CS 437 Lecture Notes

Andrew Li

Fall Quarter 2025

Original lecture notes for **CS 437: Approximation Algorithms**, from Fall Quarter 2025, taught by Professor Konstantin Makarychev.

Table of Contents

§1 September 16, 2025

I joined this class after this lecture.

§1.1 Macros

Below is an example algorithm using the macros in this repository. For simplicity, this algorithm computes the largest element of a fixed size array.

Algorithm 1.1: Algorithm to compute max(list)

```
input list
curmax ← list[0]
for n ∈ list do
    curmax ← max(n, curmax)
return curmax
```

There are also other environments, namely

Lemma 1.1 This is a lemma.

Proposition 1.2 and a proposition.

Definition 1.3 and a definition.

Example 1.4 These boxes are for examples.

Note These boxes are sparingly used, for asides.

Theorem 1.5 And finally, we've got the theorem.

As is standard, we can use the proof environment for proofs.

Proof. Trivial. □

§1.2 Set Cover

Definition 1.6 Set Cover Let V be some universe, with $|V| = n$. Let

$$S_1, \ldots, S_m \subseteq V \tag{1.1}$$

such that $\bigcup_i S_i = V$. Select the smallest $I \subseteq \{1, \ldots, m\}$ such that $\bigcup_{i \in I} S_i = V$.

Example 1.7 Let $V \equiv \{1, 2, 3, 4, 5\}$ and sets be pairs $\{i, j\}$ such that $i \neq j$. Then, an optimal solution is

$$I \equiv \{\{1, 2\}, \{3, 4\}, \{1, 5\}\} \tag{1.2}$$

In this case, $\text{opt}(I) = 3$

Definition 1.8 The approximation factor of an algorithm is α_n if for every I of size n, we have

$$\text{alg}(I) \leq \alpha_n \cdot \text{opt}(I) \tag{1.3}$$

The first theorem of this course is

Theorem 1.9 There exists a polynomial time algorithm with approximation factor $\log n$.

Algorithm 1.2: Polynomial time set cover approximation algorithm

$U_0 \leftarrow V$ `// set of not yet covered elements in` V
$t \leftarrow 0$ `// iteration counter`
for $U_t \neq \emptyset$ **do**
  Select S_i from sets that maximises $|S_i \cap U_t|$
  Include S_i in soln
  $U_t \leftarrow U_t \setminus S_i$
  $t \leftarrow t + 1$
return *soln*

1.2.1 Proof

Let $k = \text{opt}$ be the number of sets in the optimal solution. Let S_{i_1} be the first selected set. Then,

$$|S_{i_1}| \geq \frac{n}{k} \tag{1.4}$$

Then it follows that

$$|U_1| = \left|\underbrace{U_0}_{V} \setminus S_{i_1}\right| = \underbrace{|U_0|}_{n} - |S_{i_1}| \tag{1.5}$$

$$\leq n - \frac{n}{k} = n\left(1 - \frac{1}{k}\right) \tag{1.6}$$

Let $S_{i_{t+1}}$ be the set chosen at iteration t.

Lemma 1.10

$$\bigcup_{i \in I^*} S_i \cap U_t = U_t \tag{1.7}$$

Proof. We can prove that LHS $\subseteq$ RHS and RHS $\subseteq$ LHS. To prove the first,

$$u \in \bigcup_{i \in I^*} S_i \cap U_t \implies u \in \text{at least one } S_i \cap U_t \implies u \in U_t \tag{1.8}$$

Thus, every element in one of the chosen sets' intersection with U_t is in U_t.

$$\begin{aligned} u \in U_t &\implies u \in \text{at least one } S_i && I^* \text{ spans universe; } S_i \text{ must exist} && (1.9) \\ &\implies u \in \text{at least one } S_i \cap U_t && && (1.10) \\ &\implies u \in \bigcup_{i \in I^*} S_i \cap U_t && && (1.11) \end{aligned}$$

And, every element in U_t is in at least one set. □

It follows that, because S_i are not necessarily disjoint sets,

$$\sum_{i \in I^*} |S_i \cap U_t| \geq |U_t| \tag{1.12}$$

Thus, given there are k sets in I^*, by pigeonhole,

$$\exists i \qquad |S_i \cap U_t| \geq \frac{|U_t|}{k} \tag{1.13}$$

Then,

$$\begin{aligned} |U_{t+1}| &= \left|U_t \setminus (S_{i_{t+1}} \cap U_t)\right| && (1.14) \\ &= |U_t| - \left|S_{i_{t+1}} \cap U_t\right| && (1.15) \\ &\leq |U_t| - \frac{|U_t|}{k} = \left(1 - \frac{1}{k}\right)|U_t| && (1.16) \end{aligned}$$

Trivially,

$$|U_t| \leq \left(1 - \frac{1}{k}\right)^t \cdot n \tag{1.17}$$

Proposition 1.11 For $t = k \log n$,

$$\left(1 - \frac{1}{k}\right)^t < \frac{1}{n} \tag{1.18}$$

§2 September 18, 2025

§2.1 Finishing Previous Proof

Recall some universe V, some family of sets $S_1, \dots, S_m \subseteq V$, want to minimise size of family that spans entire V.

Note All solutions are feasible, as the algorithm stops when $U_t = \emptyset$, i.e. when the selected sets span V. If there is no feasible solution, then the algorithm can just terminate when there are no more sets to select.

Recall k is the number in the optimal solution.

Lemma 2.1 For $t^* = k \log n$,

$$\left(1 - \frac{1}{k}\right)^{t^*} < \frac{1}{n} \tag{2.1}$$

If this is true, then

$$|U_{t^*}| < \frac{1}{n} \cdot n < 1 \tag{2.2}$$

which implies $|U_{t^*}| \equiv 0$. That imposes an upper bound on the time steps t needed to cover all elements.

Proof. Use the well-known definition of e

$$\left(1 - \frac{1}{k}\right)^{k \log n} = \left(\left(1 - \frac{1}{k}\right)^{k}\right)^{\log n} \tag{2.3}$$

$$< (1/e)^{\log n} \tag{2.4}$$

$$< 1/n \tag{2.5}$$

□

Note When $x \approx 0$,

$$e^{-x} \approx 1 - x \tag{2.6}$$

In general,

$$1 - x < e^{-x} \tag{2.7}$$

§2.2 Weighted Set Cover Problem

Definition 2.2 Weighted Set Cover Problem Let V be some universe, $S_1, \ldots, S_m \subseteq V$. Select sets of minimum cost that cover V, where set S_i has cost/weight w_i.

WLOG, we can assume strictly-positive costs (zero cost can be dealt with in pre-processing).

Theorem 2.3 The algorithm for this is the same as before, but we select sets differently. We cannot ignore the cost.

Algorithm 2.1: Polynomial time set cover approximation algorithm

```
U_0 ← V // set of not yet covered elements in V
t ← 0 // iteration counter
for U_t ≠ ∅ do
    Select S_i from sets that maximises new elements per cost |S_i ∩ U_t| / w_i
    Include S_i in soln
    U_t ← U_t \ S_i
    t ← t + 1
return soln
```

So we maximise new elements per cost, or minimise cost per new element.

Proof. Prove by induction, on $|U_t| \leq \left(1 - \frac{1}{k}\right)^t \cdot n$. In this problem, that is analogous to

$$|U_t| \leq \exp\left(-\frac{W_t}{\text{opt}}\right) \cdot n \tag{2.8}$$

Base case, if $t = 0$, then $w_t = 0$ and obviously

$$|U_0| = n \tag{2.9}$$

Inductive step, assume inequality holds for some t,

$$|U_t| \leq \exp\left(-\frac{W_t}{\text{opt}}\right) \cdot n \tag{2.10}$$

we can prove for $t + 1$

$$|U_{t+1}| \leq \exp\left(-\frac{W_{t+1}}{\text{opt}}\right) \cdot n \tag{2.11}$$

Let S_{i_t} be the set we select at step t. Then

$$\frac{|S_{i_t} \cap U_t|}{w_{i_t}} \tag{2.12}$$

is as large as possible per the greedy algorithm.

Lemma 2.4 Claim

$$\frac{|S_{i_t} \cap U_t|}{w_{i_t}} \geq \frac{|U_t|}{\text{opt}} \tag{2.13}$$

Proof of Claim. Let I^* be the set of indices of sets in `opt`.

$$\bigcup_{i \in I^*} S_i \cap U_t = U_t \tag{2.14}$$

This was proven earlier. Based on the proof from before,

$$\sum_{i \in I^*} \frac{|S_i \cap U_t|}{\text{opt}} \geq \frac{|U_t|}{\text{opt}} \tag{2.15}$$

This expands into

$$\sum_{i \in I^*} \frac{|S_i \cap U_t|}{w_i} \cdot \frac{w_i}{\text{opt}} \geq \frac{|U_t|}{\text{opt}} \tag{2.16}$$

What if we only sum the w_i/opt? We get 1. This above dot product is then a weighted sum of elements per cost. We conclude that

$$\exists i \qquad \frac{|S_i \cap U_t|}{w_i} \geq \frac{U_t}{\text{opt}} \tag{2.17}$$

The greedy algorithm will choose the maximum so it will pick this S_i. □

Then,

$$\begin{aligned} |U_{t+1}| &= |U_t| - |(S_{i_t} \cap U_t)| && (2.18) \\ &\leq n \cdot e^{-W_t/\text{opt}} \left(1 - \frac{w_{i_t}}{\text{opt}}\right) && (2.19) \\ &\leq n \cdot e^{-W_t/\text{opt}} \cdot e^{-w_{i_t}/\text{opt}} && (2.20) \\ &\leq n \cdot e^{-W_{t+1}/\text{opt}} && (2.21) \end{aligned}$$

completing the proof. □

§2.3 Similar Problems

Instead of covering all elements, try to cover as many elements as possible

Definition 2.5 Max k Coverage Choose k sets to cover as many elements as possible. Can just look at the unweighted case.

§2.4 Submodular Maximisation

Take some set X, and some subsets 2^X. Let

$$f : 2^X \longrightarrow \mathbb{R}^+ \tag{2.22}$$

Example 2.6 Let $A \subseteq X$, $S_1, \ldots \in A$. Let f be the coverage function,

$$f(A) = \left| \bigcup_{S \in A} S \right| \tag{2.23}$$

Take $A, B \subseteq X$. Obviously,

$$f(A \cup B) \leq f(A) + f(B) \tag{2.24}$$

is always true.

Definition 2.7 Subadditive Function A function

$$f : 2^X \longrightarrow \mathbb{R}^+ \tag{2.25}$$

is subadditive if

$$f(A) + f(B) \geq f(A \cup B) \tag{2.26}$$

Definition 2.8 Submodular Function A function

$$f : 2^X \longrightarrow \mathbb{R}^+ \tag{2.27}$$

is submodular if

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B) \tag{2.28}$$

All submodular functions are also subadditive.

§3 September 23, 2025

§3.1 Submodular Maximisation (cont)

Recall the set cover problem, with some universe, and some set of sets that covers the entire universe. Now, maybe we want to pick a minimal subset that covers the entire universe. Or, we want to pick k sets and maximise the coverage of the universe. Recall the definitions

Definition 3.1 Subadditive Function A function

$$f : 2^X \longrightarrow \mathbb{R}^+ \tag{3.1}$$

is subadditive if

$$f(A) + f(B) \geq f(A \cup B) \tag{3.2}$$

Definition 3.2 Submodular Function A function

$$f : 2^X \longrightarrow \mathbb{R}^+ \tag{3.3}$$

is submodular if

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B) \tag{3.4}$$

Equivalently, we can write

Definition 3.3 Submodular Function A function

$$f : s^X \longrightarrow \mathbb{R}^+ \tag{3.5}$$

is submodular if for two subsets $T \subseteq S \subset X$, and some $x \in X \setminus S$. The submodular property gives

$$f(T \cup \{x\}) - f(T) \geq f(S \cup \{x\}) - f(S) \tag{3.6}$$

A concrete example of this is diminishing utility of some commodity (e.g. money, some ETF, bananas).

Proposition 3.4 These definitions of submodular function are indeed equivalent.

Proof. We can prove this as 3.2 $\iff$ 3.3

(3.2 $\implies$ 3.3) We want to prove

$$f(T \cup \{x\}) - f(T) \overset{?}{\geq} f(S \cup \{x\}) - f(S) \tag{3.7}$$

which is equivalent to

$$f(T \cup \{x\}) + f(S) \overset{?}{\geq} f(S \cup \{x\}) + f(T) \tag{3.8}$$

Let $A = T \cup \{x\}$ and $B = S$. Then,

$$(T \cup \{x\}) \cup S = S \cup \{x\} = A \cup B \tag{3.9}$$

$$(T \cup \{x\}) \cap S = T = A \cap B \tag{3.10}$$

(This is trivial to see with a picture) Thus, the second definition is a consequence of the first.

(3.2 $\impliedby$ 3.3) We want to prove

$$f(A \cup B) - f(A) \leq f(B) - f(A \cap B) \tag{3.11}$$

which is pretty obviously equivalent to the first definition. To show this, initially, set $S = \emptyset$, then grow it to $S = B \setminus A$ one element at a time, which follows from the second definition. Then, the end result is

$$f(S \cup A) - f(A) \leq f(S \cup (A \cap B)) - f(A \cap B) \tag{3.12}$$

which for $S = B \setminus A$, is equivalent to

$$f(A \cup B) - f(A) \leq f(B) - f(A \cap B) \tag{3.13}$$

(This is true via some very basic basic set theory, but is not basic to see. It is more clear with a picture)

□

§3.2 Back to Coverage Functions

Let U be some universe, with subsets $S_1, \ldots, S_m \subseteq U$, and $X = \{S_1, \ldots, S_m\}$. Let the coverage function

$$f(A \in X) = \left| \bigcup_{S_i \in A} S_i \right| \tag{3.14}$$

Concretely, suppose we have

$$f(\{S_1, S_3\} \cup \{S_2\}) - f(\{S_1, S_3\}) = |S_2 \setminus S_1 \setminus S_3| \tag{3.15}$$

If we also have S_4, then

$$f(\{S_1, S_3, S_4\} \cup \{S_2\}) - f(\{S_1, S_3, S_4\}) = |S_2 \setminus S_1 \setminus S_3 \setminus S_4| \leq |S_2 \setminus S_1 \setminus S_3| \tag{3.16}$$

i.e. there are 'fewer elements' in the delta of the coverage function. (This is an obvious example, but it is still quite concrete and thus useful.)

§3.3 Maximisation

Theorem 3.5 The greedy algorithm gives a $1 - e^{-1}$ approximation for monotone submodular maximisation problem in which we need to select a given number of elements.

Note Let f be some monotone function such that

$$f(S \cup \{x\}) \geq f(S) \tag{3.17}$$

Goal is to select k elements $x_1, \ldots, x_k$ to maximise

$$f(\{x_1, \ldots, x_k\}) \tag{3.18}$$

Assume that $f(\emptyset) \equiv 0$

Proposition 3.6 We specifically want to prove

$$\texttt{ALG} \geq \left(1 - \frac{1}{e}\right) \cdot \texttt{OPT} \tag{3.19}$$

Proof. Denote $\texttt{ALG}_i$ as the value that the greedy algorithm gets after i steps. Also denote

$$\Lambda \equiv \{x_1^*, \ldots, x_k^*\} \tag{3.20}$$

be an optimal solution. Finally, denote

$$\Delta_i \equiv \texttt{OPT} - \texttt{ALG}_i \tag{3.21}$$

which we can show shrinks fast. We thus need

$$\texttt{ALG}_{i+1} - \texttt{ALG}_i = ? \tag{3.22}$$

Suppose the algorithm has some set A_j at step i. Then,

$$\texttt{ALG}_i \equiv f(A_j) \implies f(A_i \cup \Lambda) \geq \texttt{OPT} \tag{3.23}$$

Every time we add some $x_j^* \in \Lambda$, we are always lower-bounded on f by $\texttt{OPT}$.

$$f(A_j \cup \{x_1^*, \ldots, x_{i+1}^*\}) - f(A_j \cup \{x_1^*, \ldots, x_i^*\}) \geq \frac{\texttt{OPT} - \texttt{ALG}_j}{k} \tag{3.24}$$

We can use some submodular math to rewrite

$$f(A_j \cup \{x_{i+1}^*\}) - f(A_j) \geq \texttt{above} \tag{3.25}$$

We ultimately conclude that

$$f(A_{j+1}) - f(A_j) \geq \frac{\texttt{OPT} - \texttt{ALG}_j}{k} \tag{3.26}$$

The desired result thus 'follows very easily'[1].

$$\texttt{OPT} - f(A_{j+1}) \leq \texttt{OPT} - \left(f(A_j) + \frac{\texttt{OPT} - \texttt{ALG}_j}{k} \right) \tag{3.27}$$

$$= (\texttt{OPT} - f(A_j)) \cdot (1 - 1/k) \tag{3.28}$$

After k steps, the upper-bound becomes

$$\left((1 - 1/k)^k \leq 1/e \right) \cdot \texttt{OPT} \tag{3.29}$$

So we finally get

$$f(A_k) \geq \texttt{OPT} \cdot (1 - 1/e) \tag{3.30}$$

□

[1] does it?