

CS675: Convex and Combinatorial Optimization
Fall 2016
Submodular Function Optimization

Instructor: Shaddin Dughmi

- 1 Introduction to Submodular Functions
- 2 Unconstrained Submodular Minimization
 - Definition and Examples
 - The Convex Closure and the Lovasz Extension
 - Wrapping up
- 3 Monotone Submodular Maximization s.t. a Matroid Constraint
 - Definition and Examples
 - Warmup: Cardinality Constraint
 - General Matroid Constraints

- We saw how matroids form a class of feasible sets over which optimization of modular objectives is tractable
- If matroids are discrete analogues of convex sets, then submodular functions are discrete analogues of convex/concave functions
 - Submodular functions behave like convex functions sometimes (minimization) and concave other times (maximization)
- Today we will introduce submodular functions, go through some examples, and mention some of their properties

Set Functions

- A **set function** takes as input a set, and outputs a real number
 - Inputs are subsets of some **ground set** X
 - $f : 2^X \rightarrow \mathbb{R}$
- We will focus on set functions where X is finite, and denote $n = |X|$

Set Functions

- A **set function** takes as input a set, and outputs a real number
 - Inputs are subsets of some **ground set** X
 - $f : 2^X \rightarrow \mathbb{R}$
- We will focus on set functions where X is finite, and denote $n = |X|$
- Equivalently: map points in the hypercube $\{0, 1\}^n$ to the real numbers
 - Can be plotted as 2^n points in $n + 1$ dimensional space

Set Functions

- We have already seen **modular** set functions
 - Associate a weight w_i with each $i \in X$, and set $f(S) = \sum_{i \in S} w_i$
 - Discrete analogue of linear functions

Set Functions

- We have already seen **modular** set functions
 - Associate a weight w_i with each $i \in X$, and set $f(S) = \sum_{i \in S} w_i$
 - Discrete analogue of linear functions
 - Direct definition of modularity: $f(A) + f(B) = f(A \cap B) + f(A \cup B)$

Set Functions

- We have already seen **modular** set functions
 - Associate a weight w_i with each $i \in X$, and set $f(S) = \sum_{i \in S} w_i$
 - Discrete analogue of linear functions
 - Direct definition of modularity: $f(A) + f(B) = f(A \cap B) + f(A \cup B)$
- **Supmodular/supermodular** functions are weak analogues to convex/concave functions (in no particular order!)

Set Functions

- We have already seen **modular** set functions
 - Associate a weight w_i with each $i \in X$, and set $f(S) = \sum_{i \in S} w_i$
 - Discrete analogue of linear functions
 - Direct definition of modularity: $f(A) + f(B) = f(A \cap B) + f(A \cup B)$
- **Supmodular/supermodular** functions are weak analogues to convex/concave functions (in no particular order!)
- Other possibly useful properties a set function may have:
 - **Monotone** increasing or decreasing
 - **Nonnegative**: $f(A) \geq 0$ for all $S \subseteq X$
 - **Normalized**: $f(\emptyset) = 0$.

Submodular Functions

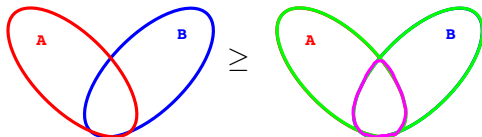
Definition 1

A set function $f : 2^X \rightarrow \mathbb{R}$ is **submodular** if and only if

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

for all $A, B \subseteq X$.

- “Uncrossing” two sets reduces their total function value



Submodular Functions

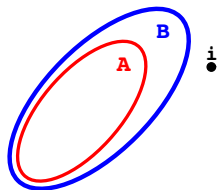
Definition 2

A set function $f : 2^X \rightarrow \mathbb{R}$ is **submodular** if and only if

$$f(B \cup \{i\}) - f(B) \leq f(A \cup \{i\}) - f(A)$$

for all $A \subseteq B \subseteq X$ and $i \notin B$.

- The marginal value of an additional element exhibits “diminishing marginal returns”
- Should remind of concavity: second “derivative” is negative



Supermodular Functions

Definition 0

A set function $f : 2^X \rightarrow \mathbb{R}$ is **supermodular** if and only if $-f$ is submodular.

Supermodular Functions

Definition 0

A set function $f : 2^X \rightarrow \mathbb{R}$ is **supermodular** if and only if $-f$ is submodular.

Definition 1

A set function $f : 2^X \rightarrow \mathbb{R}$ is **supermodular** if and only if

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

for all $A, B \subseteq X$.

Supermodular Functions

Definition 0

A set function $f : 2^X \rightarrow \mathbb{R}$ is **supermodular** if and only if $-f$ is submodular.

Definition 1

A set function $f : 2^X \rightarrow \mathbb{R}$ is **supermodular** if and only if

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

for all $A, B \subseteq X$.

Definition 2

A set function $f : 2^X \rightarrow \mathbb{R}$ is **supermodular** if and only if

$$f(B \cup \{i\}) - f(B) \geq f(A \cup \{i\}) - f(A)$$

for all $A \subseteq B \subseteq X$ and $i \notin B$.

Many common examples are monotone, normalized, and submodular. We mention some.

Coverage Functions

X is the left hand side of a graph, and $f(S)$ is the total number of neighbors of S .

- Can think of $i \in X$ as a set, and $f(S)$ as the total “coverage” of S .

Examples

Many common examples are monotone, normalized, and submodular. We mention some.

Coverage Functions

X is the left hand side of a graph, and $f(S)$ is the total number of neighbors of S .

- Can think of $i \in X$ as a set, and $f(S)$ as the total “coverage” of S .

Probability

X is a set of probability events, and $f(S)$ is the probability at least one of them occurs.

Social Influence

- X is the family of nodes in a social network
- A meme, idea, or product is adopted at a set of nodes S
- The idea propagates through the network through some random diffusion process
 - Many different models
- $f(S)$ is the expected number of nodes in the network which end up adopting the idea.

Social Influence

- X is the family of nodes in a social network
- A meme, idea, or product is adopted at a set of nodes S
- The idea propagates through the network through some random diffusion process
 - Many different models
- $f(S)$ is the expected number of nodes in the network which end up adopting the idea.

Utility Functions

When X is a set of goods, $f(S)$ can represent the utility of an agent for a bundle of these goods. Utilities which exhibit diminishing marginal returns are natural in many settings.

Entropy

X is a set of random variables, and $f(S)$ is the entropy of the joint distribution of a subset of them S .

Examples

Entropy

X is a set of random variables, and $f(S)$ is the entropy of the joint distribution of a subset of them S .

Matroid Rank

The rank function of a matroid is monotone, submodular, and normalized.

Examples

Entropy

X is a set of random variables, and $f(S)$ is the entropy of the joint distribution of a subset of them S .

Matroid Rank

The rank function of a matroid is monotone, submodular, and normalized.

Clustering Quality

X is the set of nodes in a graph G , and $f(S) = E(S)$ is the internal connectedness of cluster S .

- Supermodular

Examples

There are fewer examples of non-monotone submodular/supermodular functions, which are nonetheless fundamental.

Graph Cuts

X is the set of nodes in a graph G , and $f(S)$ is the number of edges crossing the cut $(S, X \setminus S)$.

- Submodular
- Non-monotone.

Examples

There are fewer examples of non-monotone submodular/supermodular functions, which are nonetheless fundamental.

Graph Cuts

X is the set of nodes in a graph G , and $f(S)$ is the number of edges crossing the cut $(S, X \setminus S)$.

- Submodular
- Non-monotone.

Graph Density

X is the set of nodes in a graph G , and $f(S) = \frac{E(S)}{|S|}$ where $E(S)$ is the number of edges with both endpoints in S .

- Non-monotone
- Neither submodular nor supermodular

Examples

There are fewer examples of non-monotone submodular/supermodular functions, which are nonetheless fundamental.

Graph Cuts

X is the set of nodes in a graph G , and $f(S)$ is the number of edges crossing the cut $(S, X \setminus S)$.

- Submodular
- Non-monotone.

Graph Density

X is the set of nodes in a graph G , and $f(S) = \frac{E(S)}{|S|}$ where $E(S)$ is the number of edges with both endpoints in S .

- Non-monotone
- Neither submodular nor supermodular
- However, maximizing it reduces to maximizing supermodular function $E(S) - \alpha|S|$ for various $\alpha > 0$ (binary search)

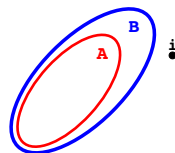
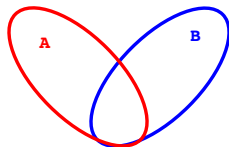
Equivalence of Both Definitions

Definition 1

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

Definition 2

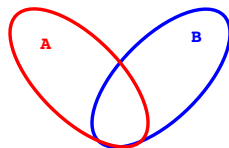
$$f(B \cup \{i\}) - f(B) \leq f(A \cup \{i\}) - f(A)$$



Equivalence of Both Definitions

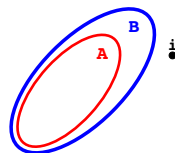
Definition 1

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



Definition 2

$$f(B \cup \{i\}) - f(B) \leq f(A \cup \{i\}) - f(A)$$



Definition 1 \Rightarrow Definition 2

- To prove (2), let $A' = A \cup \{i\}$ and $B' = B$ and apply (1)

$$\begin{aligned} f(A \cup \{i\}) + f(B) &= f(A') + f(B') \\ &\geq f(A' \cap B') + f(A' \cup B') \\ &= f(A) + f(B \cup \{i\}) \end{aligned}$$

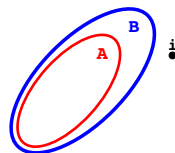
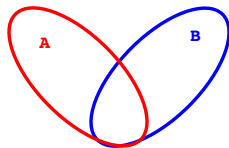
Equivalence of Both Definitions

Definition 1

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

Definition 2

$$f(B \cup \{i\}) - f(B) \leq f(A \cup \{i\}) - f(A)$$



Definition 2 \Rightarrow Definition 1

- To prove (1), start with $A = B$ and repeatedly elements to one but not the other
- At each step, (2) implies that the LHS of inequality (1) increases more than the RHS

Operations Preserving Submodularity

- **Nonnegative-weighted combinations** (a.k.a. conic combinations):
If f_1, \dots, f_k are submodular, and $w_1, \dots, w_k \geq 0$, then $g(S) = \sum_i w_i f_i(S)$ is also submodular
 - Special case: adding or subtracting a modular function

Operations Preserving Submodularity

- **Nonnegative-weighted combinations** (a.k.a. conic combinations):
If f_1, \dots, f_k are submodular, and $w_1, \dots, w_k \geq 0$, then
 $g(S) = \sum_i w_i f_i(S)$ is also submodular
 - Special case: adding or subtracting a modular function
- **Restriction**: If f is a submodular function on X , and $T \subseteq X$, then
 $g(S) = f(S \cap T)$ is submodular

Operations Preserving Submodularity

- **Nonnegative-weighted combinations** (a.k.a. conic combinations):
If f_1, \dots, f_k are submodular, and $w_1, \dots, w_k \geq 0$, then $g(S) = \sum_i w_i f_i(S)$ is also submodular
 - Special case: adding or subtracting a modular function
- **Restriction**: If f is a submodular function on X , and $T \subseteq X$, then $g(S) = f(S \cap T)$ is submodular
- **Contraction** (a.k.a. **conditioning**): If f is a submodular function on X , and $T \subseteq X$, then $f_T(S) = f(S \cup T)$ is submodular

Operations Preserving Submodularity

- **Nonnegative-weighted combinations** (a.k.a. conic combinations):
If f_1, \dots, f_k are submodular, and $w_1, \dots, w_k \geq 0$, then $g(S) = \sum_i w_i f_i(S)$ is also submodular
 - Special case: adding or subtracting a modular function
- **Restriction**: If f is a submodular function on X , and $T \subseteq X$, then $g(S) = f(S \cap T)$ is submodular
- **Contraction** (a.k.a. **conditioning**): If f is a submodular function on X , and $T \subseteq X$, then $f_T(S) = f(S \cup T)$ is submodular
- **Reflection**: If f is a submodular function on X , then $\bar{f}(S) = f(X \setminus S)$ is also submodular

Operations Preserving Submodularity

- **Nonnegative-weighted combinations** (a.k.a. conic combinations):
If f_1, \dots, f_k are submodular, and $w_1, \dots, w_k \geq 0$, then $g(S) = \sum_i w_i f_i(S)$ is also submodular
 - Special case: adding or subtracting a modular function
- **Restriction**: If f is a submodular function on X , and $T \subseteq X$, then $g(S) = f(S \cap T)$ is submodular
- **Contraction** (a.k.a. **conditioning**): If f is a submodular function on X , and $T \subseteq X$, then $f_T(S) = f(S \cup T)$ is submodular
- **Reflection**: If f is a submodular function on X , then $\bar{f}(S) = f(X \setminus S)$ is also submodular
- Others: Dilworth truncation, convolution with modular functions, ...

Operations Preserving Submodularity

- **Nonnegative-weighted combinations** (a.k.a. conic combinations):
If f_1, \dots, f_k are submodular, and $w_1, \dots, w_k \geq 0$, then $g(S) = \sum_i w_i f_i(S)$ is also submodular
 - Special case: adding or subtracting a modular function
- **Restriction**: If f is a submodular function on X , and $T \subseteq X$, then $g(S) = f(S \cap T)$ is submodular
- **Contraction** (a.k.a. **conditioning**): If f is a submodular function on X , and $T \subseteq X$, then $f_T(S) = f(S \cup T)$ is submodular
- **Reflection**: If f is a submodular function on X , then $\bar{f}(S) = f(X \setminus S)$ is also submodular
- Others: Dilworth truncation, convolution with modular functions, ...

Note

The minimum or maximum of two submodular functions is not necessarily submodular

Optimizing Submodular Functions

- As our examples suggest, optimization problems involving submodular functions are very common
- These can be classified on two axes: constrained/unconstrained and maximization/minimization

	Maximization	Minimization
Unconstrained	NP-hard $\frac{1}{2}$ approximation	Polynomial time via convex opt
Constrained	Usually NP-hard $1 - 1/e$ (mono, matroid) $O(1)$ ("nice" constraints)	Usually NP-hard to apx. Few easy special cases

Optimizing Submodular Functions

- As our examples suggest, optimization problems involving submodular functions are very common
- These can be classified on two axes: constrained/unconstrained and maximization/minimization

	Maximization	Minimization
Unconstrained	NP-hard $\frac{1}{2}$ approximation	Polynomial time via convex opt
Constrained	Usually NP-hard $1 - 1/e$ (mono, matroid) $O(1)$ ("nice" constraints)	Usually NP-hard to apx. Few easy special cases

Optimizing Submodular Functions

- As our examples suggest, optimization problems involving submodular functions are very common
- These can be classified on two axes: constrained/unconstrained and maximization/minimization

	Maximization	Minimization
Unconstrained	NP-hard $\frac{1}{2}$ approximation	Polynomial time via convex opt
Constrained	Usually NP-hard $1 - 1/e$ (mono, matroid) $O(1)$ ("nice" constraints)	Usually NP-hard to apx. Few easy special cases

Representation

In order to generalize all our examples, algorithmic results are often posed in the **value oracle** model. Namely, we only assume we have access to a subroutine evaluating $f(S)$.

- 1 Introduction to Submodular Functions
- 2 Unconstrained Submodular Minimization
 - Definition and Examples
 - The Convex Closure and the Lovasz Extension
 - Wrapping up
- 3 Monotone Submodular Maximization s.t. a Matroid Constraint
 - Definition and Examples
 - Warmup: Cardinality Constraint
 - General Matroid Constraints

Recall: Optimizing Submodular Functions

	Maximization	Minimization
Unconstrained	NP-hard $\frac{1}{2}$ approximation	Polynomial time via convex opt
Constrained	Usually NP-hard $1 - 1/e$ (mono, matroid) $O(1)$ (“nice” constraints)	Usually NP-hard to apx. Few easy special cases

Recall: Optimizing Submodular Functions

	Maximization	Minimization
Unconstrained	NP-hard $\frac{1}{2}$ approximation	Polynomial time via convex opt
Constrained	Usually NP-hard $1 - 1/e$ (mono, matroid) $O(1)$ ("nice" constraints)	Usually NP-hard to apx. Few easy special cases

Problem Definition

Given a submodular function $f : 2^X \rightarrow \mathbb{R}$ on a finite ground set X ,

$$\begin{array}{ll} \text{minimize} & f(S) \\ \text{subject to} & S \subseteq X \end{array}$$

- We denote $n = |X|$
- We assume $f(S)$ is a rational number with at most b bits

Problem Definition

Given a submodular function $f : 2^X \rightarrow \mathbb{R}$ on a finite ground set X ,

$$\begin{array}{ll} \text{minimize} & f(S) \\ \text{subject to} & S \subseteq X \end{array}$$

- We denote $n = |X|$
- We assume $f(S)$ is a rational number with at most b bits

Representation

In order to generalize all our examples, algorithmic results are often posed in the **value oracle** model. Namely, we only assume we have access to a subroutine evaluating $f(S)$ in constant time.

Problem Definition

Given a submodular function $f : 2^X \rightarrow \mathbb{R}$ on a finite ground set X ,

$$\begin{array}{ll} \text{minimize} & f(S) \\ \text{subject to} & S \subseteq X \end{array}$$

- We denote $n = |X|$
- We assume $f(S)$ is a rational number with at most b bits

Representation

In order to generalize all our examples, algorithmic results are often posed in the **value oracle** model. Namely, we only assume we have access to a subroutine evaluating $f(S)$ in constant time.

Goal

An algorithm which runs in time polynomial in n and b .

Problem Definition

Given a submodular function $f : 2^X \rightarrow \mathbb{R}$ on a finite ground set X ,

$$\begin{array}{ll} \text{minimize} & f(S) \\ \text{subject to} & S \subseteq X \end{array}$$

- We denote $n = |X|$
- We assume $f(S)$ is a rational number with at most b bits

Representation

In order to generalize all our examples, algorithmic results are often posed in the **value oracle** model. Namely, we only assume we have access to a subroutine evaluating $f(S)$ in constant time.

Goal

An algorithm which runs in time polynomial in n and b .

Note: weakly polynomial. There are strongly polytime algorithms.

Minimum Cut

Given a graph $G = (V, E)$, find a set $S \subseteq V$ minimizing the number of edges crossing the cut $(S, V \setminus S)$.

- G may be directed or undirected.
- Extends to hypergraphs.

Minimum Cut

Given a graph $G = (V, E)$, find a set $S \subseteq V$ minimizing the number of edges crossing the cut $(S, V \setminus S)$.

- G may be directed or undirected.
- Extends to hypergraphs.

Densest Subgraph

Given an undirected graph $G = (V, E)$, find a set $S \subseteq V$ maximizing the average internal degree.

- Reduces to supermodular maximization via binary search for the right density.

Continuous Extensions of a Set Function

Recall

A set function f on $X = \{1, \dots, n\}$ can be thought of as a map from the vertices $\{0, 1\}^n$ of the n -dimensional hypercube to the real numbers.

Continuous Extensions of a Set Function

Recall

A set function f on $X = \{1, \dots, n\}$ can be thought of as a map from the vertices $\{0, 1\}^n$ of the n -dimensional hypercube to the real numbers.

We will consider extensions of a set function to the entire hypercube.

Extension of a Set Function

Given a set function $f : \{0, 1\}^n \rightarrow \mathbb{R}$, an **extension** of f to the hypercube $[0, 1]^n$ is a function $g : [0, 1]^n \rightarrow \mathbb{R}$ satisfying $g(x) = f(x)$ for every $x \in \{0, 1\}^n$.

Continuous Extensions of a Set Function

Recall

A set function f on $X = \{1, \dots, n\}$ can be thought of as a map from the vertices $\{0, 1\}^n$ of the n -dimensional hypercube to the real numbers.

We will consider extensions of a set function to the entire hypercube.

Extension of a Set Function

Given a set function $f : \{0, 1\}^n \rightarrow \mathbb{R}$, an **extension** of f to the hypercube $[0, 1]^n$ is a function $g : [0, 1]^n \rightarrow \mathbb{R}$ satisfying $g(x) = f(x)$ for every $x \in \{0, 1\}^n$.

Long story short...

We will exhibit an extension which is convex when f is submodular, and can be minimized efficiently. We will then show that minimizing it yields a solution to the submodular minimization problem.

The Convex Closure

Convex Closure

Given a set function $f : \{0, 1\}^n \rightarrow \mathbb{R}$, the convex closure $f^- : [0, 1]^n \rightarrow \mathbb{R}$ of f is the point-wise greatest convex function under-estimating f on $\{0, 1\}^n$.

The Convex Closure

Convex Closure

Given a set function $f : \{0, 1\}^n \rightarrow \mathbb{R}$, the convex closure $f^- : [0, 1]^n \rightarrow \mathbb{R}$ of f is the point-wise greatest convex function under-estimating f on $\{0, 1\}^n$.

Geometric Intuition

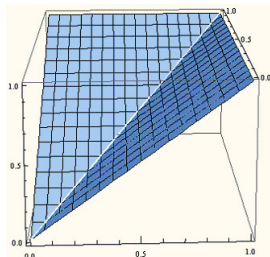
What you would get by placing a blanket under the plot of f and pulling up.

$$f(\emptyset) = 0$$

$$f(\{1\}) = f(\{2\}) = 1$$

$$f(\{1, 2\}) = 1$$

$$f^-(x_1, x_2) = \max(x_1, x_2)$$



The Convex Closure

Convex Closure

Given a set function $f : \{0, 1\}^n \rightarrow \mathbb{R}$, the convex closure $f^- : [0, 1]^n \rightarrow \mathbb{R}$ of f is the point-wise greatest convex function under-estimating f on $\{0, 1\}^n$.

Claim

The convex closure exists for any set function.

Proof

- If $g_1, g_2 : [0, 1]^n \rightarrow \mathbb{R}$ are convex under-estimators of f , then so is $\max\{g_1, g_2\}$
- Holds for infinite set of convex under-estimators
- Therefore $f^- = \max\{g : g \text{ is a convex underestimator of } f\}$ is the point-wise greatest convex underestimator of f .

Claim

The value of the convex closure f^- at $x \in [0, 1]^n$ is the solution of the following optimization problem:

$$\begin{aligned} &\text{minimize} && \sum_{y \in \{0,1\}^n} \lambda_y f(y) \\ &\text{subject to} && \sum_{y \in \{0,1\}^n} \lambda_y y = x \\ &&& \sum_{y \in \{0,1\}^n} \lambda_y = 1 \\ &&& \lambda_y \geq 0, && \text{for } y \in \{0, 1\}^n. \end{aligned}$$

Interpretation

- The minimum expected value of f over all distributions on $\{0, 1\}^n$ with expectation x .
- Equivalently: the minimum expected value of f for a random set $S \subseteq X$ including each $i \in X$ with probability x_i .
- The upper bound on $f^-(x)$ implied by applying Jensen's inequality to every convex combination of $\{0, 1\}^n$.

Claim

The value of the convex closure f^- at $x \in [0, 1]^n$ is the solution of the following optimization problem:

$$\begin{aligned} & \text{minimize} && \sum_{y \in \{0,1\}^n} \lambda_y f(y) \\ & \text{subject to} && \sum_{y \in \{0,1\}^n} \lambda_y y = x \\ & && \sum_{y \in \{0,1\}^n} \lambda_y = 1 \\ & && \lambda_y \geq 0, && \text{for } y \in \{0, 1\}^n. \end{aligned}$$

Implication

- f^- is an extension of f .
- $f^-(x)$ has no “integrality gap”
 - For every $x \in [0, 1]^n$, there is a random integer vector $y \in \{0, 1\}^n$ such that $\mathbf{E}_y f(y) = f^-(x)$.
 - Therefore, there is an integer vector y such that $f(y) \leq f^-(x)$.

Claim

The value of the convex closure f^- at $x \in [0, 1]^n$ is the solution of the following optimization problem:

$$\begin{aligned} & \text{minimize} && \sum_{y \in \{0,1\}^n} \lambda_y f(y) \\ & \text{subject to} && \sum_{y \in \{0,1\}^n} \lambda_y y = x \\ & && \sum_{y \in \{0,1\}^n} \lambda_y = 1 \\ & && \lambda_y \geq 0, \end{aligned} \quad \text{for } y \in \{0, 1\}^n .$$

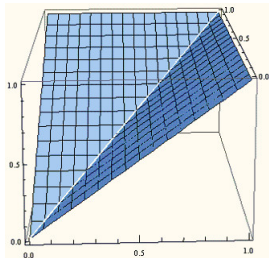
$$f(\emptyset) = 0$$

$$f(\{1\}) = f(\{2\}) = 1$$

$$f(\{1, 2\}) = 1$$

When $x_1 \leq x_2$

$$\begin{aligned} f^-(x_1, x_2) &= x_1 f(\{1, 2\}) \\ &\quad + (x_2 - x_1) f(\{2\}) \\ &\quad + (1 - x_2) f(\emptyset) \end{aligned}$$



Claim

The value of the convex closure f^- at $x \in [0, 1]^n$ is the solution of the following optimization problem:

$$\begin{aligned} & \text{minimize} && \sum_{y \in \{0,1\}^n} \lambda_y f(y) \\ & \text{subject to} && \sum_{y \in \{0,1\}^n} \lambda_y y = x \\ & && \sum_{y \in \{0,1\}^n} \lambda_y = 1 \\ & && \lambda_y \geq 0, && \text{for } y \in \{0, 1\}^n. \end{aligned}$$

Proof

- $OPT(x)$ is at least $f^-(x)$ for every x : By Jensen's inequality

Claim

The value of the convex closure f^- at $x \in [0, 1]^n$ is the solution of the following optimization problem:

$$\begin{array}{ll} \text{minimize} & \sum_{y \in \{0,1\}^n} \lambda_y f(y) \\ \text{subject to} & \sum_{y \in \{0,1\}^n} \lambda_y y = x \\ & \sum_{y \in \{0,1\}^n} \lambda_y = 1 \\ & \lambda_y \geq 0, \end{array} \quad \text{for } y \in \{0, 1\}^n .$$

Proof

- $OPT(x)$ is at least $f^-(x)$ for every x : By Jensen's inequality
- To show that $OPT(x)$ is equal to $f^-(x)$, suffices to show that it is a convex under-estimate of f

Claim

The value of the convex closure f^- at $x \in [0, 1]^n$ is the solution of the following optimization problem:

$$\begin{aligned} & \text{minimize} && \sum_{y \in \{0,1\}^n} \lambda_y f(y) \\ & \text{subject to} && \sum_{y \in \{0,1\}^n} \lambda_y y = x \\ & && \sum_{y \in \{0,1\}^n} \lambda_y = 1 \\ & && \lambda_y \geq 0, && \text{for } y \in \{0, 1\}^n. \end{aligned}$$

Proof

- $OPT(x)$ is at least $f^-(x)$ for every x : By Jensen's inequality
- To show that $OPT(x)$ is equal to $f^-(x)$, suffices to show that it is a convex under-estimate of f
- Under-estimate: $OPT(x) = f(x)$ for $x \in \{0, 1\}^n$

Claim

The value of the convex closure f^- at $x \in [0, 1]^n$ is the solution of the following optimization problem:

$$\begin{aligned} & \text{minimize} && \sum_{y \in \{0,1\}^n} \lambda_y f(y) \\ & \text{subject to} && \sum_{y \in \{0,1\}^n} \lambda_y y = x \\ & && \sum_{y \in \{0,1\}^n} \lambda_y = 1 \\ & && \lambda_y \geq 0, && \text{for } y \in \{0, 1\}^n. \end{aligned}$$

Proof

- $OPT(x)$ is at least $f^-(x)$ for every x : By Jensen's inequality
- To show that $OPT(x)$ is equal to $f^-(x)$, suffices to show that it is a convex under-estimate of f
- Under-estimate: $OPT(x) = f(x)$ for $x \in \{0, 1\}^n$
- Convex: The value of a minimization LP is convex in its right hand side constants (check)

Using the Convex Closure

Fact

The minimum of f^- is equal to the minimum of f , and moreover is attained at minimizers $y \in \{0, 1\}^n$ of f .

Proof

Using the Convex Closure

Fact

The minimum of f^- is equal to the minimum of f , and moreover is attained at minimizers $y \in \{0, 1\}^n$ of f .

Proof

- $f^-(y) = f(y)$ for every $y \in \{0, 1\}^n$
- Therefore $\min_{x \in [0, 1]^n} f^-(x) \leq \min_{y \in \{0, 1\}^n} f(y)$

Using the Convex Closure

Fact

The minimum of f^- is equal to the minimum of f , and moreover is attained at minimizers $y \in \{0, 1\}^n$ of f .

Proof

- $f^-(y) = f(y)$ for every $y \in \{0, 1\}^n$
- Therefore $\min_{x \in [0, 1]^n} f^-(x) \leq \min_{y \in \{0, 1\}^n} f(y)$
- For every x , $f^-(x)$ is the expected value of $f(y)$, for a random variable $y \in \{0, 1\}^n$ with expectation x .
- Therefore, $\min_{x \in [0, 1]^n} f^-(x) \geq \min_{y \in \{0, 1\}^n} f(y)$

Using the Convex Closure

Fact

The minimum of f^- is equal to the minimum of f , and moreover is attained at minimizers $y \in \{0, 1\}^n$ of f .

Good News?

We reduced minimizing set function f to minimizing a convex function f^- over a convex set $[0, 1]^n$. Are we done?

Using the Convex Closure

Fact

The minimum of f^- is equal to the minimum of f , and moreover is attained at minimizers $y \in \{0, 1\}^n$ of f .

Good News?

We reduced minimizing set function f to minimizing a convex function f^- over a convex set $[0, 1]^n$. Are we done?

Problem

In general, it is hard to evaluate f^- efficiently, let alone its derivative. This is indispensable for convex optimization algorithms.

Using the Convex Closure

Fact

The minimum of f^- is equal to the minimum of f , and moreover is attained at minimizers $y \in \{0, 1\}^n$ of f .

Good News?

We reduced minimizing set function f to minimizing a convex function f^- over a convex set $[0, 1]^n$. Are we done?

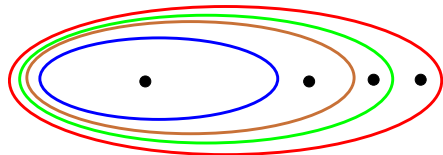
Problem

In general, it is hard to evaluate f^- efficiently, let alone its derivative. This is indispensable for convex optimization algorithms.

We will show that, when f is submodular, f^- is in fact equivalent to another extension which is easier to evaluate.

Chain Distribution

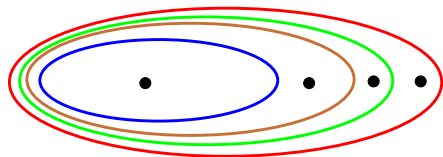
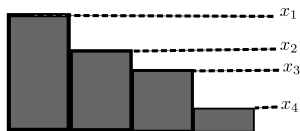
A **chain distribution** on the ground set X is a distribution over $S \subseteq X$ whose support forms a chain in the inclusion order.



Chain Distributions

Chain Distribution with Given Marginals

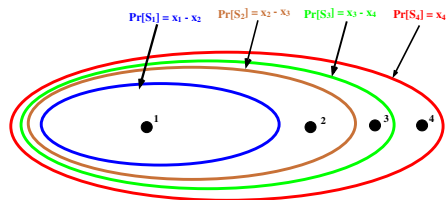
Fix the ground set $X = \{1, \dots, n\}$. The **chain distribution with marginals** $x \in [0, 1]^n$ is the unique chain distribution $D^{\mathcal{L}}(x)$ satisfying $\Pr_{S \sim D^{\mathcal{L}}(x)}[i \in S] = x_i$ for all $i \in X$.



Chain Distributions

Chain Distribution with Given Marginals

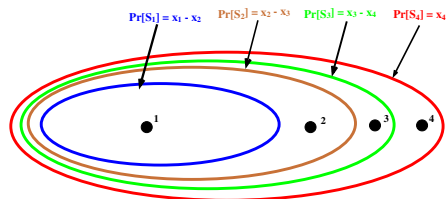
Fix the ground set $X = \{1, \dots, n\}$. The **chain distribution with marginals** $x \in [0, 1]^n$ is the unique chain distribution $D^{\mathcal{L}}(x)$ satisfying $\Pr_{S \sim D^{\mathcal{L}}(x)}[i \in S] = x_i$ for all $i \in X$.



Chain Distributions

Chain Distribution with Given Marginals

Fix the ground set $X = \{1, \dots, n\}$. The **chain distribution with marginals** $x \in [0, 1]^n$ is the unique chain distribution $D^{\mathcal{L}}(x)$ satisfying $\Pr_{S \sim D^{\mathcal{L}}(x)}[i \in S] = x_i$ for all $i \in X$.



$D^{\mathcal{L}}(x)$ is the distribution given by the following process:

- Sort $x_1 \geq x_2 \dots \geq x_n$
- Let $S_i = \{x_1, \dots, x_i\}$
- Let $\Pr[S_i] = x_i - x_{i+1}$

The Lovasz Extension

Definition

The **Lovasz extension** of a set function f is defined as follows.

$$f^{\mathcal{L}}(x) = \mathbf{E}_{S \sim D^{\mathcal{L}}(x)} f(S)$$

i.e. the Lovasz extension at x is the expected value of a set drawn from the unique chain distribution with marginals x .

Observations

- $f^{\mathcal{L}}$ is an extension, since the chain distribution with marginals $y \in \{0, 1\}^n$ is the point distribution at y .

The Lovasz Extension

Definition

The **Lovasz extension** of a set function f is defined as follows.

$$f^{\mathcal{L}}(x) = \mathbf{E}_{S \sim D^{\mathcal{L}}(x)} f(S)$$

i.e. the Lovasz extension at x is the expected value of a set drawn from the unique chain distribution with marginals x .

Observations

- $f^{\mathcal{L}}$ is an extension, since the chain distribution with marginals $y \in \{0, 1\}^n$ is the point distribution at y .
- $f^{\mathcal{L}}(x)$ is the expected value of f on some distribution on $\{0, 1\}^n$ with marginals x , therefore $f^{\mathcal{L}}(x) \geq f^-(x)$.

The Lovasz Extension

Definition

The **Lovasz extension** of a set function f is defined as follows.

$$f^{\mathcal{L}}(x) = \mathbf{E}_{S \sim D^{\mathcal{L}}(x)} f(S)$$

i.e. the Lovasz extension at x is the expected value of a set drawn from the unique chain distribution with marginals x .

Observations

- $f^{\mathcal{L}}$ is an extension, since the chain distribution with marginals $y \in \{0, 1\}^n$ is the point distribution at y .
- $f^{\mathcal{L}}(x)$ is the expected value of f on some distribution on $\{0, 1\}^n$ with marginals x , therefore $f^{\mathcal{L}}(x) \geq f^-(x)$.
- Together, those imply: if $f^{\mathcal{L}}$ is convex, then $f^{\mathcal{L}} = f^-$.

Equivalence of the Convex Closure and Lovasz Extension

Theorem

If f is submodular, then $f^{\mathcal{L}} = f^-$.

Converse holds: if f not submodular, then $f^{\mathcal{L}}$ not convex. (won't prove)

Equivalence of the Convex Closure and Lovasz Extension

Theorem

If f is submodular, then $f^{\mathcal{L}} = f^-$.

Converse holds: if f not submodular, then $f^{\mathcal{L}}$ not convex. (won't prove)

Intuition

- Recall: $f^-(x)$ evaluates f on the “lowest” distribution with marginals x
- It turns out that, when f is submodular, this lowest distribution is the chain distribution $D^{\mathcal{L}}(x)$.

Equivalence of the Convex Closure and Lovasz Extension

Theorem

If f is submodular, then $f^{\mathcal{L}} = f^-$.

Converse holds: if f not submodular, then $f^{\mathcal{L}}$ not convex. (won't prove)

Intuition

- Recall: $f^-(x)$ evaluates f on the “lowest” distribution with marginals x
- It turns out that, when f is submodular, this lowest distribution is the chain distribution $D^{\mathcal{L}}(x)$.
- Contingent on marginals x , submodularity implies that cost is minimized by “packing” as many elements together as possible
 - diminishing marginal returns
- This gives the chain distribution

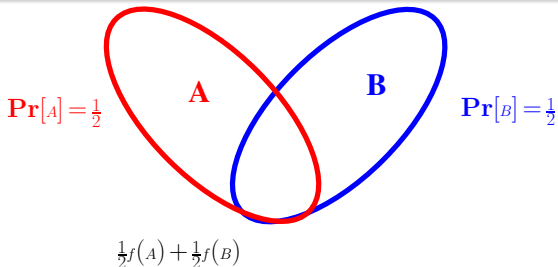
It suffices to show that the chain distribution with marginals x is in fact the “lowest” distribution with marginals x .

Proof (Special case)

It suffices to show that the chain distribution with marginals x is in fact the “lowest” distribution with marginals x .

Proof (Special case)

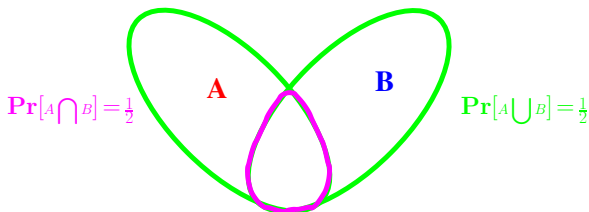
- Consider a distribution \mathcal{D} on two “crossing” sets A and B , with probability 0.5 each.



It suffices to show that the chain distribution with marginals x is in fact the “lowest” distribution with marginals x .

Proof (Special case)

- Consider a distribution \mathcal{D} on two “crossing” sets A and B , with probability 0.5 each.
- “uncrossing” implies that replacing them with $A \cap B$ and $A \cup B$, with probability 0.5 each, gives a chain distribution with lower expected value of f .

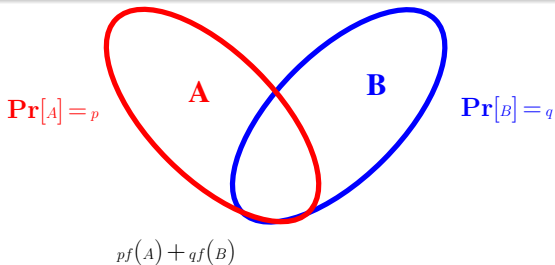


$$\frac{1}{2}f(A) + \frac{1}{2}f(B) \geq \frac{1}{2}f(A \cap B) + \frac{1}{2}f(A \cup B)$$

Proof (Slightly Less Special Case)

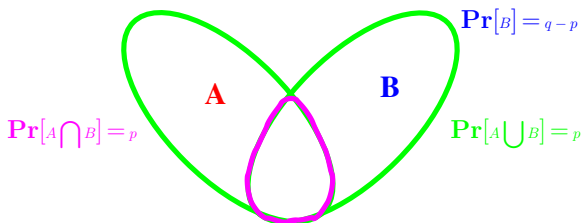
Proof (Slightly Less Special Case)

- Consider a distribution \mathcal{D} on two “crossing” sets A and B , with probabilities $p \leq q$.



Proof (Slightly Less Special Case)

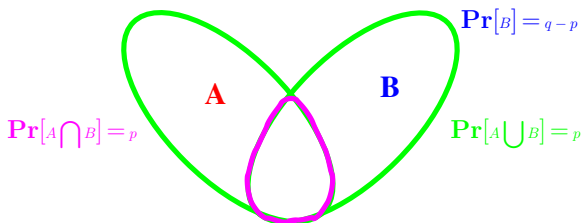
- Consider a distribution \mathcal{D} on two “crossing” sets A and B , with probabilities $p \leq q$.
- Can “uncross” a probability mass of p of each, decreasing the expected value of f



$$pf(A) + qf(B) \geq pf(A \cap B) + pf(A \cup B) + (q - p)f(B)$$

Proof (Slightly Less Special Case)

- Consider a distribution \mathcal{D} on two “crossing” sets A and B , with probabilities $p \leq q$.
- Can “uncross” a probability mass of p of each, decreasing the expected value of f
- Now a chain distribution

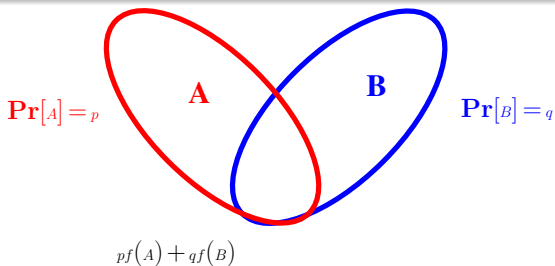


$$pf(A) + qf(B) \geq pf(A \cap B) + pf(A \cup B) + (q - p)f(B)$$

Proof (General Case)

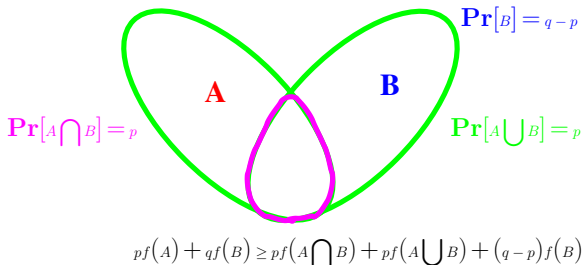
Proof (General Case)

- Consider a distribution \mathcal{D} which includes two “crossing” sets A and B in its support



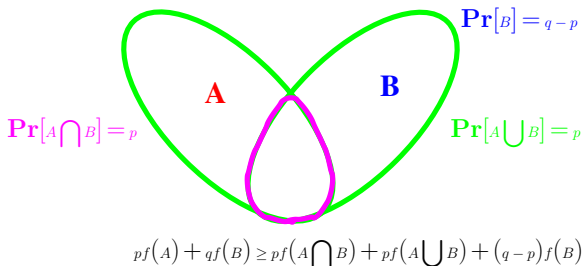
Proof (General Case)

- Consider a distribution \mathcal{D} which includes two “crossing” sets A and B in its support
- Can “uncross” a probability mass of $\min(\Pr[A], \Pr[B])$ of each, decreasing expected value of f



Proof (General Case)

- Consider a distribution \mathcal{D} which includes two “crossing” sets A and B in its support
- Can “uncross” a probability mass of $\min(\Pr[A], \Pr[B])$ of each, decreasing expected value of f
- Makes it closer to being a chain distribution
 - The bounded potential function $\mathbf{E}_{S \sim \mathcal{D}}[|S|^2]$ increases



Minimizing the Lovasz Extension

Because $f^{\mathcal{L}} = f^-$, we know the following:

Fact

The minimum of $f^{\mathcal{L}}$ is equal to the minimum of f , and moreover is attained at minimizers $y \in \{0, 1\}^n$ of f .

Minimizing the Lovasz Extension

Because $f^{\mathcal{L}} = f^-$, we know the following:

Fact

The minimum of $f^{\mathcal{L}}$ is equal to the minimum of f , and moreover is attained at minimizers $y \in \{0, 1\}^n$ of f .

Therefore, minimizing f reduces to the following convex optimization problem

Minimizing the Lovasz Extension

$$\begin{array}{ll} \text{minimize} & f^{\mathcal{L}}(x) \\ \text{subject to} & x \in [0, 1]^n \end{array}$$

Weak Solvability

An algorithm **weakly solves** our optimization problem if it takes in approximation parameter $\epsilon > 0$, runs in $\text{poly}(n, \log \frac{1}{\epsilon})$ time, and returns $x \in [0, 1]^n$ which is ϵ -optimal:

$$f^{\mathcal{L}}(x) \leq \min_{y \in [0, 1]^n} f^{\mathcal{L}}(y) + \epsilon \left[\max_{y \in [0, 1]^n} f^{\mathcal{L}}(y) - \min_{y \in [0, 1]^n} f^{\mathcal{L}}(y) \right]$$

Polynomial Solvability of CP

In order to **weakly** minimize $f^{\mathcal{L}}$, we need the following operations to run in $\text{poly}(n)$ time:

- 1 Compute a **starting ellipsoid** $E \supseteq [0, 1]^n$ with
$$\frac{\text{vol}(E)}{\text{vol}([0, 1]^n)} = O(\exp(n)).$$
- 2 A **separation oracle** for the feasible set $[0, 1]^n$
- 3 A **first order oracle** for $f^{\mathcal{L}}$: evaluates $f^{\mathcal{L}}(x)$ and a subgradient of $f^{\mathcal{L}}$ at x .

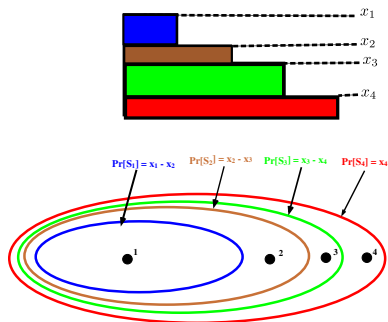
Polynomial Solvability of CP

In order to **weakly** minimize $f^{\mathcal{L}}$, we need the following operations to run in $\text{poly}(n)$ time:

- 1 Compute a **starting ellipsoid** $E \supseteq [0, 1]^n$ with
$$\frac{\text{vol}(E)}{\text{vol}([0, 1]^n)} = O(\exp(n)).$$
- 2 A **separation oracle** for the feasible set $[0, 1]^n$
- 3 A **first order oracle** for $f^{\mathcal{L}}$: evaluates $f^{\mathcal{L}}(x)$ and a subgradient of $f^{\mathcal{L}}$ at x .

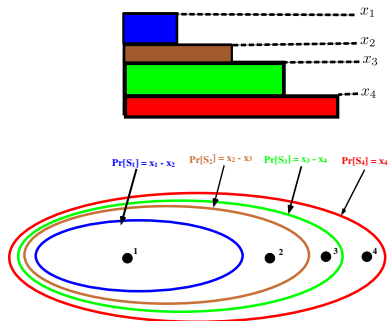
1 and 2 are trivial.

First order Oracle for $f^{\mathcal{L}}$



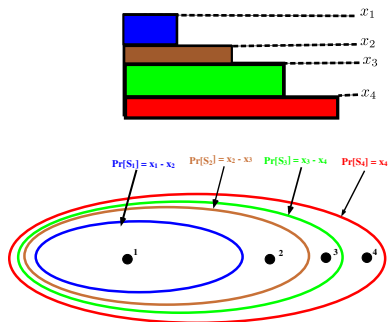
- Recall: the chain distribution with marginals x
 - Sort $x_1 \geq x_2 \dots \geq x_n$
 - Let $S_i = \{x_1, \dots, x_i\}$
 - Let $\Pr[S_i] = x_i - x_{i+1}$

First order Oracle for $f^{\mathcal{L}}$



- Recall: the chain distribution with marginals x
 - Sort $x_1 \geq x_2 \dots \geq x_n$
 - Let $S_i = \{x_1, \dots, x_i\}$
 - Let $\Pr[S_i] = x_i - x_{i+1}$
- Can evaluate $f^{\mathcal{L}}(x) = \sum_i f(S_i)(x_i - x_{i+1})$

First order Oracle for $f^{\mathcal{L}}$



- Recall: the chain distribution with marginals x
 - Sort $x_1 \geq x_2 \dots \geq x_n$
 - Let $S_i = \{x_1, \dots, x_i\}$
 - Let $\text{Pr}[S_i] = x_i - x_{i+1}$
- Can evaluate $f^{\mathcal{L}}(x) = \sum_i f(S_i)(x_i - x_{i+1})$
- $f^{\mathcal{L}}$ is peicwise linear, so can compute a sub-gradient.

Recovering an Optimal Set

We can get an ϵ -optimal solution x^* to the optimization problem in $\text{poly}(n, \log \frac{1}{\epsilon})$ time.

Minimizing the Lovasz Extension

$$\begin{array}{ll} \text{minimize} & f^{\mathcal{L}}(x) \\ \text{subject to} & x \in [0, 1]^n \end{array}$$

Recovering an Optimal Set

We can get an ϵ -optimal solution x^* to the optimization problem in $\text{poly}(n, \log \frac{1}{\epsilon})$ time.

Minimizing the Lovasz Extension

$$\begin{array}{ll} \text{minimize} & f^{\mathcal{L}}(x) \\ \text{subject to} & x \in [0, 1]^n \end{array}$$

- Set $\epsilon < 2^{-b}$, runtime is $\text{poly}(n, b)$.

Recovering an Optimal Set

We can get an ϵ -optimal solution x^* to the optimization problem in $\text{poly}(n, \log \frac{1}{\epsilon})$ time.

Minimizing the Lovasz Extension

$$\begin{array}{ll} \text{minimize} & f^{\mathcal{L}}(x) \\ \text{subject to} & x \in [0, 1]^n \end{array}$$

- Set $\epsilon < 2^{-b}$, runtime is $\text{poly}(n, b)$.
- $\min f(S) \leq f^{\mathcal{L}}(x^*) < \min 2 f(S)$

Recovering an Optimal Set

We can get an ϵ -optimal solution x^* to the optimization problem in $\text{poly}(n, \log \frac{1}{\epsilon})$ time.

Minimizing the Lovasz Extension

$$\begin{array}{ll} \text{minimize} & f^{\mathcal{L}}(x) \\ \text{subject to} & x \in [0, 1]^n \end{array}$$

- Set $\epsilon < 2^{-b}$, runtime is $\text{poly}(n, b)$.
- $\min f(S) \leq f^{\mathcal{L}}(x^*) < \min 2 f(S)$
- $f^{\mathcal{L}}(x^*)$ is the expectation f over a distribution of sets
 - It must include an optimal set in its support

Recovering an Optimal Set

We can get an ϵ -optimal solution x^* to the optimization problem in $\text{poly}(n, \log \frac{1}{\epsilon})$ time.

Minimizing the Lovasz Extension

$$\begin{array}{ll} \text{minimize} & f^{\mathcal{L}}(x) \\ \text{subject to} & x \in [0, 1]^n \end{array}$$

- Set $\epsilon < 2^{-b}$, runtime is $\text{poly}(n, b)$.
- $\min f(S) \leq f^{\mathcal{L}}(x^*) < \min 2 f(S)$
- $f^{\mathcal{L}}(x^*)$ is the expectation f over a distribution of sets
 - It must include an optimal set in its support
- We can identify this set by examining the chain distribution with marginals x^*

Outline

- 1 Introduction to Submodular Functions
- 2 Unconstrained Submodular Minimization
 - Definition and Examples
 - The Convex Closure and the Lovasz Extension
 - Wrapping up
- 3 **Monotone Submodular Maximization s.t. a Matroid Constraint**
 - **Definition and Examples**
 - **Warmup: Cardinality Constraint**
 - **General Matroid Constraints**

Recall: Optimizing Submodular Functions

	Maximization	Minimization
Unconstrained	NP-hard $\frac{1}{2}$ approximation	Polynomial time via convex opt
Constrained	Usually NP-hard $1 - 1/e$ (mono, matroid) $O(1)$ (“nice” constraints)	Usually NP-hard to apx. Few easy special cases

Recall: Optimizing Submodular Functions

	Maximization	Minimization
Unconstrained	NP-hard $\frac{1}{2}$ approximation	Polynomial time via convex opt
Constrained	Usually NP-hard $1 - 1/e$ (mono, matroid) $O(1)$ (“nice” constraints)	Usually NP-hard to apx. Few easy special cases

Problem Definition

Given a **non-decreasing** and **normalized** submodular function $f : 2^X \rightarrow \mathbb{R}_+$ on a finite ground set X , and a matroid $M = (X, \mathcal{I})$

$$\begin{array}{ll} \text{maximize} & f(S) \\ \text{subject to} & S \in \mathcal{I} \end{array}$$

- Non-decreasing: $f(S) \leq f(T)$ for $S \subseteq T$
- Normalized: $f(\emptyset) = 0$.

Problem Definition

Given a **non-decreasing** and **normalized** submodular function $f : 2^X \rightarrow \mathbb{R}_+$ on a finite ground set X , and a matroid $M = (X, \mathcal{I})$

$$\begin{array}{ll} \text{maximize} & f(S) \\ \text{subject to} & S \in \mathcal{I} \end{array}$$

- Non-decreasing: $f(S) \leq f(T)$ for $S \subseteq T$
- Normalized: $f(\emptyset) = 0$.
- We denote $n = |X|$

Problem Definition

Given a **non-decreasing** and **normalized** submodular function $f : 2^X \rightarrow \mathbb{R}_+$ on a finite ground set X , and a matroid $M = (X, \mathcal{I})$

$$\begin{array}{ll} \text{maximize} & f(S) \\ \text{subject to} & S \in \mathcal{I} \end{array}$$

- Non-decreasing: $f(S) \leq f(T)$ for $S \subseteq T$
- Normalized: $f(\emptyset) = 0$.
- We denote $n = |X|$

Representation

As before, we work in the **value oracle** and **independence oracle** models. Namely, we assume we have access to a subroutine evaluating $f(S)$, and a subroutine for checking whether $S \in \mathcal{I}$, each in constant time.

Maximum Coverage

X is the left hand side of a graph, and $f(S)$ is the total number of neighbors of S .

- Can think of $i \in X$ as a set, and $f(S)$ as the total “coverage” of S .

Goal is to cover as much of the RHS as possible with k LHS nodes.

Social Influence

- X is the family of nodes in a social network
- A meme, idea, or product is adopted at a set of nodes S
- $f(S)$ is the expected number of nodes in the network which end up adopting the idea.
- Goal is to obtain maximum influence subject to a constraint
 - Cardinality
 - Transversal
 - ...

Combinatorial Allocation

- G is a set of goods
- $f_i(B)$ is submodular utility of agent $i \in N$ for bundle $B \subseteq G$
- Allocation: A partition (B_1, \dots, B_n) of G among agents.
- Aggregate utility is $\sum_i f_i(B_i)$.

Combinatorial Allocation

- G is a set of goods
- $f_i(B)$ is submodular utility of agent $i \in N$ for bundle $B \subseteq G$
- Allocation: A partition (B_1, \dots, B_n) of G among agents.
- Aggregate utility is $\sum_i f_i(B_i)$.
- Let $X = G \times N$ be the set of good/agent pairs
- Allocations correspond to subsets S of X in which at most one “copy” of each good is chosen
 - Partition matroid constraint
- $f(S) = \sum_{i \in N} f_i(\{j \in G : (j, i) \in S\})$
 - Submodular

Theorem

Maximizing a submodular function subject to a matroid constraint is NP-hard, and NP-hard to approximate to within any better than a factor of $1 - 1/e$.

- Holds even for max coverage subject to a cardinality constraint (Feige '98)

Theorem

Maximizing a submodular function subject to a matroid constraint is NP-hard, and NP-hard to approximate to within any better than a factor of $1 - 1/e$.

- Holds even for max coverage subject to a cardinality constraint (Feige '98)

Goal

An algorithm in the value oracle and independence oracle models which

- Runs in time $\text{poly}(n)$
- Returns a feasible set $S^* \in \mathcal{I}$ satisfying $f(S^*) \geq (1 - 1/e) \max_{S \in \mathcal{I}} f(S)$.

Theorem

Maximizing a submodular function subject to a matroid constraint is NP-hard, and NP-hard to approximate to within any better than a factor of $1 - 1/e$.

- Holds even for max coverage subject to a cardinality constraint (Feige '98)

Goal

An algorithm in the value oracle and independence oracle models which

- Runs in time $\text{poly}(n)$
- Returns a feasible set $S^* \in \mathcal{I}$ satisfying $f(S^*) \geq (1 - 1/e) \max_{S \in \mathcal{I}} f(S)$.

Holds for arbitrary matroid, but much simpler for uniform matroids.

Problem Definition

Given a **non-decreasing** and **normalized** submodular function $f : 2^X \rightarrow \mathbb{R}_+$ on a finite ground set X with $|X| = n$, and an integer $k \leq n$

$$\begin{array}{ll} \text{maximize} & f(S) \\ \text{subject to} & |S| \leq k \end{array}$$

- k -uniform matroid constraint

The Greedy Algorithm

The following is the straightforward adaptation of the greedy algorithm for maximizing modular functions over a matroid.

The Greedy Algorithm

- 1 $S \leftarrow \emptyset$
- 2 While $|S| \leq k$
 - Choose $e \in X$ maximizing $f(S \cup \{e\})$
 - $S \leftarrow S \cup \{e\}$

The Greedy Algorithm

The following is the straightforward adaptation of the greedy algorithm for maximizing modular functions over a matroid.

The Greedy Algorithm

- 1 $S \leftarrow \emptyset$
- 2 While $|S| \leq k$
 - Choose $e \in X$ maximizing $f(S \cup \{e\})$
 - $S \leftarrow S \cup \{e\}$

Theorem

The greedy algorithm is a $(1 - 1/e)$ approximation algorithm for maximizing a monotone, normalized, and submodular function subject to a cardinality constraint.

Contraction/Conditioning

Let $f : 2^X \rightarrow \mathbb{R}$ and $A \subseteq X$. Define $f_A(S) = f(A \cup S) - f(A)$.

Lemma

If f is monotone and submodular, then f_A is monotone, submodular, and normalized for any A .

Contraction/Conditioning

Let $f : 2^X \rightarrow \mathbb{R}$ and $A \subseteq X$. Define $f_A(S) = f(A \cup S) - f(A)$.

Lemma

If f is monotone and submodular, then f_A is monotone, submodular, and normalized for any A .

Proof

- Normalized: trivial

Contraction/Conditioning

Let $f : 2^X \rightarrow \mathbb{R}$ and $A \subseteq X$. Define $f_A(S) = f(A \cup S) - f(A)$.

Lemma

If f is monotone and submodular, then f_A is monotone, submodular, and normalized for any A .

Proof

- Normalized: trivial
- Monotone:
 - Let $S \subseteq T$
 - $f_A(S) = f(S \cup A) - f(A) \leq f(T \cup A) - f(A) = f_A(T)$.

Contraction/Conditioning

Let $f : 2^X \rightarrow \mathbb{R}$ and $A \subseteq X$. Define $f_A(S) = f(A \cup S) - f(A)$.

Lemma

If f is monotone and submodular, then f_A is monotone, submodular, and normalized for any A .

Proof

- Normalized: trivial
- Monotone:
 - Let $S \subseteq T$
 - $f_A(S) = f(S \cup A) - f(A) \leq f(T \cup A) - f(A) = f_A(T)$.
- Submodular:

$$\begin{aligned} f_A(S) + f_A(T) &= f(S \cup A) - f(A) + f(T \cup A) - f(A) \\ &\geq f(S \cup T \cup A) - f(A) + f((S \cap T) \cup A) - f(A) \\ &= f_A(S \cup T) + f_A(S \cap T) \end{aligned}$$

Lemma

If f is normalized and submodular, and $A \subseteq X$, then there is $j \in A$ such that $f(\{j\}) \geq \frac{1}{|A|} f(A)$.

Lemma

If f is normalized and submodular, and $A \subseteq X$, then there is $j \in A$ such that $f(\{j\}) \geq \frac{1}{|A|}f(A)$.

Proof

- If A_1, A_2 partition A , then

$$f(A_1) + f(A_2) \geq f(A_1 \cup A_2) + f(A_1 \cap A_2) = f(A)$$

Lemma

If f is normalized and submodular, and $A \subseteq X$, then there is $j \in A$ such that $f(\{j\}) \geq \frac{1}{|A|}f(A)$.

Proof

- If A_1, A_2 partition A , then

$$f(A_1) + f(A_2) \geq f(A_1 \cup A_2) + f(A_1 \cap A_2) = f(A)$$

- Applying recursively, we get

$$\sum_{j \in A} f(\{j\}) \geq f(A)$$

Lemma

If f is normalized and submodular, and $A \subseteq X$, then there is $j \in A$ such that $f(\{j\}) \geq \frac{1}{|A|}f(A)$.

Proof

- If A_1, A_2 partition A , then

$$f(A_1) + f(A_2) \geq f(A_1 \cup A_2) + f(A_1 \cap A_2) = f(A)$$

- Applying recursively, we get

$$\sum_{j \in A} f(\{j\}) \geq f(A)$$

- Therefore, $\max_{j \in A} f(\{j\}) \geq \frac{1}{|A|}f(A)$

Theorem

The greedy algorithm is a $(1 - 1/e)$ approximation algorithm for maximizing a monotone, normalized, and submodular function subject to a cardinality constraint.

Proof

- Let S be the working set in the algorithm

Theorem

The greedy algorithm is a $(1 - 1/e)$ approximation algorithm for maximizing a monotone, normalized, and submodular function subject to a cardinality constraint.

Proof

- Let S be the working set in the algorithm
- Let S^* be optimal solution with $f(S^*) = OPT$.

Theorem

The greedy algorithm is a $(1 - 1/e)$ approximation algorithm for maximizing a monotone, normalized, and submodular function subject to a cardinality constraint.

Proof

- Let S be the working set in the algorithm
- Let S^* be optimal solution with $f(S^*) = OPT$.
- We will show that the suboptimality $OPT - f(S)$ shrinks by a factor of $(1 - 1/k)$ each iteration

Theorem

The greedy algorithm is a $(1 - 1/e)$ approximation algorithm for maximizing a monotone, normalized, and submodular function subject to a cardinality constraint.

Proof

- Let S be the working set in the algorithm
- Let S^* be optimal solution with $f(S^*) = OPT$.
- We will show that the suboptimality $OPT - f(S)$ shrinks by a factor of $(1 - 1/k)$ each iteration
- After k iterations, it has shrunk to $(1 - 1/k)^k \leq 1/e$ from its original value

$$OPT - f(S) \leq \frac{1}{e} OPT$$

$$(1 - 1/e)OPT \leq f(S)$$

Theorem

The greedy algorithm is a $(1 - 1/e)$ approximation algorithm for maximizing a monotone, normalized, and submodular function subject to a cardinality constraint.

Proof

- By definition, in each iteration $f(S)$ increases by $\max_j f_S(\{j\})$

Theorem

The greedy algorithm is a $(1 - 1/e)$ approximation algorithm for maximizing a monotone, normalized, and submodular function subject to a cardinality constraint.

Proof

- By definition, in each iteration $f(S)$ increases by $\max_j f_S(\{j\})$
- By our lemmas, there is $j' \in S^*$ s.t.

$$f_S(\{j'\}) \geq \frac{1}{|S^*|} f_S(S^*)$$

Theorem

The greedy algorithm is a $(1 - 1/e)$ approximation algorithm for maximizing a monotone, normalized, and submodular function subject to a cardinality constraint.

Proof

- By definition, in each iteration $f(S)$ increases by $\max_j f_S(\{j\})$
- By our lemmas, there is $j' \in S^*$ s.t.

$$\begin{aligned} f_S(\{j'\}) &\geq \frac{1}{|S^*|} f_S(S^*) \\ &= \frac{1}{k} (f(S \cup S^*) - f(S)) \end{aligned}$$

Theorem

The greedy algorithm is a $(1 - 1/e)$ approximation algorithm for maximizing a monotone, normalized, and submodular function subject to a cardinality constraint.

Proof

- By definition, in each iteration $f(S)$ increases by $\max_j f_S(\{j\})$
- By our lemmas, there is $j' \in S^*$ s.t.

$$\begin{aligned} f_S(\{j'\}) &\geq \frac{1}{|S^*|} f_S(S^*) \\ &= \frac{1}{k} (f(S \cup S^*) - f(S)) \\ &\geq \frac{1}{k} (OPT - f(S)) \end{aligned}$$

Theorem

The greedy algorithm is a $(1 - 1/e)$ approximation algorithm for maximizing a monotone, normalized, and submodular function subject to a cardinality constraint.

Proof

- By definition, in each iteration $f(S)$ increases by $\max_j f_S(\{j\})$
- By our lemmas, there is $j' \in S^*$ s.t.

$$\begin{aligned} f_S(\{j'\}) &\geq \frac{1}{|S^*|} f_S(S^*) \\ &= \frac{1}{k} (f(S \cup S^*) - f(S)) \\ &\geq \frac{1}{k} (OPT - f(S)) \end{aligned}$$

- Therefore, suboptimality decreases by factor of $1 - \frac{1}{k}$, as needed.

From Uniform to Arbitrary Matroid

Problem Definition

Given a **non-decreasing** and **normalized** submodular function $f : 2^X \rightarrow \mathbb{R}_+$ on a finite ground set X , and a matroid $M = (X, \mathcal{I})$

$$\begin{array}{ll} \text{maximize} & f(S) \\ \text{subject to} & S \in \mathcal{I} \end{array}$$

From Uniform to Arbitrary Matroid

Problem Definition

Given a **non-decreasing** and **normalized** submodular function $f : 2^X \rightarrow \mathbb{R}_+$ on a finite ground set X , and a matroid $M = (X, \mathcal{I})$

$$\begin{array}{ll} \text{maximize} & f(S) \\ \text{subject to} & S \in \mathcal{I} \end{array}$$

- The discrete greedy algorithm is now only a $1/2$ approximation
 - Partition matroid with parts $\{a\}$ and $\{b, c\}$ and budgets 1
 - $f(a) = f(b) = 1$, $f(c) = f(ac) = 1 + \epsilon$, $f(ab) = f(bc) = f(abc) = 2$

From Uniform to Arbitrary Matroid

Problem Definition

Given a **non-decreasing** and **normalized** submodular function $f : 2^X \rightarrow \mathbb{R}_+$ on a finite ground set X , and a matroid $M = (X, \mathcal{I})$

$$\begin{array}{ll} \text{maximize} & f(S) \\ \text{subject to} & S \in \mathcal{I} \end{array}$$

- The discrete greedy algorithm is now only a $1/2$ approximation
 - Partition matroid with parts $\{a\}$ and $\{b, c\}$ and budgets 1
 - $f(a) = f(b) = 1$, $f(c) = f(ac) = 1 + \epsilon$, $f(ab) = f(bc) = f(abc) = 2$
- Nevertheless, a continuous greedy algorithm gives $1 - 1/e$

From Uniform to Arbitrary Matroid

Problem Definition

Given a **non-decreasing** and **normalized** submodular function $f : 2^X \rightarrow \mathbb{R}_+$ on a finite ground set X , and a matroid $M = (X, \mathcal{I})$

$$\begin{array}{ll} \text{maximize} & f(S) \\ \text{subject to} & S \in \mathcal{I} \end{array}$$

- The discrete greedy algorithm is now only a $1/2$ approximation
 - Partition matroid with parts $\{a\}$ and $\{b, c\}$ and budgets 1
 - $f(a) = f(b) = 1$, $f(c) = f(ac) = 1 + \epsilon$, $f(ab) = f(bc) = f(abc) = 2$
- Nevertheless, a continuous greedy algorithm gives $1 - 1/e$
- Approach resembles that for minimization
 - Define a continuous extension of f
 - Optimize continuous extension over matroid polytope
 - Extract an integer point

The Multilinear Extension

Multilinear Extension

Given a set function $f : \{0, 1\}^n \rightarrow \mathbb{R}$, its **multilinear extension** $F : [0, 1]^n \rightarrow \mathbb{R}$ evaluated at $x \in [0, 1]^n$ gives the expected value of $f(S)$ for the random set S which includes each i independently with probability x_i .

$$F(x) = \sum_{S \subseteq X} f(S) \prod_{i \in S} x_i \prod_{i \notin S} (1 - x_i)$$

The Multilinear Extension

Multilinear Extension

Given a set function $f : \{0, 1\}^n \rightarrow \mathbb{R}$, its **multilinear extension** $F : [0, 1]^n \rightarrow \mathbb{R}$ evaluated at $x \in [0, 1]^n$ gives the expected value of $f(S)$ for the random set S which includes each i independently with probability x_i .

$$F(x) = \sum_{S \subseteq X} f(S) \prod_{i \in S} x_i \prod_{i \notin S} (1 - x_i)$$

- For each point x , evaluates f on the independent distribution $D(x)$

The Multilinear Extension

Multilinear Extension

Given a set function $f : \{0, 1\}^n \rightarrow \mathbb{R}$, its **multilinear extension** $F : [0, 1]^n \rightarrow \mathbb{R}$ evaluated at $x \in [0, 1]^n$ gives the expected value of $f(S)$ for the random set S which includes each i independently with probability x_i .

$$F(x) = \sum_{S \subseteq X} f(S) \prod_{i \in S} x_i \prod_{i \notin S} (1 - x_i)$$

- For each point x , evaluates f on the independent distribution $D(x)$
- Clearly an extension of f

The Multilinear Extension

Multilinear Extension

Given a set function $f : \{0, 1\}^n \rightarrow \mathbb{R}$, its **multilinear extension** $F : [0, 1]^n \rightarrow \mathbb{R}$ evaluated at $x \in [0, 1]^n$ gives the expected value of $f(S)$ for the random set S which includes each i independently with probability x_i .

$$F(x) = \sum_{S \subseteq X} f(S) \prod_{i \in S} x_i \prod_{i \notin S} (1 - x_i)$$

- For each point x , evaluates f on the independent distribution $D(x)$
- Clearly an extension of f
- Not concave (or convex) in general
 - Recall f with $f(\emptyset) = 0$ and $f(\{1\}) = f(\{2\}) = f(\{1, 2\}) = 1$
 - $F(x) = 1 - (1 - x_1)(1 - x_2)$

Easy Properties of the Multilinear Extension

Normalized

When f is normalized, $F(0) = 0$

Follows from the fact that F is an extension of f

Easy Properties of the Multilinear Extension

Normalized

When f is normalized, $F(0) = 0$

Follows from the fact that F is an extension of f

Nondecreasing

When f is monotone non-decreasing, $F(x) \leq F(y)$ whenever $x \preceq y$ component-wise.

Increasing the probability of selecting each element increases the expected value.

Up-concavity

Even though F is not concave, it is concave in “upwards” directions.

Up-concavity

Assume f is submodular. For every $\vec{a} \in [0, 1]^n$ and $\vec{d} \in [0, 1]^n$ satisfying $d \succeq 0$, the function $g(t) = F(\vec{a} + \vec{d}t)$ is a concave function of $t \in \mathbb{R}$.

Proof Sketch

- By multivariate chain rule: $\frac{d^2g}{dt^2} = d^T (\nabla^2 F) d$
- The Hessian $\nabla^2 F$ is not negative semi-definite, so can't conclude that g is concave for arbitrary directions d
- Multilinearity implies second partial derivatives $\frac{\partial^2 F}{\partial x_i^2}$ are zero
- Submodularity implies mixed derivatives $\frac{\partial^2 F}{\partial x_i \partial x_j}$ are nonpositive
 - Diminishing marginal returns + coupling argument
- Therefore $\frac{d^2g}{dt^2} = d^T (\nabla^2 F) d \leq 0$ for $\vec{d} \succeq 0$

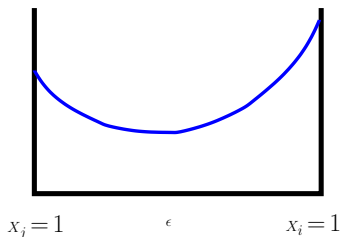
Cross-convexity

Nevertheless, F is convex in “cross” directions.

Cross-convexity

Assume f is submodular. For every $a \in [0, 1]^n$ and $\vec{d} = e_i - e_j$ for some $i, j \in X$, the function $g(t) = F(\vec{a} + \vec{d}t)$ is a convex function of $t \in \mathbb{R}$.

- Trading off one item’s probability for another’s gives convex curve
- Follows from submodularity: as we “remove” j , the marginal benefit of “adding” i increases



Cross-convexity

Nevertheless, F is convex in “cross” directions.

Cross-convexity

Assume f is submodular. For every $a \in [0, 1]^n$ and $\vec{d} = e_i - e_j$ for some $i, j \in X$, the function $g(t) = F(\vec{a} + \vec{d}t)$ is a convex function of $t \in \mathbb{R}$.

Proof

- $\frac{d^2g}{dt^2} = d^T (\nabla^2 F) d = \frac{\partial^2 F}{\partial x_i^2} + \frac{\partial^2 F}{\partial x_j^2} - 2 \frac{\partial^2 F}{\partial x_i \partial x_j}$
- By multilinearity, $\frac{\partial^2 F}{\partial x_i^2} = \frac{\partial^2 F}{\partial x_j^2} = 0$
- We already argued that submodularity implies $\frac{\partial^2 F}{\partial x_i \partial x_j} \leq 0$

Step A: Continuous Greedy Algorithm

Computes a $1 - 1/e$ approximation to the following continuous (non-convex) optimization problem.

$$\begin{array}{ll} \text{maximize} & F(x) \\ \text{subject to} & x \in \mathcal{P}(\mathcal{M}) \end{array}$$

- i.e. Computes x^* s.t. $F(x^*) \geq (1 - 1/e) \max \{F(x) : x \in \mathcal{P}(\mathcal{M})\}$

Step A: Continuous Greedy Algorithm

Computes a $1 - 1/e$ approximation to the following continuous (non-convex) optimization problem.

$$\begin{array}{ll} \text{maximize} & F(x) \\ \text{subject to} & x \in \mathcal{P}(\mathcal{M}) \end{array}$$

- i.e. Computes x^* s.t. $F(x^*) \geq (1 - 1/e) \max \{F(x) : x \in \mathcal{P}(\mathcal{M})\}$
- Note: $\max \{F(x) : x \in \mathcal{P}(\mathcal{M})\} \geq \max \{f(S) : S \in \mathcal{I}\}$

Step A: Continuous Greedy Algorithm

Computes a $1 - 1/e$ approximation to the following continuous (non-convex) optimization problem.

$$\begin{array}{ll} \text{maximize} & F(x) \\ \text{subject to} & x \in \mathcal{P}(\mathcal{M}) \end{array}$$

- i.e. Computes x^* s.t. $F(x^*) \geq (1 - 1/e) \max \{F(x) : x \in \mathcal{P}(\mathcal{M})\}$
- Note: $\max \{F(x) : x \in \mathcal{P}(\mathcal{M})\} \geq \max \{f(S) : S \in \mathcal{I}\}$
- $D(x^*)$ is a distribution over sets with expected value at least $(1 - 1/e)$ of our target
- Would we be done?

Step A: Continuous Greedy Algorithm

Computes a $1 - 1/e$ approximation to the following continuous (non-convex) optimization problem.

$$\begin{array}{ll} \text{maximize} & F(x) \\ \text{subject to} & x \in \mathcal{P}(\mathcal{M}) \end{array}$$

- i.e. Computes x^* s.t. $F(x^*) \geq (1 - 1/e) \max \{F(x) : x \in \mathcal{P}(\mathcal{M})\}$
- Note: $\max \{F(x) : x \in \mathcal{P}(\mathcal{M})\} \geq \max \{f(S) : S \in \mathcal{I}\}$
- $D(x^*)$ is a distribution over sets with expected value at least $(1 - 1/e)$ of our target
- Would we be done?

No! $D(x^*)$ may be mostly supported on infeasible sets (i.e. not independent in matroid \mathcal{M}).

Step B: Pipage Rounding

“Rounds” x^* to some vertex y^* of the matroid polytope (i.e. an independent set) satisfying

$$f(y^*) = F(y^*) \geq F(x^*)$$

Step B: Pipage Rounding

“Rounds” x^* to some vertex y^* of the matroid polytope (i.e. an independent set) satisfying

$$f(y^*) = F(y^*) \geq F(x^*)$$

- A-priori, not obvious that such a y^* exists

- The following “continuous” descent algorithm works for an arbitrary nondecreasing and up-concave function F , and solvable downwards-closed polytope $\mathcal{P} \subseteq \mathbb{R}_+^n$.
- Continuously moves a particle inside the matroid polytope, starting at 0, for a total of 1 time unit.
 - Position at time t given by $x(t)$.
- Discretized to time steps of ϵ , which we will assume to be arbitrarily small for convenience of analysis, but may be taken to be $1/\text{poly}(n)$ in the actual implementation.

- The following “continuous” descent algorithm works for an arbitrary nondecreasing and up-concave function F , and solvable downwards-closed polytope $\mathcal{P} \subseteq \mathbb{R}_+^n$.
- Continuously moves a particle inside the matroid polytope, starting at 0 , for a total of 1 time unit.
 - Position at time t given by $x(t)$.
- Discretized to time steps of ϵ , which we will assume to be arbitrarily small for convenience of analysis, but may be taken to be $1/\text{poly}(n)$ in the actual implementation.

Continuous Greedy Algorithm $(F, \mathcal{P}, \epsilon)$

- 1 $x(0) \leftarrow \vec{0}$
- 2 For $t \in [0, \epsilon, 2\epsilon, \dots, 1 - \epsilon]$
 - $x(t + \epsilon) \leftarrow x(t) + \epsilon \operatorname{argmax}_{y \in \mathcal{P}} \{\nabla F(x(t)) \cdot y\}$
- 3 Return $x(1)$

Continuous Greedy Algorithm (F, \mathcal{P}, ϵ)

- 1 $x(0) \leftarrow \vec{0}$
 - 2 For $t \in [0, \epsilon, 2\epsilon, \dots, 1 - \epsilon]$
 - $x(t + \epsilon) \leftarrow x(t) + \epsilon \operatorname{argmax}_{y \in \mathcal{P}} \{\nabla F(x(t)) \cdot y\}$
 - 3 Return $x(1)$
- I.e. When the particle is at x , it moves in direction y maximizing the linear function $\nabla F(x) \cdot y$ over $y \in \mathcal{P}$
 - The direction is actually a vertex of our matroid polytope
 - This is **NOT** gradient descent

Continuous Greedy Algorithm (F, \mathcal{P}, ϵ)

- 1 $x(0) \leftarrow \vec{0}$
 - 2 For $t \in [0, \epsilon, 2\epsilon, \dots, 1 - \epsilon]$
 - $x(t + \epsilon) \leftarrow x(t) + \epsilon \operatorname{argmax}_{y \in \mathcal{P}} \{\nabla F(x(t)) \cdot y\}$
 - 3 Return $x(1)$
-
- I.e. When the particle is at x , it moves in direction y maximizing the linear function $\nabla F(x) \cdot y$ over $y \in \mathcal{P}$
 - The direction is actually a vertex of our matroid polytope
 - This is **NOT** gradient descent
 - Observe: Algorithm forms a convex combination of $\frac{1}{\epsilon}$ vertices of the polytope \mathcal{P} , each with weight ϵ .
 - $x(1) \in \mathcal{P}$.

Theorem

Let F be nondecreasing and up-concave, and \mathcal{P} be a downwards closed polytope. In the limit as $\epsilon \rightarrow 0$, the continuous greedy algorithm outputs a $1 - 1/e$ approximation to maximizing $F(x)$ over \mathcal{P} .

Theorem

Let F be nondecreasing and up-concave, and \mathcal{P} be a downwards closed polytope. In the limit as $\epsilon \rightarrow 0$, the continuous greedy algorithm outputs a $1 - 1/e$ approximation to maximizing $F(x)$ over \mathcal{P} .

Proof Sketch

- Denote $y(t) = \operatorname{argmax}_{y \in \mathcal{P}} \nabla F(x(t)) \cdot y$
 - $\frac{d\vec{x}}{dt} = y(t)$

Theorem

Let F be nondecreasing and up-concave, and \mathcal{P} be a downwards closed polytope. In the limit as $\epsilon \rightarrow 0$, the continuous greedy algorithm outputs a $1 - 1/e$ approximation to maximizing $F(x)$ over \mathcal{P} .

Proof Sketch

- Denote $y(t) = \operatorname{argmax}_{y \in \mathcal{P}} \nabla F(x(t)) \cdot y$
 - $\frac{d\vec{x}}{dt} = y(t)$
- Let x_{opt} be the vertex of $\mathcal{P}(\mathcal{M})$ maximizing $F(x)$.
 - $F(x_{opt}) = f(x_{opt}) = OPT$

Theorem

Let F be nondecreasing and up-concave, and \mathcal{P} be a downwards closed polytope. In the limit as $\epsilon \rightarrow 0$, the continuous greedy algorithm outputs a $1 - 1/e$ approximation to maximizing $F(x)$ over \mathcal{P} .

Proof Sketch

- Denote $y(t) = \operatorname{argmax}_{y \in \mathcal{P}} \nabla F(x(t)) \cdot y$
 - $\frac{d\vec{x}}{dt} = y(t)$
- Let x_{opt} be the vertex of $\mathcal{P}(\mathcal{M})$ maximizing $F(x)$.
 - $F(x_{opt}) = f(x_{opt}) = OPT$

$$\frac{dF(x(t))}{dt}$$

$$\geq OPT - F(x(t))$$

Theorem

Let F be nondecreasing and up-concave, and \mathcal{P} be a downwards closed polytope. In the limit as $\epsilon \rightarrow 0$, the continuous greedy algorithm outputs a $1 - 1/e$ approximation to maximizing $F(x)$ over \mathcal{P} .

Proof Sketch

- Denote $y(t) = \operatorname{argmax}_{y \in \mathcal{P}} \nabla F(x(t)) \cdot y$
 - $\frac{d\vec{x}}{dt} = y(t)$
- Let x_{opt} be the vertex of $\mathcal{P}(\mathcal{M})$ maximizing $F(x)$.
 - $F(x_{opt}) = f(x_{opt}) = OPT$

$$\frac{dF(x(t))}{dt} = \nabla F(x(t)) \cdot \frac{d\vec{x}}{dt}$$

$$\geq OPT - F(x(t))$$

Theorem

Let F be nondecreasing and up-concave, and \mathcal{P} be a downwards closed polytope. In the limit as $\epsilon \rightarrow 0$, the continuous greedy algorithm outputs a $1 - 1/e$ approximation to maximizing $F(x)$ over \mathcal{P} .

Proof Sketch

- Denote $y(t) = \operatorname{argmax}_{y \in \mathcal{P}} \nabla F(x(t)) \cdot y$
 - $\frac{d\vec{x}}{dt} = y(t)$
- Let x_{opt} be the vertex of $\mathcal{P}(\mathcal{M})$ maximizing $F(x)$.
 - $F(x_{opt}) = f(x_{opt}) = OPT$

$$\begin{aligned}\frac{dF(x(t))}{dt} &= \nabla F(x(t)) \cdot \frac{d\vec{x}}{dt} \\ &= \nabla F(x(t)) \cdot y(t)\end{aligned}$$

$$\geq OPT - F(x(t))$$

Theorem

Let F be nondecreasing and up-concave, and \mathcal{P} be a downwards closed polytope. In the limit as $\epsilon \rightarrow 0$, the continuous greedy algorithm outputs a $1 - 1/e$ approximation to maximizing $F(x)$ over \mathcal{P} .

Proof Sketch

- Denote $y(t) = \operatorname{argmax}_{y \in \mathcal{P}} \nabla F(x(t)) \cdot y$
 - $\frac{d\vec{x}}{dt} = y(t)$
- Let x_{opt} be the vertex of $\mathcal{P}(\mathcal{M})$ maximizing $F(x)$.
 - $F(x_{opt}) = f(x_{opt}) = OPT$

$$\begin{aligned} \frac{dF(x(t))}{dt} &= \nabla F(x(t)) \cdot \frac{d\vec{x}}{dt} \\ &= \nabla F(x(t)) \cdot y(t) \\ &\geq \nabla F(x(t)) \cdot [x_{opt} - x(t)]^+ \\ &\geq OPT - F(x(t)) \end{aligned}$$

Theorem

Let F be nondecreasing and up-concave, and \mathcal{P} be a downwards closed polytope. In the limit as $\epsilon \rightarrow 0$, the continuous greedy algorithm outputs a $1 - 1/e$ approximation to maximizing $F(x)$ over \mathcal{P} .

Proof Sketch

- Denote $y(t) = \operatorname{argmax}_{y \in \mathcal{P}} \nabla F(x(t)) \cdot y$
 - $\frac{d\vec{x}}{dt} = y(t)$
- Let x_{opt} be the vertex of $\mathcal{P}(\mathcal{M})$ maximizing $F(x)$.
 - $F(x_{opt}) = f(x_{opt}) = OPT$

$$\begin{aligned} \frac{dF(x(t))}{dt} &= \nabla F(x(t)) \cdot \frac{d\vec{x}}{dt} \\ &= \nabla F(x(t)) \cdot y(t) \\ &\geq \nabla F(x(t)) \cdot [x_{opt} - x(t)]^+ \\ &= \nabla F(x(t)) \cdot [\max(x_{opt}, x(t)) - x(t)] \\ &\geq OPT - F(x(t)) \end{aligned}$$

Theorem

Let F be nondecreasing and up-concave, and \mathcal{P} be a downwards closed polytope. In the limit as $\epsilon \rightarrow 0$, the continuous greedy algorithm outputs a $1 - 1/e$ approximation to maximizing $F(x)$ over \mathcal{P} .

Proof Sketch

- Denote $y(t) = \operatorname{argmax}_{y \in \mathcal{P}} \nabla F(x(t)) \cdot y$
 - $\frac{d\vec{x}}{dt} = y(t)$
- Let x_{opt} be the vertex of $\mathcal{P}(\mathcal{M})$ maximizing $F(x)$.
 - $F(x_{opt}) = f(x_{opt}) = OPT$

$$\begin{aligned} \frac{dF(x(t))}{dt} &= \nabla F(x(t)) \cdot \frac{d\vec{x}}{dt} \\ &= \nabla F(x(t)) \cdot y(t) \\ &\geq \nabla F(x(t)) \cdot [x_{opt} - x(t)]^+ \\ &= \nabla F(x(t)) \cdot [\max(x_{opt}, x(t)) - x(t)] \\ &\geq F(\max(x_{opt}, x(t))) - F(x(t)) \\ &\geq OPT - F(x(t)) \end{aligned}$$

Theorem

Let F be nondecreasing and up-concave, and \mathcal{P} be a downwards closed polytope. In the limit as $\epsilon \rightarrow 0$, the continuous greedy algorithm outputs a $1 - 1/e$ approximation to maximizing $F(x)$ over \mathcal{P} .

Proof Sketch

- $v(t) = F(x(t))$ satisfies $\frac{dv}{dt} \geq OPT - v$.
- Differential equation $\frac{dv}{dt} = OPT - v$ with boundary condition $v(0) = 0$ has a unique solution

$$v(t) = OPT(1 - e^{-t})$$

- $v(1) \geq OPT(1 - 1/e)$

Continuous Greedy Algorithm $(F, \mathcal{P}, \epsilon)$

- 1 $x(0) \leftarrow \vec{0}$
 - 2 For $t \in [0, \epsilon, 2\epsilon, \dots, 1 - \epsilon]$
 - $x(t + \epsilon) \leftarrow x(t) + \epsilon \operatorname{argmax}_{y \in \mathcal{P}} \{\nabla F(x(t)) \cdot y\}$
 - 3 Return $x(1)$
-
- $\nabla F(x)$ is not readily available, but can be estimated “accurately enough” using $\operatorname{poly}(n)$ random samples from $D(x)$, w.h.p.
 - Step 2 can be implemented because \mathcal{P} is solvable
 - Discretization: Taking $\epsilon = 1/O(n^2)$ is “fine enough”
 - Both the above introduce error into the approximation guarantee, yielding $1 - 1/e - 1/O(n)$ w.h.p
 - This can be shaved off to $1 - 1/e$ with some additional “tricks”.

- The following algorithm takes x in matroid base polytope $\mathcal{P}_{base}(\mathcal{M})$, and non-decreasing cross-convex function F , and outputs integral y with $F(y) \geq F(x)$

PipageRounding (\mathcal{M}, x, F)

While x contains a fractional entry

- 1 Let T be a minimum-size **tight set** containing a fractional entry
 - i.e. $x(T) = rank_{\mathcal{M}}(T)$, $i \in T$ for some i with $x_i \in (0, 1)$, and $|T|$ is as small as possible.
- 2 Let $j \in T$ be such that $j \neq i$ and x_j is fractional.
- 3 Let $x(\mu) = x + \mu(e_i - e_j)$, and maximize $F(x(\mu))$ subject to $x(\mu) \in \mathcal{P}(\mathcal{M})$.
- 4 $x \leftarrow x(\mu)$.

- The following algorithm takes x in matroid base polytope $\mathcal{P}_{base}(\mathcal{M})$, and non-decreasing cross-convex function F , and outputs integral y with $F(y) \geq F(x)$

PipageRounding (\mathcal{M}, x, F)

While x contains a fractional entry

- 1 Let T be a minimum-size **tight set** containing a fractional entry
 - i.e. $x(T) = rank_{\mathcal{M}}(T)$, $i \in T$ for some i with $x_i \in (0, 1)$, and $|T|$ is as small as possible.
- 2 Let $j \in T$ be such that $j \neq i$ and x_j is fractional.
- 3 Let $x(\mu) = x + \mu(e_i - e_j)$, and maximize $F(x(\mu))$ subject to $x(\mu) \in \mathcal{P}(\mathcal{M})$.
- 4 $x \leftarrow x(\mu)$.

Theorem

On input $x \in \mathcal{P}_{base}(\mathcal{M})$, Pipage rounding terminates in $O(n^2)$ iterations, and outputs a matroid vertex y with $f(y) = F(y) \geq F(x)$.

PipageRounding (\mathcal{M}, x, F)

While x contains a fractional entry

- 1 Let T be a minimum-size **tight set** containing a fractional entry
 - i.e. $x(T) = \text{rank}_{\mathcal{M}}(T)$, $i \in T$ for some i with $x_i \in (0, 1)$, and $|T|$ is as small as possible.
- 2 Let $j \in T$ be such that $j \neq i$ and x_j is fractional.
- 3 Let $x(\mu) = x + \mu(e_i - e_j)$, and maximize $F(x(\mu))$ subject to $x(\mu) \in \mathcal{P}(\mathcal{M})$.
- 4 $x \leftarrow x(\mu)$.

Step 1

- T is a subset of every other tight set containing i , because tight sets form a **lattice**
 - A lattice is a family of sets closed under intersection and union.
- Proof:
 - Tight sets are the minimizers of the set function $\text{rank}_{\mathcal{M}}(S) - x(S)$
 - This set function is submodular.
 - Minimizers of a submodular function form a lattice (implied by **submodular inequality**)

PipageRounding (\mathcal{M}, x, F)

While x contains a fractional entry

- 1 Let T be a minimum-size **tight set** containing a fractional entry
 - i.e. $x(T) = \text{rank}_{\mathcal{M}}(T)$, $i \in T$ for some i with $x_i \in (0, 1)$, and $|T|$ is as small as possible.
- 2 Let $j \in T$ be such that $j \neq i$ and x_j is fractional.
- 3 Let $x(\mu) = x + \mu(e_i - e_j)$, and maximize $F(x(\mu))$ subject to $x(\mu) \in \mathcal{P}(\mathcal{M})$.
- 4 $x \leftarrow x(\mu)$.

Step 2

- Since rank is integer valued, any tight set containing fractional variable should have another.

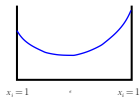
PipageRounding (\mathcal{M}, x, F)

While x contains a fractional entry

- 1 Let T be a minimum-size **tight set** containing a fractional entry
 - i.e. $x(T) = \text{rank}_{\mathcal{M}}(T)$, $i \in T$ for some i with $x_i \in (0, 1)$, and $|T|$ is as small as possible.
- 2 Let $j \in T$ be such that $j \neq i$ and x_j is fractional.
- 3 Let $x(\mu) = x + \mu(e_i - e_j)$, and maximize $F(x(\mu))$ subject to $x(\mu) \in \mathcal{P}(\mathcal{M})$.
- 4 $x \leftarrow x(\mu)$.

Step 3+4

- Either the number of fractional variables decreases, or a smaller tight set containing x_i or x_j is created.
 - Why smaller? T remains tight, and if R is a new tight set then by lattice property so is $T \cap R$
- Therefore this terminates in $O(n^2)$ iterations
- $F(x)$ does not decrease by definition



To summarize

Theorem

Let F be nondecreasing and up-concave, and \mathcal{P} be a downwards closed polytope. In the limit as $\epsilon \rightarrow 0$, the continuous greedy algorithm outputs a $1 - 1/e$ approximation to maximizing $F(x)$ over \mathcal{P} .

Theorem

On input x , Pipage rounding terminates in $O(n^2)$ iterations, and outputs a matroid vertex y with $f(y) = F(y) \geq F(x)$

To summarize

Theorem

Let F be nondecreasing and up-concave, and \mathcal{P} be a downwards closed polytope. In the limit as $\epsilon \rightarrow 0$, the continuous greedy algorithm outputs a $1 - 1/e$ approximation to maximizing $F(x)$ over \mathcal{P} .

Theorem

On input x , Pipage rounding terminates in $O(n^2)$ iterations, and outputs a matroid vertex y with $f(y) = F(y) \geq F(x)$

- Efficient implementation of continuous greedy algorithm follows from matroid optimization and basic sampling/concentration bounds
- Efficient implementation of Pipage rounding will be on HW5

Together, these imply a polynomial-time $1 - 1/e$ approximation algorithm for monotone submodular maximization subject to a matroid constraint