Grothendieck's inequality

Thm (Grothendieck's inequality).
Consider an $m \times n$ matrix (a_{ij}) of real numbers.
Suppose that for any $x_i, y_j \in \mathbb{R}$

$$\left|\sum_{ij} a_{ij} x_i y_j\right| \le \max_i |x_i| \max_j |y_j|$$

(i.e. $\forall i j$ multiply the i'th row by x_i, & the j'th column by y_j & sum the entries).
Then for any Hilbert space H and any vectors $u_i, v_j \in H$ we have

$$\left|\sum_{ij} a_{ij} \langle u_i, u_j \rangle\right| \le K \max_i \|u_i\| \cdot \max_j \|v_j\|.$$

$K \le 1.783$ is an absolute constant.

Rmk: While the statement holds for a constant $K \le 1.783$ we will give a proof that gives $K \le 8$.

Pf:
1) Given an $m \times n$ matrix A, let $K = K(A)$ be the smallest K which makes the statement true for every Hilbert space H.
Note that $\tilde{K} = \sum_{ij} |a_{ij}|$ works, so the set of K's that work is not empty.

The key point of the thm is that $K(A)$ in fact does <u>not</u> depend on A, n or m.

2) Given $u_i, v_j \in H$ we need to show we can find $K < 288$ s.t.

$$\left|\sum_{ij} a_{ij} \langle u_i, u_j\rangle\right| \le K \max_i \|u_i\| \max_j \|v_j\| .$$

Once u_i, v_j are selected, the space H does not play any role any more, so we can replace H by its subspace $\tilde{H}$ spanned by all the u_i's & v_j's.

$\tilde{H}$ is of dimension $\le N := m+n$, so it is isometric with a subspace of $\mathbb{R}^N$. Thus, without loss of generality we can assume $H = \mathbb{R}^N$ with the std inner product.

3) We need to bound

$$\left|\sum_{ij} a_{ij} \langle u_i, v_j\rangle\right| .$$

Let's realize $\langle u_i, v_j\rangle$ via random gaussian vectors.

Let $g \sim N(0, I_N)$ & define $U_i = \langle g, u_i\rangle$
$V_j = \langle g, v_j\rangle \quad \forall i,j.$

U_i, V_j are linear combinations of independent mean-zero gaussians, so they are mean-zero gaussians.

Moreover

$$E\, U_i V_j = E\,(\overbrace{u_i^T g}^{\langle g, u_i\rangle}\, \overbrace{g^T v_j}^{\langle g, v_j\rangle}) = u_i^T (E\, g g^T)\, v_j = u_i^T v_j = \langle u_i, v_j\rangle$$

so

$$\sum_{ij} a_{ij} \langle u_i, v_j\rangle = E\left(\sum_{ij} a_{ij} U_i V_j\right).$$

This way we could turn the inner product $\langle u_i, v_j \rangle$ into the product $U_i V_j$ (at the cost of adding the expectation) to which we can apply the assumption of the thm: for a given realization of U_i, V_j we can use the assumption in the thm to write

$$\left| \sum_{i,j} a_{ij} U_i V_j \right| \leq \max_i |U_i| \max_j |V_j|.$$

The issue here is that U_i, V_j are normal, so they are not bounded, so can be arbitrarily large.

4) Truncate the RVs U_i, V_j - separate into two parts, the 1st is bdd, the 2nd is unlikely (has small probability).

Given R, let

$$U_i^- := U_i \mathbb{1}_{|U_i| \leq R\|u_i\|} \quad U_i^+ := U_i \mathbb{1}_{|u_i| > R\|u_i\|}$$

Similarly define

$$V_j^- := V_j \mathbb{1}_{|V_j| \leq R\|v_j\|} \quad V_j^+ := V_j \mathbb{1}_{|v_j| > R\|v_j\|}$$

We have
$$U_i = U_i^+ + U_i^-$$
$$V_j = V_j^+ + V_j^-.$$

We have

$$\sum a_{ij} U_i V_j = \underbrace{\sum a_{ij} U_i^- V_j^-}_{S_1} + \underbrace{\sum a_{ij} U_i^+ V_j^-}_{S_2} + \underbrace{\sum a_{ij} U_i^- V_j^+}_{S_3} + \underbrace{\sum_{ij} a_{ij} U_i^+ V_j^+}_{S_4}$$

For S_1 by the hypothesis on the thm

$$|S_1| \le \max_i \|U_i^-\| \max_j \|V_j^-\| \le R^2 \max_i |u_i| \max_j |v_j|$$

So $$E|S_1| \le R^2 \max_i |u_i| \max_j |v_j|$$

5) For S_2 we write

$$ES_2 = \sum_{ij} a_{ij} E(U_i^+ V_j^-).$$

Consider U_i^-, V_j^+ as elements of the Hilbert space L_2 with the inner product

$$\langle X, Y \rangle_{L_2} = EXY.$$

Our $K = K(A)$ works for any Hilbert Space so we have

$$|ES_2| \le K \max_i \|U_i^+\|_{L_2} \max_j \|V_j^-\|_{L_2}$$

Since $U_i = \langle g, u_i \rangle$, we have $U_i \sim N(0, \|u_i\|^2) \sim \|u_i\| N(0,1)$

Thus

$$\|U_i^+\|_{L_2}^2 = E\, U_i^2 1_{|U_i| > R\|u_i\|} = \|u_i\|^2 E(g^2 1_{|g| > R})$$

where $g \sim N(0,1)$.

A simple integration by parts gives

$$\tfrac{1}{2} E g^2 1_{|g|>R} = E g^2 1_{g>R} = R \cdot \frac{1}{\sqrt{2\pi}} e^{-R^2/2} + P(g > R) \text{ so}$$

$$E g^2 1_{|g|>R} \le 2\left(R + \frac{1}{R}\right) \frac{1}{\sqrt{2\pi}} e^{-R^2/2} =: C_R$$

We get $\|U_i^+\|_{L_2}^2 \le \|u_i\|^2 C_R$, $\|V_j^-\|_{L_2} \le \|V_j\|_{L_2} = \|v_j\|$

Thus $|ES_2| \le K \cdot C_R \max_i \|u_i\| \max_j \|v_j\|$

Similarly $|ES_3| \le K \cdot C_R$ —— " ——

& $|ES_4| \le K \, C_R^2$ —— " ——

So $\left|E \sum_{ij} a_{ij} u_i v_j\right| \le \left(R^2 + K(2C_R + C_R^2)\right)$ "

K was the smallest which made ∫ work for all H, so

$$K \le R^2 + K(2C_R + C_R^2) \text{ so } K \le \frac{R^2}{1-(2C_R+C_R^2)}.$$

Plug in $R = 2.3$, get $K \le 8$ $\triangle$

<u>Rmk:</u> The hypothesis in the thm is that $\forall x_i, y_j \in \mathbb{R}$

$$(*) \qquad \left|\sum_{ij} a_{ij} x_i y_j\right| \le \max_i |x_i| \max_j |y_j|$$

This is in fact equivalent to

$$(\#) \qquad \left|\sum_{ij} a_{ij} x_i y_j\right| \le 1 \quad \forall \; x_i, y_j \in \{\pm 1\}.$$

That $(*) \Rightarrow (\#)$ is trivial.

Suppose $(\#)$ holds. Let S be the subset of $\mathbb{R}^{n+m}$ consisting of all the vectors $s = (x_1, \ldots, x_m, y_1, \ldots, y_j)$ such that

$$-1 \le \sum_{ij} a_{ij} x_i y_j \le 1.$$

$(\#)$ implies all the vectors $(\pm 1, \pm 1, \ldots, \pm 1) \in S$.

Since S is defined by a collection of linear inequalities, it is convex, so S must contain the convex hull of $(\pm 1, \ldots, \pm 1)$, so $\forall \; x_1, \ldots, x_m, y_1, \ldots, y_n \in \mathbb{R}$,

$$\left(\frac{x_1}{\max_i |x_i|}, \ldots, \frac{x_m}{\max_i |x_i|}, \frac{y_1}{\max_j |y_j|}, \ldots, \frac{y_n}{\max_j |y_j|}\right) \in S$$

which means $(*)$ holds.

Applications of Grothendieck's inequality

In computationally difficult problems often approximate solutions are seeked. Grothendieck's inequality can be used to guarantee the approximation will be good.
We will look at examples of computationally difficult problems which can be approximated by semi-definite programming a generalization of linear programming.

Def: A semi-definite program is an optimization problem of the following type:
given $n\times n$ matrices $A, B_1, \dots, B_m$, and real numbers $b_1, \dots, b_m$ find an $n\times n$ positive semi-definite matrix X ($X \succeq 0$) which maximizes $\langle A, X\rangle$ under the constraints

$$X \succeq 0, \quad \langle B_i, X\rangle = b_i \quad \text{for } i=1,\dots,m.$$

Note that the inner product of $n\times n$ matrices A, B is

$$\langle A, B\rangle = \sum_{ij} a_{ij} b_{ij}$$

which can be written as

$$\langle A, B\rangle = tr(A^T B).$$

Rmk: The main difference from linear programming is in the constraints: non-negativity in linear programming is replaced by positive semi-definiteness here

Rmk: The set of semi-definite matrices forms a convex set in the space of $n\times n$ matrices (Check!) & the intersection of that set with the

constraint hyperplanes $\langle B_i, X\rangle = b_i$ is still convex,
so the semi-definite program is optimizing
$\langle A, X\rangle$ over a convex set, which makes it
computationally tractable

We'll now look at some examples of algorithms which we will
approximate by semi-definite programming.

<u>Problem:</u> maximize $\sum_{ij} A_{ij} x_i x_j$ $\quad x_i \in \{\pm 1\}, i = 1, \ldots, n$

where A is a fixed $n \times n$ symmetric matrix.

let $\bar{x} = (\bar{x}_1, \ldots, \bar{x}_n) \in \{\pm 1\}^n$ be the maximizer &
$\mathrm{Int}(A)$ be the maximum value achieved by $\sum A_{ij} x_i x_j$.

The problem is known to be NP-hard.

Instead of solving it, we can find the maximum approximately, up to a constant factor.

Replace the numbers $x_i = \pm 1$ by unit vectors x_i in $\mathbb{R}^n$ &
solve the optimization problem:

$(*)$ $\quad$ maximize $\sum_{i,j=1}^{n} A_{ij} \langle x_i, x_j\rangle$ $\quad$ s.t. $\|x_i\|_2 = 1 \ \forall i$.

Let X be the $n \times n$ matrix $X_{ij} = \langle x_i, x_j\rangle$.

<u>Check:</u> X is positive semi-definite, & every positive semi-definite
matrix can be realized in such a way.

We have

$$\sum_{i,j=1}^{n} A_{ij} \langle x_i, x_j\rangle = \sum_{i,j=1}^{n} A_{ij} X_{ij} = \langle A, X\rangle \quad , \text{so}$$

(*) becomes :

$$\text{maximize } \langle A, X \rangle \text{ under the constraints} \qquad (\#)$$
$$X \succcurlyeq 0, \quad X_{ii} = 1 \quad \forall i.$$

This is a semi-definite program. Let $\bar{X}$ be the maximizer & $Sdp(A)$ be the maximum achieved by $\bar{X}$.

By setting $X_i = (\bar{x}_i, 0, \ldots, 0)$ we see that $\bar{X}$ can not do worse than $\bar{x}$ so

$$Int(A) \le Spd(A).$$

On the other hand, if we replace the matrix A by

$$\hat{A} = \frac{A}{Int(A)}, \text{ then}$$

$$\forall x_i \in \{\pm 1\} \qquad \left| \sum \hat{A}_{ij} x_i x_j \right| \le 1$$

so by Grothendieck's inequality

$$\left| \sum \hat{A}_{ij} \langle X_i, X_j \rangle \right| \le 2K \quad \forall |X_i| = 1, i = 1, \ldots, n.$$

where K is Grothendieck's constant.

(the reason we need $2K$ instead of K is because we are dealing w/ the symmetric version $y_i = x_i \; \forall i$ of Grothendieck)

It follows that $Spd(A) \le 2K \, Int(A)$.

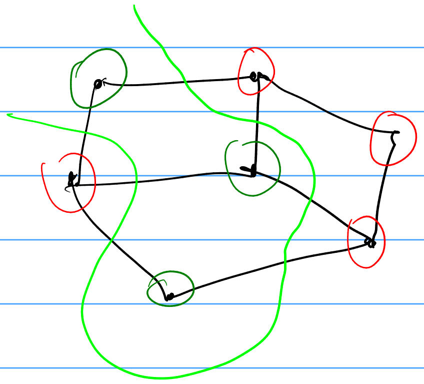
So

$$Int(A) \le Spd(A) \le 2k \, Int(A)$$

So by solving the semi-definite problem, instead of the original integer optimization problem, we over-estimate the maximum by a factor at most $2k \simeq 3.7$.

Maximum cuts on graphs

Let $G=(V,E)$ be a graph.
Partition the vertices into 2 disjoint sets & count the number of edges between them — this is called a cut.



Let max-cut(G) be the maximum possible cut.

Given G, computing max-cut(G) is NP-Hard.

Can use semi-definite programming to approximate it.

To do that, let's phrase the problem in terms of linear algebra.

Number the vertices $1, 2, \ldots, n$ & let A be the adjacency matrix $A_{ij} = \begin{cases} 1 & \text{if edge } ij \\ 0 & \text{o/w} \end{cases}$

A - symmetric

Let $x = (x_1, \ldots, x_n) \in \{\pm 1\}^n$ indicate the partition of a vertex.
The cut of this partition $\mathrm{Cut}(G,x)$ can be written as

$$\mathrm{Cut}(G,x) = \frac{1}{2}\sum_{ij} A_{ij}\, \mathbb{1}_{x_i \neq x_j} = \frac{1}{2}\sum_{ij} A_{ij}\, \mathbb{1}_{x_i x_j = -1} = \frac{1}{4}\sum_{ij} A_{ij}(1 - x_i x_j)$$

(w/o $\frac{1}{2}$ would double count).

So the integer optimization problem is

$$\text{max-cut}(G) = \frac{1}{4}\max\Big\{ \sum_{ij} A_{ij}(1 - x_i x_j) : x_i = \pm 1 \ \forall i \Big\}.$$

Instead, as before, consider the semi-definite program

$$Sdp(G) = \frac{1}{4} \max\left\{ \sum_{i,j=1}^{n} A_{ij}\left(1-\langle X_i, X_j\rangle\right) : X_i \in \mathbb{R}^n;\ \|X_i\|_2 = 1\ \forall i \right\}$$

Easy to see that $Sdp(G) \geq \text{Max-cut}(G)$.

Given the optimizer $X = (X_1, \ldots, X_n)$ for $Sdp(G)$

Can get a partition for G w/ a cut $\geq 0.878\, Sdp(G)$.

We have vectors $X_1, X_2, \ldots, X_n \in \mathbb{R}^n$.

Choose a random hyperplane through the origin & set $x_i = \pm 1$ depending on which side of the hyperplane it is.

We can choose the hyperplane by choosing its normal vector uniformly at random from the unit sphere, or choosing $g \sim N(0, I_n)$ & setting

$$x_i = \operatorname{sign}\langle g, X_i\rangle.$$

Let $x = (x_1, \ldots, x_n)$ be this random partition obtained from rounding $(X_1, \ldots, X_n)$. We have

Thus we have

$$\text{max-cut}(G) \geq \mathbb{E}\, \text{Cut}(G, x) \geq 0.878\, Sdp(G) \geq 0.878\, \text{Max-cut}(G).$$