Submodular Functions, Optimization, and Applications to Machine Learning

— Fall Quarter, Lecture 14 —

http://www.ee.washington.edu/people/faculty/bilmes/classes/ee563_spring_2018/

Prof. Jeff Bilmes

University of Washington, Seattle
Department of Electrical Engineering
http://melodi.ee.washington.edu/~bilmes

Nov 18th, 2020

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

- \( f(A) + 2f(C) + f(B) \)
- \( f(A) + f(C) + f(B) \)
- \( f(A \cap B) \)
Homework 3, out, due next Wednesday, Nov 25th, 2020, 11:59pm.

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. I can meet Monday night at 10:00pm as well (let me know if you want to meet then).

Reminder, all lectures are being recorded and posted to YouTube. To get the links, see our announcements (https://canvas.uw.edu/courses/1397085/announcements).
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
L–(11/11): Veterans Day, Holiday
L13(11/16): Polymatroids and Greedy, Possible Polytopes, Extreme Points, Cardinality Constrained Maximization
L14(11/18): Cardinality Constrained Maximization, Curvature
L15(11/23):
L16(11/25): L17(11/30):
L18(12/2): L19(12/7):
L20(12/9): maximization.

Last day of instruction, Fri. Dec 11th. Finals Week: Dec 12-18, 2020
Logistics Review

Polymatroid with labeled edge lengths

- 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 $x^T w = x(e_1)w(e_1) + x(e_2)w(e_2)$.
- If $w(e_2) > w(e_1)$ the upper extreme point indicated maximizes $x^T w$ over $x \in P_f^+$. If $w(e_2) < w(e_1)$ the lower extreme point indicated maximizes $x^T w$ over $x \in P_f^+$. 

Diagram:

- Maximal point in $P_f^+$ for $w$ in this region.
- Maximal point in $P_f^+$ for $w$ in this region.

Prof. Jeff Bilmes
EE563/Spring 2020/Submodularity - Lecture 14 - Nov 18th, 2020
F5/37 (pg.5/144)
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 $\{E_i\}_{i=1}^m$ be a set of $m$ subsets $E_i \subseteq E$ with $\bigcup_i E_i = E$.
- 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.,

$$\text{minimize} |X| \text{ subject to } f(X) \geq |E| \quad (14.1)$$

- 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 ratio better than $(1 - \epsilon) \log n$ for any $\epsilon > 0$ unless NP is slightly superpolynomial ($n^{O((\log \log n))}$).
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:

$$\max f(X) \text{ subject to } |X| \leq k$$  \hspace{1cm} (14.1)

- This problem is NP-hard, and Feige in 1998 showed that it cannot be approximated with a ratio better than $(1 - 1/e)$. 
Recall the (uncapacitated) facility location function

\[ f(A) = \sum_{v \in V} \max_{a \in A} s_{a,v} \]

where \( s_{a,v} \) is the similarity (affinity, value) 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 \( s_{v,a} \)).
Recall the (uncapacitated) facility location function
\[ f(A) = \sum_{v \in V} \max_{a \in A} s_{a,v} \]
where \( s_{a,v} \) is the similarity (affinity, value) 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:

\[
\begin{align*}
    s_{i,j} &= \exp(-d(x_i, x_j)) \\
    d(x_i, x_j) &= \|x_i - x_j\|_2
\end{align*}
\]
Recall the (uncapacitated) facility location function
\[ f(A) = \sum_{v \in V} \max_{a \in A} s_{a,v} \text{ where } s_{a,v} \text{ is the similarity (affinity, value) between } a \text{ and } v. \]
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 the (uncapacitated) facility location function
  \[ f(A) = \sum_{v \in V} \max_{a \in A} s_{a,v} \]  
  where \( s_{a,v} \) is the similarity (affinity, value) 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:
Recall the (uncapacitated) facility location function
\[ f(A) = \sum_{v \in V} \max_{a \in A} s_{a,v} \]
where \( s_{a,v} \) is the similarity (affinity, value) 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 \( k = 10 \).
Recall the (uncapacitated) facility location function
\[ f(A) = \sum_{v \in V} \max_{a \in A} s_{a,v} \]
where \( s_{a,v} \) is the similarity (affinity, value) 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 \( k = 10 \).

Multiple uniformly-at-random random summaries.
Now we are given an arbitrary polymatroid function $f$. 
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 \( A^* \in \text{argmax} \{ f(A) : |A| \leq k \} \)

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.
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.

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

\[
S_{i+1} = S_i \cup \left\{ \arg\max_{v \in V \setminus S_i} f(S_i \cup \{v\}) \right\}
\]  \hspace{1cm} (14.1)
The Greedy Algorithm for Submodular Max

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:
      \[
      v_i \in \arg\max_{v \in V \setminus S_i} f(\{v\}|S_i) = \arg\max_{v \in V \setminus S_i} f(S_i \cup \{v\}) ;
      \]
4.   Set $S_{i+1} \leftarrow S_i \cup \{v_i\}$;

\[
O(n^2 \cdot OE) \quad f(v \mid S_i) = f(S_i \cup v) - f(S_i)
\]
This algorithm has a guarantee
This algorithm has a guarantee

**Theorem 14.3.1**

*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 14.3.1**

*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 we have selected \( k \) elements in Algorithm 4.
This algorithm has a guarantee

**Theorem 14.3.1**

*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 we have selected $k$ elements in Algorithm 4.
- We can think of a “greedy operator” $\tilde{A} \in \text{gargmax} \{f(A) : |A| \leq k\}$.
This algorithm has a guarantee

**Theorem 14.3.1**

*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 we have selected \( k \) elements in Algorithm 4.
- We can think of a “greedy operator” \( \tilde{A} \in \arg\max \{ f(A) : |A| \leq k \} \).
- 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: $1 - \frac{1}{e}$ intuition.

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

Equation (14.11) will show that Equation (14.2):

$$\text{OPT} f(S_i + 1) \leq (1 - \frac{1}{e}) \text{OPT} f(S_i)$$

so

$$\text{OPT} f(S_k) \leq (1 - \frac{1}{e}) \text{OPT} f(S_0)$$
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:

$$f(v|S_i) = f(S_i + v|S_i)$$

$$\geq \frac{1}{k}(\text{OPT} - f(S_i))$$

Equation (14.2) will show that Equation (14.1):

$$\text{OPT} \geq \frac{1}{e} \text{OPT}$$
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:

$$\exists v \in V \setminus S_i : f(v|S_i) = f(S_i + v|S_i) \geq \frac{1}{k}(\text{OPT} - f(S_i)) \quad (14.2)$$

\[ \begin{array}{c}
\forall \alpha \in \mathbb{R}^+ \quad (\text{OPT} - f(S_i)) \\
\Rightarrow \quad f(S_i) \\
\end{array} \]
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:

$$\exists v \in V \setminus S_i : f(v|S_i) = f(S_i + v|S_i) \geq \frac{1}{k}(\text{OPT} - f(S_i)) \quad (14.2)$$

Equation (14.11) will show that Equation (14.2) $\Rightarrow$:

$$\begin{align*}
\text{OPT} - f(S_{i+1}) &\leq (1 - 1/k)(\text{OPT} - f(S_i)) \\
\Rightarrow \quad \text{OPT} - f(S_k) &\leq (1 - 1/k)^k \text{OPT} \\
&\leq 1/e \text{OPT} \\
\Rightarrow \quad \text{OPT}(1 - 1/e) &\leq f(S_k)
\end{align*}$$
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:

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

(14.2)

Equation (14.11) will show that Equation (14.2) $\Rightarrow$:

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

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

$\Rightarrow$ $OPT(1 - 1/e) \leq f(S_k)$
Theorem 14.3.2 (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. (14.1)). Then for all \( k, \ell \in \mathbb{Z}_++ \), we have:

\[
f(S_\ell) \geq (1 - e^{-\ell/k}) \max_{S:|S|\leq k} f(S) \tag{14.3}
\]

and in particular, for \( \ell = k \), we have \( f(S_k) \geq (1 - 1/e) \max_{S:|S|\leq k} f(S) \).
Theorem 14.3.2 (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. (14.1)).
Then for all $k, \ell \in \mathbb{Z}_{++}$, we have:

$$f(S_\ell) \geq (1 - e^{-\ell/k}) \max_{S:|S| \leq k} f(S) \tag{14.3}$$

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 14.3.2 (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. (14.1)).

Then for all \( k, \ell \in \mathbb{Z}_{++} \), we have:

\[
f(S_\ell) \geq (1 - e^{-\ell/k}) \max_{S:|S| \leq k} f(S)
\] (14.3)

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 14.3.2 (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. (14.1)). Then for all \( k, \ell \in \mathbb{Z}_{++} \), we have:

\[
f(S_\ell) \geq (1 - e^{-\ell/k}) \max_{S : |S| \leq k} f(S) \tag{14.3}
\]

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 \( k \).
**Cardinality Constrained Polymatroid Max Theorem**

**Theorem 14.3.2 (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. (14.1)). Then for all $k, \ell \in \mathbb{Z}_{++}$, we have:

$$f(S_{\ell}) \geq (1 - e^{-\ell/k}) \max_{S:|S| \leq k} f(S) \quad (14.3)$$

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 $\ell > k$. 

---

Prof. Jeff Bilmes

EE563/Spring 2020/Submodularity - Lecture 14 - Nov 18th, 2020

F14/37 (pg.35/144)
Cardinality Constrained Polymatroid Max Theorem

Proof of Theorem 14.3.2.

Fix \((\text{number of items greedy will choose})\) and \(k\) (size of optimal set to compare against).

Set \(S^\ast \subseteq \arg\max\{f(S) : |S| \leq k\}\).

w.l.o.g. assume \(|S^\ast| = k\).

Order \(S^\ast = (v^\ast_1, v^\ast_2, \ldots, v^\ast_k)\) arbitrarily.

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

Then the following inequalities (on the next slide) follow:

\[ \ldots \]
Proof of Theorem 14.3.2.

- Fix \( \ell \) (number of items greedy will chose) and \( k \) (size of optimal set to compare against).
Proof of Theorem 14.3.2.

- Fix $\ell$ (number of items greedy will chose) and $k$ (size of optimal set to compare against).
- Set $S^{*} \in \arg\max \{ f(S) : |S| \leq k \}$
Proof of Theorem 14.3.2.

- 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 14.3.2.

- Fix \( \ell \) (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 \( S^* = (v_1^*, v_2^*, \ldots, v_k^*) \) arbitrarily.
Proof of Theorem 14.3.2.

- 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 \( i \in \{1, 2, \ldots, \ell\} \).

...
Proof of Theorem 14.3.2.

- 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 \( i \in \{1, 2, \ldots, \ell\} \).
- Then the following inequalities (on the next slide) follow:
Cardinality Constrained Polymatroid Max Theorem

... proof of Theorem 14.3.2 cont.

\[ f(S^*) \geq f(S_i) + k \sum_{j=1}^{k} f(S_i) \mid v_j \mid \leq S_i \]

(14.4)

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

(14.5)

\[ f(S_i) + k \sum_{v_j \in S^*} f(S_i) \mid v_j \mid + 1 \]

(14.6)

\[ f(S_i) + k f(S_i) \mid S_i + 1 \mid \]

(14.7)

\[ f(S_i) \geq k f(S_i) \mid S_i + 1 \mid \]

(14.9)
For all $i < \ell$, we have
\[ f(S^*) \]
Cardinality Constrained Polymatroid Max Theorem

... proof of Theorem 14.3.2 cont.

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

\[
f(S^*) \leq f(S^* \cup S_i)
\]
For all $i < \ell$, we have

$$f(S^*) \leq f(S^* \cup S_i) = f(S_i) + f(S^*|S_i) \quad (14.4)$$
For all $i < \ell$, we have

$$f(S^*) \leq f(S^* \cup S_i) = f(S_i) + f(S^*|S_i) \quad (14.4)$$

$$= f(S_i) + \sum_{j=1}^{k} f(v_j^*|S_i \cup \{v_1^*, v_2^*, \ldots, v_{j-1}^*\}) \quad (14.5)$$

Therefore, we have shown Equation 14.2 above, i.e.,

$$f(S^*) \leq f(S_i) + k \cdot f(S_i) + \sum_{j=1}^{k} f(v_j^*|S_i \cup \{v_1^*, v_2^*, \ldots, v_{j-1}^*\}) \quad (14.6)$$

$$= f(S_i) + kf(S_i) + \sum_{j=1}^{k} f(v_j^*|S_i \cup \{v_1^*, v_2^*, \ldots, v_{j-1}^*\}) \quad (14.7)$$

$$= f(S_i) + kf(S_{i+1}|S_i) \quad (14.8)$$

...
For all \( i < \ell \), we have
\[
f(S^*) \leq f(S^* \cup S_i) = f(S_i) + f(S^* \mid S_i)
\]
\[
= f(S_i) + \sum_{j=1}^{k} f(v_j^* \mid S_i \cup \{v_1^*, v_2^*, \ldots, v_{j-1}^*\})
\]
\[
\leq f(S_i) + \sum_{v \in S^*} f(v \mid S_i)
\]
...proof of Theorem 14.3.2 cont.

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

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

(14.4)

\[
= f(S_i) + \sum_{j=1}^{k} f(v^*_j|S_i \cup \{v^*_1, v^*_2, \ldots, v^*_{j-1}\})
\]  

(14.5)

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

(14.6)

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

(14.7)
For all $i < \ell$, we have
\[ f(S^*) \leq f(S^* \cup S_i) = f(S_i) + f(S^* | S_i) \tag{14.4} \]
\[ = f(S_i) + \sum_{j=1}^{k} f(v_j^* | S_i \cup \{v_1^*, v_2^*, \ldots, v_{j-1}^*\}) \tag{14.5} \]
\[ \leq f(S_i) + \sum_{v \in S^*} f(v | S_i) \tag{14.6} \]
\[ \leq f(S_i) + \sum_{v \in S^*} f(v_{i+1} | S_i) = f(S_i) + \sum_{v \in S^*} f(S_{i+1} | S_i) \tag{14.7} \]
\[ f(v_{i+1} | S_i) = f(S_i + v_{i+1}) - f(S_i) \]
\[ = f(S_i + v_{i+1}) - f(S_i) \]
\[ = f(S_{i+1} | S_i) \]

...
Cardinality Constrained Polymatroid Max Theorem

...proof of Theorem 14.3.2 cont.

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

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

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

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

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

$$= f(S_i) + kf(S_{i+1}|S_i)$$
For all $i < \ell$, we have

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

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

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

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

$$= f(S_i) + kf(S_{i+1}|S_i)$$

Therefore, we have shown Equation 14.2 above, i.e.,

$$f(S^*) - f(S_i) \leq kf(S_{i+1}|S_i) = k(f(S_{i+1}) - f(S_i))$$  \hspace{1cm} (14.9)
Define the gap $i$, $f(S^{\ast}) - f(S_i)$. Thus $i \leq k \leq i + 1 = f(S_{i+1}) - f(S_i)$.

Equation (14.2) becomes $i \leq k \leq i + 1 (14.10)$ or rearranging slightly $i + 1 \leq (1 - 1/k)i (14.11)$.

The relationship between $0$ and $\beta$ is then $\beta \leq (1 - 1/k) \leq 0 (14.12)$.

Now, $0 = f(S^{\ast}) - f(\cdot)$ since $f_0$.

Also, by variation bound $1 - x \leq e^x$ for $x \in \mathbb{R}$, we have $\beta \leq e^{\beta/k} - f(S^{\ast}) (14.13)$. ...
Define the gap $\delta_i \triangleq f(S^*) - f(S_i)$. Thus $\delta_i - \delta_{i+1} = f(S_{i+1}) - f(S_i)$.
Define the gap $\delta_i \triangleq f(S^*) - f(S_i)$. Thus $\delta_i - \delta_{i+1} = f(S_{i+1}) - f(S_i)$, and Equation 14.2 becomes

$$\delta_i \leq k(\delta_i - \delta_{i+1}) \quad (14.10)$$

or rearranging slightly

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

$$\exists v \in V \setminus S_i : f(v|S_i) = f(S_i + v|S_i) \geq \frac{1}{k}(\text{OPT} - f(S_i)) \quad (14.2)$$

Equation (14.11) will show that Equation (14.2) $\Rightarrow$:

$$\begin{align*}
\text{OPT} - f(S_{i+1}) &\leq (1 - 1/k)(\text{OPT} - f(S_i)) \\
\Rightarrow \quad \text{OPT} - f(S_k) &\leq (1 - 1/k)^k \text{OPT} \\
&\leq 1/e \text{OPT} \\
\Rightarrow \quad \text{OPT}(1 - 1/e) &\leq f(S_k)
\end{align*}$$
Define the gap $\delta_i \triangleq f(S^*) - f(S_i)$. Thus $\delta_i - \delta_{i+1} = f(S_{i+1}) - f(S_i)$, and Equation 14.2 becomes

$$\delta_i \leq k(\delta_i - \delta_{i+1}) \quad (14.10)$$

or rearranging slightly

$$\delta_{i+1} \leq (1 - \frac{1}{k})\delta_i \quad (14.11)$$
Define the gap $\delta_i \triangleq f(S^*) - f(S_i)$. Thus $\delta_i - \delta_{i+1} = f(S_{i+1}) - f(S_i)$, and Equation 14.2 becomes

$$\delta_i \leq k(\delta_i - \delta_{i+1}) \quad (14.10)$$

or rearranging slightly

$$\delta_{i+1} \leq (1 - \frac{1}{k})\delta_i \quad (14.11)$$

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

$$\delta_{\ell} \leq (1 - \frac{1}{k})\ell \delta_0 \quad (14.12)$$
... proof of Theorem 14.3.2 cont.

- Define the gap $\delta_i \triangleq f(S^*) - f(S_i)$. Thus $\delta_i - \delta_{i+1} = f(S_{i+1}) - f(S_i)$, and Equation 14.2 becomes

$$\delta_i \leq k(\delta_i - \delta_{i+1}) \quad (14.10)$$

or rearranging slightly

$$\delta_{i+1} \leq (1 - \frac{1}{k})\delta_i \quad (14.11)$$

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

$$\delta_{\ell} \leq (1 - \frac{1}{k})^\ell \delta_0 \quad (14.12)$$

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

---

Prof. Jeff Bilmes  
EE563/Spring 2020/Submodularity - Lecture 14 - Nov 18th, 2020  
F17/37 (pg.58/144)
Cardinality Constrained Polymatroid Max Theorem

... proof of Theorem 14.3.2 cont.

- Define the gap $\delta_i \triangleq f(S^*) - f(S_i)$. Thus $\delta_i - \delta_{i+1} = f(S_{i+1}) - f(S_i)$, and Equation 14.2 becomes

$$\delta_i \leq k(\delta_i - \delta_{i+1}) \quad (14.10)$$

- or rearranging slightly

$$\delta_{i+1} \leq (1 - \frac{1}{k})\delta_i \quad (14.11)$$

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

$$\delta_\ell \leq (1 - \frac{1}{k})^\ell \delta_0 \quad (14.12)$$

- 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

$$\delta_\ell \leq (1 - \frac{1}{k})^\ell \delta_0 \leq e^{-\ell/k} f(S^*) \quad (14.13)$$
... proof of Theorem 14.3.2 cont.

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

$$f(S_\ell) \geq (1 - e^{-\ell/k})f(S^*)$$  \hspace{1cm} (14.14)
... proof of Theorem 14.3.2 cont.

- When we identify $\delta_\ell = f(S^*) - f(S_\ell)$, a bit of rearranging then gives:
  \[
f(S_\ell) \geq (1 - e^{-\ell/k}) f(S^*) \tag{14.14}
  \]

- 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 $k$, $f(S_k) \geq (1 - 1/e) f(S^*) \approx 0.6321 f(S^*)$. 

---

Polymatroids, Greedy, and Cardinality Constrained Maximization
Curvature
Cardinality Constrained Polymatroid Max Theorem
---

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

$$f(S_{\ell}) \geq (1 - e^{-\ell/k}) f(S^*)$$ (14.14)

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 $k$, $f(S_k) \geq (1 - 1/e)f(S^*) \approx 0.6321f(S^*)$.

What if we want to guarantee a solution no worse than $.95f(S^*)$ where $|S^*| = k$?
Cardinality Constrained Polymatroid Max Theorem

... proof of Theorem 14.3.2 cont.

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

\[
f(S_\ell) \geq (1 - e^{-\ell/k}) f(S^*)
\]  
(14.14)

- 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

\[
\ell = \left\lceil -k \ln(1 - 0.95) \right\rceil = 4k.
\]
Cardinality Constrained Polymatroid Max Theorem

... proof of Theorem 14.3.2 cont.

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

  $$f(S_\ell) \geq (1 - e^{-\ell/k}) f(S^*)$$  \hspace{1cm} (14.14)

- 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 $0.95f(S^*)$ where $|S^*| = k$? Set $0.95 = (1 - e^{-\ell/k})$, which gives

  $\ell = \lceil -k \ln(1 - 0.95) \rceil = 4k$. And $\lceil -\ln(1 - 0.999) \rceil = 7$.

  If you set $f(S_{\ell + 1}) < 0$, at some point $i < k$, then you have OPT.
Cardinality Constrained Polymatroid Max Theorem

... proof of Theorem 14.3.2 cont.

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

$$f(S_\ell) \geq (1 - e^{-\ell/k}) f(S^*) \quad (14.14)$$

- 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 $\ell = \lceil -k \ln(1 - 0.95) \rceil = 4k$. 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 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.

$$\max_{v \in V \setminus S_i} f(v|S_i)$$
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.
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 $S_{i+1} \leftarrow S_i + v$. 

Prof. Jeff Bilmes
EE563/Spring 2020/Submodularity - Lecture 14 - Nov 18th, 2020
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.
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 a 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 $v \neq v'$ in the queue, then since

$$f(v'|S_{i+1}) \geq \alpha_v = f(v|S_i) \geq f(v|S_{i+1})$$  \hspace{1cm} (14.15)

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 $v \neq v'$ in the queue, then since

$$f(v'|S_{i+1}) \geq \alpha_v = f(v|S_i) \geq f(v|S_{i+1}) \quad (14.15)$$

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

- Strategy is: find $v' \in \arg\max_{v \in V \setminus S_{i+1}} \alpha_v$, and then compute the real “fresh” $f(v'|S_{i+1})$. If it is greater than all other $\alpha_v$’s in the queue, then that’s the next greedy step. Otherwise, replace $\alpha_v'$ with its fresh value, resort ($O(\log n)$), and repeat.
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 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 set 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 set 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 set 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 set 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 $f(v|S_{i+1}) \geq f(u|S_i)$ for many other elements $u$, for many $i$. 
Priority Queue

- Use a priority queue $Q$ as a data structure: operations include:

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

$$insert(Q, (v, \rho)) \quad (14.16)$$

2. **Pop** the item with maximum value off the queue.

$$pop(Q) \quad (14.17)$$

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

$$\max(Q) \quad (14.18)$$

On next slide, we call a popped item "fresh" if the value $(v, \rho)$ popped has the correct value $\rho = f(v | S_i)$. See extra slide to store this info.
Use a priority queue $Q$ as a data structure: operations include:

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

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

(14.16)
Use a priority queue $Q$ as a data structure: operations include:

- Insert an item $(v, \alpha)$ into queue, with $v \in V$ and $\alpha \in \mathbb{R}$.
  \[
  \text{insert}(Q, (v, \alpha)) \quad (14.16)
  \]

- Pop the item $(v, \alpha)$ with maximum value $\alpha$ off the queue.
  \[
  (v, \alpha) \leftarrow \text{pop}(Q) \quad (14.17)
  \]
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 $\alpha \in \mathbb{R}$.
    \[
    \text{insert}(Q, (v, \alpha)) \quad \mathcal{O}(\log n) \quad (14.16)
    \]
  - Pop the item $(v, \alpha)$ with maximum value $\alpha$ off the queue.
    \[
    (v, \alpha) \leftarrow \text{pop}(Q) \quad \mathcal{O}(\log n) \quad (14.17)
    \]
  - Query the value of the max item in the queue
    \[
    \max(Q) \in \mathbb{R} \quad \mathcal{O}(1) \quad (14.18)
    \]
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 $\alpha \in \mathbb{R}$.
    
    \[
    \text{insert}(Q, (v, \alpha)) \quad (14.16)
    \]
  - Pop the item $(v, \alpha)$ with maximum value $\alpha$ off the queue.
    
    \[
    (v, \alpha) \leftarrow \text{pop}(Q) \quad (14.17)
    \]
  - Query the value of the max item in the queue
    
    \[
    \max(Q) \in \mathbb{R} \quad (14.18)
    \]

- 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.
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.   INSERT($Q, f(v)$)
   4. repeat
      5.     $(v, \alpha) \leftarrow \text{pop}(Q)$;
      6.     if $\alpha$ not “fresh” then
         7.        recompute $\alpha \leftarrow f(v|S_i)$
      8.     if (popped $\alpha$ in line 5 was “fresh”) OR ($\alpha \geq \max(Q)$) then
         9.         Set $S_{i+1} \leftarrow S_i \cup \{v\}$; and mark other items in $Q$ as stale
         10.        $i \leftarrow i + 1$;
      else
         12.        insert($Q, (v, \alpha)$) as a fresh item
   13. until $i = |E|$;
On the accelerated greedy algorithm

- 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 next iteration at which point it is fresh — thereby avoid extra queue check.
On the accelerated greedy algorithm

- 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 next iteration at which point it is fresh — thereby avoid extra queue check.

- Note, we can avoid keeping track of fresh/stale bit and always do an unconditional evaluation at the top of each loop.
Given polymatroid $f$, goal is to find a covering set of minimum cost:

$$S^* \in \arg\min_{S \subseteq V} |S| \text{ such that } f(S) \geq \alpha$$  \hspace{1cm} (14.19)

where $\alpha$ is a “cover” requirement.
Given polymatroid $f$, goal is to find a covering set of minimum cost:

$$S^* \in \arg\min_{S \subseteq V} |S| \text{ such that } f(S) \geq \alpha$$  \hspace{1cm} (14.19)

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:

$$S^* \in \arg\min_{S \subseteq V} |S| \text{ such that } f'(S) \geq f'(V)$$  \hspace{1cm} (14.20)
(Minimum) Submodular Set Cover

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

$$S^* \in \text{argmin} \{ |S| \text{ such that } f(S) \geq \alpha \} \quad (14.19)$$

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:

$$S^* \in \text{argmin} \{ |S| \text{ such that } f'(S) \geq f'(V) \} \quad (14.20)$$

- Note that this immediately generalizes standard set cover, in which case $f(A)$ is the cardinality of the union of sets indexed by $A$. 
Given polymatroid $f$, goal is to find a covering set of minimum cost:

$$S^* \in \arg\min_{S \subseteq V} |S| \text{ such that } f(S) \geq \alpha$$  \hspace{1cm} (14.19)

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:

$$S^* \in \arg\min_{S \subseteq V} |S| \text{ such that } f'(S) \geq f'(V)$$  \hspace{1cm} (14.20)

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

$$|S^G| \leq |S^*| H(\max_{s \in V} f(\{s\})) = |S^*| O(\log_e(\max_{s \in V} f(\{s\})))$$ \hspace{1cm} (14.21)

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

If $f$ is not integer valued, then bounds we get are of the form:

$$|S^G| \leq |S^*| \cdot 1 + \log_e f(\{s\}) f(\{s\}) f(S_T^1) \cdot \ldots \cdot (14.22)$$

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

$$|S^G| \leq |S^*|H(\max_{s \in V} f(\{s\})) = |S^*|O(\log_e(\max_{s \in V} f(\{s\}))) \quad (14.21)$$

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:

$$|S^G| \leq |S^*|(1 + \log_e \frac{f(V)}{f(V) - f(S^G_{T-1})}) \quad (14.22)$$

where $S^G_{T}$ is the greedy solution that occurs at step $T$, where $T$ is the number of iterations the algorithm runs until threshold is reached.
For integer valued \( f \), this greedy algorithm has an \( O(\log(\max_{s \in V} f(\{s\}))) \) approximation. Let \( S^* \) be optimal, and \( S^G \) be greedy solution, then

\[
|S^G| \leq |S^*|H(\max_{s \in V} f(\{s\})) = |S^*|O(\log_e(\max_{s \in V} f(\{s\})))
\] (14.21)

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:

\[
|S^G| \leq |S^*|\left(1 + \log_e \frac{f(V)}{f(V) - f(S_{T-1})}\right)
\] (14.22)

where \( S_T \) is the greedy solution that occurs at step \( T \), where \( T \) is the number of iterations the algorithm runs until threshold is reached.

As we mentioned earlier, even set cover (a special case of submodular 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 $OPT = f(S^*)$. By submodularity, we will show:

$$\exists v \in V \setminus S_i: f(v|S_i) = f(S_i + v|S_i) \geq \frac{1}{k}(OPT - f(S_i)) \tag{14.2}$$

Equation (14.11) will show that Equation (14.2) $\Rightarrow$:

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

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

$$\leq 1/e OPT$$

$$\Rightarrow OPT(1 - 1/e) \leq f(S_k)$$
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:

$$E[f(a_{i+1}|A_i)] \geq \frac{f(OPT) - f(A_i)}{k} \quad (14.23)$$

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

$$E[f(a_{i+1}|A_i)] \geq \frac{f(OPT) - f(A_i)}{k}$$  \hspace{1cm} (14.23)

where $\mathcal{A}_i = (a_1, a_2, \ldots, a_i)$ are the first $i$ elements chosen by the strategy.

Will be on Homework 4.
Curvature of a Submodular function

For any submodular function, we have $f(j|S) \leq f(j|\emptyset)$ so that $f(j|S)/f(j|\emptyset) \leq 1$ whenever $f(j|\emptyset) \neq 0$.

$$f(\emptyset) > 0$$

$$0 = f(\emptyset) \geq f(\emptyset|S) \geq 0$$

$$< 0$$
Curvature of a Submodular function

- For any submodular function, we have $f(j|S) \leq f(j|\emptyset)$ so that $f(j|S)/f(j|\emptyset) \leq 1$ whenever $f(j|\emptyset) \neq 0$.

- For $f : 2^V \rightarrow \mathbb{R}_+$ (non-negative) functions, we also have that $f(j|S)/f(j|\emptyset) \geq 0$. We get $f(j|S)/f(j|\emptyset) = 0$ whenever $j$ is “spanned” by the set $S$, using a “matroid span” concept.

$$f(j|S) = 0$$

$\emptyset \in \text{span}(S)$
Curvature of a Submodular function

- For any submodular function, we have $f(j|S) \leq f(j|\emptyset)$ so that $f(j|S)/f(j|\emptyset) \leq 1$ whenever $f(j|\emptyset) \neq 0$.
- For $f : 2^V \to \mathbb{R}_+$ (non-negative) functions, we also have that $f(j|S)/f(j|\emptyset) \geq 0$. We get $f(j|S)/f(j|\emptyset) = 0$ whenever $j$ is “spanned” by the set $S$, using a “matroid span” concept.
- The total curvature of a submodular function is defined as follows:

$$c_f \overset{\Delta}{=} 1 - \min_{S,j \notin S : f(j|\emptyset) \neq 0} \frac{f(j|S)}{f(j|\emptyset)} = 1 - \min_{j : f(j) \neq 0} \frac{f(j|V \setminus j)}{f(j)}$$  \hspace{1cm} (14.24)

$$\min_{s \subseteq V \setminus j \neq S} \frac{f(\emptyset|s)}{f(\emptyset)} = \min_{\emptyset \subseteq S} \frac{f(\emptyset|V \setminus S)}{f(\emptyset)}$$
Curvature of a Submodular function

- For any submodular function, we have $f(j|S) \leq f(j|\emptyset)$ so that $f(j|S)/f(j|\emptyset) \leq 1$ whenever $f(j|\emptyset) \neq 0$.

- For $f: 2^V \to \mathbb{R}_+$ (non-negative) functions, we also have that $f(j|S)/f(j|\emptyset) \geq 0$. We get $f(j|S)/f(j|\emptyset) = 0$ whenever $j$ is “spanned” by the set $S$, using a “matroid span” concept.

- The total curvature of a submodular function is defined as follows:

  $$c_f \triangleq 1 - \min_{S, j \notin S: f(j|\emptyset) \neq 0} \frac{f(j|S)}{f(j|\emptyset)} = 1 - \min_{j: f(j) \neq 0} \frac{f(j|V \setminus j)}{f(j)}$$

  (14.24)

- Thus, $c_f \in [0, 1]$. Let's consider the extremes:
Curvature of a Submodular function

- For any submodular function, we have \( f(j|S) \leq f(j|\emptyset) \) so that \( f(j|S)/f(j|\emptyset) \leq 1 \) whenever \( f(j|\emptyset) \neq 0 \).
- For \( f : 2^V \rightarrow \mathbb{R}_+ \) (non-negative) functions, we also have that \( f(j|S)/f(j|\emptyset) \geq 0 \). We get \( f(j|S)/f(j|\emptyset) = 0 \) whenever \( j \) is “spanned” by the set \( S \), using a “matroid span” concept.
- The total curvature of a submodular function is defined as follows:

\[
 c_f \overset{\Delta}{=} 1 - \min_{S,j \notin S: f(j|\emptyset) \neq 0} \frac{f(j|S)}{f(j|\emptyset)} = 1 - \min_{j: f(j) \neq 0} \frac{f(j|V \setminus j)}{f(j)} \tag{14.24}
\]

- Thus, \( c_f \in [0, 1] \). Let’s consider the extremes:
- When \( c_f = 0 \), \( f(j|S) = f(j|\emptyset) \) for all \( S,j \), a sufficient condition for modularity. Greedy is optimal for a modular function (i.e., optimal for \( \max_{A \subseteq V: |A| \leq k} m(A) \)). Hence, whenever \( c_f = 0 \) greedy is optimal.
Curvature of a Submodular function

- For any submodular function, we have $f(j|S) \leq f(j|\emptyset)$ so that $f(j|S)/f(j|\emptyset) \leq 1$ whenever $f(j|\emptyset) \neq 0$.

- For $f : 2^V \to \mathbb{R}_+$ (non-negative) functions, we also have that $f(j|S)/f(j|\emptyset) \geq 0$. We get $f(j|S)/f(j|\emptyset) = 0$ whenever $j$ is “spanned” by the set $S$, using a “matroid span” concept.

- The total curvature of a submodular function is defined as follows:

$$
c_f \overset{\Delta}{=} 1 - \min_{S,j \notin S : f(j|\emptyset) \neq 0} \frac{f(j|S)}{f(j|\emptyset)} = 1 - \min_{j : f(j) \neq 0} \frac{f(j|V \setminus j)}{f(j)} \tag{14.24}
$$

- Thus, $c_f \in [0, 1]$. Let's consider the extremes:

- When $c_f = 0$, $f(j|S) = f(j|\emptyset)$ for all $S, j$, a sufficient condition for modularity. Greedy is optimal for a modular function (i.e., optimal for $\max_{A \subseteq V : |A| \leq k} m(A)$). Hence, whenever $c_f = 0$ greedy is optimal.

- For a matroid rank function, we might have $c_r = 1$. E.g., $r(A) = \min(|A|, n - 1)$ has $r(j|V \setminus j)/r(j) = 0$. Theorem 9.5.1 said that greedy is optimal for max weight indep. set of a matroid.
Curvature of a Submodular function

- For any submodular function, we have $f(j|S) \leq f(j|\emptyset)$ so that $f(j|S)/f(j|\emptyset) \leq 1$ whenever $f(j|\emptyset) \neq 0$.
- For $f : 2^V \to \mathbb{R}_+$ (non-negative) functions, we also have that $f(j|S)/f(j|\emptyset) \geq 0$. We get $f(j|S)/f(j|\emptyset) = 0$ whenever $j$ is “spanned” by the set $S$, using a “matroid span” concept.

The total curvature of a submodular function is defined as follows:

$$c_f \triangleq 1 - \min_{S,j \notin S : f(j|\emptyset) \neq 0} \frac{f(j|S)}{f(j|\emptyset)} = 1 - \min_{j : f(j) \neq 0} \frac{f(j|V \setminus j)}{f(j)}$$ (14.24)

- Thus, $c_f \in [0,1]$. Lets consider the extremes:
- When $c_f = 0$, $f(j|S) = f(j|\emptyset)$ for all $S, j$, a sufficient condition for modularity. Greedy is optimal for a modular function (i.e., optimal for $\max_{A \subseteq V : |A| \leq k} m(A)$). Hence, whenever $c_f = 0$ greedy is optimal.
- For a matroid rank function, we might have $c_r = 1$. E.g.,
  $r(A) = \min(|A|, n-1)$ has $r(j|V \setminus j) / r(j) = 0$. Theorem 9.5.1 said that greedy is optimal for max weight indep. set of a matroid.
- Thus far, curvature extremes doesn’t tell us much. However ...
Curvature bound

For $f$ with curvature $c_f$, then $\forall A \subseteq V, \forall v \notin a, \forall c' \geq c_f$:

$$f(v \mid A) = f(A + v) - f(A) \geq (1 - c')f(v)$$  (14.25)
For $f$ with curvature $c_f$, then $\forall A \subseteq V, \forall v \notin A, \forall c' \geq c_f$:

$$f(A + v) - f(A) \geq (1 - c')f(v)$$  \hspace{1cm} (14.25)

This is because

$$f(v) \geq f(v|A) = f(v) \frac{f(v|A)}{f(v)} \geq f(v) \min_{v'} \frac{f(v'|A)}{f(v')}$$  \hspace{1cm} (14.26)

$$= (1 - c_f)f(v) \geq (1 - c')f(v)$$  \hspace{1cm} (14.27)
Curvature bound

For $f$ with curvature $c_f$, then $\forall A \subseteq V, \forall v \notin a, \forall c' \geq c_f$:

$$f(A + v) - f(A) \geq (1 - c')f(v) \quad (14.25)$$

This is because

$$f(v) \geq f(v|A) = f(v) \frac{f(v|A)}{f(v)} \geq f(v) \min_{v'} \frac{f(v'|A)}{f(v')} \quad (14.26)$$

$$= (1 - c_f)f(v) \geq (1 - c')f(v) \quad (14.27)$$

Hence both upper and lower bound: $f(v) \geq f(v|A) \geq (1 - c_f)f(v)$. 
Curvature bound

For $f$ with curvature $c_f$, then $\forall A \subseteq V, \forall v \notin a, \forall c' \geq c_f$:

$$f(A + v) - f(A) \geq (1 - c')f(v) \quad (14.25)$$

This is because

$$f(v) \geq f(v|A) = f(v)\frac{f(v|A)}{f(v)} \geq f(v)\min_{v'} \frac{f(v'|A)}{f(v')}$$

$$= (1 - c_f)f(v) \geq (1 - c')f(v) \quad (14.26)$$

Hence both upper and lower bound: $f(v) \geq f(v|A) \geq (1 - c_f)f(v)$.

When $c_f = 1$ then the submodular function said to be “maximally” (or fully, or totally) curved, i.e., there exists a subset that fully spans some other element.
Curvature of a Submodular function

- Curvature definition again (by submodularity, both forms are the same):

\[
c_f \triangleq 1 - \min_{S, j \notin S: f(j|\emptyset) \neq 0} \frac{f(j|S)}{f(j|\emptyset)} = 1 - \min_{j: f(j|\emptyset) \neq 0} \frac{f(j|V \setminus \{j\})}{f(j|\emptyset)} \tag{14.28}
\]
Curvature of a Submodular function

Curvature definition again (by submodularity, both forms are the same):

\[
c_f \triangleq 1 - \min_{S,j \not\in S : f(j|\emptyset) \neq 0} \frac{f(j|S)}{f(j|\emptyset)} = 1 - \min_{j : f(j|\emptyset) \neq 0} \frac{f(j|V \setminus \{j\})}{f(j|\emptyset)} \tag{14.28}
\]

Note: Matroid rank is either modular \( c_r = 0 \) or maximally curved \( c_r = 1 \). Thus, matroid rank can have only the extreme points of curvature, namely 0 or 1.
Curvature of a Submodular function

- Curvature definition again (by submodularity, both forms are the same):

\[ c_f \triangleq 1 - \min_{S,j \notin S : f(j|\emptyset) \neq 0} \frac{f(j|S)}{f(j|\emptyset)} = 1 - \min_{j : f(j|\emptyset) \neq 0} \frac{f(j|V \setminus \{j\})}{f(j|\emptyset)} \] (14.28)

- Note: Matroid rank is either modular \( c_r = 0 \) or maximally curved \( c_r = 1 \). thus, matroid rank can have only the extreme points of curvature, namely 0 or 1.

- Polymatroid functions are, however, more nuanced, in that they allow non-extreme curvature, with \( c_f \in (0,1) \).
Curvature of a Submodular function

- Curvature definition again (by submodularity, both forms are the same):

\[ c_f \overset{\Delta}{=} 1 - \min_{S, j \not\in S: f(j|\emptyset) \neq 0} \frac{f(j|S)}{f(j|\emptyset)} = 1 - \min_{j: f(j|\emptyset) \neq 0} \frac{f(j|V \setminus \{j\})}{f(j|\emptyset)} \quad (14.28) \]

- Note: Matroid rank is either modular \( c_r = 0 \) or maximally curved \( c_r = 1 \). Thus, matroid rank can have only the extreme points of curvature, namely 0 or 1.

- Polymatroid functions are, however, more nuanced, in that they allow non-extreme curvature, with \( c_f \in (0, 1) \).

- Recall the notion of “partial dependence” within polymatroid functions.
Curvature and approximation: key theorem

- Curvature-based approximation bound for max $k$ cardinality.

Theorem 14.4.1
Given a polymatroid function $f: 2^V \to \mathbb{R}_+$ with curvature $c_f \in [0, 1]$ defined above. Then the greedy algorithm's solution to the problem

$$\max_{A \in 2^V : |A| \leq k} f(A)$$

has the following approximation bound:

$$\frac{1}{c_f}(1 - e^{c_f}) \frac{f(\tilde{S}_{\text{greedy}})}{\text{OPT}} \leq 1$$

Equation (14.30)
Curvature and approximation: key theorem

- Curvature-based approximation bound for max $k$ cardinality.

**Theorem 14.4.1**

*Given a polymatroid function $f : 2^V \rightarrow \mathbb{R}_+$ with curvature $c_f \in [0, 1]$ defined above. Then the greedy algorithm’s solution to the problem $\max_{A \subseteq V : |A| \leq k} f(A)$ has the following approximation bound:*

\[
\frac{1}{c_f} (1 - e^{-c_f})
\]

(14.29)

\[
\begin{align*}
c_f = 1 & \Rightarrow 1 - e^{-1} \\
\lim_{c_f \to 0} & \frac{1}{c_f} (1 - e^{-c_f}) = 1
\end{align*}
\]
Curvature and approximation: key theorem

- Curvature-based approximation bound for max \( k \) cardinality.

**Theorem 14.4.1**

*Given a polymatroid function \( f : 2^V \rightarrow \mathbb{R}^+ \) with curvature \( c_f \in [0, 1] \) defined above. Then the greedy algorithm’s solution to the problem \( \max_{A \subseteq V : |A| \leq k} f(A) \) has the following approximation bound:*  

\[
\frac{1}{c_f} \left( 1 - e^{-c_f} \right) \tag{14.29}
\]

\[
f(\tilde{S}_{\text{greedy}}) \geq \frac{1}{c_f} \left( 1 - e^{-c_f} \right) \text{OPT} \tag{14.30}
\]
Example: curvature for \( f(S) = \sqrt{|S|} \)

Curvature of \( f(S) = \sqrt{|S|} \) as function of \( |V| = n \)

- \( f(S) = \sqrt{|S|} \) with \( |V| = n \) has curvature \( 1 - (\sqrt{n} - \sqrt{n-1}) \).

![Graph showing curvature of sqrt(|A|) as function of |V| = n](image-url)
Example: curvature for \( f(S) = \sqrt{|S|} \)

Curvature of \( f(S) = \sqrt{|S|} \) as function of \( |V| = n \)

- \( f(S) = \sqrt{|S|} \) with \( |V| = n \) has curvature 1 - (\( \sqrt{n} - \sqrt{n-1} \)).
- Approximation gets worse with bigger ground set.
Example: curvature for $f(S) = \sqrt{|S|}$

Curvature of $f(S) = \sqrt{|S|}$ as function of $|V| = n$

- $f(S) = \sqrt{|S|}$ with $|V| = n$ has curvature $1 - (\sqrt{n} - \sqrt{n-1})$.
- Approximation gets worse with bigger ground set.
- Functions of the form $f(S) = \sqrt{m(S)}$ where $m : V \rightarrow \mathbb{R}_+$, approximation worse with $n$ if $\min_{i \neq j} |m(i) - m(j)|$ has a fixed lower bound with increasing $n$. 

\[
\min_{\sim} \frac{f(n/|V|v)}{f(n)}
\]
Example: curvature for $f(S) = \sqrt{|S|}$

Curvature of $f(S) = \sqrt{|S|}$ as function of $|V| = n$

- $f(S) = \sqrt{|S|}$ with $|V| = n$ has curvature $1 - (\sqrt{n} - \sqrt{n-1})$.
- Approximation gets worse with bigger ground set.
- Functions of the form $f(S) = \sqrt{m(S)}$ where $m : V \rightarrow \mathbb{R}_+$, approximation worse with $n$ if $\min_{i \neq j} |m(i) - m(j)|$ has a fixed lower bound with increasing $n$.
- Quite good news for greedy and submodular functions.
Submodular and Supermodular Curvature Approximation

Let $f$ be a polymatroid function and let $g$ be a non-negative monotone non-decreasing supermodular function (e.g., $g(A) = \phi(m(A))$ where $\phi()$ is non-decreasing convex and $m: V \rightarrow \mathbb{R}_+$).
Let $f$ be a polymatroid function and let $g$ be a non-negative monotone non-decreasing supermodular function (e.g., $g(A) = \phi(m(A))$ where $\phi()$ is non-decreasing convex and $m : V \rightarrow \mathbb{R}_+$).  

Let $\kappa_f = 1 - \min_v \frac{f(v|V\backslash\{v\})}{f(v)}$ be the submodular total curvature,
Let $f$ be a polymatroid function and let $g$ be a non-negative monotone non-decreasing supermodular function (e.g., $g(A) = \phi(m(A))$ where $\phi()$ is non-decreasing convex and $m : V \to \mathbb{R}_+$).

Let $\kappa_f = 1 - \min_v \frac{f(v|V\backslash\{v\})}{f(v)}$ be the submodular total curvature,

Define $\kappa_g = 1 - \min_v \frac{g(v)}{g(v|V\backslash\{v\})}$ as a “supermodular curvature”
Let $f$ be a polymatroid function and let $g$ be a non-negative monotone non-decreasing supermodular function (e.g., $g(A) = \phi(m(A))$ where $\phi()$ is non-decreasing convex and $m : V \rightarrow \mathbb{R}_+$).

Let $\kappa_f = 1 - \min_v \frac{f(v|V\backslash\{v\})}{f(v)}$ be the submodular total curvature,

Define $\kappa^g = 1 - \min_v \frac{g(v)}{g(v|V\backslash\{v\})}$ as a “supermodular curvature”

$\kappa^g \in [0, 1]$ and $\kappa^g = 0$ means $g$ is modular, $\kappa^g = 1$ means $g$ is “fully curved”
Let $f$ be a polymatroid function and let $g$ be a non-negative monotone non-decreasing supermodular function (e.g., $g(A) = \phi(m(A))$ where $\phi()$ is non-decreasing convex and $m : V \rightarrow \mathbb{R}_+$).

Let $\kappa_f = 1 - \min_v \frac{f(v|V\backslash\{v\})}{f(v)}$ be the submodular total curvature,

Define $\kappa^g = 1 - \min_v \frac{g(v)}{g(v|V\backslash\{v\})}$ as a “supermodular curvature”

$\kappa^g \in [0, 1]$ and $\kappa^g = 0$ means $g$ is modular, $\kappa^g = 1$ means $g$ is “fully curved”

Form function $h(A) = f(A) + g(A)$, then $h$ is neither submodular nor supermodular, but is known as a BP-function.
We have the following:
We have the following:

**Theorem 14.4.2**

*Given a polymatroid function* $f : 2^V \to \mathbb{R}_+$ *with curvature* $\kappa_f \in [0, 1]$ *and a non-negative monotone non-decreasing supermodular function* $g : 2^V \to \mathbb{R}_+$ *with curvature* $\kappa_g$, *and* $h = f + g$. *Then the greedy algorithm’s solution to the problem* $\max_{A \subseteq V : |A| \leq k} h(A)$ *has the following approximation bound:*

$$\frac{1}{\kappa_f} \left(1 - e^{- (1 - \kappa_g) \kappa_f}\right)$$  (14.31)
We have the following:

**Theorem 14.4.2**

Given a polymatroid function $f : 2^V \to \mathbb{R}_+$ with curvature $\kappa_f \in [0, 1]$ and a non-negative monotone non-decreasing supermodular function $g : 2^V \to \mathbb{R}_+$ with curvature $\kappa_g$, and $h = f + g$. Then the greedy algorithm’s solution to the problem $\max_{A \subseteq V : |A| \leq k} h(A)$ has the following approximation bound:

$$\frac{1}{\kappa_f} \left(1 - e^{-(1-\kappa_g)\kappa_f}\right)$$

For purely supermodular optimization (i.e., $\kappa_f = 0$) we get that greedy has a guarantee of $1 - \kappa_g$. 
We have the following:

**Theorem 14.4.2**

Given a polymatroid function $f : 2^V \rightarrow \mathbb{R}_+$ with curvature $\kappa_f \in [0, 1]$ and a non-negative monotone non-decreasing supermodular function $g : 2^V \rightarrow \mathbb{R}_+$ with curvature $\kappa_g$, and $h = f + g$. Then the greedy algorithm’s solution to the problem $\max_{A \subseteq V : |A| \leq k} h(A)$ has the following approximation bound:

$$\frac{1}{\kappa_f} \left( 1 - e^{-\left( 1 - \kappa_g \right) \kappa_f} \right)$$

(14.31)

- For purely supermodular optimization (i.e., $\kappa_f = 0$) we get that greedy has a guarantee of $1 - \kappa_g$.
- Both curvatures are very easy to compute given given BP decomposition.
BP functions are an example of when quality of solutions to non-submodular problems can be analyzed via submodularity since BP functions are neither submodular nor supermodular.
Submodular Analysis for Non-Submodular Problems

- BP functions are an example of when quality of solutions to non-submodular problems can be analyzed via submodularity since BP functions are neither submodular nor supermodular.

- Another example: “deviation from submodularity” can be measured using the submodularity ratio (Das & Kempe) that we saw in HW1:

\[
\gamma_{U,k}(f) \triangleq \min_{L \subseteq U, S : |S| \leq k, S \cap L = \emptyset} \frac{\sum_{S \in S} f(x|L)}{f(S|L)}
\]  

(14.32)
Submodular Analysis for Non-Submodular Problems

- BP functions are an example of when quality of solutions to non-submodular problems can be analyzed via submodularity since BP functions are neither submodular nor supermodular.

- Another example: “deviation from submodularity” can be measured using the submodularity ratio (Das & Kempe) that we saw in HW1:

\[
\gamma_{U,k}(f) \triangleq \min_{L \subseteq U, |S| \leq k, S \cap L = \emptyset} \frac{\sum_{s \in S} f(x | L)}{f(S | L)}
\]

(14.32)

- \( f \) is submodular if and only if \( \gamma_{V,|V|} = 1 \).
BP functions are an example of when quality of solutions to non-submodular problems can be analyzed via submodularity since BP functions are neither submodular nor supermodular.

Another example: “deviation from submodularity” can be measured using the submodularity ratio (Das & Kempe) that we saw in HW1:

$$\gamma_{U,k}(f) \triangleq \min_{L \subseteq U, S : |S| \leq k, S \cap L = \emptyset} \frac{\sum_{S \in S} f(x|L)}{f(S|L)}$$  \hspace{1cm} (14.32)

- $f$ is submodular if and only if $\gamma_{V,|V|} = 1$.
- For some variable selection problems, can get bounds of the form:

$$\text{Solution} \geq (1 - \frac{1}{e^{\gamma_{U^*,k}}}) \text{OPT}$$  \hspace{1cm} (14.33)

where $U^*$ is the solution set of a variable selection algorithm.
Submodular Analysis for Non-Submodular Problems

- BP functions are an example of when quality of solutions to non-submodular problems can be analyzed via submodularity since BP functions are neither submodular nor supermodular.
- Another example: “deviation from submodularity” can be measured using the submodularity ratio \( \gamma_{U,k}(f) \) (Das & Kempe) that we saw in HW1:

\[
\gamma_{U,k}(f) \triangleq \min_{L \subseteq U, S \mid |S| \leq k, S \cap L = \emptyset} \frac{\sum_{s \in S} f(x \mid L)}{f(S \mid L)}
\]

(14.32)

- \( f \) is submodular if and only if \( \gamma_{V,|V|} = 1 \).
- For some variable selection problems, can get bounds of the form:

\[
\text{Solution} \geq (1 - \frac{1}{e \gamma_{U^*,k}}) \text{OPT}
\]

(14.33)

where \( U^* \) is the solution set of a variable selection algorithm.
- This gradually get worse as we move away from an objective being submodular (see Das & Kempe, 2011).
Submodular Analysis for Non-Submodular Problems

- BP functions are an example of when quality of solutions to non-submodular problems can be analyzed via submodularity since BP functions are neither submodular nor supermodular.

- Another example: “deviation from submodularity” can be measured using the submodularity ratio \( \gamma_{U,k}(f) \) (Das & Kempe) that we saw in HW1:

\[
\gamma_{U,k}(f) \triangleq \min_{L \subseteq U, S: |S| \leq k, S \cap L = \emptyset} \frac{\sum_{S \in S} f(x | L)}{f(S | L)}
\]  

(14.32)

- \( f \) is submodular if and only if \( \gamma_{V,|V|} = 1 \).

- For some variable selection problems, can get bounds of the form:

\[
\text{Solution} \geq (1 - \frac{1}{e^{\gamma_{U^*,k}}})\text{OPT}
\]

(14.33)

where \( U^* \) is the solution set of a variable selection algorithm.

- This gradually get worse as we move away from an objective being submodular (see Das & Kempe, 2011).

- Another analogous concepts, submodular degree.