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

Minimize: $f(x)$
Motivation - numerical optimization

Minimize: $f(x)$

Numerical optimization uses gradients

$$x \leftarrow x - \alpha \nabla f$$

Gradient descent

More efficient methods need second derivatives

$$x \leftarrow x - \frac{\partial^2 f}{\partial x \partial x}^{-1} \nabla f$$

Newton's method
Motivation - numerical optimization

Minimize: $f(x)$

Numerical optimization uses gradients

$$x \leftarrow x - \alpha \nabla f$$  Gradient descent

More efficient methods need second derivatives

$$x \leftarrow x - \left( \frac{\partial^2 f}{\partial x \partial x} \right)^{-1} \nabla f$$  Newton’s method
potential energy: $\phi(x)$

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

Required for conservative forces.
Forces are often formulated via energy.
energy density: \( \psi(F) \)

stress: \( P = \frac{\partial \psi}{\partial F} \)

Note that \( F \) and \( P \) are matrices.
Implicit methods require derivatives

Backward Euler, trapezoid rule

Solved with Newton’s method

Second derivatives:

\[
\frac{\partial f}{\partial x} = - \frac{\partial^2 \phi}{\partial x \partial x}
\]

\[
\frac{\partial P}{\partial F} = \frac{\partial^2 \psi}{\partial F \partial F}
\]
Functions can be very complex

From a graphics paper:

\[ \alpha = \left( z - x \right) \cdot \left( y - x \right) / \parallel z - x \parallel \times \left( y - x \right) / \parallel z - x \parallel \]

\[ \beta = \left( x - y \right) \cdot \left( z - y \right) / \parallel x - y \parallel \times \left( z - y \right) / \parallel x - y \parallel \]

\[ \gamma = \left( y - z \right) \cdot \left( x - z \right) / \parallel y - z \parallel \times \left( x - z \right) / \parallel y - z \parallel \]

\[ d = \left( z - x \right) \times \left( y - x \right) / \parallel z - x \parallel \times \left( y - x \right) / \parallel z - x \parallel \cdot \left( x - c \right) \]

\[ E_d = 1 / d^2 \left( \alpha / \parallel y - z \parallel^2 + \beta / \parallel x - z \parallel^2 + \gamma / \parallel x - y \parallel^2 \right) \]

\[ E_a = 1 / k d^2 / \parallel x - z \parallel \times \left( y - z \right) / \parallel x - z \parallel^2 \]

\[ E = a \cdot E_d + b \cdot E_a \]

(Used Maple)

Need:

\[ \frac{\partial E}{\partial x}, \frac{\partial E}{\partial y}, \frac{\partial E}{\partial z}, \frac{\partial^2 E}{\partial x \partial x}, \frac{\partial^2 E}{\partial x \partial y}, \ldots, \frac{\partial^2 E}{\partial z \partial z} \]
Functions can be very complex

From a graphics paper:

\[ \alpha = \frac{(z - x) \cdot (y - x)}{\| (z - x) \times (y - x) \|} \]
\[ \gamma = \frac{(y - z) \cdot (x - z)}{\| (y - z) \times (x - z) \|} \]

\[ \beta = \frac{(x - y) \cdot (z - y)}{\| (x - y) \times (z - y) \|} \]
\[ d = \frac{(z - x) \times (y - x)}{\| (z - x) \times (y - x) \|} \cdot (x - c) \]

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

\[
E_a = \frac{1}{kd^2} \| (x - z) \times (y - z) \|^2
\]

\[
E = a \cdot E_d + b \cdot E_a
\]
Functions can be very complex

From a graphics paper:

$$\alpha = \frac{(z - x) \cdot (y - x)}{\left\| (z - x) \times (y - x) \right\|}$$

$$\beta = \frac{(x - y) \cdot (z - y)}{\left\| (x - y) \times (z - y) \right\|}$$

$$\gamma = \frac{(y - z) \cdot (x - z)}{\left\| (y - z) \times (x - z) \right\|}$$

$$d = \frac{(z - x) \times (y - x)}{\left\| (z - x) \times (y - x) \right\|} \cdot (x - c)$$

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

$$E_a = \frac{1}{kd^2} \|(x - z) \times (y - z)\|^2$$

$$E = a \cdot E_d + b \cdot E_a$$

Need:

$$\frac{\partial E}{\partial x}, \frac{\partial E}{\partial y}, \frac{\partial E}{\partial z}, \frac{\partial^2 E}{\partial x \partial x}, \frac{\partial^2 E}{\partial x \partial y}, \cdots, \frac{\partial^2 E}{\partial z \partial z}$$
Functions can be very complex

From a graphics paper:

\[ \alpha = \frac{(z - x) \cdot (y - x)}{\| (z - x) \times (y - x) \|} \]

\[ \beta = \frac{(x - y) \cdot (z - y)}{\| (x - y) \times (z - y) \|} \]

\[ \gamma = \frac{(y - z) \cdot (x - z)}{\| (y - z) \times (x - z) \|} \]

\[ d = \frac{(z - x) \times (y - x)}{\| (z - x) \times (y - x) \|} \cdot (x - c) \]

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

\[ E_a = \frac{1}{kd^2} \| (x - z) \times (y - z) \|^2 \]

\[ E = a \cdot E_d + b \cdot E_a \]

Need: \( \frac{\partial E}{\partial x}, \frac{\partial E}{\partial y}, \frac{\partial E}{\partial z}, \frac{\partial^2 E}{\partial x \partial x}, \frac{\partial^2 E}{\partial x \partial y}, \cdots, \frac{\partial^2 E}{\partial z \partial z} \) (Used Maple)
It may be hard to know it is right.

Sometimes the only symptom is \textit{slow convergence}.
With the right ideas, we can do this

This course will show you how.
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

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

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

Expensive for gradients/Hessians
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
Pros:
- Compute derivatives automatically
- Can generate code automatically.
Modest example: \( f(u, v) = \| u(u \cdot v)^2 - v \| u \|_3 \|_2^2 \)
How bad can it really be?

Modest example: \( f(u, v) = \| u(u \cdot v)^2 - v \| u \|_3 \|^2 \)

What I did in Maple:

- Compute Hessian \( H = \frac{\partial^2 f}{\partial u \partial u} \)
- Simplify
- Generate C code for just \( H_{11} \).
- Simplify the code
\( t_1 = v_2 \times v_2; \ t_4 = v_1 \times v_1; \ t_6 = t_1 \times t_1; \ t_8 = t_1 / 10; \ t_9 = v_3 \times v_3; \ t_{10} = t_9 / 10; \ t_{12} = u_2 \times u_2; \ t_{13} = t_{12} \times t_{12}; \ t_{19} = u_1 \times u_1; \ t_{24} = t_{12} \times u_2; \ t_{31} = t_9 / 5; \ t_{33} = u_3 \times u_3; \ t_{35} = 3 \times t_1; \ t_{41} = u_1 \times u_1; \ t_{42} = t_4 \times t_4; \ t_{46} = 3 \times t_9; \ t_{52} = u_3 \times v_3; \ t_{61} = 0.8 \times u_1 \times u_3 \times v_1 \times v_3; \ t_{73} = t_{33} \times t_{33}; \ t_{77} = t_{33} \times u_3; \ t_{88} = t_{41} \times u_1; \ t_{94} = t_{41} \times t_{41}; \ t_{100} = \sqrt{t_{41} + t_{12} + t_{33}}; \ t_{102} = t_4 / 5; \)

\[
H_{11} = -60 / t_{100} \times (t_{100} \times (t_{13} \times (t_4 \times (-t_1 / 5 - 0.1) - t_6 / 30 - t_8 - t_{10}) - 0.4 \times t_{24} \times v_2 \times (u_3 \times (t_4 + t_1 / 3) \times v_3 + (t_4 + t_1) \times t_{19}) + t_{12} \times (t_{33} \times (t_4 \times (-t_1 - t_9 - 1) / 5 + t_1 \times (-t_9 - 1) / 5 - t_{31}) - 0.4 \times u_3 \times (t_4 + t_{35}) \times v_3 \times t_{19} - (t_{42} + t_4 \times (6 \times t_1 + 3) + t_{35} + t_{46}) \times t_{41} / 5) - 4. / 3 \times u_2 \times (t_{33} \times (0.3 \times t_4 + t_{10}) + t_{61} + t_4 \times t_{41}) \times v_2 \times (t_{19} + t_{52}) + t_{73} \times (t_4 \times (-t_{31} - 0.1) - t_8 - (t_9 + 3) \times t_9 / 30) - 0.4 \times t_{77} \times (t_4 + t_9) \times v_3 \times t_{19} - t_{33} \times (t_{42} + t_4 \times (6 \times t_9 + 3) + t_{35} + t_{46}) \times t_{41} / 5 - 4. / 3 \times v_3 \times t_4 \times v_1 \times u_3 \times t_{88} - (t_{42} + t_4 + t_1 + t_9) \times t_{94} / 2) + (u_2 \times v_2 + t_{19} + t_{52}) \times (t_{13} \times (t_{102} + t_8) + 0.8 \times t_{24} \times v_2 \times (t_{19} + t_{52} / 4) + t_{12} \times (t_{33} \times (0.4 \times t_4 + t_8 + t_{10}) + t_{61} + 1.1 \times (t_4 + 2. / 11 \times t_1) \times t_{41}) + u_2 \times (t_{88} \times v_1 + 0.4 \times t_{41} \times t_{52} + 0.8 \times u_1 \times v_1 \times t_{33} + v_3 \times t_{77} / 5) \times v_2 + t_{73} \times (t_{102} + t_{10}) + 0.8 \times v_3 \times u_1 \times v_1 \times t_{77} + 1.1 \times t_{33} \times t_{41} \times (t_4 + 2. / 11 \times t_9) + t_{88} \times v_3 \times u_3 \times v_1 + t_4 \times t_{94}));
\]
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) + (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: \( a' = f'(g(x))g'(x) \)

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

Pieces: \( g = x^2 + 1, \quad f = \sqrt{g} \)
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: \( g' = 2x, \ f' = \frac{g'}{2f} \) \hfill (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.

\[
\begin{align*}
  a &= 1 + x^2 \\
  b &= x^3 \\
  c &= \sqrt{a} \\
  d &= b + c \\
  f &= d^2
\end{align*}
\]

Note the use of the chain rule.
The chain rule is the key

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

**Step 1:** break it into small pieces.

\[
a = 1 + x^2 \quad b = x^3 \quad c = \sqrt{a} \quad d = b + c \quad f = d^2
\]

**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.

\[ a = 1 + x^2 \quad b = x^3 \quad c = \sqrt{a} \quad d = b + c \quad f = d^2 \]

**Step 2:** compute the derivative of each step

\[ a' = 2x \quad b' = 3x^2 \quad c' = \frac{a'}{2c} \quad d' = b' + c' \quad f' = 2dd' \]

Note the use of the chain rule.
This is good code

\[ a = 1 + x^2 \quad b = x^3 \quad c = \sqrt{a} \quad d = b + c \quad f = d^2 \]

\[ a' = 2x \quad b' = 3x^2 \quad c' = \frac{a'}{2c} \quad d' = b' + c' \quad f' = 2dd' \]
Second derivatives are easy, too

\[
\begin{align*}
a &= 1 + x^2 & b &= x^3 & c &= \sqrt{a} & d &= b + c & f &= d^2 \\
a' &= 2x & b' &= 3x^2 & c' &= \frac{a'}{2c} & d' &= b' + c' & f' &= 2dd' \\
a'' &= 2 & b'' &= 6x & c'' &= \frac{a''c - a'c'}{2a} & d'' &= b'' + c'' & f'' &= 2(d')^2 + 2dd''
\end{align*}
\]
This works for more complex stuff

Earlier example: \[ f(u, v) = \| u(u \cdot v)^2 - v\| u \| ^3 \| ^2 \]
This works for more complex stuff

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

Step 1:

\[
\begin{align*}
  a &= u \cdot v & b &= \|u\| \\
  w &= cu - dv & c &= a^2 & d &= b^3 \\
  f &= \|w\|^2
\end{align*}
\]
This works for more complex stuff

Earlier example: \( f(u, v) = \|u(u \cdot v)^2 - v\| u \|^3\|^2 \)

Step 1:

\[
\begin{align*}
  a &= u \cdot v \\
  b &= \|u\| \\
  c &= a^2 \\
  d &= b^3 \\
  w &= cu - dv
\end{align*}
\]

Step 2:

\[
\begin{align*}
  a_u &= v \\
  b_u &= \frac{u}{b} \\
  c_u &= 2aa_u \\
  d_u &= 3b^2b_u \\
  w_u &= cI + uc_u^T - vd_u^T \\
  f_u &= 2w \cdot w_u
\end{align*}
\]
But wait, we needed second derivatives

The first few are not too bad.

\[ a = \mathbf{u} \cdot \mathbf{v} \quad b = \| \mathbf{u} \| \quad c = a^2 \quad d = b^3 \]

\[ a_u = \mathbf{v} \quad b_u = \frac{\mathbf{u}}{b} \quad c_u = 2 a a_u \quad d_u = 3 b^2 b_u \]

\[ a_{uu} = 0 \quad b_{uu} = \frac{1}{b} (\mathbf{I} - b_u b_u^T) \quad c_{uu} = 2 a_u a_u^T \quad d_{uu} = 6 b b_u b_u^T + 3 b^2 b_{uu} \]
Complication: tensors

\[ w = cu - dv \]
\[ w_u = cI + uc_u^T - vd_u^T \]
\[ w_{uu} = \text{?!?} \]

\[ f = \|w\|^2 \]
\[ f_u = 2w \cdot w_u \]

\( w \) is a vector.
\( w_u \) is a matrix.
\( w_{uu} \) is a rank-3 tensor.
Complication: tensors

\[
\begin{align*}
w &= cu - dv \\
w_u &= cI + uc_u^T - vd_u^T \\
w_{uu} &= ???
\end{align*}
\]

\[
\begin{align*}
f &= \|w\|^2 \\
f_u &= 2w \cdot w_u \\
f_{uu} &= 2w \cdot w_{uu} + 2w_u^T w_u
\end{align*}
\]

\(w\) is a vector.

\(w_u\) is a matrix.

\(w_{uu}\) is a rank-3 tensor.

Note the usage of \(w_{uu}\).
Complication: tensors

\[ w = cu - dv \]
\[ w_u = cI + uc^T_u - vd^T_u \]
\[ w_{uu} = ?? \]

\[ f = \|w\|^2 \]
\[ f_u = 2w \cdot w_u \]
\[ f_{uu} = 2w \cdot w_{uu} + 2w_u^T w_u \]

\( w \) is a vector.
\( w_u \) is a matrix.
\( w_{uu} \) is a rank-3 tensor.

Note the usage of \( w_{uu} \). Only need matrix \( z \).
Clever idea: avoid computing $w_{uu}$

Compute $z = w \cdot w_{uu}$ instead of $w_{uu}$. $z$ is a matrix.
Clever idea: avoid computing $w_{uu}$

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

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

$$f_{uu} = 2z + 2w_u^T w_u$$
Clever idea: avoid computing $w_{uu}$

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

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

$$f_{uu} = 2z + 2w_u^T w_u$$

This is all of $H = f_{uu}$, 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: \( \frac{\partial \mathbf{P}}{\partial F} \)
Tensor index notation solves two problems

- Deal with tensors
  - Gradient of matrix: $w_{uu}$
  - Rank-4 tensor: $\frac{\partial P}{\partial F}$

- Forgotten derivative rules
  - $\nabla (u \cdot v)$
  - $\nabla (fu)$
  - $\nabla \cdot (u \times v)$
Refer to objects by their components

Scalar: $a \rightarrow a$

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

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

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  \[ A = uv^T \] calculation  \[ A_{ik} = u_i v_k \] index notation  \[ A_{ik} = u_i v_k \]
vector notation

\[ A = uv^T \]

\[ a = u \cdot v \]

calculation

\[ A_{ik} = u_i v_k \]

\[ a = \sum_i u_i v_i \]

index notation

\[ A_{ik} = u_i v_k \]

\[ a = u_i v_i \]
vector notation             calculation             index notation

\( \mathbf{A} = \mathbf{u} \mathbf{v}^T \)            \( A_{ik} = u_i v_k \)            \( A_{ik} = u_i v_k \)

\( a = \mathbf{u} \cdot \mathbf{v} \)            \( a = \sum_i u_i v_i \)            \( a = u_i v_i \)

\( \mathbf{v} = \mathbf{A} \mathbf{u} \)            \( v_i = \sum_k A_{ik} u_k \)            \( v_i = A_{ik} u_k \)
<table>
<thead>
<tr>
<th>Vector Notation</th>
<th>Calculation</th>
<th>Index Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathbf{A} = \mathbf{uv}^T )</td>
<td>( A_{ik} = u_i v_k )</td>
<td>( A_{ik} = u_i v_k )</td>
</tr>
<tr>
<td>( \mathbf{a} = \mathbf{u} \cdot \mathbf{v} )</td>
<td>( a = \sum_i u_i v_i )</td>
<td>( a = u_i v_i )</td>
</tr>
<tr>
<td>( \mathbf{v} = \mathbf{Au} )</td>
<td>( v_i = \sum_k A_{ik} u_k )</td>
<td>( v_i = A_{ik} u_k )</td>
</tr>
<tr>
<td>( \mathbf{A} = \mathbf{BC} )</td>
<td>( A_{ir} = \sum_k B_{ik} C_{kr} )</td>
<td>( A_{ir} = B_{ik} C_{kr} )</td>
</tr>
</tbody>
</table>
vector notation                  calculation                  index notation
\[ A = uv^T \]                      \[ A_{ik} = u_i v_k \]                      \[ A_{ik} = u_i v_k \]
\[ a = u \cdot v \]     \[ a = \sum_i u_i v_i \]                         \[ a = u_i v_i \]
\[ v = Au \]                           \[ v_i = \sum_k A_{ik} u_k \]        \[ v_i = A_{ik} u_k \]
\[ A = BC \]                              \[ A_{ir} = \sum_k B_{ik} C_{kr} \] \[ A_{ir} = B_{ik} C_{kr} \]
\[ A = B^T C \]                          \[ A_{ir} = \sum_k B_{ki} C_{kr} \] \[ A_{ir} = B_{ki} C_{kr} \]
<table>
<thead>
<tr>
<th>Vector Notation</th>
<th>Calculation</th>
<th>Index Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathbf{A} = \mathbf{uv}^T )</td>
<td>( A_{ik} = u_i v_k )</td>
<td>( A_{ik} = u_i v_k )</td>
</tr>
<tr>
<td>( a = \mathbf{u} \cdot \mathbf{v} )</td>
<td>( a = \sum_i u_i v_i )</td>
<td>( a = u_i v_i )</td>
</tr>
<tr>
<td>( \mathbf{v} = \mathbf{A}\mathbf{u} )</td>
<td>( v_i = \sum_k A_{ik} u_k )</td>
<td>( v_i = A_{ik} u_k )</td>
</tr>
<tr>
<td>( \mathbf{A} = \mathbf{BC} )</td>
<td>( A_{ir} = \sum_k B_{ik} C_{kr} )</td>
<td>( A_{ir} = B_{ik} C_{kr} )</td>
</tr>
<tr>
<td>( \mathbf{A} = \mathbf{B}^T \mathbf{C} )</td>
<td>( A_{ir} = \sum_k B_{ki} C_{kr} )</td>
<td>( A_{ir} = B_{ki} C_{kr} )</td>
</tr>
<tr>
<td>( a = \text{tr}(\mathbf{A}) )</td>
<td>( a = \sum_i A_{ii} )</td>
<td>( a = A_{ii} )</td>
</tr>
</tbody>
</table>
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 $(u_i v_i)(u_i v_i)$. 
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

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

\[ u = v \times w \] 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

\[ f_{,r} = \frac{\partial f}{\partial x_r} \quad u_{i,r} = \frac{\partial u_i}{\partial x_r} \]

\[ f_{,rs} = \frac{\partial^2 f}{\partial x_r \partial x_s} \quad u_{i,rs} = \frac{\partial^2 u_i}{\partial x_r \partial x_s} \]

Special case:

\[ x_{i,r} = \frac{\partial x_i}{\partial x_r} = \delta_{ir} \]

Constants:

\[ \delta_{ik,r} = 0, \quad e_{ikr,s} = 0 \]
Derivatives in index notation

Differentiation denoted with a comma

\[ f_{,r} = \frac{\partial f}{\partial x_r} \]

\[ u_{i,r} = \frac{\partial u_i}{\partial x_r} \]

\[ f_{,rs} = \frac{\partial^2 f}{\partial x_r \partial x_s} \]

\[ u_{i,rs} = \frac{\partial^2 u_i}{\partial x_r \partial x_s} \]

Special case: \[ x_{i,r} = \frac{\partial x_i}{\partial x_r} = \delta_{ir} \]
Derivatives in index notation

Differentiation denoted with a comma

\[ f_{,r} = \frac{\partial f}{\partial x_r} \]
\[ f_{,rs} = \frac{\partial^2 f}{\partial x_r \partial x_s} \]
\[ u_{i,r} = \frac{\partial u_i}{\partial x_r} \]
\[ u_{i,rs} = \frac{\partial^2 u_i}{\partial x_r \partial x_s} \]

Special case: \( x_{i,r} = \frac{\partial x_i}{\partial x_r} = \delta_{ir} \)

Constants: \( \delta_{ik,r} = 0, \ e_{ikr,s} = 0 \)
gradient

$\nabla f$

$\frac{\partial f}{\partial x_r}$

$f_{,r}$
gradient $\nabla f$

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

divergence $\nabla \cdot \mathbf{u}$

$\sum_r \frac{\partial u_r}{\partial x_r}$ $u_{r,r}$
gradient \quad \nabla f \quad \frac{\partial f}{\partial x_r} \quad f_{,r}

divergence \quad \nabla \cdot \mathbf{u} \quad \sum_r \frac{\partial u_r}{\partial x_r} \quad u_{r,r}

curl \quad \nabla \times \mathbf{u} \quad e^{ikr} u_{k,r}
gradient \quad \nabla f \quad \frac{\partial f}{\partial x_r} \quad f_{,r}

divergence \quad \nabla \cdot \mathbf{u} \quad \sum_r \frac{\partial u_r}{\partial x_r} \quad u_{r,r}

curl \quad \nabla \times \mathbf{u} \quad e^{ikr} u_{k,r}

Laplacian \quad \nabla^2 f \quad \sum_r \frac{\partial^2 f}{\partial x_r \partial x_r} \quad f_{,rr}
<table>
<thead>
<tr>
<th><strong>Operator</strong></th>
<th><strong>Symbol</strong></th>
<th><strong>Mathematical Expression</strong></th>
<th><strong>Variables</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Gradient</strong></td>
<td>( \nabla f )</td>
<td>( \frac{\partial f}{\partial x_r} )</td>
<td>( f, r )</td>
</tr>
<tr>
<td><strong>Divergence</strong></td>
<td>( \nabla \cdot \mathbf{u} )</td>
<td>( \sum_r \frac{\partial u_r}{\partial x_r} )</td>
<td>( u_r, r )</td>
</tr>
<tr>
<td><strong>Curl</strong></td>
<td>( \nabla \times \mathbf{u} )</td>
<td>( e_{ikr} u_{k,r} )</td>
<td></td>
</tr>
<tr>
<td><strong>Laplacian</strong></td>
<td>( \nabla^2 f )</td>
<td>( \sum_r \frac{\partial^2 f}{\partial x_r \partial x_r} )</td>
<td>( f, rr )</td>
</tr>
<tr>
<td><strong>Vector Laplacian</strong></td>
<td>( \nabla^2 \mathbf{u} )</td>
<td>( \sum_r \frac{\partial^2 u_i}{\partial x_r \partial x_r} )</td>
<td>( u_{i, rr} )</td>
</tr>
</tbody>
</table>
Components are *scalars*, so scalar rules apply.
Scalar derivative rules apply

Components are *scalars*, so scalar rules apply.

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

Index: $(u_i w_i)_r = u_{i,r} w_i + u_i w_{i,r} \implies \nabla u^T w + \nabla w^T u$
Scalar derivative rules apply

Components are *scalars*, so scalar rules apply.

Vector: $\nabla (u \cdot w) =$?
Index: $(u_i w_i)_r = u_{i,r} w_i + u_i w_{i,r} \implies \nabla u^T w + \nabla w^T u$

Vector: $\nabla \cdot (u \times w) =$?
Index: $(e_{ikr} u_k w_r)_s = e_{ikr} u_{k,s} w_r + e_{ikr} u_k w_{r,s}$
Recall: $u_{i,s} = \delta_{is}$ \hspace{1em} $v_{i,s} = 0$

$$w_u = cI + uc_u^T - vd_u^T$$
Unfinished business

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

\[
\mathbf{w}_u = c \mathbf{I} + \mathbf{uc}_u^T - \mathbf{vd}_u^T
\]

\[
\omega_{i,r} = c\delta_{ir} + u_i c_{r} - v_i d_{r}
\]
Unfinished business

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

\[
\begin{align*}
\mathbf{w}_u &= c\mathbf{I} + \mathbf{uc}_u^T - \mathbf{vd}_u^T \\
\mathbf{w}_{i,r} &= c\delta_{ir} + u_i\mathbf{c}_{r} - \mathbf{v}\mathbf{d}_{r} \\
\mathbf{w}_{i,rs} &= c_{,s}\delta_{ir} + u_i\mathbf{c}_{rs} + u_{i,s}\mathbf{c}_{r} - \mathbf{v}\mathbf{d}_{rs}
\end{align*}
\]
Unfinished business

Recall: \( u_{i,s} = \delta_{is} \) \( v_{i,s} = 0 \)

\[
\mathbf{w}_u = c \mathbf{I} + \mathbf{uc}_u^T - \mathbf{vd}_u^T
\]

\[
w_{i,r} = c \delta_{ir} + u_{i}c_{,r} - v_{i}d_{,r}
\]

\[
w_{i,rs} = c,s \delta_{ir} + u_{i}c_{,rs} + u_{i,s}c_{,r} - v_{i}d_{,rs}
\]

\[
w_i w_{i,rs} = w_i c,s \delta_{ir} + w_i u_{i}c_{,rs} + w_i \delta_{is}c_{,r} - w_i v_{i}d_{,rs}
\]
Unfinished business

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

$$w_u = cI + uc_u^T - vd_u^T$$

$$w_{i,r} = c\delta_{ir} + u_ic_{,r} - v_id_{,r}$$

$$w_{i,rs} = c_{,s}\delta_{ir} + u_ic_{,rs} + u_{i,s}c_{,r} - v_id_{,rs}$$

$$w_iw_{i,rs} = w_ic_{,s}\delta_{ir} + w_iu_ic_{,rs} + w_i\delta_{is}c_{,r} - w_iv_id_{,rs}$$

$$w_iw_{i,rs} = w_r c_{,s} + (w_iu_i)c_{,rs} + c_{,r}w_s - (w_iv_i)d_{,rs}$$
Recall: $u_{i,s} = \delta_{i,s}$, $v_{i,s} = 0$

\[
\begin{align*}
\mathbf{w}_u &= c\mathbf{I} + \mathbf{uc}_u^T - \mathbf{vd}_u^T \\
\omega_{i,r} &= c\delta_{ir} + u_ic_{r} - v_id_{r} \\
\omega_{i,rs} &= c_{s}\delta_{ir} + u_ic_{rs} + u_{i,s}c_{r} - v_id_{rs} \\
\omega_i\omega_{i,rs} &= \omega_ic_{s}\delta_{ir} + \omega_iu_ic_{rs} + \omega_i\delta_{is}c_{r} - \omega_iv_id_{rs} \\
\omega_i\omega_{i,rs} &= \omega_rc_{s} + (\omega_iu_i)c_{rs} + c_{r}\omega_s - (\omega_iv_i)d_{rs} \\
\mathbf{z} &= \mathbf{w} \cdot \mathbf{w}_{uu} = \mathbf{wc}_u^T + (\mathbf{w} \cdot \mathbf{u})c_{uu} + c_u\mathbf{w}^T - (\mathbf{w} \cdot \mathbf{v})d_{uu}
\end{align*}
\]
Derivatives in many variables at once

E.g., $f(u, w)$. Need $\frac{\partial f}{\partial u}$ and $\frac{\partial f}{\partial w}$.
Derivatives in many variables at once

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

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

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

Simplify after it works.
Derivatives in many variables at once

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

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

For $\frac{\partial f}{\partial w}$, let $u_{i,r} = 0$ and $w_{i,r} = \delta_{ir}$
Derivatives in many variables at once

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

Work out \( f_r \). Do not assume \( f_r = \frac{\partial f}{\partial u} \) or \( f_r = \frac{\partial f}{\partial w} \).

Make two copies of the code:

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

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

Simplify \textit{after} it works.
Different indices for different variables

\[ r \text{ for } x \]
\[ \alpha \text{ for } y \]

\[ f, r = \frac{\partial f}{\partial x_r} \]
\[ f, \alpha = \frac{\partial f}{\partial y_\alpha} \]
$\psi_{,(rs)} = \frac{\partial \psi}{\partial F_{rs}}$
\[ \psi_{, (rs)} = \frac{\partial \psi}{\partial F_{rs}} \]
\[ F_{ik, (rs)} = \delta_{ir} \delta_{ks} \]
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: $a = a(x)$, $b = b(a)$, $c = c(b)$, $y = y(c)$

\[
\frac{\partial b}{\partial x} = \frac{\partial b}{\partial a} \frac{\partial a}{\partial x} \quad \frac{\partial c}{\partial x} = \frac{\partial c}{\partial b} \frac{\partial b}{\partial x} \quad \frac{\partial y}{\partial x} = \frac{\partial y}{\partial c} \frac{\partial c}{\partial x}
\]
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: $\frac{\partial ?}{\partial x}$
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: \( a = a(x), b = b(a), c = c(b), y = y(c) \)

\[
\frac{\partial y}{\partial b} = \frac{\partial y}{\partial c} \frac{\partial c}{\partial b}, \quad \frac{\partial y}{\partial a} = \frac{\partial y}{\partial b} \frac{\partial b}{\partial a}, \quad \frac{\partial y}{\partial x} = \frac{\partial y}{\partial a} \frac{\partial a}{\partial x}
\]

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: \( x \rightarrow \mathbb{R}^6, \ a \rightarrow \mathbb{R}^3, \ b \rightarrow \mathbb{R}^3, \ y \rightarrow \mathbb{R}^1 \)
Cost

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

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

Forward: $54 + 18$
Cost

Sizes: \( x \rightarrow \mathbb{R}^6, \ a \rightarrow \mathbb{R}^3, \ b \rightarrow \mathbb{R}^3, \ y \rightarrow \mathbb{R}^1 \)

\[
\frac{\partial y}{\partial x} = \frac{\partial y}{\partial b} \left( \frac{\partial b}{\partial a} \frac{\partial a}{\partial x} \right)_{1 \times 3} \left( \begin{array}{c} 3 \times 6; (54*) \\ \end{array} \right)
\]

Forward: \( 54 + 18 \)

\[
\frac{\partial y}{\partial x} = \left( \frac{\partial y}{\partial b} \frac{\partial b}{\partial a} \right) \frac{\partial a}{\partial x} \left( \begin{array}{c} 1 \times 3; (9*) \\ 3 \times 6 \end{array} \right)
\]

Reverse: \( 9 + 18 \)
Calculating $\frac{\partial y}{\partial x}$

Forward is cheaper if $x \ll y$

- Easier to write

Efficiency of forward vs reverse modes
Efficiency of forward vs reverse modes

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

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

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

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!

\[ z(x + \Delta x) - z(x) \Delta x - z'(x) = O(\Delta x) \]

How small should \( \Delta x \) be?

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

Refinement test?
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 $\Delta x$ be?
Test derivatives against definition!

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

How small should $\Delta x$ 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 \( \Delta x \) 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 \( \Delta x \approx \epsilon^{1/3} \sim 10^{-5} \quad \epsilon \approx 2 \times 10^{-16} \)
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 $\Delta x \approx \epsilon^{1/3} \sim 10^{-5}$ \quad $\epsilon \approx 2 \times 10^{-16}$

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

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 \(\Delta x \approx \epsilon^{1/3} \sim 10^{-5}\) \(\epsilon \approx 2 \times 10^{-16}\)

Fail error: \(O(1)\)

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

Vector \(\Delta x\)?
“Multiply through” by $\Delta \mathbf{x}$

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

$$\|\Delta \mathbf{x}\|_{\infty} \leq \delta$$
Non-scalar second-order derivative test

“Multiply through” by \( \Delta x \)

\[
\begin{align*}
 z(x + \Delta x) - z(x) - \frac{\nabla z(x + \Delta x) + \nabla z(x)}{2} \cdot \Delta x &= O(\delta^3) \\
\|\Delta x\|_\infty &\leq \delta \\
\text{Fail is small: } O(\delta)
\end{align*}
\]
Non-scalar second-order derivative test

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

\[
\|\Delta x\|_\infty \leq \delta
\]
Non-scalar second-order derivative test

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

\[\|\Delta x\|_\infty \leq \delta\]

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.

\[
\frac{a_1 - a_0}{x_1 - x_0} - \frac{a'_1 + a'_0}{2} = O(\Delta x^2)
\]

\[
\frac{b_1 - b_0}{x_1 - x_0} - \frac{b'_1 + b'_0}{2} = O(\Delta x^2)
\]

\[
\frac{c_1 - c_0}{x_1 - x_0} - \frac{c'_1 + c'_0}{2} = O(\Delta x^2)
\]

\[
\frac{d_1 - d_0}{x_1 - x_0} - \frac{d'_1 + d'_0}{2} = O(\Delta x^2)
\]
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
  \[ \frac{\partial c}{\partial a} \approx \frac{c_1 - c_0}{a_1 - a_0} \]
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
Given $x$, compute $y$ from $f(x, y) = 0$. 
Given $x$, compute $y$ from $f(x, y) = 0$.

Compute $y'$ from $x'$. 
Implicit differentiation

Equation: \( f(x, y) = 0 \)
Implicit differentiation

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

Differentiate: \( f_x(x, y)x' + f_y(x, y)y' = 0 \)
Implicit differentiation

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

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

Solve: \( y' = -\frac{x' f_x}{f_y} \)
Rule derivation: vector magnitude

\[ \| \mathbf{x} \|^2 = \mathbf{x} \cdot \mathbf{x} \]

\[ 2 \| \mathbf{x} \| \| \mathbf{x} \|' = 2 \mathbf{x} \cdot \mathbf{x}' \]

\[ \| \mathbf{x} \|' = \frac{\mathbf{x}}{\| \mathbf{x} \|} \cdot \mathbf{x}' \]
Rule derivation: matrix inverse

\[ B = A^{-1} \]

\[ AB - I = 0 \]

\[ A'B + AB' = 0 \]

\[ AB' = -A'B \]

\[ B' = -BA'B \]
Differentiating the **function**, not the **algorithm** used to compute it.
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 \((A^{-1})' = -A^{-1}A'A^{-1}\).

- 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 decomposition: \( F = U \Sigma V^T \)
Singular value defines *principle stretches*

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

Singular values: \( \Sigma = \begin{pmatrix} \sigma_1 & & \\ & \sigma_2 & \\ & & \sigma_3 \end{pmatrix} \)
Singular value defines *principle stretches*

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

Singular values: \( \Sigma = \begin{pmatrix} \sigma_1 & \sigma_2 \\ \sigma_2 & \sigma_3 \end{pmatrix} \)

Naturally separates deformation into

*rotations*: \( \mathbf{U} \), \( \mathbf{V} \)

*stretching*: \( \Sigma \)
Stretching takes energy

\[ \psi(F) = \hat{\psi}(\Sigma) \]
Stretching takes energy

\[ \psi(F) = \hat{\psi}(\Sigma) \]

Popular model in graphics (co-rotated):

\[ \hat{\psi}(\Sigma) = \mu \sum_k (\sigma_k - 1)^2 + \frac{\lambda}{2} \left( \sum_k (\sigma_k - 1) \right)^2 \]
Stretching takes energy

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

Popular model in graphics (co-rotated):

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

Its derivatives are sometimes “simplified.”
Here is where it gets tough

We must differentiate this: \( P = \frac{\partial \psi}{\partial F} \).
We must differentiate this: \( \mathbf{P} = \frac{\partial \psi}{\partial \mathbf{F}} \).

Twice: \( \frac{\partial \mathbf{P}}{\partial \mathbf{F}} = \frac{\partial^2 \psi}{\partial \mathbf{F} \partial \mathbf{F}} \).
Here is where it gets tough

We must differentiate this: \( \mathbf{P} = \frac{\partial \psi}{\partial F} \).

Twice: \( \frac{\partial \mathbf{P}}{\partial F} = \frac{\partial^2 \psi}{\partial F \partial F} \).

And we can do this.
Things are simpler in diagonal space

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Diagonal</th>
<th>Relationship</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F$</td>
<td>$\Sigma$</td>
<td>$F = U\Sigma V^T$</td>
<td>diagonal</td>
</tr>
</tbody>
</table>
Things are simpler in diagonal space

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Diagonal</th>
<th>Relationship</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F$</td>
<td>$\Sigma$</td>
<td>$F = U\Sigma V^T$</td>
<td>diagonal</td>
</tr>
<tr>
<td>$P = \frac{\partial \psi}{\partial F}$</td>
<td>$\hat{P}$</td>
<td>$P = U\hat{P} V^T$</td>
<td>diagonal</td>
</tr>
</tbody>
</table>
Things are simpler in diagonal space

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Diagonal</th>
<th>Relationship</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F$</td>
<td>$\Sigma$</td>
<td>$F = U\Sigma V^T$</td>
<td>diagonal</td>
</tr>
<tr>
<td>$P = \frac{\partial \psi}{\partial F}$</td>
<td>$\hat{P}$</td>
<td>$P = U\hat{P}V^T$</td>
<td>diagonal</td>
</tr>
<tr>
<td>$T = \frac{\partial P}{\partial F}$</td>
<td>$\hat{T}$</td>
<td>$T_{ijkl} = U_{im} U_{kr} \hat{T}<em>{mnrs} V</em>{jn} V_{ls}$</td>
<td>sparse</td>
</tr>
</tbody>
</table>
Strategy

- Compute diagonal space quantity
  - \( \hat{\mathbf{P}}, \hat{T} \)
Strategy

- Compute diagonal space quantity
  - \( \hat{P}, \hat{T} \)
- Transform to original
  - \( P, T \)
Formula for $\hat{P}$

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

Notes:
Formula for $\hat{P}$

$$\hat{P}_{ii} = \dot{\psi},_i$$

Notes:

- no summation implied
Formula for $\hat{P}$

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

Notes:

- no summation implied
- $\hat{\psi}_{,i} = \frac{\partial \hat{\psi}}{\partial \sigma_i}$
Formula for $\hat{P}$

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

Notes:
- no summation implied
- $\hat{\psi}_{,i} = \frac{\partial \hat{\psi}}{\partial \sigma_{i}}$
- $\hat{P}$ is diagonal
Formula for $\hat{T}$

Hessian term:

$$\hat{T}_{iik} = \hat{\psi}_{ik}$$
Formula for $\hat{T}$

Hessian term:

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

Cross terms ($i \neq k$):

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

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

Hessian term:

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

Cross terms ($i \neq k$):

$$a_{ik} = \frac{\hat{\psi}_{i} - \hat{\psi}_{k}}{\sigma_{i} - \sigma_{k}}$$

$$b_{ik} = \frac{\hat{\psi}_{i} + \hat{\psi}_{k}}{\sigma_{i} + \sigma_{k}}$$

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

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

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

$$a_{ik} = \frac{\hat{\psi}_i - \hat{\psi}_k}{\sigma_i - \sigma_k}$$

Notes:

- $\hat{\psi}$ symmetric in $\sigma_k$
- $\sigma_i \rightarrow \sigma_k$ implies $\hat{\psi}_i \rightarrow \hat{\psi}_k$
- 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
Matrices that diagonalize as

- $A = U\hat{A}V^T$ (generalizes $P$ rule)
- $A = U\hat{A}U^T$
- $A = V\hat{A}V^T$

**Eigenvalue decomposition**

- $S = U\Lambda U^T$
- $S$ 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

```c
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**: $u'$
- **Diff_MT**: $A'$
- **Diff_TV**: $\frac{\partial f}{\partial x}$
- **Diff_VV**: $\frac{\partial u}{\partial x}$
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.dx + 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: $f(x)$
Does not scale well

- Forward mode
- Scales poorly for many inputs
  - optimization: \( f(x) \)
  - force: \( \phi(x) \)
Does not scale well

- Forward mode
- Scales poorly for many inputs
  - optimization: $f(x)$
  - force: $\phi(x)$
  - stress: $\psi(F)$
Reverse mode is tough
Compute derivatives in reverse order
Need to record the code
Result of:  \( z = 3x^2 + \cos y \)

Has type:
\[
\text{Add} < \text{Scale} < \text{Square} < \text{Var} < 0 >>>, \text{Cos} < \text{Var} < 1 >>>
\]

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
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× 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?