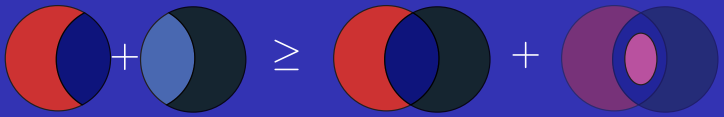


NOML: Submodularity in Machine Learning

June 17th, 2015: Intro and Applications

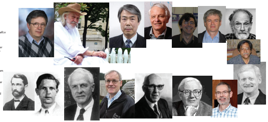
Jeffrey A. Bilmes and Rishabh Iyer

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

June 17th-19th, 2015

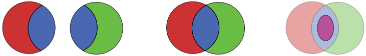
Goals of the Tutorial

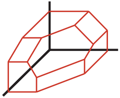
Clockwise from top left:v
Lásló Lovász
Jack Edmonds
Satoru Fujishige
George Nemhauser
Laurence Wolsey
András Frank
Lloyd Shapley
H. Narayanan
Robert Bixby
William Cunningham
William Tutte
Richard Rado
Alexander Schrijver
Garrett Birkhoff
Hassler Whitney
Richard Dedekind



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

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





- Intuitive sense for and familiarity with submodular functions.
- Survey a variety of applications of submodularity in machine learning and beyond.
- Realize why submodularity is important, worthy of study, and should be a standard tool in the tool chest of ML and AI.

On The Submodularity Tutorial

- The definition of submodularity is fairly simple: given a finite ground set V, a function $f : 2^V \to \mathbb{R}$ is said to be submodular if

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

we will revisit this in many forms today

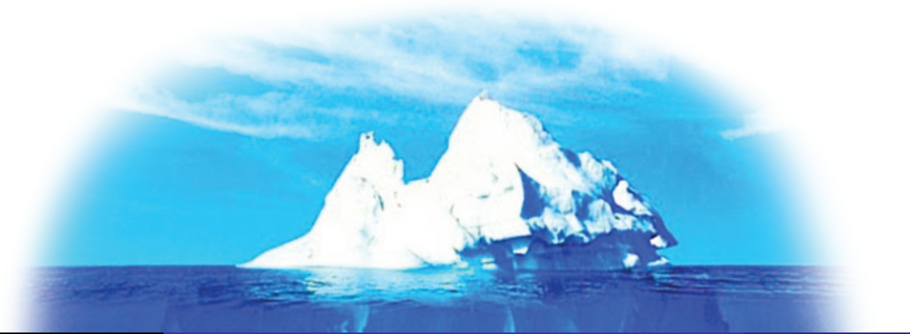
On The Submodularity Tutorial

- The definition of submodularity is fairly simple: given a finite ground set V, a function $f : 2^V \to \mathbb{R}$ is said to be submodular if

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

we will revisit this in many forms today

- The definition, however, is only the tip of the iceberg — this simple definition can lead to great mathematical and practical richness.



Overall Outline of Tutorial

1. Today (Wednesady): Basics, Examples, Properties, and Applications (presented by myself, Jeff Bilmes)
2. Tomorrow: Algorithms for constrained and unconstrained submodular optimization, details of semidifferentials, many novel submodular strucures (presented by Dr. Rishabh Iyer)

Overall Outline of Today's Tutorial

1. Part 1: Basics, Examples, and Properties
2. Part 2: Applications

Outline of Part 1: Basics, Examples, and Properties

Outline of Part 2: Submodular Applications in ML

5. Submodular Applications in Machine Learning
   - Where is submodularity useful?
6. As a model of diversity, coverage, span, or information
7. As a model of cooperative costs, complexity, roughness, and irregularity
8. As a Parameter for an ML algorithm
9. Itself, as a target for learning
10. Surrogates for optimization and analysis
11. Reading
    - Refs

Acknowledgments

Thanks to the following people (former & current students, and current colleagues):

Mukund Narasimhan, Hui Lin, Andrew Guillory, Stefanie Jegelka, Sebastian Tschiatschek, Kai Wei, Yuzong Liu, Rishabh Iyer, Jennifer Gillenwater, Yoshinobu Kawahara, Katrin Kirchhoff, Carlos Guestrin, & Bill Noble.

Outline: Part 1

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: $f(B) = 6$

Simple Costs

FUNNYRECEIPTS.com

TRADER JOE'S

[redacted]

Store [redacted]

OPEN 9:00AM TO 10:00PM DAILY

TJ'S PLAIN SOY MILK	1.69
EGGS BROWN	1.79
VEG TEMPEH ORGANIC 3 GRAIN	1.69
VEG SOY CHORIZO	1.99
PLAIN ORGANIC NONFAT YOGURT 32	2.99
LARGE BABY NON TAXABLE	1.99
GROCERY	0.49
3 @ 3 FOR 0.49	
SUBTOTAL	$12.63
TOTAL	$12.63

- Grocery store: finite set of items V that one can purchase.

Simple Costs

FUNNYRECEIPTS.com

TRADER JOE'S

[redacted]

Store [redacted]

OPEN 9:00AM TO 10:00PM DAILY

TJ'S PLAIN SOY MILK	1.69
EGGS BROWN	1.79
VEG TEMPEH ORGANIC 3 GRAIN	1.69
VEG SOY CHORIZO	1.99
PLAIN ORGANIC NONFAT YOGURT 32	2.99
LARGE BABY NON TAXABLE	1.99
GROCERY	0.49
3 @ 3 FOR 0.49	
SUBTOTAL	$12.63
TOTAL	$12.63

- Grocery store: finite set of items V that one can purchase.
- Each item $v \in V$ has a price $m(v)$.

Simple Costs

- Grocery store: finite set of items V that one can purchase.
- Each item $v \in V$ has a price $m(v)$.
- Basket of groceries $A \subseteq V$ costs:

$$m(A) = \sum_{a \in A} m(a), \tag{2}$$

the sum of individual item costs (no two-for-one discounts).

Simple Costs

- Grocery store: finite set of items V that one can purchase.
- Each item $v \in V$ has a price $m(v)$.
- Basket of groceries $A \subseteq V$ costs:

$$m(A) = \sum_{a \in A} m(a), \tag{2}$$

  the sum of individual item costs (no two-for-one discounts).
- This is known as a modular function.

Discounted Costs

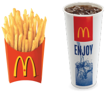
- Let f be the cost of purchasing a set of items (consumer cost).

Discounted Costs

- Let f be the cost of purchasing a set of items (consumer cost). For example, $V = \{\text{"coke"}, \text{"fries"}, \text{"hamburger"}\}$ and $f(A)$ measures the cost of any subset $A \subseteq V$.

Discounted Costs

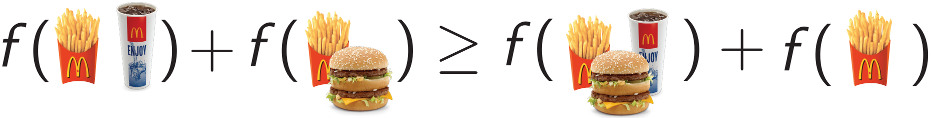
- Let f be the cost of purchasing a set of items (consumer cost). For example, $V = \{$"coke", "fries", "hamburger"$\}$ and $f(A)$ measures the cost of any subset $A \subseteq V$. Then,

$f($ 

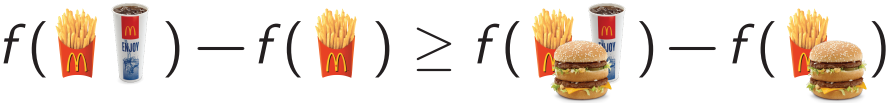
$) + f($  $) \geq f($  $) + f($  $)$

Discounted Costs

- Let f be the cost of purchasing a set of items (consumer cost). For example, $V = \{\text{"coke"}, \text{"fries"}, \text{"hamburger"}\}$ and $f(A)$ measures the cost of any subset $A \subseteq V$. Then,

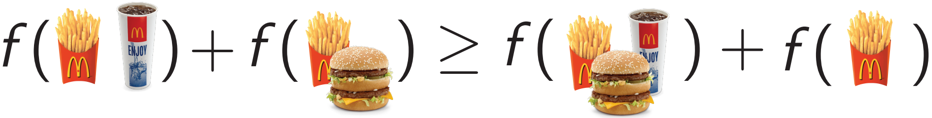


- Rearranging terms, we can see this as a form of diminishing returns:

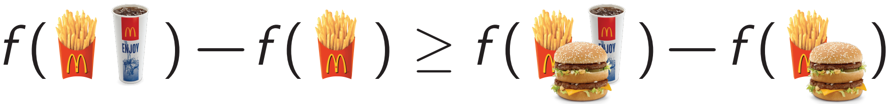


Discounted Costs

- Let f be the cost of purchasing a set of items (consumer cost). For example, $V = \{$"coke", "fries", "hamburger"$\}$ and $f(A)$ measures the cost of any subset $A \subseteq V$. Then,



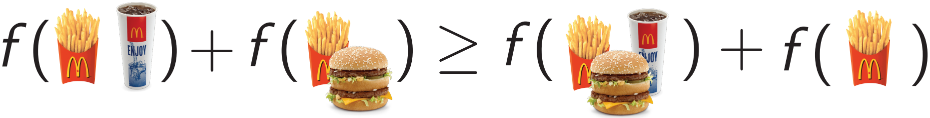
- Rearranging terms, we can see this as a form of diminishing returns:



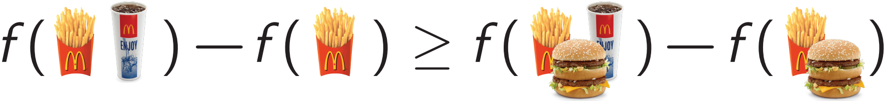
- Typical: additional cost of a coke is free only if you add it to a fries and hamburger order.

Discounted Costs

- Let f be the cost of purchasing a set of items (consumer cost). For example, $V = \{\text{"coke"}, \text{"fries"}, \text{"hamburger"}\}$ and $f(A)$ measures the cost of any subset $A \subseteq V$. Then,



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



- Typical: additional cost of a coke is free only if you add it to a fries and hamburger order.
- Such costs are submodular

Shared Fixed Costs (interacting costs)

- Costs often interact in the real world.

Shared Fixed Costs (interacting 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 (interacting 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 consumer cost of set of items A.

Shared Fixed Costs (interacting 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 consumer cost of set of items A.
- $f(\{v_1\})$ = cost to drive to and from store c_d, and cost to purchase milk c_m, so $f(\{v_1\}) = c_d + c_m$.

Shared Fixed Costs (interacting 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 consumer cost of set of items A.
- $f(\{v_1\})$ = cost to drive to and from store c_d, and cost to purchase milk c_m, so $f(\{v_1\}) = c_d + c_m$.
- $f(\{v_2\})$ = cost to drive to and from store c_d, and cost to purchase honey c_h, so $f(\{v_2\}) = c_d + c_h$.

Shared Fixed Costs (interacting 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 consumer cost of set of items A.
- $f(\{v_1\})$ = cost to drive to and from store c_d, and cost to purchase milk c_m, so $f(\{v_1\}) = c_d + c_m$.
- $f(\{v_2\})$ = cost to drive to and from store c_d, and cost to purchase honey c_h, so $f(\{v_2\}) = c_d + c_h$.
- 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 (interacting 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 consumer cost of set of items A.
- $f(\{v_1\})$ = cost to drive to and from store c_d, and cost to purchase milk c_m, so $f(\{v_1\}) = c_d + c_m$.
- $f(\{v_2\})$ = cost to drive to and from store c_d, and cost to purchase honey c_h, so $f(\{v_2\}) = c_d + c_h$.
- 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

- Let V be a set of possible items to manufacture, and let $f(S)$ for $S \subseteq V$ be the manufacture costs of items in the subset S.

Supply Side Economies of scale

- Let V be a set of possible items to manufacture, and let $f(S)$ for $S \subseteq V$ be the manufacture costs of items in the subset S.
- Ex: V might be paint colors to produce: green, red, blue, yellow, white, etc.

Supply Side Economies of scale

- Let V be a set of possible items to manufacture, and let $f(S)$ for $S \subseteq V$ be the manufacture costs of items in the subset S.
- Ex: V might be paint colors to produce: 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.

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

Supply Side Economies of scale

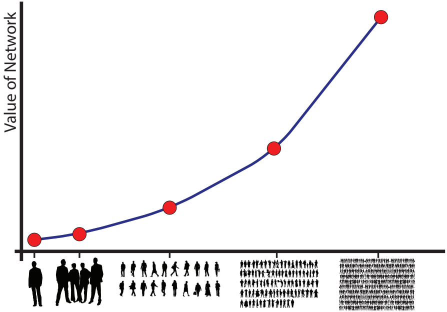
- Let V be a set of possible items to manufacture, and let $f(S)$ for $S \subseteq V$ be the manufacture costs of items in the subset S.
- Ex: V might be paint colors to produce: green, red, blue, yellow, white, etc.
- Producing green when you are already producing yellow and blue is probably cheaper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

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

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

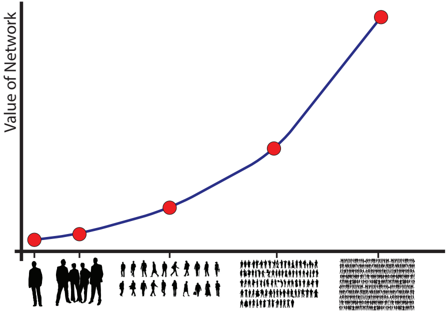
Demand side Economies of Scale: Network Externalities

- Value of a network to a user depends on the number of other users in that network. External use benefits internal use.



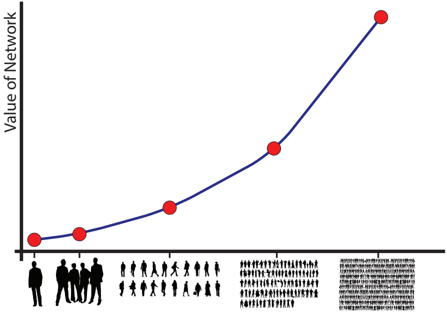
Demand side Economies of Scale: Network Externalities

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



Demand side Economies of Scale: Network Externalities

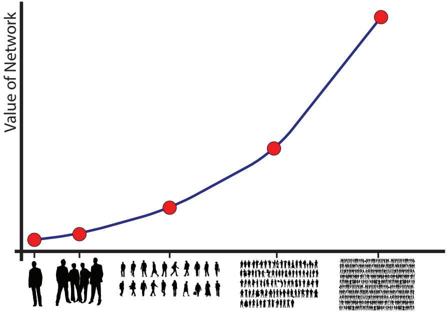
- Value of a network to a user depends on the number of other users in that network. External use benefits internal use.
- Consumers derive positive incremental value when size of the market for that good increases.



- Called network externalities (Katz & Shapiro 1986), or network effects and is a form of demand-side economies of scale

Demand side Economies of Scale: Network Externalities

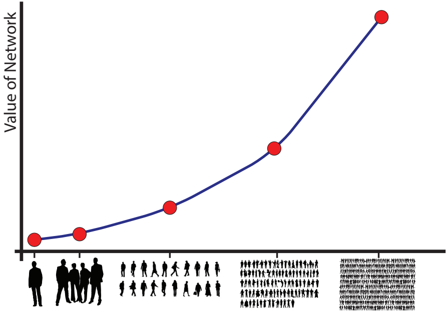
- Value of a network to a user depends on the number of other users in that network. External use benefits internal use.
- Consumers derive positive incremental value when size of the market for that good increases.



- Called network externalities (Katz & Shapiro 1986), or network effects and is a form of demand-side economies of scale
- Ex: durable goods (e.g., a car or phone), software (facebook, smartphone apps), and technology-specific human capital investment (e.g., education in a skill).

Demand side Economies of Scale: Network Externalities

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

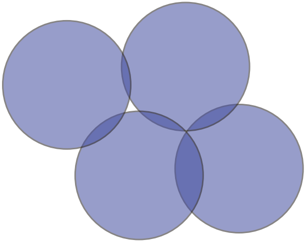


- Called network externalities (Katz & Shapiro 1986), or network effects and is a form of demand-side economies of scale
- Ex: durable goods (e.g., a car or phone), software (facebook, smartphone apps), and technology-specific human capital investment (e.g., education in a skill).
- Let V be a set of goods, A a subset and $v \notin A$. Incremental gain of good $f(A+v) - f(A)$ gets larger as size of market A grows. This is known as a supermodular function.

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.

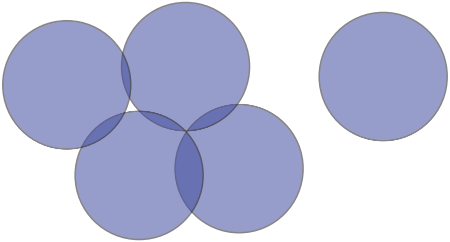
Area of the union of areas indexed by A



Union of areas of elements of A is given by:

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

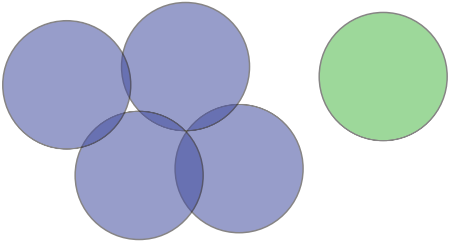
Area of the union of areas indexed by A



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

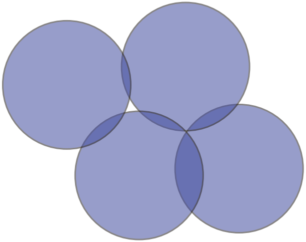


Gain (value) of v in context of A:

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

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

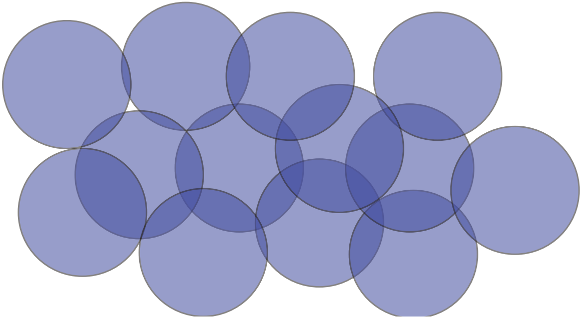
Area of the union of areas indexed by A



Area of A once again.

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

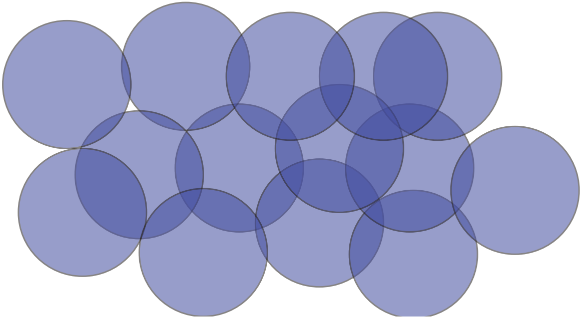
Area of the union of areas indexed by A



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

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

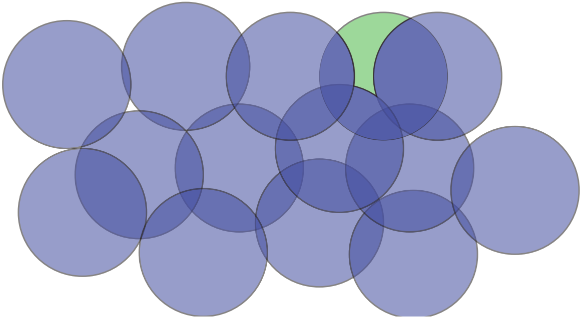
Area of the union of areas indexed by A



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 $B \supset A$.

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

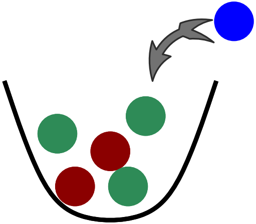
So benefit of v in the context of A is greater than the benefit of v in the context of $B \supseteq A$.

Example Submodular: Number of Colors of Balls in Urns

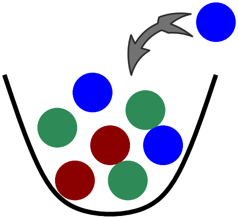
- Consider an urn containing colored balls. Given a set S of balls, $f(S)$ counts the number of distinct colors.

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.



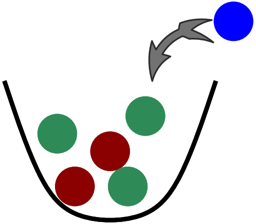
Initial value: 2 (colors in urn).
New value with added blue ball: 3

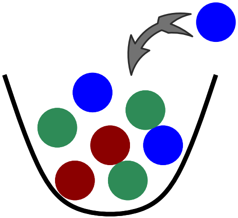


Initial value: 3 (colors in urn).
New value with added blue ball: 3

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.





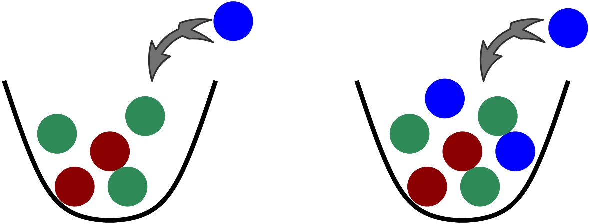
Initial value: 2 (colors in urn).
New value with added blue ball: 3

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.

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:

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B) \tag{3}$$

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:

$$f(A \cup \{v\}) - f(A) \geq f(B \cup \{v\}) - f(B) \tag{4}$$

- 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:

$$f(A) + f(B) \leq f(A \cup B) + f(A \cap B) \tag{5}$$

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:

$$f(A \cup \{v\}) - f(A) \leq f(B \cup \{v\}) - f(B) \tag{6}$$

- The incremental "value", "gain", or "cost" of v increases (improves) as the context in which v is considered grows from A to B.
- A function f is submodular iff $-f$ is supermodular.

Sets and Vectors: Some Notation Conventions

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

Sets and Vectors: Some Notation Conventions

- 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:

$$\mathbf{1}_A(v) = \begin{cases} 1 & \text{if } v \in A \\ 0 & \textit{else} \end{cases} \tag{7}$$

Sets and Vectors: Some Notation Conventions

- 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:

$$\mathbf{1}_A(v) = \begin{cases} 1 & \text{if } v \in A \\ 0 & \textit{else} \end{cases} \tag{7}$$

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

Sets and Vectors: Some Notation Conventions

- 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:

$$\mathbf{1}_A(v) = \begin{cases} 1 & \text{if } v \in A \\ 0 & \textit{else} \end{cases} \tag{7}$$

- 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: Some Notation Conventions

- 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:

$$\mathbf{1}_A(v) = \begin{cases} 1 & \text{if } v \in A \\ 0 & \textit{else} \end{cases} \tag{7}$$

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

Sets and Vectors: Some Notation Conventions

- 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:

$$\mathbf{1}_A(v) = \begin{cases} 1 & \text{if } v \in A \\ 0 & \textit{else} \end{cases} \tag{7}$$

- If $V = \{1, 2, \dots, 20\}$ and $A = \{1, 3, 5, \dots, 19\}$, then $\mathbf{1}_A = (1, 0, 1, 0, \dots)^\intercal$.
- 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.
- Also, it is a bit tedious to write $A \cup \{v\}$ so we instead occasionally write $A + v$.

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

$$m(A) = \sum_{a \in A} m(a) \tag{8}$$

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

$$m(A) = \sum_{a \in A} m(a) \tag{8}$$

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

$$m \in \mathbb{R}^V. \tag{9}$$

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
$$m(A) = \sum_{a \in A} m(a) \tag{8}$$
is called modular and normalized (meaning $m(\emptyset) = 0$).
- Any normalized modular function is identical to a vector:
$$m \in \mathbb{R}^V. \tag{9}$$
- 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

$$m(A) = \sum_{a \in A} m(a) \tag{8}$$

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

- Any normalized modular function is identical to a vector:

$$m \in \mathbb{R}^V. \tag{9}$$

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

$$m(A) = \sum_{a \in A} m(a) \tag{8}$$

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

$$m \in \mathbb{R}^V. \tag{9}$$

- 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

$$m(A) = \sum_{a \in A} m(a) \tag{8}$$

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

$$m \in \mathbb{R}^V. \tag{9}$$

- 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

$$m(A) = \sum_{a \in A} m(a) \tag{8}$$

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

$$m \in \mathbb{R}^V. \tag{9}$$

- 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 m is both submodular and supermodular, then it is modular, meaning $m(A) + m(B) = m(A \cup B) + m(A \cap B)$.

Monotone (nondecreasing) Functions

Definition (monotone function)

A function $f : 2^V \to \mathbb{R}$ is said to be monotone nondecreasing if:

$$f(A) \leq f(B) \text{ whenever } A \subseteq B \subseteq V \tag{10}$$

- Monotone nondecreasing functions are often just called monotone.

Monotone (nondecreasing) Functions

Definition (monotone function)

A function $f : 2^V \to \mathbb{R}$ is said to be monotone nondecreasing if:

$$f(A) \leq f(B) \text{ whenever } A \subseteq B \subseteq V \tag{10}$$

- Monotone nondecreasing functions are often just called monotone.
- Monotone functions have the property that

$$f(A + v) - f(A) \geq 0 \tag{11}$$

for any $A \subseteq V$ and $v \in V$.

Monotone (nondecreasing) Functions

Definition (monotone function)

A function $f : 2^V \to \mathbb{R}$ is said to be monotone nondecreasing if:

$$f(A) \leq f(B) \text{ whenever } A \subseteq B \subseteq V \tag{10}$$

- Monotone nondecreasing functions are often just called monotone.
- Monotone functions have the property that

$$f(A+v) - f(A) \geq 0 \tag{11}$$

  for any $A \subseteq V$ and $v \in V$.
- Monotonicity $\not\Rightarrow$ Submodularity.

Monotone (nondecreasing) Functions

Definition (monotone function)

A function $f : 2^V \to \mathbb{R}$ is said to be monotone nondecreasing if:

$$f(A) \leq f(B) \text{ whenever } A \subseteq B \subseteq V \tag{10}$$

- Monotone nondecreasing functions are often just called monotone.
- Monotone functions have the property that

$$f(A + v) - f(A) \geq 0 \tag{11}$$

  for any $A \subseteq V$ and $v \in V$.
- Monotonicity $\not\Rightarrow$ Submodularity.
- Submodularity $\not\Rightarrow$ Monotonicity.

Polymatroid Functions

Definition (polymatroid function)

- Any function $f : 2^V \to \mathbb{R}$ that is:

Polymatroid Functions

Definition (polymatroid function)

- Any function $f : 2^V \rightarrow \mathbb{R}$ that is:
  1. normalized ($f(\emptyset) = 0$),

Polymatroid Functions

Definition (polymatroid function)

- Any function $f : 2^V \to \mathbb{R}$ that is:
  1. normalized ($f(\emptyset) = 0$),
  2. monotone (nondecreasing), and

Polymatroid Functions

Definition (polymatroid function)

- Any function $f : 2^V \to \mathbb{R}$ that is:
  1. normalized ($f(\emptyset) = 0$),
  2. monotone (nondecreasing), and
  3. submodular

Polymatroid Functions

Definition (polymatroid function)

- Any function $f : 2^V \to \mathbb{R}$ that is:
  1. normalized ($f(\emptyset) = 0$),
  2. monotone (nondecreasing), and
  3. submodular

  is said to be a polymatroid function.

Polymatroid Functions

Definition (polymatroid function)

- Any function $f : 2^V \to \mathbb{R}$ that is:
  1. normalized ($f(\emptyset) = 0$),
  2. monotone (nondecreasing), and
  3. submodular

  is said to be a polymatroid function.

- Thus, a polymatroid function is non-negative since $f(A) \geq f(\emptyset) = 0$.

Polymatroid Functions

Definition (polymatroid function)

- Any function $f : 2^V \to \mathbb{R}$ that is:
  1. normalized ($f(\emptyset) = 0$),
  2. monotone (nondecreasing), and
  3. submodular

  is said to be a polymatroid function.

- Thus, a polymatroid function is non-negative since $f(A) \geq f(\emptyset) = 0$.
- Any submodular function can be written as a difference between a polymatroid function and a modular function. I.e., for any submodular f, we can write:

$$f(A) = f_p(A) - m(A) \tag{12}$$

where f_p is a polymatroid function and m is a modular function.

Subadditive Functions

Definition (subadditive function)

A function $f : 2^V \to \mathbb{R}$ is said to be subadditive if:

$$f(A) + f(B) \geq f(A \cup B) \text{ for all } A, B \subseteq V \tag{13}$$

- Subadditive $\not\Rightarrow$ Submodularity.

Subadditive Functions

Definition (subadditive function)

A function $f : 2^V \to \mathbb{R}$ is said to be subadditive if:

$$f(A) + f(B) \geq f(A \cup B) \text{ for all } A, B \subseteq V \tag{13}$$

- Subadditive $\not\Rightarrow$ Submodularity.
- Submodularity $\not\Rightarrow$ Subadditive.

Subadditive Functions

Definition (subadditive function)

A function $f : 2^V \to \mathbb{R}$ is said to be subadditive if:

$$f(A) + f(B) \geq f(A \cup B) \text{ for all } A, B \subseteq V \tag{13}$$

- Subadditive $\not\Rightarrow$ Submodularity.
- Submodularity $\not\Rightarrow$ Subadditive.
- However, Polymatroidal $\Rightarrow$ Subadditive.

Subadditive Functions

Definition (subadditive function)

A function $f : 2^V \to \mathbb{R}$ is said to be subadditive if:

$$f(A) + f(B) \geq f(A \cup B) \text{ for all } A, B \subseteq V \tag{13}$$

- Subadditive $\not\Rightarrow$ Submodularity.
- Submodularity $\not\Rightarrow$ Subadditive.
- However, Polymatroidal $\Rightarrow$ Subadditive.
- superadditive means that $f(A) + f(B) \leq f(A \cup B)$.

Gain of an item j in the context of A

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

Gain of an item j in the context of A

- 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:

$$
\begin{aligned}
f(A \cup \{j\}) - f(A) &\triangleq \rho_j(A) && (14)\\
&\triangleq \rho_A(j) && (15)\\
&\triangleq \nabla_j f(A) && (16)\\
&\triangleq f(\{j\}|A) && (17)\\
&\triangleq f(j|A) && (18)
\end{aligned}
$$

Gain of an item j in the context of A

- 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:

$$f(A \cup \{j\}) - f(A) \triangleq \rho_j(A) \tag{14}$$

$$\triangleq \rho_A(j) \tag{15}$$

$$\triangleq \nabla_j f(A) \tag{16}$$

$$\triangleq f(\{j\}|A) \tag{17}$$

$$\triangleq f(j|A) \tag{18}$$

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

Gain of an item j in the context of A

- 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:

$$
\begin{aligned}
f(A \cup \{j\}) - f(A) &\triangleq \rho_j(A) && (14)\\
&\triangleq \rho_A(j) && (15)\\
&\triangleq \nabla_j f(A) && (16)\\
&\triangleq f(\{j\}|A) && (17)\\
&\triangleq f(j|A) && (18)
\end{aligned}
$$

- We'll use $f(j|A)$. Also, $f(A|B) = f(A \cup B) - f(B)$.
- Submodularity's diminishing returns stated using gain:

$$\forall j,\ f(j|A) \text{ is a monotone non-increasing function of } A. \qquad (19)$$

True since submodularity means $f(j|A) \geq f(j|B)$ whenever $A \subseteq B$.

Recap: Basic Submodular/Supermodular Definitions

- Set function: map from any subset A of a ground set V to a real number:

$$f : 2^V \to \mathbb{R}$$

- Submodular functions

$$\text{for all } A, B \subseteq V, \quad f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$$

$$\text{for all } A \subseteq B \subseteq V, v \notin B, \quad f(v|A) \geq f(v|B)$$

- Supermodular functions

$$\text{for all } A, B \subseteq V, \quad f(A) + f(B) \leq f(A \cup B) + f(A \cap B)$$

$$\text{for all } A \subseteq B \subseteq V, v \notin B, \quad f(v|A) \leq f(v|B)$$

- Modular functions

$$\text{for all } A, B \subseteq V, \quad f(A) + f(B) = f(A \cup B) + f(A \cap B)$$

$$\text{for all } A \subseteq B \subseteq V, v \notin B, \quad f(v|A) = f(v|B)$$

Many (Equivalent) Definitions of Submodularity

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B), \quad \forall A, B \subseteq V \tag{20}$$

Many (Equivalent) Definitions of Submodularity

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B), \quad \forall A, B \subseteq V \tag{20}$$

$$f(j|S) \geq f(j|T), \forall S \subseteq T \subseteq V, \text{ with } j \in V \setminus T \tag{21}$$

Many (Equivalent) Definitions of Submodularity

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B), \quad \forall A, B \subseteq V \tag{20}$$

$$f(j|S) \geq f(j|T), \ \forall S \subseteq T \subseteq V, \text{ with } j \in V \setminus T \tag{21}$$

$$f(C|S) \geq f(C|T), \forall S \subseteq T \subseteq V, \text{ with } C \subseteq V \setminus T \tag{22}$$

Many (Equivalent) Definitions of Submodularity

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B), \quad \forall A, B \subseteq V \tag{20}$$

$$f(j|S) \geq f(j|T), \; \forall S \subseteq T \subseteq V, \text{ with } j \in V \setminus T \tag{21}$$

$$f(C|S) \geq f(C|T), \forall S \subseteq T \subseteq V, \text{ with } C \subseteq V \setminus T \tag{22}$$

$$f(j|S) \geq f(j|S \cup \{k\}), \; \forall S \subseteq V \text{ with } j \in V \setminus (S \cup \{k\}) \tag{23}$$

Many (Equivalent) Definitions of Submodularity

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B), \quad \forall A, B \subseteq V \tag{20}$$

$$f(j|S) \geq f(j|T), \; \forall S \subseteq T \subseteq V, \text{ with } j \in V \setminus T \tag{21}$$

$$f(C|S) \geq f(C|T), \forall S \subseteq T \subseteq V, \text{ with } C \subseteq V \setminus T \tag{22}$$

$$f(j|S) \geq f(j|S \cup \{k\}), \; \forall S \subseteq V \text{ with } j \in V \setminus (S \cup \{k\}) \tag{23}$$

$$f(A \cup B|A \cap B) \leq f(A|A \cap B) + f(B|A \cap B), \quad \forall A, B \subseteq V \tag{24}$$

Many (Equivalent) Definitions of Submodularity

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B), \quad \forall A, B \subseteq V \tag{20}$$

$$f(j|S) \geq f(j|T), \; \forall S \subseteq T \subseteq V, \text{ with } j \in V \setminus T \tag{21}$$

$$f(C|S) \geq f(C|T), \forall S \subseteq T \subseteq V, \text{ with } C \subseteq V \setminus T \tag{22}$$

$$f(j|S) \geq f(j|S \cup \{k\}), \; \forall S \subseteq V \text{ with } j \in V \setminus (S \cup \{k\}) \tag{23}$$

$$f(A \cup B|A \cap B) \leq f(A|A \cap B) + f(B|A \cap B), \quad \forall A, B \subseteq V \tag{24}$$

$$f(T) \leq f(S) + \sum_{j \in T \setminus S} f(j|S) - \sum_{j \in S \setminus T} f(j|S \cup T - \{j\}), \; \forall S, T \subseteq V \tag{25}$$

Many (Equivalent) Definitions of Submodularity

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B), \;\; \forall A, B \subseteq V \tag{20}$$

$$f(j|S) \geq f(j|T), \; \forall S \subseteq T \subseteq V, \;\text{with } j \in V \setminus T \tag{21}$$

$$f(C|S) \geq f(C|T), \forall S \subseteq T \subseteq V, \;\text{with } C \subseteq V \setminus T \tag{22}$$

$$f(j|S) \geq f(j|S \cup \{k\}), \; \forall S \subseteq V \text{ with } j \in V \setminus (S \cup \{k\}) \tag{23}$$

$$f(A \cup B|A \cap B) \leq f(A|A \cap B) + f(B|A \cap B), \;\; \forall A, B \subseteq V \tag{24}$$

$$f(T) \leq f(S) + \sum_{j \in T \setminus S} f(j|S) - \sum_{j \in S \setminus T} f(j|S \cup T - \{j\}), \; \forall S, T \subseteq V \tag{25}$$

$$f(T) \leq f(S) + \sum_{j \in T \setminus S} f(j|S), \; \forall S \subseteq T \subseteq V \tag{26}$$

Many (Equivalent) Definitions of Submodularity

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B), \quad \forall A, B \subseteq V \tag{20}$$

$$f(j|S) \geq f(j|T), \; \forall S \subseteq T \subseteq V, \text{ with } j \in V \setminus T \tag{21}$$

$$f(C|S) \geq f(C|T), \forall S \subseteq T \subseteq V, \text{ with } C \subseteq V \setminus T \tag{22}$$

$$f(j|S) \geq f(j|S \cup \{k\}), \; \forall S \subseteq V \text{ with } j \in V \setminus (S \cup \{k\}) \tag{23}$$

$$f(A \cup B|A \cap B) \leq f(A|A \cap B) + f(B|A \cap B), \quad \forall A, B \subseteq V \tag{24}$$

$$f(T) \leq f(S) + \sum_{j \in T \setminus S} f(j|S) - \sum_{j \in S \setminus T} f(j|S \cup T - \{j\}), \; \forall S, T \subseteq V \tag{25}$$

$$f(T) \leq f(S) + \sum_{j \in T \setminus S} f(j|S), \; \forall S \subseteq T \subseteq V \tag{26}$$

$$f(T) \leq f(S) - \sum_{j \in S \setminus T} f(j|S \setminus \{j\}) + \sum_{j \in T \setminus S} f(j|S \cap T) \; \forall S, T \subseteq V \tag{27}$$

Many (Equivalent) Definitions of Submodularity

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B), \quad \forall A, B \subseteq V \tag{20}$$

$$f(j|S) \geq f(j|T), \; \forall S \subseteq T \subseteq V, \text{ with } j \in V \setminus T \tag{21}$$

$$f(C|S) \geq f(C|T), \forall S \subseteq T \subseteq V, \text{ with } C \subseteq V \setminus T \tag{22}$$

$$f(j|S) \geq f(j|S \cup \{k\}), \; \forall S \subseteq V \text{ with } j \in V \setminus (S \cup \{k\}) \tag{23}$$

$$f(A \cup B|A \cap B) \leq f(A|A \cap B) + f(B|A \cap B), \quad \forall A, B \subseteq V \tag{24}$$

$$f(T) \leq f(S) + \sum_{j \in T \setminus S} f(j|S) - \sum_{j \in S \setminus T} f(j|S \cup T - \{j\}), \; \forall S, T \subseteq V \tag{25}$$

$$f(T) \leq f(S) + \sum_{j \in T \setminus S} f(j|S), \; \forall S \subseteq T \subseteq V \tag{26}$$

$$f(T) \leq f(S) - \sum_{j \in S \setminus T} f(j|S \setminus \{j\}) + \sum_{j \in T \setminus S} f(j|S \cap T) \; \forall S, T \subseteq V \tag{27}$$

$$f(T) \leq f(S) - \sum_{j \in S \setminus T} f(j|S \setminus \{j\}), \; \forall T \subseteq S \subseteq V \tag{28}$$

Many names exist for submodularity

Many names exist for submodularity

Previous names used for submodularity:

- Submodular
- Attractive
- Associative
- Regular
- Ferromagnetic
- Potts
- Subadditive (but this is now known as something different)
- Strongly Subadditive
- Upper semi-modular
- Monge (of a matrix)
- Fischer-Hadamard inequalities (after a log)
- sub-valuation
- β-functions
- ground set rank function

Many names exist for submodularity

Previous names used for submodularity:

- Submodular
- Attractive
- Associative
- Regular
- Ferromagnetic
- Potts
- Subadditive (but this is now known as something different)
- Strongly Subadditive
- Upper semi-modular
- Monge (of a matrix)
- Fischer-Hadamard inequalities (after a log)
- sub-valuation
- β-functions
- ground set rank function

"What's in a name? That which we call a submodular function,

Outline: Part 1

Set Cover and Maximum Coverage

- We are given a finite set U of m elements and a size-n set of subsets $\mathcal{U} = \{U_1, U_2, \dots, U_n\}$ of U, where $U_i \subseteq U$ and $\bigcup_i U_i = U$.

SET COVER and MAXIMUM COVERAGE

- We are given a finite set U of m elements and a size-n set of subsets $\mathcal{U} = \{U_1, U_2, \dots, U_n\}$ of U, where $U_i \subseteq U$ and $\bigcup_i U_i = U$.
- The goal of minimum SET COVER is to choose the smallest subset $A \subseteq [n] \triangleq \{1, \dots, n\} = V$ such that $\bigcup_{a \in A} U_a = U$.

SET COVER and MAXIMUM COVERAGE

- We are given a finite set U of m elements and a size-n set of subsets $\mathcal{U} = \{U_1, U_2, \dots, U_n\}$ of U, where $U_i \subseteq U$ and $\bigcup_i U_i = U$.
- The goal of minimum SET COVER is to choose the smallest subset $A \subseteq [n] \triangleq \{1, \dots, n\} = V$ such that $\bigcup_{a \in A} U_a = U$.
- Maximum k cover: The goal in MAXIMUM COVERAGE is, given an integer $k \leq n$, select k subsets, say $\{a_1, a_2, \dots, a_k\}$ with $a_i \in [n]$ such that $|\bigcup_{i=1}^{k} U_{a_i}|$ is maximized.

Set Cover and Maximum Coverage

- We are given a finite set U of m elements and a size-n set of subsets $\mathcal{U} = \{U_1, U_2, \ldots, U_n\}$ of U, where $U_i \subseteq U$ and $\bigcup_i U_i = U$.
- The goal of minimum SET COVER is to choose the smallest subset $A \subseteq [n] \triangleq \{1, \ldots, n\} = V$ such that $\bigcup_{a \in A} U_a = U$.
- Maximum k cover: The goal in MAXIMUM COVERAGE is, given an integer $k \leq n$, select k subsets, say $\{a_1, a_2, \ldots, a_k\}$ with $a_i \in [n]$ such that $|\bigcup_{i=1}^k U_{a_i}|$ 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 U of m elements and a size-n set of subsets $\mathcal{U} = \{U_1, U_2, \dots, U_n\}$ of U, where $U_i \subseteq U$ and $\bigcup_i U_i = U$.
- The goal of minimum SET COVER is to choose the smallest subset $A \subseteq [n] \triangleq \{1, \dots, n\} = V$ such that $\bigcup_{a \in A} U_a = U$.
- Maximum k cover: The goal in MAXIMUM COVERAGE is, given an integer $k \leq n$, select k subsets, say $\{a_1, a_2, \dots, a_k\}$ with $a_i \in [n]$ such that $|\bigcup_{i=1}^{k} U_{a_i}|$ 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 $f(A) = |\bigcup_{a \in A} U_a|$ is submodular!

Set Cover and Maximum Coverage

- We are given a finite set U of m elements and a size-n set of subsets $\mathcal{U} = \{U_1, U_2, \dots, U_n\}$ of U, where $U_i \subseteq U$ and $\bigcup_i U_i = U$.
- The goal of minimum SET COVER is to choose the smallest subset $A \subseteq [n] \triangleq \{1, \dots, n\} = V$ such that $\bigcup_{a \in A} U_a = U$.
- Maximum k cover: The goal in MAXIMUM COVERAGE is, given an integer $k \leq n$, select k subsets, say $\{a_1, a_2, \dots, a_k\}$ with $a_i \in [n]$ such that $|\bigcup_{i=1}^k U_{a_i}|$ 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 $f(A) = |\bigcup_{a \in A} U_a|$ is submodular!
- $f(A) = \mu(\bigcup_{i=1}^k U_{a_i})$ is still submodular if we take $U \subseteq \mathbb{R}^\ell$ and $U_i \subseteq U$ and $\mu(\cdot)$ is an additive measure (e.g., the Lebesgue measure).

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 $V \setminus X$.

$$f(X) = \left|\{(u, v) \in E : u \in X, v \in V \setminus X\}\right| \tag{29}$$

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 $V \setminus X$.

$$f(X) = \left|\{(u, v) \in E : u \in X, v \in V \setminus X\}\right| \tag{29}$$

- 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 $V \setminus X$.

$$f(X) = \left|\{(u, v) \in E : u \in X, v \in V \setminus X\}\right| \tag{29}$$

- 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 maximize 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 $V \setminus X$.

$$f(X) = \Big|\{(u, v) \in E : u \in X, v \in V \setminus X\}\Big| \quad (29)$$

- 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 maximize 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.

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

$$= \sum_{e \in \{(u,v) \in E : u \in X, v \in V \setminus X\}} w(e) \quad (31)$$

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 $V \setminus X$.

$$f(X) = \left|\{(u, v) \in E : u \in X, v \in V \setminus X\}\right| \tag{29}$$

- 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 maximize 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.

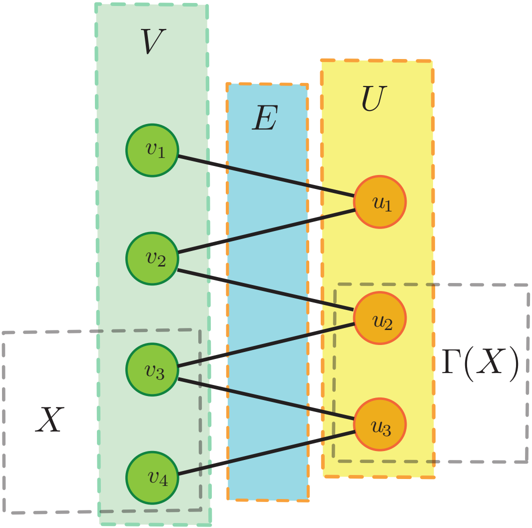
$$f(X) = w\Big(\{(u, v) \in E : u \in X, v \in V \setminus X\}\Big) \tag{30}$$

$$= \sum_{e \in \{(u,v) \in E : u \in X, v \in V \setminus X\}} w(e) \tag{31}$$

- Both functions (Equations (29) and (30)) are submodular.

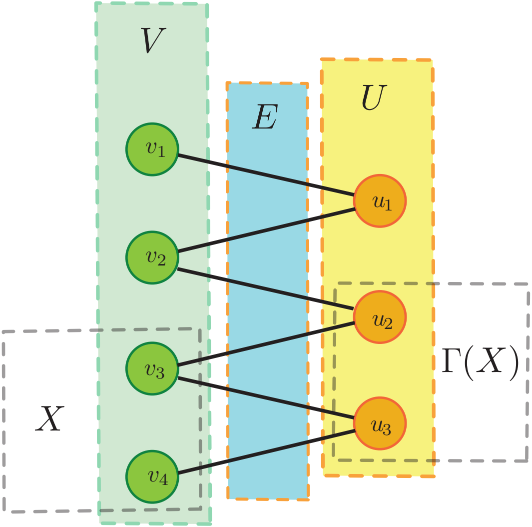
Bipartite Neighborhood Function

- Let $G = (V, U, E, w)$ be a weighted bipartite graph, where V (resp. U) is a set of left (resp. right) nodes, E is a set of edges, and $w : 2^U \to \mathbb{R}_+$ is a modular function on right nodes.



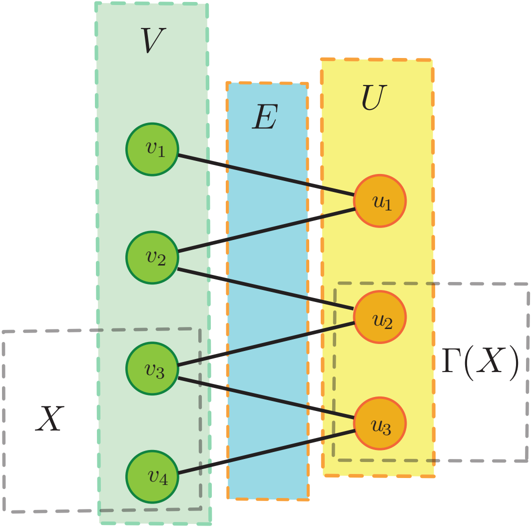
Bipartite Neighborhood Function

- Let $G = (V, U, E, w)$ be a weighted bipartite graph, where V (resp. U) is a set of left (resp. right) nodes, E is a set of edges, and $w : 2^U \to \mathbb{R}_+$ is a modular function on right nodes.
- Neighbors function: $\Gamma(X) = \{u \in U : |X \times \{u\} \cap E| \geq 1\}$ for $X \subseteq V$.



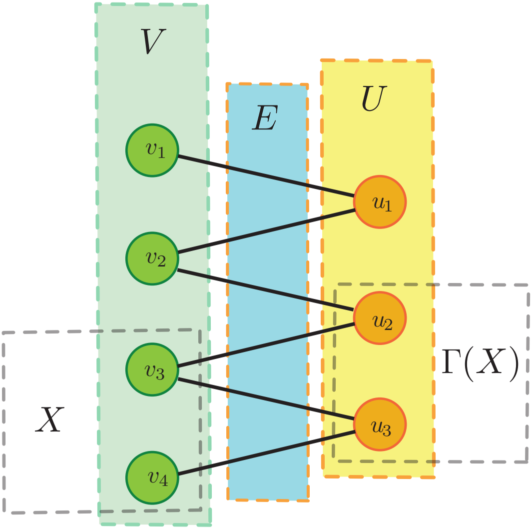
Bipartite Neighborhood Function

- Let $G = (V, U, E, w)$ be a weighted bipartite graph, where V (resp. U) is a set of left (resp. right) nodes, E is a set of edges, and $w : 2^U \to \mathbb{R}_+$ is a modular function on right nodes.
- Neighbors function: $\Gamma(X) = \{u \in U : |X \times \{u\} \cap E| \geq 1\}$ for $X \subseteq V$.
- Size of neighbors, $f(X) = |\Gamma(X)|$ is submodular.



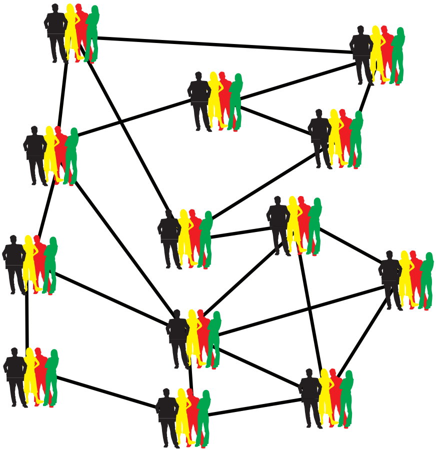
Bipartite Neighborhood Function

- Let $G = (V, U, E, w)$ be a weighted bipartite graph, where V (resp. U) is a set of left (resp. right) nodes, E is a set of edges, and $w : 2^U \to \mathbb{R}_+$ is a modular function on right nodes.
- Neighbors function: $\Gamma(X) = \{u \in U : |X \times \{u\} \cap E| \geq 1\}$ for $X \subseteq V$.
- Size of neighbors, $f(X) = |\Gamma(X)|$ is submodular.
- Weight of neighbors, $f(X) = w(\Gamma(X))$ is also submodular.



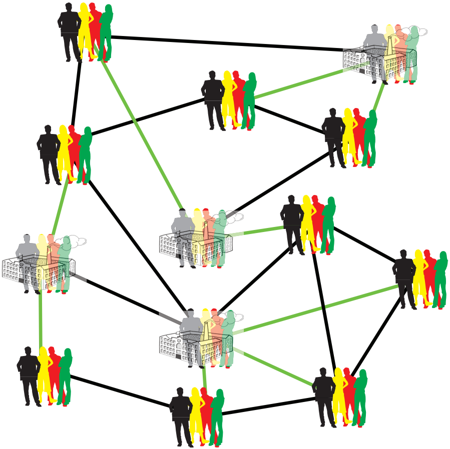
Facility/Plant Location (uncapacitated)

- Core problem in operations research, early motivation for submodularity.
- Goal: as efficiently as possible, place “facilities” (factories) at certain locations to satisfy sites (at all locations) having various demands.



Facility/Plant Location (uncapacitated)

- Core problem in operations research, early motivation for submodularity.
- Goal: as efficiently as possible, place "facilities" (factories) at certain locations to satisfy sites (at all locations) having various demands.

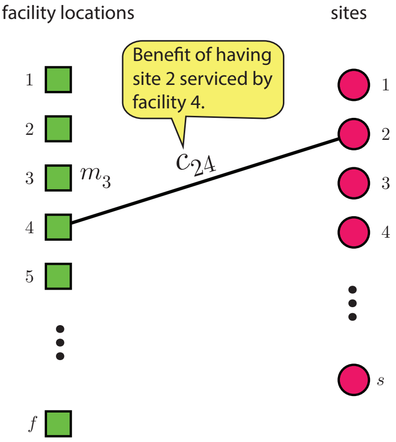


Facility/Plant Location (uncapacitated)

- Core problem in operations research, early motivation for submodularity.
- Goal: as efficiently as possible, place "facilities" (factories) at certain locations to satisfy sites (at all locations) having various demands.

- We can model this with a weighted bipartite graph $G = (F, S, E, c)$ where F is set of possible factory/plant locations, S is set of sites needing service, E are edges indicating (factory,site) service possibility pairs, and $c : E \to \mathbb{R}_+$ is the benefit of a given pair.
- Facility location function has form:

$$f(A) = \sum_{i \in F} \max_{j \in A} c_{ij}. \qquad (32)$$

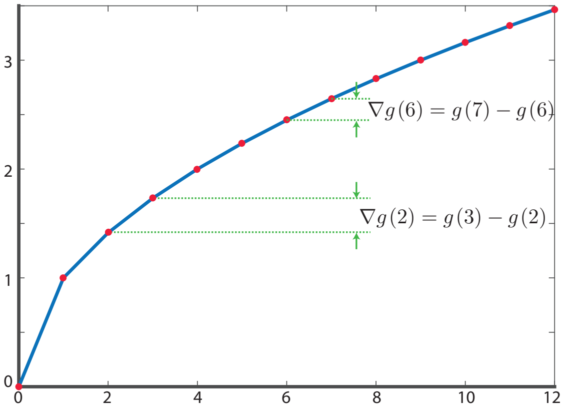


Square root of cardinality

- Define a function $f : 2^V \to \mathbb{R}_+$ as follows:

$$f(A) = \sqrt{|A|},$$

square root of cardinality of A.

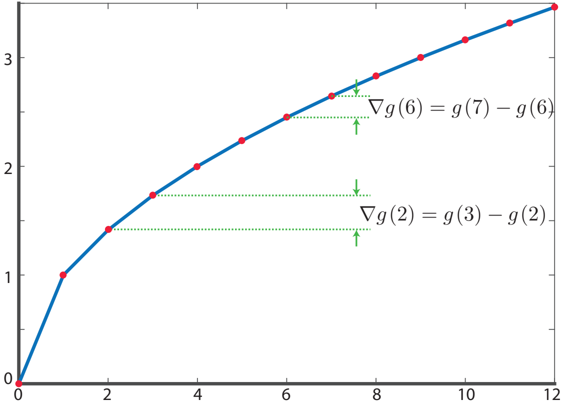


Square root of cardinality

- Define a function $f : 2^V \to \mathbb{R}_+$ as follows:

$$f(A) = \sqrt{|A|},$$

square root of cardinality of A.



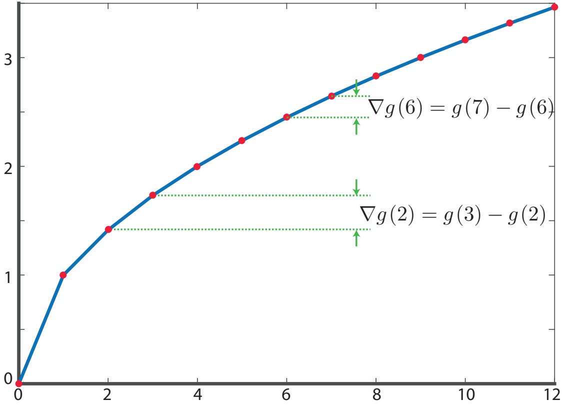
- This is a concave function (i.e., square root) composed with a modular function ($m(A) = \sum_{a \in A} m(a)$ where $m(a) = 1$).

Square root of cardinality

- Define a function $f : 2^V \to \mathbb{R}_+$ as follows:

$$f(A) = \sqrt{|A|},$$

square root of cardinality of A.



- This is a concave function (i.e., square root) composed with a modular function ($m(A) = \sum_{a \in A} m(a)$ where $m(a) = 1$).
- $\nabla g(i) > \nabla g(j)$ for $j > i$ by concavity, so f is a submodular function.

Concave function composed with a modular function

- Let $g : \mathbb{R} \to \mathbb{R}$ be any concave function.

Concave function composed with a modular function

- Let $g : \mathbb{R} \to \mathbb{R}$ be any concave function.
- Let $m : 2^V \to \mathbb{R}_+$ be any modular function with non-negative entries (i.e., $m(v) \geq 0$ for all $v \in V$).

Concave function composed with a modular function

- Let $g : \mathbb{R} \to \mathbb{R}$ be any concave function.
- Let $m : 2^V \to \mathbb{R}_+$ be any modular function with non-negative entries (i.e., $m(v) \geq 0$ for all $v \in V$).
- Then $f : 2^V \to \mathbb{R}$ defined as

$$f(A) = g(m(A)) \tag{33}$$

is a submodular function.

Concave function composed with a modular function

- Let $g : \mathbb{R} \to \mathbb{R}$ be any concave function.
- Let $m : 2^V \to \mathbb{R}_+$ be any modular function with non-negative entries (i.e., $m(v) \geq 0$ for all $v \in V$).
- Then $f : 2^V \to \mathbb{R}$ defined as

$$f(A) = g(m(A)) \tag{33}$$

  is a submodular function.
- Given a set of such concave functions $\{g_i\}$ and modular functions $\{m_i\}$, then the sum of such functions

$$f(A) = \sum_i g_i(m_i(A)) \tag{34}$$

  is also submodular.

Concave function composed with a modular function

- Let $g : \mathbb{R} \to \mathbb{R}$ be any concave function.
- Let $m : 2^V \to \mathbb{R}_+$ be any modular function with non-negative entries (i.e., $m(v) \geq 0$ for all $v \in V$).
- Then $f : 2^V \to \mathbb{R}$ defined as
$$f(A) = g(m(A)) \tag{33}$$
is a submodular function.
- Given a set of such concave functions $\{g_i\}$ and modular functions $\{m_i\}$, then the sum of such functions
$$f(A) = \sum_i g_i(m_i(A)) \tag{34}$$
is also submodular.
- Very large class of functions, including graph cut, bipartite neighborhoods, set cover (Stobbe & Krause).

Concave function composed with a modular function

- Let $g : \mathbb{R} \to \mathbb{R}$ be any concave function.
- Let $m : 2^V \to \mathbb{R}_+$ be any modular function with non-negative entries (i.e., $m(v) \geq 0$ for all $v \in V$).
- Then $f : 2^V \to \mathbb{R}$ defined as

$$f(A) = g(m(A)) \tag{33}$$

  is a submodular function.
- Given a set of such concave functions $\{g_i\}$ and modular functions $\{m_i\}$, then the sum of such functions

$$f(A) = \sum_i g_i(m_i(A)) \tag{34}$$

  is also submodular.
- Very large class of functions, including graph cut, bipartite neighborhoods, set cover (Stobbe & Krause).
- However, Vondrak showed that a simple matroid rank function (defined below) which is submodular is not a member.

Example: Rank function of a matrix

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

$$\mathbf{X} = \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & & m \end{array} \\ \begin{pmatrix} | & | & | & | & & | \\ x_1 & x_2 & x_3 & x_4 & \dots & x_m \\ | & | & | & | & & | \end{pmatrix} \end{array} \tag{35}$$

- 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

Ex: a 4×8 matrix with column index set $V = \{1,2,3,4,5,6,7,8\}$.

$$\begin{array}{c|cccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \hline 1 & & & & & & & & \\ 2 & \multicolumn{8}{c}{\left(\begin{array}{cccccccc} 0 & 2 & 2 & 3 & 0 & 1 & 3 & 1 \\ 0 & 3 & 0 & 4 & 0 & 0 & 2 & 4 \\ 0 & 0 & 0 & 0 & 3 & 0 & 0 & 5 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 5 \end{array}\right)} \\ 3 & & & & & & & & \\ 4 & & & & & & & & \end{array}$$

$$\begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array}$$

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

- 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

Ex: a 4×8 matrix with column index set $V = \{1,2,3,4,5,6,7,8\}$.

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

- 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

Ex: a 4×8 matrix with column index set $V = \{1,2,3,4,5,6,7,8\}$.

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

- 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

Ex: a 4×8 matrix with column index set $V = \{1,2,3,4,5,6,7,8\}$.

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

- 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

Ex: a 4×8 matrix with column index set $V = \{1,2,3,4,5,6,7,8\}$.

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

- 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

Ex: a 4×8 matrix with column index set $V = \{1,2,3,4,5,6,7,8\}$.

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

- 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

Ex: a 4×8 matrix with column index set $V = \{1,2,3,4,5,6,7,8\}$.

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

- 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

Ex: a 4×8 matrix with column index set $V = \{1, 2, 3, 4, 5, 6, 7, 8\}$.

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

- 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

Ex: a 4×8 matrix with column index set $V = \{1, 2, 3, 4, 5, 6, 7, 8\}$.

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

- 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

Ex: a 4×8 matrix with column index set $V = \{1,2,3,4,5,6,7,8\}$.

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

- 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

Ex: a 4×8 matrix with column index set $V = \{1,2,3,4,5,6,7,8\}$.

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

- 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

Ex: a 4×8 matrix with column index set $V = \{1,2,3,4,5,6,7,8\}$.

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

- 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

Ex: a 4×8 matrix with column index set $V = \{1,2,3,4,5,6,7,8\}$.

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

- 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

Ex: a 4×8 matrix with column index set $V = \{1,2,3,4,5,6,7,8\}$.

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

- 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

Ex: a 4×8 matrix with column index set $V = \{1,2,3,4,5,6,7,8\}$.

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

- 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

Ex: a 4×8 matrix with column index set $V = \{1, 2, 3, 4, 5, 6, 7, 8\}$.

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

- 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

Ex: a 4×8 matrix with column index set $V = \{1,2,3,4,5,6,7,8\}$.

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

- 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

Ex: a 4×8 matrix with column index set $V = \{1, 2, 3, 4, 5, 6, 7, 8\}$.

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

- 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

Ex: a 4×8 matrix with column index set $V = \{1, 2, 3, 4, 5, 6, 7, 8\}$.

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

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

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.

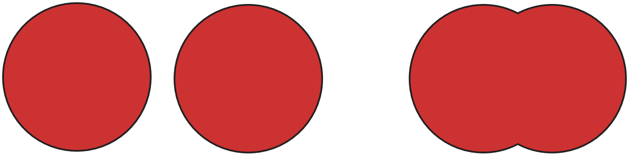
Rank function of a matrix

- 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.

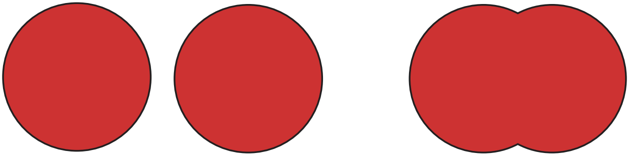
$$r(A) \quad + \quad r(B) \qquad \geq \qquad r(A \cup B)$$



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.

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

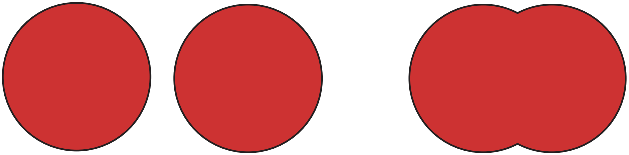


- If some of the dimensions spanned by A overlap some of the dimensions spanned by B (i.e., if $\exists$ common span), then that area is counted twice in $r(A) + r(B)$, so the inequality will be strict.

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 $\exists$ common span), 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.,

$$r(A) + r(B) = r(A_r) + 2r(C) + r(B_r). \tag{36}$$

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.,

$$r(A) + r(B) = r(A_r) + 2r(C) + r(B_r). \tag{36}$$

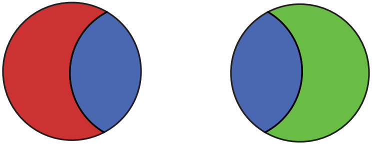
- But $r(A \cup B)$ counts the dimensions spanned by C only once.

$$r(A \cup B) = r(A_r) + r(C) + r(B_r) \tag{37}$$

Rank functions of a matrix

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

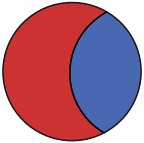
$$r(A) + r(B) = {\color{red} r(A_r)} + 2{\color{blue} r(C)} + {\color{green} r(B_r)}$$

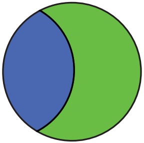


Rank functions of a matrix

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

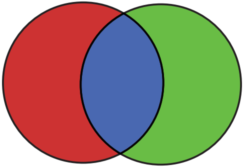
$$r(A) + r(B) = r(A_r) + 2r(C) + r(B_r)$$





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

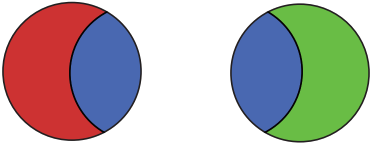
$$r(A \cup B) = r(A_r) + r(C) + r(B_r)$$



Rank functions of a matrix

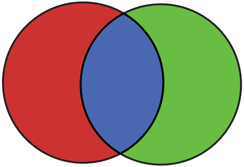
- Then $r(A) + r(B)$ counts the dimensions spanned by C twice, i.e.,

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



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

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

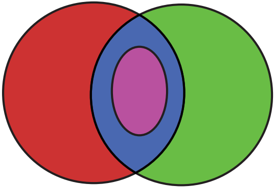


- Thus, we have subadditivity: $r(A) + r(B) \geq r(A \cup B)$. Can we add more to the r.h.s. and still have an inequality? Yes.

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).

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

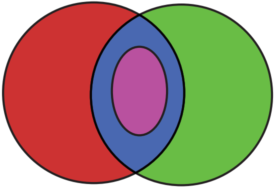


In short:

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).

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



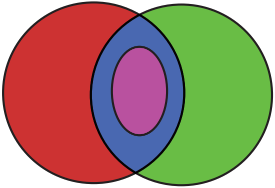
In short:

- 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).

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

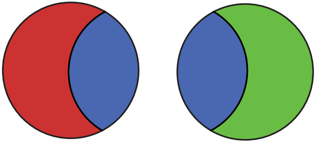


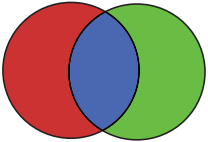
In short:

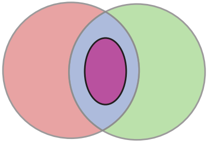
- Common span (blue) is “more” (no less) than span of common index (magenta).
- More generally, common information (blue) is “more” (no less) than information within common index (magenta).

The Venn and Art of Submodularity

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







Matroid

Definition (set system)

A (finite) ground set V and a set of subsets of V, $\emptyset \neq \mathcal{I} \subseteq 2^V$ is called a set system, notated $(V, \mathcal{I})$.

Matroid

Definition (set system)

A (finite) ground set V and a set of subsets of V, $\emptyset \neq \mathcal{I} \subseteq 2^V$ is called a set system, notated $(V, \mathcal{I})$.

Definition (independence (or hereditary) system)

A set system $(V, \mathcal{I})$ is an independence system if

$$\emptyset \in \mathcal{I} \quad \text{(emptyset containing)} \tag{I1}$$

and

$$\forall I \in \mathcal{I}, J \subset I \Rightarrow J \in \mathcal{I} \quad \text{(subclusive)} \tag{I2}$$

Matroid

Definition (set system)

A (finite) ground set V and a set of subsets of V, $\emptyset \neq \mathcal{I} \subseteq 2^V$ is called a set system, notated $(V, \mathcal{I})$.

Definition (independence (or hereditary) system)

A set system $(V, \mathcal{I})$ is an independence system if

$$\emptyset \in \mathcal{I} \quad \text{(emptyset containing)} \tag{I1}$$

and

$$\forall I \in \mathcal{I}, J \subset I \Rightarrow J \in \mathcal{I} \quad \text{(subclusive)} \tag{I2}$$

Definition (Matroid)

A set system $(V, \mathcal{I})$ is a Matroid if

Matroid

Definition (set system)

A (finite) ground set V and a set of subsets of V, $\emptyset \neq \mathcal{I} \subseteq 2^V$ is called a set system, notated $(V, \mathcal{I})$.

Definition (independence (or hereditary) system)

A set system $(V, \mathcal{I})$ is an independence system if

$$\emptyset \in \mathcal{I} \quad \text{(emptyset containing)} \tag{I1}$$

and

$$\forall I \in \mathcal{I}, J \subset I \Rightarrow J \in \mathcal{I} \quad \text{(subclusive)} \tag{I2}$$

Definition (Matroid)

A set system $(V, \mathcal{I})$ is a Matroid if

(I1) $\emptyset \in \mathcal{I}$ (emptyset containing)

Matroid

Definition (set system)

A (finite) ground set V and a set of subsets of V, $\emptyset \neq \mathcal{I} \subseteq 2^V$ is called a set system, notated $(V, \mathcal{I})$.

Definition (independence (or hereditary) system)

A set system $(V, \mathcal{I})$ is an independence system if

$$\emptyset \in \mathcal{I} \quad \text{(emptyset containing)} \tag{I1}$$

and

$$\forall I \in \mathcal{I}, J \subset I \Rightarrow J \in \mathcal{I} \quad \text{(subclusive)} \tag{I2}$$

Definition (Matroid)

A set system $(V, \mathcal{I})$ is a Matroid if

(I1) $\emptyset \in \mathcal{I}$ (emptyset containing)

(I2) $\forall I \in \mathcal{I}, J \subset I \Rightarrow J \in \mathcal{I}$ (down-closed or subclusive)

Matroid

Definition (set system)

A (finite) ground set V and a set of subsets of V, $\emptyset \neq \mathcal{I} \subseteq 2^V$ is called a set system, notated $(V, \mathcal{I})$.

Definition (independence (or hereditary) system)

A set system $(V, \mathcal{I})$ is an independence system if

$$\emptyset \in \mathcal{I} \quad \text{(emptyset containing)} \tag{I1}$$

and

$$\forall I \in \mathcal{I}, J \subset I \Rightarrow J \in \mathcal{I} \quad \text{(subclusive)} \tag{I2}$$

Definition (Matroid)

A set system $(V, \mathcal{I})$ is a Matroid if

(I1) $\emptyset \in \mathcal{I}$ (emptyset containing)

(I2) $\forall I \in \mathcal{I}, J \subset I \Rightarrow J \in \mathcal{I}$ (down-closed or subclusive)

(I3) $\forall I, J \in \mathcal{I}$, with $|I| = |J| + 1$, then $\exists\ x \in I \setminus J$ s.t. $J \cup \{x\} \in \mathcal{I}$.

A matroid rank function is submodular

We can a bit more formally define the rank function this way.

Definition

The rank of a matroid is a function $r : 2^V \to \mathbb{Z}_+$ defined by

$$r(A) = \max \{|X| : X \subseteq A, X \in \mathcal{I}\} = \max_{X \in \mathcal{I}} |A \cap X| \tag{38}$$

A matroid rank function is submodular

We can a bit more formally define the rank function this way.

Definition

The rank of a matroid is a function $r : 2^V \to \mathbb{Z}_+$ defined by

$$r(A) = \max\{|X| : X \subseteq A, X \in \mathcal{I}\} = \max_{X \in \mathcal{I}} |A \cap X| \tag{38}$$

- From the above, we immediately see that $r(A) \leq |A|$.

A matroid rank function is submodular

We can a bit more formally define the rank function this way.

Definition

The rank of a matroid is a function $r : 2^V \to \mathbb{Z}_+$ defined by

$$r(A) = \max\{|X| : X \subseteq A, X \in \mathcal{I}\} = \max_{X \in \mathcal{I}} |A \cap X| \tag{38}$$

- From the above, we immediately see that $r(A) \leq |A|$.
- Moreover, if $r(A) = |A|$, then $A \in \mathcal{I}$, meaning A is independent

A matroid rank function is submodular

We can a bit more formally define the rank function this way.

Definition

The rank of a matroid is a function $r : 2^V \to \mathbb{Z}_+$ defined by

$$r(A) = \max\{|X| : X \subseteq A, X \in \mathcal{I}\} = \max_{X \in \mathcal{I}} |A \cap X| \tag{38}$$

- From the above, we immediately see that $r(A) \leq |A|$.
- Moreover, if $r(A) = |A|$, then $A \in \mathcal{I}$, meaning A is independent

Lemma

The rank function $r : 2^V \to \mathbb{Z}_+$ of a matroid is submodular, that is $r(A) + r(B) \geq r(A \cup B) + r(A \cap B)$

Example: Partition Matroid

Ground set of objects, $V = \{$

$\}$

Example: Partition Matroid

Partition of V into six blocks, $V_1, V_2, \dots, V_6$

Example: Partition Matroid

Limit associated with each block, $\{k_1, k_2, \ldots, k_6\}$

Example: Partition Matroid

Independent subset but not maximally independent.

Example: Partition Matroid

Maximally independent subset, what is called a base.

Example: Partition Matroid

Not independent since over limit in set six.

Information and Complexity functions

- Given a collection of random variables $X_1, X_2, \ldots, X_n$ then entropy $H(X_1, \ldots, X_n)$ is the information in those n random variables.

Information and Complexity functions

- Given a collection of random variables $X_1, X_2, \ldots, X_n$ then entropy $H(X_1, \ldots, X_n)$ is the information in those n random variables.
- Define $V = \{1, 2, \ldots, n\} \triangleq [n]$ to be the set of integers (indices).

Information and Complexity functions

- Given a collection of random variables $X_1, X_2, \ldots, X_n$ then entropy $H(X_1, \ldots, X_n)$ is the information in those n random variables.
- Define $V = \{1, 2, \ldots, n\} \triangleq [n]$ to be the set of integers (indices).
- Consider a function $f : 2^V \to \mathbb{R}_+$ where $f(A)$ is entropy of the subset $A = \{a_1, a_2, \ldots, a_k\} \subseteq V$ of random variables:

$$f(A) = H(X_A) = H(X_{a_1}, X_{a_2}, \ldots, X_{a_k}) = -\sum_{x_A} \Pr(x_A) \log \Pr(x_A)$$

Information and Complexity functions

- Given a collection of random variables $X_1, X_2, \dots, X_n$ then entropy $H(X_1, \dots, X_n)$ is the information in those n random variables.
- Define $V = \{1, 2, \dots, n\} \triangleq [n]$ to be the set of integers (indices).
- Consider a function $f : 2^V \to \mathbb{R}_+$ where $f(A)$ is entropy of the subset $A = \{a_1, a_2, \dots, a_k\} \subseteq V$ of random variables:

$$f(A) = H(X_A) = H(X_{a_1}, X_{a_2}, \dots, X_{a_k}) = -\sum_{x_A} \Pr(x_A) \log \Pr(x_A)$$

- Entropy is submodular due to non-negativity of conditional mutual information. Given $A, B, C \subseteq V$,

$$\begin{aligned} &I(X_{A\setminus B}; X_{B\setminus A} | X_{A\cap B}) \\ &\quad = H(X_A) + H(X_B) - H(X_{A\cup B}) - H(X_{A\cap B}) \geq 0 \end{aligned} \tag{39}$$

Information and Complexity functions

- Given a collection of random variables $X_1, X_2, \ldots, X_n$ then entropy $H(X_1, \ldots, X_n)$ is the information in those n random variables.
- Define $V = \{1, 2, \ldots, n\} \triangleq [n]$ to be the set of integers (indices).
- Consider a function $f : 2^V \to \mathbb{R}_+$ where $f(A)$ is entropy of the subset $A = \{a_1, a_2, \ldots, a_k\} \subseteq V$ of random variables:

$$f(A) = H(X_A) = H(X_{a_1}, X_{a_2}, \ldots, X_{a_k}) = -\sum_{x_A} \Pr(x_A) \log \Pr(x_A)$$

- Entropy is submodular due to non-negativity of conditional mutual information. Given $A, B, C \subseteq V$,

$$\begin{aligned} &I(X_{A \setminus B}; X_{B \setminus A} | X_{A \cap B}) \\ &\quad = H(X_A) + H(X_B) - H(X_{A \cup B}) - H(X_{A \cap B}) \geq 0 \end{aligned} \tag{39}$$

- This was realized as early as 1954 (McGill) but it was not called submodularity then.

Gaussian entropy, and the log-determinant function

Definition (differential entropy $h(X)$)

$$h(X) = -\int_S f(x) \log f(x) dx \tag{40}$$

- When $x \sim \mathcal{N}(\mu, \Sigma)$ is multivariate Gaussian, the (differential) entropy of the r.v. X is given by

$$h(X) = \log \sqrt{|2\pi e \mathbf{\Sigma}|} = \log \sqrt{(2\pi e)^n |\mathbf{\Sigma}|} \tag{41}$$

Gaussian entropy, and the log-determinant function

Definition (differential entropy $h(X)$)

$$h(X) = -\int_S f(x) \log f(x) dx \tag{40}$$

- When $x \sim \mathcal{N}(\mu, \Sigma)$ is multivariate Gaussian, the (differential) entropy of the r.v. X is given by

$$h(X) = \log \sqrt{|2\pi e \mathbf{\Sigma}|} = \log \sqrt{(2\pi e)^n |\mathbf{\Sigma}|} \tag{41}$$

- For matrix M, define M_A as the principle submatrix of M, obtained from M by deleting rows and columns in the set $V \setminus A$.

Gaussian entropy, and the log-determinant function

Definition (differential entropy $h(X)$)

$$h(X) = -\int_S f(x) \log f(x) dx \tag{40}$$

- When $x \sim \mathcal{N}(\mu, \Sigma)$ is multivariate Gaussian, the (differential) entropy of the r.v. X is given by

$$h(X) = \log \sqrt{|2\pi e \mathbf{\Sigma}|} = \log \sqrt{(2\pi e)^n |\mathbf{\Sigma}|} \tag{41}$$

- For matrix M, define M_A as the principle submatrix of M, obtained from M by deleting rows and columns in the set $V \setminus A$.
- For $A \subseteq V$ and a constant γ, define

$$f(A) = h(X_A) = \log \sqrt{(2\pi e)^{|A|} |\mathbf{\Sigma}_A|} = \gamma |A| + \frac{1}{2} \log |\mathbf{\Sigma}_A| \tag{42}$$

Gaussian entropy, and the log-determinant function

Definition (differential entropy $h(X)$)

$$h(X) = -\int_S f(x) \log f(x) dx \tag{40}$$

- When $x \sim \mathcal{N}(\mu, \Sigma)$ is multivariate Gaussian, the (differential) entropy of the r.v. X is given by

$$h(X) = \log \sqrt{|2\pi e \mathbf{\Sigma}|} = \log \sqrt{(2\pi e)^n |\mathbf{\Sigma}|} \tag{41}$$

- For matrix M, define M_A as the principle submatrix of M, obtained from M by deleting rows and columns in the set $V \setminus A$.
- For $A \subseteq V$ and a constant γ, define

$$f(A) = h(X_A) = \log \sqrt{(2\pi e)^{|A|} |\mathbf{\Sigma}_A|} = \gamma |A| + \frac{1}{2} \log |\mathbf{\Sigma}_A| \tag{42}$$

- Submodularity of differential entropy follows from: $I(X_{A \setminus B}; X_{B \setminus A} | X_{A \cap B}) = h(X_A) + h(X_B) - h(X_{A \cup B}) - h(X_{A \cap B}) \geq 0$,

Gaussian entropy, and the log-determinant function

Definition (differential entropy $h(X)$)

$$h(X) = -\int_S f(x) \log f(x) dx \tag{40}$$

- When $x \sim \mathcal{N}(\mu, \Sigma)$ is multivariate Gaussian, the (differential) entropy of the r.v. X is given by

$$h(X) = \log \sqrt{|2\pi e \mathbf{\Sigma}|} = \log \sqrt{(2\pi e)^n |\mathbf{\Sigma}|} \tag{41}$$

- For matrix M, define M_A as the principle submatrix of M, obtained from M by deleting rows and columns in the set $V \setminus A$.
- For $A \subseteq V$ and a constant γ, define

$$f(A) = h(X_A) = \log \sqrt{(2\pi e)^{|A|} |\mathbf{\Sigma}_A|} = \gamma |A| + \frac{1}{2} \log |\mathbf{\Sigma}_A| \tag{42}$$

- Submodularity of differential entropy follows from: $I(X_{A \setminus B}; X_{B \setminus A} | X_{A \cap B}) = h(X_A) + h(X_B) - h(X_{A \cup B}) - h(X_{A \cap B}) \geq 0$,
- Hence, logdet function $f(A) = \log \det(\Sigma_A)$ is submodular.

Spectral Functions of a Matrix

- Given a positive definite matrix M, then $\log\det M = \mathrm{Tr}[\log M]$, where $\log M$ is the log of the matrix M (which is a matrix).

Spectral Functions of a Matrix

- Given a positive definite matrix M, then $\log\det M = \text{Tr}[\log M]$, where $\log M$ is the log of the matrix M (which is a matrix).
- Seen as a submodular function, we have that $f(A) = \text{Tr}[\log M_A]$ is submodular (again M_A is the principle submatrix of M)

Spectral Functions of a Matrix

- Given a positive definite matrix M, then $\log\det M = \text{Tr}[\log M]$, where $\log M$ is the log of the matrix M (which is a matrix).
- Seen as a submodular function, we have that $f(A) = \text{Tr}[\log M_A]$ is submodular (again M_A is the principle submatrix of M)
- Friedland and Gaubert (2010) generalization: if M is a Hermitian matrix (equal to its own conjugate transpose), and g is matrix-to-matrix function similar to a form of concavity (i.e., g is the "primitive" (like an integral) of a function that is operator antitone), then:

$$f(A) = \text{Tr}[g(M[A])] \tag{43}$$

  is a submodular function.

Spectral Functions of a Matrix

- Given a positive definite matrix M, then $\log\det M = \mathrm{Tr}[\log M]$, where $\log M$ is the log of the matrix M (which is a matrix).
- Seen as a submodular function, we have that $f(A) = \mathrm{Tr}[\log M_A]$ is submodular (again M_A is the principle submatrix of M)
- Friedland and Gaubert (2010) generalization: if M is a Hermitian matrix (equal to its own conjugate transpose), and g is matrix-to-matrix function similar to a form of concavity (i.e., g is the "primitive" (like an integral) of a function that is operator antitone), then:

$$f(A) = \mathrm{Tr}[g(M[A])] \tag{43}$$

  is a submodular function.
- This covers not only logdet, but also generalizes and shows submodularity of quantum entropy (used in quantum physics) with $g(x) = x \ln x$ and other functions such as $g(x) = x^p$ for $0 < p < 1$.

Are all polymatroid functions entropy functions?

Are all polymatroid functions entropy functions?

No, entropy functions must also satisfy the following:

Theorem (Yeung, 1998)

For any four discrete random variables $\{X, Y, Z, U\}$, *then*

$$I(X;Y) = I(X;Y|Z) = 0 \tag{44}$$

implies that

$$I(X;Y|Z,U) \leq I(Z;U|X,Y) + I(X;Y|U) \tag{45}$$

where $I(\cdot;\cdot|\cdot)$ *is the standard Shannon entropic mutual information function.*

- Not required for all polymatroid conditional mutual information functions $I_f(A;B|C) = f(A \cup C) + f(B \cup C) - f(C) - f(A \cup B \cup C)$.

Are all polymatroid functions entropy functions?

No, entropy functions must also satisfy the following:

Theorem (Yeung, 1998)

For any four discrete random variables $\{X, Y, Z, U\}$, *then*

$$I(X;Y) = I(X;Y|Z) = 0 \tag{44}$$

implies that

$$I(X;Y|Z,U) \leq I(Z;U|X,Y) + I(X;Y|U) \tag{45}$$

where $I(\cdot;\cdot|\cdot)$ *is the standard Shannon entropic mutual information function.*

- Not required for all polymatroid conditional mutual information functions $I_f(A;B|C) = f(A \cup C) + f(B \cup C) - f(C) - f(A \cup B \cup C)$.
- Open: Are all polymatroid functions spectral functions of a matrix?

Outline: Part 1

Other Submodular Properties

- We've defined submodular functions, and seen some of them.

Other Submodular Properties

- We've defined submodular functions, and seen some of them.
- Are there other properties, besides their ubiquity, that are useful?

Other Submodular Properties

- We've defined submodular functions, and seen some of them.
- Are there other properties, besides their ubiquity, that are useful?
- Also, as this tutorial ultimately will cover, they seem to be useful for a variety of problems in machine learning.

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:

$$\boxed{\min_{X \subseteq V} f(X)} \tag{46}$$

$$\boxed{\max_{X \subseteq V} f(X)} \tag{47}$$

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:

$$\boxed{\min_{X \subseteq V} f(X)} \tag{46}$$

$$\boxed{\max_{X \subseteq V} f(X)} \tag{47}$$

- 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:

$$\min_{X \subseteq V} f(X) \tag{46}$$

$$\max_{X \subseteq V} f(X) \tag{47}$$

- 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. (46) is polytime, and Eq. (47) 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 $\{S \subseteq V : \sum_{s \in S} w(s) \leq b\}$.
- 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:

$$\begin{array}{ll} \underset{S \subseteq V}{\text{maximize}} & f(S) \\ \text{subject to} & S \in \mathcal{S} \end{array} \quad (48)$$

$$\begin{array}{ll} \underset{S \subseteq V}{\text{minimize}} & f(S) \\ \text{subject to} & S \in \mathcal{S} \end{array} \quad (49)$$

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:

$$\begin{array}{ll} \underset{S \subseteq V}{\text{maximize}} & f(S) \\ \text{subject to} & S \in \mathcal{S} \end{array} \quad (48) \qquad \begin{array}{ll} \underset{S \subseteq V}{\text{minimize}} & f(S) \\ \text{subject to} & S \in \mathcal{S} \end{array} \quad (49)$$

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

Ex: Cardinality Constrained Max. of Polymatroid Functions

- Given an arbitrary polymatroid function f.

Ex: Cardinality Constrained Max. of Polymatroid Functions

- Given an arbitrary polymatroid function f.
- Given k, goal is: find $A^* \in \operatorname{argmax}\{f(A) : |A| \leq k\}$

Ex: Cardinality Constrained Max. of Polymatroid Functions

- Given an arbitrary polymatroid function f.
- Given k, goal is: find $A^* \in \operatorname{argmax}\{f(A) : |A| \leq k\}$
- w.l.o.g., we can find $A^* \in \operatorname{argmax}\{f(A) : |A| = k\}$

Ex: Cardinality Constrained Max. of Polymatroid Functions

- Given an arbitrary polymatroid function f.
- Given k, goal is: find $A^* \in \operatorname{argmax}\{f(A) : |A| \leq k\}$
- w.l.o.g., we can find $A^* \in \operatorname{argmax}\{f(A) : |A| = k\}$
- Nemhauser et. al. (1978) states that for normalized ($f(\emptyset) = 0$) monotone submodular functions (i.e., polymatroids) can be approximately maximized using a simple greedy algorithm.

Ex: Cardinality Constrained Max. of Polymatroid Functions

- Given an arbitrary polymatroid function f.
- Given k, goal is: find $A^* \in \operatorname{argmax}\{f(A) : |A| \leq k\}$
- w.l.o.g., we can find $A^* \in \operatorname{argmax}\{f(A) : |A| = k\}$
- Nemhauser et. al. (1978) states that for normalized ($f(\emptyset) = 0$) monotone submodular functions (i.e., polymatroids) can be approximately maximized using a simple greedy algorithm.

Algorithm 5: The Greedy Algorithm

Set $S_0 \leftarrow \emptyset$;
for $i \leftarrow 1 \dots |V|$ **do**
  Choose v_i as follows: $v_i \in \left\{\operatorname{argmax}_{v \in V \setminus S_i} f(\{v\}|S_{i-1})\right\}$;
  Set $S_i \leftarrow S_{i-1} \cup \{v_i\}$;

Ex: Cardinality Constrained Max. of Polymatroid Functions

- Given an arbitrary polymatroid function f.
- Given k, goal is: find $A^* \in \operatorname{argmax}\{f(A) : |A| \leq k\}$
- w.l.o.g., we can find $A^* \in \operatorname{argmax}\{f(A) : |A| = k\}$
- Nemhauser et. al. (1978) states that for normalized ($f(\emptyset) = 0$) monotone submodular functions (i.e., polymatroids) can be approximately maximized using a simple greedy algorithm.

Algorithm 6: The Greedy Algorithm

Set $S_0 \leftarrow \emptyset$;
for $i \leftarrow 1 \dots |V|$ **do**
  Choose v_i as follows: $v_i \in \left\{\operatorname{argmax}_{v \in V \setminus S_i} f(\{v\}|S_{i-1})\right\}$;
  Set $S_i \leftarrow S_{i-1} \cup \{v_i\}$;

- This yields a chain of sets $\emptyset = S_0 \subset S_1 \subset S_2 \subset \cdots \subset S_n = V$, with $|S_i| = i$, having very nice properties.

Greedy Algorithm for Card. Constrained Submodular Max

- This algorithm has the celebrated guarantee of $1 - 1/e$. That is

Greedy Algorithm for Card. Constrained Submodular Max

- This algorithm has the celebrated guarantee of $1 - 1/e$. That is

Theorem (Nemhauser et. al. (1978))

Given a polymatroid function $f : 2^V \to \mathbb{R}_+$, then the above greedy algorithm returns chain of sets $\{S_1, S_2, \dots, S_i\}$ such that for each i we have $f(S_i) \geq (1 - 1/e) \max_{|S| \leq i} f(S)$.

Greedy Algorithm for Card. Constrained Submodular Max

- This algorithm has the celebrated guarantee of $1 - 1/e$. That is

Theorem (Nemhauser et. al. (1978))

Given a polymatroid function $f : 2^V \to \mathbb{R}_+$, then the above greedy algorithm returns chain of sets $\{S_1, S_2, \dots, S_i\}$ such that for each i we have $f(S_i) \geq (1 - 1/e) \max_{|S| \leq i} f(S)$.

- To find $A^* \in \operatorname{argmax}\{f(A) : |A| \leq k\}$, we stop greedy at step k.

Greedy Algorithm for Card. Constrained Submodular Max

- This algorithm has the celebrated guarantee of $1 - 1/e$. That is

Theorem (Nemhauser et. al. (1978))

Given a polymatroid function $f : 2^V \to \mathbb{R}_+$, then the above greedy algorithm returns chain of sets $\{S_1, S_2, \dots, S_i\}$ such that for each i we have $f(S_i) \geq (1 - 1/e) \max_{|S| \leq i} f(S)$.

- To find $A^* \in \operatorname{argmax}\{f(A) : |A| \leq k\}$, we stop greedy at step k.
- The greedy chain also addresses the problem:

$$\text{minimize } |A| \text{ subject to } f(A) \geq \alpha \tag{50}$$

i.e., the submodular set cover problem (approximation factor $O(\log(\max_{s \in V} f(s)))$.

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

- At step $i < k$, greedy chooses v_i that maximizes $f(v|S_i)$.

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

- At step $i < k$, greedy chooses v_i that maximizes $f(v|S_i)$.
- Let S^* be optimal solution (of size k) and $\text{OPT} = f(S^*)$.

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

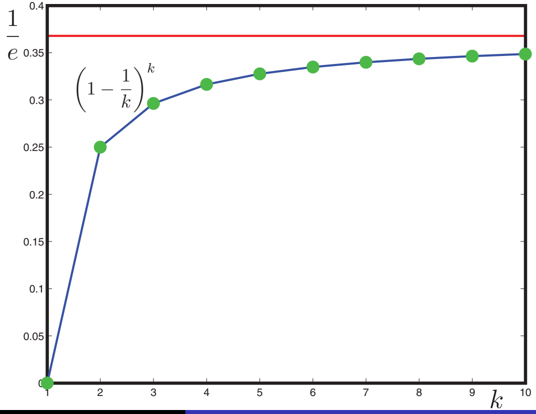
- At step $i < k$, greedy chooses v_i that maximizes $f(v|S_i)$.
- Let S^* be optimal solution (of size k) and $\text{OPT} = f(S^*)$. By submodularity, we can show:

$$\exists v \in V \setminus S_i : f(v|S_i) \geq \frac{1}{k}(\text{OPT} - f(S_i)) \tag{51}$$

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

- At step $i < k$, greedy chooses v_i that maximizes $f(v|S_i)$.
- Let S^* be optimal solution (of size k) and $\text{OPT} = f(S^*)$. By submodularity, we can show:

$$\exists v \in V \setminus S_i : f(v|S_i) \geq \frac{1}{k}(\text{OPT} - f(S_i)) \tag{51}$$



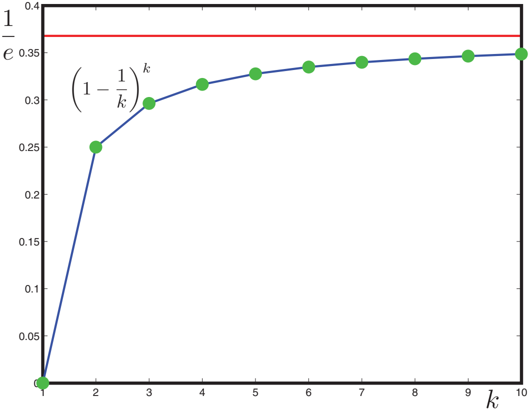
Equation (51) $\Rightarrow$:

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

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

- At step $i < k$, greedy chooses v_i that maximizes $f(v|S_i)$.
- Let S^* be optimal solution (of size k) and $\text{OPT} = f(S^*)$. By submodularity, we can show:

$$\exists v \in V \setminus S_i : f(v|S_i) \geq \frac{1}{k}(\text{OPT} - f(S_i)) \quad (51)$$



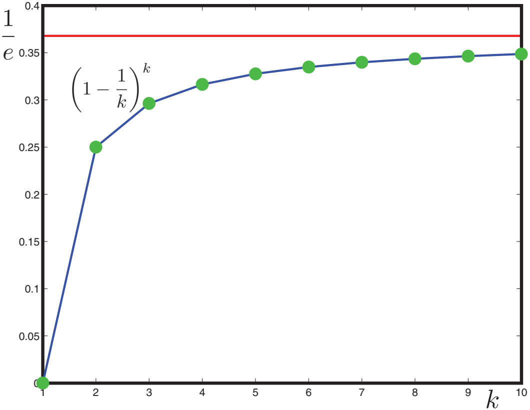
Equation (51) $\Rightarrow$:

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

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

- At step $i < k$, greedy chooses v_i that maximizes $f(v|S_i)$.
- Let S^* be optimal solution (of size k) and $\text{OPT} = f(S^*)$. By submodularity, we can show:

$$\exists v \in V \setminus S_i : f(v|S_i) \geq \frac{1}{k}(\text{OPT} - f(S_i)) \tag{51}$$



Equation (51) $\Rightarrow$:

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

Externally Non-Submodular/Internally Submodular

- Even when $h : 2^V \to \mathbb{R}$ is not submodular, submodularity can help.

Externally Non-Submodular/Internally Submodular

- Even when $h : 2^V \to \mathbb{R}$ is not submodular, submodularity can help.
- Example: difference of submodular (DS) functions
  $h(X) = f(X) - g(X)$ for f and g submodular (Narasimhan & B., Iyer & B.)

Externally Non-Submodular/Internally Submodular

- Even when $h : 2^V \to \mathbb{R}$ is not submodular, submodularity can help.
- Example: difference of submodular (DS) functions $h(X) = f(X) - g(X)$ for f and g submodular (Narasimhan & B., Iyer & B.)
- Any set function is a DS function. When naturally expressible as a DS function, there are good heuristics for optimization (minimization or maximization) that often work well in practice.

Externally Non-Submodular/Internally Submodular

- Even when $h : 2^V \to \mathbb{R}$ is not submodular, submodularity can help.
- Example: difference of submodular (DS) functions
  $h(X) = f(X) - g(X)$ for f and g submodular (Narasimhan & B., Iyer & B.)
- Any set function is a DS function. When naturally expressible as a DS function, there are good heuristics for optimization (minimization or maximization) that often work well in practice.
- Cooperative cut functions (Jegelka & B.):

$$f(X) = g(\{(u, v) \in E : u \in X, v \in V \setminus X\}) \tag{52}$$

where $g : 2^E \to \mathbb{R}_+$ is a submodular function defined on subsets of edges of the graph.

Externally Non-Submodular/Internally Submodular

- Even when $h : 2^V \to \mathbb{R}$ is not submodular, submodularity can help.
- Example: difference of submodular (DS) functions $h(X) = f(X) - g(X)$ for f and g submodular (Narasimhan & B., Iyer & B.)
- Any set function is a DS function. When naturally expressible as a DS function, there are good heuristics for optimization (minimization or maximization) that often work well in practice.
- Cooperative cut functions (Jegelka & B.):

$$f(X) = g(\{(u,v) \in E : u \in X, v \in V \setminus X\}) \tag{52}$$

  where $g : 2^E \to \mathbb{R}_+$ is a submodular function defined on subsets of edges of the graph.
- Frankenstein Cuts (Kawahara, Iyer, & B): $h(X) = f(X) + g(X)$ where f is submodular and g is a supermodular tree (submodular optimization for f, dynamic programming for g).

Outline: Part 2

5. Submodular Applications in Machine Learning
   - Where is submodularity useful?
6. As a model of diversity, coverage, span, or information
7. As a model of cooperative costs, complexity, roughness, and irregularity
8. As a Parameter for an ML algorithm
9. Itself, as a target for learning
10. Surrogates for optimization and analysis
11. Reading
    - Refs

Submodularity's utility in ML

- A model of a physical process:

Submodularity's utility in ML

- 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.

Submodularity's utility in ML

- 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:

Submodularity's utility in ML

- 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,

Submodularity's utility in ML

- 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.

Submodularity's utility in ML

- 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).

Submodularity's utility in ML

- 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.

Submodularity's utility in ML

- 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.
- A surrogate or relaxation strategy for optimization or analysis

Submodularity's utility in ML

- 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.
- A surrogate or relaxation strategy for optimization or analysis
  - An alternate to factorization, decomposition, or sum-product based simplification (as one typically finds in a graphical model). I.e., a means towards tractable surrogates for graphical models.

Submodularity's utility in ML

- 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.
- A surrogate or relaxation strategy for optimization or analysis
  - An alternate to factorization, decomposition, or sum-product based simplification (as one typically finds in a graphical model). I.e., a means towards tractable surrogates for graphical models.
  - Also, we can "relax" a problem to a submodular one where it can be efficiently solved and offer a bounded quality solution.

Submodularity's utility in ML

- 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.
- A surrogate or relaxation strategy for optimization or analysis
  - An alternate to factorization, decomposition, or sum-product based simplification (as one typically finds in a graphical model). I.e., a means towards tractable surrogates for graphical models.
  - Also, we can "relax" a problem to a submodular one where it can be efficiently solved and offer a bounded quality solution.
  - Non-submodular problems can be analyzed via submodularity.

Outline: Part 2

5. Submodular Applications in Machine Learning
   - Where is submodularity useful?
6. As a model of diversity, coverage, span, or information
7. As a model of cooperative costs, complexity, roughness, and irregularity
8. As a Parameter for an ML algorithm
9. Itself, as a target for learning
10. Surrogates for optimization and analysis
11. Reading
    - Refs

Extractive Document Summarization

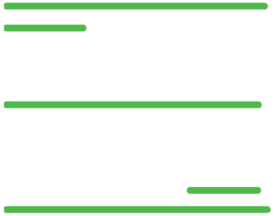
- The figure below represents the sentences of a document

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.

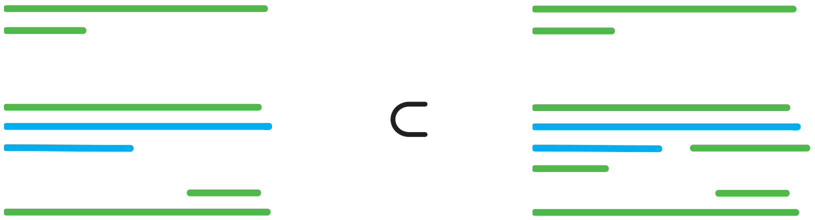
Extractive Document Summarization

- 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.

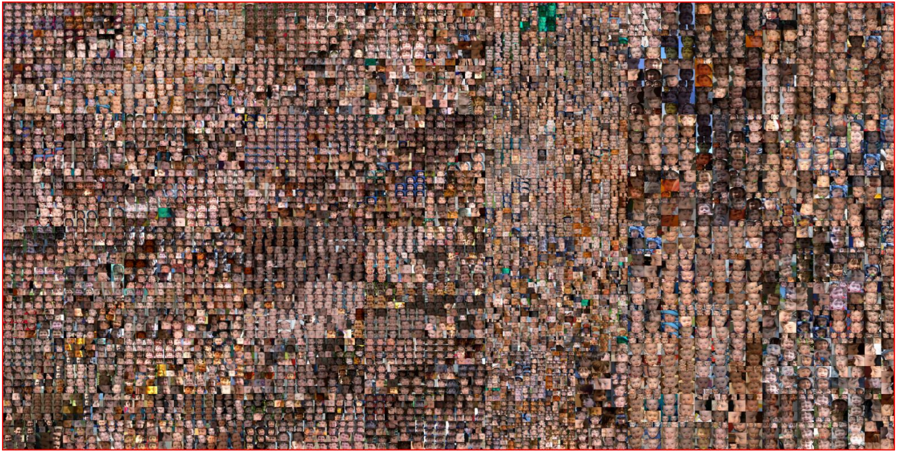


Image Summarization

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 (Tschiatschek, Iyer, & B, NIPS 2014).

Variable Selection in Classification/Regression

- Let Y be a random variable we wish to accurately predict based on at most n observed measurement variables $(X_1, X_2, \ldots, X_n) = X_V$ in a presumed probability model $\Pr(Y, X_1, X_2, \ldots, X_n)$.

Variable Selection in Classification/Regression

- Let Y be a random variable we wish to accurately predict based on at most n observed measurement variables $(X_1, X_2, \ldots, X_n) = X_V$ in a presumed probability model $\Pr(Y, X_1, X_2, \ldots, X_n)$.
- Too costly to use all variables. Goal is to choose a good subset $A \subseteq V$ of variables within budget $|A| \leq k$.

Variable Selection in Classification/Regression

- Let Y be a random variable we wish to accurately predict based on at most n observed measurement variables $(X_1, X_2, \ldots, X_n) = X_V$ in a presumed probability model $\Pr(Y, X_1, X_2, \ldots, X_n)$.
- Too costly to use all variables. Goal is to choose a good subset $A \subseteq V$ of variables within budget $|A| \leq k$.
- The mutual information function $f(A) = I(Y; X_A)$ measures how well variables A can predicting Y (entropy reduction, reduction of uncertainty of Y).

Variable Selection in Classification/Regression

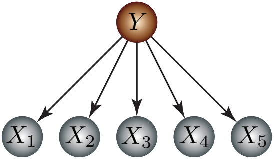
- Let Y be a random variable we wish to accurately predict based on at most n observed measurement variables $(X_1, X_2, \ldots, X_n) = X_V$ in a presumed probability model $\Pr(Y, X_1, X_2, \ldots, X_n)$.
- Too costly to use all variables. Goal is to choose a good subset $A \subseteq V$ of variables within budget $|A| \leq k$.
- The mutual information function $f(A) = I(Y; X_A)$ measures how well variables A can predicting Y (entropy reduction, reduction of uncertainty of Y).
- The mutual information function $f(A) = I(Y; X_A)$ is defined as:

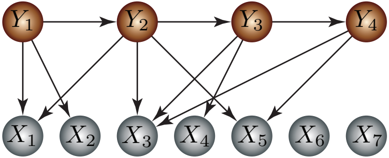
$$I(Y; X_A) = \sum_{y, x_A} \Pr(y, x_A) \log \frac{\Pr(y, x_A)}{\Pr(y)\Pr(x_A)} = H(Y) - H(Y|X_A) \quad (53)$$

$$= H(X_A) - H(X_A|Y) = H(X_A) + H(Y) - H(X_A, Y) \quad (54)$$

Feature Selection in Pattern Classification: Naïve Bayes

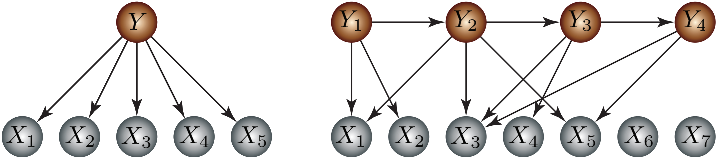
- Naïve Bayes property: $X_A \perp\!\!\!\perp X_B | Y$ for all A, B.





Feature Selection in Pattern Classification: Naïve Bayes

- Naïve Bayes property: $X_A \perp\!\!\!\perp X_B | Y$ for all A, B.



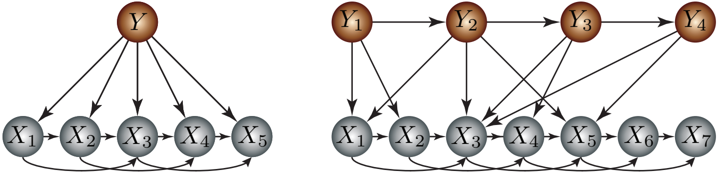
- When $X_A \perp\!\!\!\perp X_B | Y$ for all A, B (the Naïve Bayes assumption holds), then

$$f(A) = I(Y; X_A) = H(X_A) - H(X_A|Y) = H(X_A) - \sum_{a \in A} H(X_a|Y) \tag{55}$$

is submodular (submodular minus modular).

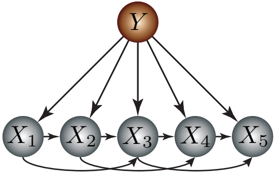
Variable Selection in Pattern Classification

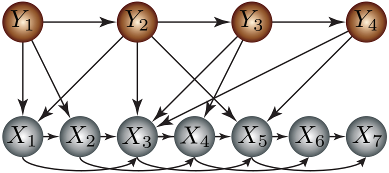
- Naïve Bayes property fails:



Variable Selection in Pattern Classification

- Naïve Bayes property fails:





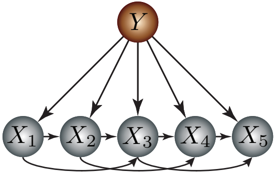
- $f(A)$ naturally expressed as a difference of two submodular functions

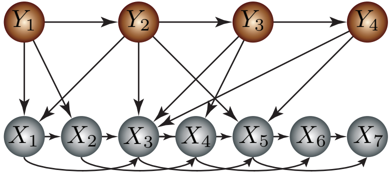
$$f(A) = I(Y; X_A) = H(X_A) - H(X_A|Y), \tag{56}$$

which is a DS (difference of submodular) function.

Variable Selection in Pattern Classification

- Naïve Bayes property fails:





- $f(A)$ naturally expressed as a difference of two submodular functions

$$f(A) = I(Y; X_A) = H(X_A) - H(X_A|Y), \tag{56}$$

which is a DS (difference of submodular) function.

- Alternatively, when Naïve Bayes assumption is false, we can make a submodular approximation (Peng-2005). E.g., functions of the form:

$$f(A) = \sum_{a \in A} I(X_a; Y) - \lambda \sum_{a, a' \in A} I(X_a; X_{a'}|Y) \tag{57}$$

where $\lambda \geq 0$ is a tradeoff constant.

Variable Selection: Linear Regression Case

- Here Z is continuous and predictor is linear $\tilde{Z}_A = \sum_{i \in A} \alpha_i X_i$.

Variable Selection: Linear Regression Case

- Here Z is continuous and predictor is linear $\tilde{Z}_A = \sum_{i \in A} \alpha_i X_i$.
- Error measure is the residual variance

$$R^2_{Z,A} = \frac{\text{Var}(Z) - E[(Z - \tilde{Z}_A)^2]}{\text{Var}(Z)} \tag{58}$$

Variable Selection: Linear Regression Case

- Here Z is continuous and predictor is linear $\tilde{Z}_A = \sum_{i \in A} \alpha_i X_i$.
- Error measure is the residual variance

$$R^2_{Z,A} = \frac{\mathrm{Var}(Z) - E[(Z - \tilde{Z}_A)^2]}{\mathrm{Var}(Z)} \tag{58}$$

- $R^2_{Z,A}$'s minimizing parameters, for a given A, can be easily computed ($R^2_{Z,A} = b_A{}^\intercal (C_A^{-1})^\intercal b_A$ when $\mathrm{Var} Z = 1$, where $b_i = \mathrm{Cov}(Z, X_i)$ and $C = E[(X - E[X])^\intercal (X - E[X])]$ is the covariance matrix).

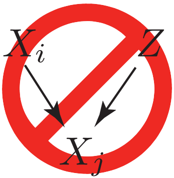
Variable Selection: Linear Regression Case

- Here Z is continuous and predictor is linear $\tilde{Z}_A = \sum_{i \in A} \alpha_i X_i$.
- Error measure is the residual variance

$$R^2_{Z,A} = \frac{\text{Var}(Z) - E[(Z - \tilde{Z}_A)^2]}{\text{Var}(Z)} \tag{58}$$

- $R^2_{Z,A}$'s minimizing parameters, for a given A, can be easily computed ($R^2_{Z,A} = {b_A}^\intercal (C_A^{-1})^\intercal b_A$ when $\text{Var} Z = 1$, where $b_i = \text{Cov}(Z, X_i)$ and $C = E[(X - E[X])^\intercal (X - E[X])]$ is the covariance matrix).
- When there are no "suppressor" variables (essentially, no v-structures that converge on X_j with parents X_i and Z), then

$$f(A) = R^2_{Z,A} = {b_A}^\intercal (C_A^{-1})^\intercal b_A \tag{59}$$



is a polymatroid function (so the greedy algorithm gives the $1 - 1/e$ guarantee). (Das&Kempe).

Data Subset Selection

- Suppose we are given a data set $\mathcal{D} = \{x_i\}_{i=1}^n$ of n data items $V = \{v_1, v_2, \ldots, v_n\}$ and we wish to choose a subset $A \subset V$ of items that is good in some way.

Data Subset Selection

- Suppose we are given a data set $\mathcal{D} = \{x_i\}_{i=1}^n$ of n data items $V = \{v_1, v_2, \dots, v_n\}$ and we wish to choose a subset $A \subset V$ of items that is good in some way.
- Suppose moreover each data item $v \in V$ is described by a vector of non-negative scores for a set U of "features" (or properties, or characteristics, etc.) of each data item.

Data Subset Selection

- Suppose we are given a data set $\mathcal{D} = \{x_i\}_{i=1}^n$ of n data items $V = \{v_1, v_2, \ldots, v_n\}$ and we wish to choose a subset $A \subset V$ of items that is good in some way.
- Suppose moreover each data item $v \in V$ is described by a vector of non-negative scores for a set U of "features" (or properties, or characteristics, etc.) of each data item.
- That is, for $u \in U$ and $v \in V$, let $m_u(v)$ represent the "degree of u-ness" possessed by data item v. Then $m_u \in \mathbb{R}_+^V$ for all $u \in U$.

Data Subset Selection

- Suppose we are given a data set $\mathcal{D} = \{x_i\}_{i=1}^n$ of n data items $V = \{v_1, v_2, \ldots, v_n\}$ and we wish to choose a subset $A \subset V$ of items that is good in some way.
- Suppose moreover each data item $v \in V$ is described by a vector of non-negative scores for a set U of "features" (or properties, or characteristics, etc.) of each data item.
- That is, for $u \in U$ and $v \in V$, let $m_u(v)$ represent the "degree of u-ness" possessed by data item v. Then $m_u \in \mathbb{R}_+^V$ for all $u \in U$.
- Example: U could be a set of colors, and for an image $v \in V$, $m_u(v)$ could represent the number of pixels that are of color u.

Data Subset Selection

- Suppose we are given a data set $\mathcal{D} = \{x_i\}_{i=1}^n$ of n data items $V = \{v_1, v_2, \dots, v_n\}$ and we wish to choose a subset $A \subset V$ of items that is good in some way.
- Suppose moreover each data item $v \in V$ is described by a vector of non-negative scores for a set U of "features" (or properties, or characteristics, etc.) of each data item.
- That is, for $u \in U$ and $v \in V$, let $m_u(v)$ represent the "degree of u-ness" possessed by data item v. Then $m_u \in \mathbb{R}_+^V$ for all $u \in U$.
- Example: U could be a set of colors, and for an image $v \in V$, $m_u(v)$ could represent the number of pixels that are of color u.
- Example: U might be a set of textual features (e.g., ngrams), and $m_u(v)$ is the number of ngrams of type u in sentence v. E.g., if a document consists of the sentence

$$v = \text{“Whenever I go to New York City, I visit the New York City museum.”}$$

then $m_{\text{'the'}}(v) = 1$ while $m_{\text{'New York City'}}(v) = 2$.

Data Subset Selection

- For $X \subseteq V$, define $m_u(X) = \sum_{x \in X} m_u(x)$, so $m_u(X)$ is a modular function representing the "degree of u-ness" in subset X.

Data Subset Selection

- For $X \subseteq V$, define $m_u(X) = \sum_{x \in X} m_u(x)$, so $m_u(X)$ is a modular function representing the "degree of u-ness" in subset X.
- Since $m_u(X)$ is modular, it does not have a diminishing returns property. I.e., as we add to X, the degree of u-ness grows additively.

Data Subset Selection

- For $X \subseteq V$, define $m_u(X) = \sum_{x \in X} m_u(x)$, so $m_u(X)$ is a modular function representing the "degree of u-ness" in subset X.
- Since $m_u(X)$ is modular, it does not have a diminishing returns property. I.e., as we add to X, the degree of u-ness grows additively.
- With g non-decreasing concave, $g(m_u(X))$ grows subadditively (if we add v to a context A with less u-ness, the u-ness benefit is more than if we add v to a context $B \supseteq A$ having more u-ness).

Data Subset Selection

- For $X \subseteq V$, define $m_u(X) = \sum_{x \in X} m_u(x)$, so $m_u(X)$ is a modular function representing the "degree of u-ness" in subset X.
- Since $m_u(X)$ is modular, it does not have a diminishing returns property. I.e., as we add to X, the degree of u-ness grows additively.
- With g non-decreasing concave, $g(m_u(X))$ grows subadditively (if we add v to a context A with less u-ness, the u-ness benefit is more than if we add v to a context $B \supseteq A$ having more u-ness). That is

$$g(m_u(A+v)) - g(m_u(A)) \geq g(m_u(B+v)) - g(m_u(B)) \tag{60}$$

Data Subset Selection

- For $X \subseteq V$, define $m_u(X) = \sum_{x \in X} m_u(x)$, so $m_u(X)$ is a modular function representing the "degree of u-ness" in subset X.
- Since $m_u(X)$ is modular, it does not have a diminishing returns property. I.e., as we add to X, the degree of u-ness grows additively.
- With g non-decreasing concave, $g(m_u(X))$ grows subadditively (if we add v to a context A with less u-ness, the u-ness benefit is more than if we add v to a context $B \supseteq A$ having more u-ness). That is

$$g(m_u(A + v)) - g(m_u(A)) \geq g(m_u(B + v)) - g(m_u(B)) \tag{60}$$

- Consider the following class of feature functions $f : 2^V \to \mathbb{R}_+$

$$f(X) = \sum_{u \in U} \alpha_u g_u(m_u(X)) \tag{61}$$

where g_u is a non-decreasing concave, and $\alpha_u \geq 0$ is a feature importance weight. Thus, f is submodular.

Data Subset Selection

- For $X \subseteq V$, define $m_u(X) = \sum_{x \in X} m_u(x)$, so $m_u(X)$ is a modular function representing the "degree of u-ness" in subset X.
- Since $m_u(X)$ is modular, it does not have a diminishing returns property. I.e., as we add to X, the degree of u-ness grows additively.
- With g non-decreasing concave, $g(m_u(X))$ grows subadditively (if we add v to a context A with less u-ness, the u-ness benefit is more than if we add v to a context $B \supseteq A$ having more u-ness). That is

$$g(m_u(A+v)) - g(m_u(A)) \geq g(m_u(B+v)) - g(m_u(B)) \tag{60}$$

- Consider the following class of feature functions $f : 2^V \to \mathbb{R}_+$

$$f(X) = \sum_{u \in U} \alpha_u g_u(m_u(X)) \tag{61}$$

  where g_u is a non-decreasing concave, and $\alpha_u \geq 0$ is a feature importance weight. Thus, f is submodular.
- $f(X)$ measures X's ability to represent set of features U as measured by $m_u(X)$, with diminishing returns function g, and importance weights α_u.

Data Subset Selection, KL-divergence

- Let $p = \{p_u\}_{u \in U}$ be a desired probability distribution over features (i.e., $\sum_u p_u = 1$ and $p_u \geq 0$ for all $u \in U$).

Data Subset Selection, KL-divergence

- Let $p = \{p_u\}_{u \in U}$ be a desired probability distribution over features (i.e., $\sum_u p_u = 1$ and $p_u \geq 0$ for all $u \in U$).
- Next, normalize the modular weights for each feature:

$$\bar{m}_u(X) = \frac{m_u(X)}{\sum_{u' \in U} m_{u'}(X)} = \frac{m_u(X)}{m(X)} \tag{62}$$

where $m(X) \triangleq \sum_{u' \in U} m_{u'}(X)$.

Data Subset Selection, KL-divergence

- Let $p = \{p_u\}_{u \in U}$ be a desired probability distribution over features (i.e., $\sum_u p_u = 1$ and $p_u \geq 0$ for all $u \in U$).
- Next, normalize the modular weights for each feature:

$$\bar{m}_u(X) = \frac{m_u(X)}{\sum_{u' \in U} m_{u'}(X)} = \frac{m_u(X)}{m(X)} \tag{62}$$

  where $m(X) \triangleq \sum_{u' \in U} m_{u'}(X)$.
- Then $\bar{m}_u(X)$ can also be seen as a distribution over features since $\bar{m}_u(X) \geq 0$ and $\sum_u \bar{m}_u(X) = 1$ for any $X \subseteq V$.

Data Subset Selection, KL-divergence

- Let $p = \{p_u\}_{u \in U}$ be a desired probability distribution over features (i.e., $\sum_u p_u = 1$ and $p_u \geq 0$ for all $u \in U$).
- Next, normalize the modular weights for each feature:

$$\bar{m}_u(X) = \frac{m_u(X)}{\sum_{u' \in U} m_{u'}(X)} = \frac{m_u(X)}{m(X)} \tag{62}$$

where $m(X) \triangleq \sum_{u' \in U} m_{u'}(X)$.
- Then $\bar{m}_u(X)$ can also be seen as a distribution over features since $\bar{m}_u(X) \geq 0$ and $\sum_u \bar{m}_u(X) = 1$ for any $X \subseteq V$.
- Consider the KL-divergence between these two distributions:

$$D(p||\{\bar{m}_u(X)\}_{u \in U}) = \sum_{u \in U} p_u \log p_u - \sum_{u \in U} p_u \log(\bar{m}_u(X)) \tag{63}$$

$$= \sum_{u \in U} p_u \log p_u - \sum_{u \in U} p_u \log(m_u(X)) + \log(m(X))$$

$$= -H(p) + \log m(X) - \sum_{u \in U} p_u \log(m_u(X)) \tag{64}$$

Data Subset Selection, KL-divergence

- The objective once again, treating entropy $H(p)$ as a constant,

$$D(p||\{\bar{m}_u(X)\}) = \text{const.} + \log m(X) - \sum_{u \in U} p_u \log(m_u(X)) \tag{65}$$

Data Subset Selection, KL-divergence

- The objective once again, treating entropy $H(p)$ as a constant,

$$D(p||\{\bar{m}_u(X)\}) = \text{const.} + \log m(X) - \sum_{u \in U} p_u \log(m_u(X)) \quad (65)$$

- But seen as a function of X, both $\log m(X)$ and $\sum_{u \in U} p_u \log m_u(X)$ are submodular functions.

Data Subset Selection, KL-divergence

- The objective once again, treating entropy $H(p)$ as a constant,

$$D(p||\{\bar{m}_u(X)\}) = \text{const.} + \log m(X) - \sum_{u \in U} p_u \log(m_u(X)) \quad (65)$$

- But seen as a function of X, both $\log m(X)$ and $\sum_{u \in U} p_u \log m_u(X)$ are submodular functions.
- Hence the KL-divergence, seen as a function of X, i.e., $f(X) = D(p||\{\bar{m}_u(X)\})$ is quite naturally represented as a difference of submodular functions.

Data Subset Selection, KL-divergence

- The objective once again, treating entropy $H(p)$ as a constant,

$$D(p||\{\bar{m}_u(X)\}) = \text{const.} + \log m(X) - \sum_{u \in U} p_u \log(m_u(X)) \quad (65)$$

- But seen as a function of X, both $\log m(X)$ and $\sum_{u \in U} p_u \log m_u(X)$ are submodular functions.
- Hence the KL-divergence, seen as a function of X, i.e., $f(X) = D(p||\{\bar{m}_u(X)\})$ is quite naturally represented as a difference of submodular functions.
- Alternatively, if we define (Shinohara, 2014)

$$g(X) \triangleq \log m(X) - D(p||\{\bar{m}_u(X)\}) = \sum_{u \in U} p_u \log(m_u(X)) \quad (66)$$

we have a submodular function g that represents a combination of its quantity of X via its features (i.e., $\log m(X)$) and its feature distribution closeness to some distribution p (i.e., $D(p||\{\bar{m}_u(X)\})$).

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 $0 < \alpha \leq 1$ (recall the submodular set cover problem)

Sensor Placement

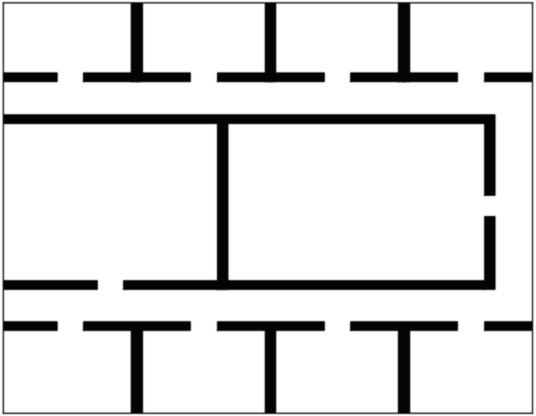
- Information gain applicable not only in pattern recognition, but in the sensor coverage problem as well, where Y is whatever question we wish to ask about an environment.
- Given an environment, there is a set V of candidate locations for placement of a sensor (e.g., temperature, gas, audio, video, 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 $0 < \alpha \leq 1$ (recall the submodular set cover problem)
- 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 $0 < \alpha \leq 1$ (recall the submodular set cover problem)
- 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.

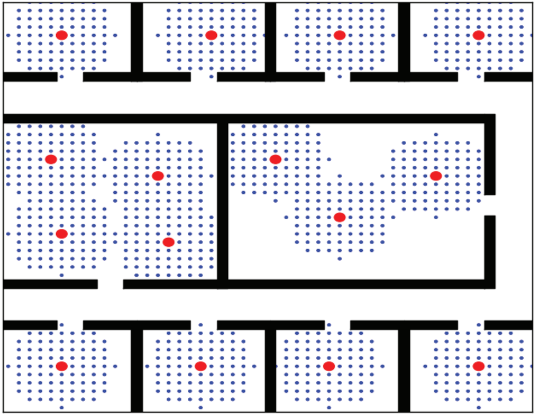
Sensor Placement within Buildings

- 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.



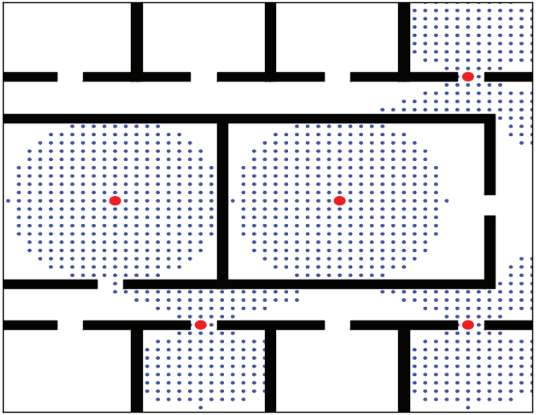
Sensor Placement within Buildings

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



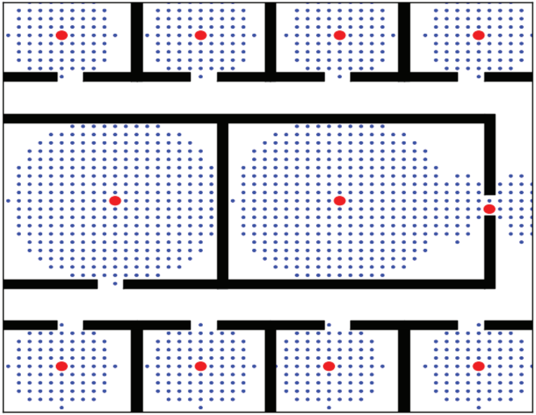
Sensor Placement within Buildings

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



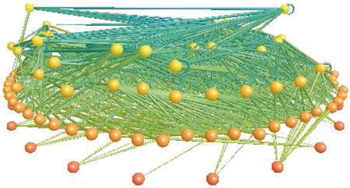
Sensor Placement within Buildings

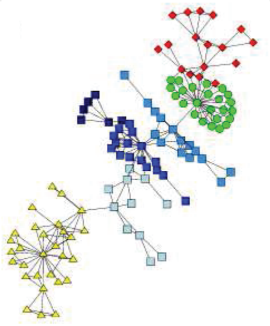
- Example sensor placement using mixed range sensors (located at red dots).

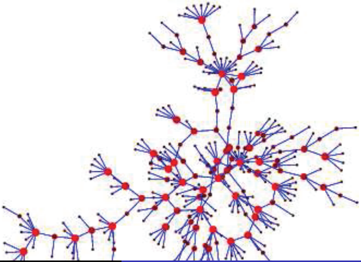


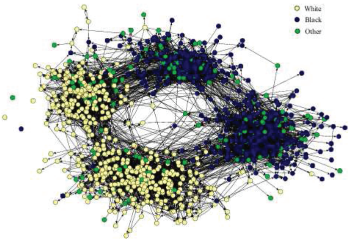
Social Networks

(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 set of individuals, how valuable is a given friend $v \in V$?

The value of a friend

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

The value of a friend

- Let V be a set of individuals, how valuable is a given friend $v \in V$?
- It depends on how many friends you have.
- Valuate a group of friends $S \subseteq V$ via set function $f(S)$.

The value of a friend

- Let V be a set of individuals, how valuable is a given friend $v \in V$?
- It depends on how many friends you have.
- Valuate a group of friends $S \subseteq V$ via set function $f(S)$.
- A submodular model: a friend becomes less marginally valuable as your set of friends grows.

The value of a friend

- Let V be a set of individuals, how valuable is a given friend $v \in V$?
- It depends on how many friends you have.
- Valuate a group of friends $S \subseteq V$ via set function $f(S)$.
- A submodular model: a friend becomes less marginally valuable as your set of friends grows.
- Supermodular model: a friend becomes more valuable the more friends you have ("I'd get by with a little help from my friends").

The value of a friend

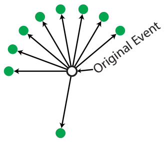
- Let V be a set of individuals, how valuable is a given friend $v \in V$?
- It depends on how many friends you have.
- Valuate a group of friends $S \subseteq V$ via set function $f(S)$.
- A submodular model: a friend becomes less marginally valuable as your set of friends grows.
- Supermodular model: a friend becomes 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).

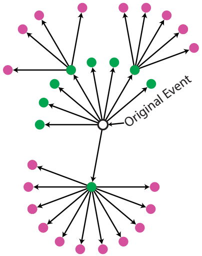
Information Cascades, Diffusion Networks

- How to model flow of information from source to the point it reaches users — information used in its common sense (like news events).



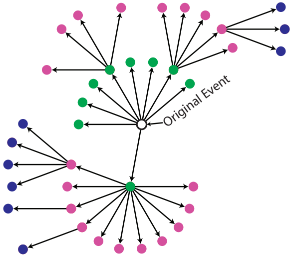
Information Cascades, Diffusion Networks

- How to model flow of information from source to the point it reaches users — information used in its common sense (like news events).



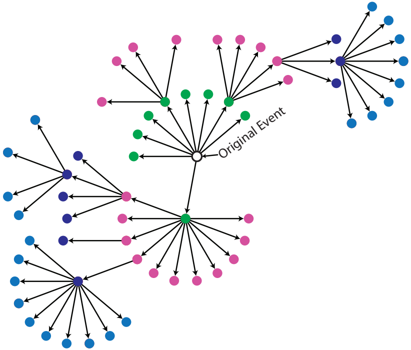
Information Cascades, Diffusion Networks

- How to model flow of information from source to the point it reaches users — information used in its common sense (like news events).



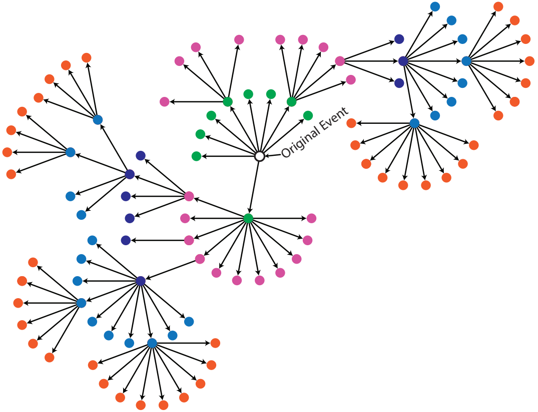
Information Cascades, Diffusion Networks

- How to model flow of information from source to the point it reaches users — information used in its common sense (like news events).



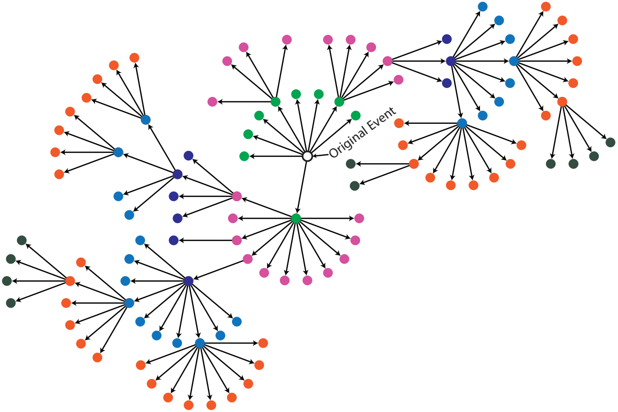
Information Cascades, Diffusion Networks

- How to model flow of information from source to the point it reaches users — information used in its common sense (like news events).



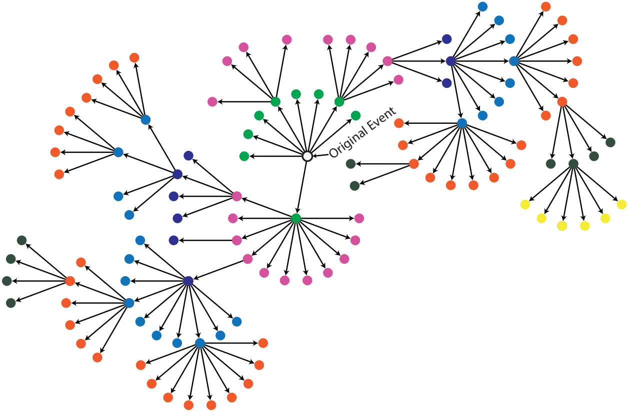
Information Cascades, Diffusion Networks

- How to model flow of information from source to the point it reaches users — information used in its common sense (like news events).



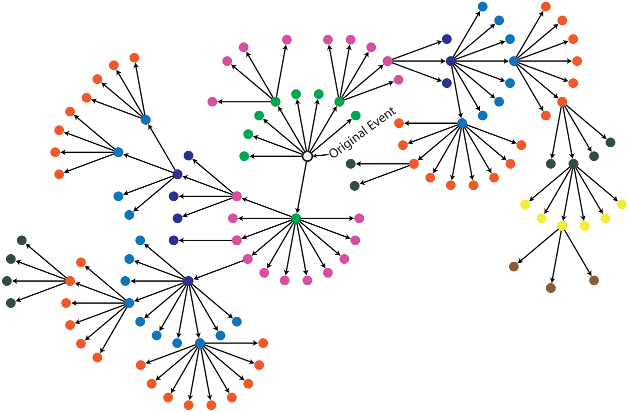
Information Cascades, Diffusion Networks

- How to model flow of information from source to the point it reaches users — information used in its common sense (like news events).



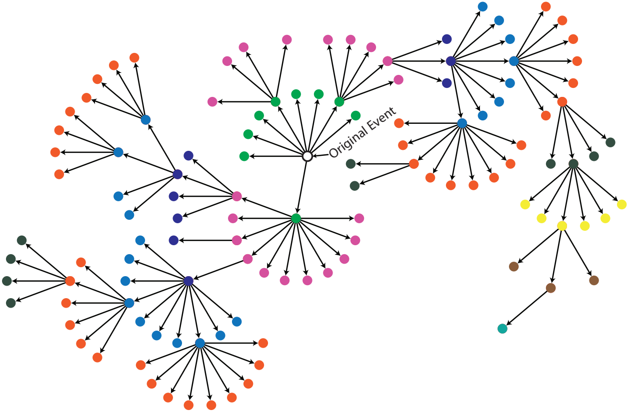
Information Cascades, Diffusion Networks

- How to model flow of information from source to the point it reaches users — information used in its common sense (like news events).



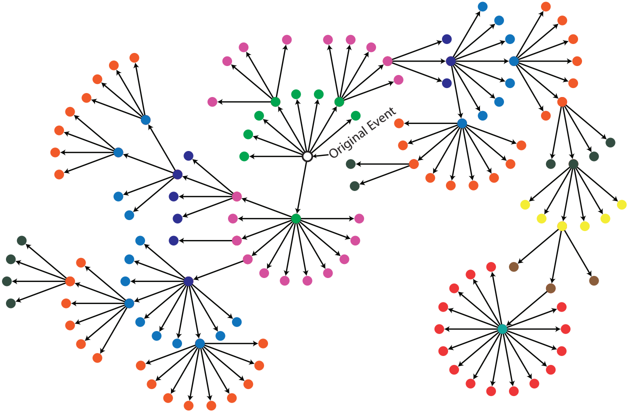
Information Cascades, Diffusion Networks

- How to model flow of information from source to the point it reaches users — information used in its common sense (like news events).



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).



- Goal: How to find the most influential sources, the ones that often set off cascades, which are like large "waves" of information flow?

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).

Graphical Model Structure Learning

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

$$p(x) = \frac{1}{Z} \exp(-E(x)) \tag{67}$$

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

Graphical Model Structure Learning

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

$$p(x) = \frac{1}{Z} \exp(-E(x)) \tag{67}$$

  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 $p : \{0,1\}^V \to [0,1]$:

$$p(x) = \frac{1}{Z} \exp(-E(x)) \tag{67}$$

  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:

$$E(x) = \sum_{c \in \mathcal{C}} E_c(x_c) \tag{68}$$

Graphical Model Structure Learning

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

$$p(x) = \frac{1}{Z} \exp(-E(x)) \tag{67}$$

  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:

$$E(x) = \sum_{c \in \mathcal{C}} E_c(x_c) \tag{68}$$

- 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 $p : \{0,1\}^V \to [0,1]$:

$$p(x) = \frac{1}{Z} \exp(-E(x)) \tag{67}$$

  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:

$$E(x) = \sum_{c \in \mathcal{C}} E_c(x_c) \tag{68}$$

- 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 $V \times V$.

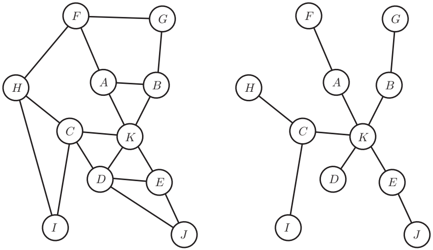
Graphical Models: Learning Tree Distributions

- 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:

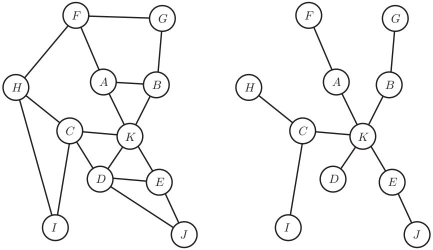
$$\begin{aligned} \underset{p_t \in \mathcal{F}(G,\mathcal{M})}{\text{minimize}} \quad & D(p||p_t) \\ \text{subject to} \quad & p_t \in \mathcal{F}(T, \mathcal{M}). \\ & T = (V, F) \text{ is a tree} \end{aligned}$$



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:

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

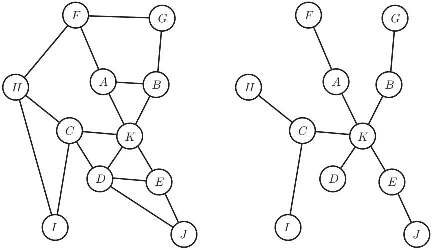


- Discrete problem: choose the optimal set of edges $A \subseteq E$ that constitute tree (i.e., find a spanning tree of G of best quality).

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:

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

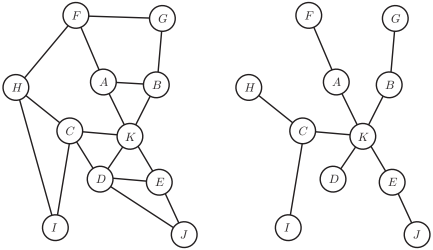


- Discrete problem: choose the optimal set of edges $A \subseteq E$ that constitute tree (i.e., find a spanning tree of G of best quality).
- Define $f : 2^E \to \mathbb{R}_+$ where f is a weighted cycle matroid rank function (a type of submodular function), with weights $w(e) = w(u, v) = I(X_u; X_v)$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

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



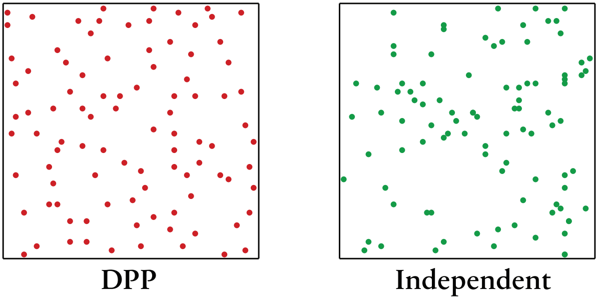
- Discrete problem: choose the optimal set of edges $A \subseteq E$ that constitute tree (i.e., find a spanning tree of G of best quality).
- Define $f : 2^E \to \mathbb{R}_+$ where f is a weighted cycle matroid rank function (a type of submodular function), with weights $w(e) = w(u, v) = I(X_u; X_v)$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

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)

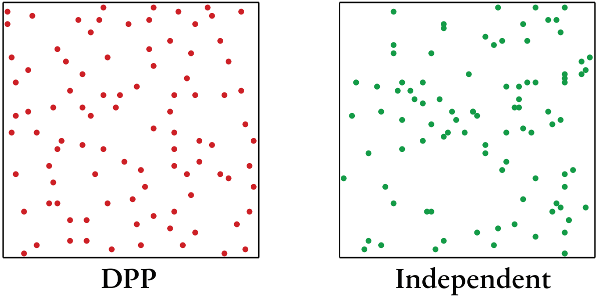
- Sometimes we wish not only to valuate subsets $A \subseteq V$ but to induce probability distributions over all subsets.
- We may wish to prefer samples where elements of A are diverse (i.e., given a sample A, for $a, b \in A$, we prefer a and b to be different).



(Kulesza, Gillenwater, & 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).

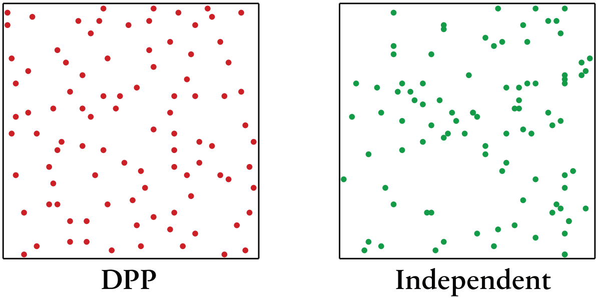


(Kulesza, Gillenwater, & Taskar, 2011)

- 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).



(Kulesza, Gillen-water, & 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 the $|X| \times |X|$ principle submatrix as we've seen before.

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 the $|X| \times |X|$ principle submatrix as we've seen before.
- A Determinantal Point Process (DPP) is a distribution of the form:

$$\Pr(\mathbf{X} = x) = \frac{|M_{X(x)}|}{|M + I|} = \exp\left(\log\left(\frac{|M_{X(x)}|}{|M + I|}\right)\right) \propto \det(M_{X(x)}) \quad (69)$$

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 the $|X| \times |X|$ principle submatrix as we've seen before.
- A Determinantal Point Process (DPP) is a distribution of the form:

$$\Pr(\mathbf{X} = x) = \frac{|M_{X(x)}|}{|M + I|} = \exp\left(\log\left(\frac{|M_{X(x)}|}{|M + I|}\right)\right) \propto \det(M_{X(x)}) \quad (69)$$

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

- Equivalently, defining K as $K = M(M+I)^{-1}$, we have:

$$\sum_{x \in \{0,1\}^V : x \geq y} \Pr(\mathbf{X} = x) = \Pr(\mathbf{X} \geq y) = \exp\left(\log\left(|K_{Y(y)}|\right)\right) \quad (70)$$

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 the $|X| \times |X|$ principle submatrix as we've seen before.
- A Determinantal Point Process (DPP) is a distribution of the form:

$$\Pr(\mathbf{X} = x) = \frac{|M_{X(x)}|}{|M + I|} = \exp\left(\log\left(\frac{|M_{X(x)}|}{|M + I|}\right)\right) \propto \det(M_{X(x)}) \quad (69)$$

  where I is $n \times n$ identity matrix, and $\mathbf{X} \in \{0,1\}^V$ is a random vector.
- Equivalently, defining K as $K = M(M + I)^{-1}$, we have:

$$\sum_{x \in \{0,1\}^V : x \geq y} \Pr(\mathbf{X} = x) = \Pr(\mathbf{X} \geq y) = \exp\left(\log\left(|K_{Y(y)}|\right)\right) \quad (70)$$

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

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 the $|X| \times |X|$ principle submatrix as we've seen before.
- A Determinantal Point Process (DPP) is a distribution of the form:

$$\Pr(\mathbf{X} = x) = \frac{|M_{X(x)}|}{|M + I|} = \exp\left(\log\left(\frac{|M_{X(x)}|}{|M + I|}\right)\right) \propto \det(M_{X(x)}) \quad (69)$$

  where I is $n \times n$ identity matrix, and $\mathbf{X} \in \{0,1\}^V$ is a random vector.
- Equivalently, defining K as $K = M(M+I)^{-1}$, we have:

$$\sum_{x \in \{0,1\}^V : x \geq y} \Pr(\mathbf{X} = x) = \Pr(\mathbf{X} \geq y) = \exp\left(\log\left(|K_{Y(y)}|\right)\right) \quad (70)$$

- Given positive definite matrix M, function $f : 2^V \to \mathbb{R}$ with $f(A) = \log|M_A|$ (the logdet function) is submodular.
- Therefore, a DPP is a log-submodular probability distribution.

Outline: Part 2

5. Submodular Applications in Machine Learning
   - Where is submodularity useful?
6. As a model of diversity, coverage, span, or information
7. As a model of cooperative costs, complexity, roughness, and irregularity
8. As a Parameter for an ML algorithm
9. Itself, as a target for learning
10. Surrogates for optimization and analysis
11. Reading
    - Refs

Graphical Models and fast MAP Inference

- Given distribution that factors w.r.t. a graph:

$$p(x) = \frac{1}{Z} \exp(-E(x)) \tag{71}$$

where $E(x) = \sum_{c \in \mathcal{C}} E_c(x_c)$ and $\mathcal{C}$ are cliques of graph $G = (V, \mathcal{E})$.

Graphical Models and fast MAP Inference

- Given distribution that factors w.r.t. a graph:

$$p(x) = \frac{1}{Z} \exp(-E(x)) \tag{71}$$

where $E(x) = \sum_{c \in \mathcal{C}} E_c(x_c)$ and $\mathcal{C}$ are cliques of graph $G = (V, \mathcal{E})$.

- MAP inference problem is important in ML: compute

$$x^* \in \operatorname*{argmax}_{x \in \{0,1\}^V} p(x) \tag{72}$$

Graphical Models and fast MAP Inference

- Given distribution that factors w.r.t. a graph:

$$p(x) = \frac{1}{Z} \exp(-E(x)) \tag{71}$$

where $E(x) = \sum_{c \in \mathcal{C}} E_c(x_c)$ and $\mathcal{C}$ are cliques of graph $G = (V, \mathcal{E})$.

- MAP inference problem is important in ML: compute

$$x^* \in \operatorname*{argmax}_{x \in \{0,1\}^V} p(x) \tag{72}$$

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

Graphical Models and fast MAP Inference

- Given distribution that factors w.r.t. a graph:

$$p(x) = \frac{1}{Z} \exp(-E(x)) \tag{71}$$

where $E(x) = \sum_{c \in \mathcal{C}} E_c(x_c)$ and $\mathcal{C}$ are cliques of graph $G = (V, \mathcal{E})$.

- MAP inference problem is important in ML: compute

$$x^* \in \operatorname*{argmax}_{x \in \{0,1\}^V} p(x) \tag{72}$$

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

Graphical Models and fast MAP Inference

- Given distribution that factors w.r.t. a graph:

$$p(x) = \frac{1}{Z}\exp(-E(x)) \tag{71}$$

where $E(x) = \sum_{c \in \mathcal{C}} E_c(x_c)$ and $\mathcal{C}$ are cliques of graph $G = (V, \mathcal{E})$.

- MAP inference problem is important in ML: compute

$$x^* \in \operatorname*{argmax}_{x \in \{0,1\}^V} p(x) \tag{72}$$

- Easy when G a tree, exponential in k (tree-width of G) in general.
- Even worse, NP-hard to find the tree-width.
- Tree-width can be large even when degree is small (e.g., regular grid graphs have low-degree but $\Omega(\sqrt{n})$ tree-width).

Graphical Models and fast MAP Inference

- Given distribution that factors w.r.t. a graph:

$$p(x) = \frac{1}{Z}\exp(-E(x)) \tag{71}$$

where $E(x) = \sum_{c \in \mathcal{C}} E_c(x_c)$ and $\mathcal{C}$ are cliques of graph $G = (V, \mathcal{E})$.

- MAP inference problem is important in ML: compute

$$x^* \in \operatorname*{argmax}_{x \in \{0,1\}^V} p(x) \tag{72}$$

- Easy when G a tree, exponential in k (tree-width of G) in general.
- Even worse, NP-hard to find the tree-width.
- Tree-width can be large even when degree is small (e.g., regular grid graphs have low-degree but $\Omega(\sqrt{n})$ tree-width).
- Many approximate inference strategies utilize additional factorization assumptions (e.g., mean-field, variational inference, expectation propagation, etc).

Graphical Models and fast MAP Inference

- Given distribution that factors w.r.t. a graph:

$$p(x) = \frac{1}{Z}\exp(-E(x)) \tag{71}$$

  where $E(x) = \sum_{c \in \mathcal{C}} E_c(x_c)$ and $\mathcal{C}$ are cliques of graph $G = (V, \mathcal{E})$.
- MAP inference problem is important in ML: compute

$$x^* \in \operatorname*{argmax}_{x \in \{0,1\}^V} p(x) \tag{72}$$

- Easy when G a tree, exponential in k (tree-width of G) in general.
- Even worse, NP-hard to find the tree-width.
- Tree-width can be large even when degree is small (e.g., regular grid graphs have low-degree but $\Omega(\sqrt{n})$ tree-width).
- Many approximate inference strategies utilize additional factorization assumptions (e.g., mean-field, variational inference, expectation propagation, etc).
- Can we do exact MAP inference in polynomial time regardless of the tree-width, without even knowing the tree-width?

Order-two (edge) graphical models

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

$$E(x) = \sum_{v \in V(G)} e_v(x_v) + \sum_{(i,j) \in E(G)} e_{ij}(x_i, x_j) \tag{73}$$

Order-two (edge) graphical models

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

$$E(x) = \sum_{v \in V(G)} e_v(x_v) + \sum_{(i,j) \in E(G)} e_{ij}(x_i, x_j) \tag{73}$$

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

Order-two (edge) graphical models

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

$$E(x) = \sum_{v \in V(G)} e_v(x_v) + \sum_{(i,j) \in E(G)} e_{ij}(x_i, x_j) \tag{73}$$

- $e_v(x_v)$ 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.

Order-two (edge) graphical models

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

$$E(x) = \sum_{v \in V(G)} e_v(x_v) + \sum_{(i,j) \in E(G)} e_{ij}(x_i, x_j) \tag{73}$$

- $e_v(x_v)$ 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.
- Further, say that $\mathrm{D}_{X_v} = \{0, 1\}$ (binary), so we have binary random vectors distributed according to $p(x)$.

Order-two (edge) graphical models

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

$$E(x) = \sum_{v \in V(G)} e_v(x_v) + \sum_{(i,j) \in E(G)} e_{ij}(x_i, x_j) \tag{73}$$

- $e_v(x_v)$ 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.
- Further, say that $\mathrm{D}_{X_v} = \{0, 1\}$ (binary), so we have binary random vectors distributed according to $p(x)$.
- Thus, $x \in \{0, 1\}^V$, and finding MPE solution is setting some of the variables to 0 and some to 1, i.e.,

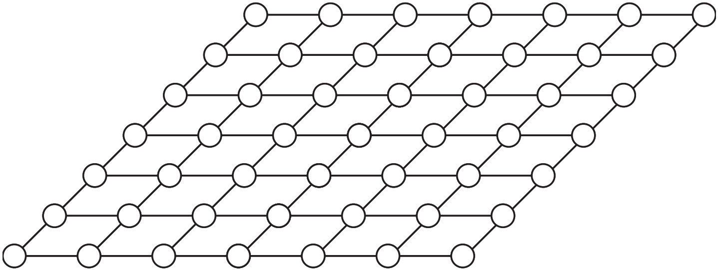
$$\min_{x \in \{0,1\}^V} E(x) \tag{74}$$

MRF example

Markov random field

$$\log p(x) \propto \sum_{v \in V(G)} e_v(x_v) + \sum_{(i,j) \in E(G)} e_{ij}(x_i, x_j) \tag{75}$$

When G is a 2D grid graph, we have

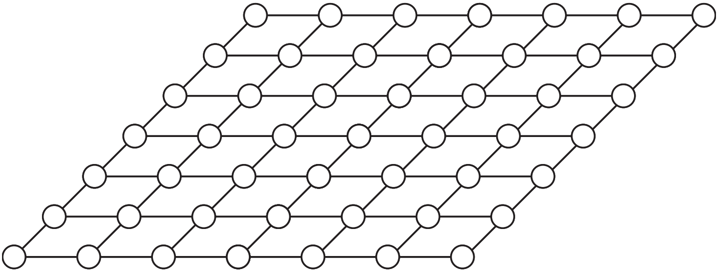


Create an auxiliary graph

- We can create auxiliary graph G_a that involves two new "terminal" nodes s and t and all of the original "non-terminal" nodes $v \in V(G)$.
- The non-terminal nodes represent the original random variables $x_v, v \in V$.
- Starting with the original grid-graph amongst the vertices $v \in V$, we connect each of s and t to all of the original nodes.
- I.e., we form $G_a = (V \cup \{s, t\}, E + \cup_{v \in V}((s, v) \cup (v, t)))$.

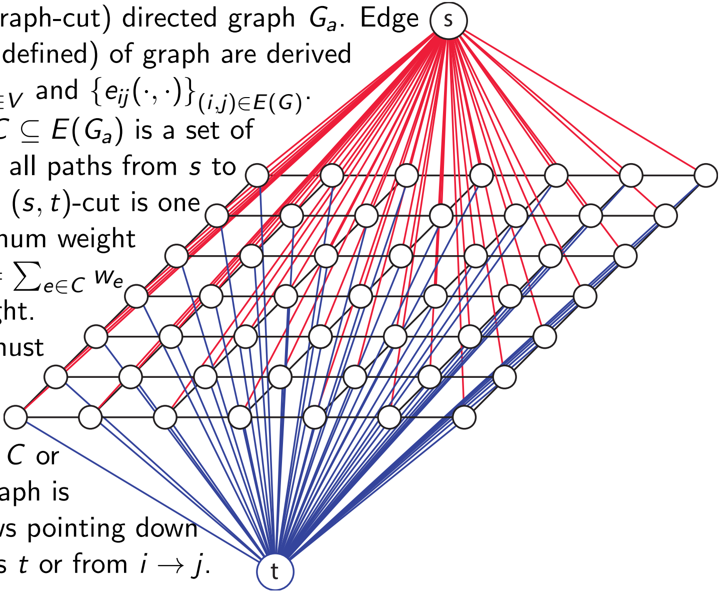
Transformation from graphical model to auxiliary graph

Original 2D-grid graphical model G and energy function $E(x) = \sum_{v \in V(G)} e_v(x_v) + \sum_{(i,j) \in E(G)} e_{ij}(x_i, x_j)$ needing to be minimized over $x \in \{0,1\}^V$. Recall, tree-width is $O(\sqrt{|V|})$.



Transformation from graphical model to auxiliary graph

Augmented (graph-cut) directed graph G_a. Edge weights (soon defined) of graph are derived from $\{e_v(\cdot)\}_{v \in V}$ and $\{e_{ij}(\cdot,\cdot)\}_{(i,j)\in E(G)}$.
An (s,t)-cut $C \subseteq E(G_a)$ is a set of edges that cut all paths from s to t. A minimum (s,t)-cut is one that has minimum weight where $w(C) = \sum_{e \in C} w_e$ is the cut weight.
To be a cut, must have that, for every $v \in V$, either $(s,v) \in C$ or $(v,t) \in C$. Graph is directed, arrows pointing down from s towards t or from $i \to j$.

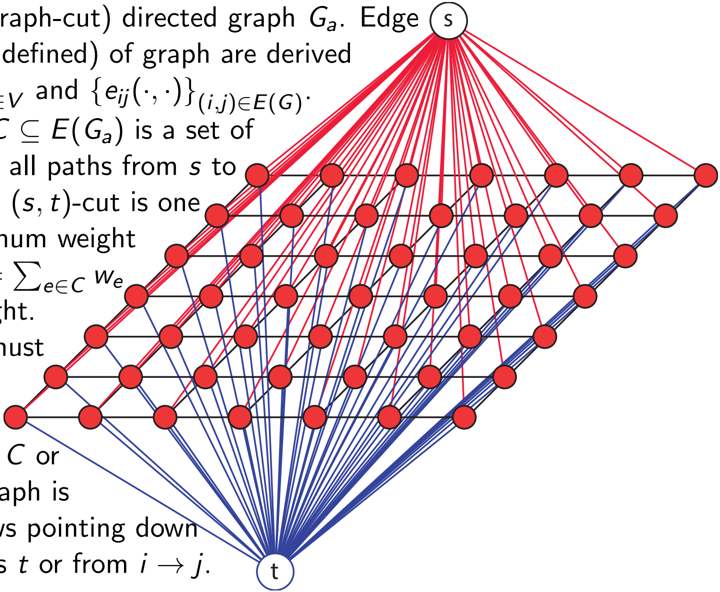


Transformation from graphical model to auxiliary graph

Augmented (graph-cut) directed graph G_a. Edge weights (soon defined) of graph are derived from $\{e_v(\cdot)\}_{v \in V}$ and $\{e_{ij}(\cdot,\cdot)\}_{(i,j)\in E(G)}$.

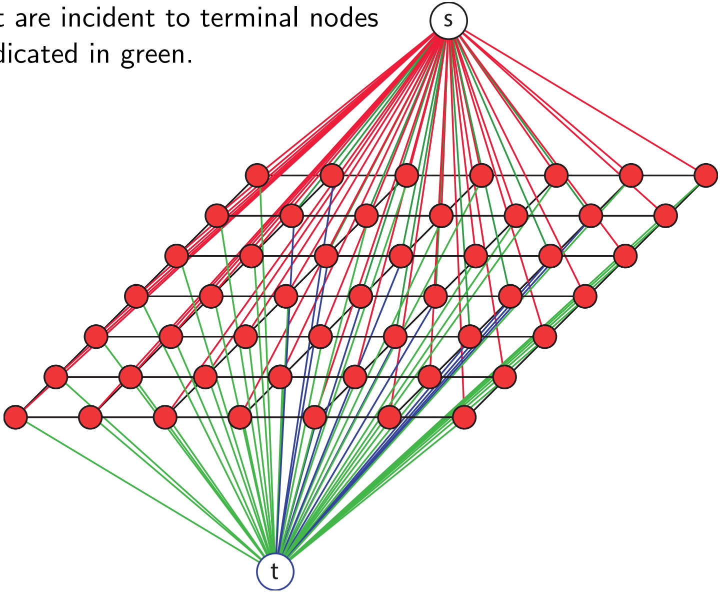
An (s,t)-cut $C \subseteq E(G_a)$ is a set of edges that cut all paths from s to t. A minimum (s,t)-cut is one that has minimum weight where $w(C) = \sum_{e \in C} w_e$ is the cut weight.

To be a cut, must have that, for every $v \in V$, either $(s,v) \in C$ or $(v,t) \in C$. Graph is directed, arrows pointing down from s towards t or from $i \to j$.



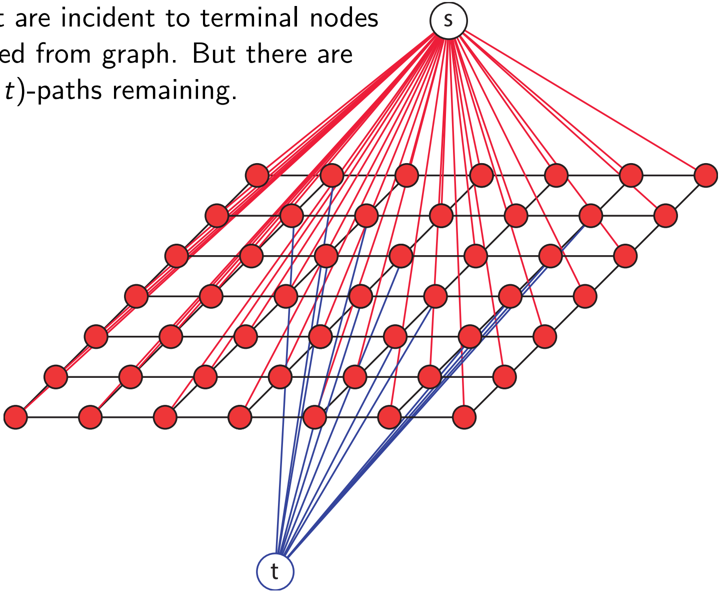
Transformation from graphical model to auxiliary graph

Cut edges that are incident to terminal nodes s and t are indicated in green.



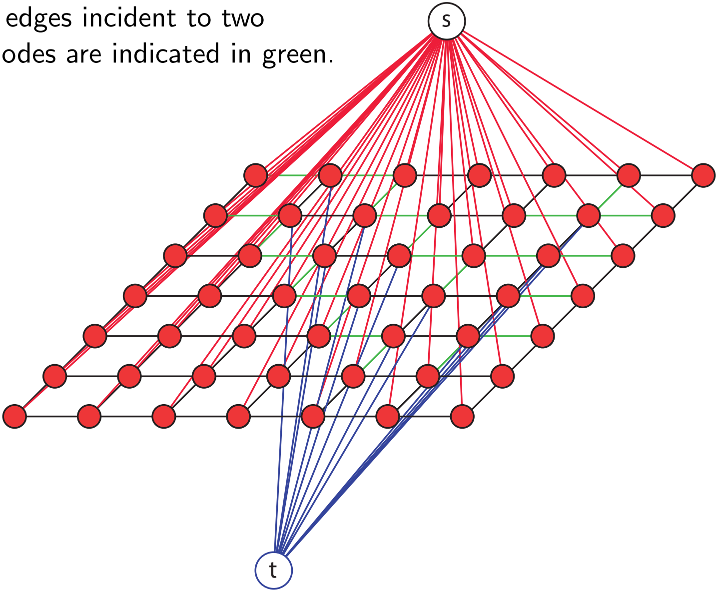
Transformation from graphical model to auxiliary graph

Cut edges that are incident to terminal nodes s and t removed from graph. But there are still un-cut (s, t)-paths remaining.



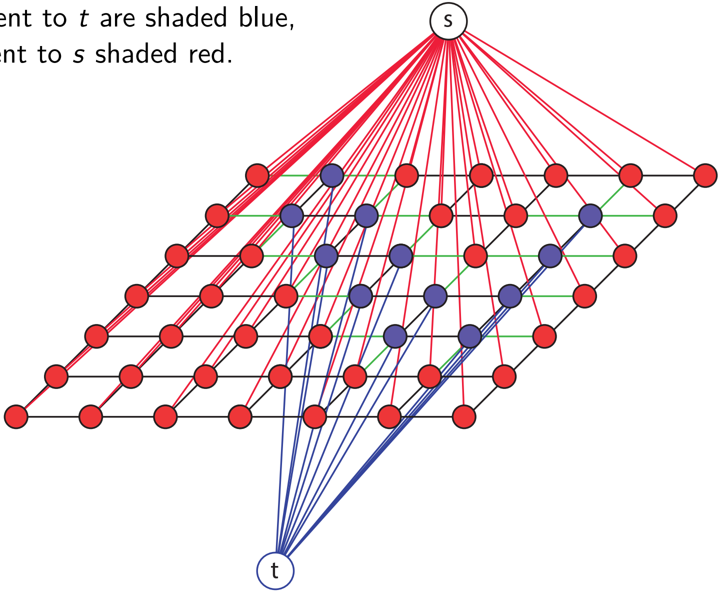
Transformation from graphical model to auxiliary graph

Additional cut edges incident to two non-terminal nodes are indicated in green.



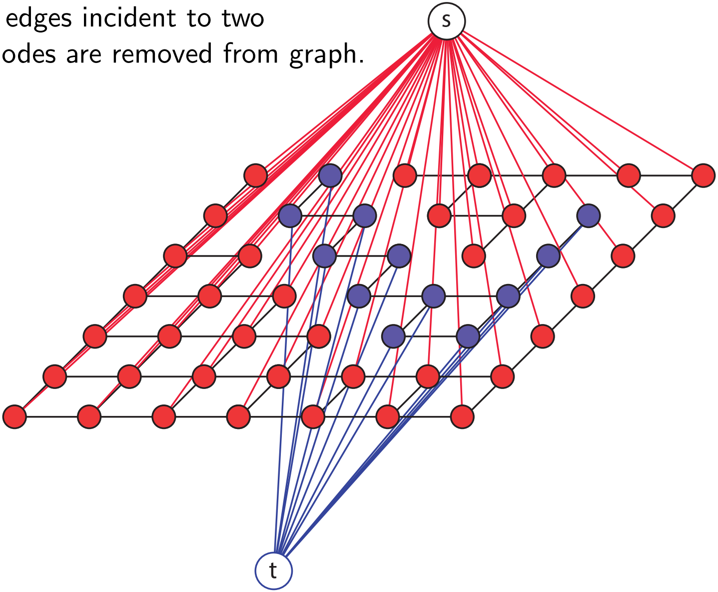
Transformation from graphical model to auxiliary graph

Vertices adjacent to t are shaded blue, vertices adjacent to s shaded red.



Transformation from graphical model to auxiliary graph

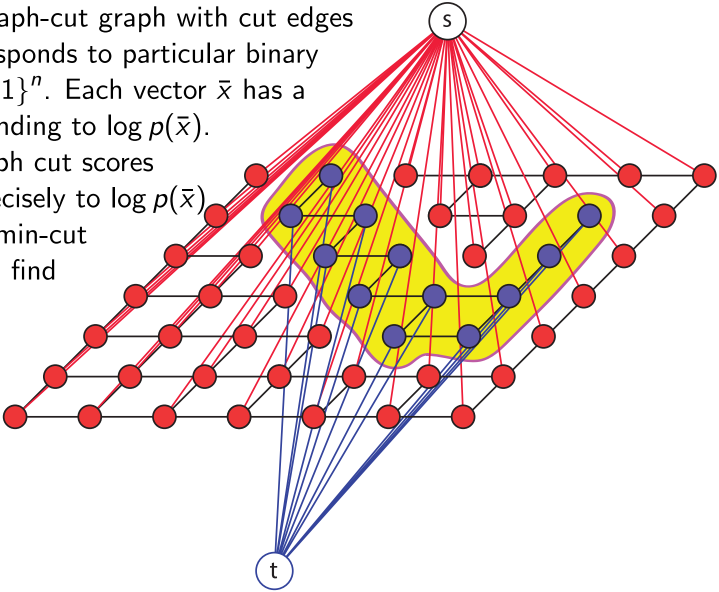
Additional cut edges incident to two non-terminal nodes are removed from graph.



Transformation from graphical model to auxiliary graph

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

When can graph cut scores correspond precisely to $\log p(\bar{x})$ in a way that min-cut algorithms can find minimum of energy $E(x)$?



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 of all edges, except those involving terminals s and t, are non-negative, graph cut computable in polynomial time via max-flow (many algorithms, e.g., Edmonds&Karp $O(nm^2)$ or $O(n^2 m \log(nC))$; Goldberg&Tarjan $O(nm \log(n^2/m))$, see Schrijver, page 161).

Setting of the weights in the auxiliary cut graph

- Any graph cut corresponds to a vector $\bar{x} \in \{0,1\}^n$.
- If weights of all edges, except those involving terminals s and t, are non-negative, graph cut computable in polynomial time via max-flow (many algorithms, e.g., Edmonds&Karp $O(nm^2)$ or $O(n^2 m \log(nC))$; Goldberg&Tarjan $O(nm \log(n^2/m))$, see Schrijver, page 161).
- 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 of all edges, except those involving terminals s and t, are non-negative, graph cut computable in polynomial time via max-flow (many algorithms, e.g., Edmonds&Karp $O(nm^2)$ or $O(n^2 m \log(nC))$; Goldberg&Tarjan $O(nm \log(n^2/m))$, see Schrijver, page 161).
- 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 $E(x) = \log p(\bar{x}) + \text{const.}$.
- 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 of all edges, except those involving terminals s and t, are non-negative, graph cut computable in polynomial time via max-flow (many algorithms, e.g., Edmonds&Karp $O(nm^2)$ or $O(n^2 m \log(nC))$; Goldberg&Tarjan $O(nm \log(n^2/m))$, see Schrijver, page 161).
- 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 $E(x) = \log p(\bar{x}) + \text{const.}$.
- Hence, poly time graph cut, can find the optimal MPE assignment, regardless of the graphical model's tree-width!
- In general, finding MPE is an NP-hard optimization problem.

Setting of the weights in the auxiliary cut graph

Edge weight assignments. Start with all weights set to zero.

- For (s, v) with $v \in V(G)$, set edge

$$w_{s,v} = (e_v(1) - e_v(0))\mathbf{1}(e_v(1) > e_v(0)) \tag{76}$$

- For (v, t) with $v \in V(G)$, set edge

$$w_{v,t} = (e_v(0) - e_v(1))\mathbf{1}(e_v(0) \geq e_v(1)) \tag{77}$$

- For original edge $(i, j) \in E$, $i, j \in V$, set weight

$$w_{i,j} = e_{ij}(1,0) + e_{ij}(0,1) - e_{ij}(1,1) - e_{ij}(0,0) \tag{78}$$

and if $e_{ij}(1,0) > e_{ij}(0,0)$, and $e_{ij}(1,1) > e_{ij}(0,1)$,

$$w_{s,i} \leftarrow w_{s,i} + (e_{ij}(1,0) - e_{ij}(0,0)) \tag{79}$$

$$w_{j,t} \leftarrow w_{j,t} + (e_{ij}(1,1) - e_{ij}(0,1)) \tag{80}$$

and analogous increments if inequalities are flipped.

Restricted clique functions

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

$$e_{ij}(0,1) + e_{ij}(1,0) \geq e_{ij}(1,1) + e_{ij}(0,0) \tag{81}$$

  which is a special case of more general submodular functions.

Restricted clique functions

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

$$e_{ij}(0,1) + e_{ij}(1,0) \geq e_{ij}(1,1) + e_{ij}(0,0) \tag{81}$$

  which is a special case of more general submodular functions.
- In probability form $p(x) \propto \prod \psi$, we get $\psi_{ij}(1,0)\psi_{ij}(0,1) \leq \psi_{ij}(0,0)\psi_{ij}(1,1)$, so geometric mean of factor scores (thus probability) is higher when neighboring pixels have the same value - reasonable assumption in natural scenes and signals.

Restricted clique functions

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

$$e_{ij}(0,1) + e_{ij}(1,0) \geq e_{ij}(1,1) + e_{ij}(0,0) \tag{81}$$

  which is a special case of more general submodular functions.
- In probability form $p(x) \propto \prod \psi$, we get $\psi_{ij}(1,0)\psi_{ij}(0,1) \leq \psi_{ij}(0,0)\psi_{ij}(1,1)$, so geometric mean of factor scores (thus probability) is higher when neighboring pixels have the same value - reasonable assumption in natural scenes and signals.
- So weights w_{ij} in s,t-graph above are always non-negative.

Restricted clique functions

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

$$e_{ij}(0,1) + e_{ij}(1,0) \geq e_{ij}(1,1) + e_{ij}(0,0) \tag{81}$$

  which is a special case of more general submodular functions.
- In probability form $p(x) \propto \prod \psi$, we get $\psi_{ij}(1,0)\psi_{ij}(0,1) \leq \psi_{ij}(0,0)\psi_{ij}(1,1)$, so geometric mean of factor scores (thus probability) is higher when neighboring pixels have the same value - reasonable assumption in natural scenes and signals.
- So weights w_{ij} in s,t-graph above are always non-negative.

Theorem

If edge functions are submodular and edge weights in s,t-graph are set as above, then finding the minimum s,t-cut in the auxiliary graph will yield a variable assignment having maximum probability.

Non-negative edge weights

- The inequalities ensures that we are adding non-negative weights to each of the edges. I.e., we do $w_{s,i} \leftarrow w_{s,i} + (e_{ij}(1,0) - e_{ij}(0,0))$ only if $e_{ij}(1,0) > e_{ij}(0,0)$.

Non-negative edge weights

- The inequalities ensures that we are adding non-negative weights to each of the edges. I.e., we do $w_{s,i} \leftarrow w_{s,i} + (e_{ij}(1,0) - e_{ij}(0,0))$ only if $e_{ij}(1,0) > e_{ij}(0,0)$.
- For (i,j) edge weight, it takes the form:

$$w_{i,j} = e_{ij}(1,0) + e_{ij}(0,1) - e_{ij}(1,1) - e_{ij}(0,0) \tag{82}$$

Non-negative edge weights

- The inequalities ensures that we are adding non-negative weights to each of the edges. I.e., we do $w_{s,i} \leftarrow w_{s,i} + (e_{ij}(1,0) - e_{ij}(0,0))$ only if $e_{ij}(1,0) > e_{ij}(0,0)$.
- For (i,j) edge weight, it takes the form:

$$w_{i,j} = e_{ij}(1,0) + e_{ij}(0,1) - e_{ij}(1,1) - e_{ij}(0,0) \tag{82}$$

- For this to be non-negative, we need:

$$e_{ij}(1,0) + e_{ij}(0,1) \geq e_{ij}(1,1) + e_{ij}(0,0) \tag{83}$$

Non-negative edge weights

- The inequalities ensures that we are adding non-negative weights to each of the edges. I.e., we do $w_{s,i} \leftarrow w_{s,i} + (e_{ij}(1,0) - e_{ij}(0,0))$ only if $e_{ij}(1,0) > e_{ij}(0,0)$.
- For (i,j) edge weight, it takes the form:

$$w_{i,j} = e_{ij}(1,0) + e_{ij}(0,1) - e_{ij}(1,1) - e_{ij}(0,0) \tag{82}$$

- For this to be non-negative, we need:

$$e_{ij}(1,0) + e_{ij}(0,1) \geq e_{ij}(1,1) + e_{ij}(0,0) \tag{83}$$

- Thus weights w_{ij} in s,t-graph above are always non-negative, so graph-cut solvable exactly.

Submodular potentials

- Edge functions must be submodular (in the binary case, equivalent to "associative", "attractive", "regular", "Potts", or "ferromagnetic"): for all $(i, j) \in E(G)$, must have:

$$e_{ij}(0,1) + e_{ij}(1,0) \geq e_{ij}(1,1) + e_{ij}(0,0) \tag{84}$$

Submodular potentials

- Edge functions must be submodular (in the binary case, equivalent to "associative", "attractive", "regular", "Potts", or "ferromagnetic"): for all $(i, j) \in E(G)$, must have:

$$e_{ij}(0,1) + e_{ij}(1,0) \geq e_{ij}(1,1) + e_{ij}(0,0) \tag{84}$$

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

Submodular potentials

- Edge functions must be submodular (in the binary case, equivalent to "associative", "attractive", "regular", "Potts", or "ferromagnetic"): for all $(i, j) \in E(G)$, must have:

$$e_{ij}(0,1) + e_{ij}(1,0) \geq e_{ij}(1,1) + e_{ij}(0,0) \tag{84}$$

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

$$f(X) = \sum_{\{i,j\} \in \mathcal{E}(G)} f_{i,j}(X \cap \{i,j\}) \tag{85}$$

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

Submodular potentials

- Edge functions must be submodular (in the binary case, equivalent to "associative", "attractive", "regular", "Potts", or "ferromagnetic"): for all $(i, j) \in E(G)$, must have:

$$e_{ij}(0,1) + e_{ij}(1,0) \geq e_{ij}(1,1) + e_{ij}(0,0) \tag{84}$$

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

$$f(X) = \sum_{\{i,j\} \in \mathcal{E}(G)} f_{i,j}(X \cap \{i, j\}) \tag{85}$$

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

- A special case of more general submodular functions – unconstrained submodular function minimization is solvable in polytime.

On log-supermodular vs. log-submodular distributions

- Log-supermodular distributions.

$$\log \Pr(x) = f(x) + \text{const.} = -E(x) + \text{const.} \tag{86}$$

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.

$$\log \Pr(x) = f(x) + \text{const.} = -E(x) + \text{const.} \tag{86}$$

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:

$$\log \Pr(x) = f(x) + \text{const.} \tag{87}$$

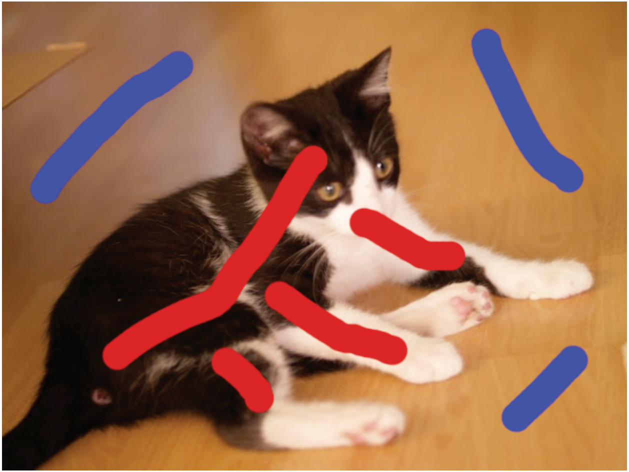
where f is submodular. MAP or high-probable assignments should be "diverse", or "complex", or "covering", like in determinantal point processes.

Submodular potentials in GMs: Image Segmentation

- an image needing to be segmented.

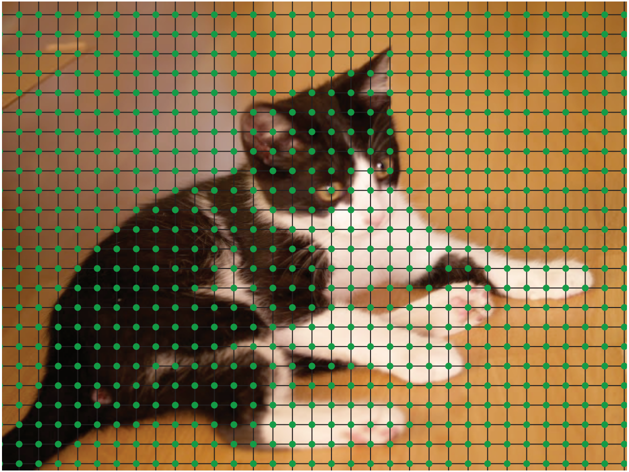
Submodular potentials in GMs: Image Segmentation

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



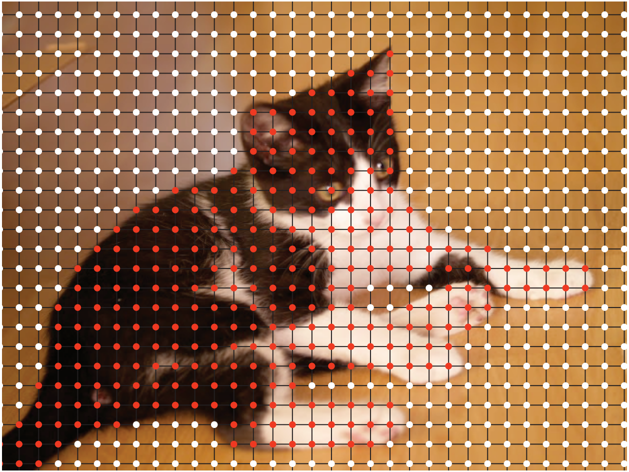
Submodular potentials in GMs: Image Segmentation

- 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.

Graph Cut Marginalization

- What to do when potentials are not submodular?

Graph Cut Marginalization

- What to do when potentials are not submodular? QPBO, quadratic pseudo Boolean optimization (computes only a partial solution).

Graph Cut Marginalization

- What to do when potentials are not submodular? QPBO, quadratic pseudo Boolean optimization (computes only a partial solution).
- Move making algorithms ($\alpha - \beta$-swaps, α-expansions, fusion moves, etc.)

Graph Cut Marginalization

- What to do when potentials are not submodular? QPBO, quadratic pseudo Boolean optimization (computes only a partial solution).
- Move making algorithms ($\alpha - \beta$-swaps, α-expansions, fusion moves, etc.)
- Is submodularity sufficient to make standard marginalization possible?

Graph Cut Marginalization

- What to do when potentials are not submodular? QPBO, quadratic pseudo Boolean optimization (computes only a partial solution).
- Move making algorithms ($\alpha - \beta$-swaps, α-expansions, fusion moves, etc.)
- Is submodularity sufficient to make standard marginalization possible?
- Unfortunately, even in submodular case, computing partition function is a #P-complete problem (if it was possible to do it in poly time, that would require $P = NP$).

Graph Cut Marginalization

- What to do when potentials are not submodular? QPBO, quadratic pseudo Boolean optimization (computes only a partial solution).
- Move making algorithms ($\alpha - \beta$-swaps, α-expansions, fusion moves, etc.)
- Is submodularity sufficient to make standard marginalization possible?
- Unfortunately, even in submodular case, computing partition function is a #P-complete problem (if it was possible to do it in poly time, that would require $P = NP$).
- On the other hand, for pairwise MRFs, computing partition function in submodular potential case is approximable (has low error with high probability).

Graph Cut Marginalization

- What to do when potentials are not submodular? QPBO, quadratic pseudo Boolean optimization (computes only a partial solution).
- Move making algorithms ($\alpha - \beta$-swaps, α-expansions, fusion moves, etc.)
- Is submodularity sufficient to make standard marginalization possible?
- Unfortunately, even in submodular case, computing partition function is a #P-complete problem (if it was possible to do it in poly time, that would require $P = NP$).
- On the other hand, for pairwise MRFs, computing partition function in submodular potential case is approximable (has low error with high probability).
- SPPs and log(SPP)s (Rishabh's talk) will also talk about how submodularity allows further approximations via semigradients.

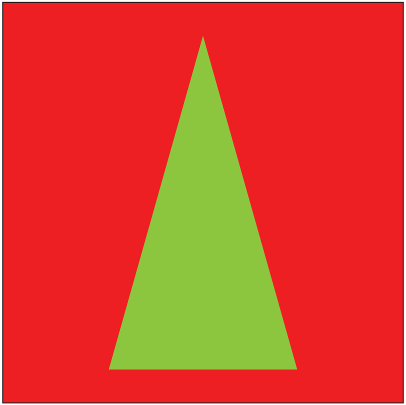
Shrinking bias in graph cut image segmentation

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

Shrinking bias in graph cut image segmentation

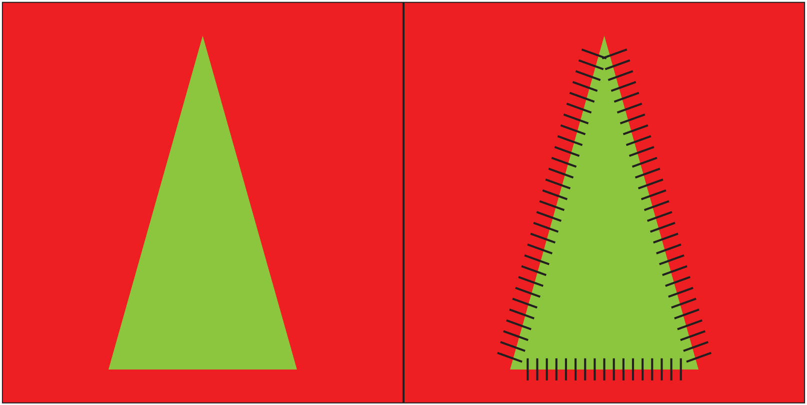
Shrinking bias in image segmentation

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



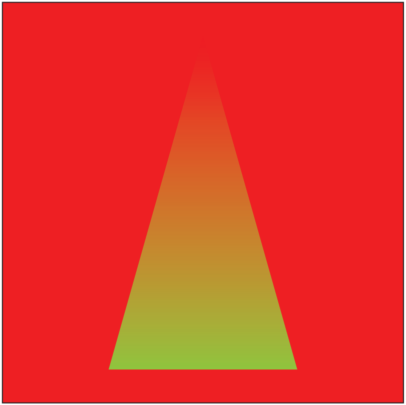
Shrinking bias in image segmentation

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



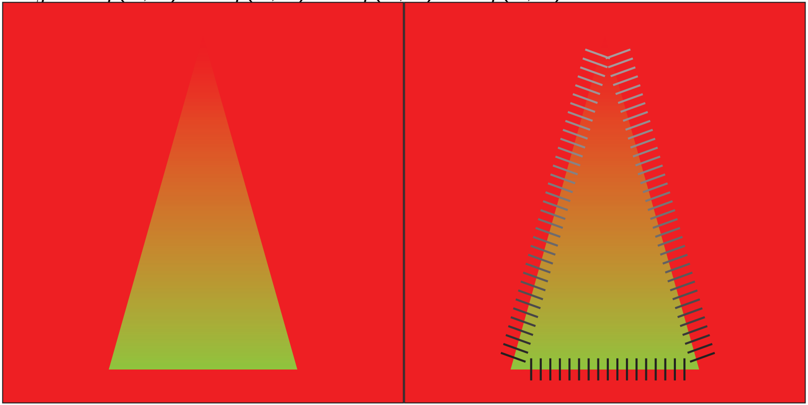
Shrinking bias in image segmentation

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



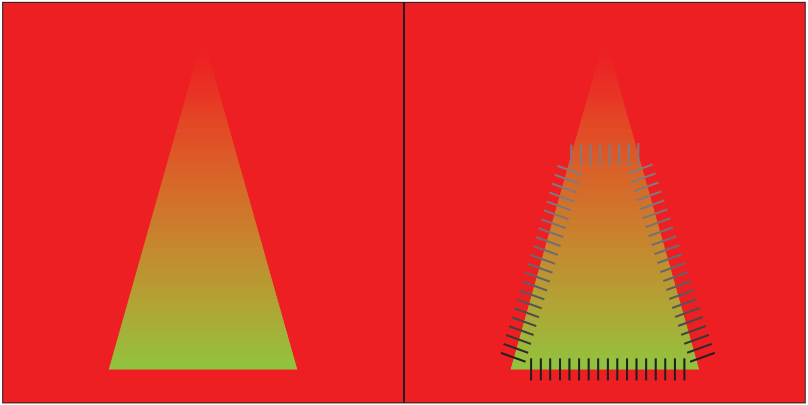
Shrinking bias in image segmentation

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



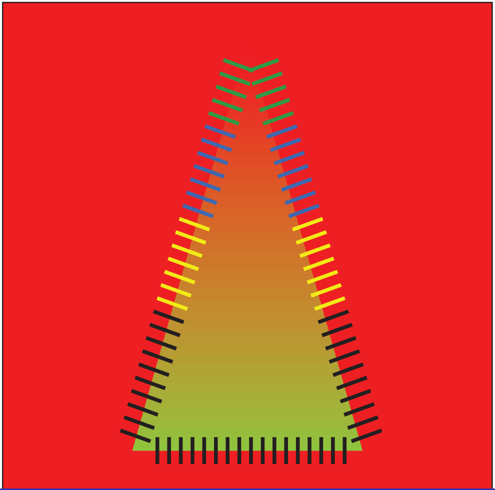
Shrinking bias in image segmentation

- 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.

$$f_w(X) = w\Big(\{(u, v) \in E : u \in X, v \in V \setminus X\}\Big) \tag{88}$$

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.

$$f_w(X) = w\Big(\{(u, v) \in E : u \in X, v \in V \setminus X\}\Big) \tag{88}$$

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

$$f_g(X) = g\Big(\{(u, v) \in E : u \in X, v \in V \setminus X\}\Big) \tag{89}$$

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.

$$f_w(X) = w\Big(\{(u,v) \in E : u \in X, v \in V \setminus X\}\Big) \tag{88}$$

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

$$f_g(X) = g\Big(\{(u,v) \in E : u \in X, v \in V \setminus X\}\Big) \tag{89}$$

- 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.

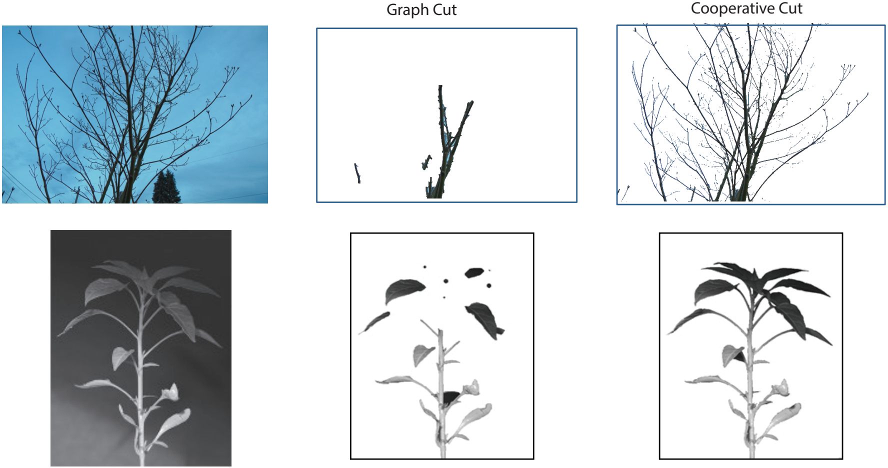
$$f_w(X) = w\Big(\{(u,v) \in E : u \in X, v \in V \setminus X\}\Big) \tag{88}$$

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

$$f_g(X) = g\Big(\{(u,v) \in E : u \in X, v \in V \setminus X\}\Big) \tag{89}$$

- 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 & B., 2011).

Graph-cut vs. cooperative-cut comparisons



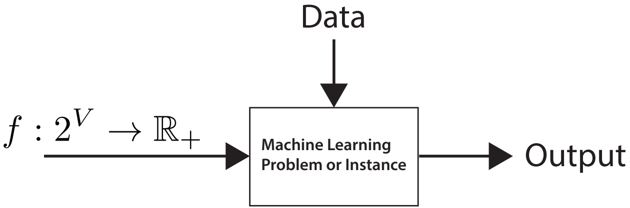
(Jegelka&Bilmes,'11). There are fast algorithms for solving as well.

Outline: Part 2

5. Submodular Applications in Machine Learning
   - Where is submodularity useful?
6. As a model of diversity, coverage, span, or information
7. As a model of cooperative costs, complexity, roughness, and irregularity
8. As a Parameter for an ML algorithm
9. Itself, as a target for learning
10. Surrogates for optimization and analysis
11. Reading
   - Refs

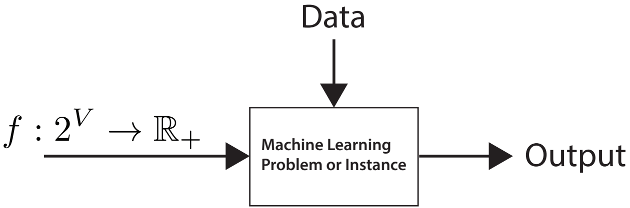
A submodular function as a parameter

- 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

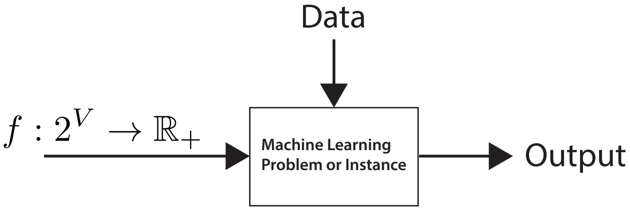
- In some cases, it may be useful to view a submodular function $f : 2^V \to \mathbb{R}$ as a input "parameter" to a machine learning algorithm.



- 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

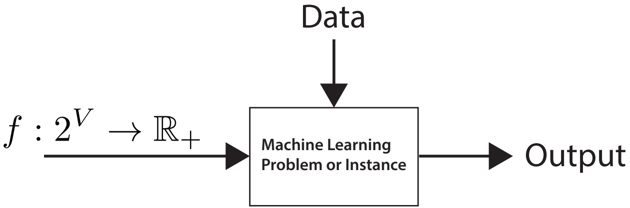
- In some cases, it may be useful to view a submodular function $f : 2^V \to \mathbb{R}$ as a input "parameter" to a machine learning algorithm.



- 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 $f + f_\epsilon \in \mathbb{S}$.

Supervised 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:

$$\min_{w \in \mathbb{R}^n} \frac{1}{m} \sum_{i=1}^m \ell(y_i, w^\intercal x_i) + \lambda \Omega(w), \tag{90}$$

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

Supervised 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:

$$\min_{w \in \mathbb{R}^n} \frac{1}{m} \sum_{i=1}^m \ell(y_i, w^\intercal x_i) + \lambda \Omega(w), \tag{90}$$

  where $\ell(\cdot)$ is a loss function (e.g., squared error) and $\Omega(w)$ is a norm.
- When data has multiple (k) responses $(x_i, y_i) \in \mathbb{R}^n \times \mathbb{R}^k$ for each of the m samples, learning becomes:

$$\min_{w^1, \dots, w^k \in \mathbb{R}^n} \sum_{j=1}^k \frac{1}{m} \sum_{i=1}^m \ell(y_i^j, (w^j)^\intercal x_i) + \lambda \Omega(w^j), \tag{91}$$

Dictionary Learning and Selection

- When only the multiple responses $\{y_i\}_{i \in [m]}$ are observed, we get either dictionary learning

$$\min_{x_1,\ldots,x_m} \min_{w^1,\ldots,w^k \in \mathbb{R}^n} \sum_{j=1}^{k} \frac{1}{m} \sum_{i=1}^{m} \ell(y_i^j, (w^j)^\top x_i) + \lambda \Omega(w^j), \tag{92}$$

Dictionary Learning and Selection

- When only the multiple responses $\{y_i\}_{i\in[m]}$ are observed, we get either dictionary learning

$$\min_{x_1,\ldots,x_m} \min_{w^1,\ldots,w^k\in\mathbb{R}^n} \sum_{j=1}^{k} \frac{1}{m} \sum_{i=1}^{m} \ell(y_i^j, (w^j)^\intercal x_i) + \lambda\Omega(w^j), \tag{92}$$

- or when we select sub-dimensions of x, we get dictionary selection (Cevher & Krause, Das & Kempe).

$$f(D) = \sum_{j=1}^{k} \min_{S\subseteq D,|S|\leq k} \min_{w^j\in\mathbb{R}^S} \left(\sum_{i=1}^{m} \ell(y_i^j, (w^j)^\intercal x_i^S) + \lambda\Omega(w^j) \right) \tag{93}$$

where D is the dictionary (indices of x that are allowed), and x^S is a sub-vector of x. Each regression allows at most $k \leq |D|$ variables.

Dictionary Learning and Selection

- When only the multiple responses $\{y_i\}_{i\in[m]}$ are observed, we get either dictionary learning

$$\min_{x_1,\dots,x_m} \min_{w^1,\dots,w^k\in\mathbb{R}^n} \sum_{j=1}^{k} \frac{1}{m}\sum_{i=1}^{m} \ell(y_i^j,(w^j)^\intercal x_i) + \lambda\Omega(w^j), \tag{92}$$

- or when we select sub-dimensions of x, we get dictionary selection (Cevher & Krause, Das & Kempe).

$$f(D) = \sum_{j=1}^{k} \min_{S\subseteq D,|S|\leq k} \min_{w^j\in\mathbb{R}^S} \left(\sum_{i=1}^{m} \ell(y_i^j,(w^j)^\intercal x_i^S) + \lambda\Omega(w^j)\right) \tag{93}$$

  where D is the dictionary (indices of x that are allowed), and x^S is a sub-vector of x. Each regression allows at most $k \leq |D|$ variables.

- In each case of the above cases, the regularizer $\Omega(\cdot)$ is critical.

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:

$$\Omega(w) = \sum_{i=2}^{N} |w_i - w_{i-1}| \tag{94}$$

- 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

$$\tilde{f}(w) = \sum_{i=1}^{n} w_{\sigma_i}\left(f(\sigma_1, \dots, \sigma_i) - f(\sigma_1, \dots, \sigma_{i-1})\right) \tag{95}$$

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

$$\tilde{f}(w) = \sum_{i=1}^{n} w_{\sigma_i}(f(\sigma_1, \ldots, \sigma_i) - f(\sigma_1, \ldots, \sigma_{i-1})) \tag{95}$$

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

Submodular Generalized Dependence

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

$$f(A \cup B) = f(A) + f(B), \tag{96}$$

Submodular Generalized Dependence

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

$$f(A \cup B) = f(A) + f(B), \tag{96}$$

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

$$f(A \cup B \cup C) + f(C) = f(A \cup C) + f(B \cup C) \tag{97}$$

Submodular Generalized Dependence

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

$$f(A \cup B) = f(A) + f(B), \tag{96}$$

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

$$f(A \cup B \cup C) + f(C) = f(A \cup C) + f(B \cup C) \tag{97}$$

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

$$f(A|B) \triangleq f(A \cup B) - f(B) < f(A), \tag{98}$$

Submodular Generalized Dependence

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

$$f(A \cup B) = f(A) + f(B), \tag{96}$$

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

$$f(A \cup B \cup C) + f(C) = f(A \cup C) + f(B \cup C) \tag{97}$$

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

$$f(A|B) \triangleq f(A \cup B) - f(B) < f(A), \tag{98}$$

- 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., $A \perp\!\!\!\perp B$:

$$f(A \cup B) = f(A) + f(B), \tag{96}$$

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

$$f(A \cup B \cup C) + f(C) = f(A \cup C) + f(B \cup C) \tag{97}$$

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

$$f(A|B) \triangleq f(A \cup B) - f(B) < f(A), \tag{98}$$

- 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":

$$I_f(S_1; S_2; \ldots; S_k) = \sum_{i=1}^{k} f(S_k) - f(S_1 \cup S_2 \cup \cdots \cup S_k) \tag{99}$$

$$I'_f(S_1; S_2; \ldots; S_k) = \sum_{A \subseteq \{1,2,\ldots,k\}} (-1)^{|A|+1} f(\bigcup_{j \in A} S_j) \tag{100}$$

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 $A_1^* \in \text{argmin}_{A \subseteq V} I_f(A; V \setminus A)$ and $A_2^* = V \setminus A_1^*$.

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 $A_1^* \in \operatorname{argmin}_{A \subseteq V} I_f(A; V \setminus A)$ and $A_2^* = V \setminus A_1^*$.
- Then partition the partitions: $A_{11}^* \in \operatorname{argmin}_{A \subseteq A_1^*} I_f(A; A_1^* \setminus A)$, $A_{12}^* = A_1^* \setminus A_{11}^*$, and $A_{21}^* \in \operatorname{argmin}_{A \subseteq A_2^*} I_f(A; A_2^* \setminus A)$, etc.

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 $A_1^* \in \text{argmin}_{A \subseteq V} I_f(A; V \setminus A)$ and $A_2^* = V \setminus A_1^*$.
- Then partition the partitions: $A_{11}^* \in \text{argmin}_{A \subseteq A_1^*} I_f(A; A_1^* \setminus A)$, $A_{12}^* = A_1^* \setminus A_{11}^*$, and $A_{21}^* \in \text{argmin}_{A \subseteq A_2^*} I_f(A; A_2^* \setminus A)$, etc.
- 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 $A_1^* \in \operatorname{argmin}_{A \subseteq V} I_f(A; V \setminus A)$ and $A_2^* = V \setminus A_1^*$.
- Then partition the partitions: $A_{11}^* \in \operatorname{argmin}_{A \subseteq A_1^*} I_f(A; A_1^* \setminus A)$, $A_{12}^* = A_1^* \setminus A_{11}^*$, and $A_{21}^* \in \operatorname{argmin}_{A \subseteq A_2^*} I_f(A; A_2^* \setminus A)$, etc.
- 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 $A_1^* \in \operatorname{argmin}_{A \subseteq V} I_f(A; V \setminus A)$ and $A_2^* = V \setminus A_1^*$.
- Then partition the partitions: $A_{11}^* \in \operatorname{argmin}_{A \subseteq A_1^*} I_f(A; A_1^* \setminus A)$, $A_{12}^* = A_1^* \setminus A_{11}^*$, and $A_{21}^* \in \operatorname{argmin}_{A \subseteq A_2^*} I_f(A; A_2^* \setminus A)$, etc.
- 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.

Is Submodular Maximization Just Clustering?

1. Clustering objectives often NP-hard and inapproximable, submodular maximization is approximable for any submodular function.

Is Submodular Maximization Just Clustering?

1. Clustering objectives often NP-hard and inapproximable, submodular maximization is approximable for any submodular function.
2. To have guarantee, clustering typically needs metricity, submodularity parameterized via any non-negative pairwise values.

Is Submodular Maximization Just Clustering?

1. Clustering objectives often NP-hard and inapproximable, submodular maximization is approximable for any submodular function.
2. To have guarantee, clustering typically needs metricity, submodularity parameterized via any non-negative pairwise values.
3. Clustering often requires separate process to choose representatives within each cluster. Submodular max does this automatically. Can also do submodular data partitioning (like clustering).

Is Submodular Maximization Just Clustering?

1. Clustering objectives often NP-hard and inapproximable, submodular maximization is approximable for any submodular function.
2. To have guarantee, clustering typically needs metricity, submodularity parameterized via any non-negative pairwise values.
3. Clustering often requires separate process to choose representatives within each cluster. Submodular max does this automatically. Can also do submodular data partitioning (like clustering).
4. Submodular max covers clustering objectives such as k-medoids.

Is Submodular Maximization Just Clustering?

1. Clustering objectives often NP-hard and inapproximable, submodular maximization is approximable for any submodular function.
2. To have guarantee, clustering typically needs metricity, submodularity parameterized via any non-negative pairwise values.
3. Clustering often requires separate process to choose representatives within each cluster. Submodular max does this automatically. Can also do submodular data partitioning (like clustering).
4. Submodular max covers clustering objectives such as k-medoids.
5. Can learn submodular functions (hence, learn clustering objective).

Is Submodular Maximization Just Clustering?

1. Clustering objectives often NP-hard and inapproximable, submodular maximization is approximable for any submodular function.
2. To have guarantee, clustering typically needs metricity, submodularity parameterized via any non-negative pairwise values.
3. Clustering often requires separate process to choose representatives within each cluster. Submodular max does this automatically. Can also do submodular data partitioning (like clustering).
4. Submodular max covers clustering objectives such as k-medoids.
5. Can learn submodular functions (hence, learn clustering objective).
6. We can choose quality guarantee for any submodular function via submodular set cover (only possible for some clustering algorithms).

Is Submodular Maximization Just Clustering?

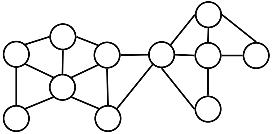
1. Clustering objectives often NP-hard and inapproximable, submodular maximization is approximable for any submodular function.
2. To have guarantee, clustering typically needs metricity, submodularity parameterized via any non-negative pairwise values.
3. Clustering often requires separate process to choose representatives within each cluster. Submodular max does this automatically. Can also do submodular data partitioning (like clustering).
4. Submodular max covers clustering objectives such as k-medoids.
5. Can learn submodular functions (hence, learn clustering objective).
6. We can choose quality guarantee for any submodular function via submodular set cover (only possible for some clustering algorithms).
7. Submodular max with constraints, ensures representatives are feasible (e.g., knapsack, matroid independence, combinatorial, submodular level set, etc.)

Is Submodular Maximization Just Clustering?

1. Clustering objectives often NP-hard and inapproximable, submodular maximization is approximable for any submodular function.
2. To have guarantee, clustering typically needs metricity, submodularity parameterized via any non-negative pairwise values.
3. Clustering often requires separate process to choose representatives within each cluster. Submodular max does this automatically. Can also do submodular data partitioning (like clustering).
4. Submodular max covers clustering objectives such as k-medoids.
5. Can learn submodular functions (hence, learn clustering objective).
6. We can choose quality guarantee for any submodular function via submodular set cover (only possible for some clustering algorithms).
7. Submodular max with constraints, ensures representatives are feasible (e.g., knapsack, matroid independence, combinatorial, submodular level set, etc.)
8. Submodular functions may be more general than clustering objectives (submodularity allows high-order interactions between elements).

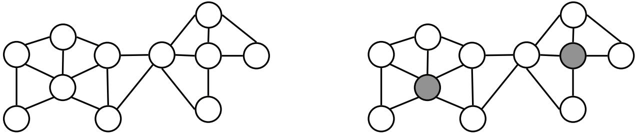
Active Transductive Semi-Supervised Learning

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

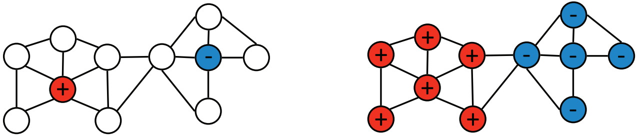


Active Transductive Semi-Supervised Learning

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

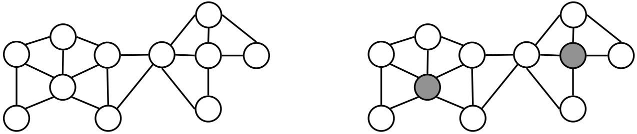


- Nature reveals labels $y_L \in \{0,1\}^L$, learner predicts labels $\hat{y} \in \{0,1\}^V$

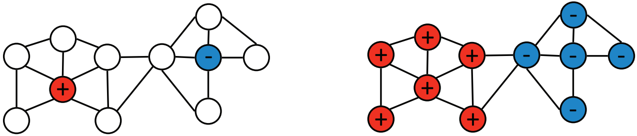


Active Transductive Semi-Supervised Learning

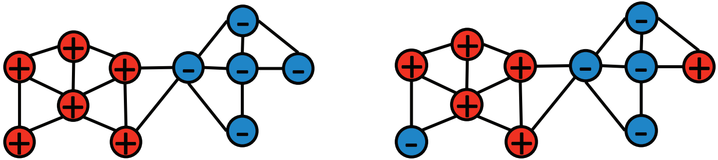
- Batch/Offline active learning: Given a set V of unlabeled data items, learner chooses subset $L \subseteq V$ of items to be labeled



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



- Learner suffers loss $\|\hat{y} - y\|_1$, where y is truth. Below, $\|\hat{y} - y\|_1 = 2$.



Choosing labels: how to select L

- Consider the following objective

$$\Psi(L) = \min_{T \subseteq V \setminus L : T \neq \emptyset} \frac{\Gamma(T)}{|T|} \tag{101}$$

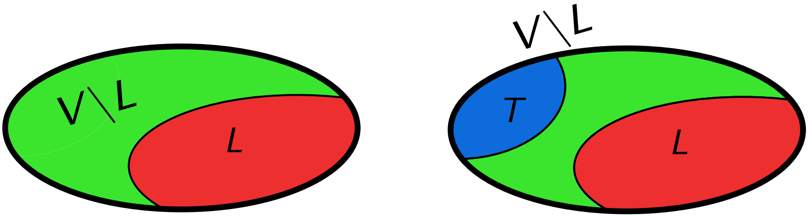
where $\Gamma(T) = I_f(T; V \setminus T) = f(T) + f(V \setminus T) - f(V)$ is an arbitrary symmetric submodular function (e.g., graph cut value between T and $V \setminus T$, or combinatorial mutual information).

Choosing labels: how to select L

- Consider the following objective

$$\Psi(L) = \min_{T \subseteq V \setminus L : T \neq \emptyset} \frac{\Gamma(T)}{|T|} \tag{101}$$

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



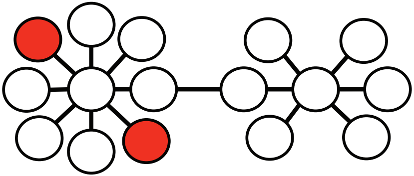
Choosing labels: how to select L

- Consider the following objective

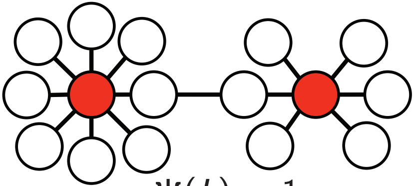
$$\Psi(L) = \min_{T \subseteq V \setminus L : T \neq \emptyset} \frac{\Gamma(T)}{|T|} \tag{101}$$

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

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



$\Psi(L) = 1/8$



$\Psi(L) = 1$

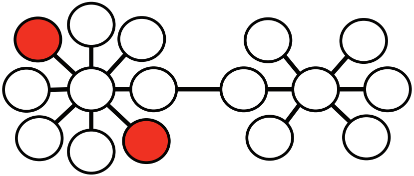
Choosing labels: how to select L

- Consider the following objective

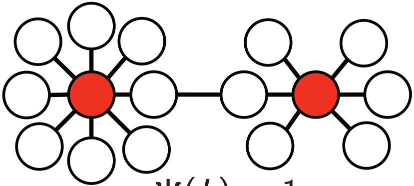
$$\Psi(L) = \min_{T \subseteq V \setminus L : T \neq \emptyset} \frac{\Gamma(T)}{|T|} \tag{101}$$

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

- 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 remaining labels: semi-supervised learning

- Once given labels for L, how to complete the remaining labels?

Choosing remaining labels: semi-supervised learning

- Once given labels for L, how to complete the remaining 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 remaining labels: semi-supervised learning

- Once given labels for L, how to complete the remaining 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 remaining labels: semi-supervised learning

- Once given labels for L, how to complete the remaining 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 remaining labels: semi-supervised learning

- Once given labels for L, how to complete the remaining 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 $A \subseteq V \setminus L$,

$$g(A) = \Gamma(A \cup \{v \in L : y_L(v) = 1\}) \tag{102}$$

Choosing remaining labels: semi-supervised learning

- Once given labels for L, how to complete the remaining 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 $A \subseteq V \setminus L$,

$$g(A) = \Gamma(A \cup \{v \in L : y_L(v) = 1\}) \tag{102}$$

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

Generalized Error Bound

Theorem (Guillory & B., '11)

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

$$\|\hat{y} - y\|_1 \leq 2\frac{\Gamma(Y(y))}{\Psi(L)} \tag{103}$$

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:

$$\Psi(S) = \min_{T \subseteq V \setminus S : T \neq \emptyset} \frac{\Gamma(T)}{|T|} \tag{104}$$

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

Discrete Submodular Divergences

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

Discrete Submodular Divergences

- A convex function parameterizes a Bregman 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 gradient when ϕ is everywhere differentiable), the generalized Bregman divergence is defined as:

$$d_\phi^{\mathcal{H}_\phi}(x, y) = \phi(x) - \phi(y) - \langle \mathcal{H}_\phi(y), x - y \rangle, \forall x, y \in \text{dom}(\phi) \quad (105)$$

Discrete Submodular Divergences

- A convex function parameterizes a Bregman 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 gradient when ϕ is everywhere differentiable), the generalized Bregman divergence is defined as:

$$d_\phi^{\mathcal{H}_\phi}(x, y) = \phi(x) - \phi(y) - \langle \mathcal{H}_\phi(y), x - y \rangle, \forall x, y \in \text{dom}(\phi) \quad (105)$$

- A submodular function parameterizes a discrete submodular Bregman divergence (Iyer & B., 2012).

Discrete Submodular Divergences

- A convex function parameterizes a Bregman 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 gradient when ϕ is everywhere differentiable), the generalized Bregman divergence is defined as:

$$d_\phi^{\mathcal{H}_\phi}(x, y) = \phi(x) - \phi(y) - \langle \mathcal{H}_\phi(y), x - y \rangle, \forall x, y \in \text{dom}(\phi) \quad (105)$$

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

$$d_f^{\mathcal{H}_f}(X, Y) = f(X) - f(Y) - \langle \mathcal{H}_f(Y), 1_X - 1_Y \rangle \quad (106)$$

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

Discrete Submodular Divergences

- A convex function parameterizes a Bregman 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 gradient when ϕ is everywhere differentiable), the generalized Bregman divergence is defined as:

$$d_\phi^{\mathcal{H}_\phi}(x, y) = \phi(x) - \phi(y) - \langle \mathcal{H}_\phi(y), x - y \rangle, \forall x, y \in \text{dom}(\phi) \quad (105)$$

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

$$d_f^{\mathcal{H}_f}(X, Y) = f(X) - f(Y) - \langle \mathcal{H}_f(Y), 1_X - 1_Y \rangle \quad (106)$$

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

Discrete Submodular Divergences

- A convex function parameterizes a Bregman 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 gradient when ϕ is everywhere differentiable), the generalized Bregman divergence is defined as:

$$d_\phi^{\mathcal{H}_\phi}(x, y) = \phi(x) - \phi(y) - \langle \mathcal{H}_\phi(y), x - y \rangle, \forall x, y \in \text{dom}(\phi) \quad (105)$$

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

$$d_f^{\mathcal{H}_f}(X, Y) = f(X) - f(Y) - \langle \mathcal{H}_f(Y), 1_X - 1_Y \rangle \quad (106)$$

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

examples: submodular parameterization

- Combinatorial independence, generalized entropy, and “information” or “complexity” functions (seen above).

examples: submodular parameterization

- Combinatorial independence, generalized entropy, and "information" or "complexity" functions (seen above).
- Simultaneous batch active-learning/semi-supervised learning (Guillory & Bilmes).

examples: submodular parameterization

- Combinatorial independence, generalized entropy, and “information” or “complexity” functions (seen above).
- Simultaneous batch active-learning/semi-supervised learning (Guillory & Bilmes).
- Rank-order based divergences (Submodular Bregman Divergence, and the Lovász-Bregman Divergences) (Iyer & Bilmes, 2013).

examples: submodular parameterization

- Combinatorial independence, generalized entropy, and "information" or "complexity" functions (seen above).
- Simultaneous batch active-learning/semi-supervised learning (Guillory & Bilmes).
- Rank-order based divergences (Submodular Bregman Divergence, and the Lovász-Bregman Divergences) (Iyer & Bilmes, 2013).
- Feature and dictionary selection (Krause & Guestrin, Das & Kempe)

examples: submodular parameterization

- Combinatorial independence, generalized entropy, and "information" or "complexity" functions (seen above).
- Simultaneous batch active-learning/semi-supervised learning (Guillory & Bilmes).
- Rank-order based divergences (Submodular Bregman Divergence, and the Lovász-Bregman Divergences) (Iyer & Bilmes, 2013).
- Feature and dictionary selection (Krause & Guestrin, Das & Kempe)
- Computer vision (Kolmogorov, Boykov, Kohli, Ladicky, Torr, etc.).

examples: submodular parameterization

- Combinatorial independence, generalized entropy, and "information" or "complexity" functions (seen above).
- Simultaneous batch active-learning/semi-supervised learning (Guillory & Bilmes).
- Rank-order based divergences (Submodular Bregman Divergence, and the Lovász-Bregman Divergences) (Iyer & Bilmes, 2013).
- Feature and dictionary selection (Krause & Guestrin, Das & Kempe)
- Computer vision (Kolmogorov, Boykov, Kohli, Ladicky, Torr, etc.).
- Data subset (or core set) selection in machine learning (Lin & Bilmes, Wei & Bilmes). Data summarization, summarizing big redundant data.

examples: submodular parameterization

- Combinatorial independence, generalized entropy, and “information” or “complexity” functions (seen above).
- Simultaneous batch active-learning/semi-supervised learning (Guillory & Bilmes).
- Rank-order based divergences (Submodular Bregman Divergence, and the Lovász-Bregman Divergences) (Iyer & Bilmes, 2013).
- Feature and dictionary selection (Krause & Guestrin, Das & Kempe)
- Computer vision (Kolmogorov, Boykov, Kohli, Ladicky, Torr, etc.).
- Data subset (or core set) selection in machine learning (Lin & Bilmes, Wei & Bilmes). Data summarization, summarizing big redundant data.
- Influence determination in social networks (Kempe, Kleinberg, & Tardos)

Outline: Part 2

5. Submodular Applications in Machine Learning
   - Where is submodularity useful?
6. As a model of diversity, coverage, span, or information
7. As a model of cooperative costs, complexity, roughness, and irregularity
8. As a Parameter for an ML algorithm
9. Itself, as a target for learning
10. Surrogates for optimization and analysis
11. Reading
    - Refs

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:

$$\hat{\mathbf{y}} \in h_{\mathbf{w}}(\bar{\mathbf{x}}) = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} s(\bar{\mathbf{x}}, \mathbf{y}) = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} \mathbf{w}^\intercal \mathbf{f}(\bar{\mathbf{x}}, \mathbf{y}) \tag{107}$$

- 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 Learning of Submodular Mixtures

- Constraints specified in inference form:

$$\underset{\mathbf{w},\xi_t}{\text{minimize}} \quad \frac{1}{T}\sum_t \xi_t + \frac{\lambda}{2}\|\mathbf{w}\|^2 \tag{108}$$

$$\text{subject to} \quad \mathbf{w}^\top \mathbf{f}_t(\mathbf{y}^{(t)}) \geq \max_{\mathbf{y}\in\mathcal{Y}_t}\left(\mathbf{w}^\top \mathbf{f}_t(\mathbf{y}) + \ell_t(\mathbf{y})\right) - \xi_t, \forall t \tag{109}$$

$$\xi_t \geq 0, \forall t. \tag{110}$$

Structured Learning of Submodular Mixtures

- Constraints specified in inference form:

$$\underset{\mathbf{w},\xi_t}{\text{minimize}} \quad \frac{1}{T}\sum_t \xi_t + \frac{\lambda}{2}\|\mathbf{w}\|^2 \tag{108}$$

$$\text{subject to} \quad \mathbf{w}^\top \mathbf{f}_t(\mathbf{y}^{(t)}) \geq \max_{\mathbf{y}\in\mathcal{Y}_t}\left(\mathbf{w}^\top \mathbf{f}_t(\mathbf{y}) + \ell_t(\mathbf{y})\right) - \xi_t, \forall t \tag{109}$$

$$\xi_t \geq 0, \forall t. \tag{110}$$

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

Structured Learning of Submodular Mixtures

- Constraints specified in inference form:

$$\underset{\mathbf{w},\xi_t}{\text{minimize}} \quad \frac{1}{T}\sum_t \xi_t + \frac{\lambda}{2}\|\mathbf{w}\|^2 \tag{108}$$

$$\text{subject to} \quad \mathbf{w}^\top \mathbf{f}_t(\mathbf{y}^{(t)}) \geq \max_{\mathbf{y}\in\mathcal{Y}_t}\left(\mathbf{w}^\top \mathbf{f}_t(\mathbf{y}) + \ell_t(\mathbf{y})\right) - \xi_t, \forall t \tag{109}$$

$$\xi_t \geq 0, \forall t. \tag{110}$$

- Exponential set of constraints reduced to an embedded optimization problem, "loss-augmented inference."
- $\mathbf{w}^\top \mathbf{f}_t(\mathbf{y})$ is a mixture of submodular components.

Structured Learning of Submodular Mixtures

- Constraints specified in inference form:

$$\underset{\mathbf{w},\xi_t}{\text{minimize}} \quad \frac{1}{T}\sum_t \xi_t + \frac{\lambda}{2}\|\mathbf{w}\|^2 \tag{108}$$

$$\text{subject to} \quad \mathbf{w}^\top \mathbf{f}_t(\mathbf{y}^{(t)}) \geq \max_{\mathbf{y}\in\mathcal{Y}_t}\Big(\mathbf{w}^\top \mathbf{f}_t(\mathbf{y}) + \ell_t(\mathbf{y})\Big) - \xi_t, \forall t \tag{109}$$

$$\xi_t \geq 0, \forall t. \tag{110}$$

- Exponential set of constraints reduced to an embedded optimization problem, "loss-augmented inference."
- $\mathbf{w}^\top \mathbf{f}_t(\mathbf{y})$ is a mixture of submodular components.
- If loss is also submodular, then loss-augmented inference is submodular optimization.

Structured Learning of Submodular Mixtures

- Constraints specified in inference form:

$$\underset{\mathbf{w},\xi_t}{\text{minimize}} \quad \frac{1}{T}\sum_t \xi_t + \frac{\lambda}{2}\|\mathbf{w}\|^2 \tag{108}$$

$$\text{subject to} \quad \mathbf{w}^\top \mathbf{f}_t(\mathbf{y}^{(t)}) \geq \max_{\mathbf{y}\in\mathcal{Y}_t}\left(\mathbf{w}^\top \mathbf{f}_t(\mathbf{y}) + \ell_t(\mathbf{y})\right) - \xi_t, \forall t \tag{109}$$

$$\xi_t \geq 0, \forall t. \tag{110}$$

- Exponential set of constraints reduced to an embedded optimization problem, "loss-augmented inference."
- $\mathbf{w}^\top \mathbf{f}_t(\mathbf{y})$ is a mixture of submodular components.
- If loss is also submodular, then loss-augmented inference is submodular optimization.
- If loss is supermodular, this is a difference-of-submodular (DS) function optimization.

Learning Submodular Mixtures: Unconstrained Form

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

$$\min_{\mathbf{w}\geq 0} \frac{1}{T} \sum_t \left[\max_{\mathbf{y}\in\mathcal{Y}_t} \left(\mathbf{w}^\top \mathbf{f}_t(\mathbf{y}) + \ell_t(\mathbf{y}) \right) - \mathbf{w}^\top \mathbf{f}_t(\mathbf{y}^{(t)}) \right] + \frac{\lambda}{2} \|\mathbf{w}\|^2 \quad (111)$$

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

$$\max_{\mathbf{y}\in\mathcal{Y}_t} \left(\mathbf{w}^\top \mathbf{f}_t(\mathbf{y}) + \ell_t(\mathbf{y}) \right) \quad (112)$$

- 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. (112) can be solved at least approximately well.

Structured Prediction: Subgradient

- Subgradient, evaluated at $\mathbf{w}$, of the following

$$\max_{\mathbf{y}\in\mathcal{Y}_t}\left(\mathbf{w}^\top \mathbf{f}_t(\mathbf{y}) + \ell_t(\mathbf{y})\right) - \mathbf{w}^\top \mathbf{f}_t(\mathbf{y}^{(t)}) + \frac{\lambda}{2}\|\mathbf{w}\|^2 \tag{113}$$

can be found by computing or approximating

$$\mathbf{y}^* \in \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}_t}\left(\mathbf{w}^\top \mathbf{f}_t(\mathbf{y}) + \ell_t(\mathbf{y})\right) - \mathbf{w}^\top \mathbf{f}_t(\mathbf{y}^{(t)}) \tag{114}$$

and then finding subgradient of

$$\mathbf{w}^\top \mathbf{f}_t(\mathbf{y}^*) + \ell_t(\mathbf{y}^*) - \mathbf{w}^\top \mathbf{f}_t(\mathbf{y}^{(t)}) + \frac{\lambda}{2}\|\mathbf{w}\|^2 \tag{115}$$

which has the form

$$\mathbf{f}_t(\mathbf{y}^*) - \mathbf{f}_t(\mathbf{y}^{(t)}) + \lambda\mathbf{w}. \tag{116}$$

Structured Prediction: Subgradient Learning

- Solvable with simple sub-gradient descent algorithm using structured variant of hinge-loss (Taskar, 2004).
- Loss-augmented inference is either submodular optimization (Lin & B. 2012) or DS optimization (Tschiatschek, Iyer, & B. 2014).

Algorithm 7: 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: Part 2

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

$$\Pr(x) = \frac{1}{Z}\exp(-E(x)) \tag{117}$$

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

$$\Pr(x) = \frac{1}{Z}\exp(-E(x)) \tag{117}$$

  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

$$\Pr(x) = \frac{1}{Z}\exp(-E(x)) \tag{117}$$

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.

Submodular Analysis for Non-Submodular Problems

- Sometimes the quality of solutions to non-submodular problems can be analyzed via submodularity.

Submodular Analysis for Non-Submodular Problems

- Sometimes the quality of solutions to non-submodular problems can be analyzed via submodularity.
- For example, "deviation from submodularity" can be measured using the submodularity ratio (Das & Kempe):

$$\gamma_{U,k}(f) = \min_{L \subseteq U, S: |S| \le k, S \cap L = \emptyset} \frac{\sum_{s \in S} f(x|L)}{f(S|L)} \tag{118}$$

Submodular Analysis for Non-Submodular Problems

- Sometimes the quality of solutions to non-submodular problems can be analyzed via submodularity.
- For example, "deviation from submodularity" can be measured using the submodularity ratio (Das & Kempe):

$$\gamma_{U,k}(f) = \min_{L \subseteq U, S: |S| \leq k, S \cap L = \emptyset} \frac{\sum_{s \in S} f(x|L)}{f(S|L)} \tag{118}$$

- f is submodular if $\gamma_{U,k} \geq 1$ for all U and k.

Submodular Analysis for Non-Submodular Problems

- Sometimes the quality of solutions to non-submodular problems can be analyzed via submodularity.
- For example, "deviation from submodularity" can be measured using the submodularity ratio (Das & Kempe):

$$\gamma_{U,k}(f) = \min_{L \subseteq U, S : |S| \leq k, S \cap L = \emptyset} \frac{\sum_{s \in S} f(x|L)}{f(S|L)} \tag{118}$$

- f is submodular if $\gamma_{U,k} \geq 1$ for all U and k.
- For some variable selection problems, can get bounds of the form:

$$\text{Solution} \geq \left(1 - \frac{1}{e^{\gamma_{U^*,k}}}\right)\text{OPT} \tag{119}$$

where U^* is the solution set of a variable selection algorithm.

Submodular Analysis for Non-Submodular Problems

- Sometimes the quality of solutions to non-submodular problems can be analyzed via submodularity.
- For example, "deviation from submodularity" can be measured using the submodularity ratio (Das & Kempe):

$$\gamma_{U,k}(f) = \min_{L \subseteq U, S: |S| \leq k, S \cap L = \emptyset} \frac{\sum_{s \in S} f(x|L)}{f(S|L)} \tag{118}$$

- f is submodular if $\gamma_{U,k} \geq 1$ for all U and k.
- For some variable selection problems, can get bounds of the form:

$$\text{Solution} \geq (1 - \frac{1}{e^{\gamma_{U^*,k}}})\text{OPT} \tag{119}$$

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

Outline: Part 2

5. Submodular Applications in Machine Learning
   - Where is submodularity useful?
6. As a model of diversity, coverage, span, or information
7. As a model of cooperative costs, complexity, roughness, and irregularity
8. As a Parameter for an ML algorithm
9. Itself, as a target for learning
10. Surrogates for optimization and analysis
11. Reading
   - Refs

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/. Lectures available 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!