

Week 3: Matrices: manipulation and computation

Algorithmic Data Science

2022-23

A puzzle for you

- The probability of it being sunny is $\frac{2}{9}$
- The probability of it raining is $\frac{4}{9}$
- The probability of it being overcast is $\frac{1}{3}$
- The probability that I go to the park if it is sunny is $\frac{3}{4}$
- The probability that I go to the park if it is raining is $\frac{1}{2}$
- The probability that I go to the park if it is overcast is $\frac{2}{3}$
- What is the probability that I go to the park?

- The probability of it being sunny is $\frac{2}{9}$
- The probability of it raining is $\frac{4}{9}$
- The probability of it being overcast is $\frac{1}{3}$
- The probability that I go to the park if it is sunny is $\frac{3}{4}$
- The probability that I go to the park if it is raining is $\frac{1}{2}$
- The probability that I go to the park if it is overcast is $\frac{2}{3}$
- What is the probability that I go to the park?

$$P(p) = P(p|s)P(s) + P(p|r)P(r) + P(p|o)P(o)$$

$$= \frac{1}{6} + \frac{2}{9} + \frac{2}{9} = \frac{11}{18}$$

Week	Who	Topic
1	Barrett	Data structures and data formats
2	Barrett	Algorithmic complexity. Sorting.
3	Barrett	Matrices: Manipulation and computation
4	Barrett	Similarity analysis
5	Rosas	Processes and concurrency
6	Rosas	Distributed computation
7	Barrett	Map/reduce
8	Barrett	Clustering, graphs/networks
9	Barrett	Graphs/networks, PageRank algorithm
10	Barrett	Databases
<i>11</i>		<i>independent study</i>

This session

- Another set of notes on the elementary matrix operations.
- An application of matrices
- Algorithms for matrix multiplication
- Inverting matrices and solving systems of linear equations:
 - Gaussian elimination and LUP decomposition
- Cosine similarity
- Eigenvalues and eigenvectors.

A matrix is

- a structured collection of numbers, e.g.,

$$A = \begin{pmatrix} 0 & 3 \\ -2 & 5 \\ 0.2 & 10 \end{pmatrix}$$

- matrix A has 3 rows and 2 columns. Its **dimensionality** is 3x2
- Individual elements, a_{ij} , can be referred to by subscripts.
- i refers to the row
- j refers to the column
- So here, $a_{21} = -2$

Matrix terminology

- A **vector** is a 1 dimensional matrix (dimensionality = $1 \times n$ or $n \times 1$)
- A **row vector** is $1 \times n$ whereas a **column vector** is $n \times 1$
- A **zero matrix** is a matrix where every entry is 0
- A **square** matrix has dimensionality $n \times n$
- A **diagonal** matrix is a square matrix with $a_{ij} = 0$ if $i \neq j$
- An **identity** matrix, I , is a diagonal matrix with $a_{ij} = 1$ if $i = j$
- Let's see an example of each of these.

- A **row vector**: $(5 \ 2 \ 3)$
- A **column vector**: $\begin{pmatrix} 1 \\ 4 \\ 2 \end{pmatrix}$
- A **zero matrix**: $\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$
- A **square matrix**: $\begin{pmatrix} 1 & 3 \\ 4 & 2 \end{pmatrix}$
- A **diagonal matrix**: $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 5 \end{pmatrix}$
- An **identity matrix**, I : $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$

Matrix operations: transpose

- The transpose of a matrix A is the matrix A^T obtained by exchanging the rows and columns of A .

$$A = \begin{pmatrix} 0 & 3 \\ -2 & 5 \\ 0.2 & 10 \end{pmatrix} \rightarrow A^T = \begin{pmatrix} 0 & -2 & 0.2 \\ 3 & 5 & 10 \end{pmatrix}$$

- A **symmetric** matrix satisfies the condition $A = A^T$
- Write down a symmetric matrix.

$$\begin{pmatrix} 1 & 4 \\ 4 & 2 \end{pmatrix}$$

Matrix operations: addition

- Addition can only be carried out for matrices which have the same dimensions
- Addition is defined component-wise:

$$C = A + B \leftrightarrow \forall_{ij} (c_{ij} = a_{ij} + b_{ij})$$

- For example:

$$\begin{pmatrix} 1 & 3 & 1 \\ 1 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 5 \\ 7 & 5 & 0 \end{pmatrix} = \begin{pmatrix} 1+0 & 3+0 & 1+5 \\ 1+7 & 0+5 & 0+0 \end{pmatrix} = \begin{pmatrix} 1 & 3 & 6 \\ 8 & 5 & 0 \end{pmatrix}$$

- For an $m \times n$ matrix, what is the asymptotic run-time of matrix addition in O notation?

$$O(m\ n)$$

Matrix operations: multiplication by a scalar

- If λ is a scalar and $A = (a_{ij})$ is a matrix, then $\lambda A = (\lambda a_{ij})$ is the scalar multiple of A obtained by multiplying each of its elements by λ .
- The **negative** of a matrix is defined as $-A = -1.A$
- Hence $-A = (-a_{ij})$
- Hence, $A + (-A) = 0 = (-A) + A$
- For an $n \times n$ matrix, what is the asymptotic run-time of multiplication by a scalar?

$$O(n^2)$$

Matrix operations: matrix subtraction

- **matrix subtraction** is defined as the addition of the negative of a matrix: $B-A = B+(-A)$

$$1. \begin{pmatrix} 10 & 0 \\ 3 & -2 \\ 5 & -9 \end{pmatrix} - \begin{pmatrix} 0 & 8 \\ -2 & -2 \\ 5 & 1 \end{pmatrix} = \begin{pmatrix} 10 & -8 \\ 5 & 0 \\ 0 & -10 \end{pmatrix}$$

$$2. \begin{pmatrix} 2 & -1 \\ 5 & 3 \end{pmatrix} - \begin{pmatrix} 0 & 5 & 2 \\ 3 & 1 & -5 \end{pmatrix} = \text{Doesn't exist (not compatible)}$$

$$3. 4 \begin{pmatrix} 3 & -1 \\ 0 & 2 \end{pmatrix} - 2I = \begin{pmatrix} 10 & -4 \\ 0 & 6 \end{pmatrix}$$

Matrix operations: matrix multiplication

- Two matrices, A and B , can only be multiplied if they are **compatible**: the number of columns of A equals the number of rows of B .
- If $A = (a_{ij})$ is an $m \times n$ matrix and $B = (b_{jk})$ is an $n \times p$ matrix, then their matrix product $C = AB$ is the $m \times p$ matrix $C = (c_{ik})$ where:
$$c_{ik} = \sum_{j=1}^n a_{ij}b_{jk}$$
- For example:
$$\begin{pmatrix} 2 & 0 & 1 \\ 0 & -1 & 3 \end{pmatrix} \times \begin{pmatrix} 3 & -2 \\ 1 & 5 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} 6 & -4 \\ -1 & -5 \end{pmatrix}$$
- Note that matrix multiplication is **not commutative**:
$$\begin{pmatrix} 3 & -2 \\ 1 & 5 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 2 & 0 & 1 \\ 0 & -1 & 3 \end{pmatrix} = \begin{pmatrix} 6 & 2 & -3 \\ 2 & -5 & 16 \\ 0 & 0 & 0 \end{pmatrix}$$
- For $n \times n$ matrices, what is the asymptotic run-time of (naïve method) matrix multiplication?

$$O(n^3)$$

Identity matrix is multiplicative identity

- Check for yourselves by example, that for any square matrix A , and identity matrix I of the same dimensions as A ,

$$AI = IA = A.$$

$$\begin{pmatrix} 4 & 2 \\ 1 & 3 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 4 & 2 \\ 1 & 3 \end{pmatrix}$$

- *Exercise: Can you prove this in general, using algebra?*

Matrix operations: matrix division?

- How do you find B such that $AB = C$?
- There is no 'division' operator for matrices.
- However, we define the **inverse** of an $n \times n$ matrix A to be the $n \times n$ matrix, denoted A^{-1} (if it exists), such that $AA^{-1} = I = A^{-1}A$
- Hence, in the above example where $AB = C$, it follows that $B = A^{-1}C$

Matrix inverses

- We can test whether B is the inverse of A using matrix multiplication e.g.,

$$AB = \begin{pmatrix} 4 & 7 \\ 2 & 6 \end{pmatrix} \begin{pmatrix} 0.6 & -0.7 \\ -0.2 & 0.4 \end{pmatrix} = \begin{pmatrix} 4 \times 0.6 + 7 \times -0.2 & 4 \times -0.7 + 7 \times 0.4 \\ 2 \times 0.6 + 6 \times -0.2 & 2 \times -0.7 + 6 \times 0.4 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = I$$

- So $A = B^{-1}$ and $B = A^{-1}$
- Many nonzero square matrices do not have inverses. A matrix without an inverse is called **noninvertible** or **singular**.
- If a matrix has an inverse, it is called **invertible** or **non-singular**.
- The transpose operation commutes with the inverse operation:

$$(A^{-1})^T = (A^T)^{-1}$$

- Test this for yourself.

Finding inverses: 2x2 matrices

1. Find the determinant. For a 2x2 matrix A this is:

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$$

2. If (and only if) the determinant is 0 (which it will be if any row or column contains only 0's), then A is singular. Otherwise:

$$A^{-1} = \frac{1}{|A|} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

3. It is straightforward to prove that this is the inverse for a 2x2 matrix.

Finding inverses: larger matrices

- The ij th **minor** of an $n \times n$ matrix A , for $n > 1$, is the $(n-1) \times (n-1)$ matrix $A_{[ij]}$ obtained by deleting the i th row and the j th column of A .

- The determinant of A is given by the recursive procedure:

$$|A| = \begin{cases} a_{11} & \text{if } n = 1 \\ a_{11}|A_{[11]}| - a_{12}|A_{[12]}| + \dots + (-1)^{n+1}a_{1n}|A_{[1n]}| & \text{if } n > 1 \end{cases}$$

$$\begin{vmatrix} 4 & 3 & 5 \\ 1 & 2 & 3 \\ 4 & 1 & 2 \end{vmatrix} = 4 \begin{vmatrix} 2 & 3 \\ 1 & 2 \end{vmatrix} - 3 \begin{vmatrix} 1 & 3 \\ 4 & 2 \end{vmatrix} + 5 \begin{vmatrix} 1 & 2 \\ 4 & 1 \end{vmatrix}$$

- Make a matrix where each element is replaced by the determinant of its minor
- Change the signs of alternate cells (this is called the matrix of cofactors)
- Transpose this matrix (this is called the adjugate or adjoint)
- Multiply by the reciprocal of the determinant.
- For an $n \times n$ matrix, what is the asymptotic run-time of naïve computation of the determinant?

$$O(n!)$$

Applications of matrices: solving systems of linear equations

Imagine we have a set of 3 simultaneous linear equations:

$$\begin{aligned}3x + 2y - z &= 10 \\ -x + 5y - 3z &= -2 \\ 2x - y + 2z &= 0\end{aligned}$$

This can be written as a matrix equation:

$$\begin{pmatrix} 3 & 2 & -1 \\ -1 & 5 & -3 \\ 2 & -1 & 2 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 10 \\ -2 \\ 0 \end{pmatrix}$$

Therefore the solution can be found (if there is one) by calculating:

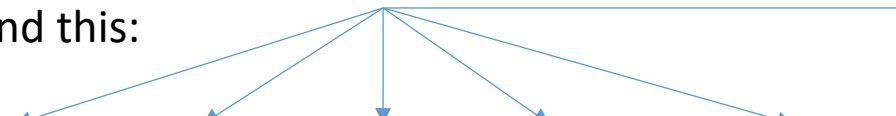
$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 3 & 2 & -1 \\ -1 & 5 & -3 \\ 2 & -1 & 2 \end{pmatrix}^{-1} \begin{pmatrix} 10 \\ -2 \\ 0 \end{pmatrix}$$

Applications of matrices: calculating marginal distributions

The example from the beginning of the lecture can be written:

$$\begin{pmatrix} P(\text{park}|\text{sunny}) & P(\text{park}|\text{raining}) & P(\text{park}|\text{overcast}) \end{pmatrix} \begin{pmatrix} P(\text{sunny}) \\ P(\text{raining}) \\ P(\text{overcast}) \end{pmatrix} = (P(\text{park}))$$

We can easily extend this:



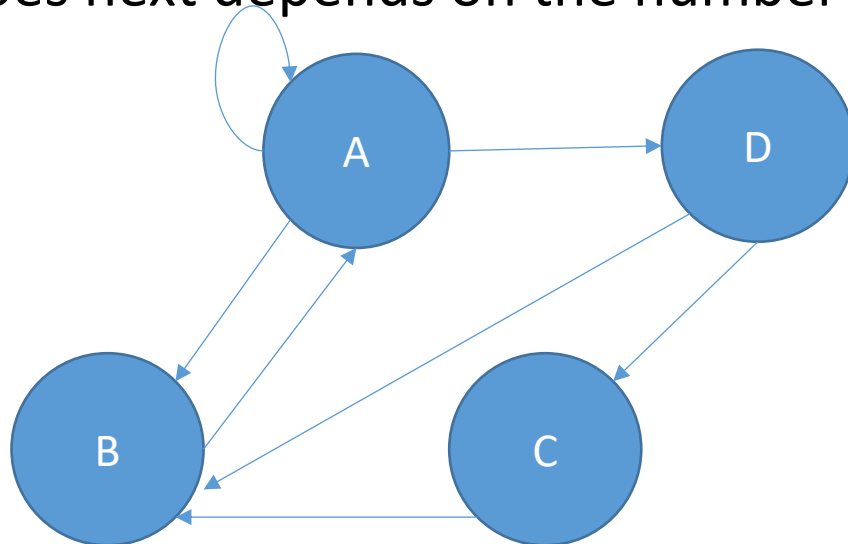
$$\begin{pmatrix} P(p|s) & P(p|r) & P(p|o) \\ P(b|s) & P(b|r) & P(b|o) \\ P(h|s) & P(h|r) & P(h|o) \end{pmatrix} \begin{pmatrix} P(s) \\ P(r) \\ P(o) \end{pmatrix} = \begin{pmatrix} P(p) \\ P(b) \\ P(h) \end{pmatrix}$$

This is **stochastic** if each column sums to 1, i.e., represents a complete probability distribution over the variables (variables must be mutually exclusive and exhaustive)

$$\begin{pmatrix} 3/4 & 1/2 & 2/3 \\ 1/8 & 0 & 1/9 \\ 1/8 & 1/2 & 2/9 \end{pmatrix} \begin{pmatrix} 2/9 \\ 4/9 \\ 1/3 \end{pmatrix} = \begin{pmatrix} 66/108 \\ 7/108 \\ 35/108 \end{pmatrix}$$

The PageRank Algorithm

- Ranks pages on the web by their perceived importance
- Pages are considered more important if they have more links TO them from other more important pages
- Imagine a random surfer on a web with 4 pages. If he is truly random, then there is a uniform probability of him starting anywhere. The probability of where he goes next depends on the number of outlinks from a page



At time 0, t_0 : $P(A) = P(B) = P(C) = P(D) = 1/4$

At time 1, t_1 :

$$P(A|A_0) = 1/3$$

$$P(B|A_0) = 1/3$$

$$P(C|A_0) = 0$$

$$P(D|A_0) = 1/3$$

$$P(A|B_0) = 1$$

$$P(B|B_0) = 0$$

$$P(C|B_0) = 0$$

$$P(D|B_0) = 0$$

....

The PageRank Algorithm

At t_1 :

$$\begin{pmatrix} P(A) \\ P(B) \\ P(C) \\ P(D) \end{pmatrix} = \begin{pmatrix} 1/3 & 1 & 0 & 0 \\ 1/3 & 0 & 1 & 1/2 \\ 0 & 0 & 0 & 1/2 \\ 1/3 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{pmatrix} = \begin{pmatrix} 8/24 \\ 11/24 \\ 3/24 \\ 2/24 \end{pmatrix}$$

Transition matrix: T

At t_n :

$$\begin{pmatrix} P(A) \\ P(B) \\ P(C) \\ P(D) \end{pmatrix} = T^n \begin{pmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{pmatrix}$$

- This tells us where the random surfer is likely to be after n steps.
- So matrix-vector multiplication is the foundation of the PageRank algorithm, named after Larry Page, one of the co-founders of Google.
- Without it, we would still be using Yahoo, Altavista and other search engines which have now all but vanished
- And the matrices are very large at Google

Algorithms for matrix multiplication: naïve method

```
Matrix-Multiply (A,B):  
  if A and B are nxn matrices:  
    let C be an nxn matrix  
    for i from 1 to n:  
      for j from 1 to n:  
         $C_{ij} = 0$   
        for k from 1 to n:  
           $C_{ij} += a_{ik} * b_{kj}$   
    return C
```

The number of multiplications is n^3 .

The number of additions is n^3

So it is straightforward to see that an upper bound on the running time of this algorithm is $O(n^3)$

Strassen's Method

First we note that any multiplication of 2 $n \times n$ matrices where n is a power of 2 can be broken down recursively into the multiplication of $(n/2) \times (n/2)$ matrices.

$$\begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & \dots & \dots \\ k & l & \dots & \dots \end{pmatrix} \begin{pmatrix} z & y & x & w \\ v & u & t & s \\ r & q & \dots & \dots \\ p & o & \dots & \dots \end{pmatrix} = \begin{pmatrix} az + bv + cr + dp & ay + bu + cq + do & \dots & \dots \\ ez + fv + gr + hp & ey + fu + gq + ho & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \end{pmatrix}$$

This can alternatively be written as:

$$\begin{pmatrix} \begin{pmatrix} a & b \\ e & f \end{pmatrix} & \begin{pmatrix} c & d \\ g & h \end{pmatrix} \\ \begin{pmatrix} i & j \\ k & l \end{pmatrix} & \begin{pmatrix} \dots & \dots \\ \dots & \dots \end{pmatrix} \end{pmatrix} \begin{pmatrix} \begin{pmatrix} z & y \\ v & u \end{pmatrix} & \begin{pmatrix} x & w \\ t & s \end{pmatrix} \\ \begin{pmatrix} r & q \\ p & o \end{pmatrix} & \begin{pmatrix} \dots & \dots \\ \dots & \dots \end{pmatrix} \end{pmatrix} = \begin{pmatrix} \begin{pmatrix} a & b \\ e & f \end{pmatrix} \begin{pmatrix} z & y \\ v & u \end{pmatrix} + \begin{pmatrix} c & d \\ g & h \end{pmatrix} \begin{pmatrix} r & q \\ p & o \end{pmatrix} & \dots \\ \dots & \dots \end{pmatrix}$$

Strassen's Method

So for any $n \times n$ matrix, where n is a power of 2, it is straightforward to write matrix multiplication as a recurrence, where the components of the matrices may be numbers or matrices:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} e & g \\ f & h \end{pmatrix} = \begin{pmatrix} r & s \\ t & u \end{pmatrix}$$
$$\begin{aligned} r &= ae + bf \\ s &= ag + bh \\ t &= ce + df \\ u &= cg + dh \end{aligned}$$

This is an example of a divide-and-conquer strategy. We split the problem into smaller problems, solve each smaller problem and then combine the results. Here the smaller problem is of size $n/2$. There are 2^3 of them to solve. Combining requires 4 matrix additions.

Strassen's method

- The running time for this basic recursive approach is given by solving the recurrence formula:

$$T(n) = 8T(n/2) + O(n^2)$$

Since matrix addition is $O(n^2)$. Doing 4 of them only affects the constant.

- The solution of this recurrence formula is: $T(n) = O(8^{\log_2 n}) = O(n^{\log_2 8}) = O(n^3)$
- This is no faster than the naïve method for matrix multiplication
- However Strassen discovered a recursive method which requires only 7 recursive multiplications at each step (but many more additions and subtractions)

$$T(n) = 7T(n/2) + O(n^2) = O(n^{\log_2 7})$$

Strassen's Method

Remember:

$$r = ae + bf$$

$$s = ag + bh$$

$$t = ce + df$$

$$u = cg + dh$$

Calculate (recursively):

$$P_1 = a(g - h)$$

$$P_2 = (a + b)h$$

$$P_3 = (c + d)e$$

$$P_4 = d(f - e)$$

$$P_5 = (a + d)(e + h)$$

$$P_6 = (b - d)(f + h)$$

$$P_7 = (a - c)(e + g)$$

These are equivalent to:

$$P_1 = ag - ah$$

$$P_2 = ah + bh$$

$$P_3 = ce + de$$

$$P_4 = df - de$$

$$P_5 = ae + ah + de + dh$$

$$P_6 = bf + bh - df - dh$$

$$P_7 = ae + ag - ce - cg$$



7 multiplications and 10 additions/subtractions

Strassen's Method

Remember:

$$r = ae + bf$$

$$s = ag + bh$$

$$t = ce + df$$

$$u = cg + dh$$

Calculate (recursively):

$$P_1 = a(g - h)$$

$$P_2 = (a + b)h$$

$$P_3 = (c + d)e$$

$$P_4 = d(f - e)$$

$$P_5 = (a + d)(e + h)$$

$$P_6 = (b - d)(f + h)$$

$$P_7 = (a - c)(e + g)$$

These are equivalent to:

$$P_1 = ag - ah$$

$$P_2 = ah + bh$$

$$P_3 = ce + de$$

$$P_4 = df - de$$

$$P_5 = ae + ah + de + dh$$

$$P_6 = bf + bh - df - dh$$

$$P_7 = ae + ag - ce - cg$$

$$P_1 + P_2 = ag - ah + ah + bh = s$$

1 addition

Strassen's Method

Remember:

$$r = ae + bf$$

$$s = ag + bh$$

$$t = ce + df$$

$$u = cg + dh$$

Calculate (recursively):

$$P_1 = a(g - h)$$

$$P_2 = (a + b)h$$

$$P_3 = (c + d)e$$

$$P_4 = d(f - e)$$

$$P_5 = (a + d)(e + h)$$

$$P_6 = (b - d)(f + h)$$

$$P_7 = (a - c)(e + g)$$

These are equivalent to:

$$P_1 = ag - ah$$

$$P_2 = ah + bh$$

$$P_3 = ce + de$$

$$P_4 = df - de$$

$$P_5 = ae + ah + de + dh$$

$$P_6 = bf + bh - df - dh$$

$$P_7 = ae + ag - ce - cg$$

$$P_3 + P_4 = ce + \cancel{de} + df - \cancel{de} = t$$

1 addition

Strassen's Method

Remember:

$$r = ae + bf$$

$$s = ag + bh$$

$$t = ce + df$$

$$u = cg + dh$$

Calculate (recursively):

$$P_1 = a(g - h)$$

$$P_2 = (a + b)h$$

$$P_3 = (c + d)e$$

$$P_4 = d(f - e)$$

$$P_5 = (a + d)(e + h)$$

$$P_6 = (b - d)(f + h)$$

$$P_7 = (a - c)(e + g)$$

These are equivalent to:

$$P_1 = ag - ah$$

$$P_2 = ah + bh$$

$$P_3 = ce + de$$

$$P_4 = df - de$$

$$P_5 = ae + ah + de + dh$$

$$P_6 = bf + bh - df - dh$$

$$P_7 = ae + ag - ce - cg$$

$$P_5 + P_4 - P_2 + P_6 = ae + \cancel{ah} + \cancel{de} + \cancel{dh} + df - \cancel{de} - \cancel{ah} - \cancel{bh} + bf + \cancel{bh} - \cancel{df} - \cancel{dh} = r$$

3 additions / subtractions

Strassen's Method

Remember:

$$r = ae + bf$$

$$s = ag + bh$$

$$t = ce + df$$

$$u = cg + dh$$

Calculate (recursively):

$$P_1 = a(g - h)$$

$$P_2 = (a + b)h$$

$$P_3 = (c + d)e$$

$$P_4 = d(f - e)$$

$$P_5 = (a + d)(e + h)$$

$$P_6 = (b - d)(f + h)$$

$$P_7 = (a - c)(e + g)$$

These are equivalent to:

$$P_1 = ag - ah$$

$$P_2 = ah + bh$$

$$P_3 = ce + de$$

$$P_4 = df - de$$

$$P_5 = ae + ah + de + dh$$

$$P_6 = bf + bh - df - dh$$

$$P_7 = ae + ag - ce - cg$$

$$P_5 - P_3 - P_7 + P_1 = \cancel{ae} + \cancel{ah} + \cancel{de} + dh - \cancel{ce} - \cancel{de} - \cancel{ae} - \cancel{ag} + \cancel{ce} + cg + \cancel{ag} - \cancel{ah} = u$$

3 additions / subtractions

Strassen's Method

- So we can carry out matrix multiplication with just 7 recursive multiplications of matrices size $n/2$ (but we now have 18 additions / subtractions rather than 4). So:

$$T(n) = 7T(n/2) + O(n^2) = O(n^{\log_2 7}) = O(n^{2.81})$$

- It is possible to modify Strassen's algorithm to work when n is not a power of 2
- In practice, the large constant hidden in the running time makes Strassen's algorithm impractical unless n is large (>45) and dense (few zero entries).
- For sparse matrices, there are special sparse-matrix algorithms which can beat this.
- There are even more advanced techniques which can beat Strassen for dense matrices - $O(n^{2.376})$ is achievable, maybe even better.

Inverting Large Matrices

- The naïve recursive approach is slow (the dimensionality only reduces by 1 each time and you have to do n of the smaller problems).
- Also suffers from **numerical instability**: round-off errors tend to accumulate unduly because floating-point representations are used rather than ideal real numbers.
- Alternative approach is Gaussian elimination / LUP decomposition – which is numerically stable and much faster

Addition and multiplication, simple exercise

$$\begin{pmatrix} 3 & 9 \\ 4 & 7 \end{pmatrix} + \begin{pmatrix} 5 & 3 \\ 4 & 2 \end{pmatrix} =$$

$$\begin{pmatrix} 4 & 7 \\ 2 & 6 \end{pmatrix} \begin{pmatrix} 3 & 1 \\ 4 & 2 \end{pmatrix} =$$

LU(P) decomposition and Gaussian elimination

- We have $Ax = b$, where A is $n \times n$ matrix. We want to find A^{-1}
- Find 3 $n \times n$ matrices, L, U , and P such that: $PA = LU$
- where L is a lower-triangular matrix, U is an upper-triangular matrix and P is a permutation matrix.
- For example:

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 3 & 2 \\ 1 & 5 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 1/3 & 1 \end{pmatrix} \begin{pmatrix} 3 & 2 \\ 0 & 4 \frac{1}{3} \end{pmatrix}$$

P is a permutation matrix

L is a lower triangular matrix

U is a upper triangular matrix

LU(P) Decomposition

- Recursive procedure for carrying out LU(P) decomposition:

$$A = \begin{pmatrix} a_{11} & w^T \\ v & A_{[11]} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ v/a_{11} & L_{n-1} \end{pmatrix} \begin{pmatrix} a_{11} & w^T \\ 0 & U_{n-1} \end{pmatrix}$$

where

$$L_{n-1}U_{n-1} = A_{[11]} - \frac{vw^T}{a_{11}} = S_{n-1}$$

- Permutation matrix P can just be identity matrix, unless $a_{11} = 0$ (permute rows to avoid procedure failing due to a division by 0).
- Carry on until you get the expression for $S_1 = L_1 U_1$. Then set $L_1=1$ and $S_1 = U_1$

e.g.

$$A = \begin{pmatrix} 3 & 1 & -2 \\ 1 & 1 & 2 \\ 4 & -1 & 3 \end{pmatrix}$$

$$A = \begin{pmatrix} a_{11} & w^T \\ v & A_{[11]} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ v/a_{11} & L_2 \end{pmatrix} \begin{pmatrix} a_{11} & w^T \\ 0 & U_2 \end{pmatrix}$$

$$L_2 U_2 = A_{[11]} - \frac{vw^T}{a_{11}} = \begin{pmatrix} 1 & 2 \\ -1 & 3 \end{pmatrix} - \frac{1}{3} \begin{pmatrix} 1 & -2 \\ 4 & -8 \end{pmatrix} = \begin{pmatrix} \frac{2}{3} & \frac{8}{3} \\ -\frac{7}{3} & \frac{17}{3} \end{pmatrix} = S$$

$$S = \begin{pmatrix} 1 & 0 \\ -7/2 & L_1 \end{pmatrix} \begin{pmatrix} 2/3 & 8/3 \\ 0 & U_1 \end{pmatrix} \quad L_1 U_1 = \frac{17}{3} + \frac{28}{3} = 15 \quad \rightarrow \quad L_2 = \begin{pmatrix} 1 & 0 \\ -7/2 & 1 \end{pmatrix}, U_2 = \begin{pmatrix} 2/3 & 8/3 \\ 0 & 15 \end{pmatrix}$$

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 1/3 & 1 & 0 \\ 4/3 & -7/2 & 1 \end{pmatrix} \begin{pmatrix} 3 & 1 & -2 \\ 0 & 2/3 & 8/3 \\ 0 & 0 & 15 \end{pmatrix}$$

Using the LU(P) decomposition

- We know $Ax = b$ and $PA = LU$

➤ $PAx = Pb$

➤ $LUx = Pb$

➤ Let $y = Ux$

➤ Solve $Ly = Pb$ using forward substitution

➤ Solve $Ux = y$ using backward substitution

e.g.

$$Ly = Pb : \begin{pmatrix} 1 & 0 & 0 \\ 2 & -1 & 0 \\ 3 & 2 & 4 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 3 \\ 2 \end{pmatrix}$$

$$\rightarrow y_1 = 1, \quad 2y_1 - y_2 = 3, \quad 3y_1 + 2y_2 + 4y_3 = 2$$

Which is easy.

$$Ux = y: \begin{pmatrix} 2 & 1 & 3 \\ 0 & 1 & 5 \\ 0 & 0 & 6 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \\ 1/4 \end{pmatrix}$$

$$\rightarrow 6x_3 = 1/4, \quad x_2 + 5x_3 = -1, \quad 2x_1 + x_2 + 3x_3 = 1$$

Also easy!

Using LU(P) decomposition to find an inverse

- Solve $AX = I$ for X since $X = A^{-1}$
- Use the LU(P) decomposition on each column of X (and corresponding column of I).
- Each of the n columns can be solved in time $O(n^2)$
- So the computation of X from LUP decomposition is $O(n^3)$
- LUP itself is also $O(n^3)$.

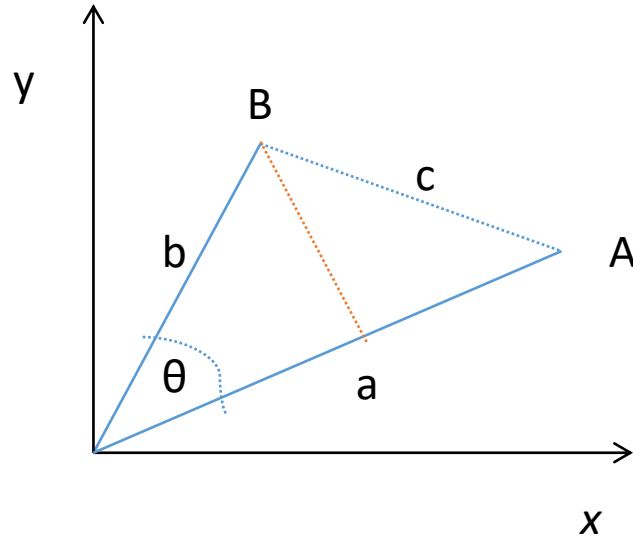
$$A \begin{pmatrix} x_1 & x_2 & \dots & x_n \end{pmatrix} = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{pmatrix}$$

n columns n rows

Application: Cosine similarity

- The similarity of two members of a population (customers, documents etc.) can be measured as the cosine of the angle between their two vectors.
 - E.g. each vector element is number of a certain item purchased.

In 2 dimensions, we can visualize it:



Remember $\cos(\theta) = 1$ when $\theta=0$

And consider the cosine rule:

$$c^2 = a^2 + b^2 - 2ab \cos(\theta)$$

rearranging we get

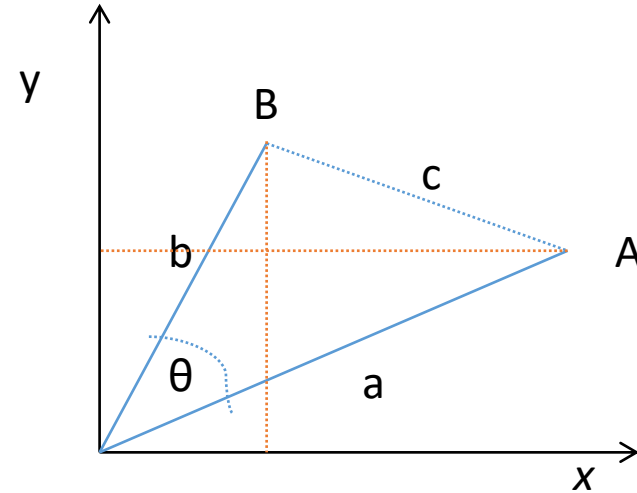
$$\cos(\theta) = \frac{a^2 + b^2 - c^2}{2ab}$$

Then generalize this to any number of dimensions.

Cosine similarity

but if A and B are the vectors:

$$A = \begin{pmatrix} x_A \\ y_A \end{pmatrix}, B = \begin{pmatrix} x_B \\ y_B \end{pmatrix}$$



$$\cos(\theta) = \frac{x_A^2 + y_A^2 + x_B^2 + y_B^2 - (x_A - x_B)^2 - (y_A - y_B)^2}{2\sqrt{(x_A^2 + y_A^2)(x_B^2 + y_B^2)}}$$

$$\cos(\theta) = \frac{(x_A x_B) + (y_A y_B)}{\sqrt{(x_A^2 + y_A^2)(x_B^2 + y_B^2)}}$$

Cosine similarity

But

$$A^T . B = \begin{pmatrix} x_A & y_A \end{pmatrix} \begin{pmatrix} x_B \\ y_B \end{pmatrix} = x_A x_B + y_A y_B$$

So

$$\cos(\theta) = \frac{A^T . B}{\sqrt{A^T . A \times B^T . B}}$$

This is the **dot product** for vectors:

$$\boldsymbol{v} \cdot \boldsymbol{w} = \sum_{i=1}^n v_i w_i$$

$$\cos(\theta) = \frac{\boldsymbol{v} \cdot \boldsymbol{w}}{\sqrt{(\boldsymbol{v} \cdot \boldsymbol{v})(\boldsymbol{w} \cdot \boldsymbol{w})}}$$

All Pairs Similarity

- If A is an $n \times m$ matrix containing *the* n -dimensional vectors (purchase histories) for m customers, we can compute all pairs similarity very straightforwardly.
- First compute all of the dot products using $A^T A$ e.g.

$$\begin{pmatrix} 1 & 2 & 3 \\ -1 & 0 & 3 \end{pmatrix} \begin{pmatrix} 1 & -1 \\ 2 & 0 \\ 3 & 3 \end{pmatrix} = \begin{pmatrix} 14 & 8 \\ 8 & 10 \end{pmatrix}$$

This is almost the covariance matrix – BUT we haven't subtracted the means of individual variables before doing the dot products.

- Then take every element on the leading diagonal and divide all of the elements in its containing row and column by it's square root:

$$\begin{pmatrix} 14/\sqrt{14 \times 14} & 8/\sqrt{14 \times 10} \\ 8/\sqrt{10 \times 14} & 10/\sqrt{10 \times 10} \end{pmatrix} = \begin{pmatrix} 1 & 4/\sqrt{35} \\ 4/\sqrt{35} & 1 \end{pmatrix}$$

[Divide i, j component by (length of vector i times length of vector j)]

All Pairs similarity

- Matrix multiplication can be done in $O(m^2n)$, but can use Strassen's algorithm for large data.
- Dividing every element is $O(m^2)$ so...
- All pairs similarity can be done in less than $O(m^2n)$
- Very important if we want to find clusters of similar objects (where objects are represented by vectors of real-valued features).

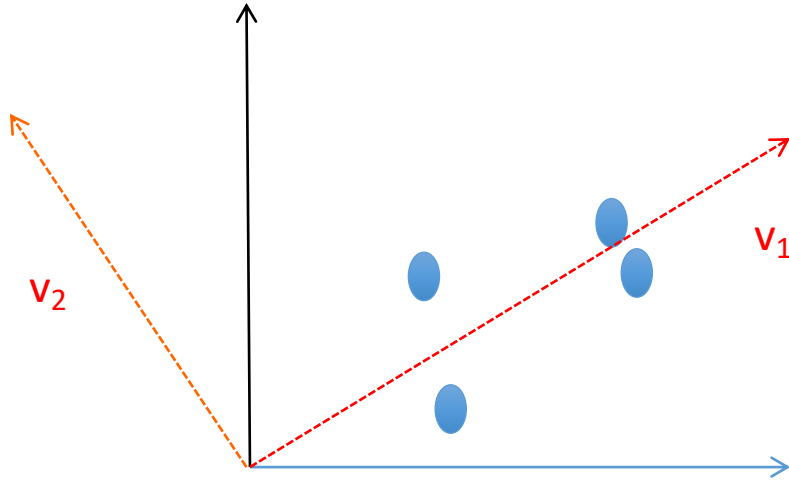
Eigenvalues and Eigenvectors

- Let M be a square matrix. Let λ be a constant and e a non-zero vector. λ is an eigenvalue of M and e is the corresponding eigenvector if:

$$Me = \lambda e$$

- In geometry, if M represents a transformation of the space (e.g., a rotation, reflection or enlargement) then an eigenvector is a vector which is unchanged under the transformation (e.g., the axis of rotation or reflection) and the eigenvalue is the scaling factor.

Why are eigenvalues and eigenvectors important in data science?



We want to rotate this space so that the bases (axes) correspond to the directions along which the points line up best. Here most of the variation is in the direction of the vector v_1

In Principal Component Analysis (PCA), the set of points are a matrix M which have been standardized for mean and variance of each individual point. If we find the eigenvalues and eigenvectors of the covariance/correlation matrix ($\text{Cov}(x_i, x_j)$), then the principal eigenvector will be v_1 and the second eigenvector will be v_2 . This can be applied in multi-dimensional space leading to a set of orthogonal eigenvectors which can be used for dimensionality reduction or clustering.

Finding eigenvalues and eigenvectors

- Solve the equation: $Me = \lambda e$
- This means solving the characteristic equation $|M - \lambda I| = 0$
- For each value of λ found, substitute back into the original equation to find the corresponding eigenvectors.

$$M = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}$$

Multiplying a matrix by its transpose always results in a symmetric matrix. The eigenvectors of a symmetric matrix are orthogonal.

Finding eigenvalues and eigenvectors

$$\begin{vmatrix} 2-\lambda & -1 \\ -1 & 2-\lambda \end{vmatrix} = 0$$



- $(2-\lambda)(2-\lambda) - 1 = 0$
- $\lambda^2 - 4\lambda + 3 = 0$
- $\lambda=1, \lambda=3$

When $\lambda=3$:

$$\begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = 3 \begin{pmatrix} x \\ y \end{pmatrix}$$

$$2x - y = 3x \therefore x = -y$$

$$-x + 2y = 3y \therefore -x = y$$



$$e = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

What is the eigenvector for $\lambda = 1$?

What have you learnt on the following topics?

- Another set of notes on the elementary matrix operations.
- An application of matrices
- Algorithms for matrix multiplication
- Inverting matrices and solving systems of linear equations:
 - Gaussian elimination and LUP decomposition
- Cosine similarity
- Eigenvalues and eigenvectors.