Explanatory Notes for 6.390

Shaunticlair Ruiz (Current TA)

Fall 2022

7.X.16 Other Derivatives

After these, you might ask yourself, what about other derivative combinations?

$$\frac{\partial v}{\partial M}? \qquad \frac{\partial M}{\partial v}? \qquad \frac{\partial M}{\partial M^2}? \tag{1}$$

There's a problem with all of these: the total number of axes is **too large**.

What do we mean by an **axis**?

Definition 1

An **axis** is one of the **indices** we can adjust to get a different scalar in our array: each index is a "direction" we can move along our object to **store** numbers.

- A **scalar** has **0 axes**: we only have one scalar, so we have no indices to adjust.

- A **vector** has **1 axis**: we can get different scalars by moving **vertically** (for column vectors): v_1, v_2, v_3...

$$\left.\begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{bmatrix}\right\} \text{Axis 1}$$

- A **matrix** has **2 axes**: we can move **horizontally** or **vertically**.

$$\overbrace{\begin{bmatrix} m_{11} & m_{12} & \cdots & m_{1r} \\ m_{21} & m_{22} & \cdots & m_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ m_{k1} & m_{k2} & \cdots & m_{kr} \end{bmatrix}}^{\text{Axis 2: Columns}} \left.\vphantom{\begin{bmatrix} m \\ m \\ \vdots \\ m \end{bmatrix}}\right\} \text{Axis 1: Rows}$$

These can also be called **dimensions**.

Why does the number of **axes** matter? Remember that, so far, for our derivatives, each axis of the output represented an axis of the **input** or **output**.

Note that last bit: we're saying a vector has one dimension. Can't a vector have **multiple** dimensions? Jump to 7.X.17 for a clarification.

$$\frac{\partial w}{\partial v} = \begin{array}{c} \overbrace{\hphantom{\frac{\partial w_1}{\partial v_1} \quad \frac{\partial w_2}{\partial v_1} \quad \cdots \quad \frac{\partial w_n}{\partial v_1}}}^{\text{Column j: vertical axis of } w} \\ \left.\begin{bmatrix} \frac{\partial w_1}{\partial v_1} & \frac{\partial w_2}{\partial v_1} & \cdots & \frac{\partial w_n}{\partial v_1} \\ \frac{\partial w_1}{\partial v_2} & \frac{\partial w_2}{\partial v_2} & \cdots & \frac{\partial w_n}{\partial v_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial w_1}{\partial v_m} & \frac{\partial w_2}{\partial v_m} & \cdots & \frac{\partial w_n}{\partial v_m} \end{bmatrix}\right\} \text{Row i: vertical axis of } v \end{array}$$

The way we currently build derivatives, we try to get **every pair** of input-output variables: we use **one** axis for each **axis** of either the **input** or **output**.

Take some examples:

- $\partial s/\partial v$: we need one axis to represent each term v_i.
  - 0 axis + 1 axis $\rightarrow$ 1 axis: the output is a (column) **vector**.
- $\partial v/\partial s$: we need one axis to represent each term w_j.
  - 1 axis + 0 axis $\rightarrow$ 1 axis: the output is a (row) **vector**.
- $\partial w/\partial v$: we need one axis to represent each term v_i, and another to represent each term w_j.
  - 1 axis + 1 axis $\rightarrow$ 2 axes: the output is a **matrix**.
- $\partial M/\partial s$: we need one axis to represent the rows of M, and another to represent the columns of M.
  - 2 axis + 0 axis $\rightarrow$ 2 axes: the output is a **matrix**.
- $\partial s/\partial M$: we need one axis to represent the rows of M, and another to represent the columns of M.
  - 0 axis + 2 axis $\rightarrow$ 2 axes: the output is a **matrix**.

Notice the pattern!

Concept 2

A **matrix derivative** needs to be able to account for each type/**index** of variable in the input **and** the output.

So, if the **input** x has m axes, and the **output** y has n axes, then the derivative needs to have the same **total** number:

$$\text{Axes}\left(\frac{\partial y}{\partial x}\right) = \text{Axes}(y) + \text{Axes}(x) \tag{2}$$

This is where our problem comes in: if we have a vector and a matrix, we need **3 axes**! That's more than a matrix.

7.X.17 Dimensions (Optional)

Here's a quick aside to clear up possible confusion from the last section: our definition of axes and "dimensions".

We said a vector has 1 axis, or "dimension" of movement. But, can't a vector have **multiple** dimensions?

Clarification 3

We have two competing definition of **dimension**: this explains why we can say seemingly conflicting things about derivatives.

So far, by "**dimension**", we mean, "a separate **value** we can **adjust**".

- Under this definition, a $(k \times 1)$ column **vector** has k dimensions: it contains k different scalars we can **adjust**.

$$\left.\begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_k \end{bmatrix}\right\} \text{We can adjust each of our } k \text{ scalars.}$$

- You might say a $(k \times r)$ **matrix** has k dimensions, too: based on the **dimensionality** of its column vectors.
  - Since we prioritize the size of the vectors, we could say this is a very "vector-centric" definition.

In this section, by "dimension", we mean, "an **index** we can **adjust** (move along) to find another scalar.

- Under this definition, a $(k \times 1)$ column **vector** has 1 dimension: we only have 1 axis of **movement**.
- You might say a $(k \times r)$ **matrix** has 2 dimensions: a **horizontal** one, and a **vertical** one.
  - This **definition** is the kind we use in the following sections.

If you jumped here from 7.X.16, feel free to follow this link back. Otherwise, continue on.

7.X.18 Dealing with Tensors

If a vector looks like a "**line**" of numbers, and a matrix looks like a "**rectangle**" of numbers, then a **3-axis** version would look like a "**box**" of numbers. How do we make sense of this?

First, what is this kind of object we've been working with? Vectors, matrices, etc. This collection of numbers, organized neatly, is an **array**.

Definition 4

An **array** of objects is an **ordered sequence** of them, stored together.

The most typical example is a **vector**: an ordered sequence of **scalars**.

A **matrix** can be thought of as a **vector** of **vectors**. For example: it could be a row vector, where every column is a column vector.

So, we think of a matrix as a "two-dimensional array".

We can extend this to any number of dimensions. We call this kind of generalization a **tensor**.

Definition 5

In **machine learning**, we think of a **tensor** as a "**multidimensional array**" of numbers.

Each "dimension" is what we have been calling an "**axis**".

A tensor with c axes is called a **c-Tensor**.

Note that what we call a tensor is **not** a mathematical (or physics) tensor: we do not often use the "tensor product", or other tensor properties.

Our tensor can be better thought of as a "**generalized matrix**".

Example: The 3-D box we are talking about above is called a 3-Tensor. We can simply think of it as a stack of matrices.

How do we handle **tensors**? Simply, we convert them into regular **matrices** in some way, and then do our usual math on them:

These examples aren't especially important, but you'll see different variations in different softwares!

- If a tensor has a pattern of zeroes, we might be able to flatten it into a matrix.
  - For example, if we wanted to flatten a matrix into a vector (which we sometimes do!), we could do

$$\begin{bmatrix} 3 & 0 & 0 \\ 0 & 9 & 0 \\ 0 & 0 & 4 \end{bmatrix} \rightarrow \begin{bmatrix} 3 \\ 9 \\ 4 \end{bmatrix} \tag{3}$$

- We can also flatten it into a matrix or vector by placing the layers next to each other.
- We cleverly do regular matrix multiplication in a way that's compatible with our tensors.
  - Note that tensors do not have a matrix multiplication-like multiplication by default: several have been designed, however.

- We ignore the structure of the tensor, and just look at the individual elements: we take the scalar chain rule for each of them, without respecting the overall tensor.

Clarification 6

If you look into **derivatives** that would result in a **3-tensor** or higher, you'll find that there's no consistent **notation** for what these derivatives look like.

These techniques are part of why: there are **different** approaches for how to approach these objects.

As we will see in the next chapter, tensors are **very** important to machine learning.

However, because they don't have a natural matrix multiplication, we'll try to convert it into a matrix in most cases.