Practical course on computing derivatives in code

Craig Schroeder

SIGGRAPH 2019

Outline

1 Basics
- Motivation
- Don't do this
- Chain rule
- Tensors

2 Practical considerations

3 Differentiating matrix factorizations

4 Automatic differentiation

Outline

1. Basics
   - Motivation
   - Don't do this
   - Chain rule
   - Tensors
2. Practical considerations
3. Differentiating matrix factorizations
4. Automatic differentiation

Motivation - numerical optimization

$$\text{Minimize: } f(\mathbf{x})$$

Motivation - numerical optimization

Minimize: $f(\mathbf{x})$

Numerical optimization uses gradients

$\mathbf{x} \leftarrow \mathbf{x} - \alpha \nabla f$ Gradient descent

Motivation - numerical optimization

Minimize: $f(\mathbf{x})$

Numerical optimization uses gradients

$$\mathbf{x} \leftarrow \mathbf{x} - \alpha \nabla f$$

Gradient descent

More efficient methods need second derivatives

$$\mathbf{x} \leftarrow \mathbf{x} - \left(\frac{\partial^2 f}{\partial \mathbf{x} \partial \mathbf{x}} \right)^{-1} \nabla f$$

Newton's method

Motivation - physical forces

$$\text{potential energy: } \phi(\mathbf{x})$$

$$\text{force: } \mathbf{f} = -\frac{\partial \phi}{\partial \mathbf{x}}$$

Required for *conservative* forces.

Forces are often formulated via energy.

Motivation - constitutive models

$$\text{energy density: } \psi(\mathbf{F})$$

$$\text{stress: } \mathbf{P} = \frac{\partial \psi}{\partial \mathbf{F}}$$

Note that $\mathbf{F}$ and $\mathbf{P}$ are matrices.

Implicit methods require derivatives

Backward Euler, trapezoid rule

Solved with Newton's method

Second derivatives:

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}} = -\frac{\partial^2 \phi}{\partial \mathbf{x} \partial \mathbf{x}} \qquad \frac{\partial \mathbf{P}}{\partial \mathbf{F}} = \frac{\partial^2 \psi}{\partial \mathbf{F} \partial \mathbf{F}}$$

Functions can be very complex

From a graphics paper:

Functions can be very complex

From a graphics paper:

$$\alpha = \frac{(\mathbf{z}-\mathbf{x})\cdot(\mathbf{y}-\mathbf{x})}{\|(\mathbf{z}-\mathbf{x})\times(\mathbf{y}-\mathbf{x})\|} \qquad \beta = \frac{(\mathbf{x}-\mathbf{y})\cdot(\mathbf{z}-\mathbf{y})}{\|(\mathbf{x}-\mathbf{y})\times(\mathbf{z}-\mathbf{y})\|}$$

$$\gamma = \frac{(\mathbf{y}-\mathbf{z})\cdot(\mathbf{x}-\mathbf{z})}{\|(\mathbf{y}-\mathbf{z})\times(\mathbf{x}-\mathbf{z})\|} \qquad d = \frac{(\mathbf{z}-\mathbf{x})\times(\mathbf{y}-\mathbf{x})}{\|(\mathbf{z}-\mathbf{x})\times(\mathbf{y}-\mathbf{x})\|}\cdot(\mathbf{x}-\mathbf{c})$$

$$E_d = \frac{1}{d^2}\left(\alpha\|\mathbf{y}-\mathbf{z}\|^2 + \beta\|\mathbf{x}-\mathbf{z}\|^2 + \gamma\|\mathbf{x}-\mathbf{y}\|^2\right)$$

$$E_a = \frac{1}{kd^2}\|(\mathbf{x}-\mathbf{z})\times(\mathbf{y}-\mathbf{z})\|^2 \qquad E = a\cdot E_d + b\cdot E_a$$

Functions can be very complex

From a graphics paper:

$$\alpha = \frac{(\mathbf{z}-\mathbf{x})\cdot(\mathbf{y}-\mathbf{x})}{\|(\mathbf{z}-\mathbf{x})\times(\mathbf{y}-\mathbf{x})\|} \qquad \beta = \frac{(\mathbf{x}-\mathbf{y})\cdot(\mathbf{z}-\mathbf{y})}{\|(\mathbf{x}-\mathbf{y})\times(\mathbf{z}-\mathbf{y})\|}$$

$$\gamma = \frac{(\mathbf{y}-\mathbf{z})\cdot(\mathbf{x}-\mathbf{z})}{\|(\mathbf{y}-\mathbf{z})\times(\mathbf{x}-\mathbf{z})\|} \qquad d = \frac{(\mathbf{z}-\mathbf{x})\times(\mathbf{y}-\mathbf{x})}{\|(\mathbf{z}-\mathbf{x})\times(\mathbf{y}-\mathbf{x})\|}\cdot(\mathbf{x}-\mathbf{c})$$

$$E_d = \frac{1}{d^2}\left(\alpha\|\mathbf{y}-\mathbf{z}\|^2 + \beta\|\mathbf{x}-\mathbf{z}\|^2 + \gamma\|\mathbf{x}-\mathbf{y}\|^2\right)$$

$$E_a = \frac{1}{kd^2}\|(\mathbf{x}-\mathbf{z})\times(\mathbf{y}-\mathbf{z})\|^2 \qquad E = a\cdot E_d + b\cdot E_a$$

Need: $\frac{\partial E}{\partial \mathbf{x}}, \frac{\partial E}{\partial \mathbf{y}}, \frac{\partial E}{\partial \mathbf{z}}, \frac{\partial^2 E}{\partial \mathbf{x}\partial \mathbf{x}}, \frac{\partial^2 E}{\partial \mathbf{x}\partial \mathbf{y}}, \ldots, \frac{\partial^2 E}{\partial \mathbf{z}\partial \mathbf{z}}$

Functions can be very complex

From a graphics paper:

$$\alpha = \frac{(\mathbf{z}-\mathbf{x})\cdot(\mathbf{y}-\mathbf{x})}{\|(\mathbf{z}-\mathbf{x})\times(\mathbf{y}-\mathbf{x})\|} \qquad \beta = \frac{(\mathbf{x}-\mathbf{y})\cdot(\mathbf{z}-\mathbf{y})}{\|(\mathbf{x}-\mathbf{y})\times(\mathbf{z}-\mathbf{y})\|}$$

$$\gamma = \frac{(\mathbf{y}-\mathbf{z})\cdot(\mathbf{x}-\mathbf{z})}{\|(\mathbf{y}-\mathbf{z})\times(\mathbf{x}-\mathbf{z})\|} \qquad d = \frac{(\mathbf{z}-\mathbf{x})\times(\mathbf{y}-\mathbf{x})}{\|(\mathbf{z}-\mathbf{x})\times(\mathbf{y}-\mathbf{x})\|}\cdot(\mathbf{x}-\mathbf{c})$$

$$E_d = \frac{1}{d^2}(\alpha\|\mathbf{y}-\mathbf{z}\|^2 + \beta\|\mathbf{x}-\mathbf{z}\|^2 + \gamma\|\mathbf{x}-\mathbf{y}\|^2)$$

$$E_a = \frac{1}{kd^2}\|(\mathbf{x}-\mathbf{z})\times(\mathbf{y}-\mathbf{z})\|^2 \qquad E = a\cdot E_d + b\cdot E_a$$

Need: $\frac{\partial E}{\partial \mathbf{x}}, \frac{\partial E}{\partial \mathbf{y}}, \frac{\partial E}{\partial \mathbf{z}}, \frac{\partial^2 E}{\partial \mathbf{x}\partial \mathbf{x}}, \frac{\partial^2 E}{\partial \mathbf{x}\partial \mathbf{y}}, \ldots, \frac{\partial^2 E}{\partial \mathbf{z}\partial \mathbf{z}}$ (Used Maple)

It may be hard to know it is right

Sometimes the only symptom is *slow convergence*.

With the right ideas, we can do this

This course will show you how.

Outline

1. Basics
   - Motivation
   - Don't do this
   - Chain rule
   - Tensors
2. Practical considerations
3. Differentiating matrix factorizations
4. Automatic differentiation

Don't avoid the problem

It is tempting to give up on the task.

The task normally falls to a student or intern.

What not to do - finite differences

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}$$

What not to do - finite differences

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}$$

- Only approximate

What not to do - finite differences

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}$$

- Only approximate
  - May break numerical optimization routines

What not to do - finite differences

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}$$

- Only approximate
  - May break numerical optimization routines
  - Catastrophic cancellation

What not to do - finite differences

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}$$

- Only approximate
  - May break numerical optimization routines
  - Catastrophic cancellation
- Expensive for gradients/Hessians

What not to do - Maple/Mathematica

Pros:

- Compute derivatives automatically
- Can generate code automatically.

How bad can it really be?

Modest example: $f(\mathbf{u}, \mathbf{v}) = \left\| \mathbf{u}(\mathbf{u} \cdot \mathbf{v})^2 - \mathbf{v} \|\mathbf{u}\|^3 \right\|^2$

How bad can it really be?

Modest example: $f(\mathbf{u},\mathbf{v}) = \left\| \mathbf{u}(\mathbf{u}\cdot\mathbf{v})^2 - \mathbf{v}\|\mathbf{u}\|^3 \right\|^2$

What I did in Maple:

- Compute Hessian $\mathbf{H} = \frac{\partial^2 f}{\partial \mathbf{u} \partial \mathbf{u}}$
- Simplify
- Generate C code for just H_{11}.
- Simplify the code

This is the result

```
t1 = v2 * v2; t4 = v1 * v1; t6 = t1 * t1; t8 = t1 / 10; t9 = v3 * v3; t10 = t9 / 10; t12 = u2 * u2;
t13 = t12 * t12; t19 = u1 * v1; t24 = t12 * u2; t31 = t9 / 5; t33 = u3 * u3; t35 = 3 * t1;
t41 = u1 * u1; t42 = t4 * t4; t46 = 3 * t9; t52 = u3 * v3; t61 = 0.8 * u1 * u3 * v1 * v3;
t73 = t33 * t33; t77 = t33 * u3; t88 = t41 * u1; t94 = t41 * t41;
t100 = sqrt(t41 + t12 + t33); t102 = t4 / 5;
H11 = -60 / t100 * (t100 * (t13 * (t4 * (-t1 / 5 - 0.1) - t6 / 30 - t8 - t10)
    - 0.4 * t24 * v2 * (u3 * (t4 + t1 / 3) * v3 + (t4 + t1) * t19)
    + t12 * (t33 * (t4 * (-t1 - t9 - 1) / 5 + t1 * (-t9 - 1) / 5 - t31)
    - 0.4 * u3 * (t4 + t35) * v3 * t19 - (t42 + t4 * (6 * t1 + 3) + t35 + t46) * t41 / 5)
    - 4. / 3 * u2 * (t33 * (0.3 * t4 + t10) + t61 + t4 * t41) * v2 * (t19 + t52)
    + t73 * (t4 * (-t31 - 0.1) - t8 - (t9 + 3) * t9 / 30)
    - 0.4 * t77 * (t4 + t9) * v3 * t19 - t33 * (t42 + t4 * (6 * t9 + 3) + t35
    + t46) * t41 / 5 - 4. / 3 * v3 * t4 * v1 * u3 * t88 - (t42 + t4 + t1 + t9) * t94 / 2)
    + (u2 * v2 + t19 + t52) * (t13 * (t102 + t8) + 0.8 * t24 * v2 * (t19 + t52 / 4)
    + t12 * (t33 * (0.4 * t4 + t8 + t10) + t61 + 1.1 * (t4 + 2. / 11 * t1) * t41)
    + u2 * (t88 * v1 + 0.4 * t41 * t52 + 0.8 * u1 * v1 * t33 + v3 * t77 / 5) * v2
    + t73 * (t102 + t10) + 0.8 * v3 * u1 * v1 * t77 + 1.1 * t33 * t41 * (t4 + 2. / 11 * t9)
    + t88 * v3 * u3 * v1 + t4 * t94));
```

This is the result

```
t1 = v2 * v2; t4 = v1 * v1; t6 = t1 * t1; t8 = t1 / 10; t9 = v3 * v3; t10 = t9 / 10; t12 = u2 * u2;
t13 = t12 * t12; t19 = u1 * v1; t24 = t12 * u2; t31 = t9 / 5; t33 = u3 * u3; t35 = 3 * t1;
t41 = u1 * u1; t42 = t4 * t4; t46 = 3 * t9; t52 = u3 * v3; t61 = 0.8 * u1 * u3 * v1 * v3;
t73 = t33 * t33; t77 = t33 * u3; t88 = t41 * u1; t94 = t41 * t41;
t100 = sqrt(t41 + t12 + t33); t102 = t4 / 5;
H11 = -60 / t100 * (t100 * (t13 * (t4 * (-t1 / 5 - 0.1) - t6 / 30 - t8 - t10)
    - 0.4 * t24 * v2 * (u3 * (t4 + t1 / 3) * v3 + (t4 + t1) * t19)
    + t12 * (t33 * (t4 * (-t1 - t9 - 1) / 5 + t1 * (-t9 - 1) / 5 - t31)
    - 0.4 * u3 * (t4 + t35) * v3 * t19 - (t42 + t4 * (6 * t1 + 3) + t35 + t46) * t41 / 5)
    - 4. / 3 * u2 * (t33 * (0.3 * t4 + t10) + t61 + t4 * t41) * v2 * (t19 + t52)
    + t73 * (t4 * (-t31 - 0.1) - t8 - (t9 + 3) * t9 / 30)
    - 0.4 * t77 * (t4 + t9) * v3 * t19 - t33 * (t42 + t4 * (6 * t9 + 3) + t35
    + t46) * t41 / 5 - 4. / 3 * v3 * t4 * v1 * u3 * t88 - (t42 + t4 + t1 + t9) * t94 / 2)
    + (u2 * v2 + t19 + t52) * (t13 * (t102 + t8) + 0.8 * t24 * v2 * (t19 + t52 / 4)
    + t12 * (t33 * (0.4 * t4 + t8 + t10) + t61 + 1.1 * (t4 + 2. / 11 * t1) * t41)
    + u2 * (t88 * v1 + 0.4 * t41 * t52 + 0.8 * u1 * v1 * t33 + v3 * t77 / 5) * v2
    + t73 * (t102 + t10) + 0.8 * v3 * u1 * v1 * t77 + 1.1 * t33 * t41 * (t4 + 2. / 11 * t9)
    + t88 * v3 * u3 * v1 + t4 * t94));
```

This is *only* H_{11}.

This is the result

```
t1 = v2 * v2; t4 = v1 * v1; t6 = t1 * t1; t8 = t1 / 10; t9 = v3 * v3; t10 = t9 / 10; t12 = u2 * u2;
t13 = t12 * t12; t19 = u1 * v1; t24 = t12 * u2; t31 = t9 / 5; t33 = u3 * u3; t35 = 3 * t1;
t41 = u1 * u1; t42 = t4 * t4; t46 = 3 * t9; t52 = u3 * v3; t61 = 0.8 * u1 * u3 * v1 * v3;
t73 = t33 * t33; t77 = t33 * u3; t88 = t41 * u1; t94 = t41 * t41;
t100 = sqrt(t41 + t12 + t33); t102 = t4 / 5;
H11 = -60 / t100 * (t100 * (t13 * (t4 * (-t1 / 5 - 0.1) - t6 / 30 - t8 - t10)
    - 0.4 * t24 * v2 * (u3 * (t4 + t1 / 3) * v3 + (t4 + t1) * t19)
    + t12 * (t33 * (t4 * (-t1 - t9 - 1) / 5 + t1 * (-t9 - 1) / 5 - t31)
    - 0.4 * u3 * (t4 + t35) * v3 * t19 - (t42 + t4 * (6 * t1 + 3) + t35 + t46) * t41 / 5)
    - 4. / 3 * u2 * (t33 * (0.3 * t4 + t10) + t61 + t4 * t41) * v2 * (t19 + t52)
    + t73 * (t4 * (-t31 - 0.1) - t8 - (t9 + 3) * t9 / 30)
    - 0.4 * t77 * (t4 + t9) * v3 * t19 - t33 * (t42 + t4 * (6 * t9 + 3) + t35
    + t46) * t41 / 5 - 4. / 3 * v3 * t4 * v1 * u3 * t88 - (t42 + t4 + t1 + t9) * t94 / 2)
    + (u2 * v2 + t19 + t52) * (t13 * (t102 + t8) + 0.8 * t24 * v2 * (t19 + t52 / 4)
    + t12 * (t33 * (0.4 * t4 + t8 + t10) + t61 + 1.1 * (t4 + 2. / 11 * t1) * t41)
    + u2 * (t88 * v1 + 0.4 * t41 * t52 + 0.8 * u1 * v1 * t33 + v3 * t77 / 5) * v2
    + t73 * (t102 + t10) + 0.8 * v3 * u1 * v1 * t77 + 1.1 * t33 * t41 * (t4 + 2. / 11 * t9)
    + t88 * v3 * u3 * v1 + t4 * t94));
```

This is *only* H_{11}. Also need H_{12}, H_{13}, H_{22}, H_{23}, and H_{33}.

Outline

1. Basics
   - Motivation
   - Don't do this
   - Chain rule
   - Tensors
2. Practical considerations
3. Differentiating matrix factorizations
4. Automatic differentiation

The chain rule in computation

Original: $a = f(g(x))$

Derivative: $a' = f'(g(x))g'(x)$

The chain rule in computation

Original: $a = f(g(x))$

Derivative: $a' = f'(g(x))\color{red}{g'(x)}$

Example: $a = \sqrt{x^2 + 1}$

Pieces: $g = x^2 + 1,\ f = \sqrt{g}$

The chain rule in computation

Original: $a = f(g(x))$

Derivative: $a' = f'(g(x))g'(x)$

Example: $a = \sqrt{x^2 + 1}$

Pieces: $g = x^2 + 1,\ f = \sqrt{g}$

Derivative: $g' = 2x,\ f' = \frac{g'}{2f}$ (Note: reuse f)

Recall: $\frac{d}{dx}\sqrt{x} = \frac{1}{2\sqrt{x}}$

The chain rule is the key

Example: $f(x) = (x^3 + \sqrt{1 + x^2})^2$

The chain rule is the key

Example: $f(x) = (x^3 + \sqrt{1 + x^2})^2$

Step 1: break it into small pieces.

The chain rule is the key

Example: $f(x) = (x^3 + \sqrt{1+x^2})^2$

Step 1: break it into small pieces.

$$a = 1 + x^2 \qquad b = x^3 \qquad c = \sqrt{a} \qquad d = b + c \qquad f = d^2$$

The chain rule is the key

Example: $f(x) = (x^3 + \sqrt{1+x^2})^2$

Step 1: break it into small pieces.

$$a = 1 + x^2 \qquad b = x^3 \qquad c = \sqrt{a} \qquad d = b + c \qquad f = d^2$$

Step 2: compute the derivative of each step

The chain rule is the key

Example: $f(x) = (x^3 + \sqrt{1 + x^2})^2$

Step 1: break it into small pieces.

$$a = 1 + x^2 \qquad b = x^3 \qquad c = \sqrt{a} \qquad d = b + c \qquad f = d^2$$

Step 2: compute the derivative of each step

$$a' = 2x \qquad b' = 3x^2 \qquad c' = \frac{\color{red}{a'}}{2c} \qquad d' = \color{red}{b'} + \color{red}{c'} \qquad f' = 2d\color{red}{d'}$$

Note the use of the chain rule.

This is good code

$$
\begin{aligned}
a &= 1 + x^2 & b &= x^3 & c &= \sqrt{a} & d &= b + c & f &= d^2 \\
a' &= 2x & b' &= 3x^2 & c' &= \frac{a'}{2c} & d' &= b' + c' & f' &= 2dd'
\end{aligned}
$$

Second derivatives are easy, too

$$\begin{array}{lllll} a = 1 + x^2 & b = x^3 & c = \sqrt{a} & d = b + c & f = d^2 \\ a' = 2x & b' = 3x^2 & c' = \dfrac{a'}{2c} & d' = b' + c' & f' = 2dd' \\ a'' = 2 & b'' = 6x & c'' = \dfrac{a''c - a'c'}{2a} & d'' = b'' + c'' & f'' = 2(d')^2 + 2dd'' \end{array}$$

This works for more complex stuff

Earlier example: $f(\mathbf{u}, \mathbf{v}) = \left\| \mathbf{u}(\mathbf{u} \cdot \mathbf{v})^2 - \mathbf{v}\|\mathbf{u}\|^3 \right\|^2$

This works for more complex stuff

Earlier example: $f(\mathbf{u},\mathbf{v}) = \left\|\mathbf{u}(\mathbf{u}\cdot\mathbf{v})^2 - \mathbf{v}\|\mathbf{u}\|^3\right\|^2$

Step 1:

$$a = \mathbf{u}\cdot\mathbf{v} \qquad b = \|\mathbf{u}\| \qquad c = a^2 \qquad d = b^3$$
$$\mathbf{w} = c\mathbf{u} - d\mathbf{v} \qquad f = \|\mathbf{w}\|^2$$

This works for more complex stuff

Earlier example: $f(\mathbf{u},\mathbf{v}) = \left\| \mathbf{u}(\mathbf{u}\cdot\mathbf{v})^2 - \mathbf{v}\|\mathbf{u}\|^3 \right\|^2$

Step 1:

$$a = \mathbf{u}\cdot\mathbf{v} \qquad b = \|\mathbf{u}\| \qquad c = a^2 \qquad d = b^3$$
$$\mathbf{w} = c\mathbf{u} - d\mathbf{v} \qquad f = \|\mathbf{w}\|^2$$

Step 2:

$$a_u = \mathbf{v} \qquad b_u = \frac{\mathbf{u}}{b} \qquad c_u = 2aa_u \qquad d_u = 3b^2 b_u$$
$$\mathbf{w}_u = c\mathbf{I} + \mathbf{u}c_u^T - \mathbf{v}d_u^T \qquad f_u = 2\mathbf{w}\cdot\mathbf{w}_u$$

But wait, we needed second derivatives

The first few are not too bad.

$$
\begin{aligned}
a &= \mathbf{u} \cdot \mathbf{v} & b &= \|\mathbf{u}\| & c &= a^2 & d &= b^3 \\
a_u &= \mathbf{v} & b_u &= \frac{\mathbf{u}}{b} & c_u &= 2aa_u & d_u &= 3b^2 b_u \\
a_{uu} &= \mathbf{0} & b_{uu} &= \frac{1}{b}(\mathbf{I} - b_u b_u^T) & c_{uu} &= 2a_u a_u^T & d_{uu} &= 6bb_u b_u^T + 3b^2 b_{uu}
\end{aligned}
$$

Complication: tensors

$$\begin{aligned} \mathbf{w} &= c\mathbf{u} - d\mathbf{v} & f &= \|\mathbf{w}\|^2 \\ \mathbf{w}_u &= c\mathbf{I} + \mathbf{u}c_u^T - \mathbf{v}d_u^T & f_u &= 2\mathbf{w} \cdot \mathbf{w}_u \\ \mathbf{w}_{uu} &= \quad ?!? \end{aligned}$$

$\mathbf{w}$ is a vector.

$\mathbf{w}_u$ is a matrix.

$\mathbf{w}_{uu}$ is a rank-3 tensor.

Complication: tensors

$$\begin{aligned} \mathbf{w} &= c\mathbf{u} - d\mathbf{v} \\ \mathbf{w}_u &= c\mathbf{I} + \mathbf{u}c_u^T - \mathbf{v}d_u^T \\ \mathbf{w}_{uu} &= \quad ?!? \end{aligned} \qquad \begin{aligned} f &= \|\mathbf{w}\|^2 \\ f_u &= 2\mathbf{w} \cdot \mathbf{w}_u \\ f_{uu} &= 2\underbrace{\mathbf{w} \cdot \mathbf{w}_{uu}}_{\mathbf{z}} + 2\mathbf{w}_u^T\mathbf{w}_u \end{aligned}$$

$\mathbf{w}$ is a vector.

$\mathbf{w}_u$ is a matrix.

$\mathbf{w}_{uu}$ is a rank-3 tensor.

Note the usage of $\mathbf{w}_{uu}$.

Complication: tensors

$$\begin{aligned}
\mathbf{w} &= c\mathbf{u} - d\mathbf{v} & f &= \|\mathbf{w}\|^2 \\
\mathbf{w}_u &= c\mathbf{I} + \mathbf{u}c_u^T - \mathbf{v}d_u^T & f_u &= 2\mathbf{w}\cdot\mathbf{w}_u \\
\mathbf{w}_{uu} &= \quad ?!? & f_{uu} &= 2\underbrace{\mathbf{w}\cdot\mathbf{w}_{uu}}_{\mathbf{z}} + 2\mathbf{w}_u^T\mathbf{w}_u
\end{aligned}$$

$\mathbf{w}$ is a vector.

$\mathbf{w}_u$ is a matrix.

$\mathbf{w}_{uu}$ is a rank-3 tensor.

Note the usage of $\mathbf{w}_{uu}$. Only need matrix $\mathbf{z}$.

Clever idea: avoid computing $\mathbf{w}_{uu}$

Compute $\mathbf{z} = \mathbf{w} \cdot \mathbf{w}_{uu}$ instead of $\mathbf{w}_{uu}$. $\mathbf{z}$ is a matrix.

Clever idea: avoid computing $\mathbf{w}_{uu}$

Compute $\mathbf{z} = \mathbf{w} \cdot \mathbf{w}_{uu}$ instead of $\mathbf{w}_{uu}$. $\mathbf{z}$ is a matrix.

$$\mathbf{z} = (\mathbf{u} \cdot \mathbf{w}) c_{uu} + c_u \mathbf{w}^T + \mathbf{w} c_u^T + (\mathbf{v} \cdot \mathbf{w}) d_{uu}$$

$$f_{uu} = 2\mathbf{z} + 2\mathbf{w}_u^T \mathbf{w}_u$$

Clever idea: avoid computing $\mathbf{w}_{uu}$

Compute $\mathbf{z} = \mathbf{w} \cdot \mathbf{w}_{uu}$ instead of $\mathbf{w}_{uu}$. $\mathbf{z}$ is a matrix.

$$\mathbf{z} = (\mathbf{u} \cdot \mathbf{w})c_{uu} + c_u \mathbf{w}^T + \mathbf{w} c_u^T + (\mathbf{v} \cdot \mathbf{w}) d_{uu}$$

$$f_{uu} = 2\mathbf{z} + 2\mathbf{w}_u^T \mathbf{w}_u$$

This is *all* of $\mathbf{H} = f_{uu}$, not just H_{11}.

Outline

1. Basics
   - Motivation
   - Don't do this
   - Chain rule
   - Tensors
2. Practical considerations
3. Differentiating matrix factorizations
4. Automatic differentiation

Tensor index notation solves two problems

- Deal with tensors
  - Gradient of matrix: $\mathbf{w}_{uu}$
  - Rank-4 tensor: $\dfrac{\partial \mathbf{P}}{\partial \mathbf{F}}$

Tensor index notation solves two problems

- Deal with tensors
  - Gradient of matrix: $\mathbf{w}_{uu}$
  - Rank-4 tensor: $\dfrac{\partial \mathbf{P}}{\partial \mathbf{F}}$
- Forgotten derivative rules
  - $\nabla(\mathbf{u} \cdot \mathbf{v})$
  - $\nabla(f\mathbf{u})$
  - $\nabla \cdot (\mathbf{u} \times \mathbf{v})$

Refer to objects by their components

Scalar: $a \to a$

Vector: $\mathbf{u} \to u_i$

Matrix: $\mathbf{A} \to A_{ij}$

Rank-3 tensor: B_{ijk}

Rank-4 tensor: C_{ijkl}

Summation convention

Dot product: $a = \mathbf{u} \cdot \mathbf{v} = \sum_{i} u_{i} v_{i}.$

Indices that occur twice in a term are *implicitly* summed.

Index notation: $a = u_{i} v_{i}.$

Summation convention

Dot product: $a = \mathbf{u} \cdot \mathbf{v} = \sum_{i} u_i v_i$.

Indices that occur twice in a term are *implicitly* summed.

Index notation: $a = u_i v_i$.

Index names do not matter. $a = u_i v_i = u_k v_k = u_r v_r$.

vector notation calculation index notation

vector notation	calculation	index notation
$\mathbf{A} = \mathbf{u}\mathbf{v}^T$	$A_{ik} = u_i v_k$	$A_{ik} = u_i v_k$

vector notation	calculation	index notation
$\mathbf{A} = \mathbf{u}\mathbf{v}^T$	$A_{ik} = u_i v_k$	$A_{ik} = u_i v_k$
$a = \mathbf{u} \cdot \mathbf{v}$	$a = \sum_i u_i v_i$	$a = u_i v_i$

vector notation	calculation	index notation
$\mathbf{A} = \mathbf{u}\mathbf{v}^T$	$A_{ik} = u_i v_k$	$A_{ik} = u_i v_k$
$a = \mathbf{u} \cdot \mathbf{v}$	$a = \sum_i u_i v_i$	$a = u_i v_i$
$\mathbf{v} = \mathbf{A}\mathbf{u}$	$v_i = \sum_k A_{ik} u_k$	$v_i = A_{ik} u_k$

vector notation	calculation	index notation
$\mathbf{A} = \mathbf{u}\mathbf{v}^T$	$A_{ik} = u_i v_k$	$A_{ik} = u_i v_k$
$a = \mathbf{u} \cdot \mathbf{v}$	$a = \sum_i u_i v_i$	$a = u_i v_i$
$\mathbf{v} = \mathbf{A}\mathbf{u}$	$v_i = \sum_k A_{ik} u_k$	$v_i = A_{ik} u_k$
$\mathbf{A} = \mathbf{B}\mathbf{C}$	$A_{ir} = \sum_k B_{ik} C_{kr}$	$A_{ir} = B_{ik} C_{kr}$

vector notation	calculation	index notation
$\mathbf{A} = \mathbf{u}\mathbf{v}^T$	$A_{ik} = u_i v_k$	$A_{ik} = u_i v_k$
$a = \mathbf{u} \cdot \mathbf{v}$	$a = \sum_i u_i v_i$	$a = u_i v_i$
$\mathbf{v} = \mathbf{A}\mathbf{u}$	$v_i = \sum_k A_{ik} u_k$	$v_i = A_{ik} u_k$
$\mathbf{A} = \mathbf{B}\mathbf{C}$	$A_{ir} = \sum_k B_{ik} C_{kr}$	$A_{ir} = B_{ik} C_{kr}$
$\mathbf{A} = \mathbf{B}^T\mathbf{C}$	$A_{ir} = \sum_k B_{ki} C_{kr}$	$A_{ir} = B_{ki} C_{kr}$

vector notation	calculation	index notation
$\mathbf{A} = \mathbf{u}\mathbf{v}^T$	$A_{ik} = u_i v_k$	$A_{ik} = u_i v_k$
$a = \mathbf{u} \cdot \mathbf{v}$	$a = \sum_i u_i v_i$	$a = u_i v_i$
$\mathbf{v} = \mathbf{A}\mathbf{u}$	$v_i = \sum_k A_{ik} u_k$	$v_i = A_{ik} u_k$
$\mathbf{A} = \mathbf{B}\mathbf{C}$	$A_{ir} = \sum_k B_{ik} C_{kr}$	$A_{ir} = B_{ik} C_{kr}$
$\mathbf{A} = \mathbf{B}^T\mathbf{C}$	$A_{ir} = \sum_k B_{ki} C_{kr}$	$A_{ir} = B_{ki} C_{kr}$
$a = \mathrm{tr}(\mathbf{A})$	$a = \sum_i A_{ii}$	$a = A_{ii}$

Subtleties

Careful about indices: $u_i u_j v_i \neq u_i u_i v_j$

Subtleties

Careful about indices: $u_i u_j v_i \neq u_i u_i v_j$

Multiplication commutes: $A_{ik} B_{kr} = B_{kr} A_{ik}$

Subtleties

Careful about indices: $u_i u_j v_i \neq u_i u_i v_j$

Multiplication commutes: $A_{ik} B_{kr} = B_{kr} A_{ik}$

$(\mathbf{u} \cdot \mathbf{v})^2$ is $(u_i v_i)(u_r v_r)$, not $(u_i v_i)(u_i v_i)$.

Special tensors - identity matrix

Kronecker delta

$$\delta_{ik} = \begin{cases} 1 & i = k \\ 0 & i \neq k \end{cases}$$

$$\delta_{ik} = \delta_{ki}$$

$$\delta_{ik} u_k = u_i$$

Special tensors - cross product

Permutation tensor

$$e_{ikr} = \begin{cases} 1 & 123, 231, 312 \\ -1 & 132, 213, 321 \\ 0 & \text{otherwise} \end{cases}$$

$\mathbf{u} = \mathbf{v} \times \mathbf{w}$ becomes $u_i = e_{ikr} v_k w_r$.

$$e_{ikr} = e_{rik} = e_{kri}$$
$$e_{ikr} = -e_{irk}$$

Derivatives in index notation

Differentiation denoted with a comma

$$f_{,r} = \frac{\partial f}{\partial x_r} \qquad f_{,rs} = \frac{\partial^2 f}{\partial x_r \partial x_s}$$

$$u_{i,r} = \frac{\partial u_i}{\partial x_r} \qquad u_{i,rs} = \frac{\partial^2 u_i}{\partial x_r \partial x_s}$$

Derivatives in index notation

Differentiation denoted with a comma

$$f_{,r} = \frac{\partial f}{\partial x_r} \qquad f_{,rs} = \frac{\partial^2 f}{\partial x_r \partial x_s}$$

$$u_{i,r} = \frac{\partial u_i}{\partial x_r} \qquad u_{i,rs} = \frac{\partial^2 u_i}{\partial x_r \partial x_s}$$

Special case: $x_{i,r} = \dfrac{\partial x_i}{\partial x_r} = \delta_{ir}$

Derivatives in index notation

Differentiation denoted with a comma

$$f_{,r} = \frac{\partial f}{\partial x_r} \qquad f_{,rs} = \frac{\partial^2 f}{\partial x_r \partial x_s}$$

$$u_{i,r} = \frac{\partial u_i}{\partial x_r} \qquad u_{i,rs} = \frac{\partial^2 u_i}{\partial x_r \partial x_s}$$

Special case: $x_{i,r} = \dfrac{\partial x_i}{\partial x_r} = \delta_{ir}$

Constants: $\delta_{ik,r} = 0$, $e_{ikr,s} = 0$

gradient ∇f $\dfrac{\partial f}{\partial x_r}$ $f_{,r}$

gradient	∇f	$\dfrac{\partial f}{\partial x_r}$	$f_{,r}$
divergence	$\nabla \cdot \mathbf{u}$	$\displaystyle\sum_r \frac{\partial u_r}{\partial x_r}$	$u_{r,r}$

gradient	∇f	$\dfrac{\partial f}{\partial x_r}$	$f_{,r}$
divergence	$\nabla \cdot \mathbf{u}$	$\sum_r \dfrac{\partial u_r}{\partial x_r}$	$u_{r,r}$
curl	$\nabla \times \mathbf{u}$		$e_{ikr} u_{k,r}$

gradient	∇f	$\frac{\partial f}{\partial x_r}$	$f_{,r}$
divergence	$\nabla \cdot \mathbf{u}$	$\sum_r \frac{\partial u_r}{\partial x_r}$	$u_{r,r}$
curl	$\nabla \times \mathbf{u}$		$e_{ikr} u_{k,r}$
Laplacian	$\nabla^2 f$	$\sum_r \frac{\partial^2 f}{\partial x_r \partial x_r}$	$f_{,rr}$

gradient	∇f	$\dfrac{\partial f}{\partial x_r}$	$f_{,r}$
divergence	$\nabla \cdot \mathbf{u}$	$\sum_r \dfrac{\partial u_r}{\partial x_r}$	$u_{r,r}$
curl	$\nabla \times \mathbf{u}$		$e_{ikr} u_{k,r}$
Laplacian	$\nabla^2 f$	$\sum_r \dfrac{\partial^2 f}{\partial x_r \partial x_r}$	$f_{,rr}$
vector Laplacian	$\nabla^2 \mathbf{u}$	$\sum_r \dfrac{\partial^2 u_i}{\partial x_r \partial x_r}$	$u_{i,rr}$

Scalar derivative rules apply

Components are *scalars*, so scalar rules apply.

Scalar derivative rules apply

Components are *scalars*, so scalar rules apply.

Vector: $\nabla(\mathbf{u} \cdot \mathbf{w}) = ?$

Index: $(u_i w_i)_{,r} = u_{i,r} w_i + u_i w_{i,r}$

Scalar derivative rules apply

Components are *scalars*, so scalar rules apply.

Vector: $\nabla(\mathbf{u} \cdot \mathbf{w}) = ?$

Index: $(u_i w_i)_{,r} = u_{i,r} w_i + u_i w_{i,r} \implies \nabla \mathbf{u}^T \mathbf{w} + \nabla \mathbf{w}^T \mathbf{u}$

Scalar derivative rules apply

Components are *scalars*, so scalar rules apply.

Vector: $\nabla(\mathbf{u} \cdot \mathbf{w}) = ?$

Index: $(u_i w_i)_{,r} = u_{i,r} w_i + u_i w_{i,r} \implies \nabla \mathbf{u}^T \mathbf{w} + \nabla \mathbf{w}^T \mathbf{u}$

Vector: $\nabla \cdot (\mathbf{u} \times \mathbf{w}) = ?$

Index: $(e_{ikr} u_k w_r)_{,s} = e_{ikr} u_{k,s} w_r + e_{ikr} u_k w_{r,s}$

Unfinished business

Recall: $u_{i,s} = \delta_{is}$ $\qquad v_{i,s} = 0$

$$\mathbf{w}_u = c\mathbf{I} + \mathbf{u}c_u^T - \mathbf{v}d_u^T$$

Unfinished business

Recall: $u_{i,s} = \delta_{is}$ $\quad v_{i,s} = 0$

$$\mathbf{w}_u = c\mathbf{I} + \mathbf{u}c_u^T - \mathbf{v}d_u^T$$
$$w_{i,r} = c\delta_{ir} + u_i c_{,r} - v_i d_{,r}$$

Unfinished business

Recall: $u_{i,s} = \delta_{is} \qquad v_{i,s} = 0$

$$\mathbf{w}_u = c\mathbf{I} + \mathbf{u}c_u^T - \mathbf{v}d_u^T$$

$$w_{i,r} = c\delta_{ir} + u_i c_{,r} - v_i d_{,r}$$

$$w_{i,rs} = c_{,s}\delta_{ir} + u_i c_{,rs} + u_{i,s} c_{,r} - v_i d_{,rs}$$

Unfinished business

Recall: $u_{i,s} = \delta_{is}$ $\qquad v_{i,s} = 0$

$$\mathbf{w}_u = c\mathbf{I} + \mathbf{u}c_u^T - \mathbf{v}d_u^T$$

$$w_{i,r} = c\delta_{ir} + u_i c_{,r} - v_i d_{,r}$$

$$w_{i,rs} = c_{,s}\delta_{ir} + u_i c_{,rs} + u_{i,s} c_{,r} - v_i d_{,rs}$$

$$w_i w_{i,rs} = w_i c_{,s}\delta_{ir} + w_i u_i c_{,rs} + w_i \delta_{is} c_{,r} - w_i v_i d_{,rs}$$

Unfinished business

Recall: $u_{i,s} = \delta_{is}$ $\qquad v_{i,s} = 0$

$$\mathbf{w}_u = c\mathbf{I} + \mathbf{u}c_u^T - \mathbf{v}d_u^T$$

$$w_{i,r} = c\delta_{ir} + u_i c_{,r} - v_i d_{,r}$$

$$w_{i,rs} = c_{,s}\delta_{ir} + u_i c_{,rs} + u_{i,s}c_{,r} - v_i d_{,rs}$$

$$w_i w_{i,rs} = w_i c_{,s}\delta_{ir} + w_i u_i c_{,rs} + w_i \delta_{is} c_{,r} - w_i v_i d_{,rs}$$

$$w_i w_{i,rs} = w_r c_{,s} + (w_i u_i) c_{,rs} + c_{,r} w_s - (w_i v_i) d_{,rs}$$

Unfinished business

Recall: $u_{i,s} = \delta_{is} \qquad v_{i,s} = 0$

$$\mathbf{w}_u = c\mathbf{I} + \mathbf{u}c_u^T - \mathbf{v}d_u^T$$

$$w_{i,r} = c\delta_{ir} + u_i c_{,r} - v_i d_{,r}$$

$$w_{i,rs} = c_{,s}\delta_{ir} + u_i c_{,rs} + u_{i,s}c_{,r} - v_i d_{,rs}$$

$$w_i w_{i,rs} = w_i c_{,s}\delta_{ir} + w_i u_i c_{,rs} + w_i \delta_{is} c_{,r} - w_i v_i d_{,rs}$$

$$w_i w_{i,rs} = w_r c_{,s} + (w_i u_i)c_{,rs} + c_{,r}w_s - (w_i v_i)d_{,rs}$$

$$\mathbf{z} = \mathbf{w} \cdot \mathbf{w}_{uu} = \mathbf{w}c_u^T + (\mathbf{w} \cdot \mathbf{u})c_{uu} + c_u\mathbf{w}^T - (\mathbf{w} \cdot \mathbf{v})d_{uu}$$

Derivatives in many variables at once

E.g., $f(\mathbf{u}, \mathbf{w})$. Need $\frac{\partial f}{\partial \mathbf{u}}$ and $\frac{\partial f}{\partial \mathbf{w}}$

Derivatives in many variables at once

E.g., $f(\mathbf{u}, \mathbf{w})$. Need $\frac{\partial f}{\partial \mathbf{u}}$ and $\frac{\partial f}{\partial \mathbf{w}}$

Work out $f_{,r}$. Do not assume $f_{,r} = \frac{\partial f}{\partial u_r}$ or $f_{,r} = \frac{\partial f}{\partial w_r}$.

Derivatives in many variables at once

E.g., $f(\mathbf{u}, \mathbf{w})$. Need $\frac{\partial f}{\partial \mathbf{u}}$ and $\frac{\partial f}{\partial \mathbf{w}}$

Work out $f_{,r}$. Do not assume $f_{,r} = \frac{\partial f}{\partial u_r}$ or $f_{,r} = \frac{\partial f}{\partial w_r}$.

Make two copies of the code:

For $\frac{\partial f}{\partial \mathbf{u}}$, let $u_{i,r} = \delta_{ir}$ and $w_{i,r} = 0$

For $\frac{\partial f}{\partial \mathbf{w}}$, let $u_{i,r} = 0$ and $w_{i,r} = \delta_{ir}$

Derivatives in many variables at once

E.g., $f(\mathbf{u}, \mathbf{w})$. Need $\frac{\partial f}{\partial \mathbf{u}}$ and $\frac{\partial f}{\partial \mathbf{w}}$

Work out $f_{,r}$. Do not assume $f_{,r} = \frac{\partial f}{\partial u_r}$ or $f_{,r} = \frac{\partial f}{\partial w_r}$.

Make two copies of the code:

For $\frac{\partial f}{\partial \mathbf{u}}$, let $u_{i,r} = \delta_{ir}$ and $w_{i,r} = 0$

For $\frac{\partial f}{\partial \mathbf{w}}$, let $u_{i,r} = 0$ and $w_{i,r} = \delta_{ir}$

Simplify *after* it works.

Different indices for different variables

r for $\mathbf{x}$

α for $\mathbf{y}$

$$f_{,r} = \frac{\partial f}{\partial x_r}$$

$$f_{,\alpha} = \frac{\partial f}{\partial y_\alpha}$$

Parenthesis for derivative by matrix

$$\psi_{,(rs)} = \frac{\partial \psi}{\partial F_{rs}}$$

Parenthesis for derivative by matrix

$$\psi_{,(rs)} = \frac{\partial \psi}{\partial F_{rs}}$$

$$F_{ik,(rs)} = \delta_{ir}\delta_{ks}$$

Outline

1 Basics

2 Practical considerations
- Modes of differentiation
- Testing
- Implicit differentiation

3 Differentiating matrix factorizations

4 Automatic differentiation

Outline

1. Basics
2. Practical considerations
   - Modes of differentiation
   - Testing
   - Implicit differentiation
3. Differentiating matrix factorizations
4. Automatic differentiation

Forward mode differentiation

Input: x

Output: y

Calculations: $a = a(x)$, $b = b(a)$, $c = c(b)$, $y = y(c)$

Forward mode differentiation

Input: x

Output: y

Calculations: $a = a(x)$, $b = b(a)$, $c = c(b)$, $y = y(c)$

$$\frac{\partial b}{\partial x} = \frac{\partial b}{\partial a}\frac{\partial a}{\partial x} \qquad \frac{\partial c}{\partial x} = \frac{\partial c}{\partial b}\frac{\partial b}{\partial x} \qquad \frac{\partial y}{\partial x} = \frac{\partial y}{\partial c}\frac{\partial c}{\partial x}$$

Forward mode differentiation

Input: x

Output: y

Calculations: $a = a(x)$, $b = b(a)$, $c = c(b)$, $y = y(c)$

$$\frac{\partial b}{\partial x} = \frac{\partial b}{\partial a}\frac{\partial a}{\partial x} \qquad \frac{\partial c}{\partial x} = \frac{\partial c}{\partial b}\frac{\partial b}{\partial x} \qquad \frac{\partial y}{\partial x} = \frac{\partial y}{\partial c}\frac{\partial c}{\partial x}$$

Note: $\frac{\partial ?}{\partial x}$

Forward mode differentiation

Input: x

Output: y

Calculations: $a = a(x)$, $b = b(a)$, $c = c(b)$, $y = y(c)$

$$\frac{\partial b}{\partial x} = \frac{\partial b}{\partial a}\frac{\partial a}{\partial x} \quad \frac{\partial c}{\partial x} = \frac{\partial c}{\partial b}\frac{\partial b}{\partial x} \quad \frac{\partial y}{\partial x} = \frac{\partial y}{\partial c}\frac{\partial c}{\partial x}$$

Note: $\frac{\partial ?}{\partial x}$

Actual computation:

$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial c}\left(\frac{\partial c}{\partial b}\left(\frac{\partial b}{\partial a}\frac{\partial a}{\partial x}\right)\right)$$

Reverse mode differentiation

Input: x

Output: y

Calculations: $a = a(x)$, $b = b(a)$, $c = c(b)$, $y = y(c)$

$$\frac{\partial y}{\partial b} = \frac{\partial y}{\partial c}\frac{\partial c}{\partial b} \qquad \frac{\partial y}{\partial a} = \frac{\partial y}{\partial b}\frac{\partial b}{\partial a} \qquad \frac{\partial y}{\partial x} = \frac{\partial y}{\partial a}\frac{\partial a}{\partial x} \qquad \text{Note: } \frac{\partial y}{\partial ?}$$

Actual computation:

$$\frac{\partial y}{\partial x} = \left(\left(\frac{\partial y}{\partial c}\frac{\partial c}{\partial b}\right)\frac{\partial b}{\partial a}\right)\frac{\partial a}{\partial x}$$

Cost

Sizes: $\mathbf{x} \to \mathbb{R}^6$, $\mathbf{a} \to \mathbb{R}^3$, $\mathbf{b} \to \mathbb{R}^3$, $\mathbf{y} \to \mathbb{R}^1$

Cost

Sizes: $\mathbf{x} \to \mathbb{R}^6$, $\mathbf{a} \to \mathbb{R}^3$, $\mathbf{b} \to \mathbb{R}^3$, $\mathbf{y} \to \mathbb{R}^1$

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \underbrace{\frac{\partial \mathbf{y}}{\partial \mathbf{b}}}_{1\times 3} \underbrace{\left(\frac{\partial \mathbf{b}}{\partial \mathbf{a}} \frac{\partial \mathbf{a}}{\partial \mathbf{x}}\right)}_{3\times 6;\ (54*)}$$

Forward: $54 + 18$

Cost

Sizes: $\mathbf{x} \to \mathbb{R}^6$, $\mathbf{a} \to \mathbb{R}^3$, $\mathbf{b} \to \mathbb{R}^3$, $\mathbf{y} \to \mathbb{R}^1$

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \underbrace{\frac{\partial \mathbf{y}}{\partial \mathbf{b}}}_{1\times 3} \underbrace{\left(\frac{\partial \mathbf{b}}{\partial \mathbf{a}} \frac{\partial \mathbf{a}}{\partial \mathbf{x}}\right)}_{3\times 6;\ (54*)}$$

Forward: $54 + 18$

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \underbrace{\left(\frac{\partial \mathbf{y}}{\partial \mathbf{b}} \frac{\partial \mathbf{b}}{\partial \mathbf{a}}\right)}_{1\times 3;\ (9*)} \underbrace{\frac{\partial \mathbf{a}}{\partial \mathbf{x}}}_{3\times 6}$$

Reverse: $9 + 18$

Efficiency of forward vs reverse modes

- Calculating $\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$
- Forward is cheaper if $x \ll y$
  - Easier to write

Efficiency of forward vs reverse modes

- Calculating $\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$
- Forward is cheaper if $x \ll y$
  - Easier to write
- Reverse is cheaper if $x \gg y$
  - Optimization: Minimize $E(\mathbf{x})$
  - Forces: $\phi(\mathbf{x})$
  - Stresses: $\psi(\mathbf{F})$
  - Backpropagation (machine learning)

Mixed mode differentiation

Input: x

Output: y

Calculations: $a = a(x)$, $b = b(a)$, $c = c(b)$, $y = y(c)$

$$\frac{\partial c}{\partial a} = \frac{\partial c}{\partial b}\frac{\partial b}{\partial a} \qquad \frac{\partial y}{\partial a} = \frac{\partial y}{\partial c}\frac{\partial c}{\partial a} \qquad \frac{\partial y}{\partial x} = \frac{\partial y}{\partial a}\frac{\partial a}{\partial x}$$

Actual computation:

$$\frac{\partial y}{\partial x} = \left(\frac{\partial y}{\partial c}\left(\frac{\partial c}{\partial b}\frac{\partial b}{\partial a}\right)\right)\frac{\partial a}{\partial x}$$

Optimal ordering

- Optimal Jacobian accumulation
- NP-complete
- Dynamic programming heuristic

Outline

1. Basics
2. Practical considerations
   - Modes of differentiation
   - Testing
   - Implicit differentiation
3. Differentiating matrix factorizations
4. Automatic differentiation

If you cannot test it, don't write it

How do we know it is right?

If you cannot test it, don't write it

How do we know it is right?

Wrong answer?

If you cannot test it, don't write it

How do we know it is right?

Wrong answer?

Disappointing results?

If you cannot test it, don't write it

How do we know it is right?

Wrong answer?

Disappointing results?

Slow convergence?

If you cannot test it, don't write it

How do we know it is right?

Wrong answer?

Disappointing results?

Slow convergence?

Poor stability?

If you cannot test it, don't write it

How do we know it is right?

Wrong answer?

Disappointing results?

Slow convergence?

Poor stability?

How do you debug that?

Testing scalars with definition

Test derivatives against definition!

Testing scalars with definition

Test derivatives against definition!

$$\frac{z(x + \Delta x) - z(x)}{\Delta x} - z'(x) = O(\Delta x)$$

Testing scalars with definition

Test derivatives against definition!

$$\frac{z(x + \Delta x) - z(x)}{\Delta x} - z'(x) = O(\Delta x)$$

How small should Δx be?

Testing scalars with definition

Test derivatives against definition!

$$\frac{z(x + \Delta x) - z(x)}{\Delta x} - z'(x) = O(\Delta x)$$

How small should Δx be?

How small is $O(\Delta x)$?

Testing scalars with definition

Test derivatives against definition!

$$\frac{z(x + \Delta x) - z(x)}{\Delta x} - z'(x) = O(\Delta x)$$

How small should Δx be?

How small is $O(\Delta x)$?

Refinement test?

Use a second-order test instead

$$\frac{z(x+\Delta x)-z(x)}{\Delta x}-\frac{z'(x+\Delta x)+z'(x)}{2}=O(\Delta x^2)$$

Use a second-order test instead

$$\frac{z(x+\Delta x) - z(x)}{\Delta x} - \frac{z'(x+\Delta x) + z'(x)}{2} = O(\Delta x^2)$$

Choose $\Delta x \approx \epsilon^{1/3} \sim 10^{-5}$ $\quad\quad$ $\epsilon \approx 2 \times 10^{-16}$

Use a second-order test instead

$$\frac{z(x+\Delta x) - z(x)}{\Delta x} - \frac{z'(x+\Delta x) + z'(x)}{2} = O(\Delta x^2)$$

Choose $\Delta x \approx \epsilon^{1/3} \sim 10^{-5}$ $\qquad \epsilon \approx 2 \times 10^{-16}$

Fail error: $O(1)$

Use a second-order test instead

$$\frac{z(x+\Delta x)-z(x)}{\Delta x}-\frac{z'(x+\Delta x)+z'(x)}{2}=O(\Delta x^2)$$

Choose $\Delta x \approx \epsilon^{1/3} \sim 10^{-5}$ $\qquad \epsilon \approx 2 \times 10^{-16}$

Fail error: $O(1)$

Pass error: $O(\epsilon^{2/3}) \sim 10^{-10}$

Use a second-order test instead

$$\frac{z(x+\Delta x)-z(x)}{\Delta x}-\frac{z'(x+\Delta x)+z'(x)}{2}=O(\Delta x^2)$$

Choose $\Delta x \approx \epsilon^{1/3} \sim 10^{-5}$ $\qquad \epsilon \approx 2 \times 10^{-16}$

Fail error: $O(1)$

Pass error: $O(\epsilon^{2/3}) \sim 10^{-10}$

Vector Δx?

Non-scalar second-order derivative test

"Multiply through" by $\Delta\mathbf{x}$

$$z(\mathbf{x}+\Delta\mathbf{x}) - z(\mathbf{x}) - \frac{\nabla z(\mathbf{x}+\Delta\mathbf{x}) + \nabla z(\mathbf{x})}{2} \cdot \Delta\mathbf{x} = O(\delta^3)$$

$\|\Delta\mathbf{x}\|_\infty \le \delta$

Non-scalar second-order derivative test

"Multiply through" by $\Delta\mathbf{x}$

$$z(\mathbf{x}+\Delta\mathbf{x}) - z(\mathbf{x}) - \frac{\nabla z(\mathbf{x}+\Delta\mathbf{x}) + \nabla z(\mathbf{x})}{2} \cdot \Delta\mathbf{x} = O(\delta^3)$$

$\|\Delta\mathbf{x}\|_\infty \leq \delta$

Fail is small: $O(\delta)$

Non-scalar second-order derivative test

$$\frac{z(\mathbf{x} + \Delta\mathbf{x}) - z(\mathbf{x})}{\delta} - \frac{\nabla z(\mathbf{x} + \Delta\mathbf{x}) + \nabla z(\mathbf{x})}{2\delta} \cdot \Delta\mathbf{x} = O(\delta^2)$$

$$\|\Delta\mathbf{x}\|_\infty \leq \delta$$

Non-scalar second-order derivative test

$$\frac{z(\mathbf{x}+\Delta\mathbf{x})-z(\mathbf{x})}{\delta}-\frac{\nabla z(\mathbf{x}+\Delta\mathbf{x})+\nabla z(\mathbf{x})}{2\delta}\cdot\Delta\mathbf{x}=O(\delta^2)$$

$\|\Delta\mathbf{x}\|_\infty \leq \delta$

Fail error: $O(1)$

Pass error: $O(\delta^2)$

Did it pass?

Introduce an error.

Did it pass?

Introduce an error.

See what a failing score looks like.

Testing Hessians

Test first derivatives.

Test second derivatives against first derivatives.

Incremental testing

Choose random x_0, x_1; small $\Delta x = x_1 - x_0$.

Incremental testing

Choose random x_0, x_1; small $\Delta x = x_1 - x_0$.

Compute at x_0: $a_0, a'_0, b_0, b'_0, c_0, c'_0, d_0, d'_0, \ldots$

Compute at x_1: $a_1, a'_1, b_1, b'_1, c_1, c'_1, d_1, d'_1, \ldots$

Incremental testing

Choose random x_0, x_1; small $\Delta x = x_1 - x_0$.

Compute at x_0: $a_0, a'_0, b_0, b'_0, c_0, c'_0, d_0, d'_0, \ldots$

Compute at x_1: $a_1, a'_1, b_1, b'_1, c_1, c'_1, d_1, d'_1, \ldots$

Diff test on each intermediate independently.

$$\frac{a_1 - a_0}{x_1 - x_0} - \frac{a'_1 + a'_0}{2} = O(\Delta x^2) \qquad \frac{b_1 - b_0}{x_1 - x_0} - \frac{b'_1 + b'_0}{2} = O(\Delta x^2)$$

$$\frac{c_1 - c_0}{x_1 - x_0} - \frac{c'_1 + c'_0}{2} = O(\Delta x^2) \qquad \frac{d_1 - d_0}{x_1 - x_0} - \frac{d'_1 + d'_0}{2} = O(\Delta x^2)$$

Very general strategy

Compute at x_0: $a_0, a_0', b_0, b_0', c_0, c_0', d_0, d_0', \ldots$

Compute at x_1: $a_1, a_1', b_1, b_1', c_1, c_1', d_1, d_1', \ldots$

- Test *any* partial
- $\dfrac{\partial c}{\partial a} \approx \dfrac{c_1 - c_0}{a_1 - a_0}$

Optimize incrementally

- Choose ordering
- Get it working
- Incremental optimization
    - Slight change
    - Test
    - Repeat

Outline

1 Basics

2 Practical considerations
- Modes of differentiation
- Testing
- Implicit differentiation

3 Differentiating matrix factorizations

4 Automatic differentiation

Implicit functions

Given x, compute y from $f(x, y) = 0$.

Implicit functions

Given x, compute y from $f(x, y) = 0$.

Compute y' from x'.

Implicit differentiation

Equation: $f(x, y) = 0$

Implicit differentiation

$$\text{Equation:} \quad f(x, y) = 0$$

$$\text{Differentiate:} \quad f_x(x, y)x' + f_y(x, y)y' = 0$$

Implicit differentiation

Equation: $f(x, y) = 0$

Differentiate: $f_x(x, y)x' + f_y(x, y)y' = 0$

Solve: $y' = -\dfrac{x' f_x}{f_y}$

Rule derivation: vector magnitude

$$\begin{aligned} \|\mathbf{x}\|^2 &= \mathbf{x} \cdot \mathbf{x} \\ 2\|\mathbf{x}\| \, \|\mathbf{x}\|' &= 2\mathbf{x} \cdot \mathbf{x}' \\ \|\mathbf{x}\|' &= \frac{\mathbf{x}}{\|\mathbf{x}\|} \cdot \mathbf{x}' \end{aligned}$$

Rule derivation: matrix inverse

$$\begin{aligned} \mathbf{B} &= \mathbf{A}^{-1} \\ \mathbf{A}\mathbf{B} - \mathbf{I} &= \mathbf{0} \\ \mathbf{A}'\mathbf{B} + \mathbf{A}\mathbf{B}' &= \mathbf{0} \\ \mathbf{A}\mathbf{B}' &= -\mathbf{A}'\mathbf{B} \\ \mathbf{B}' &= -\mathbf{B}\mathbf{A}'\mathbf{B} \end{aligned}$$

Differentiating the algorithm

Differentiate the *function*,
not the *algorithm* used to compute it.

Differentiating elementary functions

Differentiate $\sin x$ as $\cos x$

- Don't diff the Taylor series
- Use analytic formulas
- Oscillatory approximations
  - Accurate value
  - Wrong derivative

Differentiating matrix inverse

Use $(\mathbf{A}^{-1})' = -\mathbf{A}^{-1}\mathbf{A}'\mathbf{A}^{-1}$.

- Don't diff Gaussian elimination
- Discontinuous (pivoting)

Differentiating roots of polynomials

- Use implicit differentiation
- Don't diff bisection
    - How could you?

Outline

1. Basics
2. Practical considerations
3. Differentiating matrix factorizations
4. Automatic differentiation

Singular value defines *principle stretches*

Singular value decomposition: $\mathbf{F} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$

Singular value defines *principle stretches*

Singular value decomposition: $\mathbf{F} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$

Singular values: $\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_1 & & \\ & \sigma_2 & \\ & & \sigma_3 \end{pmatrix}$

Singular value defines *principle stretches*

Singular value decomposition: $\mathbf{F} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$

Singular values: $\mathbf{\Sigma} = \begin{pmatrix} \sigma_1 & & \\ & \sigma_2 & \\ & & \sigma_3 \end{pmatrix}$

Naturally separates deformation into

rotations: $\mathbf{U}$, $\mathbf{V}$

stretching: $\mathbf{\Sigma}$

Stretching takes energy

$$\psi(\mathbf{F}) = \hat{\psi}(\mathbf{\Sigma})$$

Stretching takes energy

$$\psi(\mathbf{F}) = \hat{\psi}(\mathbf{\Sigma})$$

Popular model in graphics (co-rotated):

$$\hat{\psi}(\mathbf{\Sigma}) = \mu \sum_k (\sigma_k - 1)^2 + \frac{\lambda}{2}\left(\sum_k (\sigma_k - 1)\right)^2$$

Stretching takes energy

$$\psi(\mathbf{F}) = \hat{\psi}(\mathbf{\Sigma})$$

Popular model in graphics (co-rotated):

$$\hat{\psi}(\mathbf{\Sigma}) = \mu \sum_k (\sigma_k - 1)^2 + \frac{\lambda}{2} \left(\sum_k (\sigma_k - 1) \right)^2$$

Its derivatives are sometimes "simplified."

Here is where it gets tough

We must differentiate this: $\mathbf{P} = \dfrac{\partial \psi}{\partial \mathbf{F}}$.

Here is where it gets tough

We must differentiate this: $\mathbf{P} = \dfrac{\partial \psi}{\partial \mathbf{F}}$.

Twice: $\dfrac{\partial \mathbf{P}}{\partial \mathbf{F}} = \dfrac{\partial^2 \psi}{\partial \mathbf{F} \partial \mathbf{F}}$.

Here is where it gets tough

We must differentiate this: $\mathbf{P} = \dfrac{\partial \psi}{\partial \mathbf{F}}$.

Twice: $\dfrac{\partial \mathbf{P}}{\partial \mathbf{F}} = \dfrac{\partial^2 \psi}{\partial \mathbf{F} \partial \mathbf{F}}$.

And we can do this.

Things are simpler in diagonal space

Quantity	Diagonal	Relationship	Properties
$\mathbf{F}$	$\mathbf{\Sigma}$	$\mathbf{F} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$	diagonal

Things are simpler in diagonal space

Quantity	Diagonal	Relationship	Properties
$\mathbf{F}$	$\mathbf{\Sigma}$	$\mathbf{F} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$	diagonal
$\mathbf{P} = \frac{\partial \psi}{\partial \mathbf{F}}$	$\hat{\mathbf{P}}$	$\mathbf{P} = \mathbf{U}\hat{\mathbf{P}}\mathbf{V}^T$	diagonal

Things are simpler in diagonal space

Quantity	Diagonal	Relationship	Properties
$\mathbf{F}$	$\mathbf{\Sigma}$	$\mathbf{F} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$	diagonal
$\mathbf{P} = \frac{\partial \psi}{\partial \mathbf{F}}$	$\hat{\mathbf{P}}$	$\mathbf{P} = \mathbf{U}\hat{\mathbf{P}}\mathbf{V}^T$	diagonal
$\mathbf{T} = \frac{\partial \mathbf{P}}{\partial \mathbf{F}}$	$\hat{\mathbf{T}}$	$T_{ijkl} = U_{im}U_{kr}\hat{T}_{mnrs}V_{jn}V_{ls}$	sparse

Strategy

- Compute diagonal space quantity
  - $\hat{\mathbf{P}}$, $\hat{\mathbf{T}}$

Strategy

- Compute diagonal space quantity
  - $\hat{\mathbf{P}}$, $\hat{\mathbf{T}}$
- Transform to original
  - $\mathbf{P}$, $\mathbf{T}$

Formula for $\hat{\mathbf{P}}$

$$\hat{P}_{ii} = \hat{\psi}_{,i}$$

Notes:

Formula for $\hat{\mathbf{P}}$

$$\hat{P}_{ii} = \hat{\psi}_{,i}$$

Notes:

- no summation implied

Formula for $\hat{\mathbf{P}}$

$$\hat{P}_{ii} = \hat{\psi}_{,i}$$

Notes:

- no summation implied
- $\hat{\psi}_{,i} = \dfrac{\partial \hat{\psi}}{\partial \sigma_i}$

Formula for $\hat{\mathbf{P}}$

$$\hat{P}_{ii} = \hat{\psi}_{,i}$$

Notes:

- no summation implied
- $\hat{\psi}_{,i} = \dfrac{\partial \hat{\psi}}{\partial \sigma_i}$
- $\hat{\mathbf{P}}$ is diagonal

Formula for $\hat{\mathbf{T}}$

Hessian term:

$$\hat{T}_{iikk} = \hat{\psi}_{,ik}$$

Formula for $\hat{\mathbf{T}}$

Hessian term:

$$\hat{T}_{iikk} = \hat{\psi}_{,ik}$$

Cross terms ($i \neq k$):

$$a_{ik} = \frac{\hat{\psi}_{,i} - \hat{\psi}_{,k}}{\sigma_i - \sigma_k} \qquad b_{ik} = \frac{\hat{\psi}_{,i} + \hat{\psi}_{,k}}{\sigma_i + \sigma_k}$$

$$\hat{T}_{ikik} = \frac{a_{ik} + b_{ik}}{2} \qquad \hat{T}_{ikki} = \frac{a_{ik} - b_{ik}}{2}$$

Formula for $\hat{\mathbf{T}}$

Hessian term:

$$\hat{T}_{iikk} = \hat{\psi}_{,ik}$$

Cross terms ($i \neq k$):

$$a_{ik} = \frac{\hat{\psi}_{,i} - \hat{\psi}_{,k}}{\sigma_i - \sigma_k} \qquad b_{ik} = \frac{\hat{\psi}_{,i} + \hat{\psi}_{,k}}{\sigma_i + \sigma_k}$$

$$\hat{T}_{ikik} = \frac{a_{ik} + b_{ik}}{2} \qquad \hat{T}_{ikki} = \frac{a_{ik} - b_{ik}}{2}$$

Note: $a_{ik} = a_{ki}$, $b_{ik} = b_{ki}$, $\hat{T}_{ikik} = \hat{T}_{kiki}$, $\hat{T}_{ikki} = \hat{T}_{kiik}$

Robustness notes: a_{ik}

$$a_{ik} = \frac{\hat{\psi}_{,i} - \hat{\psi}_{,k}}{\sigma_i - \sigma_k}$$

Notes:

- $\hat{\psi}$ symmetric in σ_k
- $\sigma_i \to \sigma_k$ implies $\hat{\psi}_{,i} \to \hat{\psi}_{,k}$
- limit exists
- compute analytically

Robustness notes: b_{ik}

$$b_{ik} = \frac{\hat{\psi}_{,i} + \hat{\psi}_{,k}}{\sigma_i + \sigma_k}$$

Notes:

- *might* be unbounded
- clamp it

See course notes for formulas for . . .

- Matrices that diagonalize as
  - $\mathbf{A} = \mathbf{U}\hat{\mathbf{A}}\mathbf{V}^T$ (generalizes $\mathbf{P}$ rule)
  - $\mathbf{A} = \mathbf{U}\hat{\mathbf{A}}\mathbf{U}^T$
  - $\mathbf{A} = \mathbf{V}\hat{\mathbf{A}}\mathbf{V}^T$
- Eigenvalue decomposition
  - $\mathbf{S} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^T$
  - $\mathbf{S}$ is symmetric

Outline

1. Basics
2. Practical considerations
3. Differentiating matrix factorizations
4. Automatic differentiation

Automatic differentiation

- Automate the differentiation process

Automatic differentiation

- Automate the differentiation process
- Not symbolic differentiation
    - Do not rearrange
    - Do not simplify
    - Avoids mess

Automatic differentiation

- Automate the differentiation process
- Not symbolic differentiation
  - Do not rearrange
  - Do not simplify
  - Avoids mess
- Many ways - lets explore some

Replace scalar with special type

- Store value *and* derivative
- Compute both together
- Overload operators and functions

Sample implementation

```
struct Diff_TT
{
  double x, dx;
};

Diff_TT operator+ (Diff_TT a, Diff_TT b)
{
  return {a.x + b.x, a.dx + b.dx};
}

Diff_TT operator* (Diff_TT a, Diff_TT b)
{
  return {a.x*b.x, a.dx*b.x + a.x*b.dx};
}
// and so on ...
```

Compile-time autodiff is great

- Intuitive
- Easy to implement
- Easy to use
    - Write code for value
    - Derivative for free
- Easy for compiler to optimize
    - Everything inlines

Extends to vectors, matrices

- `Diff_VT`: $\mathbf{u}'$
- `Diff_MT`: $\mathbf{A}'$
- `Diff_TV`: $\dfrac{\partial f}{\partial \mathbf{x}}$
- `Diff_VV`: $\dfrac{\partial \mathbf{u}}{\partial \mathbf{x}}$

Extends to Hessians

```
struct Hess_TT
{
  double x, dx, ddx;
};

Hess_TT operator+ (Hess_TT a, Hess_TT b)
{
  return {a.x+b.x, a.dx+b.dx, a.ddx+b.ddx};
}

Hess_TT operator* (Hess_TT a, Hess_TT b)
{
  return {a.x*b.x, a.dx*b.x + a.x*b.dx,
    a.ddx*b.x + 2*a.dx*b.dx + a.x*b.ddx};
}
```

Does not scale well

- Forward mode
- Scales poorly for many inputs

Does not scale well

- Forward mode
- Scales poorly for many inputs
  - optimization: $f(\mathbf{x})$

Does not scale well

- Forward mode
- Scales poorly for many inputs
  - optimization: $f(\mathbf{x})$
  - force: $\phi(\mathbf{x})$

Does not scale well

- Forward mode
- Scales poorly for many inputs
  - optimization: $f(\mathbf{x})$
  - force: $\phi(\mathbf{x})$
  - stress: $\psi(\mathbf{F})$

Reverse mode compile time autodiff

- Reverse mode is tough
- Compute derivatives in reverse order
- Need to record the code

Reverse mode via expression templates

Result of: $z = 3x^2 + \cos y$

Has type:

```
Add<Scale<Square<Var<0>>>,Cos<Var<1>>>
```

Reverse order traversal by recursion

Runtime

- Record operations in a list
- Walk the list to differentiate
- Forward and reverse mode
- Can handle variable input size

Not as efficient

- List construction
- Memory allocation
- No inlining
- No compiler optimization

Code generation

- Separate program
- Input: function code
- Output: derivative code

Very flexible

- Forward mode
- Reverse mode
- Mixed mode

Offline - take your time

- Run once
- Speed does not matter
- Optimize the results

Differentiate the function

- Autodiff may trace into functions
  - `exp`, `tgamma`, `sph_bessel`
  - Differentiates the *algorithm*
- Overload functions
  - Differentiates the *function*

Automatic differentiation has uses

- Prototyping
- Debugging
- Infrequently executed code
- Expect $2\times$ slowdown
  - Better for code-gen
  - Worse for dynamic
- No numerical robustness

Autodiff is a community

- http://www.autodiff.org/
- Software tools
- Libraries
- Reading lists

Manual derivatives are possible

I hope this course has shown you how.

Questions?