CS675: Convex and Combinatorial Optimization
Fall 2023
Submodular Function Optimization

Instructor: Shaddin Dughmi

Outline

Introduction

- 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
  - $f : 2^X \to \mathbb{R}$
- 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
  - $f : 2^X \to \mathbb{R}$
- 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
  - There is a weight w_i for each $i \in X$, and a constant c, such that $f(S) = c + \sum_{i \in S} w_i$ for all sets $S \subseteq X$.
  - Discrete analogue of affine functions

Set Functions

- We have already seen modular set functions
  - There is a weight w_i for each $i \in X$, and a constant c, such that $f(S) = c + \sum_{i \in S} w_i$ for all sets $S \subseteq X$.
  - Discrete analogue of affine 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
    - There is a weight w_i for each $i \in X$, and a constant c, such that $f(S) = c + \sum_{i \in S} w_i$ for all sets $S \subseteq X$.
    - Discrete analogue of affine functions
    - Direct definition of modularity: $f(A) + f(B) = f(A \cap B) + f(A \cup B)$
- Submodular/supermodular functions are weak analogues to convex/concave functions (in no particular order!)

Set Functions

- We have already seen modular set functions
  - There is a weight w_i for each $i \in X$, and a constant c, such that $f(S) = c + \sum_{i \in S} w_i$ for all sets $S \subseteq X$.
  - Discrete analogue of affine functions
  - Direct definition of modularity: $f(A) + f(B) = f(A \cap B) + f(A \cup B)$
- Submodular/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

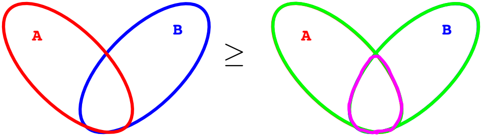
Definition 1

A set function $f : 2^X \to \mathbb{R}$ 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

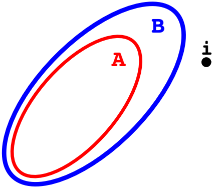
Definition 2

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

Supermodular Functions

Definition 0

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

Definition 1

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

Definition 1

A set function $f : 2^X \to \mathbb{R}$ 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 $f : 2^X \to \mathbb{R}$ 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 $i \notin B$.

Examples

Many common examples are monotone, normalized, and submodular.

Coverage Functions

- In general: X is a family of sets, and $f(S)$ is the "size" (cardinality or measure) of $\bigcup_{A \in S} A$
- Discrete special case: X the left hand side of a bipartite graph, and $f(S)$ is the total number of neighbors of S.

Examples

Many common examples are monotone, normalized, and submodular.

Coverage Functions

- In general: X is a family of sets, and $f(S)$ is the “size” (cardinality or measure) of $\bigcup_{A \in S} A$
- Discrete special case: X the left hand side of a bipartite graph, and $f(S)$ is the total number of neighbors of S.

The following two are examples of coverage functions

Probability

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

Sensor Coverage

X is a family of locations in space you can place sensors, and $f(S)$ is the total area covered if you place sensors at locations $S \subseteq X$.

Examples

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.

Examples

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.

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.

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 nontheless 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 nontheless 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 nontheless 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

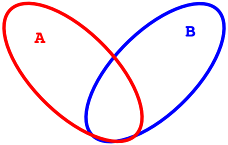
X is the set of nodes in a graph G, and $f(S) = \frac{E(S)}{|S|}$ where $E(S)$ is the number of edges with both endpoints in S.

- 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

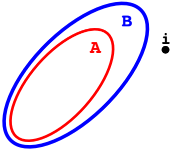
Definition 1

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



Definition 2

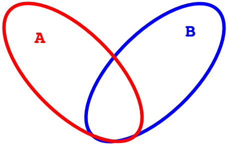
$$f(B \cup \{i\}) - f(B) \leq f(A \cup \{i\}) - f(A))$$



Equivalence of Both Definitions

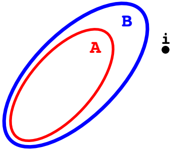
Definition 1

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



Definition 2

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



Definition 1 $\Rightarrow$ Definition 2

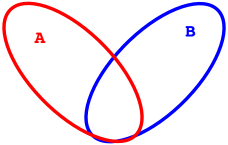
- To prove (2), let $A' = A\bigcup\{i\}$ and $B' = B$ and apply (1)

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

Equivalence of Both Definitions

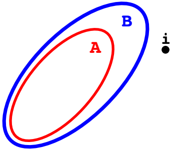
Definition 1

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



Definition 2

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



Definition 2 ⇒ Definition 1

- To prove (1), start with $A = B = A \bigcap B$ and repeatedly add 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 $f_1, \ldots, f_k$ are submodular, and $w_1, \ldots, w_k \geq 0$, 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 $f_1, \ldots, f_k$ are submodular, and $w_1, \ldots, w_k \geq 0$, 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 $f_1, \ldots, f_k$ are submodular, and $w_1, \ldots, w_k \geq 0$, 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 $f_T(S) = f(S \cup T) - f(T)$ is submodular

Operations Preserving Submodularity

- Nonnegative-weighted combinations (a.k.a. conic combinations): If $f_1, \ldots, f_k$ are submodular, and $w_1, \ldots, w_k \geq 0$, 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 $f_T(S) = f(S \cup T) - f(T)$ is submodular
- Reflection: If f is a submodular function on X, then $\overline{f}(S) = f(X \setminus S)$ is also submodular

Operations Preserving Submodularity

- Nonnegative-weighted combinations (a.k.a. conic combinations): If $f_1, \ldots, f_k$ are submodular, and $w_1, \ldots, w_k \geq 0$, 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 $f_T(S) = f(S \cup T) - f(T)$ is submodular
- Reflection: If f is a submodular function on X, then $\overline{f}(S) = f(X \setminus S)$ is also submodular
- Others: Dilworth trucation, convolution with modular functions, . . .

Operations Preserving Submodularity

- Nonnegative-weighted combinations (a.k.a. conic combinations): If $f_1, \ldots, f_k$ are submodular, and $w_1, \ldots, w_k \geq 0$, 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 $f_T(S) = f(S \cup T) - f(T)$ is submodular
- Reflection: If f is a submodular function on X, then $\overline{f}(S) = f(X \setminus S)$ is also submodular
- Others: Dilworth trucation, 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 $f(S)$.

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 submodular function $f : 2^X \to \mathbb{R}$ 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 $f : 2^X \to \mathbb{R}$ 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 $f : 2^X \to \mathbb{R}$ 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 $f : 2^X \to \mathbb{R}$ 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.

Examples

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.

Examples

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

Continuous Extensions of a Set Function

Recall

A set function f on $X = \{1, \ldots, n\}$ 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 $f : \{0,1\}^n \to \mathbb{R}$, an extension of f to the hypercube $[0,1]^n$ is a function $g : [0,1]^n \to \mathbb{R}$ 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 $f : \{0,1\}^n \to \mathbb{R}$, the convex closure $f^- : [0,1]^n \to \mathbb{R}$ of f is the point-wise greatest convex function under-estimating f on $\{0,1\}^n$.

The Convex Closure

Convex Closure

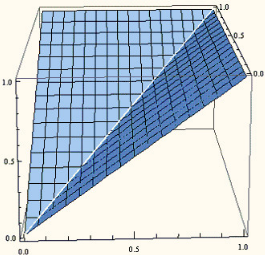
Given a set function $f : \{0,1\}^n \rightarrow \mathbb{R}$, the convex closure $f^- : [0,1]^n \rightarrow \mathbb{R}$ of f is the point-wise greatest convex function under-estimating f on $\{0,1\}^n$.

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 $f : \{0,1\}^n \to \mathbb{R}$, the convex closure $f^- : [0,1]^n \to \mathbb{R}$ 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 $g_1, g_2 : [0,1]^n \to \mathbb{R}$ 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:

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

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:

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

Implications

- 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

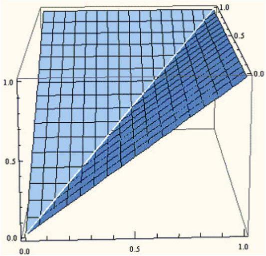
The value of the convex closure f^- at $x \in [0,1]^n$ is the solution of the following optimization problem:

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

$f(\emptyset) = 0$
$f(\{1\}) = f(\{2\}) = 1$
$f(\{1,2\}) = 1$

When $x_1 \leq x_2$

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



Claim

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

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

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:

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

Proof

- $OPT(x)$ is at least $f^-(x)$ for every x: By Jensen's inequality
- To show that $OPT(x)$ is at most $f^-(x)$, 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:

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

Proof

- $OPT(x)$ is at least $f^-(x)$ for every x: By Jensen's inequality
- To show that $OPT(x)$ is at most $f^-(x)$, 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:

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

Proof

- $OPT(x)$ is at least $f^-(x)$ for every x: By Jensen's inequality
- To show that $OPT(x)$ is at most $f^-(x)$, 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 indispensible 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

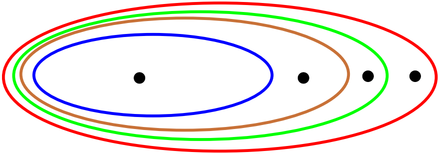
In general, it is hard to evaluate f^- efficiently, let alone its derivative. This is indispensible for convex optimization algorithms.

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

Chain Distributions

Chain Distribution

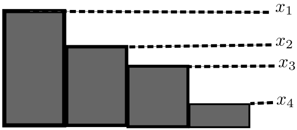
A chain distribution on the ground set X is a distribution over $S \subseteq X$ who's support forms a chain in the inclusion order.

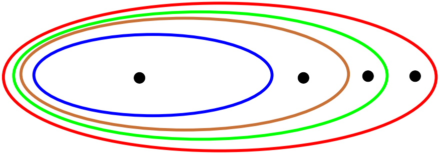


Chain Distributions

Chain Distribution with Given Marginals

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

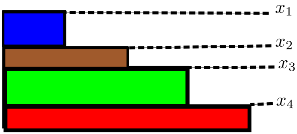


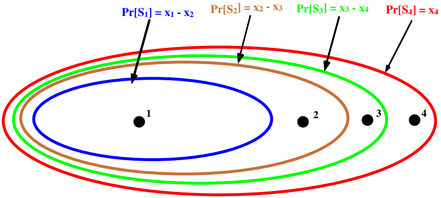


Chain Distributions

Chain Distribution with Given Marginals

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

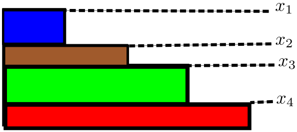


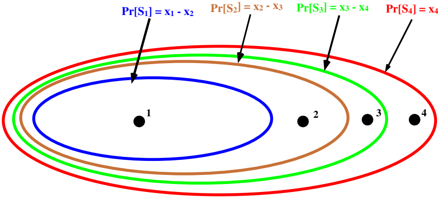


Chain Distributions

Chain Distribution with Given Marginals

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





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

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

The Lovasz Extension

Definition

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

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

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.

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

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. Since $f^{-}(x)$ chooses the "lowest" such distribution, we have $f^{\mathcal{L}}(x) \geq f^{-}(x)$.

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)

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

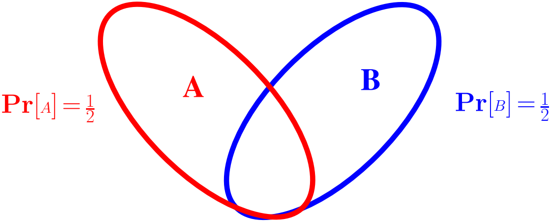
It suffices to show that the chain distribution with marginals x is in fact the “lowest” distribution with marginals x.

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)

- Take a distribution $\mathcal{D}$ on two "crossing" sets A and B, with probability 0.5 each.

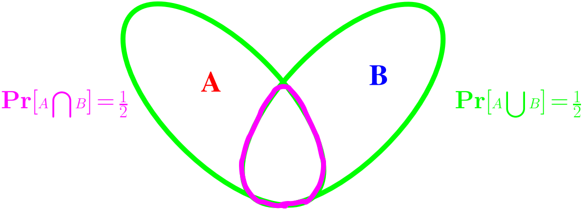


$\frac{1}{2} f(A) + \frac{1}{2} f(B)$

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

Proof (Special case)

- Take a distribution $\mathcal{D}$ on two “crossing” sets A and B, with probability 0.5 each.
- Consider “uncrossing” A and B, replacing them with $A \bigcap B$ and $A \bigcup B$, with probability 0.5 each.
  - Yields a chain distribution supported on $A \bigcap B$ and $A \bigcup B$.
  - Marginals don’t change
  - By submodularity, expected value can only go down.

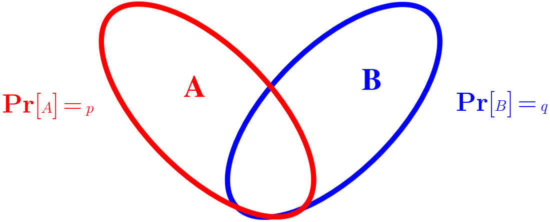


$$\frac{1}{2}f(A) + \frac{1}{2}f(B) \geq \frac{1}{2}f(A \bigcap B) + \frac{1}{2}f(A \bigcup B)$$

Proof (Slightly Less Special Case)

Proof (Slightly Less Special Case)

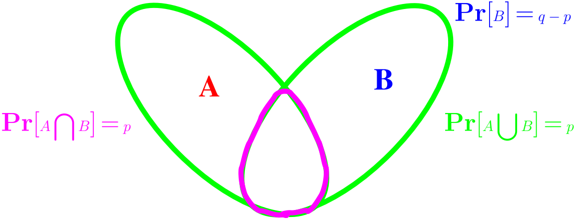
- Take a distribution $\mathcal{D}$ on two "crossing" sets A and B, with probabilities $p \leq q$.



$pf(A) + qf(B)$

Proof (Slightly Less Special Case)

- Take a distribution $\mathcal{D}$ on two "crossing" sets A and B, with probabilities $p \leq q$.
- Consider "uncrossing" a probability mass of p from each of A,B.
  - Yields a chain distribution supported on $A \bigcap B$, B, and $A \bigcup B$.
  - Marginals don't change
  - By submodularity, expected value can only go down.

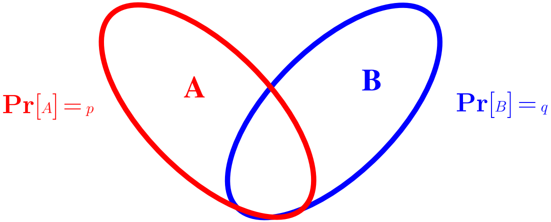


$$pf(A) + qf(B) \geq pf(A \bigcap B) + pf(A \bigcup B) + (q-p)f(B)$$

Proof (General Case)

Proof (General Case)

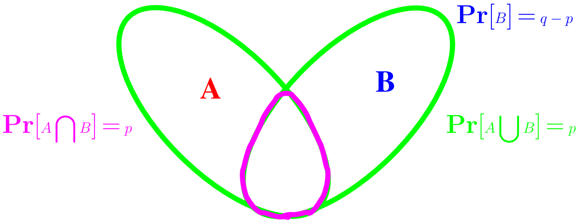
- Take a distribution $\mathcal{D}$ which includes two “crossing” sets A and B in its support, with probabilities $p \leq q$.



$pf(A) + qf(B)$

Proof (General Case)

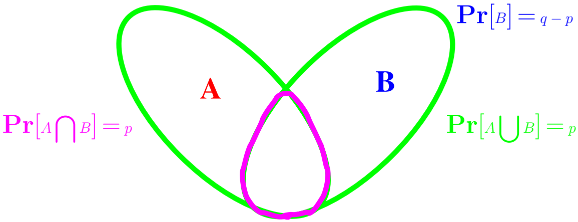
- Take a distribution $\mathcal{D}$ which includes two "crossing" sets A and B in its support, with probabilities $p \leq q$.
- Consider "uncrossing" a probability mass of p from each of A, B.
  - Marginals don't change
  - By submodularity, expected value can only go down.



$$pf(A) + qf(B) \geq pf\left(A \bigcap B\right) + pf\left(A \bigcup B\right) + (q-p)f(B)$$

Proof (General Case)

- Take a distribution $\mathcal{D}$ which includes two “crossing” sets A and B in its support, with probabilities $p \leq q$.
- Consider “uncrossing” a probability mass of p from each of A, B.
  - Marginals don’t change
  - By submodularity, expected value can only go down.
- Makes $\mathcal{D}$ “closer” to being a chain distribution
  - The bounded potential function $\mathbf{E}_{S \sim \mathcal{D}}[|S|^2]$ increases



$$pf(A) + qf(B) \geq pf\left(A \bigcap B\right) + pf\left(A \bigcup B\right) + (q-p)f(B)$$

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

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

Recall: Solvability of Convex Optimization

Weak Solvability

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

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

Recall: Solvability of Convex Optimization

Polynomial Solvability of CP

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

1. Compute a starting ellipsoid $E \supseteq [0,1]^n$ with $\frac{\mathbf{vol}(E)}{\mathbf{vol}([0,1]^n)} = O(\exp(n))$.
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.

Recall: Solvability of Convex Optimization

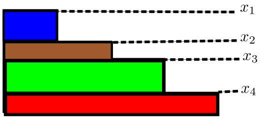
Polynomial Solvability of CP

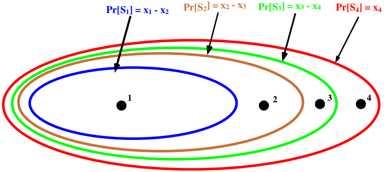
In order to weakly minimize $f^{\mathcal{L}}$, we need the following operations to run in $\text{poly}(n)$ time:

1. Compute a starting ellipsoid $E \supseteq [0,1]^n$ with $\frac{\textbf{vol}(E)}{\textbf{vol}([0,1]^n)} = O(\exp(n))$.
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.

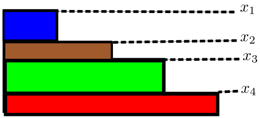
First order Oracle for $f^{\mathcal{L}}$

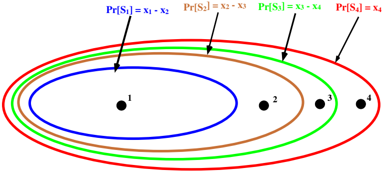




- Recall: the chain distribution with marginals x
  - Sort $x_1 \geq x_2 \ldots \geq x_n$
  - Let $S_i = \{x_1, \ldots, x_i\}$
  - Let $\mathbf{Pr}[S_i] = x_i - x_{i+1}$

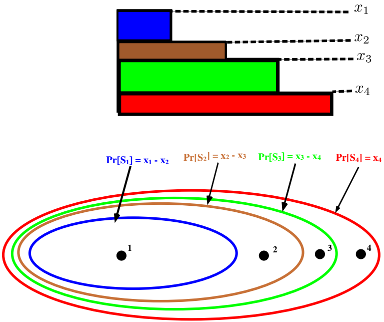
First order Oracle for $f^{\mathcal{L}}$





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

Recovering an Optimal Set

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

Recovering an Optimal Set

We can get an ϵ-optimal solution x^* to the optimization problem in $\mathrm{poly}(n, \log \frac{1}{\epsilon})$ 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 $\mathrm{poly}(n, b)$.
- $\min f(S) \leq f^{\mathcal{L}}(x^*) < \min 2\, f(S)$

Recovering an Optimal Set

We can get an ϵ-optimal solution x^* to the optimization problem in $\mathrm{poly}(n, \log \frac{1}{\epsilon})$ 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 $\mathrm{poly}(n, b)$.
- $\min f(S) \leq f^{\mathcal{L}}(x^*) < \min 2\, f(S)$
- $f^{\mathcal{L}}(x^*)$ 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 $\text{poly}(n, b)$.
- $\min f(S) \leq f^{\mathcal{L}}(x^*) < \text{min2}\, f(S)$
- $f^{\mathcal{L}}(x^*)$ 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 $f : 2^X \to \mathbb{R}_+$ 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 $f : 2^X \to \mathbb{R}_+$ 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 $f : 2^X \to \mathbb{R}_+$ 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.

Examples

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 $(B_1, \ldots, B_n)$ 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 $(B_1, \ldots, B_n)$ 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

Complexity

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)

Complexity

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

Complexity

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 $\mathrm{poly}(n)$
- 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.

Subject to a Cardinality Constraint

Problem Definition

Given a non-decreasing and normalized submodular function $f : 2^X \to \mathbb{R}_+$ 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 $|S| < k$
     - Choose $e \in X$ maximizing $f(S \bigcup \{e\})$
     - $S \leftarrow S \bigcup \{e\}$
```

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 $|S| < k$
    • Choose $e \in X$ maximizing $f(S \bigcup \{e\})$
    • $S \leftarrow S \bigcup \{e\}$
```

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

Lemma

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

Contraction/Conditioning

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

Lemma

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

Proof

- Normalized: trivial

Contraction/Conditioning

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

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

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 $f : 2^X \to \mathbb{R}_+$ 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 $f : 2^X \to \mathbb{R}_+$ 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 $f : 2^X \to \mathbb{R}_+$ 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 continous extension of f
  - Optimize continuous extension over matroid polytope
  - Extract an integer point

The Multilinear Extension

Multilinear Extension

Given a set function $f : \{0,1\}^n \to \mathbb{R}$, its multilinear extension $F : [0,1]^n \to \mathbb{R}$ 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.

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

The Multilinear Extension

Multilinear Extension

Given a set function $f : \{0,1\}^n \to \mathbb{R}$, its multilinear extension $F : [0,1]^n \to \mathbb{R}$ 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.

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

- 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.

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

- 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 $f : \{0,1\}^n \to \mathbb{R}$, its multilinear extension $F : [0,1]^n \to \mathbb{R}$ 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.

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

- 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: $\frac{d^2 g}{dt^2} = d^T (\bigtriangledown^2 F) d$
- The Hessian $\bigtriangledown^2 F$ 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 $\frac{d^2 g}{dt^2} = d^T (\bigtriangledown^2 F) d \leq 0$ for $\vec{d} \succeq 0$

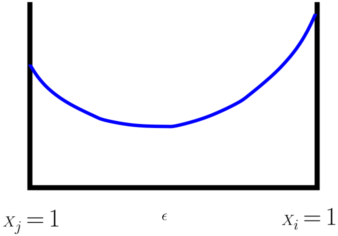
Cross-convexity

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

- $\frac{d^2 g}{dt^2} = d^T(\bigtriangledown^2 F)d = \frac{\partial^2 F}{\partial x_i^2} + \frac{\partial^2 F}{\partial x_j^2} - 2\frac{\partial^2 F}{\partial x_i \partial x_j}$
- 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 $\frac{\partial^2 F}{\partial x_i \partial x_j} \leq 0$

Algorithm Outline

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

Algorithm Outline

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: $\max \{F(x) : x \in \mathcal{P}(\mathcal{M})\} \geq \max \{f(S) : S \in \mathcal{I}\}$

Algorithm Outline

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?

Algorithm Outline

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 largely supported on infeasible sets (i.e. not independent in matroid $\mathcal{M}$).

Algorithm Outline

Step B: Pipage Rounding

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

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

Algorithm Outline

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

Step A: Continuous Greedy Algorithm

- Feasible polytope $\mathcal{P} \subseteq [0,1]^n$
  - Downwards Closed: If $y \in \mathcal{P}$ and $\vec{0} \preceq x \preceq y$ then $x \in \mathcal{P}$ also.
- Objective function $F : [0,1]^n \to \mathbb{R}_+$ which is non-decreasing, up-concave, and normalized ($F(\vec{0}) = 0$).

Step A: Continuous Greedy Algorithm

- Feasible polytope $\mathcal{P} \subseteq [0,1]^n$
  - Downwards Closed: If $y \in \mathcal{P}$ and $\vec{0} \preceq x \preceq y$ then $x \in \mathcal{P}$ also.
- Objective function $F : [0,1]^n \to \mathbb{R}_+$ which is non-decreasing, up-concave, and normalized ($F(\vec{0}) = 0$).
- Continuously moves a particle inside the polytope, starting at $\vec{0}$, for a total of 1 time unit.
  - Position at time t given by $x(t)$.

Step A: Continuous Greedy Algorithm

- Feasible polytope $\mathcal{P} \subseteq [0,1]^n$
  - Downwards Closed: If $y \in \mathcal{P}$ and $\vec{0} \preceq x \preceq y$ then $x \in \mathcal{P}$ also.
- Objective function $F : [0,1]^n \to \mathbb{R}_+$ which is non-decreasing, up-concave, and normalized ($F(\vec{0}) = 0$).
- Continuously moves a particle inside the polytope, starting at $\vec{0}$, 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 $1/\operatorname{poly}(n)$ in the actual implementation.

Step A: Continuous Greedy Algorithm

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

```
1. $x(0) \leftarrow \vec{0}$
2. For $t \in [0, \epsilon, 2\epsilon, \ldots, 1 - \epsilon]$
       Let $y(t) \in \mathrm{argmax}_{y \in \mathcal{P}} \{\bigtriangledown F(x(t)) \cdot y\}$
       $x(t + \epsilon) \leftarrow x(t) + \epsilon y(t)$
3. Return $x(1)$
```

Step A: Continuous Greedy Algorithm

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

```
1. $x(0) \leftarrow \vec{0}$
2. For $t \in [0, \epsilon, 2\epsilon, \ldots, 1-\epsilon]$
    - Let $y(t) \in \text{argmax}_{y \in \mathcal{P}} \{\triangledown F(x(t)) \cdot y\}$
    - $x(t+\epsilon) \leftarrow x(t) + \epsilon y(t)$
3. Return $x(1)$
```

- I.e. When the particle is at x, it moves in direction y maximizing the linear function $\triangledown F(x) \cdot y$ over $y \in \mathcal{P}$
    - The direction is actually a vertex of our matroid polytope
    - This is NOT gradient ascent

Step A: Continuous Greedy Algorithm

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

```
1. $x(0) \leftarrow \vec{0}$
2. For $t \in [0, \epsilon, 2\epsilon, \ldots, 1 - \epsilon]$
       Let $y(t) \in \text{argmax}_{y \in \mathcal{P}} \{\bigtriangledown F(x(t)) \cdot y\}$
       $x(t + \epsilon) \leftarrow x(t) + \epsilon y(t)$
3. Return $x(1)$
```

- I.e. When the particle is at x, it moves in direction y maximizing the linear function $\bigtriangledown F(x) \cdot y$ over $y \in \mathcal{P}$
    - The direction is actually a vertex of our matroid polytope
    - This is NOT gradient ascent
- 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

In the limit as $\epsilon \to 0$, the continuous greedy algorithm outputs a $1 - 1/e$ approximation to maximizing $F(x)$ over $\mathcal{P}$.

Theorem

In the limit as $\epsilon \to 0$, the continuous greedy algorithm outputs a $1 - 1/e$ approximation to maximizing $F(x)$ over $\mathcal{P}$.

Proof Sketch

- $\frac{d\vec{x}}{dt} = y(t)$

Theorem

In the limit as $\epsilon \to 0$, the continuous greedy algorithm outputs a $1 - 1/e$ approximation to maximizing $F(x)$ over $\mathcal{P}$.

Proof Sketch

- $\frac{d\vec{x}}{dt} = y(t)$
- Let x^* be the point in $\mathcal{P}$ maximizing $F(x)$, and $OPT = F(x^*)$.

Theorem

In the limit as $\epsilon \to 0$, the continuous greedy algorithm outputs a $1 - 1/e$ approximation to maximizing $F(x)$ over $\mathcal{P}$.

Proof Sketch

- $\frac{d\vec{x}}{dt} = y(t)$
- Let x^* be the point in $\mathcal{P}$ maximizing $F(x)$, and $OPT = F(x^*)$.

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

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

Theorem

In the limit as $\epsilon \to 0$, the continuous greedy algorithm outputs a $1 - 1/e$ approximation to maximizing $F(x)$ over $\mathcal{P}$.

Proof Sketch

- $\frac{d\vec{x}}{dt} = y(t)$
- Let x^* be the point in $\mathcal{P}$ maximizing $F(x)$, and $OPT = F(x^*)$.

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

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

Theorem

In the limit as $\epsilon \to 0$, the continuous greedy algorithm outputs a $1 - 1/e$ approximation to maximizing $F(x)$ over $\mathcal{P}$.

Proof Sketch

- $\frac{d\vec{x}}{dt} = y(t)$
- Let x^* be the point in $\mathcal{P}$ maximizing $F(x)$, and $OPT = F(x^*)$.

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

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

Theorem

In the limit as $\epsilon \to 0$, the continuous greedy algorithm outputs a $1 - 1/e$ approximation to maximizing $F(x)$ over $\mathcal{P}$.

Proof Sketch

- $\frac{d\vec{x}}{dt} = y(t)$
- Let x^* be the point in $\mathcal{P}$ maximizing $F(x)$, and $OPT = F(x^*)$.

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

Theorem

In the limit as $\epsilon \to 0$, the continuous greedy algorithm outputs a $1 - 1/e$ approximation to maximizing $F(x)$ over $\mathcal{P}$.

Proof Sketch

- $\frac{d\vec{x}}{dt} = y(t)$
- Let x^* be the point in $\mathcal{P}$ maximizing $F(x)$, and $OPT = F(x^*)$.

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

Theorem

In the limit as $\epsilon \to 0$, the continuous greedy algorithm outputs a $1 - 1/e$ approximation to maximizing $F(x)$ over $\mathcal{P}$.

Proof Sketch

- $\frac{d\vec{x}}{dt} = y(t)$
- Let x^* be the point in $\mathcal{P}$ maximizing $F(x)$, and $OPT = F(x^*)$.

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

Theorem

In the limit as $\epsilon \to 0$, 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

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

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

Implementation Details

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

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

When F is multilinear extension of submodular f, and $\mathcal{P} = \mathcal{P}(\mathcal{M})$ for matroid $\mathcal{M}$.

- $\bigtriangledown F(x)$ is not readily available, but can be estimated "accurately enough" using $\mathrm{poly}(n)$ 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. $x(T) = rank_{\mathcal{M}}(T)$, $i \in T$ 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 $rank_{\mathcal{M}}(S) - x(S)$
    - This set function is submodular.
    - Minimizers of a submodular function form a lattice.

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. $x \leftarrow x(\mu)$.

Step 2

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

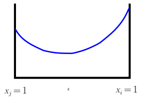
PipageRounding ($\mathcal{M}$,x, F)

While x contains a fractional entry

1. Let T be a minimum-size tight set containing a fractional entry
    - i.e. $x(T) = rank_{\mathcal{M}}(T)$, $i \in T$ 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 $T \bigcap R$
- Therefore this terminates in $O(n^2)$ iterations
- $F(x)$ does not decrease by definition of step 3



To summarize

Theorem

In the limit as $\epsilon \to 0$, 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

In the limit as $\epsilon \to 0$, 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 concentration bounds
- Efficient implementation of each iteration of Pipage rounding will be on HW

To summarize

Theorem

In the limit as $\epsilon \to 0$, 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 concentration bounds
- Efficient implementation of each iteration of Pipage rounding will be on HW

Theorem

The continuous greedy algorithm followed by Pipage rounding gives a $(1 - 1/e)$ approximation algorithm for maximizing a monotone, normalized, and submodular function subject to a matroid constraint.