Lecture 4: ML Frameworks

CS4787/5777 – Principles of Large-Scale Machine Learning Systems

Announcements

PSet 1 is due today at 11:59 PM. You may work in groups of up to four students.

You get 2 free late days for every assignment in this class → Can submit up until Friday, September 4 at 11:59 PM.

The first programming assignment will be released today.

No class on Monday.

Taking Derivatives

The derivative of F: U $\rightarrow$ V is the unique function DF that satisfies:

$$\lim_{\alpha \to 0} \frac{F(x + \alpha\Delta) - F(x)}{\alpha} = (DF(x))\Delta.$$

For any x and Δ in U.

Exercise 1: Derivatives

$$F(x) = Ax \qquad x \in \mathbb{R}^n, A \in \mathbb{R}^{n \times n}$$

Find the derivative of F(x):

$$\begin{aligned}(DF(x))\Delta &= \lim_{\alpha \to 0} \frac{F(x + \alpha\Delta) - F(x)}{\alpha} \\ &= \lim_{\alpha \to 0} \frac{Ax + \alpha A\Delta - Ax}{\alpha} \\ &= \lim_{\alpha \to 0} A\Delta \\ &= A\Delta.\end{aligned}$$

Since this holds for all Δ, we have $DF(x) = A$

Exercise 2: Derivatives

$$F(X) = X^2 \qquad X \in \mathbb{R}^{n \times n}$$

Find the derivative of F(x):

$$\begin{aligned}
(DF(X))\Delta &= \lim_{\alpha \to 0} \frac{F(X + \alpha\Delta) - F(X)}{\alpha} \\
&= \lim_{\alpha \to 0} \frac{X^2 + \alpha X\Delta + \alpha\Delta X + \alpha^2\Delta^2 - X^2}{\alpha} \\
&= \lim_{\alpha \to 0} \left(X\Delta + \Delta X + \alpha\Delta^2\right) \\
&= X\Delta + \Delta X.
\end{aligned}$$

Can't factor out Δ! We have: $DF(X) = (\Delta \mapsto X\Delta + \Delta X)$

Recap: Automatic Differentiation

- Symbolic differentiation:
  - Differentiate symbolically by applying the chain rule and other rules of calculus.
  - Can be very inefficient if implemented naively → Redundant computations.

- Numerical differentiation:

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x-h)}{2h} \approx \frac{f(x+\epsilon) - f(x-\epsilon)}{2\epsilon} \text{ for small } \epsilon$$

Recap: Automatic Differentiation

- Main idea: **Transform the program that computes the objective directly into a program that computes the gradients.**

- There are many ways of doing automatic differentiation.
  - Two broad classes: **forward mode** and **reverse mode.**

- For most ML applications, we use **backpropagation**, a particular flavor of reverse-mode automatic differentiation.
  - One specialized to compute gradients of neural network objectives.

Recap: Backpropagation (Reverse-mode AD)

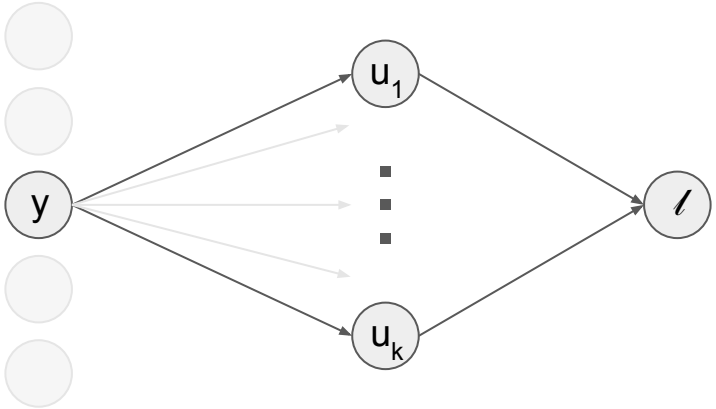
- Start with a computation graph that represents the function to be differentiated.

- **Forward pass:** Compute through the graph normally, as if we were computing the function value.
  - Save all intermediate values used in the computation → memory cost

- **Backward pass:** Now proceed backward through the graph, computing gradients of the function w.r.t intermediates.

Recap: Backpropagation

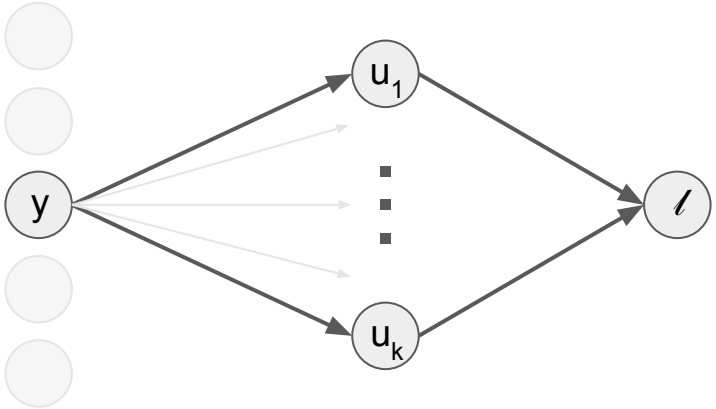
- Start with a computation graph that represents the function to be differentiated.



$$\ell = F(u_1, u_2, \ldots, u_k)$$
$$u_i = g_i(y)$$

Recap: Backpropagation

- Start with a computation graph that represents the function to be differentiated.



$$\ell = F(u_1, u_2, \ldots, u_k)$$
$$u_i = g_i(y)$$

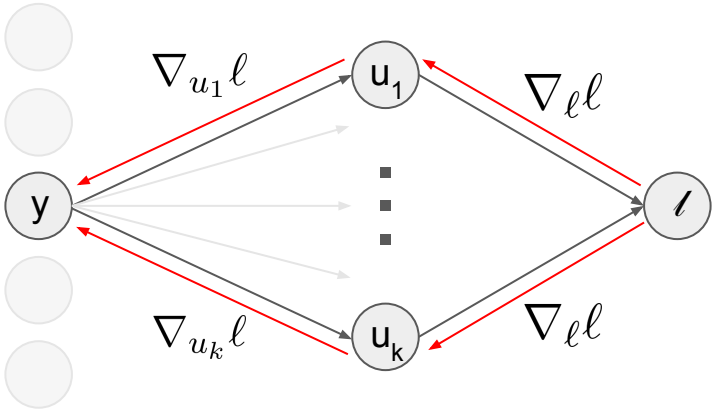
Recap: Backpropagation

- Start with a computation graph that represents the function to be differentiated.

Done locally!

$$\nabla_y \ell = \sum_{i=1}^{k} Dg_i(y)^T \nabla_{u_i} \ell$$

Passed through backprop



$$\ell = F(u_1, u_2, \ldots, u_k)$$
$$u_i = g_i(y)$$

Backward pass

Key thing to know: Automatic differentiation can compute gradients…

- For **any function** that has differentiable components.

- To **arbitrary precision**.

- Using a **small constant factor of additional compute** compared with the cost to compute the objective.

Machine Learning Frameworks

- Goal: **Make ML easier.**
  - From a software engineering perspective.
  - Make the computations more reliable, debuggable and robust.

- Goal: **Make ML scalable.**
  - To large datasets running on distributed hardware.

- Goal: **Make ML accessible.**
  - So that even people who aren't ML systems experts can get good performance.

What do ML frameworks support?

- Numerical linear algebra library.
- Hardware support (e.g. GPU)
- Backpropagation engine.
- Library for expressing deep neural networks.

Tensors

- CS way to think about it: a tensor is a **multidimensional array.**

- Math way to think about it: a **tensor is a multilinear map.**

$$T : \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} \times \cdots \times \mathbb{R}^{d_n} \to \mathbb{R}$$

$T(x_1, x_2, \ldots, x_n)$ is linear in each x_i, with other inputs fixed.

Examples of Tensors in Machine Learning

- The **CIFAR10 dataset** consists of 60,000 32x32 color images.
  - We can write the training set as a tensor.

$$T_{\text{CIFAR10}} \in \mathbb{R}^{32 \times 32 \times 3 \times 60000}$$

- **Gradients** for deep learning can also be tensors.
  - Example: Fully-connected layer with 100 input and 100 output neurons, and a mini-batch size b=32.

$$G \in \mathbb{R}^{100 \times 100 \times 32}$$

Pytorch: Most popular framework for ML research

- **Python** package that focuses on
  - Tensor computation (like numpy) with strong GPU acceleration.
  - Deep Neural Networks built with an autograd system.

- Supports **backpropagation**:
  - Fastest implementation of this method.

Demo