# Intro

- In this video we will take a look at other cases of matrix derivatives

# Jacobian

- Let's take a composition of two vector functions:

$$g(x): (x_1, \dots, x_n) \to (g_1, \dots, g_m)$$

$$f(g): (g_1, \dots, g_m) \to (f_1, \dots, f_k)$$ *this will be useful for RNN*

$$h(x) = f(g(x)): (x_1, \dots, x_n) \to (h_1, \dots, h_k) = (f_1, \dots, f_k)$$

- The matrix of partial derivatives $\frac{\partial h_i}{\partial x_j}$ is called the Jacobian:

$$J^h = \begin{pmatrix} \frac{\partial h_1}{\partial x_1} & \cdots & \frac{\partial h_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_k}{\partial x_1} & \cdots & \frac{\partial h_k}{\partial x_n} \end{pmatrix}$$

# Jacobian

- Let's take a composition of two vector functions:

$$g(x): (x_1, \ldots, x_n) \to (g_1, \ldots, g_m)$$

$$f(g): (g_1, \ldots, g_m) \to (f_1, \ldots, f_k)$$

*this will be useful for RNN*

$$h(x) = f(g(x)): (x_1, \ldots, x_n) \to (h_1, \ldots, h_k) = (f_1, \ldots, f_k)$$

- The matrix of partial derivatives $\frac{\partial h_i}{\partial x_j}$ is called the Jacobian:

$$J^h = \begin{pmatrix} \frac{\partial h_1}{\partial x_1} & \cdots & \frac{\partial h_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_k}{\partial x_1} & \cdots & \frac{\partial h_k}{\partial x_n} \end{pmatrix}$$

$$J^h_{i,j} = \frac{\partial h_i}{\partial x_j} =$$

$$\sum_l \frac{\partial f_i}{\partial g_l} \frac{\partial g_l}{\partial x_j} = \sum_l J^f_{i,l} \cdot J^g_{l,j}$$

**Chain rule**: $J^h = J^f \cdot J^g$

# Matrix by matrix derivative

- Matrix product:

$$C = AB = \begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix} \cdot \begin{pmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{pmatrix} =$$

$$\begin{pmatrix} a_{1,1}b_{1,1} + a_{1,2}b_{2,1} & a_{1,1}b_{1,2} + a_{1,2}b_{2,2} \\ a_{2,1}b_{1,1} + a_{2,2}b_{2,1} & a_{2,1}b_{1,2} + a_{2,2}b_{2,2} \end{pmatrix}$$

- How to calculate the derivative $\frac{\partial C}{\partial A}$ ?

$$\frac{\partial C}{\partial A} = \left\{ \frac{\partial c_{i,j}}{\partial a_{k,l}} \right\}_{i,j,k,l}$$

This is a **tensor** (high-dimensional array)!

# Chain rule

$$C = AB = \begin{pmatrix} a_{1,1}b_{1,1} + a_{1,2}b_{2,1} & a_{1,1}b_{1,2} + a_{1,2}b_{2,2} \\ a_{2,1}b_{1,1} + a_{2,2}b_{2,1} & a_{2,1}b_{1,2} + a_{2,2}b_{2,2} \end{pmatrix}$$

$$\frac{\partial C}{\partial A} = \begin{pmatrix} \frac{\partial c_{1,1}}{\partial A} & \frac{\partial c_{1,2}}{\partial A} \\ \frac{\partial c_{2,1}}{\partial A} & \frac{\partial c_{2,2}}{\partial A} \end{pmatrix} = \begin{pmatrix} \begin{bmatrix} b_{1,1} & b_{2,1} \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} b_{1,2} & b_{2,2} \\ 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 0 & 0 \\ b_{1,1} & b_{2,1} \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ b_{1,2} & b_{2,2} \end{bmatrix} \end{pmatrix}$$

Possible notation

- Let's say we have $\frac{\partial L}{\partial C} = \begin{bmatrix} \alpha & \beta \\ \gamma & \eta \end{bmatrix}$

$$\frac{\partial L}{\partial A} = \sum_{i,j} \frac{\partial L}{\partial c_{i,j}} \frac{\partial c_{i,j}}{\partial A} = \alpha \frac{\partial c_{1,1}}{\partial A} + \beta \frac{\partial c_{1,2}}{\partial A} + \gamma \frac{\partial c_{2,1}}{\partial A} + \eta \frac{\partial c_{2,2}}{\partial A} =$$

$$\begin{bmatrix} \alpha b_{1,1} + \beta b_{1,2} & \alpha b_{2,1} + \beta b_{2,2} \\ \gamma b_{1,1} + \eta b_{1,2} & \gamma b_{2,1} + \eta b_{2,2} \end{bmatrix}$$

# What's bad about it

- Our chain rule is a **linear combination** of matrices:

$$\frac{\partial L}{\partial A} = \sum_{i,j} \frac{\partial L}{\partial c_{i,j}} \frac{\partial c_{i,j}}{\partial A} = \alpha \frac{\partial c_{1,1}}{\partial A} + \beta \frac{\partial c_{1,2}}{\partial A} + \gamma \frac{\partial c_{2,1}}{\partial A} + \eta \frac{\partial c_{2,2}}{\partial A} =$$

$$\begin{bmatrix} \alpha b_{1,1} + \beta b_{1,2} & \alpha b_{2,1} + \beta b_{2,2} \\ \gamma b_{1,1} + \eta b_{1,2} & \gamma b_{2,1} + \eta b_{2,2} \end{bmatrix}$$

- **Crunching a lot of zeros:**

$$\frac{\partial C}{\partial A} = \begin{pmatrix} \frac{\partial c_{1,1}}{\partial A} & \frac{\partial c_{1,2}}{\partial A} \\ \frac{\partial c_{2,1}}{\partial A} & \frac{\partial c_{2,2}}{\partial A} \end{pmatrix} = \begin{pmatrix} \begin{bmatrix} b_{1,1} & b_{2,1} \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} b_{1,2} & b_{2,2} \\ 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 0 & 0 \\ b_{1,1} & b_{2,1} \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ b_{1,2} & b_{2,2} \end{bmatrix} \end{pmatrix}$$

# What's bad about it

- Our chain rule is a **linear combination** of matrices:

$$\frac{\partial L}{\partial A} = \sum_{i,j} \frac{\partial L}{\partial c_{i,j}} \frac{\partial c_{i,j}}{\partial A} = \alpha \frac{\partial c_{1,1}}{\partial A} + \beta \frac{\partial c_{1,2}}{\partial A} + \gamma \frac{\partial c_{2,1}}{\partial A} + \eta \frac{\partial c_{2,2}}{\partial A} =$$

$$\begin{bmatrix} \alpha b_{1,1} + \beta b_{1,2} & \alpha b_{2,1} + \beta b_{2,2} \\ \gamma b_{1,1} + \eta b_{1,2} & \gamma b_{2,1} + \eta b_{2,2} \end{bmatrix}$$

- **Performance issues**
  - **e.g. no such operation in BLAS**

# What's bad about it

- Our chain rule is a **linear combination** of matrices:

$$\frac{\partial L}{\partial A} = \sum_{i,j} \frac{\partial L}{\partial c_{i,j}} \frac{\partial c_{i,j}}{\partial A} = \alpha \frac{\partial c_{1,1}}{\partial A} + \beta \frac{\partial c_{1,2}}{\partial A} + \gamma \frac{\partial c_{2,1}}{\partial A} + \eta \frac{\partial c_{2,2}}{\partial A} =$$

$$\begin{bmatrix} \alpha b_{1,1} + \beta b_{1,2} & \alpha b_{2,1} + \beta b_{2,2} \\ \gamma b_{1,1} + \eta b_{1,2} & \gamma b_{2,1} + \eta b_{2,2} \end{bmatrix}$$

- **We know how to compute $\frac{\partial L}{\partial A}$ skipping $\frac{\partial C}{\partial A}$ tensor computation:**

$$\frac{\partial L}{\partial C} = \begin{bmatrix} \alpha & \beta \\ \gamma & \eta \end{bmatrix} \qquad B^T = \begin{pmatrix} b_{1,1} & b_{2,1} \\ b_{1,2} & b_{2,2} \end{pmatrix}$$

From MLP lecture: $\frac{\partial L}{\partial A} = \frac{\partial L}{\partial C} B^T = \begin{bmatrix} \alpha & \beta \\ \gamma & \eta \end{bmatrix} \begin{pmatrix} b_{1,1} & b_{2,1} \\ b_{1,2} & b_{2,2} \end{pmatrix}$

# We're still fine

- Thankfully, in deep learning frameworks we need to calculate gradients of a **scalar loss** with respect to another **scalar, vector, matrix or tensor**.

- This's how tf.gradients() in TensorFlow works.

- Deep learning frameworks have **optimized** versions of backward pass for standard layers.

# Summary

- For vector functions chain rule says that you need to multiply respective **Jacobians**.
- Matrix by matrix derivative is a **tensor**
- Chain rule for such tensors is **not very useful** in practice
- Thankfully, in deep learning frameworks we usually need to track gradients of a **scalar loss** with respect to all other parameters.