CS 294 Sum of Squares **Fall 2018**

Lecture 2: Introduction to Semidefinite Programming

Lecturer: Prasad Raghavendra Scribe: Antares Chen

In these notes, we review basic facts about positive semidefinite matrices and semidefinite programming. We then use these facts to introduce the degree-2 sum of squares relaxation for MAXCUT. Throughout these notes, we will consider vectors v as column vectors only.

2.1 Positive Semidefinite Matrices

Positive semidefinite (PSD) matrices will be fundamental to our discussion of the Sum of Squares semidefinite program. To begin, let us review some basic facts regarding real symmetric matrices. Recall that $A \in \mathbb{R}^{N \times N}$ is a *real symmetric matrix* if each entry is a real number and if $A^\top = A$. A property of real symmetric matrices that we will repeatedly use is that they always admit *spectral decompositions*:

Theorem 2.1. *Let $A \in \mathbb{R}^{N \times N}$ be a real symmetric matrix with not necessarily distinct eigenvalues $\lambda_1, \ldots, \lambda_N$. Then $A = \sum_{i=1}^N \lambda_i v_i v_i^\top$ where $v_1, \ldots, v_n$ are real eigenvectors of A and the set $\{v_1, \ldots, v_n\}$ forms an orthonormal eigenbasis of $\mathbb{R}^n$. Furthermore, the decomposition is unique if $\lambda_i \neq \lambda_j$ for all $i \neq j$.*

A real symmetric matrix $A \in \mathbb{R}^{N \times N}$ is *positive semidefinite* (denoted as $A \succeq 0$) if $\lambda_i \geq 0$ for all eigenvalues λ_i of A. We can use theorem 2.1 to provide multiple equivalent definitions of PSD matrices.

Theorem 2.2. *Given a real symmetric matrix $A \in \mathbb{R}^{N \times N}$ with not necessarily distinct eigenvalues $\lambda_1, \ldots, \lambda_N$, the following are equivalent*

(1) For all $x \in \mathbb{R}^N$, we have that $x^\top A x = \sum_{i,j} A_{ij} x_i x_j \geq 0$.

(2) $\lambda_i \geq 0$ for all $i = 1, \ldots, N$.

(3) A admits a (not necessarily unique) admits a Cholesky factorization $A = UU^\top$ for $U \in \mathbb{R}^{N \times N}$. In other words, there exists $u_1, \ldots, u_N \in \mathbb{R}^N$ such that $A_{ij} = \langle u_i, u_j \rangle$.

(4) There exists real-valued random variables $g_1, \ldots, g_N$ such that $A_{ij} = \mathbf{E}\{g_i g_j\}$.

Proof. Our proof proceeds via a chain of implications.

(1) $\implies$ (2) Consider the eigenvector v_i and observe that

$$v_i^\top A v_i = \langle v_i, A v_i \rangle = \langle v_i, \lambda v_i \rangle = \lambda_i \langle v_i, v_i \rangle = \lambda_i$$

By assumption, $x^\top A x \geq 0$ for any $x \in \mathbb{R}^N$ hence $\lambda_i \geq 0$ for all eigenvalues λ_i.

(2) $\implies$ (3) We wish to construct a Cholesky factorization of A. By theorem 2.1, we can write A as

$$A = \sum_{i=1}^{N} \lambda_i v_i v_i^\top = \begin{pmatrix} v_1 & \dots & v_N \end{pmatrix} \begin{pmatrix} \lambda_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \lambda_N \end{pmatrix} \begin{pmatrix} v_1^\top \\ \vdots \\ v_N \end{pmatrix}$$

$\lambda \geq 0$ hence we can split the center diagonal matrix into the following product.

$$\begin{pmatrix} v_1 & \dots & v_N \end{pmatrix} \begin{pmatrix} \lambda_1^{1/2} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \lambda_N^{1/2} \end{pmatrix} \begin{pmatrix} \lambda_1^{1/2} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \lambda_N^{1/2} \end{pmatrix} \begin{pmatrix} v_1^\top \\ \vdots \\ v_N \end{pmatrix} = V \Lambda^{1/2} \Lambda^{1/2} V^\top$$

With $U = V\Lambda^{1/2}$, we have that $A = UU^\top$.

(3) $\implies$ (4) By assumption, there exists vectors $v_1, \dots v_N$ such that $A_{ij} = \langle v_i, v_j \rangle$. We will select our random variables to be the projections of v_i onto a random Gaussian direction. Let $g \sim \big(\mathcal{N}(0,1)\big)^N$ and define $g_i = \langle v_i, g \rangle$ for all $i = 1, \dots, N$. The covariance between g_i, g_j is then given by

$$\mathbf{E}\{g_i g_j\} = \mathbf{E}_g\{\langle v_i, g\rangle \langle v_j, g\rangle\} = \mathbf{E}_g\Big\{ \Big(\sum_{k=1}^{N} v_{ik} g_k \Big) \Big(\sum_{k=1}^{N} v_{jk} g_k \Big) \Big\} = \sum_{k=1}^{N} \sum_{\ell=1}^{N} v_{ik} v_{j\ell} \mathbf{E}\{g_k g_\ell\}$$

However, the coordinates of g are sampled independently of one another implying $g_k g_\ell = \mathbf{1}\{k = \ell\}$. Thus

$$\sum_{k=1}^{N} \sum_{\ell=1}^{N} v_{ik} v_{j\ell} \mathbf{1}\{k = \ell\} = \sum_{k=1}^{N} v_{ik} v_{jk} = \langle v_i, v_j \rangle$$

We thus have that $A_{ij} = \mathbf{E}\{g_i g_j\}$ as required.

(4) $\implies$ (1) Given that $A_{ij} = \mathbf{E}\{g_i g_j\}$ for all coordinates i, j, we wish to show $x^\top A x \geq 0$ for all $x \in \mathbb{R}^N$. Observe

$$x^\top A x = \sum_{i=1}^{N} \sum_{j=N} A_{ij} x_i x_j = \sum_{i=1}^{N} \sum_{j=N} \mathbf{E}\{g_i g_j\} x_i x_j = \mathbf{E}\Big\{ \sum_{i=1}^{N} \sum_{j=1}^{N} g_i g_j x_i x_j \Big\} = \mathbf{E}\Big\{ \Big(\sum_{i=1}^{N} g_i x_i \Big)^2 \Big\}$$

We have that $x^\top A x \geq 0$. □

A few brief remarks regarding the previous proof and PSD matrices; first note that in demonstrating **(1) $\implies$ (2)**, appending any unitary matrix to the definition of U will not affect $A = UU^\top$ hence why the Cholesky decomposition is not necessarily unique. This case also provides a very nice way to think of PSD matrices since we will often think of a PSD matrix A as $UU^\top$[1].

[1]Sometimes it is worth thinking about a real symmetric matrix A as a linear operator that scales the vector space along the orthonormal eigenbasis since A can always be decomposed into a sum of rank one matrices by theorem 2.1 but we will not see this as often.

It might also have been strange to have used a Gaussian random variable in **(3)** $\implies$ **(4)** when any pairwise independent random variable would admit the same proof. We choose to do this because using a Gaussian provides a nice geometry for analyzing certain algorithms that we will discuss further on in the class.

Finally, we note that it is possible to define a partial ordering on PSD matrices by denoting $A \succeq B$ for PSD matrices A, B if $A - B \succeq 0$.

2.2 Semidefinite Programming

Now that we have characterized PSD matrices, we can discuss semidefinite programming. For ease of notation, let us denote $\langle A, B \rangle$ for matrices $A, B \in \mathbb{R}^{N \times N}$ by

$$\langle A, B \rangle = \sum_{i=1}^{N} \sum_{j=1}^{N} A_{ij} B_{ij}$$

Given $A_1, \ldots, A_m, C, X \in \mathbb{R}^{N \times N}$ and $b_1, \ldots, b_m \in \mathbb{R}$, a *semidefinite program* (SDP) is defined as the following optimization problem

$$\begin{array}{ll} \text{maximize} & \langle C, X \rangle \\ \text{subject to} & \langle A_1, X \rangle \leq b_1 \\ & \cdots \\ & \langle A_m, X \rangle \leq b_m \\ & X \succeq 0 \end{array}$$

We can often think of this as a linear program over N^2 variables X_{ij} with the additional constraint that $X \succeq 0$ where $X = \big(X_{ij}\big)$. It is even possible to think of the constraint $X \succeq 0$ as an infinite number of linear constraints on X as $X \succeq 0$ implies for all $v \in \mathbb{R}^N$ we have $v^\top X v \geq 0$. This view is sometimes useful when verifying the dual of an SDP.

Alternatively, we can consider this as a linear program over inner products of vectors since $X \succeq 0$ implies for all i, j we have $X_{ij} = \langle v_i, v_j \rangle$ for some v_i, v_j. Rewritting the SDP as a program on inner products creates what is sometimes called a *vector program*.

On matters computational, SDPs can be approximately solved in polynomial time using the Ellipsoid Algorithm and Matrix Multiplicative Weights (both of which will be visited later on in the course).

2.3 Degree-2 Sum of Squares Relaxation for maxcut

As an example of Semidefinite Programming, we will derive an SDP relaxation of the MAXCUT problem. Recall that in MAXCUT, we are given a graph $G = (V, E)$ with the objective of finding a cut $S \subseteq V$ maximizing the

number of crossing edges $E(S, \bar{S}) = |\{(u, v) \in E : u \in S, v \in \bar{S}\}|$. For each vertex $i \in V$, we define variables

$$x_i = \begin{cases} +1 & \text{if } i \in S \\ -1 & \text{otherwise} \end{cases}$$

Notice that $(x_i - x_j)^2 = 4$ when $i \in S, j \in \bar{S}$, otherwise $(x_i - x_j)^2 = 0$. The value of $E(S, \bar{S})$ is given by

$$E(S, \bar{S}) = \frac{1}{4} \sum_{(i,j) \in E} (x_i - x_j)^2$$

Denote $G(x) = \sum_{(i,j) \in E} (x_i, x_j)^2$ and $|V| = n$. The solution to the following optimization problem will exactly be the maxcut of G. The constant factor at the beginning of $E(S, \bar{S})$ is removed since it is positive and thus does not affect the optimal solution.

$$\begin{aligned} &\text{maximize} && G(x) \\ &\text{subject to} && x \in \{+1, -1\}^n \end{aligned} \tag{2.1}$$

Let us now use this as a starting point for deriving an SDP relaxation of MAXCUT.

2.3.1 Convexifying the Feasible Set

The first step is to convexify equation 2.1. The boolean hypercube is non-convex and difficult to optimize over, so to make the feasible set convex, we instead optimize over distributions whose supports are on points in the boolean hypercube. In particular, we solve the following program

$$\begin{aligned} &\text{maximize} && \sum_{x \in \{+1,-1\}^n} \mu(x) G(x) \\ &\text{subject to} && \mu(x) \geq 0 \qquad \forall x \in \{+1, -1\}^n \\ & && \sum_{x \in \{+1,-1\}^n} \mu(x) = 1 \end{aligned} \tag{2.2}$$

The objective for this optimization problem is simply the expectation of $G(x)$ when $x \sim \mu$. We rewrite equation 2.2 as:

$$\begin{aligned} &\text{maximize} && \mathop{\mathbf{E}}_{x \sim \mu} \{G(x)\} \\ &\text{subject to} && \mu(x) \geq 0 \qquad \forall x \in \{+1, -1\}^n \\ & && \sum_{x \in \{+1,-1\}^n} \mu(x) = 1 \end{aligned} \tag{2.3}$$

It is important to note that we have not lost any information from the original optimization problem. We have maintained the original solution because we can still recover the optimal maxcut for G. In particular, the distribution μ which maximizes the expectation of $G(x)$ always has its support on the optimal cut.

2.3.2 Reducing the Constraint Set

Next, we reduce the number of constraints. Equation 2.3 contains exponentially many non-negativity constraints on μ because it maintains a probability for each point on the boolean hypercube. This is a bit unnecessary since our goal is to calculate the expectation of $G(x)$, a degree 2 polynomial, which can be done even if given only the degree 2 moments of μ.

We define the the moments, up to degree d, of a probability distribution μ over $\mathbb{R}^n$ as

$$X_d(\mu) = \left\{ X_\sigma = \mathop{\mathbf{E}}_{x\sim\mu}\left\{ \prod_{i\in\sigma} x_i \right\} : |\sigma| \leq d \right\}$$

where σ is a multiset of elements from $[n]$. For example:

$$\begin{aligned} X_{\{i\}} &= \mathop{\mathbf{E}}_{x\sim\mu}\{x_i\} \\ X_{\{i,j\}} &= \mathop{\mathbf{E}}_{x\sim\mu}\{x_i x_j\} \\ X_{\{i,i\}} &= \mathop{\mathbf{E}}_{x\sim\mu}\{x_i x_i\} \end{aligned}$$

Indeed, the power of maintaining the moments of μ up to degree d, is that we can now calculate the expectation of any degree d polynomial due to linearity of expectations. Let us denote the set of all degree 2 moments of distributions supported on $\{+1,-1\}$ as $\mathcal{S}_2$. To make $\mathcal{S}_2$ more concrete, we note its elements (and similarly defined sets over distributions on $\mathcal{R}^n$) have a succinct representation:

$$X_2(\mu) = \begin{array}{c} \\ \emptyset \\ \{j\} \\ \\ \{n\} \end{array} \begin{array}{c} \begin{array}{cccccc} \emptyset & & \{i\} & & \{n\} \end{array} \\ \begin{pmatrix} \mathbf{E}\{1\} & \dots & \mathbf{E}\{x_i\} & \dots & \mathbf{E}\{x_n\} \\ \vdots & \ddots & \vdots & & \vdots \\ \mathbf{E}\{x_j\} & \dots & \mathbf{E}\{x_i x_j\} & \dots & \mathbf{E}\{x_j x_n\} \\ \vdots & & \vdots & \ddots & \vdots \\ \mathbf{E}\{x_n\} & \dots & \mathbf{E}\{x_i x_n\} & \dots & \mathbf{E}\{x_n^2\} \end{pmatrix} \end{array}$$

Specifically, the indices of what we call the *degree-2 moment matrix* are multisets of $[n]$ of size at most 1 with the convention that $X_\emptyset = 1$. Using this, we can see that $\mathcal{S}_2 \subseteq \mathbb{R}^{(n+1)\times(n+1)}$ is also a convex set. In particular, for any $\alpha \in [0,1]$, we have for two distributions μ, μ' over $\{+1,-1\}^n$ that

$$\alpha X_2(\mu) + (1-\alpha) X_2(\mu') = X_2(\alpha\mu + (1-\alpha)\mu')$$

since

$$\alpha \mathop{\mathbf{E}}_{x\sim\mu}\{x_i x_j\} + (1-\alpha) \mathop{\mathbf{E}}_{x\sim\mu'}\{x_i x_j\} = \mathop{\mathbf{E}}_{x\sim\alpha\mu+(1-\alpha)\mu'}\{x_i x_j\}$$

Reformulating equation 2.3 using the deg-2 moments, we have the program.

$$\begin{aligned} \text{maximize} \quad & \sum_{(i,j)\in E} (X_{ii} + X_{jj} - 2X_{ij}) \\ \text{subject to} \quad & X_{ii} = 1 \\ & X \in \mathcal{S}_2 \end{aligned} \tag{2.4}$$

The constraint $X_{ii} = 1$ corresponds to the requirement that x is on the boolean hypercube. In particular, $x_i \in \{+1, -1\}$ implies $\mathbf{E}\{x_i^2\} = 1$. Additionally, we still have not lost any information. It is remains possible to fully calculate the objective function over a feasible set which maintains the optimal maxcut for G in the support of distribution μ with moment matrix X.

2.3.3 Creating the Final Relaxation

Finally, we create the relaxation. Though equation 2.4 is a convex optimization problem, it is still NP-Hard to solve. In fact, the set $\mathcal{S}_2$ itself should be hard to optimize over, otherwise many NP-Hard optimization problems on the boolean hypercube would have efficient algorithms (such as MAXCUT).

Let us relax equation 2.4 to create something efficiently solvable. First, replace the constraint $X \in \mathcal{S}_2$ with $X \succeq 0$. This is due to the claim

Claim 2.3. *For any distribution μ supported on $\{+1, -1\}^n$, $X_2(\mu) \succeq 0$.*

which follows immediately from theorem 2.2 and that $X_2(\mu)$ is a correlation matrix. All deg-2 moment matrices are PSD, but not all PSD matrices are moment matrices for an appropriate real distribution. We have now lost information because our feasible set is strictly larger under $X \succeq 0$. Our final relaxation is

$$\begin{aligned} \text{maximize} \quad & \sum_{(i,j)\in E} (X_{ii} + X_{jj} - 2X_{ij}) \\ \text{subject to} \quad & X_{ii} = 1 \qquad \forall i \in V \\ & X \succeq 0 \end{aligned} \tag{2.5}$$

This is known as the *degree-2 sum of squares relaxation* for MAXCUT.

2.4 Conclusion

We have now reviewed basic facts regarding positive semidefinite matrices and semidefinite programming as well as introduced the degree-2 sum of squares relaxation for MAXCUT. Though the steps we took to derive the relaxation may have been a bit circuitous, we did so to highlight the generality of the construction. Our derivation does not require any properties intrinsic to MAXCUT or the hypercube, thus it can also be carried out for other discrete optimization problems. We will certainly encounter this and more as the course progresses.