AN

Can you please write a sonet for my upcoming Advanced Natural Language Processing class? i will be teaching about neural networks and NLP, transformers, syntax, morphology, and translation.

Here is a sonnet about NLP and neural networks:

In the realm of language, deep neural nets
Unlock the secrets of syntax and word forms,
With Transformers leading the way, it sets
A new standard for language norms.

Morphology and translation, once so tough,
Are now made simple by these powerful tools.
We train them on mountains of data, enough
To make language learning a breeze for fools.

But let us not forget the human touch,
For even machines can make mistakes.
In this class, we'll learn how to make them such
That language understanding they'll undertake.

So join me as we delve into this quest,
To unlock the secrets of language at its best.

CS678 - Advanced NLP

Intro:
Neural Language Models

Antonis Anastasopoulos
antonis@gmu.edu

NLP GEORGE MASON

Our goal today

What is natural language processing?

Specifically today:

- Class Logistics
- Neural Networks for NLP and Language Modeling

Hello, everyone!

Research Interests: NLP and AI

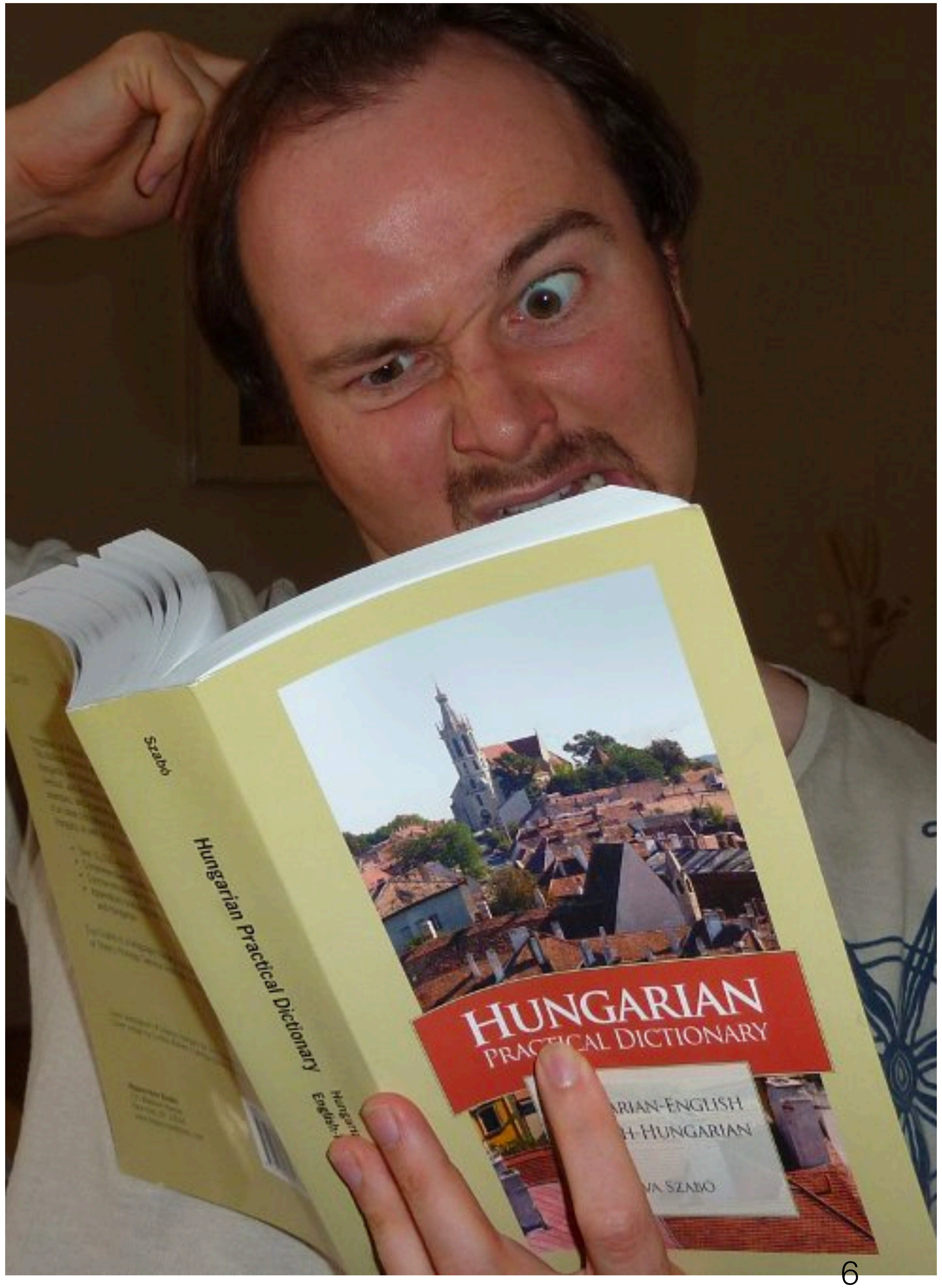
- NLP for Low-Resource Languages
- Machine Translation
- Multilinguality and Cross-Lingual Learning
- Fairness in NLP

Collaborations:

Carnegie Mellon University, U. Washington, Google, Amazon (AWS), Meta, U. of Notre Dame, Microsoft, Karya.

Antonis Anastasopoulos,
Asst. Prof.@GMU CS
Researcher @Archimedes AI

A bit about you



Language is Hard!

Are These Sentences OK?

Are These Sentences OK?

Jane went to the store.

Are These Sentences OK?

Jane went to the store.

store to Jane went the.

Are These Sentences OK?

Jane went to the store.

store to Jane went the.

Jane went store.

Are These Sentences OK?

Jane went to the store.

store to Jane went the.

Jane went store.

Jane goed to the store.

Are These Sentences OK?

Jane went to the store.

store to Jane went the.

Jane went store.

Jane goed to the store.

The store went to Jane.

Are These Sentences OK?

Jane went to the store.

store to Jane went the.

Jane went store.

Jane goed to the store.

The store went to Jane.

The food truck went to Jane.

Engineering Solutions

Jane went to the store.

store to Jane went the.

Jane went store.

Jane goed to the store.

The store went to Jane.

The food truck went to Jane.

Engineering Solutions

Jane went to the store.

store to Jane went the.

Jane went store.

Jane goed to the store.

The store went to Jane.

The food truck went to Jane.

Create a grammar of the language

Engineering Solutions

Jane went to the store.

store to Jane went the.

Jane went store.

Jane goed to the store.

The store went to Jane.

The food truck went to Jane.

Create a grammar of the language

Consider morphology and exceptions

Engineering Solutions

Jane went to the store.

store to Jane went the.

Create a grammar of the language

Jane went store.

Jane goed to the store.

Consider morphology and exceptions

The store went to Jane.

The food truck went to Jane.

Semantic categories, preferences

Engineering Solutions

Jane went to the store.

store to Jane went the.

} Create a grammar of the language

Jane went store.

} Consider morphology and exceptions

Jane goed to the store.

The store went to Jane.

} Semantic categories, preferences

The food truck went to Jane.

} And their exceptions

Are These Sentences OK?

ジェインは店へ行った。

は店行ったジェインは。

ジェインは店へ行た。

店はジェインへ行った。

屋台はジェインのところへ行った。

Μπορείτε να διαβάσετε αυτήν την πρόταση;

Potete leggere questa frase?

इस वाक्य क्या आप को पढ़ सकते हैं?

mungawerenge chiganizo ichi?

Phenomena to Handle

Morphology

Syntax

Semantics/World Knowledge

Discourse

Pragmatics

Multilinguality

Neural Nets for NLP

Neural Nets for NLP

Neural nets are a tool to do hard things!

Neural Nets for NLP

Neural nets are a tool to do hard things!

Combined with lots of data and compute, they can approximate any function!

What do you think of
when you think of NLP?

What will you learn?

What will you learn?

1. The generics of how large pre-trained neural language models are trained and operate

What will you learn?

1. The generics of how large pre-trained neural language models are trained and operate
2. Understand the limitations of current technologies

What will you learn?

1. The generics of how large pre-trained neural language models are trained and operate
2. Understand the limitations of current technologies
3. Learn about the harms associated with their use and ways to mitigate them

How will you learn?
(syllabus highlights)

Syllabus Highlights

Syllabus Highlights

Mostly lectures, but

- I'll try to have some sort of interactive element every day

Syllabus Highlights

Mostly lectures, but

- I'll try to have some sort of interactive element every day

Syllabus Highlights

Mostly lectures, but

    I’ll try to have some sort of interactive element every day

- I will provide additional readings for anyone interested

Lectures

Lectures

You should ask lots of questions

Lectures

You should ask lots of questions

- interrupting (by raising a hand) to ask your question is *strongly* encouraged

Lectures

You should ask lots of questions

- interrupting (by raising a hand) to ask your question is *strongly* encouraged
- Asking questions later (or in real time)

Lectures

You should ask lots of questions

- interrupting (by raising a hand) to ask your question is *strongly* encouraged
- Asking questions later (or in real time)

Lectures

You should ask lots of questions

- interrupting (by raising a hand) to ask your question is *strongly* encouraged
- Asking questions later (or in real time)

Interaction improves learning!

Logistics

We will use Discord for

- Announcements
- Distributing course materials before/after class

Language Models

Calculating the Probability of a Sentence

Jane went to the store .

$$P(X) = \prod_{i=1}^{n} P(x_i)$$

Calculating the Probability of a Sentence

Jane went to the store .

$$P(X) = \prod_{i=1}^{n} \underbrace{P(x_i)}_{\text{Unigram}}$$

Calculating the Probability of a Sentence

Jane went to the store .

$$P(X) = \prod_{i=1}^{n} P(x_i)$$

Unigram

$$P(\text{Jane went to the store}) = P(Jane) \times P(went) \times P(to) \times P(the) \times P(store) \times P(\,.\,) \,.$$

Calculating the Probability of a Sentence

Jane went to the store .

$$P(X) = \prod_{i=1}^{n} \underbrace{P(x_i)}_{\text{Unigram}}$$

$$P(\text{Jane went to the store}) = P(Jane) \times P(went) \times P(to) \times P(the) \times P(store) \times P(\,.\,)\,.$$

But word order and context matters!

Calculating the Probability of a Sentence

$$P(X) = \prod_{i=1}^{I} P(\underbrace{x_i}_{\text{Next Word}} \mid \underbrace{x_1, \ldots, x_{i-1}}_{\text{Context}})$$

Calculating the Probability of a Sentence

$$P(X) = \prod_{i=1}^{I} P(\underbrace{x_i}_{\text{Next Word}} \mid \underbrace{x_1, \ldots, x_{i-1}}_{\text{Context}})$$

$$\begin{aligned} P(\text{Jane went to the store}) = \; & P(Jane \mid <s>) \times P(went \mid Jane) \times \\ & P(to \mid went) \times P(the \mid to) \times \\ & P(store \mid the) \times P(. \mid store) \\ & P(</s> \mid .) \end{aligned}$$

Calculating the Probability of a Sentence

$$P(X) = \prod_{i=1}^{I} P(\underbrace{x_i}_{\text{Next Word}} \mid \underbrace{x_1, \ldots, x_{i-1}}_{\text{Context}})$$

The big problem: How do we predict

$$P(x_i \mid x_1, \ldots, x_{i-1})$$

?!?!

Count-based Language Models

Count-based Language Models

- Count up the frequency and divide:

$$P_{ML}(x_i \mid x_{i-n+1}, \ldots, x_{i-1}) := \frac{c(x_{i-n+1}, \ldots, x_i)}{c(x_{i-n+1}, \ldots, x_{i-1})}$$

Count-based Language Models

- Count up the frequency and divide:

$$P_{ML}(x_i \mid x_{i-n+1}, \ldots, x_{i-1}) := \frac{c(x_{i-n+1}, \ldots, x_i)}{c(x_{i-n+1}, \ldots, x_{i-1})}$$

Corpus:

The cat sat on the mat .
A dog chased the cat .
A mouse ate some cheese .
The mouse ran under a mat .

Count-based Language Models

- Count up the frequency and divide:

$$P_{ML}(x_i \mid x_{i-n+1}, \ldots, x_{i-1}) := \frac{c(x_{i-n+1}, \ldots, x_i)}{c(x_{i-n+1}, \ldots, x_{i-1})}$$

Corpus:

The cat sat on the mat .
A mouse ate some cheese .
A dog chased the cat .
The mouse ran under a mat .

$$p(chased \mid dog) = ? \quad p(cat \mid the) = ? \quad p(the \mid < s >) = ?$$

Count-based Language Models

- Count up the frequency and divide:

$$P_{ML}(x_i \mid x_{i-n+1}, \ldots, x_{i-1}) := \frac{c(x_{i-n+1}, \ldots, x_i)}{c(x_{i-n+1}, \ldots, x_{i-1})}$$

Corpus:

The cat sat on the mat .
A dog chased the cat .
A mouse ate some cheese .
The mouse ran under a mat .

$$p(chased \mid dog) = \frac{1}{1} = 1 \quad p(cat \mid the) = \frac{1}{4} = 0.25 \quad p(the \mid <s>) = 0.5$$

Count-based Language Models

- Count up the frequency and divide:

$$P_{ML}(x_i \mid x_{i-n+1}, \ldots, x_{i-1}) := \frac{c(x_{i-n+1}, \ldots, x_i)}{c(x_{i-n+1}, \ldots, x_{i-1})}$$

Corpus:

The cat sat on the mat .
A dog chased the cat .
A mouse ate some cheese .
The mouse ran under a mat .

$p(\text{A cat chased the mouse .}) = ?$

Count-based Language Models

Corpus:

The cat sat on the mat .
A dog chased the cat .
A mouse ate some cheese .
The mouse ran under a mat .

p(A cat chased the mouse .) =

$$p(<s>|A)\times$$
$$p(cat|a)\times$$
$$p(chased|cat)\times$$
$$p(the|chased)\times$$
$$p(mouse|the)\times$$
$$p(\,.\,|mouse)$$

Count-based Language Models

Corpus:

The cat sat on the mat .   A mouse ate some cheese .
A dog chased the cat .   The mouse ran under a mat .

p(A cat chased the mouse .) =

$$p(<s>|A) \times$$
$$p(cat|a) \times$$
$$p(chased|cat) \times$$
$$p(the|chased) \times$$
$$p(mouse|the) \times$$
$$p(\,.\,|mouse)$$

Count-based Language Models

- Count up the frequency and divide:

$$P_{ML}(x_i \mid x_{i-n+1}, \ldots, x_{i-1}) := \frac{c(x_{i-n+1}, \ldots, x_i)}{c(x_{i-n+1}, \ldots, x_{i-1})}$$

- Add smoothing to deal with zero counts:

$$p(x_i | x_{i-n+1:i-1}) = \frac{c(x_{i-n+1:i}) + \alpha}{c(x_{i-n+1:i-1}) + \alpha |V|}$$

Count-based Language Models

Corpus:

The cat sat on the mat . A mouse ate some cheese .
A dog chased the cat . The mouse ran under a mat .

$|V| = |\{the, a, cat, sat, \ldots\}| = 15$ $\alpha = 1$

Count-based Language Models

Corpus:

The cat sat on the mat .

A dog chased the cat .

A mouse ate some cheese .

The mouse ran under a mat .

$$|V| = |\{the, a, cat, sat, \ldots\}| = 15 \qquad \alpha = 1$$

$p($A cat chased the mouse .$) =$

$$p(<s> \| A) \times$$
$$p(cat \,|\, a) \times$$
$$p(chased \,|\, cat) \times$$
$$p(the \,|\, chased) \times$$
$$p(mouse \,|\, the) \times$$
$$p(\,.\,|\, mouse)$$

An Alternative:
Featurized Log-Linear Models

An Alternative:
Featurized Models

An Alternative: Featurized Models

- Calculate features of the context

An Alternative: Featurized Models

- Calculate features of the context
- Based on the features, calculate probabilities

An Alternative: Featurized Models

- Calculate features of the context
- Based on the features, calculate probabilities
- Optimize feature weights using gradient descent, etc.

Example:

Previous words: “giving a"

Example:

Previous words: "giving a"

a
the
talk
gift
hat
…

Words we're predicting

Example:

Previous words: “giving a"

Words we’re predicting	How likely are they?
a	3.0
the	2.5
talk	-0.2
gift	0.1
hat	1.2
…	…

$$b=\begin{pmatrix}3.0\\2.5\\-0.2\\0.1\\1.2\\\dots\end{pmatrix}$$

Example:

Previous words: “giving a"

$$\begin{array}{c} \text{a} \\ \text{the} \\ \text{talk} \\ \text{gift} \\ \text{hat} \\ \dots \end{array} \qquad b=\begin{pmatrix}3.0\\2.5\\-0.2\\0.1\\1.2\\\dots\end{pmatrix} \qquad w_{1,a}=\begin{pmatrix}-6.0\\-5.1\\0.2\\0.1\\0.5\\\dots\end{pmatrix}$$

Words we’re predicting

How likely are they?

How likely are they given prev. word is “a”?

Example:

Previous words: “giving a"

$$
\begin{matrix} \text{a} \\ \text{the} \\ \text{talk} \\ \text{gift} \\ \text{hat} \\ \dots \end{matrix}
\qquad
b = \begin{pmatrix} 3.0 \\ 2.5 \\ -0.2 \\ 0.1 \\ 1.2 \\ \dots \end{pmatrix}
\qquad
w_{1,a} = \begin{pmatrix} -6.0 \\ -5.1 \\ 0.2 \\ 0.1 \\ 0.5 \\ \dots \end{pmatrix}
\qquad
w_{2,giving} = \begin{pmatrix} -0.2 \\ -0.3 \\ 1.0 \\ 2.0 \\ -1.2 \\ \dots \end{pmatrix}
$$

Words we’re predicting	How likely are they?	How likely are they given prev. word is “a”?	How likely are they given 2nd prev. word is “giving”?

Example:

Previous words: “giving a"

$$
\begin{array}{c}
\text{a} \\ \text{the} \\ \text{talk} \\ \text{gift} \\ \text{hat} \\ \ldots
\end{array}
\qquad
b=\begin{pmatrix} 3.0 \\ 2.5 \\ -0.2 \\ 0.1 \\ 1.2 \\ \ldots \end{pmatrix}
\quad
w_{1,a}=\begin{pmatrix} -6.0 \\ -5.1 \\ 0.2 \\ 0.1 \\ 0.5 \\ \ldots \end{pmatrix}
\quad
w_{2,giving}=\begin{pmatrix} -0.2 \\ -0.3 \\ 1.0 \\ 2.0 \\ -1.2 \\ \ldots \end{pmatrix}
\quad
s=\begin{pmatrix} -3.2 \\ -2.9 \\ 1.0 \\ 2.2 \\ 0.6 \\ \ldots \end{pmatrix}
$$

Words we’re predicting	How likely are they?	How likely are they given prev. word is “a”?	How likely are they given 2nd prev. word is “giving”?	Total score

Softmax

- Convert scores into probabilities by taking the exponent and normalizing (softmax)

Softmax

- Convert scores into probabilities by taking the exponent and normalizing (softmax)

$$P(x_i \mid x_{i-n+1}^{i-1}) = \frac{e^{s(x_i|x_{i-n+1}^{i-1})}}{\sum_{\tilde{x}_i} e^{s(\tilde{x}_i|x_{i-n+1}^{i-1})}}$$

Softmax

- Convert scores into probabilities by taking the exponent and normalizing (softmax)

$$P(x_i \mid x_{i-n+1}^{i-1}) = \frac{e^{s(x_i \mid x_{i-n+1}^{i-1})}}{\sum_{\tilde{x}_i} e^{s(\tilde{x}_i \mid x_{i-n+1}^{i-1})}}$$

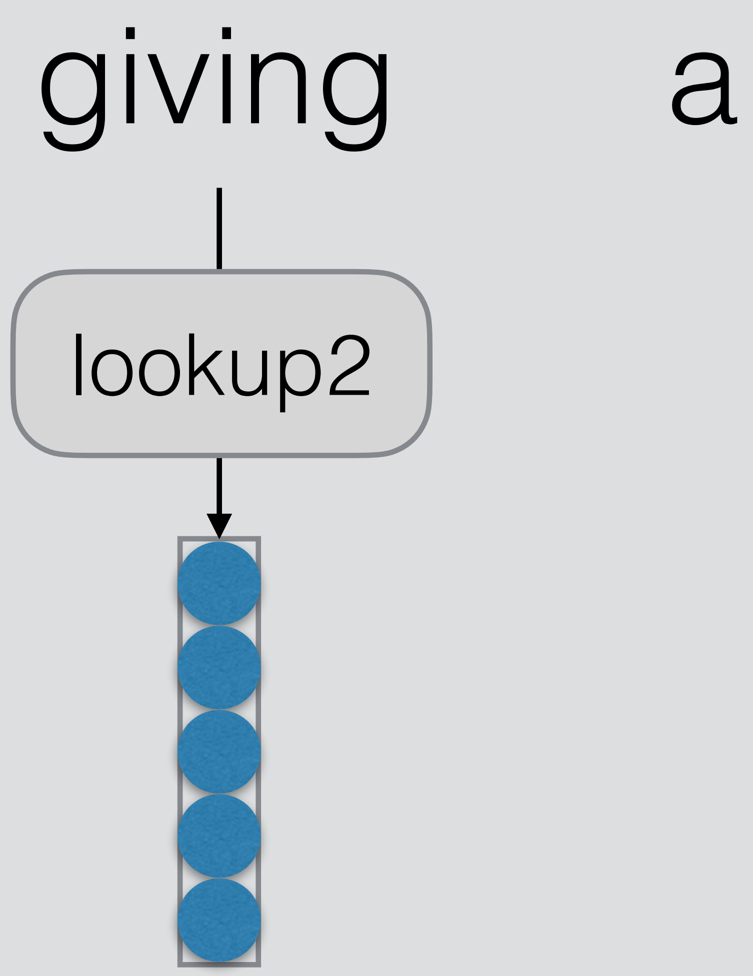
$$s = \begin{pmatrix} -3.2 \\ -2.9 \\ 1.0 \\ 2.2 \\ 0.6 \\ \dots \end{pmatrix} \longrightarrow p = \begin{pmatrix} 0.002 \\ 0.003 \\ 0.329 \\ 0.444 \\ 0.090 \\ \dots \end{pmatrix}$$

A Computation Graph View

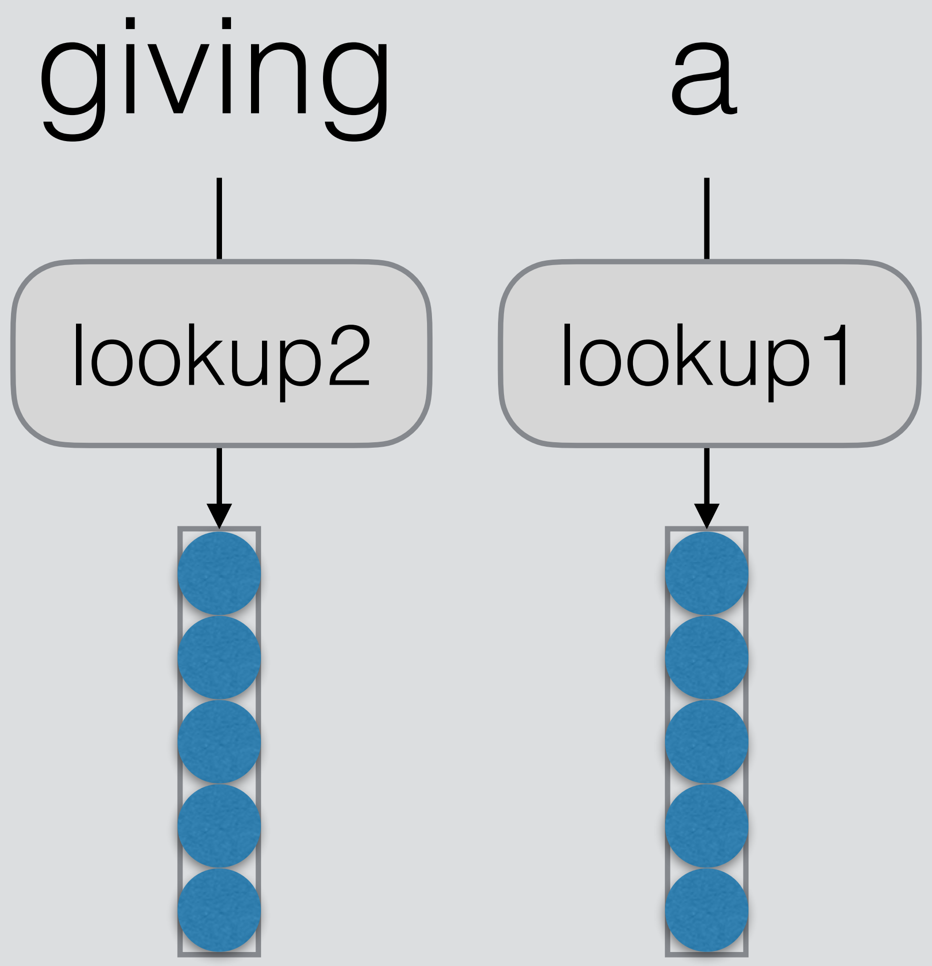
giving a

Each vector is size of output vocabulary

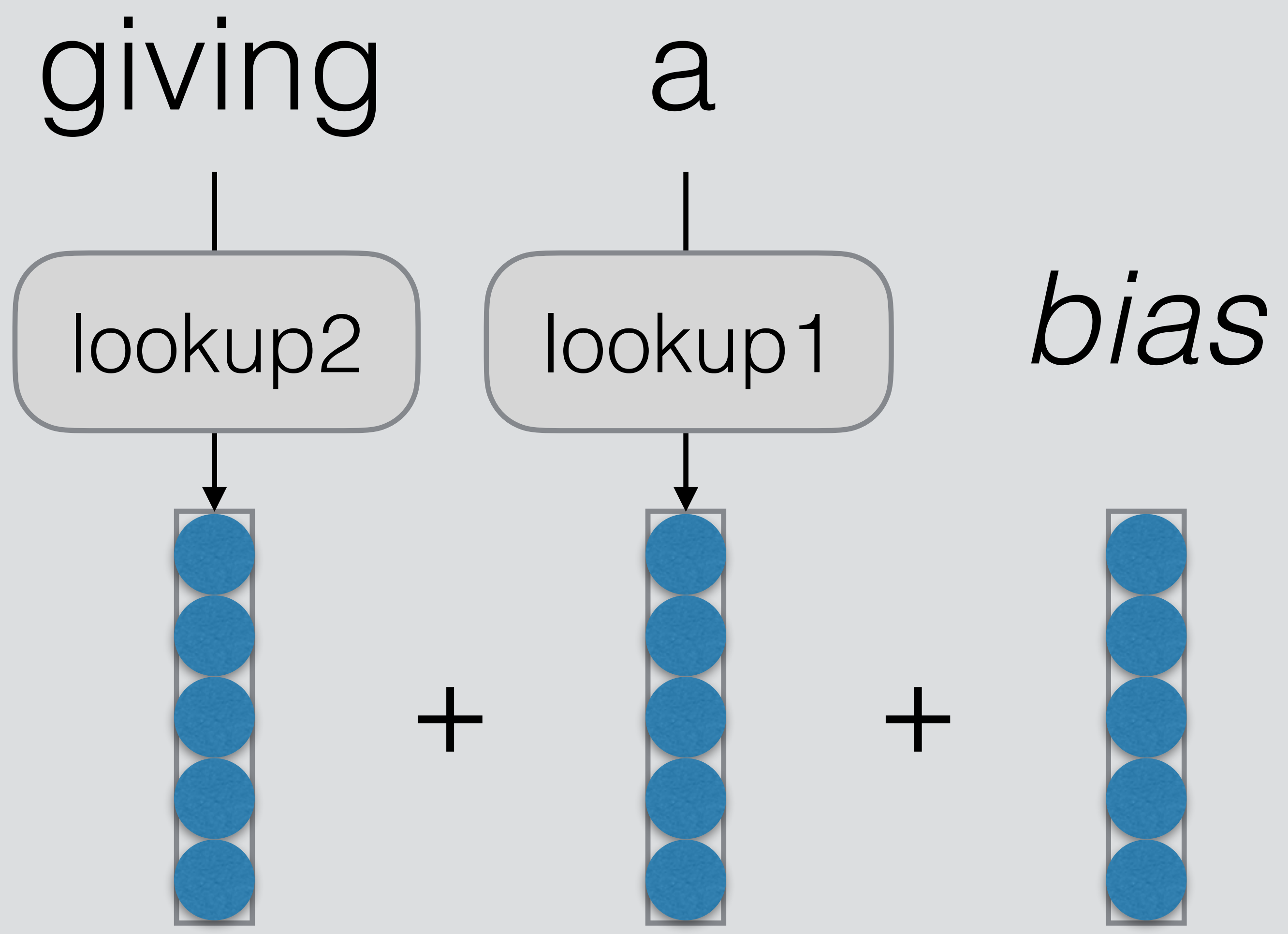
A Computation Graph View



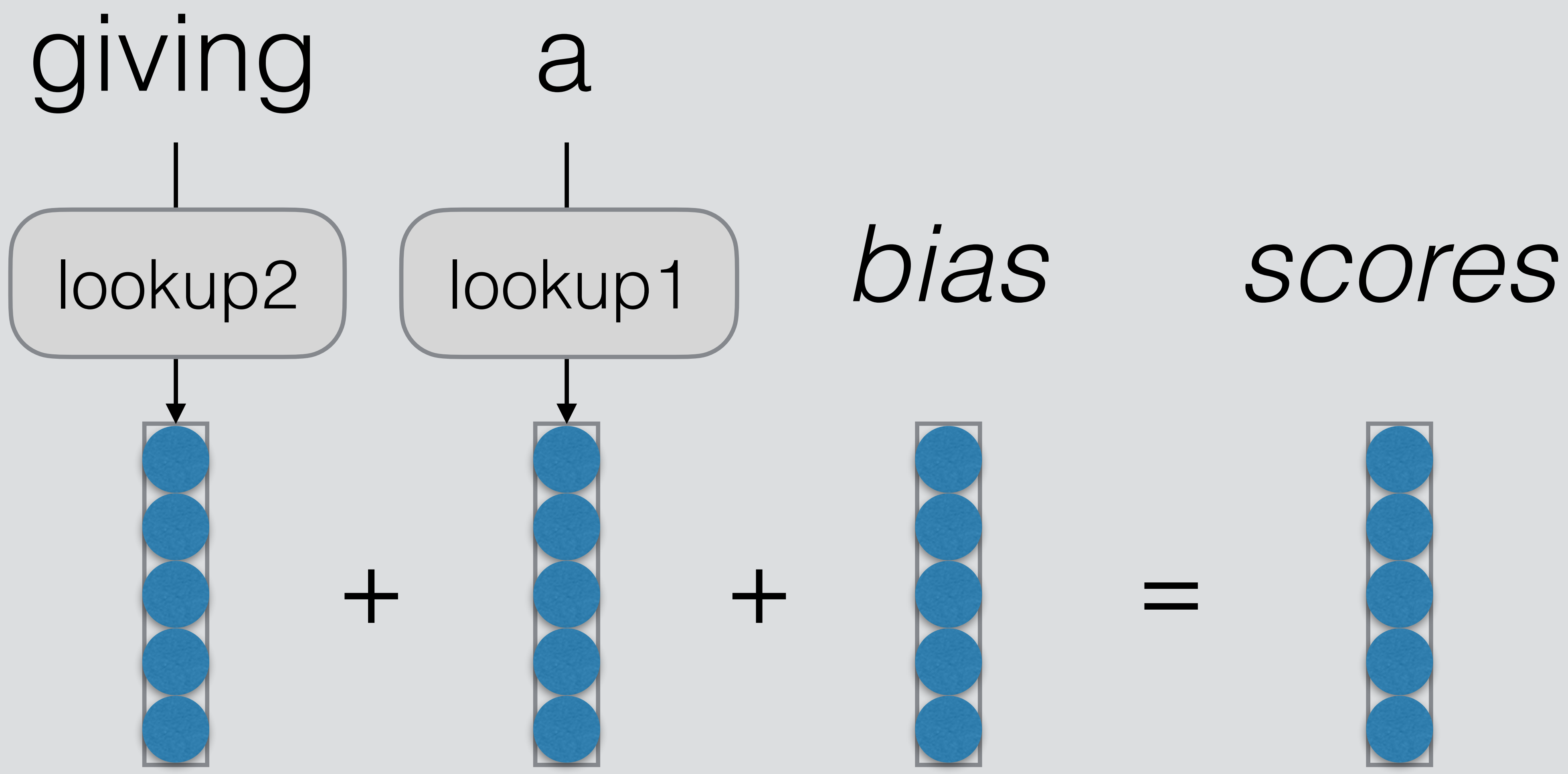
A Computation Graph View



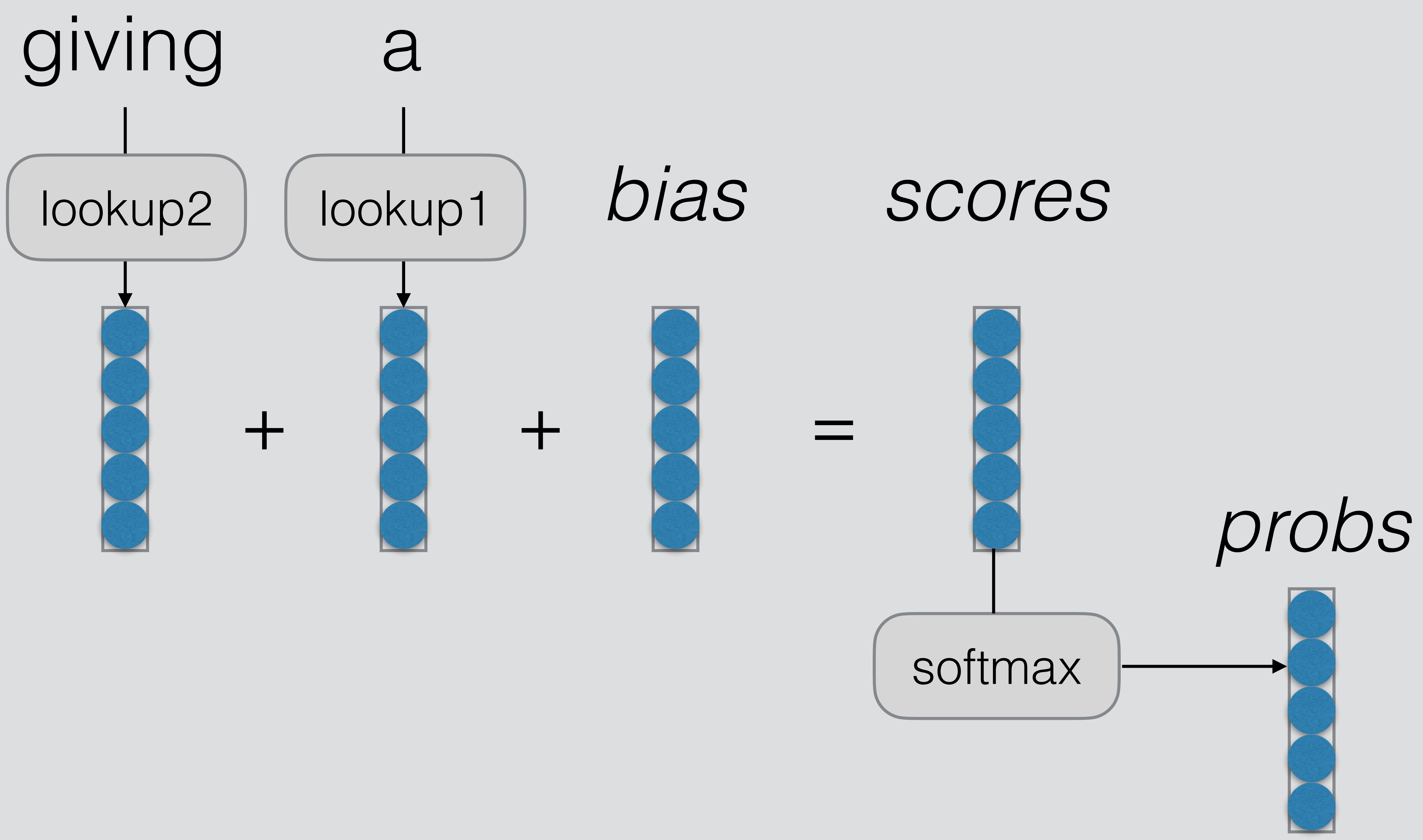
A Computation Graph View



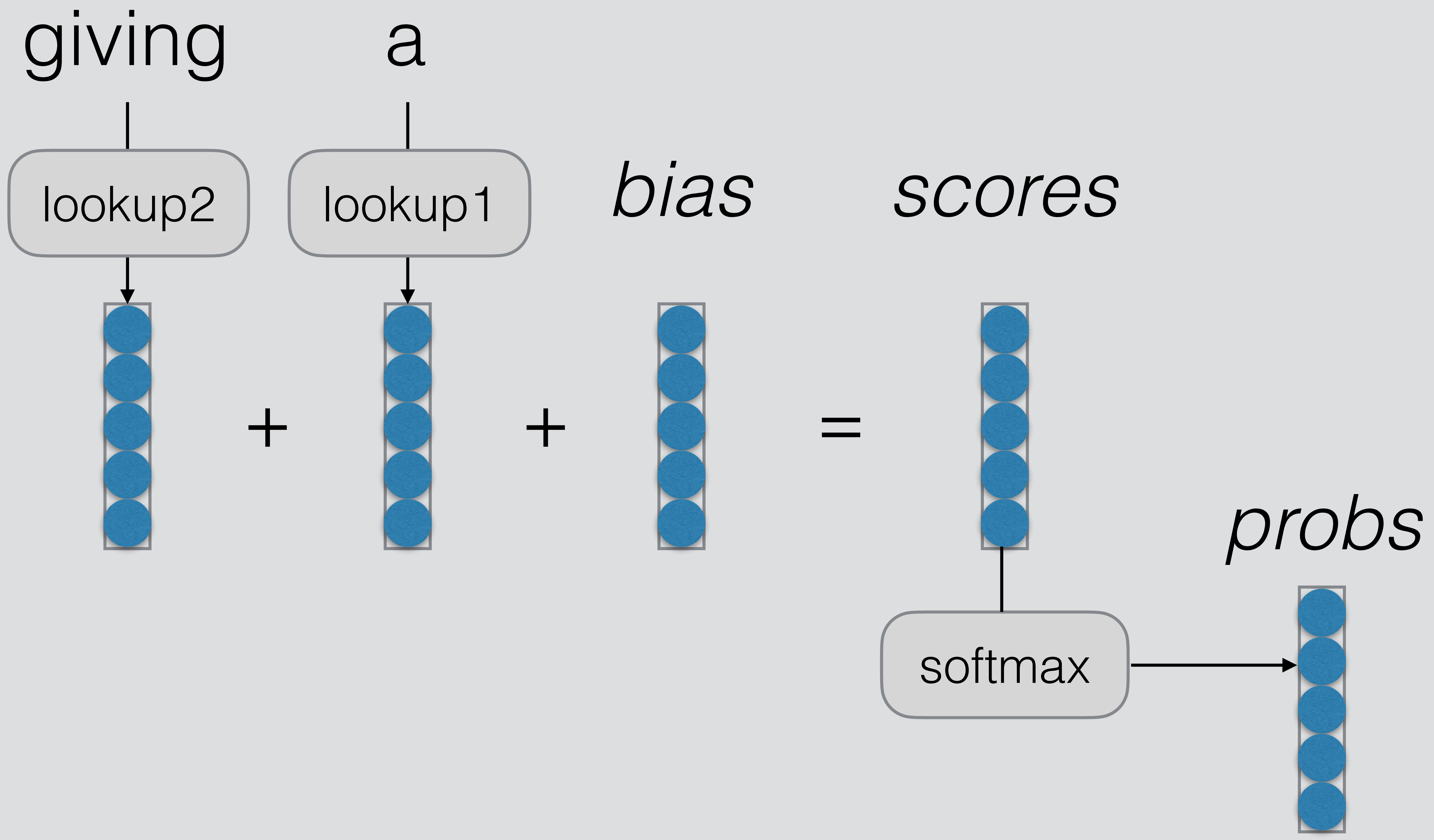
A Computation Graph View



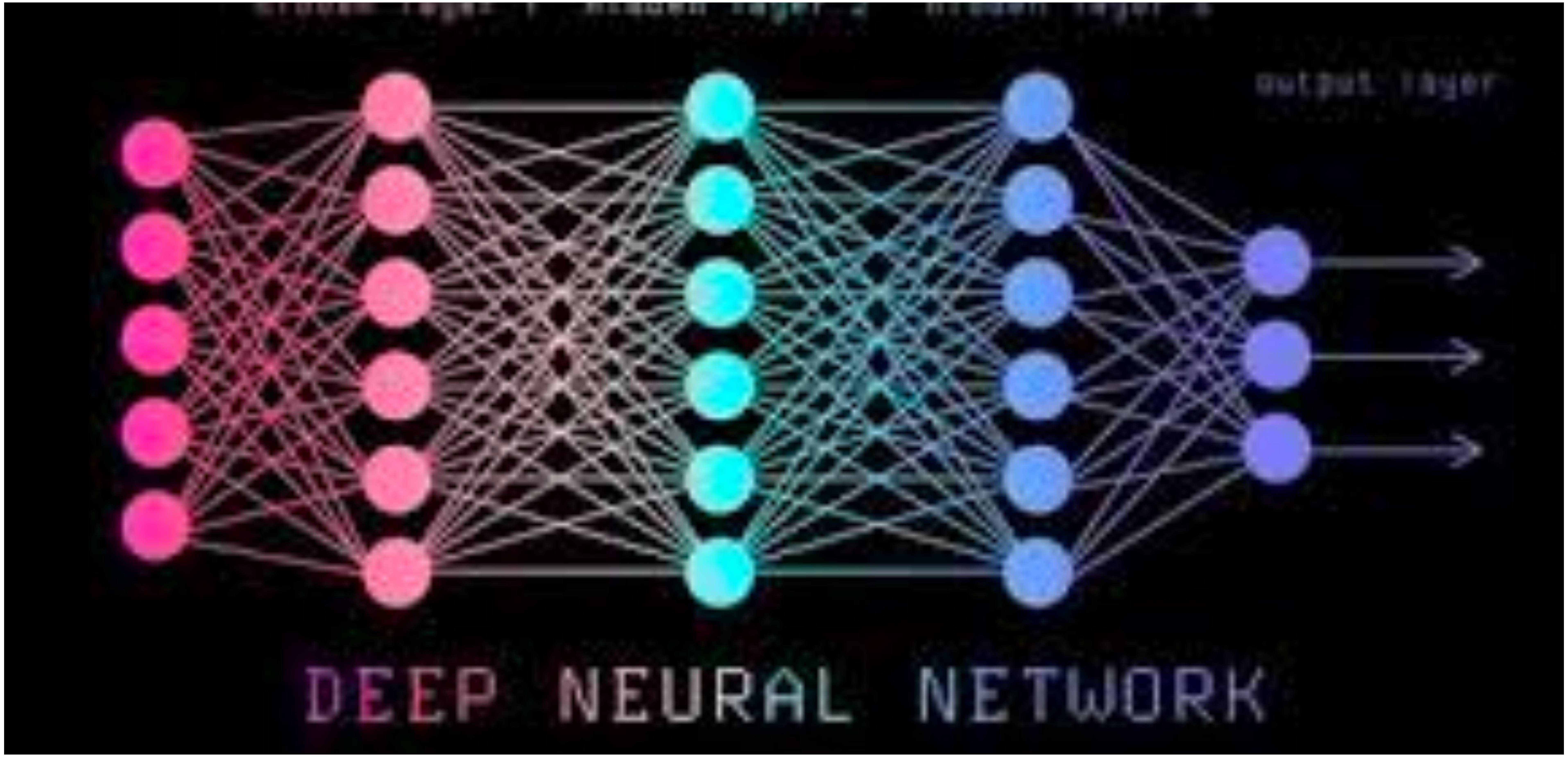
A Computation Graph View



A Computation Graph View



Each vector is size of output vocabulary



Neural Networks:
A Tool for Doing Hard Things

An Example Prediction Problem: Sentence Classification

An Example Prediction Problem: Sentence Classification

I hate this movie

An Example Prediction Problem: Sentence Classification

I hate this movie

I love this movie

An Example Prediction Problem: Sentence Classification

I hate this movie

very good
good
neutral
bad
very bad

I love this movie

An Example Prediction Problem: Sentence Classification

I hate this movie

very good
good
neutral
bad
very bad

I love this movie

very good
good
neutral
bad
very bad

An Example Prediction Problem: Sentence Classification

I hate this movie

very good
good
neutral
bad
very bad

I love this movie

very good
good
neutral
bad
very bad

An Example Prediction Problem: Sentence Classification

I hate this movie → very bad

very good
good
neutral
bad
very bad

I love this movie → very good

very good
good
neutral
bad
very bad

A First Try:
Bag of Words (BOW)

Each word has a vector of weights for each tag

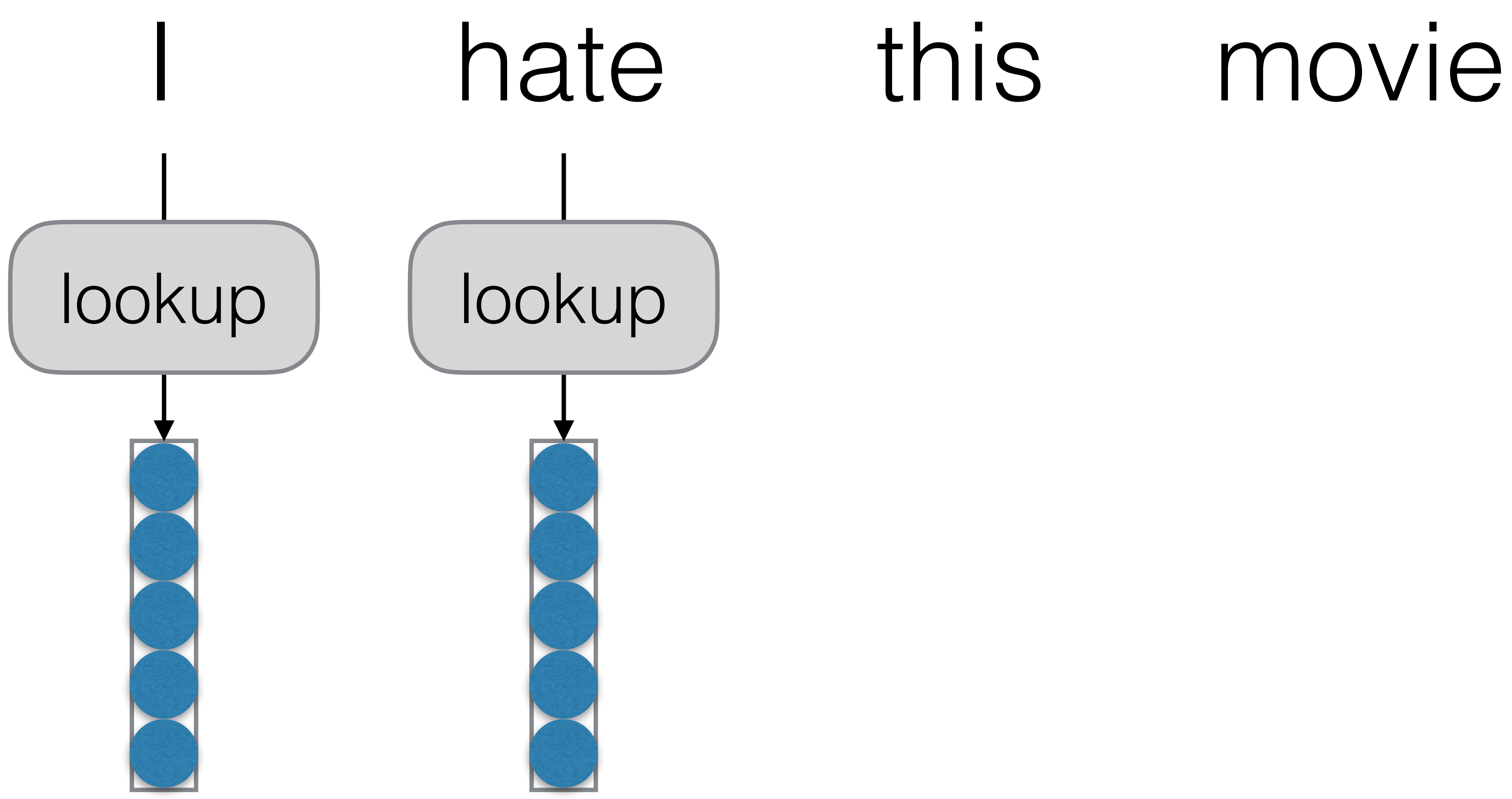
I hate this movie

A First Try:
Bag of Words (BOW)

Each word has a vector of weights for each tag

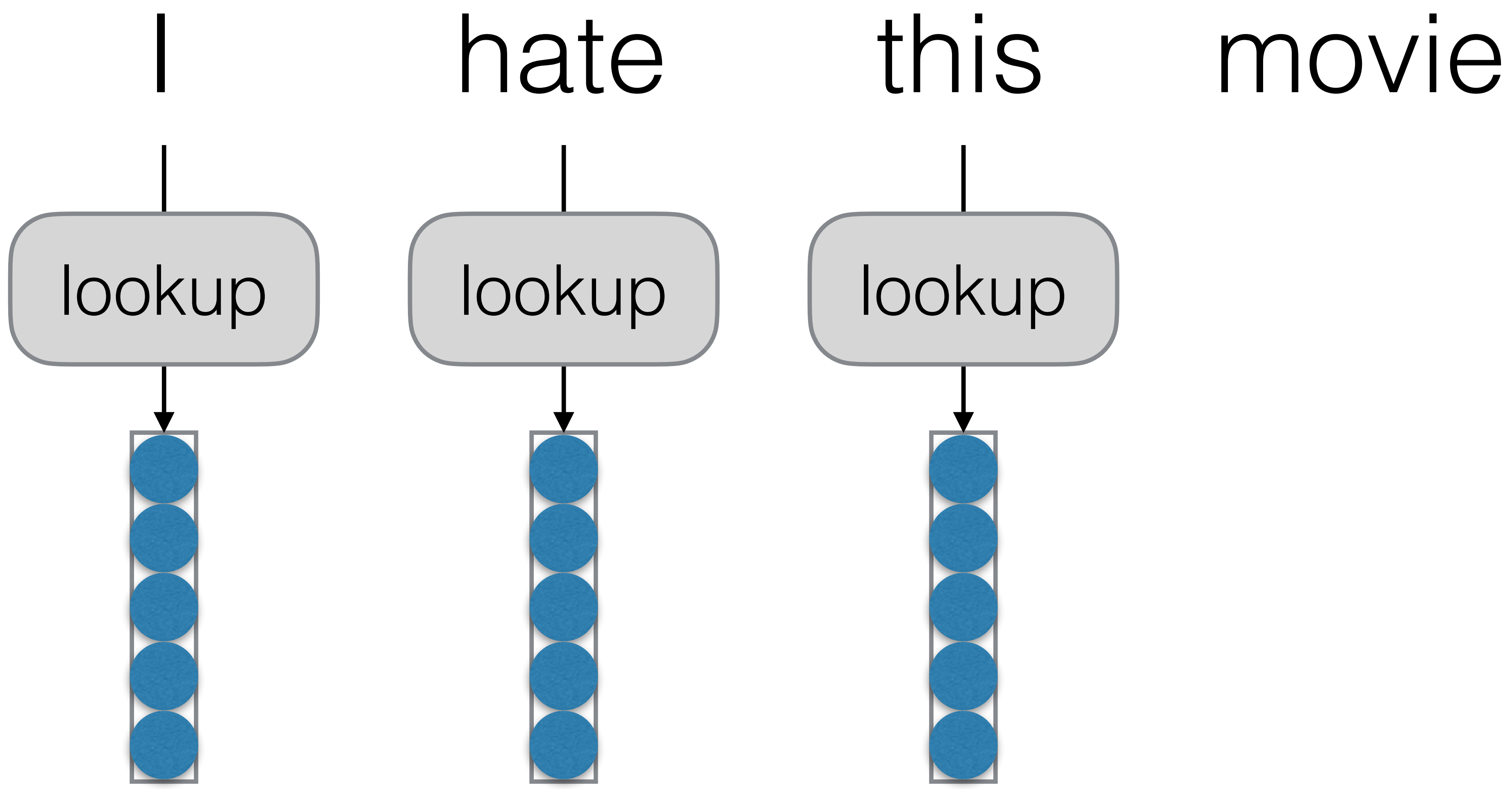
A First Try:
Bag of Words (BOW)

Each word has a vector of weights for each tag



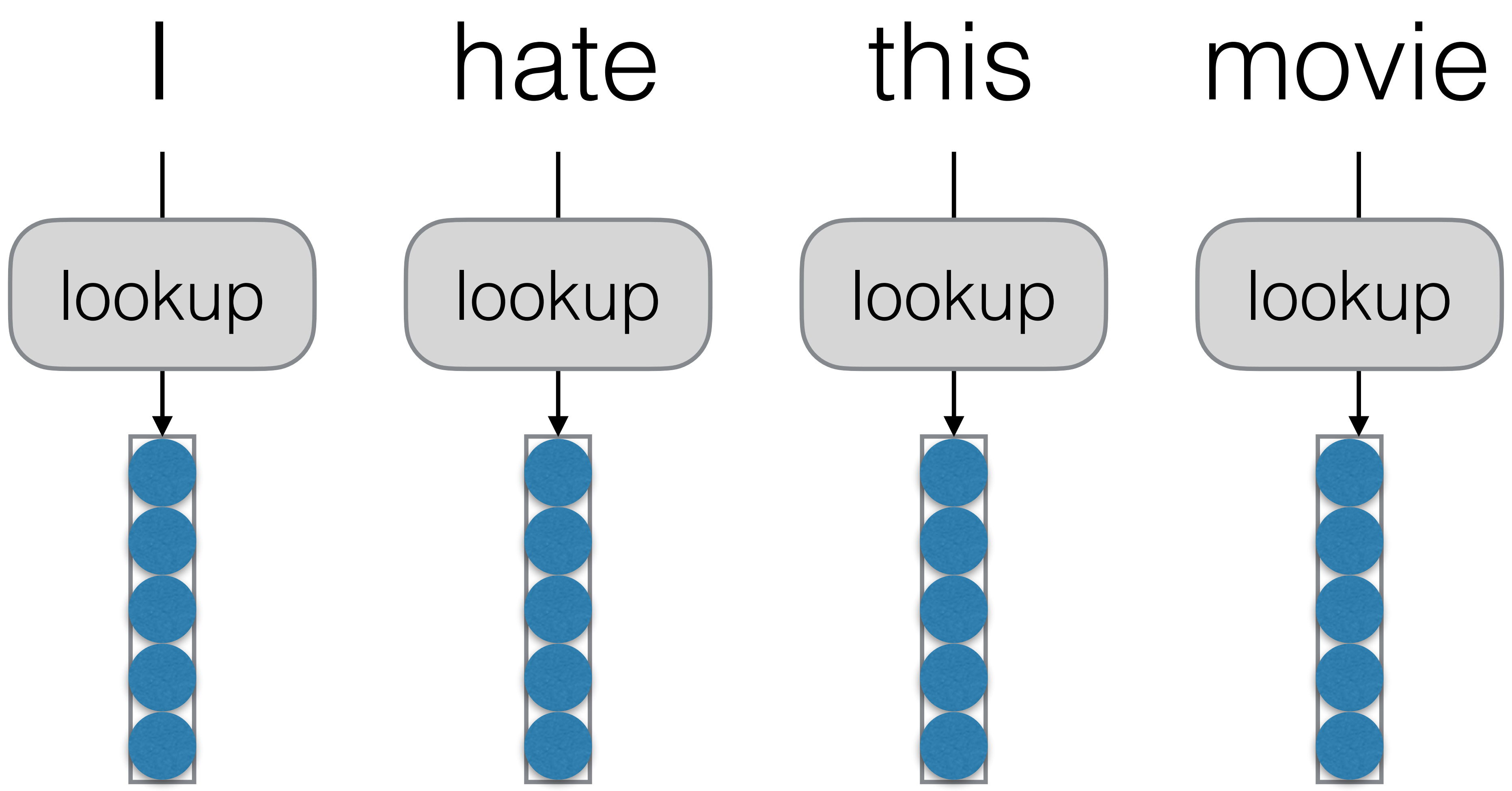
A First Try: Bag of Words (BOW)

Each word has a vector of weights for each tag



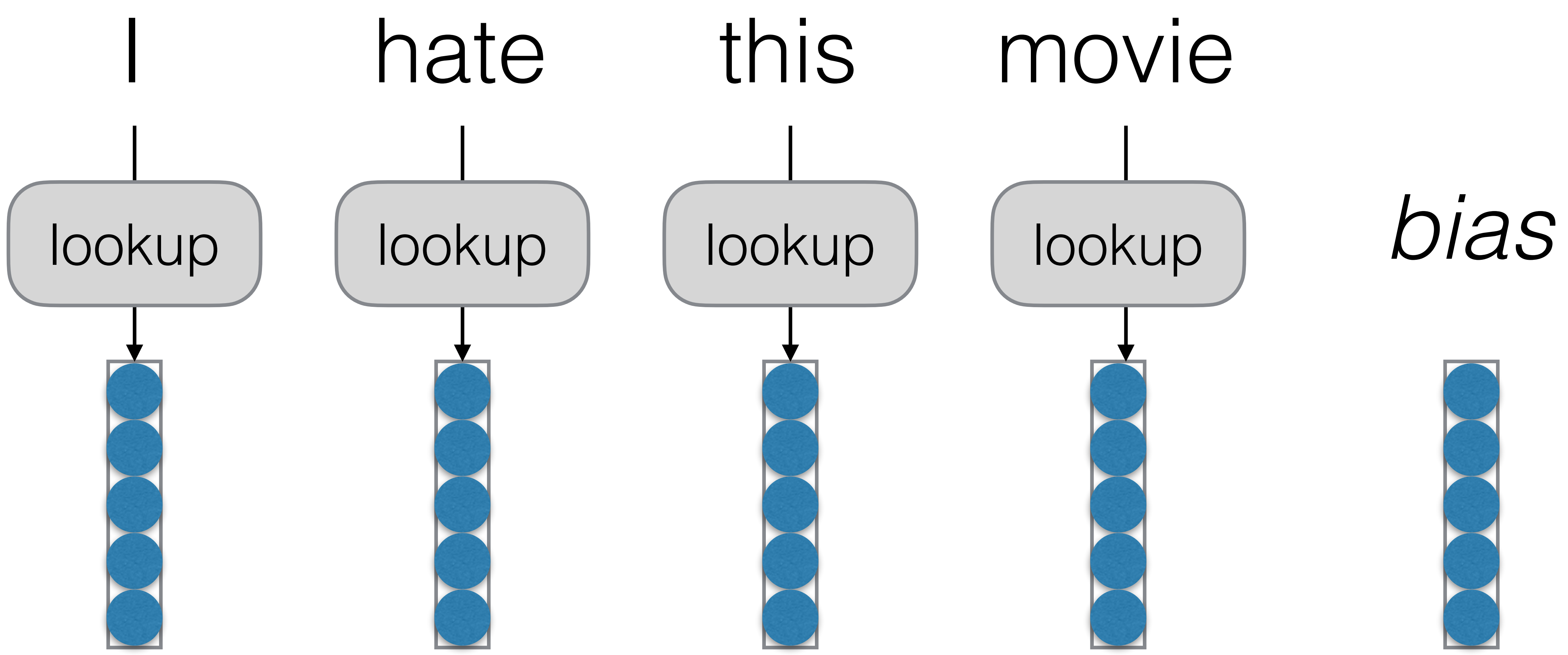
A First Try:
Bag of Words (BOW)

Each word has a vector of weights for each tag



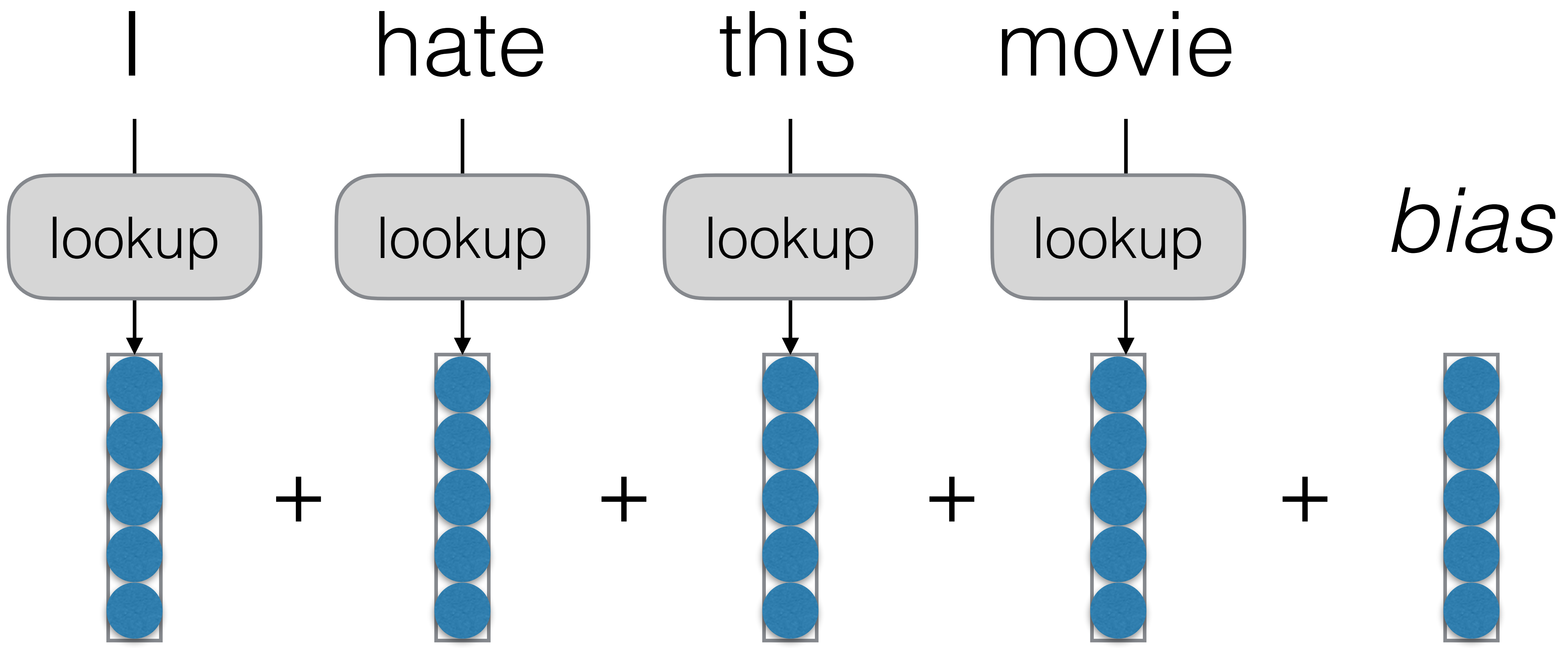
A First Try:
Bag of Words (BOW)

Each word has a vector of weights for each tag



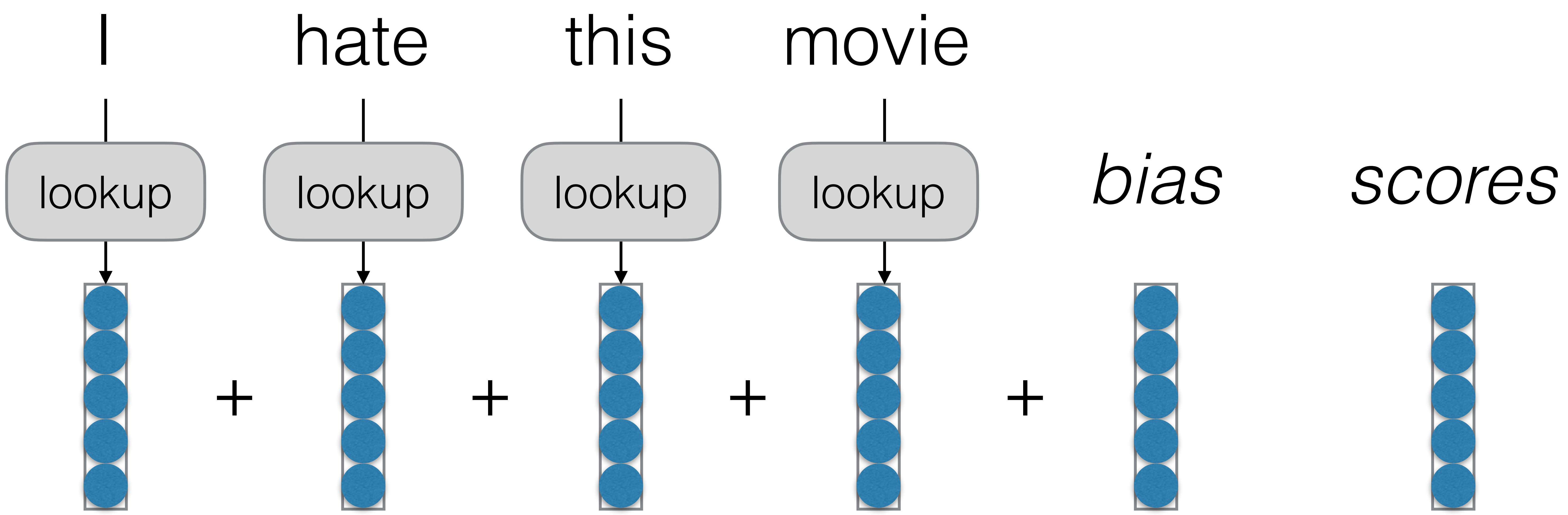
A First Try: Bag of Words (BOW)

Each word has a vector of weights for each tag



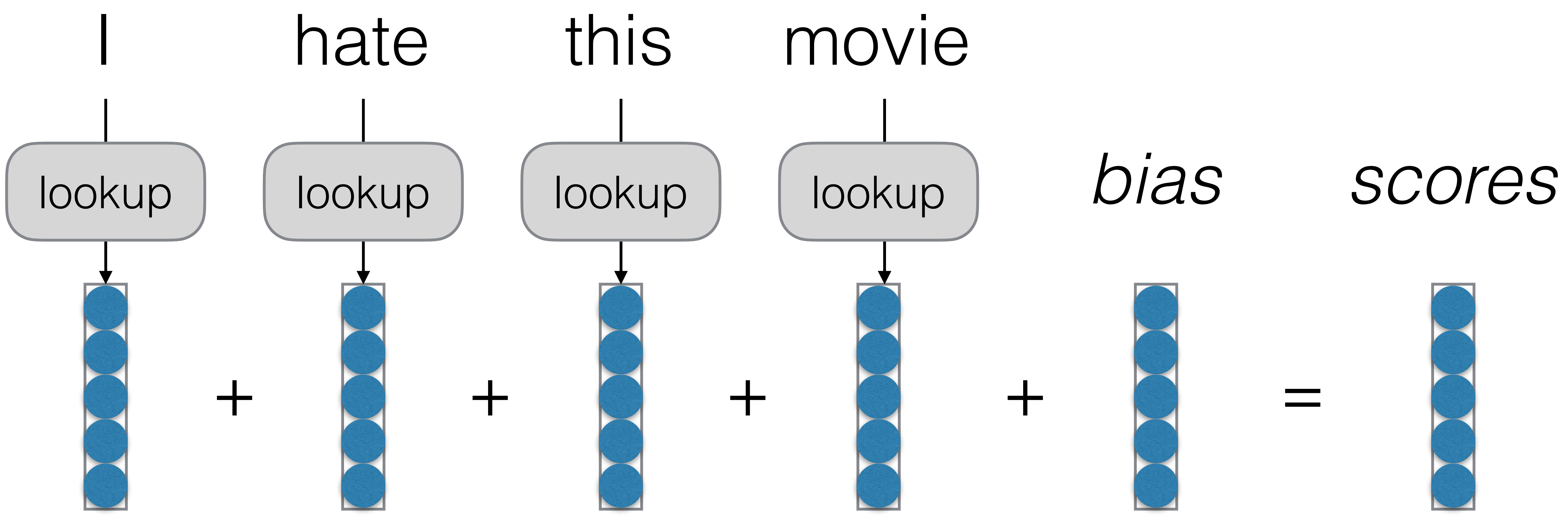
A First Try:
Bag of Words (BOW)

Each word has a vector of weights for each tag



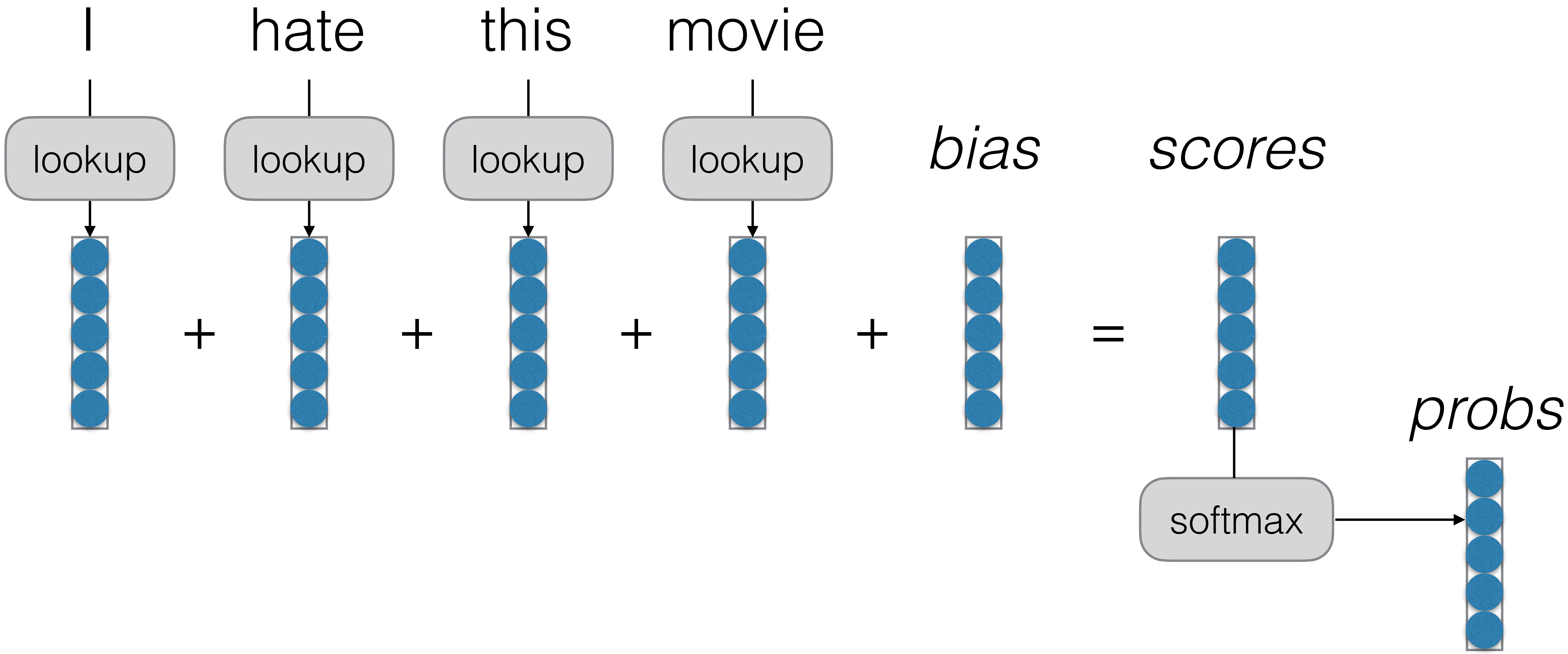
A First Try: Bag of Words (BOW)

Each word has a vector of weights for each tag



A First Try: Bag of Words (BOW)

Each word has a vector of weights for each tag



What do Our Vectors Represent?

What do Our Vectors Represent?

Each word has its own 5 elements corresponding to [very good, good, neutral, bad, very bad]

What do Our Vectors Represent?

Each word has its own 5 elements corresponding to [very good, good, neutral, bad, very bad]

"hate" will have a high value for "very bad", etc.

Build It, Break It

Build It, Break It

I don’t love this movie

Build It, Break It

I don't love this movie

very good
good
neutral
bad
very bad

Build It, Break It

I don't love this movie

very good
good
neutral
bad
very bad

Build It, Break It

I don’t love this movie

very good
good
neutral
bad
very bad

There’s nothing I don’t love about this movie

Build It, Break It

I don't love this movie

very good
good
neutral
bad
very bad

There's nothing I don't love about this movie

very good
good
neutral
bad
very bad

Build It, Break It

I don't love this movie

very good
good
neutral
bad
very bad

There's nothing I don't love about this movie

very good
good
neutral
bad
very bad

Combination Features

Combination Features

Does it contain “don’t” and “love”?

Combination Features

Does it contain “don’t” and “love”?

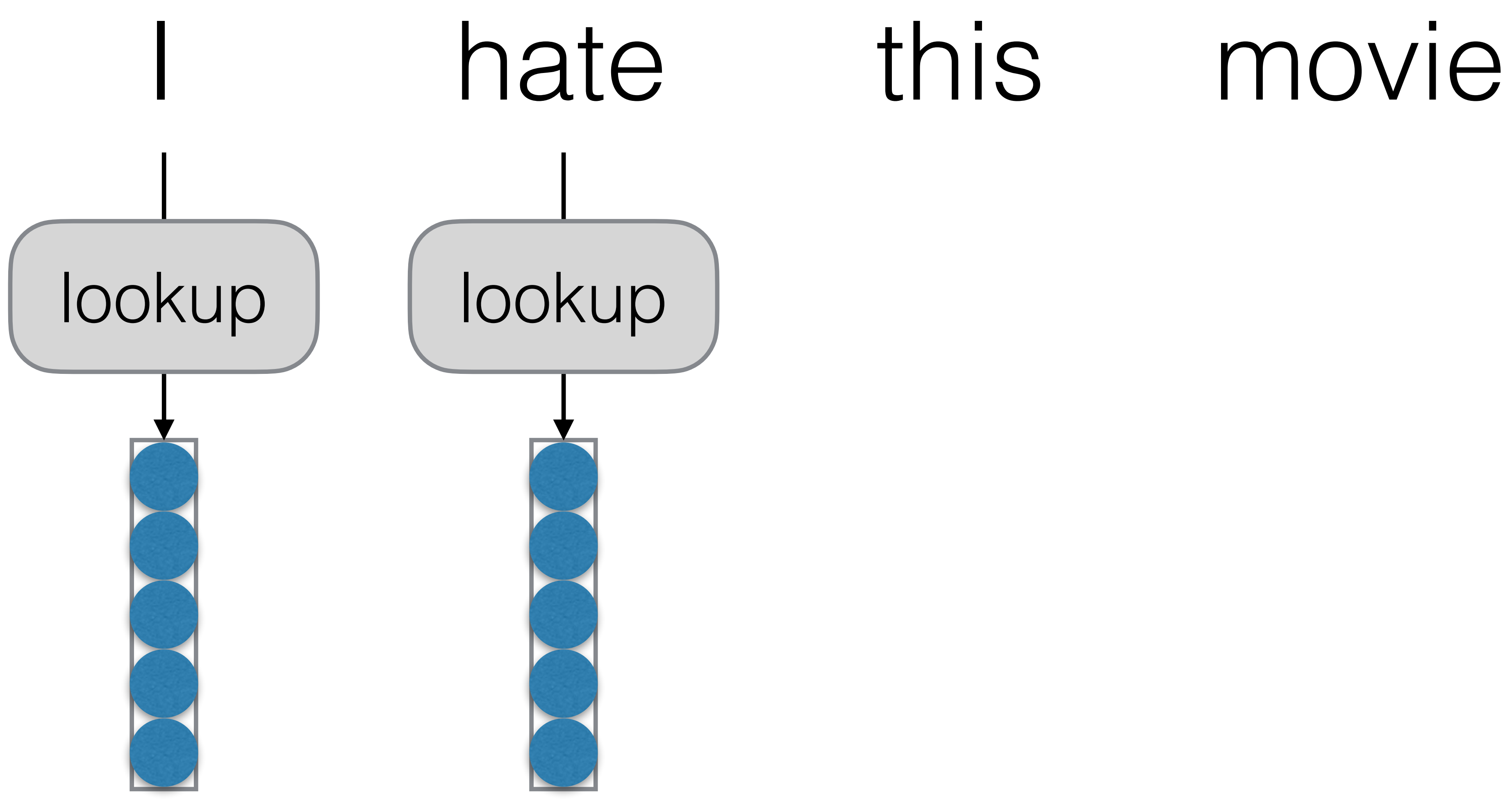
Does it contain “don’t”, “i”, “love”, and “nothing”?

Basic Idea of Neural Networks (for NLP Prediction Tasks)

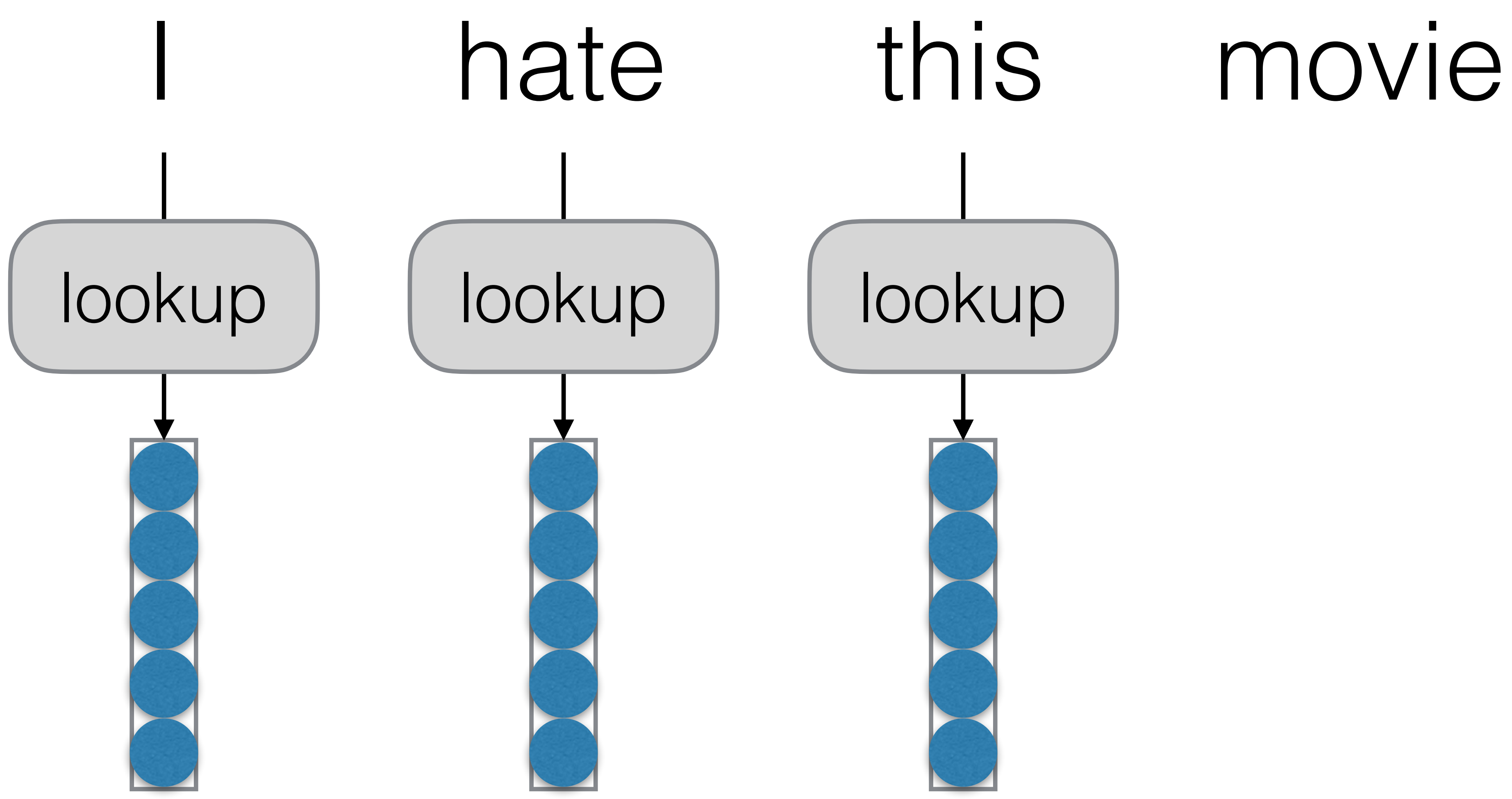
I hate this movie

Basic Idea of Neural Networks (for NLP Prediction Tasks)

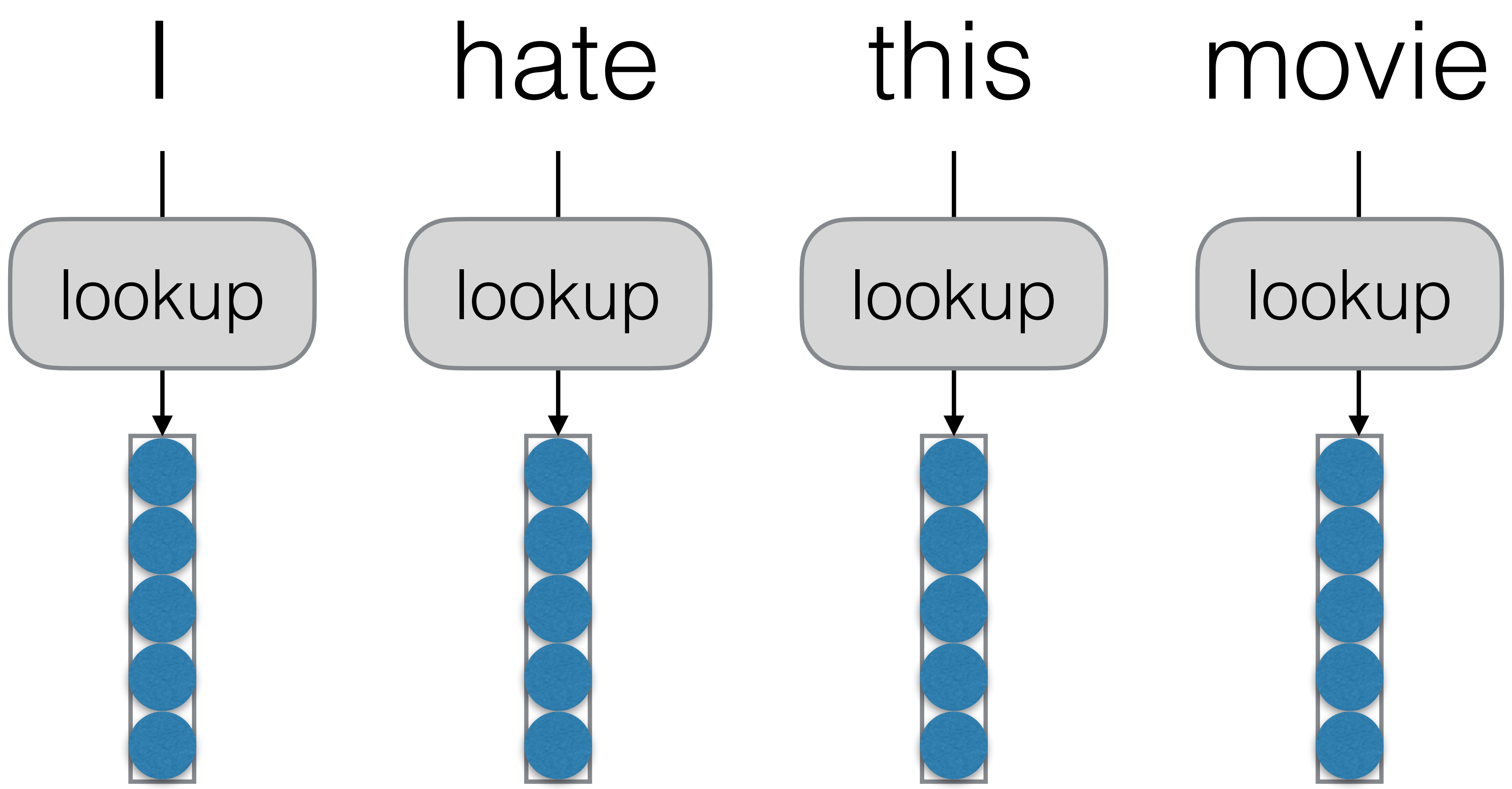
Basic Idea of Neural Networks (for NLP Prediction Tasks)



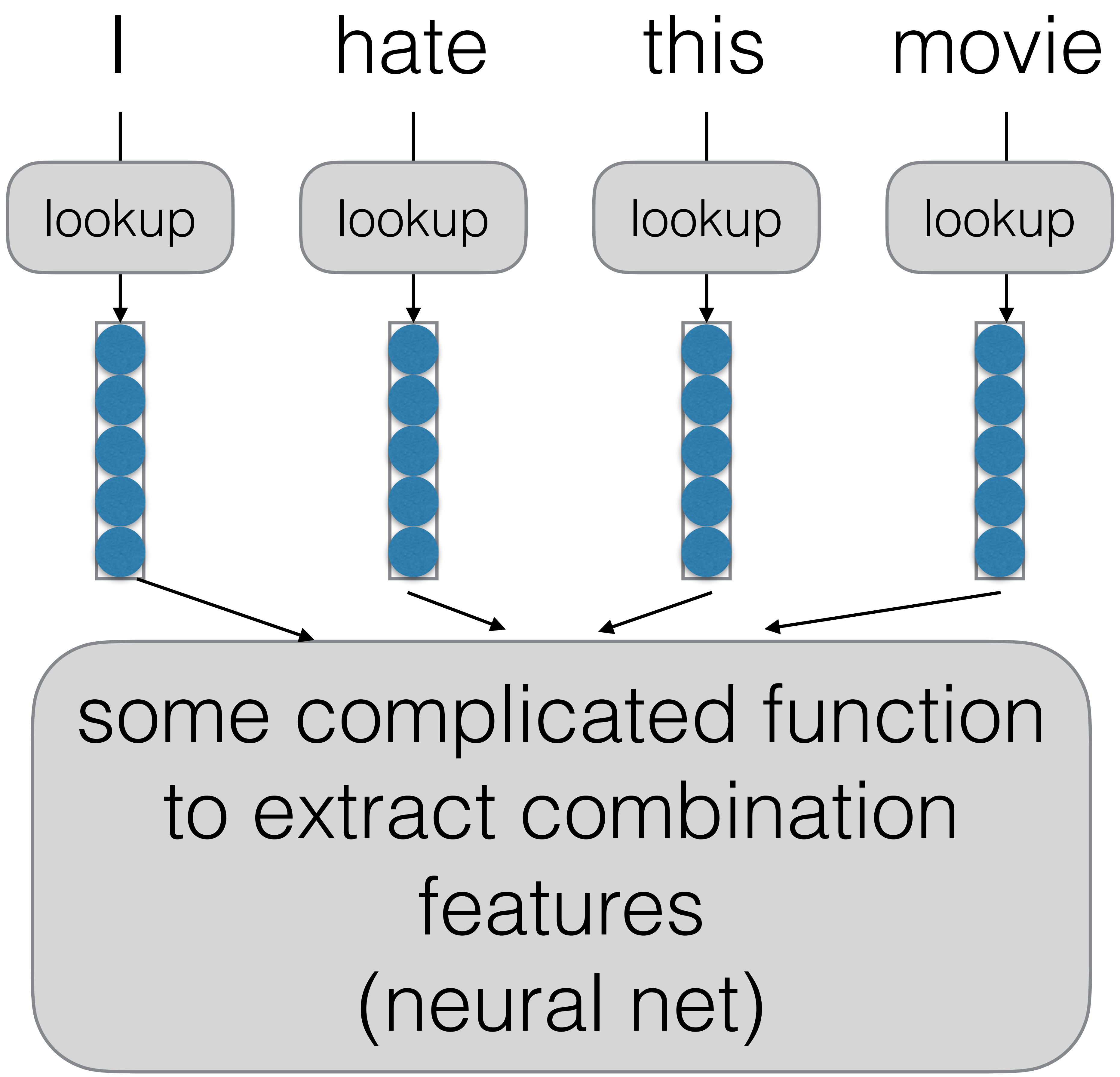
Basic Idea of Neural Networks (for NLP Prediction Tasks)



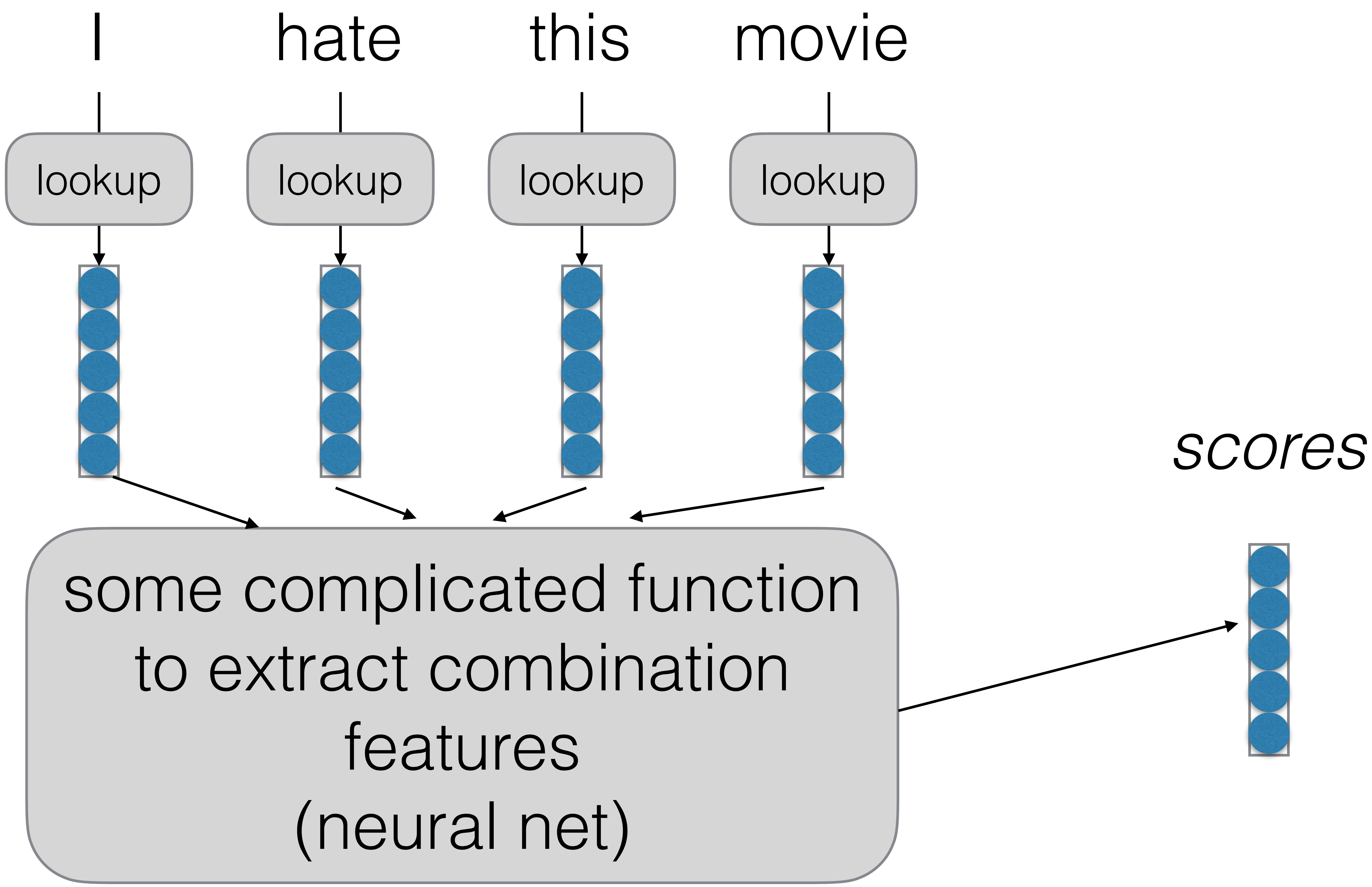
Basic Idea of Neural Networks (for NLP Prediction Tasks)



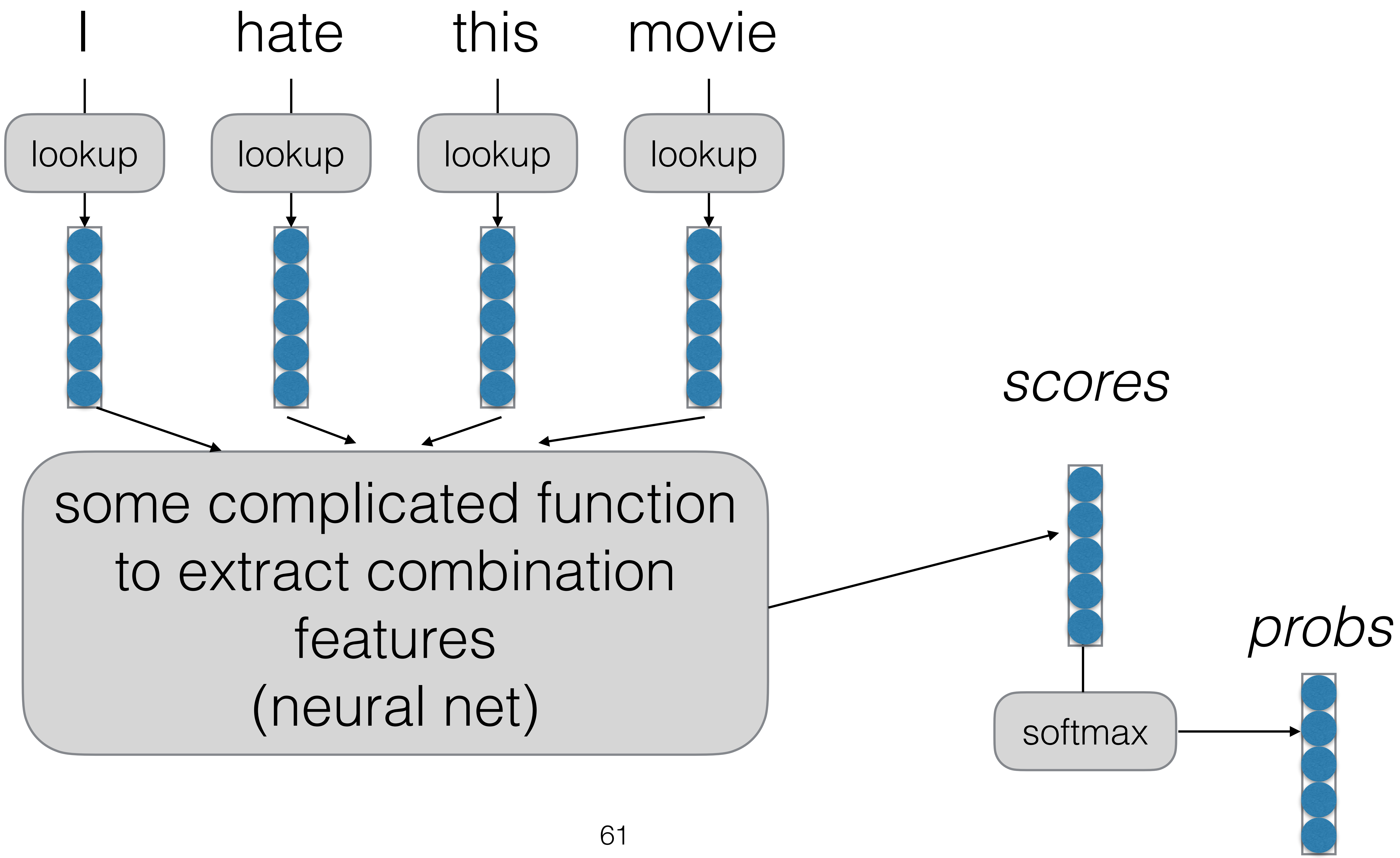
Basic Idea of Neural Networks (for NLP Prediction Tasks)



Basic Idea of Neural Networks (for NLP Prediction Tasks)



Basic Idea of Neural Networks (for NLP Prediction Tasks)



Continuous Bag of Words (CBOW)

Each word has a feature vector, each feature has weights

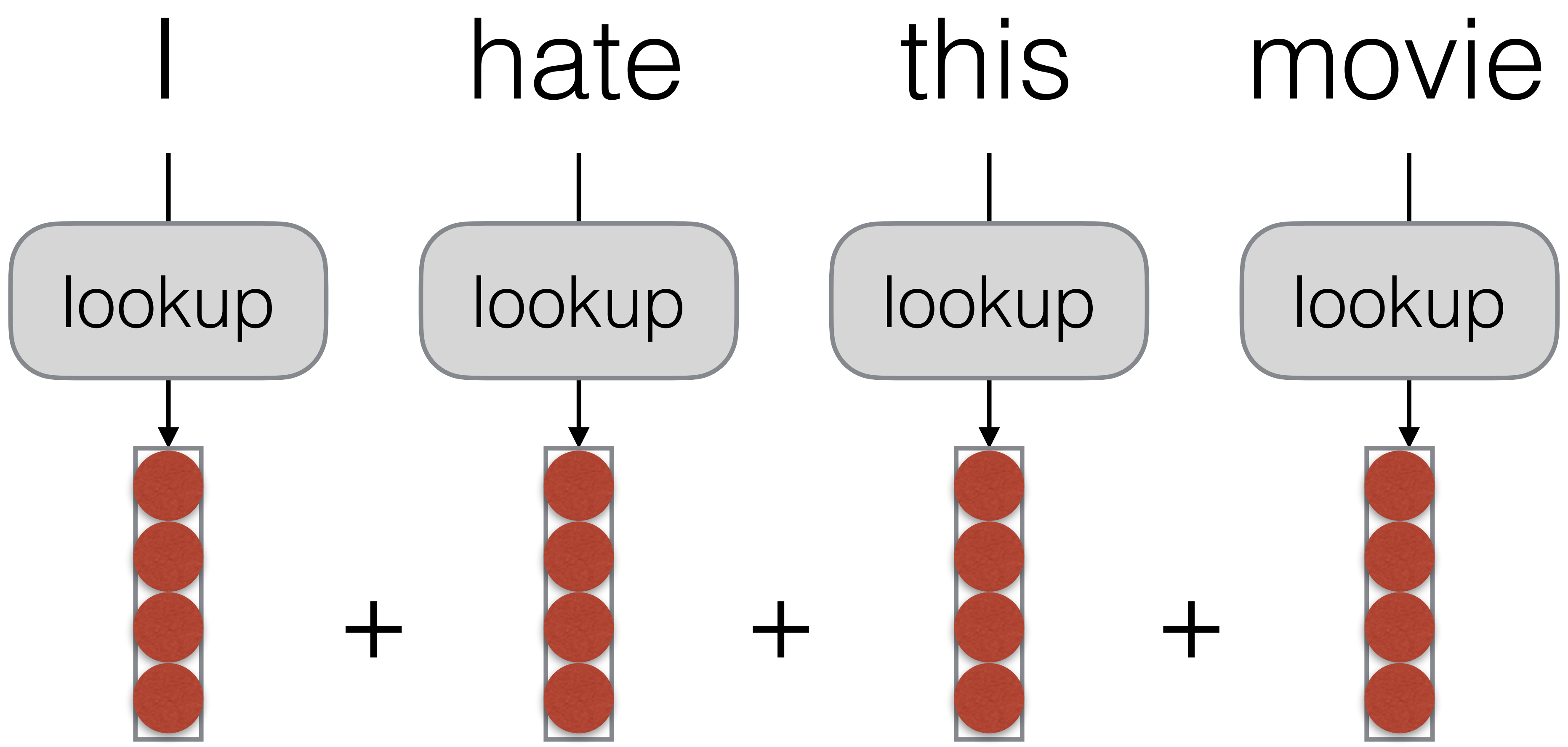
I hate this movie

Continuous Bag of Words (CBOW)

Each word has a feature vector, each feature has weights

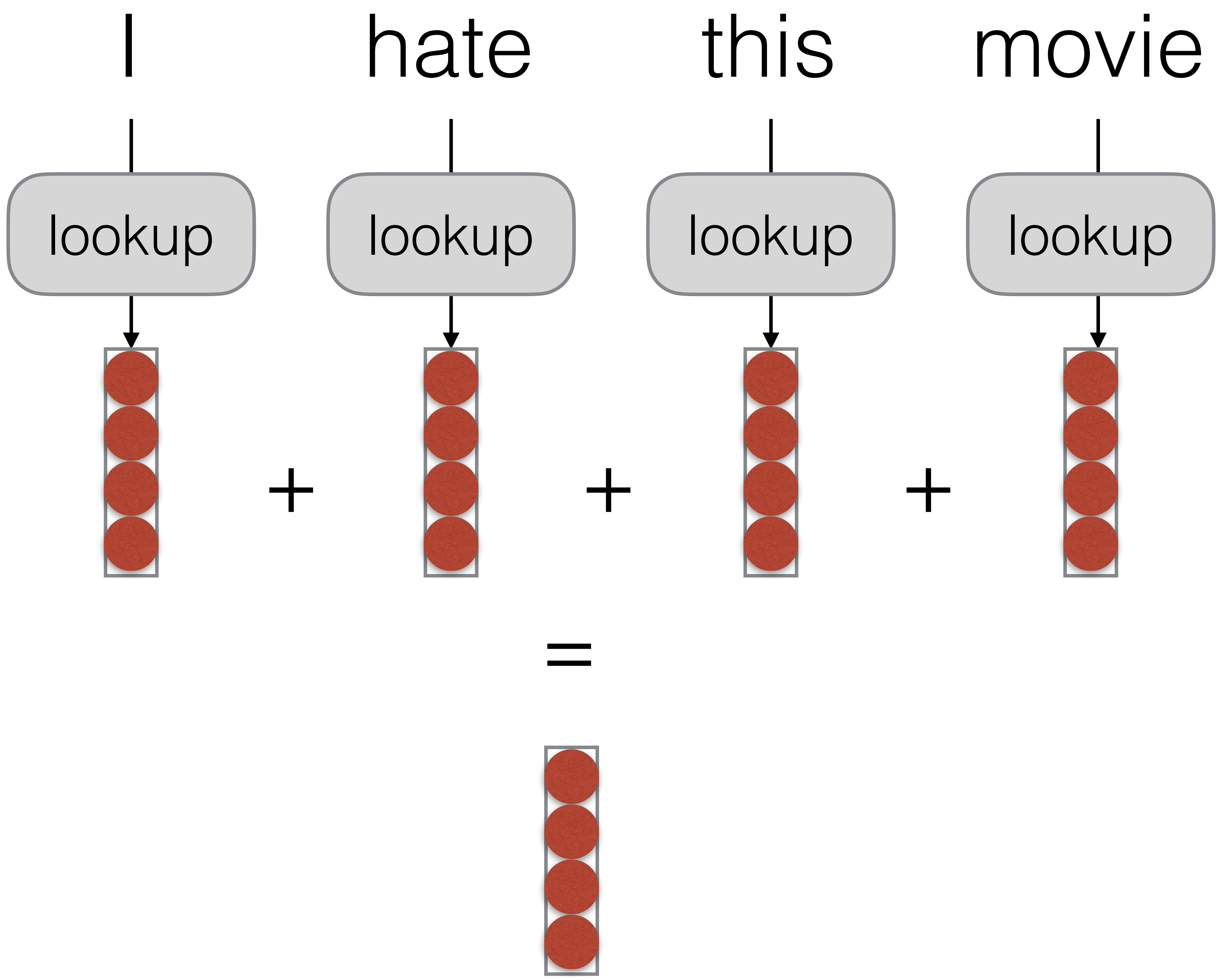
Continuous Bag of Words (CBOW)

Each word has a feature vector, each feature has weights



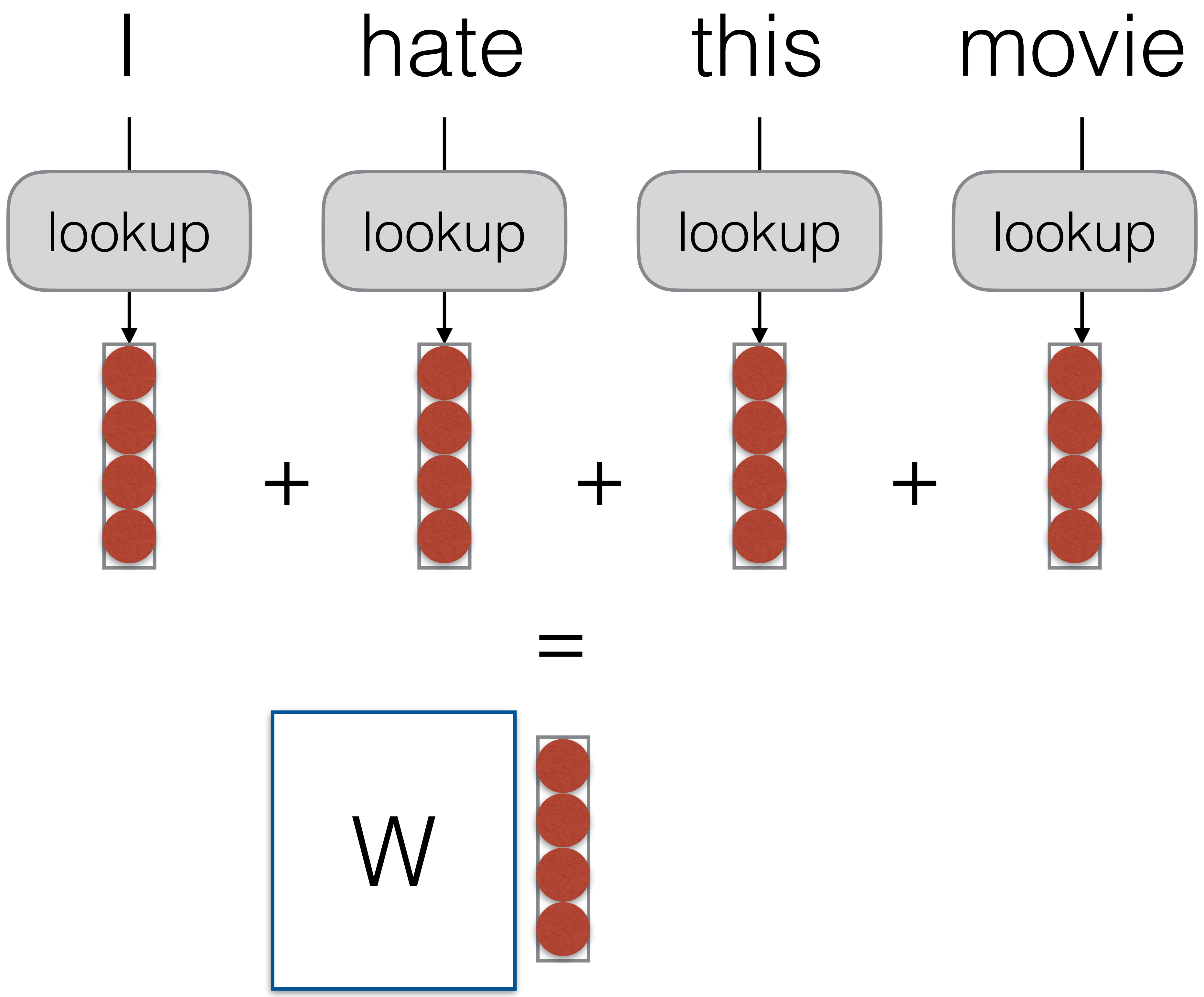
Continuous Bag of Words (CBOW)

Each word has a feature vector, each feature has weights



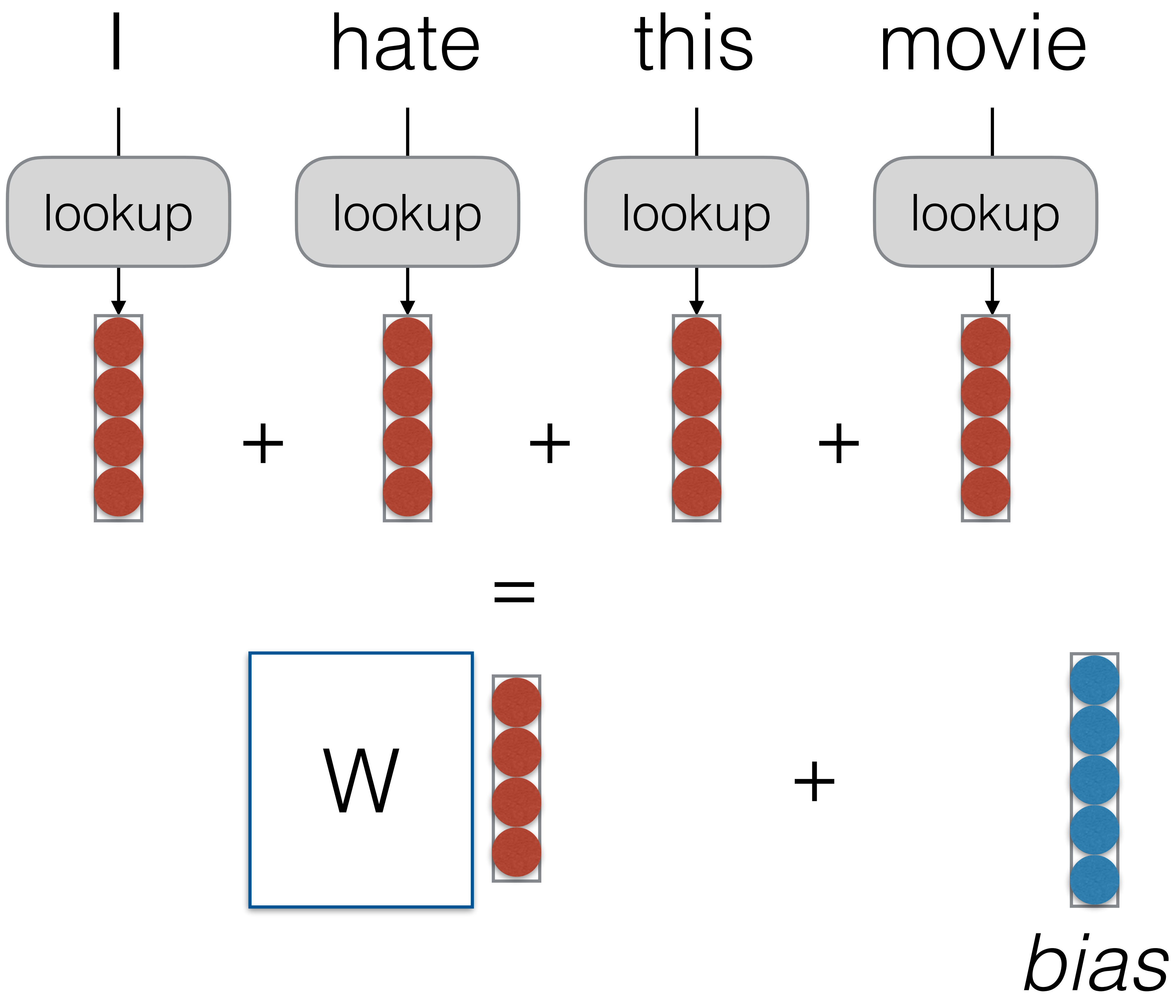
Continuous Bag of Words (CBOW)

Each word has a feature vector, each feature has weights



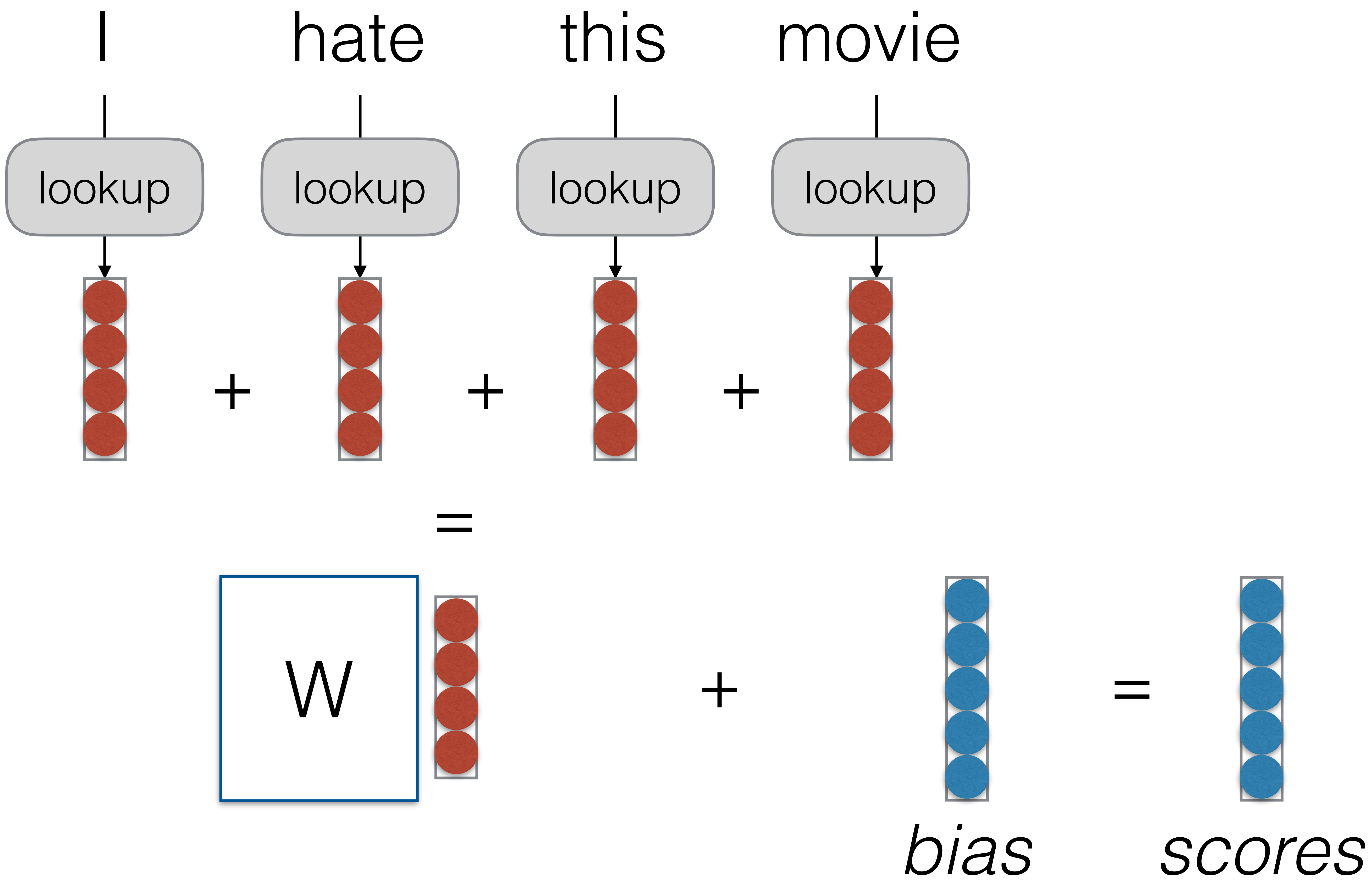
Continuous Bag of Words (CBOW)

Each word has a feature vector, each feature has weights



Continuous Bag of Words (CBOW)

Each word has a feature vector, each feature has weights



What do Our Vectors Represent?

What do Our Vectors Represent?

Each vector has “features” (e.g. is this an animate object? is this a positive word, etc.)

What do Our Vectors Represent?

Each vector has “features” (e.g. is this an animate object? is this a positive word, etc.)

We sum these features, then use these to make predictions

What do Our Vectors Represent?

Each vector has “features” (e.g. is this an animate object? is this a positive word, etc.)

We sum these features, then use these to make predictions

Still no combination features: only the expressive power of a linear model, but dimension reduced

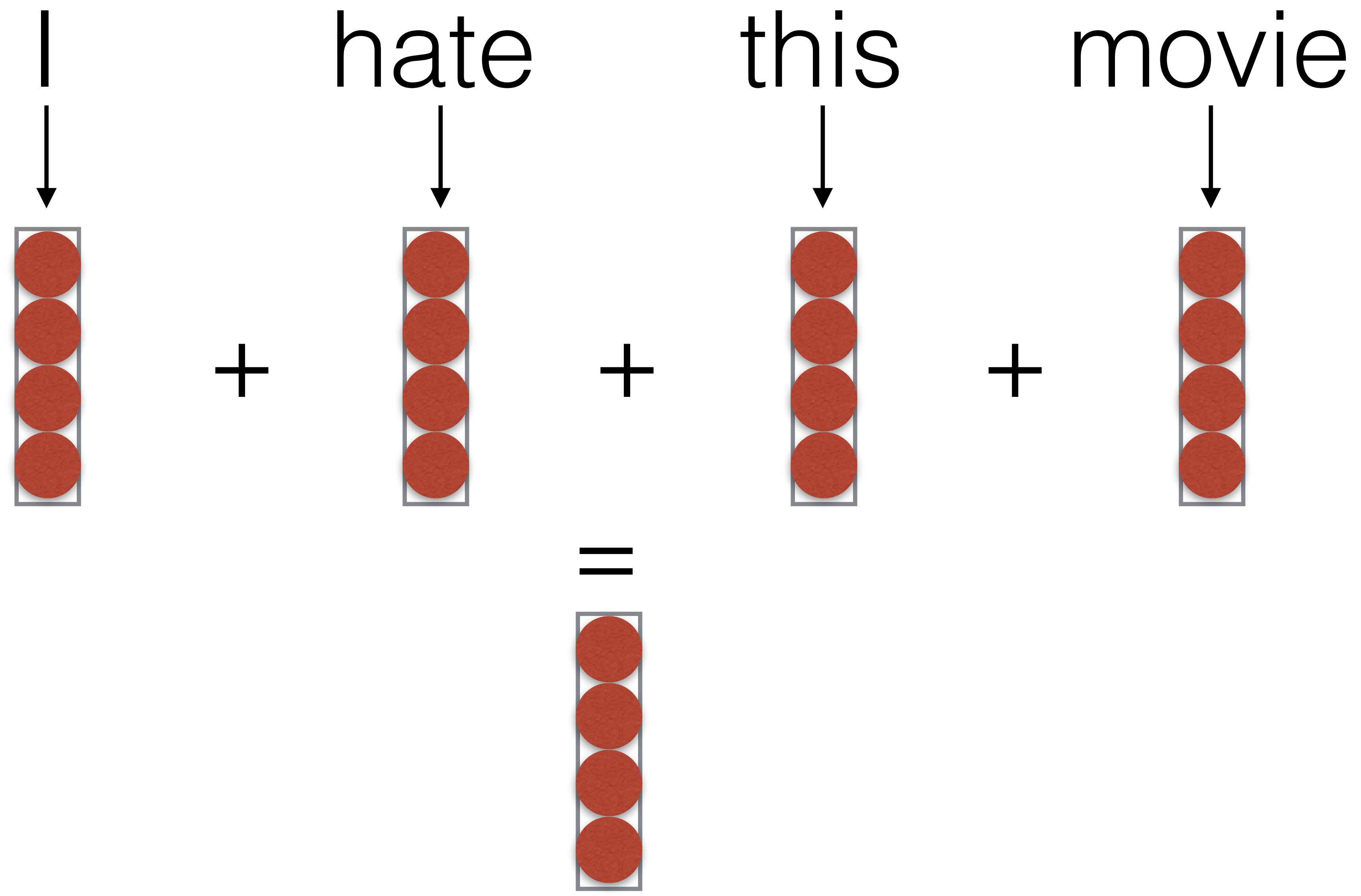
Deep CBOW

Add several feature transforms

I hate this movie

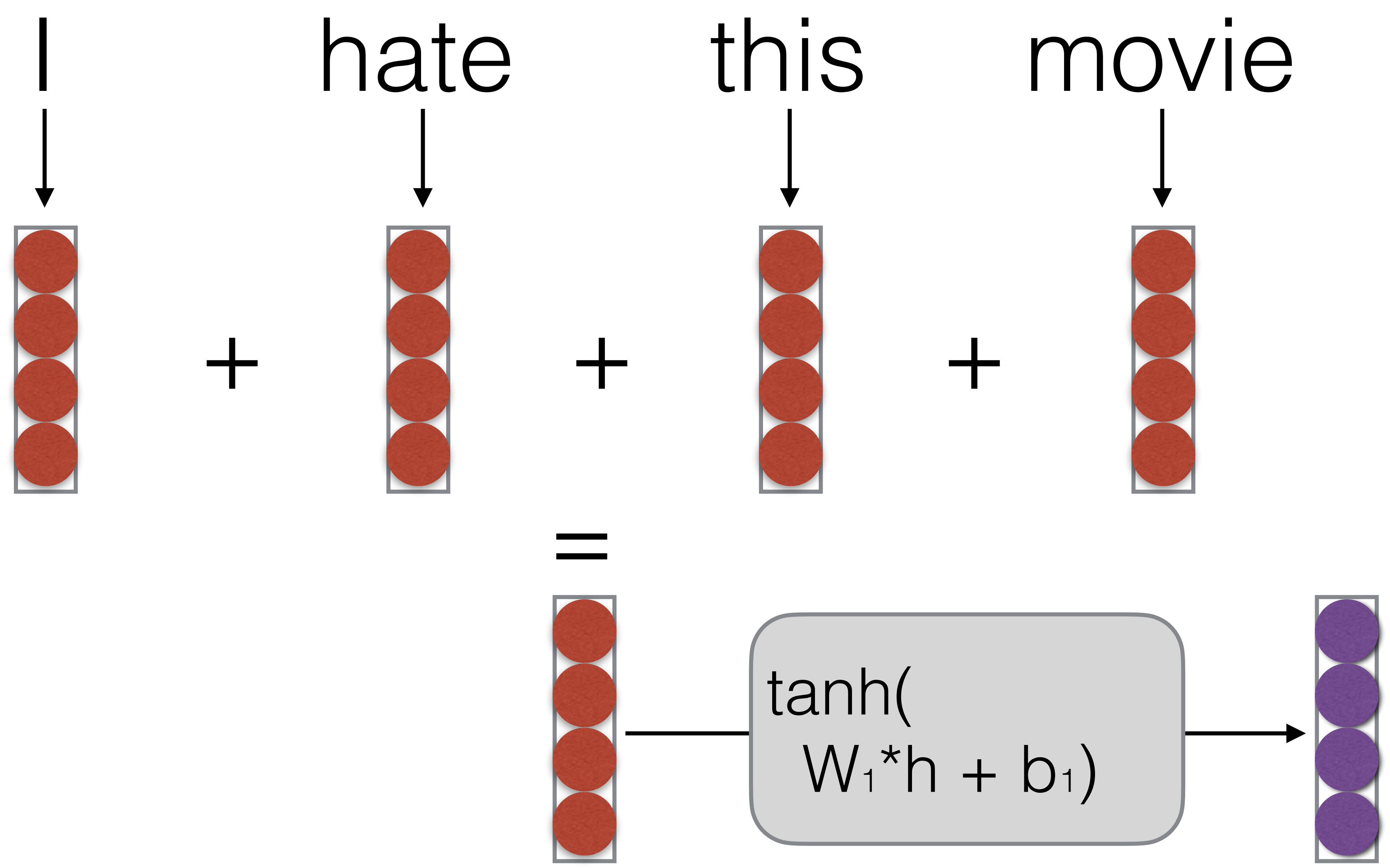
Deep CBOW

Add several feature transforms



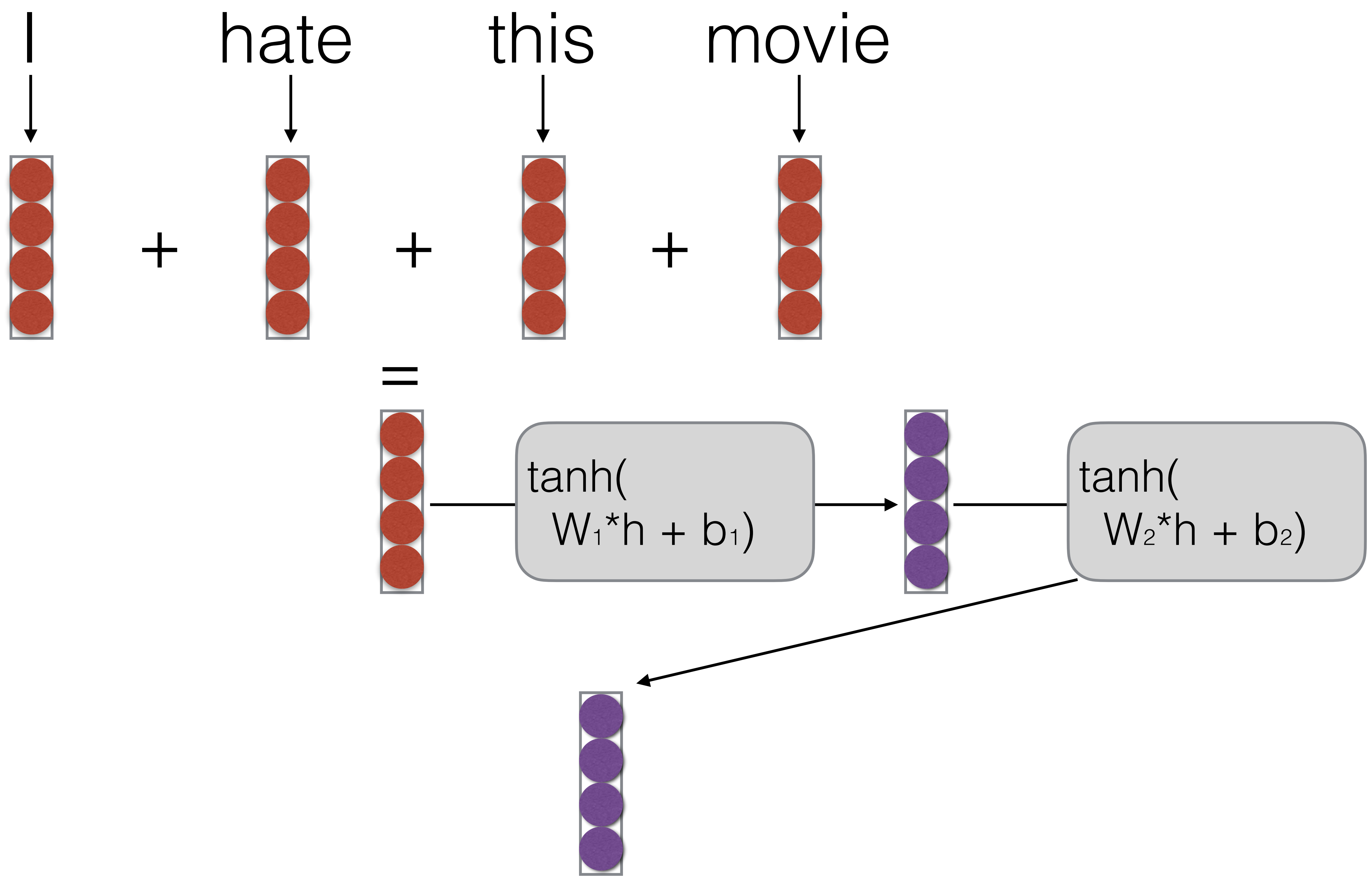
Deep CBOW

Add several feature transforms



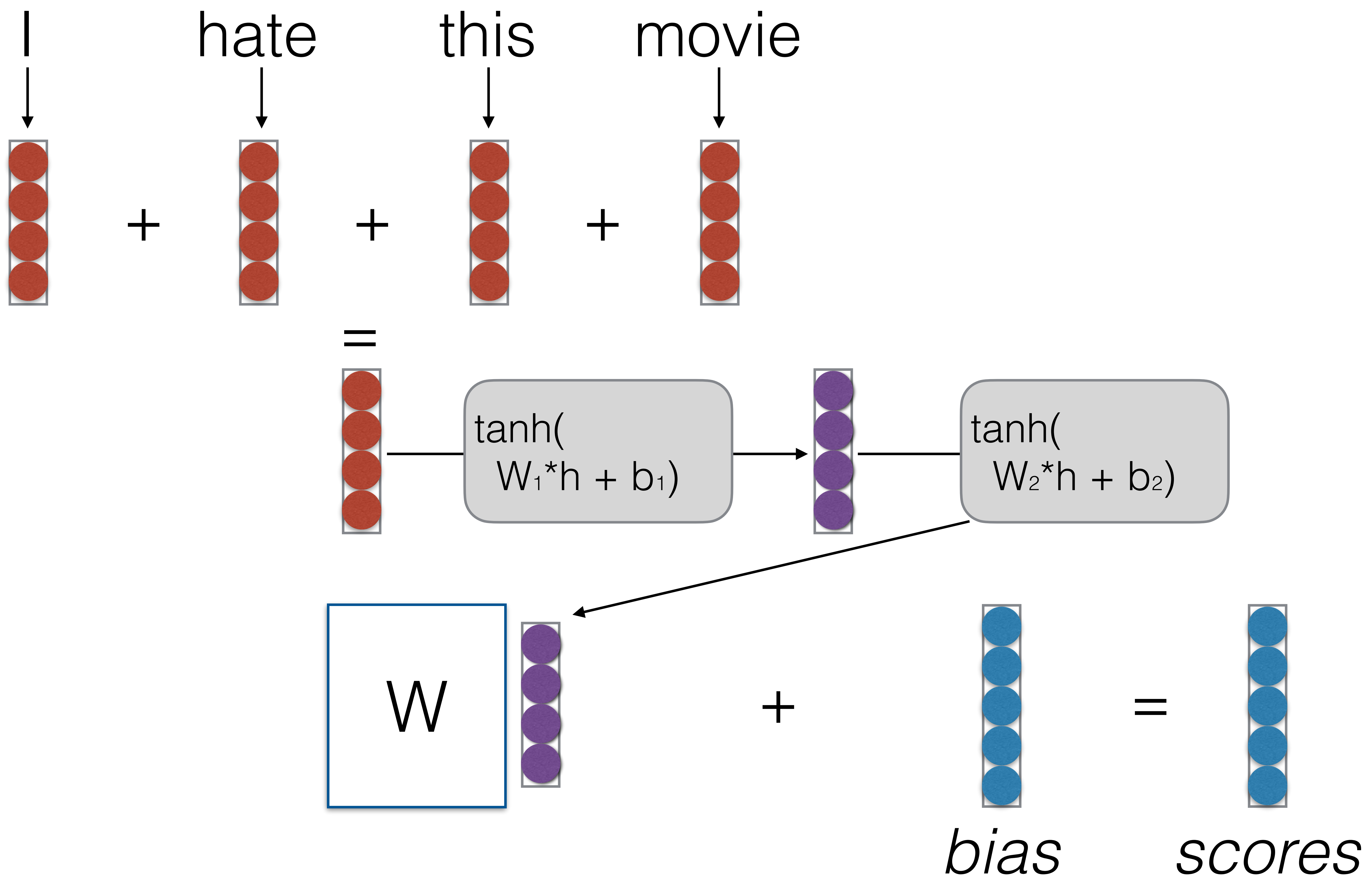
Deep CBOW

Add several feature transforms



Deep CBOW

Add several feature transforms



What do Our Vectors Represent?

What do Our Vectors Represent?

Now things are more interesting!

What do Our Vectors Represent?

Now things are more interesting!

We can learn feature combinations (a node in the second layer might be "feature 1 AND feature 5 are active")

What do Our Vectors Represent?

Now things are more interesting!

We can learn feature combinations (a node in the second layer might be "feature 1 AND feature 5 are active")

e.g. capture things such as "not" AND "hate"

What is a Neural Net?

Computation Graphs

“Neural” Nets

Original Motivation: The Brain

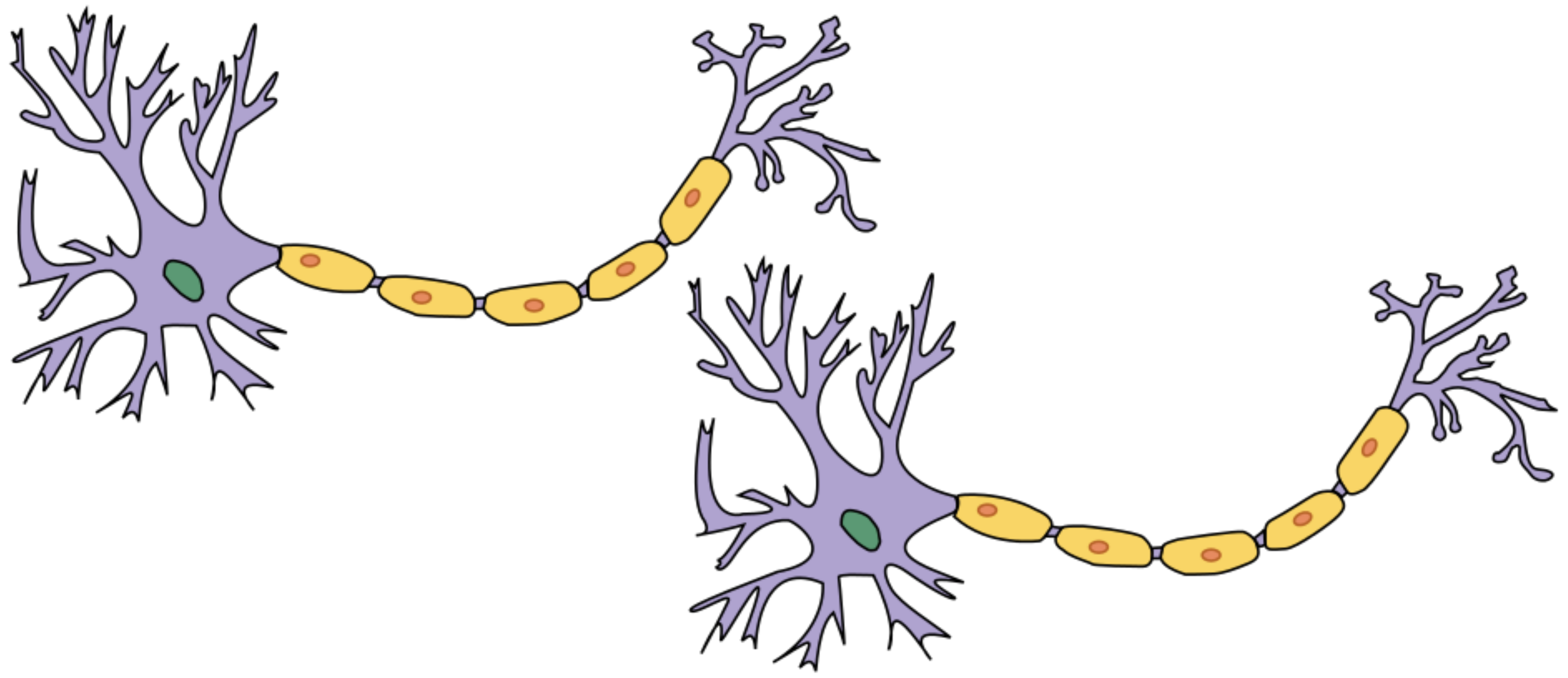
Current Implementation

Image credit: Wikipedia

“Neural” Nets

Original Motivation: The Brain

Original Motivation: Neurons in the Brain



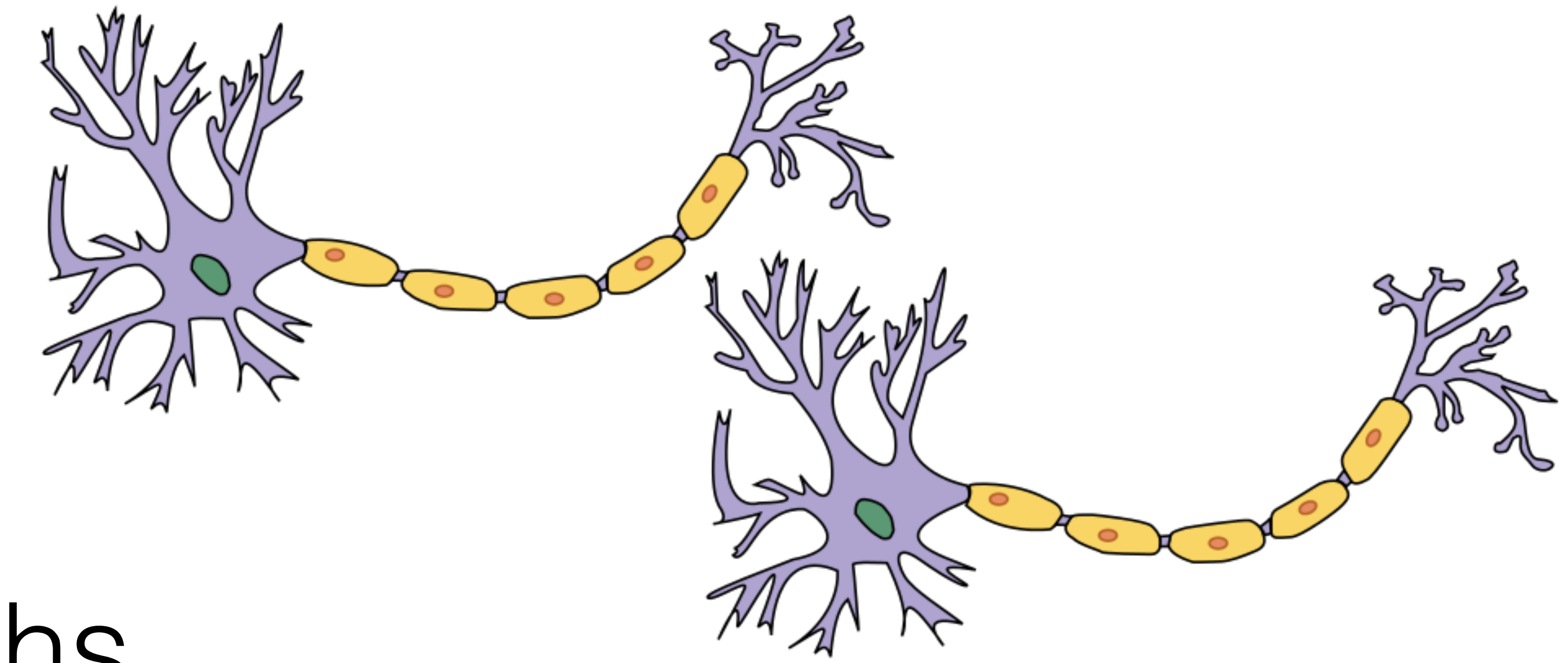
Current Implementation

Image credit: Wikipedia

"Neural" Nets

Original Motivation: The Brain

Original Motivation: Neurons in the Brain



Current Implementation

Current Conception: Computation Graphs

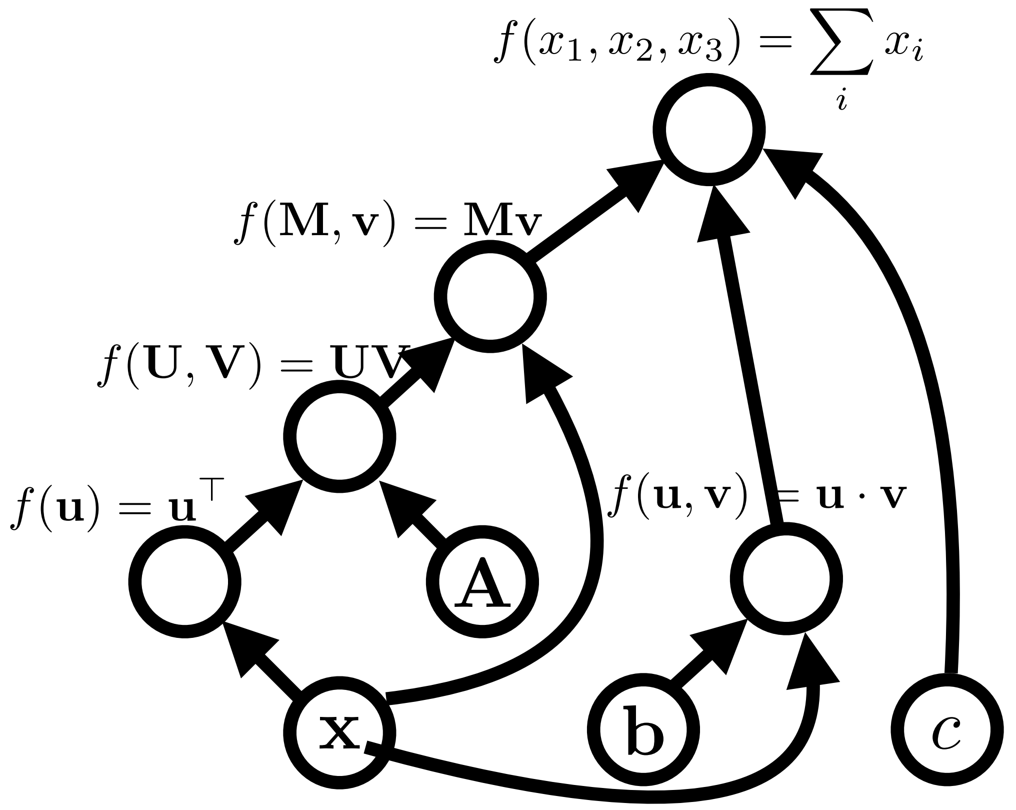


Image credit: Wikipedia

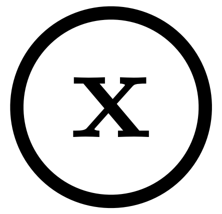
expression:

$$\mathbf{x}$$

expression:

$$\mathbf{x}$$

graph:



expression:

$$\mathbf{x}$$

graph:

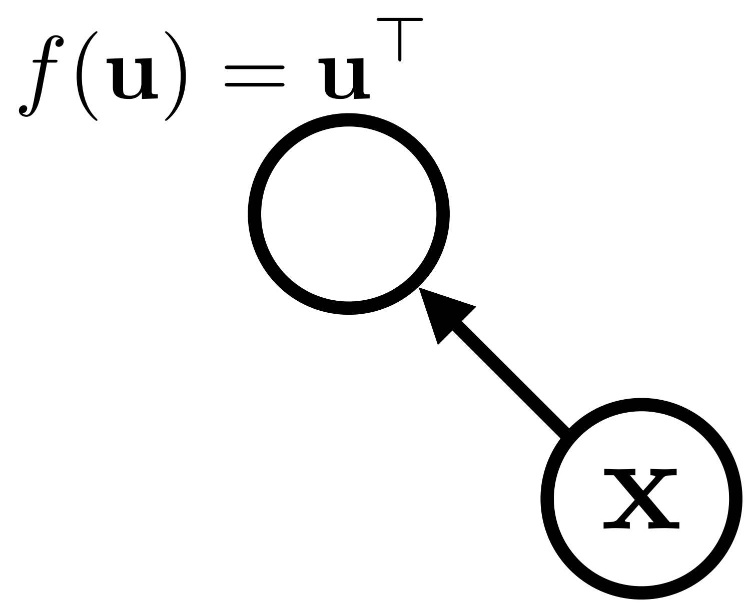
A **node** is a {tensor, matrix, vector, scalar} value

$$\mathbf{x}$$

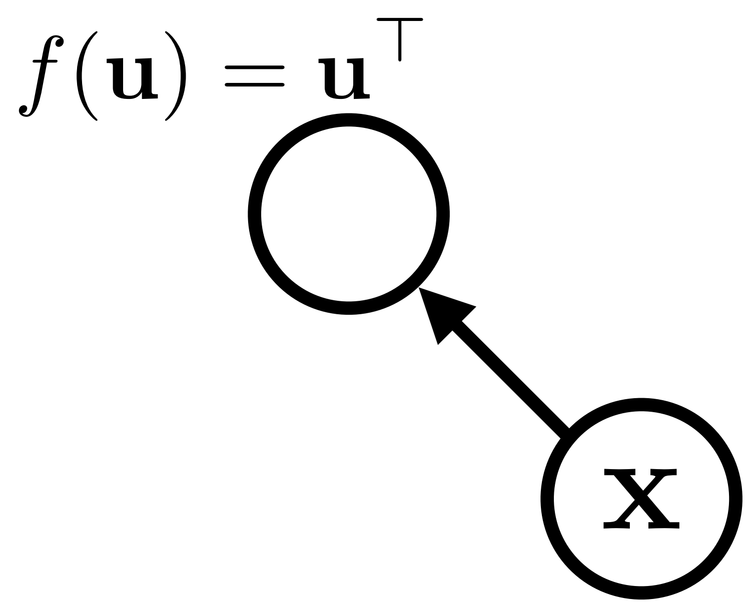
expression:

$$\mathbf{x}^\top$$

graph:

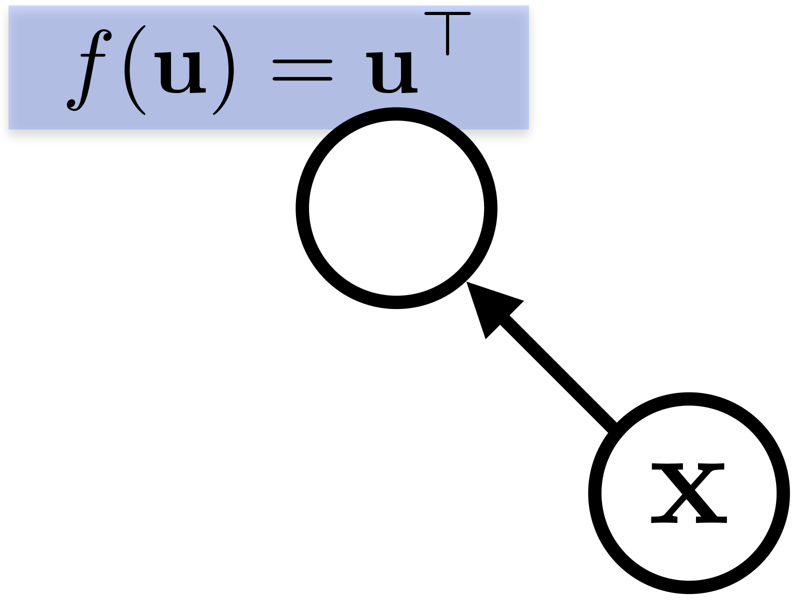


An **edge** represents a function argument (and also an data dependency). They are just pointers to nodes.



An **edge** represents a function argument (and also an data dependency). They are just pointers to nodes.

A **node** with an incoming **edge** is a **function** of that edge’s tail node.

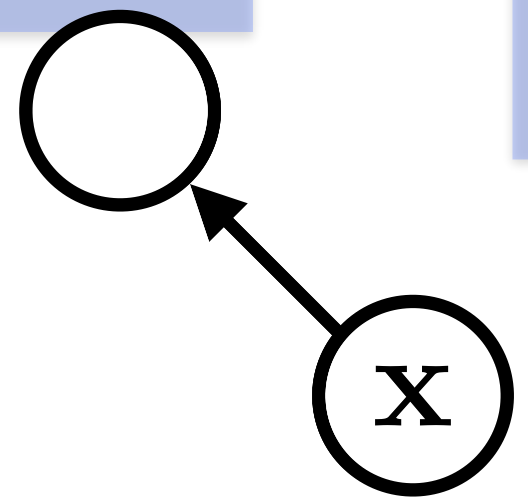


An **edge** represents a function argument (and also an data dependency). They are just pointers to nodes.

A **node** with an incoming **edge** is a **function** of that edge's tail node.

A **node** knows how to compute its value and the *value of its derivative w.r.t each argument (edge) times a derivative of an arbitrary input* $\frac{\partial \mathcal{F}}{\partial f(\mathbf{u})}$.

$$f(\mathbf{u}) = \mathbf{u}^\top$$



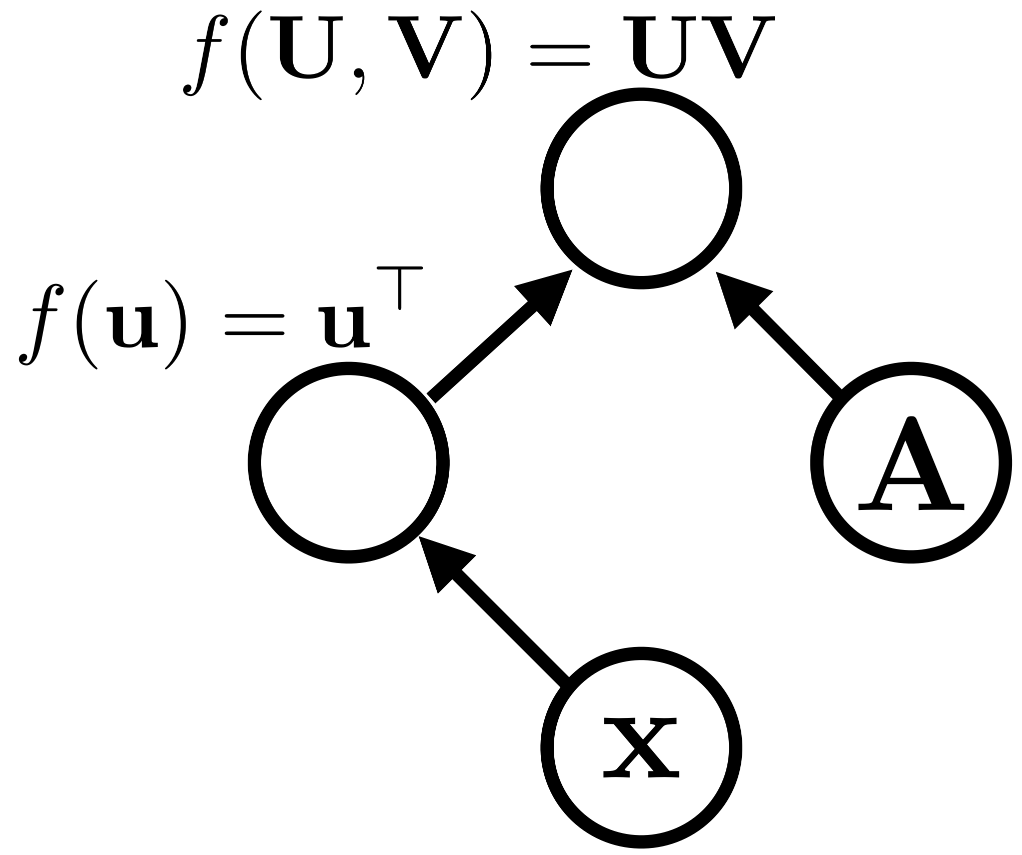
$$\frac{\partial f(\mathbf{u})}{\partial \mathbf{u}} \frac{\partial \mathcal{F}}{\partial f(\mathbf{u})} = \left(\frac{\partial \mathcal{F}}{\partial f(\mathbf{u})} \right)^\top$$

expression:

$$\mathbf{x}^\top \mathbf{A}$$

graph:

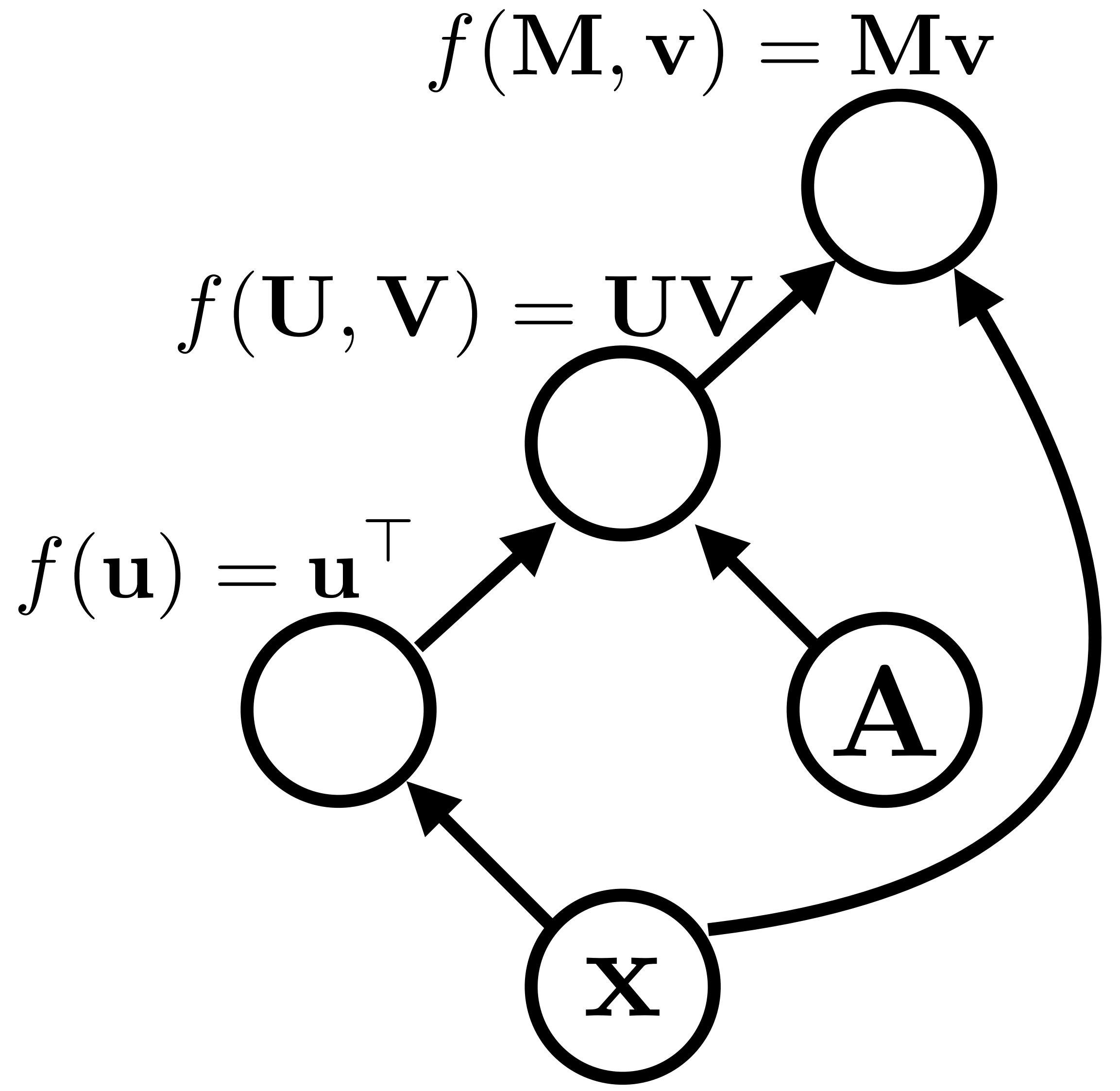
Functions can be nullary, unary, binary, … *n*-ary. Often they are unary or binary.



expression:

$$\mathbf{x}^\top \mathbf{A} \mathbf{x}$$

graph:

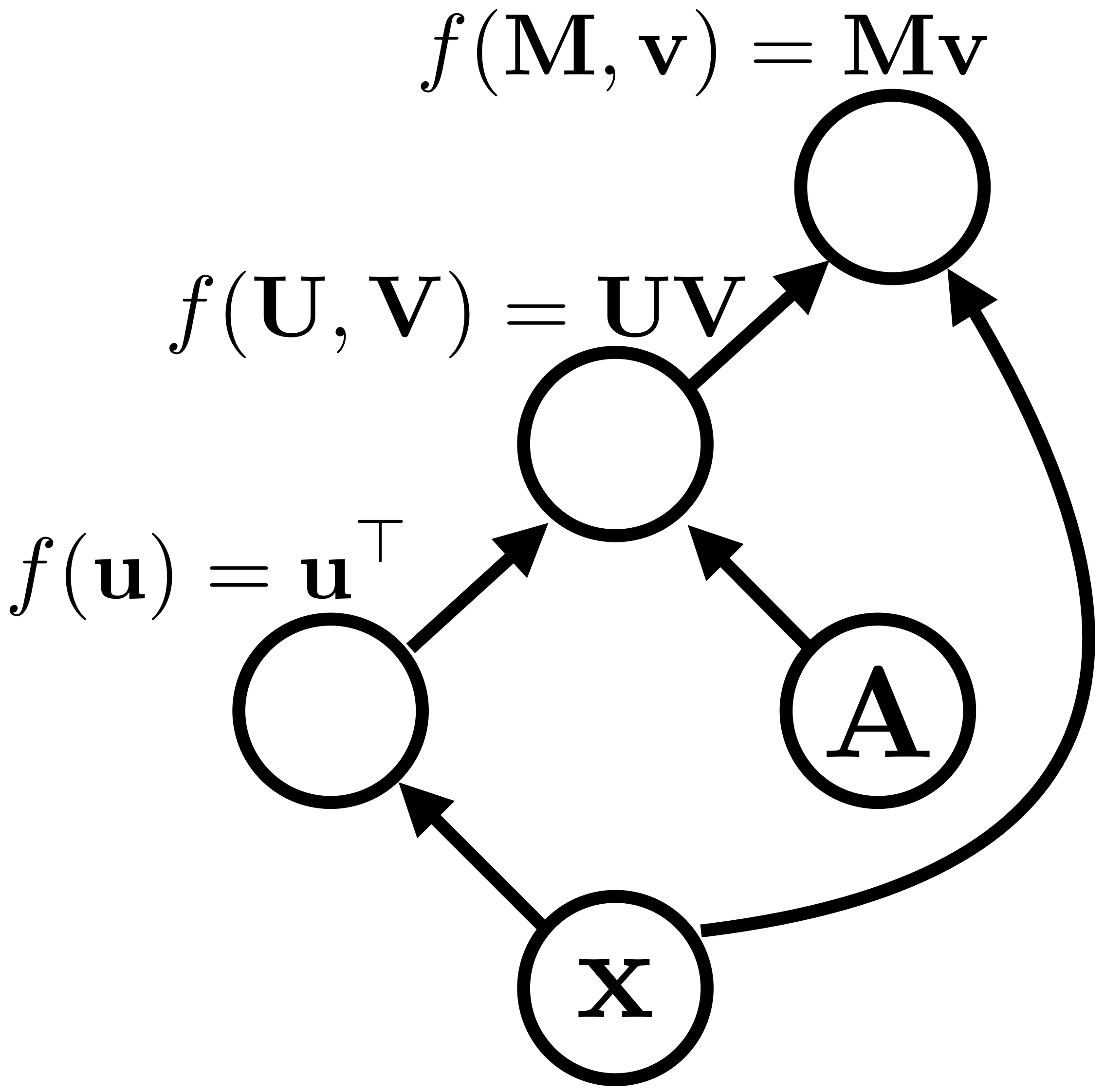


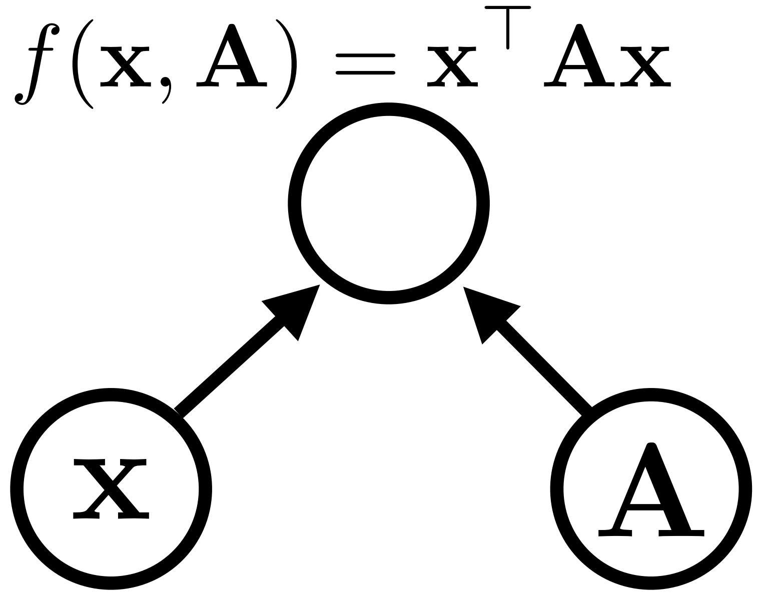
Computation graphs are directed and acyclic

expression:

$$\mathbf{x}^\top \mathbf{A}\mathbf{x}$$

graph:





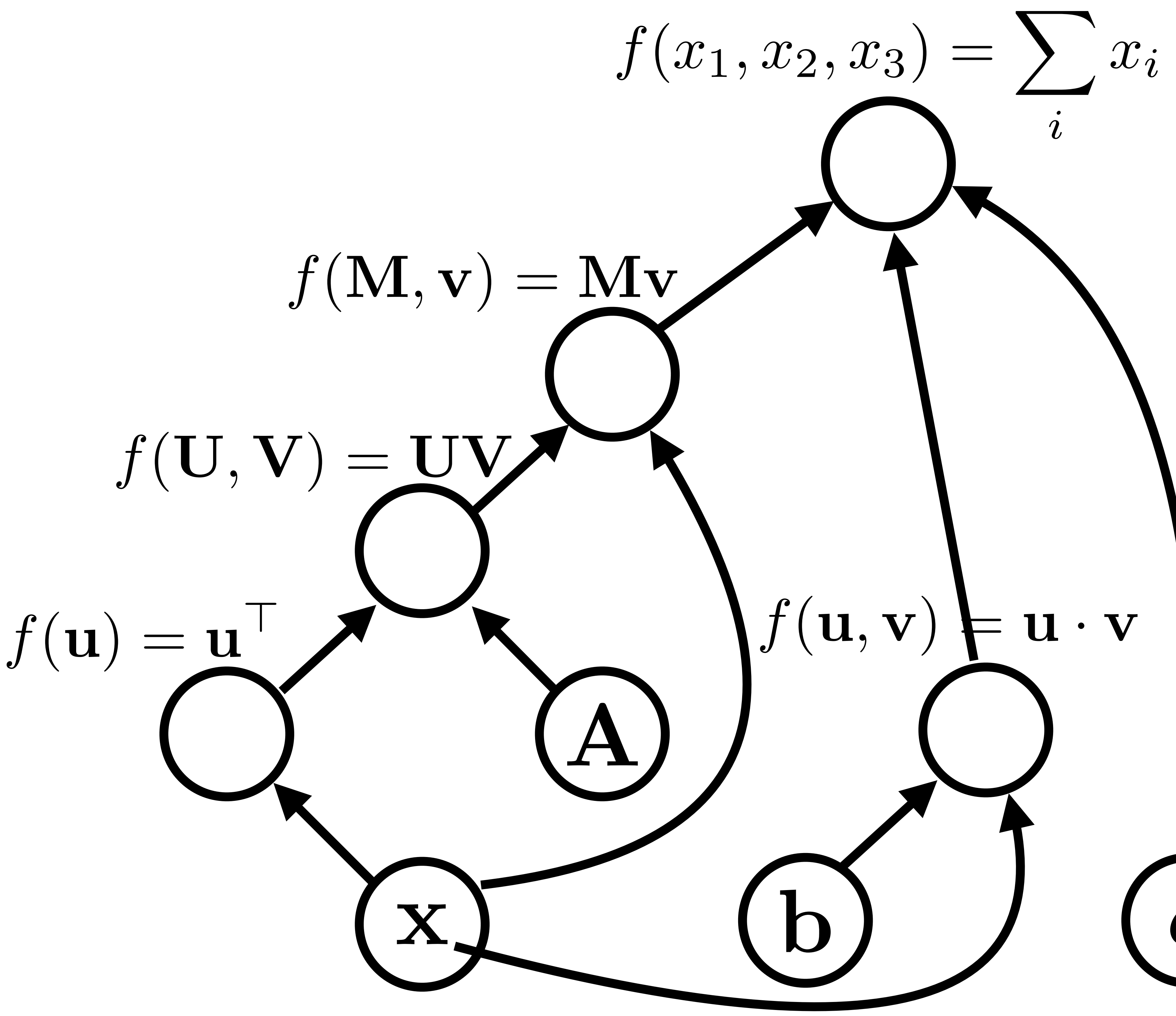
$$\frac{\partial f(\mathbf{x}, \mathbf{A})}{\partial \mathbf{x}} = (\mathbf{A}^\top + \mathbf{A})\mathbf{x}$$

$$\frac{\partial f(\mathbf{x}, \mathbf{A})}{\partial \mathbf{A}} = \mathbf{x}\mathbf{x}^\top$$

expression:

$$\mathbf{x}^\top \mathbf{A}\mathbf{x} + \mathbf{b} \cdot \mathbf{x} + c$$

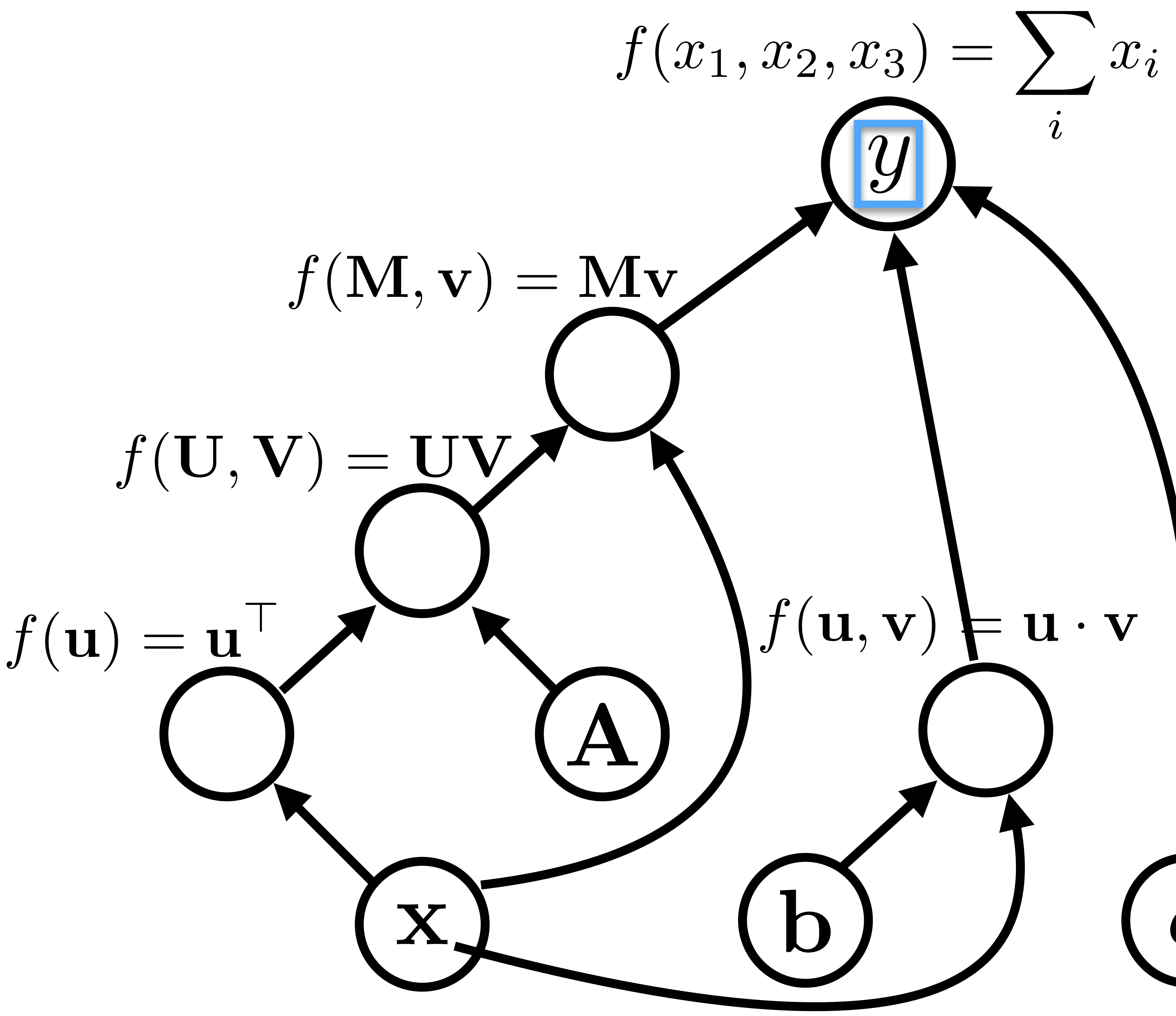
graph:



expression:

$$\boxed{y =}\, \mathbf{x}^\top \mathbf{A}\mathbf{x} + \mathbf{b}\cdot\mathbf{x} + c$$

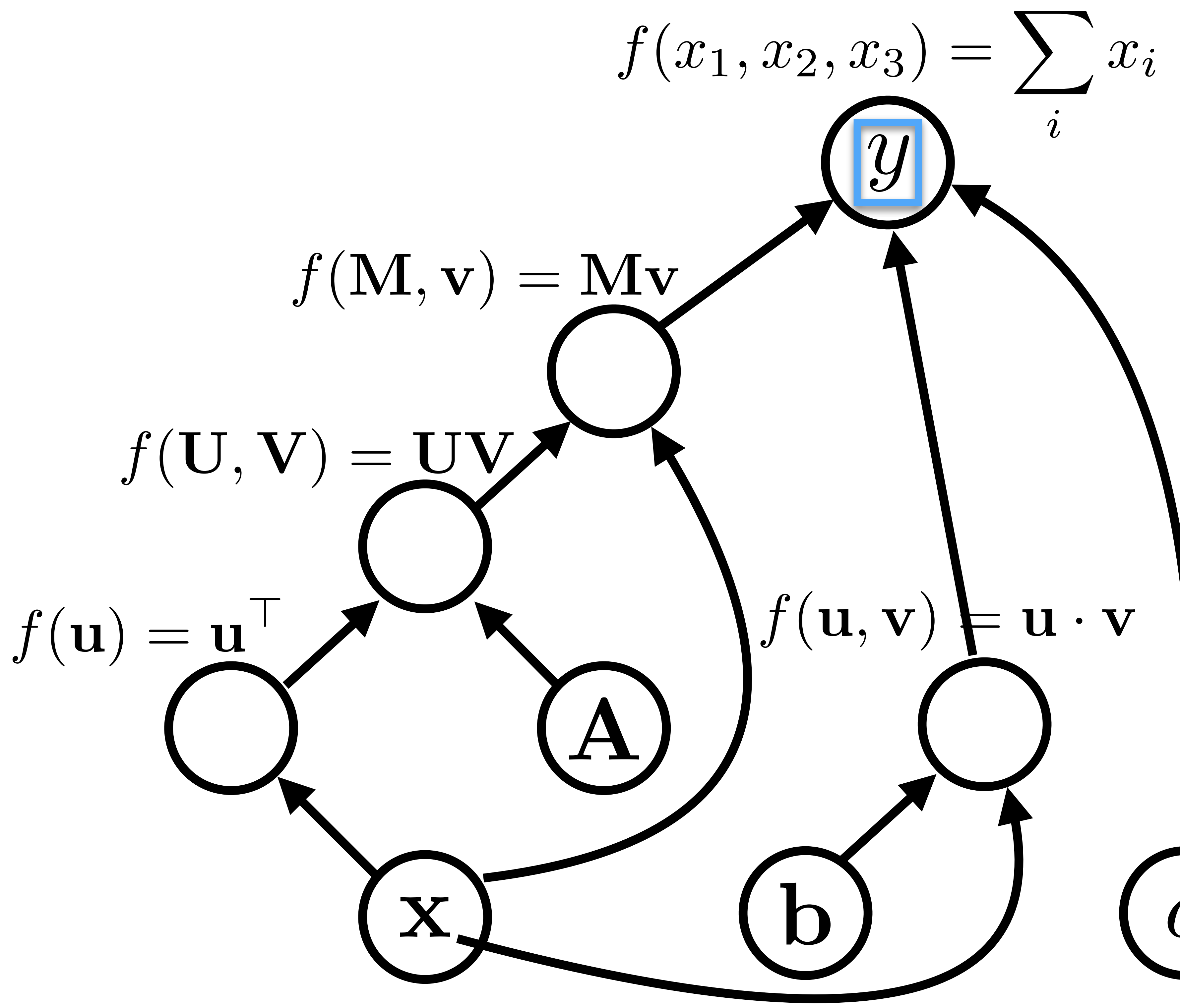
graph:



expression:

$$\boxed{y =} \mathbf{x}^\top \mathbf{A}\mathbf{x} + \mathbf{b}\cdot\mathbf{x} + c$$

graph:



variable names are just labelings of nodes.

Algorithms (1)

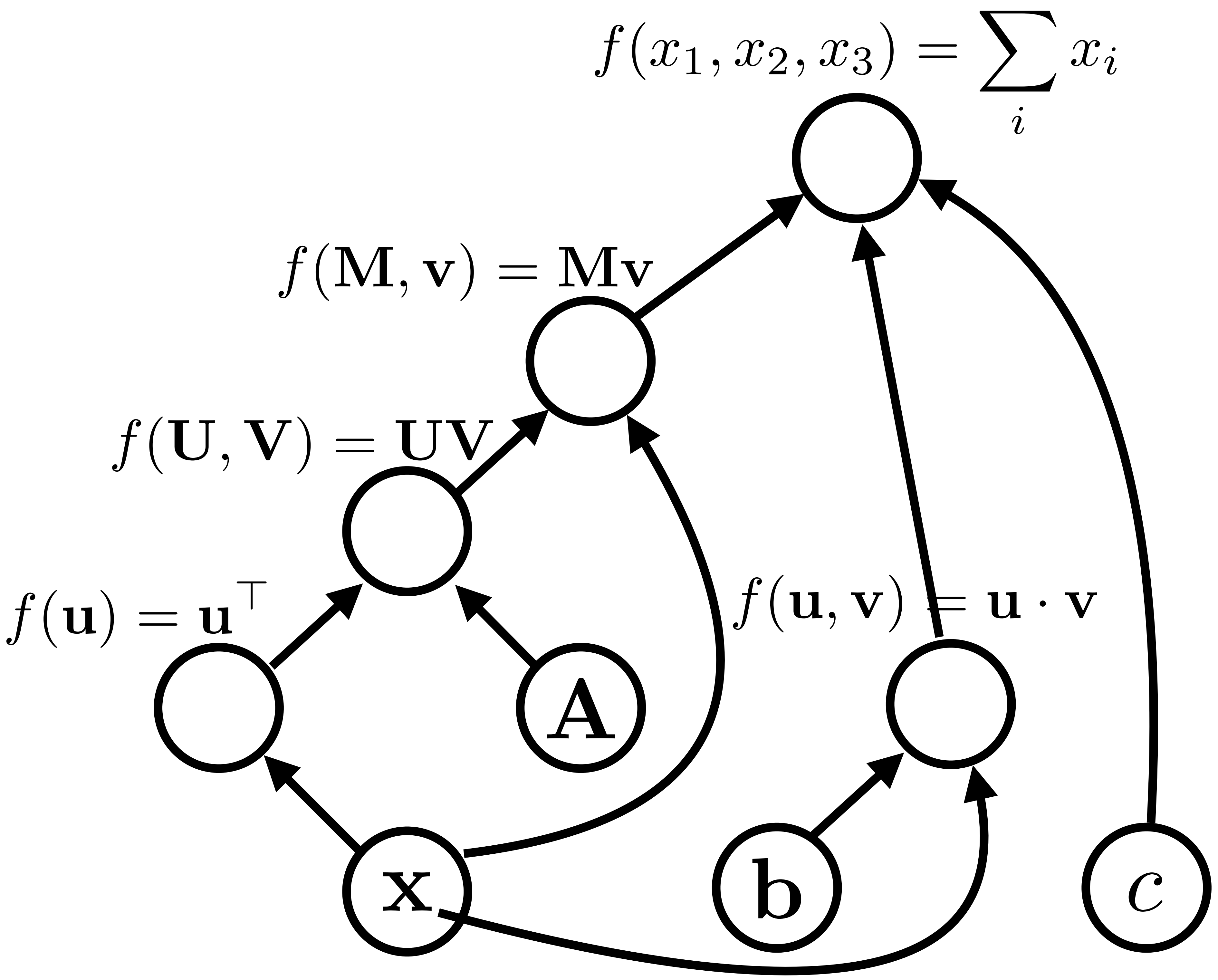
Graph construction

Forward propagation

In topological order, compute the **value** of the node given its inputs

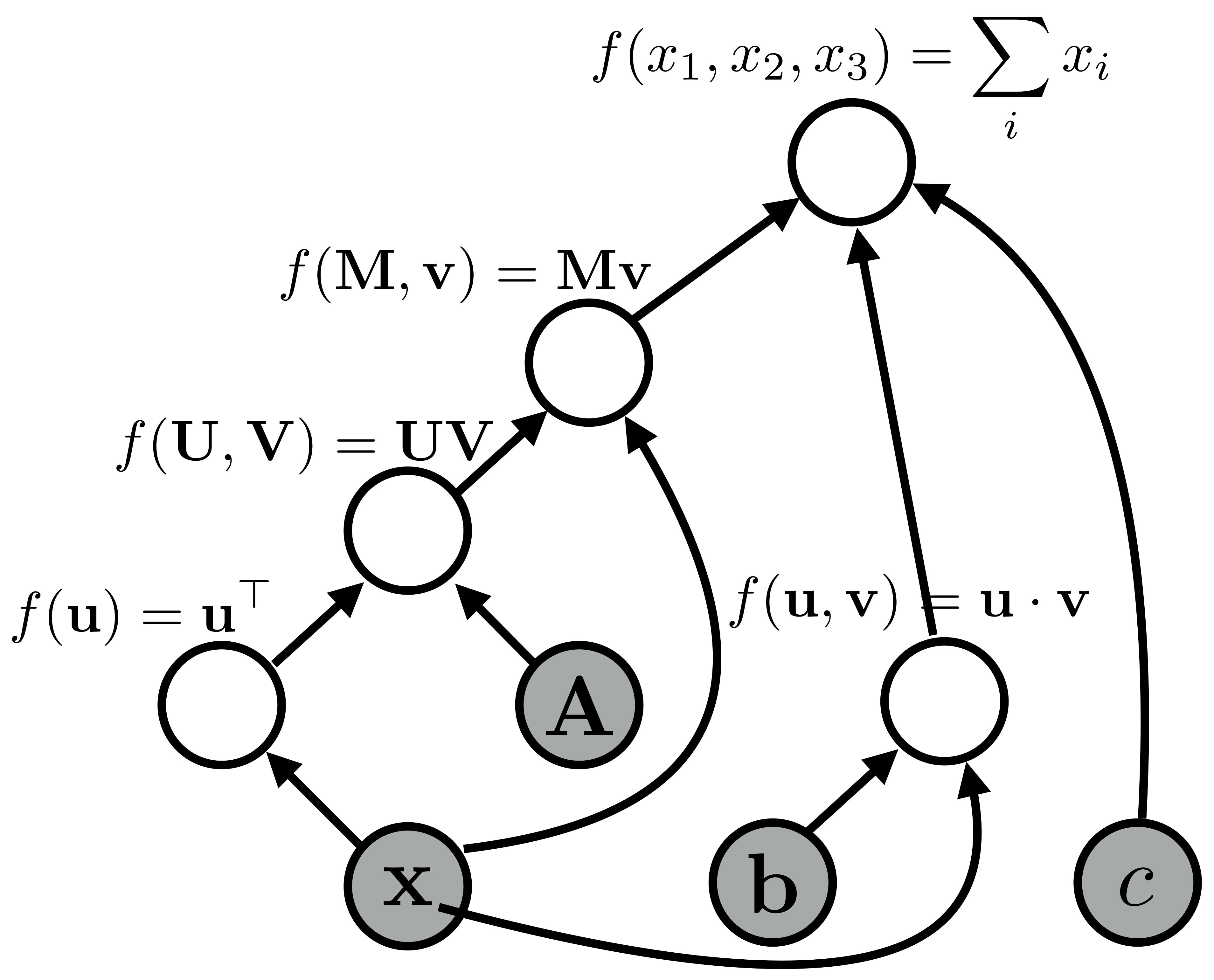
Forward Propagation

graph:



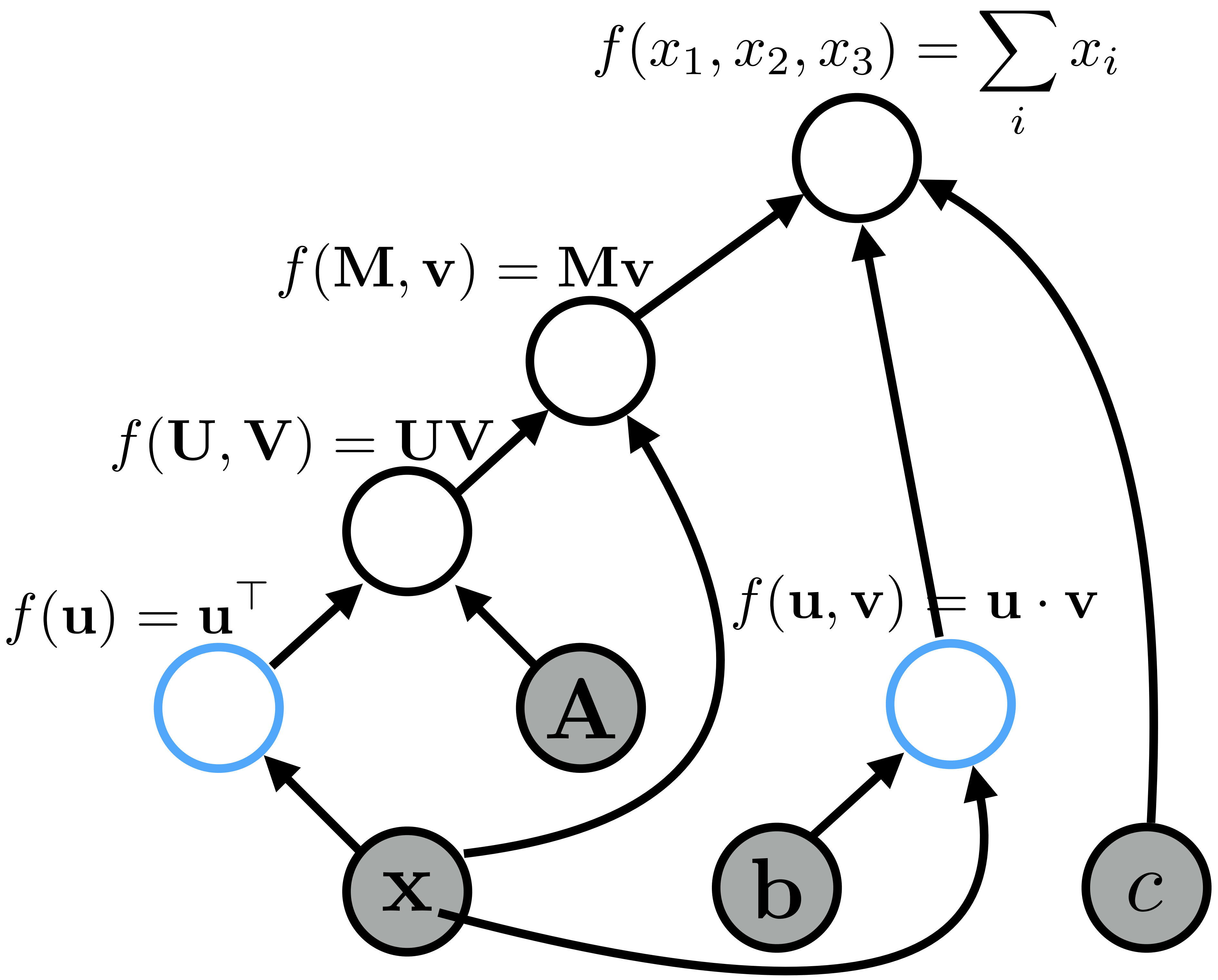
Forward Propagation

graph:



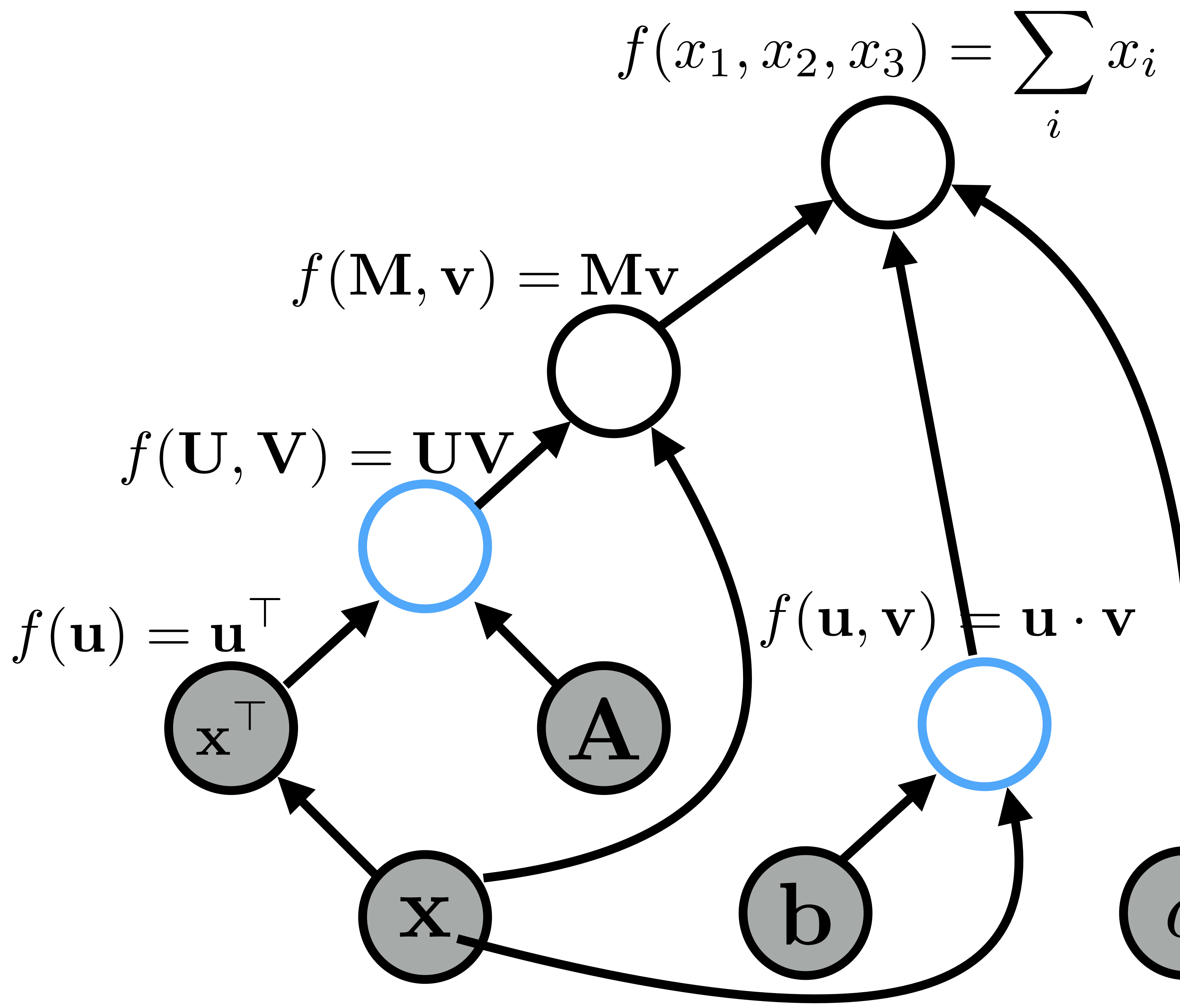
Forward Propagation

graph:



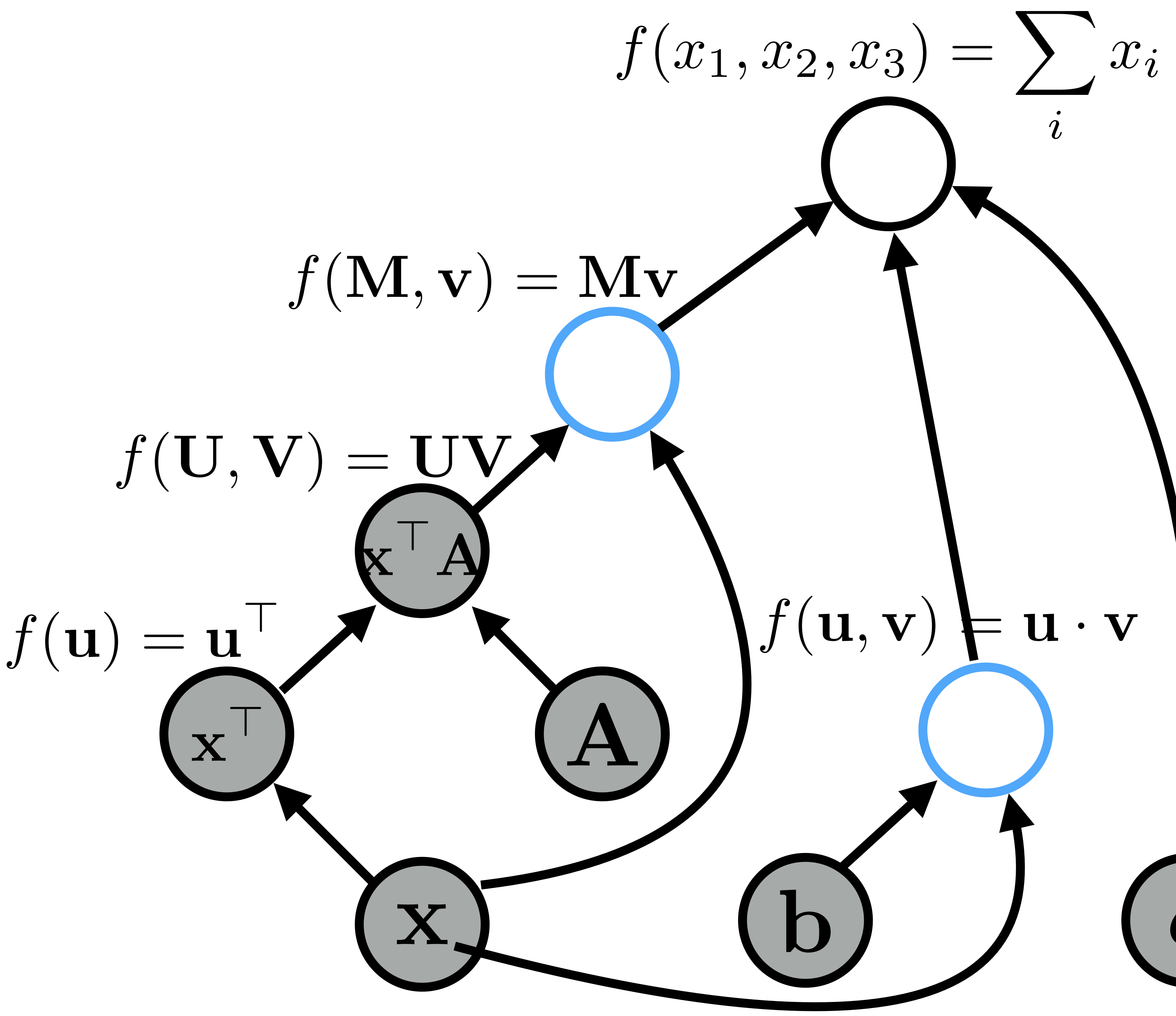
Forward Propagation

graph:



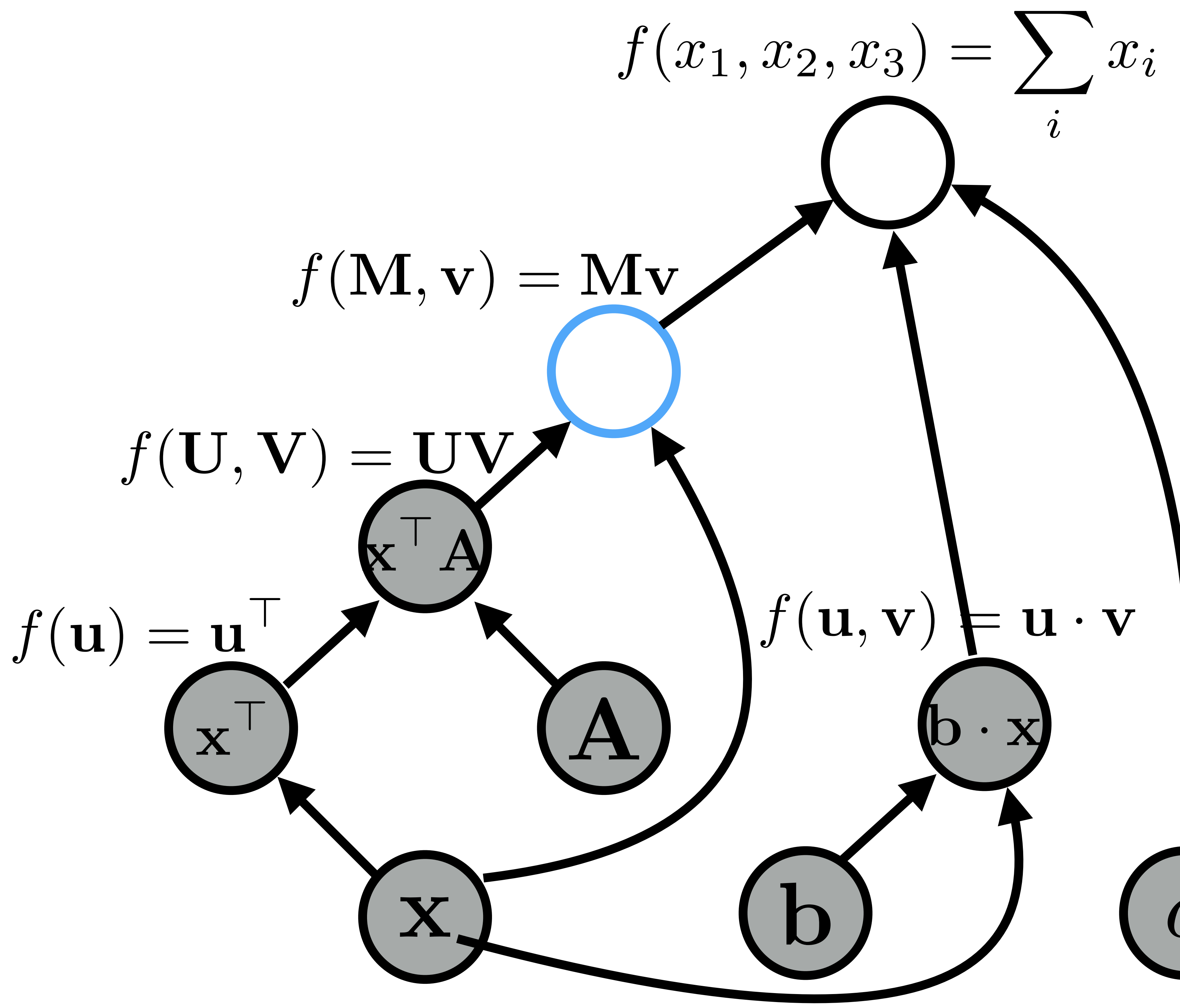
Forward Propagation

graph:



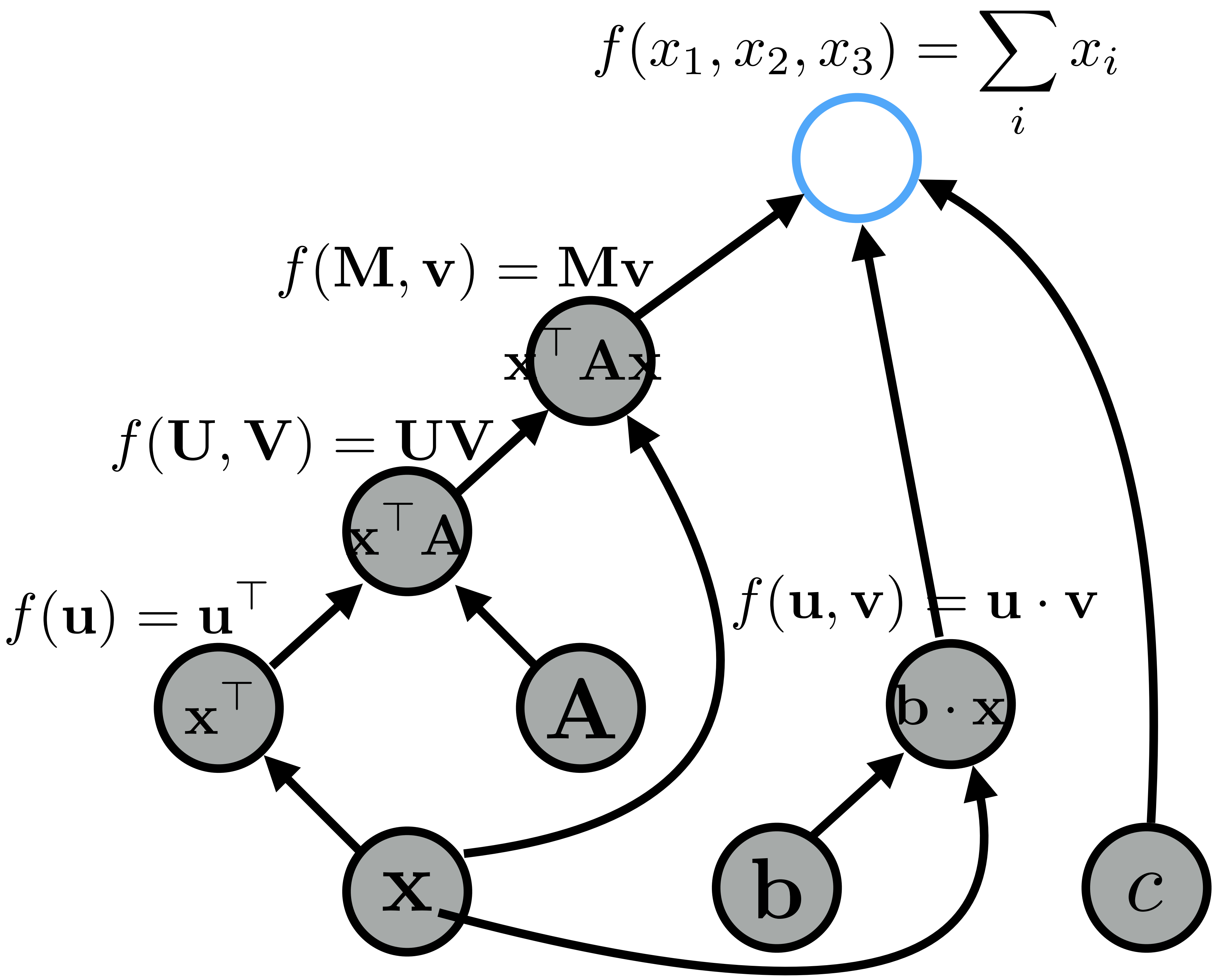
Forward Propagation

graph:



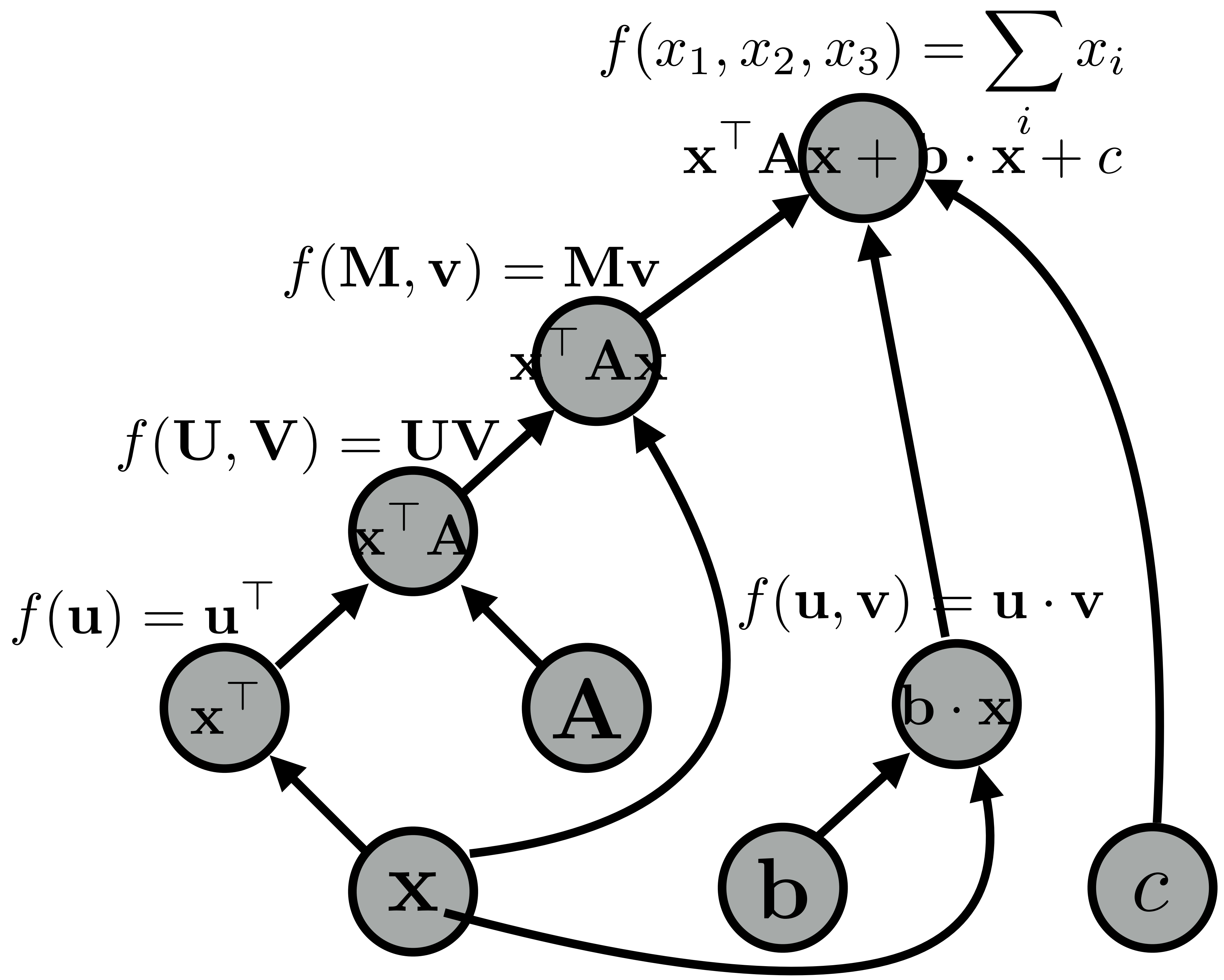
Forward Propagation

graph:



Forward Propagation

graph:



Algorithms (2)

Algorithms (2)

Back-propagation:

Process examples in reverse topological order

Calculate the derivatives of the parameters with respect to the final value
(This is usually a “loss function”, a value we want to minimize)

Algorithms (2)

Back-propagation:

Process examples in reverse topological order

Calculate the derivatives of the parameters with respect to the final value (This is usually a “loss function”, a value we want to minimize)

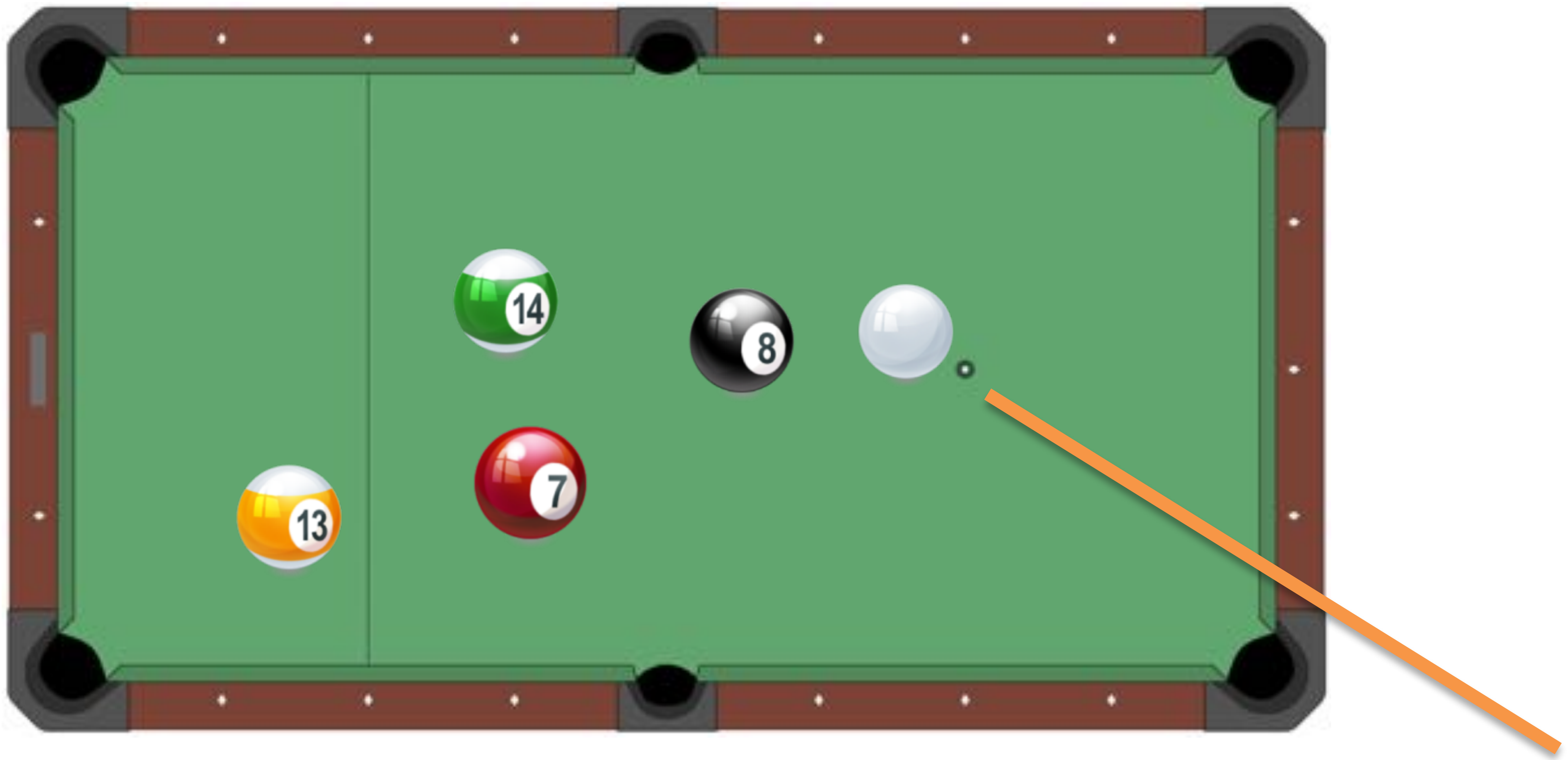
Parameter update:

Move the parameters in the direction of this derivative
W -= α * dl/dW

Intuitions

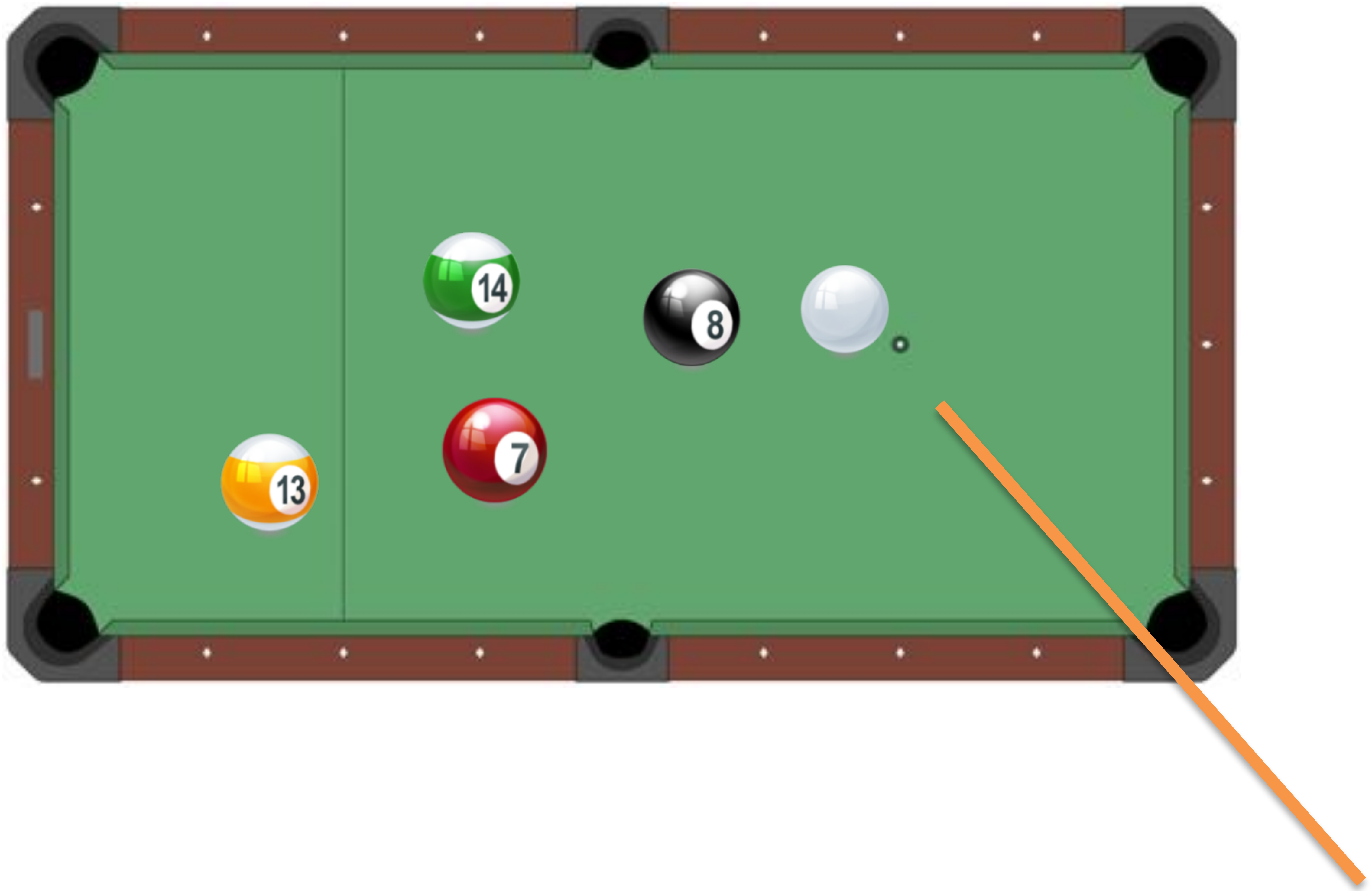
BACKPROPAGATION OF ERRORS

Error Back-Propagation



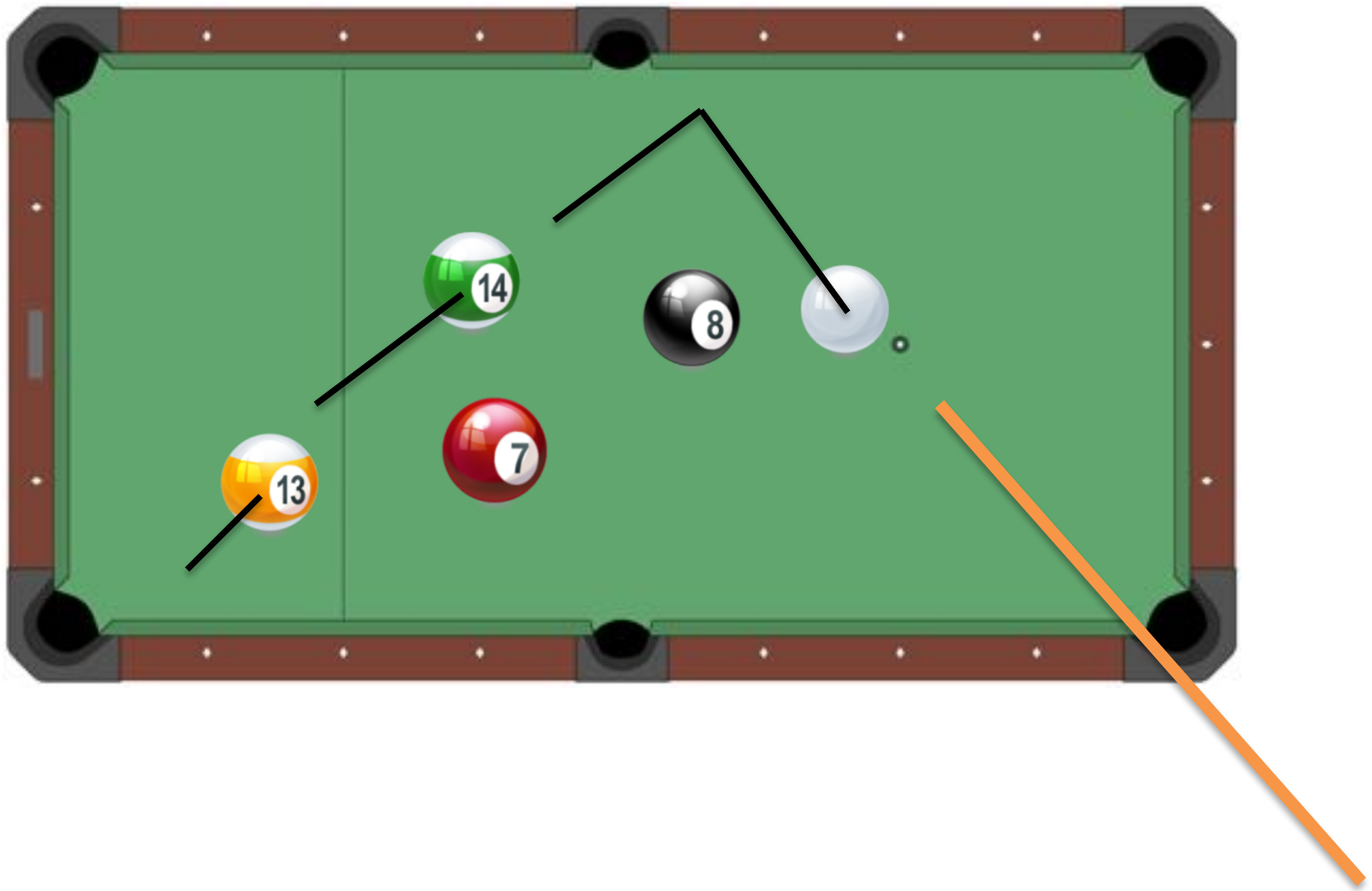
Slide from (Stoyanov & Eisner, 2012)

Error Back-Propagation



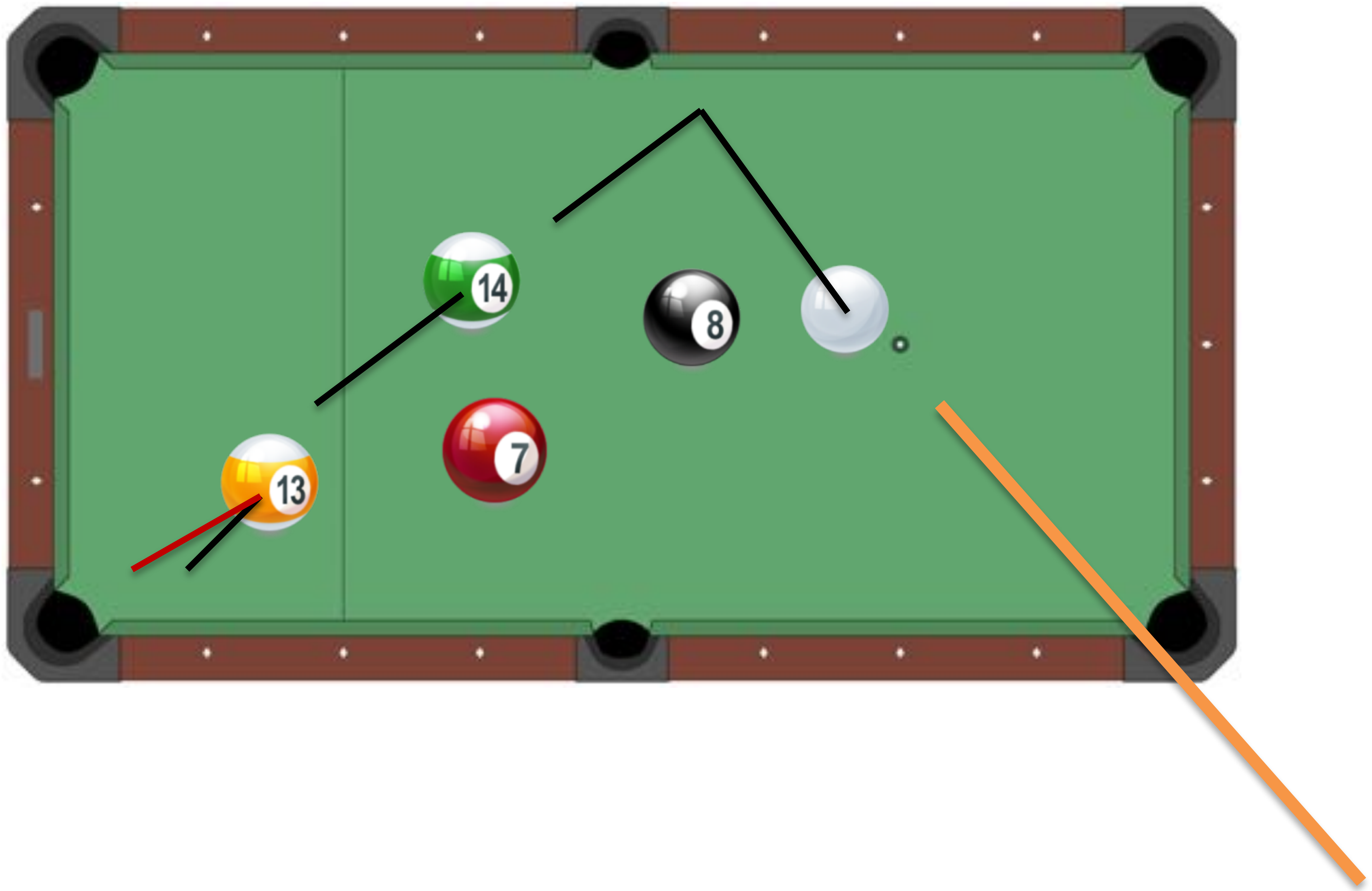
Slide from (Stoyanov & Eisner, 2012)

Error Back-Propagation



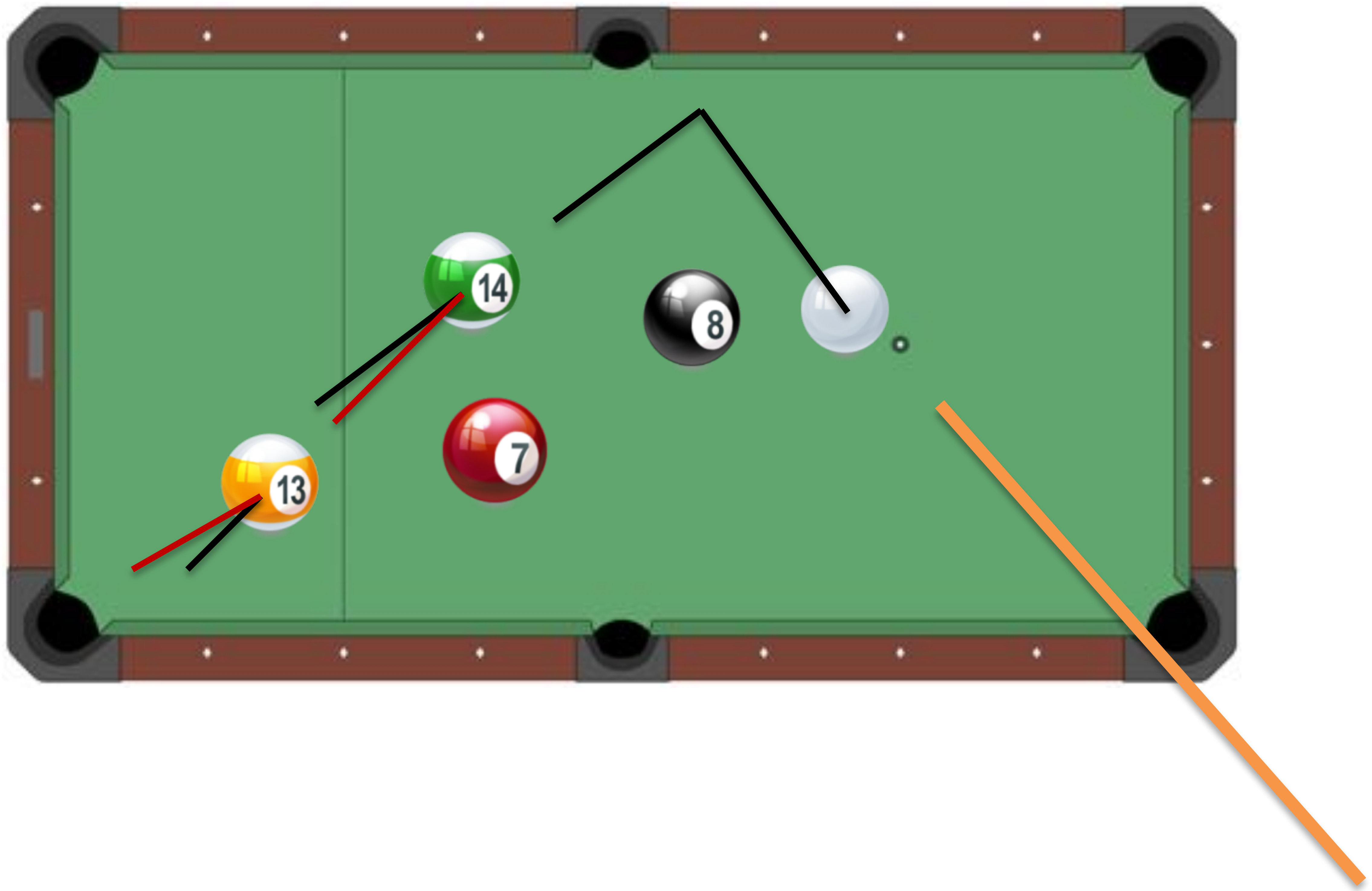
Slide from (Stoyanov & Eisner, 2012)

Error Back-Propagation



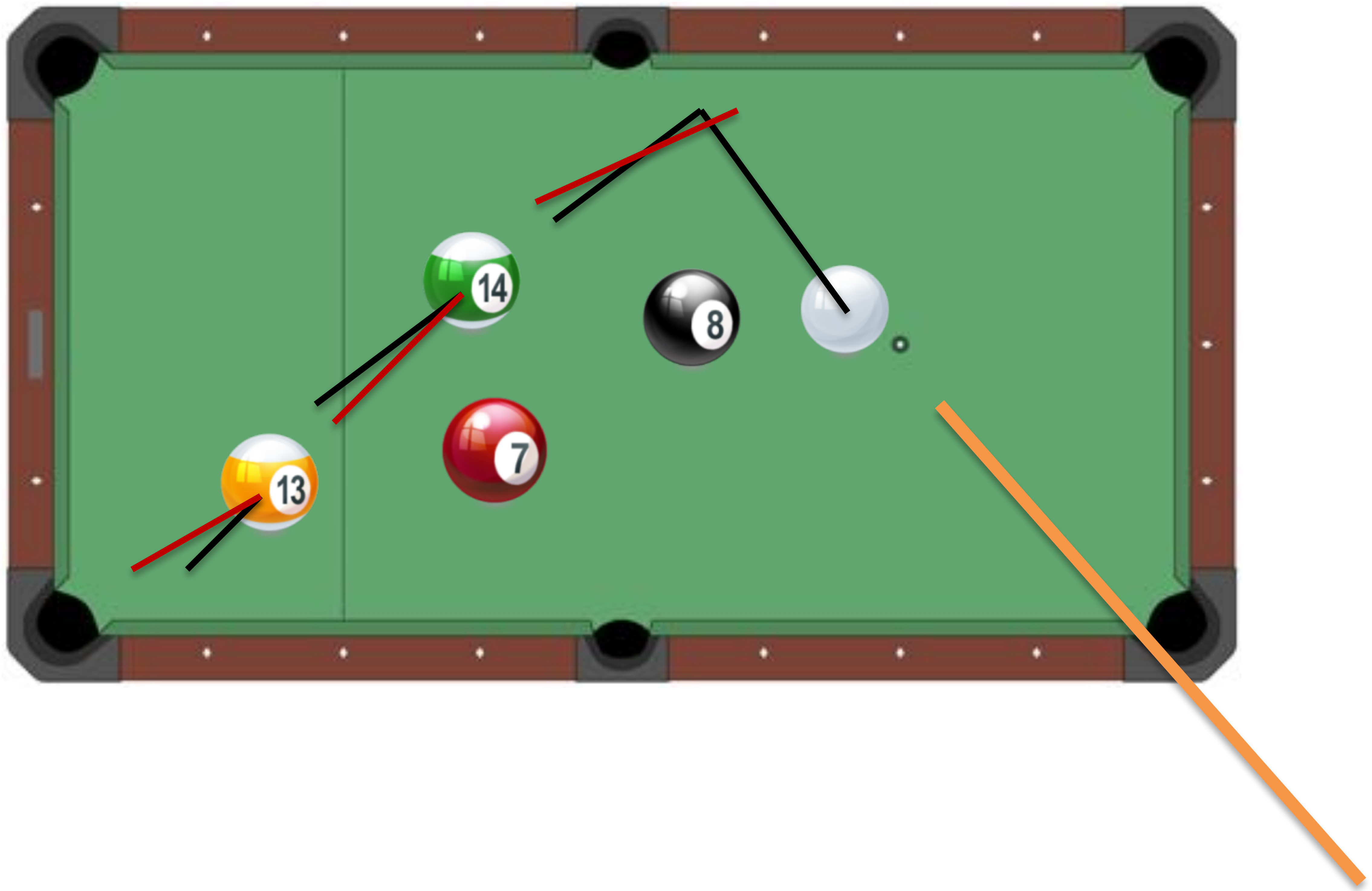
Slide from (Stoyanov & Eisner, 2012)

Error Back-Propagation



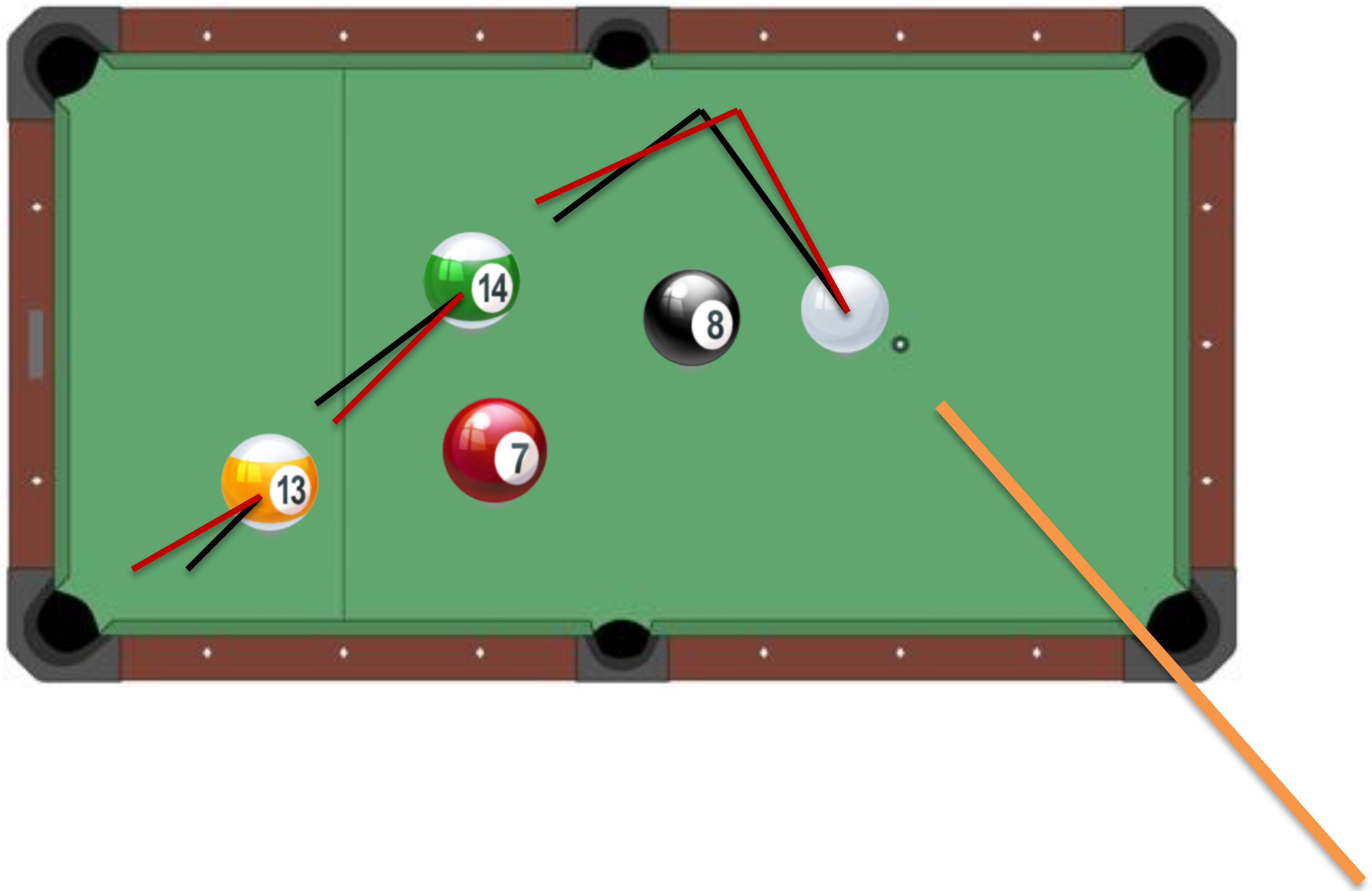
Slide from (Stoyanov & Eisner, 2012)

Error Back-Propagation



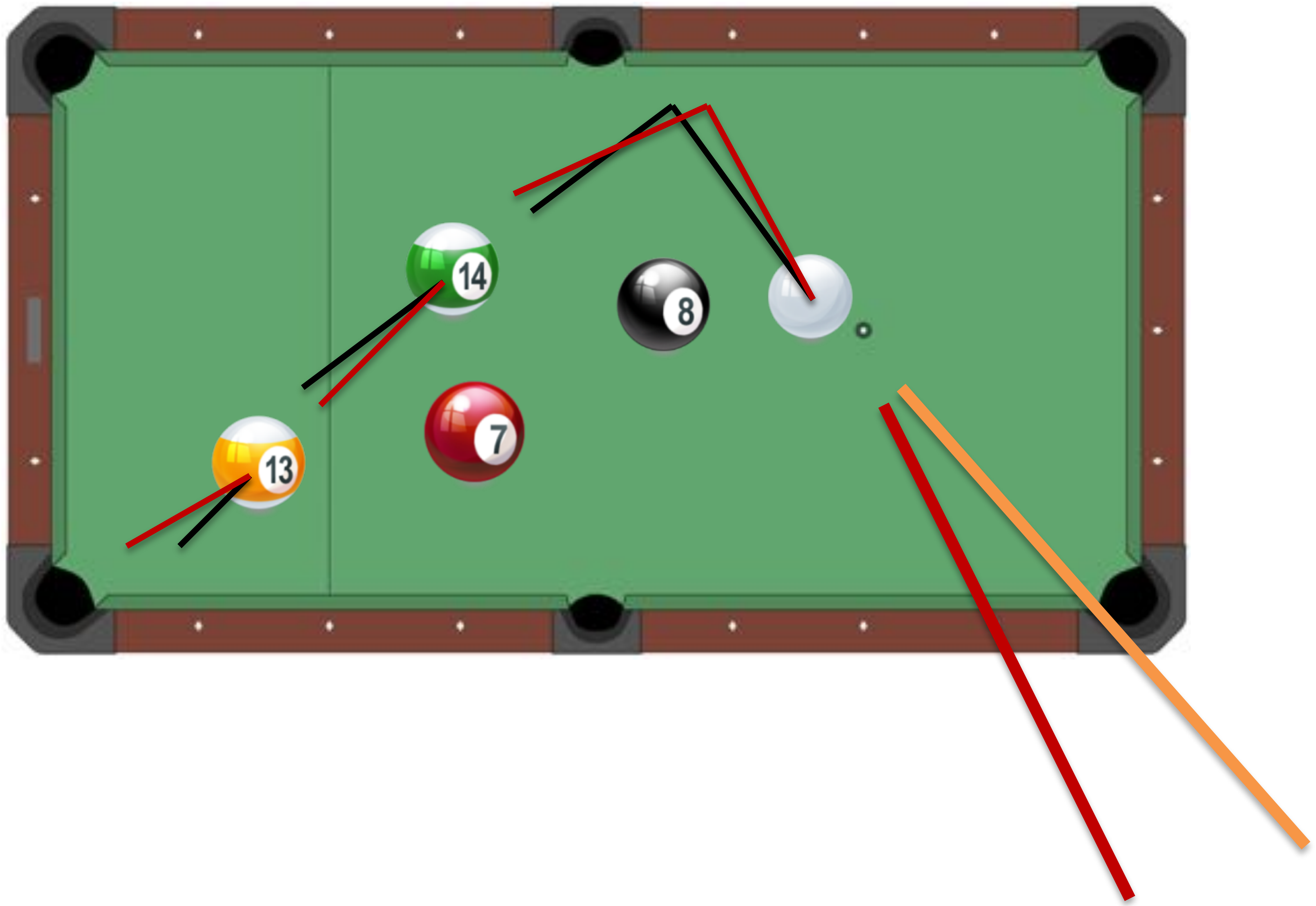
Slide from (Stoyanov & Eisner, 2012)

Error Back-Propagation



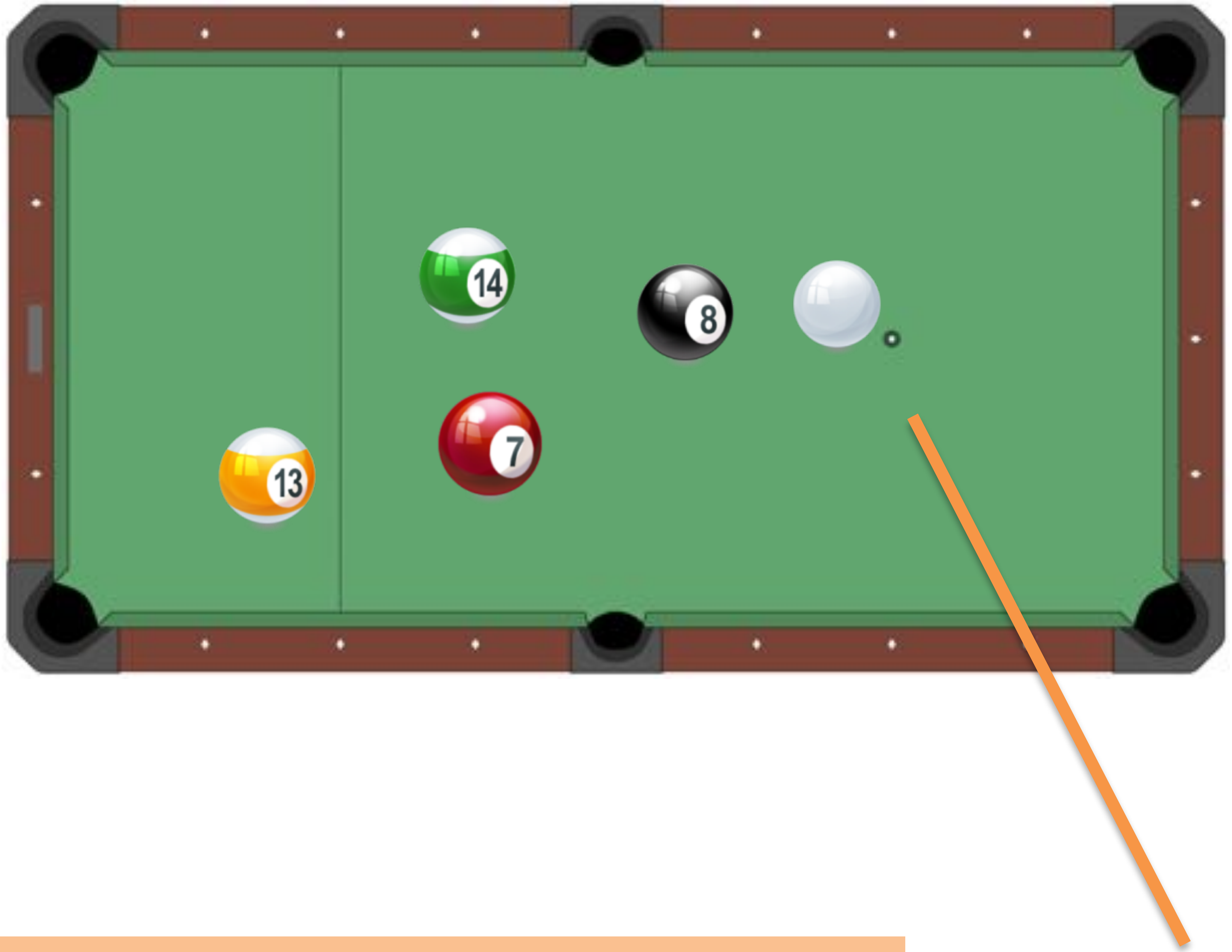
Slide from (Stoyanov & Eisner, 2012)

Error Back-Propagation



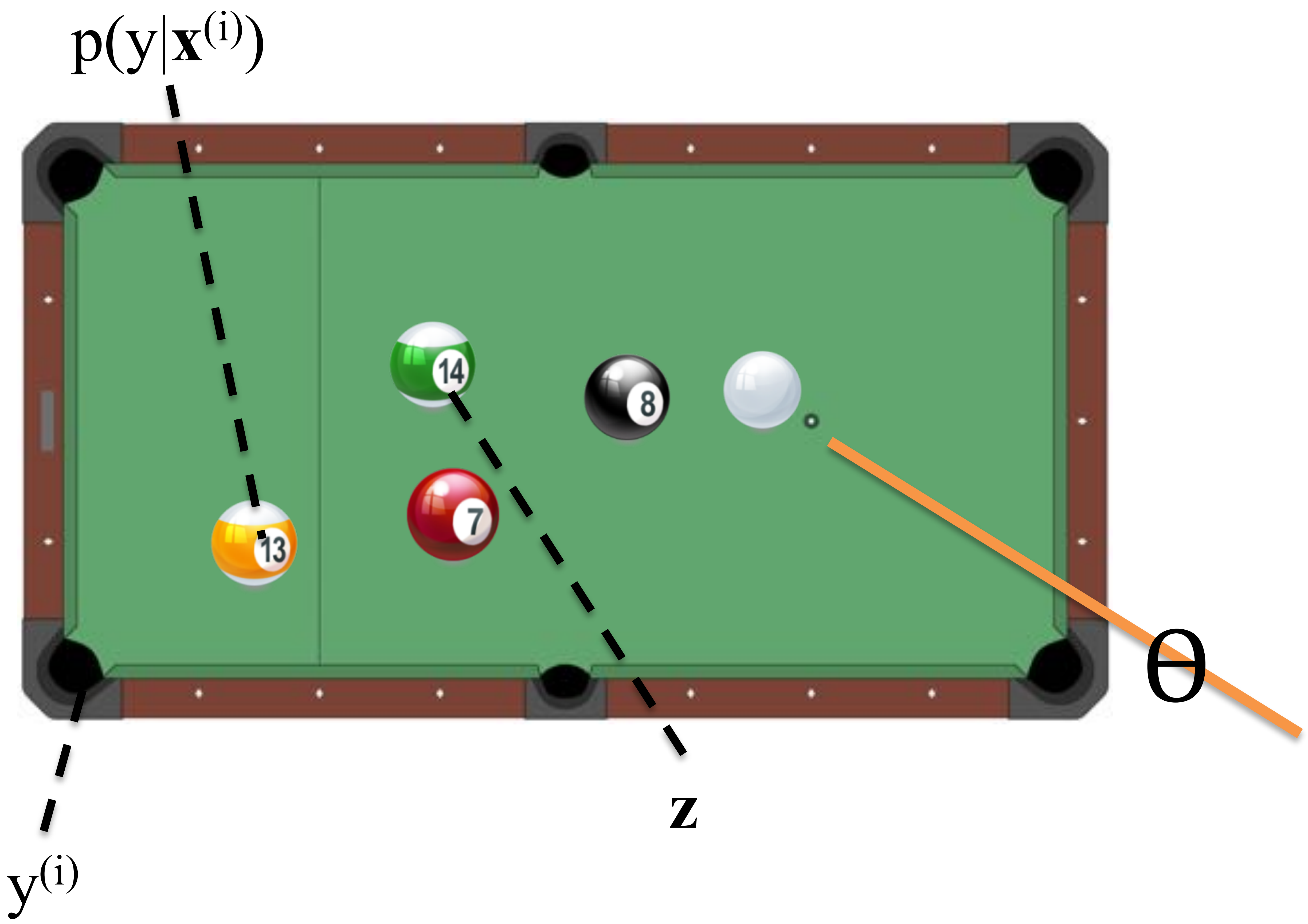
Slide from (Stoyanov & Eisner, 2012)

Error Back-Propagation



Slide from (Stoyanov & Eisner, 2012)

Error Back-Propagation



Slide from (Stoyanov & Eisner, 2012)

Language Models

- Language models are generative models of text

Text Credit: Max Deutsch (https://medium.com/deep-writing/)

Language Models

- Language models are generative models of text

$$s \sim P(x)$$

Text Credit: Max Deutsch (https://medium.com/deep-writing/)

Language Models

- Language models are generative models of text

$$s \sim P(x)$$

↓

"The Malfoys!" said Hermione.

Harry was watching him. He looked like Madame Maxime. When she strode up the wrong staircase to visit himself.

"I'm afraid I've definitely been suspended from power, no chance—indeed?" said Snape. He put his head back behind them and read groups as they crossed a corner and fluttered down onto their ink lamp, and picked up his spoon. The doorbell rang. It was a lot cleaner down in London.

Text Credit: Max Deutsch (https://medium.com/deep-writing/)

The Generation Problem

The Generation Problem

- We have a model of P(Y|X), how do we use it to generate a sentence?

The Generation Problem

- We have a model of $P(Y|X)$, how do we use it to generate a sentence?
- Two methods:

The Generation Problem

- We have a model of P(Y|X), how do we use it to generate a sentence?
- Two methods:
    - **Sampling:** Try to generate a *random* sentence according to the probability distribution.

The Generation Problem

- We have a model of P(Y|X), how do we use it to generate a sentence?
- Two methods:
  - **Sampling:** Try to generate a *random* sentence according to the probability distribution.
  - **Argmax:** Try to generate the sentence with the *highest* probability.

Ancestral Sampling

Ancestral Sampling

- **Randomly generate** words one-by-one.

Ancestral Sampling

- **Randomly generate** words one-by-one.

```
while y_{j-1} != "</s>":
  y_j ~ P(y_j | X, y_1, …, y_{j-1})
```

Ancestral Sampling

- **Randomly generate** words one-by-one.

```
while y_{j-1} != "</s>":
  y_j ~ P(y_j | X, y_1, …, y_{j-1})
```

- An **exact method** for sampling from P(X), no further work needed.

Greedy Search

Greedy Search

- One by one, pick the single highest-probability word

Greedy Search

- One by one, pick the single highest-probability word

$$
\begin{aligned}
&\text{while } y_{j-1} \mathrel{!=} \text{“</s>”}: \\
&\quad y_j = \text{argmax } P(y_j \mid X, y_1, \ldots, y_{j-1})
\end{aligned}
$$

Greedy Search

- One by one, pick the single highest-probability word

```
while y_{j-1} != "</s>":
  y_j = argmax P(y_j | X, y_1, …, y_{j-1})
```

- **Not exact, real problems:**

Greedy Search

- One by one, pick the single highest-probability word

$$\begin{array}{l} \text{while } y_{j-1} \text{ != "</s>":} \\ \quad y_j = \text{argmax } P(y_j \mid X, y_1, \ldots, y_{j-1}) \end{array}$$

- **Not exact, real problems:**
  - Will often generate the “easy” words first

Greedy Search

- One by one, pick the single highest-probability word

```
while y_{j-1} != "</s>":
  y_j = argmax P(y_j | X, y_1, ..., y_{j-1})
```

$$\text{while } y_{j-1} \neq \text{“</s>”}: \quad y_j = \operatorname{argmax} P(y_j \mid X, y_1, \ldots, y_{j-1})$$

- **Not exact, real problems:**
  - Will often generate the “easy” words first
  - Will prefer multiple common words to one rare word