Optimization of Submodular Functions
Tutorial - lecture I

Jan Vondrák

1IBM Almaden Research Center
San Jose, CA
Lecture I: outline

1. Submodular functions: what and why?
2. Convex aspects: Submodular minimization
3. Concave aspects: Submodular maximization
There are many problems that we study in combinatorial optimization... Max Matching, Min Cut, Max Cut, Min Spanning Tree, Max SAT, Max Clique, Vertex Cover, Set Cover, Max Coverage, ....

They are all problems in the form

$$\max \{ f(S) : S \in \mathcal{F} \}$$

$$\min \{ f(S) : S \in \mathcal{F} \}$$

where $\mathcal{F}$ is a discrete set of feasible solutions.
Combinatorial optimization

There are many problems that we study in combinatorial optimization... Max Matching, Min Cut, Max Cut, Min Spanning Tree, Max SAT, Max Clique, Vertex Cover, Set Cover, Max Coverage, ....

They are all problems in the form

$$\max\{f(S) : S \in \mathcal{F}\}$$
$$\min\{f(S) : S \in \mathcal{F}\}$$

where $\mathcal{F}$ is a discrete set of feasible solutions.

We can

- try to deal with each problem individually, or
There are many problems that we study in combinatorial optimization... Max Matching, Min Cut, Max Cut, Min Spanning Tree, Max SAT, Max Clique, Vertex Cover, Set Cover, Max Coverage, ....

They are all problems in the form

\[
\max \{ f(S) : S \in \mathcal{F} \} \\
\min \{ f(S) : S \in \mathcal{F} \}
\]

where \( \mathcal{F} \) is a discrete set of feasible solutions.

We can
- try to deal with each problem individually, or
- try to capture some properties of \( f, \mathcal{F} \) that make it tractable.
Continuous optimization

What are such properties in continuous optimization?
Continuous optimization

What are such properties in *continuous optimization*?

A function \( f : \mathbb{R}^n \rightarrow \mathbb{R} \) can be minimized efficiently, if it is convex.

A function \( f : \mathbb{R}^n \rightarrow \mathbb{R} \) can be maximized efficiently, if it is concave.
Continuous optimization

What are such properties in continuous optimization?

A function \( f : \mathbb{R}^n \to \mathbb{R} \)

- can be minimized efficiently, if it is convex.

A function \( f : \mathbb{R}^n \to \mathbb{R} \)

- can be maximized efficiently, if it is concave.

Discrete analogy?

Not so obvious... \( f \) is now a set function, or equivalently

\[
f : \{0, 1\}^n \to \mathbb{R}.
\]
Concavity:

\[ f : \mathbb{R} \to \mathbb{R} \text{ is concave,} \]

\[ \text{if the derivative } f'(x) \text{ is non-increasing in } x. \]
From concavity to submodularity

**Concavity:**

\( f : \mathbb{R} \rightarrow \mathbb{R} \) is concave, if the derivative \( f'(x) \) is non-increasing in \( x \).

**Submodularity:**

\( f : \{0, 1\}^n \rightarrow \mathbb{R} \) is submodular, if \( \forall i \), the discrete derivative

\[
\partial_i f(x) = f(x + e_i) - f(x)
\]

is non-increasing in \( x \).
Equivalent definitions

(1) Define the *marginal value of element j*,

\[ f_S(j) = f(S \cup \{j\}) - f(S). \]

\( f \) is submodular, if \( \forall S \subset T, j \notin T \):

\[ f_S(j) \geq f_T(j). \]
Equivalent definitions

(1) Define the *marginal value of element* \( j \),
\[
f_S(j) = f(S \cup \{j\}) - f(S).
\]

\( f \) is submodular, if \( \forall S \subseteq T, j \notin T \):
\[
f_S(j) \geq f_T(j).
\]

(2) A function \( f : 2^N \rightarrow \mathbb{R} \) is submodular if for any \( S, T \),
\[
f(S \cup T) + f(S \cap T) \leq f(S) + f(T).
\]
Examples of submodular functions

Coverage function:
Given $A_1, \ldots, A_n \subset U$,

$$f(S) = \left| \bigcup_{j \in S} A_j \right|.$$
Examples of submodular functions

Coverage function:
Given \( A_1, \ldots, A_n \subset U \),

\[
f(S) = \left| \bigcup_{j \in S} A_j \right|.
\]

Cut function:

\[
\delta(T) = \left| e(T, \overline{T}) \right|
\]
Concave or convex?

So, is submodularity more like concavity or convexity?
So, is submodularity more like concavity or convexity?

- **Argument for concavity:** Definition looks more like concavity - *non-increasing* discrete derivatives.
- **Argument for convexity:** Submodularity seems to be more useful for *minimization* than maximization.
So, is submodularity more like concavity or convexity?

- **Argument for concavity:** Definition looks more like concavity - non-increasing discrete derivatives.
- **Argument for convexity:** Submodularity seems to be more useful for minimization than maximization.


There is an algorithm that computes the minimum of any submodular function $f: \{0, 1\}^n \rightarrow \mathbb{R}$ in $\text{poly}(n)$ time (using value queries, $f(S) = ?$).

**In contrast:**
Maximizing a submodular function (e.g. Max Cut) is NP-hard.
Lecture I: outline

1. Submodular functions: what and why?
2. Convex aspects: Submodular minimization
3. Concave aspects: Submodular maximization
Convex aspects of submodular functions

Why is it possible to minimize submodular functions?

- The combinatorial algorithms are sophisticated...
- But there is a simple explanation: the Lovász extension.
Convex aspects of submodular functions

Why is it possible to minimize submodular functions?

- The combinatorial algorithms are sophisticated...
- But there is a simple explanation: the Lovász extension.

Submodular function

\[ f : \{0, 1\}^n \to \mathbb{R} \]

(Convex) Continuous function

\[ f^L : [0, 1]^n \to \mathbb{R} \]

- If \( f \) is submodular, then \( f^L \) is convex.
- Therefore, \( f^L \) can be minimized efficiently.
- A minimizer of \( f^L(x) \) can be converted into a minimizer of \( f(S) \).
The Lovász extension

Definition

Given \( f : \{0, 1\}^n \rightarrow \mathbb{R} \), its Lovász extension \( f^L : [0, 1]^n \rightarrow \mathbb{R} \) is

\[
f^L(x) = \sum_{i=0}^{n} \alpha_i f(S_i)
\]

where \( x = \sum \alpha_i 1_{S_i}, \sum \alpha_i = 1, \alpha_i \geq 0 \) and \( \emptyset = S_0 \subset S_1 \subset \ldots \subset S_n \).
The Lovász extension

Definition

Given $f : \{0, 1\}^n \rightarrow \mathbb{R}$, its Lovász extension $f^L : [0, 1]^n \rightarrow \mathbb{R}$ is

$$f^L(x) = \sum_{i=0}^{n} \alpha_i f(S_i)$$

where $x = \sum \alpha_i 1_{S_i}$, $\sum \alpha_i = 1$, $\alpha_i \geq 0$ and $\emptyset = S_0 \subset S_1 \subset \ldots \subset S_n$.

Equivalently:

$$f^L(x) = \mathbb{E}[f(T_\lambda(x))],$$

where $T_\lambda(x) = \{i : x_i > \lambda\}$, $\lambda \in [0, 1]$ uniformly random.
Lovász extension:

\[ f^L(x) = \mathbb{E}[f(T_\lambda(x))], \]
where \( T_\lambda(x) = \{i : x_i > \lambda\} \), \( \lambda \in [0, 1] \) uniformly random.

Properties:

- \( f^L \) is an extension: \( f^L(x) = f(x) \) for \( x \in \{0, 1\}^n \).
Minimizing the Lovász extension

Lovász extension:

\[ f^L(x) = \mathbb{E}[f(T_\lambda(x))], \]
where \( T_\lambda(x) = \{i : x_i > \lambda\} \), 
\( \lambda \in [0, 1] \) uniformly random.

Properties:

- \( f^L \) is an extension: \( f^L(x) = f(x) \) for \( x \in \{0, 1\}^n \).
- \( f \) is submodular \( \iff \) \( f^L \) is convex (in fact the "convex closure" of \( f \)).
Minimizing the Lovász extension

Lovász extension:

$$f^L(x) = \mathbb{E}[f(T_\lambda(x))],$$

where $$T_\lambda(x) = \{i : x_i > \lambda\}$$, $$\lambda \in [0, 1]$$ uniformly random.

Properties:

- $$f^L$$ is an extension: $$f^L(x) = f(x)$$ for $$x \in \{0, 1\}^n$$.
- $$f$$ is submodular $$\iff$$ $$f^L$$ is convex (in fact the "convex closure" of $$f$$).
- Therefore, $$f^L$$ can be minimized (by the ellipsoid method, in weakly polynomial time).
Lovász extension:

\[
f^L(x) = \mathbb{E}[f(T_\lambda(x))],
\]
where \( T_\lambda(x) = \{ i : x_i > \lambda \} \), \( \lambda \in [0, 1] \) uniformly random.

Properties:

- \( f^L \) is an extension: \( f^L(x) = f(x) \) for \( x \in \{0, 1\}^n \).
- \( f \) is submodular \(\iff\) \( f^L \) is convex (in fact the "convex closure" of \( f \)).
- Therefore, \( f^L \) can be minimized (by the ellipsoid method, in weakly polynomial time).
- Given a minimizer of \( f^L(x) \), we get a convex combination \( f^L(x) = \sum_{i=0}^{n} \alpha_i f(T_i) \), and one of the \( T_i \) is a minimizer of \( f(S) \).
Generalized submodular minimization

*Submodular functions can be minimized over restricted families of sets:*

- lattices, odd/even sets, $T$-odd sets, $T$-even sets  
  [Grötschel, Lovász, Schrijver ’81–’84]
- "parity families", including $\mathcal{L}_1 \setminus \mathcal{L}_2$ for lattices $\mathcal{L}_1, \mathcal{L}_2$  
  [Goemans, Ramakrishnan ’95]
- any down-closed constraint (excluding $\emptyset$), for symmetric submodular functions [Goemans, Soto ’10]
Generalized submodular minimization

**Submodular functions can be minimized over restricted families of sets:**

- lattices, odd/even sets, $T$-odd sets, $T$-even sets
  [Grötschel, Lovász, Schrijver ’81-’84]
- "parity families", including $\mathcal{L}_1 \setminus \mathcal{L}_2$ for lattices $\mathcal{L}_1, \mathcal{L}_2$
  [Goemans, Ramakrishnan ’95]
- any down-closed constraint (excluding $\emptyset$), for symmetric submodular functions [Goemans, Soto ’10]

However, a simple "covering" constraint can make submodular minimization hard:

- $\min \{ f(S) : |S| \geq k \}$
- $\min \{ f(T) : T \text{ is a spanning tree in } G \}$
- $\min \{ f(P) : P \text{ is a shortest path between } s - t \}$

**What about approximate solutions?**
Constrained submodular minimization

Bad news:
\[ \min\{f(S) : S \in \mathcal{F}\} \] becomes very hard for some simple constraints:
- \( n^{1/2}\)-hardness for \( \min\{f(S) : |S| \geq k\} \)
  [Goemans,Harvey,Iwata,Mirrokni ’09], [Svitkina,Fleischer ’09]
- \( n^{2/3}\)-hardness for \( \min\{f(P) : P \text{ is a shortest path}\} \)
  [Goel,Karande,Tripathi,Wang ’09]
- \( \Omega(n)\)-hardness for \( \min\{f(T) : T \text{ is a spanning tree}\} \)
  [Goel,Karande,Tripathi,Wang ’09]

Good news:
sometimes \( \min\{f(S) : S \in \mathcal{F}\} \) is equally hard for linear/submodular \( f \):
- Variants of Facility Location [Svitkina,Tardos ’06], [Chudak,Nagano ’07]
- 2-approximation for \( \min\{f(S) : S \text{ is a vertex cover}\} \)
  [Koufagiannis,Young; Iwata,Nagano; GKTW ’09]
- 2-approximation for Submodular Multiway Partition (generalizing Node-weighted Multiway Cut) [Chekuri,Ene ’11]
Bad news:
\[ \min \{ f(S) : S \in \mathcal{F} \} \] becomes very hard for some simple constraints:
- \( n^{1/2} \)-hardness for \( \min \{ f(S) : |S| \geq k \} \)
  [Goemans, Harvey, Iwata, Mirrokni ’09], [Svitkina, Fleischer ’09]
- \( n^{2/3} \)-hardness for \( \min \{ f(P) : P \) is a shortest path} \)
  [Goel, Karande, Tripathi, Wang ’09]
- \( \Omega(n) \)-hardness for \( \min \{ f(T) : T \) is a spanning tree} \)
  [Goel, Karande, Tripathi, Wang ’09]

Good news:
sometimes \( \min \{ f(S) : S \in \mathcal{F} \} \) is equally hard for linear/submodular \( f \):
- Variants of Facility Location
  [Svitkina, Tardos ’06], [Chudak, Nagano ’07]
- 2-approximation for \( \min \{ f(S) : S \) is a vertex cover} \)
  [Koufagiannis, Young; Iwata, Nagano; GKTW ’09]
- 2-approximation for Submodular Multiway Partition
  (generalizing Node-weighted Multiway Cut) [Chekuri, Ene ’11]
Submodular Vertex Cover: \( \min \{ f(S) : S \subseteq V \text{ hits every edge in } G \} \)
- formulation using the Lovász extension:

\[
\min f^L(x) : \\
\forall (i, j) \in E; x_i + x_j \geq 1; \\
x \geq 0.
\]
Submodular Vertex Cover

Submodular Vertex Cover: \( \min \{ f(S) : S \subseteq V \text{ hits every edge in } G \} \)
- formulation using the Lovász extension:

\[
\begin{align*}
\min f^L(x) : \\
\forall (i, j) \in E; x_i + x_j &\geq 1; \\
x &\geq 0.
\end{align*}
\]

Algorithm:
- Solve the convex optimization problem.
Submodular Vertex Cover

Submodular Vertex Cover: \( \min \{ f(S) : S \subseteq V \text{ hits every edge in } G \} \)
- formulation using the Lovász extension:

\[
\min f^L(x) : \\
\forall (i, j) \in E; x_i + x_j \geq 1; \\
x \geq 0.
\]

Algorithm:
- Solve the convex optimization problem.
- Given a fractional solution \( x \), take \( \lambda \in [0, \frac{1}{2}) \) uniformly random and

\[
S = T_{\lambda}(x) = \{ i : x_i > \lambda \}.
\]

\( S \) is a vertex cover because each edge has a variable \( x_i \geq 1/2 \).
Submodular Vertex Cover

**Submodular Vertex Cover:**\[ \min \{ f(S) : S \subseteq V \text{ hits every edge in } G \} \]
- formulation using the Lovász extension:

\[
\begin{align*}
\min f^L(x) : \\
\forall (i, j) \in E; x_i + x_j \geq 1; \\
x \geq 0.
\end{align*}
\]

**Algorithm:**
- Solve the convex optimization problem.
- Given a fractional solution \( x \), take \( \lambda \in [0, \frac{1}{2}) \) uniformly random and
  \[
  S = T_\lambda(x) = \{ i : x_i > \lambda \}.
  \]
- \( S \) is a vertex cover because each edge has a variable \( x_i \geq 1/2 \).
- Expected cost of the solution is
  \[
  \mathbb{E}[f(S)] = 2 \int_0^{1/2} f(T_\lambda(x))d\lambda \leq 2 \int_0^1 f(T_\lambda(x))d\lambda = 2f^L(x).
  \]
Submodular Multiway Partition

Submodular Multiway Partition: \( \min \sum_{i=1}^{k} f(S_i) \) where \( (S_1, \ldots, S_k) \) is a partition of \( V \), and \( i \in S_i \) for \( i \in \{1, 2, \ldots, k\} \) (\( k \) terminals).

\[
\min \sum_{i=1}^{k} f^L(x_i) : \\
\forall j \in V; \sum_{i=1}^{k} x_{ij} = 1; \\
\forall i \in [k]; x_{ii} = 1; \\
x \geq 0.
\]
Submodular Multiway Partition

**Submodular Multiway Partition:** $\min \sum_{i=1}^{k} f(S_i)$ where $(S_1, \ldots, S_k)$ is a partition of $V$, and $i \in S_i$ for $i \in \{1, 2, \ldots, k\}$ ($k$ terminals).

$$\min \sum_{i=1}^{k} f^L(x_i) :$$

$\forall j \in V; \sum_{i=1}^{k} x_{ij} = 1;$

$\forall i \in [k]; x_{ii} = 1;$

$x \geq 0.$

$(2 - 2/k)$-approximation algorithm:

- Given a fractional solution $x$, let $A_i = T_\lambda(x_i)$, where $\lambda \in [\frac{1}{2}, 1]$ is uniformly random. Let $U = V \setminus \bigcup_{i=1}^{k} A_i$ be the unallocated vertices.
- Return $S_{i'} = A_{i'} \cup U$ for a random $i'$, and $S_i = A_i$ for $i \neq i'$. 
<table>
<thead>
<tr>
<th>Constraint</th>
<th>Approximation</th>
<th>Hardness</th>
<th>alg. technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unconstrained</td>
<td>1</td>
<td>1</td>
<td>combinatorial</td>
</tr>
<tr>
<td>Parity families</td>
<td>1</td>
<td>1</td>
<td>combinatorial</td>
</tr>
<tr>
<td>Vertex cover</td>
<td>2</td>
<td>2</td>
<td>Lovász ext.</td>
</tr>
<tr>
<td>$k$-unif. hitting set</td>
<td>$k$</td>
<td>$k$</td>
<td>Lovász ext.</td>
</tr>
<tr>
<td>Multiway $k$-partition</td>
<td>$2 - 2/k$</td>
<td>$2 - 2/k$</td>
<td>Lovász ext.</td>
</tr>
<tr>
<td>Facility location</td>
<td>$\log n$</td>
<td>$\log n$</td>
<td>combinatorial</td>
</tr>
<tr>
<td>Set cover</td>
<td>$n$</td>
<td>$n / \log^2 n$</td>
<td>trivial</td>
</tr>
<tr>
<td>$</td>
<td>S</td>
<td>\geq k$</td>
<td>$\tilde{O}(\sqrt{n})$</td>
</tr>
<tr>
<td>Shortest path</td>
<td>$O(n^{2/3})$</td>
<td>$\Omega(n^{2/3})$</td>
<td>combinatorial</td>
</tr>
<tr>
<td>Spanning tree</td>
<td>$O(n)$</td>
<td>$\Omega(n)$</td>
<td>combinatorial</td>
</tr>
</tbody>
</table>
Lecture I: outline

1. Submodular functions: what and why?
2. Convex aspects: Submodular minimization
3. Concave aspects: Submodular maximization
Maximization of submodular functions:

- comes up naturally in allocation / welfare maximization settings
- \( f(S) = \text{value of a set of items } S \) ... often submodular due to combinatorial structure or property of diminishing returns
- in these settings, \( f(S) \) is often assumed to be monotone:
  \[
  S \subset T \implies f(S) \leq f(T).
  \]
Maximization of submodular functions:

- comes up naturally in allocation / welfare maximization settings
- \( f(S) = \text{value of a set of items } S \ldots \text{often submodular due to combinatorial structure or property of diminishing returns} \)
- in these settings, \( f(S) \) is often assumed to be monotone:
  \[
  S \subset T \implies f(S) \leq f(T).
  \]

Hence, we distinguish:

1. **Monotone submodular maximization:**
   e.g. \( \max \{ f(S) : |S| \leq k \} \), generalizing Max \( k \)-cover.

2. **Non-monotone submodular maximization:**
   e.g. \( \max f(S) \), generalizing Max Cut.
Theorem (Nemhauser,Wolsey,Fisher ’78)

The greedy algorithm gives a $(1 - 1/e)$-approximation for the problem
\[
\max \{ f(S) : |S| \leq k \} \text{ where } f \text{ is monotone submodular.}
\]
Theorem (Nemhauser, Wolsey, Fisher ’78)

The greedy algorithm gives a $(1 - 1/e)$-approximation for the problem
\[
\max\{ f(S) : |S| \leq k \} \quad \text{where } f \text{ is monotone submodular.}
\]

Generalizes a greedy $(1 - 1/e)$-approximation for Max $k$-cover:

Max $k$-cover

Choose $k$ sets
so as to maximize
\[
| \bigcup_{j \in K} A_j |.
\]

[Feige ’98]:

Unless $P = NP$, there is no $(1 - \frac{1}{e} + \epsilon)$-approximation for Max $k$-cover.
Analysis of Greedy

Greedy Algorithm: $S_i = solution after i steps;$

pick next element $a$ to maximize $f(S_i + a) - f(S_i)$. 

\[
\begin{align*}
\text{Let the optimal solution be } S^*.
\text{By submodularity: } & \exists a \in S^* \setminus S_i; \\
& f(S_i + a) - f(S_i) \geq \frac{1}{k} (OPT - f(S_i)) \\
\Rightarrow & \text{OPT} - f(S_k) \leq \frac{1}{k} OPT \leq \frac{1}{e} OPT.
\end{align*}
\]
Analysis of Greedy

**Greedy Algorithm:** \( S_i = \text{solution after } i \text{ steps}; \)

\( \text{pick next element } a \text{ to maximize } f(S_i + a) - f(S_i). \)

Let the optimal solution be \( S^*. \) By submodularity:

\[
\exists a \in S^* \setminus S_i; f(S_i + a) - f(S_i) \geq \frac{1}{k}(OPT - f(S_i)).
\]

\[
1 - \frac{1}{e}
\]

\[
\frac{f(S_i)}{OPT}
\]

\[
0 \quad 1 \quad 2 \quad 3 \quad \cdots \quad k
\]

\[
OPT - f(S_{i+1}) \leq (1 - \frac{1}{k})(OPT - f(S_i))
\]

\[
\Rightarrow OPT - f(S_k) \leq (1 - \frac{1}{k})^k OPT \leq \frac{1}{e} OPT.
\]
Submodular maximization under a matroid constraint

Nemhauser, Wolsey and Fisher considered a more general problem:

**Given:** Monotone submodular function $f$, matroid $\mathcal{M} = (N, \mathcal{I})$.

**Goal:** Find $S \in \mathcal{I}$ maximizing $f(S)$.
Submodular maximization under a matroid constraint

Nemhauser, Wolsey and Fisher considered a more general problem:

**Given:** Monotone submodular function \( f \), matroid \( \mathcal{M} = (\mathcal{N}, \mathcal{I}) \).
**Goal:** Find \( S \in \mathcal{I} \) maximizing \( f(S) \).

**Theorem (Nemhauser, Wolsey, Fisher ’78)**

*The greedy algorithm gives a \( \frac{1}{2} \)-approximation for the problem max\{\( f(S) : S \in \mathcal{I} \)\}.*

**More generally:** \( \frac{1}{k+1} \)-approximation for the problem max\{\( f(S) : S \in \mathcal{I}_1 \cap \mathcal{I}_2 \cap \ldots \cap \mathcal{I}_k \)\}.

**Motivation:** what are matroids and what can be modeled using a matroid constraint?
Matriods

Definition

A matroid on $N$ is a system of independent sets $\mathcal{I} \subset 2^N$, satisfying

1. $\forall B \in \mathcal{I}, A \subset B \Rightarrow A \in \mathcal{I}$.
2. $\forall A, B \in \mathcal{I}, |A| < |B| \Rightarrow \exists x \in B \setminus A; A \cup \{x\} \in \mathcal{I}$.

Example: partition matroid $S$ is independent, if $|S \cap Q_i| \leq 1$ for each $Q_i$. 

Jan Vondrák (IBM Almaden) Submodular Optimization Tutorial 23 / 1
Matroids

Definition

A matroid on $N$ is a system of independent sets $\mathcal{I} \subset 2^N$, satisfying

1. $\forall B \in \mathcal{I}, A \subset B \Rightarrow A \in \mathcal{I}$.
2. $\forall A, B \in \mathcal{I}, |A| < |B| \Rightarrow \exists x \in B \setminus A; A \cup \{x\} \in \mathcal{I}$.

Example: partition matroid

$S$ is independent, if $|S \cap Q_i| \leq 1$ for each $Q_i$. 
Submodular Welfare Maximization:

Given $n$ players with submodular valuation functions $w_i : 2^M \rightarrow \mathbb{R}_+$. Partition $M = S_1 \cup \ldots \cup S_n$ so as to maximize $\sum_{i=1}^{n} w_i(S_i)$. 

Reduction:
- Create $n$ clones of each item, $f(S) = \sum_i w_i(S \cap M_i)$, $I = \{ S : \forall i; |S \cap Q_i| \leq 1 \}$ (a partition matroid).

Submodular Welfare Maximization is equivalent to $\max\{f(S) : S \in I\}$ → Greedy gives $\frac{1}{2}$-approximation.
Submodular Welfare Maximization:

Given $n$ players with submodular valuation functions $w_i : 2^M \rightarrow \mathbb{R}_+$. Partition $M = S_1 \cup \ldots \cup S_n$ so as to maximize $\sum_{i=1}^n w_i(S_i)$.

Reduction:

Create $n$ clones of each item, $f(S) = \sum_i w_i(S \cap M_i)$, $\mathcal{I} = \{ S : \forall i; |S \cap Q_i| \leq 1 \}$ (a partition matroid).

Submodular Welfare Maximization is equivalent to max$\{f(S) : S \in \mathcal{I}\}$ ⇒ Greedy gives $\frac{1}{2}$-approximation.
Further combinatorial techniques

**Partial enumeration:** "guess" the first $t$ elements, then run greedy.

- $(1 - 1/e)$-approximation for monotone submodular maximization subject to a knapsack constraint, $\sum_{j \in S} w_j \leq B$ [Sviridenko ’04]
Further combinatorial techniques

Partial enumeration: "guess" the first $t$ elements, then run greedy.

- $(1 - 1/e)$-approximation for monotone submodular maximization subject to a knapsack constraint, $\sum_{j \in S} w_j \leq B$ [Sviridenko ’04]

Local search: switch up to $t$ elements, as long as it provides a (non-trivial) improvement; possibly iterate in several phases.

- $1/3$-approximation for unconstrained (non-monotone) maximization [Feige,Mirrokni,V. ’07]
- $1/(k + 2 + \frac{1}{k} + \delta_t)$-approximation for non-monotone maximization subject to $k$ matroids [Lee,Mirrokni,Nagarajan,Sviridenko ’09]
- $1/(k + \delta_t)$-approximation for monotone submodular maximization subject to $k \geq 2$ matroids [Lee,Sviridenko,V. ’10]
Continuous relaxation for submodular maximization?

Questions that don’t seem to be answered by combinatorial algorithms:

- What is the optimal approximation for \( \max \{ f(S) : S \in I \} \), in particular the Submodular Welfare Problem?
- What is the optimal approximation for multiple constraints, e.g. multiple knapsack constraints?
- In general, how can we combine different types of constraints?
Continuous relaxation for submodular maximization?

**Questions that don’t seem to be answered by combinatorial algorithms:**

- What is the optimal approximation for \( \max\{f(S) : S \in \mathcal{I}\} \), in particular the *Submodular Welfare Problem*?
- What is the optimal approximation for *multiple constraints*, e.g. multiple knapsack constraints?
- In general, how can we combine *different types of constraints*?

It would be nice to have a *continuous relaxation*, but:

1. The *Lovász extension* is convex, therefore not suitable for maximization.
2. The counterpart of the convex closure is the *concave closure*

\[
f^+(x) = \max\{\sum \alpha_S f(S) : \sum \alpha_S 1_S = x, \sum \alpha_S = 1, \alpha_S \geq 0\}.
\]

However, this extension is NP-hard to evaluate!
Multilinear relaxation

Discrete Problem
\[
\max\{f(S) : S \in \mathcal{F}\}
\]

(Multilinear) Continuous Problem
\[
\max\{F(x) : x \in P(\mathcal{F})\}
\]

Multilinear extension of \(f\):

\[
F(x) = E[f(\hat{x})],
\]

where \(\hat{x}\) is obtained by rounding each \(x_i\) randomly to 0/1 with probabilities \(x_i\).

\(F(x)\) is neither convex nor concave; it is multilinear and \(\partial^2 F / \partial x_i^2 = 0\).

\(F(x + \lambda \vec{d})\) is a concave function of \(\lambda\), if \(\vec{d} \geq 0\).
Multilinear relaxation

Discrete Problem
\[
\max\{ f(S) : S \in \mathcal{F} \}
\]

(Multilinear) Continuous Problem
\[
\max\{ F(x) : x \in P(\mathcal{F}) \}
\]

Multilinear extension of \( f \):
- \[ F(x) = \mathbb{E}[f(\hat{x})], \] where \( \hat{x} \) is obtained by rounding each \( x_i \) randomly to 0/1 with probabilities \( x_i \).
- \( F(x) \) is neither convex nor concave; it is multilinear and \( \frac{\partial^2 F}{\partial x_i^2} = 0 \).
- \( F(x + \lambda \vec{d}) \) is a concave function of \( \lambda \), if \( \vec{d} \geq 0 \).
The multilinear relaxation turns out to be useful for maximization:

1. **The continuous problem** \( \max \{ F(x) : x \in P \} \) can be solved:
   - \((1 - 1/e)\)-approximately for any monotone submodular function and solvable polytope [V. ’08]
   - \((1/e)\)-approximately for any nonnegative submodular function and downward-closed solvable polytope [Feldman, Naor, Schwartz ’11]
   (earlier constant factors: 0.325 [Chekuri, V., Zenklusen ’11], 0.13 [Fadaei, Fazli, Safari ’11])

2. **A fractional solution can be rounded:**
   - without loss for a matroid constraint [Calinescu, Chekuri, Pál, V. ’07]
   - losing \((1 - \epsilon)\) factor for a constant number of knapsack constraints [Kulik, Shachnai, Tamir ’10]
   - losing \(O(k)\) factor for \(k\) matroid constraints, in a modular fashion (to be combined with other constraints) [Chekuri, V., Zenklusen ’11]
   - e.g., \(O(k)\)-approximation for \(k\) matroids & \(O(1)\) knapsacks
The Continuous Greedy Algorithm

**Problem:** \( \max \{ F(x) : x \in P \} \).
Problem: \( \max \{ F(x) : x \in P \} \).

For each \( x \in P \), define \( v(x) \) by

\[
v(x) = \arg\max_{v \in P} (v \cdot \nabla F|_x).
\]
The Continuous Greedy Algorithm

Problem: \( \max \{ F(x) : x \in P \} \).

For each \( x \in P \), define \( v(x) \) by
\[
v(x) = \arg\max_{v \in P} (v \cdot \nabla F|_x).
\]
Define a curve \( x(t) \):
\[
\frac{dx}{dt} = v(x)
\]
Run this process for \( t \in [0, 1] \) and return \( x(1) \).
The Continuous Greedy Algorithm

**Problem:** \( \max \{ F(x) : x \in P \} \).

For each \( x \in P \), define \( v(x) \) by

\[
    v(x) = \arg\max_{v \in P} (v \cdot \nabla F|_x)。
\]

Define a curve \( x(t) \):

\[
    x(0) = 0 \\
    \frac{dx}{dt} = v(x)
\]

Run this process for \( t \in [0, 1] \) and return \( x(1) \).

**Claim:** \( x(1) \in P \) and \( F(x(1)) \geq (1 - 1/e) \text{OPT} \).
Analysis of Continuous Greedy

Evolution of the fractional solution:
- Differential equation: \( x(0) = 0, \frac{dx}{dt} = v(x) \).
Analysis of Continuous Greedy

Evolution of the fractional solution:

- Differential equation: \( x(0) = 0, \frac{dx}{dt} = v(x) \).
- Chain rule:
  \[
  \frac{dF}{dt} = \frac{dx}{dt} \cdot \nabla F(x(t)) = v(x) \cdot \nabla F(x(t)) \geq OPT - F(x(t)).
  \]
Analysis of Continuous Greedy

**Evolution of the fractional solution:**
- Differential equation: \( x(0) = 0, \frac{dx}{dt} = v(x) \).
- Chain rule:
  \[
  \frac{dF}{dt} = \frac{dx}{dt} \cdot \nabla F(x(t)) = v(x) \cdot \nabla F(x(t)) \geq OPT - F(x(t)).
  \]

Solve the differential equation:
\[
F(x(t)) \geq (1 - e^{-t}) \cdot OPT.
\]
# Submodular maximization overview

## MONOTONE MAXIMIZATION

<table>
<thead>
<tr>
<th>Constraint</th>
<th>Approximation</th>
<th>Hardness</th>
<th>technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>(</td>
<td>S</td>
<td>\leq k)</td>
<td>1 − 1/e</td>
</tr>
<tr>
<td>matroid</td>
<td>1 − 1/e</td>
<td>1 − 1/e</td>
<td>multilinear ext.</td>
</tr>
<tr>
<td>(O(1)) knapsacks</td>
<td>1 − 1/e</td>
<td>1 − 1/e</td>
<td>multilinear ext.</td>
</tr>
<tr>
<td>(k) matroids</td>
<td>(k + \epsilon)</td>
<td>(k / \log k)</td>
<td>local search</td>
</tr>
<tr>
<td>(k) matroids &amp; (O(1)) knapsacks</td>
<td>(O(k))</td>
<td>(k / \log k)</td>
<td>multilinear ext.</td>
</tr>
</tbody>
</table>

## NON-MONOTONE MAXIMIZATION

<table>
<thead>
<tr>
<th>Constraint</th>
<th>Approximation</th>
<th>Hardness</th>
<th>technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unconstrained</td>
<td>1/2</td>
<td>1/2</td>
<td>combinatorial</td>
</tr>
<tr>
<td>matroid</td>
<td>1/e</td>
<td>0.48</td>
<td>multilinear ext.</td>
</tr>
<tr>
<td>(O(1)) knapsacks</td>
<td>1/e</td>
<td>0.49</td>
<td>multilinear ext.</td>
</tr>
<tr>
<td>(k) matroids</td>
<td>(k + O(1))</td>
<td>(k / \log k)</td>
<td>local search</td>
</tr>
<tr>
<td>(k) matroids &amp; (O(1)) knapsacks</td>
<td>(O(k))</td>
<td>(k / \log k)</td>
<td>multilinear ext.</td>
</tr>
</tbody>
</table>