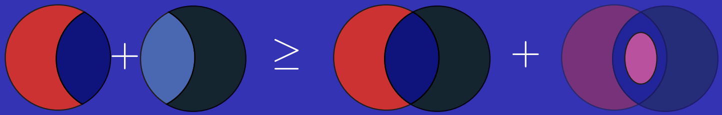


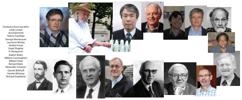
Mathematical Properties of Submodularity and Applications to Machine Learning

Jeffrey A. Bilmes

Professor
Departments of Electrical Engineering
& Computer Science and Engineering
University of Washington, Seattle
http://melodi.ee.washington.edu/~bilmes

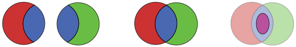
Friday, May 2nd, 2014

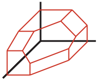
Goals of the Tutorial



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

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





- Get an intuitive sense for submodular functions, should be able to apply them.
- Learn to recognize submodularity, or recognize when it might be useful.
- Learn to realize why submodularity can be useful in machine learning. Why is it worth your time to study it.

Submodularity

- Definition: given a finite ground set V, a function $f : 2^V \to \mathbb{R}$ is said to be submodular if

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B), \quad \forall A, B \subseteq V \tag{1}$$

Submodularity

- Definition: given a finite ground set V, a function $f : 2^V \to \mathbb{R}$ is said to be submodular if

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B), \quad \forall A, B \subseteq V \tag{1}$$

- The definition is the tip of the iceberg. This simple definition can lead to great mathematical and practical richness.

Submodularity

- Definition: given a finite ground set V, a function $f : 2^V \to \mathbb{R}$ is said to be submodular if

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B), \quad \forall A, B \subseteq V \qquad (1)$$

- The definition is the tip of the iceberg. This simple definition can lead to great mathematical and practical richness.
- Goals of tutorial: will be very simple, an attempt to cover some important parts of the iceberg in 4.5 hours and in doing so give you all strong intuition and sense of applicability in ML.

Submodularity

- Definition: given a finite ground set V, a function $f : 2^V \to \mathbb{R}$ is said to be submodular if

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B), \quad \forall A, B \subseteq V \tag{1}$$

- The definition is the tip of the iceberg. This simple definition can lead to great mathematical and practical richness.
- Goals of tutorial: will be very simple, an attempt to cover some important parts of the iceberg in 4.5 hours and in doing so give you all strong intuition and sense of applicability in ML.
- The tutorial itself is the tip of the iceberg!

Overall Outline of Tutorial

1. Part 1 (now): basics and applications
2. Part 2 (later this afternoon): Theory (from matroids to polymatroids), and other submodular properties
3. Part 3 (tomorrow): Algorithms and optimization

Outline of Part 1: Basics and Applications

1. Introduction
2. Basics
3. Submodular Applications in ML
   - Where is submodularity useful?
   - Traditional combinatorial problems
   - As a model of diversity, coverage, span, or information
   - As a model of cooperative costs, complexity, roughness, and irregularity
   - As a parameter for an ML algorithm
   - Itself, as a target for learning
   - Surrogates for optimization
   - Economic applications

Outline of Part 2: Theory

4. From Matroids to Polymatroids
   - Matrix Rank
   - Venn Diagrams
   - Matroids

5. Submodular Definitions, Examples, and Properties
   - Normalization
   - Submodular Definitions
   - Submodular Composition
   - More Examples

Outline of Part 3: Algorithms

6 Discrete Semimodular Semigradients

7 Continuous Extensions
- Lovász Extension
- Concave Extension

8 Like Concave or Convex?

9 Optimization

10 Reading

Outline: Part 1

1. Introduction
2. Basics
3. Submodular Applications in ML
   - Where is submodularity useful?
   - Traditional combinatorial problems
   - As a model of diversity, coverage, span, or information
   - As a model of cooperative costs, complexity, roughness, and irregularity
   - As a parameter for an ML algorithm
   - Itself, as a target for learning
   - Surrogates for optimization
   - Economic applications

Sets and set functions

We are given a finite "ground" set of objects:

$V =$ 

Also given a set function $f : 2^V \to \mathbb{R}$ that valuates subsets $A \subseteq V$.

Ex: $f(V) = 6$

Sets and set functions

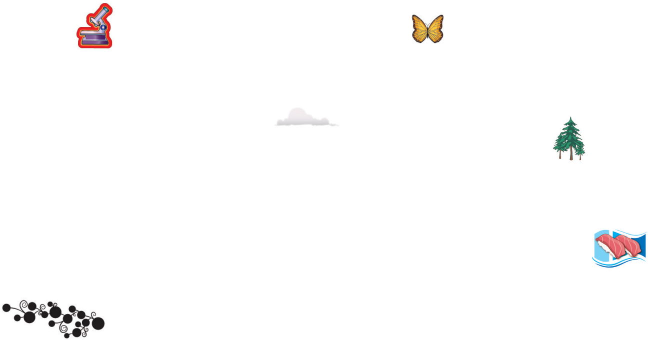
Subset $A \subseteq V$ of objects:

$A = \Bigg\{$  $\Bigg\}$

Also given a set function $f : 2^V \to \mathbb{R}$ that valuates subsets $A \subseteq V$.
Ex: $f(A) = 1$

Sets and set functions

Subset $B \subseteq V$ of objects:

$B =$ 

Also given a set function $f : 2^V \to \mathbb{R}$ that valuates subsets $A \subseteq V$.
Ex: $f(B) = 6$

Two Equivalent Submodular Definitions

Definition (submodular)

A function $f : 2^V \to \mathbb{R}$ is submodular if for any $A, B \subseteq V$, we have that:

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B) \quad (2)$$

An alternate and equivalent definition is:

Definition (submodular (diminishing returns))

A function $f : 2^V \to \mathbb{R}$ is submodular if for any $A \subseteq B \subset V$, and $v \in V \setminus B$, we have that:

$$f(A \cup \{v\}) - f(A) \geq f(B \cup \{v\}) - f(B) \quad (3)$$

This means that the incremental "value", "gain", or "cost" of v decreases (diminishes) as the context in which v is considered grows from A to B.

Two Equivalent Supermodular Definitions

Definition (submodular)

A function $f : 2^V \to \mathbb{R}$ is supermodular if for any $A, B \subseteq V$, we have that:

$$f(A) + f(B) \leq f(A \cup B) + f(A \cap B) \tag{4}$$

An alternate and equivalent definition is:

Definition (supermodular (improving returns))

A function $f : 2^V \to \mathbb{R}$ is supermodular if for any $A \subseteq B \subset V$, and $v \in V \setminus B$, we have that:

$$f(A \cup \{v\}) - f(A) \leq f(B \cup \{v\}) - f(B) \tag{5}$$

This means that the incremental "value", "gain", or "cost" of v increases (improves) as the context in which v is considered grows from A to B.

Sets and vectors

- Any set $A \subseteq V$ can be represented as a binary vector $x \in \{0, 1\}^V$.

Sets and vectors

- Any set $A \subseteq V$ can be represented as a binary vector $x \in \{0,1\}^V$.
- The characteristic vector of a set is given by $\mathbf{1}_A \in \{0,1\}^V$ where for all $v \in V$, we have:

$$\mathbf{1}_A(v) = \begin{cases} 1 & \text{if } v \in A \\ 0 & \textit{else} \end{cases} \tag{6}$$

Sets and vectors

- Any set $A \subseteq V$ can be represented as a binary vector $x \in \{0,1\}^V$.
- The characteristic vector of a set is given by $\mathbf{1}_A \in \{0,1\}^V$ where for all $v \in V$, we have:

$$\mathbf{1}_A(v) = \begin{cases} 1 & \text{if } v \in A \\ 0 & \textit{else} \end{cases} \tag{6}$$

- If $V = \{1,2,\dots,20\}$ and $A = \{1,3,5,\dots,19\}$, then $\mathbf{1}_A = (1,0,1,0,\dots)^\intercal$.

Sets and vectors

- Any set $A \subseteq V$ can be represented as a binary vector $x \in \{0,1\}^V$.
- The characteristic vector of a set is given by $\mathbf{1}_A \in \{0,1\}^V$ where for all $v \in V$, we have:

$$\mathbf{1}_A(v) = \begin{cases} 1 & \text{if } v \in A \\ 0 & \textit{else} \end{cases} \tag{6}$$

- If $V = \{1, 2, \ldots, 20\}$ and $A = \{1, 3, 5, \ldots, 19\}$, then $\mathbf{1}_A = (1, 0, 1, 0, \ldots)^\intercal$.
- It is sometimes useful to go back and forth. Given $X \subseteq V$ then $x(X) \triangleq \mathbf{1}_X$ and $X(x) = \{v \in V : x(v) = 1\}$.

Sets and vectors

- Any set $A \subseteq V$ can be represented as a binary vector $x \in \{0,1\}^V$.
- The characteristic vector of a set is given by $\mathbf{1}_A \in \{0,1\}^V$ where for all $v \in V$, we have:

$$\mathbf{1}_A(v) = \begin{cases} 1 & \text{if } v \in A \\ 0 & \textit{else} \end{cases} \tag{6}$$

- If $V = \{1, 2, \ldots, 20\}$ and $A = \{1, 3, 5, \ldots, 19\}$, then $\mathbf{1}_A = (1, 0, 1, 0, \ldots)^\intercal$.
- It is sometimes useful to go back and forth. Given $X \subseteq V$ then $x(X) \triangleq \mathbf{1}_X$ and $X(x) = \{v \in V : x(v) = 1\}$.
- $f(x) : \{0,1\}^V \to \mathbb{R}$ is a pseudo-Boolean function. A submodular function is a special case.

Modular functions, and vectors in $\mathbb{R}^V$

- Any set function $m : 2^V \to \mathbb{R}$ whose valuations, for $A \subseteq V$, take form

$$m(A) = \sum_{a \in A} m(a) \tag{7}$$

is called modular and normalized (meaning $m(\emptyset) = 0$).

Modular functions, and vectors in $\mathbb{R}^V$

- Any set function $m : 2^V \to \mathbb{R}$ whose valuations, for $A \subseteq V$, take form

$$m(A) = \sum_{a \in A} m(a) \tag{7}$$

  is called modular and normalized (meaning $m(\emptyset) = 0$).
- Any normalized modular function is identical to a vector:

$$m \in \mathbb{R}^V. \tag{8}$$

Modular functions, and vectors in $\mathbb{R}^V$

- Any set function $m : 2^V \to \mathbb{R}$ whose valuations, for $A \subseteq V$, take form

$$m(A) = \sum_{a \in A} m(a) \tag{7}$$

is called modular and normalized (meaning $m(\emptyset) = 0$).

- Any normalized modular function is identical to a vector:

$$m \in \mathbb{R}^V. \tag{8}$$

- Hence, the characteristic vector $\mathbf{1}_A$ of a set is modular.

Modular functions, and vectors in $\mathbb{R}^V$

- Any set function $m : 2^V \to \mathbb{R}$ whose valuations, for $A \subseteq V$, take form

$$m(A) = \sum_{a \in A} m(a) \tag{7}$$

  is called modular and normalized (meaning $m(\emptyset) = 0$).
- Any normalized modular function is identical to a vector:

$$m \in \mathbb{R}^V. \tag{8}$$

- Hence, the characteristic vector $\mathbf{1}_A$ of a set is modular.
- Modular functions are often called additive or linear.

Modular functions, and vectors in $\mathbb{R}^V$

- Any set function $m : 2^V \rightarrow \mathbb{R}$ whose valuations, for $A \subseteq V$, take form

$$m(A) = \sum_{a \in A} m(a) \tag{7}$$

  is called modular and normalized (meaning $m(\emptyset) = 0$).
- Any normalized modular function is identical to a vector:

$$m \in \mathbb{R}^V. \tag{8}$$

- Hence, the characteristic vector $\mathbf{1}_A$ of a set is modular.
- Modular functions are often called additive or linear.
- Modular functions are submodular since $m(A) + m(B) \geq m(A \cup B) + m(A \cap B)$.

Modular functions, and vectors in $\mathbb{R}^V$

- Any set function $m : 2^V \to \mathbb{R}$ whose valuations, for $A \subseteq V$, take form

$$m(A) = \sum_{a \in A} m(a) \tag{7}$$

  is called modular and normalized (meaning $m(\emptyset) = 0$).
- Any normalized modular function is identical to a vector:

$$m \in \mathbb{R}^V. \tag{8}$$

- Hence, the characteristic vector $\mathbf{1}_A$ of a set is modular.
- Modular functions are often called additive or linear.
- Modular functions are submodular since $m(A) + m(B) \geq m(A \cup B) + m(A \cap B)$.
- Modular functions are also supermodular since $m(A) + m(B) \leq m(A \cup B) + m(A \cap B)$.

Modular functions, and vectors in $\mathbb{R}^V$

- Any set function $m : 2^V \to \mathbb{R}$ whose valuations, for $A \subseteq V$, take form

$$m(A) = \sum_{a \in A} m(a) \tag{7}$$

  is called modular and normalized (meaning $m(\emptyset) = 0$).
- Any normalized modular function is identical to a vector:

$$m \in \mathbb{R}^V. \tag{8}$$

- Hence, the characteristic vector $\mathbf{1}_A$ of a set is modular.
- Modular functions are often called additive or linear.
- Modular functions are submodular since $m(A) + m(B) \geq m(A \cup B) + m(A \cap B)$.
- Modular functions are also supermodular since $m(A) + m(B) \leq m(A \cup B) + m(A \cap B)$.
- If f is both submodular and supermodular, then it is modular.

Discrete Optimization

- We are given a finite set of objects V of size $n = |V|$.

Discrete Optimization

- We are given a finite set of objects V of size $n = |V|$.
- There are 2^n such subsets (denoted 2^V) of the form $A \subseteq V$.

Discrete Optimization

- We are given a finite set of objects V of size $n = |V|$.
- There are 2^n such subsets (denoted 2^V) of the form $A \subseteq V$.
- We have a function $f : 2^V \to \mathbb{R}$ that judges the quality (or value, or cost, or etc.) of each subset. $f(A) =$ some real number.

Discrete Optimization

- We are given a finite set of objects V of size $n = |V|$.
- There are 2^n such subsets (denoted 2^V) of the form $A \subseteq V$.
- We have a function $f : 2^V \to \mathbb{R}$ that judges the quality (or value, or cost, or etc.) of each subset. $f(A) =$ some real number.
- Unconstrained minimization & maximization:

$$\boxed{\min_{X \subseteq V} f(X)} \qquad (9)$$

$$\boxed{\max_{X \subseteq V} f(X)} \qquad (10)$$

Discrete Optimization

- We are given a finite set of objects V of size $n = |V|$.
- There are 2^n such subsets (denoted 2^V) of the form $A \subseteq V$.
- We have a function $f : 2^V \to \mathbb{R}$ that judges the quality (or value, or cost, or etc.) of each subset. $f(A) =$ some real number.
- Unconstrained minimization & maximization:

$$\min_{X \subseteq V} f(X) \quad (9) \qquad\qquad \max_{X \subseteq V} f(X) \quad (10)$$

- Without knowing anything about f, it takes 2^n queries to be able to offer any quality assurance on a candidate solution. Otherwise, solution can be unboundedly poor.

Discrete Optimization

- We are given a finite set of objects V of size $n = |V|$.
- There are 2^n such subsets (denoted 2^V) of the form $A \subseteq V$.
- We have a function $f : 2^V \to \mathbb{R}$ that judges the quality (or value, or cost, or etc.) of each subset. $f(A) =$ some real number.
- Unconstrained minimization & maximization:

$$\boxed{\min_{X \subseteq V} f(X)} \tag{9}$$

$$\boxed{\max_{X \subseteq V} f(X)} \tag{10}$$

- Without knowing anything about f, it takes 2^n queries to be able to offer any quality assurance on a candidate solution. Otherwise, solution can be unboundedly poor.
- When f is submodular, Eq. (9) is polytime, and Eq. (10) is constant-factor approximable.

Constrained Discrete Optimization

- Often, we are interested only in a subset of the set of possible subsets, namely $\mathcal{S} \subseteq 2^V$.

Constrained Discrete Optimization

- Often, we are interested only in a subset of the set of possible subsets, namely $\mathcal{S} \subseteq 2^V$.
- Example: only sets having bounded size $\mathcal{S} = \{S \subseteq V : |S| \leq k\}$ or within a budget $\{S \subseteq V : \sum_{s \in S} w(s) \leq b\}$.

Constrained Discrete Optimization

- Often, we are interested only in a subset of the set of possible subsets, namely $\mathcal{S} \subseteq 2^V$.
- Example: only sets having bounded size $\mathcal{S} = \{S \subseteq V : |S| \leq k\}$ or within a budget $\left\{S \subseteq V : \sum_{s \in S} w(s) \leq b\right\}$.
- Example: the sets might need to correspond to a combinatorially feasible object (i.e., feasible $\mathcal{S}$ might be trees, matchings, paths, vertex covers, or cuts).

Constrained Discrete Optimization

- Often, we are interested only in a subset of the set of possible subsets, namely $\mathcal{S} \subseteq 2^V$.
- Example: only sets having bounded size $\mathcal{S} = \{S \subseteq V : |S| \leq k\}$ or within a budget $\left\{S \subseteq V : \sum_{s \in S} w(s) \leq b\right\}$.
- Example: the sets might need to correspond to a combinatorially feasible object (i.e., feasible $\mathcal{S}$ might be trees, matchings, paths, vertex covers, or cuts).
- Ex: $\mathcal{S}$ might be a function of some g (e.g., sub-level sets of g, $\mathcal{S} = \{S \subseteq V : g(S) \leq \alpha\}$, sup-level sets $\mathcal{S} = \{S \subseteq V : g(S) \geq \alpha\}$).

Constrained Discrete Optimization

- Often, we are interested only in a subset of the set of possible subsets, namely $\mathcal{S} \subseteq 2^V$.
- Example: only sets having bounded size $\mathcal{S} = \{S \subseteq V : |S| \leq k\}$ or within a budget $\left\{S \subseteq V : \sum_{s \in S} w(s) \leq b\right\}$.
- Example: the sets might need to correspond to a combinatorially feasible object (i.e., feasible $\mathcal{S}$ might be trees, matchings, paths, vertex covers, or cuts).
- Ex: $\mathcal{S}$ might be a function of some g (e.g., sub-level sets of g, $\mathcal{S} = \{S \subseteq V : g(S) \leq \alpha\}$, sup-level sets $\mathcal{S} = \{S \subseteq V : g(S) \geq \alpha\}$).
- Constrained discrete optimization problems:

$$\begin{array}{ll} \underset{S \subseteq 2^V}{\text{maximize}} & f(S) \\ \text{subject to} & S \in \mathcal{S} \end{array} \tag{11}$$

$$\begin{array}{ll} \underset{S \subseteq 2^V}{\text{minimize}} & f(S) \\ \text{subject to} & S \in \mathcal{S} \end{array} \tag{12}$$

Constrained Discrete Optimization

- Often, we are interested only in a subset of the set of possible subsets, namely $\mathcal{S} \subseteq 2^V$.
- Example: only sets having bounded size $\mathcal{S} = \{S \subseteq V : |S| \leq k\}$ or within a budget $\left\{S \subseteq V : \sum_{s \in S} w(s) \leq b\right\}$.
- Example: the sets might need to correspond to a combinatorially feasible object (i.e., feasible $\mathcal{S}$ might be trees, matchings, paths, vertex covers, or cuts).
- Ex: $\mathcal{S}$ might be a function of some g (e.g., sub-level sets of g, $\mathcal{S} = \{S \subseteq V : g(S) \leq \alpha\}$, sup-level sets $\mathcal{S} = \{S \subseteq V : g(S) \geq \alpha\}$).
- Constrained discrete optimization problems:

$$\begin{array}{ll} \underset{S \subseteq 2^V}{\text{maximize}} & f(S) \\ \text{subject to} & S \in \mathcal{S} \end{array} \qquad (11)$$

$$\begin{array}{ll} \underset{S \subseteq 2^V}{\text{minimize}} & f(S) \\ \text{subject to} & S \in \mathcal{S} \end{array} \qquad (12)$$

- Fortunately, when f (and g) are submodular, solving these problems can often be done with guarantees (and often efficiently)!

Outline: Part 1

1. Introduction
2. Basics
3. Submodular Applications in ML
   - Where is submodularity useful?
   - Traditional combinatorial problems
   - As a model of diversity, coverage, span, or information
   - As a model of cooperative costs, complexity, roughness, and irregularity
   - As a parameter for an ML algorithm
   - Itself, as a target for learning
   - Surrogates for optimization
   - Economic applications

Where is submodularity useful in ML?

- As a model of a physical process:

Where is submodularity useful in ML?

- As a model of a physical process:
  - What a submodular function is good for modeling depends on if we wish to maximize or wish to minimize it.

Where is submodularity useful in ML?

- As a model of a physical process:
    - What a submodular function is good for modeling depends on if we wish to maximize or wish to minimize it.
    - Submodular functions naturally model aspects like:

Where is submodularity useful in ML?

- As a model of a physical process:
  - What a submodular function is good for modeling depends on if we wish to maximize or wish to minimize it.
  - Submodular functions naturally model aspects like:
  - diversity, coverage, span, and information in maximization problems,

Where is submodularity useful in ML?

- As a model of a physical process:
  - What a submodular function is good for modeling depends on if we wish to maximize or wish to minimize it.
  - Submodular functions naturally model aspects like:
  - diversity, coverage, span, and information in maximization problems,
  - and cooperative costs, complexity, roughness, and irregularity in minimization problems.

Where is submodularity useful in ML?

- As a model of a physical process:
  - What a submodular function is good for modeling depends on if we wish to maximize or wish to minimize it.
  - Submodular functions naturally model aspects like:
  - diversity, coverage, span, and information in maximization problems,
  - and cooperative costs, complexity, roughness, and irregularity in minimization problems.
- A submodular function can act as a parameter for a machine learning strategy (active/semi-supervised learning, discrete divergence, convex norms for use in regularization).

Where is submodularity useful in ML?

- As a model of a physical process:
  - What a submodular function is good for modeling depends on if we wish to maximize or wish to minimize it.
  - Submodular functions naturally model aspects like:
  - diversity, coverage, span, and information in maximization problems,
  - and cooperative costs, complexity, roughness, and irregularity in minimization problems.
- A submodular function can act as a parameter for a machine learning strategy (active/semi-supervised learning, discrete divergence, convex norms for use in regularization).
- Itself, as an object or function to learn, based on data.

Where is submodularity useful in ML?

- As a model of a physical process:
  - What a submodular function is good for modeling depends on if we wish to maximize or wish to minimize it.
  - Submodular functions naturally model aspects like:
  - diversity, coverage, span, and information in maximization problems,
  - and cooperative costs, complexity, roughness, and irregularity in minimization problems.
- A submodular function can act as a parameter for a machine learning strategy (active/semi-supervised learning, discrete divergence, convex norms for use in regularization).
- Itself, as an object or function to learn, based on data.
- As a surrogate or relaxation strategy for optimization

Where is submodularity useful in ML?

- As a model of a physical process:
  - What a submodular function is good for modeling depends on if we wish to maximize or wish to minimize it.
  - Submodular functions naturally model aspects like:
  - diversity, coverage, span, and information in maximization problems,
  - and cooperative costs, complexity, roughness, and irregularity in minimization problems.
- A submodular function can act as a parameter for a machine learning strategy (active/semi-supervised learning, discrete divergence, convex norms for use in regularization).
- Itself, as an object or function to learn, based on data.
- As a surrogate or relaxation strategy for optimization
  - An alternate to factorization or decomposition based simplification (as one finds in a graphical model).

Where is submodularity useful in ML?

- As a model of a physical process:
  - What a submodular function is good for modeling depends on if we wish to maximize or wish to minimize it.
  - Submodular functions naturally model aspects like:
  - diversity, coverage, span, and information in maximization problems,
  - and cooperative costs, complexity, roughness, and irregularity in minimization problems.
- A submodular function can act as a parameter for a machine learning strategy (active/semi-supervised learning, discrete divergence, convex norms for use in regularization).
- Itself, as an object or function to learn, based on data.
- As a surrogate or relaxation strategy for optimization
  - An alternate to factorization or decomposition based simplification (as one finds in a graphical model).
  - Also, we can "relax" a problem to a submodular one where it can be efficiently solved and offer a bounded quality solution.

Outline

1 Introduction

2 Basics

3 Submodular Applications in ML
- Where is submodularity useful?
- Traditional combinatorial problems
- As a model of diversity, coverage, span, or information
- As a model of cooperative costs, complexity, roughness, and irregularity
- As a parameter for an ML algorithm
- Itself, as a target for learning
- Surrogates for optimization
- Economic applications

Set Cover and Maximum Coverage

- We are given a finite set V of n elements and a set of subsets $\mathcal{V} = \{V_1, V_2, \dots, V_m\}$ of m subsets of V, so that $V_i \subseteq V$ and $\bigcup_i V_i = V$.

SET COVER and MAXIMUM COVERAGE

- We are given a finite set V of n elements and a set of subsets $\mathcal{V} = \{V_1, V_2, \dots, V_m\}$ of m subsets of V, so that $V_i \subseteq V$ and $\bigcup_i V_i = V$.
- The goal of minimum SET COVER is to choose the smallest subset $A \subseteq [m] \triangleq \{1, \dots, m\}$ such that $\bigcup_{a \in A} V_a = V$.

SET COVER and MAXIMUM COVERAGE

- We are given a finite set V of n elements and a set of subsets $\mathcal{V} = \{V_1, V_2, \dots, V_m\}$ of m subsets of V, so that $V_i \subseteq V$ and $\bigcup_i V_i = V$.
- The goal of minimum SET COVER is to choose the smallest subset $A \subseteq [m] \triangleq \{1, \dots, m\}$ such that $\bigcup_{a \in A} V_a = V$.
- Maximum k cover: The goal in MAXIMUM COVERAGE is, given an integer $k \leq m$, select k subsets, say $\{a_1, a_2, \dots, a_k\}$ with $a_i \in [m]$ such that $|\bigcup_{i=1}^k V_{a_i}|$ is maximized.

SET COVER and MAXIMUM COVERAGE

- We are given a finite set V of n elements and a set of subsets $\mathcal{V} = \{V_1, V_2, \dots, V_m\}$ of m subsets of V, so that $V_i \subseteq V$ and $\bigcup_i V_i = V$.
- The goal of minimum SET COVER is to choose the smallest subset $A \subseteq [m] \triangleq \{1, \dots, m\}$ such that $\bigcup_{a \in A} V_a = V$.
- Maximum k cover: The goal in MAXIMUM COVERAGE is, given an integer $k \leq m$, select k subsets, say $\{a_1, a_2, \dots, a_k\}$ with $a_i \in [m]$ such that $|\bigcup_{i=1}^k V_{a_i}|$ is maximized.
- Both SET COVER and MAXIMUM COVERAGE are well known to be NP-hard, but have a fast greedy approximation algorithm.

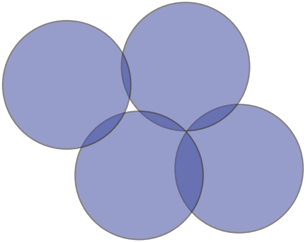
SET COVER and MAXIMUM COVERAGE

- We are given a finite set V of n elements and a set of subsets $\mathcal{V} = \{V_1, V_2, \dots, V_m\}$ of m subsets of V, so that $V_i \subseteq V$ and $\bigcup_i V_i = V$.
- The goal of minimum SET COVER is to choose the smallest subset $A \subseteq [m] \triangleq \{1, \dots, m\}$ such that $\bigcup_{a \in A} V_a = V$.
- Maximum k cover: The goal in MAXIMUM COVERAGE is, given an integer $k \leq m$, select k subsets, say $\{a_1, a_2, \dots, a_k\}$ with $a_i \in [m]$ such that $|\bigcup_{i=1}^{k} V_{a_i}|$ is maximized.
- Both SET COVER and MAXIMUM COVERAGE are well known to be NP-hard, but have a fast greedy approximation algorithm.
- The set cover function $f(A) = |\bigcup_{a \in A} V_a|$ is submodular!

Area of the union of areas indexed by A

- Let V be a set of indices, and each $v \in V$ indexes a given sub-area of some region.
- Let $\text{area}(v)$ be the area corresponding to item v.
- Let $f(S) = \bigcup_{s \in S} \text{area}(s)$ be the union of the areas indexed by elements in A.
- Then $f(S)$ is submodular.

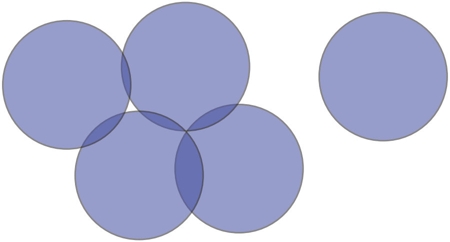
Area of the union of areas indexed by A



Union of areas of elements of A is given by:

$$f(A) = f(\{a_1, a_2, a_3, a_4\})$$

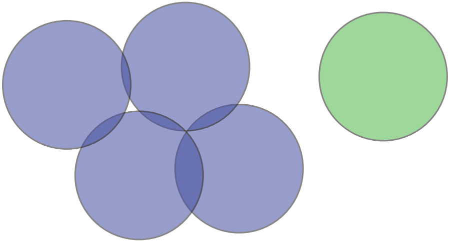
Area of the union of areas indexed by A



Area of A along with with v:

$$f(A \cup \{v\}) = f(\{a_1, a_2, a_3, a_4\} \cup \{v\})$$

Area of the union of areas indexed by A

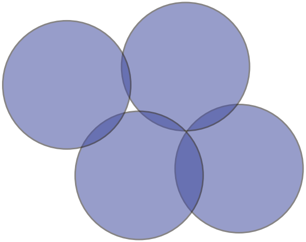


Gain (value) of v in context of A:

$$f(A \cup \{v\}) - f(A) = f(\{v\})$$

We get full value $f(\{v\})$ in this case since the area of v has no overlap with that of A.

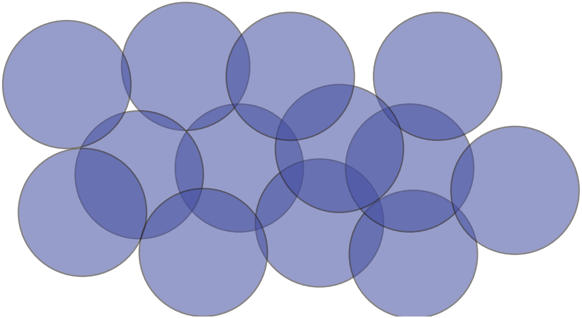
Area of the union of areas indexed by A



Area of A once again.

$$f(A) = f(\{a_1, a_2, a_3, a_4\})$$

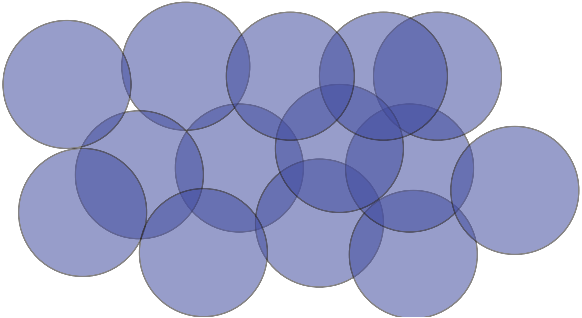
Area of the union of areas indexed by A



Union of areas of elements of $B \supset A$, where v is not included:

$$f(B) \text{ where } v \notin B \text{ and where } A \subseteq B$$

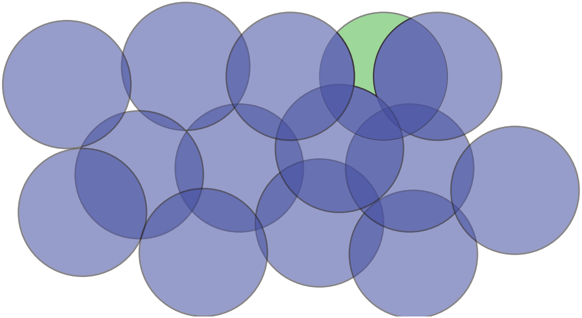
Area of the union of areas indexed by A



Area of B now also including v:

$$f(B \cup \{v\})$$

Area of the union of areas indexed by A



Incremental value of v in the context of $B \supset A$.

$$f(B \cup \{v\}) - f(B) < f(\{v\}) = f(A \cup \{v\}) - f(A)$$

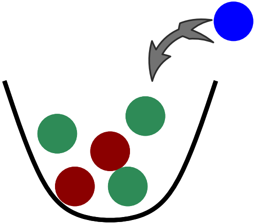
So benefit of v in the context of A is greater than the benefit of v in the context of $B \supseteq A$.

Example Submodular: Number of Colors of Balls in Urns

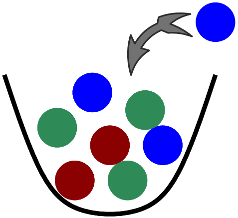
- Consider an urn containing colored balls. Given a set S of balls, $f(S)$ counts the number of distinct colors.

Example Submodular: Number of Colors of Balls in Urns

- Consider an urn containing colored balls. Given a set S of balls, $f(S)$ counts the number of distinct colors.



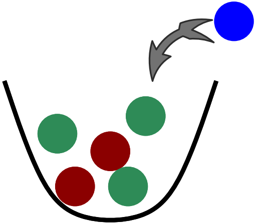
Initial value: 2 (colors in urn).
New value with added blue ball: 3



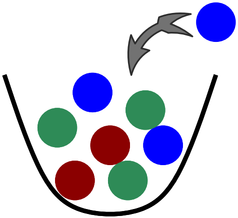
Initial value: 3 (colors in urn).
New value with added blue ball: 3

Example Submodular: Number of Colors of Balls in Urns

- Consider an urn containing colored balls. Given a set S of balls, $f(S)$ counts the number of distinct colors.



Initial value: 2 (colors in urn).
New value with added blue ball: 3

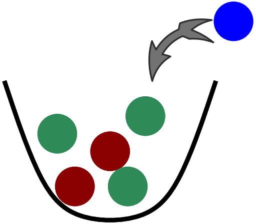


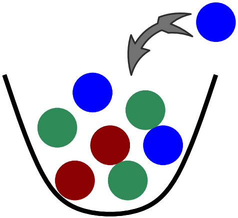
Initial value: 3 (colors in urn).
New value with added blue ball: 3

- Submodularity: Incremental Value of Object Diminishes in a Larger Context (diminishing returns).

Example Submodular: Number of Colors of Balls in Urns

- Consider an urn containing colored balls. Given a set S of balls, $f(S)$ counts the number of distinct colors.





Initial value: 2 (colors in urn).
New value with added blue ball: 3

Initial value: 3 (colors in urn).
New value with added blue ball: 3

- Submodularity: Incremental Value of Object Diminishes in a Larger Context (diminishing returns).
- Thus, f is submodular.

Vertex and Edge Covers

Definition (vertex cover)

A vertex cover (a "vertex-based cover of edges") in graph $G = (V, E)$ is a set $S \subseteq V(G)$ of vertices such that every edge in G is incident to at least one vertex in S.

Vertex and Edge Covers

Definition (vertex cover)

A vertex cover (a "vertex-based cover of edges") in graph $G = (V, E)$ is a set $S \subseteq V(G)$ of vertices such that every edge in G is incident to at least one vertex in S.

- Let $I(S)$ be the number of edges incident to vertex set S. Then we wish to find the smallest set $S \subseteq V$ subject to $I(S) = |E|$.

Vertex and Edge Covers

Definition (vertex cover)

A vertex cover (a "vertex-based cover of edges") in graph $G = (V, E)$ is a set $S \subseteq V(G)$ of vertices such that every edge in G is incident to at least one vertex in S.

- Let $I(S)$ be the number of edges incident to vertex set S. Then we wish to find the smallest set $S \subseteq V$ subject to $I(S) = |E|$.
- $I(S)$ is submodular.

Vertex and Edge Covers

Definition (vertex cover)

A vertex cover (a "vertex-based cover of edges") in graph $G = (V, E)$ is a set $S \subseteq V(G)$ of vertices such that every edge in G is incident to at least one vertex in S.

- Let $I(S)$ be the number of edges incident to vertex set S. Then we wish to find the smallest set $S \subseteq V$ subject to $I(S) = |E|$.
- $I(S)$ is submodular.

Definition (edge cover)

A edge cover (an "edge-based cover of vertices") in graph $G = (V, E)$ is a set $F \subseteq E(G)$ of edges such that every vertex in G is incident to at least one edge in F.

Vertex and Edge Covers

Definition (vertex cover)

A vertex cover (a "vertex-based cover of edges") in graph $G = (V, E)$ is a set $S \subseteq V(G)$ of vertices such that every edge in G is incident to at least one vertex in S.

- Let $I(S)$ be the number of edges incident to vertex set S. Then we wish to find the smallest set $S \subseteq V$ subject to $I(S) = |E|$.
- $I(S)$ is submodular.

Definition (edge cover)

A edge cover (an "edge-based cover of vertices") in graph $G = (V, E)$ is a set $F \subseteq E(G)$ of edges such that every vertex in G is incident to at least one edge in F.

- Let $|V|(F)$ be the number of vertices incident to edge set F. Then we wish to find the smallest set $F \subseteq E$ subject to $|V|(F) = |V|$.

Vertex and Edge Covers

Definition (vertex cover)

A vertex cover (a "vertex-based cover of edges") in graph $G = (V, E)$ is a set $S \subseteq V(G)$ of vertices such that every edge in G is incident to at least one vertex in S.

- Let $I(S)$ be the number of edges incident to vertex set S. Then we wish to find the smallest set $S \subseteq V$ subject to $I(S) = |E|$.
- $I(S)$ is submodular.

Definition (edge cover)

A edge cover (an "edge-based cover of vertices") in graph $G = (V, E)$ is a set $F \subseteq E(G)$ of edges such that every vertex in G is incident to at least one edge in F.

- Let $|V|(F)$ be the number of vertices incident to edge set F. Then we wish to find the smallest set $F \subseteq E$ subject to $|V|(F) = |V|$.
- Let $|V|(F)$ is submodular.

Graph Cut Problems

- Given a graph $G = (V, E)$, let $f : 2^V \to \mathbb{R}_+$ be the cut function, namely for any given set of nodes $X \subseteq V$, $f(X)$ measures the number of edges between nodes X and $V \setminus X$.

$$f(X) = \left|\{(u, v) \in E : u \in X, v \in V \setminus X\}\right| \tag{13}$$

Graph Cut Problems

- Given a graph $G = (V, E)$, let $f : 2^V \to \mathbb{R}_+$ be the cut function, namely for any given set of nodes $X \subseteq V$, $f(X)$ measures the number of edges between nodes X and $V \setminus X$.

$$f(X) = \Big|\{(u, v) \in E : u \in X, v \in V \setminus X\}\Big| \tag{13}$$

- MINIMUM CUT: Given a graph $G = (V, E)$, find a set of vertices $S \subseteq V$ that minimize the cut (set of edges) between S and $V \setminus S$.

Graph Cut Problems

- Given a graph $G = (V, E)$, let $f : 2^V \to \mathbb{R}_+$ be the cut function, namely for any given set of nodes $X \subseteq V$, $f(X)$ measures the number of edges between nodes X and $V \setminus X$.

$$f(X) = \left|\{(u, v) \in E : u \in X, v \in V \setminus X\}\right| \tag{13}$$

- Minimum cut: Given a graph $G = (V, E)$, find a set of vertices $S \subseteq V$ that minimize the cut (set of edges) between S and $V \setminus S$.
- Maximum cut: Given a graph $G = (V, E)$, find a set of vertices $S \subseteq V$ that minimize the cut (set of edges) between S and $V \setminus S$.

Graph Cut Problems

- Given a graph $G = (V, E)$, let $f : 2^V \to \mathbb{R}_+$ be the cut function, namely for any given set of nodes $X \subseteq V$, $f(X)$ measures the number of edges between nodes X and $V \setminus X$.

$$f(X) = \left|\{(u, v) \in E : u \in X, v \in V \setminus X\}\right| \tag{13}$$

- MINIMUM CUT: Given a graph $G = (V, E)$, find a set of vertices $S \subseteq V$ that minimize the cut (set of edges) between S and $V \setminus S$.
- MAXIMUM CUT: Given a graph $G = (V, E)$, find a set of vertices $S \subseteq V$ that minimize the cut (set of edges) between S and $V \setminus S$.
- Weighted versions, we have a non-negative modular function $w : 2^E \to \mathbb{R}_+$ defined on the edges that give cut costs.

$$f(X) = w\Big(\{(u, v) \in E : u \in X, v \in V \setminus X\}\Big) \tag{14}$$

$$= \sum_{e \in \{(u,v) \in E : u \in X, v \in V \setminus X\}} w(e) \tag{15}$$

Graph Cut Problems

- Given a graph $G = (V, E)$, let $f : 2^V \to \mathbb{R}_+$ be the cut function, namely for any given set of nodes $X \subseteq V$, $f(X)$ measures the number of edges between nodes X and $V \setminus X$.

$$f(X) = \Big|\{(u,v) \in E : u \in X, v \in V \setminus X\}\Big| \tag{13}$$

- MINIMUM CUT: Given a graph $G = (V, E)$, find a set of vertices $S \subseteq V$ that minimize the cut (set of edges) between S and $V \setminus S$.
- MAXIMUM CUT: Given a graph $G = (V, E)$, find a set of vertices $S \subseteq V$ that minimize the cut (set of edges) between S and $V \setminus S$.
- Weighted versions, we have a non-negative modular function $w : 2^E \to \mathbb{R}_+$ defined on the edges that give cut costs.

$$f(X) = w\Big(\{(u,v) \in E : u \in X, v \in V \setminus X\}\Big) \tag{14}$$

$$= \sum_{e \in \{(u,v) \in E : u \in X, v \in V \setminus X\}} w(e) \tag{15}$$

- Both functions (Equations (13) and (14)) are submodular.

Outline

1 Introduction

2 Basics

3 Submodular Applications in ML
- Where is submodularity useful?
- Traditional combinatorial problems
- As a model of diversity, coverage, span, or information
- As a model of cooperative costs, complexity, roughness, and irregularity
- As a parameter for an ML algorithm
- Itself, as a target for learning
- Surrogates for optimization
- Economic applications

Extractive Document Summarization

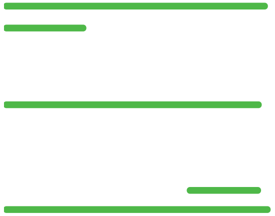
- The figure below represents the sentences of a document

Extractive Document Summarization

- We extract sentences (green) as a summary of the full document

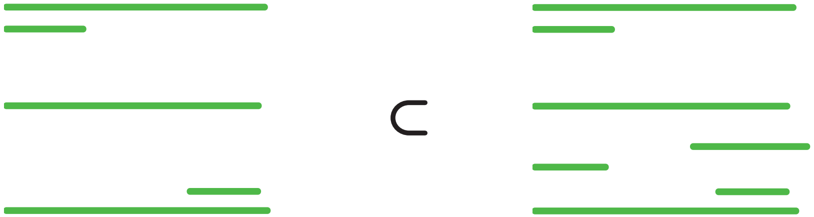
Extractive Document Summarization

- We extract sentences (green) as a summary of the full document



Extractive Document Summarization

- We extract sentences (green) as a summary of the full document



- The summary on the left is a subset of the summary on the right.

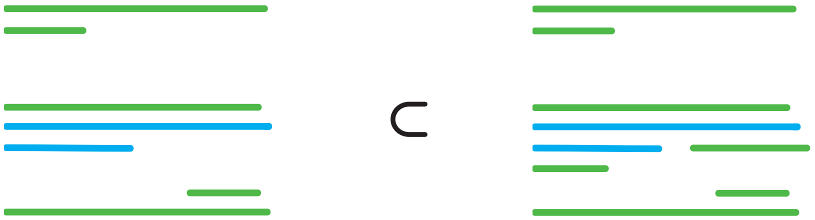
Extractive Document Summarization

- We extract sentences (green) as a summary of the full document

- The summary on the left is a subset of the summary on the right.
- Consider adding a new (blue) sentence to each of the two summaries.

Extractive Document Summarization

- We extract sentences (green) as a summary of the full document



- The summary on the left is a subset of the summary on the right.
- Consider adding a new (blue) sentence to each of the two summaries.
- The marginal (incremental) benefit of adding the new (blue) sentence to the smaller (left) summary is no more than the marginal benefit of adding the new sentence to the larger (right) summary.

Extractive Document Summarization

- We extract sentences (green) as a summary of the full document

- The summary on the left is a subset of the summary on the right.
- Consider adding a new (blue) sentence to each of the two summaries.
- The marginal (incremental) benefit of adding the new (blue) sentence to the smaller (left) summary is no more than the marginal benefit of adding the new sentence to the larger (right) summary.
- **diminishing returns** $\leftrightarrow$ **submodularity**

Image collections

Many images, also that have a higher level gestalt than just a few.

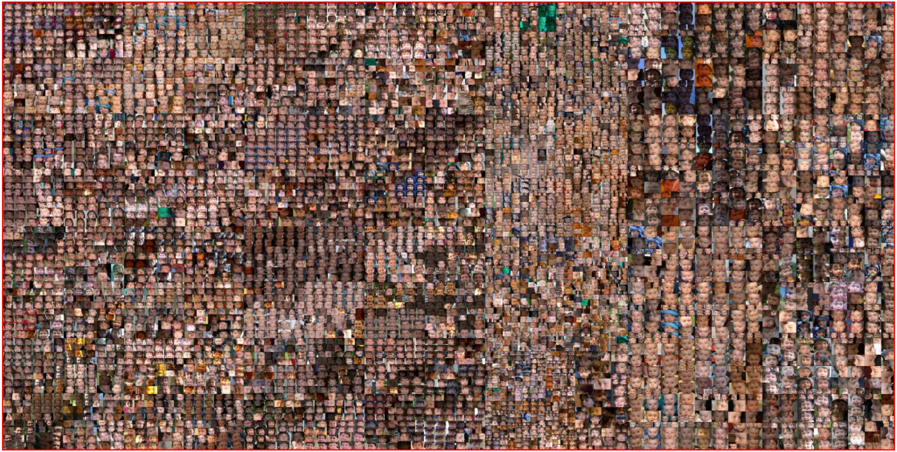


Image Summarization

10×10 image collection:

3 best summaries:

3 medium summaries:

3 worst summaries:

The three best summaries exhibit diversity. The three worst summaries exhibit redundancy.

Feature Selection in Pattern Classification

- Let Y be a random variable we wish to infer as best as possible, based on at most n measurements $(X_1, X_2, \ldots, X_n) = X_V$ (or features) in a probability model $\Pr(Y, X_1, X_2, \ldots, X_n)$.

Feature Selection in Pattern Classification

- Let Y be a random variable we wish to infer as best as possible, based on at most n measurements $(X_1, X_2, \ldots, X_n) = X_V$ (or features) in a probability model $\Pr(Y, X_1, X_2, \ldots, X_n)$.
- It is too costly to use them all, and we wish to choose a good subset $A \subseteq V$ of features to use that are within budget $|A| \leq k$.

Feature Selection in Pattern Classification

- Let Y be a random variable we wish to infer as best as possible, based on at most n measurements $(X_1, X_2, \ldots, X_n) = X_V$ (or features) in a probability model $\Pr(Y, X_1, X_2, \ldots, X_n)$.
- It is too costly to use them all, and we wish to choose a good subset $A \subseteq V$ of features to use that are within budget $|A| \leq k$.
- The mutual information function $f(A) = I(Y; X_A)$ where

$$I(Y; X_A) = \sum_{y, x_A} \Pr(y, x_A) \log \frac{\Pr(y, x_A)}{\Pr(y)\Pr(x_A)} = H(Y) - H(Y|X_A) \quad (16)$$

$$= H(X_A) - H(X_A|Y) = H(X_A) + H(Y) - H(X_A, Y) \quad (17)$$

measures how well features A are for predicting Y (entropy reduction, reduction of uncertainty of Y)

Feature Selection in Pattern Classification

- The mutual information function $f(A) = I(Y; X_A)$ where

$$I(Y;X_A) = \sum_{y,x_A} \Pr(y,x_A) \log \frac{\Pr(y,x_A)}{\Pr(y)\Pr(x_A)} = H(Y) - H(Y|X_A) \quad (18)$$

$$= H(X_A) - H(X_A|Y) = H(X_A) + H(Y) - H(X_A, Y) \quad (19)$$

measures how well X_A does for predicting Y, entropy reduction, reduction of uncertainty of Y, or information gain (Krause & Guestrin) of X_A.

Feature Selection in Pattern Classification

- The mutual information function $f(A) = I(Y; X_A)$ where

$$I(Y; X_A) = \sum_{y, x_A} \Pr(y, x_A) \log \frac{\Pr(y, x_A)}{\Pr(y)\Pr(x_A)} = H(Y) - H(Y|X_A) \quad (18)$$

$$= H(X_A) - H(X_A|Y) = H(X_A) + H(Y) - H(X_A, Y) \quad (19)$$

measures how well X_A does for predicting Y, entropy reduction, reduction of uncertainty of Y, or information gain (Krause & Guestrin) of X_A.

- Goal is to find a subset A of size k that has high information gain.

Feature Selection in Pattern Classification

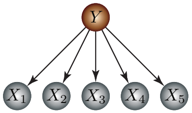
- The mutual information function $f(A) = I(Y; X_A)$ where

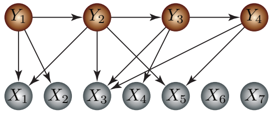
$$I(Y; X_A) = \sum_{y, x_A} \Pr(y, x_A) \log \frac{\Pr(y, x_A)}{\Pr(y)\Pr(x_A)} = H(Y) - H(Y|X_A) \quad (18)$$

$$= H(X_A) - H(X_A|Y) = H(X_A) + H(Y) - H(X_A, Y) \quad (19)$$

measures how well X_A does for predicting Y, entropy reduction, reduction of uncertainty of Y, or information gain (Krause & Guestrin) of X_A.

- Goal is to find a subset A of size k that has high information gain.
- When $X_A \perp\!\!\!\perp X_B | Y$ for all A, B (the Naïve Bayes assumption), $f(A)$ is submodular





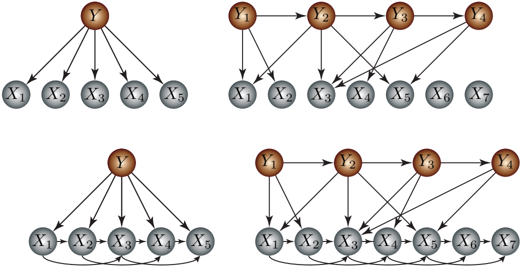
Feature Selection in Pattern Classification

- The mutual information function $f(A) = I(Y; X_A)$ where

$$I(Y;X_A) = \sum_{y,x_A} \Pr(y, x_A) \log \frac{\Pr(y,x_A)}{\Pr(y)\Pr(x_A)} = H(Y) - H(Y|X_A) \quad (18)$$

$$= H(X_A) - H(X_A|Y) = H(X_A) + H(Y) - H(X_A, Y) \quad (19)$$

  measures how well X_A does for predicting Y, entropy reduction, reduction of uncertainty of Y, or information gain (Krause & Guestrin) of X_A.
- Goal is to find a subset A of size k that has high information gain.
- When $X_A \perp\!\!\!\perp X_B | Y$ for all A, B (the Naïve Bayes assumption), $f(A)$ is submodular
- If not, $f(A)$ is naturally expressed as a difference of two submodular functions.



Sensor Placement

- Information gain applicable not only in pattern recognition, but in the sensor coverage problem as well, where Y is whatever question we wish to ask about an environment.

Sensor Placement

- Information gain applicable not only in pattern recognition, but in the sensor coverage problem as well, where Y is whatever question we wish to ask about an environment.
- Given an environment, there is a set V of candidate locations for placement of a sensor (e.g., temperature, gas, audio, video, bacteria or other environmental contaminant, etc.).

Sensor Placement

- Information gain applicable not only in pattern recognition, but in the sensor coverage problem as well, where Y is whatever question we wish to ask about an environment.
- Given an environment, there is a set V of candidate locations for placement of a sensor (e.g., temperature, gas, audio, video, bacteria or other environmental contaminant, etc.).
- We have a function $f(A)$ that measures the "coverage" of any given set A of sensor placement decisions. Then $f(V)$ is maximum possible coverage.

Sensor Placement

- Information gain applicable not only in pattern recognition, but in the sensor coverage problem as well, where Y is whatever question we wish to ask about an environment.
- Given an environment, there is a set V of candidate locations for placement of a sensor (e.g., temperature, gas, audio, video, bacteria or other environmental contaminant, etc.).
- We have a function $f(A)$ that measures the "coverage" of any given set A of sensor placement decisions. Then $f(V)$ is maximum possible coverage.
- One possible goal: choose smallest set A such that $f(A) \geq \alpha f(V)$ with $0 < \alpha \leq 1$.

Sensor Placement

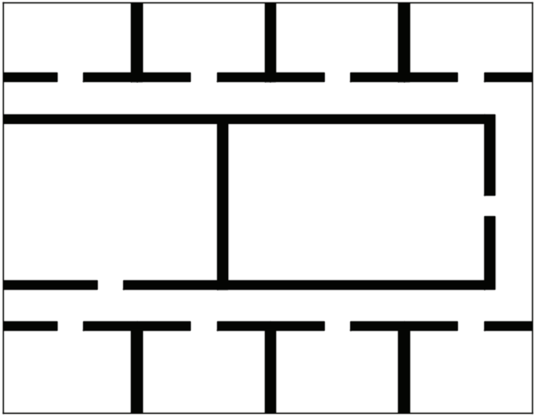
- Information gain applicable not only in pattern recognition, but in the sensor coverage problem as well, where Y is whatever question we wish to ask about an environment.
- Given an environment, there is a set V of candidate locations for placement of a sensor (e.g., temperature, gas, audio, video, bacteria or other environmental contaminant, etc.).
- We have a function $f(A)$ that measures the "coverage" of any given set A of sensor placement decisions. Then $f(V)$ is maximum possible coverage.
- One possible goal: choose smallest set A such that $f(A) \geq \alpha f(V)$ with $0 < \alpha \leq 1$.
- Another possible goal: choose size at most k set A such that $f(A)$ is maximized.

Sensor Placement

- Information gain applicable not only in pattern recognition, but in the sensor coverage problem as well, where Y is whatever question we wish to ask about an environment.
- Given an environment, there is a set V of candidate locations for placement of a sensor (e.g., temperature, gas, audio, video, bacteria or other environmental contaminant, etc.).
- We have a function $f(A)$ that measures the "coverage" of any given set A of sensor placement decisions. Then $f(V)$ is maximum possible coverage.
- One possible goal: choose smallest set A such that $f(A) \geq \alpha f(V)$ with $0 < \alpha \leq 1$.
- Another possible goal: choose size at most k set A such that $f(A)$ is maximized.
- Environment could be a floor of a building, water network, monitored ecological preservation.

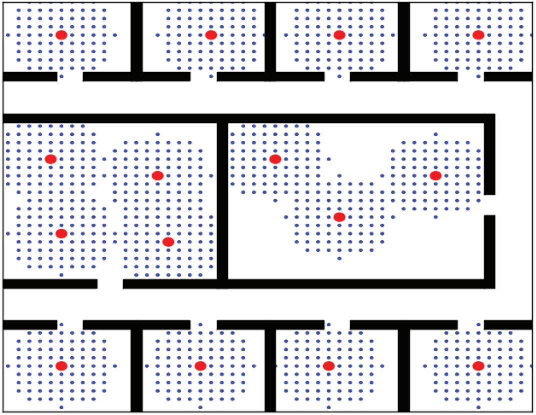
Sensor Placement within Buildings

- An example of a room layout. Should be possible to determine temperature at all points in the room. Sensors cannot sense beyond wall (thick black line) boundaries.



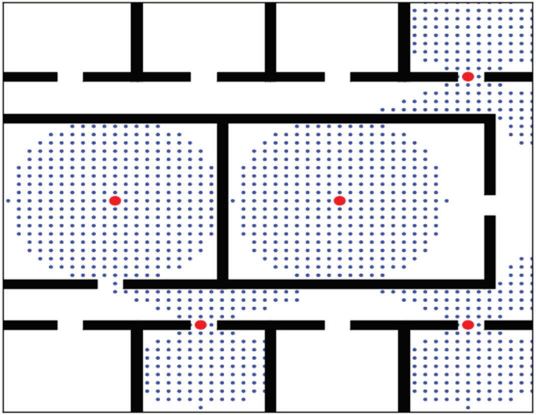
Sensor Placement within Buildings

- Example sensor placement using small range cheap sensors (located at red dots).



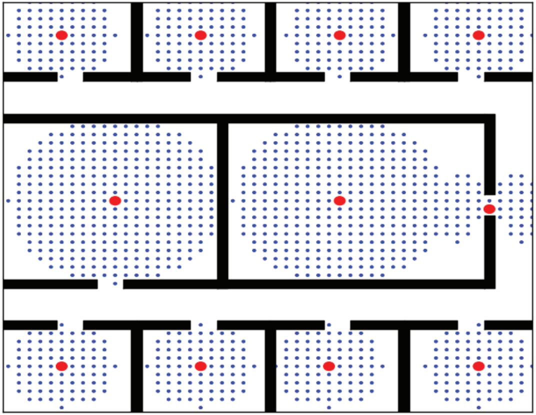
Sensor Placement within Buildings

- Example sensor placement using longer range expensive sensors (located at red dots).



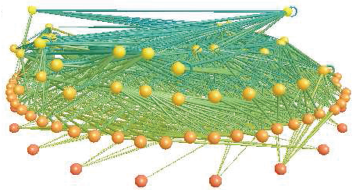
Sensor Placement within Buildings

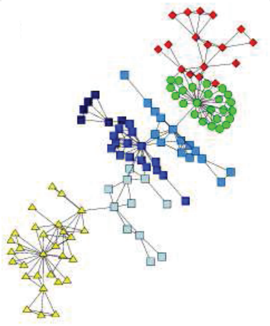
- Example sensor placement using mixed range sensors (located at red dots).

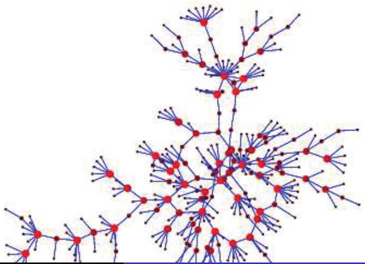


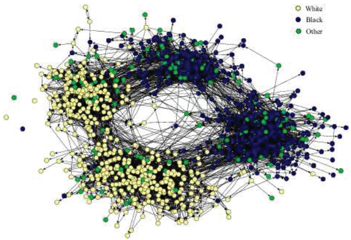
Social Networks

(from Newman, 2004). Clockwise from top left: 1) predator-prey interactions, 2) scientific collaborations, 3) sexual contact, 4) school friendships.









The value of a friend

- Let V be a group of individuals. How valuable to you is a given friend $v \in V$?

The value of a friend

- Let V be a group of individuals. How valuable to you is a given friend $v \in V$?
- It depends on how many friends you have.

The value of a friend

- Let V be a group of individuals. How valuable to you is a given friend $v \in V$?
- It depends on how many friends you have.
- Given a group of friends $S \subseteq V$, you can valuate them with a set function $f(S)$.

The value of a friend

- Let V be a group of individuals. How valuable to you is a given friend $v \in V$?
- It depends on how many friends you have.
- Given a group of friends $S \subseteq V$, you can valuate them with a set function $f(S)$.
- Let $f(S)$ be the value of the set of friends S.

The value of a friend

- Let V be a group of individuals. How valuable to you is a given friend $v \in V$?
- It depends on how many friends you have.
- Given a group of friends $S \subseteq V$, you can valuate them with a set function $f(S)$.
- Let $f(S)$ be the value of the set of friends S.
- Submodular model: a friend is less valuable the more friends you have.

The value of a friend

- Let V be a group of individuals. How valuable to you is a given friend $v \in V$?
- It depends on how many friends you have.
- Given a group of friends $S \subseteq V$, you can valuate them with a set function $f(S)$.
- Let $f(S)$ be the value of the set of friends S.
- Submodular model: a friend is less valuable the more friends you have.
- Supermodular model: a friend is more valuable the more friends you have ("I'd get by with a little help from my friends").

The value of a friend

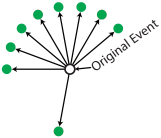
- Let V be a group of individuals. How valuable to you is a given friend $v \in V$?
- It depends on how many friends you have.
- Given a group of friends $S \subseteq V$, you can valuate them with a set function $f(S)$.
- Let $f(S)$ be the value of the set of friends S.
- Submodular model: a friend is less valuable the more friends you have.
- Supermodular model: a friend is more valuable the more friends you have ("I'd get by with a little help from my friends").
- Which is a better model?

Information Cascades, Diffusion Networks

- How to model flow of information from source to the point it reaches users — information used in its common sense (like news events).
- How to find the most influential sources, the ones that often set off cascades, which are like large "waves" of information flow?
- Example when there is one seed source shown below:

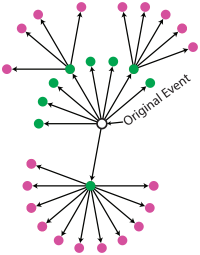
Information Cascades, Diffusion Networks

- How to model flow of information from source to the point it reaches users — information used in its common sense (like news events).
- How to find the most influential sources, the ones that often set off cascades, which are like large "waves" of information flow?
- Example when there is one seed source shown below:



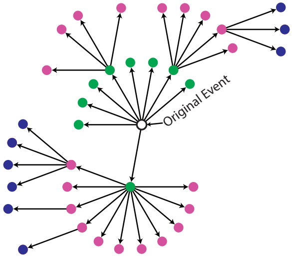
Information Cascades, Diffusion Networks

- How to model flow of information from source to the point it reaches users — information used in its common sense (like news events).
- How to find the most influential sources, the ones that often set off cascades, which are like large "waves" of information flow?
- Example when there is one seed source shown below:



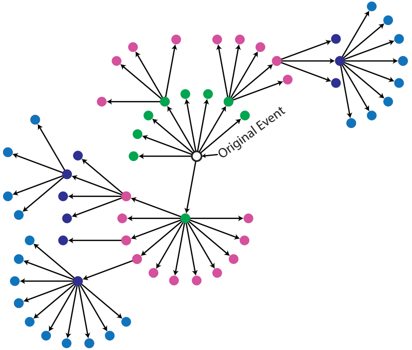
Information Cascades, Diffusion Networks

- How to model flow of information from source to the point it reaches users — information used in its common sense (like news events).
- How to find the most influential sources, the ones that often set off cascades, which are like large "waves" of information flow?
- Example when there is one seed source shown below:



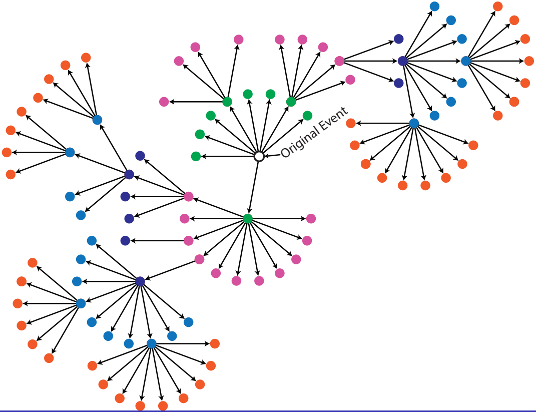
Information Cascades, Diffusion Networks

- How to model flow of information from source to the point it reaches users — information used in its common sense (like news events).
- How to find the most influential sources, the ones that often set off cascades, which are like large "waves" of information flow?
- Example when there is one seed source shown below:



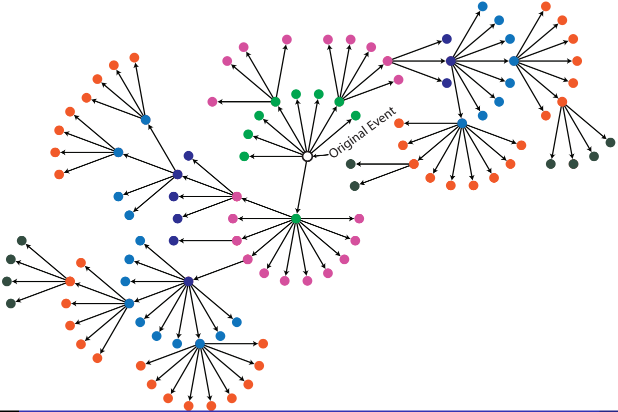
Information Cascades, Diffusion Networks

- How to model flow of information from source to the point it reaches users — information used in its common sense (like news events).
- How to find the most influential sources, the ones that often set off cascades, which are like large "waves" of information flow?
- Example when there is one seed source shown below:



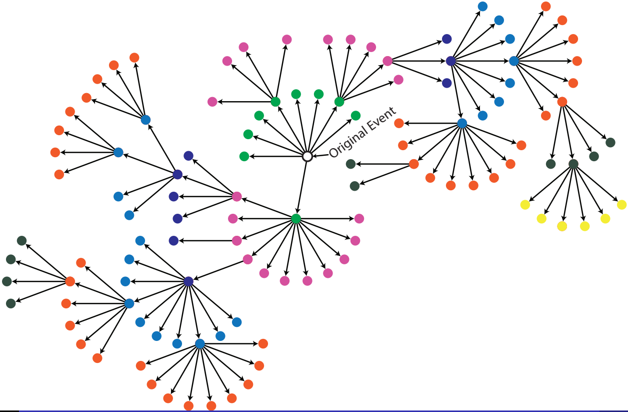
Information Cascades, Diffusion Networks

- How to model flow of information from source to the point it reaches users — information used in its common sense (like news events).
- How to find the most influential sources, the ones that often set off cascades, which are like large “waves” of information flow?
- Example when there is one seed source shown below:



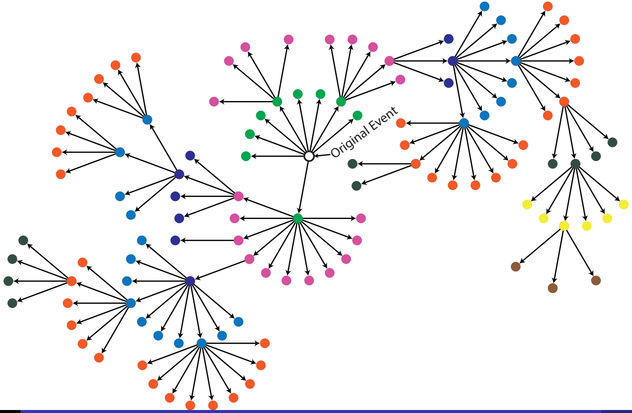
Information Cascades, Diffusion Networks

- How to model flow of information from source to the point it reaches users — information used in its common sense (like news events).
- How to find the most influential sources, the ones that often set off cascades, which are like large "waves" of information flow?
- Example when there is one seed source shown below:



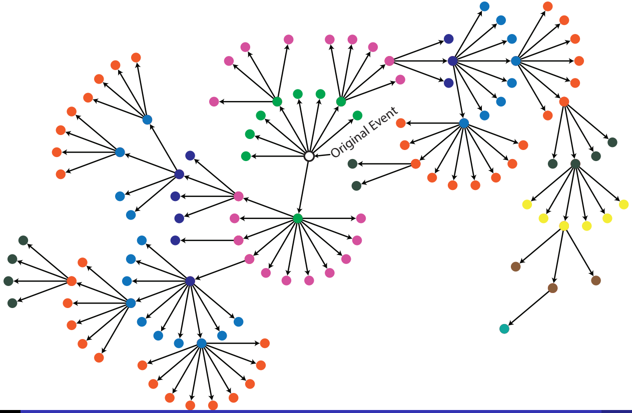
Information Cascades, Diffusion Networks

- How to model flow of information from source to the point it reaches users — information used in its common sense (like news events).
- How to find the most influential sources, the ones that often set off cascades, which are like large "waves" of information flow?
- Example when there is one seed source shown below:



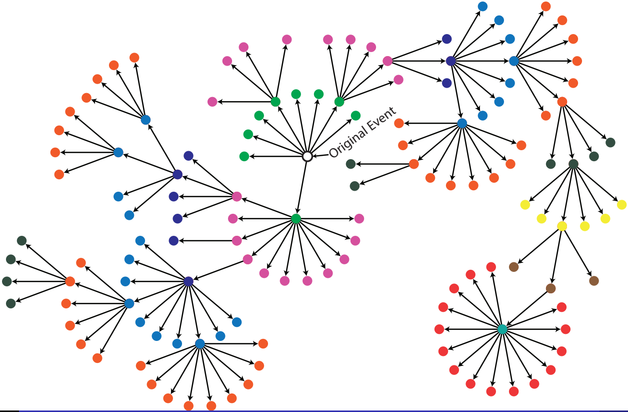
Information Cascades, Diffusion Networks

- How to model flow of information from source to the point it reaches users — information used in its common sense (like news events).
- How to find the most influential sources, the ones that often set off cascades, which are like large "waves" of information flow?
- Example when there is one seed source shown below:



Information Cascades, Diffusion Networks

- How to model flow of information from source to the point it reaches users — information used in its common sense (like news events).
- How to find the most influential sources, the ones that often set off cascades, which are like large "waves" of information flow?
- Example when there is one seed source shown below:



A model of influence in social networks

- Given a graph $G = (V, E)$, each $v \in V$ corresponds to a person, to each v we have an activation function $f_v : 2^V \to [0, 1]$ dependent only on its neighbors. I.e., $f_v(A) = f_v(A \cap \Gamma(v))$.

A model of influence in social networks

- Given a graph $G = (V, E)$, each $v \in V$ corresponds to a person, to each v we have an activation function $f_v : 2^V \to [0, 1]$ dependent only on its neighbors. I.e., $f_v(A) = f_v(A \cap \Gamma(v))$.
- Goal - Viral Marketing: find a small subset $S \subseteq V$ of individuals to directly influence, and thus indirectly influence the greatest number of possible other individuals (via the social network G).

A model of influence in social networks

- Given a graph $G = (V, E)$, each $v \in V$ corresponds to a person, to each v we have an activation function $f_v : 2^V \to [0, 1]$ dependent only on its neighbors. I.e., $f_v(A) = f_v(A \cap \Gamma(v))$.
- Goal - Viral Marketing: find a small subset $S \subseteq V$ of individuals to directly influence, and thus indirectly influence the greatest number of possible other individuals (via the social network G).
- We define a function $f : 2^V \to \mathbb{Z}^+$ that models the ultimate influence of an initial set S of nodes based on the following iterative process: At each step, a given set of nodes S are activated, and we activate new nodes $v \in V \setminus S$ if $f_v(S) \geq U[0, 1]$ (where $U[0, 1]$ is a uniform random number between 0 and 1).

A model of influence in social networks

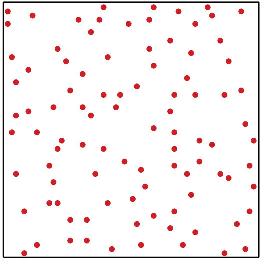
- Given a graph $G = (V, E)$, each $v \in V$ corresponds to a person, to each v we have an activation function $f_v : 2^V \to [0, 1]$ dependent only on its neighbors. I.e., $f_v(A) = f_v(A \cap \Gamma(v))$.
- Goal - Viral Marketing: find a small subset $S \subseteq V$ of individuals to directly influence, and thus indirectly influence the greatest number of possible other individuals (via the social network G).
- We define a function $f : 2^V \to \mathbb{Z}^+$ that models the ultimate influence of an initial set S of nodes based on the following iterative process: At each step, a given set of nodes S are activated, and we activate new nodes $v \in V \setminus S$ if $f_v(S) \geq U[0, 1]$ (where $U[0, 1]$ is a uniform random number between 0 and 1).
- It can be shown that for many f_v (including simple linear functions, and where f_v is submodular itself) that f is submodular (Kempe, Kleinberg, Tardos 1993).

Determinantal Point Processes (DPPs)

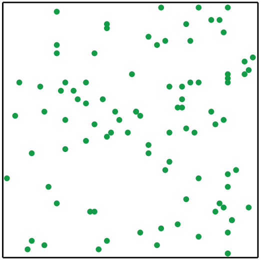
- Sometimes we wish not only to valuate subsets $A \subseteq V$ but to induce probability distributions over all subsets.

Determinantal Point Processes (DPPs)

- Sometimes we wish not only to valuate subsets $A \subseteq V$ but to induce probability distributions over all subsets.
- We may wish to prefer samples where elements of A are diverse (i.e., given a sample A, for $a, b \in A$, we prefer a and b to be different).



DPP

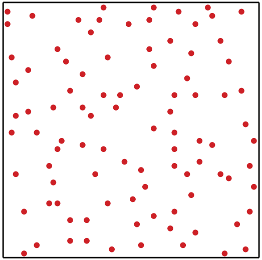


Independent

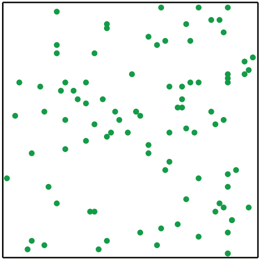
(Kulesza & Taskar, 2011)

Determinantal Point Processes (DPPs)

- Sometimes we wish not only to valuate subsets $A \subseteq V$ but to induce probability distributions over all subsets.
- We may wish to prefer samples where elements of A are diverse (i.e., given a sample A, for $a, b \in A$, we prefer a and b to be different).



DPP



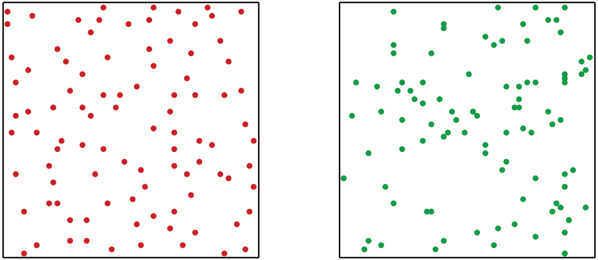
Independent

(Kulesza & Taskar, 2011)

- A Determinantal point processes (DPPs) is a probability distribution over subsets A of V where the "energy" function is submodular.

Determinantal Point Processes (DPPs)

- Sometimes we wish not only to valuate subsets $A \subseteq V$ but to induce probability distributions over all subsets.
- We may wish to prefer samples where elements of A are diverse (i.e., given a sample A, for $a, b \in A$, we prefer a and b to be different).



DPP Independent

(Kulesza & Taskar, 2011)

- A Determinantal point processes (DPPs) is a probability distribution over subsets A of V where the "energy" function is submodular.
- More "diverse" or "complex" samples are given higher probability.

DPPs and log-submodular probability distributions

- Given binary vectors $x, y \in \{0,1\}^V$, $y \leq x$ if $y(v) \leq x(v), \forall v \in V$.

DPPs and log-submodular probability distributions

- Given binary vectors $x, y \in \{0,1\}^V$, $y \leq x$ if $y(v) \leq x(v), \forall v \in V$.
- Given a positive-definite $n \times n$ matrix M and a subset $X \subseteq V$, let M_X be a submatrix (which is $|X| \times |X|$) with rows/columns specified by $X \subseteq V$.

DPPs and log-submodular probability distributions

- Given binary vectors $x, y \in \{0,1\}^V$, $y \leq x$ if $y(v) \leq x(v), \forall v \in V$.
- Given a positive-definite $n \times n$ matrix M and a subset $X \subseteq V$, let M_X be a submatrix (which is $|X| \times |X|$) with rows/columns specified by $X \subseteq V$.
- Consider the following probability distribution on binary vectors:

$$\Pr(\mathbf{X} = x) = \exp\left(\log\left(\frac{|M_{X(x)}|}{|M + I|}\right)\right) \tag{20}$$

where I is $n \times n$ identity matrix, and $\mathbf{X} \in \{0,1\}^V$ is a random vector.

DPPs and log-submodular probability distributions

- Given binary vectors $x, y \in \{0,1\}^V$, $y \leq x$ if $y(v) \leq x(v), \forall v \in V$.
- Given a positive-definite $n \times n$ matrix M and a subset $X \subseteq V$, let M_X be a submatrix (which is $|X| \times |X|$) with rows/columns specified by $X \subseteq V$.
- Consider the following probability distribution on binary vectors:

$$\Pr(\mathbf{X} = x) = \exp\left(\log\left(\frac{|M_{X(x)}|}{|M + I|}\right)\right) \tag{20}$$

where I is $n \times n$ identity matrix, and $\mathbf{X} \in \{0,1\}^V$ is a random vector.
- Equivalently,

$$\sum_{x \in \{0,1\}^V : x \geq y} \Pr(\mathbf{X} = x) = \Pr(\mathbf{X} \geq y) = \exp\left(\log\left(|K_{Y(y)}|\right)\right) \tag{21}$$

where $K = M(M+I)^{-1}$

DPPs and log-submodular probability distributions

- Given binary vectors $x, y \in \{0,1\}^V$, $y \leq x$ if $y(v) \leq x(v), \forall v \in V$.
- Given a positive-definite $n \times n$ matrix M and a subset $X \subseteq V$, let M_X be a submatrix (which is $|X| \times |X|$) with rows/columns specified by $X \subseteq V$.
- Consider the following probability distribution on binary vectors:

$$\Pr(\mathbf{X} = x) = \exp\left(\log\left(\frac{|M_{X(x)}|}{|M + I|}\right)\right) \tag{20}$$

  where I is $n \times n$ identity matrix, and $\mathbf{X} \in \{0,1\}^V$ is a random vector.
- Equivalently,

$$\sum_{x \in \{0,1\}^V : x \geq y} \Pr(\mathbf{X} = x) = \Pr(\mathbf{X} \geq y) = \exp\left(\log\left(|K_{Y(y)}|\right)\right) \tag{21}$$

  where $K = M(M + I)^{-1}$
- Given positive definite matrix M, function $f : 2^V \to \mathbb{R}$ with $f(A) = \log|M_A|$ (the logdet function) is submodular.

Graphical Model Structure Learning

- A probability distribution on binary vectors $p : \{0,1\}^V \to [0,1]$:

$$p(x) = \frac{1}{Z}\exp(-E(x)) \tag{22}$$

where $E(x)$ is the energy function.

Graphical Model Structure Learning

- A probability distribution on binary vectors $p : \{0,1\}^V \to [0,1]$:

$$p(x) = \frac{1}{Z}\exp(-E(x)) \tag{22}$$

  where $E(x)$ is the energy function.
- A graphical model $G = (V, \mathcal{E})$ represents a family of probability distributions $p \in \mathcal{F}(G)$ all of which factor w.r.t. the graph.

Graphical Model Structure Learning

- A probability distribution on binary vectors $p : \{0,1\}^V \to [0,1]$:

$$p(x) = \frac{1}{Z} \exp(-E(x)) \tag{22}$$

  where $E(x)$ is the energy function.
- A graphical model $G = (V, \mathcal{E})$ represents a family of probability distributions $p \in \mathcal{F}(G)$ all of which factor w.r.t. the graph.
- I.e., if $\mathcal{C}$ are a set of cliques of graph G, then we must have:

$$E(x) = \sum_{c \in \mathcal{C}} E_c(x_c) \tag{23}$$

Graphical Model Structure Learning

- A probability distribution on binary vectors $p : \{0,1\}^V \to [0,1]$:

$$p(x) = \frac{1}{Z} \exp(-E(x)) \tag{22}$$

  where $E(x)$ is the energy function.
- A graphical model $G = (V, \mathcal{E})$ represents a family of probability distributions $p \in \mathcal{F}(G)$ all of which factor w.r.t. the graph.
- I.e., if $\mathcal{C}$ are a set of cliques of graph G, then we must have:

$$E(x) = \sum_{c \in \mathcal{C}} E_c(x_c) \tag{23}$$

- The problem of structure learning in graphical models is to find the graph G based on data.

Graphical Model Structure Learning

- A probability distribution on binary vectors $p : \{0,1\}^V \to [0,1]$:

$$p(x) = \frac{1}{Z} \exp(-E(x)) \tag{22}$$

  where $E(x)$ is the energy function.
- A graphical model $G = (V, \mathcal{E})$ represents a family of probability distributions $p \in \mathcal{F}(G)$ all of which factor w.r.t. the graph.
- I.e., if $\mathcal{C}$ are a set of cliques of graph G, then we must have:

$$E(x) = \sum_{c \in \mathcal{C}} E_c(x_c) \tag{23}$$

- The problem of structure learning in graphical models is to find the graph G based on data.
- This can be viewed as a discrete optimization problem on the potential (undirected) edges of the graph $V \times V$.

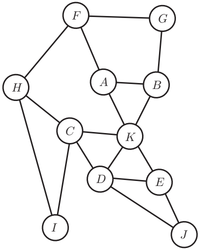
Graphical Models: Learning Tree Distributions

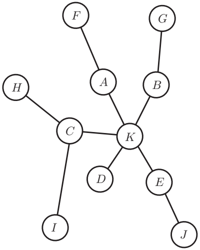
- Goal: find the closest distribution p_t to p subject to p_t factoring w.r.t. some tree $T = (V, F)$, i.e., $p_t \in \mathcal{F}(T, \mathcal{M})$.

Graphical Models: Learning Tree Distributions

- Goal: find the closest distribution p_t to p subject to p_t factoring w.r.t. some tree $T = (V, F)$, i.e., $p_t \in \mathcal{F}(T, \mathcal{M})$.
- This can be expressed as a discrete optimization problem:

$$\begin{array}{ll} \underset{p_t \in \mathcal{F}(G,\mathcal{M})}{\text{minimize}} & D(p||p_t) \\ \text{subject to} & p_t \in \mathcal{F}(T, \mathcal{M}). \\ & T = (V, F) \text{ is a tree} \end{array}$$

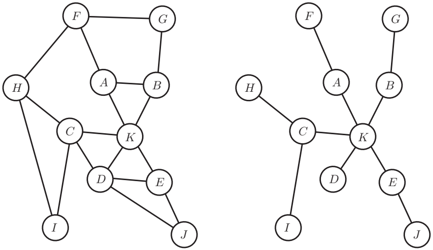




Graphical Models: Learning Tree Distributions

- Goal: find the closest distribution p_t to p subject to p_t factoring w.r.t. some tree $T = (V, F)$, i.e., $p_t \in \mathcal{F}(T, \mathcal{M})$.
- This can be expressed as a discrete optimization problem:

$$\begin{array}{ll} \underset{p_t \in \mathcal{F}(G,\mathcal{M})}{\text{minimize}} & D(p||p_t) \\ \text{subject to} & p_t \in \mathcal{F}(T, \mathcal{M}). \\ & T = (V, F) \text{ is a tree} \end{array}$$

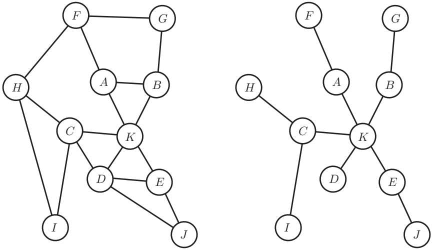


- Discrete problem: choose the optimal set of edges $A \subseteq E$ that constitute tree (i.e., find a spanning tree of G of best quality).

Graphical Models: Learning Tree Distributions

- Goal: find the closest distribution p_t to p subject to p_t factoring w.r.t. some tree $T = (V, F)$, i.e., $p_t \in \mathcal{F}(T, \mathcal{M})$.
- This can be expressed as a discrete optimization problem:

$$\begin{array}{ll} \underset{p_t \in \mathcal{F}(G,\mathcal{M})}{\text{minimize}} & D(p||p_t) \\ \text{subject to} & p_t \in \mathcal{F}(T, \mathcal{M}). \\ & T = (V, F) \text{ is a tree} \end{array}$$

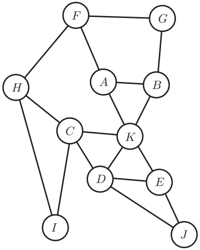


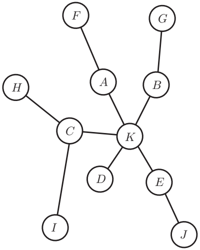
- Discrete problem: choose the optimal set of edges $A \subseteq E$ that constitute tree (i.e., find a spanning tree of G of best quality).
- Define $f : 2^E \to \mathbb{R}_+$ where f is a weighted cycle matroid rank function (a type of submodular function), with weights $w(e) = w(u, v) = I(X_u; X_v)$ for $e \in E$.

Graphical Models: Learning Tree Distributions

- Goal: find the closest distribution p_t to p subject to p_t factoring w.r.t. some tree $T = (V, F)$, i.e., $p_t \in \mathcal{F}(T, \mathcal{M})$.
- This can be expressed as a discrete optimization problem:

$$\begin{array}{ll} \underset{p_t \in \mathcal{F}(G,\mathcal{M})}{\text{minimize}} & D(p||p_t) \\ \text{subject to} & p_t \in \mathcal{F}(T, \mathcal{M}). \\ & T = (V, F) \text{ is a tree} \end{array}$$





- Discrete problem: choose the optimal set of edges $A \subseteq E$ that constitute tree (i.e., find a spanning tree of G of best quality).
- Define $f : 2^E \to \mathbb{R}_+$ where f is a weighted cycle matroid rank function (a type of submodular function), with weights $w(e) = w(u, v) = I(X_u; X_v)$ for $e \in E$.
- Then finding the maximum weight base of the matroid is solved by the greedy algorithm, and also finds the optimal tree (Chow & Liu, 1968)

Outline

1. Introduction
2. Basics
3. Submodular Applications in ML
   - Where is submodularity useful?
   - Traditional combinatorial problems
   - As a model of diversity, coverage, span, or information
   - As a model of cooperative costs, complexity, roughness, and irregularity
   - As a parameter for an ML algorithm
   - Itself, as a target for learning
   - Surrogates for optimization
   - Economic applications

Graphical Models and fast MAP Inference

- Given distribution $p(x) = \frac{1}{Z}\exp(-E(x))$ where $E(x) = \sum_{c \in \mathcal{C}} E_c(x_c)$ and $\mathcal{C}$ are the cliques of a graph $G = (V, \mathcal{E})$.

Graphical Models and fast MAP Inference

- Given distribution $p(x) = \frac{1}{Z}\exp(-E(x))$ where $E(x) = \sum_{c \in \mathcal{C}} E_c(x_c)$ and $\mathcal{C}$ are the cliques of a graph $G = (V, \mathcal{E})$.
- MAP inference problem is important in ML: compute

$$x^* \in \operatorname*{argmax}_{x \in \{0,1\}^V} p(x) \tag{24}$$

Graphical Models and fast MAP Inference

- Given distribution $p(x) = \frac{1}{Z}\exp(-E(x))$ where $E(x) = \sum_{c \in \mathcal{C}} E_c(x_c)$ and $\mathcal{C}$ are the cliques of a graph $G = (V, \mathcal{E})$.
- MAP inference problem is important in ML: compute

$$x^* \in \operatorname*{argmax}_{x \in \{0,1\}^V} p(x) \tag{24}$$

- Easy when G a tree, exponential in k (tree-width of G) in general.

Graphical Models and fast MAP Inference

- Given distribution $p(x) = \frac{1}{Z}\exp(-E(x))$ where $E(x) = \sum_{c \in \mathcal{C}} E_c(x_c)$ and $\mathcal{C}$ are the cliques of a graph $G = (V, \mathcal{E})$.
- MAP inference problem is important in ML: compute

$$x^* \in \operatorname*{argmax}_{x \in \{0,1\}^V} p(x) \tag{24}$$

- Easy when G a tree, exponential in k (tree-width of G) in general.
- NP-hard to find the tree-width.

Graphical Models and fast MAP Inference

- Given distribution $p(x) = \frac{1}{Z}\exp(-E(x))$ where $E(x) = \sum_{c \in \mathcal{C}} E_c(x_c)$ and $\mathcal{C}$ are the cliques of a graph $G = (V, \mathcal{E})$.
- MAP inference problem is important in ML: compute

$$x^* \in \operatorname*{argmax}_{x \in \{0,1\}^V} p(x) \tag{24}$$

- Easy when G a tree, exponential in k (tree-width of G) in general.
- NP-hard to find the tree-width.
- Tree-width can be large even when degree is two (i.e., $E(x) = \sum_{e \in \mathcal{E}} E_e(x_e)$ is a sum over edges).

Graphical Models and fast MAP Inference

- Given distribution $p(x) = \frac{1}{Z}\exp(-E(x))$ where $E(x) = \sum_{c \in \mathcal{C}} E_c(x_c)$ and $\mathcal{C}$ are the cliques of a graph $G = (V, \mathcal{E})$.
- MAP inference problem is important in ML: compute

$$x^* \in \operatorname*{argmax}_{x \in \{0,1\}^V} p(x) \tag{24}$$

- Easy when G a tree, exponential in k (tree-width of G) in general.
- NP-hard to find the tree-width.
- Tree-width can be large even when degree is two (i.e., $E(x) = \sum_{e \in \mathcal{E}} E_e(x_e)$ is a sum over edges).
- Many approximate inference strategies utilize additional factorization assumptions to make inference tractable (e.g., mean-field, variational inference, expectation propagation, etc).

Graphical Models and fast MAP Inference

- Given distribution $p(x) = \frac{1}{Z}\exp(-E(x))$ where $E(x) = \sum_{c \in \mathcal{C}} E_c(x_c)$ and $\mathcal{C}$ are the cliques of a graph $G = (V, \mathcal{E})$.
- MAP inference problem is important in ML: compute

$$x^* \in \operatorname*{argmax}_{x \in \{0,1\}^V} p(x) \tag{24}$$

- Easy when G a tree, exponential in k (tree-width of G) in general.
- NP-hard to find the tree-width.
- Tree-width can be large even when degree is two (i.e., $E(x) = \sum_{e \in \mathcal{E}} E_e(x_e)$ is a sum over edges).
- Many approximate inference strategies utilize additional factorization assumptions to make inference tractable (e.g., mean-field, variational inference, expectation propagation, etc).
- However, what if we could do MAP inference in polynomial time regardless of the tree-width, and without even knowing the tree-width?

Degree two (edge) graphical models

- Given G restrict $p \in \mathcal{F}(G, R^{(\mathrm{f})})$ such that we can write the global energy $E(x)$ as a sum of unary and pairwise potentials:

$$E(x) = \sum_{v \in V(G)} e_v(x_v) + \sum_{(i,j) \in \mathcal{E}(G)} e_{ij}(x_i, x_j) \tag{25}$$

Degree two (edge) graphical models

- Given G restrict $p \in \mathcal{F}(G, R^{(\mathrm{f})})$ such that we can write the global energy $E(x)$ as a sum of unary and pairwise potentials:

$$E(x) = \sum_{v \in V(G)} e_v(x_v) + \sum_{(i,j) \in \mathcal{E}(G)} e_{ij}(x_i, x_j) \tag{25}$$

- $e_v(x_v)$ and $e_{ij}(x_i, x_j)$ are like local energy potentials.

Degree two (edge) graphical models

- Given G restrict $p \in \mathcal{F}(G, R^{(\mathrm{f})})$ such that we can write the global energy $E(x)$ as a sum of unary and pairwise potentials:

$$E(x) = \sum_{v \in V(G)} e_v(x_v) + \sum_{(i,j) \in \mathcal{E}(G)} e_{ij}(x_i, x_j) \tag{25}$$

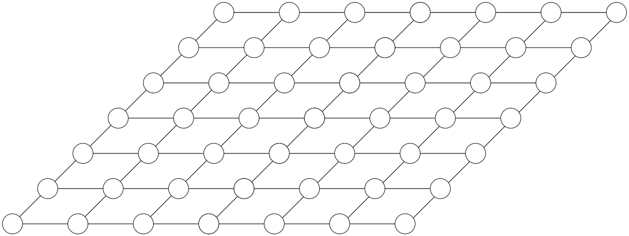
- $e_v(x_v)$ and $e_{ij}(x_i, x_j)$ are like local energy potentials.
- Since $\log p(x) = -E(x) + \text{const.}$, the smaller $e_v(x_v)$ or $e_{ij}(x_i, x_j)$ become, the higher the probability becomes.

Degree two (edge) graphical models

- Given G restrict $p \in \mathcal{F}(G, R^{(\mathrm{f})})$ such that we can write the global energy $E(x)$ as a sum of unary and pairwise potentials:

$$E(x) = \sum_{v \in V(G)} e_v(x_v) + \sum_{(i,j) \in \mathcal{E}(G)} e_{ij}(x_i, x_j) \tag{25}$$

- $e_v(x_v)$ and $e_{ij}(x_i, x_j)$ are like local energy potentials.
- Since $\log p(x) = -E(x) + \text{const.}$, the smaller $e_v(x_v)$ or $e_{ij}(x_i, x_j)$ become, the higher the probability becomes.
- When G is a 2D grid graph, we have

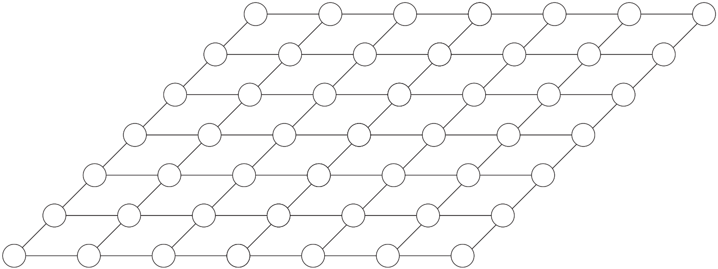


Auxiliary (s,t)-graph

- We can create auxiliary graph that involves two new terminal nodes s and t (source and sink) and connect each of s and t to all of the original nodes.
- I.e., $G_a = (V \cup \{s,t\}, E + \cup_{v\in V}((s,v) \cup (v,t)))$.

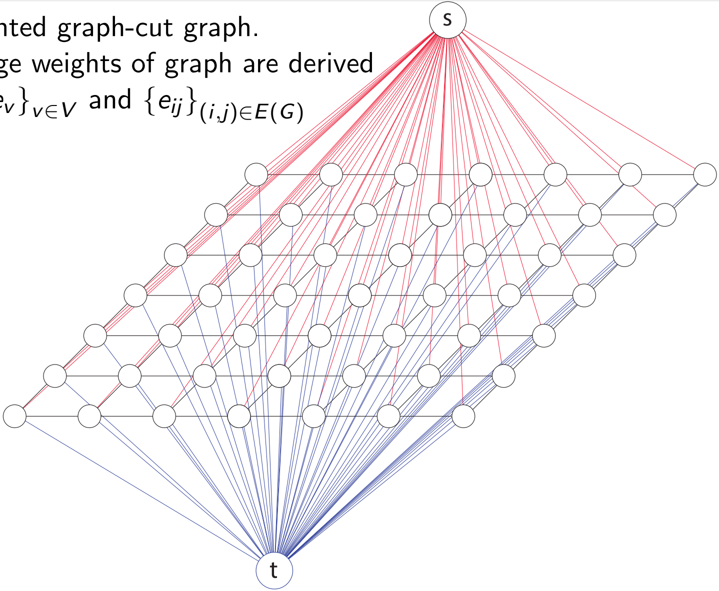
Transformation from graphical model to auxiliary graph

Original Graph: $E(x) = \sum_{v \in V(G)} e_v(x_v) + \sum_{(i,j) \in E(G)} e_{ij}(x_i, x_j)$



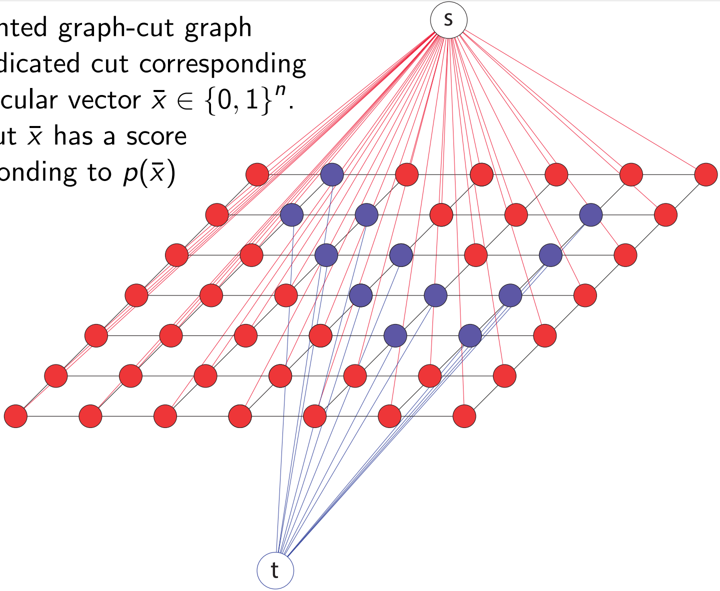
Transformation from graphical model to auxiliary graph

Augmented graph-cut graph.
The edge weights of graph are derived from $\{e_v\}_{v \in V}$ and $\{e_{ij}\}_{(i,j) \in E(G)}$



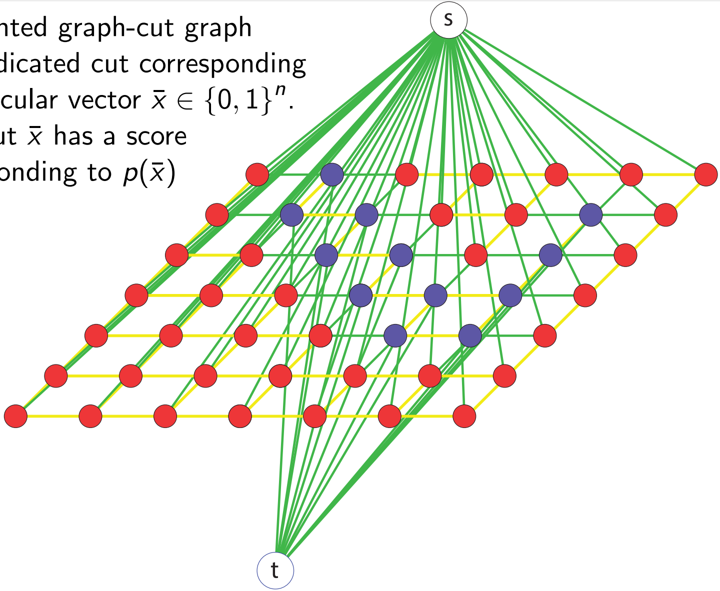
Transformation from graphical model to auxiliary graph

Augmented graph-cut graph with indicated cut corresponding to particular vector $\bar{x} \in \{0, 1\}^n$. Each cut $\bar{x}$ has a score corresponding to $p(\bar{x})$



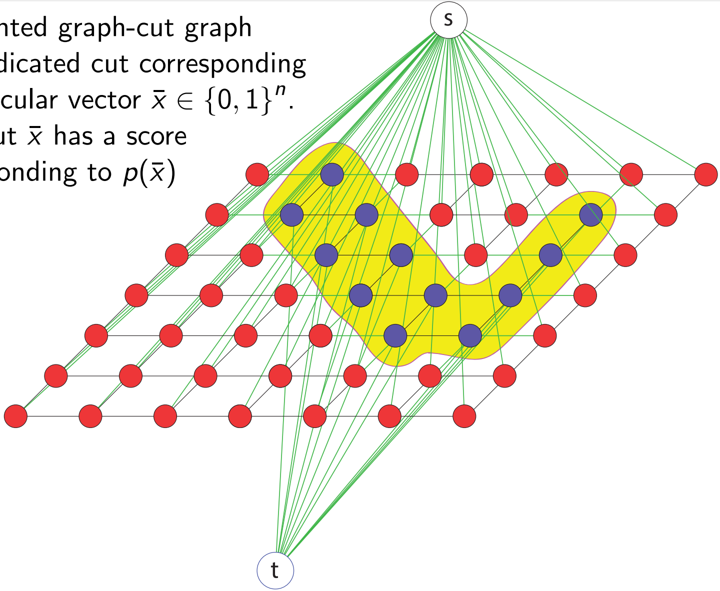
Transformation from graphical model to auxiliary graph

Augmented graph-cut graph with indicated cut corresponding to particular vector $\bar{x} \in \{0,1\}^n$. Each cut $\bar{x}$ has a score corresponding to $p(\bar{x})$



Transformation from graphical model to auxiliary graph

Augmented graph-cut graph with indicated cut corresponding to particular vector $\bar{x} \in \{0,1\}^n$. Each cut $\bar{x}$ has a score corresponding to $p(\bar{x})$



Setting of the weights in the auxiliary cut graph

- Any graph cut corresponds to a vector $\bar{x} \in \{0, 1\}^n$.

Setting of the weights in the auxiliary cut graph

- Any graph cut corresponds to a vector $\bar{x} \in \{0, 1\}^n$.
- If weights are set correctly in the cut graph, and if edge functions e_{ij} satisfy certain properties, then graph-cut score corresponding to $\bar{x}$ can be made equivalent to $E(x) = \log p(\bar{x}) + \text{const.}$.

Setting of the weights in the auxiliary cut graph

- Any graph cut corresponds to a vector $\bar{x} \in \{0,1\}^n$.
- If weights are set correctly in the cut graph, and if edge functions e_{ij} satisfy certain properties, then graph-cut score corresponding to $\bar{x}$ can be made equivalent to $E(x) = \log p(\bar{x}) + \text{const.}$.
- If weights of all edges, except those involving terminals s and t, are non-negative, graph cut computable in polynomial time via max-flow!

Setting of the weights in the auxiliary cut graph

- Any graph cut corresponds to a vector $\bar{x} \in \{0, 1\}^n$.
- If weights are set correctly in the cut graph, and if edge functions e_{ij} satisfy certain properties, then graph-cut score corresponding to $\bar{x}$ can be made equivalent to $E(x) = \log p(\bar{x}) + \text{const.}$.
- If weights of all edges, except those involving terminals s and t, are non-negative, graph cut computable in polynomial time via max-flow!
- Hence, poly time graph cut, can find the optimal MPE assignment, regardless of the graphical model's tree-width!

Setting of the weights in the auxiliary cut graph

- Any graph cut corresponds to a vector $\bar{x} \in \{0, 1\}^n$.
- If weights are set correctly in the cut graph, and if edge functions e_{ij} satisfy certain properties, then graph-cut score corresponding to $\bar{x}$ can be made equivalent to $E(x) = \log p(\bar{x}) + \text{const.}$.
- If weights of all edges, except those involving terminals s and t, are non-negative, graph cut computable in polynomial time via max-flow!
- Hence, poly time graph cut, can find the optimal MPE assignment, regardless of the graphical model's tree-width!

Setting of the weights in the auxiliary cut graph

- Any graph cut corresponds to a vector $\bar{x} \in \{0,1\}^n$.
- If weights are set correctly in the cut graph, and if edge functions e_{ij} satisfy certain properties, then graph-cut score corresponding to $\bar{x}$ can be made equivalent to $E(x) = \log p(\bar{x}) + \text{const.}$.
- If weights of all edges, except those involving terminals s and t, are non-negative, graph cut computable in polynomial time via max-flow!
- Hence, poly time graph cut, can find the optimal MPE assignment, regardless of the graphical model's tree-width!

Edge weight assignments:

- For (s, v) with $v \in V(G)$, set edge $w_{s,v} = (e_v(1) - e_v(0))\mathbf{1}(e_v(1) > e_v(0))$

Setting of the weights in the auxiliary cut graph

- Any graph cut corresponds to a vector $\bar{x} \in \{0, 1\}^n$.
- If weights are set correctly in the cut graph, and if edge functions e_{ij} satisfy certain properties, then graph-cut score corresponding to $\bar{x}$ can be made equivalent to $E(x) = \log p(\bar{x}) + \text{const.}$.
- If weights of all edges, except those involving terminals s and t, are non-negative, graph cut computable in polynomial time via max-flow!
- Hence, poly time graph cut, can find the optimal MPE assignment, regardless of the graphical model's tree-width!

Edge weight assignments:

- For (s, v) with $v \in V(G)$, set edge $w_{s,v} = (e_v(1) - e_v(0))\mathbf{1}(e_v(1) > e_v(0))$
- For (v, t) with $v \in V(G)$, set edge $w_{v,t} = (e_v(0) - e_v(1))\mathbf{1}(e_v(0) \geq e_v(1))$

Setting of the weights in the auxiliary cut graph

- Any graph cut corresponds to a vector $\bar{x} \in \{0,1\}^n$.
- If weights are set correctly in the cut graph, and if edge functions e_{ij} satisfy certain properties, then graph-cut score corresponding to $\bar{x}$ can be made equivalent to $E(x) = \log p(\bar{x}) + \text{const.}$.
- If weights of all edges, except those involving terminals s and t, are non-negative, graph cut computable in polynomial time via max-flow!
- Hence, poly time graph cut, can find the optimal MPE assignment, regardless of the graphical model's tree-width!

Edge weight assignments:

- For (s, v) with $v \in V(G)$, set edge $w_{s,v} = (e_v(1) - e_v(0))\mathbf{1}(e_v(1) > e_v(0))$
- For (v, t) with $v \in V(G)$, set edge $w_{v,t} = (e_v(0) - e_v(1))\mathbf{1}(e_v(0) \geq e_v(1))$
- For original edge $(i, j) \in E$, $i, j \in V$, set weight $w_{i,j} = e_{ij}(1,0) + e_{ij}(0,1) - e_{ij}(1,1) - e_{ij}(0,0)$.

Submodular potentials

- Edge functions must be submodular (equivalently "associative", "attractive", "regular", "Potts", or "ferromagnetic") for this to work, i.e., for all $(i,j) \in E(G)$, we must have that:

$$e_{ij}(0,1) + e_{ij}(1,0) \geq e_{ij}(1,1) + e_{ij}(0,0) \tag{26}$$

Submodular potentials

- Edge functions must be submodular (equivalently "associative", "attractive", "regular", "Potts", or "ferromagnetic") for this to work, i.e., for all $(i, j) \in E(G)$, we must have that:

$$e_{ij}(0,1) + e_{ij}(1,0) \geq e_{ij}(1,1) + e_{ij}(0,0) \tag{26}$$

- This means: on average, preservation is preferred over change.

Submodular potentials

- Edge functions must be submodular (equivalently "associative", "attractive", "regular", "Potts", or "ferromagnetic") for this to work, i.e., for all $(i, j) \in E(G)$, we must have that:

$$e_{ij}(0,1) + e_{ij}(1,0) \geq e_{ij}(1,1) + e_{ij}(0,0) \tag{26}$$

- This means: on average, preservation is preferred over change.
- As a set function, this is the same as:

$$f(X) = \sum_{\{i,j\} \in \mathcal{E}(G)} f_{i,j}(X \cap \{i,j\}) \tag{27}$$

which is submodular if each of the $f_{i,j}$'s are submodular!

Submodular potentials

- Edge functions must be submodular (equivalently "associative", "attractive", "regular", "Potts", or "ferromagnetic") for this to work, i.e., for all $(i,j) \in E(G)$, we must have that:

$$e_{ij}(0,1) + e_{ij}(1,0) \geq e_{ij}(1,1) + e_{ij}(0,0) \tag{26}$$

- This means: on average, preservation is preferred over change.
- As a set function, this is the same as:

$$f(X) = \sum_{\{i,j\} \in \mathcal{E}(G)} f_{i,j}(X \cap \{i,j\}) \tag{27}$$

  which is submodular if each of the $f_{i,j}$'s are submodular!
- Probability form $p(x) \propto \prod \psi$, so $\psi_{ij}(1,0)\psi_{ij}(0,1) \leq \psi_{ij}(0,0)\psi_{ij}(1,1)$: geometric mean of factor scores higher when neighboring pixels have the same value - a reasonable assumption about natural scenes and signals.

Submodular potentials

- Edge functions must be submodular (equivalently "associative", "attractive", "regular", "Potts", or "ferromagnetic") for this to work, i.e., for all $(i, j) \in E(G)$, we must have that:

$$e_{ij}(0,1) + e_{ij}(1,0) \geq e_{ij}(1,1) + e_{ij}(0,0) \tag{26}$$

- This means: on average, preservation is preferred over change.
- As a set function, this is the same as:

$$f(X) = \sum_{\{i,j\} \in \mathcal{E}(G)} f_{i,j}(X \cap \{i,j\}) \tag{27}$$

  which is submodular if each of the $f_{i,j}$'s are submodular!
- Probability form $p(x) \propto \prod \psi$, so $\psi_{ij}(1,0)\psi_{ij}(0,1) \leq \psi_{ij}(0,0)\psi_{ij}(1,1)$: geometric mean of factor scores higher when neighboring pixels have the same value - a reasonable assumption about natural scenes and signals.
- Weights w_{ij} in s,t-graph above are always non-negative, so graph-cut solvable.

On log-supermodular vs. log-submodular distributions

- Log-supermodular distributions.

$$\log \Pr(x) = f(x) + \text{const.} = -E(x) + \text{const.} \tag{28}$$

where f is supermodular ($E(x)$ is submodular). MAP (or high-probable) assignments should be "regular", "homogeneous", "smooth", "simple". E.g., attractive potentials in computer vision, ferromagnetic Potts models statistical physics.

On log-supermodular vs. log-submodular distributions

- Log-supermodular distributions.

$$\log \Pr(x) = f(x) + \text{const.} = -E(x) + \text{const.} \tag{28}$$

where f is supermodular ($E(x)$ is submodular). MAP (or high-probable) assignments should be "regular", "homogeneous", "smooth", "simple". E.g., attractive potentials in computer vision, ferromagnetic Potts models statistical physics.

- Log-submodular distributions:

$$\log \Pr(x) = f(x) + \text{const.} \tag{29}$$

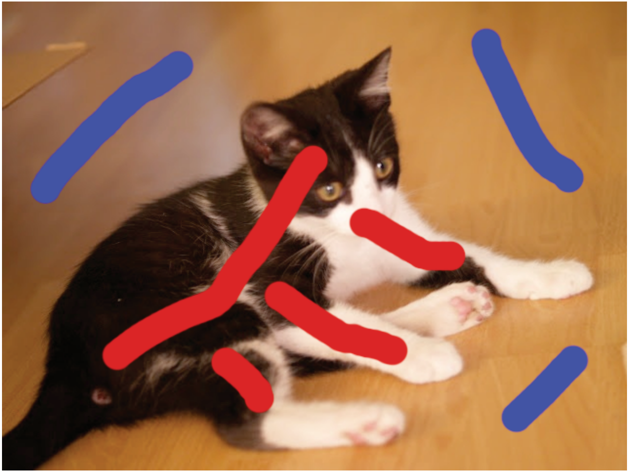
where f is submodular. MAP or high-probable assignments should be "diverse", or "complex", or "covering", like in determinantal point processes.

Submodular potentials in GMs: Image Segmentation

- an image needing to be segmented.

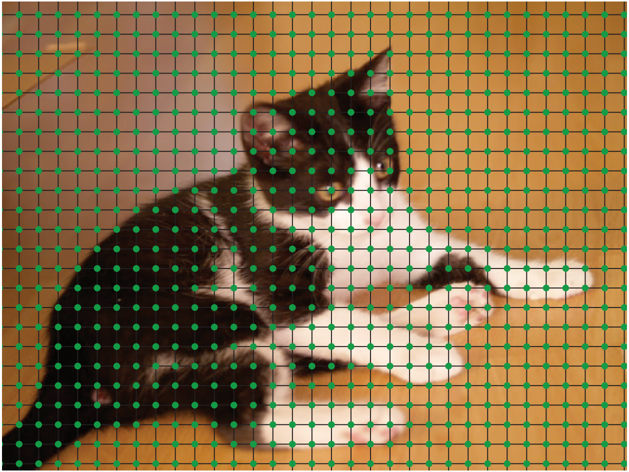
Submodular potentials in GMs: Image Segmentation

- labeled data, some pixels being marked foreground (red) and others marked background (blue) to train the unaries $\{e_v(x_v)\}_{v \in V}$.



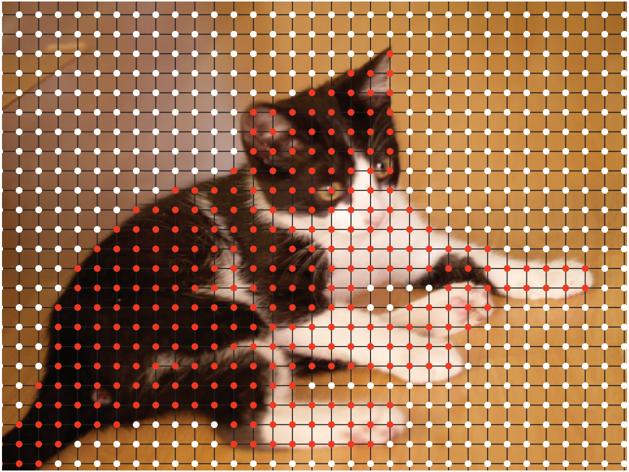
Submodular potentials in GMs: Image Segmentation

- Set of a graph over the image, graph shows binary pixel labels.



Submodular potentials in GMs: Image Segmentation

- Run graph-cut to segment the image, foreground in red, background in white.



Submodular potentials in GMs: Image Segmentation

- the foreground is removed from the background.

Shrinking bias in graph cut image segmentation

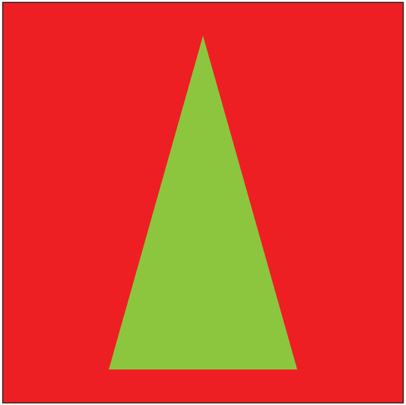
What does graph-cut based image segmentation do with elongated structures (top) or contrast gradients (bottom)?

Shrinking bias in graph cut image segmentation



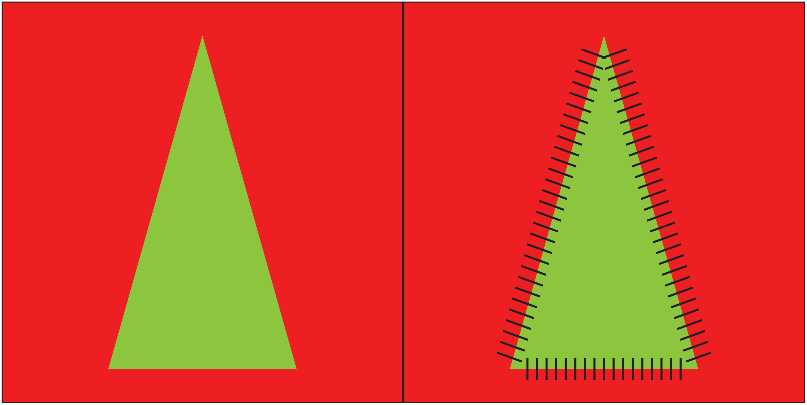

Shrinking bias in image segmentation

- An image needing to be segmented
- Clear high-contrast boundaries



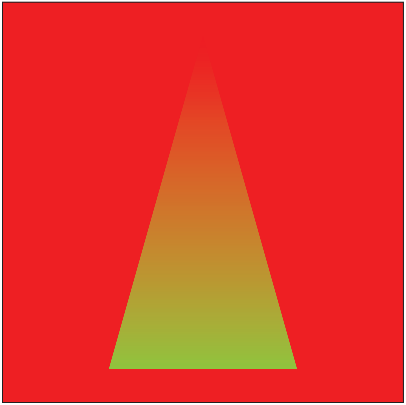
Shrinking bias in image segmentation

- Graph-cut (MRF with submodular edge potentials) works well.



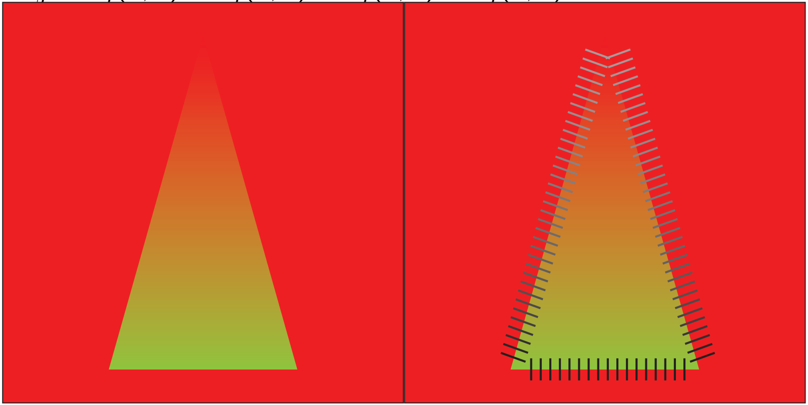
Shrinking bias in image segmentation

- Now with contrast gradient (less clear segment as we move up).
- The "elongated structure" also poses a challenge.



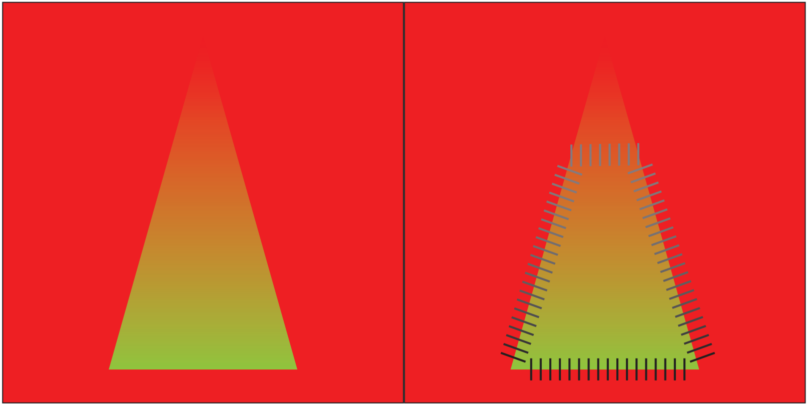
Shrinking bias in image segmentation

- Unary potentials $\{e_v(x_v)\}_{v \in V}$ prefer a different segmentation.
- Edge weights are the same regardless of where they are $w_{i,j} = e_{ij}(1,0) + e_{ij}(0,1) - e_{ij}(1,1) - e_{ij}(0,0) \geq 0$.



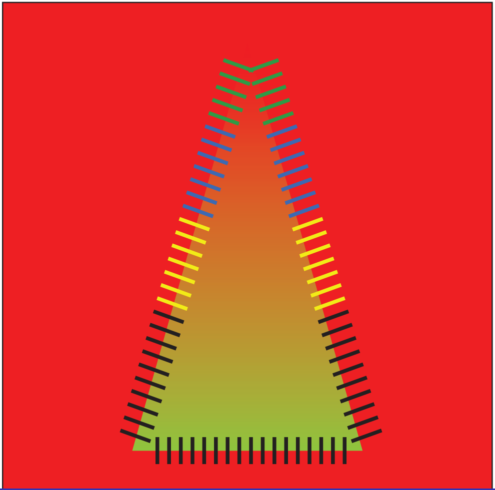
Shrinking bias in image segmentation

- And the shrinking bias occurs, truncating the segmentation since it results in lower energy.



Shrinking bias in image segmentation

- With "typed" edges, we can have cut cost be sum of edge color weights, not sum of edge weights.
- Submodularity to the rescue: balls & urns.



Addressing shrinking bias with edge submodularity

- Standard graph cut, uses a modular function $w : 2^E \to \mathbb{R}_+$ defined on the edges to measure cut costs. Graph cut node function is submodular.

$$f_w(X) = w\Big(\{(u, v) \in E : u \in X, v \in V \setminus X\}\Big) \tag{30}$$

Addressing shrinking bias with edge submodularity

- Standard graph cut, uses a modular function $w : 2^E \to \mathbb{R}_+$ defined on the edges to measure cut costs. Graph cut node function is submodular.

$$f_w(X) = w\Big(\{(u, v) \in E : u \in X, v \in V \setminus X\}\Big) \tag{30}$$

- Instead, we can use a submodular function $g : 2^E \to \mathbb{R}_+$ defined on the edges to express cooperative costs.

$$f_g(X) = g\Big(\{(u, v) \in E : u \in X, v \in V \setminus X\}\Big) \tag{31}$$

Addressing shrinking bias with edge submodularity

- Standard graph cut, uses a modular function $w : 2^E \to \mathbb{R}_+$ defined on the edges to measure cut costs. Graph cut node function is submodular.

$$f_w(X) = w\Big(\{(u, v) \in E : u \in X, v \in V \setminus X\}\Big) \tag{30}$$

- Instead, we can use a submodular function $g : 2^E \to \mathbb{R}_+$ defined on the edges to express cooperative costs.

$$f_g(X) = g\Big(\{(u, v) \in E : u \in X, v \in V \setminus X\}\Big) \tag{31}$$

- Seen as a node function, $f_g : 2^V \to \mathbb{R}_+$ is not submodular, but it uses submodularity internally to solve the shrinking bias problem.

Addressing shrinking bias with edge submodularity

- Standard graph cut, uses a modular function $w : 2^E \to \mathbb{R}_+$ defined on the edges to measure cut costs. Graph cut node function is submodular.

$$f_w(X) = w\Big(\{(u,v) \in E : u \in X, v \in V \setminus X\}\Big) \qquad (30)$$

- Instead, we can use a submodular function $g : 2^E \to \mathbb{R}_+$ defined on the edges to express cooperative costs.

$$f_g(X) = g\Big(\{(u,v) \in E : u \in X, v \in V \setminus X\}\Big) \qquad (31)$$

- Seen as a node function, $f_g : 2^V \to \mathbb{R}_+$ is not submodular, but it uses submodularity internally to solve the shrinking bias problem.
- $\Rightarrow$ cooperative-cut (Jegelka & Bilmes, 2011).

Graph-cut vs. cooperative-cut comparisons

Graph Cut Cooperative Cut

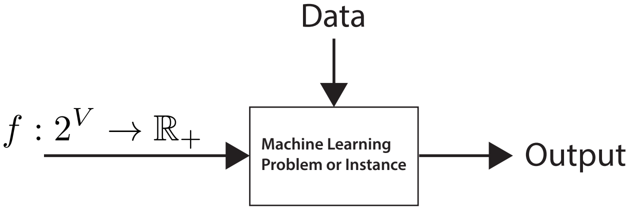
(Jegelka&Bilmes,'11). There are fast algorithms for solving as well (as we'll see tomorrow).

Outline

1. Introduction
2. Basics
3. Submodular Applications in ML
   - Where is submodularity useful?
   - Traditional combinatorial problems
   - As a model of diversity, coverage, span, or information
   - As a model of cooperative costs, complexity, roughness, and irregularity
   - As a parameter for an ML algorithm
   - Itself, as a target for learning
   - Surrogates for optimization
   - Economic applications

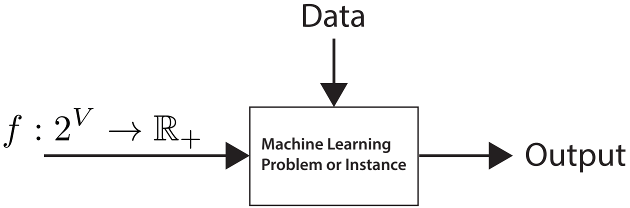
A submodular function as a parameter

- In some cases, it may be useful to view a submodular function $f : 2^V \to \mathbb{R}$ as a input "parameter" to a machine learning algorithm.



A submodular function as a parameter

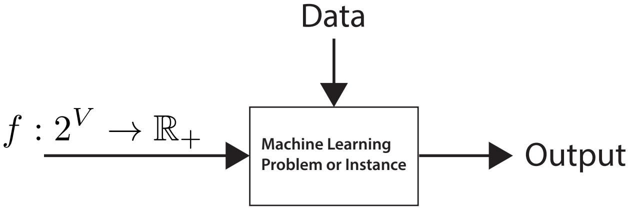
- In some cases, it may be useful to view a submodular function $f : 2^V \to \mathbb{R}$ as a input "parameter" to a machine learning algorithm.



- A given submodular function $f \in \mathbb{S} \subseteq \mathbb{R}^{2^n}$ can be seen as a vector in a 2^n-dimensional compact cone.

A submodular function as a parameter

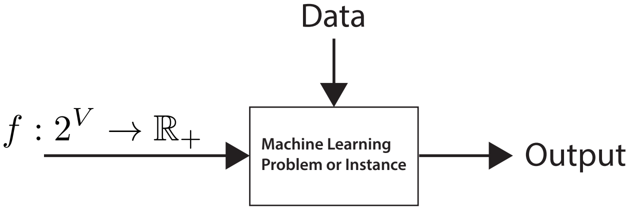
- In some cases, it may be useful to view a submodular function $f : 2^V \to \mathbb{R}$ as a input "parameter" to a machine learning algorithm.



- A given submodular function $f \in \mathbb{S} \subseteq \mathbb{R}^{2^n}$ can be seen as a vector in a 2^n-dimensional compact cone.
- $\mathbb{S}$ is a submodular cone since submodularity is closed under non-negative (conic) combinations.

A submodular function as a parameter

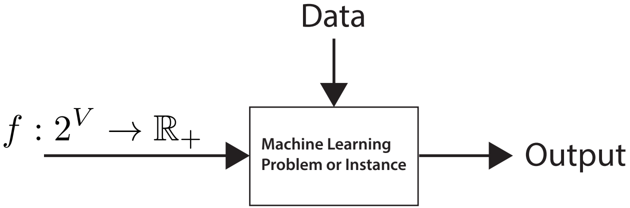
- In some cases, it may be useful to view a submodular function $f : 2^V \to \mathbb{R}$ as a input "parameter" to a machine learning algorithm.



- A given submodular function $f \in \mathbb{S} \subseteq \mathbb{R}^{2^n}$ can be seen as a vector in a 2^n-dimensional compact cone.
- $\mathbb{S}$ is a submodular cone since submodularity is closed under non-negative (conic) combinations.
- 2^n-dimensional since for certain $f \in \mathbb{S}$, there exists $f_\epsilon \in \mathbb{R}^{2^n}$ having no zero elements with $f + f_\epsilon \in \mathbb{S}$.

A submodular function as a parameter

- In some cases, it may be useful to view a submodular function $f : 2^V \to \mathbb{R}$ as a input "parameter" to a machine learning algorithm.



- A given submodular function $f \in \mathbb{S} \subseteq \mathbb{R}^{2^n}$ can be seen as a vector in a 2^n-dimensional compact cone.
- $\mathbb{S}$ is a submodular cone since submodularity is closed under non-negative (conic) combinations.
- 2^n-dimensional since for certain $f \in \mathbb{S}$, there exists $f_\epsilon \in \mathbb{R}^{2^n}$ having no zero elements with $f + f_\epsilon \in \mathbb{S}$.
- We next see how f parameterizes problems in ML, and then address learning.

Supervised And Unsupervised Machine Learning

- Given training data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^m$ with $(x_i, y_i) \in \mathbb{R}^n \times \mathbb{R}$, perform the following risk minimization problem:

$$\min_{w \in \mathbb{R}^n} \frac{1}{m} \sum_{i=1}^m \ell(y_i, w^\intercal x_i) + \lambda \Omega(w), \tag{32}$$

where $\ell(\cdot)$ is a loss function (e.g., squared error) and $\Omega(w)$ is a norm.

Supervised And Unsupervised Machine Learning

- Given training data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^m$ with $(x_i, y_i) \in \mathbb{R}^n \times \mathbb{R}$, perform the following risk minimization problem:

$$\min_{w \in \mathbb{R}^n} \frac{1}{m} \sum_{i=1}^m \ell(y_i, w^\intercal x_i) + \lambda \Omega(w), \tag{32}$$

where $\ell(\cdot)$ is a loss function (e.g., squared error) and $\Omega(w)$ is a norm.

- When data has multiple responses $(x_i, y_i) \in \mathbb{R}^n \times \mathbb{R}^k$, learning becomes:

$$\min_{w^1,\dots,w^k \in \mathbb{R}^n} \sum_{j=1}^k \frac{1}{m} \sum_{i=1}^m \ell(y_i^k, (w^k)^\intercal x_i) + \lambda \Omega(w^k), \tag{33}$$

Supervised And Unsupervised Machine Learning

- Given training data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^m$ with $(x_i, y_i) \in \mathbb{R}^n \times \mathbb{R}$, perform the following risk minimization problem:

$$\min_{w \in \mathbb{R}^n} \frac{1}{m} \sum_{i=1}^{m} \ell(y_i, w^\intercal x_i) + \lambda \Omega(w), \tag{32}$$

where $\ell(\cdot)$ is a loss function (e.g., squared error) and $\Omega(w)$ is a norm.

- When data has multiple responses $(x_i, y_i) \in \mathbb{R}^n \times \mathbb{R}^k$, learning becomes:

$$\min_{w^1, \dots, w^k \in \mathbb{R}^n} \sum_{j=1}^{k} \frac{1}{m} \sum_{i=1}^{m} \ell(y_i^k, (w^k)^\intercal x_i) + \lambda \Omega(w^k), \tag{33}$$

- When data has multiple responses only that are observed, $(y_i) \in R^k$ we get dictionary learning (Krause & Guestrin, Das & Kempe):

$$\min_{x_1, \dots, x_m} \min_{w^1, \dots, w^k \in \mathbb{R}^n} \sum_{j=1}^{k} \frac{1}{m} \sum_{i=1}^{m} \ell(y_i^k, (w^k)^\intercal x_i) + \lambda \Omega(w^k), \tag{34}$$

Norms, sparse norms, and computer vision

- Common norms include p-norm $\Omega(w) = \|w\|_p = \left(\sum_{i=1}^p w_i^p\right)^{1/p}$
- 1-norm promotes sparsity (prefer solutions with zero entries).
- Image denoising, total variation is useful, norm takes form:

$$\Omega(w) = \sum_{i=2}^{N} |w_i - w_{i-1}| \tag{35}$$

- Points of difference should be "sparse" (frequently zero).

(Rodriguez, 2009)

Submodular parameterization of a sparse convex norm

- Prefer convex norms since they can be solved.

Submodular parameterization of a sparse convex norm

- Prefer convex norms since they can be solved.
- For $w \in \mathbb{R}^V$, $\text{supp}(w) \in \{0,1\}^V$ has $\text{supp}(w)(v) = 1$ iff $w(v) > 0$

Submodular parameterization of a sparse convex norm

- Prefer convex norms since they can be solved.
- For $w \in \mathbb{R}^V$, $\text{supp}(w) \in \{0,1\}^V$ has $\text{supp}(w)(v) = 1$ iff $w(v) > 0$
- Given submodular function $f : 2^V \to \mathbb{R}_+$, $f(\text{supp}(w))$ measures the "complexity" of the non-zero pattern of w; can have more non-zero values if they cooperate (via f) with other non-zero values.

Submodular parameterization of a sparse convex norm

- Prefer convex norms since they can be solved.
- For $w \in \mathbb{R}^V$, $\text{supp}(w) \in \{0,1\}^V$ has $\text{supp}(w)(v) = 1$ iff $w(v) > 0$
- Given submodular function $f : 2^V \to \mathbb{R}_+$, $f(\text{supp}(w))$ measures the "complexity" of the non-zero pattern of w; can have more non-zero values if they cooperate (via f) with other non-zero values.
- $f(\text{supp}(w))$ is hard to optimize, but it's convex envelope $\tilde{f}(|w|)$ (i.e., largest convex under-estimator of $f(\text{supp}(w))$) is obtained via the Lovász-extension $\tilde{f}$ of f (Bolton et al. 2008, Bach 2010).

Submodular parameterization of a sparse convex norm

- Prefer convex norms since they can be solved.
- For $w \in \mathbb{R}^V$, $\text{supp}(w) \in \{0,1\}^V$ has $\text{supp}(w)(v) = 1$ iff $w(v) > 0$
- Given submodular function $f : 2^V \to \mathbb{R}_+$, $f(\text{supp}(w))$ measures the "complexity" of the non-zero pattern of w; can have more non-zero values if they cooperate (via f) with other non-zero values.
- $f(\text{supp}(w))$ is hard to optimize, but it's convex envelope $\tilde{f}(|w|)$ (i.e., largest convex under-estimator of $f(\text{supp}(w))$) is obtained via the Lovász-extension $\tilde{f}$ of f (Bolton et al. 2008, Bach 2010).
- Submodular functions thus parameterize structured convex sparse norms via the Lovász-extension!

Submodular parameterization of a sparse convex norm

- Prefer convex norms since they can be solved.
- For $w \in \mathbb{R}^V$, $\text{supp}(w) \in \{0,1\}^V$ has $\text{supp}(w)(v) = 1$ iff $w(v) > 0$
- Given submodular function $f : 2^V \to \mathbb{R}_+$, $f(\text{supp}(w))$ measures the "complexity" of the non-zero pattern of w; can have more non-zero values if they cooperate (via f) with other non-zero values.
- $f(\text{supp}(w))$ is hard to optimize, but it's convex envelope $\tilde{f}(|w|)$ (i.e., largest convex under-estimator of $f(\text{supp}(w))$) is obtained via the Lovász-extension $\tilde{f}$ of f (Bolton et al. 2008, Bach 2010).
- Submodular functions thus parameterize structured convex sparse norms via the Lovász-extension!
- The Lovász-extension (Lovász '82, Edmonds '70) is easy to get via the greedy algorithm: sort $w_{\sigma_1} \geq w_{\sigma_2} \geq \cdots \geq w_{\sigma_n}$, then

$$\tilde{f}(w) = \sum_{i=1}^{n} w_{\sigma_i}\left(f(\sigma_1, \ldots, \sigma_i) - f(\sigma_1, \ldots, \sigma_{i-1})\right) \tag{36}$$

Submodular parameterization of a sparse convex norm

- Prefer convex norms since they can be solved.
- For $w \in \mathbb{R}^V$, $\text{supp}(w) \in \{0,1\}^V$ has $\text{supp}(w)(v) = 1$ iff $w(v) > 0$
- Given submodular function $f : 2^V \to \mathbb{R}_+$, $f(\text{supp}(w))$ measures the "complexity" of the non-zero pattern of w; can have more non-zero values if they cooperate (via f) with other non-zero values.
- $f(\text{supp}(w))$ is hard to optimize, but it's convex envelope $\tilde{f}(|w|)$ (i.e., largest convex under-estimator of $f(\text{supp}(w))$) is obtained via the Lovász-extension $\tilde{f}$ of f (Bolton et al. 2008, Bach 2010).
- Submodular functions thus parameterize structured convex sparse norms via the Lovász-extension!
- The Lovász-extension (Lovász '82, Edmonds '70) is easy to get via the greedy algorithm: sort $w_{\sigma_1} \geq w_{\sigma_2} \geq \cdots \geq w_{\sigma_n}$, then

$$\tilde{f}(w) = \sum_{i=1}^{n} w_{\sigma_i}(f(\sigma_1, \ldots, \sigma_i) - f(\sigma_1, \ldots, \sigma_{i-1})) \tag{36}$$

- Ex: total variation is the Lovász-extension of graph cut

Submodular Generalized Dependence

- there is a notion of "independence" , i.e., $A \perp\!\!\!\perp B$:

$$f(A \cup B) = f(A) + f(B), \tag{37}$$

Submodular Generalized Dependence

- there is a notion of "independence" , i.e., $A \perp\!\!\!\perp B$:

$$f(A \cup B) = f(A) + f(B), \tag{37}$$

- and a notion of "conditional independence" , i.e., $A \perp\!\!\!\perp B | C$:

$$f(A \cup B \cup C) + f(C) = f(A \cup C) + f(B \cup C) \tag{38}$$

Submodular Generalized Dependence

- there is a notion of "independence" , i.e., $A \perp\!\!\!\perp B$:

$$f(A \cup B) = f(A) + f(B), \tag{37}$$

- and a notion of "conditional independence" , i.e., $A \perp\!\!\!\perp B | C$:

$$f(A \cup B \cup C) + f(C) = f(A \cup C) + f(B \cup C) \tag{38}$$

- and a notion of "dependence" (conditioning reduces valuation):

$$f(A|B) \triangleq f(A \cup B) - f(B) < f(A), \tag{39}$$

Submodular Generalized Dependence

- there is a notion of "independence" , i.e., $A \perp\!\!\!\perp B$:

$$f(A \cup B) = f(A) + f(B), \tag{37}$$

- and a notion of "conditional independence" , i.e., $A \perp\!\!\!\perp B | C$:

$$f(A \cup B \cup C) + f(C) = f(A \cup C) + f(B \cup C) \tag{38}$$

- and a notion of "dependence" (conditioning reduces valuation):

$$f(A|B) \triangleq f(A \cup B) - f(B) < f(A), \tag{39}$$

- and a notion of "conditional mutual information"

$$I_f(A; B|C) \triangleq f(A \cup C) + f(B \cup C) - f(A \cup B \cup C) - f(C) \geq 0$$

Submodular Generalized Dependence

- there is a notion of "independence", i.e., $A \perp\!\!\!\perp B$:

$$f(A \cup B) = f(A) + f(B), \tag{37}$$

- and a notion of "conditional independence", i.e., $A \perp\!\!\!\perp B | C$:

$$f(A \cup B \cup C) + f(C) = f(A \cup C) + f(B \cup C) \tag{38}$$

- and a notion of "dependence" (conditioning reduces valuation):

$$f(A|B) \triangleq f(A \cup B) - f(B) < f(A), \tag{39}$$

- and a notion of "conditional mutual information"

$$I_f(A; B|C) \triangleq f(A \cup C) + f(B \cup C) - f(A \cup B \cup C) - f(C) \geq 0$$

- and two notions of "information amongst a collection of sets":

$$I_f(S_1; S_2; \ldots; S_k) = \sum_{i=1}^{k} f(S_k) - f(S_1 \cup S_2 \cup \cdots \cup S_k) \tag{40}$$

$$I'_f(S_1; S_2; \ldots; S_k) = \sum_{A \subseteq \{1,2,\ldots,k\}} (-1)^{|A|+1} f(\bigcup_{j \in A} S_j) \tag{41}$$

Submodular Parameterized Clustering

- Given a submodular function $f : 2^V \to \mathbb{R}$, form the combinatorial dependence function $I_f(A; B) = f(A) + f(B) - f(A \cup B)$.

Submodular Parameterized Clustering

- Given a submodular function $f : 2^V \to \mathbb{R}$, form the combinatorial dependence function $I_f(A; B) = f(A) + f(B) - f(A \cup B)$.
- Consider clustering algorithm: First find partition $A_1^* \in \operatorname{argmin}_{A \subseteq V} I_f(A; V \setminus A)$.

Submodular Parameterized Clustering

- Given a submodular function $f : 2^V \to \mathbb{R}$, form the combinatorial dependence function $I_f(A; B) = f(A) + f(B) - f(A \cup B)$.
- Consider clustering algorithm: First find partition $A_1^* \in \operatorname{argmin}_{A \subseteq V} I_f(A; V \setminus A)$.
- Then partition the partitions: $A_{11}^* \in \operatorname{argmin}_{A \subseteq A_1^*} I_f(A; A_1^* \setminus A)$ and $A_{12}^* \in \operatorname{argmin}_{A \subseteq V \setminus A_1^*} I_f(A; (V \setminus A_1^*) \setminus A)$

Submodular Parameterized Clustering

- Given a submodular function $f : 2^V \to \mathbb{R}$, form the combinatorial dependence function $I_f(A; B) = f(A) + f(B) - f(A \cup B)$.
- Consider clustering algorithm: First find partition $A_1^* \in \operatorname{argmin}_{A \subseteq V} I_f(A; V \setminus A)$.
- Then partition the partitions: $A_{11}^* \in \operatorname{argmin}_{A \subseteq A_1^*} I_f(A; A_1^* \setminus A)$ and $A_{12}^* \in \operatorname{argmin}_{A \subseteq V \setminus A_1^*} I_f(A; (V \setminus A_1^*) \setminus A)$
- Recursively partition the partitions, we end up with a partition $V = V_1 \cup V_2 \cup \cdots \cup V_k$ that clusters the data.

Submodular Parameterized Clustering

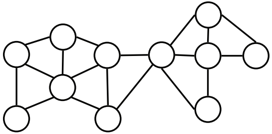
- Given a submodular function $f : 2^V \to \mathbb{R}$, form the combinatorial dependence function $I_f(A; B) = f(A) + f(B) - f(A \cup B)$.
- Consider clustering algorithm: First find partition $A_1^* \in \operatorname{argmin}_{A \subseteq V} I_f(A; V \setminus A)$.
- Then partition the partitions: $A_{11}^* \in \operatorname{argmin}_{A \subseteq A_1^*} I_f(A; A_1^* \setminus A)$ and $A_{12}^* \in \operatorname{argmin}_{A \subseteq V \setminus A_1^*} I_f(A; (V \setminus A_1^*) \setminus A)$
- Recursively partition the partitions, we end up with a partition $V = V_1 \cup V_2 \cup \cdots \cup V_k$ that clusters the data.
- Each minimization can be done using Queyranne's algorithm (alternatively can construct a Gomory-Hu tree). This gives a partition no worse than factor 2 away from optimal partition. (Narasimhan&Bilmes, 2007).

Submodular Parameterized Clustering

- Given a submodular function $f : 2^V \to \mathbb{R}$, form the combinatorial dependence function $I_f(A; B) = f(A) + f(B) - f(A \cup B)$.
- Consider clustering algorithm: First find partition $A_1^* \in \operatorname{argmin}_{A \subseteq V} I_f(A; V \setminus A)$.
- Then partition the partitions: $A_{11}^* \in \operatorname{argmin}_{A \subseteq A_1^*} I_f(A; A_1^* \setminus A)$ and $A_{12}^* \in \operatorname{argmin}_{A \subseteq V \setminus A_1^*} I_f(A; (V \setminus A_1^*) \setminus A)$
- Recursively partition the partitions, we end up with a partition $V = V_1 \cup V_2 \cup \cdots \cup V_k$ that clusters the data.
- Each minimization can be done using Queyranne's algorithm (alternatively can construct a Gomory-Hu tree). This gives a partition no worse than factor 2 away from optimal partition. (Narasimhan&Bilmes, 2007).
- Hence, family of clustering algorithms parameterized by f.

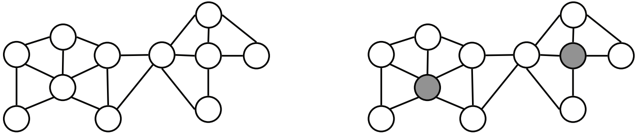
Active Transductive Semi-Supervised Learning

- Batch/Offline active learning: Given a set V of unlabeled data items, learner chooses subset $L \subseteq V$ of items to be labeled

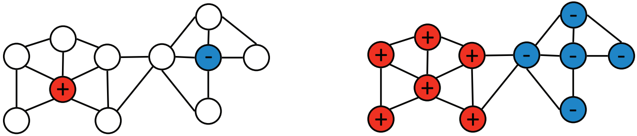


Active Transductive Semi-Supervised Learning

- Batch/Offline active learning: Given a set V of unlabeled data items, learner chooses subset $L \subseteq V$ of items to be labeled

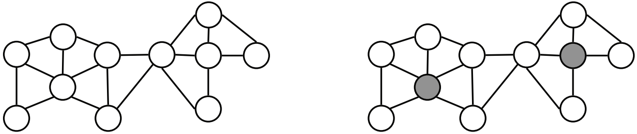


- Nature reveals labels $y_L \in \{0,1\}^L$, learner predicts labels $\hat{y} \in \{0,1\}^V$

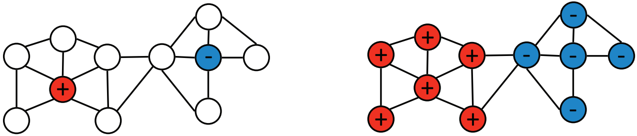


Active Transductive Semi-Supervised Learning

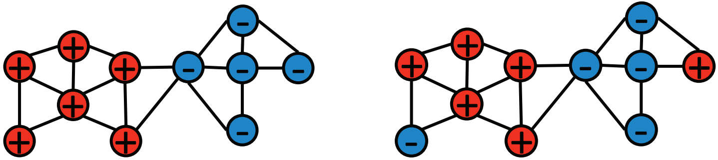
- Batch/Offline active learning: Given a set V of unlabeled data items, learner chooses subset $L \subseteq V$ of items to be labeled



- Nature reveals labels $y_L \in \{0,1\}^L$, learner predicts labels $\hat{y} \in \{0,1\}^V$



- Learner suffers loss $\|\hat{y} - y\|_1$, here $\|\hat{y} - y\|_1 = 2$.



Choosing labels: how to select L

- Consider the following objective

$$\Psi(L) = \min_{T \subseteq V \setminus L : T \neq \emptyset} \frac{\Gamma(T)}{|T|} \tag{42}$$

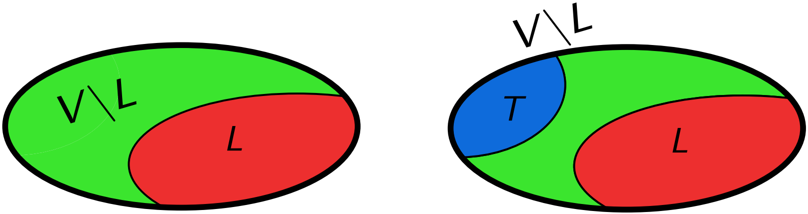
where $\Gamma(T) = f(T) + f(V \setminus T) - f(V)$ is an arbitrary symmetric submodular function (e.g., graph cut value between T and $V \setminus T$).

Choosing labels: how to select L

- Consider the following objective

$$\Psi(L) = \min_{T \subseteq V \setminus L : T \neq \emptyset} \frac{\Gamma(T)}{|T|} \tag{42}$$

  where $\Gamma(T) = f(T) + f(V \setminus T) - f(V)$ is an arbitrary symmetric submodular function (e.g., graph cut value between T and $V \setminus T$).
- Small $\Psi(L)$ means an adversary can separate away many ($|T|$ is big) combinatorially "independent" ($\Gamma(T)$ is small) points from L.



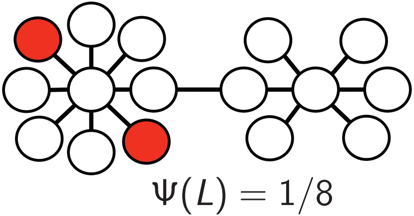
Choosing labels: how to select L

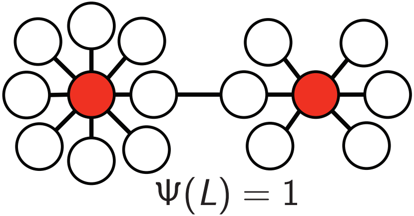
- Consider the following objective

$$\Psi(L) = \min_{T \subseteq V \setminus L : T \neq \emptyset} \frac{\Gamma(T)}{|T|} \tag{42}$$

where $\Gamma(T) = f(T) + f(V \setminus T) - f(V)$ is an arbitrary symmetric submodular function (e.g., graph cut value between T and $V \setminus T$).

- Small $\Psi(L)$ means an adversary can separate away many ($|T|$ is big) combinatorially "independent" ($\Gamma(T)$ is small) points from L.





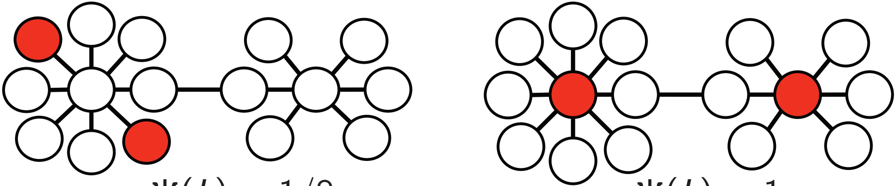
Choosing labels: how to select L

- Consider the following objective

$$\Psi(L) = \min_{T \subseteq V \setminus L : T \neq \emptyset} \frac{\Gamma(T)}{|T|} \tag{42}$$

where $\Gamma(T) = f(T) + f(V \setminus T) - f(V)$ is an arbitrary symmetric submodular function (e.g., graph cut value between T and $V \setminus T$).

- Small $\Psi(L)$ means an adversary can separate away many ($|T|$ is big) combinatorially "independent" ($\Gamma(T)$ is small) points from L.



$\Psi(L) = 1/8$ $\Psi(L) = 1$

- This suggests choosing (bounded cost) L that maximizes $\Psi(L)$.

Choosing labels: how to select L

- Given labels L, how to complete the labels?

Choosing labels: how to select L

- Given labels L, how to complete the labels?
- We form a labeling $\hat{y} \in \{0,1\}^V$ such that $\hat{y}_L = y_L$ (i.e., we agree with the known labels).

Choosing labels: how to select L

- Given labels L, how to complete the labels?
- We form a labeling $\hat{y} \in \{0,1\}^V$ such that $\hat{y}_L = y_L$ (i.e., we agree with the known labels).
- $\Gamma(T)$ measures label smoothness, how much combinatorial "information" between labels T and complement $V \setminus T$ (e.g., in graph-cut case, says label change should be across small cuts).

Choosing labels: how to select L

- Given labels L, how to complete the labels?
- We form a labeling $\hat{y} \in \{0,1\}^V$ such that $\hat{y}_L = y_L$ (i.e., we agree with the known labels).
- $\Gamma(T)$ measures label smoothness, how much combinatorial "information" between labels T and complement $V \setminus T$ (e.g., in graph-cut case, says label change should be across small cuts).
- Hence, choose labels to minimize $\Gamma(Y(\hat{y}))$ such that $\hat{y}_L = y_L$.

Choosing labels: how to select L

- Given labels L, how to complete the labels?
- We form a labeling $\hat{y} \in \{0,1\}^V$ such that $\hat{y}_L = y_L$ (i.e., we agree with the known labels).
- $\Gamma(T)$ measures label smoothness, how much combinatorial "information" between labels T and complement $V \setminus T$ (e.g., in graph-cut case, says label change should be across small cuts).
- Hence, choose labels to minimize $\Gamma(Y(\hat{y}))$ such that $\hat{y}_L = y_L$.
- This is submodular function minimization on function $g : 2^{V \setminus L} \to \mathbb{R}_+$ where for $A \subseteq V \setminus L$,

$$g(A) = \Gamma(A \cup \{v \in L : y_L(v) = 1\}) \tag{43}$$

Choosing labels: how to select L

- Given labels L, how to complete the labels?
- We form a labeling $\hat{y} \in \{0,1\}^V$ such that $\hat{y}_L = y_L$ (i.e., we agree with the known labels).
- $\Gamma(T)$ measures label smoothness, how much combinatorial "information" between labels T and complement $V \setminus T$ (e.g., in graph-cut case, says label change should be across small cuts).
- Hence, choose labels to minimize $\Gamma(Y(\hat{y}))$ such that $\hat{y}_L = y_L$.
- This is submodular function minimization on function $g : 2^{V \setminus L} \to \mathbb{R}_+$ where for $A \subseteq V \setminus L$,

$$g(A) = \Gamma(A \cup \{v \in L : y_L(v) = 1\}) \tag{43}$$

- In graph cut case, this is standard min-cut (Blum & Chawla 2001) approach to semi-supervised learning.

Generalized Error Bound

Theorem (Guillory & Bilmes, '11)

For any symmetric submodular $\Gamma(S)$*, assume* $\hat{y}$ *minimizes* $\Gamma(Y(\hat{y}))$ *subject to* $\hat{y}_L = y_L$*. Then*

$$\|\hat{y} - y\|_1 \leq 2\frac{\Gamma(Y(y))}{\Psi(L)} \tag{44}$$

where $y \in \{0,1\}^V$ *are the true labels.*

- All is defined in terms of the symmetric submodular function Γ (need not be graph cut), where:

$$\Psi(S) = \min_{T \subseteq V \setminus S : T \neq \emptyset} \frac{\Gamma(T)}{|T|} \tag{45}$$

- $\Gamma(T) = f(S) + f(V \setminus S) - f(V)$ is determined by arbitrary submodular function f, giving different error bound for each.
- Joint algorithm is "parameterized" by a submodular function f.

Discrete Submodular Divergences

- A convex function parameterizes a Bregmann divergence, useful for clustering (Banerjee et al.), includes KL-divergence, squared l2, etc.

Discrete Submodular Divergences

- A convex function parameterizes a Bregmann divergence, useful for clustering (Banerjee et al.), includes KL-divergence, squared l2, etc.
- Given a (not nec. differentiable) convex function ϕ and a sub-gradient map $\mathcal{H}_\phi$, the generalized Bregmann divergence is defined as:

$$d_\phi^{\mathcal{H}_\phi}(x, y) = \phi(x) - \phi(y) - \langle \mathcal{H}_\phi(y), x - y \rangle, \forall x, y \in \mathrm{dom}(\phi) \quad (46)$$

Discrete Submodular Divergences

- A convex function parameterizes a Bregmann divergence, useful for clustering (Banerjee et al.), includes KL-divergence, squared l2, etc.
- Given a (not nec. differentiable) convex function ϕ and a sub-gradient map $\mathcal{H}_\phi$, the generalized Bregmann divergence is defined as:

$$d_\phi^{\mathcal{H}_\phi}(x, y) = \phi(x) - \phi(y) - \langle \mathcal{H}_\phi(y), x - y \rangle, \forall x, y \in \text{dom}(\phi) \quad (46)$$

- A submodular function parameterizes a discrete submodular Bregmann divergence (Iyer & Bilmes, 2012).

Discrete Submodular Divergences

- A convex function parameterizes a Bregmann divergence, useful for clustering (Banerjee et al.), includes KL-divergence, squared l2, etc.
- Given a (not nec. differentiable) convex function ϕ and a sub-gradient map $\mathcal{H}_\phi$, the generalized Bregmann divergence is defined as:

$$d_\phi^{\mathcal{H}_\phi}(x, y) = \phi(x) - \phi(y) - \langle \mathcal{H}_\phi(y), x - y \rangle, \forall x, y \in \mathrm{dom}(\phi) \quad (46)$$

- A submodular function parameterizes a discrete submodular Bregmann divergence (Iyer & Bilmes, 2012).
- Example, lower-bound form:

$$d_f^{\mathcal{H}_f}(X, Y) = f(X) - f(Y) - \langle \mathcal{H}_f(Y), 1_X - 1_Y \rangle \quad (47)$$

where $\mathcal{H}_f(Y)$ is a sub-gradient map.

Discrete Submodular Divergences

- A convex function parameterizes a Bregmann divergence, useful for clustering (Banerjee et al.), includes KL-divergence, squared l2, etc.
- Given a (not nec. differentiable) convex function ϕ and a sub-gradient map $\mathcal{H}_\phi$, the generalized Bregmann divergence is defined as:

$$d_\phi^{\mathcal{H}_\phi}(x, y) = \phi(x) - \phi(y) - \langle \mathcal{H}_\phi(y), x - y \rangle, \forall x, y \in \text{dom}(\phi) \quad (46)$$

- A submodular function parameterizes a discrete submodular Bregmann divergence (Iyer & Bilmes, 2012).
- Example, lower-bound form:

$$d_f^{\mathcal{H}_f}(X, Y) = f(X) - f(Y) - \langle \mathcal{H}_f(Y), 1_X - 1_Y \rangle \quad (47)$$

where $\mathcal{H}_f(Y)$ is a sub-gradient map.
- Submodular Bregmann divergences also definable in terms of supergradients.

Discrete Submodular Divergences

- A convex function parameterizes a Bregmann divergence, useful for clustering (Banerjee et al.), includes KL-divergence, squared l2, etc.
- Given a (not nec. differentiable) convex function ϕ and a sub-gradient map $\mathcal{H}_\phi$, the generalized Bregmann divergence is defined as:

$$d_\phi^{\mathcal{H}_\phi}(x, y) = \phi(x) - \phi(y) - \langle \mathcal{H}_\phi(y), x - y \rangle, \forall x, y \in \text{dom}(\phi) \quad (46)$$

- A submodular function parameterizes a discrete submodular Bregmann divergence (Iyer & Bilmes, 2012).
- Example, lower-bound form:

$$d_f^{\mathcal{H}_f}(X, Y) = f(X) - f(Y) - \langle \mathcal{H}_f(Y), 1_X - 1_Y \rangle \quad (47)$$

  where $\mathcal{H}_f(Y)$ is a sub-gradient map.
- Submodular Bregmann divergences also definable in terms of supergradients.
- General: Hamming, Recall, Precision, Cond. MI, Sq. Hamming, etc.

Outline

1. Introduction

2. Basics

3. Submodular Applications in ML
   - Where is submodularity useful?
   - Traditional combinatorial problems
   - As a model of diversity, coverage, span, or information
   - As a model of cooperative costs, complexity, roughness, and irregularity
   - As a parameter for an ML algorithm
   - Itself, as a target for learning
   - Surrogates for optimization
   - Economic applications

Learning Submodular Functions

- Learning submodular functions is hard

Learning Submodular Functions

- Learning submodular functions is hard
- Goemans et al. (2009): "can one make only polynomial number of queries to an unknown submodular function f and constructs a $\hat{f}$ such that $\hat{f}(S) \leq f(S) \leq g(n)\hat{f}(S)$ where $g : \mathbb{N} \to \mathbb{R}$?"

Learning Submodular Functions

- Learning submodular functions is hard
- Goemans et al. (2009): "can one make only polynomial number of queries to an unknown submodular function f and constructs a $\hat{f}$ such that $\hat{f}(S) \leq f(S) \leq g(n)\hat{f}(S)$ where $g : \mathbb{N} \to \mathbb{R}$?" Many results, including that even with adaptive queries and monotone functions, can't do better than $\Omega(\sqrt{n}/\log n)$.

Learning Submodular Functions

- Learning submodular functions is hard
- Goemans et al. (2009): "can one make only polynomial number of queries to an unknown submodular function f and constructs a $\hat{f}$ such that $\hat{f}(S) \leq f(S) \leq g(n)\hat{f}(S)$ where $g : \mathbb{N} \to \mathbb{R}$?" Many results, including that even with adaptive queries and monotone functions, can't do better than $\Omega(\sqrt{n}/\log n)$.
- Balcan & Harvey (2011): submodular function learning problem from a learning theory perspective, given a distribution on subsets. Negative result is that can't approximate in this setting to within a constant factor.

Learning Submodular Functions

- Learning submodular functions is hard
- Goemans et al. (2009): "can one make only polynomial number of queries to an unknown submodular function f and constructs a $\hat{f}$ such that $\hat{f}(S) \leq f(S) \leq g(n)\hat{f}(S)$ where $g : \mathbb{N} \to \mathbb{R}$?" Many results, including that even with adaptive queries and monotone functions, can't do better than $\Omega(\sqrt{n}/\log n)$.
- Balcan & Harvey (2011): submodular function learning problem from a learning theory perspective, given a distribution on subsets. Negative result is that can't approximate in this setting to within a constant factor.
- But can we learn a subclass, perhaps non-negative weighted mixtures of submodular components?

Structured Prediction in Machine Learning

- Given: a finite set of training pairs $D = \left\{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\right\}_i$ where $\mathbf{x}^{(i)} \in \mathcal{X}$, $\mathbf{y}^{(i)} \in \mathcal{Y}$.
- $\mathbf{f} : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^M$ is a (fixed) vector of functions, and $\mathbf{w} \in \mathbb{R}^M$ is a vector of parameters to learn.
- Score function: $s(\mathbf{x}, \mathbf{y}) = \mathbf{w}^\intercal \mathbf{f}(\mathbf{x}, \mathbf{y}) = \sum_i w_i f_i(\mathbf{x}, \mathbf{y})$.
- Decision making (inference) for a given $\bar{\mathbf{x}}$ is based on:

$$\hat{\mathbf{y}} \in h_{\mathbf{w}}(\bar{\mathbf{x}}) = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} s(\bar{\mathbf{x}}, \mathbf{y}) = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} \mathbf{w}^\intercal \mathbf{f}(\bar{\mathbf{x}}, \mathbf{y}) \tag{48}$$

- Goal of learning: optimize $\mathbf{w}$ so that such decision making is "good"
- Let $\ell : \mathcal{Y} \times Y \to \mathbb{R}_+$ be a loss function. I.e., $\ell_{\mathbf{y}}(\hat{\mathbf{y}})$ is cost of deciding $\hat{\mathbf{y}}$ when truth is $\mathbf{y}$.
- Empirical risk minimization: adjust $\mathbf{w}$ so that $\sum_i \ell_{\mathbf{y}}(h_{\mathbf{w}}(\mathbf{x}^{(i)}))$ is small subject to other conditions (e.g., regularization).

Structured Prediction: Approach with inference

- Constraints specified in inference form:

$$\underset{\mathbf{w},\xi_t}{\text{minimize}} \quad \frac{1}{T}\sum_t \xi_t + \frac{\lambda}{2}\|\mathbf{w}\|^2 \tag{49}$$

$$\text{subject to} \quad \mathbf{w}^\top \mathbf{f}_t(\mathbf{y}^{(t)}) \geq \max_{\mathbf{y}\in\mathcal{Y}_t}\left(\mathbf{w}^\top \mathbf{f}_t(\mathbf{y}) + \ell_t(\mathbf{y})\right) - \xi_t, \forall t \tag{50}$$

$$\xi_t \geq 0, \forall t. \tag{51}$$

- Exponential set of constraints reduced to an embedded optimization problem, "inference."

Learning Submodular Mixtures: Unconstrained Form

- Unconstrained form uses a generalized hinge-loss (Taskar 2004), which is amenable to sub-gradient descent optimization:

$$\min_{\mathbf{w}\geq 0} \frac{1}{T} \sum_t \left[\max_{\mathbf{y}\in\mathcal{Y}_t} \left(\mathbf{w}^\top \mathbf{f}_t(\mathbf{y}) + \ell_t(\mathbf{y}) \right) - \mathbf{w}^\top \mathbf{f}_t(\mathbf{y}^{(t)}) \right] + \frac{\lambda}{2} \|\mathbf{w}\|^2 \quad (52)$$

- Note, $\mathbf{w} \geq 0$ critical to preserve submodularity.
- To compute a subgradient, must solve the following embedded optimization problem ("loss augmented inference"):

$$\max_{\mathbf{y}\in\mathcal{Y}_t} \left(\mathbf{w}^\top \mathbf{f}_t(\mathbf{y}) + \ell_t(\mathbf{y}) \right) \quad (53)$$

- The problem is convex in $\mathbf{w}$, and $\mathbf{w}^\top \mathbf{f}_t(\mathbf{y})$ is submodular (polymatroidal in fact), but what about $\ell_t(\mathbf{y})$?
- Often one uses Hamming loss (in general structured prediction problems) which is submodular (modular in fact).
- If loss $\ell_t(\mathbf{y})$, more generally, is submodular, then Eq. (53) can be solved at least approximately well.

Structured Prediction: Subgradient

- Subgradient, evaluated at **w**, of the following

$$\max_{\mathbf{y}\in\mathcal{Y}_t}\left(\mathbf{w}^\top \mathbf{f}_t(\mathbf{y}) + \ell_t(\mathbf{y})\right) - \mathbf{w}^\top \mathbf{f}_t(\mathbf{y}^{(t)}) + \frac{\lambda}{2}\|\mathbf{w}\|^2 \tag{54}$$

can be found by computing or approximating

$$\mathbf{y}^* \in \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}_t}\left(\mathbf{w}^\top \mathbf{f}_t(\mathbf{y}) + \ell_t(\mathbf{y})\right) - \mathbf{w}^\top \mathbf{f}_t(\mathbf{y}^{(t)}) \tag{55}$$

and then finding subgradient of

$$\mathbf{w}^\top \mathbf{f}_t(\mathbf{y}^*) + \ell_t(\mathbf{y}^*) - \mathbf{w}^\top \mathbf{f}_t(\mathbf{y}^{(t)}) + \frac{\lambda}{2}\|\mathbf{w}\|^2 \tag{56}$$

which has the form

$$\mathbf{f}_t(\mathbf{y}^*) - \mathbf{f}_t(\mathbf{y}^{(t)}) + \lambda\mathbf{w}. \tag{57}$$

Structured Prediction: Subgradient Learning

Algorithm 1: Subgradient descent learning

Input : $S = \{(\mathbf{x}^{(t)}, \mathbf{y}^{(t)})\}_{t=1}^{T}$ and a learning rate sequence $\{\eta_t\}_{t=1}^{T}$.

$w_0 = 0$;

for $t = 1, \cdots, T$ **do**

Loss augmented inference: $\mathbf{y}_t^* \in \text{argmax}_{\mathbf{y} \in \mathcal{Y}_t} \mathbf{w}_{t-1}^\top \mathbf{f}_t(\mathbf{y}) + \ell_t(\mathbf{y})$;

Compute the subgradient: $\mathbf{g}_t = \lambda \mathbf{w}_{t-1} + \mathbf{f}_t(\mathbf{y}^*) - \mathbf{f}_t(\mathbf{y}^{(t)})$;

Update the weights: $\mathbf{w}_t = \mathbf{w}_{t-1} - \eta_t \mathbf{g}_t$;

Return : the averaged parameters $\frac{1}{T} \sum_t \mathbf{w}_t$.

Outline

1. Introduction
2. Basics
3. Submodular Applications in ML
   - Where is submodularity useful?
   - Traditional combinatorial problems
   - As a model of diversity, coverage, span, or information
   - As a model of cooperative costs, complexity, roughness, and irregularity
   - As a parameter for an ML algorithm
   - Itself, as a target for learning
   - Surrogates for optimization
   - Economic applications

Submodular Relaxation

- We often are unable to optimize an objective. E.g., high tree-width graphical models (as we saw).

Submodular Relaxation

- We often are unable to optimize an objective. E.g., high tree-width graphical models (as we saw).
- If potentials are submodular, we can solve them.

Submodular Relaxation

- We often are unable to optimize an objective. E.g., high tree-width graphical models (as we saw).
- If potentials are submodular, we can solve them.
- When potentials are not, we might resort to factorization (e.g., the marginal polytope in variational inference, were we optimize over a tree-constrained polytope).

Submodular Relaxation

- We often are unable to optimize an objective. E.g., high tree-width graphical models (as we saw).
- If potentials are submodular, we can solve them.
- When potentials are not, we might resort to factorization (e.g., the marginal polytope in variational inference, were we optimize over a tree-constrained polytope).
- An alternative is submodular relaxation. I.e., given

$$\Pr(x) = \frac{1}{Z} \exp(-E(x)) \tag{58}$$

where $E(x) = E_f(x) - E_g(x)$ and both of $E_f(x)$ and $E_g(x)$ are submodular.

Submodular Relaxation

- We often are unable to optimize an objective. E.g., high tree-width graphical models (as we saw).
- If potentials are submodular, we can solve them.
- When potentials are not, we might resort to factorization (e.g., the marginal polytope in variational inference, were we optimize over a tree-constrained polytope).
- An alternative is submodular relaxation. I.e., given

$$\Pr(x) = \frac{1}{Z} \exp(-E(x)) \tag{58}$$

  where $E(x) = E_f(x) - E_g(x)$ and both of $E_f(x)$ and $E_g(x)$ are submodular.
- Any function can be expressed as the difference between two submodular functions.

Submodular Relaxation

- We often are unable to optimize an objective. E.g., high tree-width graphical models (as we saw).
- If potentials are submodular, we can solve them.
- When potentials are not, we might resort to factorization (e.g., the marginal polytope in variational inference, were we optimize over a tree-constrained polytope).
- An alternative is submodular relaxation. I.e., given

$$\Pr(x) = \frac{1}{Z}\exp(-E(x)) \tag{58}$$

  where $E(x) = E_f(x) - E_g(x)$ and both of $E_f(x)$ and $E_g(x)$ are submodular.
- Any function can be expressed as the difference between two submodular functions.
- Hence, rather than minimize $E(x)$ (hard), we can minimize $E_f(x) \geq E(x)$ (relatively easy), which is an upper bound.

Outline

1. Introduction

2. Basics

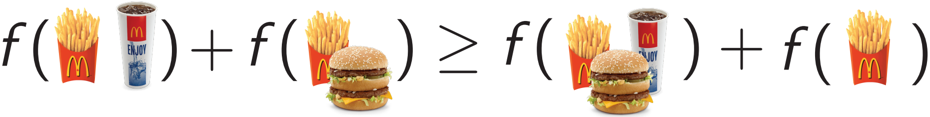
3. Submodular Applications in ML
   - Where is submodularity useful?
   - Traditional combinatorial problems
   - As a model of diversity, coverage, span, or information
   - As a model of cooperative costs, complexity, roughness, and irregularity
   - As a parameter for an ML algorithm
   - Itself, as a target for learning
   - Surrogates for optimization
   - Economic applications

Ex. Submodular: Consumer Costs of Living

- Consumer costs are very often submodular.

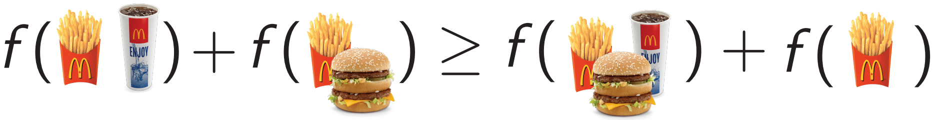
Ex. Submodular: Consumer Costs of Living

- Consumer costs are very often submodular. For example:



Ex. Submodular: Consumer Costs of Living

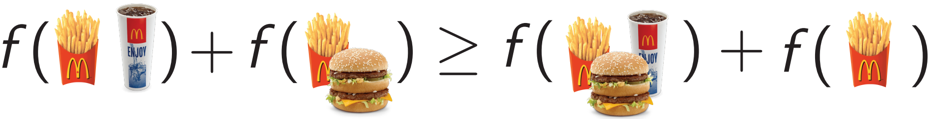
- Consumer costs are very often submodular. For example:



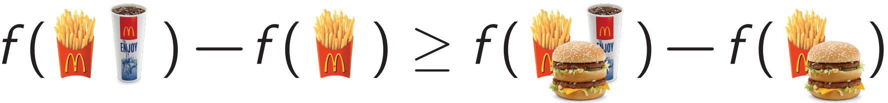
- Rearranging terms, we can see this as diminishing returns:

Ex. Submodular: Consumer Costs of Living

- Consumer costs are very often submodular. For example:

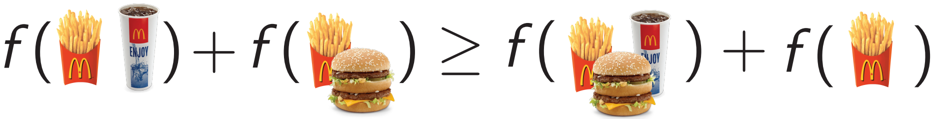


- Rearranging terms, we can see this as diminishing returns:

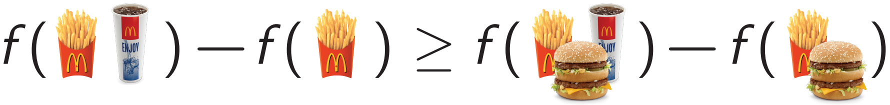


Ex. Submodular: Consumer Costs of Living

- Consumer costs are very often submodular. For example:



- Rearranging terms, we can see this as diminishing returns:



- This is very common: The additional cost of a coke is, say, free if you add it to fries and a hamburger, but when added just to an order of fries, the coke is not free.

Shared Fixed Costs

- Costs often interact in the real world.

Shared Fixed Costs

- Costs often interact in the real world.
- Ex: Let $V = \{v_1, v_2\}$ be a set of actions with:

v_1 = "buy milk at the store" v_2 = "buy honey at the store"

Shared Fixed Costs

- Costs often interact in the real world.
- Ex: Let $V = \{v_1, v_2\}$ be a set of actions with:

v_1 = "buy milk at the store" v_2 = "buy honey at the store"


- For $A \subseteq V$, let $f(A)$ be the cost of set of items A.

Shared Fixed Costs

- Costs often interact in the real world.
- Ex: Let $V = \{v_1, v_2\}$ be a set of actions with:

  v_1 = "buy milk at the store" v_2 = "buy honey at the store"

- For $A \subseteq V$, let $f(A)$ be the cost of set of items A.
- $f(\{v_1\})$ = cost to drive to and from store, and cost to purchase milk, say $c_d + c_m$.

Shared Fixed Costs

- Costs often interact in the real world.
- Ex: Let $V = \{v_1, v_2\}$ be a set of actions with:

v_1 = "buy milk at the store" v_2 = "buy honey at the store"

- For $A \subseteq V$, let $f(A)$ be the cost of set of items A.
- $f(\{v_1\})$ = cost to drive to and from store, and cost to purchase milk, say $c_d + c_m$.
- $f(\{v_2\})$ = cost to drive to and from store, and cost to purchase honey, say $c_d + c_h$.

Shared Fixed Costs

- Costs often interact in the real world.
- Ex: Let $V = \{v_1, v_2\}$ be a set of actions with:

  v_1 = "buy milk at the store" v_2 = "buy honey at the store"

- For $A \subseteq V$, let $f(A)$ be the cost of set of items A.
- $f(\{v_1\})$ = cost to drive to and from store, <u>and</u> cost to purchase milk, say $c_d + c_m$.
- $f(\{v_2\})$ = cost to drive to and from store, <u>and</u> cost to purchase honey, say $c_d + c_h$.
- But $f(\{v_1, v_2\}) = c_d + c_m + c_h < 2c_d + c_m + c_h$ since c_d (driving) is a shared fixed cost.

Shared Fixed Costs

- Costs often interact in the real world.
- Ex: Let $V = \{v_1, v_2\}$ be a set of actions with:

  v_1 = "buy milk at the store" v_2 = "buy honey at the store"

- For $A \subseteq V$, let $f(A)$ be the cost of set of items A.
- $f(\{v_1\})$ = cost to drive to and from store, <u>and</u> cost to purchase milk, say $c_d + c_m$.
- $f(\{v_2\})$ = cost to drive to and from store, <u>and</u> cost to purchase honey, say $c_d + c_h$.
- But $f(\{v_1, v_2\}) = c_d + c_m + c_h < 2c_d + c_m + c_h$ since c_d (driving) is a shared fixed cost.
- Shared fixed costs are submodular: $f(v_1) + f(v_2) \geq f(v_1, v_2) + f(\emptyset)$

Supply Side Economies of scale

- What is a good model of the cost of manufacturing a set of items?

Supply Side Economies of scale

- What is a good model of the cost of manufacturing a set of items?
- Let V be a set of possible items that a company might possibly wish to manufacture, and let $f(S)$ for $S \subseteq V$ be the cost to that company to manufacture subset S.

Supply Side Economies of scale

- What is a good model of the cost of manufacturing a set of items?
- Let V be a set of possible items that a company might possibly wish to manufacture, and let $f(S)$ for $S \subseteq V$ be the cost to that company to manufacture subset S.
- Ex: V might be colors of paint in a paint manufacturer: green, red, blue, yellow, white, etc.

Supply Side Economies of scale

- What is a good model of the cost of manufacturing a set of items?
- Let V be a set of possible items that a company might possibly wish to manufacture, and let $f(S)$ for $S \subseteq V$ be the cost to that company to manufacture subset S.
- Ex: V might be colors of paint in a paint manufacturer: green, red, blue, yellow, white, etc.
- Producing green when you are already producing yellow and blue is probably cheaper than if you were only producing some other colors.

$$f(\text{green}, \text{blue}, \text{yellow}) - f(\text{blue}, \text{yellow}) <= f(\text{green}, \text{blue}) - f(\text{blue}) \tag{59}$$

Supply Side Economies of scale

- What is a good model of the cost of manufacturing a set of items?
- Let V be a set of possible items that a company might possibly wish to manufacture, and let $f(S)$ for $S \subseteq V$ be the cost to that company to manufacture subset S.
- Ex: V might be colors of paint in a paint manufacturer: green, red, blue, yellow, white, etc.
- Producing green when you are already producing yellow and blue is probably cheaper than if you were only producing some other colors.

$$f(\text{green}, \text{blue}, \text{yellow}) - f(\text{blue}, \text{yellow}) <= f(\text{green}, \text{blue}) - f(\text{blue}) \tag{59}$$

- So diminishing returns (a submodular function) would be a good model.

Demand side Economies of Scale: Network Externalities

- consumers of a good derive positive value when size of the market increases.

Demand side Economies of Scale: Network Externalities

- consumers of a good derive positive value when size of the market increases.
- the value of a network to a user depends on the number of other users in that network. External use benefits internal use.

Demand side Economies of Scale: Network Externalities

- consumers of a good derive positive value when size of the market increases.
- the value of a network to a user depends on the number of other users in that network. External use benefits internal use.
- This is called network externalities (Katz & Shapiro 1986), and is a form of "demand" economies of scale

Demand side Economies of Scale: Network Externalities

- consumers of a good derive positive value when size of the market increases.
- the value of a network to a user depends on the number of other users in that network. External use benefits internal use.
- This is called network externalities (Katz & Shapiro 1986), and is a form of "demand" economies of scale
- Given network externalities, a consumer in today's market cares also about the future success of the product and competing products.

Demand side Economies of Scale: Network Externalities

- consumers of a good derive positive value when size of the market increases.
- the value of a network to a user depends on the number of other users in that network. External use benefits internal use.
- This is called network externalities (Katz & Shapiro 1986), and is a form of "demand" economies of scale
- Given network externalities, a consumer in today's market cares also about the future success of the product and competing products.
- If the good is durable (e.g., a car or phone) or there is human capital investment (e.g., education in a skill), the total benefits derived from a good will depend on the number of consumers who adopt compatible products in the future.

Demand side Economies of Scale: Network Externalities

- consumers of a good derive positive value when size of the market increases.
- the value of a network to a user depends on the number of other users in that network. External use benefits internal use.
- This is called network externalities (Katz & Shapiro 1986), and is a form of "demand" economies of scale
- Given network externalities, a consumer in today's market cares also about the future success of the product and competing products.
- If the good is durable (e.g., a car or phone) or there is human capital investment (e.g., education in a skill), the total benefits derived from a good will depend on the number of consumers who adopt compatible products in the future.
- So supermodularity would be a good model.

Outline: Part 2

4. From Matroids to Polymatroids
   - Matrix Rank
   - Venn Diagrams
   - Matroids

5. Submodular Definitions, Examples, and Properties
   - Normalization
   - Submodular Definitions
   - Submodular Composition
   - More Examples

Example: Rank function of a matrix

- Given an $n \times m$ matrix, thought of as m column vectors:

$$\mathbf{X} = \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & & m \end{array} \\ \begin{pmatrix} | & | & | & | & & | \\ x_1 & x_2 & x_3 & x_4 & \dots & x_m \\ | & | & | & | & & | \end{pmatrix} \end{array} \tag{60}$$

- Let set $V = \{1, 2, \dots, m\}$ be the set of column vector indices.
- For any subset of column vector indices $A \subseteq V$, let $r(A)$ be the rank of the column vectors indexed by A.
- Hence $r : 2^V \to \mathbb{Z}_+$ and $r(A)$ is the dimensionality of the vector space spanned by the set of vectors $\{x_a\}_{a \in A}$.
- Intuitively, $r(A)$ is the size of the largest set of independent vectors contained within the set of vectors indexed by A.

Example: Rank function of a matrix

Consider the following 4×8 matrix, so $V = \{1, 2, 3, 4, 5, 6, 7, 8\}$.

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{pmatrix} 0 & 2 & 2 & 3 & 0 & 1 & 3 & 1 \\ 0 & 3 & 0 & 4 & 0 & 0 & 2 & 4 \\ 0 & 0 & 0 & 0 & 3 & 0 & 0 & 5 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 5 \end{pmatrix} \end{array} = \begin{array}{c} \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{pmatrix} | & | & | & | & | & | & | & | \\ x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \\ | & | & | & | & | & | & | & | \end{pmatrix} \end{array}$$

- Let $A = \{1, 2, 3\}$, $B = \{3, 4, 5\}$, $C = \{6, 7\}$, $A_r = \{1\}$, $B_r = \{5\}$.
- Then $r(A) = 3$, $r(B) = 3$, $r(C) = 2$.
- $r(A \cup C) = 3$, $r(B \cup C) = 3$.
- $r(A \cup A_r) = 3$, $r(B \cup B_r) = 3$, $r(A \cup B_r) = 4$, $r(B \cup A_r) = 4$.
- $r(A \cup B) = 4$, $r(A \cap B) = 1 < r(C) = 2$.

Example: Rank function of a matrix

Consider the following 4×8 matrix, so $V = \{1,2,3,4,5,6,7,8\}$.

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{pmatrix} 0 & 2 & 2 & 3 & 0 & 1 & 3 & 1 \\ 0 & 3 & 0 & 4 & 0 & 0 & 2 & 4 \\ 0 & 0 & 0 & 0 & 3 & 0 & 0 & 5 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 5 \end{pmatrix} \end{array} = \begin{array}{c} \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{pmatrix} | & | & | & | & | & | & | & | \\ x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \\ | & | & | & | & | & | & | & | \end{pmatrix} \end{array}$$

- Let $A = \{1,2,3\}$, $B = \{3,4,5\}$, $C = \{6,7\}$, $A_r = \{1\}$, $B_r = \{5\}$.
- Then $r(A) = 3$, $r(B) = 3$, $r(C) = 2$.
- $r(A \cup C) = 3$, $r(B \cup C) = 3$.
- $r(A \cup A_r) = 3$, $r(B \cup B_r) = 3$, $r(A \cup B_r) = 4$, $r(B \cup A_r) = 4$.
- $r(A \cup B) = 4$, $r(A \cap B) = 1 < r(C) = 2$.

Example: Rank function of a matrix

Consider the following 4×8 matrix, so $V = \{1, 2, 3, 4, 5, 6, 7, 8\}$.

$$
\begin{array}{c|cccccccc}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\hline
1 & 0 & 2 & 2 & 3 & 0 & 1 & 3 & 1 \\
2 & 0 & 3 & 0 & 4 & 0 & 0 & 2 & 4 \\
3 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 5 \\
4 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 5
\end{array}
=
\begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8
\end{array}
\begin{pmatrix}
| & | & | & | & | & | & | & | \\
x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \\
| & | & | & | & | & | & | & |
\end{pmatrix}
$$

- Let $A = \{1, 2, 3\}$, $B = \{3, 4, 5\}$, $C = \{6, 7\}$, $A_r = \{1\}$, $B_r = \{5\}$.
- Then $r(A) = 3$, $r(B) = 3$, $r(C) = 2$.
- $r(A \cup C) = 3$, $r(B \cup C) = 3$.
- $r(A \cup A_r) = 3$, $r(B \cup B_r) = 3$, $r(A \cup B_r) = 4$, $r(B \cup A_r) = 4$.
- $r(A \cup B) = 4$, $r(A \cap B) = 1 < r(C) = 2$.

Example: Rank function of a matrix

Consider the following 4×8 matrix, so $V = \{1, 2, 3, 4, 5, 6, 7, 8\}$.

$$\begin{array}{c c} & \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{pmatrix} 0 & 2 & 2 & 3 & 0 & 1 & 3 & 1 \\ 0 & 3 & 0 & 4 & 0 & 0 & 2 & 4 \\ 0 & 0 & 0 & 0 & 3 & 0 & 0 & 5 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 5 \end{pmatrix} \end{array} = \begin{array}{c} \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{pmatrix} | & | & | & | & | & | & | & | \\ x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \\ | & | & | & | & | & | & | & | \end{pmatrix} \end{array}$$

- Let $A = \{1, 2, 3\}$, $B = \{3, 4, 5\}$, $C = \{6, 7\}$, $A_r = \{1\}$, $B_r = \{5\}$.
- Then $r(A) = 3$, $r(B) = 3$, $r(C) = 2$.
- $r(A \cup C) = 3$, $r(B \cup C) = 3$.
- $r(A \cup A_r) = 3$, $r(B \cup B_r) = 3$, $r(A \cup B_r) = 4$, $r(B \cup A_r) = 4$.
- $r(A \cup B) = 4$, $r(A \cap B) = 1 < r(C) = 2$.

Example: Rank function of a matrix

Consider the following 4×8 matrix, so $V = \{1,2,3,4,5,6,7,8\}$.

$$\begin{array}{c|cccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \hline 1 & 0 & 2 & 2 & 3 & 0 & 1 & 3 & 1 \\ 2 & 0 & 3 & 0 & 4 & 0 & 0 & 2 & 4 \\ 3 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 5 \\ 4 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 5 \end{array} = \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \begin{pmatrix} | & | & | & | & | & | & | & | \\ x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \\ | & | & | & | & | & | & | & | \end{pmatrix}$$

- Let $A = \{1,2,3\}$, $B = \{3,4,5\}$, $C = \{6,7\}$, $A_r = \{1\}$, $B_r = \{5\}$.
- Then $r(A) = 3$, $r(B) = 3$, $r(C) = 2$.
- $r(A \cup C) = 3$, $r(B \cup C) = 3$.
- $r(A \cup A_r) = 3$, $r(B \cup B_r) = 3$, $r(A \cup B_r) = 4$, $r(B \cup A_r) = 4$.
- $r(A \cup B) = 4$, $r(A \cap B) = 1 < r(C) = 2$.

Example: Rank function of a matrix

Consider the following 4×8 matrix, so $V = \{1, 2, 3, 4, 5, 6, 7, 8\}$.

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{pmatrix} 0 & 2 & 2 & 3 & 0 & 1 & 3 & 1 \\ 0 & 3 & 0 & 4 & 0 & 0 & 2 & 4 \\ 0 & 0 & 0 & 0 & 3 & 0 & 0 & 5 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 5 \end{pmatrix} \end{array} = \begin{array}{c} \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{pmatrix} | & | & | & | & | & | & | & | \\ x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \\ | & | & | & | & | & | & | & | \end{pmatrix} \end{array}$$

- Let $A = \{1, 2, 3\}$, $B = \{3, 4, 5\}$, $C = \{6, 7\}$, $A_r = \{1\}$, $B_r = \{5\}$.
- Then $r(A) = 3$, $r(B) = 3$, $r(C) = 2$.
- $r(A \cup C) = 3$, $r(B \cup C) = 3$.
- $r(A \cup A_r) = 3$, $r(B \cup B_r) = 3$, $r(A \cup B_r) = 4$, $r(B \cup A_r) = 4$.
- $r(A \cup B) = 4$, $r(A \cap B) = 1 < r(C) = 2$.

Example: Rank function of a matrix

Consider the following 4×8 matrix, so $V = \{1, 2, 3, 4, 5, 6, 7, 8\}$.

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{pmatrix} 0 & 2 & 2 & 3 & 0 & 1 & 3 & 1 \\ 0 & 3 & 0 & 4 & 0 & 0 & 2 & 4 \\ 0 & 0 & 0 & 0 & 3 & 0 & 0 & 5 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 5 \end{pmatrix} \end{array} = \begin{array}{c} \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{pmatrix} | & | & | & | & | & | & | & | \\ x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \\ | & | & | & | & | & | & | & | \end{pmatrix} \end{array}$$

- Let $A = \{1, 2, 3\}$, $B = \{3, 4, 5\}$, $C = \{6, 7\}$, $A_r = \{1\}$, $B_r = \{5\}$.
- Then $r(A) = 3$, $r(B) = 3$, $r(C) = 2$.
- $r(A \cup C) = 3$, $r(B \cup C) = 3$.
- $r(A \cup A_r) = 3$, $r(B \cup B_r) = 3$, $r(A \cup B_r) = 4$, $r(B \cup A_r) = 4$.
- $r(A \cup B) = 4$, $r(A \cap B) = 1 < r(C) = 2$.

Example: Rank function of a matrix

Consider the following 4×8 matrix, so $V = \{1, 2, 3, 4, 5, 6, 7, 8\}$.

$$\begin{array}{c|c} & \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{pmatrix} 0 & 2 & 2 & 3 & 0 & 1 & 3 & 1 \\ 0 & 3 & 0 & 4 & 0 & 0 & 2 & 4 \\ 0 & 0 & 0 & 0 & 3 & 0 & 0 & 5 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 5 \end{pmatrix} \end{array} = \begin{array}{c} \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{pmatrix} | & | & | & | & | & | & | & | \\ x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \\ | & | & | & | & | & | & | & | \end{pmatrix} \end{array}$$

- Let $A = \{1, 2, 3\}$, $B = \{3, 4, 5\}$, $C = \{6, 7\}$, $A_r = \{1\}$, $B_r = \{5\}$.
- Then $r(A) = 3$, $r(B) = 3$, $r(C) = 2$.
- $r(A \cup C) = 3$, $r(B \cup C) = 3$.
- $r(A \cup A_r) = 3$, $r(B \cup B_r) = 3$, $r(A \cup B_r) = 4$, $r(B \cup A_r) = 4$.
- $r(A \cup B) = 4$, $r(A \cap B) = 1 < r(C) = 2$.

Example: Rank function of a matrix

Consider the following 4×8 matrix, so $V = \{1,2,3,4,5,6,7,8\}$.

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{pmatrix} 0 & 2 & 2 & 3 & 0 & 1 & 3 & 1 \\ 0 & 3 & 0 & 4 & 0 & 0 & 2 & 4 \\ 0 & 0 & 0 & 0 & 3 & 0 & 0 & 5 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 5 \end{pmatrix} \end{array} = \begin{array}{c} \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{pmatrix} | & | & | & | & | & | & | & | \\ x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \\ | & | & | & | & | & | & | & | \end{pmatrix} \end{array}$$

- Let $A = \{1,2,3\}$, $B = \{3,4,5\}$, $C = \{6,7\}$, $A_r = \{1\}$, $B_r = \{5\}$.
- Then $r(A) = 3$, $r(B) = 3$, $r(C) = 2$.
- $r(A \cup C) = 3$, $r(B \cup C) = 3$.
- $r(A \cup A_r) = 3$, $r(B \cup B_r) = 3$, $r(A \cup B_r) = 4$, $r(B \cup A_r) = 4$.
- $r(A \cup B) = 4$, $r(A \cap B) = 1 < r(C) = 2$.

Example: Rank function of a matrix

Consider the following 4×8 matrix, so $V = \{1, 2, 3, 4, 5, 6, 7, 8\}$.

$$\begin{array}{c c} & \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{pmatrix} 0 & 2 & 2 & 3 & 0 & 1 & 3 & 1 \\ 0 & 3 & 0 & 4 & 0 & 0 & 2 & 4 \\ 0 & 0 & 0 & 0 & 3 & 0 & 0 & 5 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 5 \end{pmatrix} \end{array} = \begin{array}{c} \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{pmatrix} | & | & | & | & | & | & | & | \\ x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \\ | & | & | & | & | & | & | & | \end{pmatrix} \end{array}$$

- Let $A = \{1, 2, 3\}$, $B = \{3, 4, 5\}$, $C = \{6, 7\}$, $A_r = \{1\}$, $B_r = \{5\}$.
- Then $r(A) = 3$, $r(B) = 3$, $r(C) = 2$.
- $r(A \cup C) = 3$, $r(B \cup C) = 3$.
- $r(A \cup A_r) = 3$, $r(B \cup B_r) = 3$, $r(A \cup B_r) = 4$, $r(B \cup A_r) = 4$.
- $r(A \cup B) = 4$, $r(A \cap B) = 1 < r(C) = 2$.

Example: Rank function of a matrix

Consider the following 4×8 matrix, so $V = \{1, 2, 3, 4, 5, 6, 7, 8\}$.

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array}\begin{array}{c} \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{pmatrix} 0 & 2 & 2 & 3 & 0 & 1 & 3 & 1 \\ 0 & 3 & 0 & 4 & 0 & 0 & 2 & 4 \\ 0 & 0 & 0 & 0 & 3 & 0 & 0 & 5 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 5 \end{pmatrix} \end{array} = \begin{array}{c} \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{pmatrix} | & | & | & | & | & | & | & | \\ x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \\ | & | & | & | & | & | & | & | \end{pmatrix} \end{array}$$

- Let $A = \{1, 2, 3\}$, $B = \{3, 4, 5\}$, $C = \{6, 7\}$, $A_r = \{1\}$, $B_r = \{5\}$.
- Then $r(A) = 3$, $r(B) = 3$, $r(C) = 2$.
- $r(A \cup C) = 3$, $r(B \cup C) = 3$.
- $r(A \cup A_r) = 3$, $r(B \cup B_r) = 3$, $r(A \cup B_r) = 4$, $r(B \cup A_r) = 4$.
- $r(A \cup B) = 4$, $r(A \cap B) = 1 < r(C) = 2$.

Example: Rank function of a matrix

Consider the following 4×8 matrix, so $V = \{1,2,3,4,5,6,7,8\}$.

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{pmatrix} 0 & 2 & 2 & 3 & 0 & 1 & 3 & 1 \\ 0 & 3 & 0 & 4 & 0 & 0 & 2 & 4 \\ 0 & 0 & 0 & 0 & 3 & 0 & 0 & 5 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 5 \end{pmatrix} \end{array} = \begin{array}{c} \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{pmatrix} | & | & | & | & | & | & | & | \\ x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \\ | & | & | & | & | & | & | & | \end{pmatrix} \end{array}$$

- Let $A = \{1,2,3\}$, $B = \{3,4,5\}$, $C = \{6,7\}$, $A_r = \{1\}$, $B_r = \{5\}$.
- Then $r(A) = 3$, $r(B) = 3$, $r(C) = 2$.
- $r(A \cup C) = 3$, $r(B \cup C) = 3$.
- $r(A \cup A_r) = 3$, $r(B \cup B_r) = 3$, $r(A \cup B_r) = 4$, $r(B \cup A_r) = 4$.
- $r(A \cup B) = 4$, $r(A \cap B) = 1 < r(C) = 2$.

Example: Rank function of a matrix

Consider the following 4×8 matrix, so $V = \{1,2,3,4,5,6,7,8\}$.

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{pmatrix} 0 & 2 & 2 & 3 & 0 & 1 & 3 & 1 \\ 0 & 3 & 0 & 4 & 0 & 0 & 2 & 4 \\ 0 & 0 & 0 & 0 & 3 & 0 & 0 & 5 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 5 \end{pmatrix} \end{array} = \begin{array}{c} \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{pmatrix} | & | & | & | & | & | & | & | \\ x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \\ | & | & | & | & | & | & | & | \end{pmatrix} \end{array}$$

- Let $A = \{1,2,3\}$, $B = \{3,4,5\}$, $C = \{6,7\}$, $A_r = \{1\}$, $B_r = \{5\}$.
- Then $r(A) = 3$, $r(B) = 3$, $r(C) = 2$.
- $r(A \cup C) = 3$, $r(B \cup C) = 3$.
- $r(A \cup A_r) = 3$, $r(B \cup B_r) = 3$, $r(A \cup B_r) = 4$, $r(B \cup A_r) = 4$.
- $r(A \cup B) = 4$, $r(A \cap B) = 1 < r(C) = 2$.

Example: Rank function of a matrix

Consider the following 4×8 matrix, so $V = \{1, 2, 3, 4, 5, 6, 7, 8\}$.

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{pmatrix} 0 & 2 & 2 & 3 & 0 & 1 & 3 & 1 \\ 0 & 3 & 0 & 4 & 0 & 0 & 2 & 4 \\ 0 & 0 & 0 & 0 & 3 & 0 & 0 & 5 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 5 \end{pmatrix} \end{array} = \begin{array}{c} \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{pmatrix} | & | & | & | & | & | & | & | \\ x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \\ | & | & | & | & | & | & | & | \end{pmatrix} \end{array}$$

- Let $A = \{1, 2, 3\}$, $B = \{3, 4, 5\}$, $C = \{6, 7\}$, $A_r = \{1\}$, $B_r = \{5\}$.
- Then $r(A) = 3$, $r(B) = 3$, $r(C) = 2$.
- $r(A \cup C) = 3$, $r(B \cup C) = 3$.
- $r(A \cup A_r) = 3$, $r(B \cup B_r) = 3$, $r(A \cup B_r) = 4$, $r(B \cup A_r) = 4$.
- $r(A \cup B) = 4$, $r(A \cap B) = 1 < r(C) = 2$.

Example: Rank function of a matrix

Consider the following 4×8 matrix, so $V = \{1, 2, 3, 4, 5, 6, 7, 8\}$.

$$\begin{array}{c c} & \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{pmatrix} 0 & 2 & 2 & 3 & 0 & 1 & 3 & 1 \\ 0 & 3 & 0 & 4 & 0 & 0 & 2 & 4 \\ 0 & 0 & 0 & 0 & 3 & 0 & 0 & 5 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 5 \end{pmatrix} \end{array} = \begin{array}{c} \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{pmatrix} | & | & | & | & | & | & | & | \\ x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \\ | & | & | & | & | & | & | & | \end{pmatrix} \end{array}$$

- Let $A = \{1, 2, 3\}$, $B = \{3, 4, 5\}$, $C = \{6, 7\}$, $A_r = \{1\}$, $B_r = \{5\}$.
- Then $r(A) = 3$, $r(B) = 3$, $r(C) = 2$.
- $r(A \cup C) = 3$, $r(B \cup C) = 3$.
- $r(A \cup A_r) = 3$, $r(B \cup B_r) = 3$, $r(A \cup B_r) = 4$, $r(B \cup A_r) = 4$.
- $r(A \cup B) = 4$, $r(A \cap B) = 1 < r(C) = 2$.

Example: Rank function of a matrix

Consider the following 4×8 matrix, so $V = \{1, 2, 3, 4, 5, 6, 7, 8\}$.

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{pmatrix} 0 & 2 & 2 & 3 & 0 & 1 & 3 & 1 \\ 0 & 3 & 0 & 4 & 0 & 0 & 2 & 4 \\ 0 & 0 & 0 & 0 & 3 & 0 & 0 & 5 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 5 \end{pmatrix} \end{array} = \begin{array}{c} \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{pmatrix} | & | & | & | & | & | & | & | \\ x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \\ | & | & | & | & | & | & | & | \end{pmatrix} \end{array}$$

- Let $A = \{1, 2, 3\}$, $B = \{3, 4, 5\}$, $C = \{6, 7\}$, $A_r = \{1\}$, $B_r = \{5\}$.
- Then $r(A) = 3$, $r(B) = 3$, $r(C) = 2$.
- $r(A \cup C) = 3$, $r(B \cup C) = 3$.
- $r(A \cup A_r) = 3$, $r(B \cup B_r) = 3$, $r(A \cup B_r) = 4$, $r(B \cup A_r) = 4$.
- $r(A \cup B) = 4$, $r(A \cap B) = 1 < r(C) = 2$.

Example: Rank function of a matrix

Consider the following 4×8 matrix, so $V = \{1, 2, 3, 4, 5, 6, 7, 8\}$.

$$\begin{array}{c c} & \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{pmatrix} 0 & 2 & 2 & 3 & 0 & 1 & 3 & 1 \\ 0 & 3 & 0 & 4 & 0 & 0 & 2 & 4 \\ 0 & 0 & 0 & 0 & 3 & 0 & 0 & 5 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 5 \end{pmatrix} \end{array} = \begin{array}{c} \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{pmatrix} | & | & | & | & | & | & | & | \\ x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \\ | & | & | & | & | & | & | & | \end{pmatrix} \end{array}$$

- Let $A = \{1, 2, 3\}$, $B = \{3, 4, 5\}$, $C = \{6, 7\}$, $A_r = \{1\}$, $B_r = \{5\}$.
- Then $r(A) = 3$, $r(B) = 3$, $r(C) = 2$.
- $r(A \cup C) = 3$, $r(B \cup C) = 3$.
- $r(A \cup A_r) = 3$, $r(B \cup B_r) = 3$, $r(A \cup B_r) = 4$, $r(B \cup A_r) = 4$.
- $r(A \cup B) = 4$, $r(A \cap B) = 1 < r(C) = 2$.

Example: Rank function of a matrix

Consider the following 4×8 matrix, so $V = \{1,2,3,4,5,6,7,8\}$.

$$\begin{array}{c|c} & \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{pmatrix} 0 & 2 & 2 & 3 & 0 & 1 & 3 & 1 \\ 0 & 3 & 0 & 4 & 0 & 0 & 2 & 4 \\ 0 & 0 & 0 & 0 & 3 & 0 & 0 & 5 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 5 \end{pmatrix} \end{array} = \begin{array}{c} \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{pmatrix} | & | & | & | & | & | & | & | \\ x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \\ | & | & | & | & | & | & | & | \end{pmatrix} \end{array}$$

- Let $A = \{1,2,3\}$, $B = \{3,4,5\}$, $C = \{6,7\}$, $A_r = \{1\}$, $B_r = \{5\}$.
- Then $r(A) = 3$, $r(B) = 3$, $r(C) = 2$.
- $r(A \cup C) = 3$, $r(B \cup C) = 3$.
- $r(A \cup A_r) = 3$, $r(B \cup B_r) = 3$, $r(A \cup B_r) = 4$, $r(B \cup A_r) = 4$.
- $r(A \cup B) = 4$, $r(A \cap B) = 1$ $< r(C) = 2.$

Example: Rank function of a matrix

Consider the following 4×8 matrix, so $V = \{1,2,3,4,5,6,7,8\}$.

$$\begin{array}{c|cccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \hline 1 & 0 & 2 & 2 & 3 & 0 & 1 & 3 & 1 \\ 2 & 0 & 3 & 0 & 4 & 0 & 0 & 2 & 4 \\ 3 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 5 \\ 4 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 5 \end{array} = \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \end{array} \begin{pmatrix} | & | & | & | & | & | & | & | \\ x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \\ | & | & | & | & | & | & | & | \end{pmatrix}$$

- Let $A = \{1,2,3\}$, $B = \{3,4,5\}$, $C = \{6,7\}$, $A_r = \{1\}$, $B_r = \{5\}$.
- Then $r(A) = 3$, $r(B) = 3$, $r(C) = 2$.
- $r(A \cup C) = 3$, $r(B \cup C) = 3$.
- $r(A \cup A_r) = 3$, $r(B \cup B_r) = 3$, $r(A \cup B_r) = 4$, $r(B \cup A_r) = 4$.
- $r(A \cup B) = 4$, $r(A \cap B) = 1 < r(C) = 2$.
- $6 = r(A) + r(B) > r(A \cup B) + r(A \cap B) = 5$

Rank function of a matrix

- Let $A, B \subseteq V$ be two subsets of column indices.

Rank function of a matrix

- Let $A, B \subseteq V$ be two subsets of column indices.
- The rank of the two sets unioned together $A \cup B$ is no more than the sum of the two individual ranks.

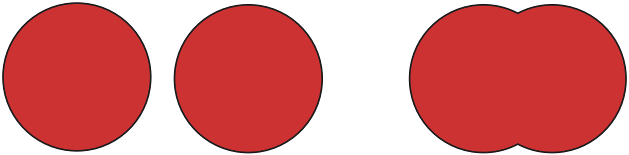
Rank function of a matrix

- Let $A, B \subseteq V$ be two subsets of column indices.
- The rank of the two sets unioned together $A \cup B$ is no more than the sum of the two individual ranks.
- In Venn diagram, Let area correspond to dimensions spanned by vectors indexed by a set. Hence, $r(A)$ can be viewed as an area.

Rank function of a matrix

- Let $A, B \subseteq V$ be two subsets of column indices.
- The rank of the two sets unioned together $A \cup B$ is no more than the sum of the two individual ranks.
- In Venn diagram, Let area correspond to dimensions spanned by vectors indexed by a set. Hence, $r(A)$ can be viewed as an area.

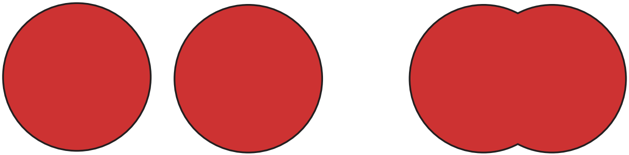
$$r(A) \quad + \quad r(B) \qquad \geq \qquad r(A \cup B)$$



Rank function of a matrix

- Let $A, B \subseteq V$ be two subsets of column indices.
- The rank of the two sets unioned together $A \cup B$ is no more than the sum of the two individual ranks.
- In Venn diagram, Let area correspond to dimensions spanned by vectors indexed by a set. Hence, $r(A)$ can be viewed as an area.

$$r(A) \quad + \quad r(B) \qquad \geq \qquad r(A \cup B)$$

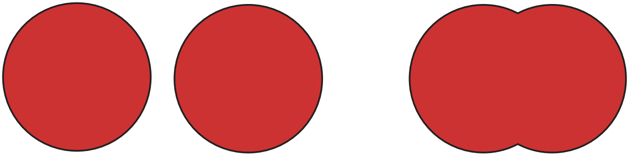


- If some of the dimensions spanned by A overlap some of the dimensions spanned by B (i.e., if $\exists$ common span), then that area is counted twice in $r(A) + r(B)$, so the inequality will be strict.

Rank function of a matrix

- Let $A, B \subseteq V$ be two subsets of column indices.
- The rank of the two sets unioned together $A \cup B$ is no more than the sum of the two individual ranks.
- In Venn diagram, Let area correspond to dimensions spanned by vectors indexed by a set. Hence, $r(A)$ can be viewed as an area.

$$r(A) \quad + \quad r(B) \qquad \geq \qquad r(A \cup B)$$



- If some of the dimensions spanned by A overlap some of the dimensions spanned by B (i.e., if $\exists$ <u>common span</u>), then that area is counted twice in $r(A) + r(B)$, so the inequality will be strict.
- Any function where the above inequality is true for all $A, B \subseteq V$ is called **subadditive**.

Rank functions of a matrix

- Vectors A and B have a (possibly empty) common span and two (possibly empty) non-common residual spans.

Rank functions of a matrix

- Vectors A and B have a (possibly empty) common span and two (possibly empty) non-common residual spans.
- Let C index vectors spanning dimensions common to A and B.

Rank functions of a matrix

- Vectors A and B have a (possibly empty) common span and two (possibly empty) non-common residual spans.
- Let C index vectors spanning dimensions common to A and B.
- Let A_r index vectors spanning dimensions spanned by A but not B.

Rank functions of a matrix

- Vectors A and B have a (possibly empty) common span and two (possibly empty) non-common residual spans.
- Let C index vectors spanning dimensions common to A and B.
- Let A_r index vectors spanning dimensions spanned by A but not B.
- Let B_r index vectors spanning dimensions spanned by B but not A.

Rank functions of a matrix

- Vectors A and B have a (possibly empty) common span and two (possibly empty) non-common residual spans.
- Let C index vectors spanning dimensions common to A and B.
- Let A_r index vectors spanning dimensions spanned by A but not B.
- Let B_r index vectors spanning dimensions spanned by B but not A.
- Then, $r(A) = r(C) + r(A_r)$

Rank functions of a matrix

- Vectors A and B have a (possibly empty) common span and two (possibly empty) non-common residual spans.
- Let C index vectors spanning dimensions common to A and B.
- Let A_r index vectors spanning dimensions spanned by A but not B.
- Let B_r index vectors spanning dimensions spanned by B but not A.
- Then, $r(A) = r(C) + r(A_r)$
- Similarly, $r(B) = r(C) + r(B_r)$.

Rank functions of a matrix

- Vectors A and B have a (possibly empty) common span and two (possibly empty) non-common residual spans.
- Let C index vectors spanning dimensions common to A and B.
- Let A_r index vectors spanning dimensions spanned by A but not B.
- Let B_r index vectors spanning dimensions spanned by B but not A.
- Then, $r(A) = r(C) + r(A_r)$
- Similarly, $r(B) = r(C) + r(B_r)$.
- Then $r(A) + r(B)$ counts the dimensions spanned by C twice, i.e.,

$$r(A) + r(B) = r(A_r) + 2r(C) + r(B_r). \tag{61}$$

Rank functions of a matrix

- Vectors A and B have a (possibly empty) common span and two (possibly empty) non-common residual spans.
- Let C index vectors spanning dimensions common to A and B.
- Let A_r index vectors spanning dimensions spanned by A but not B.
- Let B_r index vectors spanning dimensions spanned by B but not A.
- Then, $r(A) = r(C) + r(A_r)$
- Similarly, $r(B) = r(C) + r(B_r)$.
- Then $r(A) + r(B)$ counts the dimensions spanned by C twice, i.e.,

$$r(A) + r(B) = r(A_r) + 2r(C) + r(B_r). \tag{61}$$

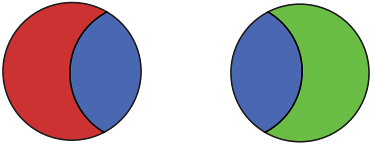
- But $r(A \cup B)$ counts the dimensions spanned by C only once.

$$r(A \cup B) = r(A_r) + r(C) + r(B_r) \tag{62}$$

Rank functions of a matrix

- Then $r(A) + r(B)$ counts the dimensions spanned by C twice, i.e.,

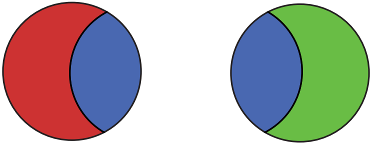
$$r(A) + r(B) = r(A_r) + 2r(C) + r(B_r)$$



Rank functions of a matrix

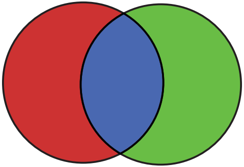
- Then $r(A) + r(B)$ counts the dimensions spanned by C twice, i.e.,

$$r(A) + r(B) = r(A_r) + 2r(C) + r(B_r)$$



- But $r(A \cup B)$ counts the dimensions spanned by C only once.

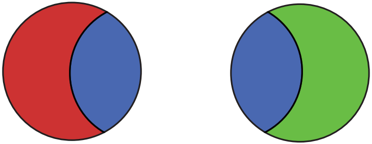
$$r(A \cup B) = r(A_r) + r(C) + r(B_r)$$



Rank functions of a matrix

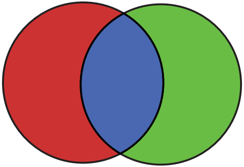
- Then $r(A) + r(B)$ counts the dimensions spanned by C twice, i.e.,

$$r(A) + r(B) = r(A_r) + 2r(C) + r(B_r)$$



- But $r(A \cup B)$ counts the dimensions spanned by C only once.

$$r(A \cup B) = r(A_r) + r(C) + r(B_r)$$

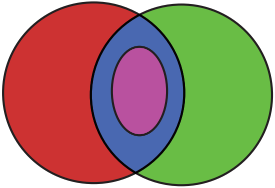


- Thus, we have subadditivity: $r(A) + r(B) \geq r(A \cup B)$. Can we add more to the r.h.s. and still have an inequality? Yes.

Rank function of a matrix

- Note, $r(A \cap B) \leq r(C)$. Why? Vectors indexed by $A \cap B$ (i.e., the common index set) span no more than the dimensions commonly spanned by A and B (namely, those spanned by the professed C).

$$r(C) \geq r(A \cap B)$$

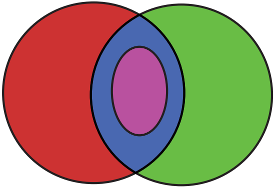


In short:

Rank function of a matrix

- Note, $r(A \cap B) \leq r(C)$. Why? Vectors indexed by $A \cap B$ (i.e., the common index set) span no more than the dimensions commonly spanned by A and B (namely, those spanned by the professed C).

$$r(C) \geq r(A \cap B)$$



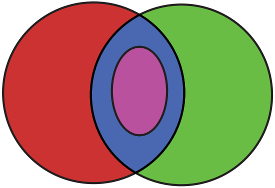
In short:

- Common span (blue) is "more" (no less) than span of common index (magenta).

Rank function of a matrix

- Note, $r(A \cap B) \leq r(C)$. Why? Vectors indexed by $A \cap B$ (i.e., the common index set) span no more than the dimensions commonly spanned by A and B (namely, those spanned by the professed C).

$$r(C) \geq r(A \cap B)$$

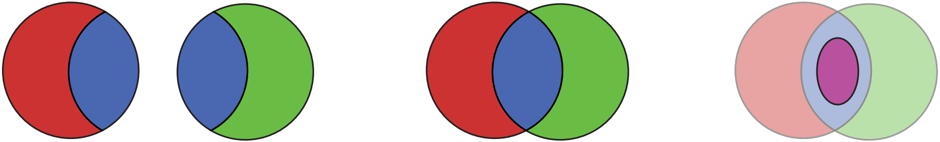


In short:

- Common span (blue) is "more" (no less) than span of common index (magenta).
- More generally, common information (blue) is "more" (no less) than information within common index (magenta).

The Venn and Art of Submodularity

$$\underbrace{r(A) + r(B)}_{= r(A_r) + 2r(C) + r(B_r)} \geq \underbrace{r(A \cup B)}_{= r(A_r) + r(C) + r(B_r)} + \underbrace{r(A \cap B)}_{= r(A \cap B)}$$



Polymatroid function and its polyhedron.

Definition

A polymatroid function is a real-valued function f defined on subsets of V which is normalized, non-decreasing, and submodular. That is:

1. $f(\emptyset) = 0$ (normalized)
2. $f(A) \leq f(B)$ for any $A \subseteq B \subseteq V$ (monotone non-decreasing)
3. $f(A \cup B) + f(A \cap B) \leq f(A) + f(B)$ for any $A, B \subseteq V$ (submodular)

We can define the polyhedron P_f^+ associated with a polymatroid function as follows

$$P_f^+ = \left\{ y \in \mathbb{R}_+^V : y(A) \leq f(A) \text{ for all } A \subseteq V \right\} \tag{63}$$

$$= \left\{ y \in \mathbb{R}^V : y \geq 0, y(A) \leq f(A) \text{ for all } A \subseteq V \right\} \tag{64}$$

Chains of sets

- Ground element $V = \{1, 2, \ldots, n\}$ set of integers w.l.o.g.

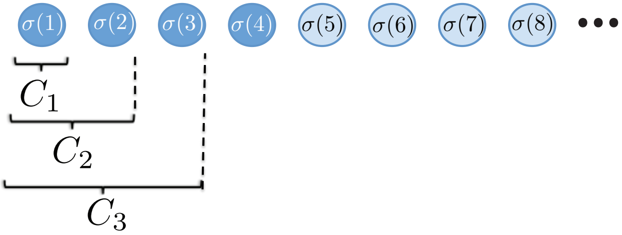
Chains of sets

- Ground element $V = \{1, 2, \ldots, n\}$ set of integers w.l.o.g.
- Given a permutation $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_n)$ of the integers.

Chains of sets

- Ground element $V = \{1, 2, \ldots, n\}$ set of integers w.l.o.g.
- Given a permutation $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_n)$ of the integers.
- From this we can form a chain of sets $\{C_i\}_i$ with $\emptyset = C_0 \subseteq C_1 \subseteq \cdots \subseteq C_n = V$ formed as:

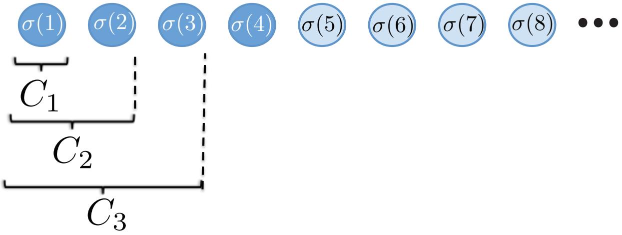
$$C_i = \{\sigma_1, \sigma_2, \ldots, \sigma_i\}, \quad \text{for } i = 1 \ldots n \tag{65}$$



Chains of sets

- Ground element $V = \{1, 2, \ldots, n\}$ set of integers w.l.o.g.
- Given a permutation $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_n)$ of the integers.
- From this we can form a chain of sets $\{C_i\}_i$ with $\emptyset = C_0 \subseteq C_1 \subseteq \cdots \subseteq C_n = V$ formed as:

$$C_i = \{\sigma_1, \sigma_2, \ldots, \sigma_i\}, \quad \text{for } i = 1 \ldots n \tag{65}$$



- Can also form a chain from a vector $w \in \mathbb{R}^V$ sorted in descending order. Choose σ so that $w(\sigma_1) \geq w(\sigma_2) \geq \cdots \geq w(\sigma_n)$.

Gain

- We often wish to express the gain of an item $j \in V$ in context A, namely $f(A \cup \{j\}) - f(A)$.

Gain

- We often wish to express the gain of an item $j \in V$ in context A, namely $f(A \cup \{j\}) - f(A)$.
- This is called the gain and is used so often, there are equally as many ways to notate this. I.e., you might see:

$$f(A \cup \{j\}) - f(A) \triangleq \rho_j(A) \tag{66}$$

$$\triangleq \rho_A(j) \tag{67}$$

$$\triangleq \nabla_j f(A) \tag{68}$$

$$\triangleq f(\{j\}|A) \tag{69}$$

$$\triangleq f(j|A) \tag{70}$$

Gain

- We often wish to express the gain of an item $j \in V$ in context A, namely $f(A \cup \{j\}) - f(A)$.
- This is called the gain and is used so often, there are equally as many ways to notate this. I.e., you might see:

$$
\begin{aligned}
f(A \cup \{j\}) - f(A) &\triangleq \rho_j(A) && (66)\\
&\triangleq \rho_A(j) && (67)\\
&\triangleq \nabla_j f(A) && (68)\\
&\triangleq f(\{j\}|A) && (69)\\
&\triangleq f(j|A) && (70)
\end{aligned}
$$

- We'll use $f(j|A)$. Also, $f(A|B) = f(A \cup B) - f(B)$.

Gain

- We often wish to express the gain of an item $j \in V$ in context A, namely $f(A \cup \{j\}) - f(A)$.
- This is called the gain and is used so often, there are equally as many ways to notate this. I.e., you might see:

$$f(A \cup \{j\}) - f(A) \triangleq \rho_j(A) \tag{66}$$
$$\triangleq \rho_A(j) \tag{67}$$
$$\triangleq \nabla_j f(A) \tag{68}$$
$$\triangleq f(\{j\}|A) \tag{69}$$
$$\triangleq f(j|A) \tag{70}$$

- We'll use $f(j|A)$. Also, $f(A|B) = f(A \cup B) - f(B)$.
- Submodularity's diminishing returns definition can be stated as saying that $f(j|A)$ is a monotone non-increasing function of A, since $f(j|A) \geq f(j|B)$ whenever $A \subseteq B$ (conditioning reduces valuation).

Polymatroidal polyhedron and greedy

- Suppose we wish to solve the following linear programming problem:

$$\begin{array}{ll} \underset{x \in \mathbb{R}^V}{\text{maximize}} & w^\intercal x \\ \text{subject to} & x \in \left\{ y \in \mathbb{R}_+^V : y(A) \leq f(A) \text{ for all } A \subseteq V \right\} \end{array} \tag{71}$$

or more simply put, $\max(wx : x \in P_f)$.

Polymatroidal polyhedron and greedy

- Suppose we wish to solve the following linear programming problem:

$$\begin{array}{ll} \underset{x \in \mathbb{R}^V}{\text{maximize}} & w^\intercal x \\ \text{subject to} & x \in \left\{ y \in \mathbb{R}_+^V : y(A) \leq f(A) \text{ for all } A \subseteq V \right\} \end{array} \tag{71}$$

  or more simply put, $\max(wx : x \in P_f)$.
- Consider greedy solution: sort elements of V w.r.t. w so that w.l.o.g. $V = (v_1, v_2, \ldots, v_m)$ has $w(v_1) \geq w(v_2) \geq \cdots \geq w(v_m)$.

Polymatroidal polyhedron and greedy

- Suppose we wish to solve the following linear programming problem:

$$\begin{aligned} \underset{x \in \mathbb{R}^V}{\text{maximize}} \quad & w^\intercal x \\ \text{subject to} \quad & x \in \left\{ y \in \mathbb{R}_+^V : y(A) \leq f(A) \text{ for all } A \subseteq V \right\} \end{aligned} \tag{71}$$

  or more simply put, $\max(wx : x \in P_f)$.
- Consider greedy solution: sort elements of V w.r.t. w so that w.l.o.g. $V = (v_1, v_2, \ldots, v_m)$ has $w(v_1) \geq w(v_2) \geq \cdots \geq w(v_m)$.
- Next, form chain of sets based on w sorted descended, giving:

$$V_i \stackrel{\text{def}}{=} \{v_1, v_2, \ldots v_i\} \tag{72}$$

  for $i = 0 \ldots m$. Note $V_0 = \emptyset$, and $f(V_0) = 0$.

Polymatroidal polyhedron and greedy

- Suppose we wish to solve the following linear programming problem:

$$\begin{aligned} &\underset{x \in \mathbb{R}^V}{\text{maximize}} && w^\intercal x \\ &\text{subject to} && x \in \left\{ y \in \mathbb{R}_+^V : y(A) \le f(A) \text{ for all } A \subseteq V \right\} \end{aligned} \tag{71}$$

  or more simply put, $\max(wx : x \in P_f)$.
- Consider greedy solution: sort elements of V w.r.t. w so that w.l.o.g. $V = (v_1, v_2, \dots, v_m)$ has $w(v_1) \ge w(v_2) \ge \dots \ge w(v_m)$.
- Next, form chain of sets based on w sorted descended, giving:

$$V_i \stackrel{\text{def}}{=} \{v_1, v_2, \dots v_i\} \tag{72}$$

  for $i = 0 \dots m$. Note $V_0 = \emptyset$, and $f(V_0) = 0$.
- The greedy solution is the vector $x \in \mathbb{R}_+^V$ with element $x(v_i)$ for $i = 1, \dots, n$ defined as:

$$x(v_i) = f(V_i) - f(V_{i-1}) = f(v_i | V_{i-1}) \tag{73}$$

Polymatroidal polyhedron and greedy

- We have the following very powerful result (which generalizes a similar one that is true for matroids).

Theorem

Let $f : 2^V \to \mathbb{R}_+$ be a given set function, and P is a polytope in $\mathbb{R}_+^V$ of the form $P = \{x \in \mathbb{R}_+^V : x(A) \leq f(A), \forall A \subseteq V\}$.
Then the greedy solution to the problem $\max(wx : x \in P)$ is optimal $\forall w$ iff f is monotone non-decreasing submodular (i.e., iff P is a polymatroid).

Polymatroid extreme points

Greedy does more than this. In fact, we have:

Theorem

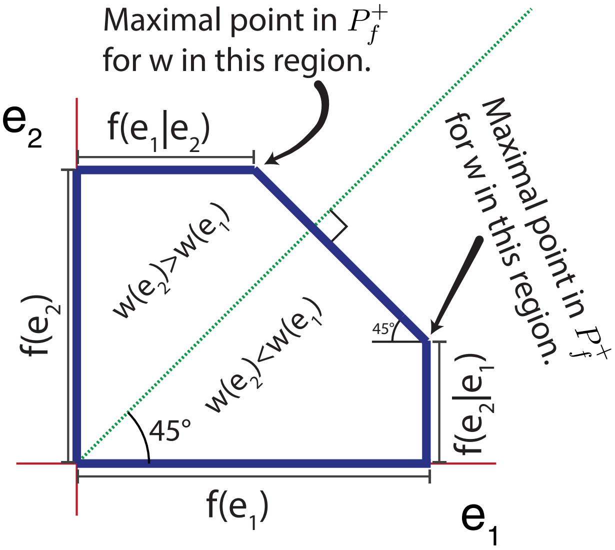
For a given ordering $V = (v_1, \dots, v_m)$ of V and a given V_i and x generated by V_i using the greedy procedure, then x is an extreme point of P_f

Corollary

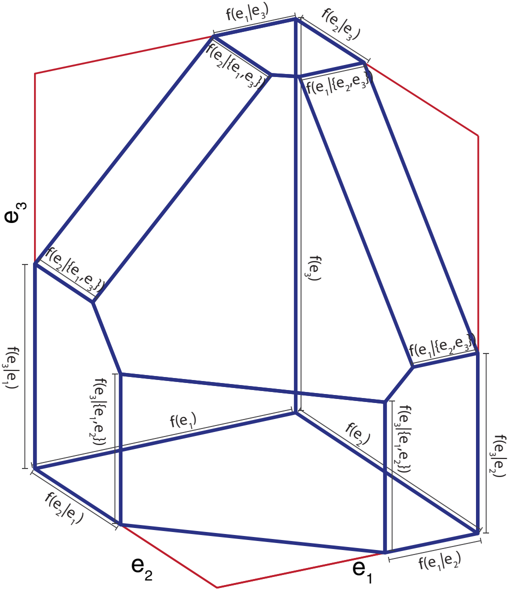
If x is an extreme point of P_f and $B \subseteq V$ is given such that $\{v \in V : x(v) \neq 0\} \subseteq B \subseteq \cup(A : x(A) = f(A))$, then x is generated using greedy by some ordering of B.

Intuition: why greedy works with polymatroids

- Given w, the goal is to find $x = (x(e_1), x(e_2))$ that maximizes $x^\intercal w = x(e_1)w(e_1) + x(e_2)w(e_2)$.
- If $w(e_2) > w(e_1)$ the upper extreme point indicated maximizes $x^\intercal w$ over $x \in P_f^+$.
- If $w(e_2) < w(e_1)$ the lower extreme point indicated maximizes $x^\intercal w$ over $x \in P_f^+$.



Polymatroid with labeled edge lengths



A polymatroid function's polyhedron vs. a polymatroid.

- Given these results, we can conclude that a polymatroid is really an extremely natural polyhedral generalization of a matroid. This was all realized by Jack Edmonds in the mid 1960s (and published in 1969 in his landmark paper "Submodular Functions, Matroids, and Certain Polyhedra").

- Jack Edmonds NIPS talk, 2011 `http://videolectures.net/nipsworkshops2011_edmonds_polymatroids/`

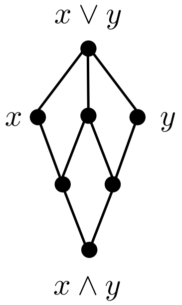
Outline: Part 2

4 From Matroids to Polymatroids
- Matrix Rank
- Venn Diagrams
- Matroids

5 Submodular Definitions, Examples, and Properties
- Normalization
- Submodular Definitions
- Submodular Composition
- More Examples

Submodular (or Upper-SemiModular) Lattices

The name "Submodular" comes from lattice theory, and refers to a property of the "height" function of an upper-semimodular lattice. Ex: consider the following lattice over 7 elements.

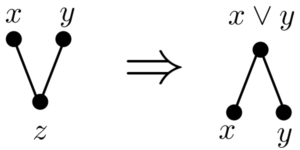


height
3
2
1
0

submodularity
$h(x)+h(y)$
$> h(x \vee y)$
$+ h(x \wedge y)$

$2 + 2 > 3 + 0$

- Such lattices require that for all x, y, z,



- The lattice is upper-semimodular (submodular), height function is submodular on the lattice.

Submodular Definitions

Definition (submodular)

A function $f : 2^V \to \mathbb{R}$ is submodular if for any $A, B \subseteq V$, we have that:

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B) \tag{74}$$

- General submodular function, f need not be monotone, non-negative, nor normalized (i.e., $f(\emptyset)$ need not be $= 0$).

Normalized Submodular Function

- Given any submodular function $f : 2^V \to \mathbb{R}$, form a normalized variant $f' : 2^V \to \mathbb{R}$, with

$$f'(A) = f(A) - f(\emptyset) \tag{75}$$

- Then $f'(\emptyset) = 0$.
- This operation does not affect submodularity, or any minima or maxima
- It is often assumed that all submodular functions are so normalized.

Submodular Polymatroidal Decomposition

- Given any arbitrary submodular function $f : 2^V \to \mathbb{R}$, consider the identity

$$f(A) = \underbrace{f(A) - m(A)}_{\bar{f}(A)} + m(A) = \bar{f}(A) + m(A) \tag{76}$$

for a modular function $m : 2^V \to \mathbb{R}$, where

$$m(a) = f(a|V \setminus \{a\}) \tag{77}$$

Submodular Polymatroidal Decomposition

- Given any arbitrary submodular function $f : 2^V \to \mathbb{R}$, consider the identity

$$f(A) = \underbrace{f(A) - m(A)}_{\bar{f}(A)} + m(A) = \bar{f}(A) + m(A) \tag{76}$$

for a modular function $m : 2^V \to \mathbb{R}$, where

$$m(a) = f(a|V \setminus \{a\}) \tag{77}$$

- Then $\bar{f}(A)$ is polymatroidal since $\bar{f}(\emptyset) = 0$ and for any a and A

$$\bar{f}(a|A) = f(a|A) - f(a|V \setminus \{a\}) \geq 0 \tag{78}$$

Totally Normalized

- $\bar{f}$ is called the totally normalized version of f

Totally Normalized

- $\bar{f}$ is called the totally normalized version of f
- polytope of $\bar{f}$ and f is the same shape, just shifted.

$$P_f = \left\{ x \in \mathbb{R}^V : x(A) \leq f(A), \forall A \subseteq V \right\} \tag{79}$$

$$= \left\{ x \in \mathbb{R}^V : x(A) \leq \bar{f}(A) + m(A), \forall A \subseteq V \right\} \tag{80}$$

Totally Normalized

- $\bar{f}$ is called the totally normalized version of f
- polytope of $\bar{f}$ and f is the same shape, just shifted.

$$P_f = \left\{x \in \mathbb{R}^V : x(A) \leq f(A), \forall A \subseteq V\right\} \tag{79}$$

$$= \left\{x \in \mathbb{R}^V : x(A) \leq \bar{f}(A) + m(A), \forall A \subseteq V\right\} \tag{80}$$

- m is like a unary score, $\bar{f}$ is where things interact . All of the real structure is in $\bar{f}$

Totally Normalized

- $\bar{f}$ is called the totally normalized version of f
- polytope of $\bar{f}$ and f is the same shape, just shifted.

$$P_f = \left\{x \in \mathbb{R}^V : x(A) \leq f(A), \forall A \subseteq V\right\} \tag{79}$$
$$= \left\{x \in \mathbb{R}^V : x(A) \leq \bar{f}(A) + m(A), \forall A \subseteq V\right\} \tag{80}$$

- m is like a unary score, $\bar{f}$ is where things interact . All of the real structure is in $\bar{f}$
- Hence, any submodular function is a sum of polymatroid and modular.

Telescoping Summation

- Given a chain set of sets $A_1 \subseteq A_2 \subseteq \cdots \subseteq A_r$

Telescoping Summation

- Given a chain set of sets $A_1 \subseteq A_2 \subseteq \cdots \subseteq A_r$
- Then the telescoping summation property of the gains is as follows:

$$\sum_{i=1}^{r-1} f(A_{i+1}|A_i) = \sum_{i=2}^{r} f(A_i) - \sum_{i=1}^{r-1} f(A_i) = f(A_r) - f(A_1) \tag{81}$$

Submodular Definitions

Theorem

Given function $f : 2^V \to \mathbb{R}$, *then*

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B) \text{ for all } A, B \subseteq V \qquad \text{(SC)}$$

if and only if

$$f(v|X) \geq f(v|Y) \text{ for all } X \subseteq Y \subseteq V \text{ and } v \notin B \qquad \text{(DR)}$$

Submodular Definitions

Theorem

Given function $f : 2^V \to \mathbb{R}$, *then*

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B) \text{ for all } A, B \subseteq V \qquad \text{(SC)}$$

if and only if

$$f(v|X) \geq f(v|Y) \text{ for all } X \subseteq Y \subseteq V \text{ and } v \notin B \qquad \text{(DR)}$$

Proof.

(SC)⇒(DR): Set $A \leftarrow X \cup \{v\}$, $B \leftarrow Y$. Then $A \cup B = B \cup \{v\}$ and $A \cap B = X$ and $f(A) - f(A \cap B) \geq f(A \cup B) - f(B)$ implies (DR).

(DR)⇒(SC): Order $A \setminus B = \{v_1, v_2, \ldots, v_r\}$ arbitrarily. Then $f(v_i|A \cap B \cup \{v_1, v_2, \ldots, v_{i-1}\}) \geq f(v_1|B \cup \{v_1, v_2, \ldots, v_{i-1}\}),\ i \in [r-1]$
Applying telescoping summation to both sides, we get:

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

□

Many (Equivalent) Definitions of Submodularity

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B), \quad \forall A, B \subseteq V \tag{82}$$

Many (Equivalent) Definitions of Submodularity

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B), \quad \forall A, B \subseteq V \tag{82}$$

$$f(j|S) \geq f(j|T), \; \forall S \subseteq T \subseteq V, \text{ with } j \in V \setminus T \tag{83}$$

Many (Equivalent) Definitions of Submodularity

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B), \quad \forall A, B \subseteq V \tag{82}$$

$$f(j|S) \geq f(j|T), \; \forall S \subseteq T \subseteq V, \; \text{with } j \in V \setminus T \tag{83}$$

$$f(C|S) \geq f(C|T), \forall S \subseteq T \subseteq V, \; \text{with } C \subseteq V \setminus T \tag{84}$$

Many (Equivalent) Definitions of Submodularity

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B), \quad \forall A, B \subseteq V \tag{82}$$

$$f(j|S) \geq f(j|T), \ \forall S \subseteq T \subseteq V, \text{ with } j \in V \setminus T \tag{83}$$

$$f(C|S) \geq f(C|T), \forall S \subseteq T \subseteq V, \text{ with } C \subseteq V \setminus T \tag{84}$$

$$f(j|S) \geq f(j|S \cup \{k\}), \ \forall S \subseteq V \text{ with } j \in V \setminus (S \cup \{k\}) \tag{85}$$

Many (Equivalent) Definitions of Submodularity

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B), \ \ \forall A, B \subseteq V \tag{82}$$

$$f(j|S) \geq f(j|T), \ \forall S \subseteq T \subseteq V, \text{ with } j \in V \setminus T \tag{83}$$

$$f(C|S) \geq f(C|T), \forall S \subseteq T \subseteq V, \text{ with } C \subseteq V \setminus T \tag{84}$$

$$f(j|S) \geq f(j|S \cup \{k\}), \ \forall S \subseteq V \text{ with } j \in V \setminus (S \cup \{k\}) \tag{85}$$

$$f(A \cup B|A \cap B) \leq f(A|A \cap B) + f(B|A \cap B), \ \ \forall A, B \subseteq V \tag{86}$$

Many (Equivalent) Definitions of Submodularity

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B), \quad \forall A, B \subseteq V \tag{82}$$

$$f(j|S) \geq f(j|T), \ \forall S \subseteq T \subseteq V, \text{ with } j \in V \setminus T \tag{83}$$

$$f(C|S) \geq f(C|T), \forall S \subseteq T \subseteq V, \text{ with } C \subseteq V \setminus T \tag{84}$$

$$f(j|S) \geq f(j|S \cup \{k\}), \ \forall S \subseteq V \text{ with } j \in V \setminus (S \cup \{k\}) \tag{85}$$

$$f(A \cup B|A \cap B) \leq f(A|A \cap B) + f(B|A \cap B), \quad \forall A, B \subseteq V \tag{86}$$

$$f(T) \leq f(S) + \sum_{j \in T \setminus S} f(j|S) - \sum_{j \in S \setminus T} f(j|S \cup T - \{j\}), \ \forall S, T \subseteq V \tag{87}$$

Many (Equivalent) Definitions of Submodularity

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B), \quad \forall A, B \subseteq V \tag{82}$$

$$f(j|S) \geq f(j|T), \ \forall S \subseteq T \subseteq V, \text{ with } j \in V \setminus T \tag{83}$$

$$f(C|S) \geq f(C|T), \forall S \subseteq T \subseteq V, \text{ with } C \subseteq V \setminus T \tag{84}$$

$$f(j|S) \geq f(j|S \cup \{k\}), \ \forall S \subseteq V \text{ with } j \in V \setminus (S \cup \{k\}) \tag{85}$$

$$f(A \cup B|A \cap B) \leq f(A|A \cap B) + f(B|A \cap B), \quad \forall A, B \subseteq V \tag{86}$$

$$f(T) \leq f(S) + \sum_{j \in T \setminus S} f(j|S) - \sum_{j \in S \setminus T} f(j|S \cup T - \{j\}), \ \forall S, T \subseteq V \tag{87}$$

$$f(T) \leq f(S) + \sum_{j \in T \setminus S} f(j|S), \ \forall S \subseteq T \subseteq V \tag{88}$$

Many (Equivalent) Definitions of Submodularity

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B), \quad \forall A, B \subseteq V \tag{82}$$

$$f(j|S) \geq f(j|T), \ \forall S \subseteq T \subseteq V, \ \text{with } j \in V \setminus T \tag{83}$$

$$f(C|S) \geq f(C|T), \forall S \subseteq T \subseteq V, \ \text{with } C \subseteq V \setminus T \tag{84}$$

$$f(j|S) \geq f(j|S \cup \{k\}), \ \forall S \subseteq V \text{ with } j \in V \setminus (S \cup \{k\}) \tag{85}$$

$$f(A \cup B|A \cap B) \leq f(A|A \cap B) + f(B|A \cap B), \quad \forall A, B \subseteq V \tag{86}$$

$$f(T) \leq f(S) + \sum_{j \in T \setminus S} f(j|S) - \sum_{j \in S \setminus T} f(j|S \cup T - \{j\}), \ \forall S, T \subseteq V \tag{87}$$

$$f(T) \leq f(S) + \sum_{j \in T \setminus S} f(j|S), \ \forall S \subseteq T \subseteq V \tag{88}$$

$$f(T) \leq f(S) - \sum_{j \in S \setminus T} f(j|S \setminus \{j\}) + \sum_{j \in T \setminus S} f(j|S \cap T) \ \forall S, T \subseteq V \tag{89}$$

Many (Equivalent) Definitions of Submodularity

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B), \quad \forall A, B \subseteq V \tag{82}$$

$$f(j|S) \geq f(j|T), \; \forall S \subseteq T \subseteq V, \text{ with } j \in V \setminus T \tag{83}$$

$$f(C|S) \geq f(C|T), \forall S \subseteq T \subseteq V, \text{ with } C \subseteq V \setminus T \tag{84}$$

$$f(j|S) \geq f(j|S \cup \{k\}), \; \forall S \subseteq V \text{ with } j \in V \setminus (S \cup \{k\}) \tag{85}$$

$$f(A \cup B|A \cap B) \leq f(A|A \cap B) + f(B|A \cap B), \quad \forall A, B \subseteq V \tag{86}$$

$$f(T) \leq f(S) + \sum_{j \in T \setminus S} f(j|S) - \sum_{j \in S \setminus T} f(j|S \cup T - \{j\}), \; \forall S, T \subseteq V \tag{87}$$

$$f(T) \leq f(S) + \sum_{j \in T \setminus S} f(j|S), \; \forall S \subseteq T \subseteq V \tag{88}$$

$$f(T) \leq f(S) - \sum_{j \in S \setminus T} f(j|S \setminus \{j\}) + \sum_{j \in T \setminus S} f(j|S \cap T) \; \forall S, T \subseteq V \tag{89}$$

$$f(T) \leq f(S) - \sum_{j \in S \setminus T} f(j|S \setminus \{j\}), \; \forall T \subseteq S \subseteq V \tag{90}$$

Basic ops: Sums, Restrictions, Conditioning

- Given submodular $f_1, f_2, \ldots, f_k$ each $\in 2^V \to \mathbb{R}$, then conic combinations are submodular. I.e.,

$$f(A) = \sum_{i=1}^{k} \alpha_i f_i(A) \tag{91}$$

where $\alpha_i \geq 0$.

Basic ops: Sums, Restrictions, Conditioning

- Given submodular $f_1, f_2, \ldots, f_k$ each $\in 2^V \to \mathbb{R}$, then conic combinations are submodular. I.e.,

$$f(A) = \sum_{i=1}^{k} \alpha_i f_i(A) \tag{91}$$

where $\alpha_i \geq 0$.

- Restrictions: $f(A) = g(A \cap C)$ is submodular whenever g is, for all C.

Basic ops: Sums, Restrictions, Conditioning

- Given submodular $f_1, f_2, \ldots, f_k$ each $\in 2^V \to \mathbb{R}$, then conic combinations are submodular. I.e.,

$$f(A) = \sum_{i=1}^{k} \alpha_i f_i(A) \tag{91}$$

  where $\alpha_i \geq 0$.
- Restrictions: $f(A) = g(A \cap C)$ is submodular whenever g is, for all C.
- Conditioning: $f(A) = g(A \cup C) - f(C) = f(A|C)$ is submodular whenever g is for all C.

The "or" of two polymatroid functions

- Given two polymatroid functions f and g, suppose feasible A are defined as $\{A : f(A) \geq \alpha_f \text{ or } g(A) \geq \alpha_g\}$ for real α_f, α_g.

The "or" of two polymatroid functions

- Given two polymatroid functions f and g, suppose feasible A are defined as $\{A : f(A) \geq \alpha_f \text{ or } g(A) \geq \alpha_g\}$ for real α_f, α_g.
- This is identical to: $\{A : \bar{f}(A) = \alpha_f \text{ or } \bar{g}(A) = \alpha_g\}$ where $\bar{f}(A) = \min(f(A), \alpha_f)$ and $\bar{g}(A) = \min(f(A), \alpha_g)$

The "or" of two polymatroid functions

- Given two polymatroid functions f and g, suppose feasible A are defined as $\{A : f(A) \geq \alpha_f \text{ or } g(A) \geq \alpha_g\}$ for real α_f, α_g.
- This is identical to: $\left\{A : \bar{f}(A) = \alpha_f \text{ or } \bar{g}(A) = \alpha_g\right\}$ where $\bar{f}(A) = \min(f(A), \alpha_f)$ and $\bar{g}(A) = \min(f(A), \alpha_g)$
- Define: $h(A) = \bar{f}(A)\bar{g}(V) + \bar{f}(V)\bar{g}(A) - \bar{f}(A)\bar{g}(A)$.

The "or" of two polymatroid functions

- Given two polymatroid functions f and g, suppose feasible A are defined as $\{A : f(A) \geq \alpha_f \text{ or } g(A) \geq \alpha_g\}$ for real α_f, α_g.
- This is identical to: $\left\{A : \bar{f}(A) = \alpha_f \text{ or } \bar{g}(A) = \alpha_g\right\}$ where $\bar{f}(A) = \min(f(A), \alpha_f)$ and $\bar{g}(A) = \min(f(A), \alpha_g)$
- Define: $h(A) = \bar{f}(A)\bar{g}(V) + \bar{f}(V)\bar{g}(A) - \bar{f}(A)\bar{g}(A)$.

Theorem (Guillory & Bilmes, 2011)

$h(A)$ so defined is polymatroidal.

The "or" of two polymatroid functions

- Given two polymatroid functions f and g, suppose feasible A are defined as $\{A : f(A) \geq \alpha_f \text{ or } g(A) \geq \alpha_g\}$ for real α_f, α_g.
- This is identical to: $\left\{A : \bar{f}(A) = \alpha_f \text{ or } \bar{g}(A) = \alpha_g\right\}$ where $\bar{f}(A) = \min(f(A), \alpha_f)$ and $\bar{g}(A) = \min(f(A), \alpha_g)$
- Define: $h(A) = \bar{f}(A)\bar{g}(V) + \bar{f}(V)\bar{g}(A) - \bar{f}(A)\bar{g}(A)$.

Theorem (Guillory & Bilmes, 2011)

$h(A)$ so defined is polymatroidal.

Theorem

$h(A) = \alpha_f \alpha_g$ if and only if $\bar{f}(A) = \alpha_f$ or $\bar{g}(A) = \alpha_g$

The "or" of two polymatroid functions

- Given two polymatroid functions f and g, suppose feasible A are defined as $\{A : f(A) \geq \alpha_f \text{ or } g(A) \geq \alpha_g\}$ for real α_f, α_g.
- This is identical to: $\left\{A : \bar{f}(A) = \alpha_f \text{ or } \bar{g}(A) = \alpha_g\right\}$ where $\bar{f}(A) = \min(f(A), \alpha_f)$ and $\bar{g}(A) = \min(f(A), \alpha_g)$
- Define: $h(A) = \bar{f}(A)\bar{g}(V) + \bar{f}(V)\bar{g}(A) - \bar{f}(A)\bar{g}(A)$.

Theorem (Guillory & Bilmes, 2011)

$h(A)$ so defined is polymatroidal.

Theorem

$h(A) = \alpha_f \alpha_g$ if and only if $\bar{f}(A) = \alpha_f$ or $\bar{g}(A) = \alpha_g$

- Therefore, h can be used as a submodular surrogate for the "or" of multiple submodular functions.

Composition and Submodular Functions

- Convex/Concave have many nice properties of composition (see Boyd & Vandenberghe, "Convex Optimization")

Composition and Submodular Functions

- Convex/Concave have many nice properties of composition (see Boyd & Vandenberghe, "Convex Optimization")
- A submodular function $f : 2^V \to \mathbb{R}$ has a different type of input and output, so composing two submodular functions directly makes no sense.

Composition and Submodular Functions

- Convex/Concave have many nice properties of composition (see Boyd & Vandenberghe, "Convex Optimization")
- A submodular function $f : 2^V \to \mathbb{R}$ has a different type of input and output, so composing two submodular functions directly makes no sense.
- However, we have a number of forms of composition results that preserve submodularity, which we turn to next:

Grouping elements, set cover, and bipartite neighborhoods

- Given submodular $f : 2^V \to \mathbb{R}$ and a grouping of $V = V_1 \cup V_2 \cup \cdots \cup V_k$ into k possibly overlapping clusters.

Grouping elements, set cover, and bipartite neighborhoods

- Given submodular $f : 2^V \to \mathbb{R}$ and a grouping of $V = V_1 \cup V_2 \cup \dots \cup V_k$ into k possibly overlapping clusters.
- Define new function $g : 2^{[k]} \to \mathbb{R}$ where $\forall D \subseteq [k] = \{1, 2, \dots, k\}$,

$$g(D) = f\Big(\bigcup_{d \in D} V_d\Big) \tag{92}$$

Grouping elements, set cover, and bipartite neighborhoods

- Given submodular $f : 2^V \to \mathbb{R}$ and a grouping of $V = V_1 \cup V_2 \cup \dots \cup V_k$ into k possibly overlapping clusters.
- Define new function $g : 2^{[k]} \to \mathbb{R}$ where $\forall D \subseteq [k] = \{1, 2, \dots, k\}$,

$$g(D) = f\Big(\bigcup_{d \in D} V_d\Big) \tag{92}$$

- Then g is submodular if either f is monotone non-decreasing or the sets $\{V_i\}$ are disjoint.

Grouping elements, set cover, and bipartite neighborhoods

- Given submodular $f : 2^V \to \mathbb{R}$ and a grouping of $V = V_1 \cup V_2 \cup \dots \cup V_k$ into k possibly overlapping clusters.
- Define new function $g : 2^{[k]} \to \mathbb{R}$ where $\forall D \subseteq [k] = \{1, 2, \dots, k\}$,

$$g(D) = f\Big(\bigcup_{d \in D} V_d\Big) \tag{92}$$

- Then g is submodular if either f is monotone non-decreasing or the sets $\{V_i\}$ are disjoint.
- Ex: Bipartite neighborhoods: Let $\Gamma : 2^V \to \mathbb{R}$ be the neighbor function in a bipartite graph $G = (V, U, E, w)$. V is set of "left" nodes, U is set of right nodes, $E \subseteq V \times U$ are edges, and $w : 2^E \to \mathbb{R}$ is a modular function on edges.

Grouping elements, set cover, and bipartite neighborhoods

- Given submodular $f : 2^V \to \mathbb{R}$ and a grouping of $V = V_1 \cup V_2 \cup \cdots \cup V_k$ into k possibly overlapping clusters.
- Define new function $g : 2^{[k]} \to \mathbb{R}$ where $\forall D \subseteq [k] = \{1, 2, \ldots, k\}$,

$$g(D) = f(\bigcup_{d \in D} V_d) \tag{92}$$

- Then g is submodular if either f is monotone non-decreasing or the sets $\{V_i\}$ are disjoint.
- Ex: Bipartite neighborhoods: Let $\Gamma : 2^V \to \mathbb{R}$ be the neighbor function in a bipartite graph $G = (V, U, E, w)$. V is set of "left" nodes, U is set of right nodes, $E \subseteq V \times U$ are edges, and $w : 2^E \to \mathbb{R}$ is a modular function on edges.
- Neighbors defined as $\Gamma(X) = \{u \in U : |X \times \{u\} \cap E| \geq 1\}$ for $X \subseteq V$.

Grouping elements, set cover, and bipartite neighborhoods

- Given submodular $f : 2^V \to \mathbb{R}$ and a grouping of $V = V_1 \cup V_2 \cup \dots \cup V_k$ into k possibly overlapping clusters.
- Define new function $g : 2^{[k]} \to \mathbb{R}$ where $\forall D \subseteq [k] = \{1, 2, \dots, k\}$,

$$g(D) = f(\bigcup_{d \in D} V_d) \tag{92}$$

- Then g is submodular if either f is monotone non-decreasing or the sets $\{V_i\}$ are disjoint.
- Ex: Bipartite neighborhoods: Let $\Gamma : 2^V \to \mathbb{R}$ be the neighbor function in a bipartite graph $G = (V, U, E, w)$. V is set of "left" nodes, U is set of right nodes, $E \subseteq V \times U$ are edges, and $w : 2^E \to \mathbb{R}$ is a modular function on edges.
- Neighbors defined as $\Gamma(X) = \{u \in U : |X \times \{u\} \cap E| \geq 1\}$ for $X \subseteq V$. Then $f(\Gamma(X))$ is submodular. Special case: set cover.

Grouping elements, set cover, and bipartite neighborhoods

- Given submodular $f : 2^V \to \mathbb{R}$ and a grouping of $V = V_1 \cup V_2 \cup \dots \cup V_k$ into k possibly overlapping clusters.
- Define new function $g : 2^{[k]} \to \mathbb{R}$ where $\forall D \subseteq [k] = \{1, 2, \dots, k\}$,

$$g(D) = f(\bigcup_{d \in D} V_d) \tag{92}$$

- Then g is submodular if either f is monotone non-decreasing or the sets $\{V_i\}$ are disjoint.
- Ex: Bipartite neighborhoods: Let $\Gamma : 2^V \to \mathbb{R}$ be the neighbor function in a bipartite graph $G = (V, U, E, w)$. V is set of "left" nodes, U is set of right nodes, $E \subseteq V \times U$ are edges, and $w : 2^E \to \mathbb{R}$ is a modular function on edges.
- Neighbors defined as $\Gamma(X) = \{u \in U : |X \times \{u\} \cap E| \geq 1\}$ for $X \subseteq V$. Then $f(\Gamma(X))$ is submodular. Special case: set cover.
- In fact, all integral polymatroid functions can be obtained in g above for f a matroid rank function and $\{V_d\}$ appropriately chosen.

Concave composed with polymatroid

We also have the following composition property with concave functions:

Theorem

Given functions $f : 2^V \to \mathbb{R}$ *and* $g : \mathbb{R} \to \mathbb{R}$*, the composition* $h = f \circ g : 2^V \to \mathbb{R}$ *(i.e.,* $h(S) = g(f(S))$*) is nondecreasing submodular, if* g *is non-decreasing concave and* f *is nondecreasing submodular.*

Concave composed with non-negative modular

Theorem

Given a ground set V. The following two are equivalent:

1. *For all modular functions $m : 2^V \to \mathbb{R}_+$, then $f : 2^V \to \mathbb{R}$ defined as $f(A) = g(m(A))$ is submodular*
2. *$g : \mathbb{R}_+ \to \mathbb{R}$ is concave.*

- If g is non-decreasing concave, then f is polymatroidal.

Concave composed with non-negative modular

Theorem

Given a ground set V. The following two are equivalent:

1. *For all modular functions $m : 2^V \to \mathbb{R}_+$, then $f : 2^V \to \mathbb{R}$ defined as $f(A) = g(m(A))$ is submodular*
2. *$g : \mathbb{R}_+ \to \mathbb{R}$ is concave.*

- If g is non-decreasing concave, then f is polymatroidal.
- Sums of concave over modular functions are submodular

$$f(A) = \sum_{i=1}^{K} g_i(m_i(A)) \tag{93}$$

Concave composed with non-negative modular

Theorem

Given a ground set V. The following two are equivalent:

1. *For all modular functions $m : 2^V \to \mathbb{R}_+$, then $f : 2^V \to \mathbb{R}$ defined as $f(A) = g(m(A))$ is submodular*
2. *$g : \mathbb{R}_+ \to \mathbb{R}$ is concave.*

- If g is non-decreasing concave, then f is polymatroidal.
- Sums of concave over modular functions are submodular

$$f(A) = \sum_{i=1}^{K} g_i(m_i(A)) \tag{93}$$

- Very large class of functions, including graph cut, bipartite neighborhoods, set cover (Stobbe & Krause).

Concave composed with non-negative modular

Theorem

Given a ground set V. The following two are equivalent:

1. *For all modular functions $m : 2^V \to \mathbb{R}_+$, then $f : 2^V \to \mathbb{R}$ defined as $f(A) = g(m(A))$ is submodular*
2. *$g : \mathbb{R}_+ \to \mathbb{R}$ is concave.*

- If g is non-decreasing concave, then f is polymatroidal.
- Sums of concave over modular functions are submodular

$$f(A) = \sum_{i=1}^{K} g_i(m_i(A)) \tag{93}$$

- Very large class of functions, including graph cut, bipartite neighborhoods, set cover (Stobbe & Krause).
- However, Vondrak showed that a graphic matroid rank function over K_4 can't be represented in this fashion.

Weighted Matroid Rank Functions

- We saw matroid rank is submodular. Given matroid $(V, \mathcal{I})$,

$$f(B) = \max\{|A| : A \subseteq B \text{ and } A \in \mathcal{I}\} \tag{94}$$

Weighted Matroid Rank Functions

- We saw matroid rank is submodular. Given matroid $(V, \mathcal{I})$,

$$f(B) = \max \{|A| : A \subseteq B \text{ and } A \in \mathcal{I}\} \tag{94}$$

- Weighted matroid rank functions. Given matroid $(V, \mathcal{I})$, and non-negative modular function $m : 2^V \to \mathbb{R}_+$,

$$f(B) = \max \{m(A) : A \subseteq B \text{ and } A \in \mathcal{I}\} \tag{95}$$

  is also submodular.

Weighted Matroid Rank Functions

- We saw matroid rank is submodular. Given matroid $(V, \mathcal{I})$,

$$f(B) = \max\{|A| : A \subseteq B \text{ and } A \in \mathcal{I}\} \tag{94}$$

- Weighted matroid rank functions. Given matroid $(V, \mathcal{I})$, and non-negative modular function $m : 2^V \to \mathbb{R}_+$,

$$f(B) = \max\{m(A) : A \subseteq B \text{ and } A \in \mathcal{I}\} \tag{95}$$

  is also submodular.
- Take a 1-partition matroid with limit k, we get:

$$f(B) = \max\{m(A) : A \subseteq B \text{ and } |A| \leq k\} \tag{96}$$

Weighted Matroid Rank Functions

- We saw matroid rank is submodular. Given matroid $(V, \mathcal{I})$,

$$f(B) = \max\{|A| : A \subseteq B \text{ and } A \in \mathcal{I}\} \tag{94}$$

- Weighted matroid rank functions. Given matroid $(V, \mathcal{I})$, and non-negative modular function $m : 2^V \to \mathbb{R}_+$,

$$f(B) = \max\{m(A) : A \subseteq B \text{ and } A \in \mathcal{I}\} \tag{95}$$

  is also submodular.
- Take a 1-partition matroid with limit k, we get:

$$f(B) = \max\{m(A) : A \subseteq B \text{ and } |A| \leq k\} \tag{96}$$

- Take a 1-partition matroid with limit 1, we get the max function:

$$f(B) = \max_{b \in B} m(b) \tag{97}$$

Facility Location

- Given a set of k matroids $(V, \mathcal{I}_i)$ and k modular weight functions m_i, the following is submodular:

$$f(A) = \sum_{i=1}^{k} \alpha_i \max \{ m_i(A) : A \subseteq B \text{ and } A \in \mathcal{I}_i \} \tag{98}$$

Facility Location

- Given a set of k matroids $(V, \mathcal{I}_i)$ and k modular weight functions m_i, the following is submodular:

$$f(A) = \sum_{i=1}^{k} \alpha_i \max\{m_i(A) : A \subseteq B \text{ and } A \in \mathcal{I}_i\} \tag{98}$$

- Take all $\alpha_i = 1$, all matroids 1-partition matroids, and set $w_{ij} = m_i(j)$, and $k = |V|$ for some weighted graph $G = (V, E, w)$, we get the uncapacitated facility location function:

$$f(A) = \sum_{i \in V} \max_{a \in A} w_{ai} \tag{99}$$

Information and Complexity functions

- Given a set V of items, we might wish to measure the "information" or "complexity" in a subset $A \subset V$.

Information and Complexity functions

- Given a set V of items, we might wish to measure the "information" or "complexity" in a subset $A \subset V$.
- Matroid rank $r(A)$ can measure the "information" or "complexity" via the dimensionality spanned by vectors with indices A.

Information and Complexity functions

- Given a set V of items, we might wish to measure the "information" or "complexity" in a subset $A \subset V$.
- Matroid rank $r(A)$ can measure the "information" or "complexity" via the dimensionality spanned by vectors with indices A.
- Unit increment $r(v|A) \in \{0, 1\}$ so no partial independence.

Information and Complexity functions

- Given a set V of items, we might wish to measure the "information" or "complexity" in a subset $A \subset V$.
- Matroid rank $r(A)$ can measure the "information" or "complexity" via the dimensionality spanned by vectors with indices A.
- Unit increment $r(v|A) \in \{0,1\}$ so no partial independence.
- Entropy of a set of random variables $\{X_v\}_{v \in V}$, where

$$f(A) = H(X_A) = H(\bigcup_{a \in A} X_a) = -\sum_{x_A} \Pr(x_A) \log \Pr(x_A) \tag{100}$$

can measure partial independence.

Information and Complexity functions

- Given a set V of items, we might wish to measure the "information" or "complexity" in a subset $A \subset V$.
- Matroid rank $r(A)$ can measure the "information" or "complexity" via the dimensionality spanned by vectors with indices A.
- Unit increment $r(v|A) \in \{0, 1\}$ so no partial independence.
- Entropy of a set of random variables $\{X_v\}_{v \in V}$, where

$$f(A) = H(X_A) = H(\bigcup_{a \in A} X_a) = -\sum_{x_A} \Pr(x_A) \log \Pr(x_A) \tag{100}$$

  can measure partial independence.
- Entropy is submodular due to non-negativity of conditional mutual information. Given $A, B, C \subseteq V$,

$$\begin{aligned} & I(X_{A \setminus B}; X_{B \setminus A} | X_{A \cap B}) \\ & = H(X_A) + H(X_B) - H(X_{A \cup B}) - H(X_{A \cap B}) \geq 0 \end{aligned} \tag{101}$$

Submodular Generalized Dependence

- there is a notion of "independence", i.e., $A \perp\!\!\!\perp B$:

$$f(A \cup B) = f(A) + f(B), \tag{37}$$

- and a notion of "conditional independence", i.e., $A \perp\!\!\!\perp B | C$:

$$f(A \cup B \cup C) + f(C) = f(A \cup C) + f(B \cup C) \tag{38}$$

- and a notion of "dependence" (conditioning reduces valuation):

$$f(A|B) \triangleq f(A \cup B) - f(B) < f(A), \tag{39}$$

- and a notion of "conditional mutual information"

$$I_f(A; B|C) \triangleq f(A \cup C) + f(B \cup C) - f(A \cup B \cup C) - f(C) \geq 0$$

- and two notions of "information amongst a collection of sets":

$$I_f(S_1; S_2; \dots; S_k) = \sum_{i=1}^{k} f(S_k) - f(S_1 \cup S_2 \cup \dots \cup S_k) \tag{40}$$

$$I'_f(S_1; S_2; \dots; S_k) = \sum_{A \subseteq \{1,2,\dots,k\}} (-1)^{|A|+1} f(\bigcup_{j \in A} S_j) \tag{41}$$

Gaussian entropy, and the log-determinant function

Definition (differential entropy $h(X)$)

$$h(X) = -\int_S f(x) \log f(x) dx \tag{102}$$

- When $x \sim \mathcal{N}(\mu, \Sigma)$ is multivariate Gaussian, the (differential) entropy of the r.v. X is given by

$$h(X) = \log \sqrt{|2\pi e \mathbf{\Sigma}|} = \log \sqrt{(2\pi e)^n |\mathbf{\Sigma}|} \tag{103}$$

and in particular, for a variable subset A and a constant γ,

$$f(A) = h(X_A) = \log \sqrt{(2\pi e)^{|A|} |\mathbf{\Sigma}_A|} = \gamma|A| + \frac{1}{2} \log |\mathbf{\Sigma}_A| \tag{104}$$

Gaussian entropy, and the log-determinant function

Definition (differential entropy $h(X)$)

$$h(X) = -\int_S f(x) \log f(x) dx \tag{102}$$

- When $x \sim \mathcal{N}(\mu, \Sigma)$ is multivariate Gaussian, the (differential) entropy of the r.v. X is given by

$$h(X) = \log \sqrt{|2\pi e \mathbf{\Sigma}|} = \log \sqrt{(2\pi e)^n |\mathbf{\Sigma}|} \tag{103}$$

and in particular, for a variable subset A and a constant γ,

$$f(A) = h(X_A) = \log \sqrt{(2\pi e)^{|A|} |\mathbf{\Sigma}_A|} = \gamma |A| + \frac{1}{2} \log |\mathbf{\Sigma}_A| \tag{104}$$

- Application of Jensen's inequality shows that $I(X_{A \setminus B}; X_{B \setminus A} | X_{A \cap B}) = h(X_A) + h(X_B) - h(X_{A \cup B}) - h(X_{A \cap B}) \geq 0$. Hence differential entropy is submodular, and thus so is the logdet function.

Are all polymatroid functions entropy functions?

Are all polymatroid functions entropy functions?

No, entropy functions must also satisfy the following:

Theorem (Yeung)

For any four discrete random variables $\{X, Y, Z, U\}$*, then*

$$I(X;Y) = I(X;Y|Z) = 0 \tag{105}$$

implies that

$$I(X;Y|Z,U) \leq I(Z;U|X,Y) + I(X;Y|U) \tag{106}$$

where $I(\cdot;\cdot|\cdot)$ *is the standard Shannon mutual information function.*

- This is not required for all polymatroid-based conditional mutual information functions $I_f(\cdot;\cdot|\cdot)$.

Containment, Gaussian Entropy, and DPPs

- Submodular functions $\supset$ Polymatroid functions $\supset$ Entropy functions $\supset$ Gaussian Entropy functions $=$ DPPs.

Containment, Gaussian Entropy, and DPPs

- Submodular functions $\supset$ Polymatroid functions $\supset$ Entropy functions $\supset$ Gaussian Entropy functions $=$ DPPs.
- DPPs (Kulesza & Taskar) are a point process where $\Pr(\mathbf{Y} = Y) \propto \det(L_Y)$ for some positive-definite matrix L, so DPPs are log-submodular, as we saw.

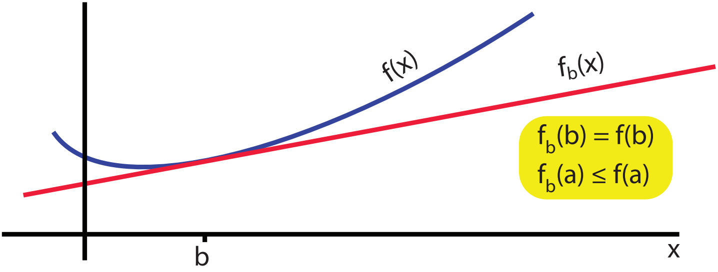
Containment, Gaussian Entropy, and DPPs

- Submodular functions $\supset$ Polymatroid functions $\supset$ Entropy functions $\supset$ Gaussian Entropy functions $=$ DPPs.
- DPPs (Kulesza & Taskar) are a point process where $\Pr(\mathbf{Y} = Y) \propto \det(L_Y)$ for some positive-definite matrix L, so DPPs are log-submodular, as we saw.
- Thanks to the properties of matrix algebra (e.g., determinants), DPPs are computationally extremely attractive and are now widely used in ML.

Outline: Part 3

6. Discrete Semimodular Semigradients
7. Continuous Extensions
   - Lovász Extension
   - Concave Extension
8. Like Concave or Convex?
9. Optimization
10. Reading

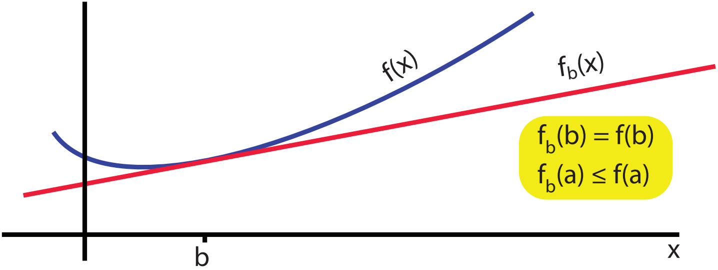
Convex Functions and Tight Subgradients



- A convex function f has a subgradient at any in-domain point b, namely there exists f_b such that

$$f(x) - f(b) \geq \langle f_b, x - b \rangle, \forall x. \tag{107}$$

Concave Functions and Tight Supergradients

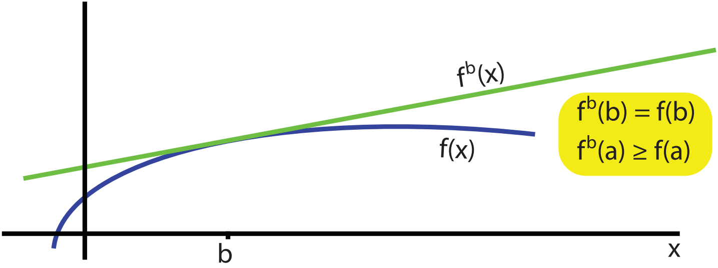


- A convex function f has a subgradient at any in-domain point b, namely there exists f_b such that

$$f(x) - f(b) \geq \langle f_b, x - b \rangle, \forall x. \tag{107}$$

- We have that $f(x)$ is convex, $f_b(x)$ is affine, and is a tight subgradient (tight at b, affine lower bound on $f(x)$).

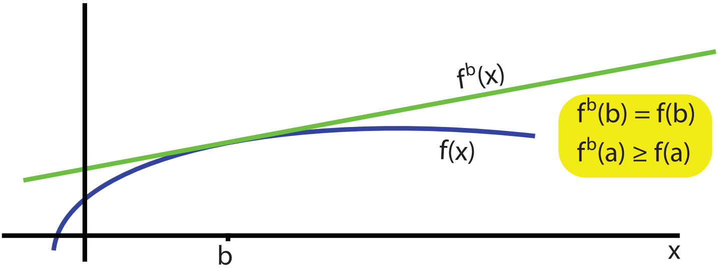
Convex Functions and Tight Subgradients



- A concave f has a supergradient at any in-domain point b, namely there exists f^b such that

$$f(x) - f(b) \leq \langle f^b, x - b \rangle, \forall x. \tag{108}$$

Concave Functions and Tight Supergradients



- A concave f has a supergradient at any in-domain point b, namely there exists f^b such that

$$f(x) - f(b) \leq \langle f^b, x - b \rangle, \forall x. \tag{108}$$

- We have that $f(x)$ is concave, $f^b(x)$ is affine, and is a tight supergradient (tight at b, affine upper bound on $f(x)$).

Trivial additive upper/lower bounds

- Any submodular function has trivial additive upper and lower bounds. That is for all $A \subseteq V$,

$$m_f(A) \leq f(A) \leq m^f(A) \tag{109}$$

where

$$m^f(A) = \sum_{a \in A} f(a) \tag{110}$$

$$m_f(A) = \sum_{a \in A} f(a|V \setminus \{a\}) \tag{111}$$

Trivial additive upper/lower bounds

- Any submodular function has trivial additive upper and lower bounds. That is for all $A \subseteq V$,

$$m_f(A) \leq f(A) \leq m^f(A) \tag{109}$$

where

$$m^f(A) = \sum_{a \in A} f(a) \tag{110}$$

$$m_f(A) = \sum_{a \in A} f(a|V \setminus \{a\}) \tag{111}$$

- $m^f \in \mathbb{R}^V$ and $m_f \in \mathbb{R}^V$ are both modular (or additive) functions.

Trivial additive upper/lower bounds

- Any submodular function has trivial additive upper and lower bounds. That is for all $A \subseteq V$,

$$m_f(A) \leq f(A) \leq m^f(A) \tag{109}$$

where

$$m^f(A) = \sum_{a \in A} f(a) \tag{110}$$

$$m_f(A) = \sum_{a \in A} f(a|V \setminus \{a\}) \tag{111}$$

- $m^f \in \mathbb{R}^V$ and $m_f \in \mathbb{R}^V$ are both modular (or additive) functions.
- A "semigradient" is customized, and at least at one point is tight.

Submodular Subgradients

- For submodular function f, the subdifferential (all subgradients tight at $X \subseteq V$) can be defined as:

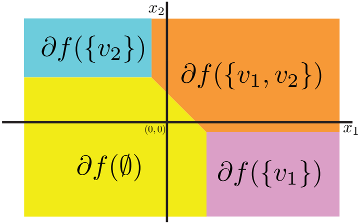
$$\partial f(X) = \{x \in \mathbb{R}^V : \forall Y \subseteq V, x(Y) - x(X) \leq f(Y) - f(X)\} \quad (112)$$

Submodular Subgradients

- For submodular function f, the subdifferential (all subgradients tight at $X \subseteq V$) can be defined as:

$$\partial f(X) = \{x \in \mathbb{R}^V : \forall Y \subseteq V, x(Y) - x(X) \leq f(Y) - f(X)\} \quad (112)$$

- This partitions $\mathbb{R}^V$:

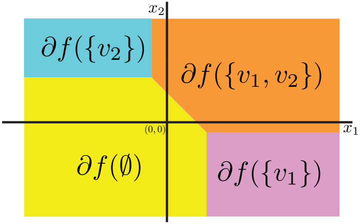


Submodular Subgradients

- For submodular function f, the subdifferential (all subgradients tight at $X \subseteq V$) can be defined as:

$$\partial f(X) = \{x \in \mathbb{R}^V : \forall Y \subseteq V, x(Y) - x(X) \leq f(Y) - f(X)\} \quad (112)$$

- This partitions $\mathbb{R}^V$:



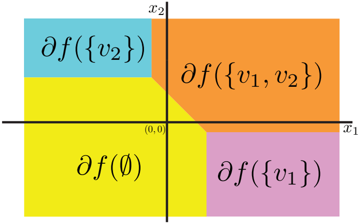
- Extreme points are easy to get via Edmonds's greedy algorithm:

Submodular Subgradients

- For submodular function f, the subdifferential (all subgradients tight at $X \subseteq V$) can be defined as:

$$\partial f(X) = \{x \in \mathbb{R}^V : \forall Y \subseteq V, x(Y) - x(X) \leq f(Y) - f(X)\} \quad (112)$$

- This partitions $\mathbb{R}^V$:



- Extreme points are easy to get via Edmonds's greedy algorithm:

Theorem (Fujishige 2005, Theorem 6.11)

A point $y \in \mathbb{R}^V$ is an extreme point of $\partial f(X)$, iff there exists a maximal chain $\emptyset = S_0 \subset S_1 \subset \cdots \subset S_n$ with $X = S_j$ for some j, such that $y(S_i \setminus S_{i-1}) = y(S_i) - y(S_{i-1}) = f(S_i) - f(S_{i-1})$.

The Submodular Subgradients (Fujishige 2005)

- For an arbitrary $Y \subseteq V$
- Let σ be a permutation of V and define $S_i^\sigma = \{\sigma(1), \sigma(2), \ldots, \sigma(i)\}$ as σ's chain where $S_k^\sigma = Y$ where $|Y| = k$.
- We can define a subgradient h_Y^f corresponding to f as:

$$h^f_{Y,\sigma}(\sigma(i)) = \begin{cases} f(S_1^\sigma) & \text{if } i = 1 \\ f(S_i^\sigma) - f(S_{i-1}^\sigma) & \text{otherwise} \end{cases}.$$

- We get a tight modular lower bound of f as follows:

$$h^f_{Y,\sigma}(X) \triangleq \sum_{x \in X} h^f_{Y,\sigma}(x) \leq f(X), \forall X \subseteq V.$$

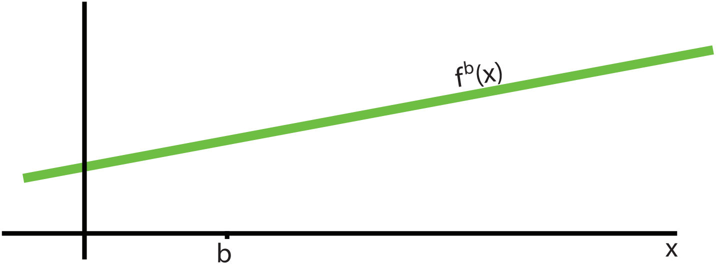
Note, tight at Y means $h^f_{Y,\sigma}(Y) = f(Y)$.

Convexity and Tight Sub- and Super-gradients?

- Can there be both a tight linear upper bound and tight linear lower bound on a convex (or concave) function, where each bound is tight at the same point?

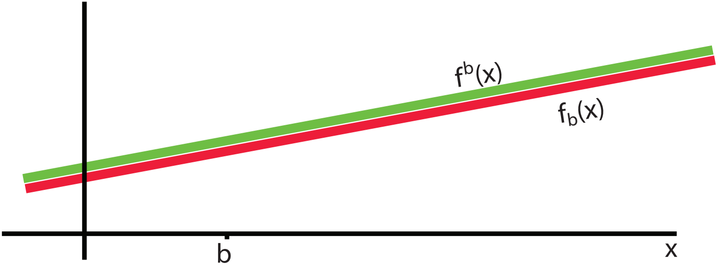
Convexity and Tight Sub- and Super-gradients?

- Can there be both a tight linear upper bound and tight linear lower bound on a convex (or concave) function, where each bound is tight at the same point?



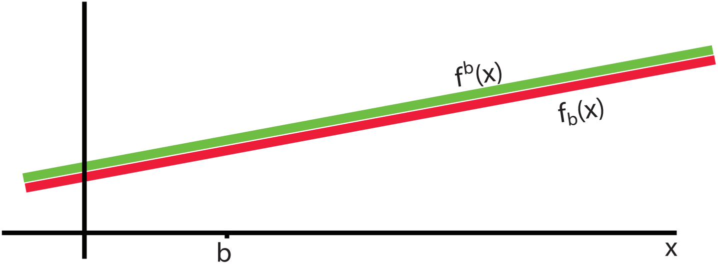
Convexity and Tight Sub- and Super-gradients?

- Can there be both a tight linear upper bound and tight linear lower bound on a convex (or concave) function, where each bound is tight at the same point?



Convexity and Tight Sub- and Super-gradients?

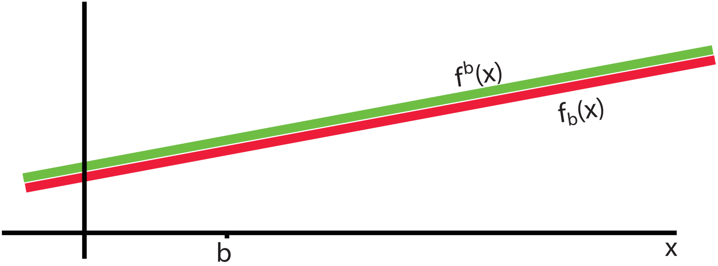
- Can there be both a tight linear upper bound and tight linear lower bound on a convex (or concave) function, where each bound is tight at the same point?



- If a continuous function has both a sub- and super-gradient at a point, then the function must be affine.

Convexity and Tight Sub- and Super-gradients?

- Can there be both a tight linear upper bound and tight linear lower bound on a convex (or concave) function, where each bound is tight at the same point?



- If a continuous function has both a sub- and super-gradient at a point, then the function must be affine.
- What about discrete set functions?

The Submodular Supergradients

- Can a submodular function also have a supergradient? We saw that in the continuous case, simultaneous sub/super gradients meant linear.
- (Nemhauser, Wolsey, & Fisher 1978) established the following iff conditions for submodularity (if either hold, f is submodular):

$$f(Y) \leq f(X) - \sum_{j \in X \setminus Y} f(j|X \setminus j) + \sum_{j \in Y \setminus X} f(j|X \cap Y),$$

$$f(Y) \leq f(X) - \sum_{j \in X \setminus Y} f(j|(X \cup Y) \setminus j) + \sum_{j \in Y \setminus X} f(j|X)$$

Recall that $f(A|B) \triangleq f(A \cup B) - f(B)$ is the gain of adding A in the context of B.

Submodular and Supergradients

- Using submodularity further, these can be relaxed to produce two tight modular upper bounds (Jegelka & Bilmes, 2011, Iyer & Bilmes 2013):

$$f(Y) \leq m^f_{X,1}(Y) \triangleq f(X) - \sum_{j \in X \setminus Y} f(j|X \setminus j) + \sum_{j \in Y \setminus X} f(j|\emptyset),$$

$$f(Y) \leq m^f_{X,2}(Y) \triangleq f(X) - \sum_{j \in X \setminus Y} f(j|V \setminus j) + \sum_{j \in Y \setminus X} f(j|X).$$

Hence, this yields three tight (at set X) modular upper bounds $m^f_{X,1}, m^f_{X,2}$ for any submodular function f.

Optimizing difference of submodular functions

Theorem

Given an arbitrary set function f, it can be expressed as a difference $f = g - h$ between two polymatroid functions, where both g and h are polymatroidal.

- The semi-gradients above offer a majorization/maximization framework to minimize any function that is naturally expressed as such a difference.
- E.g., to minimize $f = g - h$, starting with a candidate solution X, repeatedly choose a modular supergradient for g and modular subgradient for h, and perform modular minimization (easy). (see Iyer & Bilmes, 2012).
- Similar strategy used for other combinatorial constraints (.e., cooperative cut, submodular on edges, see Jegelka & Bilmes 2011)
- Opens the doors to first-order methods for discrete optimization.

Outline: Part 3

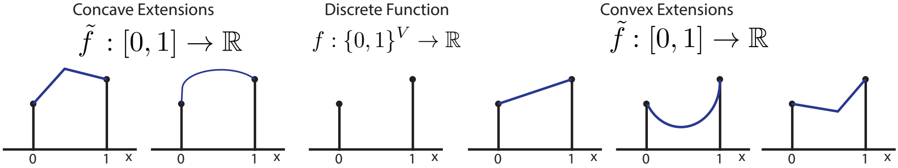
6. Discrete Semimodular Semigradients
7. Continuous Extensions
   - Lovász Extension
   - Concave Extension
8. Like Concave or Convex?
9. Optimization
10. Reading

Continuous Extensions of Discrete Set Functions

- Any function $f : 2^V \to \mathbb{R}$ (equivalently $f : \{0,1\}^V \to \mathbb{R}$) can be extended to a continuous function $\tilde{f} : [0,1]^V \to \mathbb{R}$.

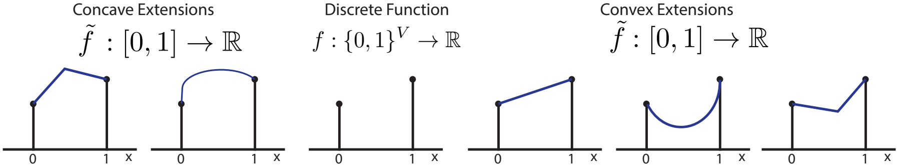
Continuous Extensions of Discrete Set Functions

- Any function $f : 2^V \to \mathbb{R}$ (equivalently $f : \{0,1\}^V \to \mathbb{R}$) can be extended to a continuous function $\tilde{f} : [0,1]^V \to \mathbb{R}$.
- In fact, any such discrete function defined on the vertices of the n-D hypercube $\{0,1\}^n$ has a variety of both convex and concave extensions tight at the vertices (Crama & Hammer). Example $n = 1$,



Continuous Extensions of Discrete Set Functions

- Any function $f : 2^V \to \mathbb{R}$ (equivalently $f : \{0,1\}^V \to \mathbb{R}$) can be extended to a continuous function $\tilde{f} : [0,1]^V \to \mathbb{R}$.
- In fact, any such discrete function defined on the vertices of the n-D hypercube $\{0,1\}^n$ has a variety of both convex and concave extensions tight at the vertices (Crama & Hammer). Example $n = 1$,

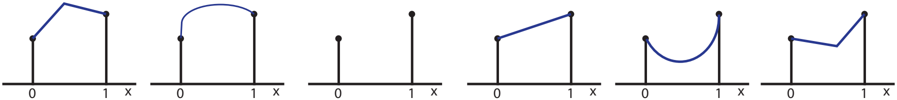


- Since there are an exponential number of vertices $\{0,1\}^n$, important questions regarding such extensions is:

Continuous Extensions of Discrete Set Functions

- Any function $f : 2^V \to \mathbb{R}$ (equivalently $f : \{0,1\}^V \to \mathbb{R}$) can be extended to a continuous function $\tilde{f} : [0,1]^V \to \mathbb{R}$.
- In fact, any such discrete function defined on the vertices of the n-D hypercube $\{0,1\}^n$ has a variety of both convex and concave extensions tight at the vertices (Crama & Hammer). Example $n = 1$,

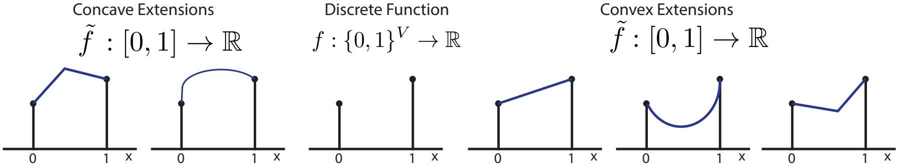
Concave Extensions $\tilde{f} : [0,1] \to \mathbb{R}$ | Discrete Function $f : \{0,1\}^V \to \mathbb{R}$ | Convex Extensions $\tilde{f} : [0,1] \to \mathbb{R}$



- Since there are an exponential number of vertices $\{0,1\}^n$, important questions regarding such extensions is:
  1. When are they computationally feasible to obtain or estimate?

Continuous Extensions of Discrete Set Functions

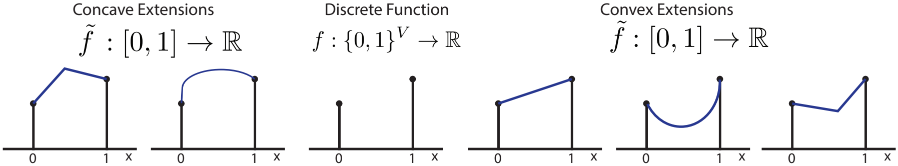
- Any function $f : 2^V \to \mathbb{R}$ (equivalently $f : \{0,1\}^V \to \mathbb{R}$) can be extended to a continuous function $\tilde{f} : [0,1]^V \to \mathbb{R}$.
- In fact, any such discrete function defined on the vertices of the n-D hypercube $\{0,1\}^n$ has a variety of both convex and concave extensions tight at the vertices (Crama & Hammer). Example $n = 1$,



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

Continuous Extensions of Discrete Set Functions

- Any function $f : 2^V \to \mathbb{R}$ (equivalently $f : \{0,1\}^V \to \mathbb{R}$) can be extended to a continuous function $\tilde{f} : [0,1]^V \to \mathbb{R}$.
- In fact, any such discrete function defined on the vertices of the n-D hypercube $\{0,1\}^n$ has a variety of both convex and concave extensions tight at the vertices (Crama & Hammer). Example $n = 1$,



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

A continuous extension of f

- Given a submodular function f, a $w \in \mathbb{R}^V$, define chain $V_i = \{v_1, v_2, \dots, v_i\}$ based on w sorted in decreasing order. Then Edmonds's greedy algorithm gives us:

$$\tilde{f}(w)$$

A continuous extension of f

- Given a submodular function f, a $w \in \mathbb{R}^V$, define chain $V_i = \{v_1, v_2, \dots, v_i\}$ based on w sorted in decreasing order. Then Edmonds's greedy algorithm gives us:

$$\tilde{f}(w) = \max(wx : x \in P_f) \tag{113}$$

A continuous extension of f

- Given a submodular function f, a $w \in \mathbb{R}^V$, define chain $V_i = \{v_1, v_2, \dots, v_i\}$ based on w sorted in decreasing order. Then Edmonds's greedy algorithm gives us:

$$\tilde{f}(w) = \max(wx : x \in P_f) \tag{113}$$

$$= \sum_{i=1}^{m} w(v_i) f(v_i | V_{i-1}) \tag{114}$$

A continuous extension of f

- Given a submodular function f, a $w \in \mathbb{R}^V$, define chain $V_i = \{v_1, v_2, \dots, v_i\}$ based on w sorted in decreasing order. Then Edmonds's greedy algorithm gives us:

$$\tilde{f}(w) = \max(wx : x \in P_f) \tag{113}$$

$$= \sum_{i=1}^{m} w(v_i) f(v_i | V_{i-1}) \tag{114}$$

$$= \sum_{i=1}^{m} w(v_i)(f(V_i) - f(V_{i-1})) \tag{115}$$

A continuous extension of f

- Given a submodular function f, a $w \in \mathbb{R}^V$, define chain $V_i = \{v_1, v_2, \dots, v_i\}$ based on w sorted in decreasing order. Then Edmonds's greedy algorithm gives us:

$$\tilde{f}(w) = \max(wx : x \in P_f) \tag{113}$$

$$= \sum_{i=1}^{m} w(v_i) f(v_i | V_{i-1}) \tag{114}$$

$$= \sum_{i=1}^{m} w(v_i)(f(V_i) - f(V_{i-1})) \tag{115}$$

$$= w(v_m) f(V_m) + \sum_{i=1}^{m-1} (w(v_i) - w(v_{i+1})) f(V_i) \tag{116}$$

A continuous extension of f

- Definition of the continuous extension, once again:

$$\tilde{f}(w) = \max(wx : x \in P_f) \tag{117}$$

A continuous extension of f

- Definition of the continuous extension, once again:

$$\tilde{f}(w) = \max(wx : x \in P_f) \tag{117}$$

- Therefore, if f is a submodular function, we can write

$$\tilde{f}(w)$$

A continuous extension of f

- Definition of the continuous extension, once again:

$$\tilde{f}(w) = \max(wx : x \in P_f) \tag{117}$$

- Therefore, if f is a submodular function, we can write

$$\tilde{f}(w) = w(v_m)f(V_m) + \sum_{i=1}^{m-1}(w(v_i) - w(v_{i+1}))f(V_i) \tag{118}$$

A continuous extension of f

- Definition of the continuous extension, once again:

$$\tilde{f}(w) = \max(wx : x \in P_f) \tag{117}$$

- Therefore, if f is a submodular function, we can write

$$\tilde{f}(w) = w(v_m)f(V_m) + \sum_{i=1}^{m-1}(w(v_i) - w(v_{i+1}))f(V_i) \tag{118}$$

$$= \sum_{i=1}^{m} \lambda_i f(V_i) \tag{119}$$

A continuous extension of f

- Definition of the continuous extension, once again:

$$\tilde{f}(w) = \max(wx : x \in P_f) \tag{117}$$

- Therefore, if f is a submodular function, we can write

$$\tilde{f}(w) = w(v_m)f(V_m) + \sum_{i=1}^{m-1}(w(v_i) - w(v_{i+1}))f(V_i) \tag{118}$$

$$= \sum_{i=1}^{m} \lambda_i f(V_i) \tag{119}$$

where $\lambda_m = w(v_m)$ and otherwise $\lambda_i = w(v_i) - w(v_{i+1})$, where the elements are sorted according to w as before.

A continuous extension of f

- Definition of the continuous extension, once again:

$$\tilde{f}(w) = \max(wx : x \in P_f) \tag{117}$$

- Therefore, if f is a submodular function, we can write

$$\tilde{f}(w) = w(v_m)f(V_m) + \sum_{i=1}^{m-1}(w(v_i) - w(v_{i+1}))f(V_i) \tag{118}$$

$$= \sum_{i=1}^{m} \lambda_i f(V_i) \tag{119}$$

  where $\lambda_m = w(v_m)$ and otherwise $\lambda_i = w(v_i) - w(v_{i+1})$, where the elements are sorted according to w as before.
- From convex analysis, we know $\tilde{f}(w) = \max(wx : x \in P)$ is always convex in w for any set $P \subseteq R^V$, since it is the maximum of a set of linear functions (true even when f is not submodular or P is not a convex set).

An extension of f

- But, for any $f : 2^V \to \mathbb{R}$, even non-submodular f, we can define an extension in this way, with

$$\tilde{f}(w) = \sum_{i=1}^{m} \lambda_i f(V_i) \tag{120}$$

with the $V_i = \{v_1, \dots, v_i\}$'s defined based on sorted descending order of w as in $w(v_1) \geq w(v_2) \geq \dots \geq w(v_m)$, and where

$$\text{for } i \in \{1, \dots, m\}, \quad \lambda_i = \begin{cases} w(v_i) - w(v_{i+1}) & \text{if } i < m \\ w(v_m) & \text{if } i = m \end{cases} \tag{121}$$

so that $w = \sum_{i=1}^{m} \lambda_i \mathbf{1}_{V_i}$

An extension of f

- But, for any $f : 2^V \to \mathbb{R}$, even non-submodular f, we can define an extension in this way, with

$$\tilde{f}(w) = \sum_{i=1}^{m} \lambda_i f(V_i) \tag{120}$$

with the $V_i = \{v_1, \dots, v_i\}$'s defined based on sorted descending order of w as in $w(v_1) \geq w(v_2) \geq \dots \geq w(v_m)$, and where

$$\text{for } i \in \{1, \dots, m\}, \quad \lambda_i = \begin{cases} w(v_i) - w(v_{i+1}) & \text{if } i < m \\ w(v_m) & \text{if } i = m \end{cases} \tag{121}$$

so that $w = \sum_{i=1}^{m} \lambda_i \mathbf{1}_{V_i}$

- Note that $w = \sum_{i=1}^{m} \lambda_i \mathbf{1}_{V_i}$ is an interpolation of certain vertices of the hypercube, and that $\tilde{f}(w) = \sum_{i=1}^{m} \lambda_i f(V_i)$ is the corresponding interpolation of the values of f at sets corresponding to each hypercube vertex.

Lovász Extension, Submodularity and Convexity

Lovász proved the following important theorem.

Theorem

A function $f : 2^V \to \mathbb{R}$ is submodular iff its its continuous extension defined above as $\tilde{f}(w) = \sum_{i=1}^m \lambda_i f(V_i)$ with $w = \sum_{i=1}^m \lambda_i \mathbf{1}_{V_i}$ is a convex function in $\mathbb{R}^V$.

Minimizing $\tilde{f}$ vs. minimizing f

Theorem

Let f be submodular and $\tilde{f}$ be its Lovász extension. Then $\min \{f(A) | A \subseteq V\} = \min_{w \in \{0,1\}^V} \tilde{f}(w) = \min_{w \in [0,1]^V} \tilde{f}(w)$.

- Let $w^* \in \operatorname{argmin} \left\{ \tilde{f}(w) | w \in [0,1]^V \right\}$ and let $A^* \in \operatorname{argmin} \{f(A) | A \subseteq V\}$.

Minimizing $\tilde{f}$ vs. minimizing f

Theorem

Let f be submodular and $\tilde{f}$ be its Lovász extension. Then $\min \{f(A) | A \subseteq V\} = \min_{w \in \{0,1\}^V} \tilde{f}(w) = \min_{w \in [0,1]^V} \tilde{f}(w)$.

- Let $w^* \in \operatorname{argmin} \left\{ \tilde{f}(w) | w \in [0,1]^V \right\}$ and let $A^* \in \operatorname{argmin} \{f(A) | A \subseteq V\}$.
- Define chain $\{V_i^*\}$ based on descending sort of w^*. Then by greedy evaluation of L.E. we have

$$\tilde{f}(w^*) = \sum_i \lambda_i^* f(V_i^*) = f(A^*) = \min \{f(A) | A \subseteq V\} \tag{122}$$

Minimizing $\tilde{f}$ vs. minimizing f

Theorem

Let f be submodular and $\tilde{f}$ be its Lovász extension. Then $\min\{f(A)|A \subseteq V\} = \min_{w\in\{0,1\}^V} \tilde{f}(w) = \min_{w\in[0,1]^V} \tilde{f}(w)$.

- Let $w^* \in \operatorname{argmin}\left\{\tilde{f}(w)|w \in [0,1]^V\right\}$ and let $A^* \in \operatorname{argmin}\{f(A)|A \subseteq V\}$.
- Define chain $\{V_i^*\}$ based on descending sort of w^*. Then by greedy evaluation of L.E. we have

$$\tilde{f}(w^*) = \sum_i \lambda_i^* f(V_i^*) = f(A^*) = \min\{f(A)|A \subseteq V\} \tag{122}$$

- Then we can show that, for each i s.t. $\lambda_i > 0$,

$$f(V_i^*) = f(A^*) \tag{123}$$

So such $\{V_i^*\}$ are also minimizers.

Duality: convex minimization of L.E. and min-norm alg.

- Let f be a submodular function with $\tilde{f}$ it's Lovász extension. Then the following two problems are duals:

$$\underset{w \in \mathbb{R}^V}{\text{minimize}} \ \tilde{f}(w) + \frac{1}{2}\|w\|_2^2 \tag{224}$$

$$\begin{aligned} &\text{maximize} && -\|x\|_2^2 && (225a) \\ &\text{subject to} && x \in B_f && (225b) \end{aligned}$$

where $B_f = P_f \cap \left\{x \in \mathbb{R}^V : x(V) = f(V)\right\}$ is the base polytope of submodular function f, and $\|x\|_2^2 = \sum_{e \in V} x(e)^2$ is the squared 2-norm.

- Minimum-norm point algorithm (Fujishige-1991, Fujishige-2005, Fujishige-2011, Bach-2013) is essentially an active-set procedure for quadratic programming, and uses Edmonds's greedy algorithm to make it efficient.
- Unknown worst-case running time, although in practice it usually performs quite well.

Other applications of Lovász Extension

- "fast" submodular function minimization, as mentioned above.
- Structured sparse-encouraging convex norms (Bach-2011), semi-supervised learning, image denoising (as mentioned yesterday).
- Non-linear measures (Denneberg), non-linear aggregation functions (Grabisch et. al), and fuzzy set theory.
- Note, many of the critical properties of the Lovász extension were given by Jack Edmonds in the 1960s. Choquet proposed an identical integral in 1954, and G. Vitali proposed a similar integral in 1925! G.Vitali, Sulla definizione di integrale delle funzioni di una variabile, Annali di Matematica Serie IV, Tomo I,(1925), 111-121

Submodular Concave Extension

- Finding a concave extension (the concave envelope, smallest concave upper bound) of a submodular function is NP-hard (Vondrak).

Submodular Concave Extension

- Finding a concave extension (the concave envelope, smallest concave upper bound) of a submodular function is NP-hard (Vondrak).
- However, a useful surrogate is the multi-linear extension.

Submodular Concave Extension

- Finding a concave extension (the concave envelope, smallest concave upper bound) of a submodular function is NP-hard (Vondrak).
- However, a useful surrogate is the multi-linear extension.

Definition

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

$$F(x) = \sum_{S \subseteq V} f(S) \prod_{i \in S} x_i \prod_{j \in V \setminus S} (1 - x_j) \qquad (226)$$

Submodular Concave Extension

- Finding a concave extension (the concave envelope, smallest concave upper bound) of a submodular function is NP-hard (Vondrak).
- However, a useful surrogate is the multi-linear extension.

Definition

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

$$F(x) = \sum_{S \subseteq V} f(S) \prod_{i \in S} x_i \prod_{j \in V \setminus S} (1 - x_j) \tag{226}$$

- Not concave, but still provides useful approximations for many constrained maximization algorithms (e.g., multiple matroid and/or knapsack constraints) via the continuous greedy algorithm followed by rounding.

Submodular Concave Extension

- Finding a concave extension (the concave envelope, smallest concave upper bound) of a submodular function is NP-hard (Vondrak).
- However, a useful surrogate is the multi-linear extension.

Definition

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

$$F(x) = \sum_{S \subseteq V} f(S) \prod_{i \in S} x_i \prod_{j \in V \setminus S} (1 - x_j) \tag{226}$$

- Not concave, but still provides useful approximations for many constrained maximization algorithms (e.g., multiple matroid and/or knapsack constraints) via the continuous greedy algorithm followed by rounding.
- Often has to be approximated.

Outline: Part 3

6 Discrete Semimodular Semigradients

7 Continuous Extensions
- Lovász Extension
- Concave Extension

8 Like Concave or Convex?

9 Optimization

10 Reading

Submodular: Concave? Convex? Neither? Both?

- Are submodular functions more like convex or more like concave functions?

Submodular is like Concave

- **Convex 1:** Like convex functions, submodular functions can be minimized efficiently (polynomial time).

Submodular is like Concave

- **Convex 1:** Like convex functions, submodular functions can be minimized efficiently (polynomial time).
- **Convex 2:** The Lovász extension of a discrete set function is convex iff the set function is submodular.

Submodular is like Concave

- **Convex 3:** Frank's discrete separation theorem: Let $f : 2^V \to \mathbb{R}$ be a submodular function and $g : 2^V \to \mathbb{R}$ be a supermodular function such that for all $A \subseteq V$,

$$g(A) \leq f(A) \tag{227}$$

Then there exists modular function $x \in \mathbb{R}^V$ such that for all $A \subseteq V$:

$$g(A) \leq x(A) \leq f(A) \tag{228}$$

Submodular is like Concave

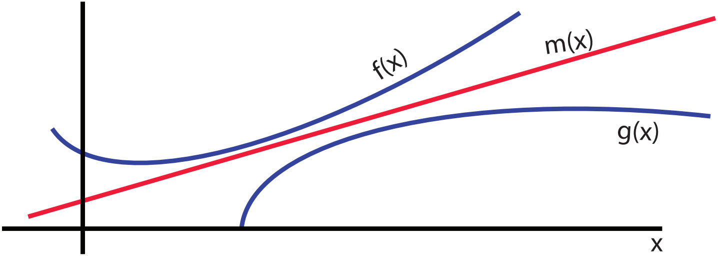
- **Convex 3:** Frank's discrete separation theorem: Let $f : 2^V \to \mathbb{R}$ be a submodular function and $g : 2^V \to \mathbb{R}$ be a supermodular function such that for all $A \subseteq V$,

$$g(A) \leq f(A) \tag{227}$$

Then there exists modular function $x \in \mathbb{R}^V$ such that for all $A \subseteq V$:

$$g(A) \leq x(A) \leq f(A) \tag{228}$$

- Compare to convex/concave case.



Submodular is like Concave

- **Convex 4:** Set of minimizers of a convex function is a convex set. Set of minimizers of a submodular function is a lattice. I.e., if $A, B \in \operatorname{argmin}_{A \subseteq V} f(A)$ then $A \cup B \in \operatorname{argmin}_{A \subseteq V} f(A)$ and $A \cap B \in \operatorname{argmin}_{A \subseteq V} f(A)$

Submodular is like Concave

- **Convex 4:** Set of minimizers of a convex function is a convex set. Set of minimizers of a submodular function is a lattice. I.e., if $A, B \in \operatorname{argmin}_{A \subseteq V} f(A)$ then $A \cup B \in \operatorname{argmin}_{A \subseteq V} f(A)$ and $A \cap B \in \operatorname{argmin}_{A \subseteq V} f(A)$
- **Convex 5:** Submodular functions have subdifferentials and subgradients tight at any point.

Submodularity and Concave

- **Concave 1:** A function is submodular if for all $X \subseteq V$ and $j, k \in V$

$$f(X + j) + f(X + k) \geq f(X + j + k) + f(X) \tag{229}$$

Submodularity and Concave

- **Concave 1:** A function is submodular if for all $X \subseteq V$ and $j, k \in V$

$$f(X + j) + f(X + k) \geq f(X + j + k) + f(X) \tag{229}$$

- With the gain defined as $\nabla_j(X) = f(X + j) - f(X)$, seen as a form of discrete gradient, this trivially becomes a second-order condition, akin to concave functions: A function is submodular if for all $X \subseteq V$ and $j, k \in V$, we have:

$$\nabla_j \nabla_k f(X) \leq 0 \tag{230}$$

Submodularity and Concave

- **Concave 1:** A function is submodular if for all $X \subseteq V$ and $j, k \in V$
$$f(X+j) + f(X+k) \geq f(X+j+k) + f(X) \tag{229}$$
- With the gain defined as $\nabla_j(X) = f(X+j) - f(X)$, seen as a form of discrete gradient, this trivially becomes a second-order condition, akin to concave functions: A function is submodular if for all $X \subseteq V$ and $j, k \in V$, we have:
$$\nabla_j \nabla_k f(X) \leq 0 \tag{230}$$
- **Concave 2:** Recall, Theorem 16: composition $h = f \circ g : 2^V \to \mathbb{R}$ (i.e., $h(S) = g(f(S))$) is nondecreasing submodular, if g is non-decreasing concave and f is nondecreasing submodular.

Submodularity and Concave

- **Concave 1:** A function is submodular if for all $X \subseteq V$ and $j, k \in V$
$$f(X + j) + f(X + k) \geq f(X + j + k) + f(X) \qquad (229)$$
- With the gain defined as $\nabla_j(X) = f(X + j) - f(X)$, seen as a form of discrete gradient, this trivially becomes a second-order condition, akin to concave functions: A function is submodular if for all $X \subseteq V$ and $j, k \in V$, we have:
$$\nabla_j \nabla_k f(X) \leq 0 \qquad (230)$$
- **Concave 2:** Recall, Theorem 16: composition $h = f \circ g : 2^V \to \mathbb{R}$ (i.e., $h(S) = g(f(S))$) is nondecreasing submodular, if g is non-decreasing concave and f is nondecreasing submodular.
- **Concave 3:** Submodular functions have superdifferentials and supergradients tight at any point.

Submodularity and Concave

- **Concave 1:** A function is submodular if for all $X \subseteq V$ and $j, k \in V$

$$f(X+j) + f(X+k) \geq f(X+j+k) + f(X) \tag{229}$$

- With the gain defined as $\nabla_j(X) = f(X+j) - f(X)$, seen as a form of discrete gradient, this trivially becomes a second-order condition, akin to concave functions: A function is submodular if for all $X \subseteq V$ and $j, k \in V$, we have:

$$\nabla_j \nabla_k f(X) \leq 0 \tag{230}$$

- **Concave 2:** Recall, Theorem 16: composition $h = f \circ g : 2^V \to \mathbb{R}$ (i.e., $h(S) = g(f(S))$) is nondecreasing submodular, if g is non-decreasing concave and f is nondecreasing submodular.
- **Concave 3:** Submodular functions have superdifferentials and supergradients tight at any point.
- **Concave 4:** Concave maximization solved via local gradient ascent. Submodular maximization is (approximately) solvable via greedy (coordinate-ascent-like) algorithms.

Submodularity and neither Concave nor Convex

- **Neither 1:** Submodular functions have simultaneous sub- and super-gradients, tight at any point.

Submodularity and neither Concave nor Convex

- **Neither 1:** Submodular functions have simultaneous sub- and super-gradients, tight at any point.
- **Neither 2:** Concave functions are closed under min, while submodular functions are not.

Submodularity and neither Concave nor Convex

- **Neither 1:** Submodular functions have simultaneous sub- and super-gradients, tight at any point.
- **Neither 2:** Concave functions are closed under min, while submodular functions are not.
- **Neither 3:** Convex functions are closed under max, while submodular functions are not.

Submodularity and neither Concave nor Convex

- **Neither 1:** Submodular functions have simultaneous sub- and super-gradients, tight at any point.
- **Neither 2:** Concave functions are closed under min, while submodular functions are not.
- **Neither 3:** Convex functions are closed under max, while submodular functions are not.
- **Neither 4:** Convex functions can't, in general, be efficiently or approximately maximized, while submodular functions can be.

Submodularity and neither Concave nor Convex

- **Neither 1:** Submodular functions have simultaneous sub- and super-gradients, tight at any point.
- **Neither 2:** Concave functions are closed under min, while submodular functions are not.
- **Neither 3:** Convex functions are closed under max, while submodular functions are not.
- **Neither 4:** Convex functions can't, in general, be efficiently or approximately maximized, while submodular functions can be.
- **Neither 5:** Convex functions have local optimality conditions of the form $\nabla_x f(x) = 0$. Analogous submodular function semi-gradient condition $m(X) = 0$ offers no such guarantee (for neither maximization nor minimization) — although there are other forms of local guarantees.

Outline: Part 3

6 Discrete Semimodular Semigradients

7 Continuous Extensions
- Lovász Extension
- Concave Extension

8 Like Concave or Convex?

9 Optimization

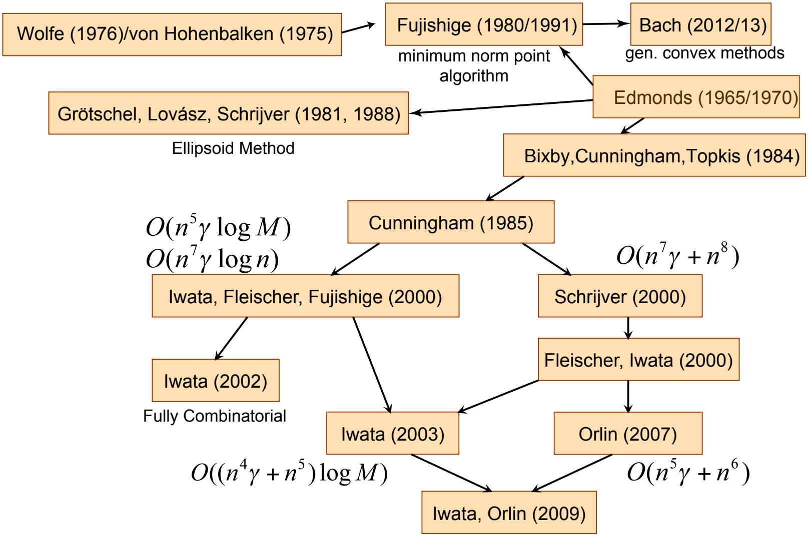
10 Reading

Submodular Optimization Results Summary

	Maximization	Minimization
Unconstrained	In general, NP-hard, greedy gives $1 - 1/e$ approximation for polymatroid cardinality constrained, improved with curvature.	Polynomial time but inefficient $O(n^5\gamma + n^6)$. Special cases (graph representable, sums of concave over modular) much faster, min-norm empirically often works well.
Constrained	NP-hard. For some constraints (matroid, knapsack), approximable with greedy (or approximate concave relaxations). Curvature dependence for combinatorial and submodular constraints.	In general, NP-hard even to approximate, but for many submodular functions still approximable. Curvature dependence for combinatorial and submodular constraints.

SFM Summary (modified from S. Iwata's slides)

General Submodular Function Minimization



Theoretical Results: Constrained Submodular Min

$$\text{minimize} \quad f(S) : S \in \mathcal{S} \tag{231}$$

- Constraint set $\mathcal{S}$ might either be cuts, paths, matchings, cardinality constraints, etc.

Theoretical Results: Constrained Submodular Min

$$\text{minimize} \quad f(S) : S \in \mathcal{S} \tag{231}$$

- Constraint set $\mathcal{S}$ might either be cuts, paths, matchings, cardinality constraints, etc.
- Minimization algorithms should have multiplicative approximation guarantee, i.e,. $f(S) \leq \alpha f(S^*)$ where S^* is optimal solution, $\alpha \geq 1$.

Theoretical Results: Constrained Submodular Min

$$\text{minimize} \quad f(S) : S \in \mathcal{S} \tag{231}$$

- Constraint set $\mathcal{S}$ might either be cuts, paths, matchings, cardinality constraints, etc.
- Minimization algorithms should have multiplicative approximation guarantee, i.e,. $f(S) \leq \alpha f(S^*)$ where S^* is optimal solution, $\alpha \geq 1$.
- In general, how good are the algorithms? Depends on the constraint:

Constraint:	**MMin**	**EA**	**Lower bound**
trees/matchings	n	$\sqrt{m}$	n
cuts	m	$\sqrt{m}$	$\sqrt{m}$
paths	n	$\sqrt{m}$	$n^{2/3}$
cardinality	k	$\sqrt{n}$	$\sqrt{n}$

Goel et al (09), Goemans et al (2009), Jegelka-Bilmes (11) ...

Theoretical Results: Constrained Submodular Min

$$\text{minimize} \quad f(S) : S \in \mathcal{S} \tag{231}$$

- Constraint set $\mathcal{S}$ might either be cuts, paths, matchings, cardinality constraints, etc.
- Minimization algorithms should have multiplicative approximation guarantee, i.e,. $f(S) \leq \alpha f(S^*)$ where S^* is optimal solution, $\alpha \geq 1$.
- In general, how good are the algorithms? Depends on the constraint:

Constraint:	**MMin**	**EA**	**Lower bound**
trees/matchings	n	$\sqrt{m}$	n
cuts	m	$\sqrt{m}$	$\sqrt{m}$
paths	n	$\sqrt{m}$	$n^{2/3}$
cardinality	k	$\sqrt{n}$	$\sqrt{n}$

Goel et al (09), Goemans et al (2009), Jegelka-Bilmes (11) . . .

- Worst case polynomial upper/lower bounds.

Theoretical Results: Constrained Submodular Min

$$\text{minimize} \quad f(S) : S \in \mathcal{S} \tag{231}$$

- Constraint set $\mathcal{S}$ might either be cuts, paths, matchings, cardinality constraints, etc.
- Minimization algorithms should have multiplicative approximation guarantee, i.e,. $f(S) \leq \alpha f(S^*)$ where S^* is optimal solution, $\alpha \geq 1$.
- In general, how good are the algorithms? Depends on the constraint:

Constraint:	**MMin**	**EA**	**Lower bound**
trees/matchings	n	$\sqrt{m}$	n
cuts	m	$\sqrt{m}$	$\sqrt{m}$
paths	n	$\sqrt{m}$	$n^{2/3}$
cardinality	k	$\sqrt{n}$	$\sqrt{n}$

Goel et al (09), Goemans et al (2009), Jegelka-Bilmes (11) ...

- Worst case polynomial upper/lower bounds.
- Other forms of constraints are "easy" (e.g., certain lattices, odd/even sets (see McCormick's SFM tutorial paper).

Submodular Maximization: Unconstrained

- In general, NP-hard. Bound take form $f(S) \geq \alpha f(S^*)$, $\alpha \leq 1$.
- The greedy algorithm for monotone submodular maximization:

Algorithm 2: The Greedy Algorithm

Set $S_0 \leftarrow \emptyset$;
for $i \leftarrow 0 \ldots |V| - 1$ **do**
  Choose v_i as follows: $v_i = \left\{ \operatorname{argmax}_{v \in V \setminus S_i} f(S_i \cup \{v\}) \right\}$;
  Set $S_{i+1} \leftarrow S_i \cup \{v_i\}$;

- has a strong guarantee:

Theorem

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

Submodular Max, Constrained

Monotone Maximization

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

Nonmonotone Maximization

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

, compiled by J. Vondrak

Constrained Submodular Minimization

- Bounds can be improved if we use a functions "curvature"

Constrained Submodular Minimization

- Bounds can be improved if we use a functions "curvature"
- Curvature of a monotone submodular function:

$$\kappa_f(X) \triangleq 1 - \min_j \frac{f(j|X \setminus j)}{f(j)}. \tag{232}$$

The solutions $\widehat{X}$ then have guarantees in terms of curvature κ_f:

$$0 \leq \kappa_f \triangleq \kappa_f(V) \leq 1 \tag{233}$$

Constrained Submodular Minimization

- Bounds can be improved if we use a functions "curvature"
- Curvature of a monotone submodular function:

$$\kappa_f(X) \triangleq 1 - \min_j \frac{f(j|X\setminus j)}{f(j)}. \tag{232}$$

The solutions $\widehat{X}$ then have guarantees in terms of curvature κ_f:

$$0 \leq \kappa_f \triangleq \kappa_f(V) \leq 1 \tag{233}$$

- Curvature dependent constrained maximization bounds:

Constraints	Method	Approximation bound	Lower bound
Cardinality	Greedy	$\frac{1}{\kappa_f}(1-e^{-\kappa_f})$	$\frac{1}{\kappa_f}(1-e^{-\kappa_f})$
Matroid	Greedy	$1/(1+\kappa_f)$	$\frac{1}{\kappa_f}(1-e^{-\kappa_f})$
Knapsack	Greedy	$1-1/e$	$1-1/e$

Constrained Submodular Minimization

- Bounds can be improved if we use a functions "curvature"
- Curvature of a monotone submodular function:

$$\kappa_f(X) \triangleq 1 - \min_j \frac{f(j|X \setminus j)}{f(j)}. \tag{232}$$

The solutions $\widehat{X}$ then have guarantees in terms of curvature κ_f:

$$0 \leq \kappa_f \triangleq \kappa_f(V) \leq 1 \tag{233}$$

- Curvature dependent constrained maximization bounds:

Constraints	Method	Approximation bound	Lower bound
Cardinality	Greedy	$\frac{1}{\kappa_f}(1 - e^{-\kappa_f})$	$\frac{1}{\kappa_f}(1 - e^{-\kappa_f})$
Matroid	Greedy	$1/(1 + \kappa_f)$	$\frac{1}{\kappa_f}(1 - e^{-\kappa_f})$
Knapsack	Greedy	$1 - 1/e$	$1 - 1/e$

- Improve curvature independent bounds when $\kappa_f < 1$.

Curvature Dependent Bounds for Constraint Minimization

- Minimization bounds take the form:

$$f(\widehat{X}) \leq \frac{|X^*|}{1 + (|X^*| - 1)(1 - \kappa_f(X^*))} f(X^*) \leq \frac{1}{1 - \kappa_f(X^*)} f(X^*)$$

Curvature Dependent Bounds for Constraint Minimization

- Minimization bounds take the form:

$$f(\widehat{X}) \leq \frac{|X^*|}{1 + (|X^*| - 1)(1 - \kappa_f(X^*))} f(X^*) \leq \frac{1}{1 - \kappa_f(X^*)} f(X^*)$$

- Lower curvature $\Rightarrow$ Better guarantees!

Curvature Dependent Bounds for Constraint Minimization

- Minimization bounds take the form:

$$f(\widehat{X}) \leq \frac{|X^*|}{1 + (|X^*| - 1)(1 - \kappa_f(X^*))} f(X^*) \leq \frac{1}{1 - \kappa_f(X^*)} f(X^*)$$

- Lower curvature $\Rightarrow$ Better guarantees!

Constraint	Semigradient	Curvature-Ind.	Lower bound
Card. LB	$\frac{k}{1+(k-1)(1-\kappa_f)}$	$\theta(n^{1/2})$	$\tilde{\Omega}(\frac{\sqrt{n}}{1+(\sqrt{n}-1)(1-\kappa_f)})$
Spanning Tree	$\frac{n}{1+(n-1)(1-\kappa_f)}$	$\theta(n)$	$\tilde{\Omega}(\frac{n}{1+(n-1)(1-\kappa_f)})$
Matchings	$\frac{n}{2+(n-2)(1-\kappa_f)}$	$\theta(n)$	$\tilde{\Omega}(\frac{n}{1+(n-1)(1-\kappa_f)})$
s-t path	$\frac{n}{1+(n-1)(1-\kappa_f)}$	$\theta(n^{2/3})$	$\tilde{\Omega}(\frac{n^{2/3}}{1+(n^{2/3}-1)(1-\kappa_f)})$
s-t cut	$\frac{m}{1+(m-1)(1-\kappa_f)}$	$\theta(\sqrt{n})$	$\tilde{\Omega}(\frac{\sqrt{n}}{1+(\sqrt{n}-1)(1-\kappa_f)})$

Summary of results for constrained minimization (Iyer, Jegelka, Bilmes, 2013).

Outline: Part 3

6. Discrete Semimodular Semigradients
7. Continuous Extensions
   - Lovász Extension
   - Concave Extension
8. Like Concave or Convex?
9. Optimization
10. Reading

Classic References

- Jack Edmonds's paper "Submodular Functions, Matroids, and Certain Polyhedra" from 1970.
- Nemhauser, Wolsey, Fisher, "A Analysis of Approximations for Maximizing Submodular Set Functions-I", 1978
- Lovász's paper, "Submodular functions and convexity", from 1983.

Classic Books

- Fujishige, "Submodular Functions and Optimization", 2005
- Narayanan, "Submodular Functions and Electrical Networks", 1997
- Welsh, "Matroid Theory", 1975.
- Oxley, "Matroid Theory", 1992 (and 2011).
- Lawler, "Combinatorial Optimization: Networks and Matroids", 1976.
- Schrijver, "Combinatorial Optimization", 2003
- Gruenbaum, "Convex Polytopes, 2nd Ed", 2003.

Recent online material with an ML slant

- My class, most proofs for above are given. http://j.ee.washington.edu/~bilmes/classes/ee596b_spring_2014/. All lectures being placed on youtube!
- Andreas Krause's web page http://submodularity.org.
- Stefanie Jegelka and Andreas Krause's ICML 2013 tutorial http://techtalks.tv/talks/submodularity-in-machine-learning-new-directions-part-i/58125/
- Francis Bach's updated 2013 text. http://hal.archives-ouvertes.fr/docs/00/87/06/09/PDF/submodular_fot_revised_hal.pdf
- Tom McCormick's overview paper on submodular minimization http://people.commerce.ubc.ca/faculty/mccormick/sfmchap8a.pdf
- Georgia Tech's 2012 workshop on submodularity: http://www.arc.gatech.edu/events/arc-submodularity-workshop

The End: Thank you!