CS11-747 Neural Networks for NLP

Introduction, Bag-of-words, and Multi-layer Perceptron

Graham Neubig

Carnegie Mellon University

Language Technologies Institute

Site

https://phontron.com/class/nn4nlp2020/

Language is Hard!

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.

} Create a grammar of the language

- Jane goed to the store. } Consider morphology and exceptions
- 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。

Phenomena to Handle

- Morphology
- Syntax
- Semantics/World Knowledge
- Discourse
- Pragmatics
- Multilinguality

Neural Nets for NLP

- Neural nets are a tool to do hard things!
- This class will give you the tools to handle the problems you want to solve in NLP.

Class Format/Structure

Class Format

- **Before class:** Read material on the topic
- **During class:**
  - *Quiz:* Simple questions about the required reading (should be easy)
  - *Summary/Questions/Elaboration:* Instructor or TAs will summarize the material, field questions, elaborate on details and talk about advanced topics
  - *Code Walk:* The TAs (or instructor) will sometimes walk through some demonstration code or equations
- **After class:** Review the code, try to run/modify it yourself. Visit office hours to talk about questions, etc.

Scope of Teaching

- **Basics of general neural network knowledge**
  -> Covered briefly (see reading and ask TAs if you are not familiar). Will have recitation.

- **Advanced training techniques for neural networks**
  -> Some coverage, like VAEs and adversarial training, mostly from the scope of NLP, not as much as other DL classes

- **Advanced NLP-related neural network architectures**
  -> Covered in detail

- **Structured prediction and structured models in neural nets**
  -> Covered in detail

- **Implementation details salient to NLP**
  -> Covered in detail

Assignments

- Course is largely group (2-3) assignment based
- **Assignment 1 - Text Classifier / Questionnaire:** *Individually* implement a text classifier and fill in questionnaire project topics
- **Assignment 2 - SOTA Survey:** Survey about your project topic and describe the state-of-the-art
- **Assignment 3 - SOTA Re-implementation:** Re-implement and reproduce results from a state-of-the-art model
- **Assignment 4 - Final Project:** Perform a unique project that either (1) improves on state-of-the-art, or (2) applies neural net models to a unique task

Instructors/Office Hours

- **Instructors:** Graham Neubig
  (Fri. 4-5PM GHC5409)
  Pengfei Liu
  (Wed. 2-3PM GHC6607)
- **TAs:**
  - Aditi Chaudhary (Mon. 10-11AM GHC6509)
  - Chunting Zhou (Fri. 10-11AM GHC5705)
  - Hiroaki Hayashi (Thu. 11AM-12PM GHC5705)
  - Pengcheng Yin (Wed. 10-11AM GHC5505)
  - Vidhisha Balachandran (Tue. 10-11AM GHC5713)
  - Zi-Yi Dou (Tue. 12-1PM GHC5417)
- **Piazza: http://piazza.com/cmu/spring2020/cs11747/home**

Neural Networks:
A Tool for Doing Hard Things

An Example Prediction Problem: Sentence Classification

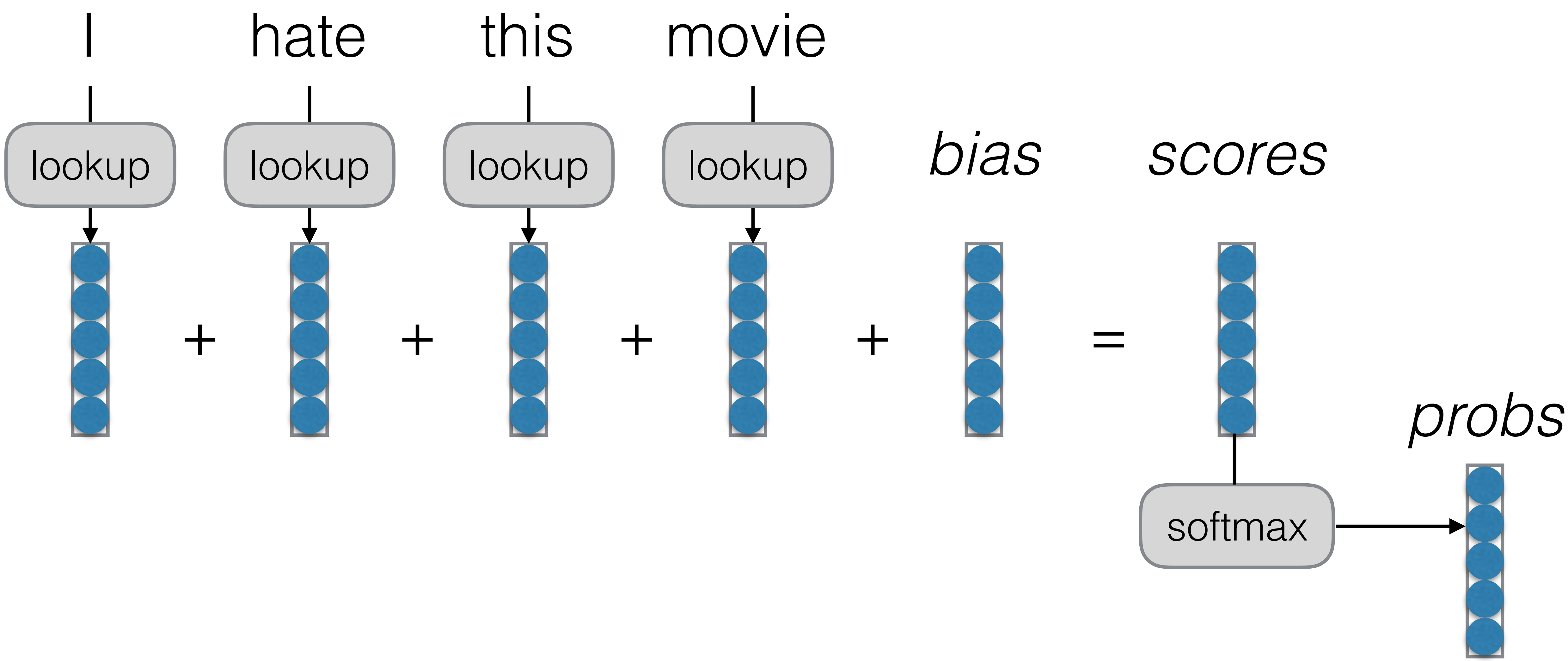
I hate this movie

very good
good
neutral
bad
very bad

I love this movie

very good
good
neutral
bad
very bad

A First Try: Bag of Words (BOW)



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

I don’t love this movie

very good
good
neutral
bad
very bad

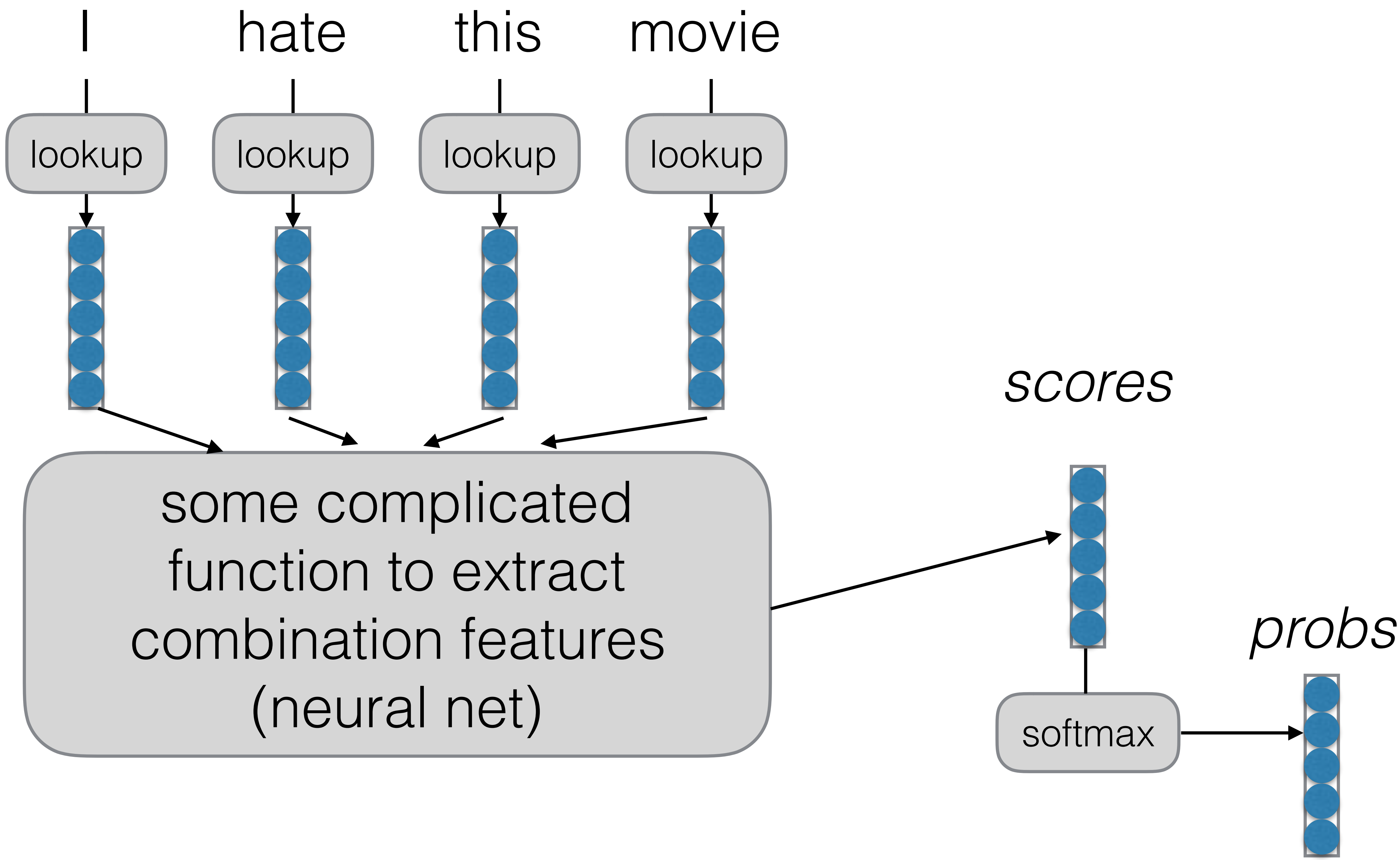
There’s nothing I don’t love about this movie

very good
good
neutral
bad
very bad

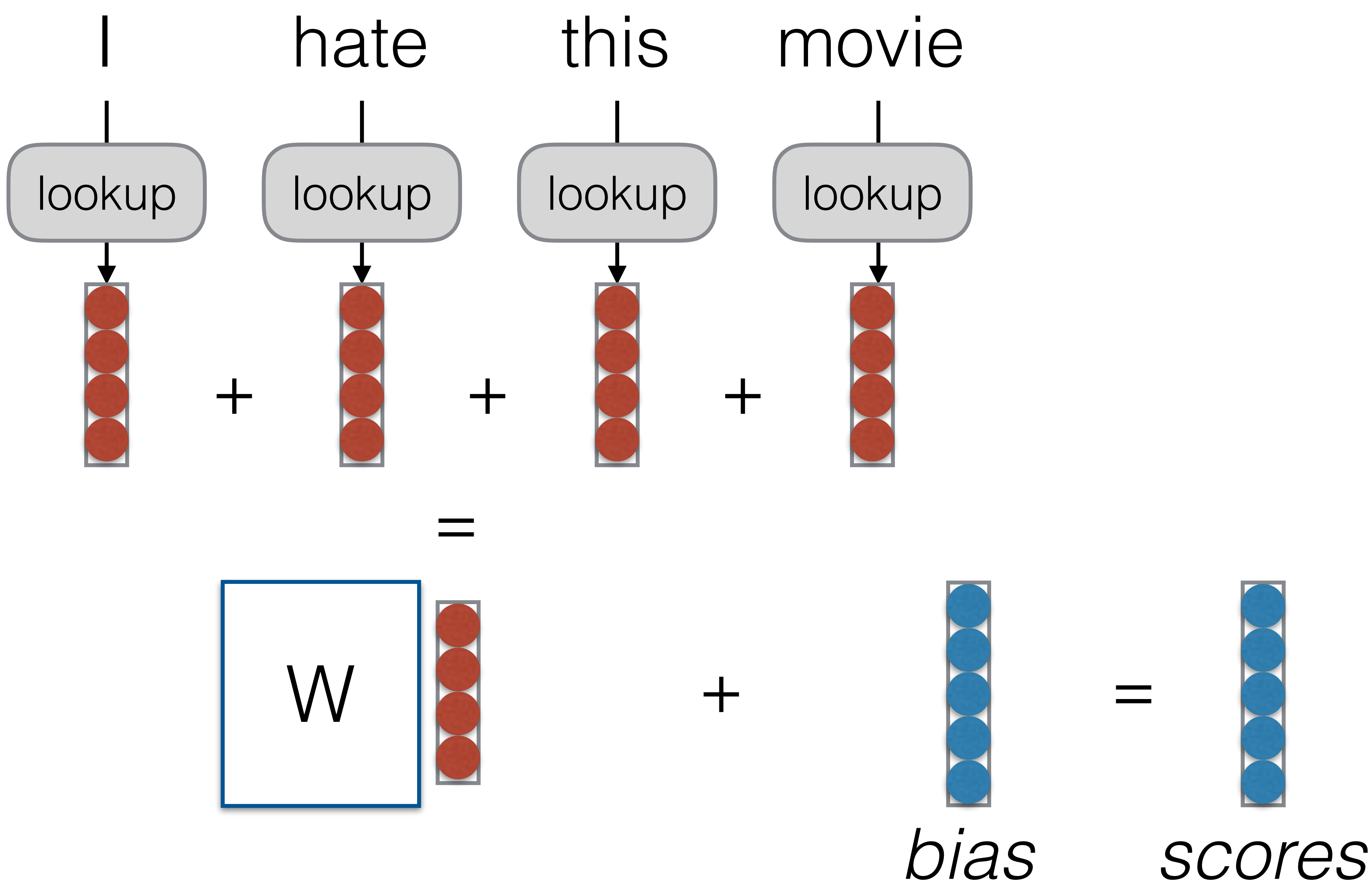
Combination Features

- 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)



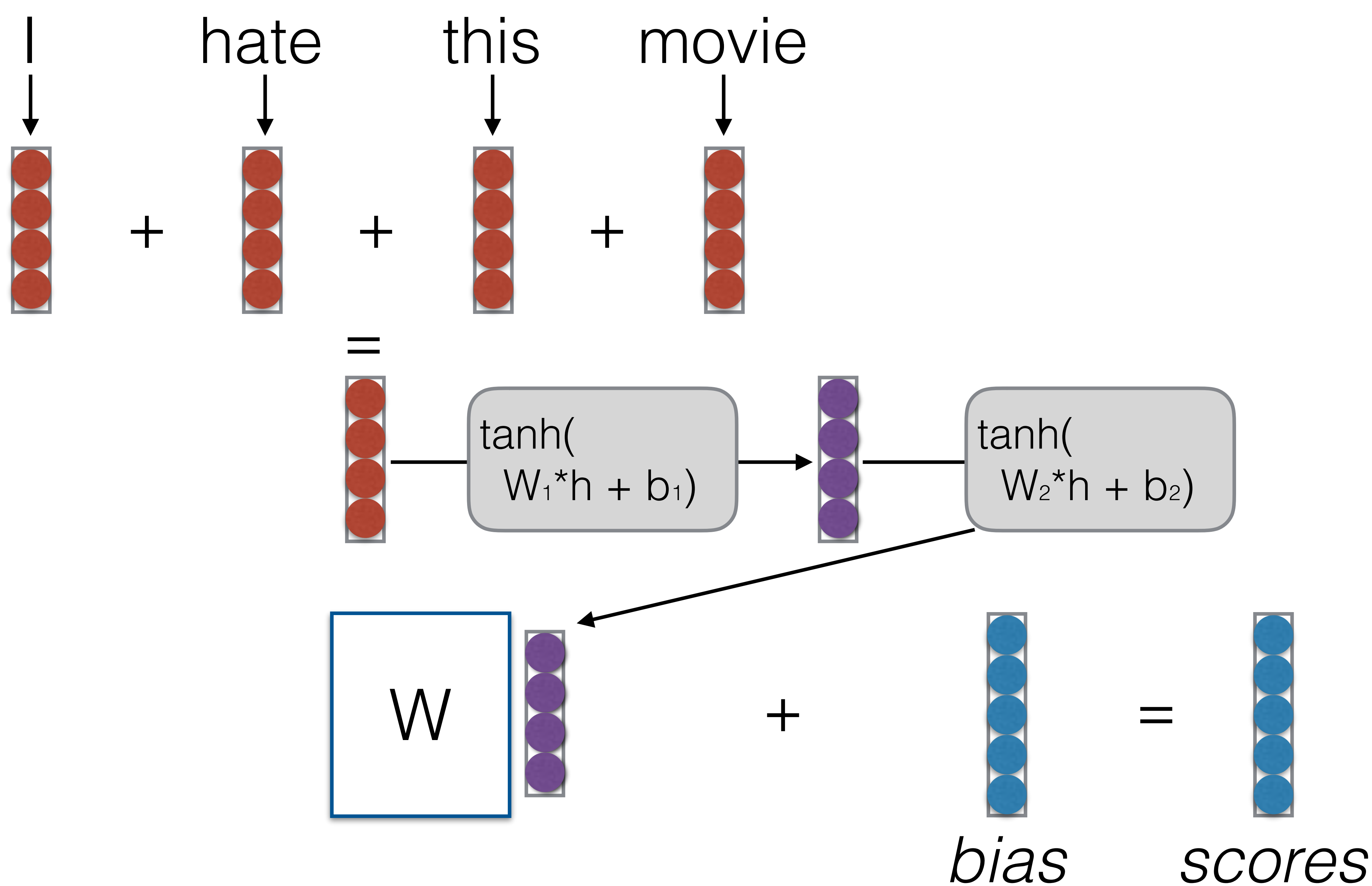
Continuous Bag of Words (CBOW)



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



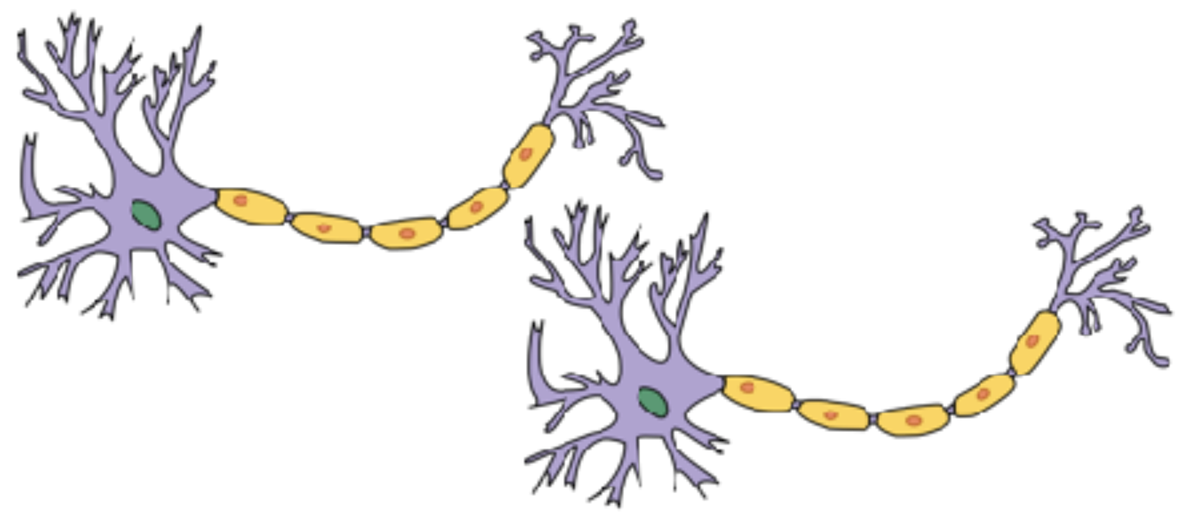
What do Our Vectors Represent?

- 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: Neurons in the Brain



Current Conception: Computation Graphs

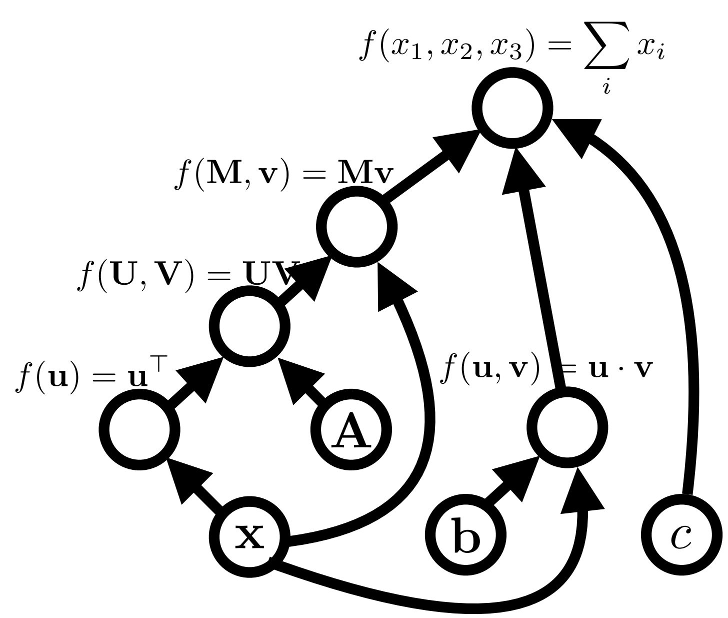


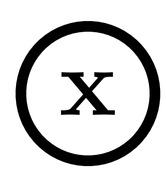
Image credit: Wikipedia

expression:

$$\mathbf{x}$$

graph:

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



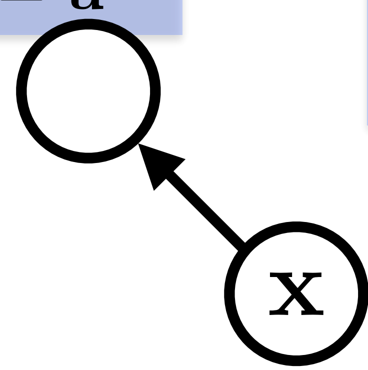
An **edge** represents a function argument (and also an data dependency). They are just pointers to nodes.

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* $\frac{\partial \mathcal{F}}{\partial f(\mathbf{u})}$.

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

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

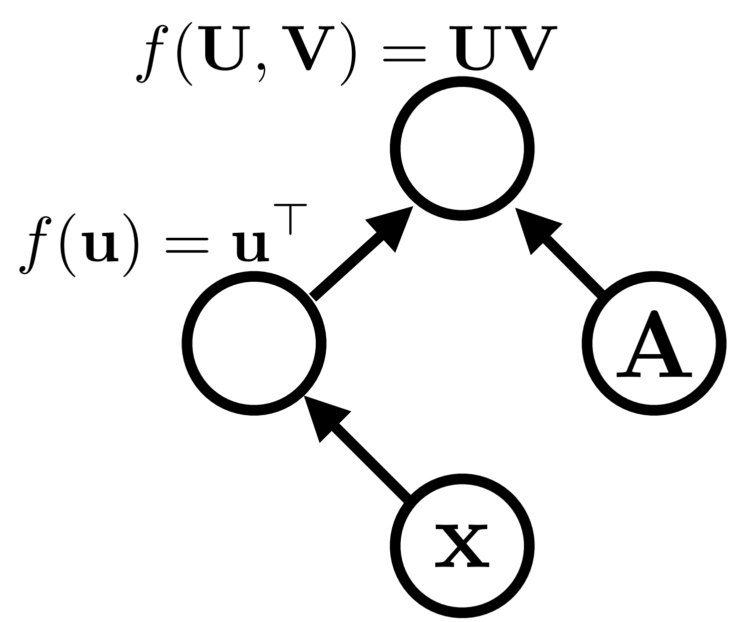


expression:

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

graph:

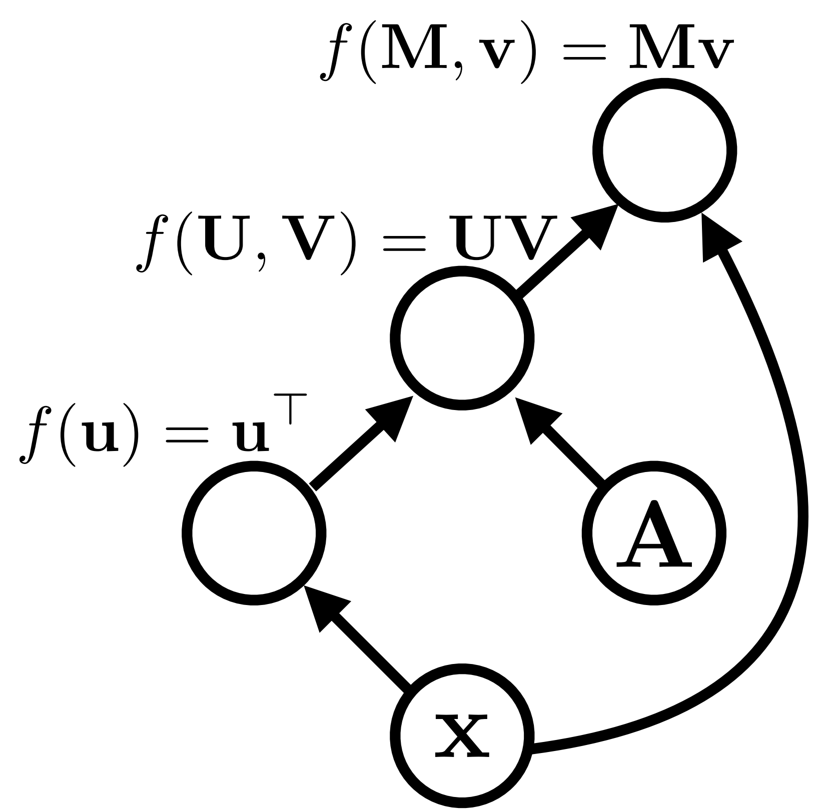
Functions can be nullary, unary, binary, … *n*-ary. Often they are unary or binary.



expression:

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

graph:

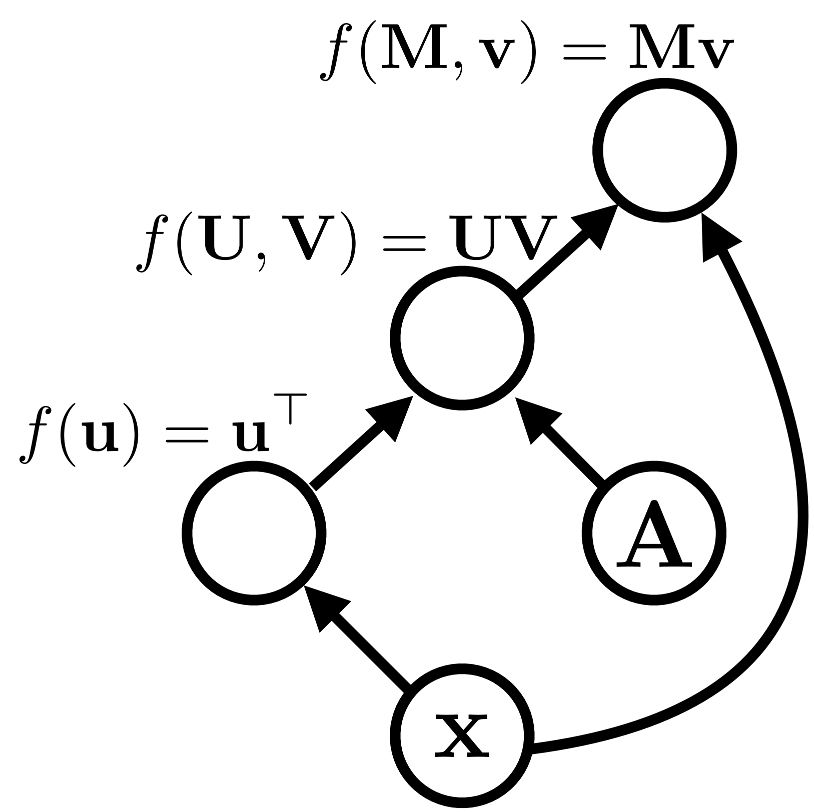


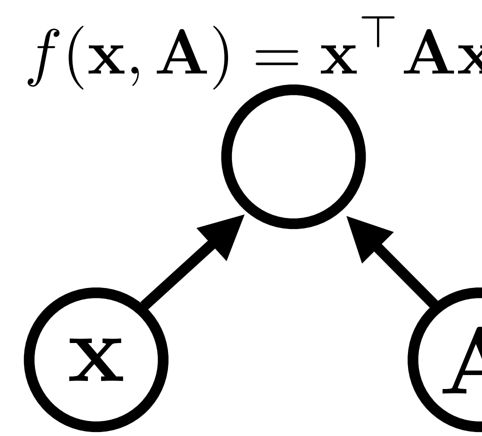
Computation graphs are directed and acyclic (in DyNet)

expression:

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

graph:





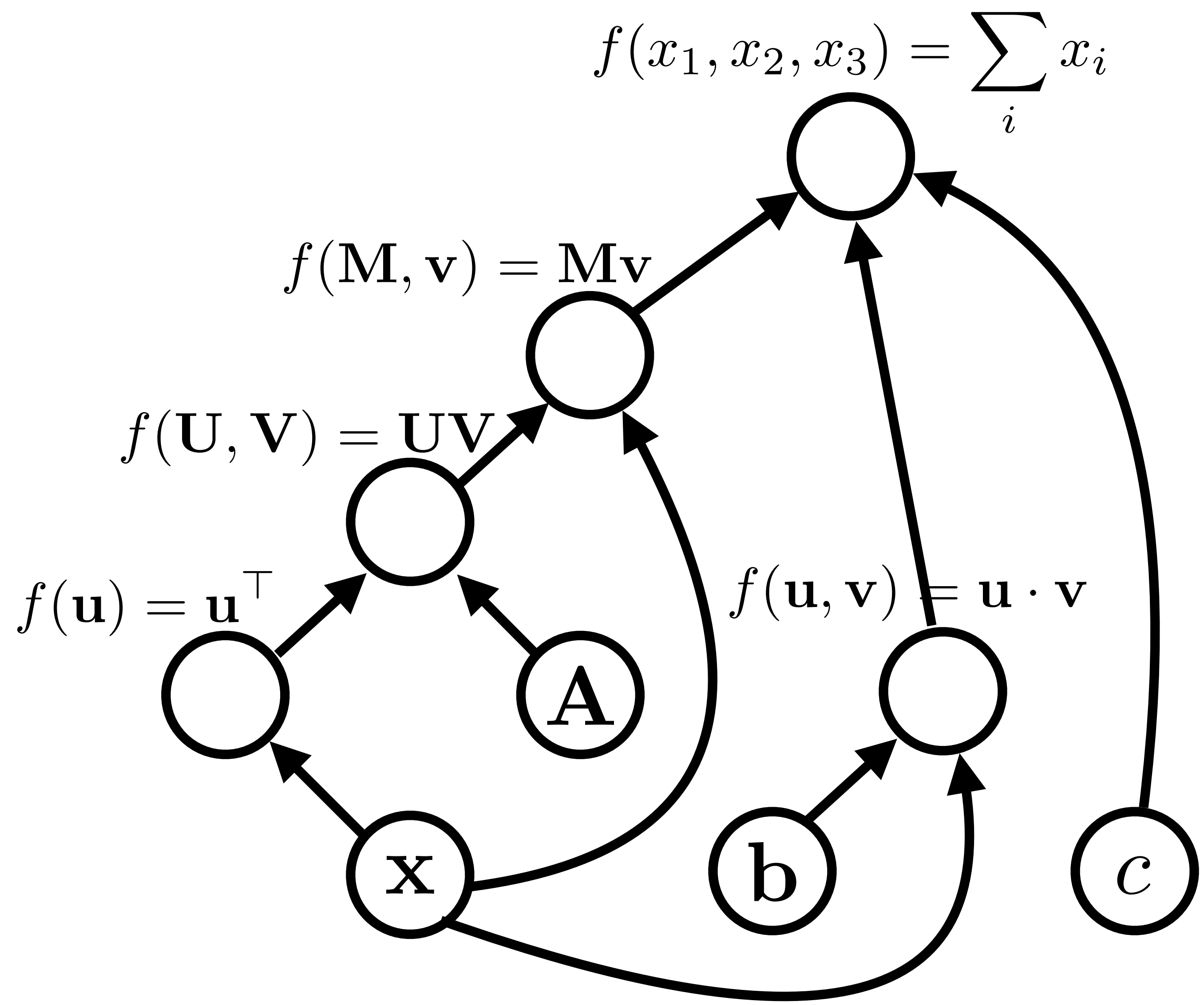
$$\frac{\partial f(\mathbf{x}, \mathbf{A})}{\partial \mathbf{x}} = (\mathbf{A}^\top + \mathbf{A})\mathbf{x}$$

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

expression:

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

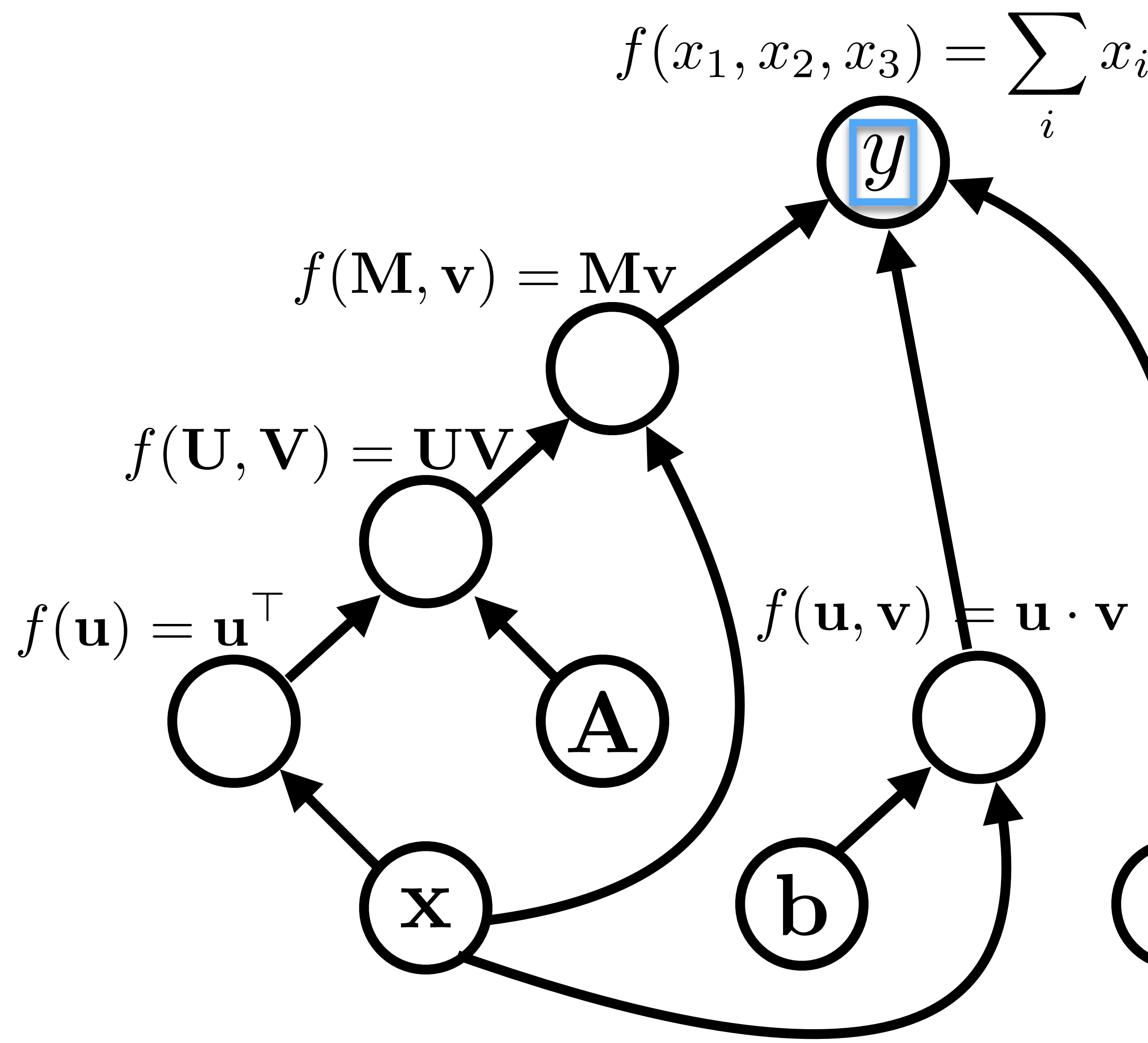
graph:



expression:

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

graph:



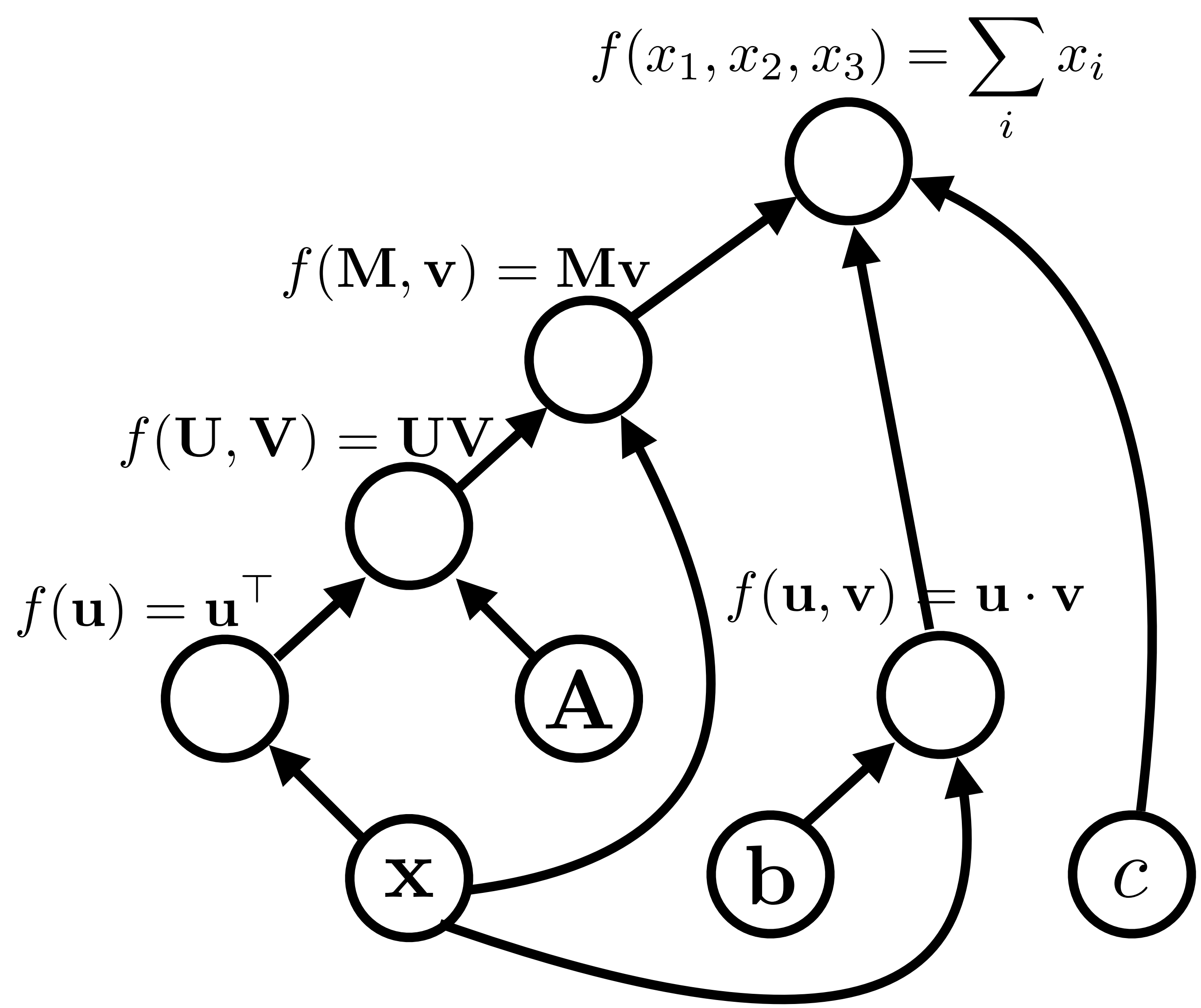
variable names are just labelings of nodes.

Algorithms (1)

- **Graph construction**
- **Forward propagation**
  - In topological order, compute the **value** of the node given its inputs

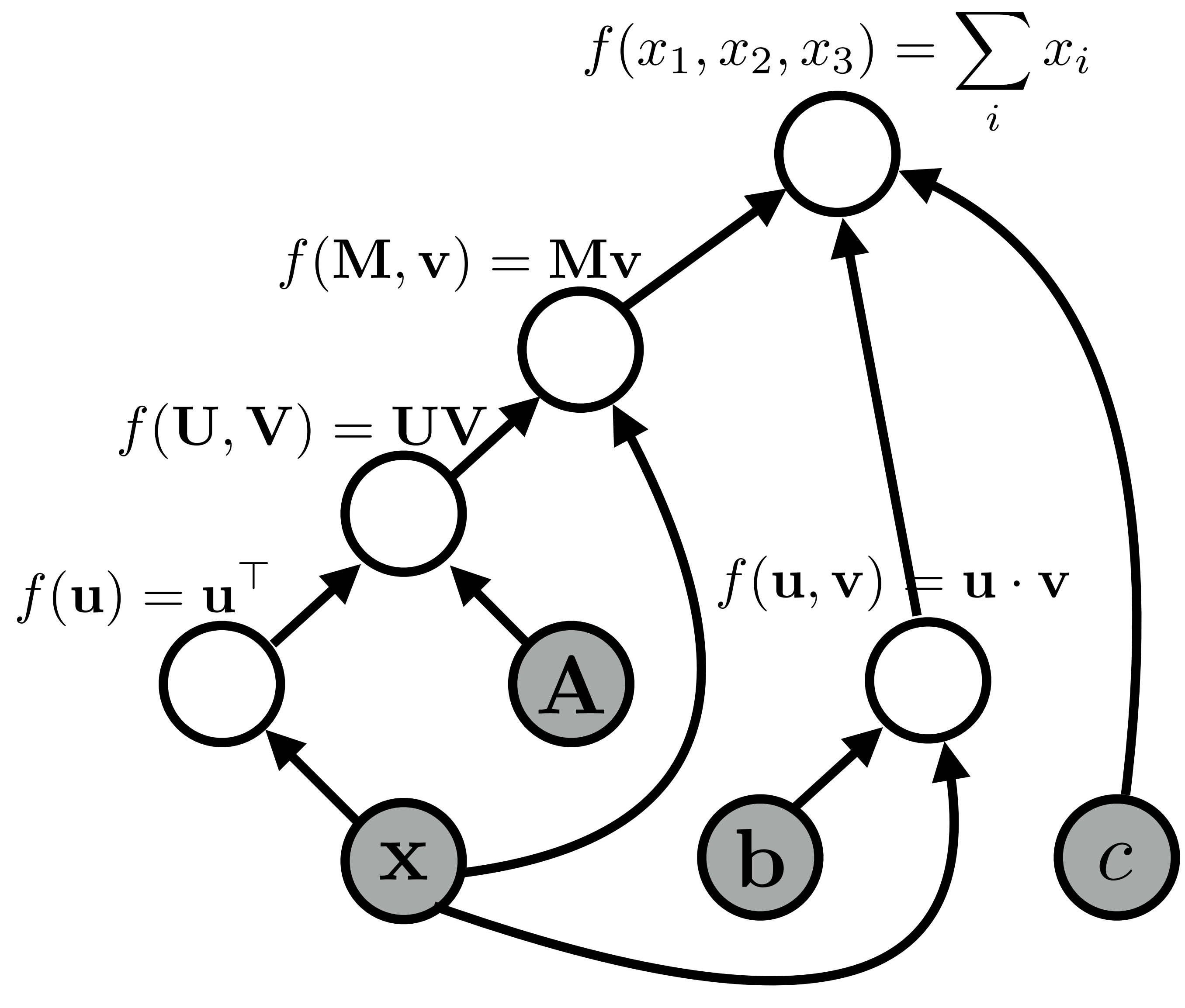
Forward Propagation

graph:



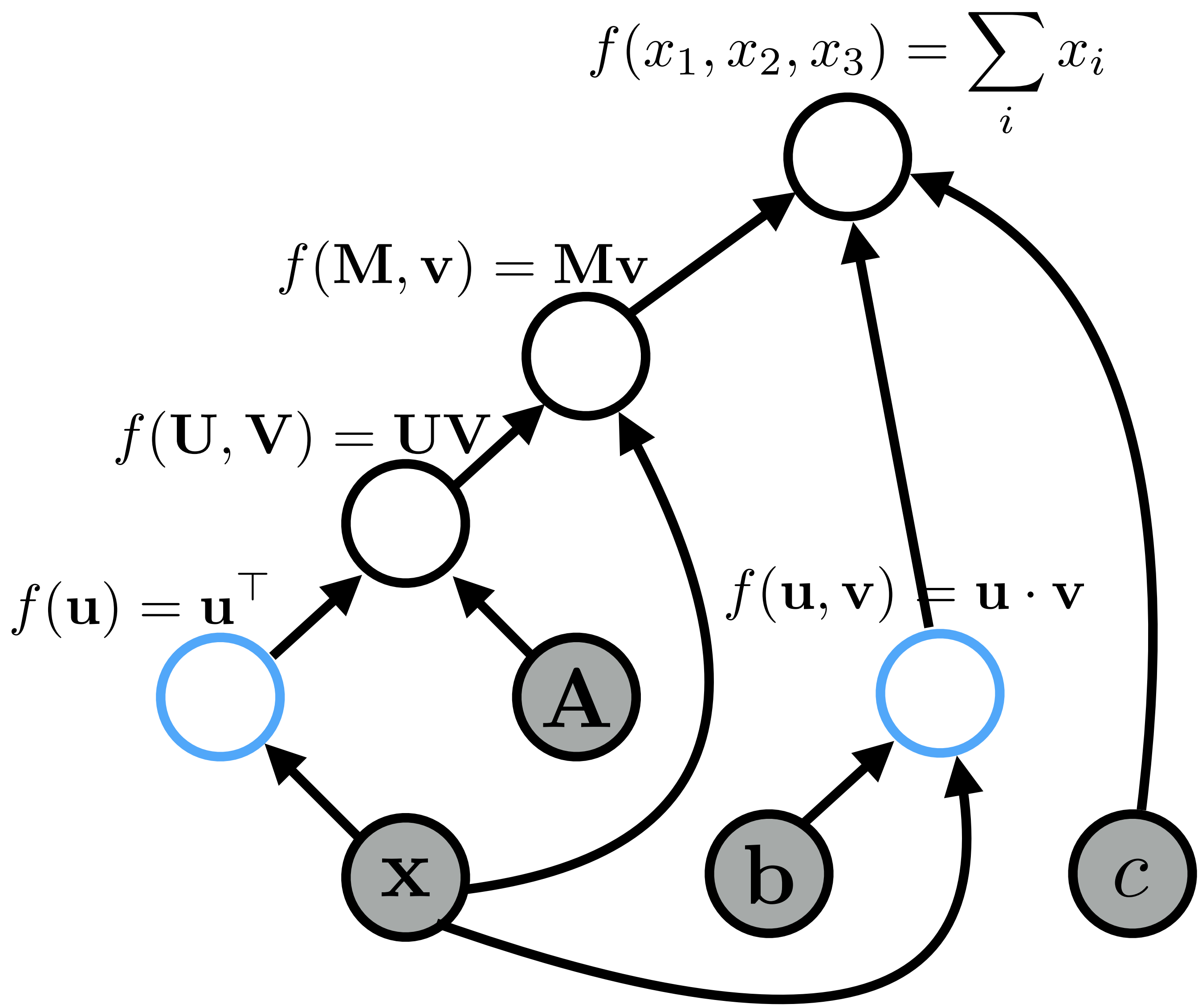
Forward Propagation

graph:



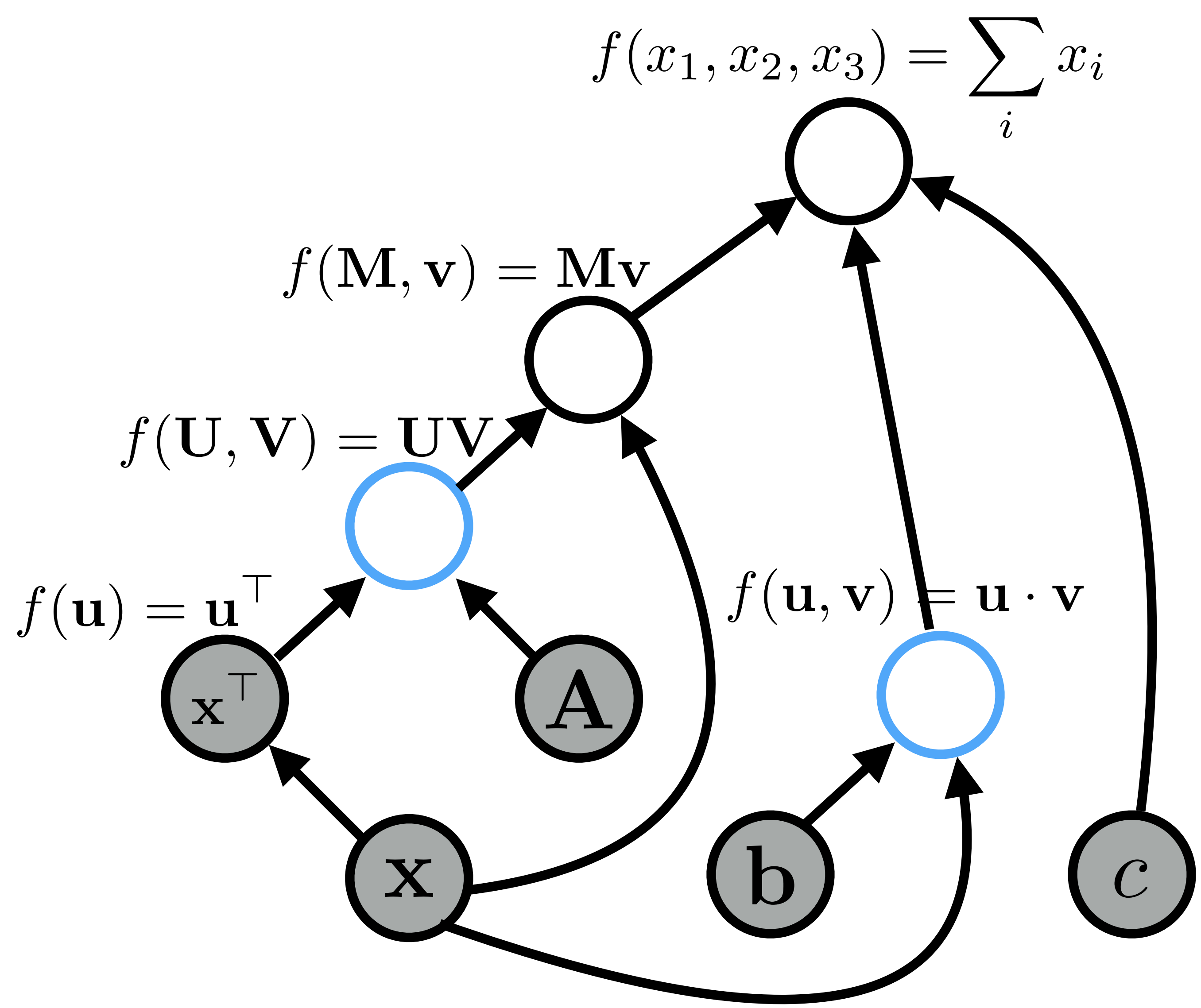
Forward Propagation

graph:



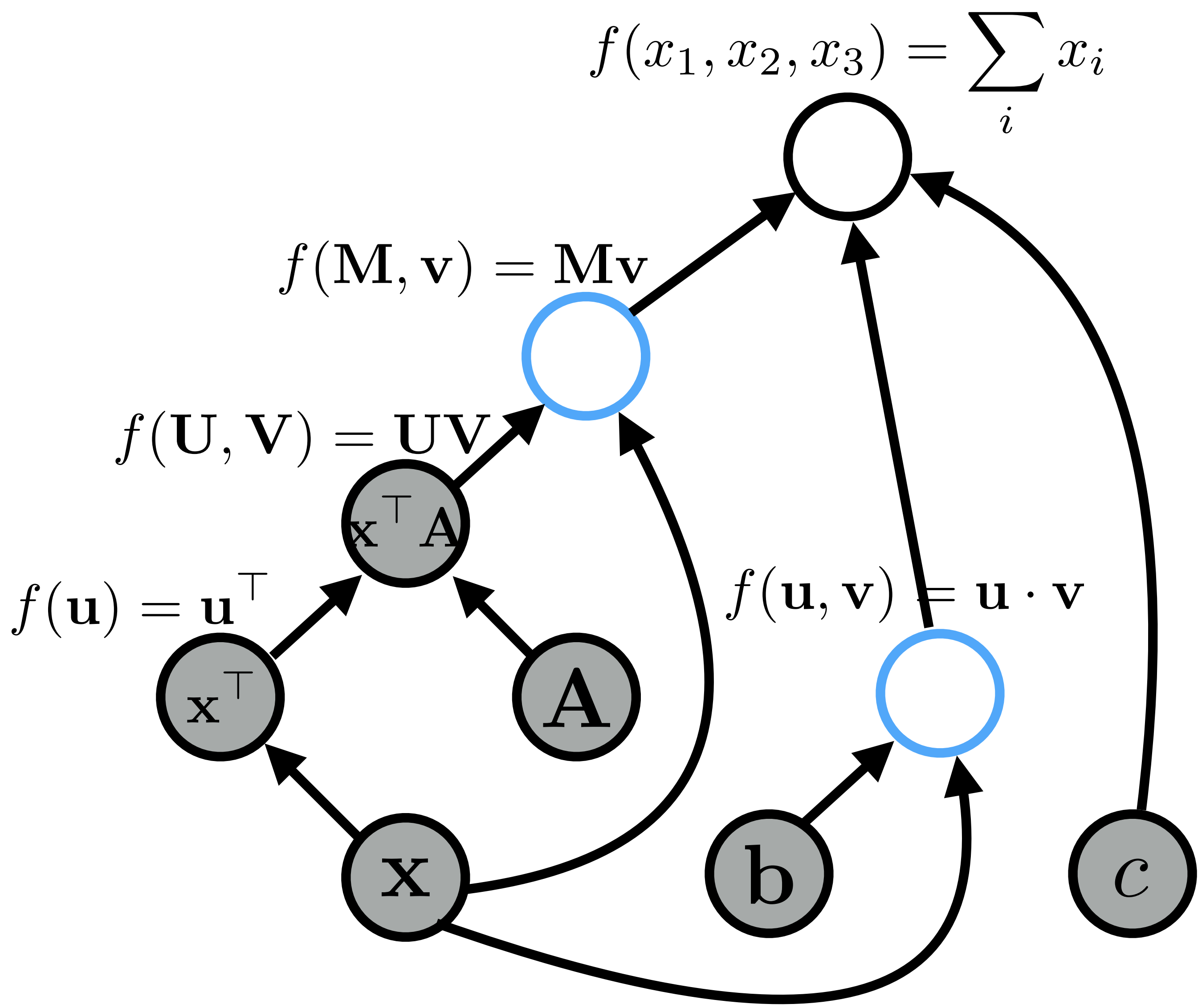
Forward Propagation

graph:



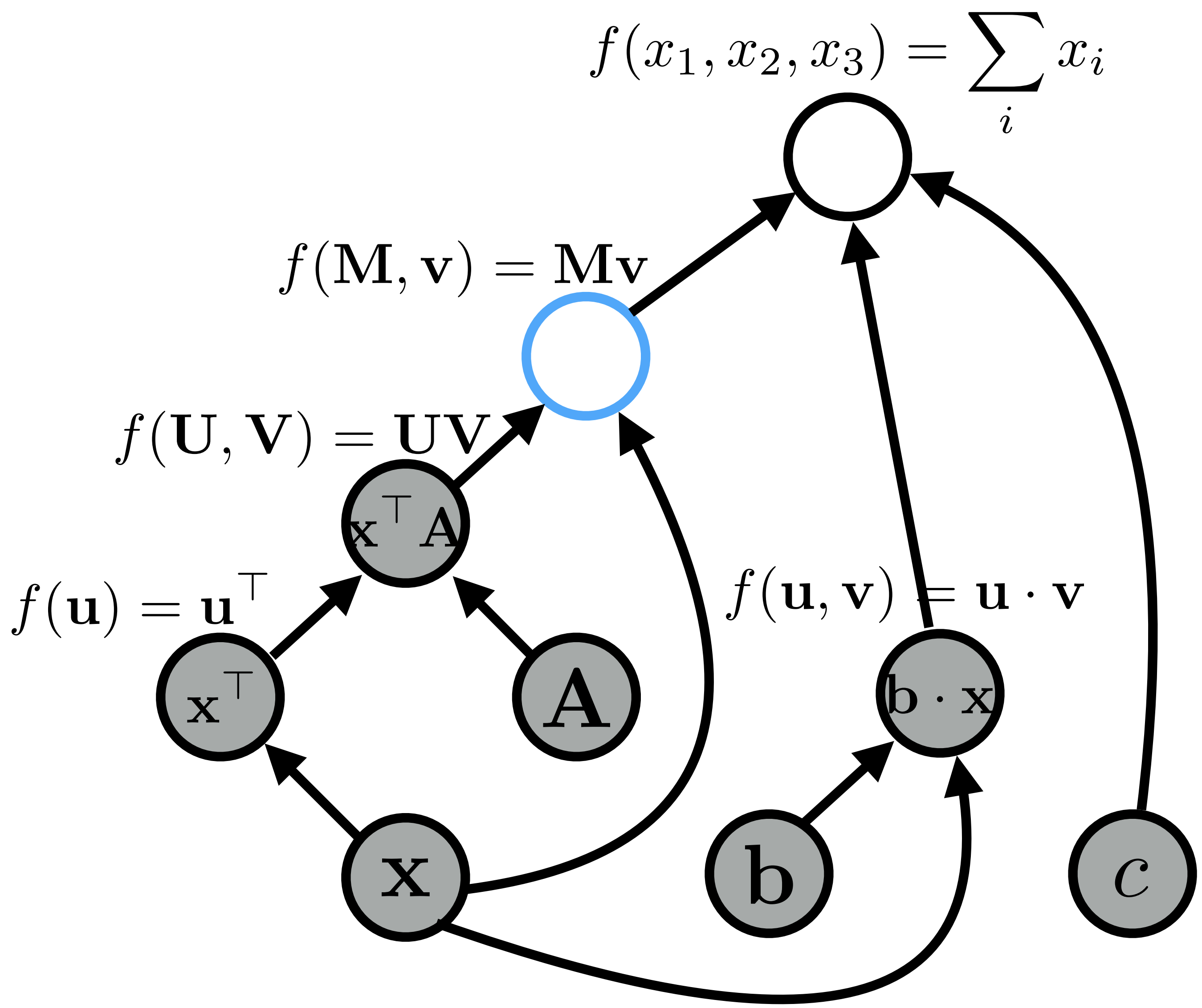
Forward Propagation

graph:



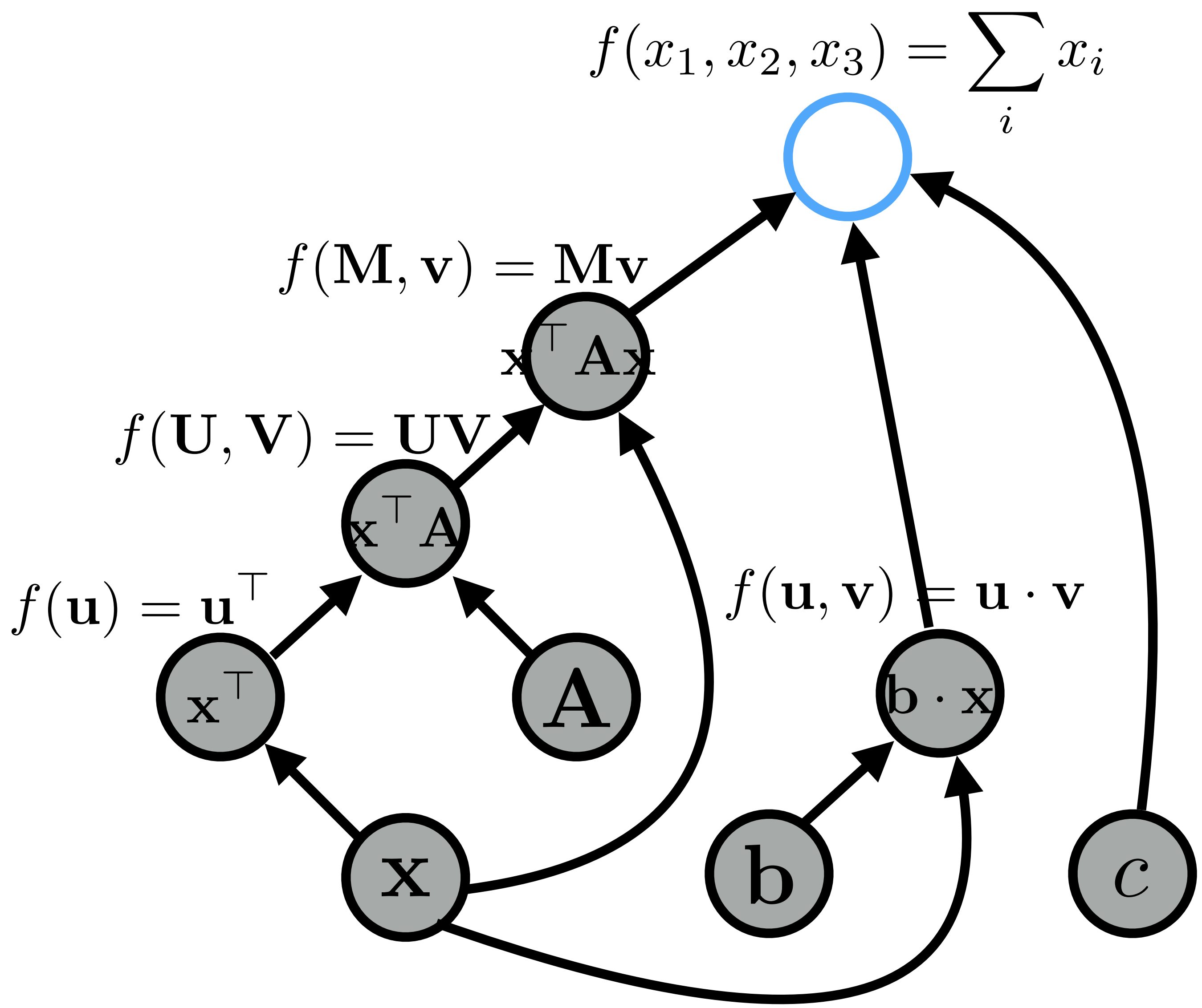
Forward Propagation

graph:



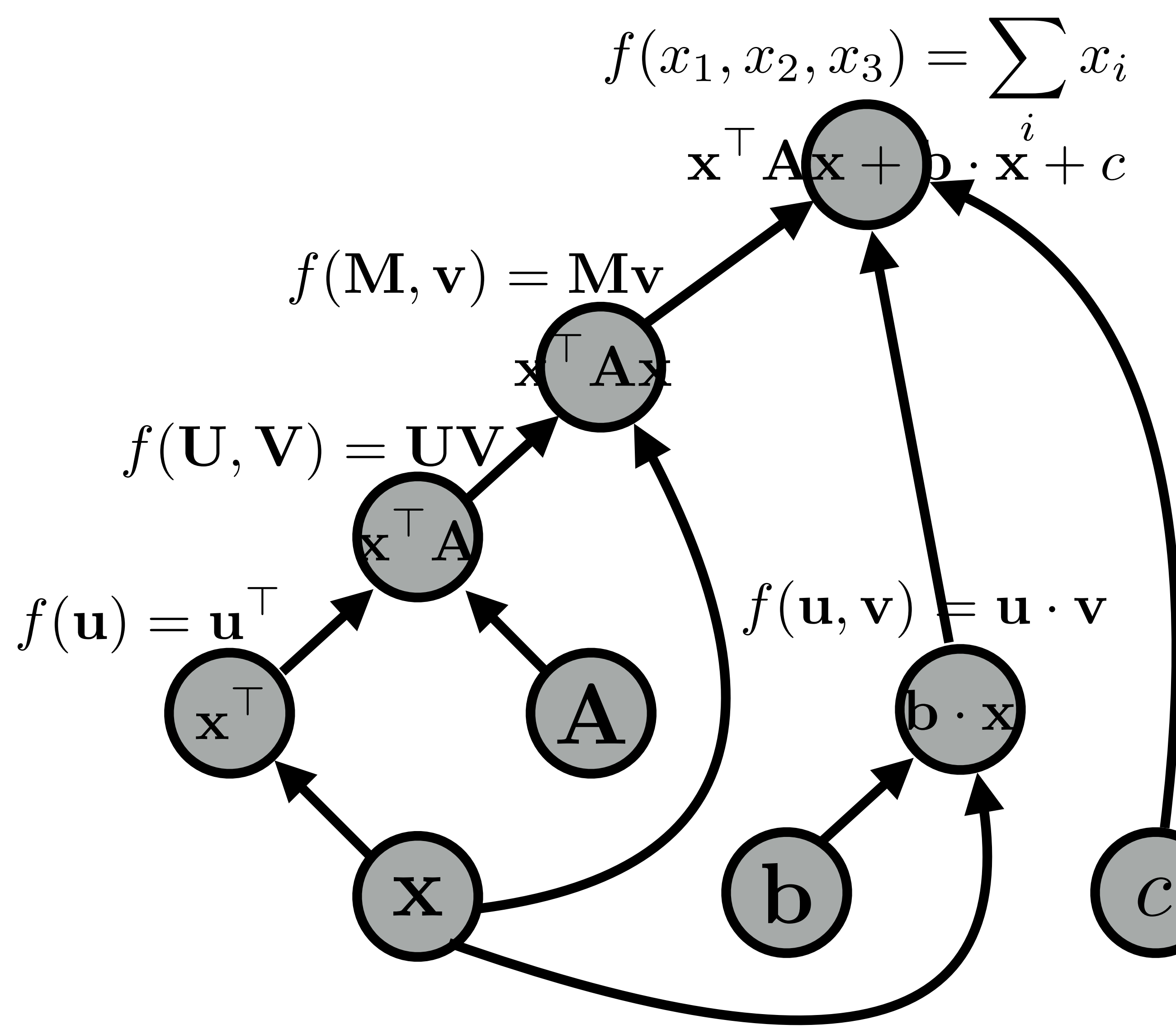
Forward Propagation

graph:



Forward Propagation

graph:



Algorithms (2)

- **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
    $W \mathrel{-}= \alpha * dl/dW$

Concrete Implementation Examples

Neural Network Frameworks

<u>Static Frameworks</u>

theano

Caffe

mxnet

TensorFlow

<u>Dynamic Frameworks (Recommended!)</u>

dy/net

Chainer

PYTORCH

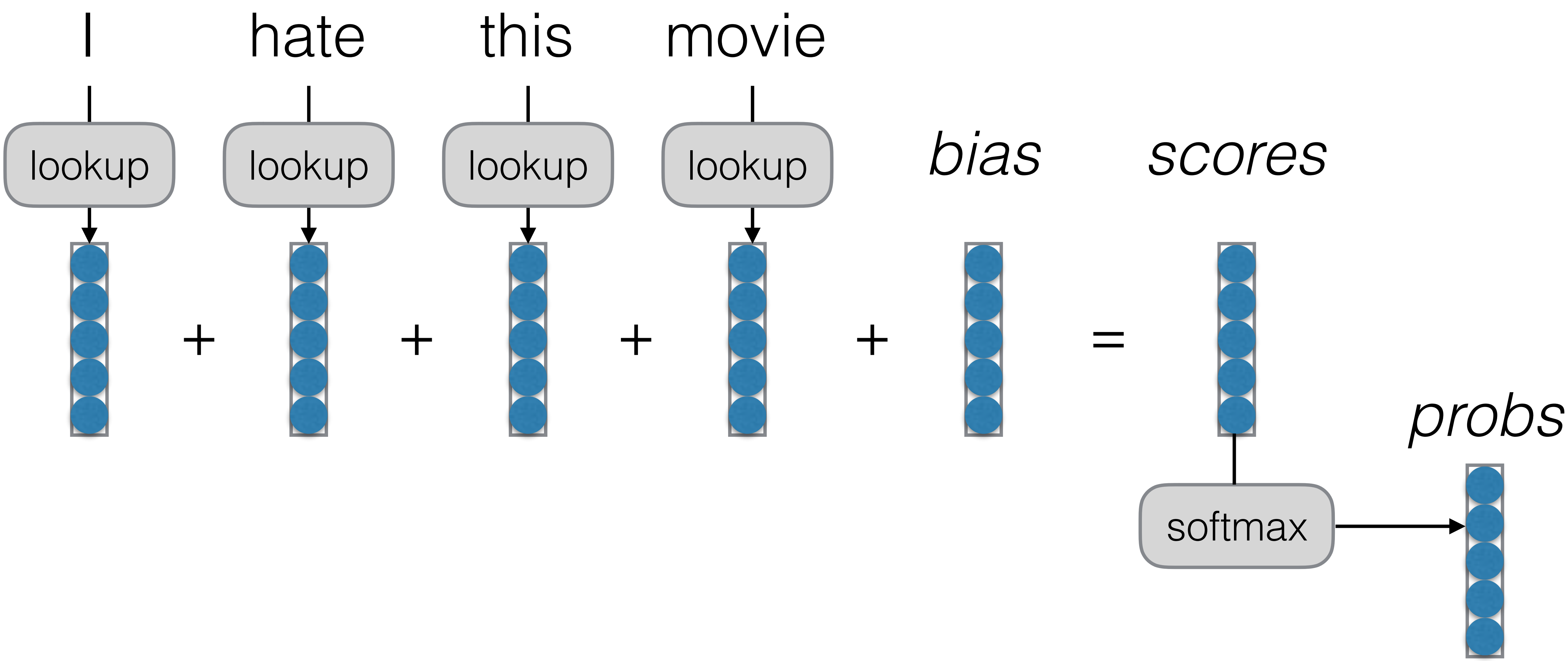
+Gluon

+Eager

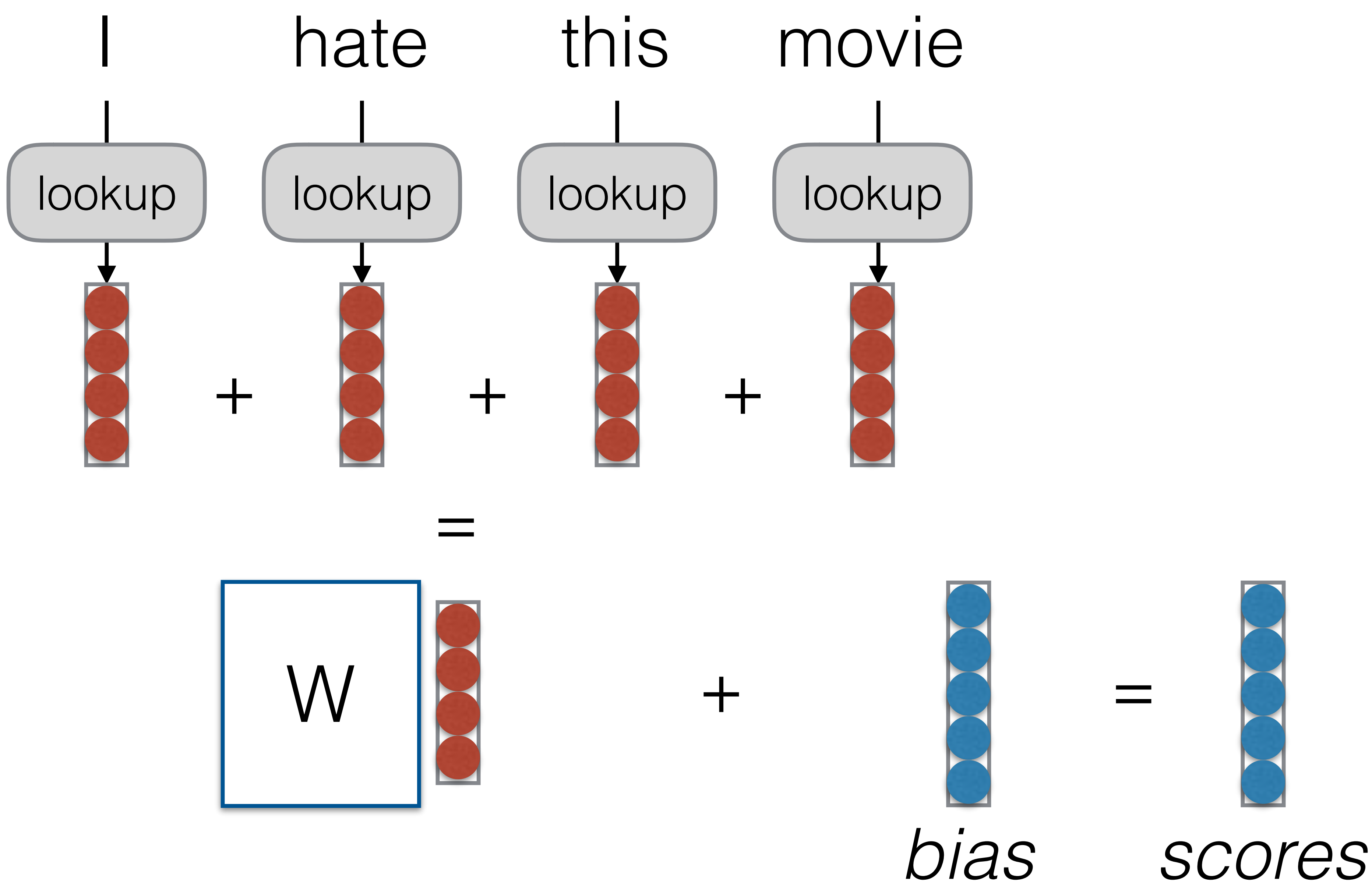
Basic Process in Dynamic Neural Network Frameworks

- Create a model
- For each example
  - **create a graph** that represents the computation you want
  - **calculate the result** of that computation
  - if training, perform **back propagation and update**

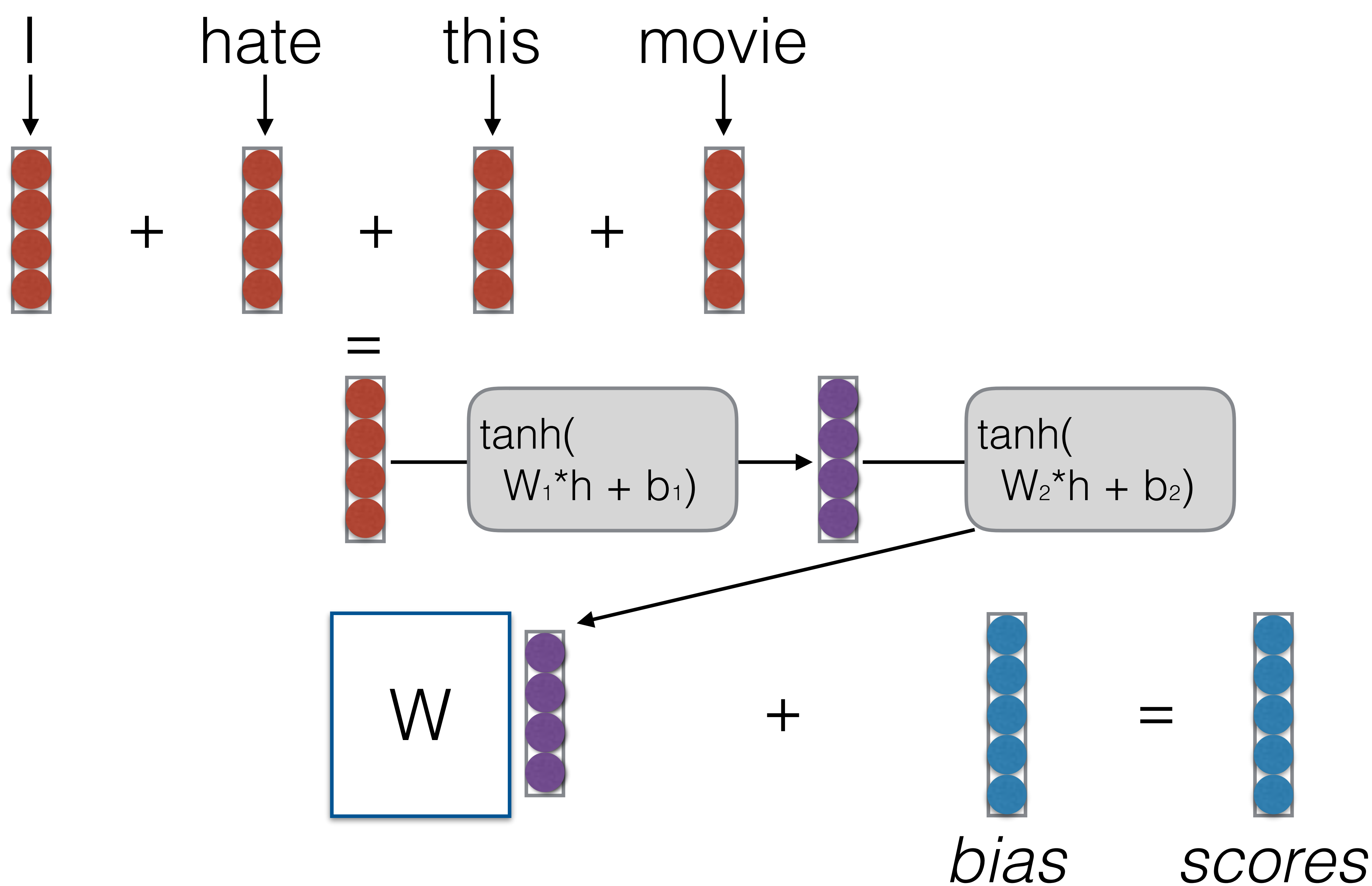
Bag of Words (BOW)



Continuous Bag of Words (CBOW)



Deep CBOW



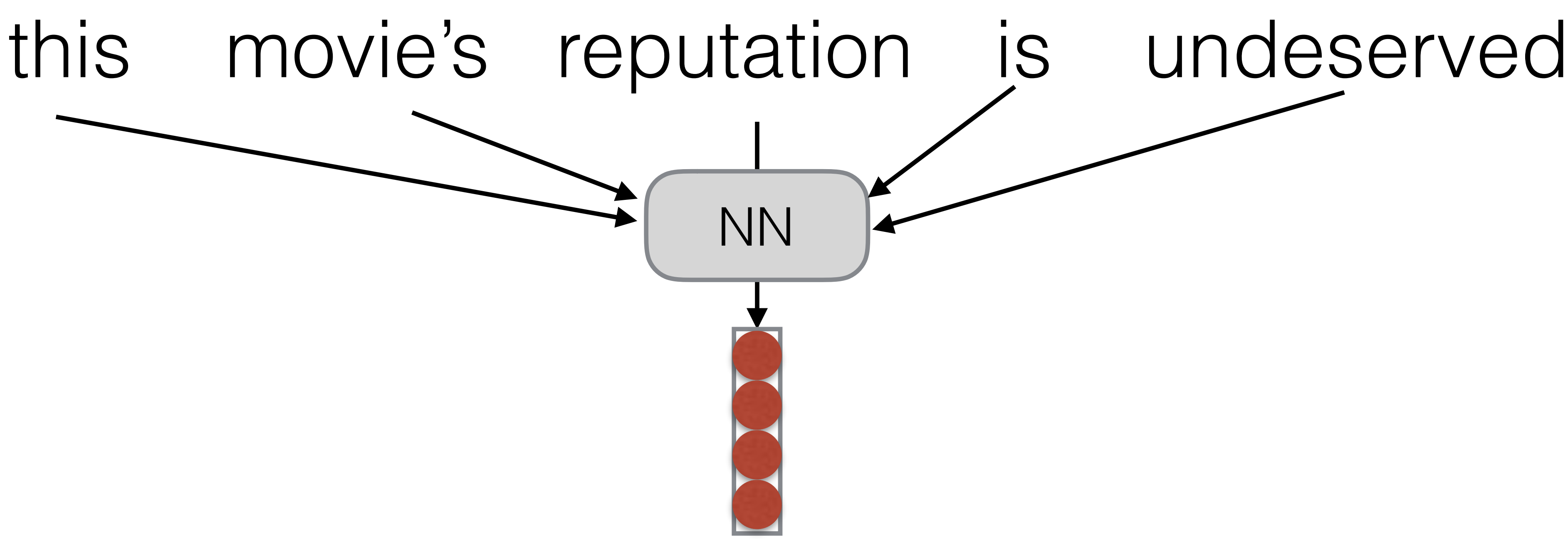
Things to Remember Going Forward

Things to Remember

- Neural nets are powerful!
    - They are universal function approximators, can calculate any continuous function
- But language is hard, and data is limited.
    - We need to design our networks to have inductive bias, to make it easy to learn things we'd like to learn.

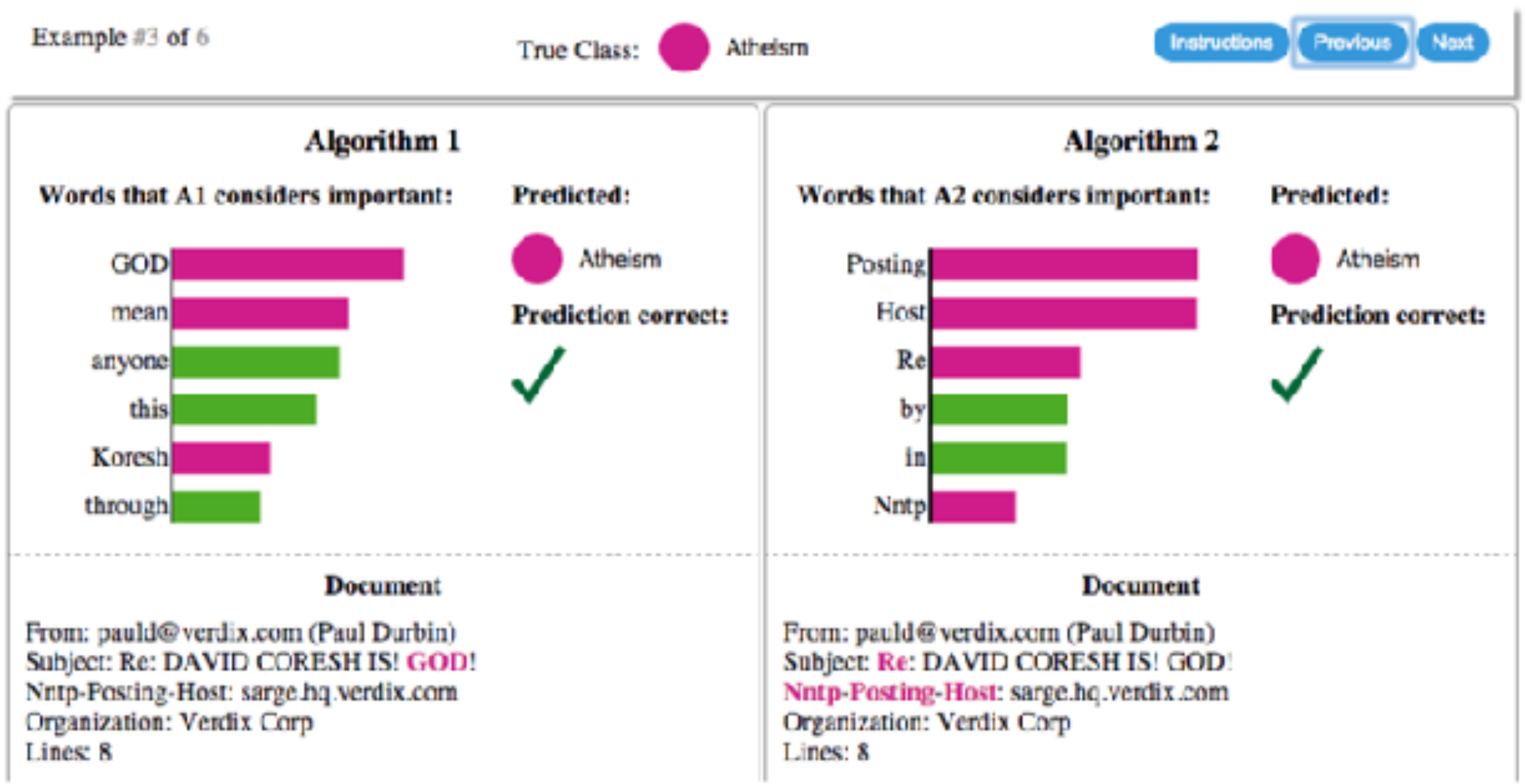
Class Plan

Topic 1:
Models of Sentences/Sequences



- Bag of words, bag of n-grams
- Convolutional nets
- Recurrent neural networks and variations
- Modeling documents and longer texts

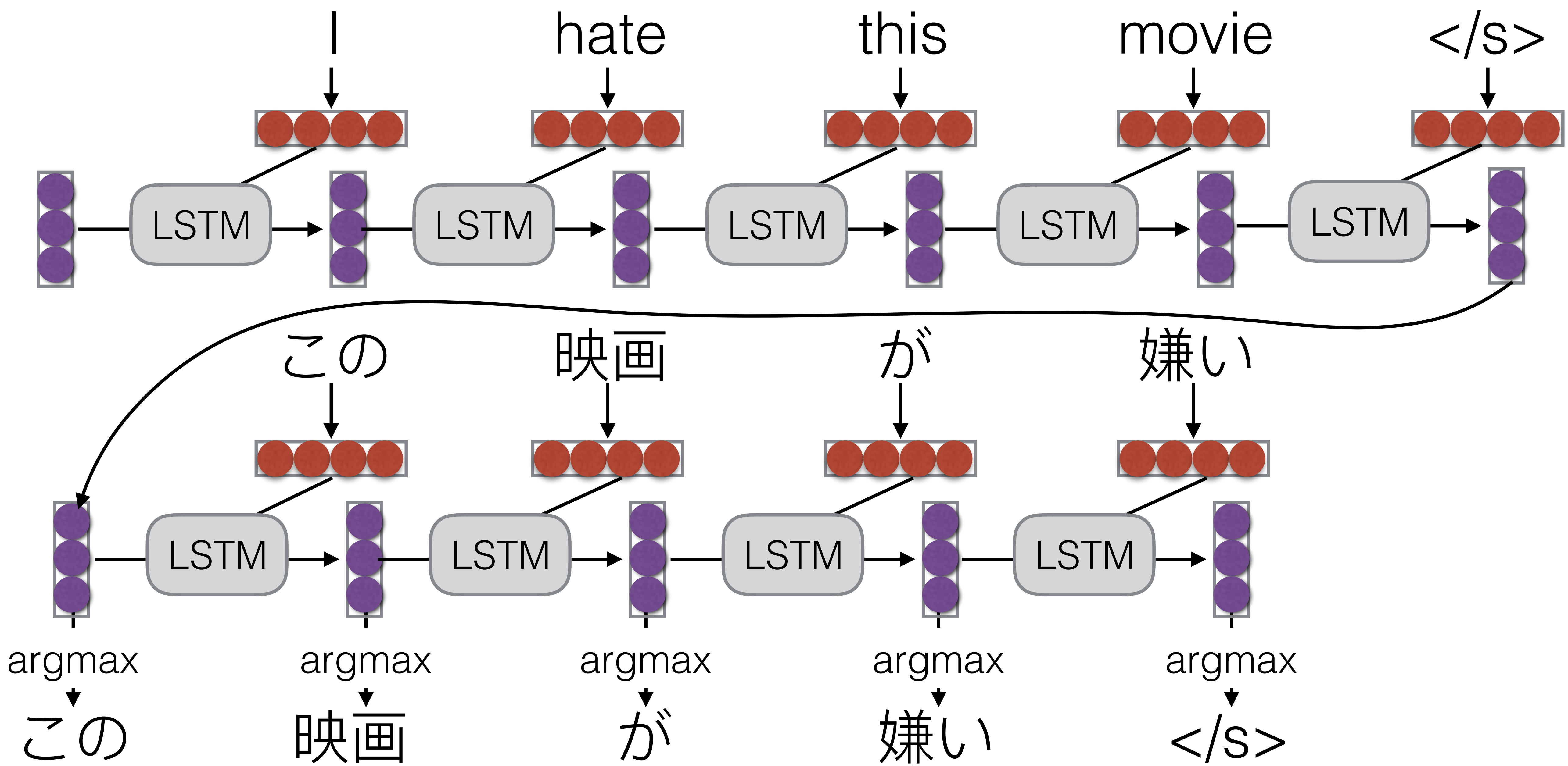
Topic 2:
Implementing, Debugging, and Interpreting



Example:
[Ribeiro+ 16]

- **Implementation:** How to efficiently and effectively implement your models
- **Debugging:** How to find problems in your implemented models
- **Interpretation:** How to find why your model made a prediction?

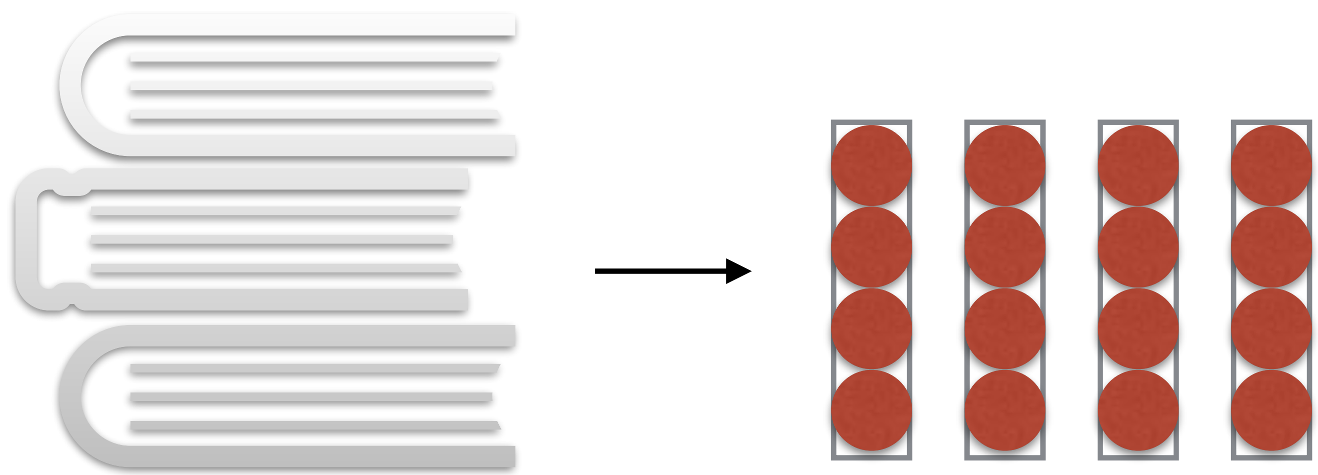
Topic 3: Conditioned Generation



- Encoder decoder models

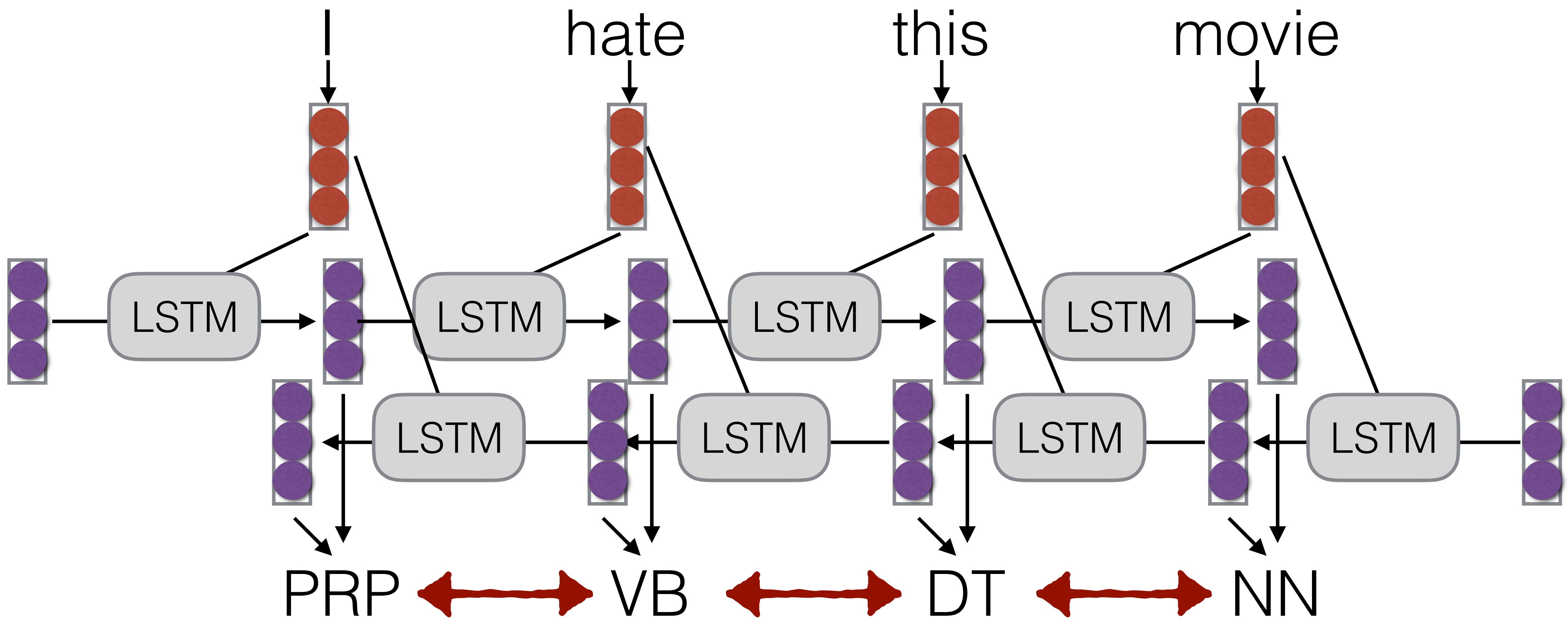
- Attentional models, self-attention (Transformers)

Topic 4: Pre-trained Embeddings



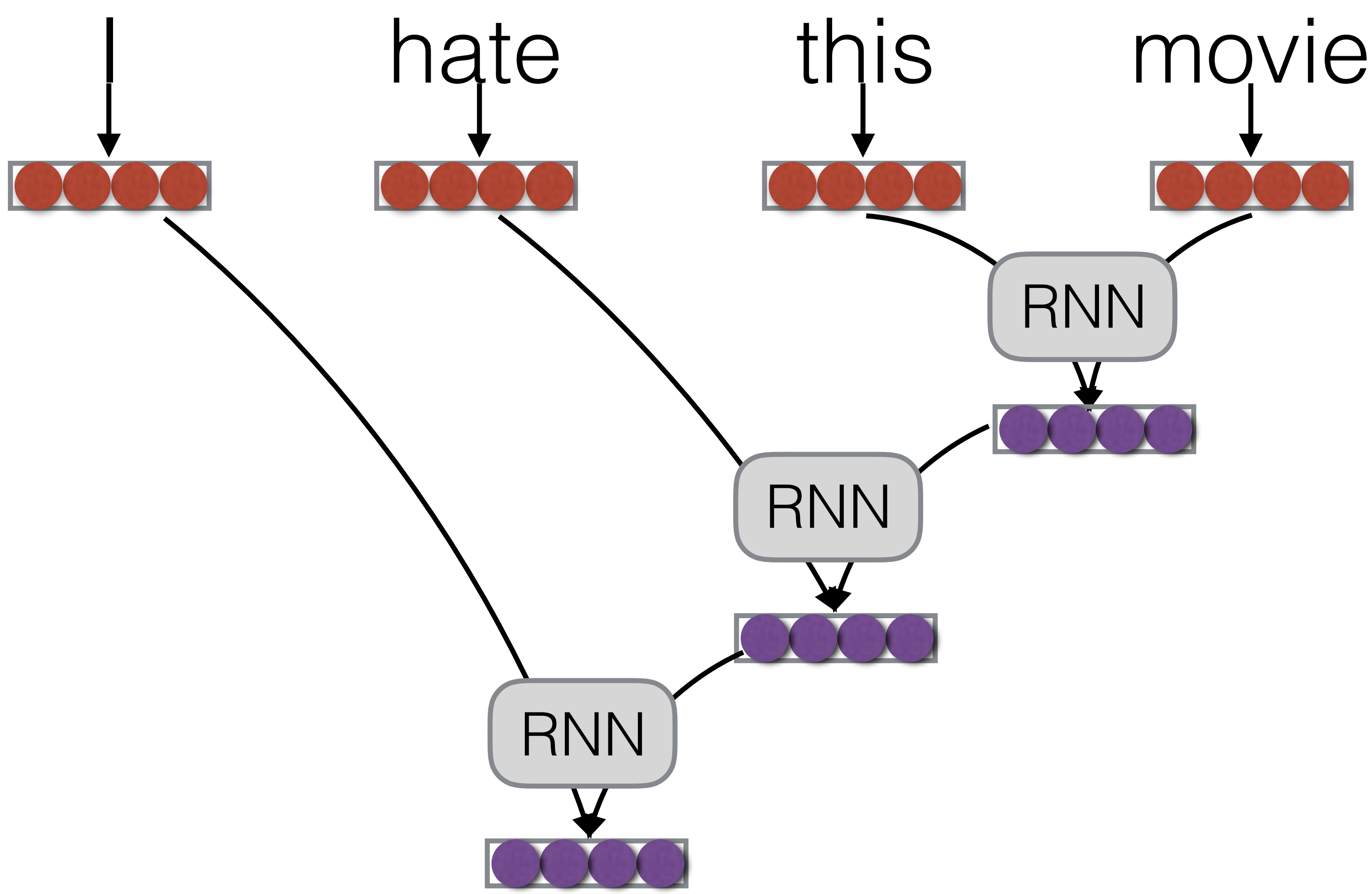
- Pre-training word embeddings, contextualized word embeddings, sentence embeddings
- Design decisions in pre-training: model, data, objective

Topic 5: Structured Prediction Models



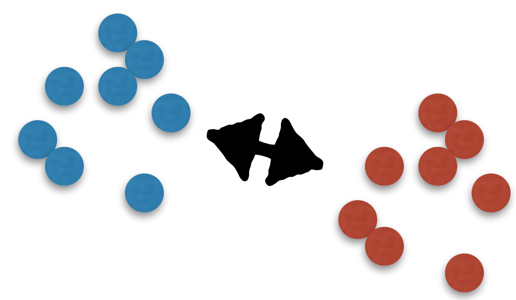
- CRFs, and other marginalization-based training
- REINFORCE, minimum risk training
- Margin-based and search-based training methods
- Advanced search algorithms

Topic 6:
Models of Tree/Graph Structures



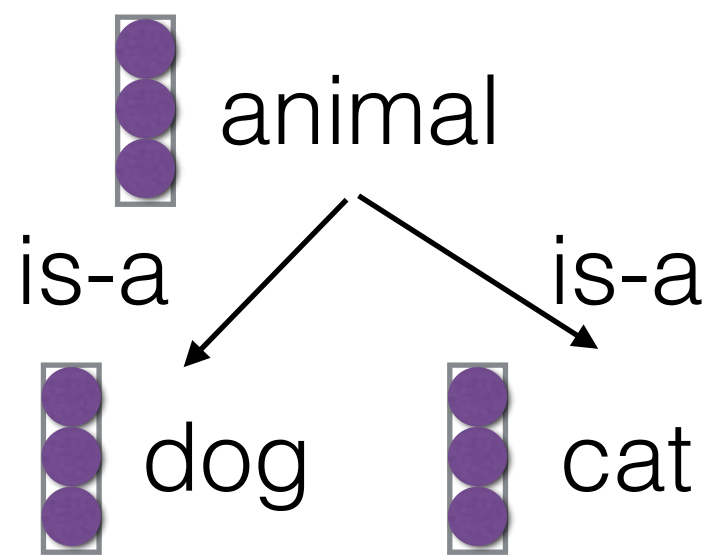
- Shift reduce, minimum spanning tree parsing
- Tree structured compositions
- Models of graph structures

Topic 7:
Advanced Learning Techniques



- Models with Latent Random Variables
- Adversarial Networks
- Semi-supervised and Unsupervised Learning

Topic 8:
Knowledge-based and Text-based QA



- Learning and QA over knowledge graphs
- Machine reading and text-based QA

Topic 9:
Multi-task and Multilingual Learning

I hate this movie

この 映画 が 嫌い

PRP VB DT NN

- Multi-task and transfer learning
- Multilingual learning of representations

Any Questions?