Submodular Functions, Optimization, and Applications to Machine Learning

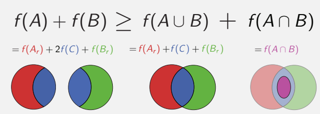
— Spring Quarter, Lecture 15 —

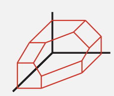
http://www.ee.washington.edu/people/faculty/bilmes/classes/ee596b_spring_2016/

Prof. Jeff Bilmes

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

May 23rd, 2016





Cumulative Outstanding Reading

- Read chapters 2 and 3 from Fujishige's book.
- Read chapter 1 from Fujishige's book.

Announcements, Assignments, and Reminders

- Homework 4, available at our assignment dropbox (`https://canvas.uw.edu/courses/1039754/assignments`), due (electronically) Wednesday (5/25) at 11:55pm.
- Homework 3, available at our assignment dropbox (`https://canvas.uw.edu/courses/1039754/assignments`), due (electronically) Monday (5/2) at 11:55pm.
- Homework 2, available at our assignment dropbox (`https://canvas.uw.edu/courses/1039754/assignments`), due (electronically) Monday (4/18) at 11:55pm.
- Homework 1, available at our assignment dropbox (`https://canvas.uw.edu/courses/1039754/assignments`), due (electronically) Friday (4/8) at 11:55pm.
- Weekly Office Hours: Mondays, 3:30-4:30, or by skype or google hangout (set up meeting via our our discussion board (`https://canvas.uw.edu/courses/1039754/discussion_topics`)).

Class Road Map - IT-I

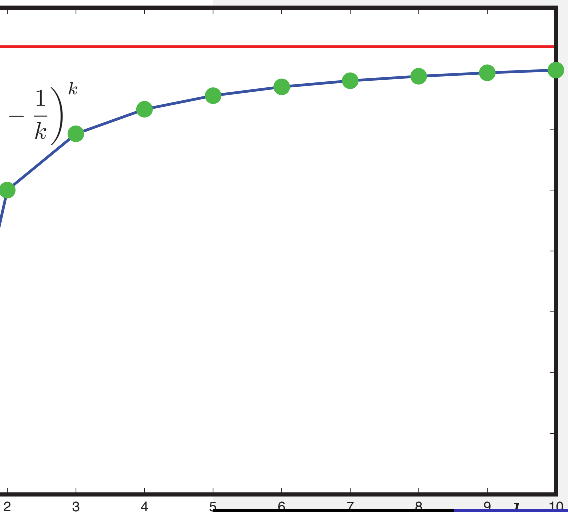
- L1(3/28): Motivation, Applications, & Basic Definitions
- L2(3/30): Machine Learning Apps (diversity, complexity, parameter, learning target, surrogate).
- L3(4/4): Info theory exs, more apps, definitions, graph/combinatorial examples, matrix rank example, visualization
- L4(4/6): Graph and Combinatorial Examples, matrix rank, Venn diagrams, examples of proofs of submodularity, some useful properties
- L5(4/11): Examples & Properties, Other Defs., Independence
- L6(4/13): Independence, Matroids, Matroid Examples, matroid rank is submodular
- L7(4/18): Matroid Rank, More on Partition Matroid, System of Distinct Reps, Transversals, Transversal Matroid,
- L8(4/20): Transversals, Matroid and representation, Dual Matroids,
- L9(4/25): Dual Matroids, Properties, Combinatorial Geometries, Matroid and Greedy
- L10(4/27): Matroid and Greedy, Polyhedra, Matroid Polytopes,
- L11(5/2): From Matroids to Polymatroids, Polymatroids
- L12(5/4): Polymatroids, Polymatroids and Greedy
- L13(5/9): Polymatroids and Greedy; Possible Polytopes; Extreme Points; Polymatroids, Greedy, and Cardinality Constrained Maximization
- L14(5/11): Cardinality Constrained Maximization; Curvature; Submodular Max w. Other Constraints
- L15(5/16): Submodular Max w. Other Constraints, Most Violated $\leq$, Matroids cont., Closure/Sat,
- L16(5/18): Closure/Sat, Fund. Circuit/Dep, Min-Norm Point and SFM, Min-Norm Point Algorithm, Proof that min-norm gives optimal.
- L17(5/23):
- L18(5/25):
- L19(6/1):
- L20(6/6): Final Presentations maximization.

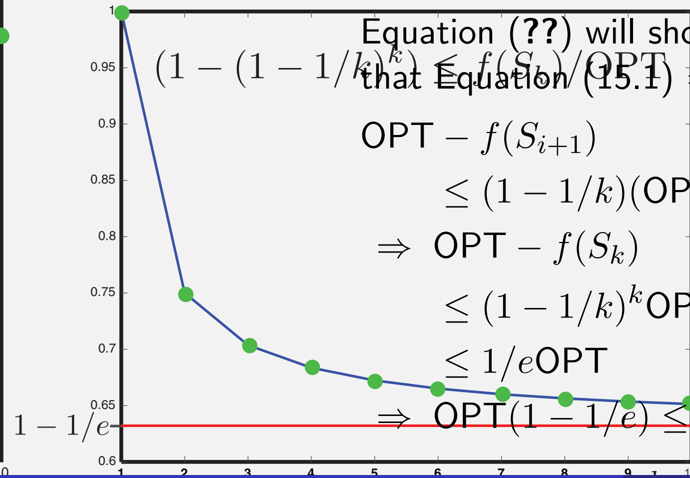
Finals Week: June 6th-10th, 2016.

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:

$$\exists v \in V \setminus S_i : f(v|S_i) = f(S_i + v|S_i) \geq \frac{1}{k}(\text{OPT} - f(S_i)) \tag{15.1}$$





Equation (**??**) will show that Equation (15.1) $\Rightarrow$:

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

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 $\alpha \in \mathbb{R}$.

$$\textsc{insert}(Q, (v, \alpha)) \tag{15.14}$$

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

$$(v, \alpha) \leftarrow \textsc{pop}(Q) \tag{15.15}$$

  - Query the value of the max item in the queue

$$\textsc{max}(Q) \in \mathbb{R} \tag{15.16}$$

- 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 Set Cover

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

$$S^* \in \underset{S \subseteq V}{\operatorname{argmin}} |S| \text{ such that } f(S) \geq \alpha \tag{15.14}$$

  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:

$$S^* \in \underset{S \subseteq V}{\operatorname{argmin}} |S| \text{ such that } f'(S) \geq f'(V) \tag{15.15}$$

- 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.

(Minimum) Submodular Set Cover: Approximation Analysis

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

$$|S^{\mathsf{G}}| \leq |S^*| H(\max_{s\in V} f(\{s\})) = |S^*| O(\log_e(\max_{s\in V} f(\{s\}))) \qquad (15.14)$$

  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:

$$|S^{\mathsf{G}}| \leq |S^*| \Big(1 + \log_e \frac{f(V)}{f(V) - f(S_{T-1})}\Big) \qquad (15.15)$$

  wehre S_T is the final greedy solution that occurs at step T.
- Set cover is hard to approximate with a factor better than $(1-\epsilon)\log \alpha$, where α is the desired cover constraint.

Curvature of a Submodular function

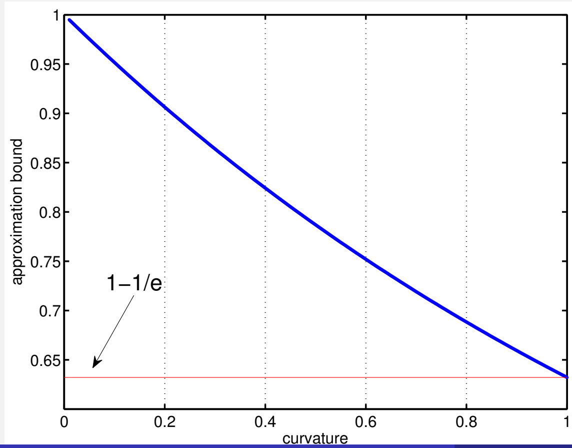
- By submodularity, total curvature can be computed in either form:

$$c \triangleq 1 - \min_{S,j\notin S: f(j|\emptyset)\neq 0} \frac{f(j|S)}{f(j|\emptyset)} = 1 - \min_{j:f(j|\emptyset)\neq 0} \frac{f(j|V\setminus\{j\})}{f(j|\emptyset)} \qquad (15.17)$$

- Note: Matroid rank is either modular $c = 0$ or maximally curved $c = 1$ — hence, matroid rank can have only the extreme points of curvature, namely 0 or 1.
- Polymatroid functions are, in this sense, more nuanced, in that they allow non-extreme curvature, with $c \in [0,1]$.
- It will be remembered the notion of “partial dependence” within polymatroid functions.

Curvature and approximation

- Curvature limitation can help the greedy algorithm in terms of approximation bounds.
- Conforti & Cornuéjols showed that greedy gives a $1/(1+c)$ approximation to $\max\{f(S) : S \in \mathcal{I}\}$ when f has total curvature c.
- Hence, greedy subject to matroid constraint is a $\max(1/(1+c), 1/2)$ approximation algorithm, and if $c < 1$ then it is better than 1/2 (e.g., with $c = 1/4$ then we have a 0.8 algorithm).
- For k-uniform matroid (i.e., k-cardinality constraints), then approximation factor becomes $\frac{1}{c}(1 - e^{-c})$



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

$$S_{i+1} = S_i \cup \left\{ \operatorname*{argmax}_{v \in V \setminus S_i \,:\, S_i + v \in \bigcap_{i=1}^{p} \mathcal{I}_i} f(S_i \cup \{v\}) \right\} \tag{15.17}$$

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

Theorem 15.2.2

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.

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.
- Consider bipartite graph $G = (E, F, V)$ where E and F are the left/right set of nodes, respectively, and V is the set of edges.
- E corresponds to, say, an English language sentence and F corresponds to a French language sentence — goal is to form a matching (an alignment) between the two.

Greedy over > 1 matroids: Multiple Language Alignment

- Consider English string and French string, set up as a bipartite graph.

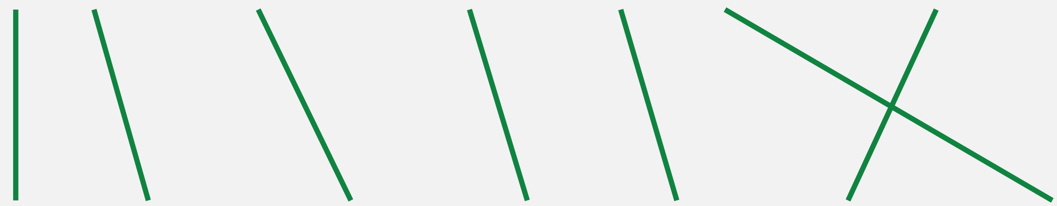
I have ... as an example of public ownership

je le ai ... comme exemple de propriété publique

Greedy over > 1 matroids: Multiple Language Alignment

- One possible alignment, a matching, with score as sum of edge weights.

I have ... as an example of public ownership

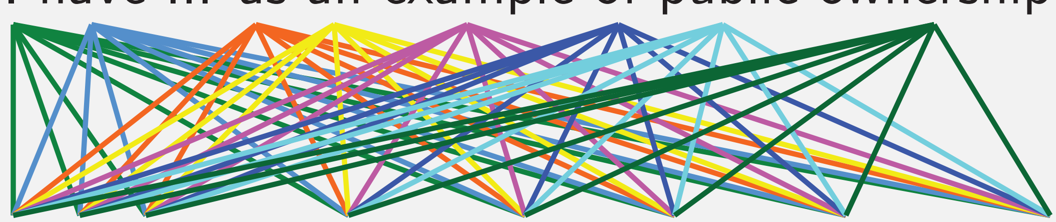


je le ai ... comme exemple de propriété publique

Greedy over > 1 matroids: Multiple Language Alignment

- Edges incident to English words constitute an edge partition

I have ... as an example of public ownership



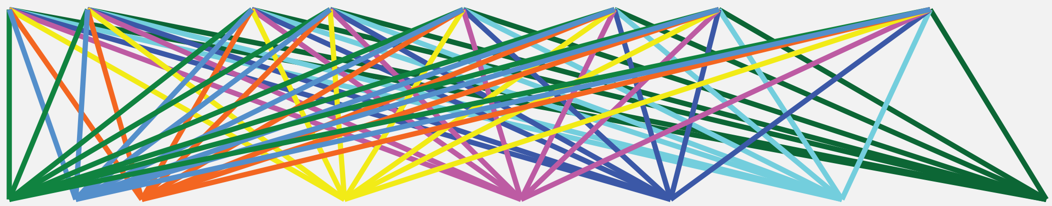
je le ai ... comme exemple de propriété publique

- The two edge partitions can be used to set up two 1-partition matroids on the edges.
- For each matroid, a set of edges is independent only if the set intersects each partition block no more than one time.

Greedy over > 1 matroids: Multiple Language Alignment

- Edges incident to French words constitute an edge partition

I have ... as an example of public ownership



je le ai ... comme exemple de propriété publique

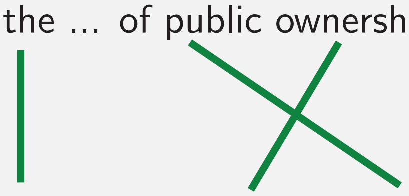
- The two edge partitions can be used to set up two 1-partition matroids on the edges.
- For each matroid, a set of edges is independent only if the set intersects each partition block no more than one time.

Greedy over > 1 matroids: Multiple Language Alignment

- Typical to use bipartite matching to find an alignment between the two language strings.
- As we saw, this is equivalent to two 1-partition matroids and a non-negative modular cost function on the edges.
- We can generalize this using a polymatroid cost function on the edges, and two k-partition matroids, allowing for "fertility" in the models:

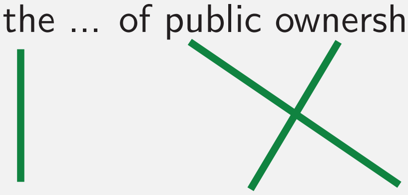
Fertility at most 1

. . . the ... of public ownership



. . . le ... de propriété publique

. . . the ... of public ownership

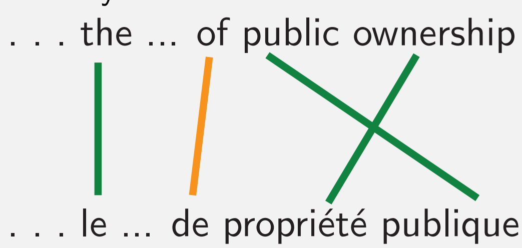


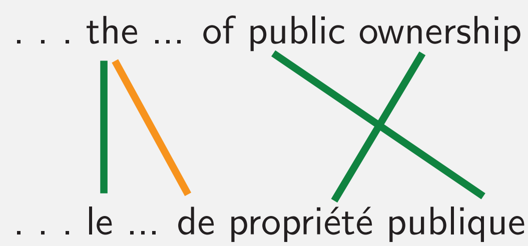
. . . le ... de propriété publique

Greedy over > 1 matroids: Multiple Language Alignment

- Typical to use bipartite matching to find an alignment between the two language strings.
- As we saw, this is equivalent to two 1-partition matroids and a non-negative modular cost function on the edges.
- We can generalize this using a polymatroid cost function on the edges, and two k-partition matroids, allowing for "fertility" in the models:

Fertility at most 2





Greedy over > 1 matroids: Multiple Language Alignment

- Generalizing further, each block of edges in each partition matroid can have its own "fertility" limit:
  $\mathcal{I} = \{X \subseteq V : |X \cap V_i| \leq k_i \text{ for all } i = 1, \dots, \ell\}$.
- Maximizing submodular function subject to multiple matroid constraints addresses this problem.

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 players has a submodular "valuation" function, $g_i : 2^E \to \mathbb{R}_+$ that measures how "desirable" or "valuable" a given subset $A \subseteq E$ of goods are to that player.
- Assumption: No good can be shared between multiple players, each good must be allocated to a single player.
- Goal of submodular welfare: Partition the goods $E = E_1 \cup E_2 \cup \cdots \cup E_n$ into n blocks in order to maximize the submodular social welfare, measured as:

$$\text{submodular-social-welfare}(E_1, E_2, \ldots, E_n) = \sum_{i=1}^{n} g_i(E_i). \tag{15.1}$$

- We can solve this via submodular maximization subject to multiple matroid independence constraints as we next describe ...

Submodular Welfare: Submodular Max over matroid partition

- Create new ground set E' as disjoint union of n copies of the ground set. I.e.,

$$E' = \underbrace{E \uplus E \uplus \cdots \uplus E}_{n\times} \tag{15.2}$$

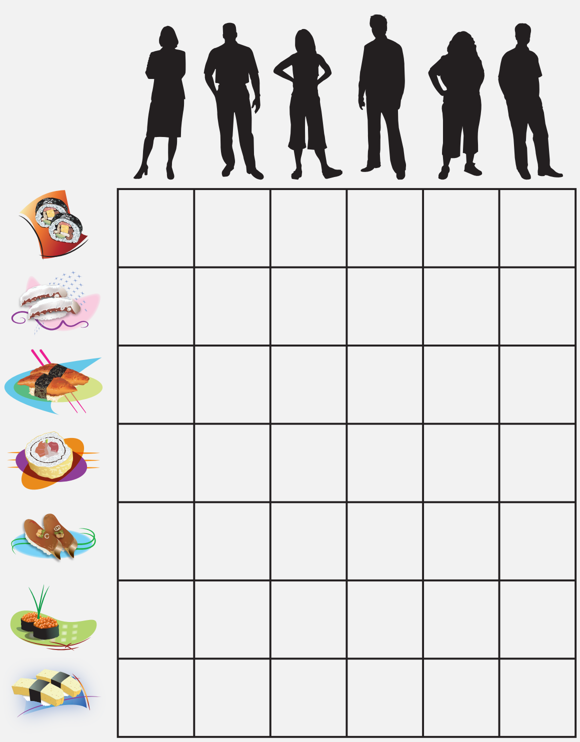
- Let $E^{(i)} \subset E'$ be the i^{th} block of E'.
- For any $e \in E$, the corresponding element in $E^{(i)}$ is called $(e, i) \in E^{(i)}$ (each original element is tagged by integer).
- For $e \in E$, define $E_e = \{(e', i) \in E' : e' = e\}$.
- Hence, $\{E_e\}_{e \in E}$ is a partition of E', each block of the partition for one of the original elements in E.
- Create a 1-partition matroid $\mathcal{M} = (E', \mathcal{I})$ where

$$\mathcal{I} = \left\{ S \subseteq E' : \forall e \in E, |S \cap E_e| \leq 1 \right\} \tag{15.3}$$

Submodular Welfare: Submodular Max over matroid partition

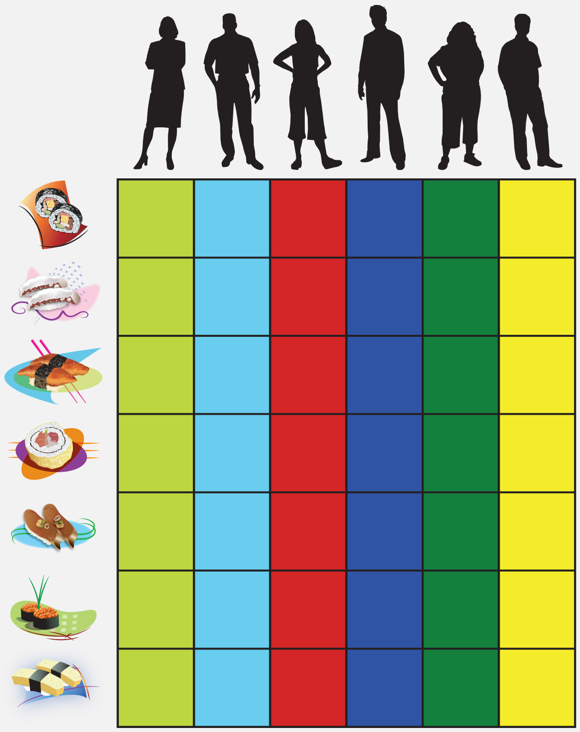
- Hence, S is independent in matroid $\mathcal{M} = (E', I)$ if S uses each original element no more than once.
- Create submodular function $f' : 2^{E'} \to \mathbb{R}_+$ with $f'(S) = \sum_{i=1}^{n} g_i(S \cap E^{(i)})$.
- Submodular welfare maximization becomes matroid constrained submodular max $\max\{f'(S) : S \in \mathcal{I}\}$, so greedy algorithm gives a $1/2$ approximation.

Submodular Social Welfare



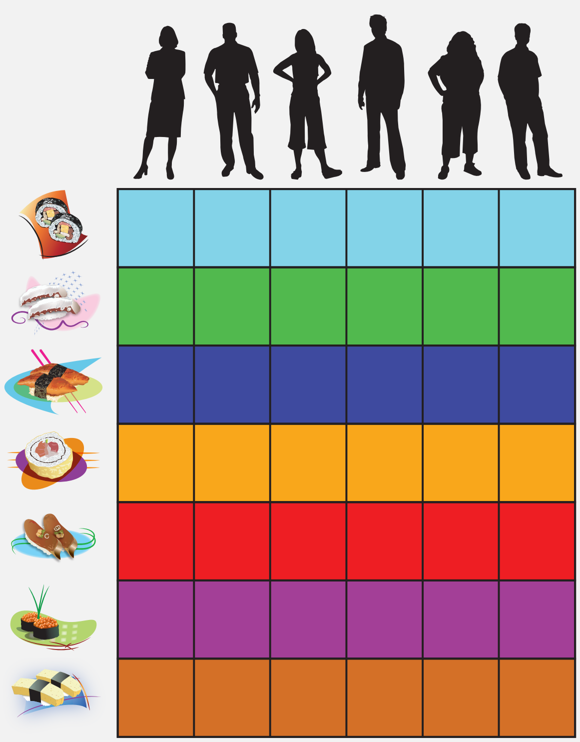
- Have $n = 6$ people (who don't like to share) and $|E| = m = 7$ pieces of sushi. E.g., $e \in E$ might be $e =$ "salmon roll".
- Goal: distribute sushi to people to maximize social welfare.
- Ground set disjoint union $E \uplus E \uplus E \uplus E \uplus E \uplus E$.
- Partition matroid partitions: $E_{e_1} \cup E_{e_2} \cup E_{e_3} \cup E_{e_4} \cup E_{e_5} \cup E_{e_6} \cup E_{e_7}$.
- independent allocation
- non-independent allocation

Submodular Social Welfare



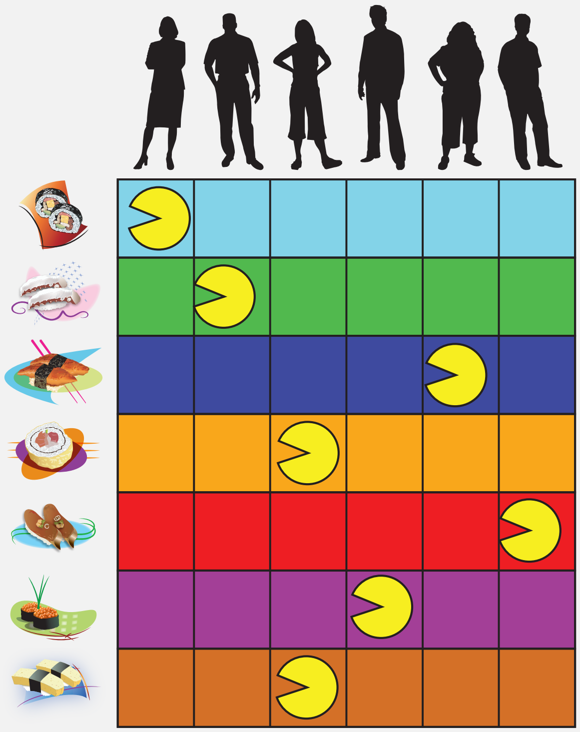
- Have $n = 6$ people (who don't like to share) and $|E| = m = 7$ pieces of sushi. E.g., $e \in E$ might be $e =$ "salmon roll".
- Goal: distribute sushi to people to maximize social welfare.
- Ground set disjoint union $E \uplus E \uplus E \uplus E \uplus E \uplus E$.
- Partition matroid partitions: $E_{e_1} \cup E_{e_2} \cup E_{e_3} \cup E_{e_4} \cup E_{e_5} \cup E_{e_6} \cup E_{e_7}$.
- independent allocation
- non-independent allocation

Submodular Social Welfare



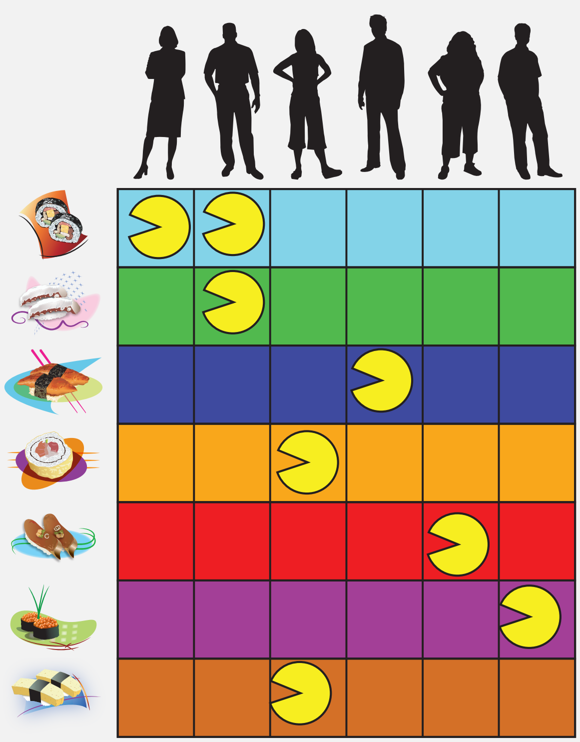
- Have $n = 6$ people (who don't like to share) and $|E| = m = 7$ pieces of sushi. E.g., $e \in E$ might be $e =$ "salmon roll".
- Goal: distribute sushi to people to maximize social welfare.
- Ground set disjoint union $E \uplus E \uplus E \uplus E \uplus E \uplus E$.
- Partition matroid partitions: $E_{e_1} \cup E_{e_2} \cup E_{e_3} \cup E_{e_4} \cup E_{e_5} \cup E_{e_6} \cup E_{e_7}$.
- independent allocation
- non-independent allocation

Submodular Social Welfare



- Have $n = 6$ people (who don't like to share) and $|E| = m = 7$ pieces of sushi. E.g., $e \in E$ might be $e =$ "salmon roll".
- Goal: distribute sushi to people to maximize social welfare.
- Ground set disjoint union $E \uplus E \uplus E \uplus E \uplus E \uplus E$.
- Partition matroid partitions: $E_{e_1} \cup E_{e_2} \cup E_{e_3} \cup E_{e_4} \cup E_{e_5} \cup E_{e_6} \cup E_{e_7}$.
- independent allocation
- non-independent allocation

Submodular Social Welfare



- Have $n = 6$ people (who don't like to share) and $|E| = m = 7$ pieces of sushi. E.g., $e \in E$ might be $e =$ "salmon roll".
- Goal: distribute sushi to people to maximize social welfare.
- Ground set disjoint union $E \uplus E \uplus E \uplus E \uplus E \uplus E$.
- Partition matroid partitions: $E_{e_1} \cup E_{e_2} \cup E_{e_3} \cup E_{e_4} \cup E_{e_5} \cup E_{e_6} \cup E_{e_7}$.
- independent allocation
- non-independent allocation

Monotone Submodular over Knapsack Constraint

- The constraint $|A| \leq k$ is a simple cardinality constraint.
- Consider a non-negative integral modular function $c : E \to \mathbb{Z}_+$.
- 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 $\max \{f(A) : A \subseteq V, c(A) \leq b\}$.
- Important: A knapsack constraint yields an independence system (down closed) but it is not a matroid!
- $c(e)$ 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

$$S_{i+1} = S_i \cup \left\{ \operatorname*{argmax}_{v \in V \setminus S_i} \Big(f(S_i \cup \{v\}) - f(S_i) \Big) \right\} \tag{15.4}$$

  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

$$S_{i+1} = S_i \cup \left\{ \operatorname*{argmax}_{v \in V \setminus S_i} \frac{f(S_i \cup \{v\}) - f(S_i)}{c(v)} \right\} \tag{15.5}$$

  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 (Minoux trick also possible for further speed)
- Partial enumeration: 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.

Local Search Algorithms

From J. Vondrak

- Local search involves switching up to t elements, as long as it provides a (non-trivial) improvement; can iterate in several phases. Some examples follow:
- 1/3 approximation to unconstrained non-monotone maximization [Feige, Mirrokni, Vondrak, 2007]
- $1/(k + 2 + \frac{1}{k} + \delta_t)$ approximation for non-monotone maximization subject to k matroids [Lee, Mirrokni, Nagarajan, Sviridenko, 2009]
- $1/(k + \delta_t)$ approximation for monotone submodular maximization subject to $k \geq 2$ matroids [Lee, Sviridenko, Vondrak, 2010].

What About Non-monotone

- 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.
- 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.
- We may get a $(\frac{1}{3} - \frac{\epsilon}{n})$ approximation for maximizing non-monotone non-negative submodular functions, with most $O(\frac{1}{\epsilon} n^3 \log n)$ function calls using approximate local maxima.

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 $v \in V \setminus S$ (i.e., local in a Hamming ball of radius 1).
- The following interesting result is true for any submodular function:

Lemma 15.3.1

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

- Idea of proof: Given $v_1, v_2 \in S$, suppose $f(S - v_1) \leq f(S)$ and $f(S - v_2) \leq f(S)$. Submodularity requires $f(S - v_1) + f(S - v_2) \geq f(S) + f(S - v_1 - v_2)$ which would be impossible unless $f(S - v_1 - v_2) \leq f(S)$.
- Similarly, given $v_1, v_2 \notin S$, and $f(S + v_1) \leq f(S)$ and $f(S + v_2) \leq f(S)$. Submodularity requires $f(S + v_1) + f(S + v_2) \geq f(S) + f(S + v_1 + v_2)$ which requires $f(S + v_1 + v_2) \leq f(S)$.

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 $v \in V \setminus S$ (i.e., local in a Hamming ball of radius 1).
- The following interesting result is true for any submodular function:

Lemma 15.3.1

Given a submodular function f, if S is a local maximum 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 maximum, 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 maximum is already hard (PLS-complete), but it is possible to find an approximate local maximum relatively efficiently.
- This is the approach that yields the $(\frac{1}{3} - \frac{\epsilon}{n})$ approximation algorithm.

Linear time algorithm unconstrained non-monotone max

- Tight randomized tight $1/2$ approximation algorithm for unconstrained non-monotone non-negative submodular maximization.
- Buchbinder, Feldman, Naor, Schwartz 2012. Recall $[a]_+ = \max(a, 0)$.

Algorithm 4: Randomized Linear-time non-monotone submodular max

Set $L \leftarrow \emptyset$; $U \leftarrow V$ $\quad$ /* Lower L, upper U. Invariant: $L \subseteq U$ */ ;
Order elements of $V = (v_1, v_2, \ldots, v_n)$ arbitrarily ;
for $i \leftarrow 0 \ldots |V|$ **do**
    $a \leftarrow [f(v_i|L)]_+$; $b \leftarrow [-f(U|U \setminus \{v_i\})]_+$;
    if $a = b = 0$ **then** $p \leftarrow 1/2$;
    ;
    else $p \leftarrow a/(a + b)$;
    ;
    if *Flip of coin with* $\Pr($*heads*$) = p$ *draws heads* **then**
        $L \leftarrow L \cup \{v_i\}$;
    Otherwise /* if the coin drew tails, an event with prob. $1 - p$ */
        $U \leftarrow U \setminus \{v\}$
return L *(which is the same as U at this point)*

Linear time algorithm unconstrained non-monotone max

- Each "sweep" of the algorithm is $O(n)$.
- Running the algorithm $1\times$ (with an arbitrary variable order) results in a $1/3$ approximation.
- The $1/2$ guarantee is in expected value (the expected solution has the $1/2$ guarantee).
- In practice, run it multiple times, each with a different random permutation of the elements, and then take the cumulative best.
- It may be possible to choose the random order smartly to get better results in practice.

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.

Some results on submodular maximization

- As we've seen, we can get $1 - 1/e$ for non-negative monotone submodular (polymatroid) functions with greedy algorithm under cardinality constraints, and this is tight.
- For general matroid, greedy reduces to $1/2$ approximation (as we've seen).
- We can recover $1 - 1/e$ approximation using the continuous greedy algorithm on the multilinear extension and then using pipage rounding to re-integerize the solution (see J. Vondrak's publications).
- More general constraints are possible too, as we see on the next table (for references, see Jan Vondrak's publications `http://theory.stanford.edu/~jvondrak/`).

Submodular Max Summary - 2012: From J. Vondrak

Monotone Maximization

Constraint	Approximation	Hardness	Technique
$\|S\| \leq k$	$1 - 1/e$	$1 - 1/e$	greedy
matroid	$1 - 1/e$	$1 - 1/e$	multilinear ext.
$O(1)$ knapsacks	$1 - 1/e$	$1 - 1/e$	multilinear ext.
k matroids	$k + \epsilon$	$k/\log k$	local search
k matroids and $O(1)$ knapsacks	$O(k)$	$k/\log k$	multilinear ext.

Nonmonotone Maximization

Constraint	Approximation	Hardness	Technique
Unconstrained	$1/2$	$1/2$	combinatorial
matroid	$1/e$	0.48	multilinear ext.
$O(1)$ knapsacks	$1/e$	0.49	multilinear ext.
k matroids	$k + O(1)$	$k/\log k$	local search
k matroids and $O(1)$ knapsacks	$O(k)$	$k/\log k$	multilinear ext.

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, probably due to the difficulty in computing the "concave extension" of a submodular function (the convex extension is easy, namely the Lovász extension).
- A paper by Chekuri, Vondrak, and Zenklusen (2011) make some progress on this front using multilinear extensions.

Multilinear extension

Definition 15.3.2

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

$$F(x) = \sum_{S \subseteq V} f(S) \prod_{i \in S} x_i \prod_{j \in V \setminus S} (1 - x_j) \tag{15.6}$$

- 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 15.3.3

Let $F : [0,1]^V \to \mathbb{R}$ be multilinear extension of set function $f : 2^V \to \mathbb{R}$, 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 $i, j\ inV$, $x \in [0,1]^V$.*

Multilinear extension

- Moreover, we have

Lemma 15.3.4

Let $F : [0,1]^V \to \mathbb{R}$ be multilinear extension of set function $f : 2^V \to \mathbb{R}$, 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 $\{\mathbf{1}_I : I \in \mathcal{I}\} \subseteq P_{\mathcal{I}}$
- 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.

Submodular Max and polyhedral approaches

- 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 (pipage 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 ℓ sparse packing integer programs, and unsplittable flow in paths and trees.
- Also, Vondrak showed that this scheme achieves the $\frac{1}{c}(1 - e^{-c})$ curvature based bound for any matroid, which matches the bound we had earlier for uniform matroids with standard greedy.
- In general, one needs to do Monte-Carlo methods to estimate the multilinear extension (so further approximations would apply).

Review from lecture 11

The next slide comes from lecture 11.

A polymatroid function's polyhedron is a polymatroid.

Theorem 15.4.1

Let f be a polymatroid function defined on subsets of E. For any $x \in \mathbb{R}_+^E$, and any P_f^+-basis $y^x \in \mathbb{R}_+^E$ of x, the component sum of y^x is

$$y^x(E) = \textit{rank}(x) = \max\left(y(E) : y \leq x, y \in P_f^+\right)$$
$$= \min\left(x(A) + f(E \setminus A) : A \subseteq E\right) \quad (15.10)$$

As a consequence, P_f^+ is a polymatroid, since r.h.s. is constant w.r.t. y^x.

Taking $E \setminus B = \text{supp}(x)$ (so elements B are all zeros in x), and for $b \notin B$ we make $x(b)$ is big enough, the r.h.s. min has solution $A^* = B$. We recover submodular function from the polymatroid polyhedron via the following:

$$\text{rank}\left(\frac{1}{\epsilon}\mathbf{1}_{E \setminus B}\right) = f(B) = \max\left\{y(B) : y \in P_f^+\right\} \quad (15.11)$$

In fact, we will ultimately see a number of important consequences of this theorem (other than just that P_f^+ is a polymatroid)

Review from lecture 12

The next slide comes from lecture 12.

Matroid instance of Theorem ??

- Considering Theorem ??, the matroid case is now a special case, where we have that:

Corollary 15.4.2

We have that:

$$\max \{y(E) : y \in P_{\textit{ind. set}}(M), y \leq x\} = \min \{r_M(A) + x(E \setminus A) : A \subseteq E\} \tag{15.30}$$

where r_M is the matroid rank function of some matroid.

Most violated inequality problem in matroid polytope case

- Consider

$$P_r^+ = \left\{x \in \mathbb{R}^E : x \geq 0, x(A) \leq r_M(A), \forall A \subseteq E\right\} \tag{15.7}$$

- Suppose we have any $x \in \mathbb{R}_+^E$ such that $x \notin P_r^+$.
- Hence, there must be a set of $\mathcal{W} \subseteq 2^V$, each member of which corresponds to a violated inequality, i.e., equations of the form $x(A) > r_M(A)$ for $A \in \mathcal{W}$.
- The most violated inequality when x is considered w.r.t. P_r^+ corresponds to the set A that maximizes $x(A) - r_M(A)$, i.e., the most violated inequality is valuated as:

$$\max \{x(A) - r_M(A) : A \in \mathcal{W}\} = \max \{x(A) - r_M(A) : A \subseteq E\} \tag{15.8}$$

- Since x is modular and $x(E \setminus A) = x(E) - x(A)$, we can express this via a min as in;:

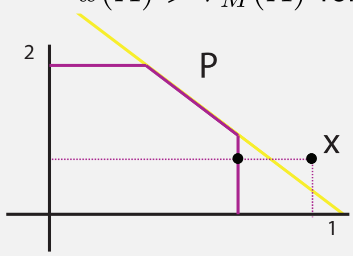
$$\min \{r_M(A) + x(E \setminus A) : A \subseteq E\} \tag{15.9}$$

Most violated inequality/polymatroid membership/SFM

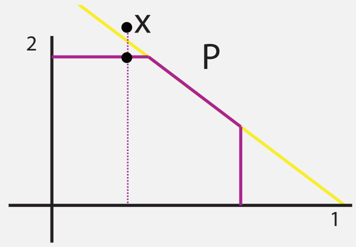
- Consider

$$P_f^+ = \{x \in \mathbb{R}^E : x \geq 0, x(A) \leq f(A), \forall A \subseteq E\} \tag{15.10}$$

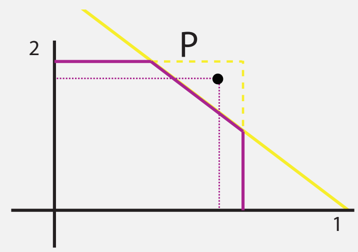
- Suppose we have any $x \in \mathbb{R}_+^E$ such that $x \notin P_f^+$.
- Hence, there must be a set of $\mathcal{W} \subseteq 2^V$, each member of which corresponds to a violated inequality, i.e., equations of the form $x(A) > r_M(A)$ for $A \in \mathcal{W}$.



$\mathcal{W} = \{\{1\}\{1,2\}\}$



$\mathcal{W} = \{\{2\},\{1,2\}\}$



$\mathcal{W} = \{\{1,2\}\}$

Most violated inequality/polymatroid membership/SFM

- The most violated inequality when x is considered w.r.t. P_f^+ corresponds to the set A that maximizes $x(A) - f(A)$, i.e., the most violated inequality is valuated as:

$$\max\{x(A) - f(A) : A \in \mathcal{W}\} = \max\{x(A) - f(A) : A \subseteq E\} \tag{15.11}$$

- Since x is modular and $x(E \setminus A) = x(E) - x(A)$, we can express this via a min as in;:

$$\min\{f(A) + x(E \setminus A) : A \subseteq E\} \tag{15.12}$$

- More importantly, $\min\{f(A) + x(E \setminus A) : A \subseteq E\}$ is a form of submodular function minimization, namely $\min\{f(A) - x(A) : A \subseteq E\}$ for a submodular f and $x \in \mathbb{R}_+^E$, consisting of a difference of polymatroid and modular function (so $f - x$ is no longer necessarily monotone, nor positive).
- We will ultimatley answer how general this form of SFM is.

Review from Lecture 6

The following three slides are review from lecture 6.

Matroids, other definitions using matroid rank $r : 2^V \to \mathbb{Z}_+$

Definition 15.5.3 (closed/flat/subspace)

A subset $A \subseteq E$ is closed (equivalently, a flat or a subspace) of matroid M if for all $x \in E \setminus A$, $r(A \cup \{x\}) = r(A) + 1$.

Definition: A hyperplane is a flat of rank $r(M) - 1$.

Definition 15.5.4 (closure)

Given $A \subseteq E$, the closure (or span) of A, is defined by $\text{span}(A) = \{b \in E : r(A \cup \{b\}) = r(A)\}$.

Therefore, a closed set A has $\text{span}(A) = A$.

Definition 15.5.5 (circuit)

A subset $A \subseteq E$ is circuit or a cycle if it is an inclusionwise-minimal dependent set (i.e., if $r(A) < |A|$ and for any $a \in A$, $r(A \setminus \{a\}) = |A| - 1$).

Matroids by circuits

A set is independent if and only if it contains no circuit. Therefore, it is not surprising that circuits can also characterize a matroid.

Theorem 15.5.3 (Matroid by circuits)

Let E be a set and $\mathcal{C}$ be a collection of subsets of E that satisfy the following three properties:

1. *(C1): $\emptyset \notin \mathcal{C}$*
2. *(C2): if $C_1, C_2 \in \mathcal{C}$ and $C_1 \subseteq C_2$, then $C_1 = C_2$.*
3. *(C3): if $C_1, C_2 \in \mathcal{C}$ with $C_1 \neq C_2$, and $e \in C_1 \cap C_2$, then there exists a $C_3 \in \mathcal{C}$ such that $C_3 \subseteq (C_1 \cup C_2) \setminus \{e\}$.*

Matroids by circuits

Several circuit definitions for matroids.

Theorem 15.5.3 (Matroid by circuits)

Let E be a set and $\mathcal{C}$ be a collection of nonempty subsets of E, such that no two sets in $\mathcal{C}$ are contained in each other. Then the following are equivalent.

1. *$\mathcal{C}$ is the collection of circuits of a matroid;*
2. *if $C, C' \in \mathcal{C}$, and $x \in C \cap C'$, then $(C \cup C') \setminus \{x\}$ contains a set in $\mathcal{C}$;*
3. *if $C, C' \in \mathcal{C}$, and $x \in C \cap C'$, and $y \in C \setminus C'$, then $(C \cup C') \setminus \{x\}$ contains a set in $\mathcal{C}$ containing y;*

Again, think about this for a moment in terms of linear spaces and matrices, and spanning trees.

Fundamental circuits in matroids

Lemma 15.5.1

Let $I \in \mathcal{I}(M)$, and $e \in E$, then $I \cup \{e\}$ contains at most one circuit in M.

Proof.

- Suppose, to the contrary, that there are two distinct circuits C_1, C_2 such that $C_1 \cup C_2 \subseteq I \cup \{e\}$.
- Then $e \in C_1 \cap C_2$, and by (C2), there is a circuit C_3 of M s.t. $C_3 \subseteq (C_1 \cup C_2) \setminus \{e\} \subseteq I$
- This contradicts the independence of I.

□

In general, let $C(I, e)$ be the unique circuit associated with $I \cup \{e\}$ (commonly called the fundamental circuit in M w.r.t. I and e).

Matroids: The Fundamental Circuit

- Define $C(I, e)$ be the unique circuit associated with $I \cup \{e\}$ (the fundamental circuit in M w.r.t. I and e, if it exists).
- If $e \in \text{span}(I) \setminus I$, then $C(I, e)$ is well defined ($I + e$ creates one circuit).
- If $e \in I$, then $I + e = I$ doesn't create a circuit. In such cases, $C(I, e)$ is not really defined.
- In such cases, we define $C(I, e) = \{e\}$, and we will soon see why.
- If $e \notin \text{span}(I)$, then $C(I, e) = \emptyset$, since no circuit is created in this case.

Union of matroid bases of a set

Lemma 15.5.2

Let $\mathcal{B}(D)$ be the set of bases of any set D. Then, given matroid $\mathcal{M} = (E, \mathcal{I})$, and any loop-free (i.e., no dependent singleton elements) set $D \subseteq E$, we have:

$$\bigcup_{B \in \mathcal{B}(D)} B = D. \tag{15.13}$$

Proof.

- Define $D' \triangleq \bigcup_{B \in \mathcal{B}(D)}$, and suppose $\exists d \in D$ such that $d \notin D'$.
- Hence, $\forall B \in \mathcal{B}(D)$ we have $d \notin B$, and $B + c$ must contain a single circuit for any B, namely $C(B, d)$.
- Then choose $d' \in C(B, d)$ with $d' \neq d$.
- Then $B + d - d'$ is independent size $|B|$ subset of D and hence spans D, and thus is a d-containing member of $\mathcal{B}(D)$, contradicting $d \notin D'$.

The sat function = Polymatroid Closure

- Thus, in a matroid, closure (span) of a set A are all items that A spans (eq. that depend on A).
- We wish to generalize closure to polymatroids.
- Consider $x \in P_f$ for polymatroid function f.
- Again, recall, tight sets are closed under union and intersection, and therefore form a distributive lattice.
- That is, we saw in Lecture 7 that for any $A, B \in \mathcal{D}(x)$, we have that $A \cup B \in \mathcal{D}(x)$ and $A \cap B \in \mathcal{D}(x)$, which can constitute a join and meet.
- Recall, for a given $x \in P_f$, we have defined this tight family as

$$\mathcal{D}(x) = \{A : A \subseteq E, x(A) = f(A)\} \tag{15.14}$$

The sat function = Polymatroid Closure

- Now given $x \in P_f^+$:

$$\mathcal{D}(x) = \{A : A \subseteq E, x(A) = f(A)\} \tag{15.15}$$
$$= \{A : f(A) - x(A) = 0\} \tag{15.16}$$

- Since $x \in P_f^+$ and f is presumed to be polymatroid function, we see $f'(A) = f(A) - x(A)$ is a non-negative submodular function, and $\mathcal{D}(x)$ are the zero-valued minimizers (if any) of $f'(A)$.
- The zero-valued minimizers of f' are thus closed under union and intersection.
- In fact, this is true for all minimizers of a submodular function as stated in the next theorem.

Minimizers of a Submodular Function form a lattice

Theorem 15.6.1

For arbitrary submodular f, the minimizers are closed under union and intersection. That is, let $\mathcal{M} = \operatorname{argmin}_{X \subseteq E} f(X)$ be the set of minimizers of f. Let $A, B \in \mathcal{M}$. Then $A \cup B \in \mathcal{M}$ and $A \cap B \in \mathcal{M}$.

Proof.

Since A and B are minimizers, we have $f(A) = f(B) \leq f(A \cap B)$ and $f(A) = f(B) \leq f(A \cup B)$.
By submodularity, we have

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B) \tag{15.17}$$

Hence, we must have $f(A) = f(B) = f(A \cup B) = f(A \cap B)$. □

Thus, the minimizers of a submodular function form a lattice, and there is a maximal and a minimal minimizer of every submodular function.

The sat function = Polymatroid Closure

- Matroid closure is generalized by the unique maximal element in $\mathcal{D}(x)$, also called the polymatroid closure or sat (saturation function).
- For some $x \in P_f$, we have defined:

$$\text{cl}(x) \stackrel{\text{def}}{=} \text{sat}(x) \stackrel{\text{def}}{=} \bigcup \{A : A \in \mathcal{D}(x)\} \tag{15.18}$$

$$= \bigcup \{A : A \subseteq E, x(A) = f(A)\} \tag{15.19}$$

$$= \{e : e \in E, \forall \alpha > 0, x + \alpha \mathbf{1}_e \notin P_f\} \tag{15.20}$$

- Hence, $\text{sat}(x)$ is the maximal (zero-valued) minimizer of the submodular function $f_x(A) \triangleq f(A) - x(A)$.
- Eq. (15.20) says that sat consists of any point x that is P_f saturated (any additional positive movement, in that dimension, leaves P_f). We'll revisit this in a few slides.
- First, we see how sat generalizes matroid closure.

The sat function = Polymatroid Closure

- Consider matroid $(E, \mathcal{I}) = (E, r)$, some $I \in \mathcal{I}$. Then $\mathbf{1}_I \in P_r$ and

$$\mathcal{D}(\mathbf{1}_I) = \{A : \mathbf{1}_I(A) = r(A)\} \tag{15.21}$$

and

$$\text{sat}(\mathbf{1}_I) = \bigcup \{A : A \subseteq E, A \in \mathcal{D}(\mathbf{1}_I)\} \tag{15.22}$$

$$= \bigcup \{A : A \subseteq E, \mathbf{1}_I(A) = r(A)\} \tag{15.23}$$

$$= \bigcup \{A : A \subseteq E, |I \cap A| = r(A)\} \tag{15.24}$$

- Notice that $\mathbf{1}_I(A) = |I \cap A| \leq |I|$.
- Intuitively, consider an $A \supset I \in \mathcal{I}$ that doesn't increase rank, meaning $r(A) = r(I)$. If $r(A) = |I \cap A| = r(I \cap A)$, as in Eqn. (15.24), then A is in I's span, so should get $\text{sat}(\mathbf{1}_I) = \text{span}(I)$.
- We formalize this next.

The sat function = Polymatroid Closure

Lemma 15.7.1 (Matroid $\text{sat} : \mathbb{R}_+^E \to 2^E$ is the same as closure.)

For $I \in \mathcal{I}$, we have
$$\text{sat}(\mathbf{1}_I) = \text{span}(I) \tag{15.25}$$

Proof.

- For $\mathbf{1}_I(I) = |I| = r(I)$, so $I \in \mathcal{D}(\mathbf{1}_I)$ and $I \subseteq \text{sat}(\mathbf{1}_I)$. Also, $I \subseteq \text{span}(I)$.
- Consider some $b \in \text{span}(I) \setminus I$.
- Then $I \cup \{b\} \in \mathcal{D}(\mathbf{1}_I)$ since $\mathbf{1}_I(I \cup \{b\}) = |I| = r(I \cup \{b\}) = r(I)$.
- Thus, $b \in \text{sat}(\mathbf{1}_I)$.
- Therefore, $\text{sat}(\mathbf{1}_I) \supseteq \text{span}(I)$.

...

The sat function = Polymatroid Closure

... proof continued.

- Now, consider $b \in \text{sat}(\mathbf{1}_I) \setminus I$.
- Choose any $A \in \mathcal{D}(\mathbf{1}_I)$ with $b \in A$, thus $b \in A \setminus I$.
- Then $\mathbf{1}_I(A) = |A \cap I| = r(A)$.
- Now $r(A) = |A \cap I| \leq |I| = r(I)$.
- Also, $r(A \cap I) = |A \cap I|$ since $A \cap I \in \mathcal{I}$.
- Hence, $r(A \cap I) = r(A) = r((A \cap I) \cup (A \setminus I))$ meaning $(A \setminus I) \subseteq \text{span}(A \cap I) \subseteq \text{span}(I)$.
- Since $b \in A \setminus I$, we get $b \in \text{span}(I)$.
- Thus, $\text{sat}(\mathbf{1}_I) \subseteq \text{span}(I)$.
- Hence $\text{sat}(\mathbf{1}_I) = \text{span}(I)$

$\square$

The sat function = Polymatroid Closure

- Now, consider a matroid (E, r) and some $C \subseteq E$ with $C \notin \mathcal{I}$, and consider $\mathbf{1}_C$. Is $\mathbf{1}_C \in P_r$? No, it might not be a vertex, or even a member, of P_r.
- $\mathrm{span}(\cdot)$ operates on more than just independent sets, so $\mathrm{span}(C)$ is perfectly sensible.
- Note $\mathrm{span}(C) = \mathrm{span}(B)$ where $\mathcal{I} \ni B \in \mathcal{B}(C)$ is a base of C.
- Then we have $\mathbf{1}_B \leq \mathbf{1}_C \leq \mathbf{1}_{\mathrm{span}(C)}$, and that $\mathbf{1}_B \in P_r$. We can then make the definition:

$$\mathrm{sat}(\mathbf{1}_C) \triangleq \mathrm{sat}(\mathbf{1}_B) \text{ for } B \in \mathcal{B}(C) \tag{15.26}$$

  In which case, we also get $\mathrm{sat}(\mathbf{1}_C) = \mathrm{span}(C)$ (in general, could define $\mathrm{sat}(y) = \mathrm{sat}(\text{P-basis}(y))$).
- However, consider the following form

$$\mathrm{sat}(\mathbf{1}_C) = \bigcup \{A : A \subseteq E, |A \cap C| = r(A)\} \tag{15.27}$$

  Exercise: is $\mathrm{span}(C) = \mathrm{sat}(\mathbf{1}_C)$? Prove or disprove it.

The sat function, span, and submodular function minimization

- Thus, for a matroid, $\mathrm{sat}(\mathbf{1}_I)$ is exactly the closure (or span) of I in the matroid. I.e., for matroid (E, r), we have $\mathrm{span}(I) = \mathrm{sat}(\mathbf{1}_B)$.
- Recall, for $x \in P_f$ and polymatroidal f, $\mathrm{sat}(x)$ is the maximal (by inclusion) minimizer of $f(A) - x(A)$, and thus in a matroid, $\mathrm{span}(I)$ is the maximal minimizer of the submodular function formed by $r(A) - \mathbf{1}_I(A)$.
- Submodular function minimization can solve "span" queries in a matroid or "sat" queries in a polymatroid.

sat, as tight polymatroidal elements

- We are given an $x \in P_f^+$ for submodular function f.
- Recall that for such an x, $\text{sat}(x)$ is defined as

$$\text{sat}(x) = \bigcup \{A : x(A) = f(A)\} \tag{15.28}$$

- We also have stated that $\text{sat}(x)$ can be defined as:

$$\text{sat}(x) = \left\{e : \forall \alpha > 0, x + \alpha \mathbf{1}_e \notin P_f^+\right\} \tag{15.29}$$

- We next show more formally that these are the same.

sat, as tight polymatroidal elements

- Lets start with one definition and derive the other.

$$\text{sat}(x) \stackrel{\text{def}}{=} \left\{e : \forall \alpha > 0, x + \alpha \mathbf{1}_e \notin P_f^+\right\} \tag{15.30}$$

$$= \{e : \forall \alpha > 0, \exists A \text{ s.t. } (x + \alpha \mathbf{1}_e)(A) > f(A)\} \tag{15.31}$$

$$= \{e : \forall \alpha > 0, \exists A \ni e \text{ s.t. } (x + \alpha \mathbf{1}_e)(A) > f(A)\} \tag{15.32}$$

- this last bit follows since $\mathbf{1}_e(A) = 1 \iff e \in A$. Continuing, we get

$$\text{sat}(x) = \{e : \forall \alpha > 0, \exists A \ni e \text{ s.t. } x(A) + \alpha > f(A)\} \tag{15.33}$$

- given that $x \in P_f^+$, meaning $x(A) \leq f(A)$ for all A, we must have

$$\text{sat}(x) = \{e : \forall \alpha > 0, \exists A \ni e \text{ s.t. } x(A) = f(A)\} \tag{15.34}$$

$$= \{e : \exists A \ni e \text{ s.t. } x(A) = f(A)\} \tag{15.35}$$

- So now, if A is any set such that $x(A) = f(A)$, then we clearly have

$$\forall e \in A, e \in \text{sat}(x), \text{ and therefore that } \text{sat}(x) \supseteq A \tag{15.36}$$

sat, as tight polymatroidal elements

- . . . and therefore, with sat as defined in Eq. (**??**),

$$\text{sat}(x) \supseteq \bigcup \{A : x(A) = f(A)\} \tag{15.37}$$

- On the other hand, for any $e \in \text{sat}(x)$ defined as in Eq. (15.35), since e is itself a member of a tight set, there is a set $A \ni e$ such that $x(A) = f(A)$, giving

$$\text{sat}(x) \subseteq \bigcup \{A : x(A) = f(A)\} \tag{15.38}$$

- Therefore, the two definitions of sat are identical.

Saturation Capacity

- Another useful concept is saturation capacity which we develop next.
- For $x \in P_f$, and $e \in E$, consider finding

$$\max \{\alpha : \alpha \in \mathbb{R}, x + \alpha \mathbf{1}_e \in P_f\} \tag{15.39}$$

- This is identical to:

$$\max \{\alpha : (x + \alpha \mathbf{1}_e)(A) \leq f(A), \forall A \supseteq \{e\}\} \tag{15.40}$$

since any $B \subseteq E$ such that $e \notin B$ does not change in a $\mathbf{1}_e$ adjustment, meaning $(x + \alpha \mathbf{1}_e)(B) = x(B)$.

- Again, this is identical to:

$$\max \{\alpha : x(A) + \alpha \leq f(A), \forall A \supseteq \{e\}\} \tag{15.41}$$

or

$$\max \{\alpha : \alpha \leq f(A) - x(A), \forall A \supseteq \{e\}\} \tag{15.42}$$

Saturation Capacity

- The max is achieved when

$$\alpha = \hat{c}(x;e) \stackrel{\text{def}}{=} \min \{f(A) - x(A), \forall A \supseteq \{e\}\} \tag{15.43}$$

- $\hat{c}(x;e)$ is known as the saturation capacity associated with $x \in P_f$ and e.
- Thus we have for $x \in P_f$,

$$\begin{aligned} \hat{c}(x;e) &\stackrel{\text{def}}{=} \min \{f(A) - x(A), \forall A \ni e\} && (15.44) \\ &= \max \{\alpha : \alpha \in \mathbb{R}, x + \alpha \mathbf{1}_e \in P_f\} && (15.45) \end{aligned}$$

- We immediately see that for $e \in E \setminus \text{sat}(x)$, we have that $\hat{c}(x;e) > 0$.
- Also, for $e \in \text{sat}(x)$, we have that $\hat{c}(x;e) = 0$.
- Note that any α with $0 \leq \alpha \leq \hat{c}(x;e)$ we have $x + \alpha \mathbf{1}_e \in P_f$.
- We also see that computing $\hat{c}(x;e)$ is a form of submodular function minimization.

Dependence Function

- Tight sets can be restricted to contain a particular element.
- Given $x \in P_f$, and $e \in \text{sat}(x)$, define

$$\begin{aligned} \mathcal{D}(x,e) &= \{A : e \in A \subseteq E, x(A) = f(A)\} && (15.46) \\ &= \mathcal{D}(x) \cap \{A : A \subseteq E, e \in A\} && (15.47) \end{aligned}$$

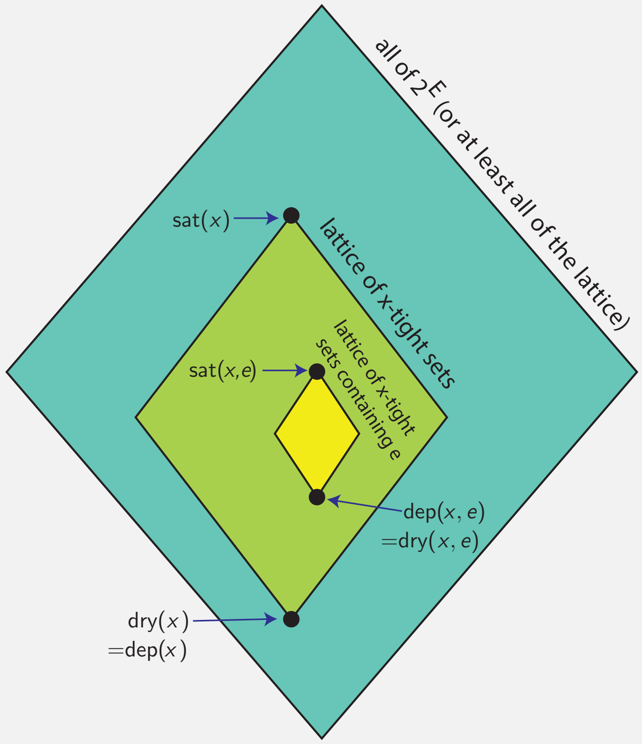
- Thus, $\mathcal{D}(x,e) \subseteq \mathcal{D}(x)$, and $\mathcal{D}(x,e)$ is a sublattice of $\mathcal{D}(x)$.
- Therefore, we can define a unique minimal element of $\mathcal{D}(x,e)$ denoted as follows:

$$\text{dep}(x,e) = \begin{cases} \bigcap \{A : e \in A \subseteq E, x(A) = f(A)\} & \text{if } e \in \text{sat}(x) \\ \emptyset & \text{else} \end{cases} \tag{15.48}$$

- I.e., $\text{dep}(x,e)$ is the minimal element in $\mathcal{D}(x)$ that contains e (the minimal x-tight set containing e).

dep and sat in a lattice

- Given some $x \in P_f$,
- The picture on the right summarizes the relationships between the lattices and sublattices.
- Note, $\bigcap_e \text{dep}(x, e) = \text{dep}(x)$.



dep and sat in a lattice

- Given $x \in P_f$, recall distributive lattice of tight sets $\mathcal{D}(x) = \{A : x(A) = f(A)\}$
- We had that $\text{sat}(x) = \bigcup \{A : A \in \mathcal{D}(x)\}$ is the "1" element of this lattice.
- Consider the "0" element of $\mathcal{D}(x)$, i.e., $\text{dry}(x) \stackrel{\text{def}}{=} \bigcap \{A : A \in \mathcal{D}(x)\}$
- We can see $\text{dry}(x)$ as the elements that are necessary for tightness.
- That is, we can equivalently define $\text{dry}(x)$ as

$$\text{dry}(x) = \left\{e' : x(A) < f(A), \forall A \not\ni e'\right\} \tag{15.49}$$

- This can be read as, for any $e' \in \text{dry}(x)$, any set that does not contain e' is not tight for x (any set A that is missing any element of $\text{dry}(x)$ is not tight).
- Perhaps, then, a better name for dry is $\text{ntight}(x)$, for the necessary for tightness (but we'll actually use neither name).
- Note that dry need not be the empty set. Exercise: give example.

An alternate expression for $\text{dep} = \text{dry}$

- Now, given $x \in P_f$, and $e \in \text{sat}(x)$, recall distributive sub-lattice of e-containing tight sets $\mathcal{D}(x, e) = \{A : e \in A, x(A) = f(A)\}$
- We can define the "1" element of this sub-lattice as $\text{sat}(x, e) \stackrel{\text{def}}{=} \bigcup \{A : A \in \mathcal{D}(x, e)\}$.
- Analogously, we can define the "0" element of this sub-lattice as $\text{dry}(x, e) \stackrel{\text{def}}{=} \bigcap \{A : A \in \mathcal{D}(x, e)\}$.
- We can see $\text{dry}(x, e)$ as the elements that are necessary for e-containing tightness, with $e \in \text{sat}(x)$.
- That is, we can view $\text{dry}(x, e)$ as

$$\text{dry}(x, e) = \left\{e' : x(A) < f(A), \forall A \not\ni e', e \in A\right\} \tag{15.50}$$

- This can be read as, for any $e' \in \text{dry}(x, e)$, any e-containing set that does not contain e' is not tight for x.
- But actually, $\text{dry}(x, e) = \text{dep}(x, e)$, so we have derived another expression for $\text{dep}(x, e)$ in Eq. (15.50).

Dependence Function and Fundamental Matroid Circuit

- Now, let $(E, \mathcal{I}) = (E, r)$ be a matroid, and let $I \in \mathcal{I}$ giving $\mathbf{1}_I \in P_r$. We have $\text{sat}(\mathbf{1}_I) = \text{span}(I) = \text{closure}(I)$.
- Given $e \in \text{sat}(\mathbf{1}_I) \setminus I$ and then consider an $A \ni e$ with $|I \cap A| = r(A)$.
- Then $I \cap A$ serves as a base for A (i.e., $I \cap A$ spans A) and any such A contains a circuit (i.e., we can add $e \in A \setminus I$ to $I \cap A$ w/o increasing rank).
- Given $e \in \text{sat}(\mathbf{1}_I) \setminus I$, and consider $\text{dep}(\mathbf{1}_I, e)$, with

$$\text{dep}(\mathbf{1}_I, e) = \bigcap \{A : e \in A \subseteq E, \mathbf{1}_I(A) = r(A)\} \tag{15.51}$$

$$= \bigcap \{A : e \in A \subseteq E, |I \cap A| = r(A)\} \tag{15.52}$$

$$= \bigcap \{A : e \in A \subseteq E, r(A) - |I \cap A| = 0\} \tag{15.53}$$

- By SFM lattice, $\exists$ a unique minimal $A \ni e$ with $|I \cap A| = r(A)$.
- Thus, $\text{dep}(\mathbf{1}_I, e)$ must be a circuit since if it included more than a circuit, it would not be minimal in this sense.

Dependence Function and Fundamental Matroid Circuit

- Therefore, when $e \in \text{sat}(\mathbf{1}_I) \setminus I$, then $\text{dep}(\mathbf{1}_I, e) = C(I, e)$ where $C(I, e)$ is the unique circuit contained in $I + e$ in a matroid (the fundamental circuit of e and I that we encountered before).
- Now, if $e \in \text{sat}(\mathbf{1}_I) \cap I$ with $I \in \mathcal{I}$, we said that $C(I, e)$ was undefined (since no circuit is created in this case) and so we defined it as $C(I, e) = \{e\}$
- In this case, for such an e, we have $\text{dep}(\mathbf{1}_I, e) = \{e\}$ since all such sets $A \ni e$ with $|I \cap A| = r(A)$ contain e, but in this case no cycle is created, i.e., $|I \cap A| \geq |I \cap \{e\}| = r(e) = 1$.
- We are thus free to take subsets of I as A, all of which must contain e, but all of which have rank equal to size.
- Also note: in general for $x \in P_f$ and $e \in \text{sat}(x)$, we have $\text{dep}(x, e)$ is tight by definition.

Summary of sat, and dep

- For $x \in P_f$, $\text{sat}(x)$ (span, closure) is the maximal saturated (x-tight) set w.r.t. x. I.e., $\text{sat}(x) = \{e : e \in E, \forall \alpha > 0, x + \alpha \mathbf{1}_e \notin P_f\}$. That is,

$$\text{cl}(x) \stackrel{\text{def}}{=} \text{sat}(x) \triangleq \bigcup \{A : A \in \mathcal{D}(x)\} \tag{15.54}$$

$$= \bigcup \{A : A \subseteq E, x(A) = f(A)\} \tag{15.55}$$

$$= \{e : e \in E, \forall \alpha > 0, x + \alpha \mathbf{1}_e \notin P_f\} \tag{15.56}$$

- For $e \in \text{sat}(x)$, we have $\text{dep}(x, e) \subseteq \text{sat}(x)$ (fundamental circuit) is the minimal (common) saturated (x-tight) set w.r.t. x containing e. I.e.,

$$\text{dep}(x, e) = \begin{cases} \bigcap \{A : e \in A \subseteq E, x(A) = f(A)\} & \text{if } e \in \text{sat}(x) \\ \emptyset & \text{else} \end{cases}$$

$$= \{e' : \exists \alpha > 0, \text{ s.t. } x + \alpha(\mathbf{1}_e - \mathbf{1}_{e'}) \in P_f\} \tag{15.57}$$

Dependence Function and exchange

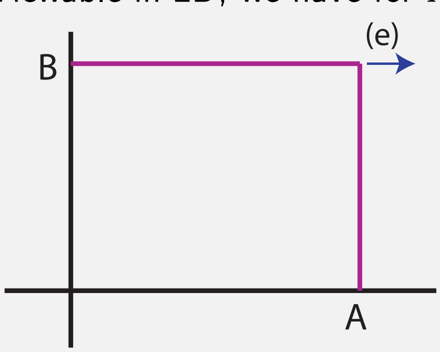
- For $e \in \text{span}(I) \setminus I$, we have that $I + e \notin \mathcal{I}$. This is a set addition restriction property.
- Analogously, for $e \in \text{sat}(x)$, any $x + \alpha \mathbf{1}_e \notin P_f$ for $\alpha > 0$. This is a vector increase restriction property.
- Recall, we have $C(I, e) \setminus e' \in \mathcal{I}$ for $e' \in C(I, e)$. I.e., $C(I, e)$ consists of elements that when removed recover independence.
- In other words, for $e \in \text{span}(I) \setminus I$, we have that

$$C(I, e) = \{a \in E : I + e - a \in \mathcal{I}\} \tag{15.58}$$

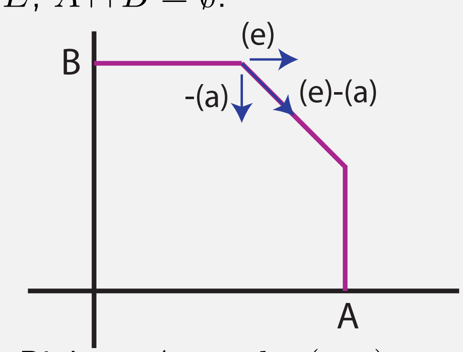
- I.e., an addition of e to I stays within $\mathcal{I}$ only if we simultaneously remove one of the elements of $C(I, e)$.
- But, analogous to the circuit case, is there an exchange property for $\text{dep}(x, e)$ in the form of vector movement restriction?
- We might expect the vector $\text{dep}(x, e)$ property to take the form: a positive move in the e-direction stays within P_f^+ only if we simultaneously take a negative move in one of the $\text{dep}(x, e)$ directions.

Dependence Function and exchange in 2D

- $\text{dep}(x, e)$ is set of neg. directions we must move if we want to move in pos. e direction, starting at x and staying within P_f.
- Viewable in 2D, we have for $A, B \subseteq E$, $A \cap B = \emptyset$:



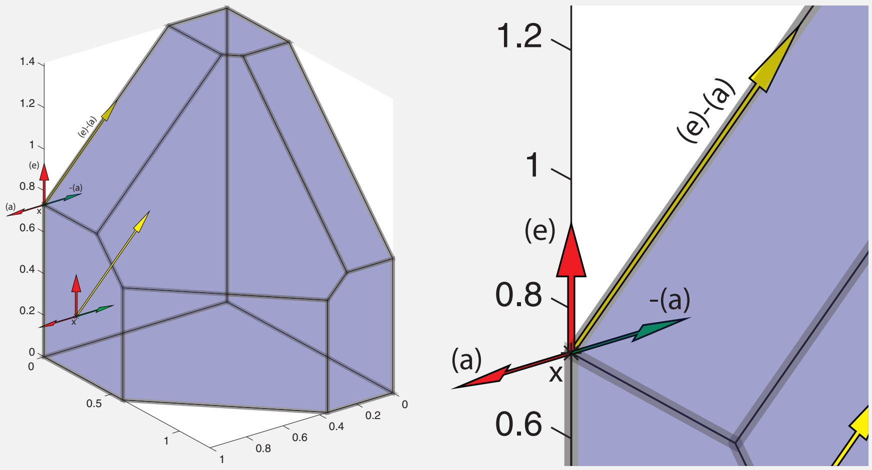
Left: $A \cap \text{dep}(x, e) = \emptyset$, and we can't move further in (e) direction, and moving in any negative $a \in A$ direction doesn't change that. Notice no dependence between (e) and any element in A.



Right: $A \subseteq \text{dep}(x, e)$, and we can't move further in the (e) direction, but we can move further in (e) direction by moving in some $a \in A$ negative direction. Notice dependence between (e) and elements in A.

Dependence Function and exchange in 3D

- We can move neither in the (e) nor the (a) direction, but we can move in the (e) direction if we simultaneously move in the -(a) direction.
- In 3D, we have:



- I.e., for $e \in \operatorname{sat}(x)$, $a \in \operatorname{sat}(x)$, $a \in \operatorname{dep}(x, e)$, $e \notin \operatorname{dep}(x, a)$, and

$$\operatorname{dep}(x, e) = \{a : a \in E, \exists \alpha > 0 : x + \alpha(\mathbf{1}_e - \mathbf{1}_a) \in P_f\} \tag{15.59}$$

- We next show this formally.

dep and exchange derived

- The derivation for $\operatorname{dep}(x, e)$ involves turning a strict inequality into a non-strict one with a strict explicit slack variable α:

$$\operatorname{dep}(x, e) = \text{ntight}(x, e) = \tag{15.60}$$

$$= \{e' : x(A) < f(A), \forall A \not\ni e', e \in A\} \tag{15.61}$$

$$= \{e' : \exists \alpha > 0, \text{ s.t. } \alpha \leq f(A) - x(A), \forall A \not\ni e', e \in A\} \tag{15.62}$$

$$= \{e' : \exists \alpha > 0, \text{ s.t. } \alpha \mathbf{1}_e(A) \leq f(A) - x(A), \forall A \not\ni e', e \in A\} \tag{15.63}$$

$$= \{e' : \exists \alpha > 0, \text{ s.t. } \alpha(\mathbf{1}_e(A) - \mathbf{1}_{e'}(A)) \leq f(A) - x(A), \forall A \not\ni e', e \in A\} \tag{15.64}$$

$$= \{e' : \exists \alpha > 0, \text{ s.t. } x(A) + \alpha(\mathbf{1}_e(A) - \mathbf{1}_{e'}(A)) \leq f(A), \forall A \not\ni e', e \in A\} \tag{15.65}$$

- Now, $1_e(A) - \mathbf{1}_{e'}(A) = 0$ if either $\{e, e'\} \subseteq A$, or $\{e, e'\} \cap A = \emptyset$.
- Also, if $e' \in A$ but $e \notin A$, then $x(A) + \alpha(\mathbf{1}_e(A) - \mathbf{1}_{e'}(A)) = x(A) - \alpha \leq f(A)$ since $x \in P_f$.

dep and exchange derived

- thus, we get the same in the above if we remove the constraint $A \not\ni e', e \in A$, that is we get

$$\text{dep}(x, e) = \left\{e' : \exists \alpha > 0, \text{ s.t. } x(A) + \alpha(\mathbf{1}_e(A) - \mathbf{1}_{e'}(A)) \leq f(A), \forall A\right\} \quad (15.66)$$

- This is then identical to

$$\text{dep}(x, e) = \left\{e' : \exists \alpha > 0, \text{ s.t. } x + \alpha(\mathbf{1}_e - \mathbf{1}_{e'}) \in P_f\right\} \quad (15.67)$$

- Compare with original, the minimal element of $\mathcal{D}(x, e)$, with $e \in \text{sat}(x)$:

$$\text{dep}(x, e) = \begin{cases} \bigcap \{A : e \in A \subseteq E, x(A) = f(A)\} & \text{if } e \in \text{sat}(x) \\ \emptyset & \text{else} \end{cases} \quad (15.68)$$

Summary of Concepts

- Most violated inequality $\max \{x(A) - f(A) : A \subseteq E\}$
- Matroid by circuits, and the fundamental circuit $C(I, e) \subseteq I + e$.
- Minimizers of submodular functions form a lattice.
- Minimal and maximal element of a lattice.
- x-tight sets, maximal and minimal tight set.
- sat function & Closure
- Saturation Capacity
- e-containing tight sets
- dep function & fundamental circuit of a matroid

Summary important definitions so far: tight, dep, & sat

- x-tight sets: For $x \in P_f$, $\mathcal{D}(x) = \{A \subseteq E : x(A) = f(A)\}$.
- Polymatroid closure/maximal x-tight set: For $x \in P_f$, $\text{sat}(x) = \cup\{A : A \in \mathcal{D}(x)\} = \{e : e \in E, \forall\alpha > 0, x + \alpha\mathbf{1}_e \notin P_f\}$.
- Saturation capacity: for $x \in P_f$, $0 \leq \hat{c}(x; e) = \min\{f(A) - x(A) | \forall A \ni e\} = \max\{\alpha : \alpha \in \mathbb{R}, x + \alpha\mathbf{1}_e \in P_f\}$
- Recall: $\text{sat}(x) = \{e : \hat{c}(x; e) = 0\}$ and $E \setminus \text{sat}(x) = \{e : \hat{c}(x; e) > 0\}$.
- e-containing x-tight sets: For $x \in P_f$, $\mathcal{D}(x, e) = \{A : e \in A \subseteq E, x(A) = f(A)\} \subseteq \mathcal{D}(x)$.
- Minimal e-containing x-tight set/polymatroidal fundamental circuit/: For $x \in P_f$,

$$\text{dep}(x, e) = \begin{cases} \bigcap\{A : e \in A \subseteq E, x(A) = f(A)\} & \text{if } e \in \text{sat}(x) \\ \emptyset & \text{else} \end{cases}$$

$$= \{e' : \exists\alpha > 0, \text{ s.t. } x + \alpha(\mathbf{1}_e - \mathbf{1}_{e'}) \in P_f\}$$