

# Symbol-to-Symbol Derivatives

Sargur N. Srihari  
srihari@buffalo.edu

# Topics (Deep Feedforward Networks)

- Overview
  - 1.Example: Learning XOR
  - 2.Gradient-Based Learning
  - 3.Hidden Units
  - 4.Architecture Design
  - 5.Backpropagation and Other Differentiation Algorithms
  - 6.Historical Notes

# Topics in Backpropagation

- Forward and Backward Propagation
  1. Computational Graphs
  2. Chain Rule of Calculus
  3. Recursively applying the chain rule to obtain backprop
  4. Backpropagation computation in fully-connected MLP
  5. Symbol-to-symbol derivatives
  6. General backpropagation
  7. Ex: backpropagation for MLP training
  8. Complications
  9. Differentiation outside the deep learning community
  10. Higher-order derivatives

# Symbol-to-Symbol Derivatives

- Both algebraic expressions and computational graphs operate on symbols, or variables that do not have specific values
- They are called symbolic representations
- When we actually use or train a neural network, we must assign specific values for these symbols
- We replace a symbolic input to the network with a specific numeric value
  - E.g.,  $[2.5, 3.75, -1.8]^T$

# Two approaches to backpropagation

## 1. Symbol-to-number differentiation

- Take a computational graph and a set of numerical values for inputs to the graph
- Return a set of numerical values describing gradient at those input values
- Used by libraries: Torch and Caffe

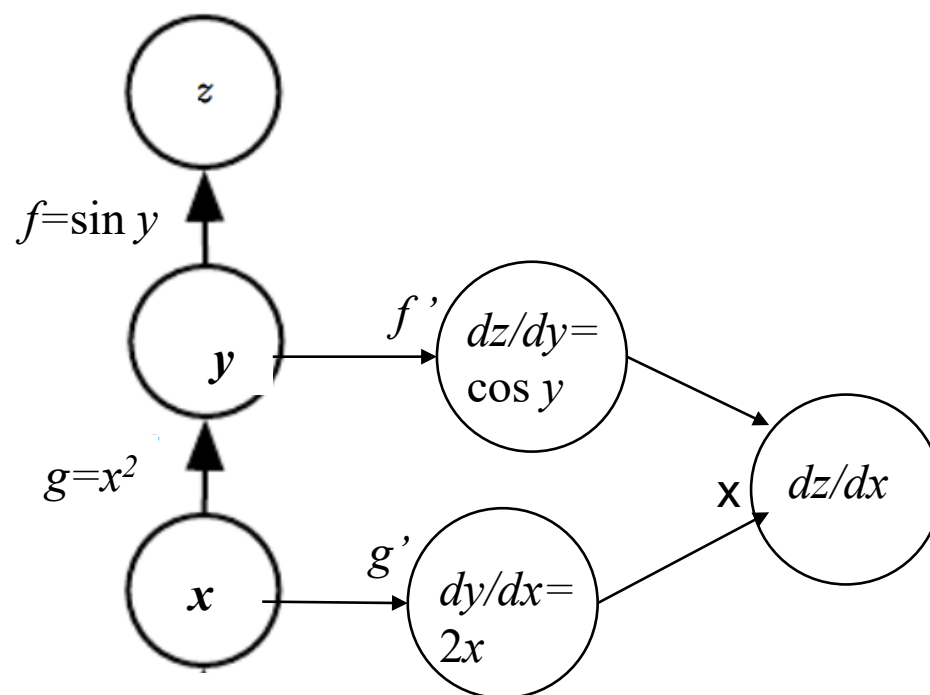
## 2. Symbol-to-symbol differentiation

- Take a computational graph
- Add additional nodes to the graph that provide a symbolic description of desired derivatives
- Used by libraries: Theano and Tensorflow

# Symbol-to-symbol Derivatives

- To compute derivative using this approach, backpropagation does not need to ever access any actual numerical values
  - Instead it adds nodes to a computational graph describing how to compute the derivatives for any specific numerical values
  - A generic graph evaluation engine can later compute derivatives for any specific numerical values

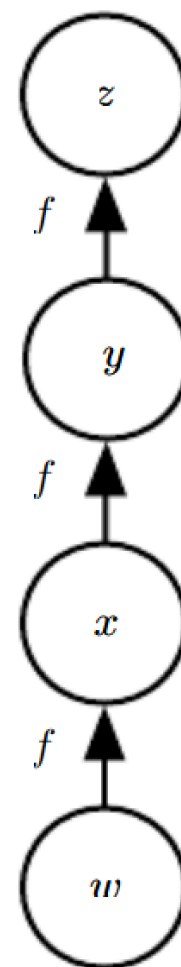
# Ex: Symbol-to-symbol derivatives



# Ex: Symbol-to-symbol Derivatives

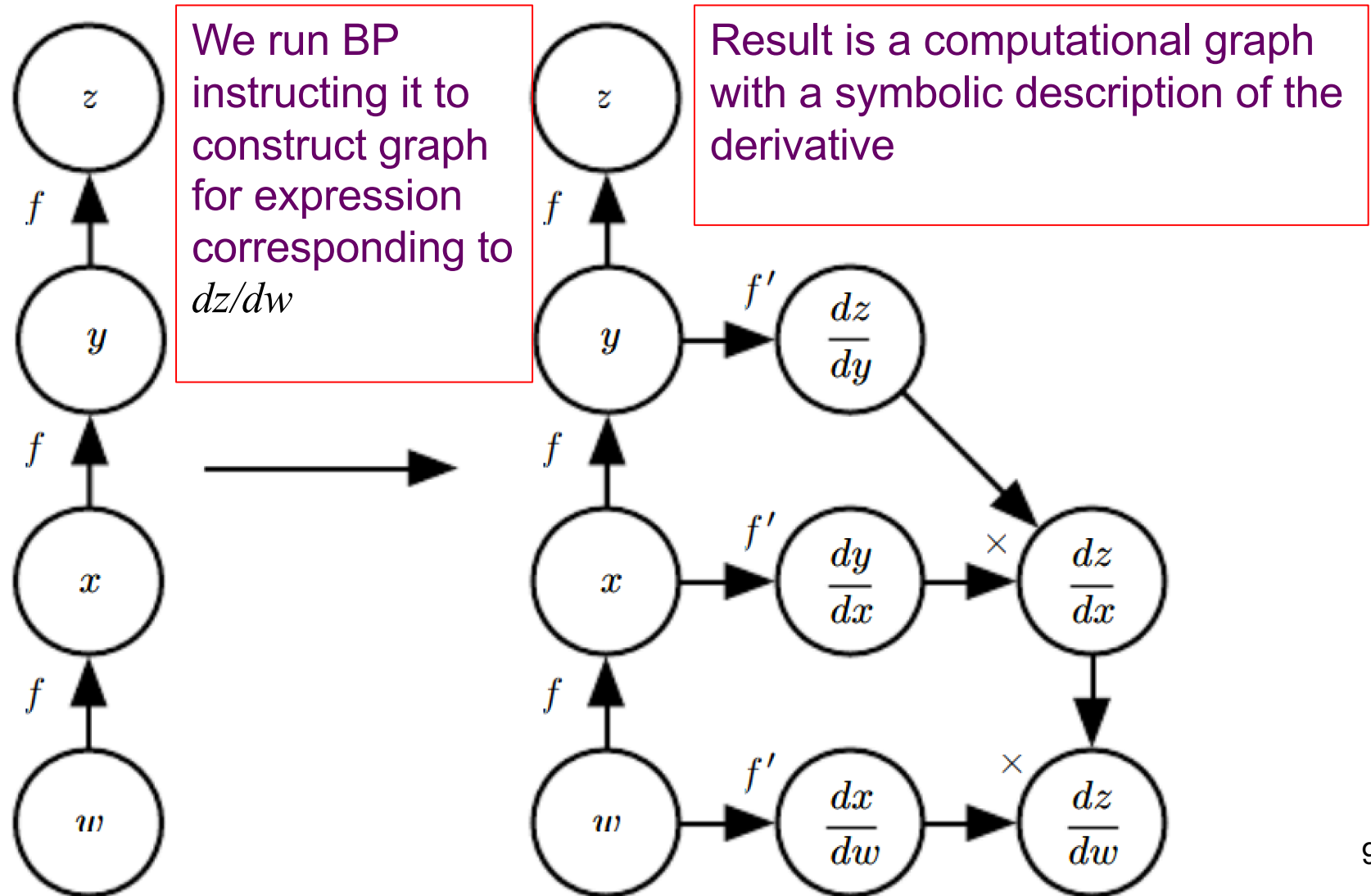
- Begin with graph representing

$$z = f(f(f(w)))$$





# Symbol-to-Symbol Derivative Computation



# Advantages of Approach

- Derivatives are described in the same language as the original expression
- Because the derivatives are just another computational graph, it is possible to run back-propagation again
  - Differentiating the derivatives
  - Yields higher-order derivatives