The Tensor Cookbook

Thomas Dybdahl Ahle

May 28, 2024

Chapter 1

Introduction

What is this? These pages are a guide to tensors, using the visual language of "tensor diagrams". For illustrating the generality of the approach, I've tried to closely follow the legendary "Matrix Cookbook". As such, most of the presentation is a collection of facts (identities, approximations, inequalities, relations, ...) about tensors and matters relating to them. You won't find many results not in the original cookbook, but hopefully the diagrams will give you a new way to understand and appreciate them.

It's ongoing: The Matrix Cookbook is a long book, and not all the sections are equally amendable to diagrams. Hence I've opted to skip certain sections and shorten others. Perhaps in the future, I, or others, will expand the coverage further.

For example, while we cover all of the results on Expectation of Linear Combinations and Gaussian moments, we skip the section on general multi-variate distributions. I have also had to rearrange the material a bit, to avoid having to introduce all the notation up front.

Complex Matrices and Covariance Tensor diagrams (or networks) are currently most often seen in Quantum Physics. Here most values are complex numbers, which introduce some extra complexity. In particular transposing a matrix now involves taking the conjugate (flipping the sign of the imaginary part), which introduces the need for co- and contra-variant tensors. None of this complexity is present with standard real valued matrices, as is common e.g. in Machine Learning applications. For simplicity I have decided to not include these complexities.

Tensorgrad The symbolic nature of tensor diagrams make the well suited for symbolic computation.

Advantages of Tensor Diagram Notation: Tensor diagram notation has many benefits compared to other notations:

Various operations, such as a trace, tensor product, or tensor contraction can be expressed simply without extra notation. Names of indices and tensors can often be omitted. This saves time and lightens the notation, and is especially useful for internal indices which exist mainly to be summed over. The order of the tensor resulting from a complicated network of contractions can be determined by inspection: it is just the number of unpaired lines. For example, a tensor network with all lines joined, no matter how complicated, must result in a scalar.

Contents

1.1 Notation and Nomenclature

Dot product	$a—b$	$y = \sum_i a_i b_i$	$[\cdot\cdot\cdot\cdot]\begin{bmatrix}\cdot\\\cdot\\\cdot\end{bmatrix}$	$= y$
Outer product	$—a \quad b—$	$Y_{i,j} = a_i b_j$	$\begin{bmatrix}\cdot\\\cdot\\\cdot\end{bmatrix}[\cdot\cdot\cdot\cdot]$	$= —Y—$
Matrix-Vector	$—A–b$	$y_i = \sum_j A_{i,j} b_j$	$\begin{bmatrix}\cdot&\cdot&\cdot&\cdot\\\cdot&\cdot&\cdot&\cdot\\\cdot&\cdot&\cdot&\cdot\\\cdot&\cdot&\cdot&\cdot\end{bmatrix}\begin{bmatrix}\cdot\\\cdot\\\cdot\\\cdot\end{bmatrix}$	$= —y$
Matrix-Matrix	$—A–B—$	$Y_{i,k} = \sum_j A_{i,j} B_{j,k}$	$\begin{bmatrix}\cdot&\cdot&\cdot&\cdot\\\cdot&\cdot&\cdot&\cdot\\\cdot&\cdot&\cdot&\cdot\\\cdot&\cdot&\cdot&\cdot\end{bmatrix}\begin{bmatrix}\cdot&\cdot&\cdot&\cdot\\\cdot&\cdot&\cdot&\cdot\\\cdot&\cdot&\cdot&\cdot\\\cdot&\cdot&\cdot&\cdot\end{bmatrix}$	$= —Y—$

1.1.1 The Copy Tensor

We define $\circ—$ to be the all-ones vector. That is $\circ_i = 1$. We generalize $\circ$ to rank-n tensors by $\circ_{i,j,k,\ldots} = [i = j = k = \ldots]$. That is, the tensor with 1 on the diagonal, and 0 everywhere else. This is also known as the "copy" or "spider" tensor, or "generalized Kronecker delta". For rank-2 tensors, $—\circ— = I$, the identity matrix.

The rank-4 copy tensor, $C_{i,j,k,l} = [i = j = k = l]$, $>\!\!\circ\!\!<$, differs from the outer product of two identity matrices (in the Cookbook denoted J), which satisfies $J_{i,j,k,l} = [i = k][j = l]$ and which we'd write as $J = \substack{—\circ—\\—\circ—}$, and satisfies, for example, $\frac{dX}{dX} = J$.

1.2 Basics

$—A— = —Q—\underset{\lambda}{\circ}—Q^{-1}—$ where λ_i is the ith eigenvalue of A.

More general principles:

1. You can contract edges in any order.
2. You can always contract connected subgraphs of Copy tensors.
3. distributive law of sums/products.
4. Sums and broadcasting.

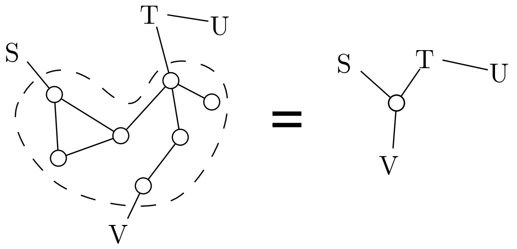


Figure 1.1: Contract spiders

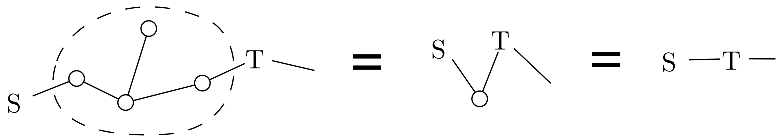


Figure 1.2: Eliminate Identity

1.2.1 Trace

The “trace” of a square matrix is defined $\mathrm{Tr}(A) = \sum_i A_{i,i}$. In tensor diagram notation, that corresponds to a self-edge: $\overgroup{A}$.

$$\sum_{i=1}^{n} A_{ii} = \mathrm{Tr}(A) = \mathrm{Tr}(AI) \qquad \overgroup{A} = \overgroup{A\text{—}\circ} \tag{11}$$

$$\mathrm{Tr}(A) = \sum_i \lambda_i = \langle 1, \lambda \rangle \qquad \overgroup{A} = \overgroup{Q\text{—}\underset{\lambda}{\circ}\text{—}Q^{-1}} \tag{12}$$

$$\mathrm{Tr}(A) = \mathrm{Tr}(A^T) \qquad \overgroup{A} = \overgroup{A} \tag{13}$$

$$\mathrm{Tr}(AB) = \mathrm{Tr}(BA) \qquad \overgroup{A\text{—}B} = \overgroup{B\text{—}A} \tag{14}$$

$$\begin{aligned} \mathrm{Tr}(A+B) = \mathrm{Tr}(A) + \mathrm{Tr}(B) \qquad \overgroup{(A+B)\text{—}\circ} &= \overgroup{A\text{—}\circ} \\ &+ \overgroup{B\text{—}\circ} \end{aligned} \tag{15}$$

$$\begin{aligned} \mathrm{Tr}(ABC) &= \mathrm{Tr}(BCA) \qquad \overgroup{A\text{—}B\text{—}C} = \overgroup{B\text{—}C\text{—}A} \\ &= \mathrm{Tr}(CAB) \qquad \qquad \qquad \quad = \overgroup{C\text{—}A\text{—}B} \end{aligned} \tag{16}$$

$$\begin{aligned} a^T a = \mathrm{Tr}(aa^T) \qquad a\text{—}a &= \mathrm{Tr}(\text{—}a \quad a\text{—}) \\ &= \overgroup{a \quad a} \end{aligned} \tag{17}$$

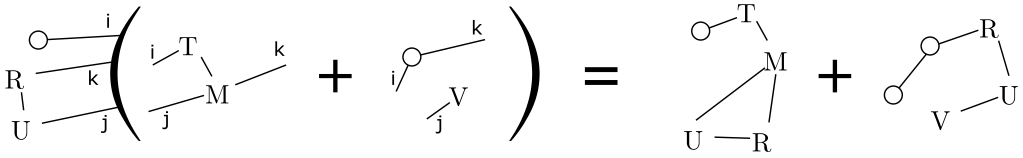


Figure 1.3: Distributive law

Chapter 2

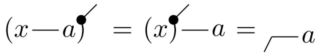
Simple Derivatives

2.1 Derivatives of Matrices, Vectors and Scalar Forms

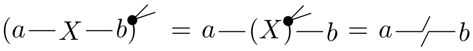
2.1.1 First Order

The Matrix Cookbook defines the single-entry matrix $J^{i,j} \in R^{n\times n}$ as the matrix which is zero everywhere except in the entry (i, j) in which it is 1. Alternatively we could write $J^{i,j}_{n,m} = [i = n][j = m]$.

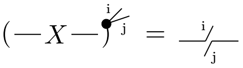
$$\frac{\partial x^T a}{\partial x} = \frac{\partial a^T x}{\partial x} = a \tag{69}$$



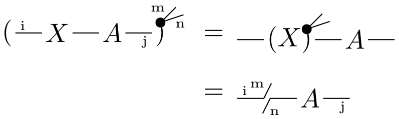
$$\frac{\partial a^T X b}{\partial X} = ab^T \tag{70}$$



$$\frac{\partial X}{\partial X_{i,j}} = J^{i,j} \tag{73}$$

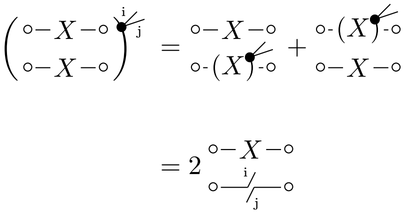


$$\frac{\partial (XA)_{i,j}}{\partial X_{m,n}} = (J^{m,n}A)_{i,j} \tag{74}$$



2.1.2 Second Order

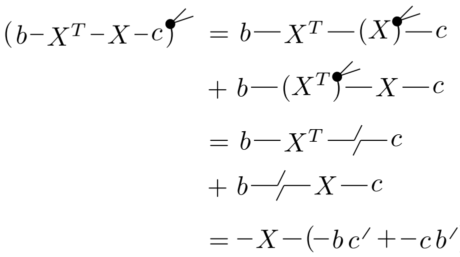
$$\frac{\partial}{\partial X_{i,j}}\sum_{k,l,m,n} X_{k,l}X_{m,n} = (\sum_{k,l} X_{k,l})^2$$



(76)

$$= 2\sum_{k,l} X_{k,l}$$

$$\frac{\partial b^T X^T X c}{\partial X} = X(bc^T + cb^T)$$

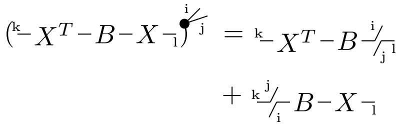


(77)

$$\frac{\partial}{\partial x}(Bx+b)^T C(Dx+d) = B^T C(Dx+d) + D^T C^T (Bx+b) \tag{78}$$

. . .

$$\frac{\partial}{\partial X_{i,j}}(X^T B X)_{k,l} = \delta_{l,j}(X^T B)_{k,i} + \delta_{k,j}(BX)_{i,l}$$

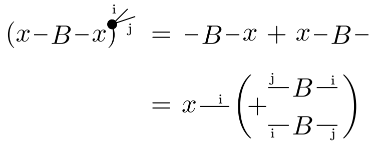


(79)

$$\frac{\partial}{\partial X_{i,j}} X^T B X = X^T B J^{i,j} + J^{j,i} B X \tag{80}$$

(same as above)

$$\frac{\partial}{\partial x} x^T B x = (B + B^T)x$$



(81)

$$\frac{\partial}{\partial X} b^T X^T D X c = D^T X b c^T + D X c b^T \tag{82}$$

. . .

$$\frac{\partial}{\partial X}(Xb+c)^T D(Xb+c) = (D+D^T)(Xb+c)b^T \tag{83}$$

. . .

Assume W is symmetric, then... (84) - (88)

2.1.3 Higher-order and non-linear

$$\frac{\partial\left(\mathbf{X}^n\right)_{kl}}{\partial X_{ij}} = \sum_{r=0}^{n-1}\left(\mathbf{X}^r \mathbf{J}^{ij} \mathbf{X}^{n-1-r}\right)_{kl}$$

For proof of the above, see B.1.3.

$$\frac{\partial}{\partial \mathbf{X}} \mathbf{a}^T \mathbf{X}^n \mathbf{b} = \sum_{r=0}^{n-1} \left(\mathbf{X}^r\right)^T \mathbf{a}\mathbf{b}^T \left(\mathbf{X}^{n-1-r}\right)^T$$

2.2 Derivatives of Traces

Assume F(X) to be a differentiable function of each of the elements of X. It then holds that

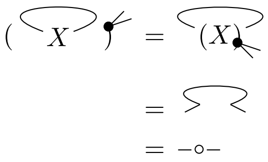
$$\frac{d\mathrm{Tr}(F(x))}{dX} = f(X)^T,$$

where $f(\cdot)$ is the scalar derivative of $F(\cdot)$.

TODO: To show this with tensor diagrams, we first need to introduce our notation for functions.

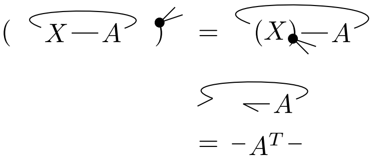
2.2.1 First Order

$$\frac{\partial}{\partial X}\mathrm{Tr}(X) = I$$



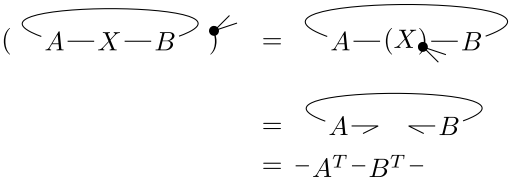
(99)

$$\frac{\partial}{\partial X}\mathrm{Tr}(XA) = A^T$$



(100)

$$\frac{\partial}{\partial X}\mathrm{Tr}(AXB) = A^T B^T$$

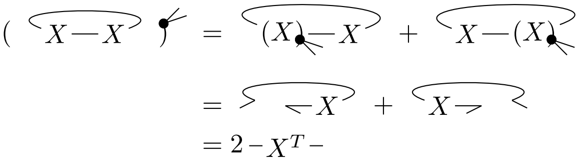


(101)

Continues for (102-105). The last one uses the Kronecker product, which we may have to introduce first.

2.2.2 Second Order

$$\frac{\partial}{\partial X}\mathrm{Tr}(X^2) = 2X^T$$



More:

$$
\begin{aligned}
&\frac{\partial}{\partial \mathbf{X}} \operatorname{Tr}\left(\mathbf{X}^2\mathbf{B}\right) = (\mathbf{XB} + \mathbf{BX})^T \\
&\frac{\partial}{\partial \mathbf{X}} \operatorname{Tr}\left(\mathbf{X}^T\mathbf{BX}\right) = \mathbf{BX} + \mathbf{B}^T\mathbf{X} \\
&\frac{\partial}{\partial \mathbf{X}} \operatorname{Tr}\left(\mathbf{BXX}^T\right) = \mathbf{BX} + \mathbf{B}^T\mathbf{X} \\
&\frac{\partial}{\partial \mathbf{X}} \operatorname{Tr}\left(\mathbf{XX}^T\mathbf{B}\right) = \mathbf{BX} + \mathbf{B}^T\mathbf{X} \\
&\frac{\partial}{\partial \mathbf{X}} \operatorname{Tr}\left(\mathbf{XBX}^T\right) = \mathbf{XB}^T + \mathbf{XB} \\
&\frac{\partial}{\partial \mathbf{X}} \operatorname{Tr}\left(\mathbf{BX}^T\mathbf{X}\right) = \mathbf{XB}^T + \mathbf{XB} \\
&\frac{\partial}{\partial \mathbf{X}} \operatorname{Tr}\left(\mathbf{X}^T\mathbf{XB}\right) = \mathbf{XB}^T + \mathbf{XB} \\
&\frac{\partial}{\partial \mathbf{X}} \operatorname{Tr}(\mathbf{AXBX}) = \mathbf{A}^T\mathbf{X}^T\mathbf{B}^T + \mathbf{B}^T\mathbf{X}^T\mathbf{A}^T \\
&\frac{\partial}{\partial \mathbf{X}} \operatorname{Tr}\left(\mathbf{X}^T\mathbf{X}\right) = \frac{\partial}{\partial \mathbf{X}} \operatorname{Tr}\left(\mathbf{XX}^T\right) = 2\mathbf{X} \\
&\frac{\partial}{\partial \mathbf{X}} \operatorname{Tr}\left(\mathbf{B}^T\mathbf{X}^T\mathbf{CXB}\right) = \mathbf{C}^T\mathbf{XBB}^T + \mathbf{CXBB}^T \\
&\frac{\partial}{\partial \mathbf{X}} \operatorname{Tr}\left[\mathbf{X}^T\mathbf{BXC}\right] = \mathbf{BXC} + \mathbf{B}^T\mathbf{XC}^T \\
&\frac{\partial}{\partial \mathbf{X}} \operatorname{Tr}\left(\mathbf{AXBX}^T\mathbf{C}\right) = \mathbf{A}^T\mathbf{C}^T\mathbf{XB}^T + \mathbf{CAXB} \\
&\frac{\partial}{\partial \mathbf{X}} \operatorname{Tr}\left[(\mathbf{AXB} + \mathbf{C})(\mathbf{AXB} + \mathbf{C})^T\right] = 2\mathbf{A}^T(\mathbf{AXB} + \mathbf{C})\mathbf{B}^T \\
&\frac{\partial}{\partial \mathbf{X}} \operatorname{Tr}(\mathbf{X} \otimes \mathbf{X}) = \frac{\partial}{\partial \mathbf{X}} \operatorname{Tr}(\mathbf{X}) \operatorname{Tr}(\mathbf{X}) = 2\operatorname{Tr}(\mathbf{X})\mathbf{I}
\end{aligned}
$$

2.2.3 Higher Order

$$
\begin{aligned}
\frac{\partial}{\partial \mathbf{X}} \operatorname{Tr}\left(\mathbf{X}^k\right) =& k\left(\mathbf{X}^{k-1}\right)^T \\
\frac{\partial}{\partial \mathbf{X}} \operatorname{Tr}\left(\mathbf{AX}^k\right) =& \sum_{r=0}^{k-1}\left(\mathbf{X}^r\mathbf{AX}^{k-r-1}\right)^T \\
\frac{\partial}{\partial \mathbf{X}} \operatorname{Tr}\left[\mathbf{B}^T\mathbf{X}^T\mathbf{CXX}^T\mathbf{CXB}\right] =& \mathbf{CXX}^T\mathbf{CXBB}^T \\
&+ \mathbf{C}^T\mathbf{XBBB}^T\mathbf{X}^T\mathbf{C}^T\mathbf{X} \\
&+ \mathbf{CXBB}^T\mathbf{X}^T\mathbf{CX} \\
&+ \mathbf{C}^T\mathbf{XX}^T\mathbf{C}^T\mathbf{XBB}
\end{aligned}
$$

2.2.4 Other

$$\frac{\partial}{\partial \mathbf{X}} \operatorname{Tr}\left(\mathbf{A}\mathbf{X}^{-1}\mathbf{B}\right) = -\left(\mathbf{X}^{-1}\mathbf{B}\mathbf{A}\mathbf{X}^{-1}\right)^T = -\mathbf{X}^{-T}\mathbf{A}^T\mathbf{B}^T\mathbf{X}^{-T}$$

Assume **B** and **C** to be symmetric, then

$$\begin{aligned}
\frac{\partial}{\partial \mathbf{X}} \operatorname{Tr}\left[\left(\mathbf{X}^T\mathbf{C}\mathbf{X}\right)^{-1}\mathbf{A}\right] &= -\left(\mathbf{C}\mathbf{X}\left(\mathbf{X}^T\mathbf{C}\mathbf{X}\right)^{-1}\right)\left(\mathbf{A}+\mathbf{A}^T\right)\left(\mathbf{X}^T\mathbf{C}\mathbf{X}\right)^{-1} \\
\frac{\partial}{\partial \mathbf{X}} \operatorname{Tr}\left[\left(\mathbf{X}^T\mathbf{C}\mathbf{X}\right)^{-1}\left(\mathbf{X}^T\mathbf{B}\mathbf{X}\right)\right] &= -2\mathbf{C}\mathbf{X}\left(\mathbf{X}^T\mathbf{C}\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{B}\mathbf{X}\left(\mathbf{X}^T\mathbf{C}\mathbf{X}\right)^{-1} \\
\frac{\partial}{\partial \mathbf{X}} \operatorname{Tr}\left[\left(\mathbf{A}+\mathbf{X}^T\mathbf{C}\mathbf{X}\right)^{-1}\left(\mathbf{X}^T\mathbf{B}\mathbf{X}\right)\right] &= -2\mathbf{B}\mathbf{X}\left(\mathbf{X}^T\mathbf{C}\mathbf{X}\right)^{-1} \\
&\quad - 2\mathbf{C}\mathbf{X}\left(\mathbf{A}+\mathbf{X}^T\mathbf{C}\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{B}\mathbf{X}\left(\mathbf{A}+\mathbf{X}^T\mathbf{C}\mathbf{X}\right)^{-1} \\
&\quad + 2\mathbf{B}\mathbf{X}\left(\mathbf{A}+\mathbf{X}^T\mathbf{C}\mathbf{X}\right)^{-1}
\end{aligned}$$

See [7].

$$\frac{\partial \operatorname{Tr}(\sin(\mathbf{X}))}{\partial \mathbf{X}} = \cos(\mathbf{X})^T$$

Chapter 3

Statistics and Probability

3.0.1 Definition of Moments

Let $x \in \mathbb{R}^n$ is a random variable. We write $m = E[x] \in \mathbb{R}^n$ for the expectation and $M = \mathrm{Var}[x] = E[(x-m)(x-m)^T]$ for the covariance (when these quantities are defined.)

In tensor diagrams, we will use square brackets:

$$m = [\text{—}x] \quad \text{and} \quad M = [\text{—}(x \ominus m) \ \ (x \div m)\text{—}]$$

Note we used the German minus, $\div$, to distinguish subtraction from contraction edges.

We can also define the third and fourth centralized moment tensors

$$M_3 = \begin{bmatrix}(x \div m)\text{—}\\(x \div m)\text{—}\\(x \div m)\text{—}\end{bmatrix} \quad \text{and} \quad M_4 = \begin{bmatrix}(x \div m)\text{—}\\(x \div m)\text{—}\\(x \div m)\text{—}\\(x \div m)\text{—}\end{bmatrix}.$$

3.0.2 Expectation of Linear Combinations

General principle: The "linearity of expectation" lets you pull out all parts of the graph not involving X.

Linear Forms

$$\mathrm{E}[AXB + C] = A\mathrm{E}[X]B + C \qquad \begin{bmatrix}-A-X-B-\\+\ -C-\end{bmatrix} = \begin{matrix}-A-[X]-B-\\+\ -C-\end{matrix} \tag{312}$$

$$\mathrm{Var}[Ax] = A\mathrm{Var}[x]A^T \qquad \begin{bmatrix}A-x \div [A-x]\\A-x \div [A-x]\end{bmatrix} = \begin{bmatrix}A-(x \div m)\\A-(x \div m)\end{bmatrix} \tag{313}$$

$$= \begin{matrix}A-\\A-\end{matrix}\!\!\begin{bmatrix}(x \div m)\\(x \div m)\end{bmatrix}$$

$$= -A-M_2-A-$$

Quadratic Forms

$$
\begin{aligned}
\mathrm{E}[x^T A x] &= \mathrm{Tr}(A\Sigma) + \mu^T A \mu \\
[x{-}A{-}x] &= [(x \div \mu){-}A{-}(x \div \mu) + \mu{-}A{-}\mu] \\
&= \begin{bmatrix}(x \div m)\\(x \div m)\end{bmatrix}\!\!> A \quad + \mu{-}A{-}\mu \\
&= \overgroup{\Sigma \text{---} A} \quad + \mu{-}A{-}\mu
\end{aligned}
$$

Cubic Forms

3.0.3 Weighted Scalar Variable

Let $y = w^T x$, and let $m = E[y]$, then

$$
\begin{aligned}
\mathrm{E}[y] &= m = w^T \mu \\
\mathrm{E}[(y-m)^2] &= w{-}M_2{-}w \\
\mathrm{E}[(y-m)^3] &= w\text{-}\overset{\overset{w}{|}}{M_3}\text{-}w \\
\mathrm{E}[(y-m)^4] &= w\text{-}\underset{\underset{w}{|}}{\overset{\overset{w}{|}}{M_4}}\text{-}w
\end{aligned}
$$

For specific distributions, like x Gaussian, we can often reduce the moment tensors further. Khintchine's inequality also gives a way to bound all of these in terms of $E[(y-m)^2]$.

3.0.4 Gaussian Moments

Mean and covariance of linear forms

Mean and variance of square forms

Cubic forms

Mean of Quartic Forms

Gaussian Integration by Parts

General principle for Gaussian expectations.

Chapter 4

Kronecker and Vec Operator

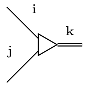
4.1 Flattening

Flattening is a common operation for programmers. In the language of numpy, we may write `np.ones((2,3,4)).reshape(2, 12)` to flatten a shape (2,3,4) tensor into a shape (2,12) matrix. Similarly, in mathematical notation, vec(X) is commonly used to denote the flattening of a matrix into a vector.

Typically the main reason to do this is as a cludge for dealing with bad general notation for tensors. Hence, with tensor diagrams, we can avoid this operation entirely. However, it is still interesting to see how tensor diagrams can make a lot of properties of flattening much more transparent.

To begin with we note that flattening is a linear operation, and hence can be represented as a simple tensor. We'll use a triangle to denote this:

$$\triangleright_{i,j,k} =$$

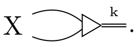


$$= [i + jn = k].$$

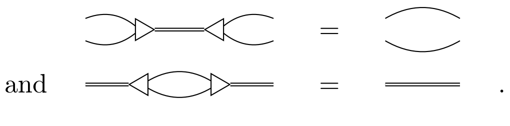
Here n is the dimension of the i edge. Note we use a double line to denote the output of the flattening operation. This is simply a syntactic choice to remind ourselves that the output is a bundle of two edges.

Using this notation we can write

$$\mathrm{vec}(X)_k = \sum_{i,j} \triangleright_{i,j,k} X_{i,j} =$$



The basic property of $\triangleright$ is that opposing triangles cancel:

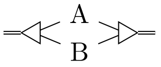


4.2 The Kronecker Product

The Kronecker product of an $m \times n$ matrix A and an $r \times q$ matrix B, is an $mr \times nq$ matrix, $A \otimes B$ defined as

$$A \otimes B = \begin{bmatrix} A_{1,1}B & A_{1,2}B & \cdots & A_{1,n}B \\ A_{2,1}B & A_{2,2}B & \cdots & A_{2,n}B \\ \vdots & \vdots & \ddots & \vdots \\ A_{m,1}B & A_{m,2}B & \cdots & A_{m,n}B \end{bmatrix}.$$

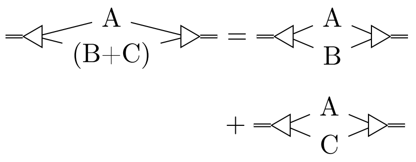
Using index notation we can also write this as $(A \otimes B)_{p(r-1)+v,q(s-1)+w} = A_{rs}B_{vw}$, but it's pretty hard to read.

In tensor notation the Kronecker Product is simply the outer product of two matrices, flattened "on both sides": $A \otimes B =$ 

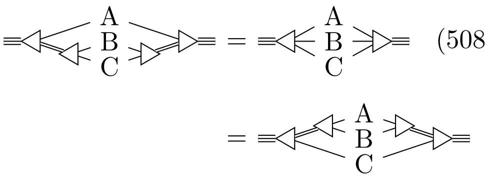
.

The Kronecker product has the following properties:

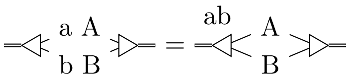
$$A \otimes (B + C) = A \otimes B + A \otimes C \tag{506}$$



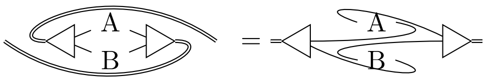
$$A \otimes (B \otimes C) = (A \otimes B) \otimes C \tag{508}$$



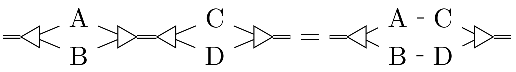
$$aA \otimes bB = ab(A \otimes B) \tag{509}$$



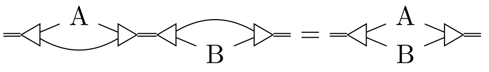
$$(A \otimes B)^T = A^T \otimes B^T \tag{510}$$



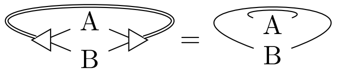
$$(A \otimes B)(C \otimes D) = AC \otimes BD \tag{511}$$

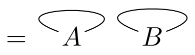


$$(A \otimes I)(I \otimes B) = A \otimes B \tag{511b}$$

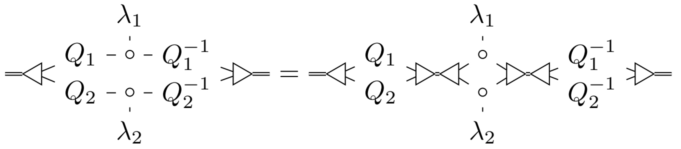


$$\text{Tr}(A \otimes B) = \text{Tr}(A)\text{Tr}(B) \tag{515}$$

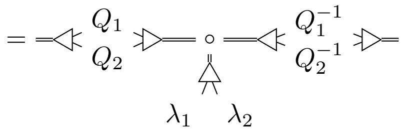




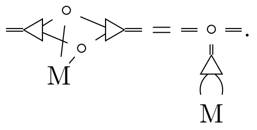
$\mathrm{eig}(A \otimes B) = \mathrm{eig}(A)\mathrm{eig}(B)$



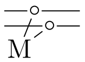
(519)



Here the last equation shows an interesting general property of ▷:



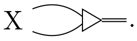
(4.1)

This is easier to see when we consider that $V =$  represents the tensor where $V_{i,j,i,j} = M_{i,j}$ and 0 otherwise. So flattening V on both sides is the same as diag(vec(M)), which is the rhs of (4.1).

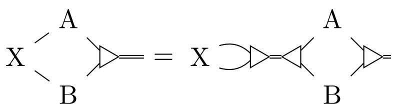
4.3 The Vec Operator

The vec-operator applied on a matrix A stacks the columns into a vector, i.e. for a 2×2 matrix

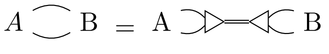
$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \qquad \mathrm{vec}(A) = \begin{bmatrix} A_{11} \\ A_{21} \\ A_{12} \\ A_{22} \end{bmatrix}$$

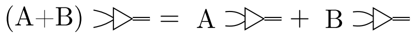
At the start of the chapter we showed how to represent the vec-operator using the flattening tensor: vec(X) = . The Matrix Cookbook gives the following properties of the vec-operator:

$\mathrm{vec}(A^T X B) = \mathrm{vec}(X)^T (A \otimes B)$

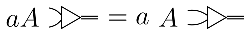


(520)

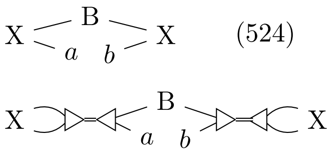
$\mathrm{Tr}(A^T B) = \mathrm{vec}(A)^T \mathrm{vec}(B)$  (521)

$\mathrm{vec}(A + B) = \mathrm{vec}(A) + \mathrm{vec}(B)$  (522)

$$\mathrm{vec}(aA) = a\,\mathrm{vec}(A) \qquad aA \quad (523)$$

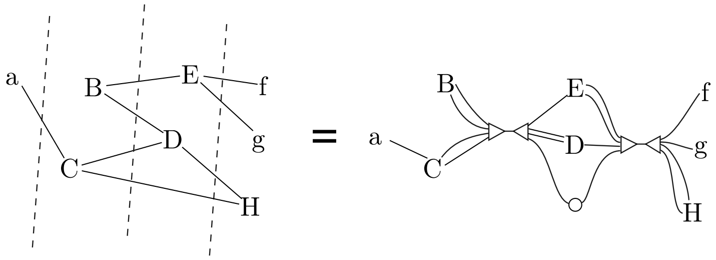


$$a^T XBX^T c = \mathrm{vec}(X)^T (B \otimes ca^T)\mathrm{vec}(X) \qquad a{-}X{-}B{-}X{-}c = \quad (524)$$



4.4 General Matrification

The last equation is an example of a general idea: Any tensor network can be transformed into a series of matrix multiplications by applying the vec-operator to all tensors and the flattening tensor to all edges. For example, the following complicated graph:



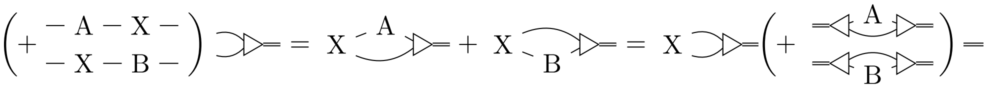
Can be written as a simple vector-matrix-matrix-vector product, aM_1M_2b, where $M_1 = \mathrm{vec}(B) \otimes C'$, $M_2 = E' \otimes D' \otimes I$ and $b = f \otimes g \otimes \mathrm{vec}(H)$, where C', D' and E' are rank 3 tensors flattened on one side, and $\mathrm{vec}(B)$ is interpreted as a matrix with a single column.

4.4.1 The Lyapunov Equation

A nice application of Kronecker product rewritings is to solve equations like

$$AX + XB = C. \tag{272}$$

We use the rewriting $\mathrm{vec}(AX + XB) = (I \otimes A + B^T \otimes I)\mathrm{vec}(X)$, which follows from the tensor diagram massaging:



after which we can take the normal matrix inverse to get

$$\mathrm{vec}(X) = (I \otimes A + B^T \otimes I)^{-1}\mathrm{vec}(C). \tag{273}$$

4.4.2 Encapsulating Sum

This is a generalization of the previous equation.

$$\sum_n A_n X B_n = C \tag{274}$$

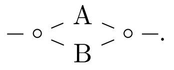
$$\mathrm{vec}(X) = (\sum_n B_n^T \otimes A_n)^{-1}\mathrm{vec}(C) \tag{275}$$

4.5 The Hadamard Product

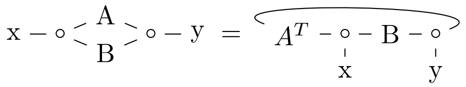
The Hadamard product, also known as element-wise multiplication, is not described in the Matrix Cookbook. Yet, it is a very useful operation, and has some interesting properties in connection with the Kronecker product.

We define the Hadamard product of two 2×2 matrices A and B as

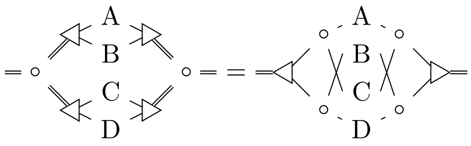
$$A \circ B = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \circ \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} = \begin{bmatrix} A_{11}B_{11} & A_{12}B_{12} \\ A_{21}B_{21} & A_{22}B_{22} \end{bmatrix}.$$

In tensor notation, the Hadamard product can be represented using two rank-3 copy tensors: . Some properties of the Hadamard product are:

$$x^T(A \circ B)y = \mathrm{tr}(A^T D_x B D_y)$$



$$(A \otimes B) \circ (C \otimes D) = (A \circ C) \otimes (B \circ D)$$



To see why the last equation holds, it suffices to follow the double lines to see that A and C both use the "upper" part of the double edge, while B and D use the lower part.

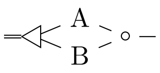
4.6 Khatri–Rao product

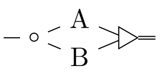
Also known as the column-wise Kronecker, row-wise Kronecker or "Face-splitting Product". We use the symbols $*$ and $\bullet$ for the column and row-wise Kronecker products, respectively.

$$A * B = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} * \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} = \begin{bmatrix} A_{11}B_{11} & A_{12}B_{12} \\ A_{11}B_{21} & A_{22}B_{22} \\ A_{21}B_{11} & A_{22}B_{12} \\ A_{21}B_{21} & A_{22}B_{22} \end{bmatrix}$$

$$A \bullet B = \dots = \begin{bmatrix} A_{11}B_{11} & A_{11}B_{12} & A_{12}B_{11} & A_{12}B_{12} \\ A_{21}B_{21} & A_{21}B_{22} & A_{22}B_{21} & A_{22}B_{22} \end{bmatrix}$$

In terms of tensor diagrams, these products correspond simply to flattening the product on one side, and using a copy tensor on the other:

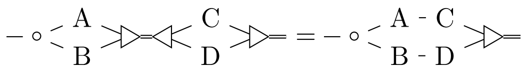
$A * B =$ 

$A \bullet B =$ 

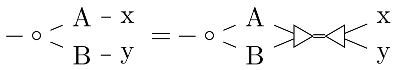
Clearly the two are identical up to transpose. Indeed, $(A*B)^T = B^T \bullet A^T$ and $(A \bullet B)^T = B^T * A^T$.

There are multiple "mixed product" identities:

$$(A \bullet B)(C \otimes D) = (AC) \bullet (BD)$$



$$(Ax) \circ (By) = (A \bullet C)(x \otimes y)$$



4.6.1 Stacking

Can be part of Kronecker section

From Josh: Proposition 2.5. For any field $\mathbb{F}$, integers d_1, d_2, d_3, d_4 and matrices $X_1 \in \mathbb{F}^{d_1 \times d_2}$, $X_2 \in \mathbb{F}^{d_2 \times d_3}$, $X_3 \in \mathbb{F}^{d_1 \times d_4}$, and $X_4 \in \mathbb{F}^{d_4 \times d_3}$, we have

$$X_1 \times X_2 + X_3 \times X_4 = (X_1 \mid X_3) \times \begin{pmatrix} X_2 \\ \hline X_4 \end{pmatrix},$$

where we are writing '|'to denote matrix concatenation.

With tensor diagrams we can write stacking along a new axis i as

$$\text{stack}_i(X,Y) = \overset{e^{(0)}\,\overline{}\,{}_i}{—X—} + \overset{e^{(1)}\,\overline{}\,{}_i}{—Y—}$$

where $e_i^{(i)} = 1$ and 0 elsewhere.

Fro this we easily get the identity

$$\begin{aligned}
(A \mid C)\begin{pmatrix} B \\ \hline D \end{pmatrix} &= \text{stack}_i(A,C)\,\text{stack}_i(B,D) && (4.2)\\
&= \left(\overset{e^{(0)}\,\overline{}\,{}_i}{{}_k—A—{}_j} + \overset{e^{(1)}\,\overline{}\,{}_i}{{}_k—C—{}_j} \right) \left(\overset{e^{(0)}\,\overline{}\,{}_i}{{}_j—B—{}_m} + \overset{e^{(1)}\,\overline{}\,{}_i}{{}_j—D—{}_m} \right) && (4.3)\\
&= \overset{e^{(0)}\,-\,-\,e^{(0)}}{{}_k—A——B—{}_m} + \overset{e^{(1)}\,-\,-\,e^{(1)}}{{}_k—C——D—{}_m} && (4.4)\\
&= AB + CD && (4.5)
\end{aligned}$$

TODO: Relation to direct sum, which is basically stacking + flattening. Also have a bunch of properties like $A \otimes (B \oplus C) = A \otimes B \oplus A \otimes C$.

Chapter 5

Functions

Function to scalar	$f : \mathbb{R}^n \to \mathbb{R}$	$f(x) \in \mathbb{R}$	$\{f \leftarrow x\}$
Function to vector	$g : \mathbb{R}^n \to \mathbb{R}^m$	$g(x) \in \mathbb{R}^m$	$\{\overset{/}{g} \leftarrow x\}$
Element-wise function	$h : \mathbb{R}^n \to \mathbb{R}$	$h(x) \in \mathbb{R}^m$	$\{h \quad \overset{/}{x}\}$
Vector times vector function	$u : \mathbb{R}^n \to \mathbb{R}^m, v \in \mathbb{R}^m$	$v^T u(x) \in \mathbb{R}$	$\overparen{v \quad \{u} \leftarrow x\}$
Vector times matrix function	$A : \mathbb{R}^n \to \mathbb{R}^{m \times n}, v \in \mathbb{R}^n$	$A(x)v \in \mathbb{R}^m$	$\{\overset{\text{m}\diagdown}{\overparen{A \leftarrow x\} \quad v}}$
Batch function application	$f : \mathbb{R}^d \to \mathbb{R}, X \in \mathbb{R}^{b \times d}$	$f(X) \in \mathbb{R}^b$	$\{f \leftarrow \overset{\overset{\text{b}}{/}}{X}\}$

As an example of a more complicated function, let $\exp : \mathbb{R} \to \mathbb{R}$ be the element-wise exponential function, and $\mathrm{pow}^{-1} : \mathbb{R} \to \mathbb{R}$ be the element-wise inverse power function. Then we can write $\mathrm{softmax} : \mathbb{R}^d \to \mathbb{R}^d$ as the tensor diagram:

$$\mathrm{softmax}(x) = \{\exp \quad \overset{/}{x}\} \qquad \{\mathrm{pow}^{-1} \quad \{\exp \quad \overparen{x\} \quad \circ}\}\ .$$

Note in particular how $\{\exp \quad \overparen{x\} \quad \circ}$ is the diagram way of writing $\sum_i \exp(x_i)$. Alternatively we could have used a function $\mathrm{sum} : \mathbb{R}^n \to \mathbb{R}$, but the more we can express in terms of tensor diagrams, the more we can use the powerful tensor diagram calculus.

Stuff about analytical matrix functions. Such as Exponential Matrix Function. I'd rather talk about my general function notation. And maybe about taylor series?

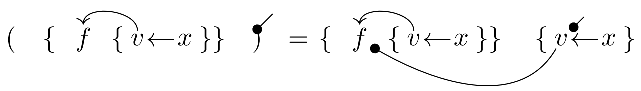
5.1 The Chain Rule

Sometimes the objective is to find the derivative of a matrix which is a function of another matrix.

E.g. $f : \mathbb{R}^n - > \mathbb{R}^n$

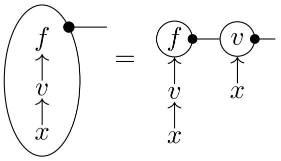
Standard chain rule. Here we let $f \in \mathbb{R}^d \to \mathbb{R}$ be a scalar function, and $v \in \mathbb{R}^d \to \mathbb{R}^d$ be a vector function as used in backprop.

Visualization of the Chain Rule: $J_{f\circ v}(x) = \nabla_f(v(x))J_v(x)$.



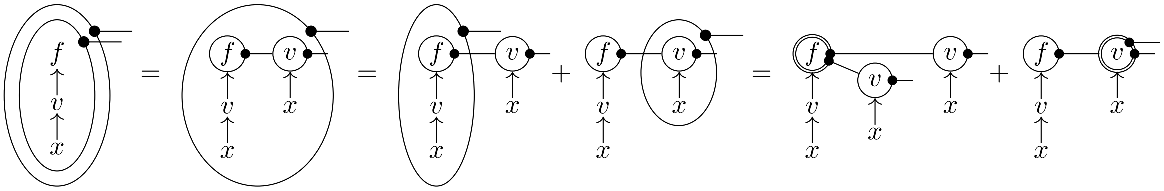
5.1.1 The Chain Rule

Let $f \in \mathbb{R}^d \to \mathbb{R}$ be a scalar function, and $v \in \mathbb{R}^d \to \mathbb{R}^d$ be a vector function, as used in backprop. Then we can write the chain rule as:



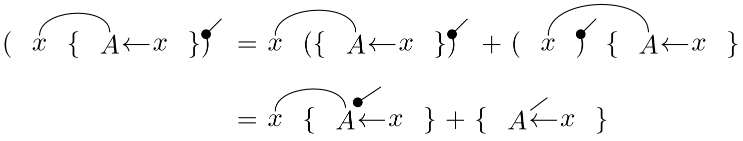
Using standard notation: $J_{f\circ v}(x) = \nabla_f(v(x))J_v(x)$.

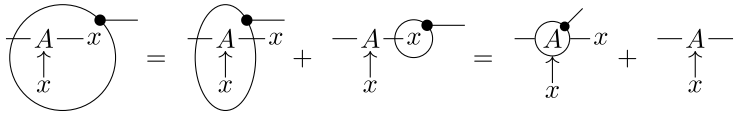
The second derivative, (or Hessian Chain rule):



Using standard notation: $H_{f\circ v}(x) = Dv(x)^T \cdot D^2 f(v(x)) \cdot Dv(x) + \sum_{k=1}^{d} \frac{\partial f}{\partial u_k}(v(x)) \frac{\partial^2 v_k}{\partial x \partial x^T}(x)$.

$$\frac{\partial A(x)x}{\partial x} = (x^T \otimes I)\frac{\partial}{\partial x}\text{vec}[A(x)] + A(x)$$





are common. All pixel-adaptive filters like non-local means, bilateral, etc, and the so-called attention mechanism in transformers can be written this way

Gradient of this f(x) is important & has a form worth remembering. . .

Pseudo-linear form

Maybe this should just be an example in a table?

Derivation of Peyman Milanfar's gradient

$$\begin{aligned}
\mathrm{d}[\mathbf{f}(\mathbf{x})] &= \mathrm{d}[\mathbf{A}(\mathbf{x})\mathbf{x}] \\
&= \mathrm{d}[\mathbf{A}(\mathbf{x})]\mathbf{x} + \mathbf{A}(\mathbf{x})\mathrm{d}\mathbf{x} \\
&= \mathrm{vec}\{\mathrm{d}[\mathbf{A}(\mathbf{x})]\mathbf{x}\} + \mathbf{A}(\mathbf{x})\mathrm{d}\mathbf{x} \\
&= \mathrm{vec}\{\mathbf{I}\mathrm{d}[\mathbf{A}(\mathbf{x})]\mathbf{x}\} + \mathbf{A}(\mathbf{x})\mathrm{d}\mathbf{x} \\
&= \left(\mathbf{x}^T \otimes \mathbf{I}\right) \mathrm{vec}\{\mathrm{d}[\mathbf{A}(\mathbf{x})]\} + \mathbf{A}(\mathbf{x})\mathrm{d}\mathbf{x} \\
&= \left(\mathbf{x}^T \otimes \mathbf{I}\right) \mathrm{D}\,\mathrm{vec}[\mathbf{A}(\mathbf{x})]\mathrm{d}\mathbf{x} + \mathbf{A}(\mathbf{x})\mathrm{d}\mathbf{x} \\
&= \left[\left(\mathbf{x}^T \otimes \mathbf{I}\right) \mathrm{D}\,\mathrm{vec}[\mathbf{A}(\mathbf{x})] + \mathbf{A}(\mathbf{x})\right] \mathrm{d}\mathbf{x}
\end{aligned}$$

5.1.2 Taylor

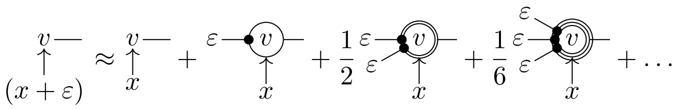
For an n-times differentiable function $v : \mathbb{R}^d \to \mathbb{R}^d$ we can write the Taylor expansion:

$$\begin{aligned}
v(x+\varepsilon) &\approx v(x) + \left[\frac{\partial}{\partial x} v(x)\right] \varepsilon + \frac{1}{2}\left[\frac{\partial}{\partial x}\left[\frac{\partial}{\partial x} v(x)\right] \varepsilon\right] \varepsilon + \frac{1}{6}\left[\frac{\partial}{\partial x}\left[\frac{\partial}{\partial x}\left[\frac{\partial}{\partial x} v(x)\right] \varepsilon\right] \varepsilon\right] \varepsilon + \dots \\
&= v(x) + \left[\frac{\partial}{\partial x} v(x)\right] \varepsilon + \frac{1}{2}(I \otimes \varepsilon)\left[\frac{\partial \mathrm{vec}}{\partial x}\left[\frac{\partial v(x)}{\partial x}\right]\right] \varepsilon + \frac{1}{6}(I \otimes \varepsilon \otimes \varepsilon)\left[\frac{\partial \mathrm{vec}}{\partial x}\left[\frac{\partial \mathrm{vec}}{\partial x}\left[\frac{\partial v(x)}{\partial x}\right]\right]\right] \varepsilon + \dots
\end{aligned}$$

Or with indices:

$$v_i(x+\varepsilon) \approx v_i(x) + \sum_j \frac{\partial v_i(x)}{\partial x_j}\varepsilon_j + \frac{1}{2}\sum_{j,k} \frac{\partial v_i(x)}{\partial x_j \partial x_k}\varepsilon_j\varepsilon_k + \frac{1}{6}\sum_{j,k,\ell} \frac{\partial v_i(x)}{\partial x_j \partial x_k \partial x_\ell}\varepsilon_j\varepsilon_k\varepsilon_\ell$$

Or diagrams:



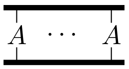
TODO: Examples based on idempotent matrices etc.

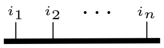
Chapter 6

Determinant and Inverses

6.1 Determinant

It's convenient to write the determinant in tensor notation as

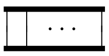
$$\det(A) = \frac{1}{n!}$$ 

where 

$= \varepsilon_{i_1,\dots,i_n}$ is the rank-n Levi-Civita tensor defined by

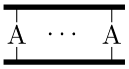
$$\varepsilon_{i_1,\dots,i_n} = \begin{cases} \text{sign}(\sigma) & \sigma = (i_1,\dots,i_n) \text{ is a permutation} \\ 0 & \text{otherwise.} \end{cases}$$

To see that the definition makes sense, let's first consider

$$\det(I) = \frac{1}{n!}$$ 

$$= \frac{1}{n!} \sum_{i_1,\dots,i_n,j_1,\dots,j_n} \varepsilon_{i_1,\dots,i_n}\varepsilon_{j_1,\dots,j_n}[i=j] = \frac{1}{n!}\sum_{i_1,\dots,i_n} \varepsilon^2_{i_1,\dots,i_n} = 1.$$

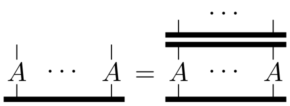
In general we get from the permutation definition of the determinant:



$$\begin{aligned}
&= \sum_{i_1,\dots,i_n,j_1,\dots,j_n} \varepsilon_{i_1,\dots,i_n}\varepsilon_{j_1,\dots,j_n} A_{i_1,j_1}\cdots A_{i_n,j_n} \\
&= \sum_{\sigma,\tau} \text{sign}(\sigma)\text{sign}(\tau) A_{\sigma_1,\tau_1}\cdots A_{\sigma_n,\tau_n} \\
&= \sum_{\sigma} \text{sign}(\sigma) \sum_{\tau} \text{sign}(\tau) A_{\sigma_1,\tau_1}\cdots A_{\sigma_n,\tau_n} \\
&= \sum_{\sigma} \text{sign}(\sigma)^2 \det(A) \\
&= n!\det(A).
\end{aligned}$$

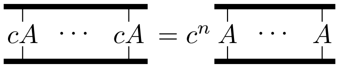
The definition generalizes to Cayley's "hyper determinants" by ....

A curious property is that



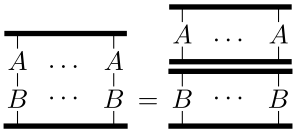
(18) $\det(A) = \prod_i \lambda_i$ . . .

(19) $\det(cA) = c^n \det(A)$



(20) $\det(A) = \det(A^T)$ . . .

(21) $\det(AB) = \det(A)\det(B)$



(22) $\det(A^{-1}) = 1/\det(A)$ . . .

(23) $\det(A^n) = \det(A)^n$ . . .

(24) $\det(I + uv^T) = 1 + u^T v$ . . .

6.2 Inverses

Might be reduced, unless cofactor matrices have a nice representation?

Chapter 7

Advanced Derivatives

7.1 Derivatives of vector norms

7.1.1 Two-norm

$$\frac{d}{dx}\|x-a\|_2 = \frac{x-a}{\|x-a\|_2} \tag{7.1}$$

$$\frac{d}{dx}\frac{x-a}{\|x-a\|_2} = \frac{I}{\|x-a\|_2} - \frac{(x-a)(x-a)^T}{\|x-a\|_2^3} \tag{7.2}$$

$$\frac{d}{dx}\|x\|_2^2 = \frac{d}{dx}\|x^T x\|_2 = 2x \tag{7.3}$$

7.2 Derivatives of matrix norms

7.3 Derivatives of Structured Matrices

7.3.1 Symmetric

7.3.2 Diagonal

7.3.3 Toeplitz

7.4 Derivatives of a Determinant

7.5 General forms

7.6 Linear forms

7.7 Square forms

7.8 Derivatives of an Inverse

7.9 Derivatives of Eigenvalues

Chapter 8

Special Matrices

8.0.1 Block matrices

Stuff like Schur complements is interesting. But can we say anything useful using tensor diagrams?

8.0.2 The Discrete Fourier Transform Matrix

I think FFT can be nicely described with diagrams

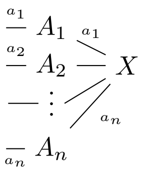
Let's start with the Hadamard matrix: $H_n = \left[\begin{smallmatrix}1 & 1\\ 1 & -1\end{smallmatrix}\right]^{\otimes n}$. Hm. It's just a bunch of matrices below each other, kinda boring.

What about the FFT? Does that require a bit more?

8.0.3 Fast Kronecker Multiplication

Say we want to compute $(A_1 \otimes A_2 \cdots A_n)x$, where A_i is a $a_i \times a_i$ matrix, and $x \in \mathbb{R}^{a_1 a_2 \cdots a_n}$. If we first compute the Kronecker product, and then the matrix-vector multiplication, this would take $(a_1 \cdots a_n)^2$ time.

Instead we can reshape x into a $a_1 \times \cdots a_n$ tensor and perform the multiplication



by contracting the a_i edges one by one. This takes time

$$a_1^2(a_2 \cdots a_n) + a_2^2(a_1 a_3 \cdots a_n) + \cdots + a_n^2(a_1 \cdots a_{n-1}) = (a_1 + \cdots + a_n)(a_1 \cdots a_n),$$

which is the basis of many fast algorithm as we will see.

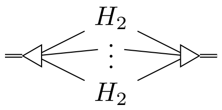
Hadamard

The Hadamard matrix is defined as $H_{2^n} = H_2^{\otimes n} = H_2 \otimes \cdots \otimes H_2$ where $H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$.

For example

$$H_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}.$$

This gives a very simple tensor diagram representation as:

$H_{2^n} =$  .

The Fast Hadamard Transform (FHT) transform is usually described recursively by:

$$H_{2^n} x = \begin{bmatrix} H_{2^{n-1}} & H_{2^{n-1}} \\ H_{2^{n-1}} & -H_{2^{n-1}} \end{bmatrix} \begin{bmatrix} x^{(1)} \\ x^{(2)} \end{bmatrix},$$

where $\left[\begin{smallmatrix} x^{(1)} \\ x^{(2)} \end{smallmatrix}\right]$ is the first and second half of x. Because of the redundancy in the matrix multiplication (it only depends on $H_{2^{n-1}} x^{(1)}$ and $H_{2^{n-1}} x^{(2)}$, the algorithm computes $H_N x$ in $O(N \log N)$ time.

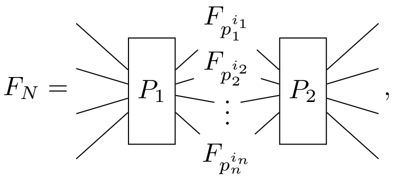
Alternatively we could just use the general fact, as described above, where $a_i = 2$ for all i. Then the "fast Kronecker multiplication" method takes time $(a_1 a_2 \cdots a_n)(a_1 + a_2 + \cdots a_n) = 2n \log_2 n$.

Fourier

A more common matrix is the Discrete Fourier Matrix. It is defined by $(F_N)_{i,j} = \omega^{ij}$, where $\omega = e^{-2\pi i/N}$. We can write it out as

$$F_N = \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \cdots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \cdots & \omega^{2(N-1)} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \cdots & \omega^{3(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \omega^{3(N-1)} & \cdots & \omega^{(N-1)(N-1)} \end{bmatrix}.$$

The Good-Thomas Fast Fourier Transformer (FFT) uses a decomposition based on the Chinese Remainder Theorem:



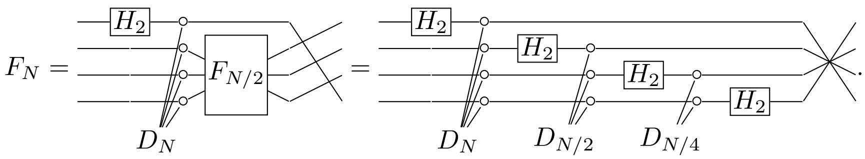
where $N = p_1^{i_1} p_2^{i_2} \cdots p_n^{i_n}$ is the prime factorisation of N, and P_1 and P_2 are some permutation matrices.

Using fast Kronecker multiplication, the algorithm this takes $(p_1^{i_1} + \cdots + p_n^{i_n})N$ time. By padding x with zeros, we can increase N by a constant factor to get a string of $n = O(\log(N)/\log\log(N))$ primes, the sum of which is $\sim n^2/\log n = O(\log(N)^2)$. The complete algorithm thus takes time $O(N\log(N)^2)$. Next we will see how to reduce this to $O(N\log N)$.

The classical Cooley-Tukey FFT algorithm uses a recursion:

$$F_N = \begin{bmatrix} I & I \\ I & -I \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & D_{N/2} \end{bmatrix} \begin{bmatrix} F_{N/2} & 0 \\ 0 & F_{N/2} \end{bmatrix} \begin{bmatrix} \text{even-odd} \\ \text{permutation} \end{bmatrix},$$

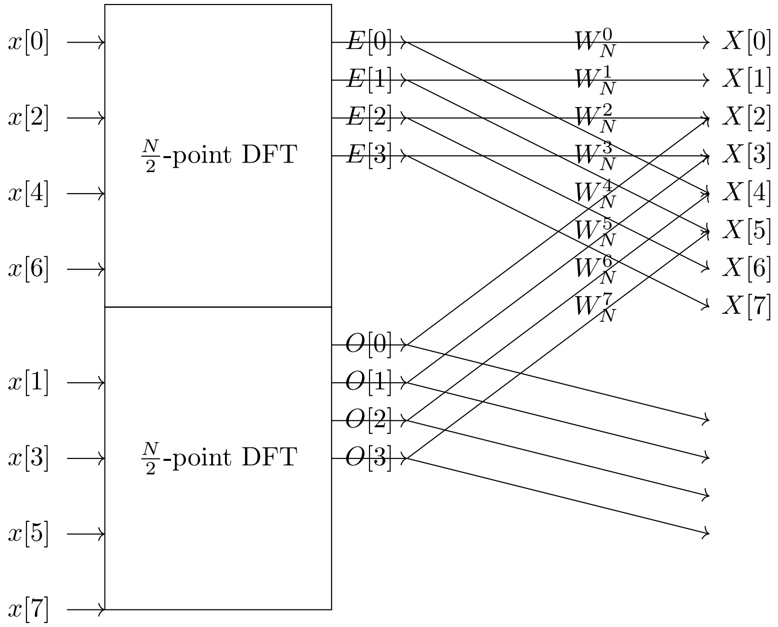
where $D_N = [1, w^N, w^{2N}, \ldots]$. The even-odd permutation moves all the even values to the start. If we reshape I_{2^n} as $I_2 \otimes \cdots \otimes I_2$, this permutation is just $P_N =$, or in pytorch: `x.permute([3,0,1,2])`. Also note that $\left[\begin{smallmatrix} I & I \\ I & -I \end{smallmatrix}\right] = H_2 \otimes I$ and $\left[\begin{smallmatrix} F_{N/2} & 0 \\ 0 & F_{N/2} \end{smallmatrix}\right] = I_2 \otimes F_{N/2}$. So we can write in tensor diagram notation:



Since one can multiply with the permutation and diagonal matrices in linear time, the $O(n \log n)$ time complexity follows from the same argument as for Hadamard.

TODO: We can also talk about the row/column version of Cooley-Tukey (Bailey's FFT algorithm). The nice thing is that we can compose arbitrarily (not necessarily in H_2 chunks), and the D_N matrices have a nice form (they are just fourier matrices!) So they could in principle be decomposed too, by the same algorithm...

Note that this figure may look different from some FFT diagrams you have seen. These typically look like this:



and have 2^n rows. The tensor diagram only has n rows (or $\log_2 N$).

Multi-dimensional Fourier Transform

This is just taking the Fourier transform along each axis.

8.0.4 Hermitian Matrices and skew-Hermitian

Complex. Skip

8.0.5 Idempotent Matrices

Skip

8.0.6 Orthogonal matrices

Skip

8.0.7 Positive Definite and Semi-definite Matrices

Skip

8.0.8 Singleentry Matrix, The

Describes the matrix J. All of this is trivial with diagrams.

8.0.9 Symmetric, Skew-symmetric/Antisymmetric

Could introduce Penrose's symmetric tensors here?

8.0.10 Toeplitz Matrices

Could talk about the convolution tensor here...

8.0.11 Units, Permutation and Shift

Not that interesting...

8.0.12 Vandermonde Matrices

Does this have a nice description? Not a lot of properties are given in the Cookbook.

Chapter 9

Decompositions

9.1 Higher-order singular value decomposition

Say we have an order n tensor A. We “unfold” A along each dimension. This means pulling the edge i to the left, and flattening the rest to the right. Then we compute the SVD, USV. Here U is a square matrix, which we keep. We multiply the ith edge of A by U^T (which is also the inverse of U). The result is a “core” tensor as well as a sequence of U tensors. If we want a more compact SVD, we can make each U low rank, like normal SVD. There is also the “Interlacing computation“ where we multiply the U^Ts onto A as we go along.

For order 3 tensors, this method is called a “Tucker decomposition”.

If the “core matrix” is diagonal, this is called tensor rank decomposition. If we were good at that, we could use it to factor $I^{\otimes 3}$ to get better matrix multiplication algorithms. Unfortunately tensor rank decomposition is NP hard.

I guess HOSVD gives a rank decomposition if we diagonalize the core tensor. It just won’t be an efficient one.

9.2 Rank Decomposition

9.3 Fast Matrix Multiplication

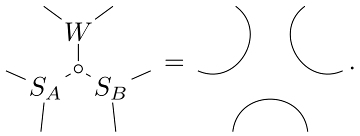
Strassen defines 3 tensors of shape 7 × 2 × 2:

$$S_A = \left[\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} -1 & 0 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 0 & -1 \end{bmatrix} \right]$$

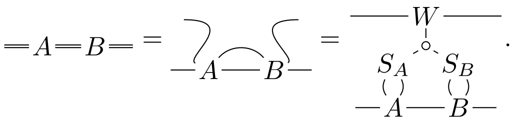
$$S_B = \left[\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 0 & -1 \end{bmatrix}, \begin{bmatrix} -1 & 0 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix} \right]$$

$$W = \left[\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}, \begin{bmatrix}0 & 1\\ 0 & -1\end{bmatrix}, \begin{bmatrix}0 & 0\\ 1 & 1\end{bmatrix}, \begin{bmatrix}1 & 1\\ 0 & 0\end{bmatrix}, \begin{bmatrix}-1 & 0\\ 1 & 0\end{bmatrix}, \begin{bmatrix}0 & 0\\ 0 & 1\end{bmatrix}, \begin{bmatrix}1 & 0\\ 0 & 0\end{bmatrix}\right]$$

These tensors have the neat property that they factor $I_2 \otimes I_2 \otimes I_2$:

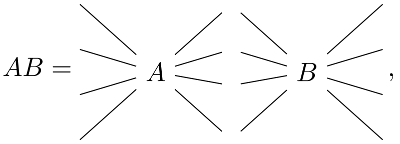


To multiply two matrices, A and B, faster than the normal n^3 time, we reshape them as block matrices, shape $(2, \frac{n}{2}, 2, \frac{n}{2})$ and use Strassen's tensor:



Contracting the edges in the right order, uses only $7/8n^3 + O(n^2)$ operations.

If we instead reshape to $(2, 2, \ldots, 2)$,



and using Strassen's tensor along each axis reduces the work by $(7/8)^{\log_2(n)}$, giving us matrix multiplication in time $n^{3+\log_2(7/8)} = n^{2.80735}$.

Contracting the double edges, $S_A - A$ and $S_B - B$, is both $O(n^2)$ time.

It remains to verify that this is actually faster than the naive matrix multiplication: Contracting $S_A - A$ takes $7 \cdot 2^2(n/2)^2$ operations, and likewise $S_B - B$. Next we contract $S_A A - S_B B$ which takes $7(n/2)^3$ time. And finally we contract the edge with W which takes $2^2{\cdot}7(n/2)^2$. The important term is the cubic $7/8n^3$, which if instead done recursively, leads to the classical $O(n^{\log_2 7})$ algorithm.

FIXME: What "edge with W"? I think we have to/want to contract the hyperedge with W immediately?

Other

If we instead wrote A and B using (n, m) and (m, p) shaped blocks, we could factor $I_n \otimes I_m \otimes I_p$ and get a matrix multiplication algorithm using the same approach as the Strassen $(2, 2, 2)$ tensors above. Lots of papers have investigated this problem, which has led to the best algorithms by Josh Alman and others. For example, Deep Mind found a rank 47 factorization of $I_3 \otimes I_4 \otimes I_5$.

Maybe a more interesting example is the $(4, 4, 4)$ tensor, for which they find a rank 47 factorization. This an easy way to create a rank 49 is to take Strassen and double it. Would this be a nice thing to show? Maybe too messy? Well, actually their rank 47 construction only works in the "modular" case. Then $(3, 4, 5)$ is general.

Chapter 10

Machine Learning Applications

10.1 Least Squares

10.2 Hessian of Cross Entropy Loss

10.3 Convolutional Neural Networks

10.4 Transformers / Attention

10.5 Tensor Sketch

Chapter 11

Tensorgrad

Implementation details

11.1 Simplification Rules

11.2 Functions

11.3 Isomorphisms

Chapter 12

Appendix

Contains some proofs, such as of equation 524 or 571. They are pretty long and could be useful for contrasting with the diagram proofs.