Lecture 7: Semidefinite programming

Rajat Mittal

IIT Kanpur

Our next task will be to generalize ideas learnt in linear programming to a more general class of optimization problems, called *semidefinite programming*. This class, even though being more general, preserves the three important properties of linear programming: wide applicability, efficient algorithm and duality theory.

Semidefinite programming is a class of convex optimization where:

- Cost/optimization function is linear,
- Constraints are either linear equalities/inequalities, and/or membership constraints with respect to the semidefinite cone; these membership constraints are also called *generalized inequalities*.

Hence, it can be viewed as linear programming with the additional power of generalized inequalities for the positive semidefinite cone ($\mathcal{S}_n$). We will learn the meaning of generalized inequalities in this lecture note.

The main aim of the first part is to prepare your background for semidefinite programming. We have already seen some linear algebra. Now, we will see the concept of eigenvalues and eigenvectors, spectral decomposition and special classes of matrices. The class of positive semidefinite matrices will be of special interest to us. We will look at the properties of positive semidefinite matrices and the cone formed by them.

Later, we will look at the standard form of a semidefinite program. The lecture will finish with some situations which can be modelled as a semidefinite program. Specifically, we will cover Goemans-Williamson approximation algorithm for max cut using relaxation and rounding technique.

Remember, matrices are linear operators and every linear operator can be represented by a matrix (if we fix a basis). There were two important theorems covered earlier.

Theorem 1 (Rank-nullity theorem). *Let M be an $m \times n$ matrix. The dimension of the image of M is known as the* rank *of M ($rank(M)$) and the dimension of kernel of M is known as the nullity of M ($nullity(M)$). Then,*

$$rank(M) + nullity(M) = n.$$

Theorem 2 (Column rank and row rank). *The column rank of a matrix M is same as the row rank of M.*

1 Eigenvalues and eigenvectors

Consider two vector spaces V and W over real numbers. A matrix $M \in L(V, W)$ is square if $dim(V) = dim(W)$. In particular, a matrix $M \in L(V)$ is always square.

Consider a matrix $M \in L(V)$, any vector $v \in V$ satisfying,

$$Mv = \lambda v \text{ for some } \lambda \in \mathbb{R},$$

is called the *eigenvector* of matrix M with *eigenvalue* λ.

Exercise 1. Given two eigenvectors v, w, when is their linear combination an eigenvector itself?

The previous exercise can be used to show that all the eigenvectors corresponding to a particular eigenvalue form a subspace. This subspace is called the *eigenspace* of the corresponding eigenvalue.

An eigenvalue λ of an $n \times n$ matrix M satisfies the equation

$$det(\lambda I - M) = 0,$$

where $det(M)$ denotes the determinant of the matrix M. The polynomial $det(\lambda I - M) = 0$, in λ, is called the *characteristic polynomial* of M. The characteristic polynomial has degree n and will have n roots in the field of complex numbers. Though, these roots might not be real.

It can be shown that if λ is a root of characteristic polynomial then there exist at least one eigenvector corresponding to λ. We leave it as an exercise (Hint: look at the null space of $\lambda I - M$ for that λ).

Exercise 2. Give an example of a matrix with no real roots of the characteristic polynomial.

The next theorem says that eigenvalues are preserved under basis transformation.

Theorem 3. *Given a matrix P of full rank, matrix M and matrix $P^{-1}MP$ have the same set of eigenvalues.*

Proof. Suppose λ is an eigenvalue of $P^{-1}MP$, we need to show that it is an eigenvalue for M too. Say λ is an eigenvalue with eigenvector v. Then,

$$P^{-1}MPv = \lambda v \Rightarrow M(Pv) = \lambda Pv.$$

Hence Pv is an eigenvector with eigenvalue λ.

The opposite direction follows similarly. Given an eigenvector v of M, it can be shown that $P^{-1}v$ is an eigenvector of $P^{-1}MP$.

$$P^{-1}MP(P^{-1}v) = P^{-1}Mv = \lambda P^{-1}v$$

Hence proved. □

Exercise 3. Where did we use the fact that P is a full rank matrix?

The problem $Ax = b$ is easy to solve when the matrix is upper/lower triangular (we convert a general matrix into that form using Gaussian elimination). Similarly $Ax = \lambda x$ is easy to solve when the matrix is diagonal.

Exercise 4. What are the eigenvalues of a diagonal matrix?

Using Thm. 3, it is easy to find eigenvalues of a matrix, if it can be diagonalized (for M, find P such that $P^{-1}MP$ is diagonal). It is not always possible to diagonalize a matrix. There is a beautiful theory of diagonalizable matrices, we will not go into details here. Interested readers can refer Gilbert Strang's book on linear algebra [2].

One of the cases, when it can be diagonalized, is symmetric matrices. In this case, the eigenvectors are orthogonal to each other. This fact about symmetric matrices is also known as *spectral decomposition.*

Theorem 4. *Spectral decomposition: For a symmetric matrix $M \in \mathbb{R}^{n\times n}$, there exists an orthonormal basis $x_1, \cdots, x_n$ of $\mathbb{R}^n$, s.t.,*

$$M = \sum_{i=1}^{n} \lambda_i x_i x_i^T.$$

Here, $\lambda_i \in \mathbb{R}$ for all i.

The proof of this theorem is given in the next section and is optional.

Exercise 5. Show that x_i is an eigenvector of M with eigenvalue λ_i.

Note 1. $u^T w$ is a scalar, but uw^T is a matrix.

Note 2. The λ_i's need not be different. If we collect all the x_i's corresponding to a particular eigenvalue λ, the space spanned by those x_i's is the eigenspace of λ.

A matrix U is orthogonal if $UU^T = U^TU = I$. In other words, the columns of U form an orthonormal basis of the whole space.

Orthogonal matrices can be viewed as matrices which implement a change of basis. Hence, they preserve the angle (inner product) between the vectors (prove it). So for an orthogonal U,

$$u^Tv = (Uu)^T(Uv).$$

If two matrices A, B are related by $A = U^{-1}BU$, where U is orthogonal, then they are called *orthogonally similar*. If two matrices are orthogonally similar then they are similar.

Spectral theorem can be stated as the fact that symmetric matrices are orthogonally similar to a diagonal matrix. It means that any symmetric matrix M can be expressed as U^TDU, where D is the diagonal matrix with eigenvalues and U is the orthogonal matrix with columns as eigenvectors.

1.1 Extra reading: proof of spectral decomposition

Exercise 6. Let v_1, v_2 be two eigenvectors of a matrix M with distinct eigenvalues. Show that these two eigenvectors are linearly independent.

This exercise also shows: sum of the dimensions of eigenspaces of an $n \times n$ matrix M can't exceed n.

Given an $n \times n$ matrix M, it need not have n linearly independent eigenvectors. The matrix M is called *diagonalizable* iff the set of eigenvectors of M span the complete space $\mathbb{R}^n$.

For a diagonalizable matrix, the basis of eigenvectors need not be an orthogonal basis. We will be interested in matrices which have an orthonormal basis of eigenvectors.

Suppose a matrix M has an orthonormal basis of eigenvectors. Let $\lambda_1, \lambda_2, \cdots, \lambda_n \in \mathbb{R}$ be the n eigenvalues with the corresponding eigenvectors $u_1, u_2, \cdots, u_n$. Define D to be the diagonal matrix with $D_{ii} = \lambda_i$ for every i. Let U be the matrix with i-th column being u_i. Since u_i's are orthonormal, $U^T = U^{-1}$.

Exercise 7. Show that $M = UDU^T$. Remember, to show that two matrices are same, we only need to show that their action on a basis is same.

So, if a matrix M has an orthonormal set of eigenvectors, then it can be written as UDU^T. This implies that $M = M^T$. We call such matrices *symmetric*.

What about the reverse direction? *Spectral decomposition* shows that every symmetric matrix has an orthonormal set of eigenvectors. Before proving spectral decomposition, let us look at the eigenvalues and eigenvectors of a symmetric matrix.

Lemma 1. *Let $Mu = \lambda_1 u$ and $Mw = \lambda_2 w$, where λ_1 and λ_2 are not equal. Then*

$$u^Tw = 0.$$

Proof. Notice that $\lambda_1, \lambda_2, u, w$ need not be real. From the conditions given in the lemma,

$$\lambda_1 u^Tw = (Mu)^Tw = u^T(Mw) = \lambda_2 u^Tw.$$

Where did we use the fact that M is symmetric?

Since $\lambda_1 \neq \lambda_2$, we get that $u^Tw = 0$. □

Let M be a symmetric matrix. A priori, it is not even clear that all the roots of $det(M - \lambda I)$ are real. Let us first prove that all roots of the characteristic polynomial are real.

Lemma 2. *Given an $n \times n$ symmetric matrix M, all roots of $det(\lambda I - M)$ are real.*

Proof. Let λ be a root of $det(\lambda I - M)$. Suppose it is not real,

$$\lambda = a + ib, \text{ where } b \neq 0.$$

Since $\lambda I - M$ is zero, the kernel of $\lambda I - M$ is not empty. Hence, there exists a vector $v = x + iy$, such that

$$M(x + iy) = (a + ib)(x + iy).$$

Taking the adjoint of this equation and noting $M^* = M$ (M is real and symmetric),

$$M(x - iy) = (a - ib)(x - iy).$$

Using Lem. 1, we know that $x + iy$ and $x - iy$ should be orthogonal to each other.

Exercise 8. Prove that $x + iy$ and $x - iy$ can not be orthogonal to each other by taking their inner product.

□

Now we are ready to prove spectral decomposition.

Proof of Thm. 4. Proof of spectral theorem essentially hinges on the following lemma.

Lemma 3. *Given an eigenspace S (of eigenvalue λ) of a symmetric matrix M, the matrix M acts on the space S and $S^{\perp}$ separately. In other words, $Mv \in S$ if $v \in S$ and $Mv \in S^{\perp}$ if $v \in S^{\perp}$.*

Proof of lemma. Since S is an eigenspace, $Mv \in S$ if $v \in S$. This shows that M preserves the subspace S.

Suppose $v_1 \in S^{\perp}, \; v_2 \in S$, then $Mv_2 = M^T v_2 \in S$. So,

$$0 = v_1^T M^T v_2 = (Mv_1)^T (v_2).$$

This shows that $Mv_1 \in S^{\perp}$. Hence, matrix M acts separately on S and $S^{\perp}$. □

We have already shown that the eigenvectors of a symmetric matrix corresponding to different eigenvalues are orthogonal (Lem. 1). Also, it was shown that every root of the characteristic polynomial is real, so there are n real roots (Lem. 2). Though some roots might be present with multiplicities more than 1.

Assume that we list out all possible eigenvalues $\lambda_1, \lambda_2, \cdots, \lambda_k$ with their eigenspaces $P_1, P_2, \cdots, P_k$. If $\sum_{i=1}^{k} dim(P_i) = n$, then we are done. If not say the remaining space is P_{k+1}.

Since eigenvalues do not change under a basis transformation (Thm. 3), we can look at M in the bases of $P_1, P_2, \cdots, P_{k+1}$. Lem. 3 implies that matrix M looks like,

$$\begin{pmatrix} \lambda_1 & & & & & & \\ & \lambda_1 & & & & & \\ & & \ddots & & & & \\ & & & \lambda_k & & & \\ & & & & \ddots & & \\ & & & & & \lambda_k & \\ & & & & & & C \end{pmatrix}.$$

We can assume that C is non-zero (why?). Then fundamental theorem of algebra says that $det(\lambda I - C)$ has a root. Since that will also be a root of $det(\lambda I - M)$, it has to be real. But then this real root will have at least one eigenvector. This is a contradiction, since we had listed all possible eigenvalues and their eigenspaces.

Exercise 9. Given the spectral decomposition of M, what is the spectral decomposition of M^T?

□

It was shown before that any matrix with orthonormal set of eigenvectors is a symmetric matrix. Hence, spectral decomposition provides another characterization of symmetric matrices.

Clearly the spectral decomposition is not unique (essentially because of the multiplicity of eigenvalues). But the eigenspaces corresponding to each eigenvalue are fixed. So there is a unique decomposition in terms of eigenspaces and then any orthonormal basis of these eigenspaces can be chosen.

2 Positive semidefinite matrices

A matrix M is called *positive semidefinite (psd)* if it is symmetric and all its eigenvalues are non-negative. If all eigenvalues are strictly positive then it is called a *positive definite* matrix.

In many references, you might find another definition of positive semidefiniteness. A matrix $M \in L(V)$ will be called positive semidefinite iff,

1. M is symmetric,
2. $v^T M v \geq 0$ for all $v \in V$.

The two definitions for positive semidefinite matrix turn out be equivalent. In the next theorem, we identify many different definitions of positive semidefinite (psd) matrices to be equivalent.

Theorem 5. *For a symmetric $n \times n$ matrix $M \in L(V)$, following are equivalent.*

1. *$v^T M v \geq 0$ for all $v \in V$.*
2. *All the eigenvalues are non-negative.*
3. *There exist a matrix B, s.t., $B^T B = M$.*
4. *M is the Gram matrix of vectors $u_1, \cdots, u_n \in U$, where U is some vector space.*

$$\forall i, j : \; M_{i,j} = u_i^T u_j$$

Note 3. The vector space U (space of vectors for Gram matrix) is not the same as V, space over which M acts. In general, they might have different dimensions.

Proof. $1 \Rightarrow 2$: Say λ is an eigenvalue of M. Then there exist eigenvector $v \in V$, s.t., $Mv = \lambda v$. So $v^T M v = \lambda v^T v \geq 0$. Since $v^T v$ is positive for all v, implies λ is non-negative.

$2 \Rightarrow 3$: Since the matrix M is symmetric, it has a spectral decomposition.

$$M = \sum_i \lambda_i x_i x_i^T$$

Define $y_i = \sqrt{\lambda_i} x_i$ (this definition is possible because λ_i's are non-negative). Then,

$$M = \sum_i y_i y_i^T.$$

Define B to be the matrix whose rows are y_i. Then it is clear that $B^T B = M$.

$3 \Rightarrow 4$: We are given a matrix B, s.t., $B^T B = M$. Say the columns of B are $v_1, \cdots, v_n$. Then, from the definition of matrix multiplication,

$$\forall i, j : \; M_{i,j} = v_i^T v_j$$

Exercise 10. Convince yourself that M is a gram matrix of $v_1 \cdots, v_n$.

$4 \Rightarrow 1$: Suppose M is the gram matrix of vectors $u_1, \cdots, u_n$. Then,

$$x^T M x = \sum_{i,j} M_{i,j} x_i x_j = \sum_{i,j} x_i x_j (v_i^T v_j),$$

where x_i is the i^{th} element of vector x. Define $y = \sum_i x_i v_i$, then,

$$0 \geq y^T y = \sum_{i,j} x_i x_j (v_i^T v_j) = x^T M x.$$

Hence $x^T M x \geq 0$ for all x.

Exercise 11. Prove that $2 \Rightarrow 1$ and $3 \Rightarrow 1$ directly.

□

Remark: A matrix M of the form $M = \sum_i x_i x_i^T$ is positive semidefinite (Exercise: Prove it), even if x_i's are not orthogonal to each other.

Remark: A matrix of the form yx^T is a rank one matrix. It is rank one because all columns are scalar multiples of y. Similarly, all rank one matrices can be expressed in this form.

Exercise 12. A rank one matrix yx^T is positive semi-definite iff y is a positive scalar multiple of x.

From the construction in the proof for matrix B such that $B^T B = M$, B's columns are orthogonal. In general, any matrix of the form $B^T B$ is positive semi-definite. The matrix B need not have orthogonal columns (it can even be rectangular).

This representation, $B^T B = M$ is not unique, but there always exists a matrix B with orthogonal columns for M, s.t., $B^T B = M$. This decomposition (with B having orthogonal columns) is unique (gives spectral decomposition). The positive semidefinite B, s.t., $B^T B = M$, is called the square root of M.

Exercise 13. Prove that the square root of a matrix is unique.

Hint: Use the spectral decomposition to find one of the square root. Suppose A is any square root of M. Then use the spectral decomposition of A and show the square root is unique (remember the decomposition to eigenspaces is unique) .

2.1 Some examples

- An $n \times n$ identity matrix is positive semidefinite. It has rank n. All the eigenvalues are 1 and every vector is an eigenvector. It is the only symmetric matrix with all eigenvalues 1 (prove it).
- The all 1's matrix J ($n \times n$) is a rank one psd matrix. It has one eigenvalue n and rest are zero.
- The matrix

$$M = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix},$$

  is positive semidefinite. Because, the quadratic form $x^T M x = (x_1 - x_2)^2$, where x_1, x_2 are two components of x.
- Let λ be the maximum eigenvalue for a matrix M. The matrix $\lambda' I - M$, where $\lambda' \geq \lambda$ is positive semidefinite.

Exercise 14. What is the maximum value of $x^T M x$ for a unit vector x?

2.2 Extra reading: properties of semidefinite matrices

Principal submatrix A principal submatrix P of a matrix M is obtained by selecting a subset of rows and the same subset of columns. If M is positive semidefinite then all its principal submatrices are also positive semidefinite.

This follows by considering the quadratic form $x^T M x$ and looking at the components of x corresponding to the defining subset of principal submatrix. The converse is trivially true.

Exercise 15. Show that the determinant of a psd matrix is non-negative. Hence, show that all the principal minors are non-negative. Actually the converse also holds true, i.e., if all the principal minors are non-negative then the matrix is positive semidefinite.

Diagonal elements If the matrix is positive semidefinite then its diagonal elements should *dominate* the non-diagonal elements. The quadratic form for M is,

$$x^T M x = \sum_{i,j} M_{i,j} x_i x_j. \tag{1}$$

Here x_i's are the respective components of x. If M is positive semidefinite then Eqn. 1 should be non-negative for every choice of x.

By choosing x to be a standard basis vector e_i, we get $M_{ii} \geq 0, \ \forall i$. Hence, all diagonal elements are non-negative and $tr(M) \geq 0$.

If x is chosen to have only two nonzero entries, let's say 1 at i and -1 at j position, then Eqn. 1 implies,

$$M_{i,j} \leq \frac{M_{ii} + M_{jj}}{2}.$$

A stronger version can be obtained (remember AM-GM inequality) by observing that the determinant of 2×2 submatrix (i, j rows and columns) is non-negative. Notice that the determinant of any positive semidefinite matrix is non-negative (all eigenvalues are non-negative).

$$M_{i,j} \leq \sqrt{M_{ii} M_{jj}}.$$

This shows that any off diagonal element is less than or equal to either the diagonal element in its row or in its column.

Composition of semidefinite matrices

– The direct sum matrix $A \oplus B$,

$$\begin{pmatrix} A & 0 \\ 0 & B \end{pmatrix},$$

is positive semidefinite iff A and B both are positive semidefinite. This can most easily be seen by looking at the quadratic form $x^T(A \oplus B)x$. Divide x into x_1 and x_2 of the required dimensions, then

$$x^T(A \oplus B)x = x_1^T A x_1 + x_2^T B x_2.$$

– The tensor product $A \otimes B$ is positive semidefinite iff A and B are both positive semidefinite or both are negative semidefinite. This follows from the fact that given the eigenvalues $\lambda_1, \cdots, \lambda_n$ for A and $\mu_1, \cdots, \mu_m$ for B; the eigenvalues of $A \otimes B$ are

$$\forall i, j : \ \lambda_i \mu_j.$$

– The sum of two psd matrices is psd.
– The product of two positive semidefinite matrices need not be positive semidefinite.

Exercise 16. Give an example of two positive semidefinite matrices whose product is not positive semidefinite.

– The Hadamard product of two positive semidefinite matrices A and B, $A \circ B$, is also positive semidefinite. Since A and B are positive semidefinite, they are Gram matrix of some vectors $u_1, \cdots, u_n$ and $v_1, \cdots v_n$. The Hadamard product will be the Gram matrix of $u_i \otimes v_i$'s.
– The inverse of a *positive definite* matrix is positive definite. The eigenvalues of the inverse are inverses of the eigenvalues.
– The matrix $P^T M P$ is positive semidefinite if M is positive semidefinite.

2.3 Positive semidefinite cone

Consider the vector space of real symmetric $n \times n$ matrices, $\mathbb{R}^{\frac{n(n+1)}{2}}$. What kind of structure does the set of psd matrices form in this space?

If M, N are positive semidefinite, then $\alpha M + \beta N$ is also psd for $\alpha, \beta \geq 0$. Hence, the set of positive semidefinite matrices is a *convex cone* in $\mathbb{R}^{\frac{n(n+1)}{2}}$. The cone is denoted as $\mathcal{S}_n$.

This cone has more nice properties. If $M \succeq 0$ then $-M$ is not positive semi-definite. So, the cone $\mathcal{S}_n$ does not contain a line. Also, if we look at the positive definite matrices. They form the interior of the cone. Identity is positive definite, so interior is not empty.

Hence, $\mathcal{S}_n$ is a convex cone that does not contain a line and has non-empty interior. Such cones are called *proper* cones; in other words, $\mathcal{S}_n$ is proper. We can define a generalized inequality with respect to such a proper cone.

$$M \succeq N \Leftrightarrow M - N \succeq 0 \Leftrightarrow M - N \in \mathcal{S}_n$$

Note 4. In general, any proper cone C gives rise to a generalized inequality, $M \succeq N \Leftrightarrow M - N \in C$. The simplest case is when C is positive orthant ($x \geq 0$). In this case, generalized inequality corresponds to our usual inequality.

The positive semidefinite cone is generated by all rank one matrices xx^T. They form the extreme rays of the cone. As mentioned before, the positive definite matrices lie in the interior of the cone. The positive semidefinite matrices with at least one zero eigenvalue are on the boundary.

3 Semidefinite programs (SDP)

Let $A \bullet B = tr(A^T B)$ be the *Hadamard product* between two matrices. It can be viewed as the standard inner product of A, B in n^2 dimensional vector space.

A semidefinite program (SDP) in a standard form looks like,

$$\begin{aligned} \max \quad & C \bullet X \\ \text{s.t.} \quad & A_i \bullet X = b_i, \ i = 1, \cdots, m \\ & X \succeq 0. \end{aligned}$$

In the semidefinite program above, X is the *variable matrix* of dimension $n \times n$. The matrix C is called the cost or *objective matrix.* A_i's are the *constraint matrices.* C and A_i's have the same dimension as X ($n \times n$). b_i's are scalars and the vector b (with b_i) as components is known as *constraint vector.*

Note 5. Only change from a linear program: instead of variables being in a positive orthant, they should be in positive semidefinite cone.

Many of the standard tricks used in linear programming to convert non-standard form into standard form can also be used here. For example, converting inequalities into equalities, changing minimum to maximum and change of variables.

Let us take a look at the following program,

$$\begin{aligned} \max \quad & Tr(X) \\ \text{s.t.} \quad & X = \begin{pmatrix} 1 & x \\ 1 & x \end{pmatrix} \succeq 0. \end{aligned}$$

Show that it is a semidefinite program. In other words, what are the constraint matrices and constraint vector?

Exercise 17. Find the value of this semidefinite program.

Generalizing the above example, a semidefinite program can be viewed as,

- variables $x_{i,j}$ arranged in a matrix X,
- linear constraints and objective over these variable $x_{i,j}$,
- and the positive semidefinite constraint $X \succeq 0$.

You will prove in the assignment that the positive semidefinite constraint can be viewed as infinite number of linear constraints in variables $x_{i,j}$.

Exercise 18. Find the value of this semidefinite program.

$$\begin{array}{rl} & \inf / \min \quad x_1 \\ \text{s.t.} & \begin{pmatrix} x_1 & 1 \\ 1 & x_2 \end{pmatrix} \succeq 0. \end{array}$$

3.1 Equivalent definitions

In general, many totally different looking programs can be transformed into an sdp.

Form with positive semidefinite constraints Another standard form for sdp's is,

$$\begin{array}{rl} & \min b^T y \\ \text{s.t.} & \sum_{i=1}^m y_i A_i - C \succeq 0. \end{array}$$

Let us prove that this form can be converted to the standard form.

Denote the matrix $\sum_i y_i A_i - C$ by a new variable matrix Z. Now the matrix variable in the program is Z and we also have scalar variables y_i's. The linear constraints can be changed into

$$\forall\ i, j;\ z_{ij} = (\sum_k y_k (A_k)_{ij}) - C_{i,j}.$$

Here z_{ij} are the entries of matrix Z. So, the program changes to,

$$\begin{array}{rl} & \min b^T y \\ \text{s.t.} & \forall\ i, j;\ z_{ij} = (\sum_k y_k (A_k)_{ij}) - C_{i,j} \\ & Z \succeq 0 \end{array}$$

It almost looks like the standard form but variables y_i's do not occur in the semidefinite constraint.

Use the old trick, convert unrestricted variables to positive variables. Replace y_i by two positive variables, i.e., $y_i = y_i' - y_i''$ and $y_i', y_i'' \geq 0$. Now, these positive variables y_i', y_i'' can be put in a separate block and included in the semidefinite constraint.

$$\begin{array}{rl} & \min \sum_k b_k (y_k' - y_k'') \\ \text{s.t.} & \forall\ i, j;\ z_{ij} = (\sum_k (y_k' - y_k'')(A_k)_{ij}) - C_{i,j} \\ & Z = \begin{pmatrix} Z & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & y_1' & \cdots & 0 & 0 & 0 & 0 \\ 0 & \vdots & \ddots & \vdots & 0 & 0 & 0 \\ 0 & 0 & 0 & y_m' & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & y_1'' & \cdots & 0 \\ 0 & 0 & 0 & 0 & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & y_m'' \end{pmatrix} \succeq 0 \end{array}$$

You will show in the assignment that we don't need explicit constraints to force the off-diagonal entries to be zero.

Gram matrix formulation An $n \times n$ positive semidefinite matrix X can be expressed as the Gram matrix of vectors $u_1, u_2, \cdots, u_n$. Write the variable matrix X as the gram matrix of $u_1, \cdots, u_n \in \mathbb{R}^k$. Notice, $u_1, u_2, \cdots, u_n \in \mathbb{R}^k$ are vector variables without any constraint on k.

Then, the semidefinite program looks like,

$$\begin{aligned} \max \quad & \textstyle\sum_{ij} C_{ij} u_i^T u_j \\ \text{s.t.} \quad & \textstyle\sum_{ij} A_{ij}^{(k)} u_i^T u_j = b_k, \ \forall k = 1, \cdots, m \end{aligned}$$

We have removed the $X \succeq 0$ constraint from the new form.

The new form can be understood as,

- $u_1, u_2, \cdots, u_n$ as vector variables,
- and linear constraints over the *inner product* of these vector variables.

This form is specially useful in combinatorial optimization. The reason is, many problems can be expressed as integer programs in combinatorial optimization. When you relax these variables to be vectors instead of integers, this form arises naturally. We will see such an example later in the course.

Note 6. This is not a linear program, since constraints are on the inner-products.

3.2 Examples

Minimizing the maximum eigenvalue Suppose, we are given a matrix $M(x)$, which depends affinely on the variables in x. That means every entry in $M(x)$ can be written as an affine function of variables in x ($M(x)_{ij} = a_1 x_1 + \cdots + a_n x_n + b$). The problem is to minimize the maximum eigenvalue of $M(x)$ over all x, i.e.,

$$\min_x \max_i \lambda_i(M(x))$$

Clearly, this is not in the standard form of SDP. The trick here is to introduce another variable η to change $\min\max$ to only min. Suppose $\lambda_{\max}(M)$ represents the maximum eigenvalue of M, then

$$\begin{aligned} & \min \eta \\ \text{s.t.} \quad & \eta \geq \lambda_{\max}(M(x)) \end{aligned}$$

Now, we use the fact that $\lambda_{\max} I - M(x) \succeq 0$. Hence,

$$\begin{aligned} & \min \eta \\ \text{s.t.} \quad & \eta I - M(x) \succeq 0 \end{aligned}$$

This is one of the alternative form discussed in the last section (why?). Here the variables are (η, x).

Exercise 19. Write an SDP to find the maximum eigenvalue of a matrix M.

Linear programs as a special case Linear programming is a special case of sdp's. It is obtained by restricting matrices to be diagonal in the standard form of semidefinite programming. Suppose the linear program is,

$$\begin{aligned} \max \quad & c^T x \\ \text{s.t.} \quad & a_i^T x = b_i, \ \forall i = 1, \cdots, m \\ & x \geq 0. \end{aligned}$$

Look at the semidefinite program,

$$\begin{aligned} \max \quad & C \bullet X \\ \text{s.t.} \quad & A_i \bullet X = b_i, \ \forall i = 1, \cdots, m \\ & X \succeq 0; \end{aligned}$$

Here, C is the diagonal matrix with entries from c and A_i's are the diagonal matrices with diagonals a_i. Then the above mentioned two programs are actually equal.

Given a solution of the linear program, it can be converted into a solution for semidefinite program by taking X to be the diagonal matrix with diagonal x. Similarly, if X is any solution for the semidefinite program, then $x = diag(X)$ will be a solution for the linear program with the same objective value.

Hence, any linear program can be converted into a semidefinite program by taking corresponding diagonal matrices for the constraints as well as the objective matrix.

Sum of squares If a polynomial $p(x_1, x_2, \cdots, x_n)$ can be written as a sum of squares then it is clearly positive for all values of variable x. In general, this gives a sufficient condition for positivity of the polynomial.

To check whether a polynomial can be written as a sum of squares turns out to be a semidefinite programming feasibility problem. Notice that a polynomial (of degree d) can be written as the dot product between two vectors $p(x) = p^T x_d$. Here, x_d is the list of all $\leq d$ degree monomials. In other words, p is the vector of coefficients corresponding to those monomials.

Suppose $p(x) = \sum_i q_i(x)^2$, i.e., it can be written as the sum of squares. Say the degree of p is $2d$. Hence,

$$p(x) = \sum_i q_i(x)^2 = \sum_i x_d^T q_i^T q_i x_d = x_d^T (\sum_i q_i^T q_i) x_d$$

This shows that a polynomial is a sum of squares, iff, it can be written as $x_d^T Q x_d$ for some positive semidefinite Q. So, a polynomial is a sum of squares iff

$$\begin{aligned} p(x) &= x_d^T Q x_d \\ Q &\succeq 0. \end{aligned}$$

It might be confusing that why this is a semidefinite program. First, the constraint $p(x) = x_d^T Q x_d$ is a linear constraint on the elements of Q (variables for us). These constraints can be obtained by comparing coefficient of each monomial of p.

Also the absence of max/min might be confusing. This kind of problem without objective function and only constraint is called a semidefinite programming feasibility problem. It is a special case of semidefinite programming (why ?).

This semidefinite program is important in giving bounds on the minimum value of a polynomial. Consider,

$$\begin{aligned} & \max \lambda \\ \text{s.t.} \quad & p(x) - \lambda = x_d^T Q x_d \\ & Q \succeq 0. \end{aligned}$$

The value of this program (say s^*) satisfies $p(x) \geq s^*$ for all x. So, it gives a lower bound on the minimum value of the polynomial.

This might not be the minimum value of p, i.e., biggest s such that $p(x) \geq s$ for all x; since the representation as sum of squares is only a sufficient condition for positivity. Though if $n = 1$, it is known to give the tight bound.

4 Duality theory for semidefinite programming

We have already seen the strong connections of dual and primal program in case of linear programming. Similar connections exist in the world of semidefinite programming. To extend the theory of duality for semidefinite program, we need the concept of *dual cone*.

In this section, we will introduce dual cones, learn how to take dual of a general cone program and apply this knowledge to semidefinite programming. Please note that *strong duality* does not need to always hold in the case of a semidefinite program, as was the case in linear programs.

4.1 Dual cone

Exercise 20. What is the definition of a cone, convex cone and a finitely generated cone?

Given a cone C in $\mathbb{R}^n$, we can define the dual of cone C by

$$C^* = \{y \in \mathbb{R}^n : y^T x \geq 0,\ \forall x \in C\}$$

Here $y^T x$ is the inner product on $\mathbb{R}^n$.

Suppose $y \in C^*$, then the entire cone C lies on one side of the hyperplane normal to y (passing through origin). The cone C contains origin (0). In other words, the hyperplane with normal y (passing through origin) is a supporting hyperplane at 0.

Exercise 21. Read about supporting hyperplane.

Given two hyperplanes which are supporting at 0, their convex combination (convex combination of normals) will also give a supporting hyperplane. Geometrically, the convex combination will lie *in between* the original two hyperplanes. Hence,

$$y_1, y_2 \in C^* \rightarrow \theta y_1 + (1-\theta) y_2 \in C^*,\ \forall \theta \in \{0, 1\}$$

It is again easy to see (by this geometric picture),

$$y \in C^*,\ \theta \geq 0 \Rightarrow \theta y \in C^*.$$

So the dual cone is a convex cone irrespective of whether the original cone was convex or not. Notice from the definition, the dual cone is always closed (all limits exist in the dual cone). Actually, the starting set C need not be a cone. For any set C, we can define the dual and it will be a convex cone.

Exercise 22. Can you describe, what will be the dual cone of a set C?

Exercise 23. Show that if $C \subseteq D$, then $D^* \subseteq C^*$.

Examples Let us find the dual of some cones.

- Subspace: First, we can show that a linear subspace L (all linear combinations are present in the set) is a cone. We leave it as an easy exercise.
  A vector y is in L^* iff $y^T x \geq 0,\ \forall x \in L$. But in a subspace, if x is present then so is $-x$.
  So,
  $$y^T x = 0,\ \forall x \in L.$$
  Hence, the dual cone to the subspace contains all vectors normal to it and nothing else.
- Positive orthant ($\mathbb{R}^n_+$): We discussed that this set is a cone before and the usual $\geq$ inequality is the generalized inequality with respect to this cone.

  Exercise 24. Show that the positive orthant is the dual cone of itself

Such cones are called *self dual cones*.

– Set of all positive semidefinite matrices are called the *positive semidefinite cone*. We will show later that it is also a self dual cone.
– Another interesting case of a dual cone is the case of a finitely generated cone. Suppose we are given a cone $C = cone(x_1, \cdots, x_k)$. How will its dual cone look?

$$C^* = \{y : y^T x \geq 0, \ \forall x \in C\}$$

Clearly, for all the generators of C $(x_1, \cdots, x_k)$, $y^T x_i \geq 0$. Since any element of the cone is a conic combination of these vectors, this condition is actually sufficient too. So,

$$C^* = \{y : y^T x_i \geq 0, \ i = 1, \cdots, k\}.$$

Notice that this gives the definition of cone in terms of inequalities. Affine Weyl theorem tells us that these two are equivalent formulations. So, C^* is also a finitely generated cone.
It is important to note that the generators of the original cone are the inequalities for the dual cone and vice versa. So, it is easy to specify the dual cone in the case of finitely generated cone.

Exercise 25. What will be the dual cone in $\mathbb{R}^2$ of a finitely generated cone which is generated by two vectors.

Dual of the dual cone The natural question is, what is the dual cone of C^* for a closed convex cone C?

Suppose $x \in C$ and $y \in C^*$, then we know $y^T x \geq 0$. Since this condition is symmetric, we get that every element of C will be in C^{**}, i.e., $C \subseteq C^{**}$. We will show that these two sets are indeed equal.

For the other direction, we need to show that any element x not in C has negative inner product with at least one $y \in C^*$.

Suppose $x \notin C$, then by Farkas lemma, there exists a hyperplane which separates the point x and the cone C. So there exists y, s.t.,

$$y^T x < 0 \leq y^T z, \ \forall z \in C.$$

Then, $y \in C^*$ by definition of dual cone. We also know, $y^T x < 0$, so $x \notin C^{**}$. Hence, for a closed convex cone C,

$$C = C^{**}.$$

Exercise 26. Dual cone can be defined for arbitrary sets also. What will be the dual of the dual in this case?

As discussed last time, we will mostly be interested in proper cones. Remember that they are convex, closed cones which do not contain any line and have nonempty interior. It is known that the dual of a proper cone is a proper cone. Using a proper cone S, we can define corresponding *generalized inequality* as a partial order.

$$x \geq y \Leftrightarrow x - y \in S$$

Exercise 27. Show that the positive orthant is a proper cone. What is the generalized inequality with respect to that cone?

4.2 $\mathcal{S}_n$ is a self dual cone

The inner product in the space of matrices is the $\bullet$ operation between matrices.

$$A \bullet B = \sum_{i,j} A_{ij} B_{ij} = tr(A^T B)$$

The dual cone of $\mathcal{S}_n$ is the cone $\mathcal{S}_n'$, s.t.,

$$\mathcal{S}_n' = \{M : M \bullet N \geq 0, \ \forall N \in \mathcal{S}_n\}$$

It was proved before that the Hadamard product of two positive semidefinite matrices is also positive semidefinite.

Exercise 28. If you don't remember the proof, try it now.

The easy exercise below shows that if M, N are positive semidefinite, then $M \bullet N \geq 0$.

Exercise 29. If $M \circ N \succeq 0$, prove that $M \bullet N \geq 0$.

Hence, every positive semidefinite matrix is part of the dual cone $\mathcal{S}'_n$. It implies

$$\mathcal{S}_n \subseteq \mathcal{S}'_n. \tag{2}$$

Now consider a symmetric matrix $M \notin \mathcal{S}_n$. There exist at least one negative eigenvalue λ and an eigenvector v corresponding to it. So,

$$0 > \lambda = v^T M v = M \bullet (vv^T).$$

Since vv^T is a psd matrix (why?), this implies that $M \notin \mathcal{S}'_n$. Hence,

$$\mathcal{S}'_n \subseteq \mathcal{S}_n. \tag{3}$$

From Eqn. 2 and Eqn. 3, we get $\mathcal{S}'_n = \mathcal{S}_n$. The cone of positive semidefinite cone is a self dual cone.

4.3 Dual for a cone program

Let us see how the dual of a general cone program can be obtained. We are given an optimization program,

$$\begin{aligned} &\max c^T x \\ \text{s.t.} \quad &Ax \leq b \\ x \in S \quad &(S \text{ is some cone.}) \end{aligned}$$

For this case, say we take the linear combination y of the rows of matrix A. Then, we want $y^T b$ to be an upper bound on the value of $c^T x$ for any feasible x. One possible way is to satisfy,

$$c^T x \leq y^T A x \leq y^T b.$$

For the second inequality, y should be positive and for the first one $y^T A - c \in S^*$. Here, S^* is the dual cone of S. So, the dual program looks like,

$$\begin{aligned} &\min y^T b \\ \text{s.t.} \quad &y^T A - c \in S^* \\ &y \geq 0, \end{aligned}$$

Let's look at the two programs slightly differently.

$$\begin{aligned} &\max c^T x \\ \text{s.t.} \quad &Ax - b \leq 0 \Leftrightarrow Ax - b \in C \\ &x \in S. \quad (S, C \text{ are cones}) \end{aligned} \qquad\qquad \begin{aligned} &\min y^T b \\ \text{s.t.} \quad &y^T A - c \in S^* \\ &-y \in C^*, \end{aligned}$$

Cone C is the cone of all negative vectors, $\mathbb{R}^n_-$.

If the given program is a maximization problem, then if the primal variable is in cone S then the dual constraint is the membership in S^*. If the primal constraint is membership in cone C then the negative of dual variable is in cone C^*. For the minimization problem the relationship is opposite.

	Primal variable in S	Primal constraint in C
Max	Dual constraint in S^*	Negative of dual variable in C^*
Min	Negative of Dual constraint in S^*	Dual variable in C^*

Exercise 30. Check and verify the table.

4.4 Dual of a semidefinite program

We can take the dual of the standard form of SDP using the technique discussed in the previous section.

$$\begin{aligned} &\text{Primal} \\ &\max C \bullet X \\ \text{s.t. } &A_i \bullet X = b_i \quad \forall i \\ &X \in \mathcal{S}_n \end{aligned} \qquad\qquad \begin{aligned} &\text{Dual} \\ &\min y^T b \\ \text{s.t. } &\sum_i y_i A_i - C \in \mathcal{S}_n \end{aligned}$$

The dual variable y is unconstrained because there is equality in the primal constraint. The dual program is in the second standard form previously discussed in the class. You will verify in the assignment that dual of the dual is the primal SDP.

Exercise 31. What is the dual of the following program?

$$\begin{aligned} &\max \textstyle\sum_{(i,j)\in E} \frac{1-y_i^T y_j}{2} \\ \text{s.t.} \quad &\|y_i\| = 1 \quad \forall i \end{aligned} \tag{4}$$

From the formulation of a dual program, it turns out that the dual program gave an upper/lower bound depending on whether the problem was maximization/minimization respectively.

Are these bounds tight? It turns out that these bounds are tight in most cases of interest (like linear programming). The bounds are not tight in a few contrived cases.

When the primal and dual values agree, *strong duality* is said to hold. *Slater's conditions* provide an easy way to show strong duality for an SDP.

Theorem 6 (Slater's condition). *Given a semidefinite program in standard form with parameters C, A_i, b, suppose the feasible set of primal is $\mathcal{P}$ and feasible set of dual is $\mathcal{D}$. Then strong duality holds if either*

- *If $\mathcal{D} \neq \varnothing$ and there exists a strictly feasible $X \in \mathcal{P}$, i.e., $X \succ 0, A_i \bullet X = b_i \quad \forall i$.*
  or
- *If $\mathcal{P} \neq \varnothing$ and there exists a strictly feasible $y \in \mathcal{D}$, i.e., $\sum_i y_i A_i - C \succ 0$.*

If interested, you can read more about them in any standard text on convex programming, for example [1].

Weak duality and complementary slackness It is instructive to see the direct proof of weak duality. Given any feasible solution X for primal and y for dual,

$$C \bullet X \leq (\sum_i y_i A_i) \bullet X = \sum_i y_i b_i = y^T b. \tag{5}$$

Notice that the above equation implies, all feasible solutions of dual give an upper bound on the optimal value of primal. Similarly any feasible solution for primal gives a lower bound on the optimal value of dual.

Suppose the optimal value of primal is p^*, attained at X^*. Similarly the optimal value of dual is d^* and obtained for y^*. Weak duality implies that $p^* \leq d^*$. Assume that this two values are equal, i.e., $p^* = d^*$ (strong duality). Then from Eqn. 5,

$$C \bullet X = (\sum_i y_i A_i) \bullet X$$
$$\Rightarrow (\sum_i y_i A_i - C) \bullet X = 0$$

The condition above is called the *complementary slackness* condition. So, for optimal X^*, y^* with strong duality, the complementary slackness condition holds. Conversely, if X, y are feasible solutions of primal and dual respectively and satisfy the complementary slackness condition then strong duality holds and $p^* = d^*$.

Exercise 32. What does the complementary slackness condition tell us in case of linear programming?

The above discussion is done for semidefinite program, but with little more effort can be generalized to cone programs.

5 SDP for maximum cut problem

Before we finish, let us look at an application of semidefinite programming, giving the best known approximation algorithm for max-cut problem.

This algorithm was discovered by Goemans and Williamson in 1990's and it gave rise to various other approximation algorithms using similar techniques. The algorithm and the analysis is not difficult, but it introduced a bunch of new ideas.

5.1 Max cut

You have already seen the minimum cut problem in a directed graph. We consider the problem of finding maximum cut in an undirected graph.

Given an *undirected graph* $G = (V, E)$, the *max-cut* problem asks for a subset $S \subseteq V$ such that the number of edges going from S to $\bar{S}$ are maximized.

Many problems in the real world can be formulated as a max-cut instance. For example, suppose the graph gives information about the friendships in a set of students. More concretely, the vertices correspond to students in the class and there is an edge present between two students if they are friends.

The problem is to put them in two different classroom so that the number of friendships (edges) across the two rooms is maximized (one measure to stop copying). This is essentially solving a max-cut problem on the given graph.

The max-cut problem seems very similar to *min-cut* discussed in the course before. We saw some algorithms for the min cut problem. Can those algorithms be used to solve max-cut problem?

One natural approach is to solve min-cut problem in the complementary graph. We take the original graph and switch edges, i.e., we will have edges where there were no edges in the original graph and vice versa.

Exercise 33. Show that this strategy does not work.

5.2 Integer programming formulation

Our first approach would be to look at the integer linear program (ILP) for s-t min-cut and convert it into an ILP for max-cut. For simplicity, we assume that the capacity of every edge is 1.

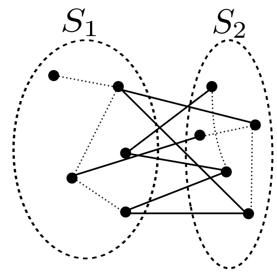

Fig. 1. A cut with edges in solid line, $S_1 = S$ and $S_2 = \bar{S}$

$$\begin{aligned} \max \sum_{(u,v)\in E} d_{u,v} & \\ \text{s.t. } d_{u,v} \geq p_u - p_v & \qquad \forall (u,v) \in E \\ p_u,\ d_{u,v} \in \{0,1\} & \qquad \forall u \in V,\ (u,v) \in E \end{aligned}$$

Notice that the variables p_u indicate the side of cut vertex u is on. Similarly, d_{uv} indicates if the edge (u,v) is part of the cut or not.

Exercise 34. Since this is a maximization problem, show that the optimal value is $|E|$ irrespective of the graph.

We can make another attempt to create an ILP without a trivial solution.

$$\begin{aligned} \max \sum_{(u,v)\in E} d_{u,v} & \\ \text{s.t. } d_{u,v} \leq p_u + p_v & \qquad \forall (u,v) \in E \\ d_{u,v} \leq 2 - (p_u + p_v) & \qquad \forall (u,v) \in E \\ p_u,\ d_{u,v} \in \{0,1\} & \qquad \forall u \in V,\ (u,v) \in E \end{aligned}$$

This will give us a solution of max-cut but is hard to solve. We relax it to an LP.

$$\begin{aligned} \max \sum_{(u,v)\in E} d_{u,v} & \\ \text{s.t. } d_{u,v} \leq p_u + p_v & \qquad \forall (u,v) \in E \\ d_{u,v} \leq 2 - (p_u + p_v) & \qquad \forall (u,v) \in E \\ p_u,\ d_{u,v} \in [0,1] & \qquad \forall u \in V,\ (u,v) \in E \end{aligned}$$

Notice the only change, variables are in $[0,1]$ instead of being integers. Unfortunately, this LP also suffers a setback, its optimal value is $|E|$ irrespective of the graph.

Exercise 35. Show that $p_u = 1/2$ is a feasible solution for the above LP. What should be the values of variables d_{uv}?

The next idea is to let go of linear programming and target semidefinite programming. An integer program is an optimization problem where variables are constrained to be a set of integers. We will create an integer program for max-cut, whose relaxation will be an SDP.

Given a graph $G = (V, E)$, which has $|V| = n$ vertices and $|E| = m$ edges, introduce variable y_i for every vertex $i \in [n]$. Say, $y_i = 1$ if the vertex is assigned to S_1 and $y_i = -1$ if it is assigned to S_2. Then the task is to maximize the number of edges between S_1 and S_2.

If the edge i, j is part of the cut (between S_1 and S_2), $y_i y_j = -1$ otherwise it is $+1$. Then $\sum_{(i,j)\in E} \frac{1-y_iy_j}{2}$ counts the number of edges in the cut. Hence, the following integer program gives us the maximum cut.

$$\begin{array}{rl} & \max \sum_{(i,j)\in E} \frac{1-y_iy_j}{2} \\ \text{s.t.} & y_i \in \{-1, 1\} \quad \forall i \end{array} \tag{6}$$

We have seen before that NP hard problems can be formulated as integer programs. Hence, we can't hope to solve integer programs efficiently (in general).

Even the specific problem of max-cut is known to be NP-hard. So, we do not expect to have a polynomial time algorithm for this problem. Instead, we are interested in finding an approximation algorithm for this problem.

Remember, an *approximation algorithm* outputs a solution which is provably not far from the optimal solution. In our case, we are interested in an algorithm which outputs a cut S, s.t.,

$$\text{number of edges in } S \geq c. \text{ number of edges in max-cut.}$$

Here, $c \leq 1$ is a constant and the algorithm will be called a c-approximation algorithm.

You will show in the assignment that a random cut will have half the edges of the graph and hence will give a randomized .5-approximation algorithm. Can we improve this factor of half? Goemans and Williamson improved this factor to around .87 using semidefinite programming.

Exercise 36. Can you give a deterministic algorithm which gets at least half the edges?

6 SDP relaxation and rounding

Goeman and Williamson gave a *randomized algorithm* for max-cut with approximation factor of around .87. The main ideas/steps for Goemans and Williamson is the following.

1. Convert the integer program into a semidefinite program.
2. Solve the semidefinite program.
3. Then, convert the solution of SDP into an integer solution ($\{-1, +1\}$) again.

This is similar to the technique for designing approximation algorithm of set cover.

Again, the first step of converting the integer program into an SDP is known as *relaxation.* A relaxation of an optimization program is another optimization program which, ideally, is easier to solve and every solution of original program is also a solution of the new program with the same (or related) objective value.

In the case of Eqn. 6, we change the domain of y_i's to be unit vectors instead of integers $\{-1, +1\}$.

$$\begin{array}{rl} & \max \sum_{(i,j)\in E} \frac{1-y_i^T y_j}{2} \\ \text{s.t.} & \|y_i\| = 1 \quad \forall i \end{array} \tag{7}$$

Only change from the integer program is that the y_i's are vectors instead of integers in the new program.

Exercise 37. Show that the relaxed program is an SDP.

It is clear that any solution of Eqn. 6 will still be a solution of Eqn. 7 with same objective value. We just need to consider the integers as one dimensional vectors. Hence, the objective value of the SDP is at least the value of the integer program. The SDP (Eqn. 7) is called the relaxation of the integer program Eqn. 6.

Since the SDP can be solved in polynomial time (with some precision), the solutions of Eqn. 7 can be obtained [1]. Our next goal is to convert these vector solutions into integer ($\{-1, +1\}$) solutions. The process of converting vector solution into integers is called *rounding*.

The SDP value in general will be higher than the integer program. So, while rounding a solution, we expect to lose some factor in the objective. The gap between the SDP value and the integer program value is known as the *integrality gap* of the relaxation.

Exercise 38. What kind of rounding techniques can be possible?

If the solutions were integers, one idea would be put positive numbers on one side and negative numbers on other side. When solutions are vectors, we can say that if a vector lies on one side of the hyperplane, it is one side otherwise on the other side.

Exercise 39. Which hyperplane should we choose?

Notice that the SDP solutions are invariant under rotation, since the value only depends on inner products. The rounding technique given by Goemans and Williamson is very simple and is a randomized rounding procedure. It is done by choosing a random hyperplane, if vectors lie on one side of the hyperplane (say positive) then they are assigned $+1$ otherwise -1.

The complete Goemans and Williamson algorithm can be written as,

1. Given a graph G, solve the SDP relaxation, Eqn. 7, for that graph and obtain the solution $\{y_1, \cdots, y_n\}$.
2. Say the vectors y_i belong to $\mathbb{R}^n$. Choose a random vector $v \in \mathbb{R}^n$.
3. If $y_i^T v \geq 0$ then $y_i' = 1$ otherwise $y_i' = -1$.

How good is this algorithm? Does it produce a cut? If yes, what is the relation between max-cut and the cut obtained by rounding?

The new integer assignment y_i' specifies a cut. Vertices with $y_i' = 1$ go on one side of the cut and $y_i' = -1$ go on the other side. What about the value of this cut?

If it can be shown that $\sum_{(i,j)\in E} \frac{1-y_i'y_j'}{2}$ is a significant proportion of $\sum_{(i,j)\in E} \frac{1-y_i^T y_j}{2}$, we will get a good approximation algorithm.

Exercise 40. What could be the best possible upper bound on

$$\frac{\sum_{(i,j)\in E} \frac{1-y_i'y_j'}{2}}{\sum_{(i,j)\in E} \frac{1-y_i^T y_j}{2}}$$

We know that $\sum_{(i,j)\in E} \frac{1-y_i^T y_j}{2}$ for optimal y is greater than the value of max-cut (why?). We want to show that the ratio,

$$\frac{\sum_{(i,j)\in E} \frac{1-y_i'y_j'}{2}}{\sum_{(i,j)\in E} \frac{1-y_i^T y_j}{2}},$$

is close to 1 with high probability for all possible graphs (worst case bound).

[1] The same algorithms as for linear programs work for SDP's, we will not discuss these algorithms in this course.

7 Analysis of the algorithm

The optimum value of the SDP is

$$S = \sum_{(i,j)\in E} \frac{1 - y_i^T y_j}{2}.$$

We will show that the rounded solution $\{y_1', \cdots, y_n'\}$ has expected value at least cS with $c \approx .8785$. The expectation is taken over random choice of the hyperplane. The idea is, if $y_i^T y_j$ is small, there is a big angle between them. In that case most of the v's will separate y_i and y_j, giving value $y_i' y_j'$ to be close to -1.

Since there are n vectors, we can assume that they live in $\mathbb{R}^n$.

The expected value of the integer solution, using linearity of expectation, can be written as,

$$\mathbb{E}_{v\in\mathbb{R}^n} \sum_{(i,j)\in E} \frac{1 - y_i' y_j'}{2} = \sum_{(i,j)\in E} \mathbb{E}_{v\in\mathbb{R}^n} \frac{1 - y_i' y_j'}{2}.$$

To calculate this, we need the probability that a random v separates vectors y_i and y_j.

$$\mathbb{E}_{v\in\mathbb{R}^n} \frac{1 - y_i' y_j'}{2} = Pr_{v\in\mathbb{R}^n}(y_i' \neq y_j')$$

The angle between vectors y_i and y_j is $\cos^{-1} y_i^T y_j$ (they are unit vectors). Hence, the probability that a random v separates them is

$$\frac{\cos^{-1} y_i^T y_j}{\pi}.$$

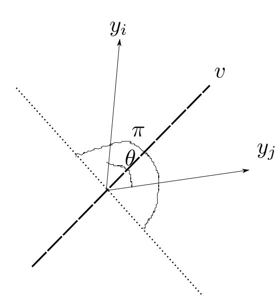

Fig. 2. Random hyperplane separating two vectors

So, the expected value is

$$\sum_{(i,j)\in E} \frac{\cos^{-1} y_i^T y_j}{\pi}$$

Now use the change of variable $y_i^T y_j = \cos\theta_{ij}$.

$$
\begin{aligned}
& \sum_{(i,j)\in E} \frac{\cos^{-1} y_i^T y_j}{\pi} \\
= & \sum_{(i,j)\in E} \frac{\theta_{ij}}{\pi} \\
= & \frac{1}{\pi} \sum_{(i,j)\in E} \frac{\theta_{ij}}{1-\cos\theta ij}(1 - y_i^T y_j) \\
\geq & \left(\min_{0\leq\theta\leq\pi} \frac{2\theta}{\pi(1-\cos\theta)}\right) S \\
\geq & \; cS
\end{aligned}
$$

Exercise 41. Give a lower bound on $c = \min_{0\leq\theta\leq\pi} \frac{2\theta}{\pi(1-\cos\theta)}$.

Using calculus we can show that $c \geq .8785$. Hence,

$$E(Obj(y_i')) \geq c \;\; Obj(y_i) \geq c \;\; max - cut(G).$$

Here $Obj(x)$ denotes the objective value for the solution x and $max - cut(G)$ is the maximum cut in G. The second inequality follows from the fact that SDP is relaxation of original maximum cut integer program. Hence, the algorithm given above is a c approximation algorithm for max-cut. Like before, we can use Markov inequality to convert the statement into: with high probablity we will get a big cut.

Exercise 42. Convince yourself that above equation implies a c-approximation algorithm for max-cut.

Consequences of the algorithm One consequence from the previous analysis is that the SDP gives a nearly tight bound on the value of the integer program. In general, once an optimization program is relaxed, there is no guarantee about the value of the relaxed program. The feasible set is increased and hence the objective value can be arbitrarily higher (in case of maximization problem) or lower (minimization problems) as compared to the optimum value of the original program.

The *rounding* provides a proof that the relaxed value is comparable to the original value. In the case of max-cut, the SDP value is definitely at least the value of integer program. Rounding shows that the SDP value is less than $\frac{1}{c} \;\; Obj(y_i')$, which is less than $\frac{1}{c} \;\; max - cut(G)$.

This shows that SDP gives a pretty tight bound on the value of the integer program.

$$max - cut(G) \leq Opt(SDP) \leq \frac{1}{c} \;\; max - cut(G).$$

8 Assignment

Exercise 43. Prove that if λ is a root of the characteristic polynomial, then there exist at least one eigenvector for λ.

Exercise 44. Show that the matrix M and M^T have the same singular values.

Exercise 45. Read about polar decomposition and prove it using singular value decomposition.

Exercise 46. Read about tensor product of two matrices in Wikipedia.

Exercise 47. What are the eigenvalues of $A \otimes B$, where A, B are symmetric matrices and $\otimes$ denotes the tensor product?

Exercise 48. Give a characterization of the linear operators over $V \otimes W$ in terms of linear operators over V and W. Remember that they form a vector space.

Exercise 49. Show that $\langle v|A|w\rangle = \sum_{ij} A_{ij} v_i^* w_j$.

Exercise 50. The multiplicity of a root of characteristic polynomial is known as the *algebraic multiplicity*. The dimension of the eigenspace of that root is known as the *geometric multiplicity*. It is known that geometric multiplicity is less than algebraic multiplicity.

Show that the geometric multiplicity is same as algebraic multiplicity for a symmetric matrix M.

Exercise 51. Show that semidefinite program can be viewed as an optimization problem with linear cost function and infinite linear constraints.

Exercise 52. Prove that we don't need to put the explicit constraint that the off-diagonal entries (blocks) are zero, when we converted form with positive semidefinite constraint into standard form.

Exercise 53. Prove that for a single variate polynomial p, it is positive iff it can be written as a sum of squares (Hint: Look at the factorization of p in complex domain).

Exercise 54. Show that the following problem can be converted into a SDP.

$$\begin{aligned} &\min \tfrac{(c^T x)^2}{d^T x} \\ \text{s.t.}\quad &Ax + b \geq 0 \end{aligned}$$

Here, x is a vector and c, d are vectors of the same dimension as x.

Exercise 55. Show that the programs given in the *counterexample to strong duality* section are dual of each other.

Exercise 56. Convert the dual of the standard SDP into the standard form, take dual and show that you get the original SDP.

Exercise 57. Remember the SDP for finding the maximum eigenvalue of a matrix M. Take its dual and show that it computes the maximum eigenvalue (without using duality).

Exercise 58. Learn about Lovasz theta number and the associated semidefinite program. Show that it is a relaxation of maximum independent set problem.

Exercise 59. Given an undirected graph $G = (V, E)$, show that a random cut in a graph will have expected value $\frac{|E|}{2}$. How does this give a randomized approximation algorithm with factor .5?

Exercise 60. Give a deterministic .5 approximation algorithm for max-cut.

Hint: use induction.

References

1. Stephen Boyd and Lieven Vandenberghe. *Convex optimization.* Cambridge university press, 2004.
2. Gilbert Strang. *Introduction to Linear Algebra.* Wellesley-Cambridge Press, 2009.