Submodular Functions, Optimization, and Applications to Machine Learning

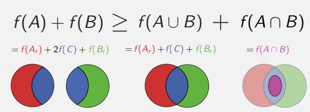
— Spring Quarter, Lecture 19 —

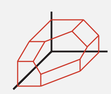
http://j.ee.washington.edu/~bilmes/classes/ee596b_spring_2014/

Prof. Jeff Bilmes

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

June 4th, 2014





Cumulative Outstanding Reading

- Good references for today: Schrijver-2003, Oxley-1992/2011, Welsh-1973, Goemans-2010, Cunningham-1984, Edmonds-1969, Choquet-1955, Grabisch/Marichal/Mesiar/Pap “Aggregation Functions”, Lovász-1983, Bach-2011.
- Read Tom McCormick’s overview paper on SFM http://people.commerce.ubc.ca/faculty/mccormick/sfmchap8a.pdf
- Read chapters 1 - 4 from Fujishige book.
- Matroid properties http://www-math.mit.edu/~goemans/18433S09/matroid-notes.pdf
- Read lecture 14 slides on lattice theory at our web page (http://j.ee.washington.edu/~bilmes/classes/ee596b_spring_2014/)
- Wolfe “Finding the Nearest Point in a Polytope”, 1976.
- Fujishige & Isotani, “A Submodular Function Minimization Algorithm Based on the Minimum-Norm Base”, 2009.

Sources for Today's Lecture

- "Submodular Function Maximization", Krause and Golovin.
- Chekuri, Vondrak, Zenklusen, "Submodular Function Maximization via the Multilinear Relaxation and Contention Resolution Schemes", 2011 (a recent paper (appeared yesterday) that, among other things, has a nice up-to-date summary on all the results on submodular max).
- Minoux, "Accelerated Greedy Algorithms for Maximizing Submodular Set Functions", 1977.
- Feige, Mirrokni, Vondrak, "Maximizing non-monotone submodular functions", 2007.
- Fujishige, "Submodular Functions and Optimization", 2005.
- Fujishige, "Submodular Systems and Related Topics", 1984.
- Fisher, Nemhauser, Wolsey, "An Analysis of Approximations for Maximizing Submodular Set Functions - II", 1978.
- Lin & Bilmes, "A Class Of Submodular Functions for Document Summarization", 2011.

Other readings

- J. Vondrak, "Submodularity and curvature: the optimal algorithm" in RIMS Kokyuroku Bessatsu B23, Kyoto, 2010.
- M. Conforti and G. Cornuéjols. Submodular set functions, matroids and the greedy algorithm: tight worst-case bounds and some generalizations of the Rado-Edmonds theorem. Discrete Applied Math, 7(3):251-274, 1984.

Announcements, Assignments, and Reminders

- Weekly Office Hours: Wednesdays, 5:00-5:50, or by skype or google hangout (email me).

Class Road Map - IT-I

- L1 (3/31): Motivation, Applications, & Basic Definitions
- L2: (4/2): Applications, Basic Definitions, Properties
- L3: More examples and properties (e.g., closure properties), and examples, spanning trees
- L4: proofs of equivalent definitions, independence, start matroids
- L5: matroids, basic definitions and examples
- L6: More on matroids, System of Distinct Reps, Transversals, Transversal Matroid, Matroid and representation
- L7: Dual Matroids, other matroid properties, Combinatorial Geometries
- L8: Combinatorial Geometries, matroids and greedy, Polyhedra, Matroid Polytopes,
- L9: From Matroid Polytopes to Polymatroids.
- L10: Polymatroids and Submodularity
- L11: More properties of polymatroids, SFM special cases
- L12: polymatroid properties, extreme points polymatroids,
- L13: sat, dep, supp, exchange capacity, examples
- L14: Lattice theory: partially ordered sets; lattices; distributive, modular, submodular, and boolean lattices; ideals and join irreducibles.
- L15: Supp, Base polytope, polymatroids and entropic Venn diagrams, exchange capacity,
- L16: proof that minimum norm point yields min of submodular function, and the lattice of minimizers of a submodular function, Lovasz extension
- L17: Lovasz extension, Choquet Integration, more properties/examples of Lovasz extension, convex minimization and SFM.
- L18: Lovasz extension examples and structured convex norms, The Min-Norm Point Algorithm detailed.
- L19: symmetric submodular function minimization, maximizing monotone submodular function w. card constraints.
- L20: maximizing monotone submodular function w. other constraints, non-monotone maximization.

Finals Week: June 9th-13th, 2014.

Symmetric Submodular Functions

- Given: $\breve{f} : 2^E \to \mathbb{R}$, if $\breve{f}$ is submodular and also has the property that $\breve{f}(A) = \breve{f}(E \setminus A)$ for all A, then $\breve{f}$ is said to be symmetric submodular
- Given any non-symmetric submodular function f, we can always symmetrize it, $f_{\text{symmetric}}(A) = f(A) + f(E \setminus A)$.
- *Symmetrize* and *normalize* f as $f \to \breve{f}$ via the operation: $\breve{f}(A) = f(A) + f(E \setminus A) - f(E)$, so that $\breve{f}(\emptyset) = 0$ if $f(\emptyset) = 0$.
- Such an $\breve{f}$ is also non-negative since

$$2\breve{f}(A) = \breve{f}(A) + \breve{f}(E \setminus A) \geq \breve{f}(\emptyset) + \breve{f}(E) = 2\breve{f}(\emptyset) \geq 0 \quad (19.1)$$

- Equivalence class: $f \to \breve{f}$ same up to modular shift since $\breve{f} = \breve{g}$ if $f = g + m$ with m modular $\Rightarrow$ consider only polymatroidal f.
- Combinatorial mutual information function, so $\breve{f}(A) = I_f(A; V \setminus A)$ where $I_f(A; B) = f(A) + f(B) - f(A \cup B) - f(A \cap B)$.
- Example: $f(A) = H(X_A) =$ entropy, then $\breve{f} = I(X_A; X_{E \setminus A}) =$ symmetric mutual information.

Separators of submodular function via symmetrized version

- Such a symmetrized submodular function measures a form of "dependence" between A and $\bar{A} \triangleq E \setminus A$.

Theorem 19.3.1

We are given an f that is normalized & submodular. If $\exists A$ s.t. $\breve{f}(A) \triangleq f(A) + f(\bar{A}) - f(E) = 0$ then f is "decomposable" w.r.t. A — this means $f(B) = f(B \cap A) + f(B \cap \bar{A})$, $\forall B$.

Proof.

- By submodularity (subadditivity for non-intersecting sets), we have:

$$f(B) = f\Big((B \cap A) \cup (B \cap \bar{A})\Big) \leq f(B \cap A) + f(B \cap \bar{A}) \quad (19.2)$$

- Hence, $f(B) \leq f(B \cap A) + f(B \cap \bar{A})$.

...

Separators of submodular function via symmetrized version

. . . proof of Theorem 19.3.1 cont.

- By submodularity

$$f(B) - f(B \cap A) - f(B \cap \bar{A}) \geq f(A \cup B) - f(A) - f(B \cap \bar{A}) \quad (19.3)$$
$$\geq f((A \cup B) \cup \bar{A}) - f(A) - f(\bar{A}) \quad (19.4)$$
$$= f(E) - f(A) + f(\bar{A}) = 0 \quad (19.5)$$

- Eqn. (19.3) follows since $f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$, and Eqn. (19.4) follows since $B \cap \bar{A} = (A \cup B) \cap \bar{A}$ and $f(A \cup B) + f(\bar{A}) \geq f((A \cup B) \cup \bar{A}) + f((A \cup B) \cap \bar{A})$.
- Hence, both $f(B) \geq f(B \cap A) + f(B \cap \bar{A})$ (from above) and $f(B) \leq f(B \cap A) + f(B \cap \bar{A})$ (previous slide).

□

Separators of submodular function via symmetrized version

- Again, let $\breve{f}$ be the symmetrized version of f.
- Definition: If $\breve{f}(A) = 0$, then any $A' \subseteq A$ and $\bar{A}' \subseteq E \setminus A$ are “independent” w.r.t. submodular g, and A is called a separator.
- random variables: $X_A \perp\!\!\!\perp X_B \Rightarrow X_{A'} \perp\!\!\!\perp X_{B'} \;\forall\; A' \subseteq A$ and $B' \subseteq B$.
- Set of separators of $\breve{f}$ is closed under intersection, union, and complementation. Hence, the separators partition E.
- In following slides, $\breve{f}$ is symmetrized & normalized version of f.

Review

Next slide is from Lecture 4.

Many (Equivalent) Definitions of Submodularity

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B), \ \ \forall A, B \subseteq V \tag{19.6}$$

$$f(j|S) \geq f(j|T), \ \forall S \subseteq T \subseteq V, \text{ with } j \in V \setminus T \tag{19.7}$$

$$f(C|S) \geq f(C|T), \forall S \subseteq T \subseteq V, \text{ with } C \subseteq V \setminus T \tag{19.8}$$

$$f(j|S) \geq f(j|S \cup \{k\}), \ \forall S \subseteq V \text{ with } j \in V \setminus (S \cup \{k\}) \tag{19.9}$$

$$f(A \cup B|A \cap B) \leq f(A|A \cap B) + f(B|A \cap B), \ \ \forall A, B \subseteq V \tag{19.10}$$

$$f(T) \leq f(S) + \sum_{j \in T \setminus S} f(j|S) - \sum_{j \in S \setminus T} f(j|S \cup T - \{j\}), \ \forall S, T \subseteq V \tag{19.11}$$

$$f(T) \leq f(S) + \sum_{j \in T \setminus S} f(j|S), \ \forall S \subseteq T \subseteq V \tag{19.12}$$

$$f(T) \leq f(S) - \sum_{j \in S \setminus T} f(j|S \setminus \{j\}) + \sum_{j \in T \setminus S} f(j|S \cap T) \ \forall S, T \subseteq V \tag{19.13}$$

$$f(T) \leq f(S) - \sum_{j \in S \setminus T} f(j|S \setminus \{j\}), \ \forall T \subseteq S \subseteq V \tag{19.14}$$

Minimization of a Symmetric Submodular Functions

- Minimizing symmetric submodular functions can be done in strongly polynomial time $O(n^3)$. The algorithm by Nagamochi & Ibaracki 1992 for graph cuts shown by Queyranne in 1995 to work for sym. SFM.
- The algorithm finds (as a subroutine) MA (maximum adjacency) or a maximum back orders (not same as greedy order).

```
Choose v_1 arbitrarily ;
W_1 ← (v_1)                      /* The first of an ordered list W_i. */ ;
for i ← 1 ... |V| − 1 do
    Choose v_{i+1} ∈ argmin_{u ∈ V \ W_i} f(W_i | {u}) ;
    W_{i+1} ← (W_i, v_{i+1}) ; /* Append v_{i+1} to end of W_i */
```

- Note algorithm operates on non-symmetric function f. If f is already symmetric and normalized, then $f = \breve{f}$.
- The final ordered set $W_n = (v_1, v_2, \dots, v_n)$ is special in that the last two nodes (v_{n-1}, v_n) serve as a surrogate minimizer for a special case.

Pendent pair

- A ordered pair of elements (t, u) is called a pendent pair if u is a minimizer amongst all sets that separate u and t.
- That is (t, u) is a pendent pair if

$$\{u\} \in \operatorname*{argmin}_{A \subseteq V : u \in A, t \notin A} \breve{f}(A) \tag{19.6}$$

- That is,

$$\breve{f}(\{u\}) \leq \breve{f}(A) \;\; \forall A \text{ s.t. } t \notin A \ni u \tag{19.7}$$

Theorem 19.3.2

In the ordered set $W = (v_1, \dots, v_n)$ generated by the MA algorithm, then (v_{n-1}, v_n) is a pendent pair.

- Interestingly, this algorithm is the same as maximum cardinality search (MCS), when f represents a graph cut function (recall, MCS is used to efficiently test graph chordality).

Minimization of a Symmetric Submodular Functions

- Now, given a pendent pair (t, u) there are two cases.
- Either: The global minimizer, say X^* of $\breve{f}$ is such that $t \notin X^* \ni u$ or we, by symmetry, can w.l.o.g. choose the minimizer so that both $\{t, u\} \in X^*$.
- We store the score (min value) in the first case, then, consider a new element "tu" and clustered ground set $V' = V \setminus \{t, u\} \cup \{tu\}$, and new symmetric submodular function $f' : 2^{V'} \to \mathbb{R}$ with

$$\breve{f}'(X) = \begin{cases} \breve{f}(X) & \text{if } tu \notin X \\ \breve{f}(X \cup \{t, u\} \setminus \{tu\}) & \text{if } tu \in X \end{cases} \tag{19.8}$$

- We then find a new pendent pair on f' using the above algorithm, store the new min value, and merge, and repeat.
- We do this n times. We take the min over all of the stored values.
- The pendent pair corresponding to the min element, say (t', u') will (most probability) correspond to nested clusters, so we use the original ground elements corresponding to u'.

Minimization of a Symmetric Submodular Functions

Theorem 19.3.3

The final resultant u' when expanded to original ground elements minimizes the symmetric submodular function f in $O(n^3)$ time.

- This has become known as Queyranne's algorithm for symmetric submodular function minimization.
- This was done in 1995 and it is said that this result, at that time, rekindled the efforts to find general combinatorial SFM.
- The actual algorithm was originally developed by Nagamochi and Ibaraki for a simple algorithm for finding graph cut. Queyranne showed it worked for any symmetric submodular function.
- Hence, it seems reasonable that symmetric SFM is faster than general SFM (although this question is still unknown).
- Quoting Fujishige from NIPS 2012, he said that he "hopes general purpose SFM is $O(n^4)$" ☺.

Maximization of Submodular Functions

- We spent much time on submodular function minimization (SFM) and saw this can be done in polynomial time.
- Submodular maximization is also 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).
- For polymatroid function (or any monotone non-decreasing function), unconstrained maximization is trivial (take ground set).
- Thus, when we do monotone submodular maximization, we either
  - Find the maximum under some constraint
  - Find the maximum for a non-polymatroid submodular function
  - Do both.
- 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 ground 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.,

$$\text{minimize}|X| \text{ subject to } f(X) \geq |E| \tag{19.9}$$

- 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).

The Max k-Cover Problem

- Let E be a ground set and let $E_1, E_2, \dots, E_m$ be a set of subsets.
- Let $V = \{1, 2, \dots, m\}$ 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:

$$\max f(X) \text{ subject to } |X| \leq k \tag{19.10}$$

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

Cardinality Constrained Max. of Polymatroid Functions

- Now we are given an arbitrary polymatroid function f.
- Given k, goal is: find $A^* \in \operatorname{argmax}\{f(A) : |A| \leq k\}$
- w.l.o.g., we can find $A^* \in \operatorname{argmax}\{f(A) : |A| = k\}$
- 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 $i = 0 \ldots (k-1)$:

$$S_{i+1} = S_i \cup \left\{ \underset{v \in V \setminus S_i}{\operatorname{argmax}} f(S_i \cup \{v\}) \right\} \tag{19.11}$$

The Greedy Algorithm for Submodular Max

A bit more precisely:

Algorithm 2: The Greedy Algorithm

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

The Greedy Algorithm for Submodular Max

- This algorithm has a guarantee

Theorem 19.4.1

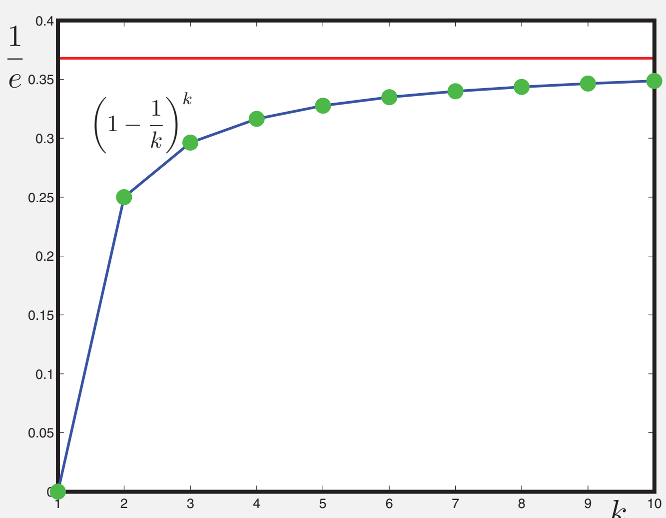
Given a polymatroid function f, the above greedy algorithm returns sets S_i such that for each i we have $f(S_i) \geq (1 - 1/e) \max_{|S| \leq i} f(S)$.

- To find $A^* \in \operatorname{argmax}\{f(A) : |A| \leq k\}$, we repeat the greedy step until $k = i + 1$:
- 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 $f(v|S_i)$.
- Let S^* be optimal solution (of size k) and OPT $= f(S^*)$. By submodularity, we will show:

$$\exists v \in S^* \setminus S_i : f(S_i + v|S_i) \geq \frac{1}{k}(\text{OPT} - f(S_i)) \tag{19.12}$$



Equation (19.21) will show that Equation (19.12) $\Rightarrow$:

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

Cardinality Constrained Polymatroid Max Theorem

Theorem 19.4.2 (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. (19.11)). Then for all $k, \ell \in \mathbb{Z}_{++}$, we have:

$$f(S_\ell) \geq (1 - e^{-\ell/k}) \max_{S:|S| \leq k} f(S) \quad (19.13)$$

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

Cardinality Constrained Polymatroid Max Theorem

Proof of Theorem 19.4.2.

- Fix ℓ (number of items greedy will chose) and k (size of optimal set to compare against).
- Set $S^* \in \operatorname{argmax} \{f(S) : |S| \leq k\}$
- 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 19.4.2 cont.

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

$$f(S^*) \leq f(S^* \cup S_i) \tag{19.14}$$

$$= f(S_i) + \sum_{j=1}^{k} f(v_j^*|S_i \cup \{v_1^*, v_2^*, \ldots, v_{j-1}^*\}) \tag{19.15}$$

$$\leq f(S_i) + \sum_{v \in S^*} f(v|S_i) \tag{19.16}$$

$$\leq f(S_i) + \sum_{v \in S^*} f(v_{i+1}|S_i) = f(S_i) + \sum_{v \in S^*} f(S_{i+1}|S_i) \tag{19.17}$$

$$= f(S_i) + kf(S_{i+1}|S_i) \tag{19.18}$$

- Therefore, we have Equation 19.12, i.e.,:

$$f(S^*) - f(S_i) \leq kf(S_{i+1}|S_i) = k(f(S_{i+1}) - f(S_i)) \tag{19.19}$$

...

Cardinality Constrained Polymatroid Max Theorem

... proof of Theorem 19.4.2 cont.

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

$$\delta_i \leq k(\delta_i - \delta_{i+1}) \tag{19.20}$$

or

$$\delta_{i+1} \leq (1 - \frac{1}{k})\delta_i \tag{19.21}$$

- The relationship between δ_0 and δ_ℓ is then

$$\delta_l \leq (1 - \frac{1}{k})^\ell \delta_0 \tag{19.22}$$

- 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

$$\delta_\ell \leq (1 - \frac{1}{k})^\ell \delta_0 \leq e^{-\ell/k} f(S^*) \tag{19.23}$$

...

Cardinality Constrained Polymatroid Max Theorem

. . . proof of Theorem 19.4.2 cont.

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

$$f(S_\ell) \geq (1 - e^{-\ell/k}) f(S^*) \tag{19.24}$$

□

- 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 $\lceil -\ln(1 - 0.999) \rceil = 7$.
- 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.
- 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 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 (typically $n \log n$) 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.
- 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

$$f(v'|S_{i+1}) \geq \alpha_v = f(v|S_i) \geq f(v|S_{i+1}) \tag{19.25}$$

  we have the true max, and we need not re-evaluate gains of other elements again.
- Strategy is: find the $\operatorname{argmax}_{v' \in V \setminus S_{i+1}} \alpha_{v'}$, and then compute the real $f(v'|S_{i+1})$. If it is greater than all other α_v's then that's the next greedy step. Otherwise, replace $\alpha_{v'}$ with its real value, resort, and repeat.

Minoux's Accelerated Greedy for Submodular Functions

- Minoux's algorithm is exact, in that it has the same guarantees as does the $O(n^2)$ greedy Algorithm 2 (this means it will return either the same answers, or answers that have 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 $O(n + k)$, 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 used for "big data" sets (e.g., social networks, selecting blogs of greatest influence, document summarization, etc.).

Priority Queue

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

$$\textsc{insert}(Q, (v, \alpha)) \tag{19.26}$$

  - Pop the item (v, α) with maximum value α off the queue.

$$(v, \alpha) \leftarrow \textsc{pop}(Q) \tag{19.27}$$

  - Query the value of the max item in the queue

$$\textsc{max}(Q) \in \mathbb{R} \tag{19.28}$$

- On next slide, we call a popped item "fresh" if the value (v, α) 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.

Minoux's Accelerated Greedy Algorithm Submodular Max

Algorithm 3: Minoux's Accelerated Greedy Algorithm

```
1  Set S_0 ← ∅ ; i ← 0 ; Initialize priority queue Q ;
2  for v ∈ E do
3      INSERT(Q, f(v))
4  repeat
5      (v, α) ← POP(Q) ;
6      if α not "fresh" then
7          recompute α ← f(v|S_i)
8      if (popped α in line 5 was "fresh") OR (α ≥ MAX(Q)) then
9          Set S_{i+1} ← S_i ∪ {v} ;
10         i ← i + 1 ;
11     else
12         INSERT(Q, (v, α))
13 until i = |E|;
```

Minimum Submodular Cover

- Given polymatroid f, goal is to find a covering set of minimum cost:
$$S^* \in \operatorname*{argmin}_{S \subseteq V} |S| \text{ such that } f(S) \geq \alpha \tag{19.29}$$
where α is a "cover" requirement.
- Normally take $\alpha = f(V)$ but defining $f'(A) = \min\{f(A), \alpha\}$ we can take any α. Hence, we have equivalent formulation:
$$S^* \in \operatorname*{argmin}_{S \subseteq V} |S| \text{ such that } f'(S) \geq f'(V) \tag{19.30}$$
- Note that this immediately generalizes standard set cover, in which case $f(A)$ is the cardinality of the union of sets indexed by A.
- Algorithm: Pick the first S_i chosen by aforementioned greedy algorithm such that $f(S_i) \geq \alpha$.
- For integer valued f, this greedy algorithm an $O(\log(\max_{s \in V} f(\{s\})))$ approximation. Set cover is hard to approximate with a factor better than $(1-\epsilon)\log \alpha$, where α is the desired cover 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.
- Simple greedy algorithm gets $1 - e^{-\ell/k}$ approximation, where k is size of optimal set we compare against, and ℓ is size of set greedy algorithm chooses.
- Submodular cover: min. $|S|$ s.t. $f(S) \geq \alpha$.
- Minoux's accelerated greedy trick.

Generalizations

- Consider a k-uniform matroid $\mathcal{M} = (V, \mathcal{I})$ where $\mathcal{I} = \{S \subseteq V : |S| \leq k\}$, and consider problem $\max\{f(A) : A \in \mathcal{I}\}$
- Hence, the greedy algorithm is $1 - 1/e$ optimal for maximizing polymatroidal f subject to a k-uniform matroid constraint.
- Might be useful to allow an arbitrary matroid (e.g., partition matroid $\mathcal{I} = \{X \subseteq V : |X \cap V_i| \leq k_i$ for all $i = 1, \ldots, \ell\}$., or a transversal, etc).
- Knapsack constraint: if each item $v \in V$ has a cost $c(v)$, we may ask for $c(S) \leq b$ where b is a budget, in units of costs. Q: Is $\mathcal{I} = \{I : c(I) \leq b\}$ the independent sets of a matroid?
- We may wish to maximize f subject to multiple matroid constraints. I.e., $S \in \mathcal{I}_1, S \in \mathcal{I}_2, \ldots, S \in \mathcal{I}_p$ where $\mathcal{I}_i$ are independent sets of the i^{th} matroid.
- Combinations of the above (e.g., knapsack & multiple matroid constraints).

Greedy over multiple matroids

- Obvious heuristic is to use the greedy step but always stay feasible.
- I.e., Starting with $S_0 = \emptyset$, we repeat the following greedy step

$$S_{i+1} = S_i \cup \left\{ \operatorname*{argmax}_{v \in V \setminus S_i \,:\, S_i + v \in \bigcap_{i=1}^{p} \mathcal{I}_i} f(S_i \cup \{v\}) \right\} \tag{19.31}$$

- That is, we keep choosing next whatever feasible element looks best.
- This algorithm is simple and also has a guarantee

Theorem 19.5.1

Given a polymatroid function f, and set of matroids $\{M_j = (E, \mathcal{I}_j)\}_{j=1}^{p}$, the above greedy algorithm returns sets S_i such that for each i we have $f(S_i) \geq \frac{1}{p+1} \max_{|S| \leq i, S \in \bigcap_{i=1}^{p} \mathcal{I}_i} f(S)$, assuming such sets exists.

- For one matroid, we have a 1/2 approximation.
- Very easy algorithm, Minoux trick still possible, while addresses multiple matroid constraints — but the bound is not that good when there are many matroids.