Submodular Functions, Optimization, and Applications to Machine Learning
— Fall Quarter, Lecture 13 —

Prof. Jeff Bilmes

University of Washington, Seattle
Department of Electrical Engineering

http://melodi.ee.washington.edu/~bilmes

Nov 16th, 2020

\[
f(A) + f(B) \geq f(A \cup B) + f(A \cap B)
\]

f(A) + f(B) = f(A \cup B) + f(A \cap B)

f(A) + f(B) = f(A \cup B) + f(A \cap B)

f(A) + f(B) = f(A \cup B) + f(A \cap B)

f(A) + f(B) = f(A \cup B) + f(A \cap B)

f(A) + f(B) = f(A \cup B) + f(A \cap B)

f(A) + f(B) = f(A \cup B) + f(A \cap B)

f(A) + f(B) = f(A \cup B) + f(A \cap B)
Homework 3, out, due next Wednesday, Nov 25th, 2020, 11:59pm.

Reminder, all lectures are being recorded and posted to youtube. To get the links, see our announcements (https://canvas.uw.edu/courses/1397085/announcements).

Office hours this week, Wed & Thur, 10:00pm at our class zoom link.

Next week office hours, Tues (11/24) & Wed (11/25), 10:00pm at our class zoom link.
Class Road Map - EE563

L1(9/30): Motivation, Applications, Definitions, Properties
L2(10/5): Sums concave(modular), uses (diversity/costs, feature selection), information theory
L3(10/7): Monge, More Definitions, Graph and Combinatorial Examples,
L4(10/12): Graph & Combinatorial Examples, Matrix Rank, Properties, Other Defs, Independence
L5(10/14): Properties, Defs of Submodularity, Independence
L6(10/19): Matroids, Matroid Examples, Matroid Rank,
L8(10/26): Transversal Matroid, Matroid and representation, Dual Matroid
L9(10/28): Other Matroid Properties, Combinatorial Geometries, Matroid and Greedy, Polyhedra, Matroid Polytopes
L10(11/2): Matroid Polytopes, Matroids → Polymatroids
L11(11/4): Matroids → Polymatroids, Polymatroids
L12(11/9): Polymatroids, Polymatroids and Greedy
L13(11/11): Veterans Day, Holiday
L13(11/16): Polymatroids and Greedy, Possible Polytopes, Extreme Points, Cardinality Constrained Maximization

Last day of instruction, Fri. Dec 11th. Finals Week: Dec 12-18, 2020
A polymatroid is a polymatroid function’s polytope

- So, when $f$ is a polymatroid function, $P_f^+$ is a polymatroid.
- Is it the case that, conversely, for any polymatroid $P$, there is an associated polymatroidal function $f$ such that $P = P_f^+$?

**Theorem 13.2.1**

For any polymatroid $P$ (compact subset of $\mathbb{R}_{+}^E$, zero containing, down-monotone, and $\forall x \in \mathbb{R}_{+}^E$ any maximal independent subvector $y \leq x$ has same component sum $y(E) = \text{rank}(x)$), there is a polymatroid function $f : 2^E \rightarrow \mathbb{R}$ (normalized, monotone non-decreasing, submodular) such that $P = P_f^+$ where $P_f^+ = \{ x \in \mathbb{R}^E : x \geq 0, x(A) \leq f(A), \forall A \subseteq E \}$. 
Tight sets $\mathcal{D}(y)$ are closed, and max tight set $\text{sat}(y)$

Recall the definition of the set of tight sets at $y \in P_f^+$:

$$\mathcal{D}(y) \triangleq \{ A : A \subseteq E, \ y(A) = f(A) \}$$  \hspace{1cm} (13.1)

### Theorem 13.2.1

For any $y \in P_f^+$, with $f$ a polymatroid function, then $\mathcal{D}(y)$ is closed under union and intersection.

### Proof.

We have already proven this as part of Theorem ??

Also recall the definition of $\text{sat}(y)$, the maximal set of tight elements relative to $y \in \mathbb{R}_+^E$.

$$\text{sat}(y) \overset{\text{def}}{=} \bigcup \{ T : T \in \mathcal{D}(y) \}$$  \hspace{1cm} (13.2)
Join $\lor$ and meet $\land$ for $x, y \in \mathbb{R}_+^E$

For $x, y \in \mathbb{R}_+^E$, define vectors $x \land y \in \mathbb{R}_+^E$ and $x \lor y \in \mathbb{R}_+^E$ such that, for all $e \in E$

\[
(x \lor y)(e) = \max(x(e), y(e)) \quad \text{(13.1)}
\]
\[
(x \land y)(e) = \min(x(e), y(e)) \quad \text{(13.2)}
\]

Hence,

\[
x \lor y \triangleq \left( \max(x(e_1), y(e_1)), \max(x(e_2), y(e_2)), \ldots, \max(x(e_n), y(e_n)) \right)
\]

and similarly

\[
x \land y \triangleq \left( \min(x(e_1), y(e_1)), \min(x(e_2), y(e_2)), \ldots, \min(x(e_n), y(e_n)) \right)
\]

From this, we can define things like an lattices, and other constructs.
Vector rank, $\text{rank}(x)$, is submodular

- Recall that the matroid rank function $r(A) = \max(|I| : I \subseteq A : I \in \mathcal{I})$ is submodular.
- The vector rank function $\text{rank}(x) = \max (y(E) : y \leq x, y \in P)$ also satisfies a form of submodularity, namely one defined on the real lattice.

**Theorem 13.2.1 (vector rank and submodularity)**

Let $P$ be a polymatroid polytope. The vector rank function $\text{rank} : \mathbb{R}^E_+ \to \mathbb{R}$ with $\text{rank}(x) = \max (y(E) : y \leq x, y \in P)$ satisfies, for all $u, v \in \mathbb{R}^E_+$

$$\text{rank}(u) + \text{rank}(v) \geq \text{rank}(u \vee v) + \text{rank}(u \wedge v) \quad (13.1)$$

- Note what happens when $u, v \in \{0, 1\}^E \subseteq \mathbb{R}^E_+$. 

Polymatroidal polyhedron and the greedy solution

- What is the greedy solution for $\max \{wx : x \in P^+_f\}$, when $w \in \mathbb{R}^E$?
- Sort elements of $E$ w.r.t. $w$ so that, w.l.o.g. $E = (e_1, e_2, \ldots, e_m)$ with $w(e_1) \geq w(e_2) \geq \cdots \geq w(e_m)$.
- Let $k + 1$ be the first point (if any) at which we are non-positive, i.e., $w(e_k) > 0$ and $0 \geq w(e_{k+1})$.
- Next define partial accumulated sets $E_i$, for $i = 0 \ldots m$, we have w.r.t. the above sorted order:

$$E_i \overset{\text{def}}{=} \{e_1, e_2, \ldots e_i\} \quad (13.22)$$

(note $E_0 = \emptyset$, $f(E_0) = 0$, and $E$ and $E_i$ is always sorted w.r.t. $w$).

- The greedy solution is the vector $x \in \mathbb{R}^E_+$ with elements defined as:

$$x(e_1) \overset{\text{def}}{=} f(E_1) = f(e_1) = f(e_1|E_0) = f(e_1|\emptyset) \quad (13.23)$$

$$x(e_i) \overset{\text{def}}{=} f(E_i) - f(E_{i-1}) = f(e_i|E_{i-1}) \text{ for } i = 2 \ldots k \quad (13.24)$$

$$x(e_i) \overset{\text{def}}{=} 0 \text{ for } i = k + 1 \ldots m = |E| \quad (13.25)$$
Theorem 13.2.2

The vector $x \in \mathbb{R}^E_+$ as previously defined using the greedy algorithm maximizes $wx$ over $P^+_f$, with $w \in \mathbb{R}^E_+$, if $f$ is submodular.

Proof.

Consider the LP strong duality equation:

$$\max(wx : x \in P^+_f) = \min \left( \sum_{A \subseteq E} y_A f(A) : y \in \mathbb{R}^{2E}_+, \sum_{A \subseteq E} y_A 1_A \geq w \right)$$

(13.21)

Sort $E$ by $w$ descending, and define the following vector $y \in \mathbb{R}^{2E}_+$ as

$$y_{E_i} \leftarrow w(e_i) - w(e_{i+1}) \text{ for } i = 1 \ldots (m - 1),$$

(13.22)

$$y_E \leftarrow w(e_m), \text{ and}$$

(13.23)

$$y_A \leftarrow 0 \text{ otherwise}$$

(13.24)
**Theorem 13.2.2**

Conversely, suppose $P_f^+$ is a polytope of form

$$P_f^+ = \left\{ x \in \mathbb{R}_+^E : x(A) \leq f(A), \forall A \subseteq E \right\},$$

then the greedy solution to

$$\text{max}(wx : x \in P_f^+)$$

is optimum only if $f$ is submodular.

**Proof.**

- Choose $A$ and $B$ arbitrarily, and then order elements of $E$ as $(e_1, e_2, \ldots, e_m)$, with $E_i = (e_1, e_2, \ldots, e_i)$, so the following is true:
  - For $1 \leq p \leq q \leq m$, $A = \{e_1, e_2, \ldots, e_k, e_{k+1}, \ldots, e_p\} = E_p$ and $B = \{e_1, e_2, \ldots, e_k, e_{p+1}, \ldots, e_q\} = E_k \cup (E_q \setminus E_p) = (A \cap B) \cup (B \setminus A)$
  - Note, then we have $A \cap B = \{e_1, \ldots, e_k\} = E_k$, and $A \cup B = E_q$. 

The next slide comes from lecture 9.
Matroid and the greedy algorithm

- Let \((E, \mathcal{I})\) be an independence system, and we are given a non-negative modular weight function \(w: E \to \mathbb{R}_+\).

**Algorithm 1:** The Matroid Greedy Algorithm

1. Set \(X \leftarrow \emptyset\);
2. while \(\exists v \in E \setminus X\) s.t. \(X \cup \{v\} \in \mathcal{I}\) do
3. \(v \in \arg\max \{w(v) : v \in E \setminus X, X \cup \{v\} \in \mathcal{I}\}\);
4. \(X \leftarrow X \cup \{v\}\);

- Same as sorting items by decreasing weight \(w\), and then choosing items in that order that retain independence.

**Theorem 13.3.4**

Let \((E, \mathcal{I})\) be an independence system. Then the pair \((E, \mathcal{I})\) is a matroid if and only if for each weight function \(w \in \mathcal{R}_+^E\), Algorithm ?? above leads to a set \(I \in \mathcal{I}\) of maximum weight \(w(I)\).
Thus, restating the above results into a single complete theorem, we have a result very similar to what we saw for matroids (i.e., Theorem 10.3.2)

**Theorem 13.3.1**

If $f : 2^E \to \mathbb{R}_+$ is given, and $P$ is a polytope in $\mathbb{R}_+^E$ of the form

$$P = \{ x \in \mathbb{R}_+^E : x(A) \leq f(A), \forall A \subseteq E \},$$

then the greedy solution to the problem $\max (w^\top x : x \in P)$ is $\forall w$ optimum iff $f$ is monotone non-decreasing submodular (i.e., iff $P$ is a polymatroid).
Multiple Polytopes associated with arbitrary $f$

• Given an arbitrary submodular function $f : 2^V \rightarrow \mathbb{R}$ (not necessarily a polymatroid function, so it need not be positive, monotone, etc.).
Multiple Polytopes associated with arbitrary $f$

- Given an arbitrary submodular function $f : 2^V \rightarrow R$ (not necessarily a polymatroid function, so it need not be positive, monotone, etc.).
- If $f(\emptyset) \neq 0$, can set $f'(A) = f(A) - f(\emptyset)$ without destroying submodularity. This does not change any minima, (i.e., $\arg\min_A f(A) = \arg\min_{A'} f'(A)$) so we often assume all functions are normalized $f(\emptyset) = 0$.

Note that due to constraint $x(\emptyset) \leq f(\emptyset)$, we must have $f(\emptyset) \geq 0$ since if not (i.e., if $f(\emptyset) < 0$), then $P_f^+$ doesn’t exist.

Another form of normalization takes the form:

$$f'(A) = \begin{cases} f(A) & \text{if } A \neq \emptyset \\ 0 & \text{if } A = \emptyset \end{cases} \quad (13.1)$$

This preserves submodularity due to $f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$, and if $A \cap B = \emptyset$ then r.h.s. only gets smaller when $f(\emptyset) \geq 0$. 

---

**Realizing $f$ using different polytopes**

Given a function $f$.

1. $P_f$: All $x$ such that $x(A) \leq f(A)$ for each $A$. This is the extended polytope.
2. $P_f^+$: $P_f$ restricted to the positive orthant.
3. $B_f$: $P_f$ restricted such that $x(E) = f(E)$.

For $P_f^+$, we can be greedy.

For $B_f$, we need to consider all $x_1, \ldots, x_k$ such that $x_i(E) = f(E)$. For this, a polynomial-time approach is to:

- For some $i$, $x_i(E) = f(E)$:
  - $B_f$ is a polyhedron in $R^n$ with $n \leq |E|$. It is convex, the convex hull of these points, and we can use a LP approach.
  - $P_f$ is a polyhedron with a number of points $O(2^n)$.

For $B_f$:

- We can solve this using linear programming.
- Alternatively, we can use a greedy algorithm as long as:
  - $P_f^+$ is a polyhedron.
  - $f$ is submodular.

For $P_f$: We can use a greedy algorithm. However, it is not polyhedral, and we need a different approach.
Multiple Polytopes associated with arbitrary $f$

- Given an arbitrary submodular function $f : 2^V \rightarrow R$ (not necessarily a polymatroid function, so it need not be positive, monotone, etc.).
- If $f(\emptyset) \neq 0$, can set $f'(A) = f(A) - f(\emptyset)$ without destroying submodularity. This does not change any minima, (i.e., $\arg\min_A f(A) = \arg\min_A f'(A)$) so we often assume all functions are normalized $f(\emptyset) = 0$.

- We can define several polytopes:

$$P_f = \{x \in \mathbb{R}^E : x(S) \leq f(S), \forall S \subseteq E\} \quad (13.1)$$

$$P_f^+ = P_f \cap \{x \in \mathbb{R}^E : x \geq 0\} \quad (13.2)$$

$$B_f = P_f \cap \{x \in \mathbb{R}^E : x(E) = f(E)\} \quad (13.3)$$
Given an arbitrary submodular function \( f : 2^V \rightarrow R \) (not necessarily a polymatroid function, so it need not be positive, monotone, etc.).

If \( f(\emptyset) \neq 0 \), can set \( f'(A) = f(A) - f(\emptyset) \) without destroying submodularity. This does not change any minima, (i.e., \( \arg\min_A f(A) = \arg\min_A f'(A) \)) so we often assume all functions are normalized \( f(\emptyset) = 0 \).

We can define several polytopes:

\[
P_f = \{ x \in \mathbb{R}^E : x(S) \leq f(S), \forall S \subseteq E \} \tag{13.1}
\]

\[
P_f^+ = P_f \cap \{ x \in \mathbb{R}^E : x \geq 0 \} \tag{13.2}
\]

\[
B_f = P_f \cap \{ x \in \mathbb{R}^E : x(E) = f(E) \} \tag{13.3}
\]

\( P_f \) is what is sometimes called the extended polytope (sometimes notated as \( EP_f \)).
Multiple Polytopes associated with arbitrary $f$

- Given an arbitrary submodular function $f : 2^V \to \mathbb{R}$ (not necessarily a polymatroid function, so it need not be positive, monotone, etc.).
- If $f(\emptyset) \neq 0$, can set $f'(A) = f(A) - f(\emptyset)$ without destroying submodularity. This does not change any minima, (i.e., $\text{argmin}_A f(A) = \text{argmin}_A f'(A)$) so we often assume all functions are normalized $f(\emptyset) = 0$.
- We can define several polytopes:

$$P_f = \{ x \in \mathbb{R}^E : x(S) \leq f(S), \forall S \subseteq E \} \quad (13.1)$$

$$P_f^+ = P_f \cap \{ x \in \mathbb{R}^E : x \geq 0 \} \quad (13.2)$$

$$B_f = P_f \cap \{ x \in \mathbb{R}^E : x(E) = f(E) \} \quad (13.3)$$

- $P_f$ is what is sometimes called the extended polytope (sometimes notated as $EP_f$).
- $P_f^+$ is $P_f$ restricted to the positive orthant.
Multiple Polytopes associated with arbitrary $f$

- Given an arbitrary submodular function $f : 2^V \rightarrow R$ (not necessarily a polymatroid function, so it need not be positive, monotone, etc.).

- If $f(\emptyset) \neq 0$, can set $f'(A) = f(A) - f(\emptyset)$ without destroying submodularity. This does not change any minima, (i.e., $\arg\min_A f(A) = \arg\min_A f'(A)$) so we often assume all functions are normalized $f(\emptyset) = 0$.

- We can define several polytopes:

  $$P_f = \{ x \in \mathbb{R}^E : x(S) \leq f(S), \forall S \subseteq E \} \quad (13.1)$$

  $$P_f^+ = P_f \cap \{ x \in \mathbb{R}^E : x \geq 0 \} \quad (13.2)$$

  $$B_f = P_f \cap \{ x \in \mathbb{R}^E : x(E) = f(E) \} \quad (13.3)$$

- $P_f$ is what is sometimes called the extended polytope (sometimes notated as $EP_f$).

- $P_f^+$ is $P_f$ restricted to the positive orthant.

- $B_f$ is called the base polytope, analogous to the base in matroid.
Multiple Polytopes in 2D associated with $f$

$$P_f^+ = P_f \cap \{x \in \mathbb{R}^E : x \geq 0\}$$  \hspace{1cm} (13.4)

$$P_f = \{x \in \mathbb{R}^E : x(S) \leq f(S), \forall S \subseteq E\}$$  \hspace{1cm} (13.5)

$$B_f = P_f \cap \{x \in \mathbb{R}^E : x(E) = f(E)\}$$  \hspace{1cm} (13.6)
Multiple Polytopes in 2D associated with $f$

\[ P_f^+ = P_f \cap \{ x \in \mathbb{R}^E : x \geq 0 \} \]  
(13.4)

\[ P_f = \{ x \in \mathbb{R}^E : x(S) \leq f(S), \forall S \subseteq E \} \]  
(13.5)

\[ B_f = P_f \cap \{ x \in \mathbb{R}^E : x(E) = f(E) \} \]  
(13.6)
Multiple Polytopes in 2D associated with \( f \)

\[
P_f^+ = P_f \cap \{ x \in \mathbb{R}^E : x \geq 0 \}\]

\[
P_f = \{ x \in \mathbb{R}^E : x(S) \leq f(S), \forall S \subseteq E \}\]

\[
B_f = P_f \cap \{ x \in \mathbb{R}^E : x(E) = f(E) \}\]
Base Polytope in 3D

\[ P_f = \{ x \in \mathbb{R}^E : x(S) \leq f(S), \forall S \subseteq E \} \]  
\[ B_f = P_f \cap \{ x \in \mathbb{R}^E : x(E) = f(E) \} \]
A polymatroid function’s polyhedron is a polymatroid.

**Theorem 13.4.1**

Let \( f \) be a submodular function defined on subsets of \( E \). For any \( x \in \mathbb{R}^E \), we have:

\[
\text{rank}(x) = \max (y(E) : y \leq x, y \in P_f) = \min (x(A) + f(E \setminus A) : A \subseteq E)
\]

(13.9)

Essentially the same theorem as Theorem 11.4.1, but note \( P_f \) rather than \( P_f^+ \). Taking \( x = 0 \) we get:

**Corollary 13.4.2**

Let \( f \) be a submodular function defined on subsets of \( E \). We have:

\[
\text{rank}(0) = \max (y(E) : y \leq 0, y \in P_f) = \min (f(A) : A \subseteq E)
\]

(13.10)
Proof of Theorem 13.4.1

Proof Thm 13.4.1: \( \max (y(E) : y \leq x, y \in P_f) = \min (x(A) + f(E \setminus A) : A \subseteq E) \).

- Let \( y^* \) be optimal solution of the l.h.s. and let \( A \subseteq E \) be any subset.

\[
\begin{align*}
\text{Let } y^* \text{ be optimal solution of the l.h.s. and let } A \subseteq E \text{ be any subset.}
\end{align*}
\]
Proof Thm 13.4.1: \( \max (y(E) : y \leq x, y \in P_f) = \min (x(A) + f(E \setminus A) : A \subseteq E) \).

- Let \( y^* \) be optimal solution of the l.h.s. and let \( A \subseteq E \) be any subset.
- Then \( y^*(E) = y^*(A) + y^*(E \setminus A) \leq f(A) + x(E \setminus A) \) (since if \( y^* \in P_f \) then \( y^*(A) \leq f(A) \), and since \( y^* \leq x \) then \( y^*(E \setminus A) \leq x(E \setminus A) \)).

This is a form of weak duality.
Proof of Theorem 13.4.1

Proof Thm 13.4.1: \( \max (y(E) : y \leq x, y \in P_f) = \min (x(A) + f(E \setminus A) : A \subseteq E) \).

- Let \( y^* \) be optimal solution of the l.h.s. and let \( A \subseteq E \) be any subset.
- Then \( y^*(E) = y^*(A) + y^*(E \setminus A) \leq f(A) + x(E \setminus A) \) (since if \( y^* \in P_f \) then \( y^*(A) \leq f(A) \), and since \( y^* \leq x \) then \( y^*(E \setminus A) \leq x(E \setminus A) \)).
  This is a form of weak duality.
- For any \( e \in E \), if \( y^*(e) < x(e) \), must be some reason other than constraint \( y^* \leq x \), namely must be that \( \exists T \in D(y^*) \) with \( e \in T \) (i.e., \( e \) is a member of at least one of the tight sets).
Proof Thm 13.4.1: \( \max (y(E) : y \leq x, y \in P_f) = \min (x(A) + f(E \setminus A) : A \subseteq E) \).

- Let \( y^* \) be optimal solution of the l.h.s. and let \( A \subseteq E \) be any subset.
- Then \( y^*(E) = y^*(A) + y^*(E \setminus A) \leq f(A) + x(E \setminus A) \) (since if \( y^* \in P_f \) then \( y^*(A) \leq f(A) \), and since \( y^* \leq x \) then \( y^*(E \setminus A) \leq x(E \setminus A) \)). This is a form of weak duality.
- For any \( e \in E \), if \( y^*(e) < x(e) \), must be some reason other than constraint \( y^* \leq x \), namely must be that \( \exists T \in D(y^*) \) with \( e \in T \) (i.e., \( e \) is a member of at least one of the tight sets). I.e., given \( e \notin \text{sat}(y^*) \), then \( y^*(A) < f(A) \forall A \ni e \) including \( \{e\} \), hence \( x(e) < f(e) \).
Proof of Theorem 13.4.1

Proof Thm 13.4.1: \[ \max (y(E) : y \leq x, y \in P_f) = \min (x(A) + f(E \setminus A) : A \subseteq E). \]

- Let \( y^* \) be optimal solution of the l.h.s. and let \( A \subseteq E \) be any subset.
- Then \( y^*(E) = y^*(A) + y^*(E \setminus A) \leq f(A) + x(E \setminus A) \) (since if \( y^* \in P_f \) then \( y^*(A) \leq f(A) \), and since \( y^* \leq x \) then \( y^*(E \setminus A) \leq x(E \setminus A) \)).
  This is a form of weak duality.

- For any \( e \in E \), if \( y^*(e) < x(e) \), must be some reason other than constraint \( y^* \leq x \), namely must be that \( \exists T \in D(y^*) \) with \( e \in T \) (i.e., \( e \) is a member of at least one of the tight sets). I.e., given \( e \notin \text{sat}(y^*) \), then \( y^*(A) < f(A) \forall A \ni e \) including \( \{e\} \), hence \( x(e) < f(e) \).
  Conversely, \( e \in \text{sat}(y^*) \) means \( \exists T \in D(y^*) \), w. \( e \in T \) & \( y^*(T) = f(T) \).
**Proof of Theorem 13.4.1**

Let $y^*$ be optimal solution of the l.h.s. and let $A \subseteq E$ be any subset.

Then $y^*(E) = y^*(A) + y^*(E \setminus A) \leq f(A) + x(E \setminus A)$ (since if $y^* \in P_f$ then $y^*(A) \leq f(A)$, and since $y^* \leq x$ then $y^*(E \setminus A) \leq x(E \setminus A)$).

This is a form of weak duality.

For any $e \in E$, if $y^*(e) < x(e)$, must be some reason other than constraint $y^* \leq x$, namely must be that $\exists T \in \mathcal{D}(y^*)$ with $e \in T$ (i.e., $e$ is a member of at least one of the tight sets). I.e., given $e \notin \text{sat}(y^*)$, then $y^*(A) < f(A) \forall A \ni e$ including $\{e\}$, hence $x(e) < f(e)$.

Conversely, $e \in \text{sat}(y^*)$ means $\exists T \in \mathcal{D}(y^*)$, w. e $\in T$ & $y^*(T) = f(T)$.

Hence, for all $e \notin \text{sat}(y^*)$ we have $y^*(e) = x(e)$, and moreover $y^*(\text{sat}(y^*)) = f(\text{sat}(y^*))$ by definition.
Proof of Theorem 13.4.1

Proof Thm 13.4.1: \( \max (y(E) : y \leq x, y \in P_f) = \min (x(A) + f(E \setminus A) : A \subseteq E) \).

- Let \( y^* \) be optimal solution of the l.h.s. and let \( A \subseteq E \) be any subset.
- Then \( y^*(E) = y^*(A) + y^*(E \setminus A) \leq f(A) + x(E \setminus A) \) (since if \( y^* \in P_f \) then \( y^*(A) \leq f(A) \), and since \( y^* \leq x \) then \( y^*(E \setminus A) \leq x(E \setminus A) \)). This is a form of weak duality.
- For any \( e \in E \), if \( y^*(e) < x(e) \), must be some reason other than constraint \( y^* \leq x \), namely must be that \( \exists T \in D(y^*) \) with \( e \in T \) (i.e., \( e \) is a member of at least one of the tight sets). I.e., given \( e \notin \text{sat}(y^*) \), then \( y^*(A) < f(A) \forall A \ni e \) including \( \{e\} \), hence \( x(e) < f(e) \).
  Conversely, \( e \in \text{sat}(y^*) \) means \( \exists T \in D(y^*) \), w. \( e \in T \) & \( y^*(T) = f(T) \).
- Hence, for all \( e \notin \text{sat}(y^*) \) we have \( y^*(e) = x(e) \), and moreover \( y^*(\text{sat}(y^*)) = f(\text{sat}(y^*)) \) by definition.
- Thus \( y^*(\text{sat}(y^*)) + y^*(E \setminus \text{sat}(y^*)) = f(\text{sat}(y^*)) + x(E \setminus \text{sat}(y^*)) \), strong duality, showing that the two sides are equal for \( y^* \).
Greedy and $P_f$

- In Theorem 12.4.1 (i.e., greedy solution in $P_f^+$), we can relax $P_f^+$ to $P_f$ (prime and dual feasibility still hold as does strong duality).
In Theorem 12.4.1 (i.e., greedy solution in $P^+_f$), we can relax $P^+_f$ to $P_f$ (prime and dual feasibility still hold as does strong duality). That is, the proof still shows that $x \in P_f$ in this relaxed case.
Greedy and $P_f$

- In Theorem 12.4.1 (i.e., greedy solution in $P_f^+$), we can relax $P_f^+$ to $P_f$ (prime and dual feasibility still hold as does strong duality). That is, the proof still shows that $x \in P_f$ in this relaxed case.

- If $\exists e$ such that $w(e) < 0$, however, then $\max(wx : x \in P_f) = \infty$ since we can let $x_e \to \infty$, unless we ignore the negative elements or assume $w \geq 0$. 

Prof. Jeff Bilmes
Greedy and $P_f$

- In Theorem 12.4.1 (i.e., greedy solution in $P_f^+$), we can relax $P_f^+$ to $P_f$ (prime and dual feasibility still hold as does strong duality). That is, the proof still shows that $x \in P_f$ in this relaxed case.
- If $\exists e$ such that $w(e) < 0$, however, then $\max(wx : x \in P_f) = \infty$ since we can let $x_e \to \infty$, unless we ignore the negative elements or assume $w \geq 0$.
- Moreover, in either $P_f$, or $P_f^+$ case, since the greedy constructed an $x$ has $x(E) = f(E)$, we have that the greedy $x \in B_f$. 

In Theorem 12.4.1 (i.e., greedy solution in $P_f^+$), we can relax $P_f^+$ to $P_f$ (prime and dual feasibility still hold as does strong duality). That is, the proof still shows that $x \in P_f$ in this relaxed case.

If $\exists e$ such that $w(e) < 0$, however, then $\max(wx : x \in P_f) = \infty$ since we can let $x_e \to \infty$, unless we ignore the negative elements or assume $w \geq 0$.

Moreover, in either $P_f$, or $P_f^+$ case, since the greedy constructed an $x$ has $x(E) = f(E)$, we have that the greedy $x \in B_f$.

We might thus be more interested in $\max(wx : x \in B_f)$ when $w$ is an arbitrary vector.
**Greedy and $P_f$**

- In Theorem 12.4.1 (i.e., greedy solution in $P_f^+$), we can relax $P_f^+$ to $P_f$ (prime and dual feasibility still hold as does strong duality). That is, the proof still shows that $x \in P_f$ in this relaxed case.

- If $\exists e$ such that $w(e) < 0$, however, then $\max(wx : x \in P_f) = \infty$ since we can let $x_e \to \infty$, unless we ignore the negative elements or assume $w \geq 0$.

- Moreover, in either $P_f$, or $P_f^+$ case, since the greedy constructed an $x$ has $x(E) = f(E)$, we have that the greedy $x \in B_f$.

- We might thus be more interested in $\max(wx : x \in B_f)$ when $w$ is an arbitrary vector.

- In fact, we will see, in the next section, that the full run of the greedy algorithm producing $x$ is in fact a vertex of $B_f$. 
Recall that Theorem 11.4.1 states that
\[
\max \left( y(E) : y \leq x, y \in P_f^+ \right) = \min (x(A) + f(E \setminus A) : A \subseteq E)
\]
Greedy and $P_f$

- Recall that Theorem 11.4.1 states that
  \[ \max \left( y(E) : y \leq x, y \in P_f^+ \right) = \min \left( x(A) + f(E \setminus A) : A \subseteq E \right) \]

- Theorem 12.4.1 states that greedy algorithm maximizes $wx$ over $P_f^+$ for $w \in \mathbb{R}_E^+$ with $f$ being submodular.
Recall that Theorem 11.4.1 states that
\[
\max \left( y(E) : y \leq x, y \in P_f^+ \right) = \min \left( x(A) + f(E \setminus A) : A \subseteq E \right)
\]

Theorem 12.4.1 states that greedy algorithm maximizes \(wx\) over \(P_f^+\) for \(w \in \mathbb{R}_+^E\) with \(f\) being submodular.

Above implies that Theorem 12.4.1 can be generalized to over \(P_f\) and that greedy solution gives a point in \(B_f\), even for arbitrary finite \(w\).
Polymatroid extreme points

- The greedy algorithm does more than solve $\max(wx : x \in P_f^+)$.
  - We can use it to generate vertices of polymatroidal polytopes.
Polymatroid extreme points

- The greedy algorithm does more than solve $\max(wx : x \in P_f^+)$. We can use it to generate vertices of polymatroidal polytopes.
- Consider $P_f^+$ and also $C_f^+ \overset{\text{def}}{=} \{ x : x \in \mathbb{R}^E_+, x(e) \leq f(e), \forall e \in E \}$
Polymatroid extreme points

- The greedy algorithm does more than solve $\max(wx : x \in P_f^+)$. We can use it to generate vertices of polymatroidal polytopes.

- Consider $P_f^+$ and also $C_f^+ \overset{\text{def}}{=} \{ x : x \in \mathbb{R}^E_+, x(e) \leq f(e), \forall e \in E \}$

- Then ordering $A = (a_1, \ldots, a_{|A|})$ arbitrarily with $A_i = \{a_1, \ldots, a_i\}$, $f(A) = \sum_i f(a_i | A_{i-1}) \leq \sum_i f(a_i)$, and hence $P_f^+ \subseteq C_f^+$. 
Polymatroid extreme points

- The greedy algorithm does more than solve $\max(wx : x \in P_f^+)$. We can use it to generate vertices of polymatroidal polytopes.
- Consider $P_f^+$ and also $C_f^+ \overset{\text{def}}{=} \{ x : x \in \mathbb{R}^E_+, x(e) \leq f(e), \forall e \in E \}$
- Then ordering $A = (a_1, \ldots, a_{|A|})$ arbitrarily with $A_i = \{a_1, \ldots, a_i\}$, $f(A) = \sum_i f(a_i|A_{i-1}) \leq \sum_i f(a_i)$, and hence $P_f^+ \subseteq C_f^+$. 
Polymatroid extreme points

Since $w \in \mathbb{R}^E_+$ is arbitrary, it may be that any $e \in E$ is max (i.e., is such that $w(e) > w(e')$ for $e' \in E \setminus \{e\}$).
Polymatroid extreme points

- Since $w \in \mathbb{R}^E_+$ is arbitrary, it may be that any $e \in E$ is max (i.e., is such that $w(e) > w(e')$ for $e' \in E \setminus \{e\}$).

- Thus, intuitively, any first vertex of the polytope away from the origin might be obtained by advancing along the corresponding axis.
Polymatroid extreme points

- Since $w \in \mathbb{R}_+^E$ is arbitrary, it may be that any $e \in E$ is max (i.e., is such that $w(e) > w(e')$ for $e' \in E \setminus \{e\}$).

- Thus, intuitively, any first vertex of the polytope away from the origin might be obtained by advancing along the corresponding axis.

- Recall, base polytope defined as the extreme face of $P_f$. I.e.,

$$B_f = P_f \cap \{ x \in \mathbb{R}_+^E : x(E) = f(E) \} \quad (13.11)$$
Polymatroid extreme points

- Since \( w \in \mathbb{R}_+^E \) is arbitrary, it may be that any \( e \in E \) is max (i.e., is such that \( w(e) > w(e') \) for \( e' \in E \setminus \{e\} \)).

- Thus, intuitively, any first vertex of the polytope away from the origin might be obtained by advancing along the corresponding axis.

- Recall, base polytope defined as the extreme face of \( P_f \). I.e.,

\[
B_f = P_f \cap \{ x \in \mathbb{R}_+^E : x(E) = f(E) \} \quad (13.11)
\]
Polymatroid extreme points

- Since \( w \in \mathbb{R}^E_+ \) is arbitrary, it may be that any \( e \in E \) is max (i.e., is such that \( w(e) > w(e') \) for \( e' \in E \setminus \{e\} \)).

- Thus, intuitively, any first vertex of the polytope away from the origin might be obtained by advancing along the corresponding axis.

- Recall, base polytope defined as the extreme face of \( P_f \). I.e.,

\[
B_f = P_f \cap \{ x \in \mathbb{R}^E_+ : x(E) = f(E) \} \quad (13.11)
\]

- Also, intuitively, we can continue advancing along the skeletal edges of the polytope to reach any other vertex, given the appropriate ordering. If we advance in all \( E \) dimensions, we’ll reach a vertex in \( B_f \), and if we advance only in some dimensions, we’ll reach a vertex in \( P_f \cap \{ x \in \mathbb{R}^E_+ : x(A) = 0 \text{ for some } A \} \).
Polymatroid extreme points

- Since $w \in \mathbb{R}^E_+$ is arbitrary, it may be that any $e \in E$ is max (i.e., is such that $w(e) > w(e')$ for $e' \in E \setminus \{e\}$).

- Thus, intuitively, any first vertex of the polytope away from the origin might be obtained by advancing along the corresponding axis.

- Recall, base polytope defined as the extreme face of $P_f$. I.e.,

$$B_f = P_f \cap \{x \in \mathbb{R}^E_+ : x(E) = f(E)\} \quad (13.11)$$

- Also, intuitively, we can continue advancing along the skeletal edges of the polytope to reach any other vertex, given the appropriate ordering. If we advance in all $E$ dimensions, we’ll reach a vertex in $B_f$, and if we advance only in some dimensions, we’ll reach a vertex in $P_f \cap \{x \in \mathbb{R}^E_+ : x(A) = 0$ for some $A\}$.

- We formalize this next:
Polymatroid extreme points

Given any arbitrary order of \( E = (e_1, e_2, \ldots, e_m) \), define \( E_i = (e_1, e_2, \ldots, e_i) \).
Polymatroid extreme points

- Given any arbitrary order of $E = (e_1, e_2, \ldots, e_m)$, define $E_i = (e_1, e_2, \ldots, e_i)$.
- As before, a vector $x$ is generated by $E_i$ using the greedy procedure as follows

\begin{align*}
x(e_1) &= f(E_1) = f(e_1) \\
x(e_j) &= f(E_j) - f(E_{j-1}) = f(e_j|E_{j-1}) \text{ for } 2 \leq j \leq i
\end{align*}
Polymatroid extreme points

- Given any arbitrary order of $E = (e_1, e_2, \ldots, e_m)$, define $E_i = (e_1, e_2, \ldots, e_i)$.
- As before, a vector $x$ is generated by $E_i$ using the greedy procedure as follows
  \[
  x(e_1) = f(E_1) = f(e_1) \quad (13.12)
  \]
  \[
  x(e_j) = f(E_j) - f(E_{j-1}) = f(e_j|E_{j-1}) \text{ for } 2 \leq j \leq i \quad (13.13)
  \]
- An extreme point of $P_f$ is a point that is not a convex combination of two other distinct points in $P_f$. Equivalently, an extreme point corresponds to setting certain inequalities ($|E|$ of them) in the specification of $P_f$ to be equalities, so that there is a unique single point solution.
Polymatroid extreme points

Theorem 13.5.1

For a given ordering $E = (e_1, \ldots, e_m)$ of $E$ and a given $E_i = (e_1, \ldots, e_i)$ and $x$ generated by $E_i$ using the greedy procedure ($x(e_i) = f(e_i|E_{i-1})$), then $x$ is an extreme point of $P_f$ when $f$ is submodular.
Polymatroid extreme points

**Theorem 13.5.1**

For a given ordering $E = (e_1, \ldots, e_m)$ of $E$ and a given $E_i = (e_1, \ldots, e_i)$ and $x$ generated by $E_i$ using the greedy procedure ($x(e_i) = f(e_i | E_{i-1})$), then $x$ is an extreme point of $P_f$ when $f$ is submodular.

**Proof.**

- We already saw that $x \in P_f$ (Theorem 12.4.1).
Polymatroid extreme points

Theorem 13.5.1

For a given ordering $E = (e_1, \ldots, e_m)$ of $E$ and a given $E_i = (e_1, \ldots, e_i)$ and $x$ generated by $E_i$ using the greedy procedure ($x(e_i) = f(e_i|E_{i-1})$), then $x$ is an extreme point of $P_f$ when $f$ is submodular.

Proof.

- We already saw that $x \in P_f$ (Theorem 12.4.1).
- To show that $x$ is an extreme point of $P_f$, note that it is the unique solution of the following system of equations

$$x(E_j) = f(E_j) \text{ for } 1 \leq j \leq i \leq m \quad (13.14)$$

$$x(e) = 0 \text{ for } e \in E \setminus E_i \quad (13.15)$$

There are $i \leq m$ equations and $i \leq m$ unknowns, and simple Gaussian elimination gives us back the $x$ constructed via the Greedy algorithm!!
Polymatroid extreme points

- As an example, we have $x(E_1) = x(e_1) = f(e_1)$
Polymatroid extreme points

- As an example, we have \( x(E_1) = x(e_1) = f(e_1) \)
- \( x(E_2) = x(e_1) + x(e_2) = f(e_1, e_2) \) so
  \[ x(e_2) = f(e_1, e_2) - x(e_1) = f(e_1, e_2) - f(e_1) = f(e_2 | e_1). \]
Polymatroid extreme points

As an example, we have $x(E_1) = x(e_1) = f(e_1)$

$x(E_2) = x(e_1) + x(e_2) = f(e_1, e_2)$ so

$x(e_2) = f(e_1, e_2) - x(e_1) = f(e_1, e_2) - f(e_1) = f(e_2|e_1)$.

$x(E_3) = x(e_1) + x(e_2) + x(e_3) = f(e_1, e_2, e_3)$ so $x(e_3) = f(e_1, e_2, e_3) - x(e_2) - x(e_1) = f(e_1, e_2, e_3) - f(e_1, e_2) = f(e_3|e_1, e_2)$
Polymatroid extreme points

- As an example, we have $x(E_1) = x(e_1) = f(e_1)$
- $x(E_2) = x(e_1) + x(e_2) = f(e_1, e_2)$ so
  $x(e_2) = f(e_1, e_2) - x(e_1) = f(e_1, e_2) - f(e_1) = f(e_2|e_1)$.
- $x(E_3) = x(e_1) + x(e_2) + x(e_3) = f(e_1, e_2, e_3)$ so $x(e_3) = f(e_1, e_2, e_3) - x(e_2) - x(e_1) = f(e_1, e_2, e_3) - f(e_1, e_2) = f(e_3|e_1, e_2)$
- And so on . . . , but we see that this is just Gaussian elimination.
Polymatroid extreme points

- As an example, we have \( x(E_1) = x(e_1) = f(e_1) \)
- \( x(E_2) = x(e_1) + x(e_2) = f(e_1, e_2) \) so 
  \[ x(e_2) = f(e_1, e_2) - x(e_1) = f(e_1, e_2) - f(e_1) = f(e_2 | e_1). \]
- \( x(E_3) = x(e_1) + x(e_2) + x(e_3) = f(e_1, e_2, e_3) \) so 
  \[ x(e_3) = f(e_1, e_2, e_3) - x(e_2) - x(e_1) = f(e_1, e_2, e_3) - f(e_1, e_2) = f(e_3 | e_1, e_2) \]
- And so on . . ., but we see that this is just Gaussian elimination.
- Also, since \( x \in P_f \), for each \( i \), we see that,

\[
\begin{align*}
x(E_j) &= f(E_j) \quad \text{for } 1 \leq j \leq i \quad (13.16) \\
x(A) &\leq f(A), \forall A \subseteq E \quad (13.17)
\end{align*}
\]
Polymatroid extreme points

As an example, we have

\[ x(E_1) = x(e_1) = f(e_1) \]

\[ x(E_2) = x(e_1) + x(e_2) = f(e_1, e_2) \text{ so } x(e_2) = f(e_1, e_2) - x(e_1) = f(e_1, e_2) - f(e_1) = f(e_2 | e_1). \]

\[ x(E_3) = x(e_1) + x(e_2) + x(e_3) = f(e_1, e_2, e_3) \text{ so } x(e_3) = f(e_1, e_2, e_3) - x(e_2) - x(e_1) = f(e_1, e_2, e_3) - f(e_1, e_2) = f(e_3 | e_1, e_2) \]

And so on . . . , but we see that this is just Gaussian elimination.

Also, since \( x \in P_f \), for each \( i \), we see that,

\[ x(E_j) = f(E_j) \text{ for } 1 \leq j \leq i \quad (13.16) \]

\[ x(A) \leq f(A), \forall A \subseteq E \quad (13.17) \]

Thus, the greedy procedure provides a modular function lower bound on \( f \) that is tight on all points \( E_i \) in the order. This can be useful in its own right, as it provides subgradients and subdifferential structure.
Polymatroid extreme points

some examples
Polymatroid extreme points

Moreover, we can show that

Corollary 13.5.2

If \( x \) is an extreme point of \( P_f \) and \( B \subseteq E \) is given such that
\[
supp(x) = \{ e \in E : x(e) \neq 0 \} \subseteq B \subseteq \bigcup \{ A : x(A) = f(A) \} = \text{sat}(x),
\]
then \( x \) is generated using greedy by some ordering of \( B \).
Polymatroid extreme points

Moreover, we can show that

**Corollary 13.5.2**

*If* $x$ *is an extreme point of* $P_f$ *and* $B \subseteq E$ *is given such that*

$$\text{supp}(x) = \{ e \in E : x(e) \neq 0 \} \subseteq B \subseteq \bigcup (A : x(A) = f(A)) = \text{sat}(x),$$

*then* $x$ *is generated using greedy by some ordering of* $B$.

*Note,* $\text{sat}(x) = \text{cl}(x) = \bigcup (A : x(A) = f(A))$ *is also called the closure of* $x$ *(recall that sets* $A$ *such that* $x(A) = f(A)$ *are called tight, and such sets are closed under union and intersection, as seen in Lecture 10, Theorem 12.3.2)*
Polymatroid extreme points

Moreover, we can show that

**Corollary 13.5.2**

*If* \( x \) *is an extreme point of* \( P_f \) *and* \( B \subseteq E \) *is given such that*

\[
\text{supp}(x) = \{ e \in E : x(e) \neq 0 \} \subseteq B \subseteq \bigcup (A : x(A) = f(A)) = \text{sat}(x),
\]

*then* \( x \) *is generated using greedy by some ordering of* \( B \).

*Note,* \( \text{sat}(x) = \text{cl}(x) = \bigcup (A : x(A) = f(A)) \) *is also called the closure of* \( x \) *(recall that sets* \( A \) *such that* \( x(A) = f(A) \) *are called tight, and such sets are closed under union and intersection, as seen in Lecture 10, Theorem 12.3.2)*

*Thus,* \( \text{cl}(x) \) *is a tight set.*
Polymatroid extreme points

Moreover, we can show that

Corollary 13.5.2

If $x$ is an extreme point of $P_f$ and $B \subseteq E$ is given such that

$$\text{supp}(x) = \{e \in E : x(e) \neq 0\} \subseteq B \subseteq \bigcup (A : x(A) = f(A)) = \text{sat}(x),$$

then $x$ is generated using greedy by some ordering of $B$.

Note, $\text{sat}(x) = \text{cl}(x) = \bigcup (A : x(A) = f(A))$ is also called the closure of $x$ (recall that sets $A$ such that $x(A) = f(A)$ are called tight, and such sets are closed under union and intersection, as seen in Lecture 10, Theorem 12.3.2)

Thus, $\text{cl}(x)$ is a tight set.

Also, $\text{supp}(x) = \{e \in E : x(e) \neq 0\}$ is called the support of $x$. 
Polymatroid extreme points

Moreover, we can show that

**Corollary 13.5.2**

If \( x \) is an extreme point of \( P_f \) and \( B \subseteq E \) is given such that
\[
\text{supp}(x) = \{e \in E : x(e) \neq 0\} \subseteq B \subseteq \bigcup (A : x(A) = f(A)) = \text{sat}(x),
\]
then \( x \) is generated using greedy by some ordering of \( B \).

- Note, \( \text{sat}(x) = \text{cl}(x) = \bigcup (A : x(A) = f(A)) \) is also called the closure of \( x \) (recall that sets \( A \) such that \( x(A) = f(A) \) are called tight, and such sets are closed under union and intersection, as seen in Lecture 10, Theorem 12.3.2)
- Thus, \( \text{cl}(x) \) is a tight set.
- Also, \( \text{supp}(x) = \{e \in E : x(e) \neq 0\} \) is called the support of \( x \).
- For arbitrary \( x \), \( \text{supp}(x) \) is not necessarily tight, but for an extreme point, \( \text{supp}(x) \) is.
Recall
\[ f(e|A) = f(A+e) - f(A) \]

Notice how submodularity, 
\[ f(e|B) \leq f(e|A) \text{ for } A \subseteq B, \]
defines the shape of the polytope.

In fact, we have strictness here
\[ f(e|B) < f(e|A) \text{ for } A \subset B. \]

Also, consider how the greedy algorithm proceeds along the edges of the polytope.
Recall 
\[ f(e|A) = f(A+e) - f(A) \]

Notice how submodularity,
\[ f(e|B) \leq f(e|A) \] for \( A \subseteq B \), defines the shape of the polytope.

In fact, we have strictness here
\[ f(e|B) < f(e|A) \] for \( A \subset B \).

Also, consider how the greedy algorithm proceeds along the edges of the polytope.
Intuition: why greedy works with polymatroids

Given \( w \), the goal is to find
\[
x = (x(e_1), x(e_2))
\]
that maximizes
\[
x^\top w = x(e_1)w(e_1) + x(e_2)w(e_2).
\]

If \( w(e_2) > w(e_1) \) the upper extreme point indicated maximizes \( x^\top w \) over \( x \in P_f^+ \).

If \( w(e_2) < w(e_1) \) the lower extreme point indicated maximizes \( x^\top w \) over \( x \in P_f^+ \).
Submodular maximization is quite useful.
Maximization of Submodular Functions

- Submodular maximization is quite useful.
- Applications: sensor placement, facility location, document summarization, or any kind of covering problem (choose a small set of elements that cover some domain as much as possible).
Maximization of Submodular Functions

- Submodular maximization is quite useful.
- Applications: sensor placement, facility location, document summarization, or any kind of covering problem (choose a small set of elements that cover some domain as much as possible).
- $f$ in this case is a model of dispersion, diversity, representativeness, or information.
Maximization of Submodular Functions

- Submodular maximization is quite useful.
- Applications: sensor placement, facility location, document summarization, or any kind of covering problem (choose a small set of elements that cover some domain as much as possible).
- $f$ in this case is a model of dispersion, diversity, representativeness, or information.
- For polymatroid function (or any monotone non-decreasing function), unconstrained maximization is trivial (take ground set).
Maximization of Submodular Functions

- Submodular maximization is quite useful.
- Applications: sensor placement, facility location, document summarization, or any kind of covering problem (choose a small set of elements that cover some domain as much as possible).
- \( f \) in this case is a model of dispersion, diversity, representativeness, or information.
- For polymatroid function (or any monotone non-decreasing function), unconstrained maximization is trivial (take ground set).
- Thus, when we do monotone submodular maximization we find the maximum under some constraint.
Maximization of Submodular Functions

- Submodular maximization is quite useful.
- Applications: sensor placement, facility location, document summarization, or any kind of covering problem (choose a small set of elements that cover some domain as much as possible).
- \( f \) in this case is a model of dispersion, diversity, representativeness, or information.
- For polymatroid function (or any monotone non-decreasing function), unconstrained maximization is trivial (take ground set).
- Thus, when we do monotone submodular maximization we find the maximum under some constraint.
- There is also a sort of dual problem that is often considered together with max, and those are minimum cover problems (to be defined).
The Set Cover Problem

- Let $E$ be a set and let $E_1, E_2, \ldots, E_m$ be a set of subsets.
The Set Cover Problem

- Let $E$ be a set and let $E_1, E_2, \ldots, E_m$ be a set of subsets.
- Let $V = \{1, 2, \ldots, m\}$ be the set of integers.
The Set Cover Problem

- Let $E$ be a set and let $E_1, E_2, \ldots, E_m$ be a set of subsets.
- Let $V = \{1, 2, \ldots, m\}$ be the set of integers.
- Define $f: 2^V \rightarrow \mathbb{Z}_+$ as $f(X) = |\bigcup_{v \in X} E_v|$
The Set Cover Problem

- Let $E$ be a set and let $E_1, E_2, \ldots, E_m$ be a set of subsets.
- Let $V = \{1, 2, \ldots, m\}$ be the set of integers.
- Define $f : 2^V \to \mathbb{Z}_+ \text{ as } f(X) = |\bigcup_{v \in X} E_v|$
- Then $f$ is the set cover function. As we say, $f$ is monotone submodular (a polymatroid).
The Set Cover Problem

- Let $E$ be a set and let $E_1, E_2, \ldots, E_m$ be a set of subsets.
- Let $V = \{1, 2, \ldots, m\}$ be the set of integers.
- Define $f : 2^V \to \mathbb{Z}_+$ as $f(X) = |\bigcup_{v \in X} E_v|$
- Then $f$ is the set cover function. As we say, $f$ is monotone submodular (a polymatroid).
- The set cover problem asks for the smallest subset $X$ of $V$ such that $f(X) = |E|$ (smallest subset of the subsets of $E$) where $E$ is still covered. I.e.,

$$\text{minimize} |X| \text{ subject to } f(X) \geq |E| \quad (13.18)$$
The Set Cover Problem

- Let $E$ be a set and let $E_1, E_2, \ldots, E_m$ be a set of subsets.
- Let $V = \{1, 2, \ldots, m\}$ be the set of integers.
- Define $f : 2^V \rightarrow \mathbb{Z}_+$ as $f(X) = |\bigcup_{v \in X} E_v|$
- Then $f$ is the set cover function. As we say, $f$ is monotone submodular (a polymatroid).
- The set cover problem asks for the smallest subset $X$ of $V$ such that $f(X) = |E|$ (smallest subset of the subsets of $E$) where $E$ is still covered. I.e.,

$$\text{minimize} \ |X| \ \text{subject to} \ f(X) \geq |E|$$

We might wish to use a more general modular function $m(X)$ rather than cardinality $|X|$.
The Set Cover Problem

- Let $E$ be a set and let $E_1, E_2, \ldots, E_m$ be a set of subsets.
- Let $V = \{1, 2, \ldots, m\}$ be the set of integers.
- Define $f : 2^V \to \mathbb{Z}_+ \text{ as } f(X) = |\bigcup_{v \in X} E_v|$
- Then $f$ is the set cover function. As we say, $f$ is monotone submodular (a polymatroid).
- The set cover problem asks for the smallest subset $X$ of $V$ such that $f(X) = |E|$ (smallest subset of the subsets of $E$) where $E$ is still covered. I.e.,

$$\text{minimize} |X| \text{ subject to } f(X) \geq |E| \quad (13.18)$$

- We might wish to use a more general modular function $m(X)$ rather than cardinality $|X|$.
- This problem is NP-hard, and Feige in 1998 showed that it cannot be approximated with a ratio better than $(1 - \epsilon) \log n$ unless NP is slightly superpolynomial ($n^{O(\log \log n)}$).
What About Non-monotone

- So even simple case of cardinality constrained submodular function maximization is NP-hard.
- This will be true of most submodular max (and related) problems.
- Hence, the only hope is approximation algorithms. Question is, what is the tradeoff between running time and approximation quality, and is it possible to get tight bounds (i.e., an algorithm that achieves an approximation ratio, and a proof that one can’t do better than that unless some extremely unlike event were to be true, such as P=NP).
Let $E$ be a set and let $E_1, E_2, \ldots, E_m$ be a set of subsets.
The Max $k$-Cover Problem

- Let $E$ be a set and let $E_1, E_2, \ldots, E_m$ be a set of subsets.
- Let $V = \{1, 2, \ldots, m\}$ be the set of integers.
The Max \( k \)-Cover Problem

- Let \( E \) be a set and let \( E_1, E_2, \ldots, E_m \) be a set of subsets.
- Let \( V = \{1, 2, \ldots, m\} \) be the set of integers.
- Define \( f : 2^V \rightarrow \mathbb{Z}_+ \) as \( f(X) = |\bigcup_{v \in V} E_v| \)
The Max \( k \)-Cover Problem

- Let \( E \) be a set and let \( E_1, E_2, \ldots, E_m \) be a set of subsets.
- Let \( V = \{1, 2, \ldots, m\} \) be the set of integers.
- Define \( f : 2^V \to \mathbb{Z}_+ \) as \( f(X) = |\bigcup_{v \in V} E_v| \)
- Then \( f \) is the set cover function. As we saw, \( f \) is monotone submodular (a polymatroid).
The Max $k$-Cover Problem

- Let $E$ be a set and let $E_1, E_2, \ldots, E_m$ be a set of subsets.
- Let $V = \{1, 2, \ldots, m\}$ be the set of integers.
- Define $f : 2^V \to \mathbb{Z}_+$ as $f(X) = |\bigcup_{v \in V} E_v|$.
- Then $f$ is the set cover function. As we saw, $f$ is monotone submodular (a polymatroid).
- The max $k$ cover problem asks, given a $k$, what sized $k$ set of sets $X$ can we choose that covers the most? I.e., that maximizes $f(X)$ as in:

$$\max f(X) \text{ subject to } |X| \leq k$$ (13.19)
The Max $k$-Cover Problem

- Let $E$ be a set and let $E_1, E_2, \ldots, E_m$ be a set of subsets.
- Let $V = \{1, 2, \ldots, m\}$ be the set of integers.
- Define $f : 2^V \rightarrow \mathbb{Z}_+$ as $f(X) = |\bigcup_{v \in V} E_v|$
- Then $f$ is the set cover function. As we saw, $f$ is monotone submodular (a polymatroid).
- The max $k$ cover problem asks, given a $k$, what sized $k$ set of sets $X$ can we choose that covers the most? I.e., that maximizes $f(X)$ as in:

$$\max f(X) \text{ subject to } |X| \leq k \quad (13.19)$$

- This problem is NP-hard, and Feige in 1998 showed that it cannot be approximated with a ratio better than $(1 - 1/e)$. 
Recall facility location function $f(A) = \sum_{v \in V} \max_{a \in A} s_{a,v}$ where $s_{a,v}$ is the similarity between $a$ and $v$. Alternatively, we can think of this as a representativeness matrix, where $s_{a,v}$ is seen as how how good $a$ is as acting as a representative for $v$ (which might not be the same as $s_{v,a}$).
Cardinality Constrained Max. of Facility Location

- Recall facility location function $f(A) = \sum_{v \in V} \max_{a \in A} s_{a,v}$ where $s_{a,v}$ is the similarity between $a$ and $v$. Alternatively, we can think of this as a representativeness matrix, where $s_{a,v}$ is seen as how good $a$ is as acting as a representative for $v$ (which might not be the same as $s_{v,a}$).

Example:
Cardinality Constrained Max. of Facility Location

• Recall facility location function $f(A) = \sum_{v \in V} \max_{a \in A} s_{a,v}$ where $s_{a,v}$ is the similarity between $a$ and $v$. Alternatively, we can think of this as a representativeness matrix, where $s_{a,v}$ is seen as how good $a$ is as acting as a representative for $v$ (which might not be the same as $s_{v,a}$). Example:

![Graphical representation of facility location function with V=1000, Gauscmps=5, concn=0.7, k_RD=10]
Recall facility location function $f(A) = \sum_{v \in V} \max_{a \in A} s_{a,v}$ where $s_{a,v}$ is the similarity between $a$ and $v$. Alternatively, we can think of this as a representativeness matrix, where $s_{a,v}$ is seen as how good $a$ is as acting as a representative for $v$ (which might not be the same as $s_{v,a}$).

Example:
Cardinality Constrained Max. of Facility Location

- Recall facility location function $f(A) = \sum_{v \in V} \max_{a \in A} s_{a,v}$ where $s_{a,v}$ is the similarity between $a$ and $v$. Alternatively, we can think of this as a representativeness matrix, where $s_{a,v}$ is seen as how good $a$ is as acting as a representative for $v$ (which might not be the same as $s_{v,a}$).

- Middle example is estimate of $\max_{A \subseteq V: |A| \leq k} f(A)$, right is uniformly-at-random randomly chosen set of size $k$, for $k = 10$. 

![Graphs showing facility location function examples]
Now we are given an arbitrary polymatroid function $f$. 
Cardinality Constrained Max. of Polymatroid Functions

- Now we are given an arbitrary polymatroid function $f$.
- Given $k$, goal is: find $A^* \in \text{argmax} \{ f(A) : |A| \leq k \}$
Now we are given an arbitrary polymatroid function $f$.

Given $k$, goal is: find $A^* \in \text{argmax} \{ f(A) : |A| \leq k \}$

w.l.o.g., we can find $A^* \in \text{argmax} \{ f(A) : |A| = k \}$
Now we are given an arbitrary polymatroid function $f$.

Given $k$, goal is: find $A^* \in \arg\max \{ f(A) : |A| \leq k \}$

w.l.o.g., we can find $A^* \in \arg\max \{ f(A) : |A| = k \}$

An important result by Nemhauser et. al. (1978) states that for normalized ($f(\emptyset) = 0$) monotone submodular functions (i.e., polymatroids) can be approximately maximized using a simple greedy algorithm.
Now we are given an arbitrary polymatroid function $f$.

Given $k$, goal is: find $A^* \in \text{argmax} \{ f(A) : |A| \leq k \}$

w.l.o.g., we can find $A^* \in \text{argmax} \{ f(A) : |A| = k \}$

An important result by Nemhauser et. al. (1978) states that for normalized ($f(\emptyset) = 0$) monotone submodular functions (i.e., polymatroids) can be approximately maximized using a simple greedy algorithm.

Starting with $S_0 = \emptyset$, we repeat the following greedy step for $i = 0 \ldots (k - 1)$:

$$S_{i+1} = S_i \cup \left\{ \text{argmax}_{v \in V \setminus S_i} f(S_i \cup \{v\}) \right\}$$

(13.20)
A bit more precisely:

**Algorithm 1: The Greedy Algorithm**

1. Set $S_0 \leftarrow \emptyset$;
2. for $i \leftarrow 0 \ldots |E| - 1$ do
3.   Choose $v_i$ as follows:
       $v_i \in \text{argmax}_{v \in V \setminus S_i} f(\{v\} | S_i) = \text{argmax}_{v \in V \setminus S_i} f(S_i \cup \{v\})$;
4.   Set $S_{i+1} \leftarrow S_i \cup \{v_i\}$;
This algorithm has a guarantee
This algorithm has a guarantee

**Theorem 13.6.1**

Given a polymatroid function $f$, the above greedy algorithm returns sets $S_i$ such that for each $i$ we have $f(S_i) \geq (1 - 1/e) \max_{|S| \leq i} f(S)$. 
This algorithm has a guarantee

**Theorem 13.6.1**

Given a polymatroid function $f$, the above greedy algorithm returns sets $S_i$ such that for each $i$ we have $f(S_i) \geq (1 - 1/e) \max_{|S| \leq i} f(S)$.

To approximately find $A^* \in \arg\max \{f(A) : |A| \leq k\}$, we repeat the greedy step until $k = i + 1$ in Algorithm 4:
This algorithm has a guarantee

Theorem 13.6.1

Given a polymatroid function $f$, the above greedy algorithm returns sets $S_i$ such that for each $i$ we have $f(S_i) \geq (1 - 1/e) \max_{|S| \leq i} f(S)$.

To approximately find $A^* \in \arg\max \{f(A) : |A| \leq k\}$, we repeat the greedy step until $k = i + 1$ in Algorithm 4:

- Again, since this generalizes max $k$-cover, Feige (1998) showed that this can’t be improved. Unless $P = NP$, no polynomial time algorithm can do better than $(1 - 1/e + \epsilon)$ for any $\epsilon > 0$. 
The Greedy Algorithm: $1 - 1/e$ intuition.

- At step $i < k$, greedy chooses $v_i$ to maximize $f(v|S_i)$. 

\[ \text{Equation (13.30) will show that Equation (13.21)} \]
\[ \Rightarrow \]
\[ \text{OPT} - f(S_{i+1}) \leq (1 - 1/k) (\text{OPT} - f(S_i)) \]
\[ \Rightarrow \]
\[ \text{OPT} \leq 1/e \text{OPT} \]
\[ \Rightarrow \]
\[ \text{OPT} \left(1 - 1/e\right) \leq f(S_k) \]
The Greedy Algorithm: \( 1 - \frac{1}{e} \) intuition.

- At step \( i < k \), greedy chooses \( v_i \) to maximize \( f(v|S_i) \).
- Let \( S^* \) be optimal solution (of size \( k \)) and \( \text{OPT} = f(S^*) \).
The Greedy Algorithm: $1 - 1/e$ intuition.

- At step $i < k$, greedy chooses $v_i$ to maximize $f(v|S_i)$.
- Let $S^*$ be optimal solution (of size $k$) and $OPT = f(S^*)$. By submodularity, we will show:

$$\exists v \in V \setminus S_i : f(v|S_i) = f(S_i + v|S_i) \geq \frac{1}{k}(OPT - f(S_i)) \quad (13.21)$$
The Greedy Algorithm: $1 - 1/e$ intuition.

- At step $i < k$, greedy chooses $v_i$ to maximize $f(v|S_i)$.
- Let $S^*$ be optimal solution (of size $k$) and $OPT = f(S^*)$. By submodularity, we will show:

$$
\exists v \in V \setminus S_i : f(v|S_i) = f(S_i + v|S_i) \geq \frac{1}{k}(OPT - f(S_i))
$$

(13.21)

Equation (13.30) will show that Equation (13.21) $\Rightarrow$:

$$
OPT - f(S_i+1) \\
\leq (1 - 1/k)(OPT - f(S_i)) \\
\Rightarrow OPT - f(S_k) \\
\leq (1 - 1/k)^kOPT \\
\leq 1/eOPT \\
\Rightarrow OPT(1 - 1/e) \leq f(S_k)
$$
The Greedy Algorithm: $1 - 1/e$ intuition.

- At step $i < k$, greedy chooses $v_i$ to maximize $f(v|S_i)$.
- Let $S^*$ be optimal solution (of size $k^*$) and $OPT = f(S^*)$. By submodularity, we will show:

$$\exists v \in V \setminus S_i : f(v|S_i) = f(S_i + v|S_i) \geq \frac{1}{k}(OPT - f(S_i)) \quad (13.21)$$

Equation (13.30) will show that Equation (13.21) $\Rightarrow$:

$$OPT - f(S_{i+1}) \leq (1 - 1/k)(OPT - f(S_i))$$

$\Rightarrow$ $OPT - f(S_k) \leq (1 - 1/k)^k OPT$

$\leq 1/e OPT$

$\Rightarrow$ $OPT(1 - 1/e) \leq f(S_k)$
Theorem 13.6.2 (Nemhauser et al. 1978)

Given non-negative monotone submodular function $f : 2^V \to \mathbb{R}_+$, define $\{S_i\}_{i \geq 0}$ to be the chain formed by the greedy algorithm (Eqn. (13.20)). Then for all $k, \ell \in \mathbb{Z}_{++}$, we have:

$$f(S_\ell) \geq (1 - e^{-\ell/k}) \max_{S : |S| \leq k} f(S) \quad (13.22)$$

and in particular, for $\ell = k$, we have $f(S_k) \geq (1 - 1/e) \max_{S : |S| \leq k} f(S)$. 
Theorem 13.6.2 (Nemhauser et al. 1978)

Given non-negative monotone submodular function $f : 2^V \rightarrow \mathbb{R}_+$, define \( \{S_i\}_{i \geq 0} \) to be the chain formed by the greedy algorithm (Eqn. (13.20)). Then for all $k, \ell \in \mathbb{Z}^+$, we have:

$$f(S_\ell) \geq (1 - e^{-\ell/k}) \max_{S : |S| \leq k} f(S) \quad (13.22)$$

and in particular, for $\ell = k$, we have $f(S_k) \geq (1 - 1/e) \max_{S : |S| \leq k} f(S)$.

- $k$ is size of optimal set, i.e., $\text{OPT} = f(S^*)$ with $|S^*| = k$. 

Theorem 13.6.2 (Nemhauser et al. 1978)

Given non-negative monotone submodular function $f : 2^V \rightarrow \mathbb{R}_+$, define $\{S_i\}_{i \geq 0}$ to be the chain formed by the greedy algorithm (Eqn. (13.20)). Then for all $k, \ell \in \mathbb{Z}_{++}$, we have:

$$f(S_\ell) \geq (1 - e^{-\ell/k}) \max_{S:|S|\leq k} f(S) \tag{13.22}$$

and in particular, for $\ell = k$, we have $f(S_k) \geq (1 - 1/e) \max_{S:|S|\leq k} f(S)$.

- $k$ is size of optimal set, i.e., $\text{OPT} = f(S^*)$ with $|S^*| = k$
- $\ell$ is size of set we are choosing (i.e., we choose $S_\ell$ from greedy chain).
Theorem 13.6.2 (Nemhauser et al. 1978)

Given non-negative monotone submodular function \( f : 2^V \rightarrow \mathbb{R}_+ \), define \( \{S_i\}_{i \geq 0} \) to be the chain formed by the greedy algorithm (Eqn. (13.20)). Then for all \( k, \ell \in \mathbb{Z}_{++} \), we have:

\[
f(S_\ell) \geq (1 - e^{-\ell/k}) \max_{S:|S| \leq k} f(S) \tag{13.22}
\]

and in particular, for \( \ell = k \), we have

\[
f(S_k) \geq (1 - 1/e) \max_{S:|S| \leq k} f(S).
\]

- \( k \) is size of optimal set, i.e., \( \text{OPT} = f(S^*) \) with \( |S^*| = k \).
- \( \ell \) is size of set we are choosing (i.e., we choose \( S_\ell \) from greedy chain).
- Bound is how well does \( S_\ell \) (of size \( \ell \)) do relative to \( S^* \), the optimal set of size \( k \).
Theorem 13.6.2 (Nemhauser et al. 1978)

Given non-negative monotone submodular function \( f : 2^V \rightarrow \mathbb{R}_+ \), define \( \{S_i\}_{i \geq 0} \) to be the chain formed by the greedy algorithm (Eqn. (13.20)). Then for all \( k, \ell \in \mathbb{Z}_{++} \), we have:

\[
f(S_{\ell}) \geq (1 - e^{-\ell/k}) \max_{S:|S| \leq k} f(S) \tag{13.22}
\]

and in particular, for \( \ell = k \), we have \( f(S_k) \geq (1 - 1/e) \max_{S:|S| \leq k} f(S) \).

- \( k \) is size of optimal set, i.e., \( \text{OPT} = f(S^*) \) with \( |S^*| = k \)
- \( \ell \) is size of set we are choosing (i.e., we choose \( S_\ell \) from greedy chain).
- Bound is how well does \( S_\ell \) (of size \( \ell \)) do relative to \( S^* \), the optimal set of size \( k \).
- Intuitively, bound should get worse when \( \ell < k \) and get better when \( \ell > k \).
Proof of Theorem 13.6.2.

Fix ℓ (number of items greedy will choose) and k (size of optimal set to compare against).

Set $S^* \in \arg\max \{ f(S) : |S| \leq k \}$ w.l.o.g. assume $|S^*| = k$.

Order $S^* = (v^*_1, v^*_2, ..., v^*_k)$ arbitrarily.

Let $S_i = (v_1, v_2, ..., v_i)$ be the greedy order chain chosen by the algorithm, for $i \in \{1, 2, ..., \ell\}$.

Then the following inequalities (on the next slide) follow:
Proof of Theorem 13.6.2.

- Fix \( \ell \) (number of items greedy will chose) and \( k \) (size of optimal set to compare against).
Proof of Theorem 13.6.2.

1. Fix \( \ell \) (number of items greedy will chose) and \( k \) (size of optimal set to compare against).
2. Set \( S^* \in \text{argmax} \{ f(S) : |S| \leq k \} \)
Proof of Theorem 13.6.2.

- Fix $\ell$ (number of items greedy will chose) and $k$ (size of optimal set to compare against).
- Set $S^* \in \arg\max \{f(S) : |S| \leq k\}$
- w.l.o.g. assume $|S^*| = k$. 

...
Proof of Theorem 13.6.2.

- Fix \( \ell \) (number of items greedy will chose) and \( k \) (size of optimal set to compare against).
- Set \( S^{*} \in \text{argmax} \{ f(S) : |S| \leq k \} \)
- w.l.o.g. assume \( |S^{*}| = k \).
- Order \( S^{*} = (v_1^{*}, v_2^{*}, \ldots, v_k^{*}) \) arbitrarily.
Proof of Theorem 13.6.2.

- Fix \( \ell \) (number of items greedy will chose) and \( k \) (size of optimal set to compare against).
- Set \( S^* \in \text{argmax}\ \{f(S) : |S| \leq k\} \)
- w.l.o.g. assume \( |S^*| = k \).
- Order \( S^* = (v_1^*, v_2^*, \ldots, v_k^*) \) arbitrarily.
- Let \( S_i = (v_1, v_2, \ldots, v_i) \) be the greedy order chain chosen by the algorithm, for \( i \in \{1, 2, \ldots, \ell\} \).
Proof of Theorem 13.6.2.

- Fix \( \ell \) (number of items greedy will chose) and \( k \) (size of optimal set to compare against).
- Set \( S^* \in \arg\max \{ f(S) : |S| \leq k \} \)
- w.l.o.g. assume \( |S^*| = k \).
- Order \( S^* = (v_1^*, v_2^*, \ldots, v_k^*) \) arbitrarily.
- Let \( S_i = (v_1, v_2, \ldots, v_i) \) be the greedy order chain chosen by the algorithm, for \( i \in \{1, 2, \ldots, \ell\} \).
- Then the following inequalities (on the next slide) follow:
Cardinality Constrained Polymatroid Max Theorem

...proof of Theorem 13.6.2 cont.

For all \( i < \ell \), we have

\[
\begin{align*}
    f(S^*) & \leq f(S^* \cup S_i) = f(S_i) + f(S^* \mid S_i) \quad (13.23) \\
    & \leq f(S_i) + k \sum_{j=1}^{v^*_{i-1}} f(v^*_{j} \mid S_i \cup \{v^*_1, v^*_2, \ldots, v^*_j-1\}) \quad (13.24) \\
    & \leq f(S_i) + \sum_{v \in S^*} f(v \mid S_i) \quad (13.25) \\
    & \leq f(S_i) + \sum_{v \in S^*} f(S_i+1 \mid S_i) = f(S_i) + kf(S_i+1 \mid S_i) \quad (13.26)
\end{align*}
\]

Therefore, we have Equation 13.21, i.e.,

\[
\begin{align*}
    f(S^*) - f(S_i) & \leq kf(S_i+1 \mid S_i) = k(f(S_i+1) - f(S_i)) \quad (13.28)
\end{align*}
\]
...proof of Theorem 13.6.2 cont.

- For all $i < \ell$, we have

$$f(S^*)$$
Proof of Theorem 13.6.2 cont.

For all $i < \ell$, we have

$$f(S^*) \leq f(S^* \cup S_i)$$
For all $i < \ell$, we have
\[ f(S^*) \leq f(S^* \cup S_i) = f(S_i) + f(S^*|S_i) \]  
(13.23)
Cardinality Constrained Polymatroid Max Theorem

...proof of Theorem 13.6.2 cont.

For all $i < \ell$, we have

$$f(S^*) \leq f(S^* \cup S_i) = f(S_i) + f(S^*|S_i)$$  \hspace{1cm} (13.23)

$$= f(S_i) + \sum_{j=1}^{k} f(v_j^*|S_i \cup \{v_1^*, v_2^*, \ldots, v_{j-1}^*\})$$  \hspace{1cm} (13.24)
For all \( i < \ell \), we have
\[
f(S^*) \leq f(S^* \cup S_i) = f(S_i) + f(S^*|S_i) \quad (13.23)
\]
\[
= f(S_i) + \sum_{j=1}^{k} f(v_j^*|S_i \cup \{v_1^*, v_2^*, \ldots, v_{j-1}^*\}) \quad (13.24)
\]
\[
\leq f(S_i) + \sum_{v \in S^*} f(v|S_i) \quad (13.25)
\]
For all \( i < \ell \), we have

\[
f(S^*) \leq f(S^* \cup S_i) = f(S_i) + f(S^* \mid S_i)
\] (13.23)

\[
= f(S_i) + \sum_{j=1}^{k} f(v_j^* \mid S_i \cup \{v_1^*, v_2^*, \ldots, v_{j-1}^*\})
\] (13.24)

\[
\leq f(S_i) + \sum_{v \in S^*} f(v \mid S_i)
\] (13.25)

\[
\leq f(S_i) + \sum_{v \in S^*} f(v_{i+1} \mid S_i)
\]
For all \( i < \ell \), we have

\[
f(S^*) \leq f(S^* \cup S_i) = f(S_i) + f(S^* | S_i)
\]

\[
= f(S_i) + \sum_{j=1}^{k} f(v_j^* | S_i \cup \{v_1^*, v_2^*, \ldots, v_{j-1}^*\})
\]

\[
\leq f(S_i) + \sum_{v \in S^*} f(v | S_i)
\]

\[
\leq f(S_i) + \sum_{v \in S^*} f(v_{i+1} | S_i) = f(S_i) + \sum_{v \in S^*} f(S_i+1 | S_i)
\]
Cardinality Constrained Polymatroid Max Theorem

...proof of Theorem 13.6.2 cont.

- For all $i < \ell$, we have
  
  \[ f(S^*) \leq f(S^* \cup S_i) = f(S_i) + f(S^*|S_i) \]  
  \[ = f(S_i) + \sum_{j=1}^{k} f(v_j^*|S_i \cup \{v_1^*, v_2^*, \ldots, v_{j-1}^*\}) \]  
  \[ \leq f(S_i) + \sum_{v \in S^*} f(v|S_i) \]  
  \[ \leq f(S_i) + \sum_{v \in S^*} f(v_{i+1}|S_i) = f(S_i) + \sum_{v \in S^*} f(S_{i+1}|S_i) \]  
  \[ = f(S_i) + kf(S_{i+1}|S_i) \]  
  
  (13.23)
  (13.24)
  (13.25)
  (13.26)
  (13.27)
...proof of Theorem 13.6.2 cont.

For all \( i < \ell \), we have

\[
f(S^*) \leq f(S^* \cup S_i) = f(S_i) + f(S^* | S_i)
\]

(13.23)

\[
f(S_i) + \sum_{j=1}^{k} f(v_j^* | S_i \cup \{v_1^*, v_2^*, \ldots, v_{j-1}^*\})
\]

(13.24)

\[
\leq f(S_i) + \sum_{v \in S^*} f(v | S_i)
\]

(13.25)

\[
\leq f(S_i) + \sum_{v \in S^*} f(v_{i+1} | S_i) = f(S_i) + \sum_{v \in S^*} f(S_{i+1} | S_i)
\]

(13.26)

\[
f(S_i) + k f(S_{i+1} | S_i)
\]

(13.27)

Therefore, we have Equation 13.21, i.e.,

\[
f(S^*) - f(S_i) \leq k f(S_{i+1} | S_i) = k(f(S_{i+1}) - f(S_i))
\]

(13.28)
Define \( \delta_i \equiv f(S^*) - f(S_i) \), so \( \delta_i - \delta_{i+1} = f(S_{i+1}) - f(S_i) \), giving

\[
\delta_i \leq k(\delta_i - \delta_{i+1}) \quad (13.29)
\]

or

\[
\delta_{i+1} \leq (1 - \frac{1}{k})\delta_i \quad (13.30)
\]

The relationship between \( \delta_0 \) and \( \delta_\ell \) is then

\[
\delta_\ell \leq (1 - \frac{1}{k})\ell \delta_0 \quad (13.31)
\]

Now, \( \delta_0 = f(S^*) - f(\emptyset) \leq f(S^*) \) since \( f \geq 0 \).

Also, by variational bound \( 1 - x \leq e^{-x} \) for \( x \in \mathbb{R} \), we have

\[
\delta_\ell \leq (1 - \frac{1}{k})\ell \delta_0 \leq e^{-\ell/k} f(S^*) \quad (13.32)
\]

... proof of Theorem 13.6.2 cont.
Define gap \( \delta_i \triangleq f(S^*) - f(S_i) \), so \( \delta_i - \delta_{i+1} = f(S_{i+1}) - f(S_i) \),
... proof of Theorem 13.6.2 cont.

Define gap \( \delta_i \triangleq f(S^*) - f(S_i) \), so \( \delta_i - \delta_{i+1} = f(S_{i+1}) - f(S_i) \), giving

\[
\delta_i \leq k(\delta_i - \delta_{i+1}) \tag{13.29}
\]

or
Cardinality Constrained Polymatroid Max Theorem

... proof of Theorem 13.6.2 cont.

- Define gap \( \delta_i \triangleq f(S^*) - f(S_i) \), so \( \delta_i - \delta_{i+1} = f(S_{i+1}) - f(S_i) \), giving
  \[
  \delta_i \leq k(\delta_i - \delta_{i+1}) \tag{13.29}
  \]
  or
  \[
  \delta_{i+1} \leq (1 - \frac{1}{k})\delta_i \tag{13.30}
  \]
Cardinality Constrained Polymatroid Max Theorem

...proof of Theorem 13.6.2 cont.

- Define gap \( \delta_i \triangleq f(S^*) - f(S_i) \), so \( \delta_i - \delta_{i+1} = f(S_{i+1}) - f(S_i) \), giving
  \[
  \delta_i \leq k(\delta_i - \delta_{i+1}) \tag{13.29}
  \]
  or
  \[
  \delta_{i+1} \leq (1 - \frac{1}{k})\delta_i \tag{13.30}
  \]
- The relationship between \( \delta_0 \) and \( \delta_\ell \) is then
  \[
  \delta_\ell \leq (1 - \frac{1}{k})^\ell \delta_0 \tag{13.31}
  \]
... proof of Theorem 13.6.2 cont.

- Define gap $\delta_i \triangleq f(S^*) - f(S_i)$, so $\delta_i - \delta_{i+1} = f(S_{i+1}) - f(S_i)$, giving

$$\delta_i \leq k(\delta_i - \delta_{i+1})$$

(13.29)

or

$$\delta_{i+1} \leq (1 - \frac{1}{k})\delta_i$$

(13.30)

- The relationship between $\delta_0$ and $\delta_\ell$ is then

$$\delta_{\ell} \leq (1 - \frac{1}{k})^\ell \delta_0$$

(13.31)

- Now, $\delta_0 = f(S^*) - f(\emptyset) \leq f(S^*)$ since $f \geq 0$. 

...
Cardinality Constrained Polymatroid Max Theorem

... proof of Theorem 13.6.2 cont.

- Define gap $\delta_i \triangleq f(S^*) - f(S_i)$, so $\delta_i - \delta_{i+1} = f(S_{i+1}) - f(S_i)$, giving
  \[ \delta_i \leq k(\delta_i - \delta_{i+1}) \]  
  (13.29)

  or
  \[ \delta_{i+1} \leq (1 - \frac{1}{k})\delta_i \]  
  (13.30)

- The relationship between $\delta_0$ and $\delta_\ell$ is then
  \[ \delta_\ell \leq (1 - \frac{1}{k})^\ell \delta_0 \]  
  (13.31)

- Now, $\delta_0 = f(S^*) - f(\emptyset) \leq f(S^*)$ since $f \geq 0$.

- Also, by variational bound $1 - x \leq e^{-x}$ for $x \in \mathbb{R}$, we have
  \[ \delta_\ell \leq (1 - \frac{1}{k})^\ell \delta_0 \leq e^{-\ell/k} f(S^*) \]  
  (13.32)
When we identify \( \delta_{\ell} = f(S^*) - f(S_{\ell}) \), a bit of rearranging then gives:

\[
f(S_{\ell}) \geq (1 - e^{-\ell/k}) f(S^*) \tag{13.33}
\]
Cardinality Constrained Polymatroid Max Theorem

...proof of Theorem 13.6.2 cont.

- When we identify $\delta \ell = f(S^*) - f(S_\ell)$, a bit of rearranging then gives:

\[
f(S_\ell) \geq (1 - e^{-\ell/k}) f(S^*) \tag{13.33}
\]

- With $\ell = k$, when picking $k$ items, greedy gets $(1 - 1/e) \approx 0.6321$ bound. This means that if $S_k$ is greedy solution of size $k$, and $S^*$ is an optimal solution of size $k$, $f(S_k) \geq (1 - 1/e) f(S^*) \approx 0.6321 f(S^*)$. 
Cardinality Constrained Polymatroid Max Theorem

... proof of Theorem 13.6.2 cont.

- When we identify $\delta_\ell = f(S^*) - f(S_\ell)$, a bit of rearranging then gives:

$$f(S_\ell) \geq (1 - e^{-\ell/k}) f(S^*)$$  \hfill (13.33)

- With $\ell = k$, when picking $k$ items, greedy gets $(1 - 1/e) \approx 0.6321$ bound. This means that if $S_k$ is greedy solution of size $k$, and $S^*$ is an optimal solution of size $k$, $f(S_k) \geq (1 - 1/e) f(S^*) \approx 0.6321 f(S^*)$.

- What if we want to guarantee a solution no worse than $0.95 f(S^*)$ where $|S^*| = k$?
When we identify $\delta_l = f(S^*) - f(S_l)$, a bit of rearranging then gives:

$$f(S_l) \geq (1 - e^{-\ell/k}) f(S^*) \quad (13.33)$$

With $\ell = k$, when picking $k$ items, greedy gets $(1 - 1/e) \approx 0.6321$ bound. This means that if $S_k$ is greedy solution of size $k$, and $S^*$ is an optimal solution of size $k$, $f(S_k) \geq (1 - 1/e) f(S^*) \approx 0.6321 f(S^*)$.

What if we want to guarantee a solution no worse than $0.95 f(S^*)$ where $|S^*| = k$? Set $0.95 = (1 - e^{-\ell/k})$, which gives

$$\ell = \lceil -k \ln(1 - 0.95) \rceil = 4k.$$
When we identify $\delta_l = f(S^*) - f(S_\ell)$, a bit of rearranging then gives:

$$f(S_\ell) \geq (1 - e^{-\ell/k}) f(S^*) \quad (13.33)$$

With $\ell = k$, when picking $k$ items, greedy gets $(1 - 1/e) \approx 0.6321$ bound. This means that if $S_k$ is greedy solution of size $k$, and $S^*$ is an optimal solution of size $k$, $f(S_k) \geq (1 - 1/e) f(S^*) \approx 0.6321 f(S^*)$.

What if we want to guarantee a solution no worse than $.95 f(S^*)$ where $|S^*| = k$? Set $0.95 = (1 - e^{-\ell/k})$, which gives $\ell = \lceil -k \ln(1 - 0.95) \rceil = 4k$. And $\lceil -\ln(1 - 0.999) \rceil = 7$. 
... proof of Theorem 13.6.2 cont.

- When we identify $\delta = f(S^*) - f(S_\ell)$, a bit of rearranging then gives:
  \[ f(S_\ell) \geq (1 - e^{-\ell/k}) f(S^*) \]  
  (13.33)

- With $\ell = k$, when picking $k$ items, greedy gets $(1 - 1/e) \approx 0.6321$ bound. This means that if $S_k$ is greedy solution of size $k$, and $S^*$ is an optimal solution of size $k$, $f(S_k) \geq (1 - 1/e) f(S^*) \approx 0.6321 f(S^*)$.

- What if we want to guarantee a solution no worse than $0.95 f(S^*)$ where $|S^*| = k$? Set $0.95 = (1 - e^{-\ell/k})$, which gives $\ell = -k \ln(1 - 0.95) = 4k$. And $\lceil -\ln(1 - 0.999) \rceil = 7$.

- So solution, in the worst case, quickly gets very good. Typical/practical case is much better.
Greedy running time

- Greedy computes a new maximum \( n = |V| \) times, and each maximum computation requires \( O(n) \) comparisons, leading to \( O(n^2) \) computation for greedy.
Greedy running time

- Greedy computes a new maximum $n = |V|$ times, and each maximum computation requires $O(n)$ comparisons, leading to $O(n^2)$ computation for greedy.
- This is the best we can do for arbitrary functions, but $O(n^2)$ is not practical to some.
Greedy running time

- Greedy computes a new maximum $n = |V|$ times, and each maximum computation requires $O(n)$ comparisons, leading to $O(n^2)$ computation for greedy.
- This is the best we can do for arbitrary functions, but $O(n^2)$ is not practical to some.
- Greedy can be made much faster in practice by a simple strategy made possible, once again, via the use of submodularity.
Greedy running time

- Greedy computes a new maximum $n = |V|$ times, and each maximum computation requires $O(n)$ comparisons, leading to $O(n^2)$ computation for greedy.
- This is the best we can do for arbitrary functions, but $O(n^2)$ is not practical to some.
- Greedy can be made much faster in practice by a simple strategy made possible, once again, via the use of submodularity.
- This is called Minoux’s 1977 Accelerated Greedy strategy (and has been rediscovered a few times, e.g., “Lazy greedy”), and runs much faster while still producing same answer.
Greedy running time

- Greedy computes a new maximum $n = |V|$ times, and each maximum computation requires $O(n)$ comparisons, leading to $O(n^2)$ computation for greedy.
- This is the best we can do for arbitrary functions, but $O(n^2)$ is not practical to some.
- Greedy can be made much faster in practice by a simple strategy made possible, once again, via the use of submodularity.
- This is called Minoux’s 1977 Accelerated Greedy strategy (and has been rediscovered a few times, e.g., “Lazy greedy”), and runs much faster while still producing same answer.
- We describe it next:
Minoux’s Accelerated Greedy for Submodular Functions

At stage $i$ in the algorithm, we have a set of gains $f(v|S_i)$ for all $v \notin S_i$. Store these values $\alpha_v \leftarrow f(v|S_i)$ in sorted priority queue.
At stage $i$ in the algorithm, we have a set of gains $f(v|S_i)$ for all $v \notin S_i$. Store these values $\alpha_v \leftarrow f(v|S_i)$ in sorted priority queue.

Priority queue, $O(1)$ to find max, $O(\log n)$ to insert in right place.
Minoux’s Accelerated Greedy for Submodular Functions

- At stage $i$ in the algorithm, we have a set of gains $f(v|S_i)$ for all $v \notin S_i$. Store these values $\alpha_v \leftarrow f(v|S_i)$ in sorted priority queue.
- Priority queue, $O(1)$ to find max, $O(\log n)$ to insert in right place.
- Once we choose a max $v$, then set $S_{i+1} \leftarrow S_i + v$. 


Minoux’s Accelerated Greedy for Submodular Functions

- At stage $i$ in the algorithm, we have a set of gains $f(v|S_i)$ for all $v \notin S_i$. Store these values $\alpha_v \leftarrow f(v|S_i)$ in sorted priority queue.
- Priority queue, $O(1)$ to find max, $O(\log n)$ to insert in right place.
- Once we choose a max $v$, then set $S_{i+1} \leftarrow S_i + v$.
- For $v \notin S_{i+1}$ we have $f(v|S_{i+1}) \leq f(v|S_i)$ by submodularity.
Minoux’s Accelerated Greedy for Submodular Functions

- At stage \( i \) in the algorithm, we have a set of gains \( f(v|S_i) \) for all \( v \not\in S_i \). Store these values \( \alpha_v \leftarrow f(v|S_i) \) in sorted priority queue.
- Priority queue, \( O(1) \) to find max, \( O(\log n) \) to insert in right place.
- Once we choose a max \( v \), then set \( S_{i+1} \leftarrow S_i + v \).
- For \( v \not\in S_{i+1} \) we have \( f(v|S_{i+1}) \leq f(v|S_i) \) by submodularity.
- Therefore, if we find a \( v' \) such that \( f(v'|S_{i+1}) \geq \alpha_v \) for all \( v \neq v' \), then since

\[
f(v'|S_{i+1}) \geq \alpha_v = f(v|S_i) \geq f(v|S_{i+1}) \quad (13.34)
\]

we have the true max, and we need not re-evaluate gains of other elements again.
Minoux’s Accelerated Greedy for Submodular Functions

- At stage $i$ in the algorithm, we have a set of gains $f(v|S_i)$ for all $v \notin S_i$. Store these values $\alpha_v \leftarrow f(v|S_i)$ in sorted priority queue.
- Priority queue, $O(1)$ to find max, $O(\log n)$ to insert in right place.
- Once we choose a max $v$, then set $S_{i+1} \leftarrow S_i + v$.
- For $v \notin S_{i+1}$ we have $f(v|S_{i+1}) \leq f(v|S_i)$ by submodularity.
- Therefore, if we find a $v'$ such that $f(v'|S_{i+1}) \geq \alpha_v$ for all $v \neq v'$, then since

$$f(v'|S_{i+1}) \geq \alpha_v = f(v|S_i) \geq f(v|S_{i+1})$$

we have the true max, and we need not re-evaluate gains of other elements again.

- Strategy is: find the $\text{argmax}_{v' \in V \setminus S_{i+1}} \alpha_{v'}$, and then compute the real $f(v'|S_{i+1})$. If it is greater than all other $\alpha_v$’s then that’s the next greedy step. Otherwise, replace $\alpha_{v'}$ with its real value, resort ($O(\log n)$), and repeat.
Minoux’s Accelerated Greedy for Submodular Functions

Minoux’s algorithm is exact, in that it has the same guarantees as does the standard $O(n^2)$ greedy algorithm (will return the same answers, i.e., those having the $1 - 1/e$ guarantee).
Minoux’s Accelerated Greedy for Submodular Functions

- Minoux’s algorithm is exact, in that it has the same guarantees as does the standard $O(n^2)$ greedy algorithm (will return the same answers, i.e., those having the $1 - 1/e$ guarantee).

- In practice: Minoux’s trick has enormous speedups ($\approx 700\times$) over the standard greedy procedure due to reduced function evaluations and use of good data structures (priority queue).
Minoux’s Accelerated Greedy for Submodular Functions

- Minoux’s algorithm is exact, in that it has the same guarantees as does the standard $O(n^2)$ greedy algorithm (will return the same answers, i.e., those having the $1 - 1/e$ guarantee).

- In practice: Minoux’s trick has enormous speedups ($\approx 700\times$) over the standard greedy procedure due to reduced function evaluations and use of good data structures (priority queue).

- When choosing a of size $k$, naïve greedy algorithm is $O(nk)$ but accelerated variant at the very best does $O(n \log n + k)$, so this limits the speedup.
Minoux’s Accelerated Greedy for Submodular Functions

- Minoux’s algorithm is exact, in that it has the same guarantees as does the standard $O(n^2)$ greedy algorithm (will return the same answers, i.e., those having the $1 - 1/e$ guarantee).

- In practice: Minoux’s trick has enormous speedups ($\approx 700 \times$) over the standard greedy procedure due to reduced function evaluations and use of good data structures (priority queue).

- When choosing a of size $k$, naïve greedy algorithm is $O(nk)$ but accelerated variant at the very best does $O(n \log n + k)$, so this limits the speedup.

- Algorithm has been rediscovered (I think) independently (CELF - cost-effective lazy forward selection, Leskovec et al., 2007)
Minoux’s Accelerated Greedy for Submodular Functions

- Minoux’s algorithm is exact, in that it has the same guarantees as does the standard $O(n^2)$ greedy algorithm (will return the same answers, i.e., those having the $1 - 1/e$ guarantee).

- In practice: Minoux’s trick has enormous speedups ($\approx 700 \times$) over the standard greedy procedure due to reduced function evaluations and use of good data structures (priority queue).

- When choosing a of size $k$, naïve greedy algorithm is $O(nk)$ but accelerated variant at the very best does $O(n \log n + k)$, so this limits the speedup.

- Algorithm has been rediscovered (I think) independently (CELF - cost-effective lazy forward selection, Leskovec et al., 2007)

- Can be used for “big data” sets (e.g., social networks, selecting blogs of greatest influence, document summarization, etc.).
Minoux’s Accelerated Greedy for Submodular Functions

- Minoux’s algorithm is exact, in that it has the same guarantees as does the standard $O(n^2)$ greedy algorithm (will return the same answers, i.e., those having the $1 - 1/e$ guarantee).

- In practice: Minoux’s trick has enormous speedups ($\approx 700\times$) over the standard greedy procedure due to reduced function evaluations and use of good data structures (priority queue).

- When choosing a of size $k$, naïve greedy algorithm is $O(nk)$ but accelerated variant at the very best does $O(n \log n + k)$, so this limits the speedup.

- Algorithm has been rediscovered (I think) independently (CELF - cost-effective lazy forward selection, Leskovec et al., 2007)

- Can be used for “big data” sets (e.g., social networks, selecting blogs of greatest influence, document summarization, etc.).

- Very good if there are many elements $v$ with $f(v) < f(u|V \setminus \{u\})$ for enough $u$ elements (gain of $v$ is evaluated only once).
Use a priority queue $Q$ as a data structure: operations include:

- Insert an item $(v, \alpha)$ into queue, with $v \in V$ and $\alpha \in \mathbb{R}$.

$$\text{insert}(Q, (v, \alpha)) \quad (13.35)$$

- Pop the item $(v, \alpha)$ with maximum value $\alpha$ off the queue.

$$(v, \alpha) \leftarrow \text{pop}(Q) \quad (13.36)$$

- Query the value of the max item in the queue $\max(Q) \in \mathbb{R}$.

$$\max(Q) \quad (13.37)$$

On next slide, we call a popped item “fresh” if the value $(v, \alpha)$ popped has the correct value $\alpha = f(v | S_i)$. Use extra “bit” to store this info. If a popped item is fresh, it must be the maximum — this can happen if, at a given iteration, $v$ was first popped and neither fresh nor maximum so placed back in the queue, and it then percolates back to the top at which point it is fresh — thereby avoid extra queue check.
Use a priority queue $Q$ as a data structure: operations include:

- Insert an item $(v, \alpha)$ into queue, with $v \in V$ and $\alpha \in \mathbb{R}$.

\[
\text{insert}(Q, (v, \alpha))
\] (13.35)
Use a priority queue $Q$ as a data structure: operations include:

- Insert an item $(v, \alpha)$ into queue, with $v \in V$ and $\alpha \in \mathbb{R}$.
  
  $$\text{insert}(Q, (v, \alpha))$$  
  \hspace{1cm} (13.35)

- Pop the item $(v, \alpha)$ with maximum value $\alpha$ off the queue.
  
  $$(v, \alpha) \leftarrow \text{pop}(Q)$$  
  \hspace{1cm} (13.36)
Use a priority queue $Q$ as a data structure: operations include:

- Insert an item $(v, \alpha)$ into queue, with $v \in V$ and $\alpha \in \mathbb{R}$.

  \[ \text{insert}(Q, (v, \alpha)) \]  
  \[ (13.35) \]

- Pop the item $(v, \alpha)$ with maximum value $\alpha$ off the queue.

  \[ (v, \alpha) \leftarrow \text{pop}(Q) \]  
  \[ (13.36) \]

- Query the value of the max item in the queue

  \[ \max(Q) \in \mathbb{R} \]  
  \[ (13.37) \]
Priority Queue

- Use a priority queue $Q$ as a data structure: operations include:
  - Insert an item $(v, \alpha)$ into queue, with $v \in V$ and $\alpha \in \mathbb{R}$.
    
    $$\text{insert}(Q, (v, \alpha)) \quad (13.35)$$

  - Pop the item $(v, \alpha)$ with maximum value $\alpha$ off the queue.
    
    $$(v, \alpha) \leftarrow \text{pop}(Q) \quad (13.36)$$

  - Query the value of the max item in the queue
    
    $$\max(Q) \in \mathbb{R} \quad (13.37)$$

- On next slide, we call a popped item “fresh” if the value $(v, \alpha)$ popped has the correct value $\alpha = f(v|S_i)$. Use extra “bit” to store this info.
Priority Queue

- Use a priority queue $Q$ as a data structure: operations include:
  - Insert an item $(v, \alpha)$ into queue, with $v \in V$ and $\alpha \in \mathbb{R}$.
    \[
    \text{insert}(Q, (v, \alpha)) \tag{13.35}
    \]
  - Pop the item $(v, \alpha)$ with maximum value $\alpha$ off the queue.
    \[
    (v, \alpha) \leftarrow \text{pop}(Q) \tag{13.36}
    \]
  - Query the value of the max item in the queue
    \[
    \max(Q) \in \mathbb{R} \tag{13.37}
    \]

- On next slide, we call a popped item “fresh” if the value $(v, \alpha)$ popped has the correct value $\alpha = f(v|S_i)$. Use extra “bit” to store this info

- If a popped item is fresh, it must be the maximum — this can happen if, at given iteration, $v$ was first popped and neither fresh nor maximum so placed back in the queue, and it then percolates back to the top at which point it is fresh — thereby avoid extra queue check.
Algorithm 2: Minoux’s Accelerated Greedy Algorithm

1. Set $S_0 \leftarrow \emptyset$; $i \leftarrow 0$; Initialize priority queue $Q$;
2. for $v \in E$ do
3. \hspace{1em} INSERT($Q, f(v)$)
4. repeat
5. \hspace{2em} $(v, \alpha) \leftarrow \text{pop}(Q)$;
6. \hspace{3em} if $\alpha$ not “fresh” then
7. \hspace{4em} recomputes $\alpha \leftarrow f(v|S_i)$
8. \hspace{3em} if (popped $\alpha$ in line 5 was “fresh”) OR ($\alpha \geq \max(Q)$) then
9. \hspace{4em} Set $S_{i+1} \leftarrow S_i \cup \{v\}$;
10. \hspace{4em} $i \leftarrow i + 1$;
11. else
12. \hspace{2em} insert($Q, (v, \alpha)$)
13. until $i = |E|$;
Given polymatroid $f$, goal is to find a covering set of minimum cost:

$$S^* \in \operatorname{argmin}_{S \subseteq V} |S| \text{ such that } f(S) \geq \alpha$$  \hspace{1cm} (13.38)

where $\alpha$ is a “cover” requirement.
(Minimum) **Submodular Set Cover**

- Given polymatroid \( f \), goal is to find a covering set of minimum cost:

\[
S^* \in \operatorname{arg\,min} |S| \text{ such that } f(S) \geq \alpha
\]  

(13.38)

where \( \alpha \) is a “cover” requirement.

- Normally take \( \alpha = f(V) \) but defining \( f'(A) = \min \{ f(A), \alpha \} \) we can take any \( \alpha \). Hence, we have equivalent formulation:

\[
S^* \in \operatorname{arg\,min} |S| \text{ such that } f'(S) \geq f'(V)
\]  

(13.39)
Given polymatroid $f$, goal is to find a covering set of minimum cost:

$$S^* \in \arg\min_{S \subseteq V} |S| \text{ such that } f(S) \geq \alpha$$

(13.38)

where $\alpha$ is a “cover” requirement.

Normally take $\alpha = f(V)$ but defining $f'(A) = \min\{f(A), \alpha\}$ we can take any $\alpha$. Hence, we have equivalent formulation:

$$S^* \in \arg\min_{S \subseteq V} |S| \text{ such that } f'(S) \geq f'(V)$$

(13.39)

Note that this immediately generalizes standard set cover, in which case $f(A)$ is the cardinality of the union of sets indexed by $A$. 
(Minimum) Submodular Set Cover

- Given polymatroid \( f \), goal is to find a covering set of minimum cost:

\[
S^* \in \arg\min_{S \subseteq V} |S| \text{ such that } f(S) \geq \alpha
\]  

(13.38)

where \( \alpha \) is a “cover” requirement.

- Normally take \( \alpha = f(V) \) but defining \( f'(A) = \min\{f(A), \alpha\} \) we can take any \( \alpha \). Hence, we have equivalent formulation:

\[
S^* \in \arg\min_{S \subseteq V} |S| \text{ such that } f'(S) \geq f'(V)
\]  

(13.39)

- Note that this immediately generalizes standard set cover, in which case \( f(A) \) is the cardinality of the union of sets indexed by \( A \).

- Greedy Algorithm: Pick the first chain item \( S_i \) chosen by aforementioned greedy algorithm such that \( f(S_i) \geq \alpha \) and output that as solution.
For integer valued $f$, this greedy algorithm an $O(\log(\max_{s \in V} f(\{s\})))$ approximation. Let $S^*$ be optimal, and $S^G$ be greedy solution, then

$$|S^G| \leq |S^*| H(\max_{s \in V} f(\{s\})) = |S^*| O(\log e(\max_{s \in V} f(\{s\}))) \quad (13.40)$$

where $H$ is the harmonic function, i.e., $H(d) = \sum_{i=1}^{d} (1/i)$. 

(Minimum) Submodular Set Cover: Approximation Analysis
(Minimum) Submodular Set Cover: Approximation Analysis

- For integer valued $f$, this greedy algorithm an $O(\log(\max_{s \in V} f(\{s\})))$ approximation. Let $S^*$ be optimal, and $S^G$ be greedy solution, then

$$|S^G| \leq |S^*| H(\max_{s \in V} f(\{s\})) = |S^*| O(\log e (\max_{s \in V} f(\{s\}))) \quad (13.40)$$

where $H$ is the harmonic function, i.e., $H(d) = \sum_{i=1}^{d} (1/i)$.

- If $f$ is not integral value, then bounds we get are of the form:

$$|S^G| \leq |S^*| \left(1 + \log e \frac{f(V)}{f(V) - f(S_{T-1})}\right) \quad (13.41)$$

where $S_T$ is the final greedy solution that occurs at step $T$. 
For integer valued $f$, this greedy algorithm an $O(\log(\max_{s \in V} f(\{s\})))$ approximation. Let $S^*$ be optimal, and $S^G$ be greedy solution, then

$$|S^G| \leq |S^*| H(\max_{s \in V} f(\{s\})) = |S^*| O(\log_e(\max_{s \in V} f(\{s\}))) \quad (13.40)$$

where $H$ is the harmonic function, i.e., $H(d) = \sum_{i=1}^{d} (1/i)$.

If $f$ is not integral value, then bounds we get are of the form:

$$|S^G| \leq |S^*| \left(1 + \log_e \frac{f(V)}{f(V) - f(S_{T-1})} \right) \quad (13.41)$$

where $S_T$ is the final greedy solution that occurs at step $T$.

Set cover is hard to approximate with a factor better than $(1 - \epsilon) \log \alpha$, where $\alpha$ is the desired cover constraint.
Summary: Monotone Submodular Maximization

- Only makes sense when there is a constraint.
Summary: Monotone Submodular Maximization

- Only makes sense when there is a constraint.
- We discussed cardinality constraint
Summary: Monotone Submodular Maximization

- Only makes sense when there is a constraint.
- We discussed cardinality constraint
- Generalizes the max $k$-cover problem, and also similar to the set cover problem.
Summary: Monotone Submodular Maximization

- Only makes sense when there is a constraint.
- We discussed cardinality constraint
- Generalizes the max $k$-cover problem, and also similar to the set cover problem.
- Simple greedy algorithm gets $1 - e^{-\ell/k}$ approximation, where $k$ is size of optimal set we compare against, and $\ell$ is size of set greedy algorithm chooses.
Summary: Monotone Submodular Maximization

- Only makes sense when there is a constraint.
- We discussed cardinality constraint
- Generalizes the max $k$-cover problem, and also similar to the set cover problem.
- Simple greedy algorithm gets $1 - e^{-\ell/k}$ approximation, where $k$ is size of optimal set we compare against, and $\ell$ is size of set greedy algorithm chooses.
- Submodular cover: min. $|S|$ s.t. $f(S) \geq \alpha$. 
Summary: Monotone Submodular Maximization

- Only makes sense when there is a constraint.
- We discussed cardinality constraint.
- Generalizes the max $k$-cover problem, and also similar to the set cover problem.
- Simple greedy algorithm gets $1 - e^{-\ell/k}$ approximation, where $k$ is size of optimal set we compare against, and $\ell$ is size of set greedy algorithm chooses.
- Submodular cover: min. $|S|$ s.t. $f(S) \geq \alpha$.
- Minoux’s accelerated greedy trick.
The Greedy Algorithm: $1 - 1/e$ intuition.

- At step $i < k$, greedy chooses $v_i$ to maximize $f(v|S_i)$.

- Let $S^*$ be optimal solution (of size $k'$) and $\text{OPT} = f(S^*)$. By submodularity, we will show:

$$\exists v \in V \setminus S_i : f(v|S_i) = f(S_i + v|S_i) \geq \frac{1}{k}(\text{OPT} - f(S_i)) \tag{13.21}$$

Equation (13.30) will show that Equation (13.21) $\Rightarrow$:

$$\text{OPT} - f(S_{i+1}) \leq (1 - 1/k)(\text{OPT} - f(S_i))$$

$\Rightarrow$$\text{OPT} - f(S_k) \leq (1 - 1/k)^k \text{OPT}$

$\Rightarrow$$\text{OPT} \leq \frac{1}{e} \text{OPT}$

$\Rightarrow$$\text{OPT}(1 - 1/e) \leq f(S_k)$
How can we produce a randomized greedy strategy, one where each greedy sweep produces a set that, on average, has a $1 - 1/e$ guarantee?
How can we produce a randomized greedy strategy, one where each greedy sweep produces a set that, on average, has a $1 - 1/e$ guarantee? Suppose the following holds:

$$E[f(a_{i+1}|A_i)] \geq \frac{f(OPT) - f(A_i)}{k}$$ (13.42)

where $A_i = (a_1, a_2, \ldots, a_i)$ are the first $i$ elements chosen by the strategy.