how many types (for normalization)?

Context: x y	Word (Type): z	Raw Count	Add-1 count	Norm.	Probability p(z | x y)
The film	The	0			
The film	film	0			
The film	got	1			
The film	went	0			
The film	OOV	0			
The film	EOS	0			
		…			
a great	great	0			
a great	opening	1			
a great	and	0			
a great	the	0			
		…			

An Extended Trigram Example

The film got a great opening and the film went on to become a hit .

Q: With OOV, EOS, and BOS, how many types (for normalization)?

A: 16
(why don't we count BOS?)

Context: x y	Word (Type): z	Raw Count	Add-1 count	Norm.	Probability p(z | x y)
The film	The	0			
The film	film	0			
The film	got	1			
The film	went	0			
The film	OOV	0			
The film	EOS	0			
		…			
a great	great	0			
a great	opening	1			
a great	and	0			
a great	the	0			
		…			

An Extended Trigram Example

The film got a great opening and the film went on to become a hit .

Q: With OOV, EOS, and BOS, how many types (for normalization)?

A: 16
(why don't we count BOS?)

Context: x y	Word (Type): z	Raw Count	Add-1 count	Norm.	Probability p(z | x y)
The film	The	0	1		1/17
The film	film	0	1		1/17
The film	got	1	2		2/17
The film	went	0	1	17 (=1+16*1)	1/17
	…				…
The film	OOV	0	1		1/17
The film	EOS	0	1		1/17
			…		
a great	great	0	1		1/17
a great	opening	1	2	17	2/17
a great	and	0	1		1/17
a great	the	0	1		1/17
			…		

An Extended Trigram Example

The film got a great opening and the film went on to become a hit .

Q: With OOV, EOS, and BOS, how many types (for normalization)?

A: 16
(why don't we count BOS?)

Only one "The film *" trigram in the dataset

Context: x y	Word (Type): z	Raw Count	Add-1 count	Norm.	Probability p(z | x y)
The film	The	0	1	17 (=1+16*1)	1/17
The film	film	0	1		1/17
The film	got	1	2		2/17
The film	went	0	1		1/17
…					…
The film	OOV	0	1		1/17
The film	EOS	0	1		1/17
…					
a great	great	0	1	17	1/17
a great	opening	1	2		2/17
a great	and	0	1		1/17
a great	the	0	1		1/17
…					

Review: Calculating perplexity for our trigram model from slide 55

Trigrams	MLE p(trigram)	Smoothed p(trigram)
<BOS> <BOS> The	1	2/17
<BOS> The film	1	2/17
The film ,	0	1/17
film , a	0	1/16
, a hit	0	1/16
a hit !	0	1/17
hit ! <EOS>	0	1/16
Perplexity	Infinity	13.59

"The film , a hit !"

$$\text{perplexity} = \exp\left(\frac{-1}{M}\sum_{i=1}^{M}\log p(w_i \mid h_i)\right)$$

Adding <UNK> to trigrams

Trigrams	MLE p(trigram)
<BOS> <BOS> The	1
<BOS> The film	1
The film ,	0
film , a	0
, a hit	0
a hit !	0
hit ! <EOS>	0

Adding <UNK> to trigrams

Trigrams	MLE p(trigram)	UNK-ed trigrams
<BOS> <BOS> The	1	<BOS> <BOS> The
<BOS> The film	1	<BOS> The film
The film ,	0	The film <UNK>
film , a	0	film <UNK> a
, a hit	0	<UNK> a hit
a hit !	0	a hit <UNK>
hit ! <EOS>	0	hit <UNK> <EOS>

Adding <UNK> to trigrams

Trigrams	MLE p(trigram)	UNK-ed trigrams	Smoothed p(trigram)
<BOS> <BOS> The	1	<BOS> <BOS> The	2/17
<BOS> The film	1	<BOS> The film	2/17
The film ,	0	The film <UNK>	1/17
film , a	0	film <UNK> a	1/16
, a hit	0	<UNK> a hit	1/16
a hit !	0	a hit <UNK>	1/17
hit ! <EOS>	0	hit <UNK> <EOS>	1/16

Types of Early LMs

Maximum likelihood (MLE): simple counting

Other count-based models

- Laplace smoothing, add- λ ← Easy to implement
- Interpolation models
- Discounted backoff
- Interpolated (modified) Kneser-Ney
- Good-Turing
- Witten-Bell

(Interpolation models through Witten-Bell: Advanced/ out of scope)

Maxent n-gram models ← Featureful LMs

Neural n-gram models ← Feedforward LMs

Recurrent/autoregressive NNs ← Precursor to modern LMs

Review: LR/Maxent Equation

$$p(Y = y \mid x) = \frac{\exp(\theta_y^T f(x))}{\sum_{y'} \exp(\theta_{y'}^T f(x))}$$

$$p(Y = y \mid x) \propto \exp(\theta_y^T f(x))$$

$$p(Y \mid x) = \mathrm{softmax}(\theta f(x))$$

Maxent/LR Models as Featureful n-gram Language Models

p(Colorless green ideas sleep furiously | Label) =
p(Colorless | Label, <BOS>) * … * p(<EOS> | Label , furiously)

Chain rule

The label from our classification problem (e.g., entailed/not entailed) is now on this side of the conditional because we're interested in generating the text, not predicting the label

Model each n-gram term with a maxent model

$$p(x_i \mid y, x_{i-N+1:i-1}) = \mathrm{maxent}(y, x_{i-N+1:i-1}, x_i)$$

generatively trained:
learn to model (*class-specific*) *language*

Language Model with Maxent n-grams

$$p_n(🗎|y) = \prod_{i=1}^{M} \text{maxent}(y, x_{i-n+1:i-1}, x_i)$$

label

n-gram

$$= \prod_{i=1}^{M} \frac{\exp(\theta_{x_i}^T f(y, x_{i-n+1:i-1}))}{\sum_{x'} \exp(\theta_{x'}^T f(y, x_{i-n+1:i-1}))}$$

Iterate through all possible output vocab types x'---just like in count-based LMs

What Should These Features Do?

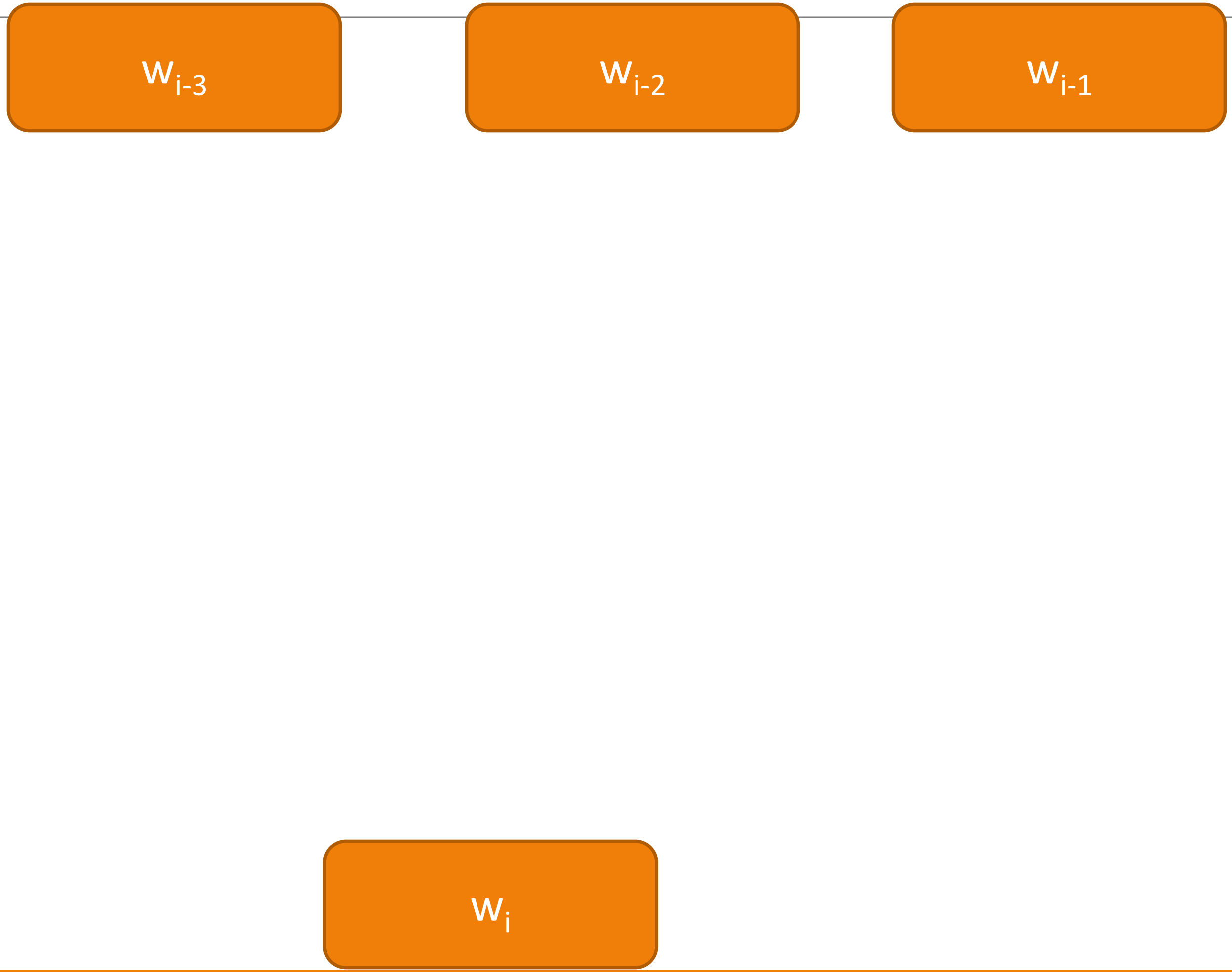
$p(x_i \mid y, x_{i-N+1:i-1}) = \text{maxent}(y, x_{i-N+1:i-1}, x_i)$, e.g.,

$$p(\text{sleep} \mid y, \text{green}, \text{ideas}) = \\ \text{maxent}\big(y, x_{i-2,i-1} = (\text{green}, \text{ideas}), x_i = \text{sleep}\big) \\ \propto \exp(\theta_{x_i=\text{sleep}}{}^T f(y, x_{i-2,i-1} = (\text{green}, \text{ideas})))$$

(in-class discussion)

N-gram Language Models

given some context…



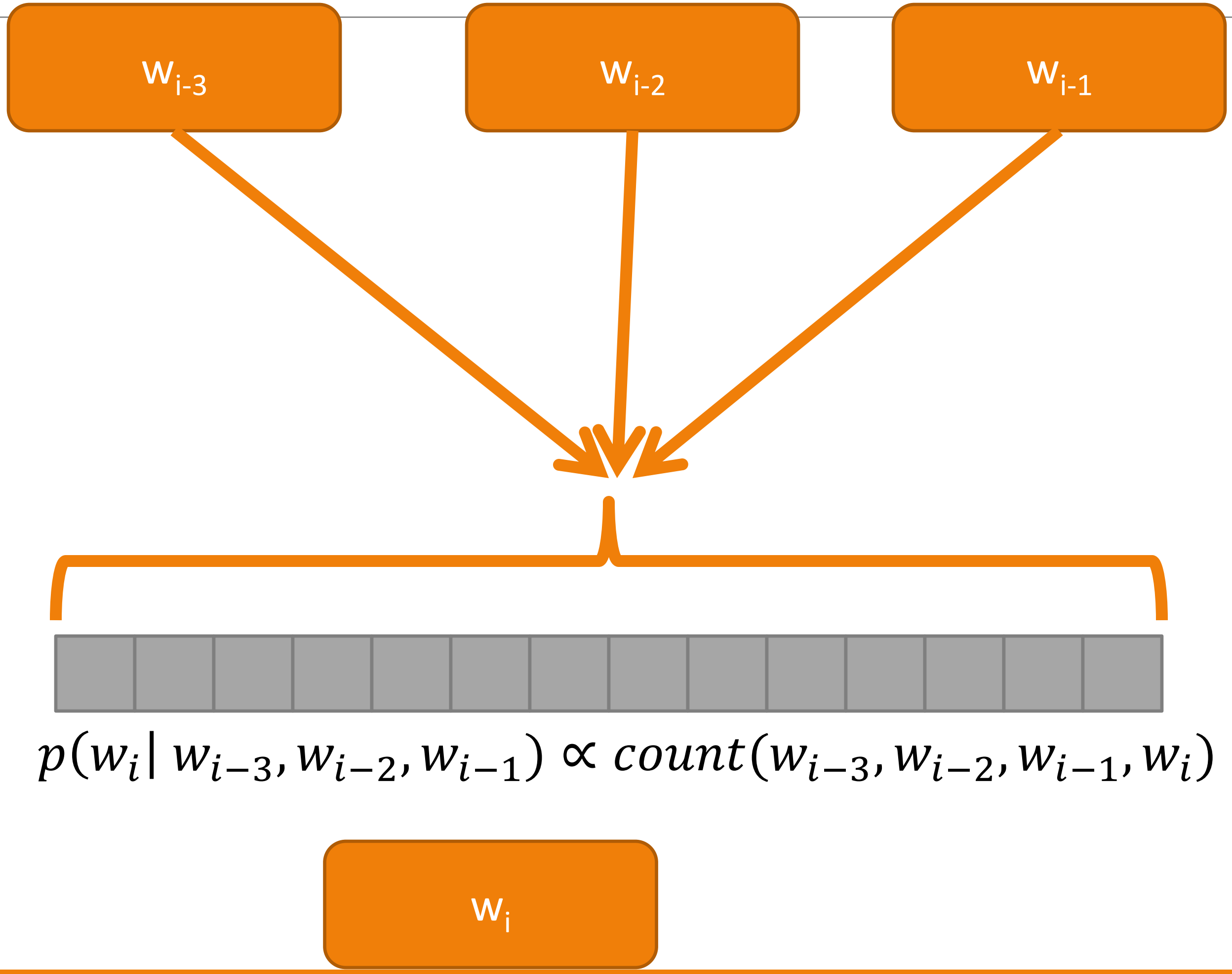
predict the next word

N-gram Language Models

given some context…

compute beliefs about what is likely…

predict the next word



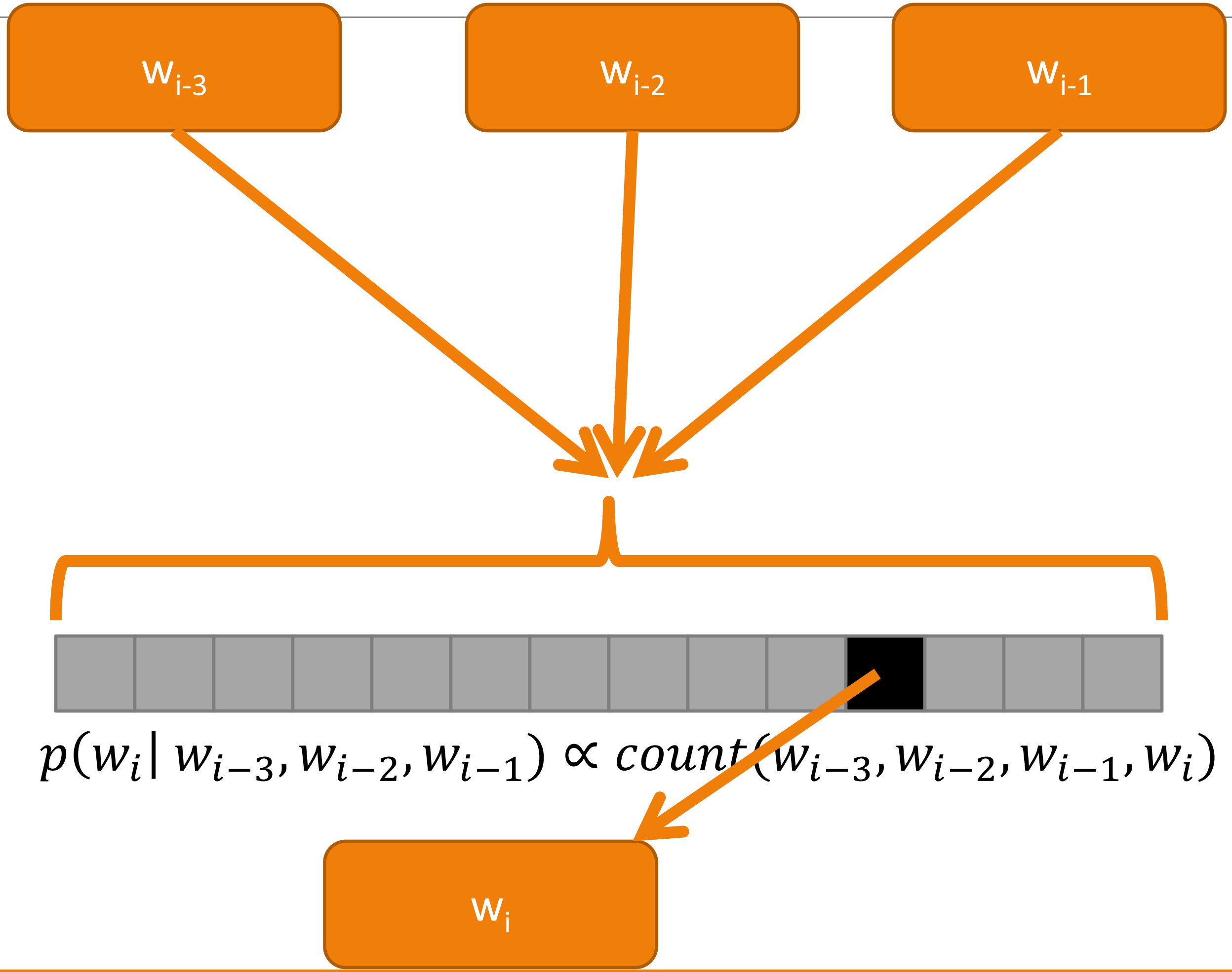
$$p(w_i |\, w_{i-3}, w_{i-2}, w_{i-1}) \propto count(w_{i-3}, w_{i-2}, w_{i-1}, w_i)$$

N-gram Language Models

given some context…

compute beliefs about what is likely…

predict the next word



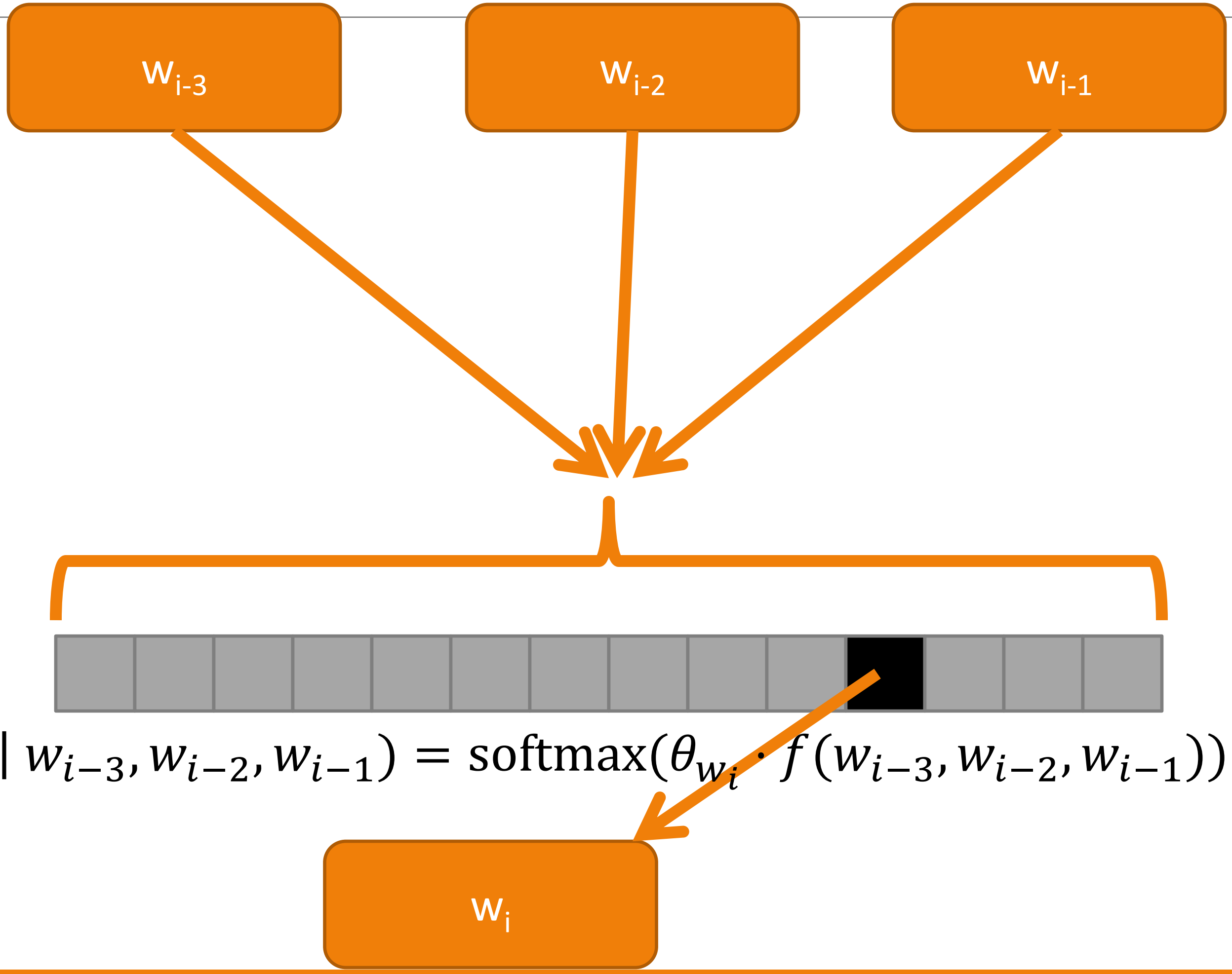
$$p(w_i | w_{i-3}, w_{i-2}, w_{i-1}) \propto count(w_{i-3}, w_{i-2}, w_{i-1}, w_i)$$

Maxent/LR Language Models

given some context…

compute beliefs about what is likely…

predict the next word



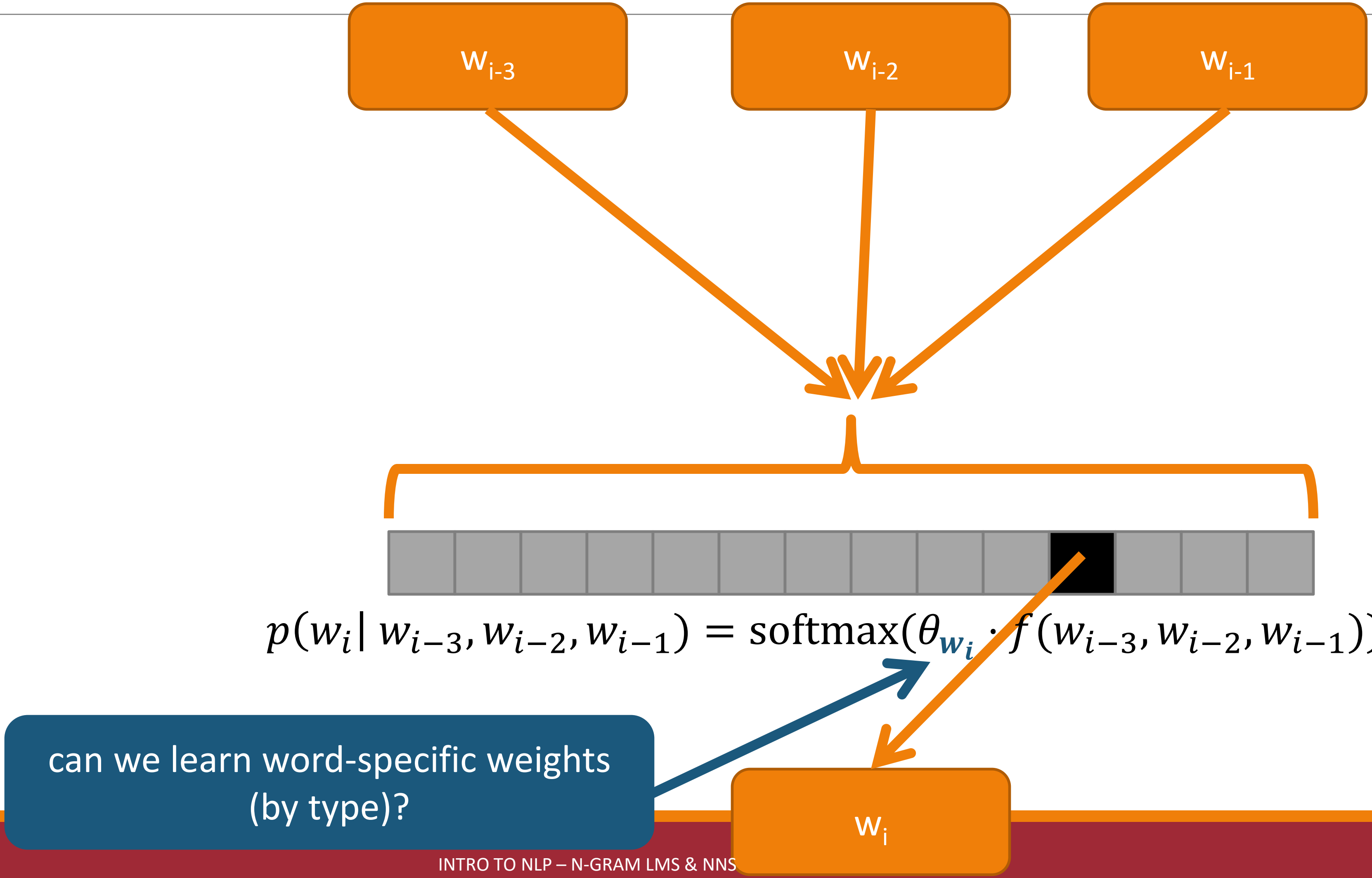
$$p(w_i | w_{i-3}, w_{i-2}, w_{i-1}) = \text{softmax}(\theta_{w_i} \cdot f(w_{i-3}, w_{i-2}, w_{i-1}))$$

Maxent/LR Language Models

given some context…

compute beliefs about what is likely…

predict the next word



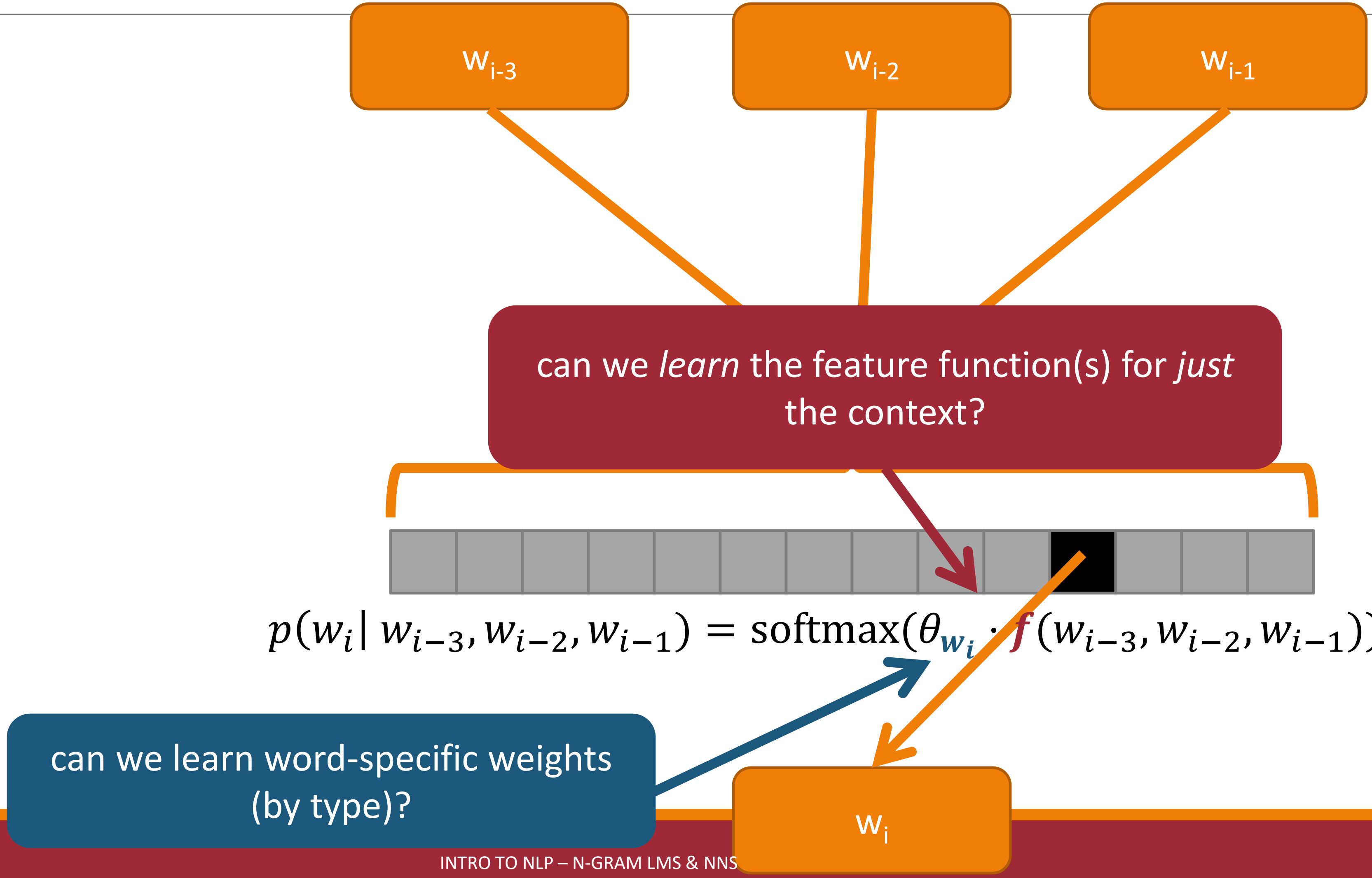
$$p(w_i \mid w_{i-3}, w_{i-2}, w_{i-1}) = \text{softmax}(\theta_{\boldsymbol{w_i}} \cdot f(w_{i-3}, w_{i-2}, w_{i-1}))$$

Neural Language Models

given some context…

compute beliefs about what is likely…

predict the next word



$$p(w_i | w_{i-3}, w_{i-2}, w_{i-1}) = \text{softmax}(\theta_{w_i} \cdot f(w_{i-3}, w_{i-2}, w_{i-1}))$$

Types of Early LMs

Maximum likelihood (MLE): simple counting

Other count-based models

- Laplace smoothing, add- λ ← Easy to implement
- Interpolation models
- Discounted backoff
- Interpolated (modified) Kneser-Ney
- Good-Turing
- Witten-Bell

(Interpolation models through Witten-Bell: Advanced/ out of scope)

Maxent n-gram models ← Featureful LMs

Neural n-gram models ← Feedforward LMs

Recurrent/autoregressive NNs ← Precursor to modern LMs

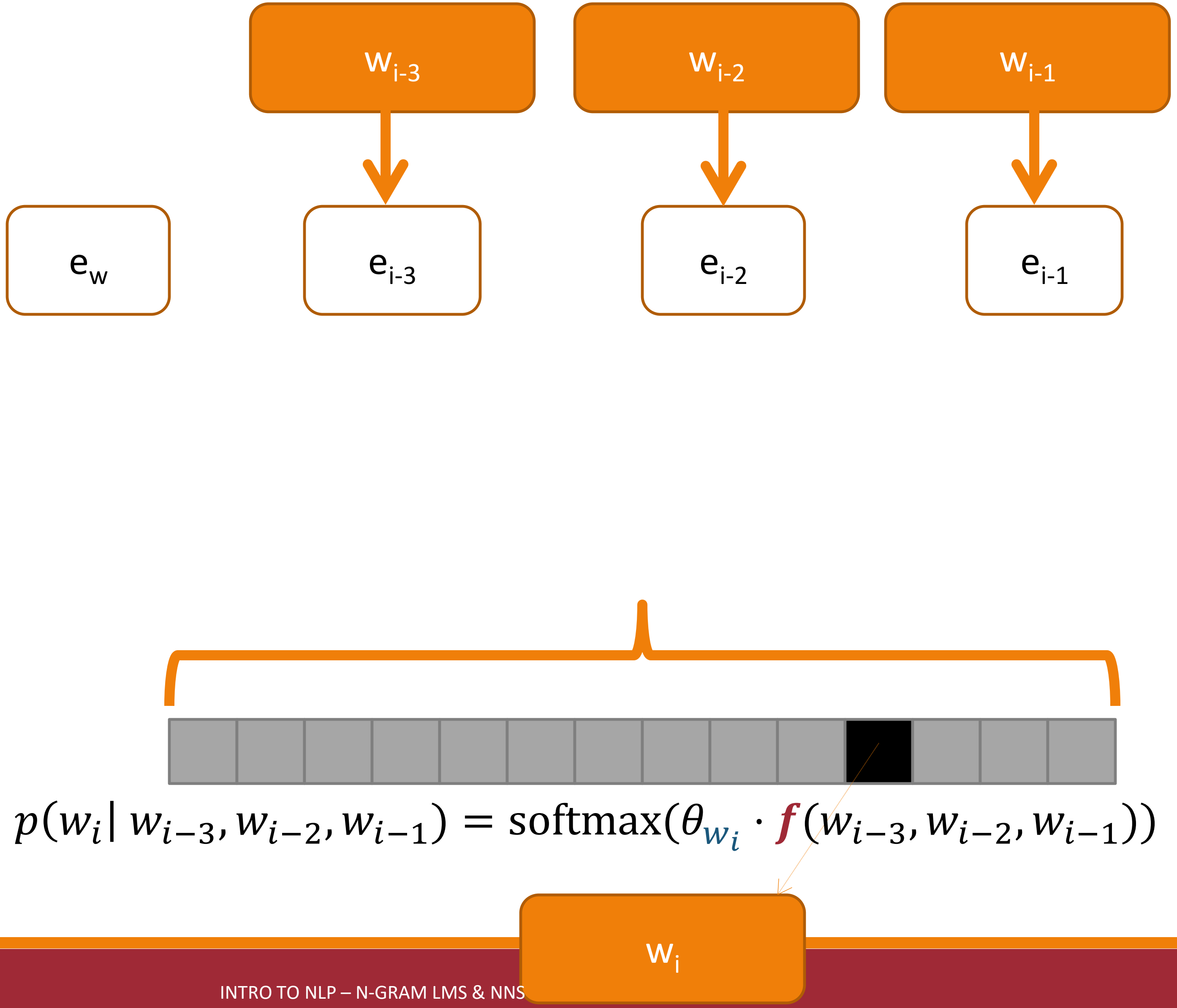
Neural Language Models

given some context…

create/use “distributed representations”…

compute beliefs about what is likely…

predict the next word



$$p(w_i | w_{i-3}, w_{i-2}, w_{i-1}) = \text{softmax}(\theta_{w_i} \cdot \boldsymbol{f}(w_{i-3}, w_{i-2}, w_{i-1}))$$

Neural Language Models

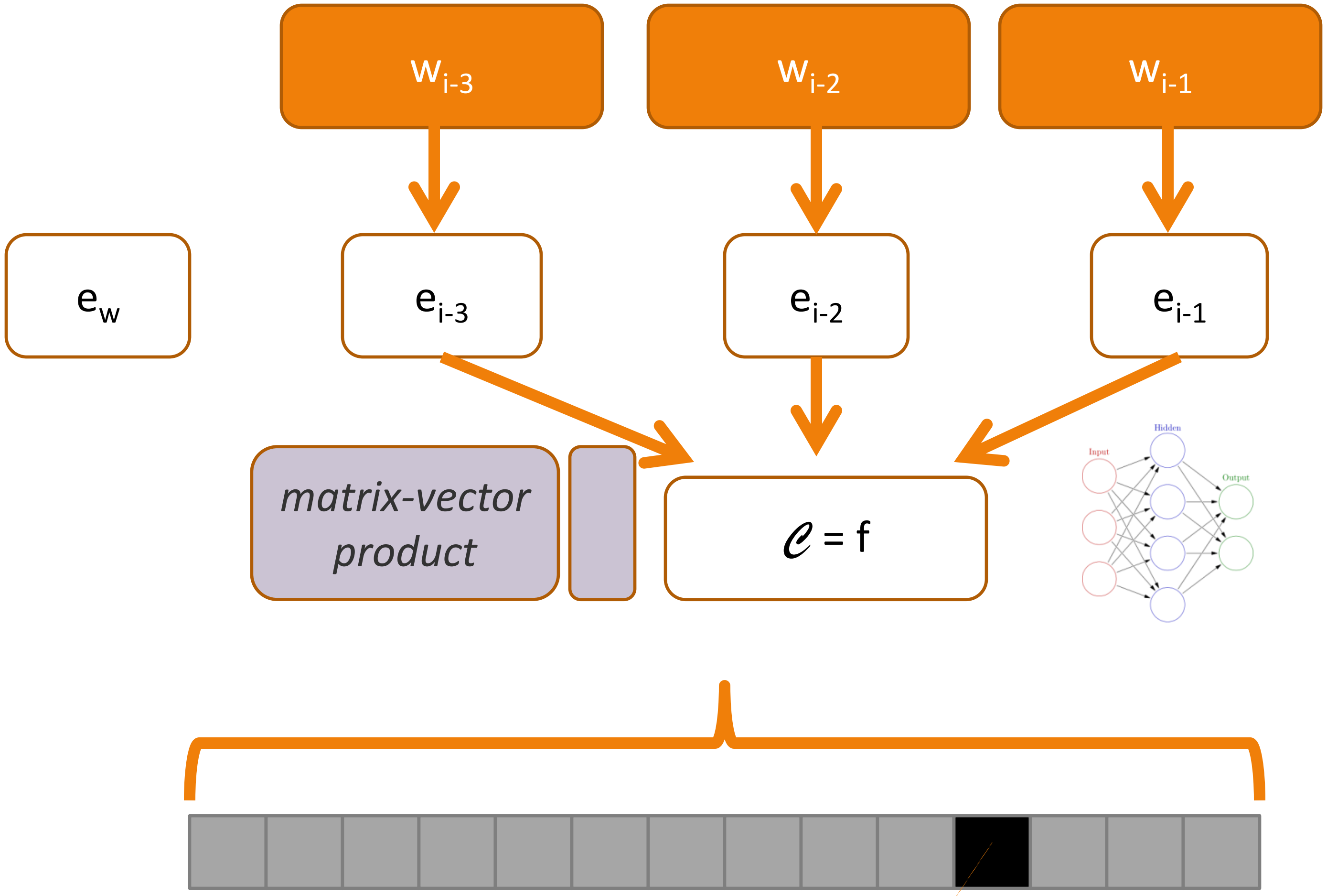
given some context…

create/use "distributed representations"…

combine these representations…

compute beliefs about what is likely…

predict the next word



$$p(w_i | w_{i-3}, w_{i-2}, w_{i-1}) = \text{softmax}(\theta_{w_i} \cdot \boldsymbol{f}(w_{i-3}, w_{i-2}, w_{i-1}))$$

w_i

Neural Language Models

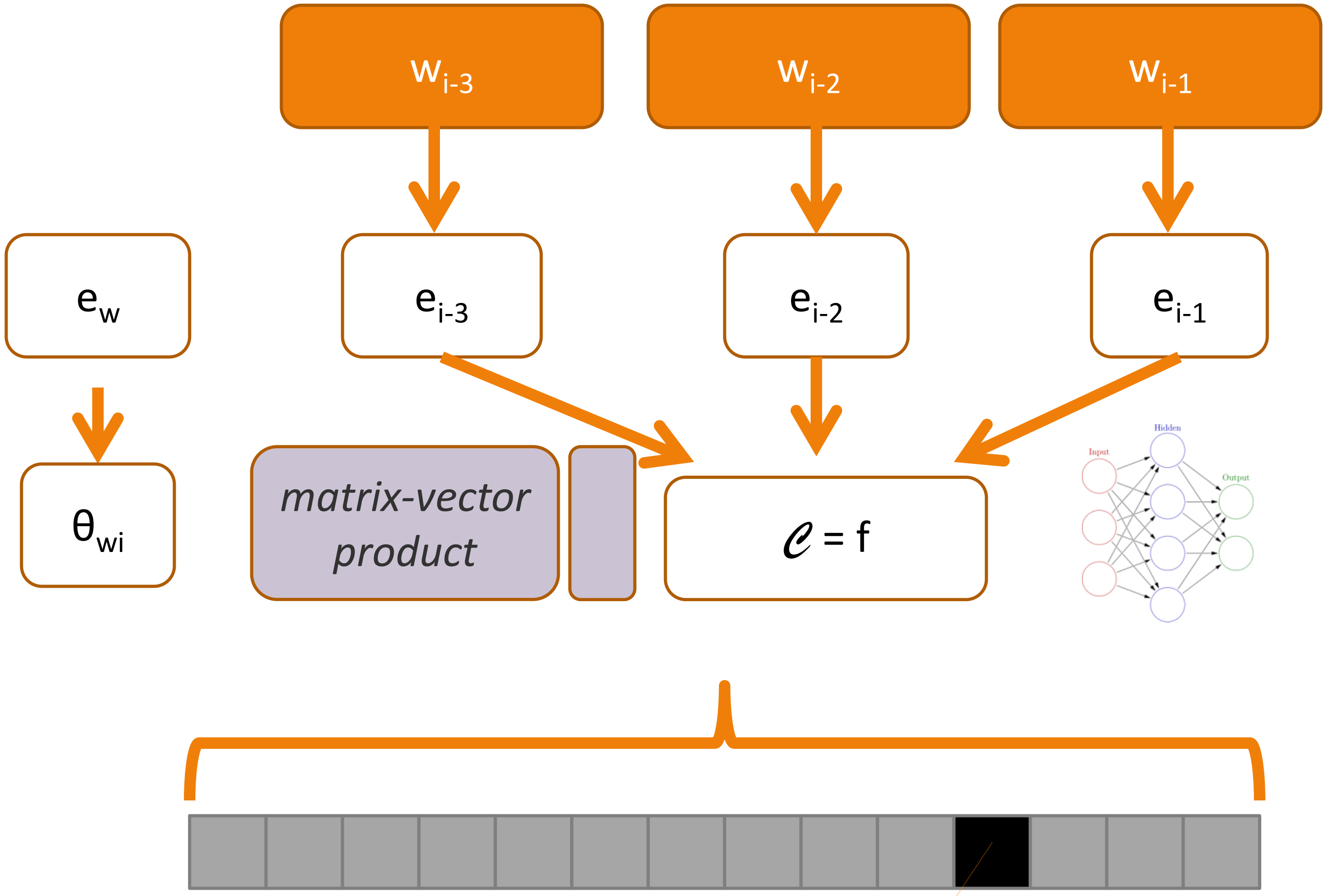
given some context…

create/use “distributed representations”…

combine these representations…

compute beliefs about what is likely…

predict the next word



$$p(w_i | w_{i-3}, w_{i-2}, w_{i-1}) = \text{softmax}(\theta_{w_i} \cdot \boldsymbol{f}(w_{i-3}, w_{i-2}, w_{i-1}))$$

w_i

Neural Language Models

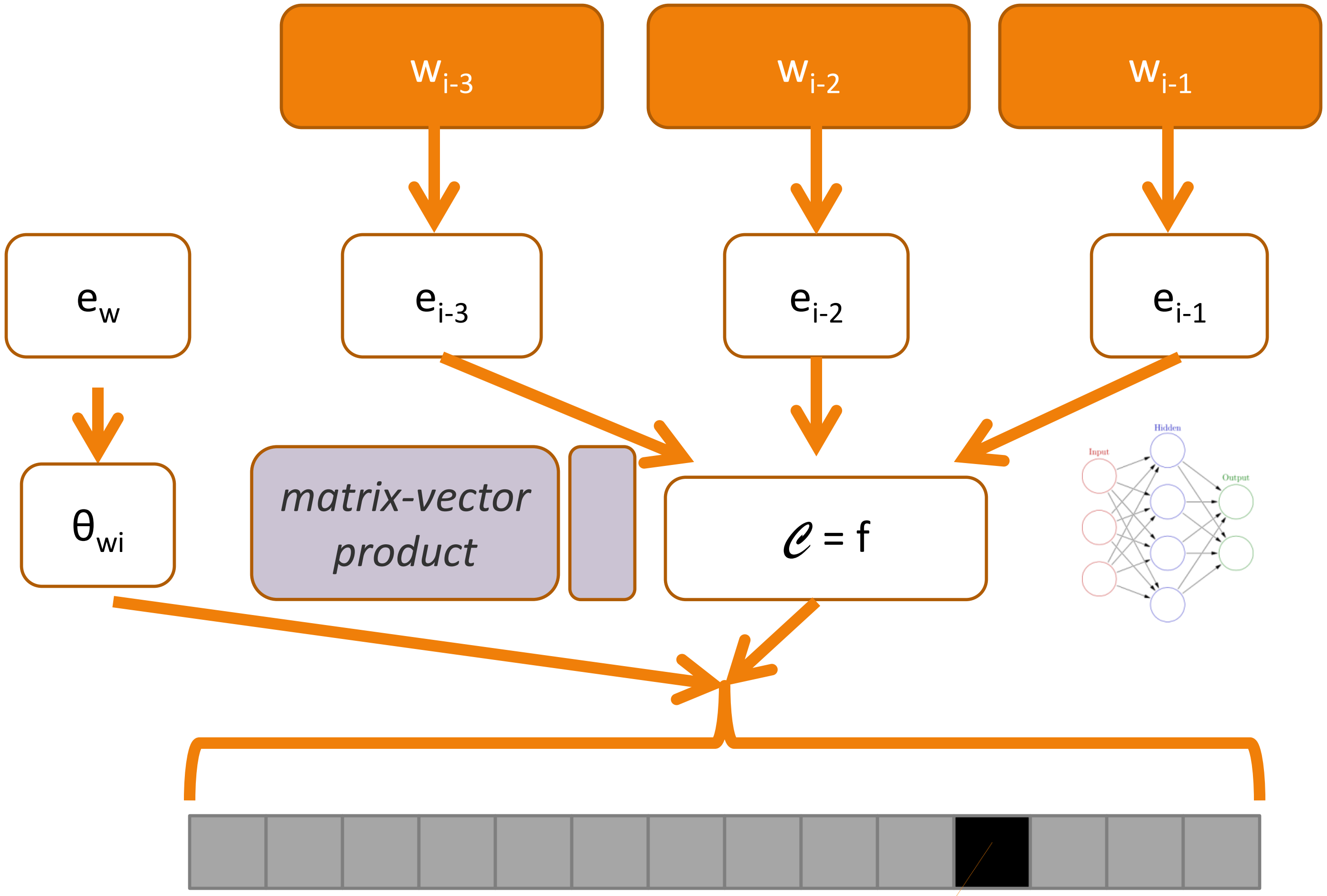
given some context…

create/use “distributed representations”…

combine these representations…

compute beliefs about what is likely…

predict the next word

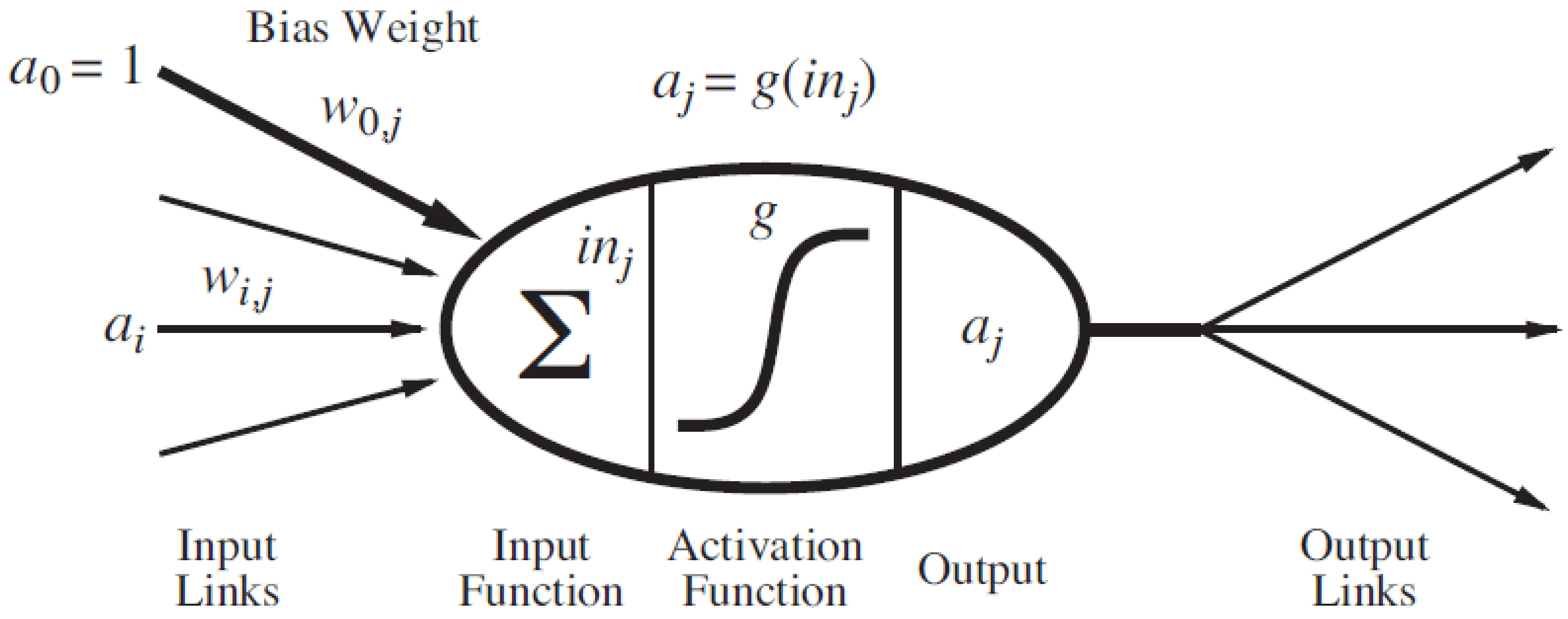


$$p(w_i | w_{i-3}, w_{i-2}, w_{i-1}) = \text{softmax}(\theta_{w_i} \cdot \boldsymbol{f}(w_{i-3}, w_{i-2}, w_{i-1}))$$

w_i

Biologically-Inspired Learning Models: Neuron Unit

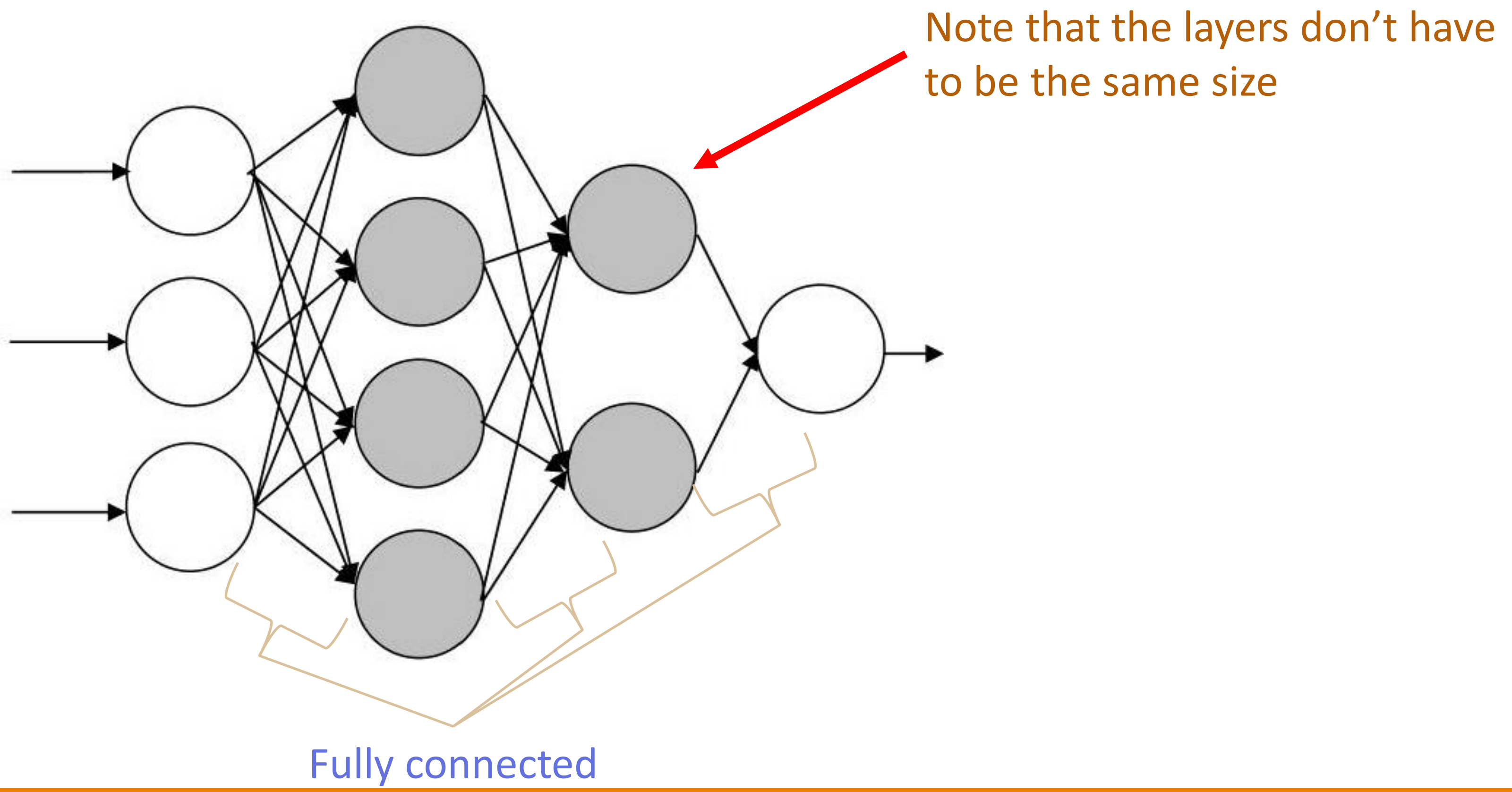
$$in_j = w_{0j} + w_{1j}a_1 + w_{2j}a_2 + \cdots + w_{ij}a_i$$



activations
$0 \leq a_i \leq 1$

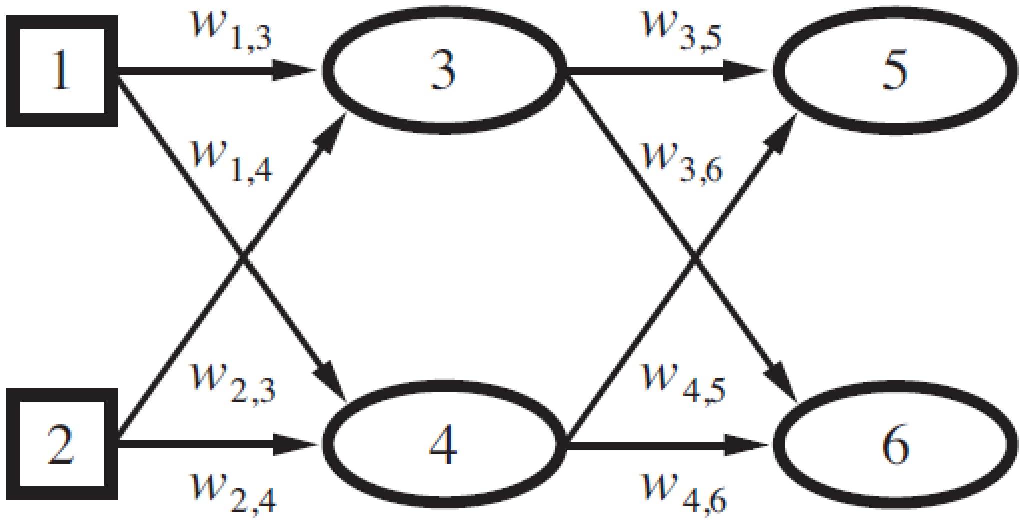
weights
$-\infty < w_{ij} < \infty$

Multi-layer Networks: General Structure Example



Multi-layer Networks: General Structure

Multi-layer perceptrons (aka neural networks) will have **inputs**, one or more **hidden layers**, and an **output layer:**



Multi-layer Networks: General Structure

Multi-layer perceptrons (aka neural networks) will have **inputs**, one or more **hidden layers**, and an **output layer:**

Number of inputs, outputs, and number and size of hidden layers can vary

Combination of **different weights** and **different structures** represent different **functions**

We will treat each layer as **fully-connected**

- Each unit in one layer connects to every unit in the next layer

Computing Values: Forward Propagation

Forward propagation calculates the output values for a given set of input values

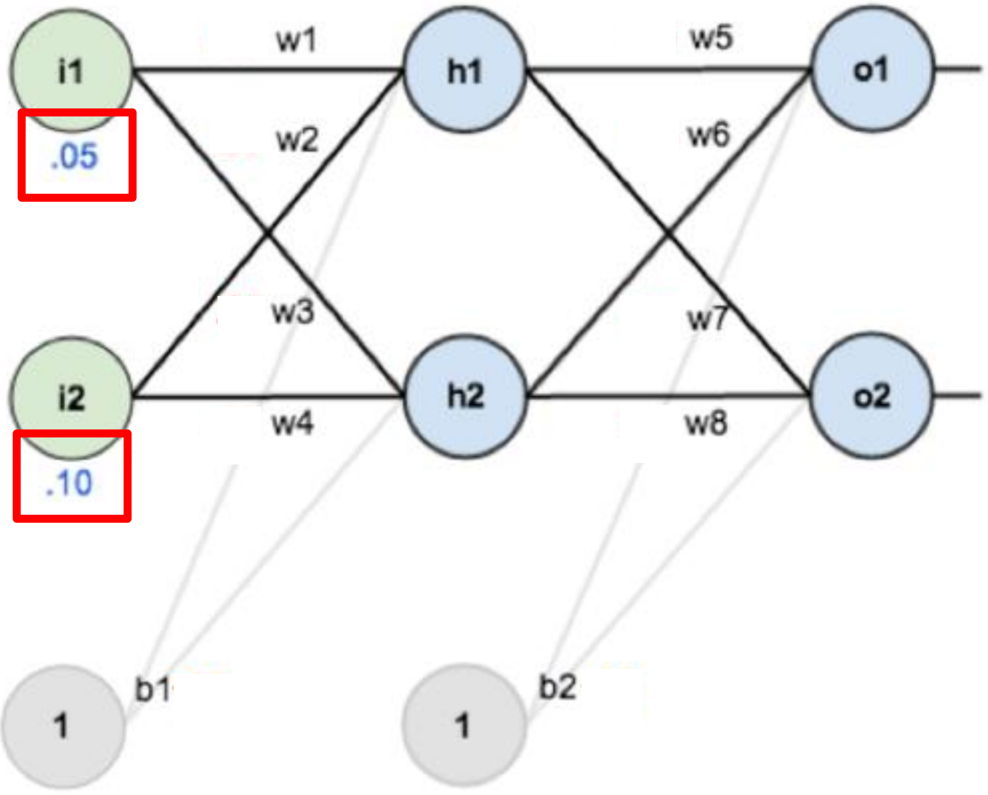
Algorithm

For each layer:

1. Calculate the weighted sum of inputs to each neuron unit
2. Evaluate the activation function to determine the output of each neuron unit
3. Use outputs as inputs for the next layer

Forward Propagation Example

Calculate the output of the network below, assuming each neuron uses a sigmoid activation function, given 0.05 and 0.1 as inputs.

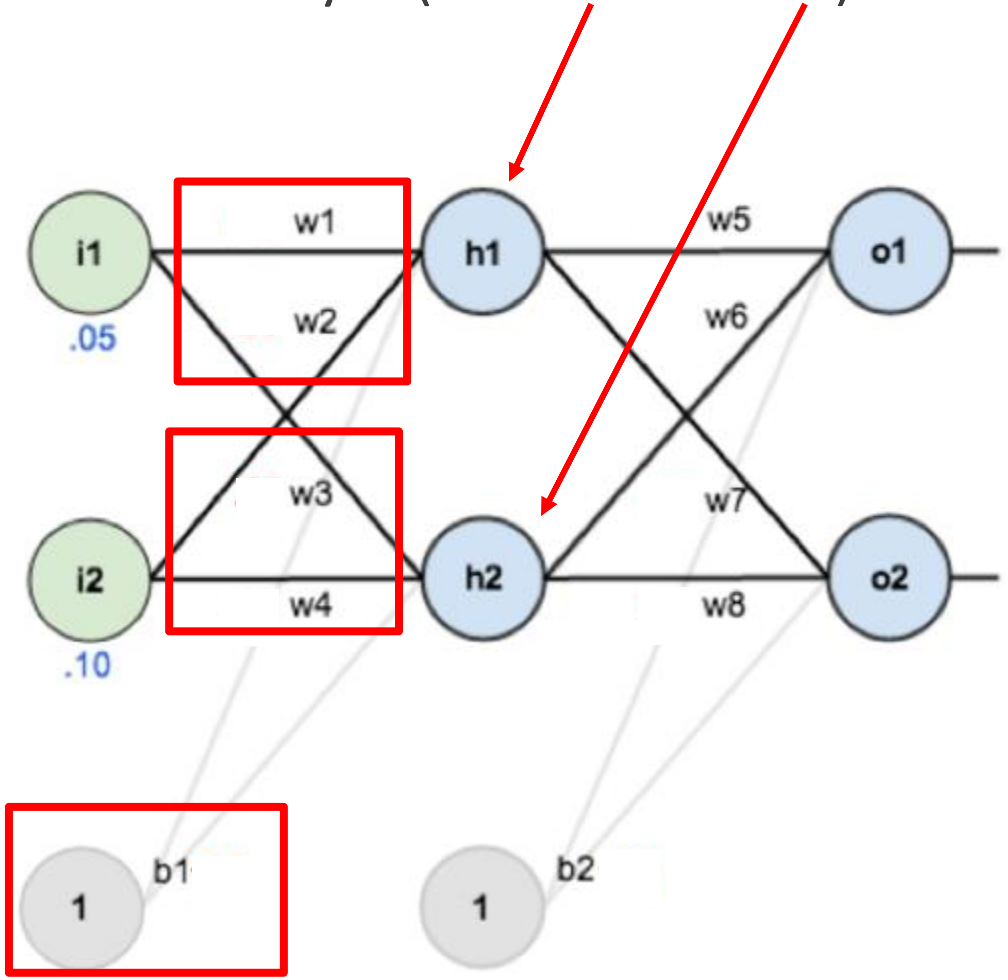


For each layer:

1. Calculate the weighted sum of inputs to each neuron unit
2. Evaluate the activation function to determine the output of each neuron unit
3. Use outputs as inputs for the next layer

Forward Propagation Example

Calculate inputs to the hidden layer (units h1 and h2):



$$
\begin{aligned}
in_{h1} &= w_1 i_1 + w_2 i_2 + b_1 \\
&= .15(.05)+.2(.1)-.35 \\
&= .0075+.02-.35 \\
&= -.3225
\end{aligned}
$$

$$
\begin{aligned}
in_{h2} &= w_3 i_1 + w_4 i_2 + b_2 \\
&= .25(.05)+.3(.1)-.35 \\
&= .0125+.03-.35 \\
&= -.3075
\end{aligned}
$$

For each layer:

1. Calculate the weighted sum of inputs to each neuron unit
2. Evaluate the activation function to determine the output of each neuron unit
3. Use outputs as inputs for the next layer

Forward Propagation Example

Calculate <u>outputs</u> to the hidden layer (units h1 and h2):

How do we do this?

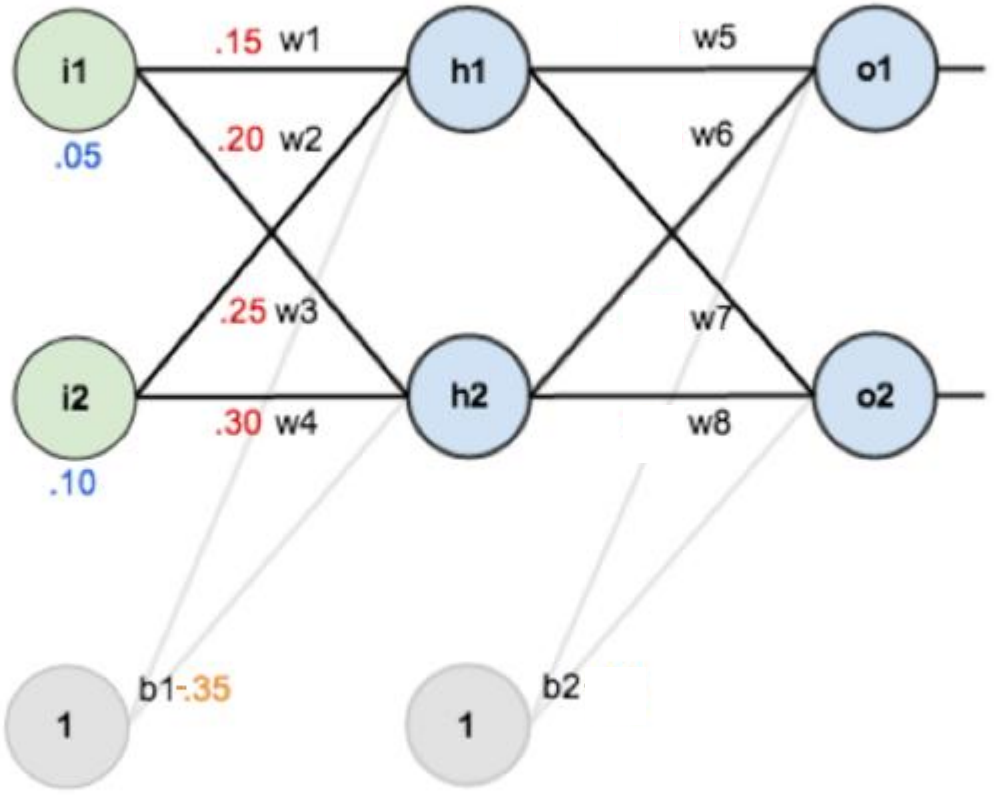
Use our activation function!

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

What will be our *x*?

in_{h1} =–.3225

in_{h2} = –.3075



$out_{h1} = g(in_{h1})$

$= \frac{1}{1+e^{-in_{h1}}}$

$= \frac{1}{1+e^{-(-.3275)}}$

= .4188

$out_{h2} = g(in_{h2})$

$= \frac{1}{1+e^{-in_{h2}}}$

$= \frac{1}{1+e^{-(-.3075)}}$

= .4237

For each layer:

1. Calculate the weighted sum of inputs to each neuron unit
2. Evaluate the activation function to determine the output of each neuron unit
3. Use outputs as inputs for the next layer