Final topic of the Course:

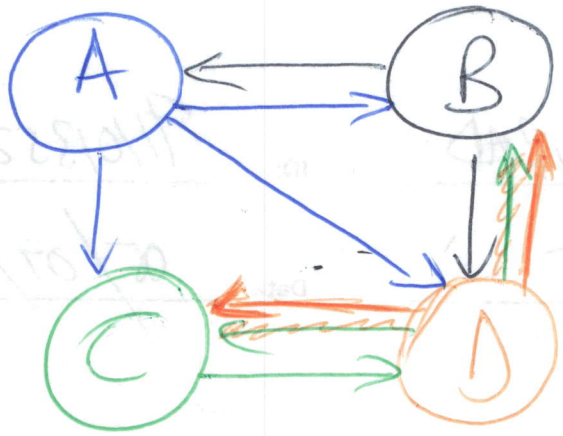
Eigen Problems:

Page Rank: which order to display of websites when they returned from search.

Central assumption (Pagerank): The importance of a website is related to its

— links to and from other website.

— and somehow Eigen theory comes up



Each Bubble is a web page, and arrow

↳ link that take you to web page.

We trying to build an Expression that tells us Based on the network structure, which of these web pages is most relevant to the person that made search.

we will be using procrastinating Pat, who is an imaginary person who goes on internet, and just randomly clicks links to avoid doing ~~the~~ his work.

By mapping all the possible links, we can build a model to estimate the amount of time we would expect Pat to spend on each webpage.

we can describe the links on Page (A) as vector, each raw either 1 or 0, ~~Based~~ on whether there's link to corresponding page. and normalize the vector by total number of links, such that they can be used to describe a probability for that page.

Eg. The vector of links for Page (A), will be (0, 1, 1, 1)

$$\therefore \quad A \xrightarrow{1} B,\quad A \xrightarrow{1} C,\quad A \xrightarrow{1} D,\quad A \xrightarrow{0} A$$

A has link to ... But does not have link to itself

Also, because there are 3 links on this page in total, we would normalize by factor of 3. So that total click probability sum to 1.

$$\therefore L_A = (0, \tfrac{1}{3}, \tfrac{1}{3}, \tfrac{1}{3})$$

$$L_B = (\tfrac{1}{2}, 0, 0, \tfrac{1}{2})$$

$$L_C = (0, 0, 0, 1)$$

$$L_D = (0, \tfrac{1}{2}, \tfrac{1}{2}, 0)$$

we can now build our link matrix L, by using each of our link vectors as a column, which will form a square matrix.

$$L = \begin{bmatrix} 0 & \frac{1}{2} & 0 & 0 \\ \frac{1}{3} & 0 & 0 & \frac{1}{2} \\ \frac{1}{3} & 0 & 0 & \frac{1}{2} \\ \frac{1}{3} & \frac{1}{2} & 1 & 0 \end{bmatrix}$$

The matrix L will show the probability of ending up on each of pages...

Eg The only way to get to (A) is to be on (B)
∴ we need to know the probability of getting to (B)

⇒ which we ~~get~~ get if can get to (B), from either (C) or (D)

∴ Page rank is self referencial, as ranks of all pages, depend on all the others.

Although we build our matrix from columns of outward links, the rows actually describe inward links normalized to w.r.t their page of origin.

Lets write an expression that summarize the approach:

Use vector r to store rank of all webpages:

To calculate the rank of page (A), we need to know 3 things of all other page:

- what's yor rank?
- Do you link to page (A)?
- How many outgoing link do you have in total?

⇒ The following expression combines these 3 pieces of information for web page (A) only:

$$r_A = \sum_{j=1}^{n} L_{A,j} \cdot r_j$$

(n = all web pages; r_j = each r_j)

This will scroll to each of our web pages.

which means that the rank of A, is the sum of ranks of all pages which link to it. weighted by the specific link probability taken from matrix L.

we need to write this expression for all pages and solve them simultaneously:

[previous lesson's say: we can write it as a simple matrix multiplication]

$$r = Lr$$

∴ we start off not knowing r

we simply assume that all ranks are equal and normalize it by total number of web pages in our analysis

$$\therefore r = \begin{bmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{bmatrix}$$

Then, each time we multiply r by our matrix L, this gives us an updated value for r

$$\therefore r^{i+1} = L r^{i}$$

- applying this expression repeatedly means that we are solving the problem iteratively

Each time we do this we update the value of r until eventually stops changing

So now r really does $= Lr$

$\therefore r$ is now an eigenvector, of matrix L and eigenvalue of $\underline{1}$.

But, if we want to multiply r by L many times, perhaps this would be best tackled by applying the diagonalization method.

(But this will require us to know all the eigenvectors, which is what we're trying to find in the first place ②)

So now, we have an equation, and computer will apply it iteratively until it converges to final ranking vector.

∴ takes about 10 iterations for it to settle down.

Result: ⟶

Although there are many approaches to calculating Eigenvectors, repeatedly multiplying a random selected initial guess vector by your matrix, which is called the power method, is still very effective for the page rank problem, for 2 reasons:

— although power method will only give you 1 Eigenvector, but we know there will be n for n × n system., it turns out the way we structured our link matrix, the vector it gives you, will always be the one you are looking for., with Eigenvalue of 1.

— although this is not true for the full webpages, when looking at real internet, you can imagine that almost every entry in the link matrix will be 0, i.e most page do not connect to most other pages
∴ referred to as sparse matrix.

One key aspect not discussed in Pagerank ~~Problem~~ algorithm, is called damping factor, d.

$$\therefore r^{t+1} = d(Lr^i) + \frac{1-d}{n}$$

$\therefore$ d is something between 0 and 1.

END

Summary:

- Goal here is to provide some underpinning of linear algebra (vectors and matrices) in order to access NN, data science course more generally.

- This is not for graduate work.

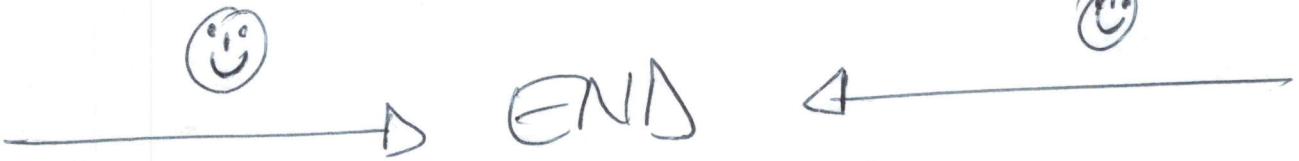
But, More than enough to develop understanding required to access other courses, and

Solve problems with ML in real world.

a lot of work we covered HERE:

- we started out by thinking of data, and sort of problems we might have, primarily optimization problems, like fitting parameters of model data, the distribution of height, and simultaneous equations (like apple and banana price discovery problem)
- we then used those to go on a journey exploring vector properties (like linear combinations, vector addition and scalar multiplication
- we then went on to find the modulus of vector, dot product and related ideas of scalar and vector projections
- that led us to basis changes and vector transformations
- which means we had to define what is meant by Basis, by linear independence and dimensionality of vector space

- we then covered matrices and how to do matrix operations, and how to solve systems of simultaneous equations, using algorithms
- we then looked at basis transformations again, using matrices, and looked at how to construct a basis the Gram-Schmidt approach
- then we covered eigenvectors/values and how to find them.
- and then how to construct Google's famous PageRank algorithm



David Bye you the Best!