QuantEcon Lunchtalk 15:

Introduction to Automatic Differentiation

Shu Hu

Motivation: Deep Neural Network

$$y = (f_K \circ f_{K-1} \circ \cdots \circ f_1)(x) = f_K(f_{K-1}(\cdots(f_1(x))))$$

- x are the inputs,
- y are the observations,
- $f_i, i = 1, \cdots K$, called the function in the i-th layer, possesses its own parameters.

Motivation: Deep Neural Network

If we have inputs x and observations y and a network structure defined by

$$f_0 := x$$

and

$$f_i := \sigma_i(A_{i-1} f_{i-1} + b_{i-1})$$

for $i = 1, \cdots, K$, where

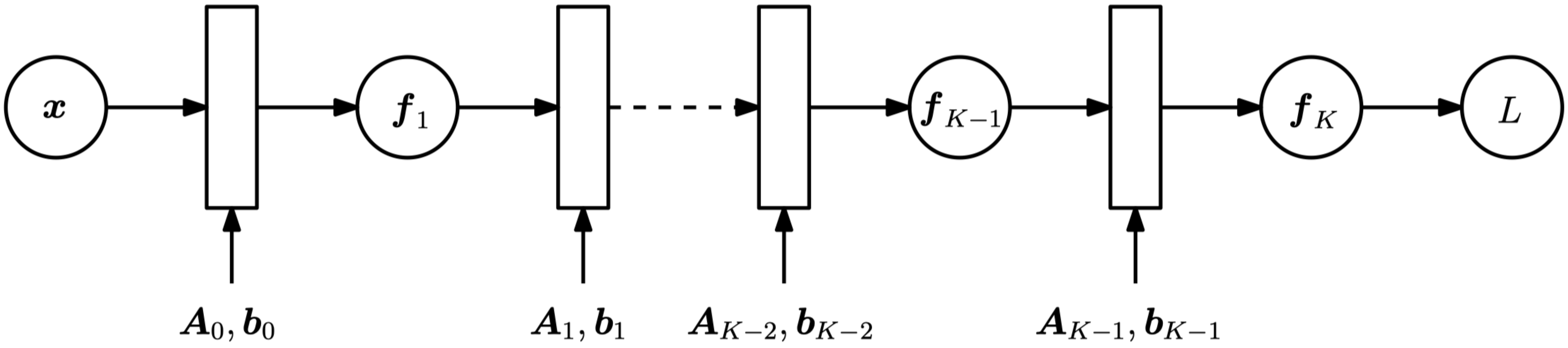
- x_{i-1} is the output of layer $i - 1$,
- σ is an activation function, and
- A_{i-1}, b_{j-1} are model parameters.

Motivation: Deep Neural Network

Then we may be interested in find $A_j, b_j, j = 0, \cdots, K-1$ s.t. the squared loss

$$L(\theta) = \|y - f_K(\theta, x)\|^2$$

is minimized, where $\theta = \{A_0, b_0, \cdots, A_{K-1}, b_{K-1}\}$.



Motivation: Deep Neural Network

To obtain the gradients w.r.t. the parameter set θ, we require partial derivatives of L w.r.t the parameters $\theta_j = \{A_j, b_j\}$ of each layer $j = 0, \cdots, K-1$, which is enabled by the chain rule

$$\frac{\partial L}{\partial \theta_{K-1}} = \frac{\partial L}{\partial f_K} \frac{\partial f_K}{\partial \theta_{K-1}}$$

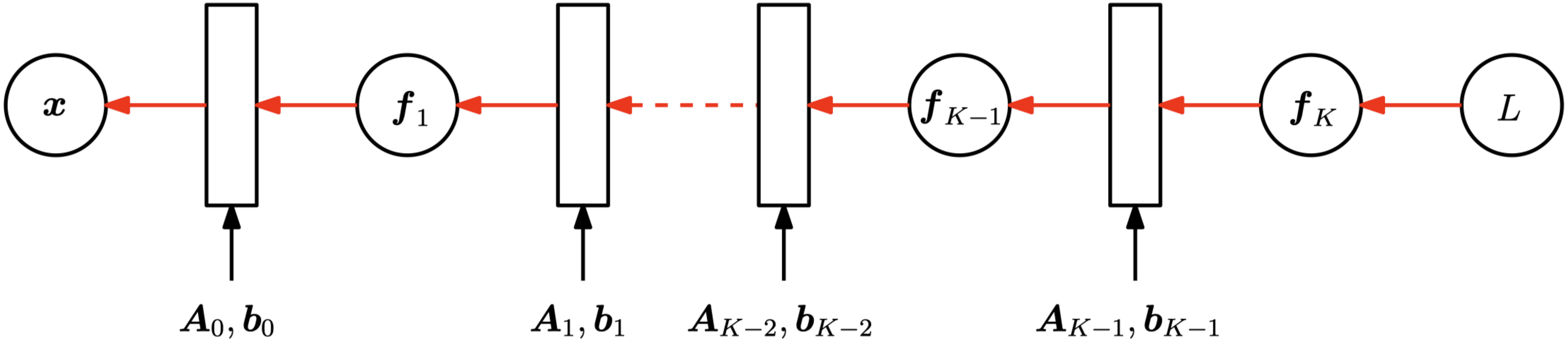
and

$$\frac{\partial L}{\partial \theta_{K-2}} = \frac{\partial L}{\partial f_K} \boxed{\frac{\partial f_K}{\partial f_{K-1}} \frac{\partial f_{K-1}}{\partial \theta_{K-2}}}$$

and

$$\frac{\partial L}{\partial \theta_i} = \frac{\partial L}{\partial f_K} \frac{\partial f_K}{\partial \theta_{K-1}} \cdots \boxed{\frac{\partial f_{i+2}}{\partial f_{i+1}} \frac{\partial f_{i+1}}{\partial \theta_i}}$$

Motivation: Deep Neural Network



Automatic Differentiation: Primer

Basic idea:

- a set of techniques to numerically evaluate the exact gradient of a function by working with intermediate variables and applying the chain rule.

Example 1, consider the data flow from input x to output y via intermediate variables a, b.

Other Differentiation Methods vs AutoDiff

- Manual Differentiation
- Numerical Differentiation
- Symbolic Differentiation
- Automatic Differentiation

Example

$$f(x) = \sqrt{x^2 + \exp(x^2)} + \cos(x^2 + \exp(x^2))$$

Intermediate Variables of the Example

$$a = x^2,$$

$$b = \exp(a)$$

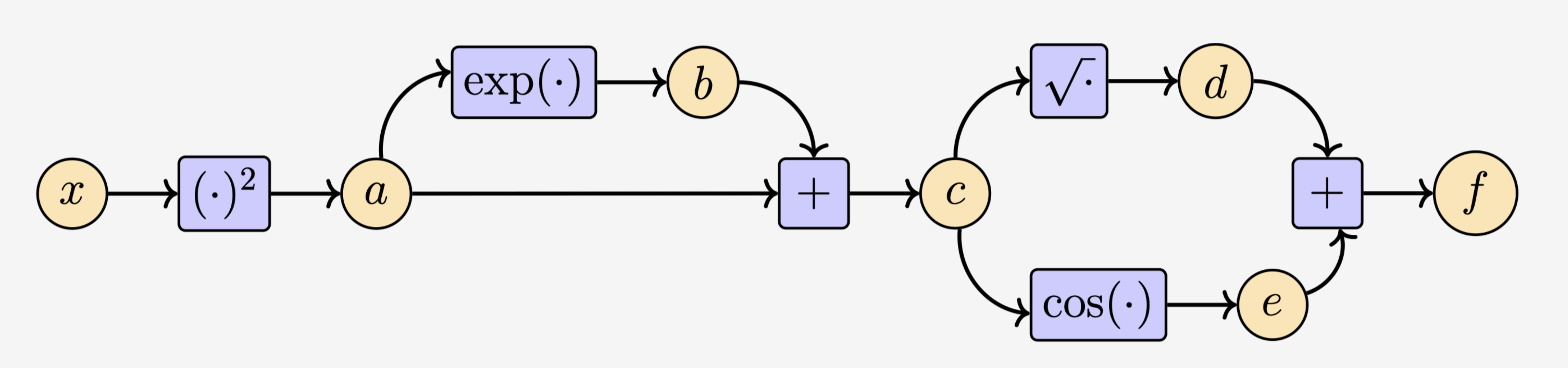
$$c = a + b$$

$$d = \sqrt{c}$$

$$e = \cos(c)$$

$$f = d + e$$

Computational Graph of the Example



Autodiff

Let

- $x_1, \cdots, x_d$ be the input variables to the function
- $x_{d+1}, \cdots, x_D$ be the intermediate variables
- x_D be the output variable.

Then the computation graph can be expressed as

$$x_i = g_i(x_{Pa(x_i)}),\ for\ i = d+1, \cdots, D$$

- $g_i(\cdot)$ are elementary functions
- $x_{Pa(x_i)}$ are the parent nodes of the variable x_i in the graph

Autodiff

Given a function defined in this way, we can use the chain rule to compute the derivative of the fucntion in a step-by-step fashion.

- since by definition $f = x_D$ and hence
$$\frac{\partial f}{\partial x_D} = 1$$
- For other variables x_i we apply the chain rule
$$\frac{\partial f}{\partial x_i} = \sum_{x_j : x_i \in Pa(x_j)} \frac{\partial f}{\partial x_j} \frac{\partial x_j}{\partial x_i} = \sum_{x_j : x_i \in Pa(x_j)} \frac{\partial f}{\partial x_j} \frac{\partial g_j}{\partial x_i}$$
    - $Pa(x_j)$ is the set of parent nodes of x_j in the computation graph.