Max-SAT

Given m CNF clauses $C_1, C_2, \ldots, C_m$

on n boolean variables $x_1, \ldots, x_n$

literal: x_i or its negation $\neg x_i$

each clause is conjunction (or) of literals: $(x_1 \vee \neg x_5 \vee x_7)$

Goal: find an assignment for the variables to maximize number of satisfied clauses. In weighted version, clause C_j has positive weight w_j and the goal is to maximize weight of satisfied clauses

SAT, in particular 3-SAT *the* NP-Complete problem

Similarly Max-SAT and Max-3SAT very imporant in approximation, especially in proving inapproximability

A simple randomized algorithm

Set each variable x_i independently to True with probability ½

Note that x_i is set to False with probability ½

Let k_j be # of literals in C_j

$\Pr[C_j \text{ is not satisfied}] \leq 2^{-k_j}$

since this happens only if each positive literal in C_j is set to False and each negative literal in C_j is set to True

A simple randomized algorithm

$\Pr[C_j \text{ is not satisfied}] \leq 2^{-k_j}$

$\Pr[C_j \text{ is satisfied}] \geq 1 - 2^{-k_j}$

X_j : random variable equal to 1 if C_j is satisfied, 0 otherwise.

$X = \sum_{j=1}^{m} X_j$

X : random variable which gives the weight of clauses satisfied

$E[X] = \sum_j E[X_i] = \sum_j \Pr[C_j \text{ is satisfied}] \geq \sum_j (1-2^{-k_j}) \geq m/2$

Since $OPT \leq m$, this gives a randomized ½ approximation

Can be derandomized (see Vazirani's book)

A simple randomized algorithm

Notice that if $k_j \geq k$ for all j then approximation is $(1\text{-}2^{-k})$

The probability of a clause being satisfied improves with size of clause.

Also note that the random assignment procedure works for other constraint satisfaction problems

Constraint satisfaction: given variables $x_1, \ldots, x_n$ from some domain (boolean or other domains)

Set of constraints over variables: $C_1, \ldots, C_m$

Goal: set variables to maximize # of satisfied constraints

Constraint satisfaction

Max-Cut can be formulated as a constraint satisfaction problem

For each vertex i we have a boolean variable x_i
For each edge $e = ij$ we have a constraint $x_i \oplus x_j$

Now show that finding an assignment that maximizes the number of constraints is the same as solving Max-Cut.

What does the random assignment correspond to?

Max-3SAT

Random assignment gives a 7/8 approximation for

Max-3SAT , that is instances of Max-SAT in which all cluases have 3 literals (or more)

Hastad showed that unless P=NP no $7/8+\varepsilon$ approximation!

LP Based algorithm for Max-SAT

y_i LP variable corresponding to x_i. y_i is 1 if x_i is set to True, 0 otherwise

z_j : variable for clause C_j, 1 if C_j satisfied, 0 otherwise

For cluase C_j,

$P_j = \{ i \mid \text{variable } x_i \text{ occurs as positive literal in } C_j \}$

$N_j = \{ i \mid \text{variable } x_i \text{ occurs as a negative literal in } C_j \}$

LP Based algorithm for Max-SAT

LP relaxation:

max $\sum_{j=1}^{m} z_j$

s.t

$z_j \leq \sum_{i \in P_j} y_i + \sum_{i \in N_j} (1-y_i)$ for each clause C_j

$y_i, z_j \in [0, 1]$ for all i, j

Check that above is a valid relaxation (if $y_i, z_j \in \{0,1\}$ then IP is exact)

Rounding LP

Let (y^*, z^*) be an optimum solution to LP

We round y^* to an assignment by setting variable i to True with probability y^*_i

Wlog assume that all literals in C_j are positive (since we are only focussing on C_j)

$\Pr[C_j \text{ is satisfied}]$

$= 1 - \prod_{i \in P_j} (1-y^*_i)$

$\geq 1 - (\sum_{i \in P_j} (1-y^*_i)/k_j)^{k_j}$ (using arithmetic-geometric ineq)

$= 1 - (1 - \sum_{i \in P_j} y^*_i/k_j)^{k_j}$

$\geq 1 - (1 - \sum_{i \in P_j} z^*_j/k_j)^{k_j}$ since in LP, $z^*_j \leq \sum_{i \in P_j} y^*_i$

Rounding LP

$\Pr[C_j \text{ is satisfied}] \geq 1 - (1 - \sum_{i \in P_i} z^*_j/k_j)^{k_j}$

Can show using elementary calculus that for $z \in [0,1]$

$1 - (1-z/k)^k \geq (1 - (1-1/k)^k)\, z$

Therefore

$\Pr[C_j \text{ is satisfied}] \geq (1 - (1-1/k_j)^{k_j}) z^*_j \geq (1-1/e) z^*_j$

Therefore the expected number of clauses satisfied

$\geq \sum_j (1-1/e) z^*_j \geq (1-1/e)\, OPT_{LP} \geq (1-1/e)\, OPT$

Note that the probability of a clause being satisfied is good when k_j is small and decreases to $1-1/e$ as $k_j \to \infty$

A ¾ approximation for Max-SAT

The simple random assignment algorithm satisfied large clauses better. The LP based randomized algorithm satisfied small cluases better.

Algorithm: Pick the better of the two solutions.

How do we analyze?

Modified Algorithm: Pick the algorithm to use randomly with probability ½

Show expected # of clauses satisfied $\geq$ ¾ OPT_{LP}

Note that we are still using OPT_{LP} as an upper bound even though the first algorithm had no LP!

A ¾ approximation for Max-SAT

The simple random assignment algorithm satisfied large clauses better. The LP based randomized algorithm satisfied small cluases better.

Algorithm: Pick the better of the two solutions.

How do we analyze?

Modified Algorithm: Pick the algorithm to use randomly with probability ½

Show expected # of clauses satisfied $\geq$ ¾ OPT_{LP}

Note that we still use OPT_{LP} as an upper bound on OPT even though the first algorithm had no LP

A ¾ approximation for Max-SAT

Analysis of algorithm:

We think of the algorithm as a single randomized algorithm

$\Pr[C_j \text{ is satisfied}] = \frac{1}{2} \Pr[C_j \text{ is satisfied in Alg 1}] + \frac{1}{2} \Pr[C_j \text{ is satisfied in Alg 2}]$

From previous analysis:

$\Pr[C_j \text{ is satisfied in Alg 1}] = 1 - 2^{-k_j} \geq (1-2^{-k_j})\, z^*_j$ (since $z^*_j \leq 1$)

$\Pr[C_j \text{ is satisfied in Alg 2}] = (1 - (1 - 1/k_j)^{k_j})\, z^*_j$

A ¾ approximation for Max-SAT

Therefore

$\Pr[C_j \text{ is satisfied}] \geq z^*_j \, (2 - 2^{-k_j} - (1-1/k_j)^{k_j})/2$

can check that for all $k_j \geq 1$, $(2 - 2^{-k_j} - (1-1/k_j)^{k_j})/2 \geq ¾$

Therefore $\Pr[C_j \text{ is satisfied }] \geq ¾$

By linearity of expectation, the expected # of clauses satisfied is $\geq ¾ \, OPT_{LP} \geq ¾ \, OPT$

Tight example

Two variables x_1, x_2

4 clauses with all possible combinations
$(x_1 \vee x_2)$, $(x_2 \vee \neg x_2)$, $(\neg x_1 \vee x_2)$, $(\neg x_1 \vee \neg x_2)$

Clear that OPT = 3

Claim: $OPT_{LP} = 4$

set $y_i = ½$ for $i = 1,2$

$z_j = 1$ for $j = 1,2,3,4$

feasible for LP

integrality gap = 3/4

Metric Methods

A metric space is given by a set V and a distance function $d : V \times V \rightarrow \mathcal{R}^+$

The distance function d satisfies
triangle inequality: $d(u,v) + d(v,w) \geq d(u,w)$ for all $u,v,w \in V$

symmetry: $d(u,v) = d(v,u)$
reflexivity: $d(u,v) = 0 \Rightarrow u = v$

If we drop the reflexivity then we get a semi-metric

If we drop symmetry we simply get a distance function

Metrics/Distances in Algorithms

Metrics/distances arise for three main reasons

- As input to a problem
- As desired output for a problem
- As part of solving a problem

Metrics/Distances as input

Directly: geometric problems with points in real space

Indirectly: a graph with edge weights defines a distance function (if G is directed) and a metric if G is undirected

The solution to many problems doesn't change if instead of G, we give the metric completion of G

Examples: Metric-TSP, Steiner tree, Steiner forest, k-Center, k-median, facility location etc

Typically G is a sparse graph and metric-completion is dense graph so it is preferable to work with G directly

Metric/distances as output

Example: Phylogenetic trees, evolutionary trees etc

Given a set of species that evolve from a common parent species, an evolutionary tree describes the branching off process. We only have today's species and some measure of distances between them. These distances are not perfect and are also noisy.

Problem: given a distance matrix D between a set of species, find a tree T with the current species at the leaves and assign lengths on edges of T so that the distances between the leaves is as close to D as possible

Note: here the output is a metric (induced by a tree).

Metrics/distances as relaxations

An important application of metrics is in solving cut problems. In a cut problem we are given some graph and the goal is to remove some edges of minimum cost to satisfy some constraints. Typically the constraints ask for separation of some sets of vertices. Metrics arise naturally in trying to solve these problems as follows.

Given a susbet S of edges in G, we can define a distance function d_S on the vertices of V as

$d_S(u, v) = 1$ if u and v are separated by removing S

$d_S(u, v) = 0$ otherwise

Metrics/distances as relaxations

Given a susbet S of edges in G, we can define a distance function d_S on the vertices of V as

$d_S(u, v) = 1$ if u and v are separated by removing S

$d_S(u, v) = 0$ otherwise

Note that if G is undirected then d_S is symmetric and hence is a semi-metric while d_S need not be symmetric in directed graphs

A cut problem consists of choosing a min-cost set of edges to remove so that some separation constraints on vertices are satisfied. We can express that as an integer program as follows

Metrics and cuts

we have a binary variable d_e for each edge e and it indicates whether e is chosen in the cut or not

c_e is cost of e

min $\sum_e c_e d_e$

s.t

$d(s, t) = 1$ for all pairs (s, t) that need to be separated

$d_e \in \{0, 1\}$

In the above $d(s,t)$ is the shortest path distance between s and t with edge lengths given by d_e

We can express shortest path distances using linear constraints as shown below

Metrics and cuts

Given d_e we wish to obtain $d(u,v)$ for each pair of vertices u,v where $d(u,v)$ is the distances between u and v in the graph with edge lengths given by d_e

We simply write the simple triangle inequalities

$d(u,v) + d_e \leq d(u, w)$ if $e = (v,w)$

A less cumbersome way to do this is to assume that the given graph is a complete graph. This is wlog since any edge which is not in the original graph can be zero cost

Metrics and cuts

If we assume above then the edge and distance variables are essentially same so we simply have variables $d(u,v)$ for each pair of vertices (u,v) then we can write cut problems as

$\min \sum_{u,v} c(u,v)\, d(u,v)$

s.t

$d(s,t) = 1$ for each pair s,t that needs to be separated

$d(u,v) + d(v,w) \geq d(u,w)$ for all u,v,w

$d(u,v) \in \{0,1\}$

Metrics and cuts

The formulation applies to directed graphs since we treat d(u,v) and d(v,u) as separate variables. Thus there are a total of n(n-1) variables

For undirected graphs there are two possibilities.

One is to use ordered pairs and add additional symmetric inequalities

d(u,v) = d(v,u)

and in the cost summation have only one of the two terms.

Or one can also formulate using variables d(uv) where uv is an unordered pair: we will have n(n-1)/2 variables

Maxflow-mincut using metrics

We will prove the maxflow min-cut theorem using metric methods and LP duality

Given a directed graph $G=(V,A)$ and two vertices s, t we wish to find the minimum cost cut separating s and t

Each arc in A has a cost c_a and the cost of of a cut (set of arcs) is the sum of the arc weights

As before we can assume that the graph is a complete graph by using zero cost arcs. This won't change the problem

Min-cut as IP/LP

We have a variable $d(u,v)$ for each ordered pair of vertices (u,v) and we write the LP now as

$\min \sum_{u,v} c(u,v)\, d(u,v)$

s.t

$d(s,t) \geq 1$

$d(u,v) + d(v,w) \geq d(u, w)$ for all u,v,w

$d(u,v) \in \{0,1\}$ for all (u,v)

We obtain an LP by relaxing the integer constraints to $d(u,v) \in [0, 1]$ for all (u,v)

Rounding LP to obtain a cut

Given a feasible solution d^* to the above LP we obtain an integer solution as follows:

Pick a number θ uniformly at random from $[0,1)$
Let $X = \{v \mid d^*(s,v) \leq \theta\}$

Output the cut $\delta_G(X) = \{ (u,v) \mid u \in X, v \notin X \}$

Claim: Output is a feasible solution (separates s and t)
Proof: t is not in X since $d^*(s, t) \geq 1$ and $\theta < 1$

All edges out of X are removed so s, t are separated

Rounding LP to obtain a cut

Claim: Expected cost of cut = OPT_{LP}

Proof: Consider arc (u,v)

$\Pr[(u,v) \text{ is cut}] = \Pr[u \text{ in } X, v \notin X]$
$= \Pr[d^*(s,u) \leq \theta \text{ and } d^*(s,v) > \theta]$

$= \Pr[\theta \text{ lies in interval } [d^*(s,u], d^*(s,v))] =$
$d^*(s,v) - d^*(s,u) \leq d^*(u,v)$

Therefore expected cost of cut
$\leq \sum_{u,v} c(u,v)\, d^*(u,v) = OPT_{LP} \leq OPT$

Therefore expected cost = OPT_{LP} = OPT where OPT is optimum (integer) cut

Maxflow-mincut

The above analysis shows that the LP has an optimum solution equal to its integer optimum

Prove the dual of the LP for s-t cut we used is the LP for s-t maximum flow

By duality, we have the maximum flow is equal to minimum cut

Multiway-cut and LP Rounding

Recall the multiway-cut problem

Given undirected graph $G=(V,E)$ with edge costs $c: E \rightarrow \mathcal{R}^+$ and terminals $T = \{t_1, t_2, \ldots, t_k\} \subset V$

Find a min-cost set of edges $E' \subseteq E$ whose removal results in separating the terminals

Previously we saw a greedy $2(1-1/k)$ approximation using two slightly different greedy algorithms

Here we use an LP relaxation to obtain the same result

LP Relaxation

One formulation is the following:
variable l_e for $e \in E$ to indicate if e is cut or not

Let $P_{ij} = \{ p \mid p \text{ is a path from } t_i \text{ to } t_j \text{ in } G\}$

$\min \sum_e c_e l_e$

s.t

$\sum_{e \in p} l_e \geq 1 \qquad p \in P_{ij}, \ 1 \leq i < j \leq k$

$l_e \geq 0$

LP Relaxation

$\min \sum_e c_e l_e$

s.t

$\sum_{e \in p} l_e \geq 1 \qquad p \in P_{ij}, \ 1 \leq i < j \leq k$

$l_e \geq 0$

Note: formulation has exponential # of constraints but LP has a polynomial time separation oracle and hance can be solved in polynomial time.

A polynomial sized formulation

One could also write a different formulation which is essentially equivalent but has polynomial size

extra vairable $d(uv)$ for each unordered pair of vertices uv.

$d(uv)$ is to model the shortest path distance between u and v in the metric induced by l

$\min \sum_e c_e l_e$

s.t

$d(uv) + d(vw) \geq d(uw)$ $\quad u,v,w \in V$

$d(uv) \leq l_e$ $\quad$ for each edge $e=uv$

$d(t_i t_j) \geq 1$ $\quad 1 \leq i < j \leq k$

$l, d \geq 0$

A polynomial sized formulation

In fact we do not need to maintain the variables l anymore since in any optimum solution $d(uv) = l_{uv}$

Thus the formulation can be written as

$$\min \sum_{uv \in E} c(uv)\, d(uv)$$

s.t

$$d(uv) + d(vw) \geq d(uw) \qquad u,v,w \in V$$

$$d(t_i t_j) \geq 1 \qquad 1 \leq i < j \leq k$$

$$d \geq 0$$

# of variables is $n(n-1)/2$ and # of constraints is $\Theta(n^3)$

Rounding the LP

Note that both LP's essentially assign lengths/distances to each e

Let us work with the first LP

we let $l(uv)$ denote the shortest path distance between u and v with edge lengths given by l

Note that the LP ensures that $l(t_i t_j) \geq 1$ for each i,j

For a vertex v and a real value r let $B(v,r) = \{u \mid l(vu) < r\}$ denote the *(open) ball of raidius r around v*

Rounding the LP

Pick θ uniformly at random from $[0, ½)$

For each t_i, remove all edges $\delta(B(t_i, \theta))$
that is remove $E' = \cup_{i=1}^{k} \delta(B(t_i, \theta))$

Claim: E' is a feasible solution

We observe that the only vertices reachable from t_i after removing E' are in $B(t_i, \theta)$

And $B(t_i, \theta) \cap B(t_j, \theta) = \emptyset$ for otherwise we would have $l(t_i t_j) < 1$

Note that $B(t_i, ½) \cap B(t_j, ½) = \emptyset$ (Why?)

Expected cost of E’

We now estimate Expect[c(E’)] the expected cost of E’
It is enough to estimate the probability that $e \in E'$

We prove the following lemma
Lemma: $\Pr[e \in E'] \leq 2\, I_e$

Assuming the lemma,
$\text{Expect}[c(E')] \leq \sum_e 2c_e\, I_e \leq 2\, \text{OPT}_{LP} \leq 2\text{OPT}$

Proof of Lemma

Focus on $e = uv$

Four cases:

Case 1: $u, v \in B(t_i, ½)$ for some i

Wlog, $l(t_iu) \leq l(t_iv)$, that is u is closer to t_i than v

Then

$$\Pr[e \text{ is cut }] = \Pr[l(t_iu) \leq \theta \text{ and } l(t_iv) > \theta)]$$

$$= 2\,(l(t_iv) - l(t_i\,u)) \leq 2\,l_e$$

Proof of Lemma

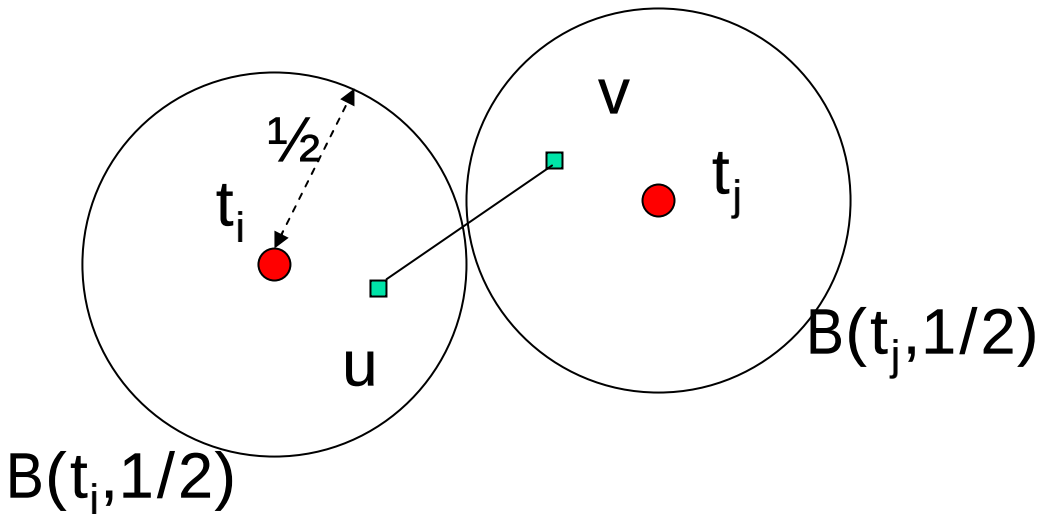
Case 2: $u \in B(t_i, ½), v \in B(t_j, ½)$ for some i, j

Note that in this case e can be cut either in $B(t_i,1/2)$ or $B(t_j,1/2)$ (see pic next slide)

Then

$\Pr[e \text{ is cut }]$
$\leq \Pr[e \text{ is cut in } B(t_i,1/2)] + \Pr[e \text{ is cut in } B(t_j, ½)]$
$\leq \Pr[l(t_iu) < \theta \leq ½] + \Pr[l(t_j,v) \leq \theta \leq ½]$
$\leq 2(1/2 - l(t_iu)) + 2(1/2 - l(t_jv))$
$\leq 2(1 - l(t_iu) - l(t_jv)) \leq 2l_e$ (Why?)

u, v in different balls



Other cases

Case 3: $u \in B(t_i, ½)$ and v is not in any ball

In this case $Pr[e \text{ is cut}] = 2(1/2 - l(t_i u)) \leq 2l_e$ (Why)

Case 4: u, v are both not in any ball

In this case uv will not be cut

Derandomization and improvement

We can derandomize the algorithm. Although θ is picked from $[0,1/2)$ there are only a polynomial number of values at which $B(t_i, \theta)$ changes when increasing from 0 to ½ corresponding to a BFS search from t_i. Hence we can try "all possible values" of θ and pick the value at which we obtain the cheapest cut.

Note that we don't have to cut in all balls. See how you can improve analysis to obtain a $2(1-1/k)$ bound

Tight example

Star with k leaves each of which is a terminal

OPT = k-1

OPT_{LP} = k/2 (why?)

So integrality gap = 2(1-1/k)