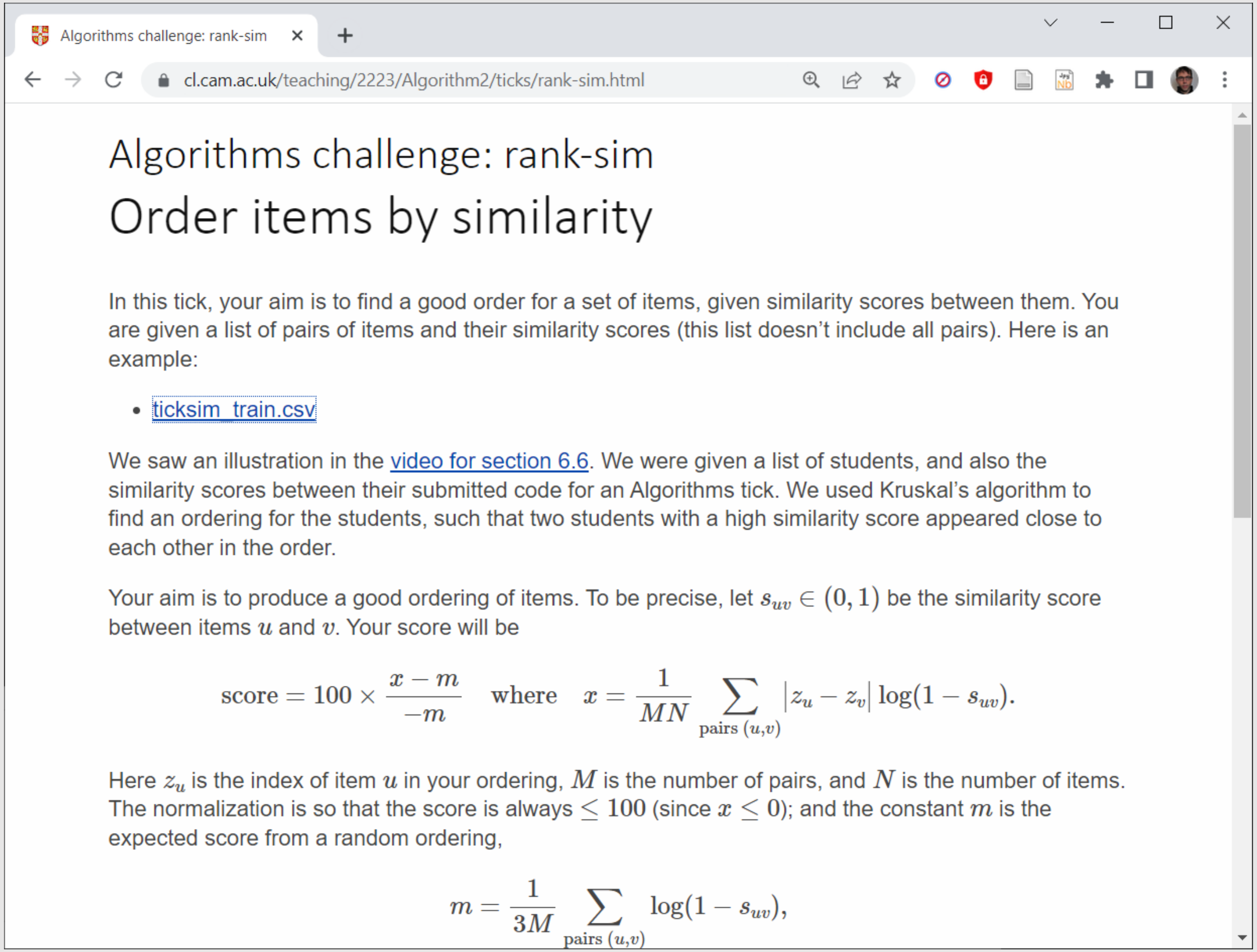


Algorithms challenge: rank-sim

Order items by similarity

In this tick, your aim is to find a good order for a set of items, given similarity scores between them. You are given a list of pairs of items and their similarity scores (this list doesn't include all pairs). Here is an example:

- ticksim_train.csv

We saw an illustration in the video for section 6.6. We were given a list of students, and also the similarity scores between their submitted code for an Algorithms tick. We used Kruskal's algorithm to find an ordering for the students, such that two students with a high similarity score appeared close to each other in the order.

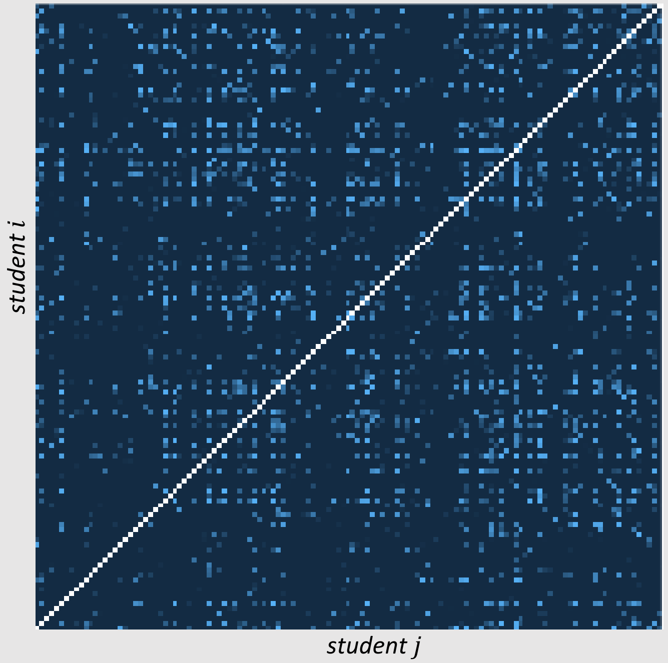
Your aim is to produce a good ordering of items. To be precise, let $s_{uv} \in (0,1)$ be the similarity score between items u and v. Your score will be

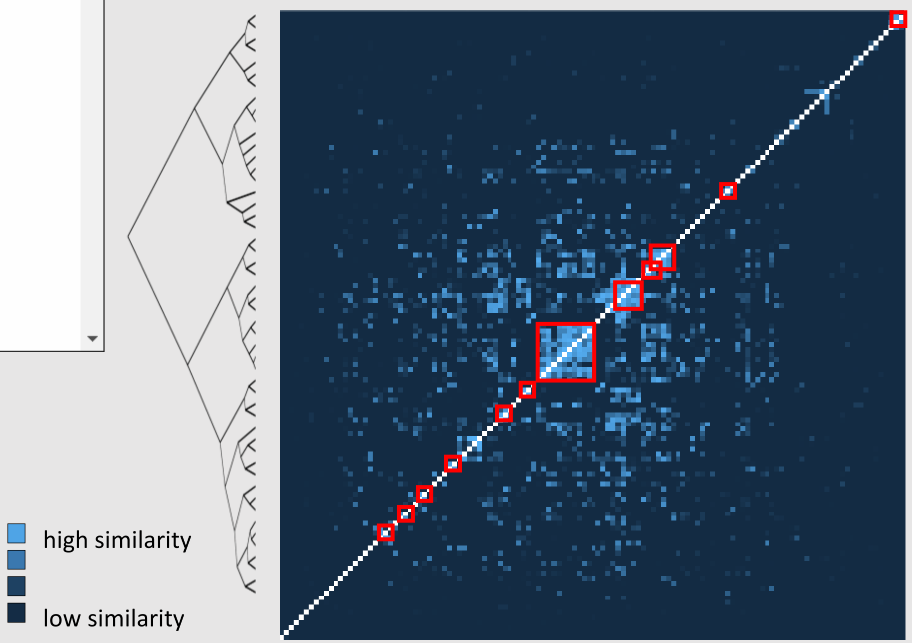
$$\text{score} = 100 \times \frac{x-m}{-m} \quad \text{where} \quad x = \frac{1}{MN} \sum_{\text{pairs }(u,v)} |z_u - z_v| \log(1 - s_{uv}).$$

Here z_u is the index of item u in your ordering, M is the number of pairs, and N is the number of items. The normalization is so that the score is always ≤ 100 (since $x \leq 0$); and the constant m is the expected score from a random ordering,

$$m = \frac{1}{3M} \sum_{\text{pairs }(u,v)} \log(1 - s_{uv}),$$

Similarity matrix of max-flow code





	score on training data (2021 tick1)	score on holdout data (2023 max-flow)
Matej Urban (Trinity)	76.39	58.20
Kevin Xie (Downing)	76.46	57.39
Reuben Carolan (Churchill)	76.85	57.36
Mario Pariona Molocho (Trinity)	71.76	57.25
Wei Chuen Sin (Hughes Hall)		57.13
Leo Takashige (Trinity)	75.50	53.55
Paul DSouza (Robinson)	71.12	48.75
Ugo Obudulu (Churchill)	71.88	47.20
Katy Thackray (New Hall)	67.50	38.26
Milos Puric (Trinity)	65.41	33.97
Leonard Ong (Hughes Hall)	65.41	33.97
Elizabeth Ho (Trinity)	64.98	32.52
Dhruv Pattem (Trinity)	31.50	22.90
Max Bowman (Robinson)	0.51	1.06
George Ogden (Trinity)	76.77	0.00

The Greatest Algorithm Anybody has ever Written

Reuben Carolan

Optimization Problem

- We have function.
- We want inputs that give function biggest possible value.

- In this case find an order of the list to maximise:

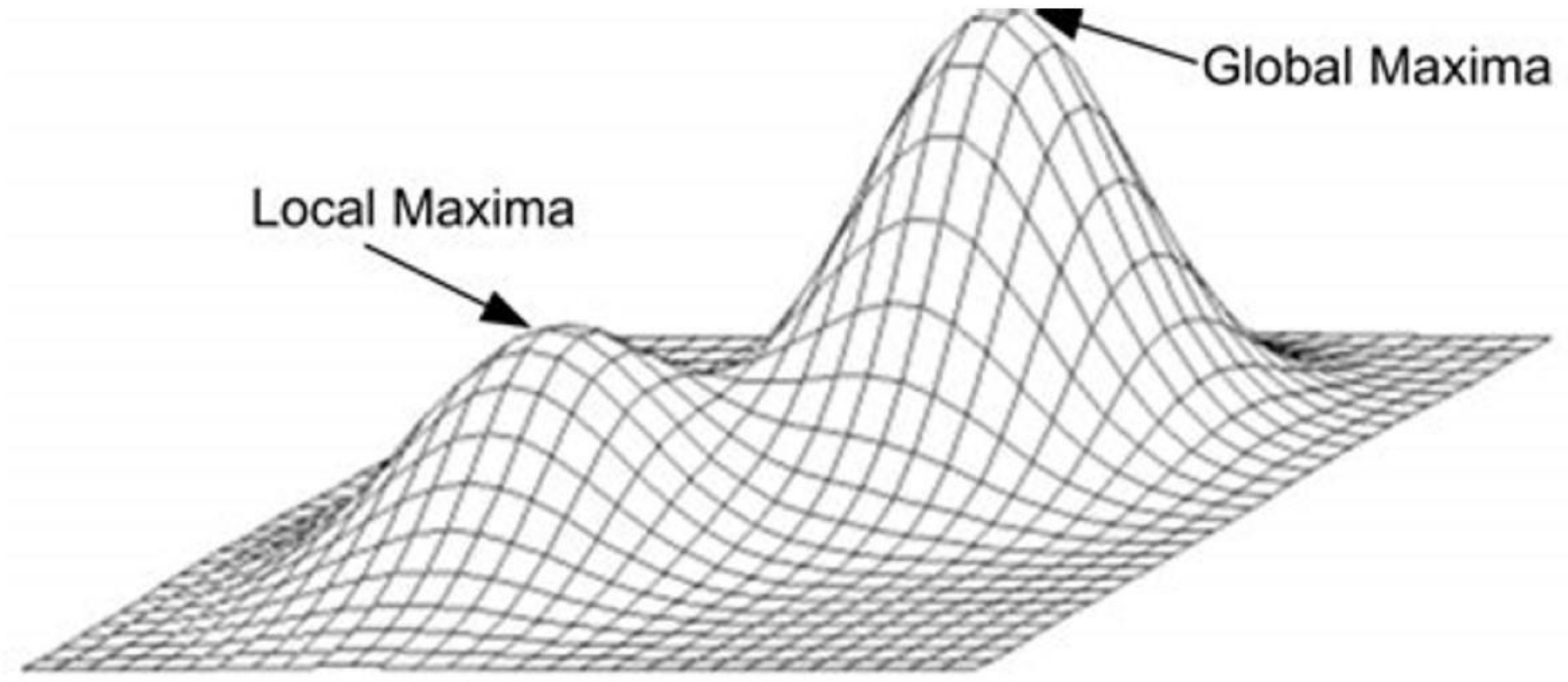
$$\sum_{\text{pairs }(u,v)} \left|z_u - z_v\right| \log(1 - s_{uv}).$$

Solution Space

- Solutions have neighbours. In this case solutions are neighbours if they are the same but with two items swapped places.

- We could naively randomly pick a neighbour and move to that solution if it scores better.

Solution Space



Simulated Annealing

- We define a Temperature value that decreases over time.

- And for each potential solution an accept probability:

```
accept_p = np.exp(( newscore - oldscore) / curr_temperature)
```

- We Accept a new solution is better if:

(New solution is better OR a random number from [0,1]<accept_p)

Why Simulated Annealing

- Idk ¯_(ツ)_/¯
- Its good when the solution space is huge but has a small diameter. (In this case we are only ever 152 swaps from the perfect solution).
- Its good at getting a ‘pretty good’ solution in a short amount of time.
- It works well on the Travelling Salesman problem which is another problem of ordering a list.

Improvements and Other Algorithms

- Restarting
- Better Candidate Generation
- Ant Colony Optimisation

SECTION X

Optimization algorithms

Non-examinable.

Max-flow

Given a directed graph $g = (V, E)$ in which each edge $u \to v$ has a capacity c_{uv}, and given a source s and a sink t, find a flow from s to t with maximum possible value.

Let f_{uv} = flow on edge $u \to v$.

We want to maximize flow value = net flow out of s = $\sum_{u: s \to u} f_{su} - \sum_{w: w \to s} f_{ws}$

For it to be a valid flow, we need $0 \le f_{uv} \le c_{uv}$ for all edges,

and flow conservation at every vertex $v \in V \setminus \{s,t\}$, ie $\sum_{u: v \to u} f_{vu} - \sum_{w: w \to v} f_{wv} = 0 \quad \forall v \in V \setminus \{s,t\}$.

Restate the problem:

maximize $\sum_{e \in E} b_e f_e$ where $b_e = \begin{cases} 1 & \text{if edge } e \text{ starts at } s \\ -1 & \text{if edge } e \text{ ends at } s \\ 0 & \text{otherwise} \end{cases}$

over $f \in \mathbb{R}^E$, $0 \le f \le c$ ← 0 and c are vectors in $\mathbb{R}^E$, and the inequality must hold elementwise

such that $\underbrace{\sum_{e \in E} A_{ve} f_e = 0 \quad \forall v \in V \setminus \{s,t\}}_{\text{we can rewrite as a matrix equation}}$ where $A_{ve} = \begin{cases} 1 & \text{if edge } e \text{ starts at } v \\ -1 & \text{if edge } e \text{ ends at } v \\ 0 & \text{otherwise} \end{cases}$

$$Af = 0$$

$$\underbrace{|V|-2}_{\text{rows}} \left\{ \overbrace{\begin{bmatrix} & A & \end{bmatrix}}^{|E| \text{ cols}} \right. \begin{bmatrix} f \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$$

```python
import pandas
import numpy as np
df = pandas.read_csv('flownetwork_02.csv')
df
```

	u	v	capacity
0	0	1	10000
1	1	2	10000
2	2	5	10000
3	0	3	8000
4	3	2	8000
5	1	4	6000
6	4	5	6000

```python
s,t = 0,5
V = set(df.u) | set(df.v)
b = np.where(df.u==s,1,0) - np.where(df.v==s,1,0)
A = np.row_stack([np.where(df.u==v,1,0) - np.where(df.v==v,1,0)
                  for v in V if v not in {s,t}])

import scipy.optimize
res = scipy.optimize.linprog(
       -b,
       bounds=np.column_stack([np.zeros(len(df)), df.capacity]),
       A_eq=A, b_eq=np.zeros(len(A))
       )
df['flow'] = res.x
```

minimize $-b \cdot x$
over $x \in \mathbb{R}^n, 0 \leq x \leq \text{capacity}$
such that $A_{\text{eq}} x = b_{\text{eq}}$

Minimum spanning tree

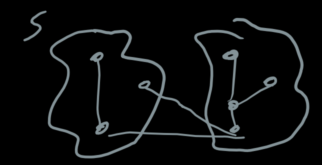
Given a connected undirected graph where the weight of edge $u — v$ is c_{uv}, find a spanning tree of minimum weight.

we want to find a subgraph, ie a subset of edges from the original graph.

Let $x_{uv} = \begin{cases} 1 & \text{if edge } u-v \text{ is in our subgraph} \\ 0 & \text{otherwise.} \end{cases}$ $\qquad x \in \{0,1\}^E$

The weight is $\displaystyle\sum_{u-v} x_{uv} c_{uv}$

For x to be spanning: for every non-trivial partition of the vertices $V = S \cup \bar{S}$, $S \neq \emptyset$, $S \neq V$,
there must be at least one edge between them.



Thus x spanning $\Rightarrow \displaystyle\sum_{v \in S, w \in \bar{S}} x_{vw} \geq 1 \quad \forall$ non-trivial partitions $(S, \bar{S})$.

EXERCISE: show that if x is not spanning then $\exists$ partition $(S,\bar{S})$ with $\displaystyle\sum_{v \in S, w \in \bar{S}} x_{vw} = 0$.

For x to be a spanning tree: we need the spanning condition, and $\displaystyle\sum_e x_e = |V| - 1$ (EXERCISE).

Restate the problem:

minimize $\displaystyle\sum_e x_e c_e$

over $x \in \{0,1\}^E$

such that $\displaystyle\sum_e x_e = |V|-1$ and $\displaystyle\sum_e A_{Se} x_e \geq 1$ for all non-triv. partitions $(S,\bar{S})$.

This is too hard! Integer constraints are tough! Let's relax the problem, and simply require $x \in \mathbb{R}^E$, $0 \leq x \leq 1$. Interpret x_e as a measure of "how much we want edge e".

What other problems from this course can be written as optimization problems?
as linear programs?

problems of the form

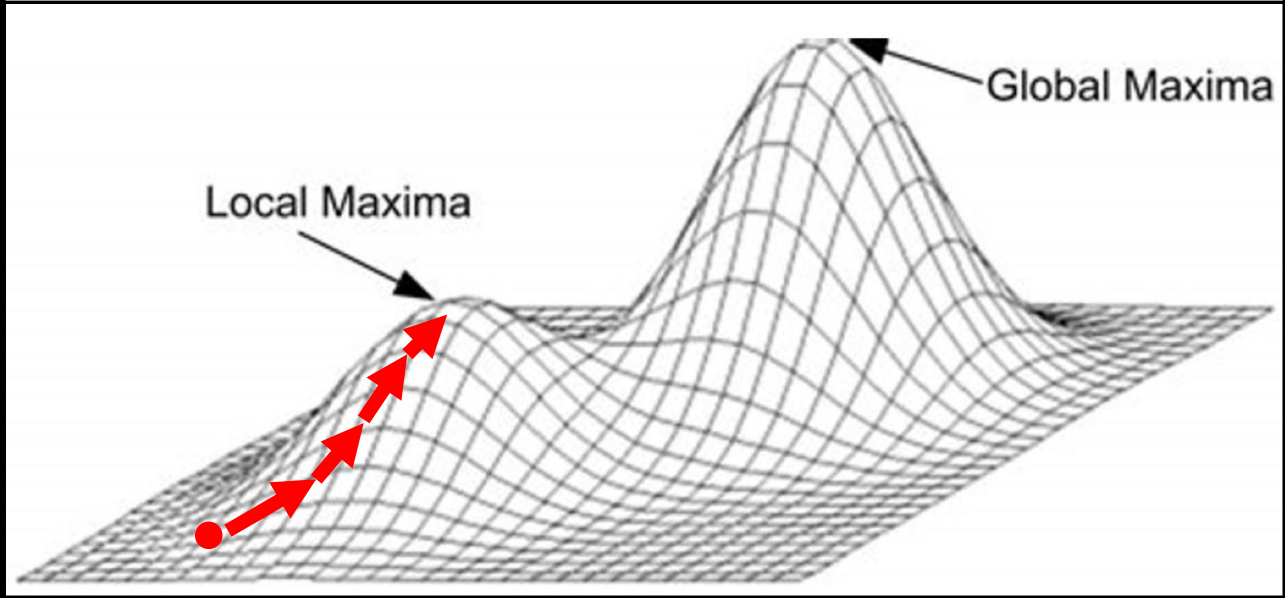
maximize $\quad c \cdot x$

over $\quad x \in \mathbb{R}^n, \quad \underbrace{l \leq x \leq u}_{\text{optional}}$

such that $\quad Ax = b$

$\quad A'x \leq b'$


Andrej Karpathy
@karpathy
Gradient descent can write code better than you. I'm sorry.
3:56 PM - 4 Aug 2017
343 Retweets 1,161 Likes
72
343
1.2K


Global Maxima
Local Maxima

- Can I express my task as an optimization problem?
- … that can be solved with off-the-shelf optimizers?
- If I can’t, is there an adjacent problem that’s more amenable?

Algorithms challenge: rank-sim

Order items by similarity

In this tick, your aim is to find a good order for a set of items, given similarity scores between them. You are given a list of pairs of items and their similarity scores (this list doesn't include all pairs). Here is an example:

- ticksim_train.csv

We saw an illustration in the video for section 6.6. We were given a list of students, and also the similarity scores between their submitted code for an Algorithms tick. We used Kruskal's algorithm to find an ordering for the students, such that two students with a high similarity score appeared close to each other in the order.

Your aim is to produce a good ordering of items. To be precise, let $s_{uv} \in (0,1)$ be the similarity score between items u and v. Your score will be

$$\text{score} = 100 \times \frac{x-m}{-m} \quad \text{where} \quad x = \frac{1}{MN} \sum_{\text{pairs }(u,v)} |z_u - z_v| \log(1 - s_{uv}).$$

Here z_u is the index of item u in your ordering, M is the number of pairs, and N is the number of items.

In this tick, your aim is to find a good order for a set of items, given similarity scores between them. You are given a list of pairs of items and their similarity scores (this list doesn't include all pairs). Here is an example:

- ticksim_train.csv

We saw an illustration in the video for section 6.6. We were given a list of students, and also the similarity scores between their submitted code for an Algorithms tick. We used Kruskal's algorithm to find an ordering for the students, such that two students with a high similarity score appeared close to each other in the order.

Your aim is to produce a good ordering of items. To be precise, let $s_{uv} \in (0,1)$ be the similarity score between items u and v. Your score will be

$$\text{score} = 100 \times \frac{x-m}{-m} \quad \text{where} \quad x = \frac{1}{MN} \sum_{\text{pairs } (u,v)} |z_u - z_v| \log(1 - s_{uv}).$$

Here z_u is the index of item u in your ordering, M is the number of pairs, and N is the number of items.

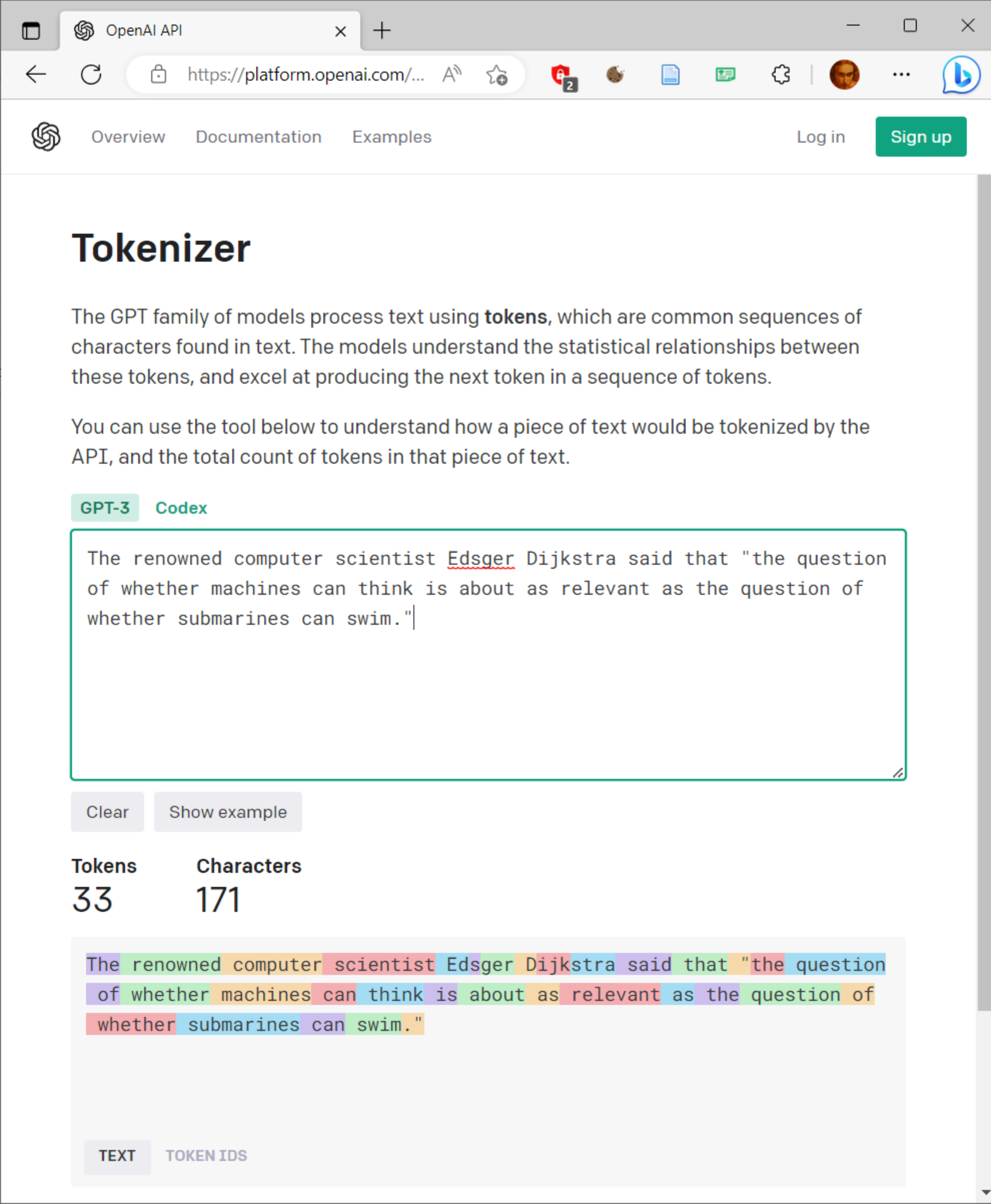
These are called embedding problems. We want to "embed" each student into some other space

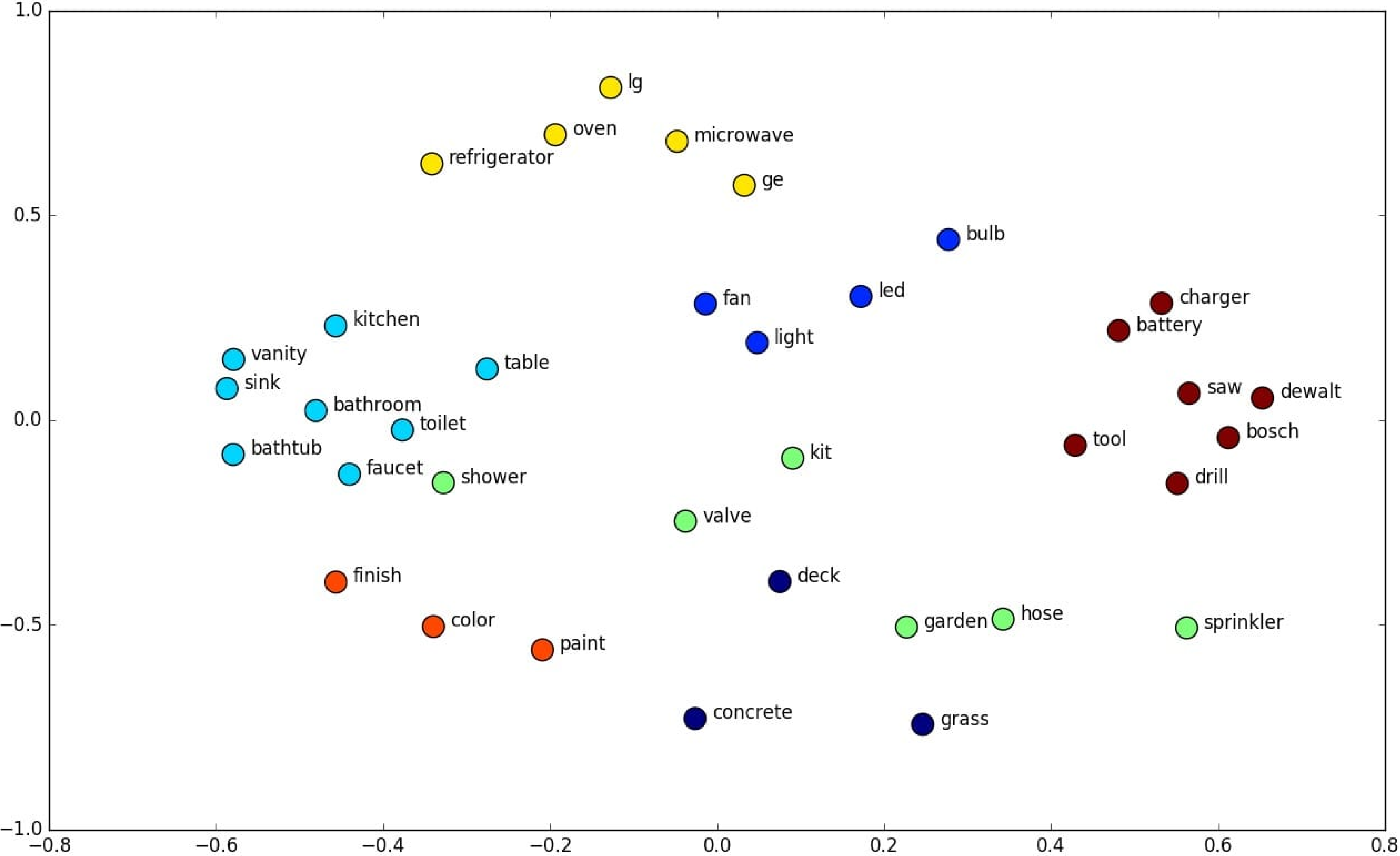
- *either $z_u \in \mathbb{N}$, an integer embedding*
- *or $z_u \in \mathbb{R}$, a real embedding*

The hope is that if two students are similar ($s_{uv} \approx 1$) then they'll end up with similar embeddings ($z_u \approx z_v$).

Related (and much easier) problem:

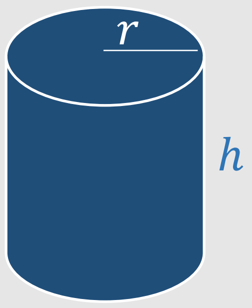
$$\begin{array}{ll} \text{maximize} & \sum_{u,v} |z_u - z_v| \log(1 - s_{uv}) \\ \text{over} & z \in \mathbb{R}^N \\ \text{such that} & \max(z) - \min(z) = N - 1 \end{array}$$


OpenAI API
https://platform.openai.com/...
Overview
Documentation
Examples
Log in
Sign up
Tokenizer
The GPT family of models process text using tokens, which are common sequences of characters found in text. The models understand the statistical relationships between these tokens, and excel at producing the next token in a sequence of tokens.
You can use the tool below to understand how a piece of text would be tokenized by the API, and the total count of tokens in that piece of text.
GPT-3
Codex
The renowned computer scientist Edsger Dijkstra said that "the question of whether machines can think is about as relevant as the question of whether submarines can swim."
Clear
Show example
Tokens
33
Characters
171
The renowned computer scientist Edsger Dijkstra said that "the question of whether machines can think is about as relevant as the question of whether submarines can swim."
TEXT
TOKEN IDS


lg
oven
microwave
refrigerator
ge
bulb
led
fan
charger
battery
light
kitchen
vanity
table
sink
saw
dewalt
bathroom
toilet
tool
bosch
bathtub
kit
faucet
shower
drill
valve
finish
deck
color
garden
hose
sprinkler
paint
concrete
grass
1.0
0.5
0.0
−0.5
−1.0
−0.8
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6
0.8

Maths for NST B lecture 14

A right-circular cylinder of radius r and height h has volume $\pi r^2 h$ and surface area $2\pi r^2 + 2\pi r h$. Given the surface area is A, find the largest possible volume.



Maths for NST B lecture 14

A right-circular cylinder of radius r and height h has volume $\pi r^2 h$ and surface area $2\pi r^2 + 2\pi r h$. Given the surface area is A, find the largest possible volume.

maximize	$\pi r^2 h$
over	$r, h \in \mathbb{R}_{\geq 0}$
such that	$2\pi r^2 + 2\pi r h - \mathrm{A} = 0$

METHOD

1. Write out the Lagrangian

$$\mathcal{L}(r, h; \lambda) = \pi r^2 h - \lambda(2\pi r^2 + 2\pi r h - A)$$

2. For a given λ, find $r \geq 0$ and $h \geq 0$ to maximize $\mathcal{L}(r, h; \lambda)$

3. Choose λ so that these r and h satisfy the constraint

Step 2:

$$\frac{\partial \mathcal{L}}{\partial r} = 2\pi r h - \lambda(4\pi r + 2\pi h) = 0$$

$$\frac{\partial \mathcal{L}}{\partial h} = \pi r^2 - \lambda(2\pi r) = 0$$

$$\Rightarrow \quad h = 4\lambda, \qquad \cancel{r = 0} \text{ or } r = 2\lambda$$

Step 3: $2\pi r^2 + 2\pi r h = A \Rightarrow 8\pi\lambda^2 + 16\pi\lambda^2 = A \Rightarrow \lambda = \sqrt{\frac{A}{24\pi}}$

This gives $V = \pi r^2 h = 16\pi\lambda^3 = 16\pi\left(\frac{A}{24\pi}\right)^{3/2}$

Maths for NST B lecture 14

A right-circular cylinder of radius r and height h has volume $\pi r^2 h$ and surface area $2\pi r^2 + 2\pi rh$. Given the surface area is A, find the largest possible volume.

maximize $\pi r^2 h$
over $r, h \in \mathbb{R}_{\geq 0}$
such that $2\pi r^2 + 2\pi rh - A = 0$

METHOD

1. Write out the Lagrangian

$$\mathcal{L}(r, h; \lambda) = \pi r^2 h - \lambda(2\pi r^2 + 2\pi rh - A)$$

= objective function $-\sum_i \lambda_i$ constraint$_i$

one λ for every constraint equation

2. For a given λ, find $r \geq 0$ and $h \geq 0$ to maximize $\mathcal{L}(r, h; \lambda)$
3. Choose λ so that these r and h satisfy the constraint

Why does this method work?

For every (r, h) such that $\text{Area}(r, h) = A$, and for every $\lambda \in \mathbb{R}$,

$$\begin{aligned}
\text{volume}(r, h) &= \pi r^2 h && \text{by definition of volume} \\
&= \pi r^2 h - \lambda(\text{Area}(r, h) - A) && \text{since Area}(r,h) = A \text{ by assumption} \\
&= \mathcal{L}(r, h; \lambda) && \text{by definition of } \mathcal{L} \\
&\leq \max_{r', h' \geq 0} \mathcal{L}(r', h'; \lambda) && \text{since we can only possibly increase } \mathcal{L} \text{ by maximizing over its first two arguments} \\
&= \lambda A - 8\pi\lambda^3 && \text{This maximization is pretty easy, with simple calculus}
\end{aligned}$$

Maths for NST B lecture 14

A right-circular cylinder of radius r and height h has volume $\pi r^2 h$ and surface area $2\pi r^2 + 2\pi r h$. Given the surface area is A, find the largest possible volume.

maximize	$\pi r^2 h$
over	$r, h \in \mathbb{R}_{\geq 0}$
such that	$2\pi r^2 + 2\pi r h - \mathrm{A} = 0$

METHOD

1. Write out the Lagrangian

$$\mathcal{L}(r, h; \lambda) = \pi r^2 h - \lambda(2\pi r^2 + 2\pi r h - A)$$

2. For a given λ, find $r \geq 0$ and $h \geq 0$ to maximize $\mathcal{L}(r, h; \lambda)$

we'll get answers as functions of λ: $r = r(\lambda)$, $h = h(\lambda)$

3. Choose λ so that these r and h satisfy the constraint

This works in some problems, but not all.
A more general method is "Find λ that minimizes the upper bound".

Why does this method work?

For every (r, h) such that $\text{Area}(r, h) = A$, and for every $\lambda \in \mathbb{R}$,

$$\begin{aligned}
\text{volume}(r, h) &= \pi r^2 h \\
&= \pi r^2 h - \lambda(\text{Area}(r, h) - A) \\
&= \mathcal{L}(r, h; \lambda) \\
&\leq \max_{r', h' \geq 0} \mathcal{L}(r', h'; \lambda) \\
&= \lambda A - 8\pi\lambda^3
\end{aligned}$$

Max-flow

Given a directed graph g in which each edge $u \to v$ has a capacity c_{uv}, and given a source s and a sink t, find a flow from s to t with maximum possible value.

$$\mathcal{L}(f;\lambda) = \text{value}(f) - \sum_{v \in V \setminus \{s,t\}} \lambda_v \left(\text{net flow}(v) - 0\right)$$

For every $\overset{\text{flow } f}{\cancel{(r,h)}}$ such that $\overset{\text{flow is conserved}}{\cancel{\text{Area}(r,h) = A}}$, and for every $\lambda \in \cancel{\mathbb{R}}$, $\mathbb{R}^{V-2}$

$$
\begin{aligned}
\text{value}(f)\ \cancel{\text{volume}(r,h)} &= \cancel{\pi r^2 h} \\
&= \cancel{\pi r^2 h - \lambda(\text{Area}(r,h) - A)} \\
&= \cancel{\mathcal{L}(r,h;\lambda)}\ \mathcal{L}(f;\lambda) \\
&\le \cancel{\max_{r',h' \ge 0} \mathcal{L}(r',h';\lambda)}\ \max_{0 \le f' \le c} \mathcal{L}(f';\lambda) \\
&= \cancel{\lambda A - 8\pi\lambda^3} = \text{capacity}(\lambda)
\end{aligned}
$$

think of λ as a "generalized cut"

The Lagrangian is

$$\mathcal{L}(f;\lambda) = \left(\sum_{u:s\to u} f_{su} - \sum_{w:w\to s} f_{ws}\right) - \sum_{v\neq s,t} \lambda_v \left(\sum_{u:v\to u} f_{vu} - \sum_{w:w\to v} f_{wv}\right)$$

The preceding argument (called “Lagrangian weak duality”) says that for any flow f and for any λ,

$$\text{val}(f) \leq \max_{f':0\leq f'\leq C} \mathcal{L}(f';\lambda)$$

$$\mathcal{L}(f';\lambda) = \sum_{v} \delta_v \left(\sum_{u:v\to u} f'_{vu} - \sum_{w:w\to v} f'_{wv}\right) \text{ where } \delta_v = \begin{cases} 1 & \text{if } v=s \\ 0 & \text{if } v=t \\ \lambda_v & \text{otherwise} \end{cases}$$

$$= \sum_{v,u:v\to u} \delta_v f'_{vu} - \sum_{v,w:w\to v} \delta_w f'_{wv}$$

$$= \sum_{a,b:a\to b} \delta_a f'_{ab} - \sum_{b,a:a\to b} \delta_b f'_{ab}$$

$$= \sum_{a,b:a\to b} f'_{ab}(\delta_a - \delta_b)$$

This is maximized at $f'_{ab} = \begin{cases} c_{ab} & \text{if } \delta_a > \delta_b \\ 0 & \text{if } \delta_a < \delta_b \\ ? & \text{if } \delta_a = \delta_b \end{cases} = \begin{cases} c_{ab} & \text{if } a\in S, b\notin S \\ 0 & \text{if } a\notin S, b\in S \\ ? & \text{if } a,b \text{ on same side} \end{cases}$

Weak duality holds for any λ. Let's consider $\lambda_v = \begin{matrix} 1 \text{ if } v\in S \\ 0 \text{ if } v\notin S \end{matrix}$ for some cut $(S,\bar{S})$

For such a λ, $\max_{0\leq f'\leq C} \mathcal{L}(f';\lambda) = \sum_{a\in S, b\notin S} c_{ab} = \text{capacity}(S,\bar{S})$

There are deep connections between optimization and graph algorithms

- Starting with the Lagrangian, it is sometimes possible to derive fast algorithms, called ‘primal-dual’ algorithms
- Ford–Fulkerson and Kruskal are examples from this general family