Computability and Complexity
COSC 4200

Approximating MAX3SAT

Decision problem:

3SAT

Input: 3CNF formula ϕ

Goal: Decide if ϕ has a satisfying assignment

Decision problem:

3SAT

Input: 3CNF formula ϕ

Goal: Decide if ϕ has a satisfying assignment

Search problem:

3SAT Search

Input: 3CNF formula ϕ

Goal: Find a satisfying assignment for ϕ, if one exists.

Decision problem:

3SAT

Input: 3CNF formula ϕ

Goal: Decide if ϕ has a satisfying assignment

Search problem:

3SAT Search

Input: 3CNF formula ϕ

Goal: Find a satisfying assignment for ϕ, if one exists.

Optimization problem:

MAX3SAT

Input: 3CNF formula ϕ

Goal: Find an assignment that satisfies as many of ϕ's clauses as possible.

We've seen that 3SAT decision is solvable in polynomial time if and only if 3SAT search is solvable in polynomial time.

We've seen that 3SAT decision is solvable in polynomial time if and only if 3SAT search is solvable in polynomial time.

In MAX3SAT we might be given a formula that is unsatisfiable. Our job is to find an assignment that satisfies as many clauses as possible.

Proposition

If $\mathrm{P} \neq \mathrm{NP}$, then there is no polynomial-time algorithm for MAX3SAT.

Proposition

If $\mathrm{P} \neq \mathrm{NP}$, then there is no polynomial-time algorithm for MAX3SAT.

Proof.

Suppose A is a polynomial-time algorithm for MAX3SAT. We show that 3SAT $\in \mathrm{P}$ via the following algorithm:

On input ϕ, run A to get an assignment τ. If τ satisfies ϕ, accept ϕ; otherwise reject ϕ. $\square$

Proposition

If $\mathrm{P} \neq \mathrm{NP}$, then there is no polynomial-time algorithm for MAX3SAT.

Proof.

Suppose A is a polynomial-time algorithm for MAX3SAT. We show that 3SAT $\in \mathrm{P}$ via the following algorithm:

On input ϕ, run A to get an assignment τ. If τ satisfies ϕ, accept ϕ; otherwise reject ϕ. $\square$

It is therefore unlikely there is an efficient algorithm that solves MAX3SAT. However, this is an optimization problem, and we can ask if there is an approximation algorithm that may not achieve the optimum solution but always comes close.

Approximation Algorithms

Let X be a minimization problem.
An algorithm $\mathcal{A}$ is an $\alpha(n)$-approximation algorithm if for all n, for all instances of size n, $\mathcal{A}$ returns a solution with value at most $\alpha(n)$ times the value of the optimal solution.

For example:

- a $\log n$-approximation algorithm
- a 2-approximation algorithm

Approximation Algorithms

Let X be a minimization problem.
An algorithm $\mathcal{A}$ is an $\alpha(n)$-approximation algorithm if for all n, for all instances of size n, $\mathcal{A}$ returns a solution with value at most $\alpha(n)$ times the value of the optimal solution.

For example:

- a $\log n$-approximation algorithm
- a 2-approximation algorithm

Let X be a maximization problem.
An algorithm $\mathcal{A}$ is an $\alpha(n)$-approximation algorithm if for all n, for all instances of size n, $\mathcal{A}$ returns a solution with value at least $\alpha(n)$ times the value of the optimal solution.

For example:

- a $\frac{7}{8}$-approximation algorithm
- a $\frac{\log n}{n}$-approximation algorithm

An algorithm for MAX3SAT is an α-approximation algorithm if it satisfies at least an α fraction of the optimal number of clauses.

Theorem

There is a polynomial-time $\frac{7}{8}$-approximation algorithm for MAX3SAT.

An algorithm for MAX3SAT is an α-approximation algorithm if it satisfies at least an α fraction of the optimal number of clauses.

Theorem

There is a polynomial-time $\frac{7}{8}$-approximation algorithm for MAX3SAT.

Proof. Let $\phi = C_1 \wedge \cdots \wedge C_m$ be a 3ECNF formula with m clauses and n variables (each clause has exactly 3 distinct literals – the case where some clauses have fewer than 3 literals is easier).

An algorithm for MAX3SAT is an α-approximation algorithm if it satisfies at least an α fraction of the optimal number of clauses.

Theorem

There is a polynomial-time $\frac{7}{8}$-approximation algorithm for MAX3SAT.

Proof. Let $\phi = C_1 \wedge \cdots \wedge C_m$ be a 3ECNF formula with m clauses and n variables (each clause has exactly 3 distinct literals – the case where some clauses have fewer than 3 literals is easier).

It is instructive to first consider choosing an assignment τ uniformly at random.

Since each clause involves 3 distinct variables, we have

$$\mathrm{Prob}[\tau \text{ satisfies } C_i] = \frac{7}{8}$$

for each i as there is only one of the 8 possible settings to the variables that does not satisfy C_i.

Let Y_i be the indicator random variable

$$Y_i = \begin{cases} 1 & \text{if } \tau \text{ satisfies } C_i \\ 0 & \text{if } \tau \text{ does not satisfies } C_i. \end{cases}$$

Let Y_i be the indicator random variable

$$Y_i = \begin{cases} 1 & \text{if } \tau \text{ satisfies } C_i \\ 0 & \text{if } \tau \text{ does not satisfies } C_i. \end{cases}$$

Then

$$E[Y_i] = \frac{7}{8}.$$

Let Y be the random variable that counts the number of clauses that τ satisfies.

Let Y_i be the indicator random variable

$$Y_i = \begin{cases} 1 & \text{if } \tau \text{ satisfies } C_i \\ 0 & \text{if } \tau \text{ does not satisfies } C_i. \end{cases}$$

Then

$$E[Y_i] = \tfrac{7}{8}.$$

Let Y be the random variable that counts the number of clauses that τ satisfies. Then $Y = \sum_{i=1}^{m} Y_i$ and

Let Y_i be the indicator random variable

$$Y_i = \begin{cases} 1 & \text{if } \tau \text{ satisfies } C_i \\ 0 & \text{if } \tau \text{ does not satisfies } C_i. \end{cases}$$

Then

$$E[Y_i] = \tfrac{7}{8}.$$

Let Y be the random variable that counts the number of clauses that τ satisfies. Then $Y = \sum_{i=1}^{m} Y_i$ and

$$E[Y] \;=\; E\left[\sum_{i=1}^{m} Y_i\right]$$

Let Y_i be the indicator random variable

$$Y_i = \begin{cases} 1 & \text{if } \tau \text{ satisfies } C_i \\ 0 & \text{if } \tau \text{ does not satisfies } C_i. \end{cases}$$

Then

$$E[Y_i] = \tfrac{7}{8}.$$

Let Y be the random variable that counts the number of clauses that τ satisfies. Then $Y = \sum_{i=1}^{m} Y_i$ and

$$\begin{aligned} E[Y] &= E\left[\sum_{i=1}^{m} Y_i\right] \\ &= \sum_{i=1}^{m} E[Y_i] \end{aligned}$$

Let Y_i be the indicator random variable

$$Y_i = \begin{cases} 1 & \text{if } \tau \text{ satisfies } C_i \\ 0 & \text{if } \tau \text{ does not satisfies } C_i. \end{cases}$$

Then

$$E[Y_i] = \tfrac{7}{8}.$$

Let Y be the random variable that counts the number of clauses that τ satisfies. Then $Y = \sum_{i=1}^{m} Y_i$ and

$$\begin{aligned} E[Y] &= E\left[\sum_{i=1}^{m} Y_i\right] \\ &= \sum_{i=1}^{m} E[Y_i] \\ &= \sum_{i=1}^{m} \tfrac{7}{8} \end{aligned}$$

Let Y_i be the indicator random variable

$$Y_i = \begin{cases} 1 & \text{if } \tau \text{ satisfies } C_i \\ 0 & \text{if } \tau \text{ does not satisfies } C_i. \end{cases}$$

Then

$$E[Y_i] = \tfrac{7}{8}.$$

Let Y be the random variable that counts the number of clauses that τ satisfies. Then $Y = \sum_{i=1}^{m} Y_i$ and

$$\begin{aligned} E[Y] &= E\left[\sum_{i=1}^{m} Y_i\right] \\ &= \sum_{i=1}^{m} E[Y_i] \\ &= \sum_{i=1}^{m} \tfrac{7}{8} \\ &= \tfrac{7}{8} m. \end{aligned}$$

On average, a randomly chosen τ will satisfy $\frac{7}{8}m$ clauses.
Therefore there must be some τ that satisfies at least $\frac{7}{8}m$ clauses.
This is called the *probabilistic method*.

On average, a randomly chosen τ will satisfy $\frac{7}{8}m$ clauses.
Therefore there must be some τ that satisfies at least $\frac{7}{8}m$ clauses.
This is called the *probabilistic method*.

Our algorithm A will search for such a τ using the *method of conditional expectations*.

Let OPT be the maximum number of clauses that can be satisfied.
Since $\mathrm{OPT} \leq m$, we will then have performance ratio at least

$$\frac{\frac{7}{8}m}{\mathrm{OPT}} \geq \frac{\frac{7}{8}m}{m} = \frac{7}{8}.$$

Given a partial assignment $\tau_j = (t_1, \ldots, t_j)$, where $0 \leq j < n$, consider extending τ_j into a full assignment $\tau = (t_1, \ldots, t_n)$ by choosing $t_{j+1}, \ldots, t_n$ for the remaining variables at random.

Given a partial assignment $\tau_j = (t_1, \ldots, t_j)$, where $0 \le j < n$, consider extending τ_j into a full assignment $\tau = (t_1, \ldots, t_n)$ by choosing $t_{j+1}, \ldots, t_n$ for the remaining variables at random.

Then for each i, we can efficiently compute

$$E[Y_i \mid \tau_j] = \mathrm{Prob}[Y_i = 1 \mid \tau_j]$$

and

$$E[Y \mid \tau_j] = \sum_{i=1}^{m} E[Y_i \mid \tau_j].$$

How to compute $\mathrm{Prob}[Y_i = 1 \mid \tau_j]$

τ_j fixes either $0, 1, 2$, or 3 literals in clause C_i.

0. No literals fixed: Then the remainder of the assignment will satifisfy the clause with probability $\frac{7}{8}$.

How to compute $\mathrm{Prob}[Y_i = 1 \mid \tau_j]$

τ_j fixes either $0, 1, 2,$ or 3 literals in clause C_i.

0. No literals fixed: Then the remainder of the assignment will satifisfy the clause with probability $\frac{7}{8}$.
1. One literal fixed:
   - If the fixed literal is true, the clause is satisfied with probability 1.
   - Otherwise, the clause will be satisfied with probability $\frac{3}{4}$.

How to compute $\mathrm{Prob}[Y_i = 1 \mid \tau_j]$

τ_j fixes either $0, 1, 2$, or 3 literals in clause C_i.

0. No literals fixed: Then the remainder of the assignment will satifisfy the clause with probability $\frac{7}{8}$.
1. One literal fixed:
   - If the fixed literal is true, the clause is satisfied with probability 1.
   - Otherwise, the clause will be satisfied with probability $\frac{3}{4}$.
2. Two literals fixed:
   - If at least one fixed literal is true, the clause is satisfied with probability 1.
   - Otherwise, the clause will be satisfied with probability $\frac{1}{2}$.

How to compute $\mathrm{Prob}[Y_i = 1 \mid \tau_j]$

τ_j fixes either $0, 1, 2$, or 3 literals in clause C_i.

0. No literals fixed: Then the remainder of the assignment will satifisfy the clause with probability $\frac{7}{8}$.
1. One literal fixed:
   - If the fixed literal is true, the clause is satisfied with probability 1.
   - Otherwise, the clause will be satisfied with probability $\frac{3}{4}$.
2. Two literals fixed:
   - If at least one fixed literal is true, the clause is satisfied with probability 1.
   - Otherwise, the clause will be satisfied with probability $\frac{1}{2}$.
3. Three literals fixed:
   - If any of the literals are true, the clause is satisfied with probability 1.
   - Otherwise, the clause is satisfied with probability 0.

For any i we have,

$$\begin{aligned}\mathrm{Prob}[Y_i = 1 \mid \tau_j] &= \mathrm{Prob}[Y_i = 1, t_{j+1} = T \mid \tau_j] \\ &\quad + \mathrm{Prob}[Y_i = 1, t_{j+1} = F \mid \tau_j]\end{aligned}$$

For any i we have,

$$\begin{aligned}\mathrm{Prob}[Y_i = 1 \mid \tau_j] &= \mathrm{Prob}[Y_i = 1, t_{j+1} = T \mid \tau_j] \\ &\quad + \mathrm{Prob}[Y_i = 1, t_{j+1} = F \mid \tau_j] \\ &= \mathrm{Prob}[Y_i = 1 \mid \tau_j,\ t_{j+1} = T] \cdot \mathrm{Prob}[t_{j+1} = T] \\ &\quad + \mathrm{Prob}[Y_i = 1 \mid \tau_j,\ t_{j+1} = F] \cdot \mathrm{Prob}[t_{j+1} = F]\end{aligned}$$

For any i we have,

$$\begin{aligned}
\mathrm{Prob}[Y_i = 1 \mid \tau_j] &= \mathrm{Prob}[Y_i = 1, t_{j+1} = T \mid \tau_j] \\
&\quad + \mathrm{Prob}[Y_i = 1, t_{j+1} = F \mid \tau_j] \\
&= \mathrm{Prob}[Y_i = 1 \mid \tau_j,\ t_{j+1} = T] \cdot \mathrm{Prob}[t_{j+1} = T] \\
&\quad + \mathrm{Prob}[Y_i = 1 \mid \tau_j,\ t_{j+1} = F] \cdot \mathrm{Prob}[t_{j+1} = F] \\
&= \tfrac{1}{2} \cdot \mathrm{Prob}[Y_i = 1 \mid \tau_j,\ t_{j+1} = T] \\
&\quad + \tfrac{1}{2} \cdot \mathrm{Prob}[Y_i = 1 \mid \tau_j,\ t_{j+1} = F],
\end{aligned}$$

For any i we have,

$$\begin{aligned}
\mathrm{Prob}[Y_i = 1 \mid \tau_j] &= \mathrm{Prob}[Y_i = 1, t_{j+1} = T \mid \tau_j] \\
&\quad + \mathrm{Prob}[Y_i = 1, t_{j+1} = F \mid \tau_j] \\
&= \mathrm{Prob}[Y_i = 1 \mid \tau_j,\ t_{j+1} = T] \cdot \mathrm{Prob}[t_{j+1} = T] \\
&\quad + \mathrm{Prob}[Y_i = 1 \mid \tau_j,\ t_{j+1} = F] \cdot \mathrm{Prob}[t_{j+1} = F] \\
&= \tfrac{1}{2} \cdot \mathrm{Prob}[Y_i = 1 \mid \tau_j,\ t_{j+1} = T] \\
&\quad + \tfrac{1}{2} \cdot \mathrm{Prob}[Y_i = 1 \mid \tau_j,\ t_{j+1} = F],
\end{aligned}$$

so it follows that

$$\begin{aligned}
E[Y \mid \tau_j] &= \tfrac{1}{2} E[Y \mid \tau_j, t_{j+1} = T] \\
&\quad + \tfrac{1}{2} E[Y \mid \tau_j, t_{j+1} = F].
\end{aligned}$$

For any i we have,

$$\begin{aligned}
\mathrm{Prob}[Y_i = 1 \mid \tau_j] &= \mathrm{Prob}[Y_i = 1, t_{j+1} = T \mid \tau_j] \\
&\quad + \mathrm{Prob}[Y_i = 1, t_{j+1} = F \mid \tau_j] \\
&= \mathrm{Prob}[Y_i = 1 \mid \tau_j,\ t_{j+1} = T] \cdot \mathrm{Prob}[t_{j+1} = T] \\
&\quad + \mathrm{Prob}[Y_i = 1 \mid \tau_j,\ t_{j+1} = F] \cdot \mathrm{Prob}[t_{j+1} = F] \\
&= \tfrac{1}{2} \cdot \mathrm{Prob}[Y_i = 1 \mid \tau_j,\ t_{j+1} = T] \\
&\quad + \tfrac{1}{2} \cdot \mathrm{Prob}[Y_i = 1 \mid \tau_j,\ t_{j+1} = F],
\end{aligned}$$

so it follows that

$$\begin{aligned}
E[Y \mid \tau_j] &= \tfrac{1}{2} E[Y \mid \tau_j, t_{j+1} = T] \\
&\quad + \tfrac{1}{2} E[Y \mid \tau_j, t_{j+1} = F].
\end{aligned}$$

Therefore we always have

$$\max(E[Y \mid \tau_j, t_{j+1} = T], E[Y \mid \tau_j, t_{j+1} = F]) \geq E[Y \mid \tau_j].$$

Approximation Algorithm

input ϕ
let n *be the number of variables in* ϕ
let τ_0 *be the empty assignment*
for $j = 1$ *to* n
    compute
        $e_T = E[Y \mid \tau_{j-1}, t_j = T]$,
        $e_F = E[Y \mid \tau_{j-1}, t_j = F]$
    if $e_T > e_F$
        $t_j = T$
    else
        $t_j = F$
    $\tau_j = (t_1, \ldots, t_j)$
output τ_n

Each iteration does not decrease the expected number of satisfied clauses. By induction, τ_n satisfies at least $\frac{7}{8}m$ clauses. $\square$

Theorem (Håstad, 1997)

If $\mathrm{P} \neq \mathrm{NP}$, then for every $\epsilon > 0$, there is no polynomial-time $(\frac{7}{8} + \epsilon)$-appoximation algorithm for MAX3SAT.

Theorem (Håstad, 1997)

If $\mathrm{P} \neq \mathrm{NP}$, then for every $\epsilon > 0$, there is no polynomial-time $(\frac{7}{8} + \epsilon)$-appoximation algorithm for MAX3SAT.

The proof of this theorem is based on the *probabilistically checkable proofs* (PCP) characterization of NP:

$$\mathrm{NP} = \mathrm{PCP}(O(\log n), 3).$$

Every NP language A has a proof system where the proofs of membership may be verified with high probability by reading only 3 bits from the proof that are randomly selected.