Feedforward Neural Networks and Backpropagation

Changyou Chen

Department of Computer Science and Engineering
Universitpy at Buffalo, SUNY
changyou@buffalo.edu

February 19, 2019

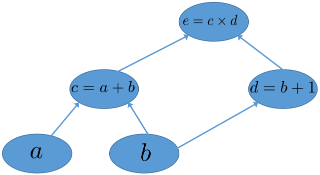
Computational Graphs

How to apply SGD in Deep Neural Networks?

1. Until now we have learned how to defined an FNN, but how to learn the model given data?
   - Learning by stochastic gradient descent (SGD).
2. To apply SGD, we need the gradients of the parameters of the neural networks for the loss function.
3. How to calculate the gradients in a general framework?
   - Backpropagation (BP).

Graph of a Math Expression

1. How to make computers understand math expressions?
2. Computational graphs are a nice way to express math expressions.
3. Consider the expression: $e = (a + b) \times (b + 1)$



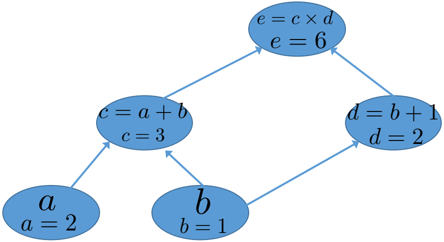
3. Introduce intermediate variables for results of each operation:

$$c = a + b; \quad d = b + 1; \quad e = c \times d$$

4. Construct a graph corresponding to these expressions:
   - operations and inputs are nodes
   - values used in operations are directed edges

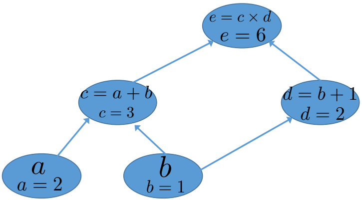
Evaluating the expression

1. Set the input variables to values and compute nodes up through the graph.
2. For $a = 2$ and $b = 1$:
3. Expression evaluates to 6.



Computational Graph Language

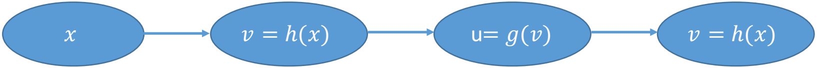
1. To describe backpropagation more precisely, we review the computational graph language.
2. Each node is either
   - a variable: scalar, vector, matrix, tensor, or other type
   - or an operation
     - ★ simple function of one or more variables
     - ★ functions more complex than operations are obtained by composing operations
   - if variable y is computed by applying operation to variable x then draw directed edge from x to y.



Derivatives of Composite function with Chain Rule

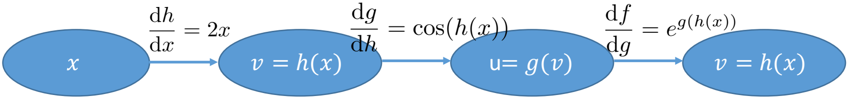
1. Computational graph provides a convenient way to calculate derivatives.
2. Let a composite function be $f(g(h(x))) \triangleq f \circ g \circ h(x)$, where $f(x) = e^x$, $g(x) = \sin(x)$, $h(x) = x^2$.
3. Using chain rule, we have

$$\begin{aligned} \frac{\mathrm{d}f}{\mathrm{d}x} &= \frac{\mathrm{d}f}{\mathrm{d}g} \cdot \frac{\mathrm{d}g}{\mathrm{d}h} \cdot \frac{\mathrm{d}h}{\mathrm{d}x} \\ &= e^{g(h(x))} \cdot \cos(h(x)) \cdot 2x \\ &= e^{\sin(x^2)} \cdot \cos(x^2) \cdot 2x \end{aligned}$$



Derivatives using Computational Graph

1. All we need to do is get the derivative of each node w.r.t. each of its inputs.



2. Get the derivative by multiplying the "connection" derivatives.

$$\frac{\mathrm{d}f}{\mathrm{d}x} = \frac{\mathrm{d}f}{\mathrm{d}g} \cdot \frac{\mathrm{d}g}{\mathrm{d}h} \cdot \frac{\mathrm{d}h}{\mathrm{d}x} = e^{\sin(x^2)} \cdot \cos(x^2) \cdot 2x$$

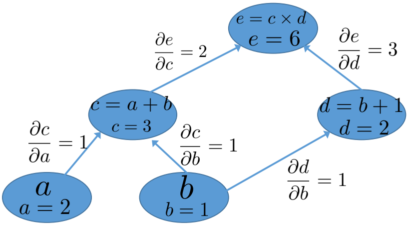
Derivatives for $e = (a + b) \times (b + 1)$ with Computational Graph

1. Indirection connection by chain rule:

$$\frac{\partial e}{\partial a} = \frac{\partial e}{\partial c} \cdot \frac{\partial c}{\partial a} = 2 \times 1 = 2$$

2. The general rule (with multiple paths) is: sum over all possible paths from one node to the other while multiplying derivatives on each path
   - *e.g.*, two paths from e to b

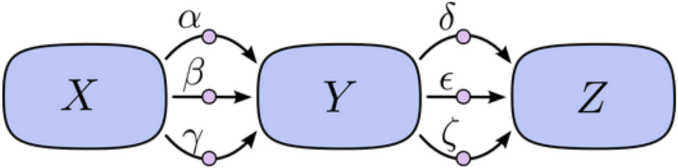
$$\frac{\partial e}{\partial b} = \frac{\partial e}{\partial c} \cdot \frac{\partial c}{\partial b} + \frac{\partial e}{\partial d} \cdot \frac{\partial d}{\partial b}$$
$$= 2 \times 1 + 3 \times 1 = 5$$



Naive Implementation of Derivative on Computational Graph

1. To compute the derivative of a node "*a*" w.r.t. another node "*b*", we need to find the number of paths from *b* to *a*.
   - This would lead to combinatorial explosion, *i.e.*, the number of path grows exponentially.
2. To get derivative $\frac{\partial Z}{\partial X}$, we need to sum over $3 \times 3 = 9$ paths:

$$\frac{\partial Z}{\partial X} = \alpha\delta + \alpha\epsilon + \alpha\varsigma + \beta\delta + \beta\epsilon + \beta\varsigma + \gamma\delta + \gamma\epsilon + \gamma\varsigma$$



3. Backpropagation is an efficient way of computing the derivatives on computational graphs, especially for deep neural networks.