Logistics

 announcements, assignments, and reminders

Please do use our discussion board 
(https://catalyst.uw.edu/gopost/board/bilmes/29948/) for all questions, comments, so that all will benefit from them being answered.

Today is makeup lecture.
Second makeup lecture is Thursday, 1-3pm
Final presentations Thursday, 3-5pm.
Cumulative Outstanding Reading

- Read overview of submodular function maximization
- Read “A Tight Linear Time (1/2)-Approximation for Unconstrained Submodular Maximization” by Niv Buchbinder and Moran Feldman and Joseph (Seffi) Naor and Roy Schwartz, in FOCS 2012
  http://theory.stanford.edu/~tim/focs12/.
- Read Tom McCormick’s overview paper on SFM
  http://people.commerce.ubc.ca/faculty/mccormick/sfmchap8a.pdf
- Read chapters 1 - 5 from Fujishige book.
- Read over lecture slides, all posted on our web page (http://j.ee.washington.edu/~bilmes/classes/ee596a_fall_2012/).
- See the summary slide at the end of each lectures for additional reading sources.

On Final Projects

- We will have final project presentations, Thursday Dec 13th.
- Final projects can be something related to your own research, or just the presentation of 2 or more papers that relate strongly to submodularity.
- Each presentation should be about 20 minutes.
- You are also to turn in a no more than 4 page write up on your project (conference style, with short abstract).
On Final Projects: deadlines

All due via our dropbox (https://catalyst.uw.edu/collectit/dropbox/bilmes/23873):

- By Wednesday Dec 12th: Final 4 page write up (see our dropbox (https://catalyst.uw.edu/collectit/dropbox/bilmes/23873))
- Time/place of presentations Thursday, TBD (they’ll be combined with final lecture of the class, goal is to have two lectures the week of Dec 10th).
Submodular functions

Submodular function minimization: Given general submodular function \( f : 2^E \to \mathbb{R} \), find largest set of sets \( \{A_1, A_2, \ldots, A_k\} \) such that for any \( i \in \{1, \ldots, k\} \), we have

\[
f(A_k) = \min_{A \subseteq E} f(A)
\]

We saw that \( \{A_1, A_2, \ldots, A_k\} \) forms a distributive lattice, and that there were multiple ways of getting this: (ellipsoid algorithm, finding the min-norm point and all dep/sat functions, combinatorial algorithms that we outlined the flavor of).

What about symmetric submodular functions?

What about submodular function maximization?
Symmetric Submodular Functions

- Given: \( \tilde{f} : 2^E \rightarrow \mathbb{R} \), if \( \tilde{f} \) is submodular and also has the property that \( \tilde{f}(A) = \tilde{f}(E \setminus A) \) for all \( A \), then \( \tilde{f} \) 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 \( \bar{f} \) via the operation:
  \[
  \bar{f}(A) = f(A) + f(E \setminus A) - f(E),
  \]
  so that \( \bar{f}(\emptyset) = 0 \) if \( f(\emptyset) = 0 \).
- Such an \( \bar{f} \) is also non-negative since
  \[
  2\bar{f}(A) = \bar{f}(A) + \bar{f}(E \setminus A) \geq \bar{f}(\emptyset) + \bar{f}(E) = 2\bar{f}(\emptyset) \geq 0 \tag{18.2}
  \]
- Equivalence class: \( f \rightarrow \bar{f} \) same up to modular shift since \( \bar{f} = \tilde{g} \) if \( f = g + m \) with \( m \) modular \( \Rightarrow \) consider only polymatroidal \( f \).
- Example: \( f(A) = H(X_A) = \) entropy, then \( \bar{f} = I(X_A; X_E \setminus A) = \) symmetric mutual information.

Separators of submodular function via symmetrized version

- Such a symmetrized submodular function measures a form of “dependence” between \( A \) and \( \bar{A} \triangleq E \setminus A \) even when

**Theorem 18.3.1**

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

**Proof.**

- By submodularity (subadditivity for non-intersecting sets), we have:
  \[
  f(B) = f((B \cap A) \cup (B \cap \bar{A})) \leq f(B \cap A) + f(B \cap \bar{A}) \tag{18.3}
  \]
- Hence, \( f(B) \leq f(B \cap A) + f(B \cap \bar{A}) \).

...
Tip/Side note: On submodular bounds

- Note: Given submodular \( f \), and given you have \( C, D \subseteq E \) with either \( D \supseteq C \) or \( D \subseteq C \), and have an expression of the form:

  \[
  f(C) - f(D) \tag{18.7}
  \]

- If you can find an \( X \) such that \( D = C \cup X \) then

  \[
  f(C) - f(D) = f(C) - f(C \cup X) \geq f(C \cap X) - f(X) \tag{18.8}
  \]
  or

  \[
  f(C \cup X | C) \leq f(X | C \cap X) \tag{18.9}
  \]

- Alternatively, if you can find an \( Y \) such that \( D = C \cap Y \) then

  \[
  f(C) - f(D) = f(C) - f(C \cap Y) \geq f(C \cup Y) - f(Y) \tag{18.10}
  \]
  or

  \[
  f(C | C \cap Y) \geq f(C \cup Y | Y) \tag{18.11}
  \]

- The various definitions of submodular from lecture 3, repeated again next, will also be remembered.
Many (Equivalent) Definitions of Submodularity

\[ f(A) + f(B) \geq f(A \cup B) + f(A \cap B), \quad \forall A, B \subseteq E \]  
(18.18)

\[ f(j|S) \geq f(j|T), \quad \forall S \subseteq T \subseteq E, \quad \text{with } j \in E \setminus T \]  
(18.19)

\[ f(C|S) \geq f(C|T), \quad \forall S \subseteq T \subseteq E, \quad \text{with } C \subseteq E \setminus T \]  
(18.20)

\[ f(j|S) \geq f(j|S \cup \{k\}), \quad \forall S \subseteq E \text{ with } j \in E \setminus (S \cup \{k\}) \]  
(18.21)

\[ f(T) \leq f(S) + \sum_{j \in T \setminus S} f(j|S) - \sum_{j \in S \setminus T} f(j|S \cup T - \{j\}), \quad \forall S, T \subseteq E \]  
(18.22)

\[ f(T) \leq f(S) + \sum_{j \in T \setminus S} f(j|S), \quad \forall S \subseteq T \subseteq E \]  
(18.23)

\[ f(T) \leq f(S) - \sum_{j \in S \setminus T} f(j|S \setminus \{j\}) + \sum_{j \in T \setminus S} f(j|S \cap T) \quad \forall S, T \subseteq E \]  
(18.24)

\[ f(T) \leq f(S) - \sum_{j \in S \setminus T} f(j|S \setminus \{j\}), \quad \forall T \subseteq S \subseteq E \]  
(18.25)

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).

1. Choose \(v_1\) arbitrarily ;
2. \(W_1 \leftarrow (v_1)\);
3. for \(i \leftarrow 1 \ldots |V| - 1\) do
   4. Choose \(v_{i+1} \in \text{argmin}_{u \in V \setminus W_i} f(W_i|\{u\})\);
   5. \(W_{i+1} \leftarrow (W_i, v_{i+1})\) ; /* 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 \(f = \hat{f}\).
- The final ordered set \(W_n = (v_1, v_2, \ldots, v_n)\) 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

$$\{u\} \in \operatorname{argmin} \left\{ \tilde{f}(A) : u \in A, t \notin A \right\} \quad (18.12)$$

- That is,

$$\tilde{f}(\{u\}) \leq \tilde{f}(A) \quad \forall A \text{ s.t. } t \notin A \ni u \quad (18.13)$$

**Theorem 18.3.2**

In the ordered set $W = (v_1, \ldots, v_n)$ 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 minimizer, say $X^*$ of $\tilde{f}$ 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 $\tilde{f}' : 2^{V'} \to \mathbb{R}$ with

$$\tilde{f}'(X) = \begin{cases} \tilde{f}(X) & \text{if } tu \notin X \\ \tilde{f}(X \cup \{t, u\} \setminus \{tu\}) & \text{if } tu \in X \end{cases} \quad (18.14)$$

- We then find a new pendent pair on $\tilde{f}'$ 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 $(t', u')$ most probably will correspond to clusters, so we use the original ground elements corresponding to $u'$. 
Minimization of a Symmetric Submodular Functions

**Theorem 18.3.3**

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 last week (at 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.,

$$\begin{align*}
\text{minimize } |X| \text{ subject to } f(X) \geq |E| \quad (18.15)
\end{align*}$$

- 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 $(1 - 1/e) \log n$ unless NP is slightly superpolynomial ($n^{O(\log \log n)}$).

The Max $k$-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 $f(X) = |\bigcup_{v \in V} E_v|$.
- 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:

$$\begin{align*}
\max f(X) \text{ subject to } |X| \leq k \quad (18.16)
\end{align*}$$

- This problem is NP-hard, and Feige in 1998 showed that it cannot be approximated with a ratio better than $(1 - 1/e)$. 
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 $i = 0 \ldots (k - 1)$:

$$S_{i+1} = S_i \cup \left\{ \arg\max_{v \in V \setminus S_i} f(S_i \cup \{v\}) \right\} \quad (18.17)$$

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. \hspace{1em} Choose $v_i$ as follows:
   $$v_i \in \left\{ \arg\max_{v \in V \setminus S_i} f(\{v\}\cup S_i) \right\} = \left\{ \arg\max_{v \in V \setminus S_i} f(S_i \cup \{v\}) \right\} ;$$
4. \hspace{1em} Set $S_{i+1} \leftarrow S_i \cup \{v_i\}$;
The Greedy Algorithm for Submodular Max

- This algorithm has a guarantee

**Theorem 18.4.1**

*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 \( A^* \in \text{argmax} \{ f(A) : |A| \leq k \} \), we repeat the greedy step until \( k = i + 1 \):
- Again, since this generalizes \( \text{max } k \)-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:

- 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 and the greedy choice, we will show:

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

(18.18)

\[
\text{OPT} - f(S_{i+1}) \\
\leq (1 - 1/k)(\text{OPT} - f(S_i)) \\
\Rightarrow \text{OPT} - f(S_k) \\
\leq (1 - 1/k)^k \text{OPT} \\
\leq 1/e \text{OPT} \\
\Rightarrow \text{OPT}(1 - 1/e) \leq f(S_k)
\]
### Theorem 18.4.2 (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. (18.17)). Then for all \( k, \ell \in \mathbb{Z}^+ \), we have:

\[
f(S_\ell) \geq (1 - e^{-\ell/k}) \max_{S:|S| \leq k} f(S) \tag{18.19}
\]

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

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

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

#### Proof of Theorem 18.4.2.

- 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 \( (v_1, v_2, \ldots, v_k) \) be the greedy order chosen by the algorithm.
- Then the following inequalities (on the next slide) follow:
Cardinality Constrained Polymatroid Max Theorem

...proof of Theorem 18.4.2 cont.

★ For all $i < \ell$, we have
\[ f(S^*) \leq f(S^* \cup S_i) \] (18.20)
\[ = f(S_i) + \sum_{j=1}^{k} f(v_j^* | S_i \cup \{v_1^*, v_2^*, \ldots, v_{j-1}^*\}) \] (18.21)
\[ \leq f(S_i) + \sum_{v \in S^*} f(v | S_i) \] (18.22)
\[ \leq f(S_i) + \sum_{v \in S^*} f(v | S_i) = f(S_i) + \sum_{v \in S^*} f(S_{i+1} | S_i) \] (18.23)
\[ = f(S_i) + kf(S_{i+1} | S_i) \] (18.24)
★ Therefore, we have
\[ f(S^*) - f(S_i) \leq kf(S_{i+1} | S_i) = k(f(S_{i+1}) - f(S_i)) \] (18.25)

Define $\delta_i \triangleq f(S^*) - f(S_i)$, so $\delta_i - \delta_{i+1} = f(S_{i+1}) - f(S_i)$, giving
\[ \delta_i \leq k(\delta_i - \delta_{i+1}) \] (18.26)
or
\[ \delta_{i+1} \leq (1 - \frac{1}{k})\delta_i \] (18.27)
★ The relationship between $\delta_0$ and $\delta_\ell$ is then
\[ \delta_\ell \leq (1 - \frac{1}{k})^\ell \delta_0 \] (18.28)
★ 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
\[ \delta_\ell \leq (1 - \frac{1}{k})^\ell \delta_0 \leq e^{-\ell/k}f(S^*) \] (18.29)
...proof of Theorem 18.4.2 cont.

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

$$f(S_l) \geq (1 - e^{-\ell/k})f(S^*) \quad (18.30)$$

★ 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 $|S^*| = k$? Set $.95 = (1 - e^{-\ell/k})$, 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 Accelerated Greedy strategy (and has been rediscovered a few times), 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 $f(v|S_{i+1}) \leq \alpha_v = f(v|S_i)$, we need not re-evaluate gain.
- Strategy is: find the $\arg\max_{v' \in V \setminus S_{i+1}} \alpha_{v'}$, and then compute the real $f(v'|S_{i+1})$. If it is greater than all other $\alpha_{v'}$’s then that’s the next greedy step. Otherwise, replace $\alpha_{v'}$ with its real value, resort, 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).
- 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 $(v, \alpha)$ into queue, with $v \in V$ and $\alpha \in \mathbb{R}$.
    
    $$\text{INSERT}(Q, (v, \alpha)) \quad (18.31)$$
  
  - Pop the item $(v, \alpha)$ with maximum value $\alpha$ off the queue.
    
    $$(v, \alpha) \leftarrow \text{POP}(Q) \quad (18.32)$$
  
  - Query the value of the max item in the queue
    
    $$\text{MAX}(Q) \in \mathbb{R} \quad (18.33)$$

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

Algorithm 3: Minoux’s Accelerated Greedy Algorithm Submodular Max

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 $(\text{popped } \alpha \text{ in line 5 was “fresh”}) \text{ OR } (\alpha \geq \text{MAX}(Q))$ then
9.     Set $S_{i+1} \leftarrow S_i \cup \{v\}$;
10.    $i \leftarrow i + 1$;
11. else
12.    INSERT$(Q, (v, \alpha))$
13. until $i = |E|$;
Minimum Submodular Cover

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

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

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:

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

(18.35)

- 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 \( \alpha \) is the desired cover constraint.

Generalizations

- There are a number of ways of generalizing these results.

- First, we can use more general constraints, rather than just cardinality constraints.

- Alternatively, we may wish to maximize non-monotone submodular functions. This includes of course graph cuts, and this problem is APX-hard, so maximizing non-monotone functions, even unconstrainedly, is hard.

- We may wish to do both.

- In either case, 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).
Submodular max with constraints

- Consider a set of sets $\mathcal{I}$ and one wishes to find $\max \{ f(A) : A \in \mathcal{I} \}$.
- In fact, if $M = (E, \mathcal{I})$ is a $k$-uniform matroid we saw in Lecture 3 ($\mathcal{I} = \{ A \subseteq E : |A| \leq k \}$), we have the cardinality constrained submodular max we just encountered.
- Might be useful to allow an arbitrary matroid (e.g., partition matroid $\mathcal{I} = \{ X \subseteq V : |X \cap V_i| \leq k_i \text{ for all } i = 1, \ldots, \ell \}$, or a transversal, etc).
- In fact, one could consider multiple matroids $M_1, M_2, \ldots, M_p$ and one may want a solution that is independent in all matroids. I.e., the constraint set is that $A \in \mathcal{I}_1 \cap \mathcal{I}_2 \cap \cdots \cap \mathcal{I}_p$.

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
  \[ S_{i+1} = S_i \cup \{ \text{argmax}_{v \in V \setminus S_i : S_i + v \in \bigcap_{i=1}^p \mathcal{I}_i} f(S_i \cup \{v\}) \} \]  

  (18.36)

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

**Theorem 18.5.1**

*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.*

- So this is a very easy algorithm to solve multiple matroid constraints, but the bound is not that good when there are many matroids.
Greedy over multiple matroids: Generalized Bipartite Matching

- Generalized bipartite matching (i.e., max bipartite matching with submodular costs on the edges). Use two partition matroids (as mentioned earlier in class).
- Useful in natural language processing: Ex. Computer language translation, find an alignment between two language strings.

Greedy over multiple matroids: Submodular Welfare

- Submodular Welfare Maximization: Consider $E$ a set of $m$ goods to be distributed/partitioned among $n$ people ("players").
- Each player has a submodular "valuation" function, $g_i : 2^E \rightarrow \mathbb{R}^+$ for a set of goods.
- Goal: Partition $E = E_1 \cup E_2 \cup \cdots \cup E_n$ to maximize $\sum_{i=1}^{n} g_i(E_i)$.
- Create new ground set $E'$ as disjoint union $E' = E \bigcup E \bigcup \cdots \bigcup E$ of original ground set. Let $E^{(i)} \subset E'$ be the $i^{th}$ block of $E'$.
- Create a 1-partition matroid $\mathcal{M} = (E', \mathcal{I})$ where $\mathcal{I} = \{S \subseteq E' : \forall i, |S \cap E^{(i)}| \leq 1\}$.
- Create submodular function $f' : 2^{E'} \rightarrow \mathbb{R}_+$ with $f'(S) = \sum_{i=1}^{n} g_i(S \cap E^{(i)})$.
- Submodular welfare maximization becomes matroid constrained submodular max \( \max \{f'(S) : S \in \mathcal{I}\} \), so greedy algorithm gives a $1/2$ approximation.
Submodular Welfare Figure

Suppose $n = 5$:

Monotone Submodular over Knapsack Constraint

- The constraint $|A| \leq k$ is a simple cardinality constraint.
- Consider a non-negative integral modular function $c : E \rightarrow \mathbb{Z}_+$.  
- A knapsack constraint would be of the form $c(A) \leq b$ where $B$ is some integer budget that must not be exceeded. That is max $\{f(A) : A \subseteq V, c(A) \leq b\}$.
- $c(e)$ may be seen as the cost of item $e$ and if $c(e) = 1$ for all $e$, then we recover the cardinality constraint we saw earlier.
Monotone Submodular over Knapsack Constraint

- Greedy can be seen as choosing the best gain: Starting with $S_0 = \emptyset$, we repeat the following greedy step

$$S_{i+1} = S_i \cup \left\{ \arg\max_{v \in V \setminus S_i} (f(S_i \cup \{v\}) - f(S_i)) \right\} \quad (18.37)$$

the gain is $f(\{v\}|S_i) = f(S_i + v) - f(S_i)$, so greedy just chooses next the currently unselected element with greatest gain.

- Core idea in knapsack case: Greedy can be extended to choose next whatever looks cost-normalized best, i.e., Starting some initial set $S_0$, we repeat the following cost-normalized greedy step

$$S_{i+1} = S_i \cup \left\{ \arg\max_{v \in V \setminus S_i} \frac{f(S_i \cup \{v\}) - f(S_i)}{c(v)} \right\} \quad (18.38)$$

which we repeat until $c(S_{i+1}) < b$ and then take $S_i$ as the solution.

A Knapsack Constraint

- There are a number of ways of getting approximation bounds using this strategy.
- If we run the normalized greedy procedure starting with $S_0 = \emptyset$, and compare the solution found with the max of the singletons $\max_{v \in V} f(\{v\})$, choosing the max, then we get a $(1 - e^{-1/2}) \approx 0.39$ approximation, in $O(n^2)$ time.
- On the other hand, we can get a $(1 - e^{-1}) \approx 0.63$ approximation in $O(n^5)$ time if we run the above procedure starting from all sets of cardinality three (so restart for all $S_0$ such that $|S_0| = 3$), and compare that with the best singleton and pairwise solution.
- Extending something similar to this to $d$ simultaneous knapsack constraints is possible as well.
What About Non-monotone

- If \( f \) is an arbitrary submodular function (so neither polymatroidal, nor necessarily positive or negative), then verifying if the maximum of \( f \) is positive or negative is already NP-hard.
- Therefore, submodular function max in such case is inapproximable unless P=NP (since any such procedure would give us the sign of the max).
- Thus, any approximation algorithm must be for unipolar submodular functions. E.g., non-negative but otherwise arbitrary submodular functions.
- It is possible to get a \( \left( \frac{1}{3} - \frac{\epsilon}{n} \right) \) approximation for maximizing non-monotone non-negative submodular functions, using an algorithm that uses at most \( O\left(\frac{1}{\epsilon}n^3 \log n\right) \) function calls.

Submodularity and local optima

- Given any submodular function \( f \), a set \( S \subseteq V \) is a local maximum of \( f \) if \( f(S - v) \leq f(S) \) for all \( v \in S \) and \( f(S + v) \leq f(S) \) for all \( v \in V \setminus S \).
- The following interesting result is true for any submodular function:

**Lemma 18.5.2**

*Given a submodular function \( f \), if \( S \) is a local optimum of \( f \), and \( I \subseteq S \) or \( I \supseteq S \), then \( f(I) \leq f(S) \).*

- In other words, once we have identified a local optimum, the two intervals in the Boolean lattice \([\emptyset, S]\) and \([S, V]\) can be ruled out as a possible improvement over \( S \).
- Finding a local optimum is already hard (PLS-complete), but it is possible to find an approximate local optimum relatively efficiently.
- This is the approach that yields the \( \left( \frac{1}{3} - \frac{\epsilon}{n} \right) \) approximation algorithm.
More general still: multiple constraints different types

- In the past several years, there has been a plethora of papers on maximizing both monotone and non-monotone submodular functions under various combinations of one or more knapsack and/or matroid constraints.
- The approximation quality is usually some function of the number of matroids, and is often not a function of the number of knapsacks.
- Often the computational costs of the algorithms are prohibitive (e.g., exponential in \( k \)) with large constants, so these algorithms might not scale.
- On the other hand, these algorithms offer deep and interesting intuition into submodular functions, beyond what we have covered here.

Submodular Max and polyhedral approaches

- We’ve spent much time discussing SFM and the polymatroidal polytope, and in general polyhedral approaches for SFM.
- Most of the approaches for submodular max have not used such an approach.
- A paper by Chekuri, Vondrak, and Zenklusen (2011) make some progress on this front using multilinear extensions.
Multilinear extension

**Definition 18.5.3**

For a set function \( f : 2^V \rightarrow \mathbb{R} \), define its **multilinear extension** \( F : [0, 1]^V \rightarrow \mathbb{R} \) by

\[
F(x) = \sum_{S \subseteq V} f(S) \prod_{i \in S} x_i \prod_{j \in V \setminus S} (1 - x_j)
\]  

(18.39)

- Note that \( F(x) = Ef(\hat{x}) \) where \( \hat{x} \) is a random binary vector over \( \{0, 1\}^V \) with elements independent w. probability \( x_i \) for \( \hat{x}_i \).
- While this is defined for any set function, we have:

**Lemma 18.5.4**

Let \( F : [0, 1]^V \rightarrow \mathbb{R} \) be multilinear extension of set function \( f : 2^V \rightarrow \mathbb{R} \), then

- If \( f \) is monotone non-decreasing, then \( \frac{\partial F}{\partial x_i} \geq 0 \) for all \( i \in V, x \in [0, 1]^V \).
- If \( f \) is submodular, then \( \frac{\partial^2 F}{\partial x_i \partial x_j} \leq 0 \) for all \( i, j \in V, x \in [0, 1]^V \).

Moreover, we have

**Lemma 18.5.5**

Let \( F : [0, 1]^V \rightarrow \mathbb{R} \) be multilinear extension of set function \( f : 2^V \rightarrow \mathbb{R} \), then

- If \( f \) is monotone non-decreasing, then \( F \) is non-decreasing along any line of direction \( d \in \mathbb{R}^E \) with \( d \geq 0 \)
- If \( f \) is submodular, then \( F \) is concave along any line of direction \( d \geq 0 \), and is convex along any line of direction \( \mathbf{1}_v - \mathbf{1}_w \) for any \( v, w \in V \).

Another connection between submodularity and convexity/concavity
- but note, unlike the Lovász extension, this function is neither.
Submodular Max and polyhedral approaches

- Basic idea: Given a set of constraints \( \mathcal{I} \), we form a polytope \( P_\mathcal{I} \) such that \( \{1_I : I \in \mathcal{I}\} \subseteq P_\mathcal{I} \)
- We find \( \max_{x \in P_\mathcal{I}} F(x) \) where \( F(x) \) is the multi-linear extension of \( f \), to find a fractional solution \( x^* \)
- We then round \( x^* \) to a point on the hypercube, thus giving us a solution to the discrete problem.

In the recent paper by Chekuri, Vondrak, and Zenklusen, they show:

1) constant factor approximation algorithm for \( \max \{F(x) : x \in P\} \)
   for any down-monotone solvable polytope \( P \) and \( F \) multilinear extension of any non-negative submodular function.
2) A randomized rounding scheme to obtain an integer solution
3) An optimal \( (1 - 1/e) \) instance of their rounding scheme that can be used for a variety of interesting independence systems, including \( O(1) \) knapsacks, \( k \) matroids and \( O(1) \) knapsacks, a \( k \)-matchoid and \( \ell \) sparse packing integer programs, and unsplittable flow in paths and trees.
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).
- Jegelka & Bilmes, ”Approximate Probabilistic Inference via Generalized Graph Cuts”, 2011.