Introduction and Syllabus for 18.S096: Matrix Calculus

IAP 2023

Profs. Alan Edelman & Steven Johnson
MWF 11am–1pm in 2-190

(MIT students, comments, fixes, request for further clarifications, welcome. Please keep mathematical)
● Lectures: Jan. 18, 20, 23, 25, 27 + Feb. 1, 3
  ○ 11am–1pm in 2-142, short break around noon
● Two Psets: Released Wednesday due following Wednesday (Jan 19 & 26) @ midnight on Canvas
● 3 Units
● Prerequisite: Linear Algebra (18.06 or similar)

Some demos and hw may use julia. (minimal programming experience assumed, though most LinAlg classes at MIT use a little Julia already)
Where does matrix calculus fit in?

- MIT 18.01: Scalar or Single Variable Calculus
- MIT 18.02: Vector or Multivariable Calculus

Perhaps an ideal world might go Scalar, Vector, Matrix, Higher Dimensional Arrays…
(0 dimensional, 1 dimensional, 2 dimensional…)
(e.g. size(scalar)=[], size(vector)=[n], size(matrix)=[m,n],...)
(some programming language do not implement this fully)

Why now?

- In the last decade or two, the role of linear algebra has taken on larger importance in lots of areas including Machine Learning, Statistics, Engineering, etc.
- Warning: googling Matrix Calculus may only give a small view of the full range of the mathematics that we hope to cover example what is the derivative of $X^2$ when $X$ is a square matrix? Should it be $2X$? (It’s not). What about $X^{-1}$? $-X^{-2}$? (Not quite).
Applications: Machine Learning
buzzwords: parameter optimization
stochastic gradient descent, autodiff, backpropagation

Matrix Calculus for Machine Learning

As Machine Learning deals with data in higher dimensions, understanding algorithms with knowledge of one and two variable calculus is cumbersome and slow. If someone asks for the derivative of \( f(x) \), without a second you will tell its \( 2x \), without using the first principles-definition of differentiability. Here, I will provide some tips and tricks to perform matrix calculus just like the differentiation of \( x^2 \).

Notes on Matrix Calculus for Deep Learning

Deep learning is an exciting field that is having a great real-world impact. This article is a collection of notes based on 'The Matrix Calculus You Need For Deep Learning' by Terence Parr and Jeremy Howard.

A matrix calculus problem in backpropagation encountered when studying Deep Learning
Applications: Physical Problems

Topology-optimized aircraft wing

~ 10^9 parameters

Goal: maximize stiffness under external loads, utilizing limited amount of material
→ Light but strong
→ 100s tonnes of fuel saving

© Springer Nature Limited. All rights reserved. This content is excluded from our Creative Commons license. For more information, see https://ocw.mit.edu/help/faq-fair-use.

Topology optimization with fluid dynamics

Switching flow channels for high vs. low viscosity

© John Wiley & Sons, Inc. All rights reserved. This content is excluded from our Creative Commons license. For more information, see https://ocw.mit.edu/help/faq-fair-use.

© Altair Engineering Inc. All rights reserved. This content is excluded from our Creative Commons license. For more information, see https://ocw.mit.edu/help/faq-fair-use.

© General Motors. All rights reserved. This content is excluded from our Creative Commons license. For more information, see https://ocw.mit.edu/help/faq-fair-use.

© Springer Nature Limited. All rights reserved. This content is excluded from our Creative Commons license. For more information, see https://ocw.mit.edu/help/faq-fair-use.

© General Motors. All rights reserved. This content is excluded from our Creative Commons license. For more information, see https://ocw.mit.edu/help/faq-fair-use.

© Alternative Engine Inc. All rights reserved. This content is excluded from our Creative Commons license. For more information, see https://ocw.mit.edu/help/faq-fair-use.

© Springer Nature Limited. All rights reserved. This content is excluded from our Creative Commons license. For more information, see https://ocw.mit.edu/help/faq-fair-use.

© Springer Nature Limited. All rights reserved. This content is excluded from our Creative Commons license. For more information, see https://ocw.mit.edu/help/faq-fair-use.

Engineering optimization (structural “topology” optimization):

Find the physical structure that optimizes some objective (e.g. focusing light, minimizing drag, supporting weight, ...).

Key point is that if you have any complicated calculation with lots of parameters, you can compute gradient (sensitivity) of a scalar output g(u) with respect to every parameter with roughly one additional calculation.

Enabling factor for large-scale optimization in machine learning [g = loss function, u = network outputs, p = network weights & other parameters], statistics, finance, and many other fields.
Applications: Data Science & Multivariable Statistics

\[
\frac{d}{dx} kx = A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}
\]

\[
\frac{d}{dx} kx^2 =
\]

Derivative of a Matrix: Data Science Basics

© ritvikmath. All rights reserved. This content is excluded from our Creative Commons license. For more information, see https://ocw.mit.edu/help/faq-fair-use.

© John Wiley & Sons, Inc. All rights reserved. This content is excluded from our Creative Commons license. For more information, see https://ocw.mit.edu/help/faq-fair-use.
The Role of Automatic Differentiation

Typical differential calculus classes are mostly symbolic calculus:

- Students learn to do what mathematica/wolfram alpha readily can do

For a small portion of the class, some numerics may show up

- approximate $f'(x)$ by finite differences $(f(x+\epsilon)-f(x)) / \epsilon$ or $(f(x+\epsilon)-f(x-\epsilon)) / 2\epsilon$
- e.g. students and professors think that “sin” is actually computed using Taylor series

Today’s automatic differentiation is neither of these two things. It is more in the field of the computer science topic of compiler technology than mathematics.

However the underlying mathematics is interesting! We will learn about this in this class.
Everything is easy with scalar functions of scalars

- The derivative of a function of one variable is a function of one variable
- The linearization of a function has the form \((y-y_0) \approx f'(x_0)(x-x_0)\)

Other notations (sometimes confusing \(x\) and \(x_0\)):

- \(\delta y \approx f'(x) \delta x\)
- \(\text{dy} = f'(x)\text{dx}\)
- \(f(x)-f(x_0) \approx f'(x_0)(x-x_0)\)
- \(\text{df} = f'(x)\text{dx} \leftarrow \text{this one is preferred here}\)

- Numerics are fairly trivial:
Numerical Example of what’s going on in the previous slide

Suppose \( f(x) = x^2 \) with \((x_0,y_0)=(3,9)\) and \( f'(x_0)=6 \)

- \( f(3.0001) = 9.00060001 \)
- \( f(3.00001) = 9.0000600001 \)
- \( f(3.000001) = 9.000006000001 \)
- \( f(3.0000001) = 9.00000060000001 \) (Notice that \( \Delta y = 6 \Delta x \))

\[ f(3 + \Delta x) \approx 9 + \Delta y = 9 + 6 \Delta x \quad (\Delta y = f'(x_0) \Delta x) \]

\[ f(x) - f(3) \approx 6(x-3) \quad \text{← linearization of } x^2 \text{ at } x=3 \text{ is the “multiply by } 6” \text{ function} \]

We write:

\[ dy = f(x_0+dx) - f(x_0) \quad \text{where } dy = f'(x_0)dx \text{ or } f(x_0+dx) = f(x_0) + f'(x_0)dx \]

I think of \( dx \) and \( dy \) as really small numbers; in math they are called \( \text{infinitesimal} \).

In rigorous mathematics, one takes limits.
Demo

http://www.matrixcalculus.org/

Notation: Elementwise vector or matrix product. We will use \( x.*y \), they use \( x \odot y \)

- \([2,3].*[10,11] = [20,33]\)
- \( \text{trace}(A) = \text{tr}(A) = \) the sum (a scalar) of the diagonal elements of matrix \( A \)
- Some limitations:
  - matrixcalculus.org will not display derivatives that involve more than 2 dimensions:
    - e.g. a derivative of a matrix with respect to a vector or a matrix

If we differentiate a scalar function of a matrix

Answer is a matrix: \(-2 \times' (Y-X\theta)\)

We will teach you to solve problems like this!

Courtesy of MIT Mathematics Department.
# Format of the first derivative, explicit notation:

<table>
<thead>
<tr>
<th>Input ↓ \ Output →</th>
<th>Scalar</th>
<th>Vector</th>
<th>Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scalar</td>
<td>Scalar</td>
<td>Vector (e.g. velocity)</td>
<td>Matrix</td>
</tr>
<tr>
<td>Vector</td>
<td>Gradient = vector (or column vector) Notation: ( \nabla f ) ( f'(x) = ) row vector ( df = f'(x)dx )</td>
<td>Matrix (Jacobian matrix)</td>
<td>Higher order array</td>
</tr>
<tr>
<td>Matrix</td>
<td>Matrix</td>
<td>Higher order array</td>
<td>Higher order array</td>
</tr>
</tbody>
</table>

# Format of the first derivative, implicit view: linear operator

- \( d(x^3) = 3x^2 \ dx \) scalar in, scalar out (multiply the infinitesimal scalar \( dx \) by \( 3x^2 \))
- \( d(x^T x) = 2x^T dx \) scalar in, vector out (take the dot product of the infinitesimal vector \( dx \) with the vector \( 2x \))
- \( d(X^2) = XdX + dX X \) matrix in, matrix out (multiply the infinitesimal matrix \( dX \) by matrix \( X \) on each side and add)

You will learn to do all of these in great detail – the purpose of this slide is just to plant the notion of linearization.
Let's check the linearization numerically

\[ f(x) = x^T x \]

\[ x_0 = [3; 4] \Rightarrow f(x_0) = x_0^T x_0 = 25 \]

\[ dx = [0.001; 0.002] \Rightarrow (3.001)^2 + (4.002)^2 = 25.022005 \]

\[ 2x_0^T dx = 2 [3; 4]^T [0.001; 0.002] = 0.022 \]

Notice that \( f(x_0 + dx) \approx f(x_0) + 2x_0^T dx = 25 + 0.022 \)
Matrix and vector product rule

$d(AB) = (dA)B + A(dB)$ is still correct but generally the products do not commute

However if $x$ is a vector:

$d(x^Tx) = dx^Tx + x^Tdx$ and since vector dot products commute (a dot b is b dot a), we in this special case can write $d(x^Tx) = (2x)^Tdx$.

Example: $x=[1;2;3;4]$; $dx$=rand(4)/100000;

$(x+dx)'(x+dx) - x'x$ # this is $d(x'x)$

$(2x)^Tdx$ # this is approximately the same as $d(x'x)$

Note: the way the product rule works for vectors and matrices is that transposes “go for the ride”

Examples:

1. $d(u^Tv) = du^Tv + u^Td v$ but note $du^Tv = v^Tdu$ because dot products commute
2. $d(uv^T)=$duv$^T + udv^T$
For the explicit form we want derivatives of all outputs w.r.t. to all inputs.

How many parameters are needed? If there are n inputs and m outputs

Answer:
Second derivatives (a few words for starters)

Explicit form: The second derivative of a scalar valued function of a vector is represented explicitly as a symmetric matrix known as the Hessian of the function.

Implicit form: All second derivatives are what is known in advanced linear algebra as a quadratic form (or a symmetric bilinear form).