FDMA: Frequency Division Multiple Access

Limit: Capacity of frequency channels

TDMA: Time Division Multiple Access

Limit: Speed of switching or pausing talking time

Ideal CDMA: Ideal Code Division Multiple Access

How to shield the other codes is a mathematical application, refer to the textbook.

Advantages: Synchronization

Practical CMDA

Not enough codes: Although you can generate 128 different codes for 128 different people, you can not guarantee that they are mathematically orthogonal.

Imperfect synchronization.

Cocktail Party:

频分多址就好比把聚会大厅分成一个个小房间，房间里面的双方可以很清楚的听到对方讲话，但是这种方式的缺点也很明显，就是能够容纳的客人太少了，只有 20 个房间就把大厅占满了，其他人都没法进场了啊；

时分多址就是在频分多址的基础上，规定每个房间的人只能连续讲 10 分钟的话，超过 10 分钟就要让给下一对儿，等下次轮到你了，你再占用这个房间讲话。通过这种方式，可以有更多的人参加到酒会里来，只要间隔时间（例子中的 10 分钟）在你可以忍受的范围内，那么就不会有什么问题；

CDMA 即码分多址是指，大家都在大厅里，没有小房间了，大厅里可以容纳很多很多人，但是，你和张三说中国话，alice 和 bob 说英文，cici 和 coco 说意大利语，虽然大家都叽叽喳喳的说着话，但是你还是可以很清楚的听明白张三说的什么，因为你只能听懂中国话，听不懂英文和意大利语，就把其他语言当成噪音直接过滤掉了。

Example 1: negative externality

Famous special case: near- far problem

Usually we want the talking power passivated the same no matter the distance is. However, the need may change at certain time. Take an example, at 11:59 pm, a student is watching video on iqiyi, and another student is uploading his homework. They want different speed and different power.

Question: How to design a kind of mechanism so that it can satisfy all people's need?

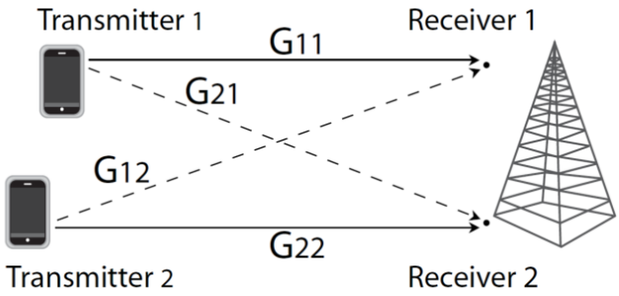
Communication between two mobile phones:

MS1 → BS1 → BS2 → MS2

voice communication in old days: distance long: higher power; distance short: lower power

different APP with different requirements

Distributed power control (DPC)



Channel gain: Several notations:

i: receiver; j: transmitter

G_{ij}: channel gain from transmitter j to receiver I; (direct channel gains, the larger the better)

G_{ii}: is enhanced by CDMA spreading codes. (interference channel gains, the smaller the better)

G_{ij} is determined by the location of the transmitter and receiver; G_{ii} is determined by the location and the code used.

SIR: Signal to Interference noise Ratio: (unit-less ratio)

$$SIR_i = \frac{G_{ii}p_i}{\sum_{j\neq i} G_{ij}p_j + n_i}, \forall i$$

n_i: noise; unit: W

p_j: unit: W

Usually, it's difficult to measure each p_j and G_{ij}, but it's easy to measure the sum product $I = \sum_{j\neq i} G_{ij}p_j$.

SISO, SIMO, MISO, MIMO: signal (multiple) input and signal (multiple) output

For proper decoding of the packets, the receiver needs to maintain a target level of SIR. Which means: $SIR_i \geq \gamma_i$ for any i. γ_i is the target level of SIR. Clearly, increasing p_1 raises the SIR for receiver 1 but lowers the SIR for all other receivers.

Divide the time into discrete slots. An iterative, distributed algorithm (DPC):

$$p_i[t+1] = \frac{\gamma_i}{SIR_i[t]} p_i[t], \forall i, \forall t$$

We need to achieve the target SIR with the minimum transmission power. If the system is stabilized, SIR=1. However, because of the limitation of the resources, it's impossible to make p_i large or eliminate the interference and noise. Only when the target level of SIR is large enough, the system can be stabilized.

Simple: in communication; in computation; in configuration

Intuitive: equilibrium looks good; convergence sounds plausible

A general theme: 1) Individual behaviors driven by self-interact; 2) Aggregate into a (hopefully fair and efficient) state across all users;3) Helped by feedback signals.

DPC as an optimization solution

objective (what wants to optimize): power minimization, $\min\sum_i p_i$;

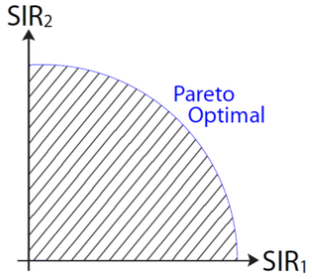
constraints (what needs to satisfy): achieve target SIRs for all users, $SIR_i \geq \gamma_i, \forall i$;

NOTICE: The target level SIR γ_i must be achievable!

variables (what may change): transmit powers, $p_i, \forall i$;

constants (what cannot change): channels, noise, target SIRs, G_{ij}, noise, γ_i.

Feasible region (in SINRs):



$$SIR_i(max) = \frac{p_i(max)G_{ii}}{n_i}$$

Interference relationship determines the Pareto Optimal boundary.

Linear programming:

Minimize a linear function subject to linear constraints;

$$SIR_i = \frac{G_{ii}p_i}{\sum_{j\neq i} G_{ij}p_j + n_i}, \forall i$$

$$\Rightarrow\ G_{ii}p_i \geq \gamma_i\left(\sum_{j\neq i} G_{ij}p_j + n_i\right), \forall i$$

How to prove convergence of DPC algorithm to a solution of this optimization?

Infeasible: set an infeasible γ_i.

Terminologies: Infeasible optimization problem; Feasible optimization problem; Global optimal (on all range of x value) and local optimal solution (larger than its neighborhood value of x).

Game (博弈)

a set of players, {1, 2, …, N};

a strategy space A_i for each player;

a payoff function, or utility function, U_i for each player to maximize (or a cost function to minimize). (Function U_i maps each combination of all players' strategies to a real number, the payoff (or cost), to player i).

Application in power control: 1) Power control is a competition; 2) Every user tries to minimize his transmission power: Subject to the SINR constraint; 3) games are models for Competition & Cooperation; 4) There is a formal (mathematical) language for games.

Introduction of game theory: samples of prisoner's dilemma and cooperation game

[PRISONER'S DILEMMA]

Two suspects are arrested. The police lack sufficient evidence to convict the suspects, unless at least one confesses. The police hold the suspects in separate rooms, and tell each of them three possible consequences. The conditions are as following:

If both deny: 1 month in jail each.

If both confess: 6 months in jail each.

If one confesses and one denies

The one confesses: walk away free of charge.

The one denies: serve 12 months in jail.

	Not Confess	Confess
Not Confess	(-1, -1)	(-5, 0)
Confess	(0, -5)	(-3, -3)

Table for two players

Confess is a strictly dominant strategy for the players, as it always leads to the best payoff among all his strategies, independent of player 2's strategy.

When the best response strategy of a player is the same no matter what strategy the other player chooses, we call that a dominant strategy. It may not exist. But when it exists, a player will obviously pick a dominant strategy.

[COORDINATION GAME]

You and your friend are trying to coordinate which movie to watch together. If you disagree, you will not go to any movie and the payoff is zero for both of you. If you agree, each will get some positive payoff but the values are different as you prefer the romance movie and your friend prefer the action movie.

	Action movie	Romance Movie
Action movie	(2, 1)	(0, 0)
Romance Movie	(0, 0)	(1, 2)

Table for two players

Symbolically, for a two-user game, suppose the two payoff functions are (U_1,U_2) and the two strategy spaces are $(\mathcal{A}, \mathcal{B})$ for the two players, respectively. We say (a*$\in \mathcal{A}$, b*$\in \mathcal{B}$) is a Nash equilibrium if:

$$U_1(a^*, b^*) \geq U_1(a, b^*), a \in \mathcal{A};$$
$$U_2(a^*, b^*) \geq U_2(a^*, b), b \in \mathcal{B}.$$

Two individuals go for a hunt. The conditions are as the following:

Each one can hunt a stag (deer) or a hare.

Successful hunt of stag requires cooperation.

Successful hunt of hare can be done individually.

Simultaneous decisions without prior communications.

	Stag	Hare
Stag	(5, 5)	(0, 2)
Hare	(2, 0)	(2, 2)

There is no strictly dominant or strictly dominated strategies.

Two equilibriums in the hunting: (5, 5) or (2, 2). (Stag, Stag) is payoff dominant: both players get the best payoff possible; (Hare, Hare) is risk dominant: minimum risk if player is uncertain of each other's choice. Therefore, there is no answer for this problem because different people may have different preference.

Nash Equilibrium: A pair of strategies form a Nash Equilibrium (NE) if each player is

choosing the best response given the other player's strategy choice. At a Nash equilibrium, no player can perform a profitable deviation unilaterally.

Define:

- Players: $\mathcal{I} = \{1, 2, 3, \dots, I\}$

- Strategy: $x_i \in \mathcal{X}_i$

- Payoff: $U_i(x_i, x_{-i})$, where $x_{-i} = (x_1, x_{i-1}, x_{i+1}, \dots, x_I)$

NE:

Application in DPC:

- Players: $\mathcal{I}$

- Strategy: $\forall i, p_i \in [0, p_i^{max}]$ $\quad$ $SIR_i(p_i, p_{-i}) \geq \gamma_i$

- Payoff: $\forall i, \min(p_i)$

Textbook for learning game theory and optimization: Convex Optimization (Steven Boyd)

Equilibrium: Socially optimal? Pareto optimal? Exist? Unique?

Example

Receiver of Link	Transmitter of Link 1	2	3	4
1	1	0.1	0.2	0.3
2	0.2	1	0.1	0.1
3	0.2	0.1	1	0.1
4	0.1	0.1	0.1	1

- Target SNRs: 2, 2.5, 1.5, 2
- Noise level: 0.1 mW

Analysis of the Second Price Auction

Strategy: a weakly dominate

Case1: painting: $W, N bidders, each give $b_i \geq 0$. Show that bidding W is a weakly dominate strategy. Let b* $= \max(b_2, \ldots, b_N)$, consider the case from bidder 1

if b*<W:

if b1=b*: the bidder 1 either is the winner with a payoff >0 or is the winner with a 0 payoff.

if b1<b*: then the bidder 1 will not win and gets a 0 payoff.

if b1>b*: then the bidder 1 is the winner with a payoff>0.

if b*=W:

if b1=W: the bidder 1 either is the winner with a payoff or the winner with a 0 payoff.

if b1>W: the bidder 1 is the winner with a 0 payoff.

if b1<W: the bidder 1 will not win and gets a 0 payoff.

if b*>W:

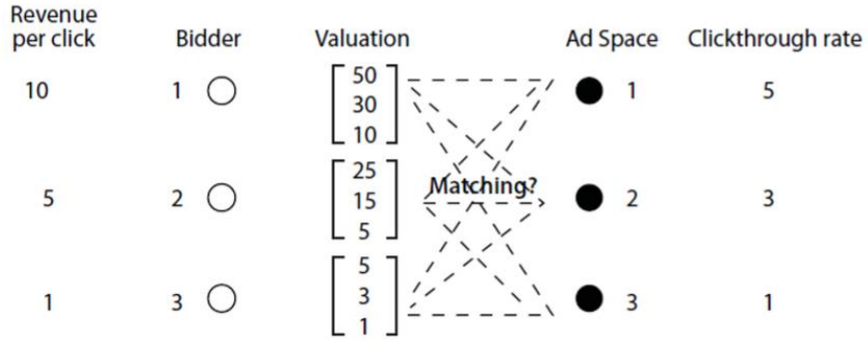
if b1>b*, then the bidder 1 is the winner with a payoff <0.

if b1=b*, then the bidder 1 either is the winner with a payoff <0 or is the winner with a 0 payoff.

if b1<b*, then the bidder 1 will not win and gets a 0 payoff.

Why not the third price?

externality: W1>W2 (bidder 1 should pay W2, if bidder 1 is not here, bidder 2 should get the painting. Therefore, we rule it to be the second price but not the third or fourth bidder.



- Payoff Calculation:

bidder 1: receive space 1; payoff=valuation-payment=50-25=25($/h)

bidder 2: receive space 2; payoff=valuation-payment=15-3=12($/h)

bidder 3: receive space 3; payoff=valuation-payment=1-?

The payment of bidder 3 depends on the price Google fixed. Take example of 0.5$/h, then its payoff is 1-0.5=0.5($/h)

Total Bidder Payoff=25+12+0.5=37.5($/h)

Total Revenue of Google=25+3+0.5=28.5($/h)

GSP: Generalized Second Price

What if the bidding is untruthful?

Buyers (R)
\$12
\$8
\$4

Ad Space (C)
400
300

1) First assume that it's a truthful bidding from everyone.

buyer 1: payoff=(12-8)*400=1600($/h)

buyer 2: payoff=(8-4)*300=1200($/h)

2) If buyer 1 change \$12 to \$7?

buyer 1: payoff=(12-4)*300=2400($/h)

3) Show that truthful bidding is not an NE in this example.

Why Google choose GSP…

In practical, there might be lots of problems, such as bidding shade or other things.

About the project

3 per group, but 2 will have a bonus.

Chapter 3: How Google rank their webpages?

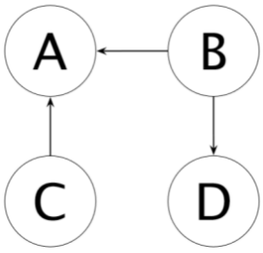
hyperlinks: directed graph: have a direction, when pointing to another page, unable to turn back.

How to quantify node importance: Count the number of links? More important links point to this page? (How to define the importance of Bill Gates?) => cyclic challenge; recursive definition

network: topology & functionality

for topology: graphs, matrices; for functionality: what you do on the graph?

Focusing on sociality, different people do different things even base on the same graph.

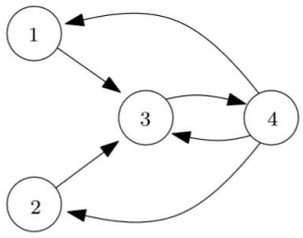


Try 0: $\pi_A = I_A = 2$

Try 1: $\pi_A = \sum_{i \to A} \pi_i = \pi_B + \pi_C$

Try 2: The spread of importance: $\pi_A = \sum_{i \to A} \frac{\pi_i}{O_i}$ O_i: outgoing degree

Scores given according to the above formulas: $\vec{\pi} = (\pi_i, \forall i)$



Systematic graph: symmetric link. According to symmetry:

$$\pi_1 = \pi_2 = x$$

Node 4:

$$\pi_4 = \frac{\pi_3}{O_3} = \pi_3 = y$$

Connect x and y then only need to solve 2 variables.

(if using Node 3: $\pi_3 = \sum_{i \to 3} \frac{\pi_i}{O_i} = \pi_1 + \pi_2 + \frac{\pi_4}{3}$)

$$\pi_1 = x = \sum_{i \to 1} \frac{\pi_i}{O_i} = \frac{\pi_4}{O_4} = \frac{\pi_4}{3} = \frac{y}{3}$$

Normalization: Let the summation to be 1. (Actually it's also fine to keep the summation to be 10 if let the proportion be the same constant of $\vec{\pi}$.)

$$\begin{cases} 2x + 2y = 1 \\ x = \frac{y}{3} \end{cases} \Rightarrow \begin{cases} x = \frac{1}{8} \\ y = \frac{3}{8} \end{cases}$$

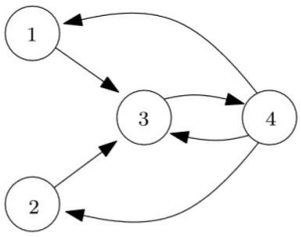
If the webpages have the same scores, then Google will randomly display them.

In reality operation, Google will always generate 2 scores, we are talking about importance score now, not the relevant score.

$$\vec{\pi} = \begin{pmatrix} \pi_1 \\ \pi_2 \\ \pi_3 \\ \vdots \\ \pi_N \end{pmatrix} = (\pi_1, .., \pi_N)^T$$

N*N matrix A: $\vec{\pi}^T A = \lambda \vec{\pi}^T$

$\vec{\pi}^T$: eigenvector of A converges to λ; λ: eigenvalue of A



$$\boldsymbol{H} = \begin{pmatrix} 0 & 0 & 1/1 & 0 \\ 0 & 0 & 1/1 & 0 \\ 0 & 0 & 0 & 1/1 \\ 1/3 & 1/3 & 1/3 & 0 \end{pmatrix}$$

→ each row => one node outgoing => summation = 1

$$\vec{\pi}^T H = \vec{\pi}^T$$

Matrix $\boldsymbol{H}$ defined by

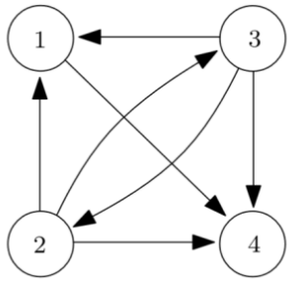
$$H_{ij} = \begin{cases} 1/O_i & \text{if } i \to j \\ 0 & \text{otherwise} \end{cases}$$

$\pi^T \boldsymbol{H} = \pi^T$, hence π is the eigenvector of $\boldsymbol{H}$.

(Take $\lambda = 1$)

For Google, H is a huge matrix. It will directly compute eigenvector (computationally intensive) use iterative algorithm $\vec{\pi}^T[t+1] = \vec{\pi}^T H$. Eventually it will (hopefully) converge to $\vec{\pi}^*$.

For matrix H, sometimes one matrix is not sufficient, we need to modify the matrices.



Identify the node liked to other nodes and note it as row vectors.

$$(\pi_1\ \pi_2\ \pi_3\ \pi_4)\overrightarrow{H_n} = \pi_n$$

$$\mathbf{H} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1/3 & 0 & 1/3 & 1/3 \\ 1/3 & 1/3 & 0 & 1/3 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \qquad \begin{cases} \frac{1}{3}(\pi_2 + \pi_3) = \pi_1 \\ \frac{1}{3}\pi_3 = \pi_2 \\ \frac{1}{3}\pi_2 = \pi_3 \\ \pi_1 + \frac{1}{3}(\pi_2 + \pi_3) = \pi_4. \end{cases}$$

$$\vec{\pi} = 0$$

However, it's not a symmetric graph so that their important scores are not equally important. The zero vector of row 4 in matrix H lead to the zero vector in the final results.

There is no outgoing when reaches node 4, thus suppose there are 25% possibility to go to every other node. The second matrix can then be modified as:

$$\widehat{H} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ \frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{bmatrix}$$

Define $\vec{1} = (1\ 1\ \ldots 1\ 1)^T$, Hence we are adding the following row vector to the row corresponding to the "dangling node":

$$\frac{1}{N}\vec{1}^T$$

Define $\overrightarrow{W} = (0\ 0\ 0\ 1)^T$:

$$\vec{W}\left(\frac{1}{N}\vec{1}^{T}\right) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{bmatrix}$$

$$\hat{H} = H + \frac{1}{N}\vec{W}\vec{1}^{T}$$

Then solve $\vec{\pi}^{T} = \hat{H}\vec{\pi}^{T}$

If the two webpages are disconnected, we need to build a bridge to prevent the errors. We find the initial setting $\vec{\pi}^{T}[0]$ may infect the results of $\vec{\pi}^{T}[t]$ according to the formula $\vec{\pi}^{T}[t+1] = \vec{\pi}^{T}H$, thus we need the third matrix, G.

$$G = \theta\hat{H} + \frac{(1-\theta)1}{N}\vec{1}\vec{1}^{T}$$

Empirically, take $\theta = 0.85$.

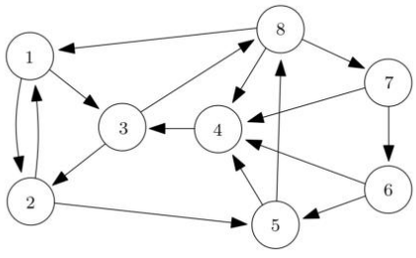
As t → infinity, the $\vec{\pi}^{*} = \vec{\pi}^{*}G$, independent with the initial value choice of $\vec{\pi}[0]$.

Normalization:

$$\vec{\pi}^{*}{}_{i} \leftarrow \frac{\vec{\pi}^{*}{}_{i}}{\sum_{j=1}^{N}\vec{\pi}^{*}{}_{j}}$$

Notice: Display pages on Google search page according to ranking (no scores)

Example:



Which node is most important?

Guess: 3 (8, 7, 6, 5 pointing to 4 and distribute to 3, another 1 pointing to 3), 2 (3 split its importance to 2 and 8 and another 1 pointing to 2) or 1.

In this example, H matrix is the same as $\hat{H}$ matrix because there are no dangling nodes.

$$\mathbf{H} = \begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 0 & 1/2 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 0 & 0 & 0 & 0 & 1/2 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/2 & 0 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/2 & 0 & 1/2 & 0 & 0 \\ 1/3 & 0 & 0 & 1/3 & 0 & 0 & 1/3 & 0 \end{bmatrix}$$

G matrix with $\theta = 0.85$:

$$\mathbf{G} = \begin{bmatrix} 0.0188 & 0.4437 & 0.4437 & 0.0188 & 0.0188 & 0.0188 & 0.0188 & 0.0188 \\ 0.4437 & 0.0188 & 0.0188 & 0.0188 & 0.4437 & 0.0188 & 0.0188 & 0.0188 \\ 0.0188 & 0.4437 & 0.0188 & 0.0188 & 0.0188 & 0.0188 & 0.0188 & 0.4437 \\ 0.0188 & 0.0188 & 0.8688 & 0.0188 & 0.0188 & 0.0188 & 0.0188 & 0.0188 \\ 0.0188 & 0.0188 & 0.0188 & 0.4437 & 0.0188 & 0.0188 & 0.0188 & 0.4437 \\ 0.0188 & 0.0188 & 0.0188 & 0.4437 & 0.4437 & 0.0188 & 0.0188 & 0.0188 \\ 0.0188 & 0.0188 & 0.0188 & 0.4437 & 0.0188 & 0.4437 & 0.0188 & 0.0188 \\ 0.3021 & 0.0188 & 0.0188 & 0.3021 & 0.0188 & 0.0188 & 0.3021 & 0.0188 \end{bmatrix}$$

According to the iteration:

- Choosing initial vector $\boldsymbol{\pi}[0] = (1/8, 1/8, \cdots, 1/8)^T$
- Iterations lead to

$$\begin{aligned} \pi[1] &= [0.1073\ 0.1250\ 0.1781\ 0.2135\ 0.1250\ 0.0719\ 0.0542\ 0.1250]^T \\ \pi[2] &= [0.1073\ 0.1401\ 0.2459\ 0.1609\ 0.1024\ 0.0418\ 0.0542\ 0.1476]^T \\ \pi[3] &= [0.1201\ 0.1688\ 0.2011\ 0.1449\ 0.0960\ 0.0418\ 0.0606\ 0.1668]^T \\ \pi[4] &= [0.1378\ 0.1552\ 0.1929\ 0.1503\ 0.1083\ 0.0445\ 0.0660\ 0.1450]^T \\ \pi[5] &= [0.1258\ 0.1593\ 0.2051\ 0.1528\ 0.1036\ 0.0468\ 0.0598\ 0.1468]^T \\ \pi[6] &= [0.1280\ 0.1594\ 0.2021\ 0.1497\ 0.1063\ 0.0442\ 0.0603\ 0.1499]^T \\ &\vdots \end{aligned}$$

- Final result
  $\pi^* = [0.1286\ 0.1590\ 0.2015\ 0.1507\ 0.1053\ 0.0447\ 0.0610\ 0.1492]^T$
- Page rank: 3, 2, 4, 8, 1, 5, 7, 6.

A more refined randomization ingredient:

if matrix A is difficult to be reversed:

- Write $\boldsymbol{A} = \boldsymbol{M} - \boldsymbol{N}$, where $\boldsymbol{M}$ is easily invertible.
- Hence we want to solve
  $$\boldsymbol{Mx} = \boldsymbol{Nx} + \boldsymbol{b}$$
- We use the following **linear stationary iteration**:
  $$\boldsymbol{x}[k] = \boldsymbol{M}^{-1}\boldsymbol{N}\boldsymbol{x}[k-1] + \boldsymbol{M}^{-1}\boldsymbol{b}$$
- If the largest eigenvalue of $\boldsymbol{M}^{-1}\boldsymbol{N}$ is smaller than 1, then the **global convergence** is guaranteed.

SEO (Search Engine Optimization):

A battle between webpage designers and search engine designers.

Chapter 4: How does Netflix recommend movies?

Rating: (u, i, r_{ui}, t_{ui}) -- input

(1-5)

Recommendation: $\hat{r}_{ui}$ -- output

Root Mean Square Error (RMSE): C is the total number of r_{ui} and $\hat{r}_{ui}$ pairs.

$$\sqrt{\sum_{(u,i)} \frac{(r_{ui} - \hat{r}_{ui})^2}{C}}$$

Without the root, it is a Mean Square Error, it is equivalent to rank. Practically, we need to minimize the RMSE or MSE so that to minimize the recommendation error.

What is special with Netflix?

Recommendation mechanism: 10% Cinematch (before 2006) & $1M and 100 Million data points. (Big, sparse, screwed data)

Now predict the value of '?'s.

Movies

Users	1	2	3	4	5	6	7	8
1		5		2	4			
2	4		3	1			3	
3		5	4		5		4	
4						1	1	2
5	3		?		?	3		
6		?	2		4		?	

The mark given by the user might depend on type of movies and preference of users.

Large and sparse data:

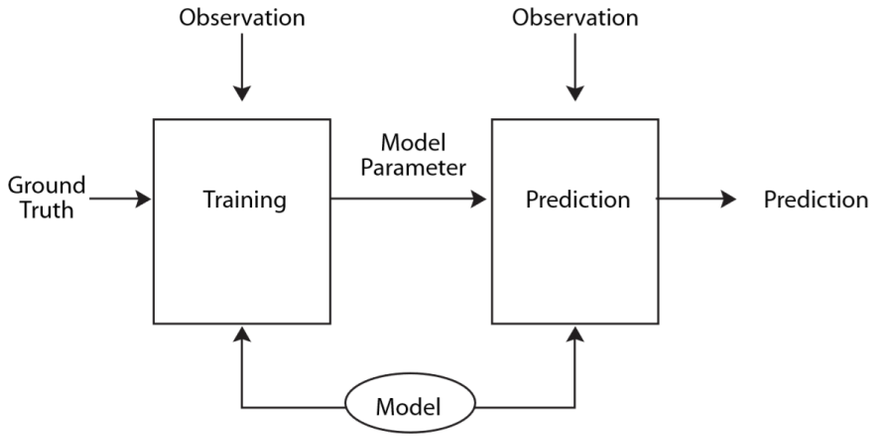
Content-based filter: look at each row or each column in isolation

Collaborative filter: jointly look at rows and columns (global structure)

Neighborhood method (today's lecture)

Latent factor method (Advanced materials)

Parameterized Models:



Average Predictor (over all existing ratings of movies and users)

$$\bar{r} = \frac{\sum_{(u,i)} r_{ui}}{C}$$

Baseline Predictor:

$$\hat{r}_{ui} = \bar{r} + b_u + b_i$$

b_u: bias of user u

b_i: bias of movie i

Why use the summation not the multiplication: minimize the calculation, so that only need to calculate 500k+80k not 500k*800k data for a pair of $\hat{r}_{ui}$.

RMSE => MSE => SE:

$$\text{minimize}_{\{b_u, b_i, \forall u,i\}} \sum_{(u,i)} (\hat{r}_{ui} - r_{ui})^2$$

test probe set: ~1.4M test the accuracy.

Example: 1 user (user 1) and two movies (A, B).

$$\bar{r} = \frac{1}{2}(r_{1A} + r_{1B})$$

$$\min_{\{b_1, b_A, b_B\}} (\bar{r} + b_1 + b_A - r_{1A})^2 + (\bar{r} + b_1 + b_B - r_{1B})^2$$

L-2 norm of a vector: $\left|\left|\vec{x}\right|\right| = \sqrt{\sum_i x_i^2}$

The problem becomes to be:

$$\min_b \left|\left|\mathbf{A}\vec{b} - \vec{c}\right|\right|_2^2$$

Notice that all vectors are column vectors.

$$\left|\left|\mathbf{A}\vec{b}-\vec{c}\right|\right|_2^2=\left|\left|\begin{bmatrix}1 & 1 & 0\\ 1 & 0 & 1\end{bmatrix}\begin{bmatrix}b_1\\ b_A\\ b_B\end{bmatrix}-\begin{bmatrix}r_{1A}-\bar{r}\\ r_{1B}-\bar{r}\end{bmatrix}\right|\right|_2^2=\left|\left|\begin{bmatrix}b_1+b_A+\bar{r}-r_{1A}\\ b_1+b_B+\bar{r}-r_{1B}\end{bmatrix}\right|\right|_2^2$$

Least squares:

$$\left|\left|\mathbf{A}\vec{b}-\vec{c}\right|\right|_2^2=\left(\mathbf{A}\vec{b}-\vec{c}\right)^{\mathrm{T}}\left(\mathbf{A}\vec{b}-\vec{c}\right)$$

Taking derivative with respect to $\vec{b}$, and setting the derivative to $\vec{0}$:

$$2(\mathbf{A}^{\mathrm{T}}\mathbf{A})\vec{b}-2\mathbf{A}^{\mathrm{T}}\vec{c}=\vec{0}$$

This leads to the following linear equations:

$$(\mathbf{A}^{\mathrm{T}}\mathbf{A})\vec{b}=\mathbf{A}^{\mathrm{T}}\vec{c}$$

Solving the above equations to obtain $\vec{b}^{*}$ that minimizes the (RM)SE.

After baseline predictor:

Error term:

$$\tilde{r}_{ui}=r_{ui}-\hat{r}_{ui}=r_{ui}-(\bar{r}+b_u+b_i)$$

Error matrix:

$$\widetilde{R}=R-\widehat{R}$$

Neighborhood method of movies: the same categories of movies.

After the calculation above, the data will be the input for the neighborhood method.

(About Neighborhood method, we will discuss tomorrow)

Convex Optimization (majority of Steven Boyd)

convex set: the line connected two points in the set will be located in the whole set.

nonconvex set: the line connected two points in the set might not totally in the set.

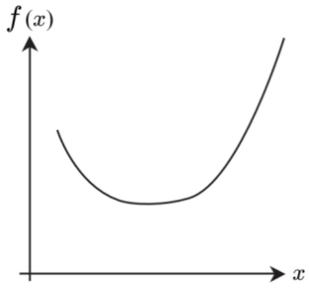
If a set C is convex: then for every point in the set ($\vec{a},\vec{b}$ can be generalized as vectors), there exist $\theta\in[0,1]$ that $\theta\vec{a}+(1-\theta)\vec{b}\in C$.

convex function: when we say convex function: $f''(x)\geq 0$

(There is no direct relationship between convex set and convex function.)

If turn a function upside down, it's a convex function, says, $-f(x)$ is a convex

function, then we call it concave function.



Hessian Matrix of a multi-variable function f(x) is: $[\frac{\partial^2 f(x)}{\partial x_i \, \partial x_j}]_{i,j}$

Function f(x) is convex if its Hessian matrix is positive semi-definite (that is, its all eigenvalues are non-negative).

If we have a convex function, find its minimum value is extremely simple. find out the point $f'(x) = 0$, $f(x)_{min}$ can be calculated.

Minimize convex objective function is important.

When calculating the least square, take the derivative of matrix A is the same method with convex function.

(Pay attention that to determine a convex function, not only the shape but also the domain is relative.)

Review:

Average predictor (over all existing ratings of movies and users) (Training)

$$\bar{r} = \frac{\sum_{(u,i)} r_{ui}}{C}$$

Baseline predictor (Training)

$$\hat{r}_{ui} = \bar{r} + b_u + b_i$$

b_u: bias of user u

b_i: bias of movie i

Next, minimization:

$$\text{minmize}_{\{b_u, b_i, \forall u,i\}} \sum_{(u,i)} (\hat{r}_{ui} - r_{ui})^2$$

⇔solving problem:

$$\text{minimize}_{\vec{b}} \left|\left|\mathbf{A}\vec{b} - \vec{c}\right|\right|_2^2$$

A (#test data point)

(#of varilab)

To adjust the categories of movies:

After baseline predictor:

error term: $\tilde{r}_{ui} = r_{ui} - \hat{r}_{ui} = r_{ui} - (\bar{r} + b_u + b_i)$

error matrix: $\widetilde{R} = R - \widehat{R}$

How to understand the similarities between two movies?

consider the error terms of two users and two movies: $\tilde{r}_A,\ \tilde{r}_B$:

$$\tilde{r}_A = (0.5, 1),\ \tilde{r}_B = (-1, -1)$$

The **similarity** of these two movies can be determined by the angle $\cos\theta$:

$$d_{AB} = \frac{\tilde{r}_A \cdot \tilde{r}_B}{\left|\left|\tilde{r}_A\right|\right|_2 \left|\left|\tilde{r}_B\right|\right|_2} = \frac{\sum_u \tilde{r}_{uA}\, \tilde{r}_{uB}}{\sqrt{\sum_u \tilde{r}_{uA}^2}\sqrt{\sum_u \tilde{r}_{uB}^2}}$$

$$d_{AB} = \frac{0.5 \times (-1) + 1 \times (-1)}{\sqrt{0.5^2 + 1^2}\sqrt{(-1)^2 + (-1)^2}} = -0.949$$

Compute an M×M matrix D, where d_{ij} is computed as:

$$d_{ij} = \frac{\tilde{r}_i \cdot \tilde{r}_j}{\left|\left|\tilde{r}_i\right|\right|_2 \left|\left|\tilde{r}_j\right|\right|_2} = \frac{\sum_u \tilde{r}_{uA}\, \tilde{r}_{uB}}{\sqrt{\sum_u \tilde{r}_{uA}^2}\sqrt{\sum_u \tilde{r}_{uB}^2}}$$

If there are three movies:

$$D_{3\times 3} = \begin{bmatrix} - & d_{12} & d_{13} \\ d_{21} & - & d_{23} \\ d_{31} & d_{32} & - \end{bmatrix}$$

Pick the top L movies as movie i's neighborhood: $\mathcal{L}_i$

⋆ L is a system design parameter

Neighborhood Adjustment:

$$\frac{\sum_{j\in\mathcal{L}_i} d_{ij}\tilde{r}_{uj}}{\sum_{j\in\mathcal{L}_i} |d_{ij}|}$$

Neiborhood Predictor = Baseline Predictor + Neighborhood Adjustment

$$\hat{r}_{ui}^{N} = (\bar{r} + b_u + b_i) + \frac{\sum_{j \in \mathcal{L}_i} d_{ij}\tilde{r}_{uj}}{\sum_{j \in \mathcal{L}_i} |d_{ij}|}$$

Briefly summary:

1) Train baseline predictor through (RM)SE minimization

$$\hat{r}_{ui} = \bar{r} + b_u + b_i \Leftrightarrow \widehat{\mathbf{R}} = [\hat{r}_{ui}]$$

2) Obtain error terms from the baseline prediction

$$\tilde{r}_{ui} = r_{ui} - \hat{r}_{ui} \Leftrightarrow \widehat{\mathbf{R}} = [\hat{r}_{ui}]$$

3) Compute similarity matrix among movies

$$d_{ij} = \frac{\tilde{r}_i \cdot \tilde{r}_j}{||\tilde{r}_i||_2 ||\tilde{r}_j||_2} \Leftrightarrow \mathbf{D} = [d_{ij}]$$

4) Define neighborhood for each movie

$$\mathcal{L}_i, \forall i$$

5) Obtain neighborhood prediction

$\hat{r}_{ui}^{N} = (\bar{r} + b_u + b_i) + \frac{\sum_{j \in \mathcal{L}_i} d_{ij}\tilde{r}_{uj}}{\sum_{j \in \mathcal{L}_i} |d_{ij}|} \Rightarrow \widehat{\mathbf{R}}^{N} = [\hat{r}_{ui}^{N}]$

$\sum_{j \in \mathcal{L}_i} d_{ij}\tilde{r}_{uj}$ term may have 1 to 5 terms so that when calculating, we need to truncate the data.

Chapter 5: When can I trust an average rating on Amazon?

Recommendation algorithm:

Google PageRank: a common list for all users

Netflix collaborative filtering: a separate list for each user

Amazon rating: aggregation of a vector of rating scores into a scalar

How to trust/not to trust the average ratings? Challenges:

Gatekeeper: who can enter the reviews? Only buyers can reach the review sections.

Scale: 1 – 7 or -5 – 5? Five choices are more friendly for the people.

Number of reviews: How big is big enough? Hundreds or Thousands.

How do you compare across products? Price/scores ranking/…

Galton's experiment: Gather 787 people in a farm in UK to guess the ox's weight in 1906. The average guess is 1197 pounds and the actual weight is 1198 pounds. Reason

for the crowd is so wise that they can guess the weight within 1% error without the help of any experts: 3 key factors: 1) The task has a correct and objective answer; 2) Unbiased (some people estimate for higher value and some people estimate for a lower value so that the average is close to the truth) and independent estimates (without communications with each other); 3) Enough people participating.

Questions: When/why can we trust an average rating? & How to rank different products based on their ratings?

Use Galton's experiment as an example:

- N users
- x correct but unknown answers
- user i's has an estimation: $y_i(x) = x + \varepsilon_i(x)$

  Unbiased error: $E_x[\varepsilon_i(x)] = 0$

  Independent: $\varepsilon_i(x)$ is independent with $\varepsilon_j(x)$ for $\forall i \neq j$

Average of errors:

- user i's error: $\varepsilon_i(x)$
- MSE (Mean Square Error): $E_x[\varepsilon_i^2(x)]$
- expected average of errors (AE):

$$E_{AE} = \frac{1}{N}\sum_{i=1}^{N} E_x[\varepsilon_i^2(x)]$$

- expected error of the average (EA):

$$E_{EA} = \frac{1}{N^2} E_x\left[\left(\sum_{i=1}^{N} \varepsilon_i(x)\right)^2\right]$$

Compare AE and EA:

Take an example of two users:

$$E_x\left[\left(\varepsilon_1(x) + \varepsilon_2(x)\right)^2\right] = E_x[\varepsilon_1(x)^2 + \varepsilon_2(x)^2 + 2\varepsilon_1(x)\varepsilon_2(x)] = E_x[\varepsilon_1(x)^2] + E_x[\varepsilon_2(x)^2]$$

- N users

- x correct but unknown answers

- user i's has an estimation: $y_i(x) = x + \varepsilon_i(x)$

Average of errors (AE):

$$E_{AE} = \frac{1}{N}\sum_{i=1}^{N} E_x[\varepsilon_i^2(x)]$$

Error of average (EA):

$$E_{EA} = \frac{1}{N^2} E_x\left[\left(\sum_{i=1}^{N} \varepsilon_i(x)\right)^2\right]$$

Combination between EA and AE:

$$E_x\left[\left(\sum_{i=1}^{N} \varepsilon_i(x)\right)^2\right] = \sum_{i=1}^{N} E_x[\varepsilon_i^2(x)]$$

Hence

$$E_{EA} = \frac{1}{N} E_{AE}$$

Conclusion: Averaging the estimation over N users reducing the error to $\frac{1}{N}$. The error of estimation will be small if the population of crowd is as large as possible. – wisdom of crowds.

Consider the extreme case: What if all estimations are dependent, says, all estimations are the same?

$$y_i(x) = y_j(x)$$

The average of the estimation will be the same as each estimation:

$$E_{EA} = E_{AE}$$

Partial dependence:

$$\frac{1}{N} E_{AE} \le E_{EA} \le E_{AE}$$

[Bayesian estimation: $P(A|B) = \frac{P(B|A)\cdot P(A)}{P(B)}$]

Question 1: If you have performed an experiment n times, and you have observed the outcome of "1" s times. What is the probability of seeing an outcome of 1 in the next experiment?

Question 2: If you are tossing a coin, with head meaning “1” and tail meaning “0”. If you have seen s times of “1” out of n experiments, what is the probability of seeing “1” in the next toss?

If we know the probability p, then we can predict the probability of next outcome. Conversely, if we have observed the outcomes for many times, how to estimate the probability p?

Bayesian ranking

We have several brands of products, indexed by i;

For each brand i: a total of n_i reviews, with an average rating of r_i;

For all brands: a total of N reviews, with an average rating of $R = \frac{\sum_i r_i n_i}{\sum_i n_i}$;

Bayesian rating of Brand i:

$$\tilde{r_i} = \frac{NR + r_i n_i}{N + n_i}$$

sometimes use the formula:

$$\tilde{r}_i = \frac{N_{min} R + r_i n_i}{N_{min} + n_i}$$

MacBook	No. Ratings	Ave. Rating	Rank	Bayesian Rating	Bayesian Rank
MB991LL	10	4.920	1	4.436	2
MB403LL	15	4.667	2	4.433	3
MB402LL	228	4.535	3	4.459	1
MC204LL	150	4.310	4	4.401	5
MB061LL	124	4.298	5	4.402	4

The Bayesian rating is different with the average rating because the formula of Bayesian rating considers the number of reviews. The third one comes to the top because the first one and the second one has far fewer reviews.

Key factors:

- Bayesian ranking
- Too few or too outdated reviews penalized
- Very high-quality reviews help a lot
- Major issues push ranking down a lot

Summary:

Average ratings scalarizes a vector and ranks

Average: factor of N multiplexing gain in wisdom of crowds, following independent and unbiased individual inputs

Ranking: number of ratings should matter, as in Bayesian ranking

- $G = (V, E)$
  - V: set of nodes (indexed by i)
  - E: set of links (in the form of (i, j))
- **Directed** graph: $(i, j) \in E$ does not imply that $(j, i) \in E$
- **Undirected** graph: $(i, j) \in E$ implies that $(j, i) \in E$ (each link is bidirectional)
- We only consider **simple** and **connected** graphs
  - Simple: each link only connects two nodes
  - Connected: there is no "disconnected" node

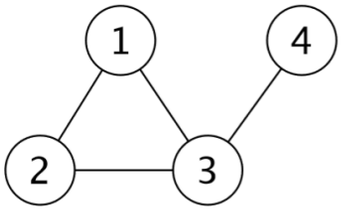
Adjacency matrix:

$\boldsymbol{A}$: $N \times N$ matrix

$$A_{ij} = \begin{cases} 1 & \text{if } i \to j \\ 0 & \text{otherwise} \end{cases}$$

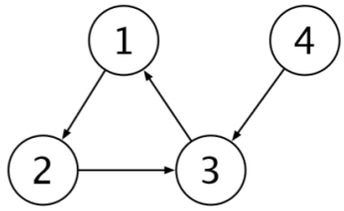
A directed adjacency matrix is always symmetric and an undirected adjacency matrix is not.

Example:



$$\boldsymbol{A} = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Undirected Graph
(Symmetric matrix)



$$\boldsymbol{A} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Directed Graph
(Asymmetric matrix)

Notice that A must be square matrix

Case 1: small server: $u_s \le \frac{u_s + \sum_i u_i}{N}$

For tree j: $r_j = \frac{u_j}{\sum_i u_i} u_s$

Feasibility:

① server: $\sum_j r_j = \sum_j \frac{u_j}{\sum_i u_i} u_s = \frac{\sum_j u_j}{\sum_i u_i} u_s = u_s$

② each node j: $(N-1) r_j = \frac{(N-1) u_j}{\sum_i u_i} u_s \le u_j$

since $u_s \le \frac{u_s + \sum_i u_i}{N} \Rightarrow (N-1) u_s \le \sum_i u_i \Rightarrow \frac{u_s (N-1)}{\sum_i u_i} \le 1$

P2P: upload: $u_s + \sum_i u_i$

$T = \max\left\{\frac{1}{u_s}, \frac{1}{u_s + \sum_i u_i}\right\}$ maximum duration rate:

$r_{max} = \frac{1}{T} = \min\left\{u_s, \frac{u_s + \sum_i u_i}{N}\right\}$

Case 2: powerful server: $u_s > \frac{u_s + \sum_i u_i}{N}$

For 2-hop tree j: $r_j = \frac{u_j}{N-1}$ For a one-hop tree: $r_{N+1} = \frac{u_s - \sum_j \frac{u_j}{N-1}}{N}$

Achievability: node j total rate:

$$r_j + \sum_{i \ne j} r_i + r_{N+1} = \frac{u_j}{N+1} + \sum_{i \ne j} \frac{u_i}{N-1} + \frac{u_s - \sum_j \frac{u_j}{N-1}}{N}$$

① server

② each node j

$$= \sum_i \frac{u_i}{N-1} + \frac{u_s - \sum_i \frac{u_i}{N-1}}{N}$$

$$= \frac{u_s}{N} + \sum_i u_i \left(\frac{N}{(N-1)N} - \frac{1}{(N-1)N} \right) = \frac{u_s + \sum_i u_i}{N}$$

chapter 18

802.11 family

1999 802.11b	0.4GHz	2Mbps 5 11Mbps
2000 → 11a	5GHz	54Mbps
2003 → 11g	2.4GHz	54Mbps
2006 → 11n	5GHz/2.4GHz	
2012 → 11ac	1.71Gbps → 7Gbps	

6Mbps ADSL (Telephone line)

(20Mbps

- TV 4Mbps
- Internet: 2Mbps
- Backend:
  - 0.5Mbps
  - 1.5Mbps