Analyzing Neural Networks with Gradient Tracing

Brian DuSell

Application

Identifying neural network components that are most influential in training

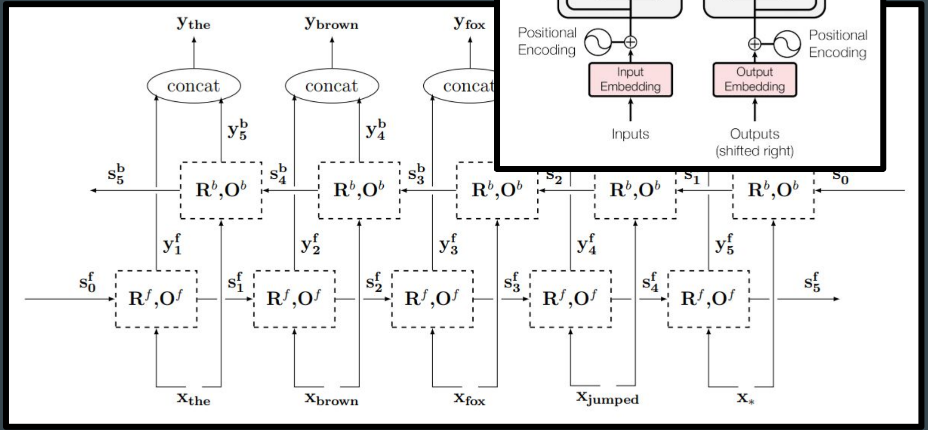
Kernel

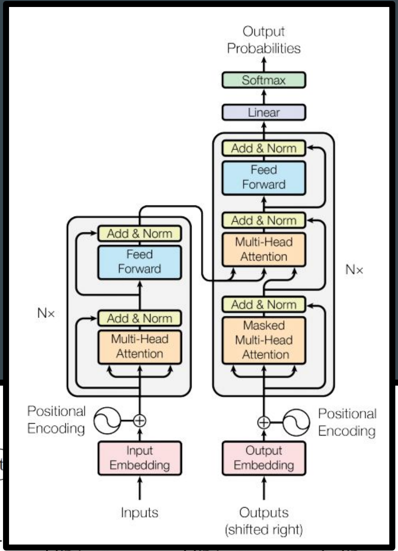
"Gradient tracing"

- Finds the path in the computation graph through which the most gradient propagates
- Starting from parameters, follow vertices which contributed most gradient

Neural Networks in Natural Language Processing

- State of the art in NLP tasks [1]
  - machine translation
  - language modeling
- Trained using backpropagation algorithm
  - Equivalent to computing partial derivative of loss w.r.t. parameters
  - Based on the chain rule of calculus
- Consist of a “computation graph” through which gradient flows
- Notorious for being black boxes





Interpreting Neural Networks

- Ablation study
  - Hack off components
- Design for interpretability
  - Attention [3]
  - Input-switched affine networks [4]
- Analytical methods
  - Rational recurrences [5]
  - Weighted sum [6]

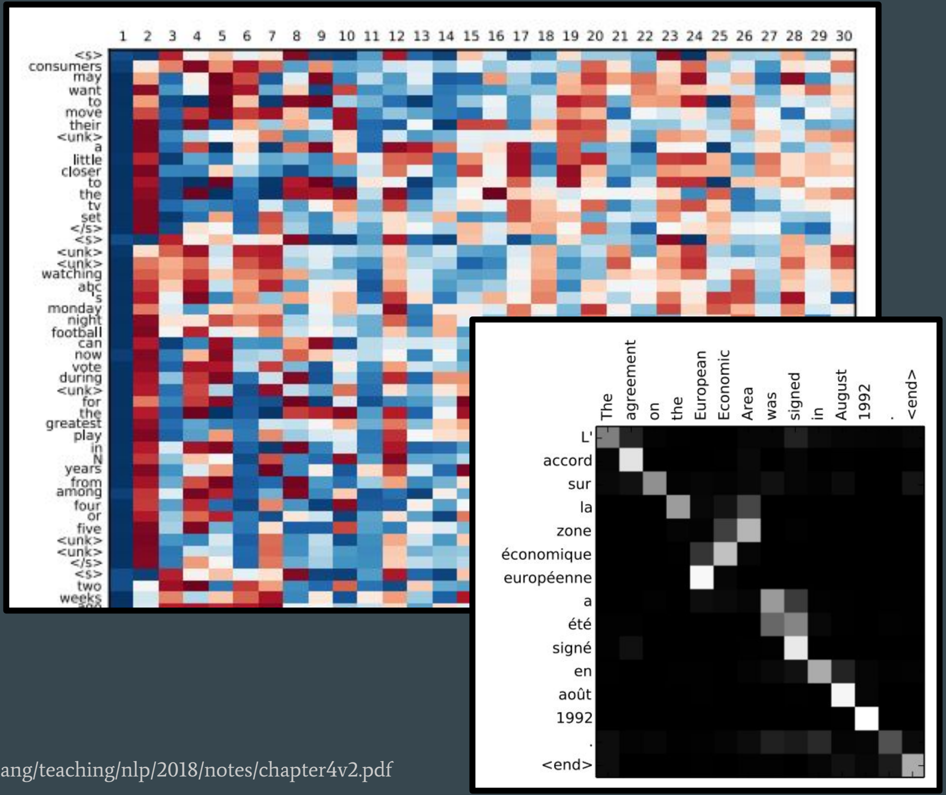
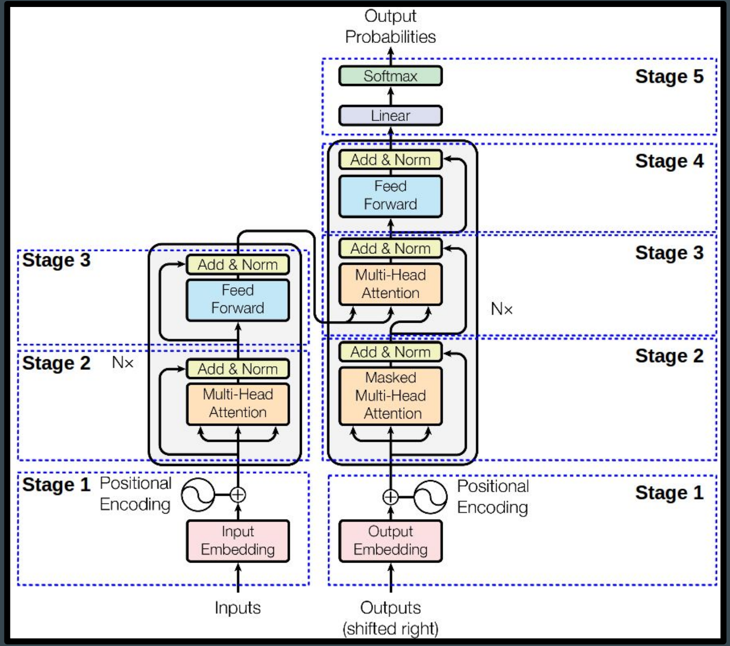


Image credits:
https://www3.nd.edu/~dchiang/teaching/nlp/2018/notes/chapter4v2.pdf
and [3]

Neural Networks with Multiple Components

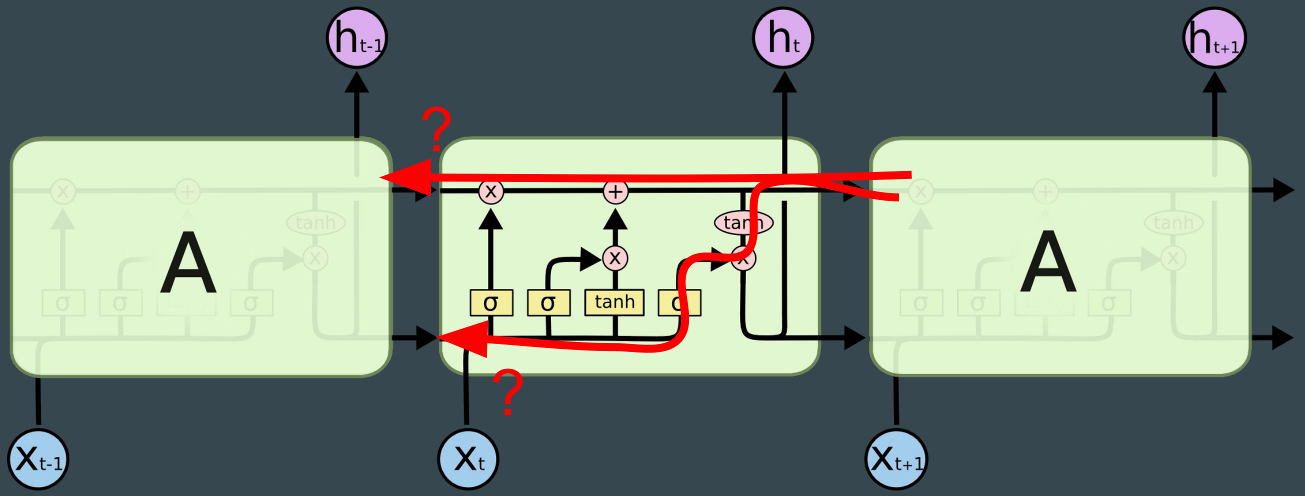
- Modern neural network architectures have multiple components
- Often used to make training feasible
- Classic example: LSTMs



The Transformer network, a state-of-the-art machine translation system [2].

Image credit: https://mchromiak.github.io/articles/2017/Sep/12/Transformer-Attention-is-all-you-need/

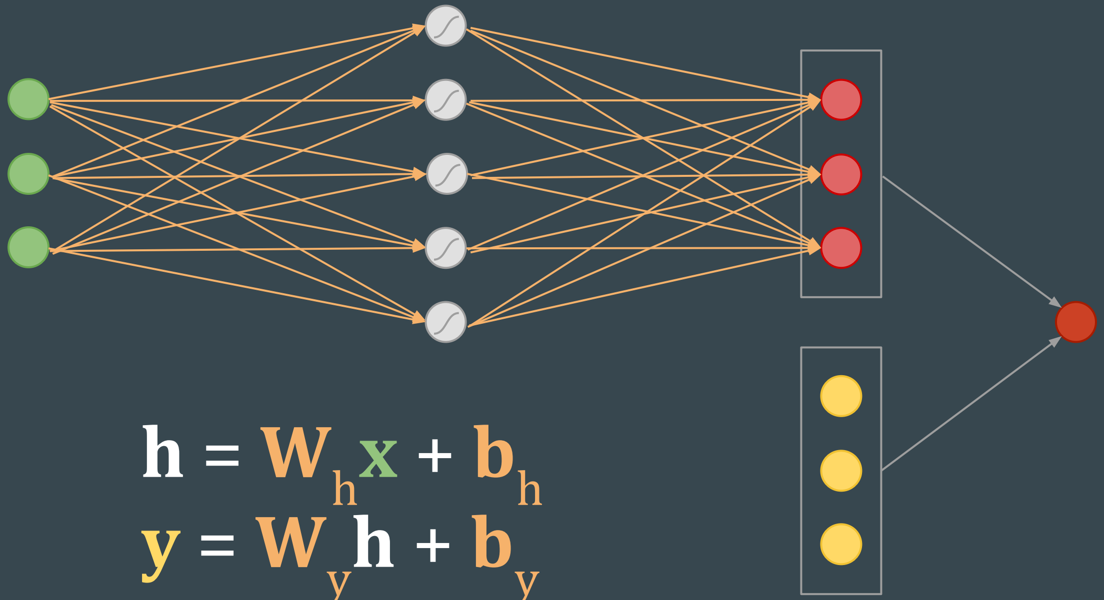
Neural Networks with Multiple Components



Architectural diagram of LSTM

Image credit: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Neural Network Basics



$$\mathbf{h} = \mathbf{W}_h \mathbf{x} + \mathbf{b}_h$$

$$\mathbf{y} = \mathbf{W}_y \mathbf{h} + \mathbf{b}_y$$

$$L(\mathbf{y}, \hat{\mathbf{y}})$$

Input

Target output

Parameters

Predicted output

Loss function

Computation Graphs

- Like an abstract syntax tree
- Always a DAG
- Vertices represent operators, constants, or parameters
- Edges are directed and represent assignments to function parameters
- Gradient propagates through graph in reverse from a root “loss” node

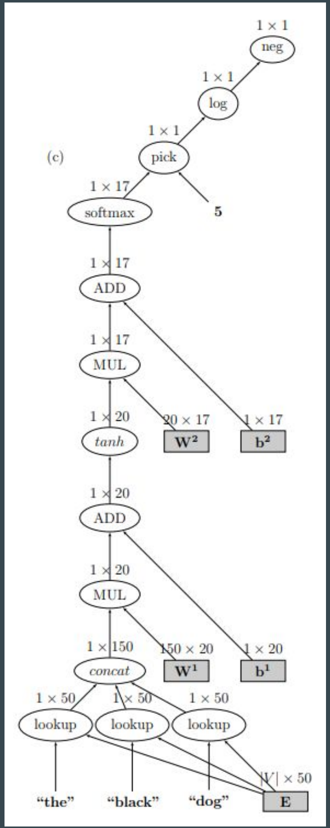


Image credit: [1]

Why Not Centrality Metrics?

- Edge weights, not graph topology, define importance
- CGs are always DAGs, which allows for optimizations
- Gradient does not “flow”
  - Amount of loss propagated depends on the operator

What’s the partial derivative of…

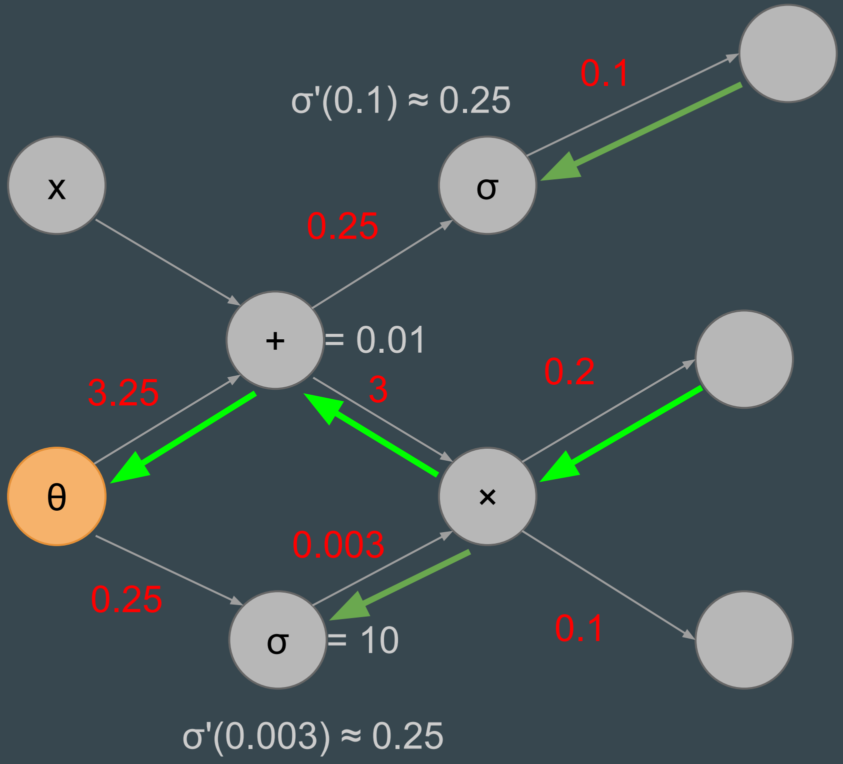
$f(x, y) = x + y$

$f(x, y) = xy$

$f(x) = 1 / (1 + e^{-x})$

Key Idea

- Follow nodes which contributed gradient with highest absolute value
- Nodes can be re-used more than once
- When a node has multiple gradients flowing back to it, the gradients are added together



Gradient Tracing Pseudocode

Assume backpropagation has already been run on the computation graph.
Let $G = (V, E)$ be the reversed computation graph, where each edge (u, v) has weight $w(u, v)$ equal to the partial derivative of the loss function L w.r.t. the parameter θ.

TraceGradient(G, θ):

  $p :=$ an empty path

  $v := v_\theta$

  while $v \neq v_L$,

    append v to p

    $v := \operatorname{argmax}_u |w(u, v)|$

  return p

Let a component C be defined as a subgraph of the computation graph.

A component C is more “important” when p contains more edges in C.

Datasets

- Graphs can be extracted from any neural model run on any dataset
- Typical “small” RNN language model
  - 10k vocab size
  - 1000 hidden units/embedding size
  - ~2.1M parameters
  - ~30M × n edges, where n is sentence length
  - LSTM or NMT system would be bigger
- Should always be small enough to fit into memory

Things I'm Still Thinking About

- How to aggregate results from multiple samples
- How to aggregate results for groups of parameters
- How to aggregate results for groups of vertices ("components")
- k-best paths

References

[1] Goldberg, Yoav. "Neural network methods for natural language processing." Synthesis Lectures on Human Language Technologies 10.1 (2017): 1-309.
[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. CoRR, abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762.
[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. CoRR, abs/1409.0473,2014. URL http://arxiv.org/abs/1409.0473.
[4] Jakob N. Foerster, Justin Gilmer, Jan Chorowski, Jascha Sohl-Dickstein, and David Sussillo. Intelligible language modeling with input switched affine networks. CoRR, abs/1611.09434, 2016. URL http://arxiv.org/abs/1611.09434.
[5] Hao Peng, Roy Schwartz, Sam Thomson, and Noah A. Smith. Rational Recurrences. URL https://arxiv.org/abs/1808.09357.
[6] Omer Levy, Kenton Lee, Nicholas FitzGerald, and Luke Zettlemoyer. Long short-term memory as a dynamically computed element-wise weighted sum. CoRR, abs/1805.03716, 2018. URL http://arxiv.org/abs/1805.03716.