Chapter 24

Approximate Max Cut

CS 573: Algorithms, Fall 2013
November 19, 2013

24.1 Normal distribution

24.1.0.1 Normal distribution – proof

\[ \tau^2 = \left( \int_{x=-\infty}^{\infty} \exp \left( -\frac{x^2}{2} \right) \, dx \right)^2 \]

\[ = \int_{(x,y) \in \mathbb{R}^2} \exp \left( -\frac{x^2 + y^2}{2} \right) \, dx \, dy \quad \text{Change of vars:} \quad x = r \cos \alpha, \quad y = r \sin \alpha \]

\[ = \int_{\alpha=0}^{2\pi} \int_{r=0}^{\infty} \exp \left( -\frac{r^2}{2} \right) \left| \det \begin{pmatrix} \frac{\partial r \cos \alpha}{\partial r} & \frac{\partial r \cos \alpha}{\partial \alpha} \\ \frac{\partial r \sin \alpha}{\partial r} & \frac{\partial r \sin \alpha}{\partial \alpha} \end{pmatrix} \right| \, dr \, d\alpha \]

\[ = \int_{\alpha=0}^{2\pi} \int_{r=0}^{\infty} \exp \left( -\frac{r^2}{2} \right) \left| \det \begin{pmatrix} \cos \alpha & -r \sin \alpha \\ \sin \alpha & r \cos \alpha \end{pmatrix} \right| \, dr \, d\alpha \]

\[ = \int_{\alpha=0}^{2\pi} \int_{r=0}^{\infty} \exp \left( -\frac{r^2}{2} \right) r \, dr \, d\alpha \]

\[ = \int_{\alpha=0}^{2\pi} \left[ -exp \left( -\frac{r^2}{2} \right) \right]_{r=0}^{\infty} \, d\alpha = \int_{\alpha=0}^{2\pi} 1 \, d\alpha = 2\pi \]

24.1.0.2 Multidimensional normal distribution

(A) A random variable \( X \) has \textbf{normal distribution} if \( \Pr[X = x] = \frac{1}{\sqrt{2\pi}} \exp(-x^2/2) \).

(B) \( X \sim N(0,1) \).

(C) A vector \( \mathbf{x} = (x_1, \ldots, x_n) \) has \( d \)-dimensional normal distributed (i.e., \( \mathbf{v} \sim N^n(0,1) \) if \( v_1, \ldots, v_n \sim N(0,1) \)).

(D) Consider a vector \( \mathbf{v} \in \mathbb{R}^n \), such that \( \| \mathbf{v} \| = 1 \). Let \( \mathbf{x} \sim N^n(0,1) \). Then \( z = \langle \mathbf{v}, \mathbf{x} \rangle \) has normal distribution!
24.2 Approximate Max Cut

24.2.1 The movie so far...

24.2.1.1 Summary: It sucks.

(A) Seen: Examples of using rounding techniques for approximation.
(B) So far: Relaxed optimization problem is LP.
(C) But... We know how to solve convex programming.
(D) Convex programming $\gg$ LP.
(E) Convex programming can be solved in polynomial time.
(F) Solving convex programming is outside scope: assume doable in polynomial time.
(G) Today’s lecture:
   (A) Revisit MAX CUT.
   (B) Show how to relax it into semi-definite programming problem.
   (C) Solve relaxation.
   (D) Show how to round the relaxed problem.

24.2.2 Problem Statement: MAX CUT

24.2.2.1 Since this is a theory class, we will define our problem.

(A) $G = (V, E)$: undirected graph.
(B) $\forall i,j \in E$: nonnegative weights $\omega_{ij}$.
(C) MAX CUT (maximum cut problem): Compute set $S \subseteq V$ maximizing weight of edges in cut $(S, \overline{S})$.
(D) $ij \notin E \implies \omega_{ij} = 0$.
(E) weight of cut: $w(S, \overline{S}) = \sum_{i \in S, j \in \overline{S}} \omega_{ij}$.
(F) Known: problem is NP-Complete.
   Hard to approximate within a certain constant.

24.2.3 Max cut as integer program

24.2.3.1 because what can go wrong?

(A) Vertices: $V = \{1, \ldots, n\}$.
(B) $\omega_{ij}$: non-negative weights on edges.
(C) max cut $w(S, \overline{S})$ is computed by the integer quadratic program:

\[
\begin{align*}
(Q) \quad \text{max} & \quad \frac{1}{2} \sum_{i<j} \omega_{ij}(1 - y_i y_j) \\
\text{subject to:} & \quad y_i \in \{-1, 1\} \quad \forall i \in V.
\end{align*}
\]

(D) Set: $S = \{i \mid y_i = 1\}$.
(E) $\omega(S, \overline{S}) = \frac{1}{2} \sum_{i,j} \omega_{ij}(1 - y_i y_j)$. 

2
24.2.4 Relaxing $-1, 1$...

24.2.4.1 Because $1$ and $-1$ are just vectors.

(A) Solving quadratic integer programming is of course **NP-Hard**.
(B) Want a relaxation...
(C) $1$ and $-1$ are just roots of unity.
(D) FFT: All roots of unity are a circle.
(E) In higher dimensions: All unit vectors are points on unit sphere.
(F) $y_i$ are just unit vectors.
(G) $y_i * y_j$ is replaced by dot product $\langle y_i, y_j \rangle$.

24.2.5 Quick reminder about dot products

24.2.5.1 Because not everybody remembers what they did in kindergarten

(A) $x = (x_1, \ldots, x_d), y = (y_1, \ldots, y_d)$.
(B) $\langle x, y \rangle = \sum_{i=1}^{d} x_i y_i$.
(C) For a vector $v \in \mathbb{R}^d$: $\|v\|^2 = \langle v, v \rangle$.
(D) $\langle x, y \rangle = \|x\| \|y\| \cos \alpha$.
   $\alpha$: Angle between $x$ and $y$.

(E) $x \perp y$: $\langle x, y \rangle = 0$.
(F) $x = y$ and $\|x\| = \|y\| = 1$: $\langle x, y \rangle = 1$.
(G) $x = -y$ and $\|x\| = \|y\| = 1$: $\langle x, y \rangle = -1$.

24.2.6 Relaxing $-1, 1$...

24.2.6.1 Because $1$ and $-1$ are just vectors.

(A) max cut $w(S, \bar{S})$ as integer quadratic program:

\[
\begin{align*}
\text{(Q)} & \quad \max \quad \frac{1}{2} \sum_{i<j} \omega_{ij} (1 - y_i y_j) \\
& \quad \text{subject to: } y_i \in \{-1, 1\} \quad \forall i \in V.
\end{align*}
\]

(B) Relaxed semi-definite programming version:

\[
\begin{align*}
\text{(P)} & \quad \max \quad \gamma = \frac{1}{2} \sum_{i<j} \omega_{ij} \left(1 - \langle v_i, v_j \rangle \right) \\
& \quad \text{subject to: } v_i \in S^{(n)} \quad \forall i \in V,
\end{align*}
\]

$S^{(n)}$: $n$ dimensional unit sphere in $\mathbb{R}^{n+1}$.

24.2.6.2 Discussion...

(A) semi-definite programming: special case of convex programming.
(B) Can be solved in polynomial time.
(C) Solve within a factor of $(1 + \varepsilon)$ of optimal, for any $\varepsilon > 0$, in polynomial time.
(D) Intuition: vectors of one side of the cut, and vertices on the other sides, would have faraway vectors.

### 24.2.6.3 Approximation algorithm

(A) Given instance, compute SDP (P).
(B) Compute optimal solution for (P).
(C) generate a random vector $\vec{r}$ on the unit sphere $S^{(n)}$.
(D) induces hyperplane $h \equiv \langle \vec{r}, x \rangle = 0$
(E) assign all vectors on one side of $h$ to $S$, and rest to $\overline{S}$.

$$S = \{v_i \mid \langle v_i, \vec{r} \rangle \geq 0\}.$$

### 24.2.7 Analysis

#### 24.2.7.1 Analysis...

Intuition: with good probability, vectors in the solution of (P) that have large angle between them would be separated by cut.

**Lemma 24.2.1.** $\Pr[\text{sign}(\langle v_i, \vec{r} \rangle) \neq \text{sign}(\langle v_j, \vec{r} \rangle)] = \frac{1}{\pi} \arccos(\langle v_i, v_j \rangle) = \frac{\tau}{\pi}$.

#### 24.2.7.2 Proof...

(A) Think $v_i, v_j$ and $\vec{r}$ as being in the plane.
(B) ... reasonable assumption!
   (A) $g$: plane spanned by $v_i$ and $v_j$.
   (B) Only care about signs of $\langle v_i, \vec{r} \rangle$ and $\langle v_j, \vec{r} \rangle$
   (C) can be decided by projecting $\vec{r}$ on $g$... and normalizing it to have length 1.
   (D) Sphere is symmetric $\implies$ sampling $\vec{r}$ from $S^{(n)}$ projecting it down to $g$, and then normalizing it
      $\equiv$ choosing uniformly a vector from the unit circle in $g$
24.2.7.3 Proof via figure...

\[ \tau = \arccos(\langle v_i, v_j \rangle) \]

24.2.7.4 Proof...

(A) Think \( v_i, v_j \) and \( \vec{r} \) as being in the plane.

(B) \( \text{sign}(\langle v_i, \vec{r} \rangle) \neq \text{sign}(\langle v_j, \vec{r} \rangle) \) happens only if \( \vec{r} \) falls in the double wedge formed by the lines perpendicular to \( v_i \) and \( v_j \).

(C) angle of double wedge = angle \( \tau \) between \( v_i \) and \( v_j \).

(D) \( v_i \) and \( v_j \) are unit vectors: \( \langle v_i, v_j \rangle = \cos(\tau) \).

\[ \tau = \angle v_i v_j \]

(E) Thus,

\[ \Pr[\text{sign}(\langle v_i, \vec{r} \rangle) \neq \text{sign}(\langle v_j, \vec{r} \rangle)] = \frac{2\tau}{2\pi} \]

\[ = \frac{1}{\pi} \cdot \arccos(\langle v_i, v_j \rangle), \]

as claimed.

24.2.7.5 Theorem

**Theorem 24.2.2.** Let \( W \) be the random variable which is the weight of the cut generated by the algorithm. We have

\[ \mathbb{E}[W] = \frac{1}{\pi} \sum_{i<j} \omega_{ij} \arccos(\langle v_i, v_j \rangle). \]

24.2.7.6 Proof

(A) \( X_{ij} \): indicator variable = 1 \( \iff \) edge \( ij \) is in the cut.

(B) \( \mathbb{E}[X_{ij}] = \Pr[\text{sign}(\langle v_i, \vec{r} \rangle) \neq \text{sign}(\langle v_j, \vec{r} \rangle)] \]

\[ = \frac{1}{\pi} \arccos(\langle v_i, v_j \rangle), \text{ by lemma.} \]

(C) \( W = \sum_{i<j} \omega_{ij} X_{ij} \), and by linearity of expectation...

\[ \mathbb{E}[W] = \sum_{i<j} \omega_{ij} \mathbb{E}[X_{ij}] = \frac{1}{\pi} \sum_{i<j} \omega_{ij} \arccos(\langle v_i, v_j \rangle). \]
Lemma 24.2.3. For \(-1 \leq y \leq 1\), we have 
\[
\frac{\arccos(y)}{\pi} \geq \alpha \cdot \frac{1}{2}(1 - y), \quad \text{where} \quad \alpha = \min_{0 \leq \psi \leq \pi} \frac{2}{\pi} \frac{\psi}{1 - \cos(\psi)}.
\]

Proof: Set \(y = \cos(\psi)\). The inequality now becomes \(\frac{\psi}{\pi} \geq \alpha \frac{1}{2}(1 - \cos(\psi))\). Reorganizing, the inequality becomes \(\frac{2}{\pi} \frac{\psi}{1 - \cos(\psi)} \geq \alpha\), which trivially holds by the definition of \(\alpha\).  

Lemma 24.2.4. \(\alpha > 0.87856\).

Proof: Using simple calculus, one can see that \(\alpha\) achieves its value for \(\psi = 2.331122\ldots\), the nonzero root of \(\cos \psi + \psi \sin \psi = 1\).  

Theorem 24.2.5. The above algorithm computes in expectation a cut with total weight \(\alpha \cdot \text{Opt} \geq 0.87856\text{Opt}\), where \(\text{Opt}\) is the weight of the maximal cut.

Proof: Consider the optimal solution to \((P)\), and let its value be \(\gamma \geq \text{Opt}\). By lemma:
\[
E[W] = \frac{1}{\pi} \sum_{i<j} \omega_{ij} \arccos(\langle v_i, v_j \rangle) \\
\geq \sum_{i<j} \omega_{ij} \alpha \frac{1}{2}(1 - \langle v_i, v_j \rangle) = \alpha \gamma \geq \alpha \cdot \text{Opt}.
\]

24.3 Semi-definite programming

24.3.0.10 SDP: Semi-definite programming

(A) \(x_{ij} = \langle v_i, v_j \rangle\).
(B) \(M\): \(n \times n\) matrix with \(x_{ij}\) as entries.
(C) \(x_{ii} = 1\), for \(i = 1, \ldots, n\).
(D) \(V\): matrix having vectors \(v_1, \ldots, v_n\) as its columns.
(E) \(M = V^T V\).
(F) \(\forall v \in \mathbb{R}^n: v^T M v = v^T A^T A v = (Av)^T (Av) \geq 0\).
(G) \(M\) is positive semidefinite (PSD).
(H) Fact: Any PSD matrix \(P\) can be written as \(P = B^T B\).
(I) Furthermore, given such a matrix \(P\) of size \(n \times n\), we can compute \(B\) such that \(P = B^T B\) in \(O(n^3)\) time.
(J) Known as Cholesky decomposition.
24.3.0.11 SDP: Semi-definite programming

(A) If PSD $P = B^TB$ has a diagonal of 1
(B) $B$ has columns which are unit vectors.
(C) If solve SDP (P), get back semi-definite matrix...
(D) ... recover the vectors realizing the solution (i.e., compute $B$)
(E) Now, do the rounding.
(F) SDP (P) can be restated as

\[ \begin{array}{ll}
\text{(SD)} & \max \frac{1}{2} \sum_{i<j} \omega_{ij} (1 - x_{ij}) \\
\text{subject to:} & x_{ii} = 1 \quad \text{for } i = 1, \ldots, n \\
& (x_{ij})_{i=1,\ldots,n,j=1,\ldots,n} \text{ is a PSD matrix.}
\end{array} \]

24.3.0.12 SDP: Semi-definite programming

(A) SDP is

\[ \begin{array}{ll}
\text{(SD)} & \max \frac{1}{2} \sum_{i<j} \omega_{ij} (1 - x_{ij}) \\
\text{subject to:} & x_{ii} = 1 \quad \text{for } i = 1, \ldots, n \\
& (x_{ij})_{i=1,\ldots,n,j=1,\ldots,n} \text{ is a PSD matrix.}
\end{array} \]

(B) find optimal value of a linear function...
(C) ... over a set which is the intersection of:
(A) linear constraints, and
(B) set of positive semi-definite matrices.

24.3.0.13 Lemma

Lemma 24.3.1. Let $\mathcal{U}$ be the set of $n \times n$ positive semidefinite matrices. The set $\mathcal{U}$ is convex.

Proof: Consider $A, B \in \mathcal{U}$, and observe that for any $t \in [0, 1]$, and vector $v \in \mathbb{R}^n$, we have:

\[ v^T \left( tA + (1 - t)B \right) v = v^T \left( tAv + (1 - t)Bv \right) = tv^T Av + (1 - t)v^T Bv \geq 0 + 0 \geq 0, \]

since $A$ and $B$ are positive semidefinite.

24.3.0.14 More on positive semidefinite matrices

(A) PSD matrices corresponds to ellipsoids.
(B) $x^TAx = 1$: the set of vectors solve this equation is an ellipsoid.
(C) Eigenvalues of a PSD are all non-negative real numbers.
(D) Given matrix: can in polynomial time decide if it is PSD.
(E) ... by computing the eigenvalues of the matrix.
(F) $\Rightarrow$ SDP: optimize a linear function over a convex domain.
(G) SDP can be solved using interior point method, or the ellipsoid method.
(I) Membership oracle: ability to decide in polynomial time, given a solution, whether its feasible or not.
24.4 Bibliographical Notes

24.4.0.15 Bibliographical Notes

(A) Approx. algorithm presented by Goemans and Williamson [Goemans and Williamson 1995].
(B) Håstad [Håstad 2001] showed that MAX CUT can not be approximated within a factor of $16/17 \approx 0.941176$.
(C) Khot et al [Khot et al. 2004] showed a hardness result that matches the constant of Goemans and Williamson (i.e., one can not approximate it better than $\alpha$, unless $P = NP$).

24.4.0.16 Bibliographical Notes

(A) Relies on two conjectures: “Unique Games Conjecture” and “Majority is Stablest”.
(B) “Majority is Stablest” conjecture was proved by Mossel et al [Mossel et al. 2005].
(C) Not clear if the “Unique Games Conjecture” is true, see the discussion in Khot et al. [2004].
(D) Goemans and Williamson work spurred wide research on using SDP for approximation algorithms.
Bibliography


