Frameworks

Jordan Boyd-Graber

University of Maryland

Introduction

Slides adapted from Chris Dyer, Yoav Goldberg, Graham Neubig

Neural Nets and Language

Language

Discrete, structured (graphs, trees)

Neural-Nets

Continuous: poor native support for structure

Big challenge: writing code that translates between the {discrete-structured, continuous} regimes

Why not do it yourself?

- Hard to compare with exting models
- Obscures difference between model and optimization
- Debugging has to be custom-built
- Hard to tweak model

Outline

- Computation graphs (general)
- Neural Nets in PyTorch
- Full example

Computation Graphs

Expression

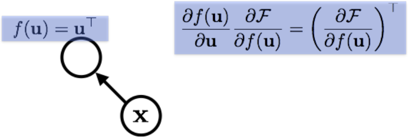
$\vec{x}$

graph:

Computation Graphs

Expression

$\vec{x}^\top$



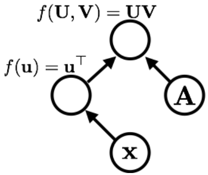
- Edge: function argument / data dependency
- A node with an incoming edge is a function $F \equiv f(u)$ edge's tail node
- A node computes its value and the value of its derivative w.r.t each argument (edge) times a derivative $\frac{\partial f}{\partial u}$

Computation Graphs

Expression

$\vec{x}^\top A$

graph:



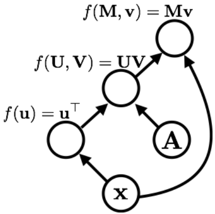
Functions can be nullary, unary, binary, . . . n-ary. Often they are unary or binary.

Computation Graphs

Expression

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

graph:



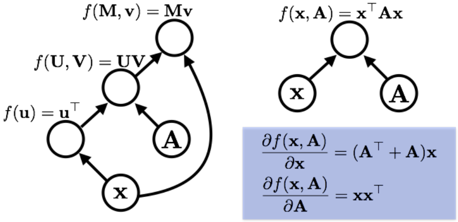
Computation graphs are (usually) directed and acyclic

Computation Graphs

Expression

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

graph:

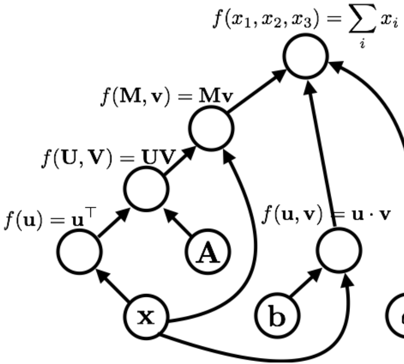


Computation Graphs

Expression

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

graph:

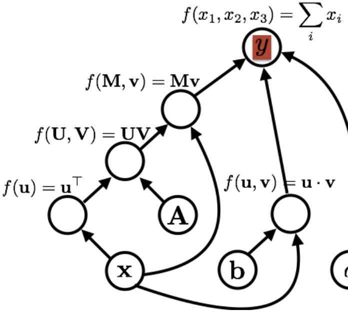


Computation Graphs

Expression

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

graph:

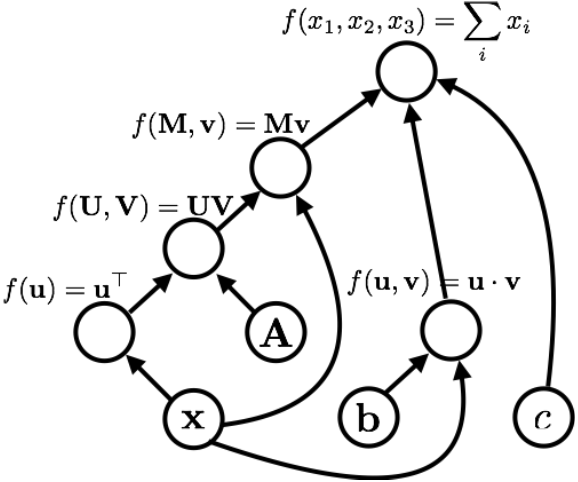


Variable names label nodes

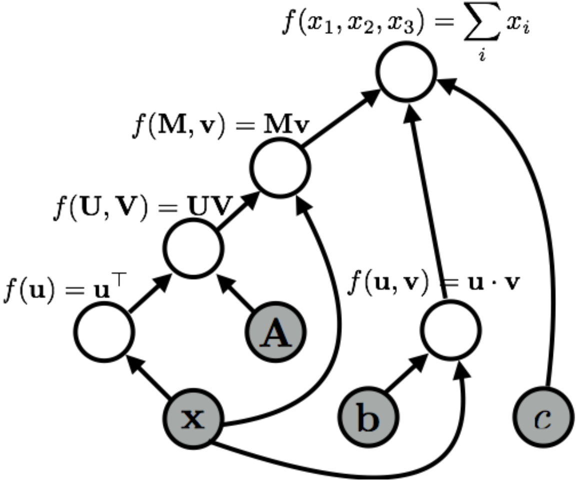
Algorithms

- Graph construction
- Forward propagation
  - Loop over nodes in topological order
  - Compute the value of the node given its inputs
  - Given my inputs, make a prediction (i.e. “error” vs. “target output”)
- Backward propagation
  - Loop over the nodes in reverse topological order, starting with goal node
  - Compute derivatives of final goal node value wrt each edge’s tail node
  - How does the output change with small change to inputs?

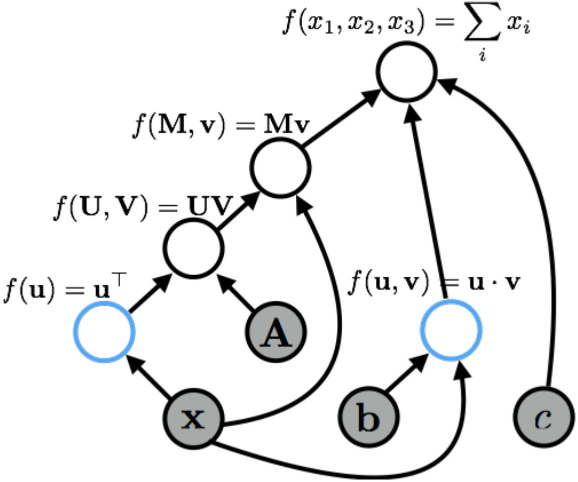
Forward Propagation



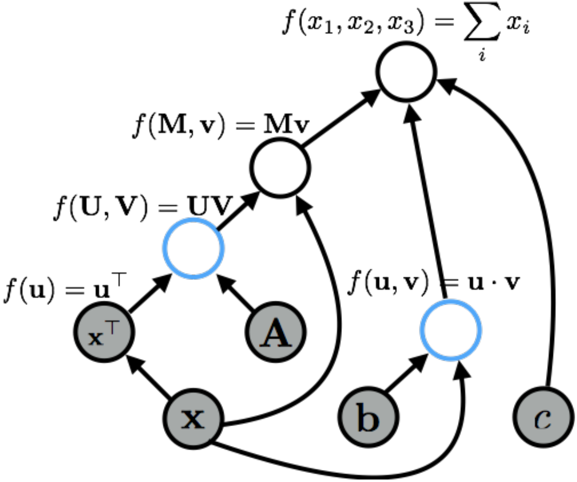
Forward Propagation



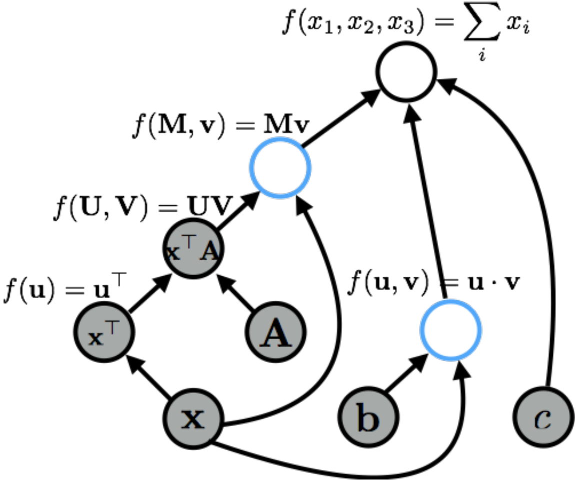
Forward Propagation



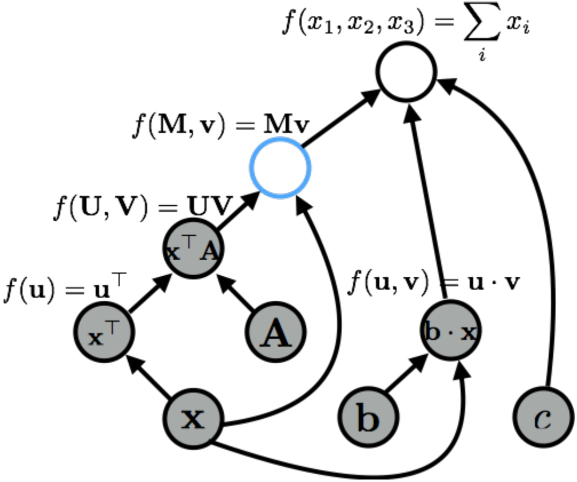
Forward Propagation



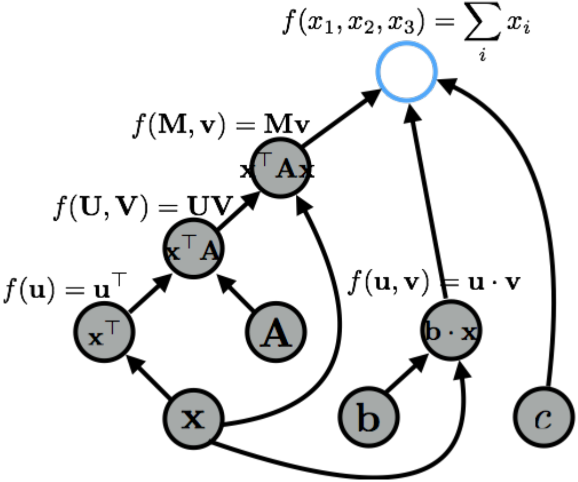
Forward Propagation



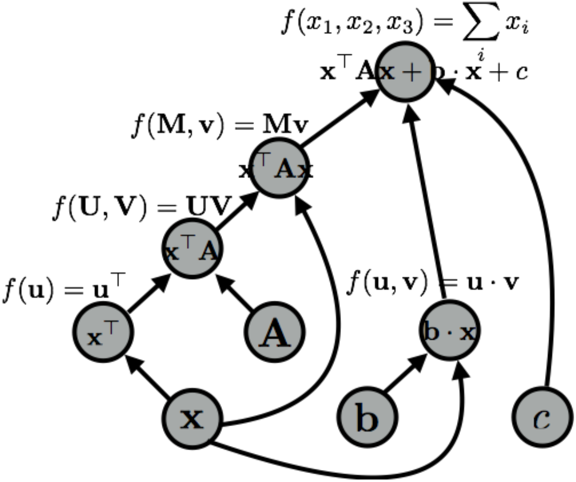
Forward Propagation



Forward Propagation



Forward Propagation



Constructing Graphs

Static declaration

- Define architecture, run data through
- PROS: Optimization, hardware support
- CONS: Structured data ugly, graph language

Theano, Tensorflow

Dynamic declaration

- Graph implicit with data
- PROS: Native language, interleave construction/evaluation
- CONS: Slower, computation can be wasted

Chainer, Dynet, PyTorch

Constructing Graphs

Static declaration

- Define architecture, run data through
- PROS: Optimization, hardware support
- CONS: Structured data ugly, graph language

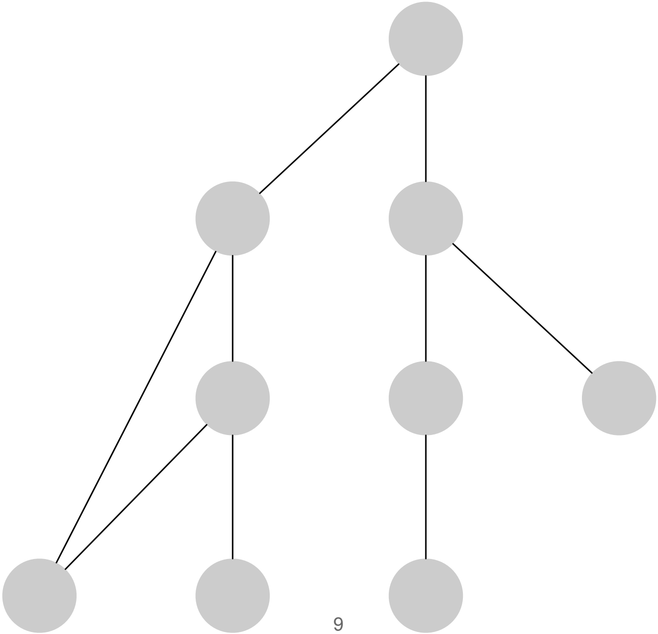
Theano, Tensorflow

Dynamic declaration

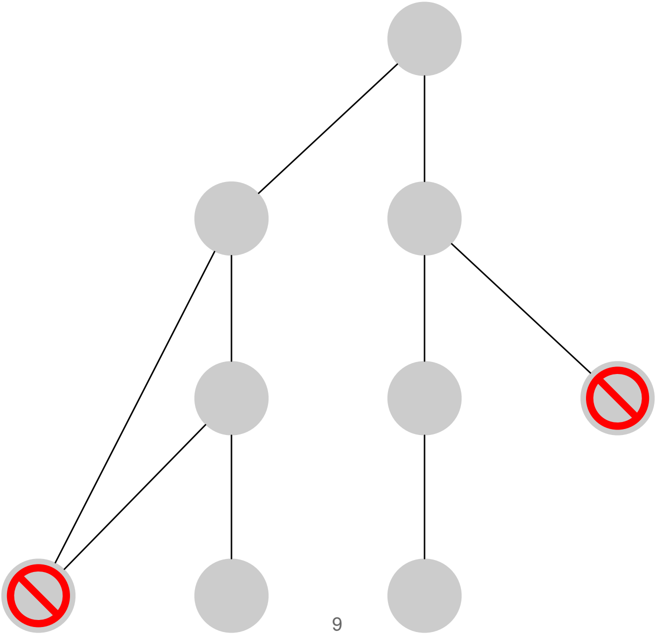
- Graph implicit with data
- PROS: Native language, interleave construction/evaluation
- CONS: Slower, computation can be wasted

Chainer, Dynet, PyTorch

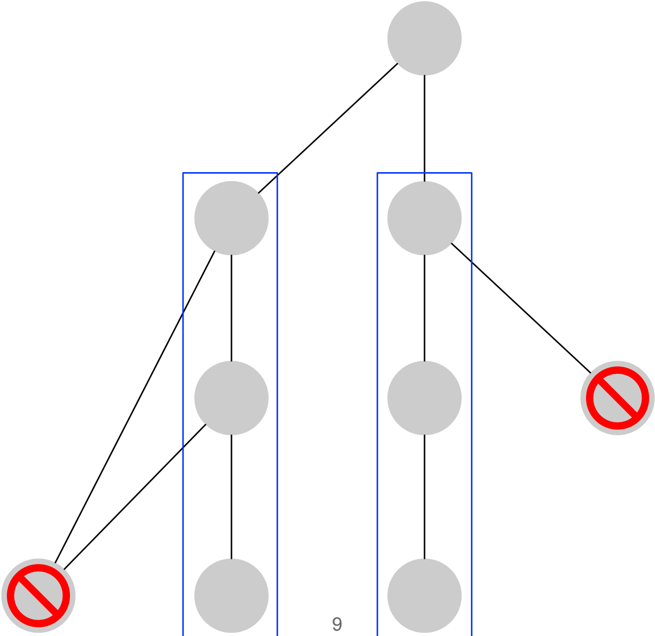
Advantage of Dynamic Declaration



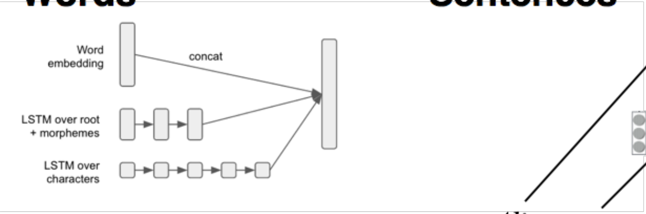
Advantage of Dynamic Declaration



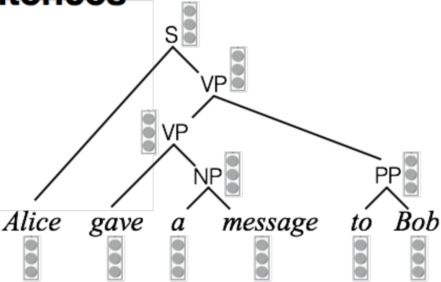
Advantage of Dynamic Declaration



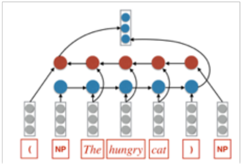
Words



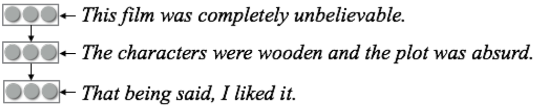
Sentences



Phrases



Documents



Language is Hierarchical

Dynamic Hierarchy in Language

- Language is hierarchical
  - Graph should reflect this reality
  - Traditional flow-control best for processing
- Combinatorial algorithms (e.g., dynamic programming)
- Exploit independencies to compute over a large space of operations tractably

PyTorch

- Torch: Facebook’s deep learning framework
- Nice, but written in Lua (C backend)
- Optimized to run computations on GPU
- Mature, industry-supported framework

Why GPU?

Why GPU?