We have seen a variety of problems ①
turn into 0,1 linear programs.

Q: which ones are "easy"?

"Easy" = solution of relaxed linear program

$$\parallel$$

soln of 0-1 linear program

0,1 LP	Relaxed LP
min $c^T x$	$c^T x$
st $Ax = b$	$Ax = b$
$x \in \{0,1\}$	$0 \leq x \leq 1$
$Cx \leq d$	$Cx \leq d$.

If this property is true we could
- solve relaxed linear program
- use soln (cause it's 0-1)

(This is hardly ever efficient!)

Mathing in bipartite graphs

A bipartite graph has
- two "types" of vertex (R, G) say.
- all edges go have one R-vertex, one G-vertex

Commonly drawn as two columns, (R, G) and edges go from one side to the other

A matching in a graph is a set of edges without common vertices so in bipartite graph, looks like (say!)

Matchings in bipartite graphs are common in applications (3)

eg DETR (paper on web page)

- Network predicts k (fixed) possible objects
- Decoder must decide for each
  - if it is an object
  - what object
  - where.

Training.
- image has $r \ll k$ boxes, containing known objects
- current state of decoder predicts a score for any object, location

- So we get a <u>weighted bipartite matching problem</u>

training labels, $r=2$

Detr decodes, $k=4$

- we would like a <u>matching</u> (one decode per object)
- that max score
  min ~~weight~~ loss.

- Then we backprop through matching.
- Notice this is indep of the ~~perm~~ order of object boxes, as it should be.

Simplest case:

- n verts on each side (complete bipartite graph).
- in this case, a matching is a permutation
- Write r_i for i'th r vert
  g_i " " g vert

$$r_i \to g_{\pi(i)} \quad \text{is matching}$$

- a permutation is a 0-1 Matrix $M_{n\times n}$ with special properties

$$M_{ij} \in \{0, 1\}$$

$$\sum_i M_{ij} = 1$$

$$\sum_j M_{ij} = 1$$

Max weighted bipartite matching in perfect case

$$\max \sum_{ij} w_{ij} M_{ij}$$

st $M_{ij} \in \{0,1\}$, M_{ij} is $n \times n$

$\sum_i M_{ij} = 1$ $\quad$ $\sum_j M_{ij} = 1$

Recall Birkhoff - Von - Neumann thm

D = set of doubly stochastic matrices

$= \left\{ A_{n\times n} \;\middle|\; \begin{array}{l} 1^T A = 1^T \ , \ A1 = 1 \\ A_{ij} \geq 0 \end{array} \right\}$

= matrices st :
row - sums = 1
col - sums = 1
entries ≥ 0
square

I : D is convex
(easy)

II : D is convex hull of
permutation matrices

III : every vert of D is a perm matrix
AND
every perm matrix is a vert of D.

In turn, relaxed LP is

$$\max \sum_{ij} w_{ij} M_{ij} \quad \left[= \mathrm{Tr}[W^T M] \right]$$

$$\text{s.t.} \quad M \in D.$$

and soln of relaxed $\equiv$ soln of 0-1 problem

- This isn't most efficient alg (later)
-

Now consider general case:

k r-verts, r g-verts, $k \neq r$.

- Matching is a

$k \times r$ matrix M

st

$$\sum_i M_{ij} \leq 1$$

$$\sum_j M_{ij} \leq 1$$

$$M_{ij} \in \{0, 1\}$$

Q: how many 1's in M?

Turn M into vector m by rearranging

We get

$$A\, m \leq 1 \quad , \; m_i \in \{0,1\}$$

for constraints.

<u>Properties of A</u>:

- each <u>col</u> of A corresponds to an entry in M
- each entry in M appears in exactly 2 inequalities (one row, one col).

$\Rightarrow$ every col of A has exactly 2 ones; all others 0.

<u>Thm</u>: every ~~kxk~~ sxs minor of A is either 1, 0, -1

$\equiv$ A is <u>totally unimodular</u> (TUM)

for the moment, assume true
(I'll prove below).

Consequences:

- every vertex of

$$A m \leq 1, \quad 0 \leq m \leq 1$$

is integer (and so 0,1).

Proof:

- there are $k.r$ vars
- $k+r+2kr$ ineqs.
- choose kr ineqs to be equalities
- solve – this gets vertex if soln exists and feasible

$$M = [\cdots, 1, \cdots 0, \cdots w \cdots]$$

coeffs where $M \leq 1$ is active

$0 \leq M$ active

others

So $Bw + v = 1$

integer

Block of A

So $Bw = z$

integer

$$w = B^{-1} z$$

if inverse exists

But $$B^{-1} = \frac{1}{\det B} \cdot \mathrm{Adj}(B)$$

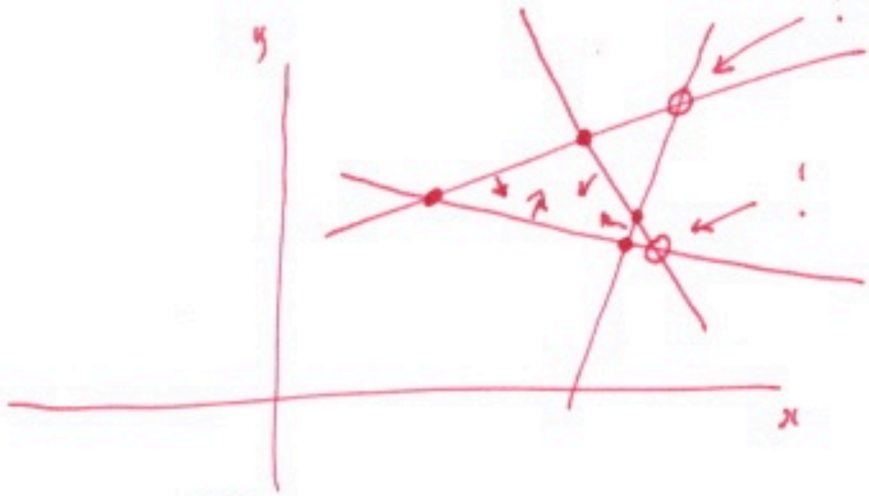
but any minor of A has $\det = \{0, 1, -1\}$

So $\det B \in \{0, 1, -1\}$

$\operatorname{adj} B$ has all entries $\{0, 1, -1\}$

So $w = B^{-1} z$ is integer, if it exists.

· Notice this construction yields more pts than the vertices and they're all integer!



example:

<u>In turn</u>. Solving relaxed LP

$\equiv$

Solving 0.1 LP

in this case.

Thm: A is TUM

Proof: (induction)

Take T a square submatrix of A $(2s \times 2s)$

Three cases:

- some col is all $0 \rightarrow \det T = 0$
- at least one col contains exactly 1 one.
  $\rightarrow$ pass to smaller submatx
- every col contains 2 ones

interesting case

Interesting case:

· consider some col of T.

$$\begin{matrix} 0 \\ \vdots \\ 1 \\ 0 \\ 0 \\ \vdots \\ 1 \\ 0 \end{matrix}$$

← one of these comes from a row constraint on M

← the other from a col constraint.

· now permute the rows of T (which at worst changes sign of det(T)) to get.

} ← s rows from col constraints

} ← s rows from row constraints

Then $[\underbrace{1 \cdots 1}_{s}, \underbrace{-1 \cdots -1}_{s}]\, T = 0$

so $\det(T) = 0$!

Notice this TUM property is very strong.

We have:

$$Ax \leq b$$
$$z_b \leq x \leq z_a$$

integer vectors

has integer verts if A is TUM (showed this already).

Notice

$$Ax = b$$
$$x \geq 0$$

has integer verts if A is TUM and b integer.

In some cases, we can guarrantee an integer soln w/o A being TUM.

consider

$$\max_x \quad \frac{x^T A x}{2} + b^T x$$

$$\text{s.t.} \quad x \in \{0,1\}^n$$

which becomes

$$\max_{x,q} \quad c^T q + b^T x$$

$$\text{s.t.} \quad q \in \{0,1\}^m, \quad x \in \{0,1\}^n$$

and

$$q_{ij} \leq x_i$$

$$q_{ij} \leq x_j$$

$$q_{ij} \geq x_i + x_j - 1$$

relax to

$$\left.\begin{aligned} q_{ij} &\leq x_i \\ q_{ij} &\leq x_j \\ q_{ij} &\leq 1 \\ q_{ij} &\geq 0 \\ x_i &\leq 1 \\ x_i &\geq 0 \\ q_{ij} &\geq x_i + x_j - 1 \end{aligned}\right] \equiv \begin{bmatrix} M \\ Q \end{bmatrix} \begin{bmatrix} x \\ q \end{bmatrix} \leq b$$

Q gets the $1 \geq x_i + x_j - q_{ij}$ constraints

<u>Notice</u>: M is TUM

(Q: can you prove?)

but this means that

as long as Q constraints <u>aren't</u> active, vertices are integer.

<u>Now notice</u> :

$$\max_{x} \quad x^T \frac{A}{2} x + b^T x$$

if $A_{ij} > 0$ then if $x_i^* = 1$, $x_j = 1$,

$q_{ij} = 1$ (because you get a better value of objective function)

This means that if $A_{ij} > 0$, we can ignore Q constraints, so we get integer verts

. This has important application consequences

. recall the foreground - background seg case

$$\max_{x \in \{0,1\}^n} \sum_i \left[U_i(x_i) + \sum_{j \in N(i)} B_{ij}(x_i, x_j) \right]$$

binary value:
- how good is it to have x_i, x_j?

unary value:
- how good is it to label i'th pixel w/ x_i?

i in pixels

U and B come from application considerations

. can transform to:

$$\max_{x \in \{0,1\}^n} \sum_i \Bigg[\left(x_i U_i(1) + (1-x_i) U_i(0) \right) + \sum_{j \in N(i)} \left\{ \begin{array}{l} x_i x_j B_{ij}(1,1) + (1-x_i) x_j B_{ij}(0,1) \\ + x_i (1-x_j) B_{ij}(1,0) + (1-x_i)(1-x_j) B_{ij}(0,0) \end{array} \right. \Bigg.$$

and quadratic term is

$$x_i x_j \left[B_{ij}(1,1) + B_{ij}(0,0) - B_{ij}(0,1) - B_{ij}(1,0) \right]$$

so if $B_{ij}(1,1) + B_{ij}(0,0) - B_{ij}(0,1) - B_{ij}(1,0) > 0$

problem can be relaxed

(= is polynomial)

"It's better to agree than disagree"