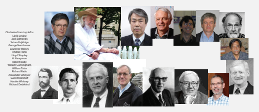
Submodular Functions, Optimization, and Applications to Machine Learning
— Fall Quarter, Lecture 15 —

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

Prof. Jeff Bilmes

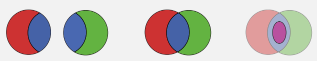
University of Washington, Seattle
Department of Electrical Engineering
http://melodi.ee.washington.edu/~bilmes

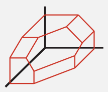
Nov 23rd, 2020



$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$$

$= f(A_r) + 2f(C) + f(B_r)$ $\quad = f(A_r) + f(C) + f(B_r)$ $\quad = f(A \cap B)$





Announcements, Assignments, and Reminders

- Homework 3, out, due Wednesday, Nov 25th, 2020, 11:59pm.
- Office hours this week, Tues (11/24) & Wed (11/25), 10:00pm at our class zoom link. I can meet Monday night at 10:00pm as well on request.

Class Road Map - EE563

- L1(9/30): Motivation, Applications, Definitions, Properties
- L2(10/5): Sums concave(modular), uses (diversity/costs, feature selection), information theory
- L3(10/7): Monge, More Definitions, Graph and Combinatorial Examples,
- L4(10/12): Graph & Combinatorial Examples, Matrix Rank, Properties, Other Defs, Independence
- L5(10/14): Properties, Defs of Submodularity, Independence
- L6(10/19): Matroids, Matroid Examples, Matroid Rank,
- L7(10/21): Matroid Rank, More on Partition Matroid, Laminar Matroids, System of Distinct Reps, Transversals
- L8(10/26): Transversal Matroid, Matroid and representation, Dual Matroid
- L9(10/28): Other Matroid Properties, Combinatorial Geometries, Matroid and Greedy, Polyhedra, Matroid Polytopes
- L10(11/2): Matroid Polytopes, Matroids $\rightarrow$ Polymatroids
- L11(11/4): Matroids $\rightarrow$ Polymatroids, Polymatroids
- L12(11/9): Polymatroids, Polymatroids and Greedy
- L–(11/11): Veterans Day, Holiday
- L13(11/16): Polymatroids and Greedy, Possible Polytopes, Extreme Points, Cardinality Constrained Maximization
- L14(11/18): Cardinality Constrained Maximization, Curvature
- L15(11/23): Curvature, Submodular Max w. Other Constraints, Start Cont. Extensions
- L16(11/25):
- L17(11/30):
- L18(12/2):
- L19(12/7):
- L20(12/9):
- L21(12/14): final meeting (presentations) maximization.

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

Rest of class

- Homework 4 will come out later this week, will be due about 1.5-2 weeks after that.
- Final project: Read and present a recent (past 5 years) paper on submodular/supermodular optimization. Paper should have both a theoretical and practical component. What is due: (1) 4-page paper summary, and (2) 10 minute presentation about the paper, will be giving presentations on Monday 12/14/2020. You must choose your paper before the 14th (this will be HW5), and you must turn in your slides and 4-page paper (this will be HW6).

The Greedy Algorithm for Submodular Max

A bit more precisely:

Algorithm 1: The Greedy Algorithm

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

Greedy Algorithm for Card. Constrained Submodular Max

- This algorithm has a guarantee

Theorem 15.2.1

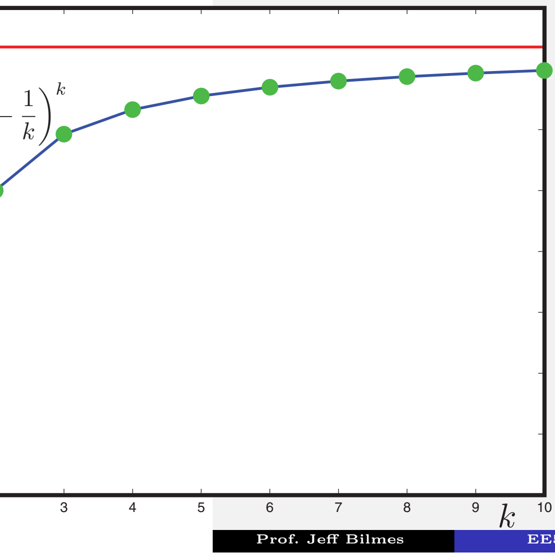
Given a polymatroid function f , the above greedy algorithm returns sets S_i such that for each i we have $f(S_i) \geq (1 - 1/e) \max_{|S| \leq i} f(S)$.

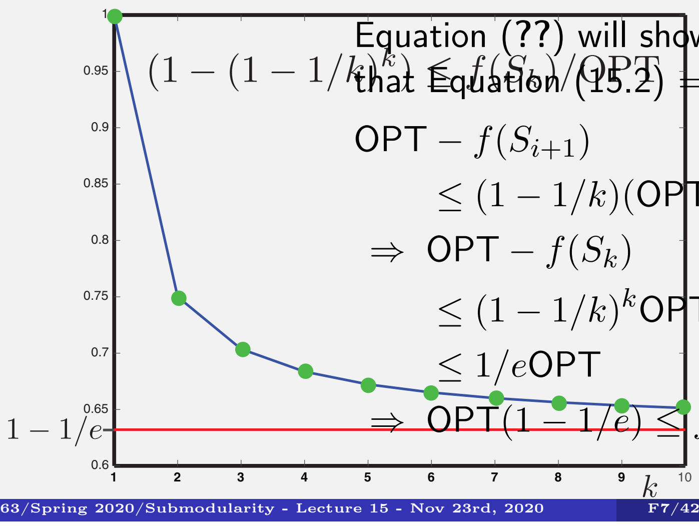
- To approximately find $A^* \in \operatorname{argmax}\{f(A) : |A| \leq k\}$, we repeat the greedy step until we have selected k elements in Algorithm 4.
- We can think of a "greedy operator" $\tilde{A} \in \operatorname{gargmax}\{f(A) : |A| \leq k\}$
- Again, since this generalizes max k-cover, Feige (1998) showed that this can't be improved. Unless $P = NP$, no polynomial time algorithm can do better than $(1 - 1/e + \epsilon)$ for any $\epsilon > 0$.

The Greedy Algorithm: $1 - 1/e$ 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:

$$\exists v \in V \setminus S_i : f(v|S_i) = f(S_i + v|S_i) \geq \frac{1}{k}(\text{OPT} - f(S_i)) \tag{15.2}$$





Equation (??) will show that Equation (15.2) $\Rightarrow$:

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

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

$$\text{insert}(Q, (v, \alpha)) \tag{15.15}$$

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

$$(v, \alpha) \leftarrow \text{pop}(Q) \tag{15.16}$$

  - Query the value of the max item in the queue

$$\max(Q) \in \mathbb{R} \tag{15.17}$$

- 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

Minoux's Accelerated Greedy Algorithm Submodular Max

Algorithm 2: Minoux's Accelerated Greedy Algorithm

```
1  Set $S_0 \leftarrow \emptyset$ ; $i \leftarrow 0$ ; Initialize priority queue $Q$ ;
2  for $v \in E$ do
3      INSERT$(Q, f(v))$
4  repeat
5      $(v, \alpha) \leftarrow$ pop$(Q)$ ;
6      if $\alpha$ not "fresh" then
7          recompute $\alpha \leftarrow f(v|S_i)$
8      if (popped $\alpha$ in line 5 was "fresh") OR ($\alpha \geq$ max$(Q)$) then
9          Set $S_{i+1} \leftarrow S_i \cup \{v\}$ ; and mark other items in $Q$ as stale
           ;
10         $i \leftarrow i + 1$ ;
11     else
12         insert$(Q, (v, \alpha))$ as a fresh item
13 until $i = |E|$;
```

(Minimum) Submodular Set Cover

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

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

  where α is a "cover" requirement.

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

$$S^* \in \operatorname*{argmin}_{S \subseteq V} |S| \text{ such that } f'(S) \geq f'(V) \tag{15.16}$$

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

$$|S^{\mathsf{G}}| \leq |S^*| H(\max_{s \in V} f(\{s\})) = |S^*| O(\log_e(\max_{s \in V} f(\{s\}))) \tag{15.15}$$

  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:

$$|S^{\mathsf{G}}| \leq |S^*| \left(1 + \log_e \frac{f(V)}{f(V) - f(S_{T-1})}\right) \tag{15.16}$$

  where S_T is the greedy solution that occurs at step T, where T is the number of iterations the algorithm runs until threshold is reached.
- As we mentioned earlier, even set cover (a special case of submodular set cover) is hard to approximate with a factor better than $(1 - \epsilon) \log \alpha$, where α is the desired cover constraint.

Curvature of a Submodular function

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

$$c_f \triangleq 1 - \min_{S, j \notin S : f(j|\emptyset) \neq 0} \frac{f(j|S)}{f(j|\emptyset)} = 1 - \min_{j : f(j|\emptyset) \neq 0} \frac{f(j|V \setminus \{j\})}{f(j|\emptyset)} \tag{15.20}$$

- Note: Matroid rank is either modular $c_r = 0$ or maximally curved $c_r = 1$. thus, matroid rank can have only the extreme points of curvature, namely 0 or 1.
- Polymatroid functions are, however, more nuanced, in that they allow non-extreme curvature, with $c_f \in (0, 1)$.
- Recall the notion of "partial dependence" within polymatroid functions.

Curvature and approximation: key theorem

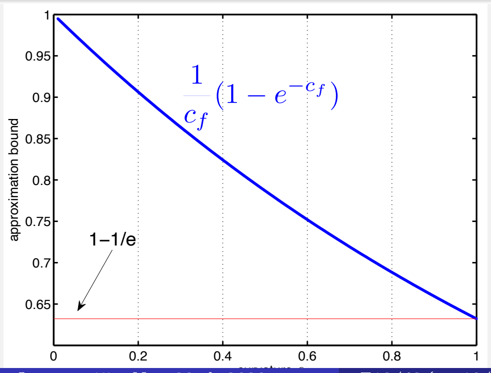
- Curvature-based approximation bound for max k cardinality.

Theorem 15.2.2

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

$$\frac{1}{c_f}(1 - e^{-c_f}) \tag{15.20}$$

- $$f(\tilde{S}_{\text{greedy}}) \geq \frac{1}{c_f}(1 - e^{-c_f})\text{OPT} \tag{15.21}$$



Submodular and Supermodular Curvature Approximation

- Let f be a polymatroid function and let g be a non-negative monotone non-decreasing supermodular function (e.g., $g(A) = \phi(m(A))$ where $\phi()$ is non-decreasing convex and $m : V \to \mathbb{R}_+$).
- Let $\kappa_f = 1 - \min_v \frac{f(v|V\setminus\{v\})}{f(v)}$ be the submodular total curvature,
- Define $\kappa^g = 1 - \min_v \frac{g(v)}{g(v|V\setminus\{v\})}$ as a "supermodular curvature"
- $\kappa^g \in [0,1]$ and $\kappa^g = 0$ means g is modular, $\kappa^g = 1$ means g is "fully curved"
- Form function $h(A) = f(A) + g(A)$, then h is neither suBmodular nor suPermodular, but is known as a BP-function.

SuBmodular and SuPermodular (BP function) Curvature Approximation

- We have the following:

Theorem 15.3.1

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

$$\frac{1}{\kappa_f}(1 - e^{-(1-\kappa_g)\kappa_f}) \tag{15.1}$$

- For purely supermodular optimization (i.e., $\kappa_f = 0$) we get that greedy has a guarantee of $1 - \kappa_g$.
- Both curvatures are very easy to compute given BP decomposition.

Submodular Analysis for Non-Submodular Problems

- BP functions are an example of when quality of solutions to non-submodular problems can be analyzed via submodularity since BP functions are neither submodular nor supermodular.
- Another example: "deviation from submodularity" can be measured using the submodularity ratio (Das & Kempe) that we saw in HW1:

$$\gamma_{U,k}(f) \triangleq \min_{L \subseteq U, S : |S| \leq k, S \cap L = \emptyset} \frac{\sum_{s \in S} f(x|L)}{f(S|L)} \tag{15.2}$$

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

$$\text{Solution} \geq (1 - \frac{1}{e^{\gamma_{U^*,k}}})\text{OPT} \tag{15.3}$$

where U^* is the solution set of a variable selection algorithm.
- This gradually get worse as we move away from an objective being submodular (see Das & Kempe, 2011).
- Another analogous concepts, submodular degree.

Generalizations

- Consider a k-uniform matroid $\mathcal{M} = (V, \mathcal{I})$ where $\mathcal{I} = \{S \subseteq V : |S| \leq k\}$, and consider problem $\max\{f(A) : A \in \mathcal{I}\}$
- Hence, the greedy algorithm is $1 - 1/e$ optimal for maximizing polymatroidal f subject to a k-uniform matroid constraint.
- Might be useful to allow an arbitrary matroid (e.g., partition matroid $\mathcal{I} = \{X \subseteq V : |X \cap V_i| \leq k_i$ for all $i = 1, \ldots, \ell\}$., or a transversal, etc).
- Knapsack constraint: if each item $v \in V$ has a cost $c(v) \geq 0$, we may ask for $c(S) \leq b$ where $b \geq 0$ is a budget, in units of costs. Q: Is $\mathcal{I} = \{I : c(I) \leq b\}$ the independent sets of a matroid?
- We may wish to maximize f subject to multiple matroid constraints. I.e., $S \in \mathcal{I}_1, S \in \mathcal{I}_2, \ldots, S \in \mathcal{I}_p$ where $\mathcal{I}_i$ are independent sets of the i^{th} matroid.
- Combinations of the above (e.g., knapsack & multiple matroid constraints).

Greedy over multiple matroids

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

$$S_{i+1} = S_i \cup \left\{ \operatorname*{argmax}_{v \in V \setminus S_i \,:\, S_i + v \in \bigcap_{i=1}^{p} \mathcal{I}_i} f(S_i \cup \{v\}) \right\} \tag{15.4}$$

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

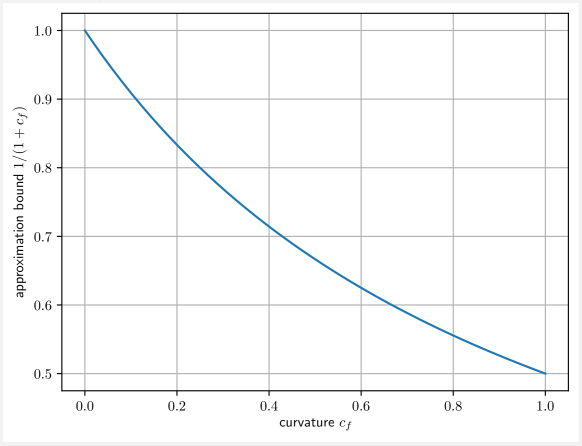
Theorem 15.4.1

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

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

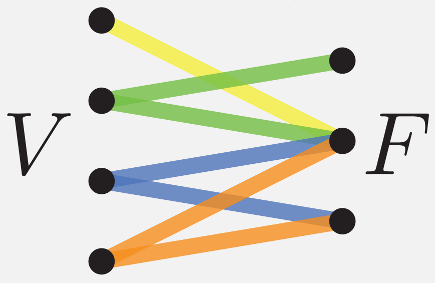
Curvature approximation with matroid constraints

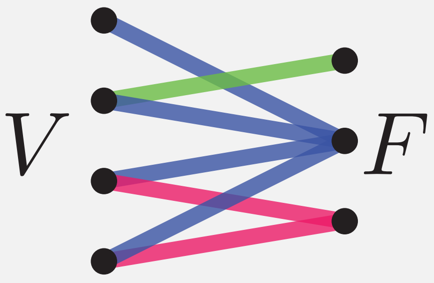
- Conforti & Cornuéjols showed that greedy gives a $1/(1+c_f)$ approximation to $\max\{f(S) : S \in \mathcal{I}\}$ when f has total curvature c.
- Hence, greedy subject to matroid constraint is a $\max(1/(1+c_f), 1/2)$ approximation algorithm, and if $c_f < 1$ then it is better than 1/2 (e.g., with $c_f = 1/4$ then we have a 0.8 algorithm).



Matroid Intersection and Bipartite Matching

- Why might we want to do matroid intersection?
- Consider bipartite graph $G = (V, F, E)$. Define two partition matroids $M_V = (E, \mathcal{I}_V)$, and $M_F = (E, \mathcal{I}_F)$.
- Independence in each matroid corresponds to:
  1. $I \in \mathcal{I}_F$ if $|I \cap (V, f)| \leq 1$ for all $f \in F$,
  2. and $I \in \mathcal{I}_V$ if $|I \cap (v, F)| \leq 1$ for all $v \in V$.





- Therefore, a matching in G is simultaneously independent in both M_V and M_F and finding the maximum matching is finding the maximum cardinality set independent in both matroids.
- In bipartite graph case, therefore, can be solved in polynomial time.

Matroid Intersection and Network Communication

- Let $G_1 = (V_1, E)$ and $G_2 = (V_2, E)$ be two graphs on an isomorphic set of edges (lets just give them same names E).
- Consider two cycle matroids associated with these graphs $M_1 = (E, \mathcal{I}_1)$ and $M_2 = (E, \mathcal{I}_2)$. They might be very different (e.g., an edge might be between two distinct nodes in G_1 but the same edge is a loop in multi-graph G_2.)
- We may wish to find the maximum size edge-induced subgraph that is still forest in both graphs (i.e., adding any edges will create a circuit in either M_1, M_2, or both).
- This is again a matroid intersection problem.

Matroid Intersection and TSP

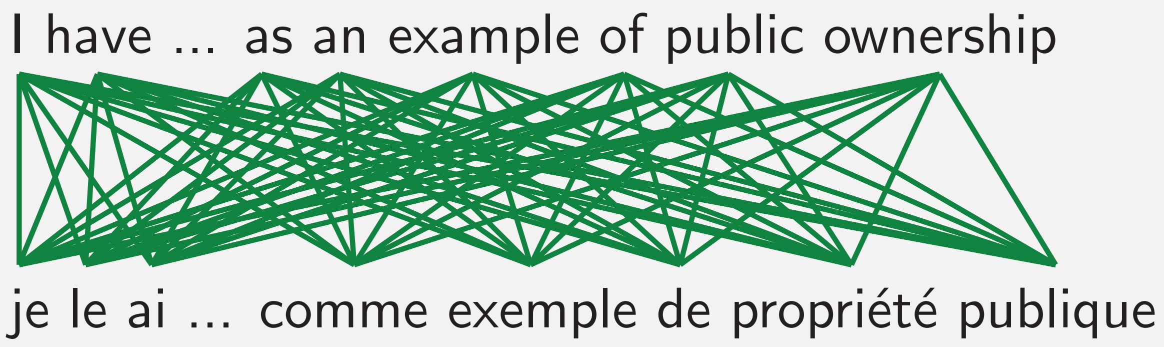
- Definition: a Hamiltonian cycle is a cycle that passes through each node of a graph exactly once.
- Given directed graph G, goal is to find such a Hamiltonian cycle.
- From G with n nodes, create G' with $n+1$ nodes by duplicating (w.l.o.g.) a particular node $v_1 \in V(G)$ to v_1^+, v_1^-, and have all outgoing edges from v_1 come instead from v_1^- and all edges incoming to v_1 go instead to v_1^+.
- Let M_1 be the cycle matroid on G' (I independent if no cycles).
- Let M_2 be the partition matroid having as independent sets those that have no more than one edge leaving any node — i.e., $I \in \mathcal{I}(M_2)$ if $|I \cap \delta^-(v)| \leq 1$ for all $v \in V(G')$.
- Let M_3 be the partition matroid having as independent sets those that have no more than one edge entering any node — i.e., $I \in \mathcal{I}(M_3)$ if $|I \cap \delta^+(v)| \leq 1$ for all $v \in V(G')$.
- Then a Hamiltonian cycle exists iff there is an n-element intersection of M_1, M_2, and M_3.
- Recall, the traveling salesperson problem (TSP) is the problem to given a directed graph, start at a node, visit all cities, and return to the starting point. Optimization version does this tour at minimum cost

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.

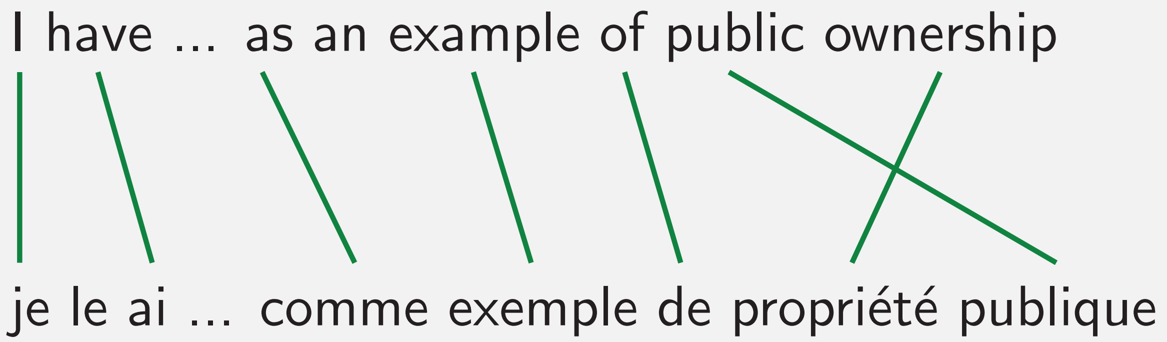
Greedy over > 1 matroids: Multiple Language Alignment

- 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

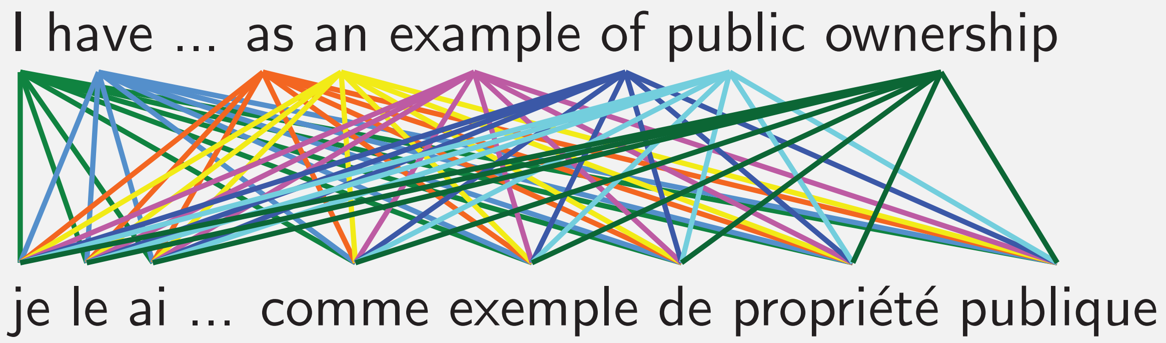
Greedy over > 1 matroids: Multiple Language Alignment

- 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

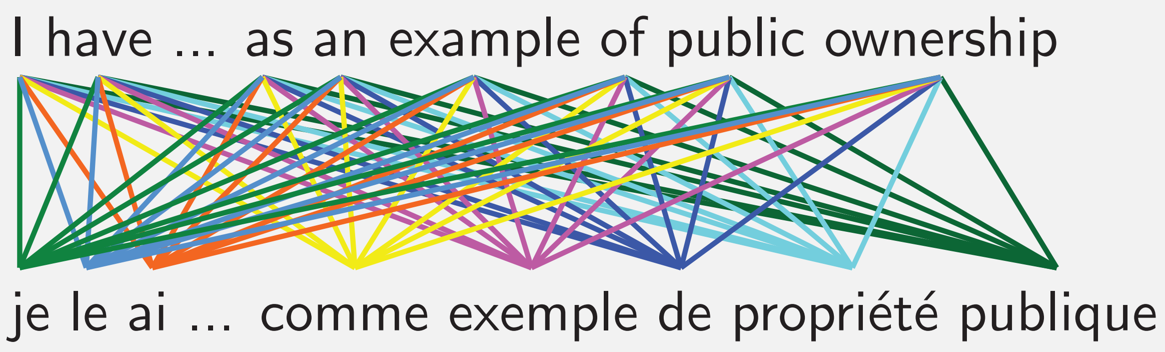
- Edges incident to English words constitute an edge partition


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

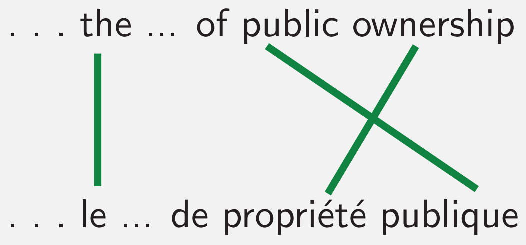

I have ... as an example of public ownership
je le ai ... comme exemple de propriété publique

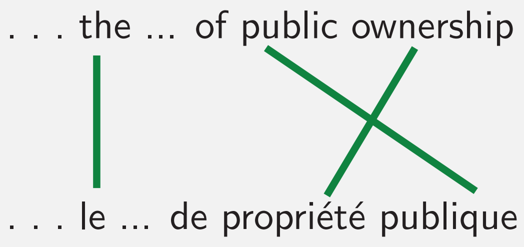
- The two edge partitions can be used to set up two 1-partition matroids on the edges.
- For each matroid, a set of edges is independent only if the set intersects each partition block no more than one time.

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

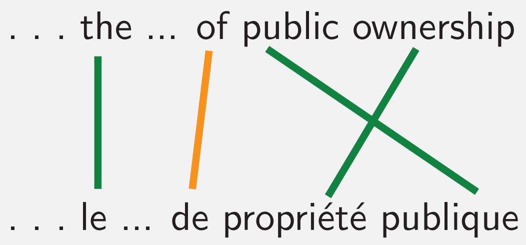

. . . the ... of public ownership
. . . le ... de propriété publique


. . . the ... of public ownership
. . . le ... de propriété publique

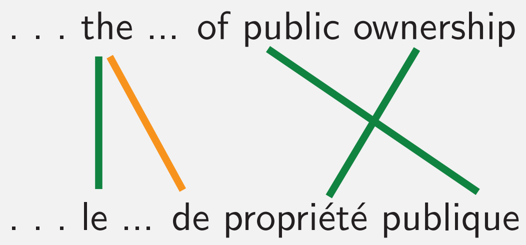
Greedy over > 1 matroids: Multiple Language Alignment

- 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



. . . the ... of public ownership
. . . le ... de propriété publique



. . . the ... of public ownership
. . . le ... de propriété publique

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 player has a submodular "valuation" function, $g_i : 2^E \to \mathbb{R}_+$, $g_i(A)$ measures how "desirable" or "valuable" 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:

$$\text{submodular-social-welfare}(E_1, E_2, \ldots, E_n) = \sum_{i=1}^{n} g_i(E_i). \tag{15.5}$$

- 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.,

$$E' = \underbrace{E \uplus E \uplus \cdots \uplus E}_{n\times} \tag{15.6}$$

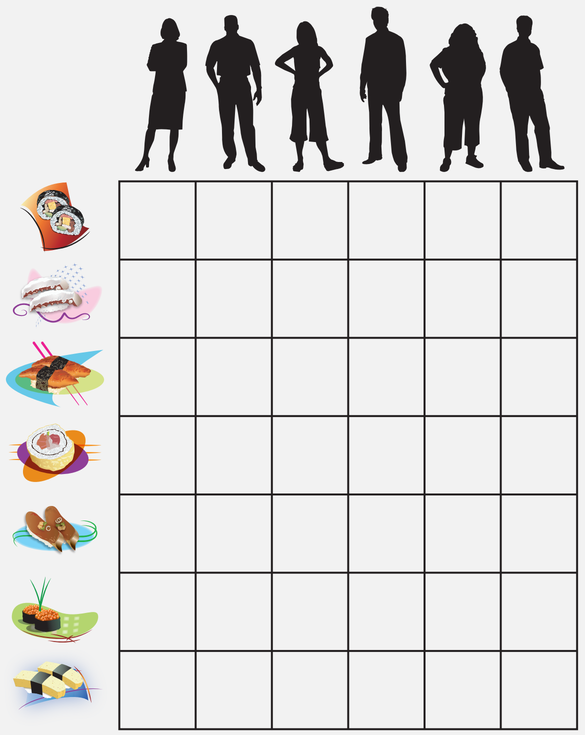
- Let $E^{(i)} \subset E'$ be the i^{th} block of E'.
- For any $e \in E$, the corresponding element in $E^{(i)}$ is called $(e,i) \in E^{(i)}$ (each original element is tagged by integer).
- For $e \in E$, define $E_e = \{(e', i) \in E' : e' = e\}$.
- Hence, $\{E_e\}_{e \in E}$ is a partition of E',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

$$\mathcal{I} = \left\{ S \subseteq E' : \forall e \in E, |S \cap E_e| \le 1 \right\} \tag{15.7}$$

Submodular Welfare: Submodular Max over matroid partition

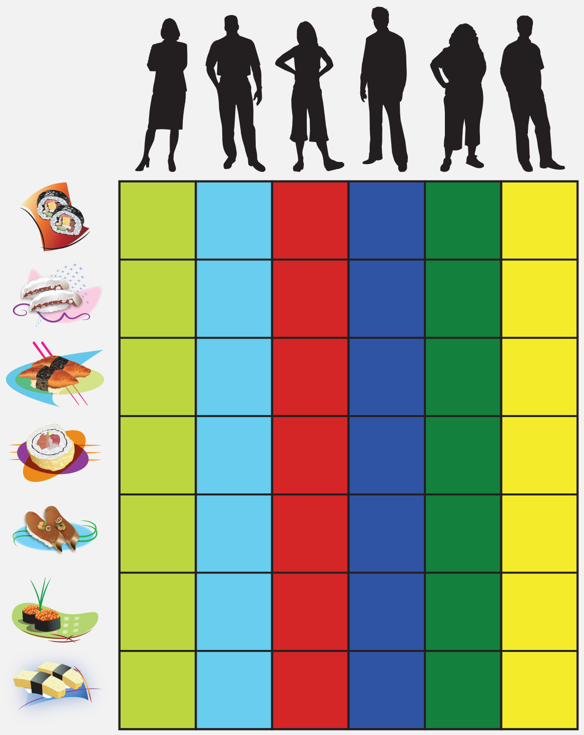
- Hence, S is independent in matroid $\mathcal{M} = (E', I)$ if S uses each original element no more than once.
- Create submodular function $f' : 2^{E'} \to \mathbb{R}_+$ with $f'(S) = \sum_{i=1}^n g_i(S \cap E^{(i)})$.
- Submodular welfare maximization becomes matroid constrained submodular max $\max\{f'(S)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

Submodular Social Welfare



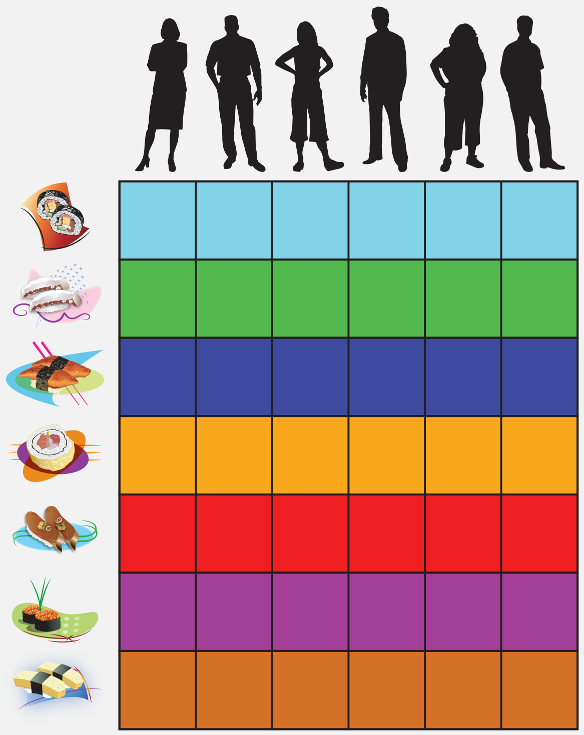
- Have $n = 6$ people (who don't like to share) and $|E| = m = 7$ pieces of sushi. E.g., $e \in E$ might be $e =$ "salmon roll".
- Goal: distribute sushi to people to maximize social welfare.
- Ground set disjoint union $E \uplus E \uplus E \uplus E \uplus E \uplus E$.
- Partition matroid partitions: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

Submodular Social Welfare



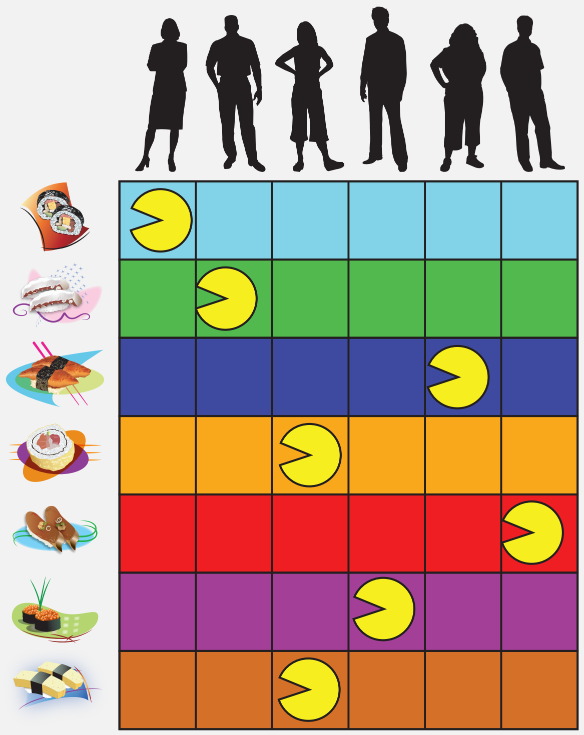
- Have $n = 6$ people (who don't like to share) and $|E| = m = 7$ pieces of sushi. E.g., $e \in E$ might be $e =$ "salmon roll".
- Goal: distribute sushi to people to maximize social welfare.
- Ground set disjoint union $E \uplus E \uplus E \uplus E \uplus E \uplus E$.
- Partition matroid partitions: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

Submodular Social Welfare



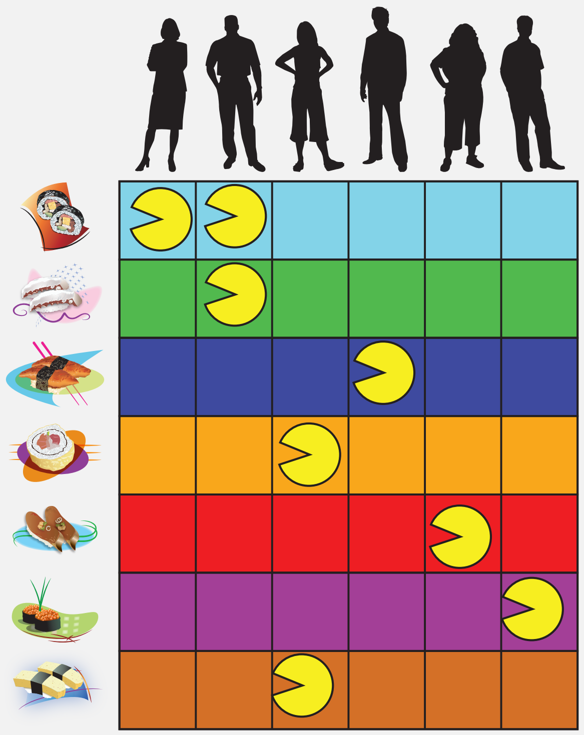
- Have $n = 6$ people (who don't like to share) and $|E| = m = 7$ pieces of sushi. E.g., $e \in E$ might be $e =$ "salmon roll".
- Goal: distribute sushi to people to maximize social welfare.
- Ground set disjoint union $E \uplus E \uplus E \uplus E \uplus E \uplus E$.
- Partition matroid partitions: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

$$S_{i+1} = S_i \cup \left\{ \operatorname*{argmax}_{v \in V \setminus S_i} \Big(f(S_i \cup \{v\}) - f(S_i) \Big) \right\} \tag{15.8}$$

  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

$$S_{i+1} = S_i \cup \left\{ \operatorname*{argmax}_{v \in V \setminus S_i} \frac{f(S_i \cup \{v\}) - f(S_i)}{c(v)} \right\} \tag{15.9}$$

  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.

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.

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 15.4.2

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 15.4.2

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

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 3: Randomized Linear-time non-monotone submodular max

```
1  Set $L \leftarrow \emptyset$ ; $U \leftarrow V$     /* Lower $L$, upper $U$. Invariant: $L \subseteq U$ */ ;
2  Order elements of $V = (v_1, v_2, \ldots, v_n)$ arbitrarily ;
3  for $i \leftarrow 0 \ldots |V|$ do
4      $a \leftarrow [f(v_i|L)]_+$; $b \leftarrow [-f(U|U \setminus \{v_i\})]_+$ ;
5      if $a = b = 0$ then $p \leftarrow 1/2$ ;
6      ;
7      else $p \leftarrow a/(a+b)$;
8      ;
9      if Flip of coin with $\Pr$(heads) $= p$ draws heads then
10         $L \leftarrow L \cup \{v_i\}$ ;
11     Otherwise /* if the coin drew tails, an event with prob. $1 - p$ */
12         $U \leftarrow U \setminus \{v_i\}$
13 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.
- But note, even a random subset is a $1/4$ approximation to the optimal solution for unconstrained non-monotone submodular maximization, in expectation (Feige, Mirrokni, Vondrak, Maximizing non-monotone submodular functions. SIAM Journal on Computing, 40(4):1133-1153, 2011.)

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/`).

Venn Family of Subclusive Constraints

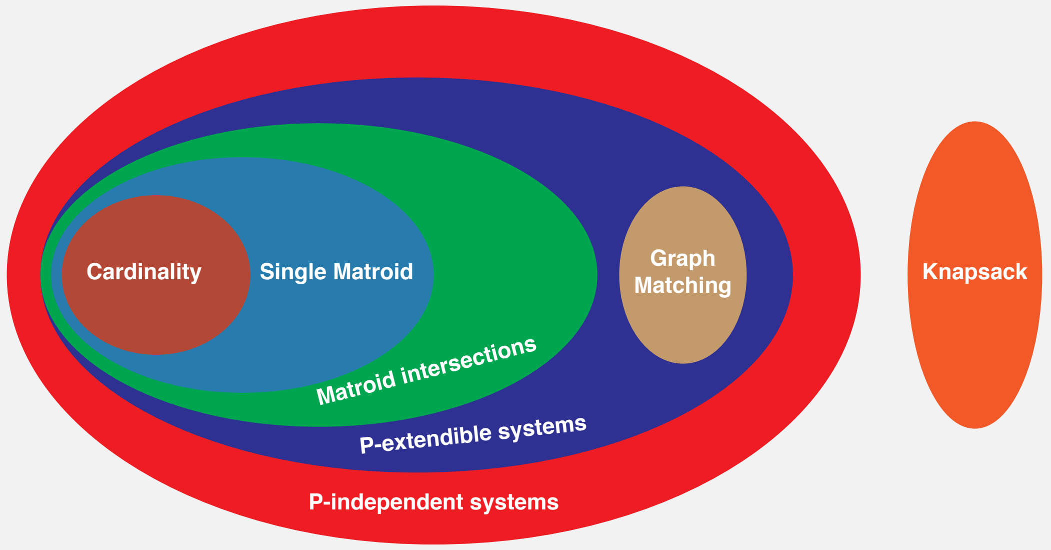


Figure idea from Amin Karbasi

Submodular Max Summary - From J. Vondrak

Monotone Maximization

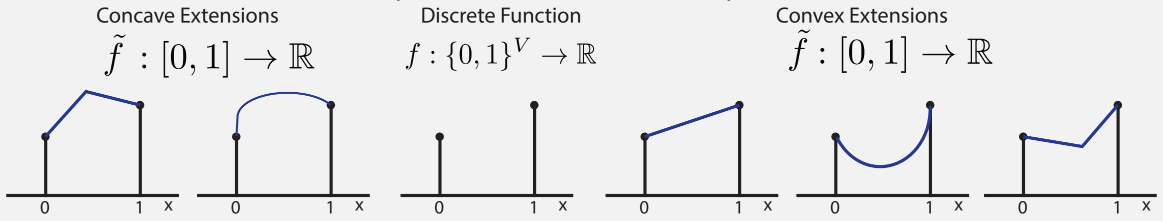
Constraint	Approximation	Hardness	Technique
$\|S\| \leq k$	$1-1/e$	$1-1/e$	greedy
matroid	$1-1/e$	$1-1/e$	multilinear ext.
$O(1)$ knapsacks	$1-1/e$	$1-1/e$	multilinear ext.
k matroids	$k+\epsilon$	$k/\log k$	local search
k matroids and $O(1)$ knapsacks	$O(k)$	$k/\log k$	multilinear ext.

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.

Continuous Extensions of Discrete Set Functions

- Any function $f : 2^V \to \mathbb{R}$ (equivalently $f : \{0,1\}^V \to \mathbb{R}$) can be extended to a continuous function in the sense $\tilde{f} : [0,1]^V \to \mathbb{R}$.
- This may be tight (i.e., $\tilde{f}(\mathbf{1}_A) = f(A)$ for all A). I.e., the extension $\tilde{f}$ coincides with f at the hypercube vertices.
- In fact, any such discrete function defined on the vertices of the n-D hypercube $\{0,1\}^n$ has a variety of both convex and concave extensions tight at the vertices (Crama & Hammer'11). Example $n = 1$,


Concave Extensions
$\tilde{f} : [0,1] \to \mathbb{R}$
Discrete Function
$f : \{0,1\}^V \to \mathbb{R}$
Convex Extensions
$\tilde{f} : [0,1] \to \mathbb{R}$
0
1
x

- Since there are an exponential number of vertices $\{0,1\}^n$, important questions regarding such extensions is:
  1. When are they computationally feasible to obtain or estimate?
  2. When do they have nice mathematical properties?
  3. When are they useful for something practical?

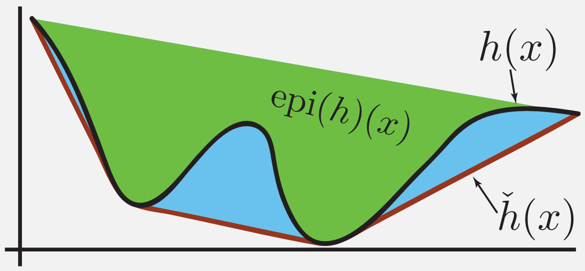
Def: Convex Envelope of a function

- Given any function $h : \mathbb{R}^n \to \mathbb{R}$, define new function $\check{h} : \mathbf{R}^n \to \mathbb{R}$ via:

$$\check{h}(x) = \sup\{g(x) : g \text{ is convex } \& \ g(y) \leq h(y), \forall y \in \mathbb{R}^n\} \qquad (15.10)$$

- I.e., (1) $\check{h}(x)$ is convex, (2) $\check{h}(x) \leq h(x), \forall x$, and (3) if $g(x)$ is any convex function having the property that $g(x) \leq h(x), \forall x$, then $g(x) \leq \check{h}(x)$.
- Alternatively,

$$\check{h}(x) = \inf\{t : (x,t) \in \text{convexhull}(\text{epigraph}(h))\} \qquad (15.11)$$


$h(x)$
$\text{epi}(h)(x)$
$\check{h}(x)$