Tutorial 1

CS5242: Neural Networks and Deep Learning

Zheng Zangwei

National University of Singapore

Agenda

1. Differentiation
2. PyTorch
3. Gradient Descend
4. Q&A Session

Differentiation

Recap: Importance of Gradient

Gradient is a key element in optimization.

In Deep learning, gradient descent is used for optimization. We must know the gradient of loss over all parameters (scalar-by-matrix differentiation).

Recap: Differentiation types

Types of matrix derivative

Types	Scalar	Vector	Matrix
Scalar	$\frac{\partial y}{\partial x}$	$\frac{\partial \mathbf{y}}{\partial x}$	$\frac{\partial \mathbf{Y}}{\partial x}$
Vector	$\frac{\partial y}{\partial \mathbf{x}}$	$\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$	
Matrix	$\frac{\partial y}{\partial \mathbf{X}}$		

1. Gradient shape (shape check)
2. Gradient value (gradient rule)

Recap: Differentiation shapes

Our layout: **denominator layout**

This means the shape of our gradient is decided according to the shape of **denominator** and the transpose of **numerator**. ($\frac{\partial y}{\partial x}$, then shape according to x and y^T)

Given scalar x, y, vector $\boldsymbol{x}$ of shape [n x 1], vector $\boldsymbol{y}$ of shape [m x 1], we have:

$$\frac{\partial y}{\partial x} : [1x1]/[1x1] \rightarrow [1x1]$$
$$\frac{\partial \mathbf{y}}{\partial x} : [mx1]/[1x1] \rightarrow [1xm]$$
$$\frac{\partial y}{\partial \mathbf{x}} : [1x1]/[nx1] \rightarrow [nx1]$$

Recap: Differentiation shapes (Cont.)

Our layout: **denominator layout**

Given scalar x, y, vector $\boldsymbol{x}$ of shape [n x 1], vector $\boldsymbol{y}$ of shape [m x 1], matrix $\boldsymbol{X}$, $\boldsymbol{Y}$ of shape [m x n]

we have:

$$\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}} : [mx1]/[nx1] \rightarrow [nxm]$$
$$\frac{\partial y}{\partial \boldsymbol{X}} : [1x1]/[mxn] \rightarrow [mxn]$$

We only use enumerator layout for the following case:

$$\frac{\partial \boldsymbol{Y}}{\partial x} : [mxn]/[1x1] \rightarrow ?$$

Recap: Differentiation rules

Ways to differentiate function with vectors:

1. By definition.
2. Use rules (Refer to Wiki and Matrix Cookbook).
3. Pushforward.

Recap: Differentiation rules (Cont.)

By definition

$$\frac{\partial y}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y}{\partial x_1} \\ \frac{\partial y}{\partial x_2} \\ \vdots \\ \frac{\partial y}{\partial x_n} \end{bmatrix}.$$

$$\frac{\partial \mathbf{y}}{\partial x} = \begin{bmatrix} \frac{\partial y_1}{\partial x} & \frac{\partial y_2}{\partial x} & \cdots & \frac{\partial y_m}{\partial x} \end{bmatrix}.$$

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_2}{\partial x_1} & \cdots & \frac{\partial y_m}{\partial x_1} \\ \frac{\partial y_1}{\partial x_2} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_m}{\partial x_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_1}{\partial x_n} & \frac{\partial y_2}{\partial x_n} & \cdots & \frac{\partial y_m}{\partial x_n} \end{bmatrix}.$$

$$\frac{\partial y}{\partial \mathbf{X}} = \begin{bmatrix} \frac{\partial y}{\partial x_{11}} & \frac{\partial y}{\partial x_{12}} & \cdots & \frac{\partial y}{\partial x_{1q}} \\ \frac{\partial y}{\partial x_{21}} & \frac{\partial y}{\partial x_{22}} & \cdots & \frac{\partial y}{\partial x_{2q}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y}{\partial x_{p1}} & \frac{\partial y}{\partial x_{p2}} & \cdots & \frac{\partial y}{\partial x_{pq}} \end{bmatrix}.$$

Recap: Differentiation rules (Cont.)

Use rules

$\boldsymbol{y} =$	$\boldsymbol{a}$	$\boldsymbol{x}$	$\boldsymbol{Ax}$	$\boldsymbol{x}^T\boldsymbol{A}$
$\dfrac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}} =$	?	?	$\boldsymbol{A}^T$	?

$\boldsymbol{y} =$	$a\boldsymbol{u}$ $\boldsymbol{u} = \boldsymbol{u}(\boldsymbol{x})$	$v\boldsymbol{u}$ $v = v(\boldsymbol{x}), \boldsymbol{u} = \boldsymbol{u}(\boldsymbol{x})$	$\boldsymbol{v} + \boldsymbol{u}$ $\boldsymbol{v} = \boldsymbol{v}(\boldsymbol{x}), \boldsymbol{u} = \boldsymbol{u}(\boldsymbol{x})$	$\boldsymbol{Au}$ $\boldsymbol{u} = \boldsymbol{u}(\boldsymbol{x})$	$g(\boldsymbol{u})$ $\boldsymbol{u} = \boldsymbol{u}(\boldsymbol{x})$
$\dfrac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}} =$	?	$v\dfrac{\partial \boldsymbol{u}}{\partial \boldsymbol{x}} + \dfrac{\partial v}{\partial \boldsymbol{x}}\boldsymbol{u}^T$	?	?	?

$y =$	a	$\boldsymbol{u}^T\boldsymbol{v}$ $\boldsymbol{v} = \boldsymbol{v}(\boldsymbol{x}), \boldsymbol{u} = \boldsymbol{u}(\boldsymbol{x})$	$g(u)$ $u = u(\boldsymbol{x})$	$\boldsymbol{x}^T A\boldsymbol{x}$
$\dfrac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}} =$	?	?	?	?

Recap: Differentiation rules (Cont.)

Why rules hold? Use definition to prove.

Let's take $\frac{\partial \boldsymbol{Ax}}{\partial \boldsymbol{x}}$ as an example.

Shape check: $\boldsymbol{x}$ [n x 1], $\boldsymbol{A}$ [m x n]; $\boldsymbol{Ax}$ [m x 1]
So the gradient $\boldsymbol{g}$ is of shape [n x m].

Consider one position:

$$\begin{aligned} \boldsymbol{g}_{i,j} &= \frac{\partial (Ax)_j}{\partial x_i} \\ &= \frac{\sum_{k=1}^{n}(A_{j,k}x_k)}{\partial x_i} \\ &= A_{j,i} \end{aligned}$$

So the gradient $\frac{\partial \boldsymbol{Ax}}{\partial \boldsymbol{x}} = A^T$

Homework discussion

- Sigmoid function

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

- Softmax

$$f(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}, 1 \leq i \leq n$$

- Softplus activation

$$f(x) = \frac{1}{\beta} \cdot \ln(1 + e^{\beta x})$$

Homework discussion (Cont.)

-
$$f(x) = x^T(Ax + z)$$
- L2 loss
$$L(w) = \frac{1}{2}(w^T x - y)^2$$
- L2 loss (multiple examples)
$$L(w) = \frac{1}{2m}\|Xw - y\|^2$$
-
$$z = Wx + b, L = \|z - y\|^2$$

Identities

PyTorch

How does program do differentiation?

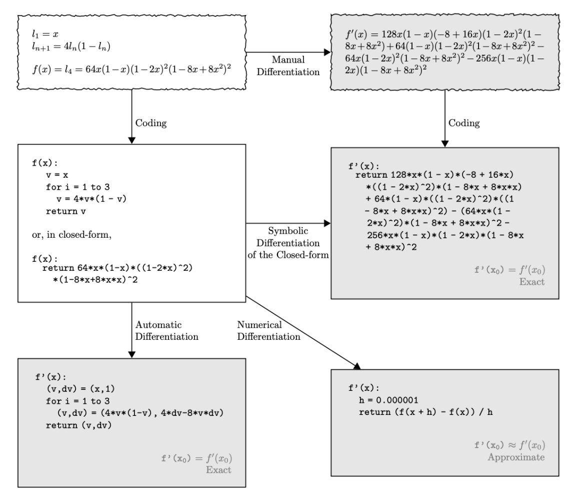

Figure 1: Source: Baydin, Atilim Gunes, et al. ”Automatic differentiation in machine learning: a survey.” Journal of Marchine Learning Research 18 (2018): 1-43.

How does program do differentiation?

- Manual Differentiation
- Symbolic Differentiation of the Closed-form (e.g., Mathematica, Maple)
- Numerical Differentiation
- Automatic Differentiation (PyTorch, Tensorflow)

How does program do differentiation? (Cont.)

Forward Primal Trace		
v_{-1}	$= x_1$	$= 2$
v_0	$= x_2$	$= 5$
v_1	$= \ln v_{-1}$	$= \ln 2$
v_2	$= v_{-1} \times v_0$	$= 2 \times 5$
v_3	$= \sin v_0$	$= \sin 5$
v_4	$= v_1 + v_2$	$= 0.693 + 10$
v_5	$= v_4 - v_3$	$= 10.693 + 0.959$
y	$= v_5$	$= 11.652$

Forward Tangent (Derivative) Trace		
$\dot{v}_{-1}$	$= \dot{x}_1$	$= 1$
$\dot{v}_0$	$= \dot{x}_2$	$= 0$
$\dot{v}_1$	$= \dot{v}_{-1}/v_{-1}$	$= 1/2$
$\dot{v}_2$	$= \dot{v}_{-1} \times v_0 + \dot{v}_0 \times v_{-1}$	$= 1 \times 5 + 0 \times 2$
$\dot{v}_3$	$= \dot{v}_0 \times \cos v_0$	$= 0 \times \cos 5$
$\dot{v}_4$	$= \dot{v}_1 + \dot{v}_2$	$= 0.5 + 5$
$\dot{v}_5$	$= \dot{v}_4 - \dot{v}_3$	$= 5.5 - 0$
$\boldsymbol{\dot{y}}$	$\boldsymbol{= \dot{v}_5}$	$\mathbf{= 5.5}$

Figure 2: Source: Baydin, Atilim Gunes, et al. ”Automatic differentiation in machine learning: a survey.” Journal of Marchine Learning Research 18 (2018): 1-43.

Get Started

1. PyTorch
2. Colab
3. Homework

Gradient Descend

Training by gradient descent

- Gradient descent algorithm for optimization
- Set w = 0.1 or a random number
- Repeat
  - For each data sample, compute $\tilde{y} = xw + b$
  - Compute the average loss, $\sum_{<x,y>\in S_{train}} L(x, y|w, b) \,/\, |S_{train}|$
  - Compute $\frac{\partial J}{\partial w}$
  - Update $\mathrm{w} = \mathrm{w} - \alpha \frac{\partial J}{\partial w}$

Update w, b repeatedly…

Learning rate

A good learning rate α make sure the process converges and converges fast.

For a simple case,

- Converge gradually
- Oscillate but converge
- Diverge

In the practice, more complicated.

- local minimal
- flatness
- ......

So learning rate is often the most important hyper-parameter to tune in DL.

Homework discussion

Consider a linear regression without intercept $y = xw, x \in \mathbb{R}, w \in \mathbb{R}$. L2 loss and gradient descent are used. Initial $w = 0$ and learning rate is α. Suppose we only have one example $x = 1, y = 100$ (which is not a setting in the reality and we use this toy example for ease of computation).

1. Show how gradient descent works for $\alpha = 0.5, 1.5, 2.5$.
2. Give the condition of α that gradient descent starts to oscillate around the optimal position. Give the condition of α that gradient descent can converge.
3. Try to prove your statement in 2). (Hint: consider $|100 - w_t|$)

Q&A Session