Submodular Functions, Optimization, and Applications to Machine Learning

— Spring Quarter, Lecture 1 —

http://www.ee.washington.edu/people/faculty/bilmes/classes/ee596b_spring_2016/

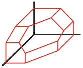
Prof. Jeff Bilmes

University of Washington, Seattle
Department of Electrical Engineering
http://melodi.ee.washington.edu/~bilmes

Mar 28th, 2016

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

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



Announcements

- Welcome to: Submodular Functions, Optimization, and Applications to Machine Learning, EE596B.
- Class: An introduction to submodular functions including methods for their optimization, and how they have been (and can be) applied in many application domains.
- Weekly Office Hours: Mondays, 3:30-4:30, 10 minutes after class ends on Mondays.
- Loew 116, class web page is at our web page (http://www.ee.washington.edu/people/faculty/bilmes/classes/ee596b_spring_2016/).
- Use our discussion board (https://canvas.uw.edu/courses/1039754/discussion_topics) for all questions, comments, so that all will benefit from them being answered.

Rough Class Outline

- Introduction to submodular functions: definitions, real-world and contrived examples, properties, operations that preserve submodularity, inequalities, variants and special submodular functions, and computational properties. Gain intution, when is submodularity and supermodularity useful?
- Applications in data science, computer vision, tractable substructures in constraint satisfaction/SAT and graphical models, game theory, social networks, economics, information theory, structured convex norms, natural language processing, genomics/proteomics, sensor networks, probabilistic inference, and other areas of machine learning.
- Submodularity is an ideal model for cooperation, complexity, and attractiveness as well as for diversity, coverage, & information

Rough Class Outline (cont. II)

- theory of matroids and lattices.
- Polyhedral properties of submodular functions, polymatroids generalize matroids.
- The Lovász extension of submodular functions, the Choquet integral, and convex and concave extensions.
- Submodular maximization algorithms under constraints, submodular cover problems, greedy algorithms, approximation guarantees.
- Submodular minimization algorithms, a history of submodular minimization, including both numerical and combinatorial algorithms, computational properties, and descriptions of both known results and currently open problems in this area.
- Submodular flow problems, the principle partition of a submodular function and its variants.

Rough Class Outline (cont. III)

- Constrained optimization problems with submodular functions, including maximization and minimization problems with various constraints. An overview of recent problems addressed in the community.

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.

Useful 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.
- Additional readings that will be announced here.

Recent online material (some with an ML slant)

- Previous version of this class http://j.ee.washington.edu/~bilmes/classes/ee596a_fall_2014/.
- Stefanie Jegelka & Andreas Krause's 2013 ICML tutorial http://techtalks.tv/talks/submodularity-in-machine-learning-new-directions-part-i/58125/
- NIPS, 2013 tutorial on submodularity http://melodi.ee.washington.edu/~bilmes/pgs/b2hd-bilmes2013-nips-tutorial.html and http://youtu.be/c4rBof38nKQ
- Andreas Krause's web page http://submodularity.org.
- 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

Facts about the class

- Prerequisites: ideally knowledge in probability, statistics, convex optimization, and combinatorial optimization these will be reviewed as necessary. The course is open to students in all UW departments. Any questions, please contact me.
- Text: We will be drawing from the book by Satoru Fujishige entitled "Submodular Functions and Optimization" 2nd Edition, 2005, but we will also be reading research papers that will be posted here on this web page, especially for some of the application areas.
- Grades and Assignments: Grades will be based on a combination of a final project (45%), homeworks (55%). There will be between 3-6 homeworks during the quarter.
- Final project: The final project will consist of a 4-page paper (conference style) and a final project presentation. The project must involve using/dealing mathematically with submodularity in some way or another, and might involve a contest!

Facts about the class

- Homework must be submitted electronically using our assignment dropbox (https://canvas.uw.edu/courses/1039754/assignments). PDF submissions only please. Photos of neatly hand written solutions, combined into one PDF, are fine
- Lecture slides - are being updated and improved this quarter. They will likely appear on the web page the night before, and the final version will appear just before class.
- Slides from previous version of this class are at http://j.ee.washington.edu/~bilmes/classes/ee596a_fall_2014/.

Cumulative Outstanding Reading

- Read chapter 1 from Fujishige's book.

Class Road Map - IT-I

- L1(3/28): Motivation, Applications, & Basic Definitions
- L2(3/30):
- L3(4/4):
- L4(4/6):
- L5(4/11):
- L6(4/13):
- L7(4/18):
- L8(4/20):
- L9(4/25):
- L10(4/27):
- L11(5/2):
- L12(5/4):
- L13(5/9):
- L14(5/11):
- L15(5/16):
- L16(5/18):
- L17(5/23):
- L18(5/25):
- L19(6/1):
- L20(6/6): Final Presentations maximization.

Finals Week: June 6th-10th, 2016.

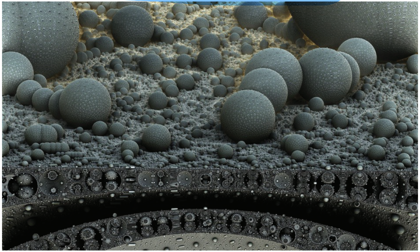
Machine Learning and Machine Intelligence

- Machine learning: our acknowledgement that humans might might be intelligent enough only to produce intelligent machines indirectly
- This is yet another instance of "All problems in computer science can be solved by another level of indirection" David Wheeler.
- Progress: natural language processing (NLP), computer vision, robotics, smart homes, genomics/proteomics, and game playing (e.g., GO).
- Promise: education, poverty, energy/climate change, and health.

The Ideal Machine Learning Methods

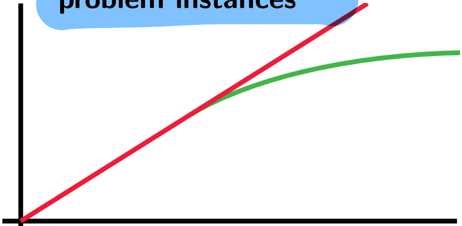
- Simple to define

- Mathematically rich

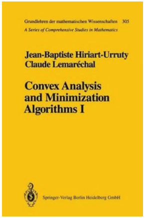


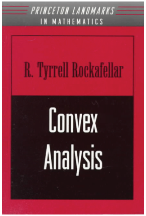
- Naturally suited to many real-world applications

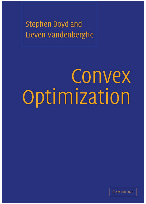
- Efficient & scalable to large problem instances

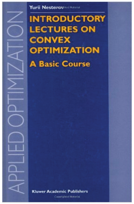


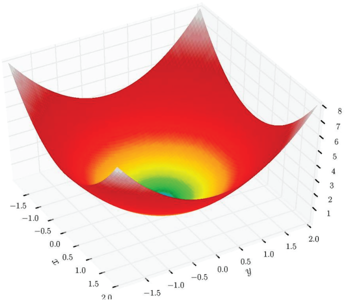
Convex Analysis in Machine Learning

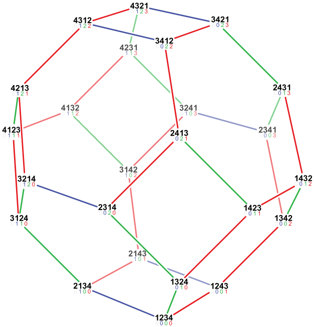












Successful Convexity in Machine Learning

- Linear and logistic regresion, surrogate loss functions.
- Convex sparse regularizers (such as the ℓ_p family and nuclear norms).
- PSD matrices (i.e., positive semidefinite cone) and Gaussian densities.
- Optimizing non-linear and even non-convex classification/regression methods such as support-vector (SVMs) and kernel machines via convex optimization.
- Maximum entropy estimation
- The expectation-maximization (EM) algorithm.
- Ideas/techniques/insight for non-convex methods, convex minimization useful even for non-convex problems, such as Deep Neural Networks (DNNs).

A Convexity Limitation: Discrete Problems

Many Machine Learning problems are **inherently discrete:**

- Active learning/label selection.
- MAP & diverse k-best discrete probabilistic inference
- Data Science: data partitioning, clustering, selection; data summarization; the science of data management.
- Sparse modeling, compressed sensing, low-rank approximation.
- Graphical models structure learning
- Variable, feature, and data selection; dictionary selection.
- Natural language processing (NLP): words, phrases, sentences, paragraphs, n-grams, syntax trees, graphs, semantic structures.
- Social choice and voting theory, social networks, viral marketing,
- Multi-label image segmentation in computer vision
- Proteomics: selecting or identifying peptides, proteins, drug trial participants
- Genomics: cell-type or assay selection, genomic summarization

Might not always be perfectly satisfied with only convex functions.

More Examples: Discrete Optimization Problems

- **Combinatorial Problems**: e.g., set cover, max k coverage, vertex cover, edge cover, graph cut problems.
- **Operations Research/Industrial Engineering**: facility and factory location, packing and covering.
- **Sensor placement** where to optimally place sensors?
- **Information:** Information gain and feature selection, information theory
- **Mathematics:** e.g., monge matrices, efficient dynamic programming
- **Networks**: Social networks, influence, viral marketing, information cascades, diffusion networks
- **Algorithms**: limits of polynomial time complexity
- **Diversity** and its models, subset selection, data summarization
- **Economics**: markets, economies of scale, mathematics of supply & demand

General Integer Programming (e.g., Integer Linear Programming (ILP), Integer Quadratic Programming (IQP), etc), but general case might ignore important, useful, and natural structures common to many problems.

Attractions of Convex Functions

Why do we like Convex Functions? (Quoting Lovász 1983):

1. *Convex functions occur in many mathematical models in economy, engineering, and other sciences. Convexity is a very natural property of various functions and domains occurring in such models; quite often the only non-trivial property which can be stated in general.*

Attractions of Convex Functions

Why do we like Convex Functions? (Quoting Lovász 1983):

1. *Convex functions occur in many mathematical models in economy, engineering, and other sciences. Convexity is a very natural property of various functions and domains occurring in such models; quite often the only non-trivial property which can be stated in general.*
2. *Convexity is preserved under many natural operations and transformations, and thereby the effective range of results can be extended, elegant proof techniques can be developed as well as unforeseen applications of certain results can be given.*

Attractions of Convex Functions

Why do we like Convex Functions? (Quoting Lovász 1983):

1. *Convex functions occur in many mathematical models in economy, engineering, and other sciences. Convexity is a very natural property of various functions and domains occurring in such models; quite often the only non-trivial property which can be stated in general.*
2. *Convexity is preserved under many natural operations and transformations, and thereby the effective range of results can be extended, elegant proof techniques can be developed as well as unforeseen applications of certain results can be given.*
3. *Convex functions and domains exhibit sufficient structure so that a mathematically beautiful and practically useful theory can be developed.*

Attractions of Convex Functions

Why do we like Convex Functions? (Quoting Lovász 1983):

1. *Convex functions occur in many mathematical models in economy, engineering, and other sciences. Convexity is a very natural property of various functions and domains occurring in such models; quite often the only non-trivial property which can be stated in general.*
2. *Convexity is preserved under many natural operations and transformations, and thereby the effective range of results can be extended, elegant proof techniques can be developed as well as unforeseen applications of certain results can be given.*
3. *Convex functions and domains exhibit sufficient structure so that a mathematically beautiful and practically useful theory can be developed.*
4. *There are theoretically and practically (reasonably) efficient methods to find the minimum of a convex function.*

Attractions of Submodular Functions

- In this course, we wish to demonstrate that submodular and supermodular functions also possess attractions of these four sorts as well.

Attractions of Submodular Functions

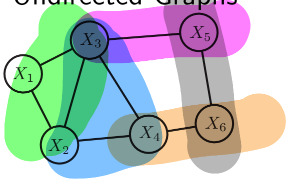
- In this course, we wish to demonstrate that submodular and supermodular functions also possess attractions of these four sorts as well.
- Next we consider graphical models. Can't they provide useful structural properties that make many discrete problems easy?

Graphical Models and Decomposition

- Let $\mathcal{B}$ be the set of cliques of a graph G. A graphical model prescribes how to write functions $f : \{0,1\}^n \to \mathbb{R}$. Let $x \in \{0,1\}^n$

$$f(x) = \sum_{B \in \mathcal{B}} f_B(x_B) \tag{1.1}$$

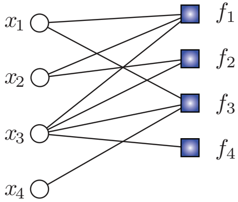
Example: Undirected Graphs



$$f(x_{1:6}) = f(x_1, x_2, x_3) + f(x_2, x_3, x_4) + f(x_3, x_5) + f(x_5, x_6) + f(x_4, x_6)$$

$$f(x_{1:6}) = f(x_1, x_2) + f(x_2, x_3) + f(x_3, x_1) + f(x_2, x_3) + f(x_3, x_4) + f(x_4, x_2) + f(x_3, x_5) + f(x_5, x_6) + f(x_4, x_6)$$

Example: Factor/Hyper Graphs



$$f(x_{1:4}) = f_1(x_1, x_2, x_3) + f_2(x_2, x_3) = f_3(x_1, x_3, x_4) + f_4(x_3)$$

Graphical Models/Decomposition: Real-Object Example

- How to valuate a set of items?

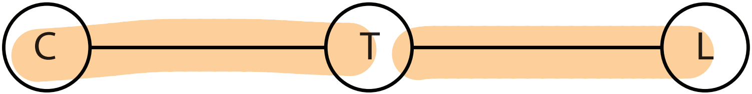
Graphical Models/Decomposition: Real-Object Example

- How to valuate a set of items?
- Let C, T, and L be binary variables indicating the presence or absence of items, and we wish to compute value(C, T, L).

Graphical Models/Decomposition: Real-Object Example

- How to valuate a set of items?
- Let C, T, and L be binary variables indicating the presence or absence of items, and we wish to compute value(C, T, L).
- Example: Value of Coffee (C), Tea (T), and Lemon (L).



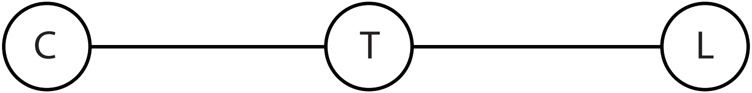


$$\text{value}(C, T, L) = \text{value}(C, T) + \text{value}(T, L) \tag{1.2}$$

Graphical Decomposition Limitation: Manner of Interaction

- Value of Coffee (C), Tea (T), and Lemon (L).



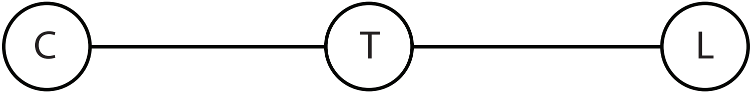


$$\text{value}(C, T, L) = \text{value}(C, T) + \text{value}(T, L) \quad (1.3)$$

Graphical Decomposition Limitation: Manner of Interaction

- Value of Coffee (C), Tea (T), and Lemon (L).





$$\text{value}(C, T, L) = \text{value}(C, T) + \text{value}(T, L) \quad (1.3)$$

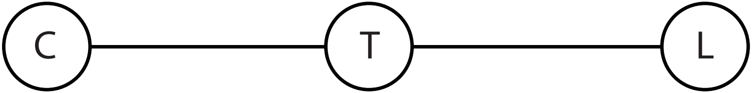
- Coffee and Tea are "substitutive"

$$\text{value}(C, T) \leq \text{value}(C) + \text{value}(T) \quad (1.4)$$

Graphical Decomposition Limitation: Manner of Interaction

- Value of Coffee (C), Tea (T), and Lemon (L).



$$\text{value}(C,T,L) = \text{value}(C,T) + \text{value}(T,L) \quad (1.3)$$

- Coffee and Tea are "substitutive"

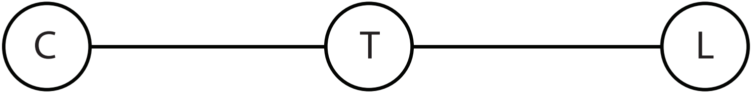
$$\text{value}(C,T) \leq \text{value}(C) + \text{value}(T) \quad (1.4)$$

- Tea and Lemon are "complementary"

$$\text{value}(T,L) \geq \text{value}(T) + \text{value}(L) \quad (1.5)$$

Graphical Decomposition Limitation: Manner of Interaction

- Value of Coffee (C), Tea (T), and Lemon (L).



$$\text{value}(C,T,L) = \text{value}(C,T) + \text{value}(T,L) \tag{1.3}$$

- Coffee and Tea are "substitutive"

$$\text{value}(C,T) \leq \text{value}(C) + \text{value}(T) \tag{1.4}$$

- Tea and Lemon are "complementary"

$$\text{value}(T,L) \geq \text{value}(T) + \text{value}(L) \tag{1.5}$$

- These are distinct non-graphically expressed manners of interaction!

Options for Cost Models via Graphical Decomposition

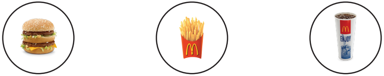
- Three items. Hamburger (H), Fries (F), Soda (S)

Options for Cost Models via Graphical Decomposition

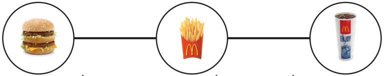
- Three items. Hamburger (H), Fries (F), Soda (S)



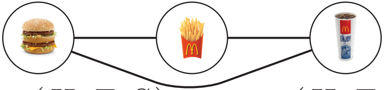
- Some graphical model options for $\text{costs}(H, F, S)$:



$$\text{costs}(H, F, S) = \text{cst}_{\text{h}}(H) + \text{cst}_{\text{f}}(F) + \text{cst}_{\text{c}}(S)$$



$$\text{costs}(H, F, S) = \text{cst}_{\text{hf}}(H, F) + \text{cst}_{\text{fc}}(F, S)$$



$$\text{costs}(H, F, S) = \text{cst}_{\text{hfc}}(H, F, S)$$

Decompositions via Manner of Interaction

- costs(H, F, S) of Hamburger (H), Fries (F), Soda (S)



Consider components of cost: consumer-costs (ccs) and health-costs (hcs), each of which is ternary.

$$\text{costs}(H, F, S) = \text{ccs}(H, F, S) + \text{hcs}(H, F, S) \tag{1.6}$$

Decompositions via Manner of Interaction

- costs(H, F, S) of Hamburger (H), Fries (F), Soda (S)

Consider components of cost: consumer-costs (ccs) and health-costs (hcs), each of which is ternary.

$$\text{costs}(H, F, S) = \text{ccs}(H, F, S) + \text{hcs}(H, F, S) \tag{1.6}$$

- Consumer costs

ccs () − ccs () ≥ ccs () − ccs ()

Decompositions via Manner of Interaction

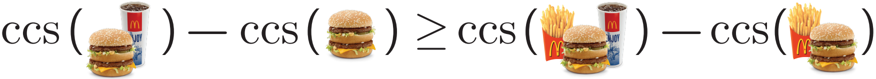
- costs(H, F, S) of Hamburger (H), Fries (F), Soda (S)



Consider components of cost: consumer-costs (ccs) and health-costs (hcs), each of which is ternary.

$$\text{costs}(H, F, S) = \text{ccs}(H, F, S) + \text{hcs}(H, F, S) \quad (1.6)$$

- Consumer costs



- Health costs

Decompositions via Manner of Interaction

- costs(H, F, S) of Hamburger (H), Fries (F), Soda (S)



Consider components of cost: consumer-costs (ccs) and health-costs (hcs), each of which is ternary.

$$\text{costs}(H, F, S) = \text{ccs}(H, F, S) + \text{hcs}(H, F, S) \tag{1.6}$$

- Consumer costs

ccs() − ccs() ≥ ccs() − ccs()

- Health costs

hcs() − hcs() ≤ hcs() − hcs()

- In both cases, graphical-only decompositions fail!

Sets and set functions $f : 2^V \to \mathbb{R}$

We are given a finite "ground" set V of objects, $2^V \triangleq \{A : A \subseteq V\}$

$V = \{$

$\}$

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

Subset $A \subseteq V$ of objects:

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

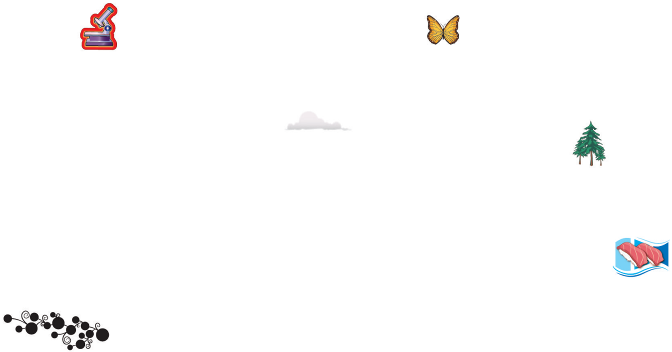
Also given a set function $f : 2^V \to \mathbb{R}$ that valuates subsets $A \subseteq V$.
Ex: $f(A) = 1$

Sets and set functions $f : 2^V \to \mathbb{R}$

$n = |V|$

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: $f(B) = 6$

Set functions are pseudo-Boolean functions

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

Set functions are pseudo-Boolean functions

- Any set $A \subseteq V$ can be represented as a binary vector $x \in \{0,1\}^V$ (a "bit vector" representation of a set).
- The characteristic vector $\mathbf{1}_A \in \{0,1\}^V$ of a set A is defined one where element $v \in V$ has value:

$$\mathbf{1}_A(v) = \begin{cases} 1 & \text{if } v \in A \\ 0 & else \end{cases} \tag{1.7}$$

Set functions are pseudo-Boolean functions

- Any set $A \subseteq V$ can be represented as a binary vector $x \in \{0,1\}^V$ (a "bit vector" representation of a set).
- The characteristic vector $\mathbf{1}_A \in \{0,1\}^V$ of a set A is defined one where element $v \in V$ has value:

$$\mathbf{1}_A(v) = \begin{cases} 1 & \text{if } v \in A \\ 0 & else \end{cases} \tag{1.7}$$

- Useful to be able to quickly map between $X = X(\mathbf{1}_X)$ and $x(X) \triangleq \mathbf{1}_X$.

Set functions are pseudo-Boolean functions

- Any set $A \subseteq V$ can be represented as a binary vector $x \in \{0,1\}^V$ (a "bit vector" representation of a set).
- The characteristic vector $\mathbf{1}_A \in \{0,1\}^V$ of a set A is defined one where element $v \in V$ has value:

$$\mathbf{1}_A(v) = \begin{cases} 1 & \text{if } v \in A \\ 0 & else \end{cases} \tag{1.7}$$

- Useful to be able to quickly map between $X = X(\mathbf{1}_X)$ and $x(X) \triangleq \mathbf{1}_X$.
- $f : \{0,1\}^V \to \{0,1\}$ are known as Boolean function.

Set functions are pseudo-Boolean functions

- Any set $A \subseteq V$ can be represented as a binary vector $x \in \{0,1\}^V$ (a "bit vector" representation of a set).
- The characteristic vector $\mathbf{1}_A \in \{0,1\}^V$ of a set A is defined one where element $v \in V$ has value:

$$\mathbf{1}_A(v) = \begin{cases} 1 & \text{if } v \in A \\ 0 & else \end{cases} \tag{1.7}$$

- Useful to be able to quickly map between $X = X(\mathbf{1}_X)$ and $x(X) \triangleq \mathbf{1}_X$.
- $f : \{0,1\}^V \to \{0,1\}$ are known as Boolean function.
- $f : \{0,1\}^V \to \mathbb{R}$ is a pseudo-Boolean function (submodular functions are a special case).

Two Equivalent Submodular Definitions

Definition 1.3.1 (submodular concave)

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

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B) \quad (1.8)$$

An alternate and (as we will soon see) equivalent definition is:

Definition 1.3.2 (diminishing returns)

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

$$f(A \cup \{v\}) - f(A) \geq f(B \cup \{v\}) - f(B) \quad (1.9)$$

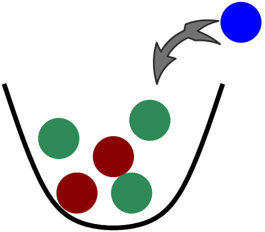
The incremental "value", "gain", or "cost" of v decreases (diminishes) as the context in which v is considered grows from A to B.

Example Submodular: Number of Colors of Balls in Urns

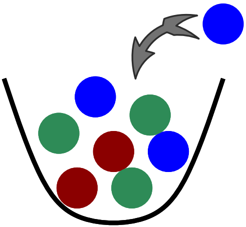
- Consider an urn containing colored balls. Given a set S of balls, $f(S)$ counts the number of distinct colors in S.

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 in S.



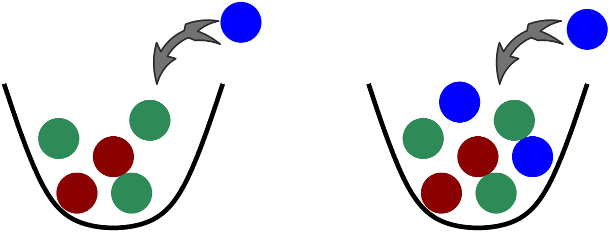
Initial value: 2 (colors in urn).
New value with added blue ball: 3



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 in S.



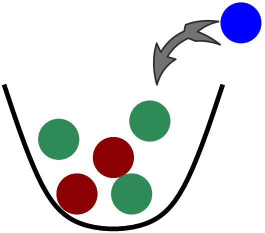
Initial value: 2 (colors in urn).
New value with added blue ball: 3

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

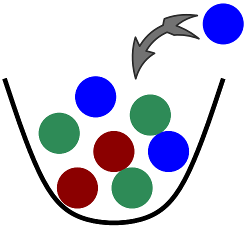
- Submodularity: Incremental Value of Object Diminishes in a Larger Context (diminishing returns).

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 in S.



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 Supermodular Definitions

Definition 1.3.3 (supermodular)

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

$$f(A) + f(B) \leq f(A \cup B) + f(A \cap B) \quad (1.10)$$

Definition 1.3.4 (supermodular (improving returns))

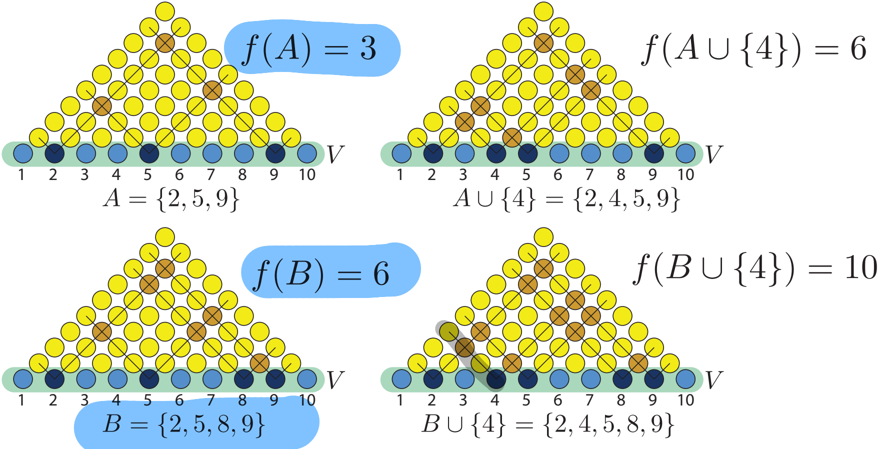
A function $f : 2^V \to \mathbb{R}$ is supermodular if for any $A \subseteq B \subset V$, and $v \in V \setminus B$, we have that:

$$f(A \cup \{v\}) - f(A) \leq f(B \cup \{v\}) - f(B) \quad (1.11)$$

- 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.
- If f both submodular and supermodular, then f is said to be **modular**, and $f(A) = c + \sum_{a \in A} f(a)$ (often $c = 0$).

Example Supermodular: Number of Balls with Two Lines

Given ball pyramid, bottom row V is size $n = |V|$. For subset $S \subseteq V$ of bottom-row balls, draw 45° and 135° diagonal lines from each $s \in S$. Let $f(S)$ be number of non-bottom-row balls with two lines $\Rightarrow f(S)$ is supermodular.



Scientific Anecdote: Emergent Properties

New York Times column (D. Brooks), March 28th, 2011 on "Tools for Thinking" was about responses to Steven Pinker's (Harvard) asking a number of scientists "What scientific concept would improve everybody's cognitive toolkit?"

See `http://edge.org/responses/what-scientific-concept-would-improve-everybodys-cognitive-toolkit`

A common theme was "emergent properties" or "emergent systems"

> *Emergent systems are ones in which many different elements interact. The pattern of interaction then produces a new element that is greater than the sum of the parts, which then exercises a top-down influence on the constituent elements.*

Emergent properties are well modeled by supermodular functions!

Submodular-Supermodular Decomposition

- As an alternative to graphical decomposition, we can decompose a function without resorting sums of local terms.

Submodular-Supermodular Decomposition

- As an alternative to graphical decomposition, we can decompose a function without resorting sums of local terms.

Theorem 1.3.5 (Additive Decomposition (Narasimhan & Bilmes, 2005))

Let $h : 2^V \to \mathbb{R}$ be **any** *set function. Then there exists a submodular function $f : 2^V \to \mathbb{R}$ and a supermodular function $g : 2^V \to \mathbb{R}$ such that h may be additively decomposed as follows: For all $A \subseteq V$,*

$$h(A) = f(A) + g(A) \tag{1.12}$$

Submodular-Supermodular Decomposition

- As an alternative to graphical decomposition, we can decompose a function without resorting sums of local terms.

Theorem 1.3.5 (Additive Decomposition (Narasimhan & Bilmes, 2005))

Let $h : 2^V \to \mathbb{R}$ be **any** *set function. Then there exists a submodular function $f : 2^V \to \mathbb{R}$ and a supermodular function $g : 2^V \to \mathbb{R}$ such that h may be additively decomposed as follows: For all $A \subseteq V$,*

$$h(A) = f(A) + g(A) \tag{1.12}$$

- For many applications (as we will see), either the submodular or supermodular component is naturally zero.

Submodular-Supermodular Decomposition

- As an alternative to graphical decomposition, we can decompose a function without resorting sums of local terms.

Theorem 1.3.5 (Additive Decomposition (Narasimhan & Bilmes, 2005))

Let $h : 2^V \to \mathbb{R}$ be **any** *set function. Then there exists a submodular function $f : 2^V \to \mathbb{R}$ and a supermodular function $g : 2^V \to \mathbb{R}$ such that h may be additively decomposed as follows: For all $A \subseteq V$,*

$$h(A) = f(A) + g(A) \tag{1.12}$$

- For many applications (as we will see), either the submodular or supermodular component is naturally zero.
- Sometimes more natural than a graphical decomposition.

Submodular-Supermodular Decomposition

- As an alternative to graphical decomposition, we can decompose a function without resorting sums of local terms.

Theorem 1.3.5 (Additive Decomposition (Narasimhan & Bilmes, 2005))

Let $h : 2^V \to \mathbb{R}$ be **any** *set function. Then there exists a submodular function $f : 2^V \to \mathbb{R}$ and a supermodular function $g : 2^V \to \mathbb{R}$ such that h may be additively decomposed as follows: For all $A \subseteq V$,*

$$h(A) = f(A) + g(A) \tag{1.12}$$

- For many applications (as we will see), either the submodular or supermodular component is naturally zero.
- Sometimes more natural than a graphical decomposition.
- Sometimes $h(A)$ has structure in terms of submodular functions but is non additively decomposed (one example is $h(A) = f(A)/g(A)$).

Submodular-Supermodular Decomposition

- As an alternative to graphical decomposition, we can decompose a function without resorting sums of local terms.

Theorem 1.3.5 (Additive Decomposition (Narasimhan & Bilmes, 2005))

Let $h : 2^V \to \mathbb{R}$ be **any** *set function. Then there exists a submodular function $f : 2^V \to \mathbb{R}$ and a supermodular function $g : 2^V \to \mathbb{R}$ such that h may be additively decomposed as follows: For all $A \subseteq V$,*

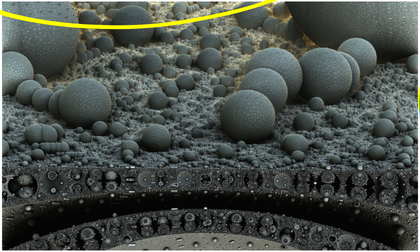
$$h(A) = f(A) + g(A) \tag{1.12}$$

- For many applications (as we will see), either the submodular or supermodular component is naturally zero.
- Sometimes more natural than a graphical decomposition.
- Sometimes $h(A)$ has structure in terms of submodular functions but is non additively decomposed (one example is $h(A) = f(A)/g(A)$).
- Complementary: simultaneous graphical/submodular-supermodular decomposition (i.e., submodular + supermodular tree).

The Ideal Machine Learning Methods

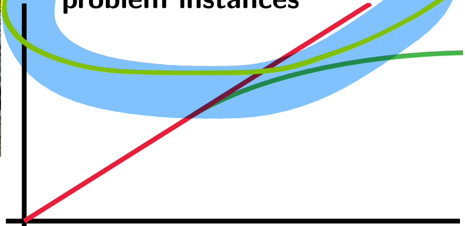
- Simple to define

- Mathematically rich



- Naturally suited to many real-world applications

- Efficient & scalable to large problem instances



Discrete Optimization

- Unconstrained minimization and maximization:

$$\min_{X \subseteq V} f(X) \qquad (1.13)$$

$$\max_{X \subseteq V} f(X) \qquad (1.14)$$

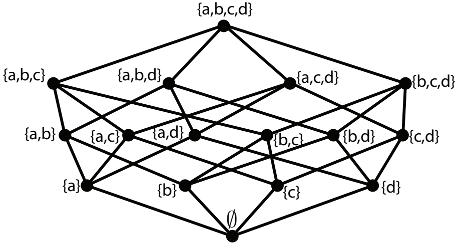
Discrete Optimization

- Unconstrained minimization and maximization:

$$\min_{X \subseteq V} f(X) \quad (1.13)$$

$$\max_{X \subseteq V} f(X) \quad (1.14)$$

- Knowing nothing about f, need 2^n queries for any quality assurance on candidate solution. Otherwise, solution can be unboundedly poor!!

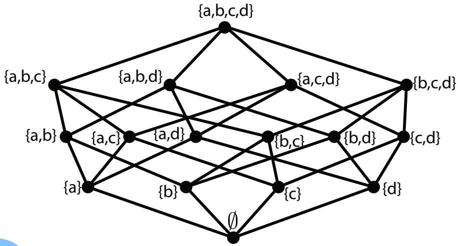


Discrete Optimization

- Unconstrained minimization and maximization:

$$\min_{X \subseteq V} f(X) \quad (1.13) \qquad \max_{X \subseteq V} f(X) \quad (1.14)$$

- Knowing nothing about f, need 2^n queries for any quality assurance on candidate solution. Otherwise, solution can be unboundedly poor!!



- Alternatively, we may partition V into (necessarily disjoint) blocks $\{V_1, V_2, \ldots\}$ that collectively are good in some way.

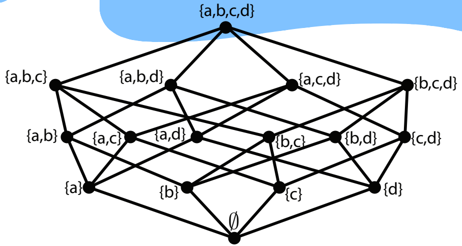
Discrete Optimization

- Unconstrained minimization and maximization:

$$\min_{X \subseteq V} f(X) \quad (1.13)$$

$$\max_{X \subseteq V} f(X) \quad (1.14)$$

- Knowing nothing about f, need 2^n queries for any quality assurance on candidate solution. Otherwise, solution can be unboundedly poor!!



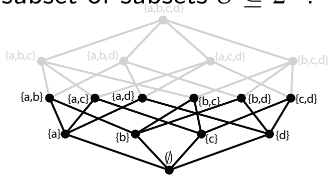
- Alternatively, we may partition V into (necessarily disjoint) blocks $\{V_1, V_2, \ldots\}$ that collectively are good in some way.
- When f is submodular, however, Eq. (1.13) is polytime, and Eq. (1.14) is constant-factor approximable. Partitionings are also approximable!

Constrained Discrete Optimization

- Constrained case: interested only in a subset of subsets $\mathcal{S} \subseteq 2^V$.

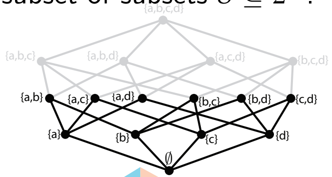
Constrained Discrete Optimization

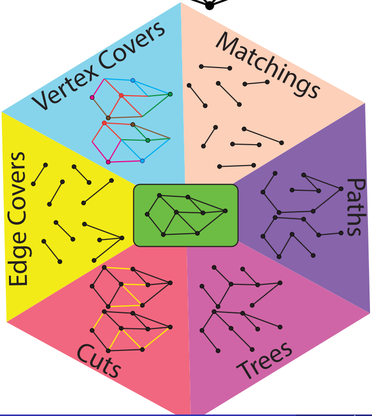
- Constrained case: interested only in a subset of subsets $\mathcal{S} \subseteq 2^V$.
- Ex: Bounded size $\mathcal{S} = \{S \subseteq V : |S| \leq k\}$, or given cost vector w and budget, bounded cost $\{S \subseteq V : \sum_{s \in S} w(s) \leq b\}$.



Constrained Discrete Optimization

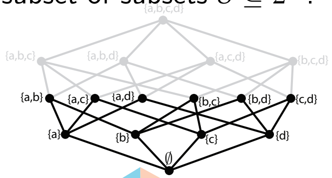
- Constrained case: interested only in a subset of subsets $\mathcal{S} \subseteq 2^V$.
- Ex: Bounded size $\mathcal{S} = \{S \subseteq V : |S| \leq k\}$, or given cost vector w and budget, bounded cost $\{S \subseteq V : \sum_{s \in S} w(s) \leq b\}$.
- Ex: feasible sets $\mathcal{S}$ as combinatorial objects



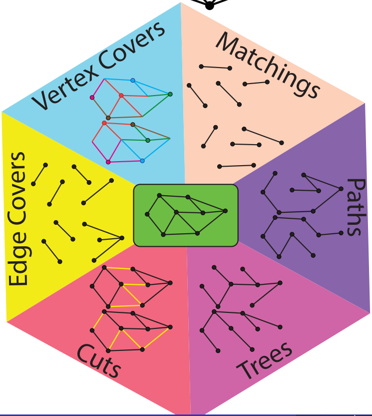


Constrained Discrete Optimization

- Constrained case: interested only in a subset of subsets $\mathcal{S} \subseteq 2^V$.
- Ex: Bounded size $\mathcal{S} = \{S \subseteq V : |S| \leq k\}$, or given cost vector w and budget, bounded cost $\{S \subseteq V : \sum_{s \in S} w(s) \leq b\}$.

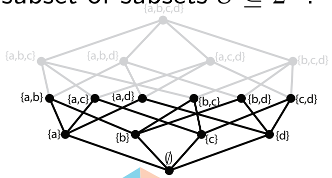


- Ex: feasible sets $\mathcal{S}$ as combinatorial objects
- Ex: feasible sets $\mathcal{S}$ as matroids.

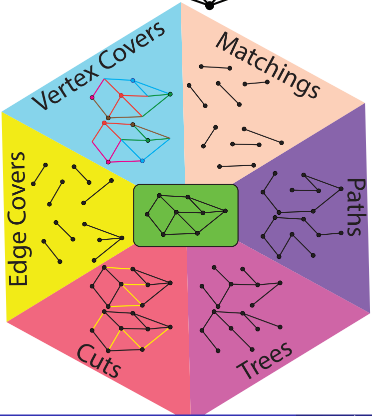


Constrained Discrete Optimization

- Constrained case: interested only in a subset of subsets $\mathcal{S} \subseteq 2^V$.
- Ex: Bounded size $\mathcal{S} = \{S \subseteq V : |S| \leq k\}$, or given cost vector w and budget, bounded cost $\{S \subseteq V : \sum_{s \in S} w(s) \leq b\}$.



- Ex: feasible sets $\mathcal{S}$ as combinatorial objects
- Ex: feasible sets $\mathcal{S}$ as matroids.
- Ex: feasible sets $\mathcal{S}$ as 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

- Constrained discrete optimization problems:

$$\begin{array}{ll} \text{maximize} & f(S) \\ \text{subject to} & S \in \mathcal{S} \end{array} \quad (1.15) \qquad \begin{array}{ll} \text{minimize} & f(S) \\ \text{subject to} & S \in \mathcal{S} \end{array} \quad (1.16)$$

where $\mathcal{S} \subseteq 2^V$ is the feasible set of sets.

Constrained Discrete Optimization

- Constrained discrete optimization problems:

$$\begin{array}{ll}\text{maximize} & f(S)\\ \text{subject to} & S \in \mathcal{S}\end{array} \quad (1.15) \qquad \begin{array}{ll}\text{minimize} & f(S)\\ \text{subject to} & S \in \mathcal{S}\end{array} \quad (1.16)$$

  where $\mathcal{S} \subseteq 2^V$ is the feasible set of sets.
- Fortunately, when f (and g) are submodular, these problems can often be solved with guarantees, often very efficiently! *(Feige, Mirrokni & Vondrák 20XX; Goel, Karande, Tripathi & Wang; Svitkina & Fleischer 2010; Jegelka & Bilmes 2011, Iyer, Jegelka, & Bilmes 2013, Iyer & Bilmes 2014, and* **many many** *others).*

Submodular and Supermodular Applications

- Algorithms: Algorithms can be developed that often are tractable (and as we will see many in this class).
- Applications: There are many seemingly different applications that are strongly related to submodularity.
- Submodularity and supermodularity is as common and natural for discrete problems in machine learning as is convexity/concavity for continuous problems.
- First, lets look at a few more very simple examples of submodular functions.

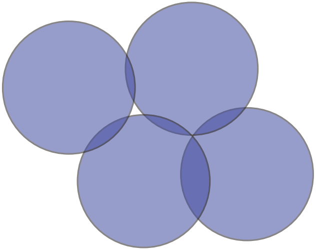
Continuous Set Cover

The area of the union of areas indexed by A

- Let V be a set of indices, and each $v \in V$ indexes a given fixed sub-area of some region in $\mathbb{R}^2$.
- Let area(v) be the area corresponding to item v.
- Let $f(S) = \bigcup_{s \in S}$ area(s) be the union of the areas indexed by elements in S.
- Then $f(S)$ is submodular, and corresponds to a continuous set cover function.

Continuous Set Cover

The area of the union of areas indexed by A — Example

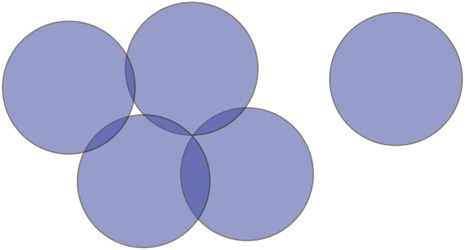


Union of areas of elements of A is given by:

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

Continuous Set Cover

The area of the union of areas indexed by A — Example

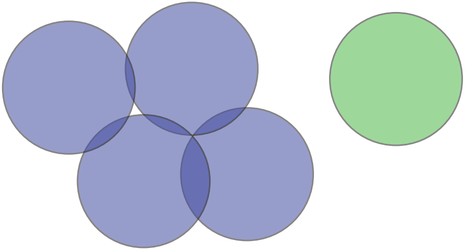


Area of A along with with v:

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

Continuous Set Cover

The area of the union of areas indexed by A — Example



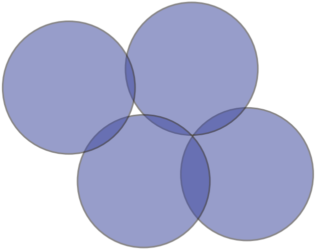
Gain (value) of v in context of A:

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

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

Continuous Set Cover

The area of the union of areas indexed by A — Example

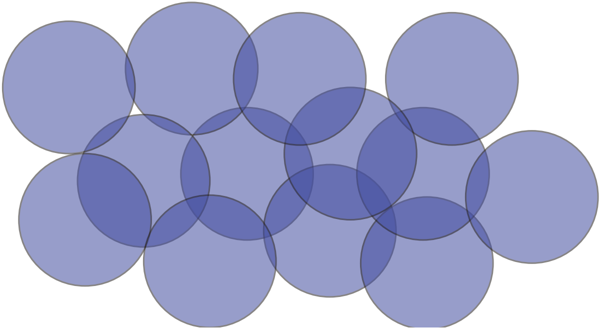


Area of A once again.

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

Continuous Set Cover

The area of the union of areas indexed by A — Example

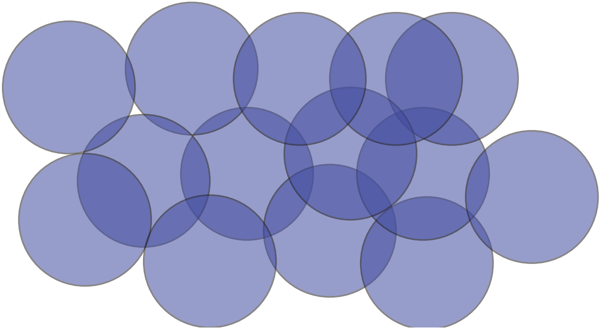


Union of areas of elements of $B \supset A$, where v is not included:

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

Continuous Set Cover

The area of the union of areas indexed by A — Example

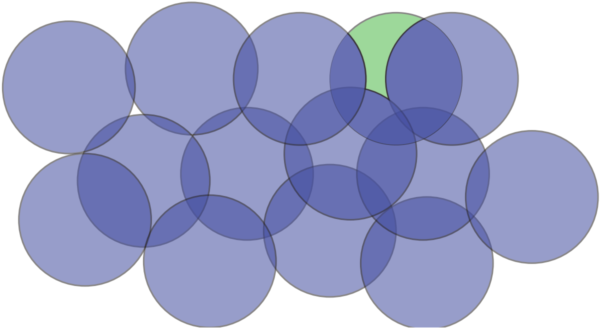


Area of B now also including v:

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

Continuous Set Cover

The area of the union of areas indexed by A — Example



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 $B \supseteq A$.

Simple Consumer 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 Consumer Costs

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

Simple Consumer 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:

$$m(A) = \sum_{a \in A} m(a), \tag{1.17}$$

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

Simple Consumer 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

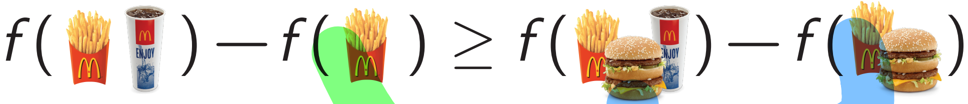
- Grocery store: finite set of items V that one can purchase.
- Each item $v \in V$ has a price $m(v)$.
- Basket of groceries $A \subseteq V$ costs:

$$m(A) = \sum_{a \in A} m(a), \tag{1.17}$$

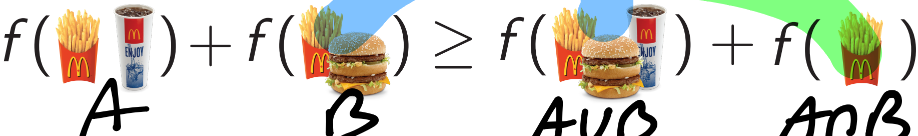
  the sum of individual item costs (no two-for-one discounts).
- This is known as a modular function.

Discounted Consumer Costs (as we saw earlier)

- 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 $A \subseteq V$.We get diminishing returns:



- Simply rearranging terms, we get the other definition of submodularity:



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

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: $f(v_1) + f(v_2) \geq f(v_1, v_2) + f(\emptyset)$

Markets: Supply Side Economies of scale

- Economies of Scale: Many goods and services can be produced at a much lower per-unit cost only if they are produced in very large quantities.
- The profit margin for producing a unit of goods is improved as more of those goods are created.
- If you already make a good, making a similar good is easier than if you start from scratch (e.g., Apple making both iPod and iPhone).
- An argument in favor of free trade is that it opens up larger markets for firms (especially in otherwise small markets), thereby enabling better economies of scale, and hence greater efficiency (lower costs and resources per unit of good produced).

Supply Side Economies of Scale

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

Supply Side Economies of Scale

- What is a good model of the cost of manufacturing a set of items?
- Let V be a set of possible items 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

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

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

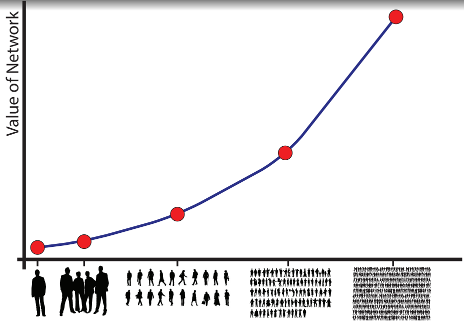
- What is a good model of the cost of manufacturing a set of items?
- Let V be a set of possible items to manufacture, and let $f(S)$ for $S \subseteq V$ be the manufacture costs of items in the subset S.
- Ex: V might be paint colors to produce: green, red, blue, yellow, white, etc.
- Producing green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

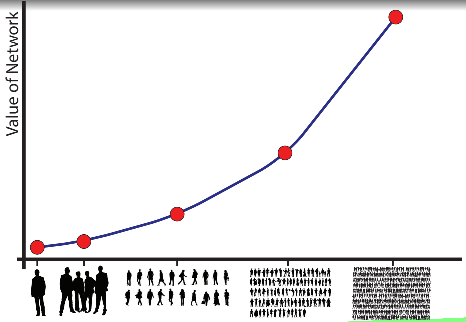
Demand side Economies of Scale: Network Externalities

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



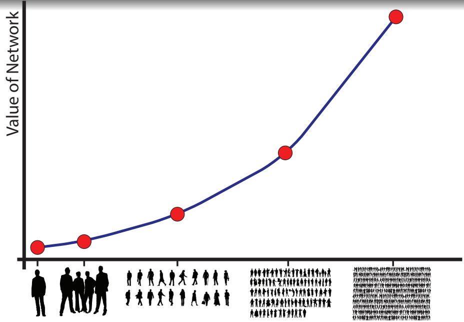
Demand side Economies of Scale: Network Externalities

- 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

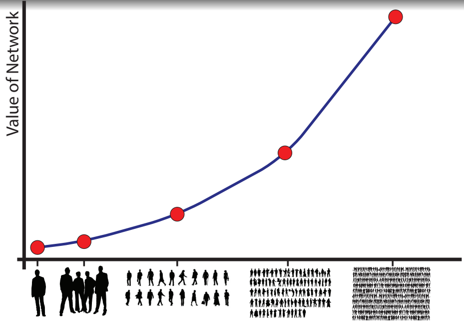
- Value of a network to a user depends on the number of other users in that network. External use benefits internal use.
- Consumers derive positive incremental value when size of the market for that good increases.



- 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

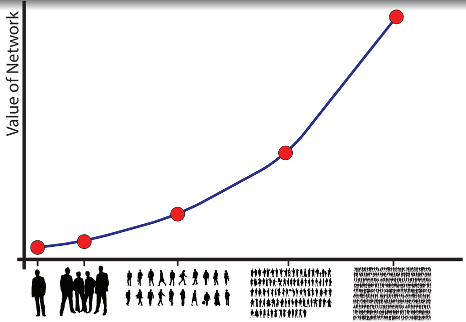
- Value of a network to a user depends on the number of other users in that network. External use benefits internal use.
- Consumers derive positive incremental value when size of the market for that good increases.



- 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), benefit depends on total user base.

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), benefit depends on total user base.
- 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.

Examples: Positive Network Effects

- railroad - standard rail format and shared access
- The telephone, who wants to talk by phone only to oneself?
- the internet, more valuable per person the more people use it.
- ebooks (the more people comment, the better it gets)
- social network sites: facebook more valuable with everyone online
- online education, Massive Open Online Courses (MOOCs) such as Coursera, edX, etc. – with many people simultaneously taking a class, all gain due to richer peer discussions due to greater pool of well matched study groups, more simultaneous similar questions/problems that are asked $\Rightarrow$ more efficient learning & training data for ML algorithms to learn how people learn.
- Software (e.g., Microsoft office, smartphone apps, etc.): more people means more bug reporting $\Rightarrow$ better & faster software evolution.
- gmail and web-based email (collaborative spam filtering).
- wikipedia, collaborative documents
- any widely used standard (job training now is useful in the future)
- the "tipping point", and "winner take all" (one platform prevails)

Examples: Other Network Effects

No Network Externalities

- food/drink - (should be) independent of how many others are eating the type of food.

Examples: Other Network Effects

No Network Externalities

- food/drink - (should be) independent of how many others are eating the type of food.
- Music - your enjoyment should (ideally) be independent of others' enjoyment (but maybe not, see Salganik, Dodds, Watts'06).

Examples: Other Network Effects

No Network Externalities

- food/drink - (should be) independent of how many others are eating the type of food.
- Music - your enjoyment should (ideally) be independent of others' enjoyment (but maybe not, see Salganik, Dodds, Watts'06).

Negative Network Effects

Examples: Other Network Effects

No Network Externalities

- food/drink - (should be) independent of how many others are eating the type of food.
- Music - your enjoyment should (ideally) be independent of others' enjoyment (but maybe not, see Salganik, Dodds, Watts'06).

Negative Network Effects

- clothing

Examples: Other Network Effects

No Network Externalities

- food/drink - (should be) independent of how many others are eating the type of food.
- Music - your enjoyment should (ideally) be independent of others' enjoyment (but maybe not, see Salganik, Dodds, Watts'06).

Negative Network Effects

- clothing
- (Halloween) costumes

Examples: Other Network Effects

No Network Externalities

- food/drink - (should be) independent of how many others are eating the type of food.
- Music - your enjoyment should (ideally) be independent of others' enjoyment (but maybe not, see Salganik, Dodds, Watts'06).

Negative Network Effects

- clothing
- (Halloween) costumes
- perfume?

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, structured sparse 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, structured sparse 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, structured sparse 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, structured sparse 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, structured sparse 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, structured sparse 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.

Diversity Functions

- Diverse web search. Given search term (e.g., "jaguar") but no other information, one probably does not want only articles about cars.

Diversity Functions

- Diverse web search. Given search term (e.g., “jaguar”) but no other information, one probably does not want only articles about cars.
- Given a set V of of items, how do we choose a subset $S \subseteq V$ that is as diverse as possible, with perhaps constraints on S such as its size.

Diversity Functions

- Diverse web search. Given search term (e.g., "jaguar") but no other information, one probably does not want only articles about cars.
- Given a set V of of items, how do we choose a subset $S \subseteq V$ that is as diverse as possible, with perhaps constraints on S such as its size.
- How do we choose the smallest set S that maintains a given quality of diversity?

Diversity Functions

- Diverse web search. Given search term (e.g., "jaguar") but no other information, one probably does not want only articles about cars.
- Given a set V of of items, how do we choose a subset $S \subseteq V$ that is as diverse as possible, with perhaps constraints on S such as its size.
- How do we choose the smallest set S that maintains a given quality of diversity?
- Goal of diversity: ensure proper representation in chosen set that, say otherwise in a random sample, could lead to poor representation of normally underrepresented groups.

Extractive Document Summarization

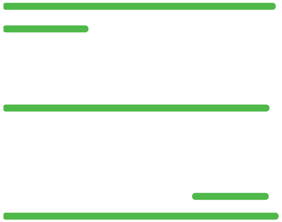
- The figure below represents the sentences of a document

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



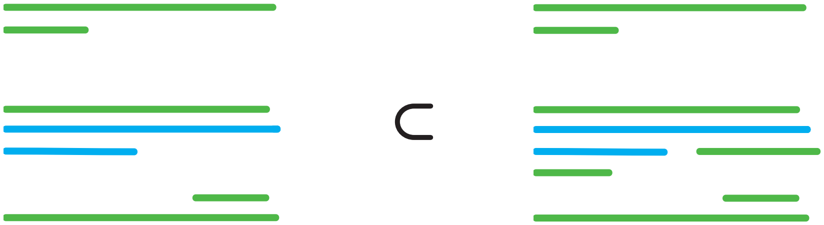
Extractive Document Summarization

- 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

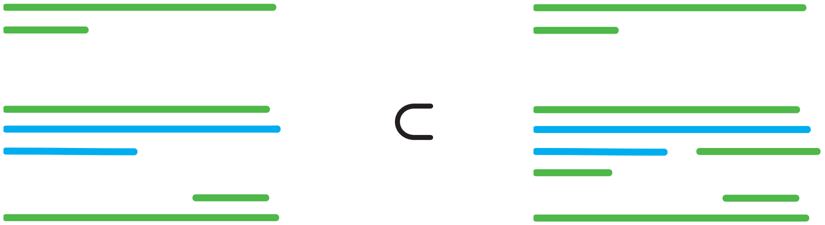
- We extract sentences (green) as a summary of the full document



- 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

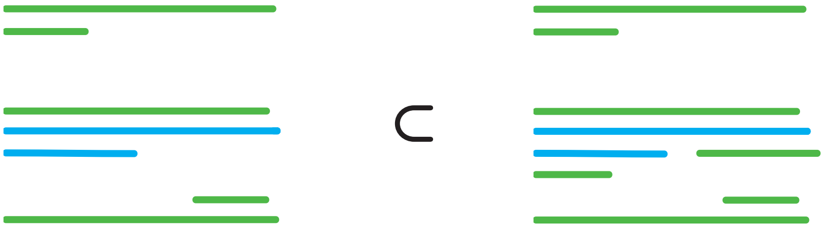
- We extract sentences (green) as a summary of the full document



- 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

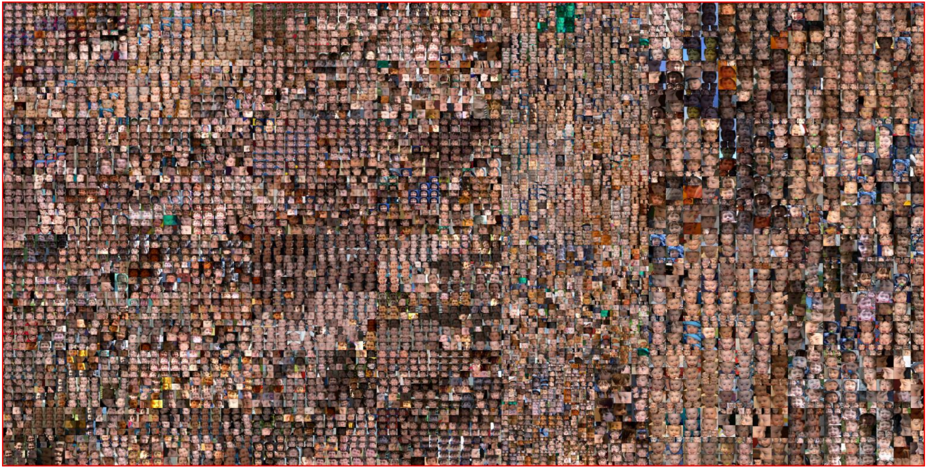
- We extract sentences (green) as a summary of the full document



- 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.



Web search and information retrieval

- A web search is a form of summarization based on query.
- Goal of a web search engine is to produce a ranked list of web pages that, conditioned on the text query entered, summarizes the most important links on the web.
- Information retrieval (the science of automatically acquiring information), book and music recommendation systems —
- Overall goal: user should quickly find information that is informative, concise, accurate, relevant (to the user's needs), and comprehensive.

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 $n = |V|$ observed measurement variables $(X_1, X_2, \ldots, X_n) = X_V$ in a 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 $n = |V|$ observed measurement variables $(X_1, X_2, \ldots, X_n) = X_V$ in a probability model $\Pr(Y, X_1, X_2, \ldots, X_n)$.
- Too costly to use all V variables. Goal: choose subset $A \subseteq V$ of variables within budget $|A| \leq k$. Predictions based on only $\Pr(y|x_A)$, hence subset A should retain accuracy.

Variable Selection in Classification/Regression

- Let Y be a random variable we wish to accurately predict based on at most $n = |V|$ observed measurement variables $(X_1, X_2, \ldots, X_n) = X_V$ in a probability model $\Pr(Y, X_1, X_2, \ldots, X_n)$.
- Too costly to use all V variables. Goal: choose subset $A \subseteq V$ of variables within budget $|A| \leq k$. Predictions based on only $\Pr(y|x_A)$, hence subset A should retain accuracy.
- The mutual information function $f(A) = I(Y; X_A)$ ("information gain") 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 $n = |V|$ observed measurement variables $(X_1, X_2, \dots, X_n) = X_V$ in a probability model $\Pr(Y, X_1, X_2, \dots, X_n)$.
- Too costly to use all V variables. Goal: choose subset $A \subseteq V$ of variables within budget $|A| \leq k$. Predictions based on only $\Pr(y|x_A)$, hence subset A should retain accuracy.
- The mutual information function $f(A) = I(Y; X_A)$ ("information gain") 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:

$$I(Y; X_A) = \sum_{y, x_A} \Pr(y, x_A) \log \frac{\Pr(y, x_A)}{\Pr(y)\Pr(x_A)} = H(Y) - H(Y|X_A) \qquad (1.18)$$

$$= H(X_A) - H(X_A|Y) = H(X_A) + H(Y) - H(X_A, Y) \qquad (1.19)$$

Variable Selection in Classification/Regression

- Let Y be a random variable we wish to accurately predict based on at most $n = |V|$ observed measurement variables $(X_1, X_2, \ldots, X_n) = X_V$ in a probability model $\Pr(Y, X_1, X_2, \ldots, X_n)$.
- Too costly to use all V variables. Goal: choose subset $A \subseteq V$ of variables within budget $|A| \leq k$. Predictions based on only $\Pr(y|x_A)$, hence subset A should retain accuracy.
- The mutual information function $f(A) = I(Y; X_A)$ ("information gain") 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:

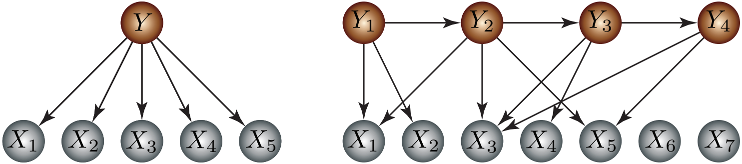
$$I(Y; X_A) = \sum_{y, x_A} \Pr(y, x_A) \log \frac{\Pr(y, x_A)}{\Pr(y)\Pr(x_A)} = H(Y) - H(Y|X_A) \tag{1.18}$$

$$= H(X_A) - H(X_A|Y) = H(X_A) + H(Y) - H(X_A, Y) \tag{1.19}$$

- Applicable in pattern recognition, also in sensor coverage problem, where Y is whatever question we wish to ask about environment.

Information Gain and Feature Selection
in Pattern Classification: Naïve Bayes

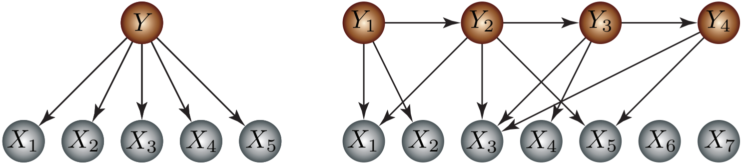
- Naïve Bayes property: $X_A \perp\!\!\!\perp X_B | Y$ for all A, B.



Information Gain and Feature Selection

in Pattern Classification: Naïve Bayes

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



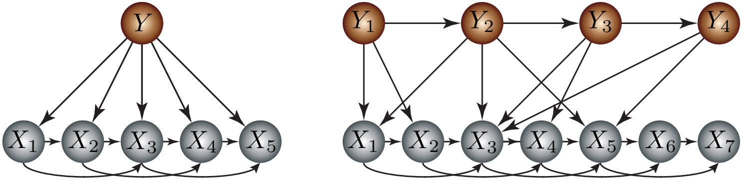
- When $X_A \perp\!\!\!\perp X_B | Y$ for all A, B (the Naïve Bayes assumption holds), then

$$f(A) = I(Y; X_A) = H(X_A) - H(X_A|Y) = H(X_A) - \sum_{a \in A} H(X_a|Y) \tag{1.20}$$

is submodular (submodular minus modular).

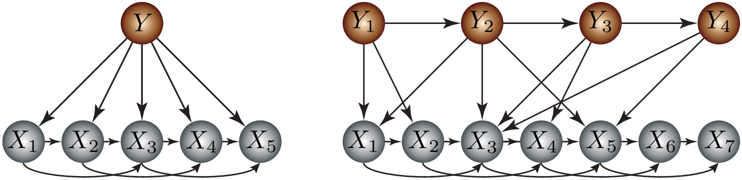
Variable Selection in Pattern Classification

- Naïve Bayes property fails:



Variable Selection in Pattern Classification

- Naïve Bayes property fails:



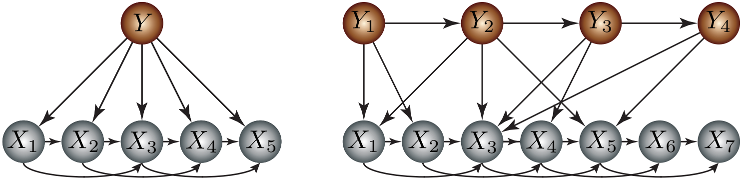
- $f(A)$ naturally expressed as a difference of two submodular functions

$$f(A) = I(Y; X_A) = H(X_A) - H(X_A|Y), \tag{1.21}$$

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

$$f(A) = I(Y; X_A) = H(X_A) - H(X_A|Y), \tag{1.21}$$

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:

$$f(A) = \sum_{a \in A} I(X_a; Y) - \lambda \sum_{a,a' \in A} I(X_a; X_{a'}|Y) \tag{1.22}$$

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

$$R^2_{Z,A} = \frac{\text{Var}(Z) - E[(Z - \tilde{Z}_A)^2]}{\text{Var}(Z)} \tag{1.23}$$

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

$$R^2_{Z,A} = \frac{\text{Var}(Z) - E[(Z - \tilde{Z}_A)^2]}{\text{Var}(Z)} \tag{1.23}$$

- $R^2_{Z,A}$'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 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).

Variable Selection: Linear Regression Case

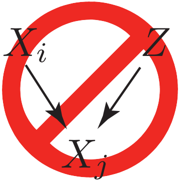
- Here Z is continuous and predictor is linear $\tilde{Z}_A = \sum_{i \in A} \alpha_i X_i$.
- Error measure is the residual variance

$$R^2_{Z,A} = \frac{\text{Var}(Z) - E[(Z - \tilde{Z}_A)^2]}{\text{Var}(Z)} \tag{1.23}$$

- $R^2_{Z,A}$'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 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

$$f(A) = R^2_{Z,A} = b_A{}^\intercal (C_A^{-1})^\intercal b_A \tag{1.24}$$

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 $V = \{v_1, v_2, \dots, v_n\}$ 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 $V = \{v_1, v_2, \dots, v_n\}$ 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 $u \in U$.

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.

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

$v =$ "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

$$g(m_u(A+v)) - g(m_u(A)) \geq g(m_u(B+v)) - g(m_u(B)) \tag{1.25}$$

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

$$g(m_u(A+v)) - g(m_u(A)) \geq g(m_u(B+v)) - g(m_u(B)) \tag{1.25}$$

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

$$f(X) = \sum_{u \in U} \alpha_u g_u(m_u(X)) \tag{1.26}$$

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

$$g(m_u(A + v)) - g(m_u(A)) \geq g(m_u(B + v)) - g(m_u(B)) \tag{1.25}$$

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

$$f(X) = \sum_{u \in U} \alpha_u g_u(m_u(X)) \tag{1.26}$$

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:

$$\bar{m}_u(X) = \frac{m_u(X)}{\sum_{u' \in U} m_{u'}(X)} = \frac{m_u(X)}{m(X)} \tag{1.27}$$

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:

$$\bar{m}_u(X) = \frac{m_u(X)}{\sum_{u' \in U} m_{u'}(X)} = \frac{m_u(X)}{m(X)} \tag{1.27}$$

  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:

$$\bar{m}_u(X) = \frac{m_u(X)}{\sum_{u' \in U} m_{u'}(X)} = \frac{m_u(X)}{m(X)} \tag{1.27}$$

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:

$$D(p || \{\bar{m}_u(X)\}_{u \in U}) = \sum_{u \in U} p_u \log p_u - \sum_{u \in U} p_u \log(\bar{m}_u(X)) \tag{1.28}$$

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

$$= -H(p) + \log m(X) - \sum_{u \in U} p_u \log(m_u(X)) \tag{1.29}$$

Data Subset Selection, KL-divergence

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

$$D(p||\{\bar{m}_u(X)\}) = \text{const.} + \log m(X) - \sum_{u \in U} p_u \log(m_u(X)) \quad (1.30)$$

Data Subset Selection, KL-divergence

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

$$D(p||\{\bar{m}_u(X)\}) = \text{const.} + \log m(X) - \sum_{u \in U} p_u \log(m_u(X)) \quad (1.30)$$

- 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,

$$D(p||\{\bar{m}_u(X)\}) = \text{const.} + \log m(X) - \sum_{u \in U} p_u \log(m_u(X)) \quad (1.30)$$

- 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,

$$D(p||\{\bar{m}_u(X)\}) = \text{const.} + \log m(X) - \sum_{u \in U} p_u \log(m_u(X)) \quad (1.30)$$

- 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)

$$g(X) \triangleq \log m(X) - D(p||\{\bar{m}_u(X)\}) = \sum_{u \in U} p_u \log(m_u(X)) \quad (1.31)$$

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

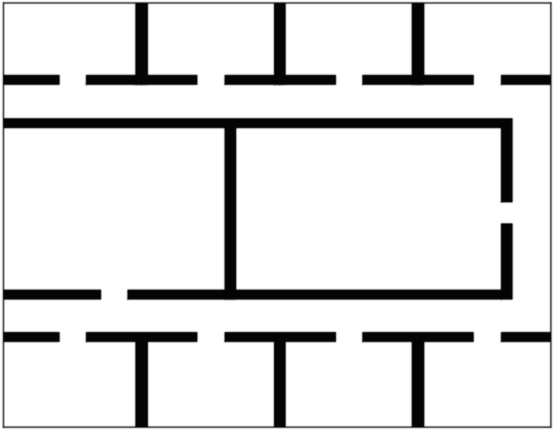
- Information gain applicable not only in pattern recognition, but in the sensor coverage problem as well, where Y is whatever question we wish to ask about an environment.
- Given an environment, there is a set V of candidate locations for placement of a sensor (e.g., temperature, gas, audio, video,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

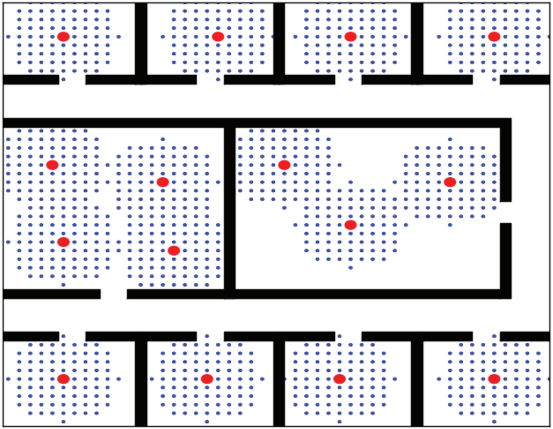
Sensor Placement within Buildings

- 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.



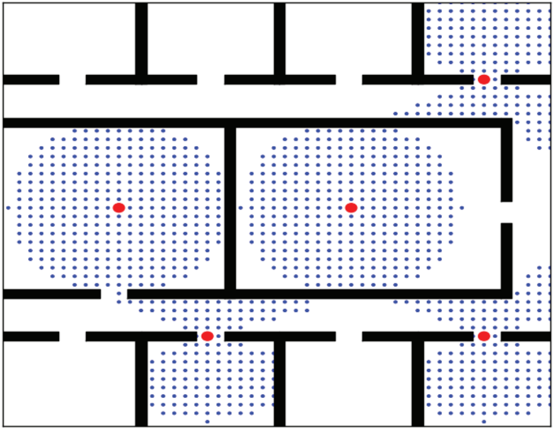
Sensor Placement within Buildings

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



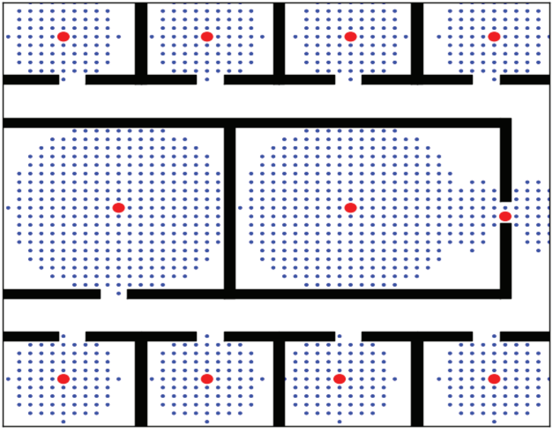
Sensor Placement within Buildings

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



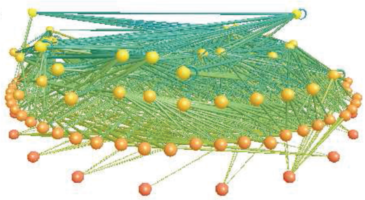
Sensor Placement within Buildings

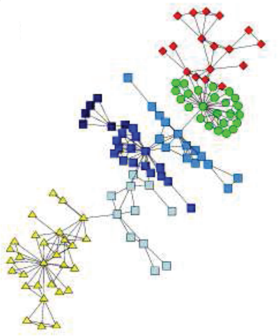
- Example sensor placement using mixed range sensors (located at red dots).

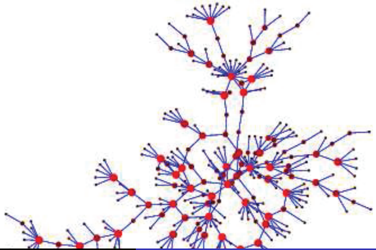


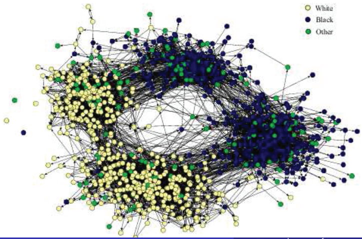
Social Networks

(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

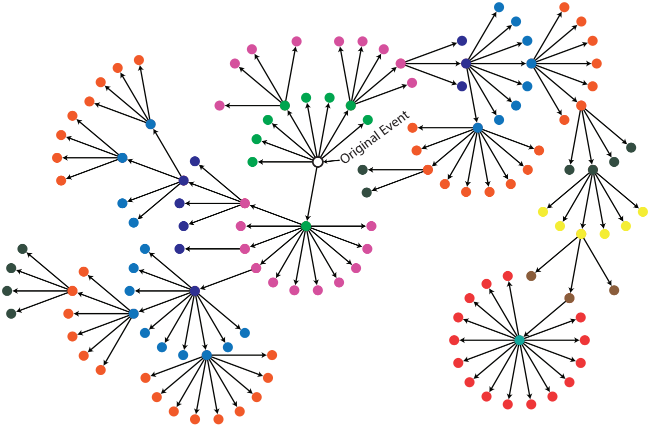
- Let V be a set of individuals, how valuable is a given friend $v \in V$?
- It depends on how many friends you have.
- Valuate a group of friends $S \subseteq V$ via set function $f(S)$.
- A submodular model: a friend becomes less marginally valuable as your set of friends grows.
- Supermodular model: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

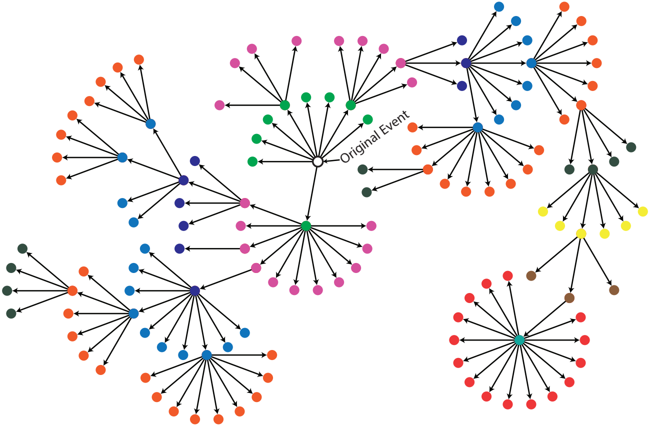
Information Cascades, Diffusion Networks

- 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).



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

Diffusion Networks

- Information propagation: when blogs or news stories break, and creates an information cascade over multiple other blogs/newspapers/magazines.
- Viral marketing: What is the pattern of trendsetters that cause an individual to purchase a product?
- Epidemiology: who got sick from whom, and what is the network of such links?
- How can we infer the connectivity of a network (of memes, purchase decisions, virusus, etc.) based only on diffusion traces (the time that each node is "infected")? How to find the most likely tree?

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

$$p(x) = \frac{1}{Z} \exp(-E(x)) \tag{1.32}$$

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

Graphical Model Structure Learning

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

$$p(x) = \frac{1}{Z} \exp(-E(x)) \tag{1.32}$$

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

$$p(x) = \frac{1}{Z} \exp(-E(x)) \tag{1.32}$$

  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:

$$E(x) = \sum_{c \in \mathcal{C}} E_c(x_c) \tag{1.33}$$

Graphical Model Structure Learning

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

$$p(x) = \frac{1}{Z} \exp(-E(x)) \tag{1.32}$$

  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:

$$E(x) = \sum_{c \in \mathcal{C}} E_c(x_c) \tag{1.33}$$

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

$$p(x) = \frac{1}{Z} \exp(-E(x)) \tag{1.32}$$

  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:

$$E(x) = \sum_{c \in \mathcal{C}} E_c(x_c) \tag{1.33}$$

- 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 $V \times V$.

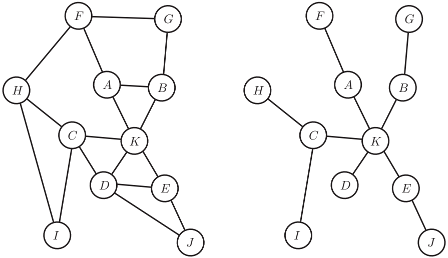
Graphical Models: Learning Tree Distributions

- 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:

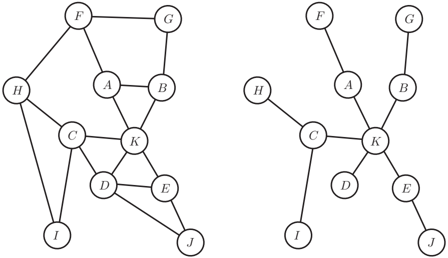
$$\begin{aligned} \underset{p_t \in \mathcal{F}(G,\mathcal{M})}{\text{minimize}} \quad & D(p||p_t) \\ \text{subject to} \quad & p_t \in \mathcal{F}(T, \mathcal{M}). \\ & T = (V, F) \text{ is a tree} \end{aligned}$$



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).

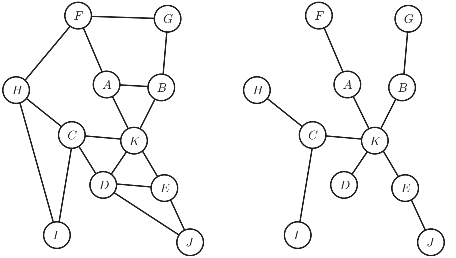
Graphical Models: Learning Tree Distributions

- 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:

$$\underset{p_t \in \mathcal{F}(G,\mathcal{M})}{\text{minimize}} \quad D(p||p_t)$$

$$\text{subject to} \quad p_t \in \mathcal{F}(T, \mathcal{M}).$$

$$T = (V, F) \text{ is a tree}$$

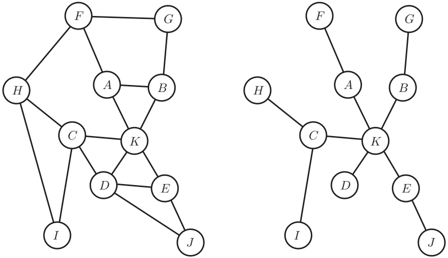


- Discrete problem: choose the optimal set of edges $A \subseteq E$ that constitute tree (i.e., find a spanning tree of G of best quality).
- Define $f : 2^E \to \mathbb{R}_+$ where f is a weighted cycle matroid rank function (a type of submodular function), with weights $w(e) = w(u, v) = I(X_u; X_v)$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

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



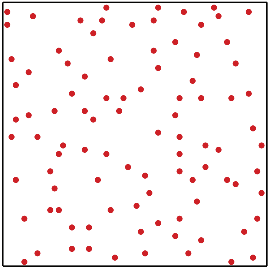
- Discrete problem: choose the optimal set of edges $A \subseteq E$ that constitute tree (i.e., find a spanning tree of G of best quality).
- Define $f : 2^E \to \mathbb{R}_+$ where f is a weighted cycle matroid rank function (a type of submodular function), with weights $w(e) = w(u, v) = I(X_u; X_v)$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

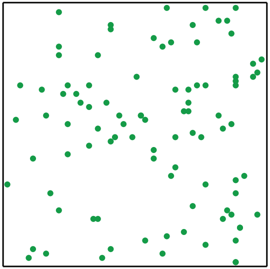
- Sometimes we wish not only to valuate subsets $A \subseteq V$ but to induce probability distributions over all subsets.

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).



DPP

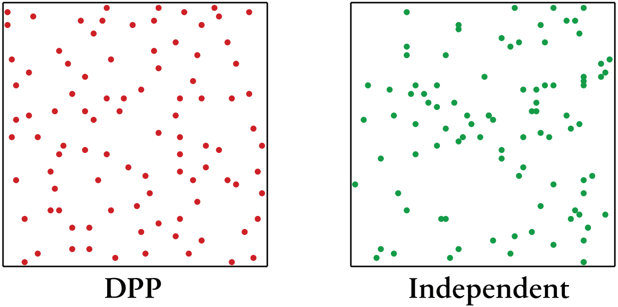


Independent

(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).

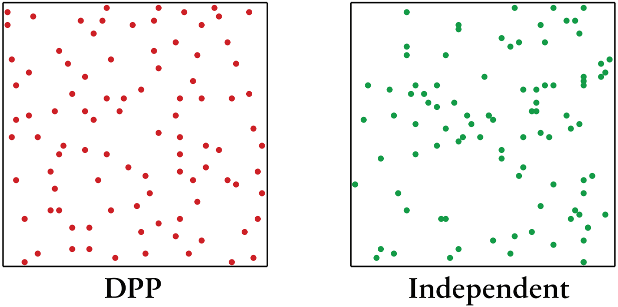


(Kulesza, Gillenwater, & Taskar, 2011)

- 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:

$$\Pr(\mathbf{X} = x) = \frac{|M_{X(x)}|}{|M + I|} = \exp\left(\log\left(\frac{|M_{X(x)}|}{|M + I|}\right)\right) \propto \det(M_{X(x)}) \tag{1.34}$$

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:

$$\Pr(\mathbf{X} = x) = \frac{|M_{X(x)}|}{|M + I|} = \exp\left(\log\Big(\frac{|M_{X(x)}|}{|M + I|}\Big)\right) \propto \det(M_{X(x)}) \tag{1.34}$$

  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:

$$\sum_{x \in \{0,1\}^V : x \geq y} \Pr(\mathbf{X} = x) = \Pr(\mathbf{X} \geq y) = \exp\left(\log\Big(|K_{Y(y)}|\Big)\right) \tag{1.35}$$

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:

$$\Pr(\mathbf{X} = x) = \frac{|M_{X(x)}|}{|M+I|} = \exp\left(\log\left(\frac{|M_{X(x)}|}{|M+I|}\right)\right) \propto \det(M_{X(x)}) \tag{1.34}$$

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:

$$\sum_{x \in \{0,1\}^V : x \geq y} \Pr(\mathbf{X} = x) = \Pr(\mathbf{X} \geq y) = \exp\left(\log\left(|K_{Y(y)}|\right)\right) \tag{1.35}$$

- 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 $x, y \in \{0,1\}^V$, $y \le x$ if $y(v) \le x(v), \forall v \in V$.
- 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:

$$\Pr(\mathbf{X} = x) = \frac{|M_{X(x)}|}{|M + I|} = \exp\left(\log\left(\frac{|M_{X(x)}|}{|M + I|}\right)\right) \propto \det(M_{X(x)}) \tag{1.34}$$

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:

$$\sum_{x \in \{0,1\}^V : x \ge y} \Pr(\mathbf{X} = x) = \Pr(\mathbf{X} \ge y) = \exp\left(\log\left(|K_{Y(y)}|\right)\right) \tag{1.35}$$

- 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.

Graphical Models and fast MAP Inference

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

$$p(x) = \frac{1}{Z} \exp(-E(x)) \tag{1.36}$$

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:

$$p(x) = \frac{1}{Z}\exp(-E(x)) \tag{1.36}$$

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

$$x^* \in \operatorname*{argmax}_{x \in \{0,1\}^V} p(x) \tag{1.37}$$

Graphical Models and fast MAP Inference

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

$$p(x) = \frac{1}{Z}\exp(-E(x)) \tag{1.36}$$

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

$$x^* \in \operatorname*{argmax}_{x \in \{0,1\}^V} p(x) \tag{1.37}$$

- 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:

$$p(x) = \frac{1}{Z} \exp(-E(x)) \tag{1.36}$$

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

$$x^* \in \operatorname*{argmax}_{x \in \{0,1\}^V} p(x) \tag{1.37}$$

- 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:

$$p(x) = \frac{1}{Z}\exp(-E(x)) \tag{1.36}$$

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

$$x^* \in \operatorname*{argmax}_{x \in \{0,1\}^V} p(x) \tag{1.37}$$

- 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:

$$p(x) = \frac{1}{Z} \exp(-E(x)) \tag{1.36}$$

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

$$x^* \in \operatorname*{argmax}_{x \in \{0,1\}^V} p(x) \tag{1.37}$$

- 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:

$$p(x) = \frac{1}{Z} \exp(-E(x)) \tag{1.36}$$

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

$$x^* \in \operatorname*{argmax}_{x \in \{0,1\}^V} p(x) \tag{1.37}$$

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

$$E(x) = \sum_{v \in V(G)} e_v(x_v) + \sum_{(i,j) \in E(G)} e_{ij}(x_i, x_j) \tag{1.38}$$

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:

$$E(x) = \sum_{v \in V(G)} e_v(x_v) + \sum_{(i,j) \in E(G)} e_{ij}(x_i, x_j) \tag{1.38}$$

- $e_v(x_v)$ 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:

$$E(x) = \sum_{v \in V(G)} e_v(x_v) + \sum_{(i,j) \in E(G)} e_{ij}(x_i, x_j) \tag{1.38}$$

- $e_v(x_v)$ 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:

$$E(x) = \sum_{v \in V(G)} e_v(x_v) + \sum_{(i,j) \in E(G)} e_{ij}(x_i, x_j) \tag{1.38}$$

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

Order-two (edge) graphical models

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

$$E(x) = \sum_{v \in V(G)} e_v(x_v) + \sum_{(i,j) \in E(G)} e_{ij}(x_i, x_j) \tag{1.38}$$

- $e_v(x_v)$ 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 $\mathsf{D}_{X_v} = \{0, 1\}$ (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.,

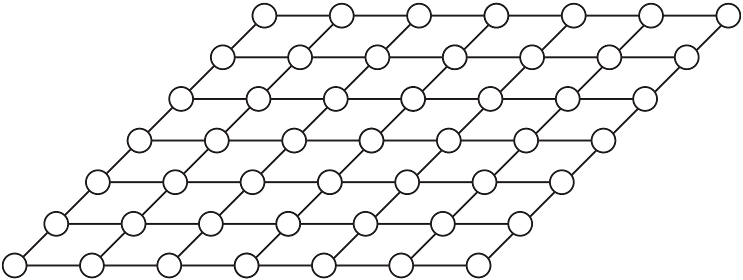
$$\min_{x \in \{0,1\}^V} E(x) \tag{1.39}$$

MRF example

Markov random field

$$\log p(x) \propto \sum_{v \in V(G)} e_v(x_v) + \sum_{(i,j) \in E(G)} e_{ij}(x_i, x_j) \quad (1.40)$$

When G is a 2D grid graph, we have

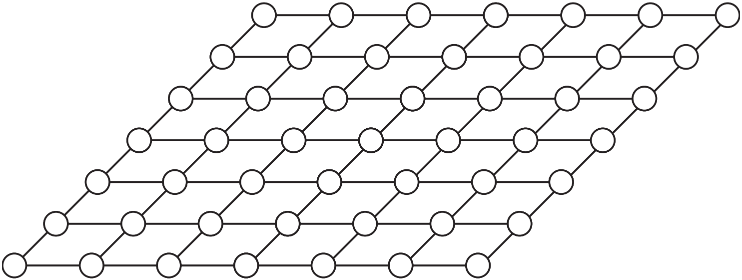


Create an auxiliary graph

- 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 $O(\sqrt{|V|})$.

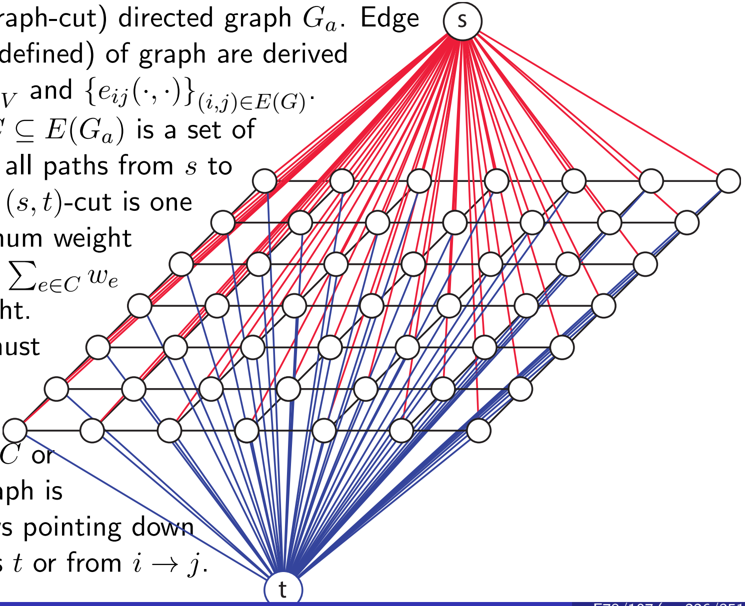


Transformation from graphical model to auxiliary graph

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

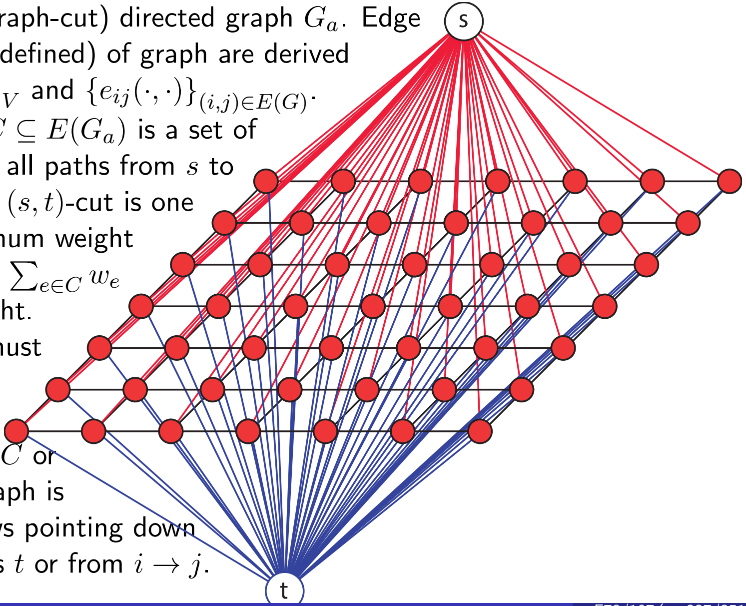
An (s,t)-cut $C \subseteq E(G_a)$ is a set of edges that cut all paths from s to t. A minimum (s,t)-cut is one that has minimum weight where $w(C) = \sum_{e \in C} w_e$ is the cut weight.

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 $i \to j$.



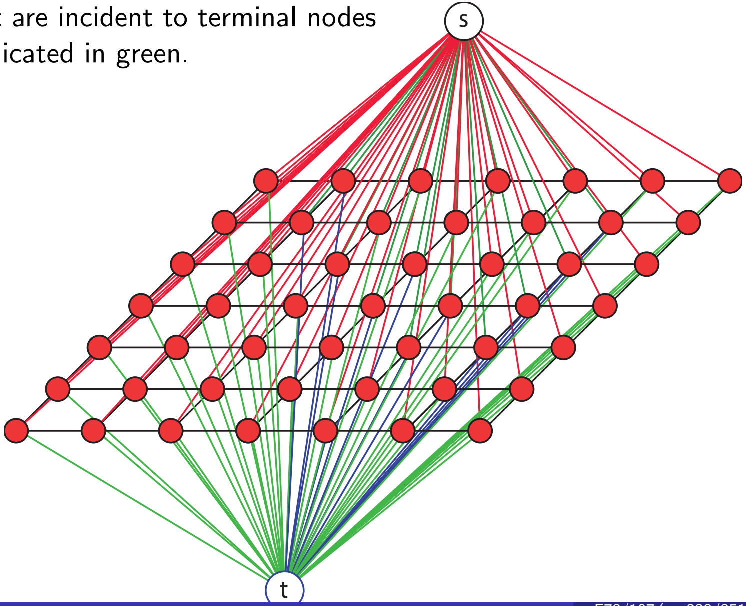
Transformation from graphical model to auxiliary graph

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 $i \to j$.



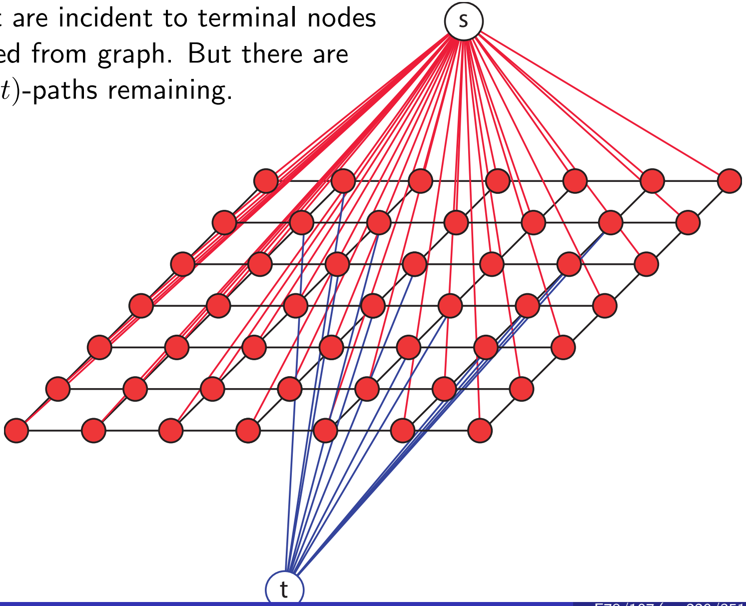
Transformation from graphical model to auxiliary graph

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



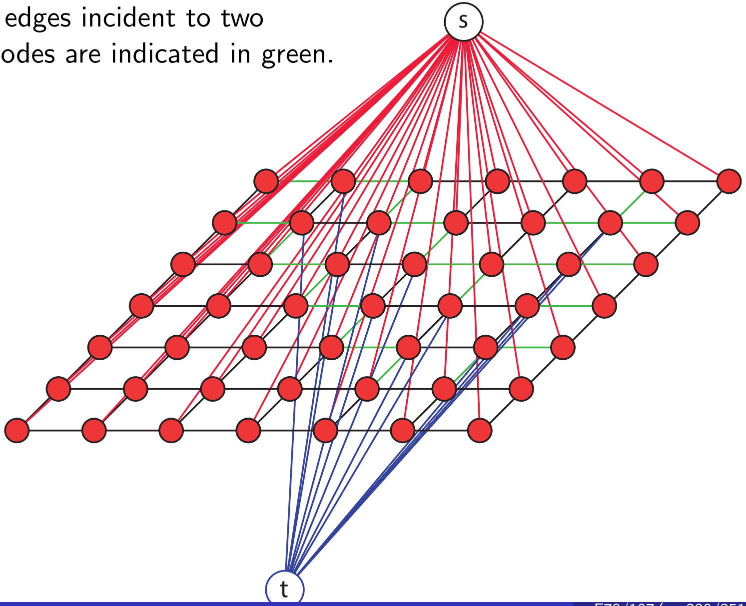
Transformation from graphical model to auxiliary graph

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



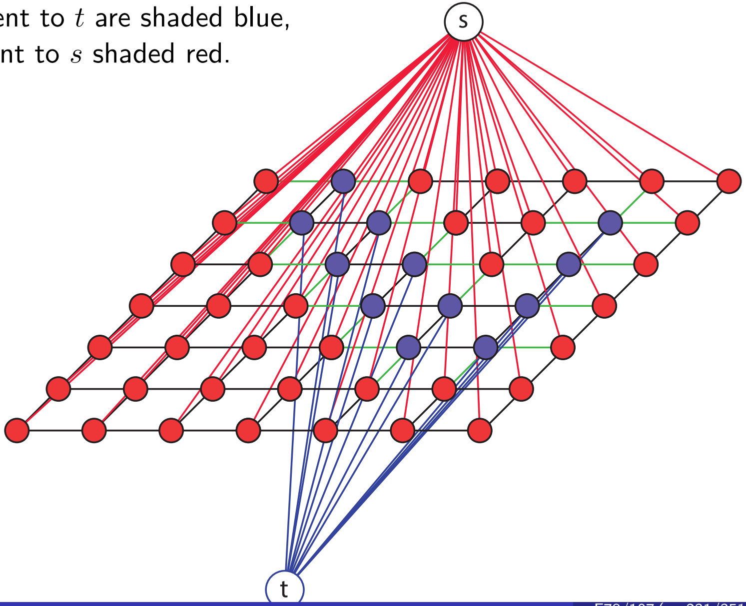
Transformation from graphical model to auxiliary graph

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



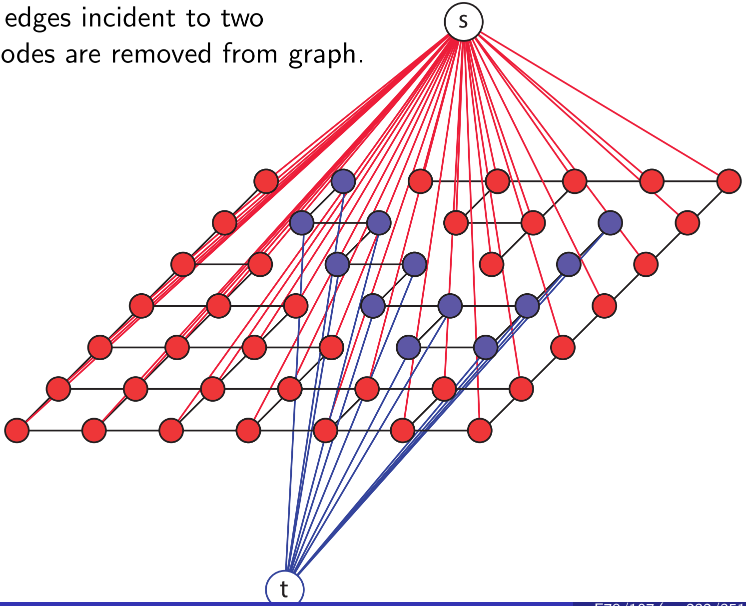
Transformation from graphical model to auxiliary graph

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



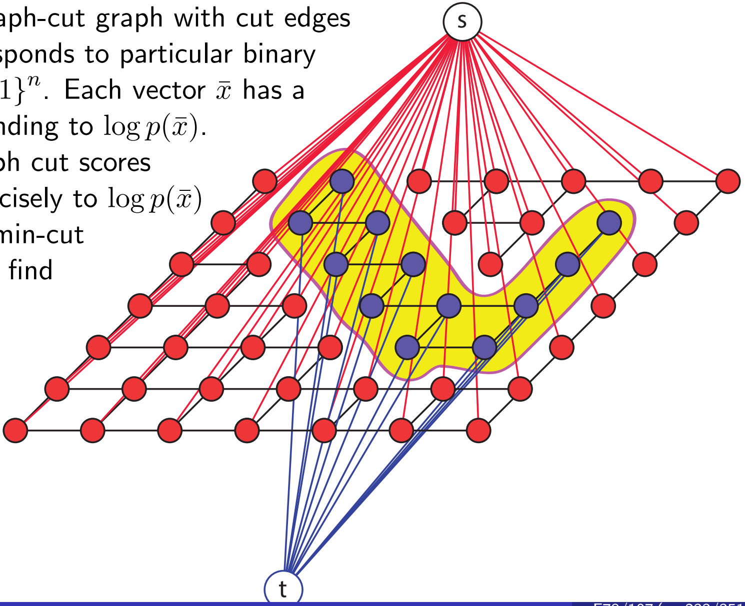
Transformation from graphical model to auxiliary graph

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 $E(x) = \log p(\bar{x}) +$ 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

$$w_{s,v} = (e_v(1) - e_v(0))\mathbf{1}(e_v(1) > e_v(0)) \tag{1.41}$$

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

$$w_{v,t} = (e_v(0) - e_v(1))\mathbf{1}(e_v(0) \geq e_v(1)) \tag{1.42}$$

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

$$w_{i,j} = e_{ij}(1,0) + e_{ij}(0,1) - e_{ij}(1,1) - e_{ij}(0,0) \tag{1.43}$$

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

$$w_{s,i} \leftarrow w_{s,i} + (e_{ij}(1,0) - e_{ij}(0,0)) \tag{1.44}$$

$$w_{j,t} \leftarrow w_{j,t} + (e_{ij}(1,1) - e_{ij}(0,1)) \tag{1.45}$$

and analogous increments if inequalities are flipped.

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:

$$w_{i,j} = e_{ij}(1,0) + e_{ij}(0,1) - e_{ij}(1,1) - e_{ij}(0,0) \tag{1.46}$$

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:

$$w_{i,j} = e_{ij}(1,0) + e_{ij}(0,1) - e_{ij}(1,1) - e_{ij}(0,0) \tag{1.46}$$

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

$$e_{ij}(1,0) + e_{ij}(0,1) \geq e_{ij}(1,1) + e_{ij}(0,0) \tag{1.47}$$

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:

$$w_{i,j} = e_{ij}(1,0) + e_{ij}(0,1) - e_{ij}(1,1) - e_{ij}(0,0) \tag{1.46}$$

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

$$e_{ij}(1,0) + e_{ij}(0,1) \geq e_{ij}(1,1) + e_{ij}(0,0) \tag{1.47}$$

- 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:

$$e_{ij}(0,1) + e_{ij}(1,0) \geq e_{ij}(1,1) + e_{ij}(0,0) \tag{1.48}$$

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:

$$e_{ij}(0,1) + e_{ij}(1,0) \geq e_{ij}(1,1) + e_{ij}(0,0) \tag{1.48}$$

- 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:

$$e_{ij}(0,1) + e_{ij}(1,0) \geq e_{ij}(1,1) + e_{ij}(0,0) \tag{1.48}$$

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

$$f(X) = \sum_{\{i,j\} \in \mathcal{E}(G)} f_{i,j}(X \cap \{i,j\}) \tag{1.49}$$

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:

$$e_{ij}(0,1) + e_{ij}(1,0) \geq e_{ij}(1,1) + e_{ij}(0,0) \quad (1.48)$$

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

$$f(X) = \sum_{\{i,j\} \in \mathcal{E}(G)} f_{i,j}(X \cap \{i, j\}) \quad (1.49)$$

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.

$$\log \Pr(x) = f(x) + \text{const.} = -E(x) + \text{const.} \tag{1.50}$$

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.

$$\log \Pr(x) = f(x) + \text{const.} = -E(x) + \text{const.} \tag{1.50}$$

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:

$$\log \Pr(x) = f(x) + \text{const.} \tag{1.51}$$

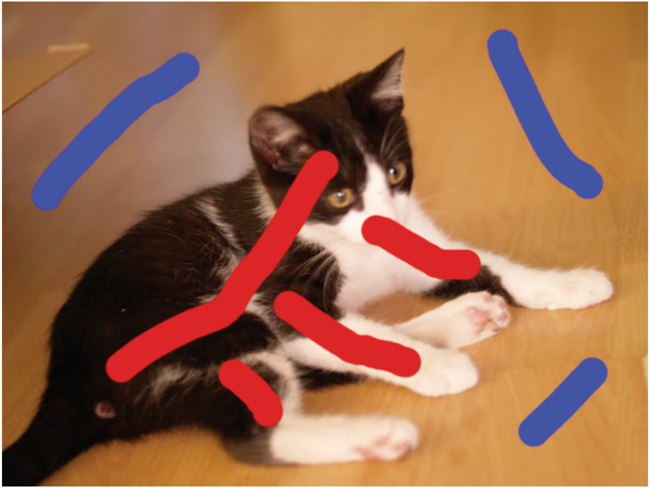
where f is submodular. MAP or high-probable assignments should be "diverse", or "complex", or "covering", like in determinantal point processes.

Submodular potentials in GMs: Image Segmentation

- an image needing to be segmented.

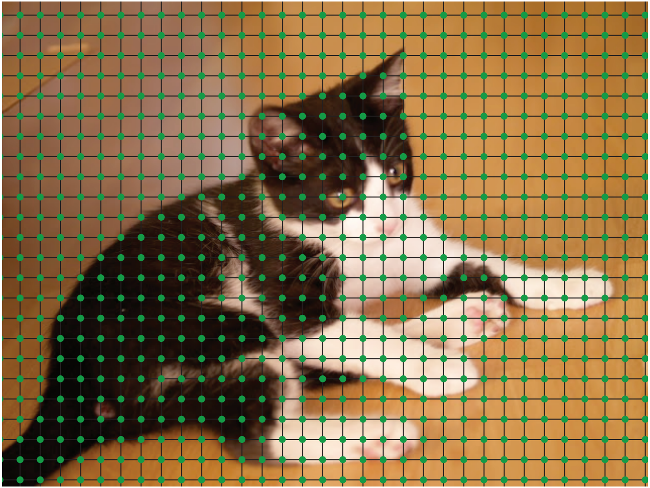
Submodular potentials in GMs: Image Segmentation

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



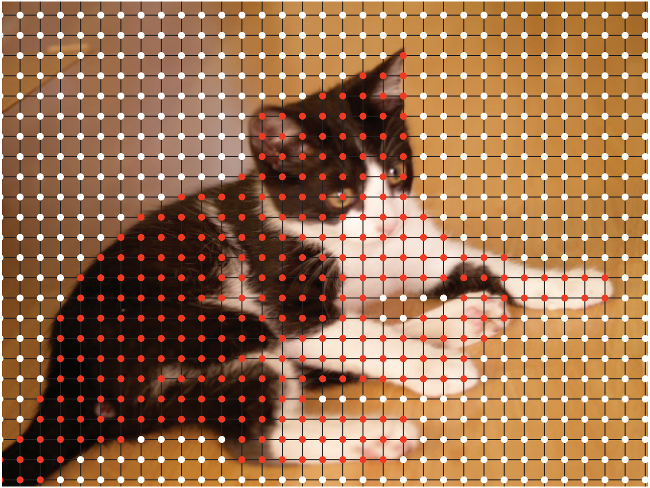
Submodular potentials in GMs: Image Segmentation

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



Submodular potentials in GMs: Image Segmentation

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



Submodular potentials in GMs: Image Segmentation

- the foreground is removed from the background.

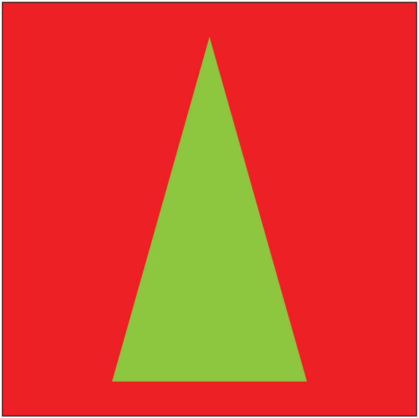
Shrinking bias in graph cut image segmentation

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

Shrinking bias in graph cut image segmentation

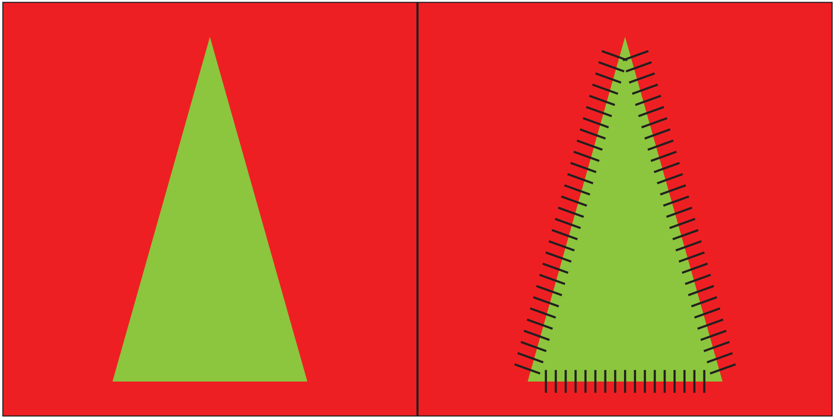
Shrinking bias in image segmentation

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



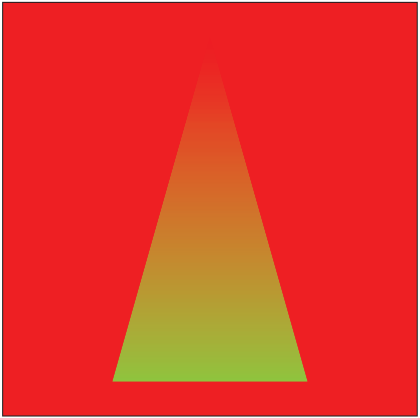
Shrinking bias in image segmentation

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



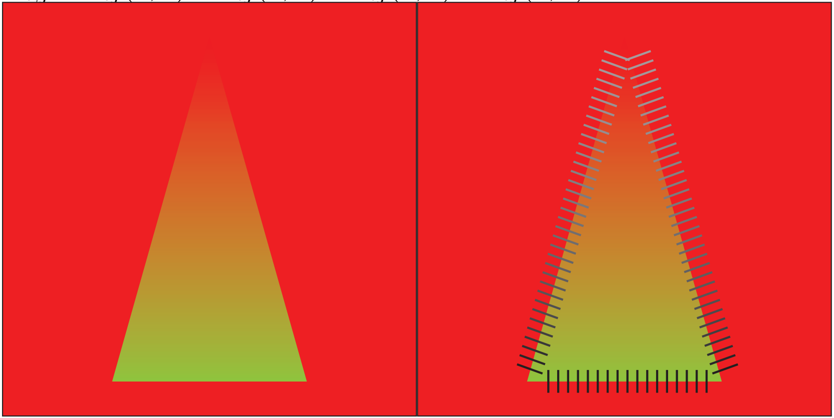
Shrinking bias in image segmentation

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



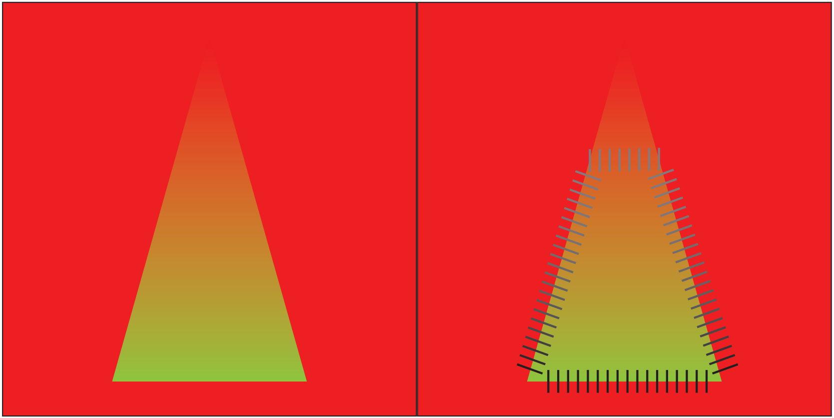
Shrinking bias in image segmentation

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



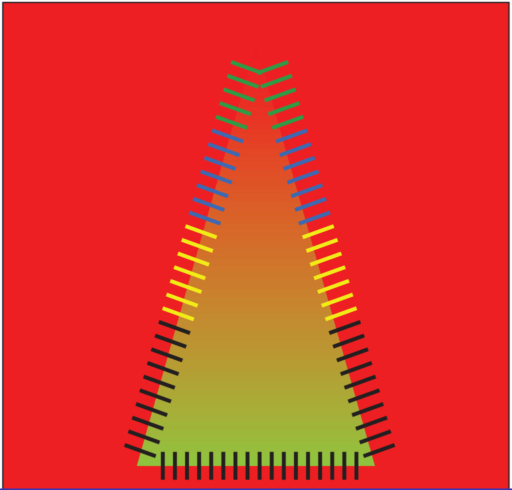
Shrinking bias in image segmentation

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



Shrinking bias in image segmentation

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



Addressing shrinking bias with edge submodularity

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

$$f_w(X) = w\Big(\{(u, v) \in E : u \in X, v \in V \setminus X\}\Big) \tag{1.52}$$

Addressing shrinking bias with edge submodularity

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

$$f_w(X) = w\Big(\{(u,v) \in E : u \in X, v \in V \setminus X\}\Big) \qquad (1.52)$$

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

$$f_g(X) = g\Big(\{(u,v) \in E : u \in X, v \in V \setminus X\}\Big) \qquad (1.53)$$

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.

$$f_w(X) = w\Big(\{(u,v) \in E : u \in X, v \in V \setminus X\}\Big) \tag{1.52}$$

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

$$f_g(X) = g\Big(\{(u,v) \in E : u \in X, v \in V \setminus X\}\Big) \tag{1.53}$$

- 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.

$$f_w(X) = w\Big(\{(u, v) \in E : u \in X, v \in V \setminus X\}\Big) \tag{1.52}$$

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

$$f_g(X) = g\Big(\{(u, v) \in E : u \in X, v \in V \setminus X\}\Big) \tag{1.53}$$

- 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

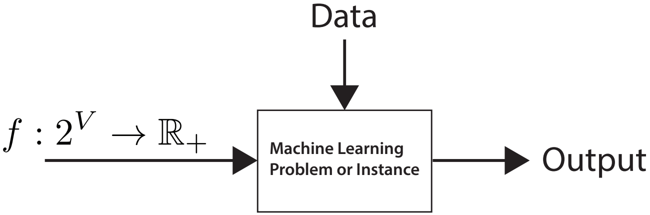
Graph Cut

Cooperative Cut

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

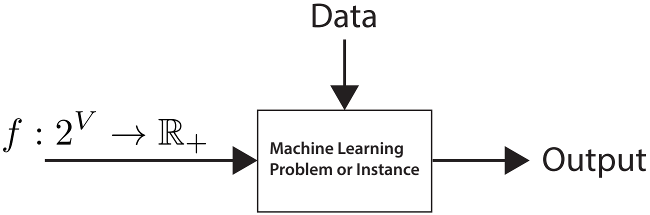
A submodular function as a parameter

- 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

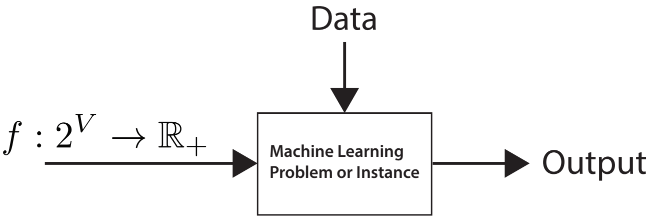
- In some cases, it may be useful to view a submodular function $f : 2^V \to \mathbb{R}$ as a input "parameter" to a machine learning algorithm.



- 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

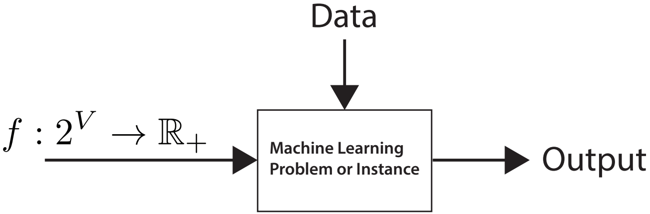
- In some cases, it may be useful to view a submodular function $f : 2^V \to \mathbb{R}$ as a input "parameter" to a machine learning algorithm.



- 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:

$$\min_{w \in \mathbb{R}^n} \frac{1}{m} \sum_{i=1}^m \ell(y_i, w^\intercal x_i) + \lambda \Omega(w), \tag{1.54}$$

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:

$$\min_{w \in \mathbb{R}^n} \frac{1}{m} \sum_{i=1}^m \ell(y_i, w^\intercal x_i) + \lambda \Omega(w), \tag{1.54}$$

  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:

$$\min_{w^1, \dots, w^k \in \mathbb{R}^n} \sum_{j=1}^k \frac{1}{m} \sum_{i=1}^m \ell(y_i^j, (w^j)^\intercal x_i) + \lambda \Omega(w^j), \tag{1.55}$$

Dictionary Learning and Selection

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

$$\min_{x_1,\dots,x_m} \min_{w^1,\dots,w^k \in \mathbb{R}^n} \sum_{j=1}^{k} \frac{1}{m} \sum_{i=1}^{m} \ell(y_i^j, (w^j)^\intercal x_i) + \lambda \Omega(w^j), \tag{1.56}$$

Dictionary Learning and Selection

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

$$\min_{x_1,\dots,x_m} \min_{w^1,\dots,w^k\in\mathbb{R}^n} \sum_{j=1}^{k} \frac{1}{m} \sum_{i=1}^{m} \ell(y_i^j, (w^j)^\intercal x_i) + \lambda\Omega(w^j), \tag{1.56}$$

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

$$f(D) = \sum_{j=1}^{k} \min_{S\subseteq D, |S|\leq k} \min_{w^j\in\mathbb{R}^S} \left(\sum_{i=1}^{m} \ell(y_i^j, (w^j)^\intercal x_i^S) + \lambda\Omega(w^j) \right) \tag{1.57}$$

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

$$\min_{x_1,\dots,x_m} \min_{w^1,\dots,w^k\in\mathbb{R}^n} \sum_{j=1}^{k} \frac{1}{m} \sum_{i=1}^{m} \ell(y_i^j, (w^j)^\intercal x_i) + \lambda\Omega(w^j), \tag{1.56}$$

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

$$f(D) = \sum_{j=1}^{k} \min_{S\subseteq D, |S|\leq k} \min_{w^j\in\mathbb{R}^S} \left(\sum_{i=1}^{m} \ell(y_i^j, (w^j)^\intercal x_i^S) + \lambda\Omega(w^j) \right) \tag{1.57}$$

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

$$\Omega(w) = \sum_{i=2}^{N} |w_i - w_{i-1}| \tag{1.58}$$

- 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

$$\tilde{f}(w) = \sum_{i=1}^{n} w_{\sigma_i}(f(\sigma_1, \ldots, \sigma_i) - f(\sigma_1, \ldots, \sigma_{i-1})) \tag{1.59}$$

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

$$\tilde{f}(w) = \sum_{i=1}^{n} w_{\sigma_i}(f(\sigma_1, \ldots, \sigma_i) - f(\sigma_1, \ldots, \sigma_{i-1})) \tag{1.59}$$

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

Submodular Generalized Dependence

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

$$f(A \cup B) = f(A) + f(B), \tag{1.60}$$

Submodular Generalized Dependence

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

$$f(A \cup B) = f(A) + f(B), \tag{1.60}$$

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

$$f(A \cup B \cup C) + f(C) = f(A \cup C) + f(B \cup C) \tag{1.61}$$

Submodular Generalized Dependence

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

$$f(A \cup B) = f(A) + f(B), \tag{1.60}$$

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

$$f(A \cup B \cup C) + f(C) = f(A \cup C) + f(B \cup C) \tag{1.61}$$

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

$$f(A|B) \triangleq f(A \cup B) - f(B) < f(A), \tag{1.62}$$

Submodular Generalized Dependence

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

$$f(A \cup B) = f(A) + f(B), \tag{1.60}$$

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

$$f(A \cup B \cup C) + f(C) = f(A \cup C) + f(B \cup C) \tag{1.61}$$

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

$$f(A|B) \triangleq f(A \cup B) - f(B) < f(A), \tag{1.62}$$

- 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., $A \perp\!\!\!\perp B$:

$$f(A \cup B) = f(A) + f(B), \quad (1.60)$$

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

$$f(A \cup B \cup C) + f(C) = f(A \cup C) + f(B \cup C) \quad (1.61)$$

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

$$f(A|B) \triangleq f(A \cup B) - f(B) < f(A), \quad (1.62)$$

- 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":

$$I_f(S_1; S_2; \ldots; S_k) = \sum_{i=1}^{k} f(S_k) - f(S_1 \cup S_2 \cup \cdots \cup S_k) \quad (1.63)$$

$$I'_f(S_1; S_2; \ldots; S_k) = \sum_{A \subseteq \{1,2,\ldots,k\}} (-1)^{|A|+1} f(\bigcup_{j \in A} S_j) \quad (1.64)$$

Submodular Parameterized Clustering

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

Submodular Parameterized Clustering

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

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.
- 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 Learning and Semi-Supervised Learning

- Given training data $\mathcal{D}_V = \{(x_i, y_i)\}_{i \in V}$ of (x, y) pairs where x is a query (data item) and y is an answer (label), goal is to learn a good mapping $y = h(x)$.

Active Learning and Semi-Supervised Learning

- Given training data $\mathcal{D}_V = \{(x_i, y_i)\}_{i \in V}$ of (x, y) pairs where x is a query (data item) and y is an answer (label), goal is to learn a good mapping $y = h(x)$.
- Often, getting y is time-consuming, expensive, and error prone (manual transcription, Amazon Turk, etc.)

Active Learning and Semi-Supervised Learning

- Given training data $\mathcal{D}_V = \{(x_i, y_i)\}_{i \in V}$ of (x, y) pairs where x is a query (data item) and y is an answer (label), goal is to learn a good mapping $y = h(x)$.
- Often, getting y is time-consuming, expensive, and error prone (manual transcription, Amazon Turk, etc.)
- Batch active learning: choose a subset $S \subset V$ so that only the labels $\{y_i\}_{i \in S}$ should be acquired.

Active Learning and Semi-Supervised Learning

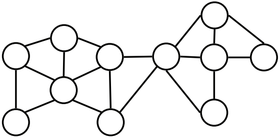
- Given training data $\mathcal{D}_V = \{(x_i, y_i)\}_{i \in V}$ of (x, y) pairs where x is a query (data item) and y is an answer (label), goal is to learn a good mapping $y = h(x)$.
- Often, getting y is time-consuming, expensive, and error prone (manual transcription, Amazon Turk, etc.)
- Batch active learning: choose a subset $S \subset V$ so that only the labels $\{y_i\}_{i \in S}$ should be acquired.
- Adaptive active learning: choose a policy whereby we choose an $i_1 \in V$, get the label y_{i_1}, choose another $i_2 \in V$, get label y_{i_2},where each chose can be based on previously acquired labels.

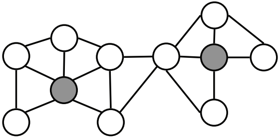
Active Learning and Semi-Supervised Learning

- Given training data $\mathcal{D}_V = \{(x_i, y_i)\}_{i \in V}$ of (x, y) pairs where x is a query (data item) and y is an answer (label), goal is to learn a good mapping $y = h(x)$.
- Often, getting y is time-consuming, expensive, and error prone (manual transcription, Amazon Turk, etc.)
- Batch active learning: choose a subset $S \subset V$ so that only the labels $\{y_i\}_{i \in S}$ should be acquired.
- Adaptive active learning: choose a policy whereby we choose an $i_1 \in V$, get the label y_{i_1}, choose another $i_2 \in V$, get label y_{i_2},where each chose can be based on previously acquired labels.
- Semi-supervised (transductive) learning: Once we have $\{y_i\}_{i \in S}$, infer the remaining labels $\{y_i\}_{i \in V \setminus S}$.

Active Transductive Semi-Supervised Learning

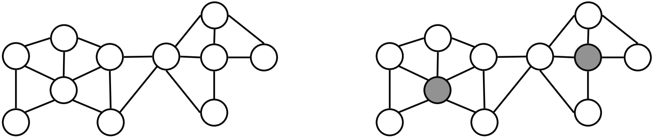
- Batch/Offline active learning: Given a set V of unlabeled data items, learner chooses subset $L \subseteq V$ of items to be labeled



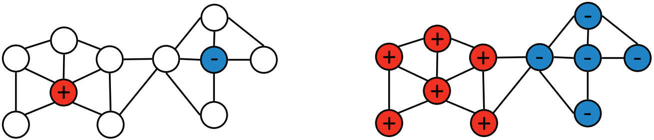


Active Transductive Semi-Supervised Learning

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

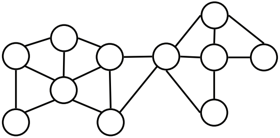
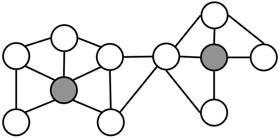


- Nature reveals labels $y_L \in \{0,1\}^L$, learner predicts labels $\hat{y} \in \{0,1\}^V$

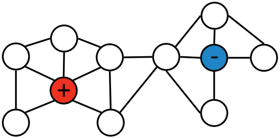
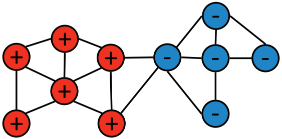


Active Transductive Semi-Supervised Learning

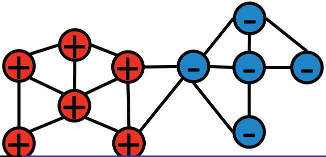
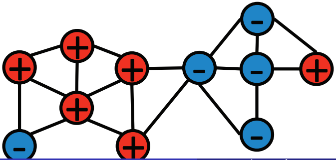
- Batch/Offline active learning: Given a set V of unlabeled data items, learner chooses subset $L \subseteq V$ of items to be labeled

- 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

$$\Psi(L) = \min_{T \subseteq V \setminus L : T \neq \emptyset} \frac{\Gamma(T)}{|T|} \tag{1.65}$$

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).

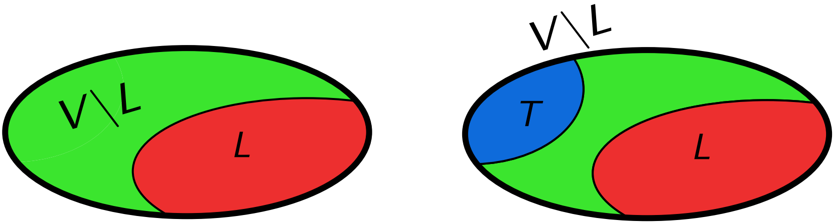
Choosing labels: how to select L

- Consider the following objective

$$\Psi(L) = \min_{T \subseteq V \setminus L : T \neq \emptyset} \frac{\Gamma(T)}{|T|} \quad (1.65)$$

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.



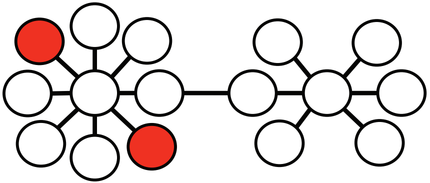
Choosing labels: how to select L

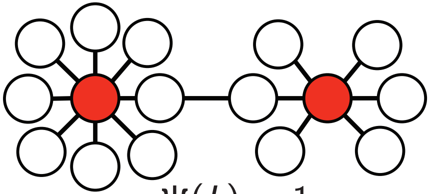
- Consider the following objective

$$\Psi(L) = \min_{T \subseteq V \setminus L : T \neq \emptyset} \frac{\Gamma(T)}{|T|} \tag{1.65}$$

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.





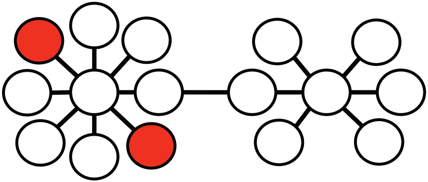
Choosing labels: how to select L

- Consider the following objective

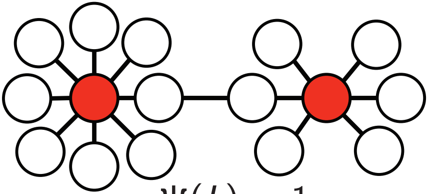
$$\Psi(L) = \min_{T \subseteq V \setminus L : T \neq \emptyset} \frac{\Gamma(T)}{|T|} \tag{1.65}$$

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 $A \subseteq V \setminus L$,

$$g(A) = \Gamma(A \cup \{v \in L : y_L(v) = 1\}) \tag{1.66}$$

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 $A \subseteq V \setminus L$,

$$g(A) = \Gamma(A \cup \{v \in L : y_L(v) = 1\}) \tag{1.66}$$

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

Generalized Error Bound

Theorem 1.8.1 (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

$$\|\hat{y} - y\|_1 \leq 2\frac{\Gamma(Y(y))}{\Psi(L)} \tag{1.67}$$

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:

$$\Psi(S) = \min_{T \subseteq V \setminus S : T \neq \emptyset} \frac{\Gamma(T)}{|T|} \tag{1.68}$$

- $\Gamma(T) = I_f(T; V \setminus T) = f(S) + f(V \setminus S) - f(V)$ 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:

$$d_\phi^{\mathcal{H}_\phi}(x, y) = \phi(x) - \phi(y) - \langle \mathcal{H}_\phi(y), x - y \rangle, \forall x, y \in \text{dom}(\phi) \quad (1.69)$$

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:

$$d_\phi^{\mathcal{H}_\phi}(x, y) = \phi(x) - \phi(y) - \langle \mathcal{H}_\phi(y), x - y \rangle, \forall x, y \in \text{dom}(\phi) \quad (1.69)$$

- 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:

$$d_\phi^{\mathcal{H}_\phi}(x, y) = \phi(x) - \phi(y) - \langle \mathcal{H}_\phi(y), x - y \rangle, \forall x, y \in \text{dom}(\phi) \quad (1.69)$$

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

$$d_f^{\mathcal{H}_f}(X, Y) = f(X) - f(Y) - \langle \mathcal{H}_f(Y), 1_X - 1_Y \rangle \quad (1.70)$$

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:

$$d_\phi^{\mathcal{H}_\phi}(x, y) = \phi(x) - \phi(y) - \langle \mathcal{H}_\phi(y), x - y \rangle, \forall x, y \in \text{dom}(\phi) \quad (1.69)$$

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

$$d_f^{\mathcal{H}_f}(X, Y) = f(X) - f(Y) - \langle \mathcal{H}_f(Y), 1_X - 1_Y \rangle \quad (1.70)$$

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:

$$d_\phi^{\mathcal{H}_\phi}(x, y) = \phi(x) - \phi(y) - \langle \mathcal{H}_\phi(y), x - y \rangle, \forall x, y \in \text{dom}(\phi) \quad (1.69)$$

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

$$d_f^{\mathcal{H}_f}(X, Y) = f(X) - f(Y) - \langle \mathcal{H}_f(Y), 1_X - 1_Y \rangle \quad (1.70)$$

  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.

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 $g : \mathbb{N} \rightarrow \mathbb{R}$?" 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 Learning of Submodular Mixtures

- Constraints specified in inference form:

$$\underset{\mathbf{w},\xi_t}{\text{minimize}} \quad \frac{1}{T}\sum_t \xi_t + \frac{\lambda}{2}\|\mathbf{w}\|^2 \tag{1.71}$$

$$\text{subject to} \quad \mathbf{w}^\top \mathbf{f}_t(\mathbf{y}^{(t)}) \geq \max_{\mathbf{y}\in\mathcal{Y}_t}\left(\mathbf{w}^\top \mathbf{f}_t(\mathbf{y}) + \ell_t(\mathbf{y})\right) - \xi_t, \forall t \tag{1.72}$$

$$\xi_t \geq 0, \forall t. \tag{1.73}$$

Structured Learning of Submodular Mixtures

- Constraints specified in inference form:

$$\underset{\mathbf{w},\xi_t}{\text{minimize}} \quad \frac{1}{T}\sum_t \xi_t + \frac{\lambda}{2}\|\mathbf{w}\|^2 \tag{1.71}$$

$$\text{subject to} \quad \mathbf{w}^\top \mathbf{f}_t(\mathbf{y}^{(t)}) \geq \max_{\mathbf{y}\in\mathcal{Y}_t}\Big(\mathbf{w}^\top \mathbf{f}_t(\mathbf{y}) + \ell_t(\mathbf{y})\Big) - \xi_t, \forall t \tag{1.72}$$

$$\xi_t \geq 0, \forall t. \tag{1.73}$$

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

Structured Learning of Submodular Mixtures

- Constraints specified in inference form:

$$\underset{\mathbf{w},\xi_t}{\text{minimize}} \quad \frac{1}{T}\sum_t \xi_t + \frac{\lambda}{2}\|\mathbf{w}\|^2 \tag{1.71}$$

$$\text{subject to} \quad \mathbf{w}^\top \mathbf{f}_t(\mathbf{y}^{(t)}) \geq \max_{\mathbf{y}\in\mathcal{Y}_t}\Big(\mathbf{w}^\top \mathbf{f}_t(\mathbf{y}) + \ell_t(\mathbf{y})\Big) - \xi_t, \forall t \tag{1.72}$$

$$\xi_t \geq 0, \forall t. \tag{1.73}$$

- 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:

$$\underset{\mathbf{w},\xi_t}{\text{minimize}} \quad \frac{1}{T}\sum_t \xi_t + \frac{\lambda}{2}\|\mathbf{w}\|^2 \tag{1.71}$$

$$\text{subject to} \quad \mathbf{w}^\top \mathbf{f}_t(\mathbf{y}^{(t)}) \geq \max_{\mathbf{y}\in\mathcal{Y}_t}\Big(\mathbf{w}^\top \mathbf{f}_t(\mathbf{y}) + \ell_t(\mathbf{y})\Big) - \xi_t, \forall t \tag{1.72}$$

$$\xi_t \geq 0, \forall t. \tag{1.73}$$

- 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:

$$\underset{\mathbf{w},\xi_t}{\text{minimize}} \quad \frac{1}{T}\sum_t \xi_t + \frac{\lambda}{2}\|\mathbf{w}\|^2 \tag{1.71}$$

$$\text{subject to} \quad \mathbf{w}^\top \mathbf{f}_t(\mathbf{y}^{(t)}) \geq \max_{\mathbf{y}\in\mathcal{Y}_t}\Big(\mathbf{w}^\top \mathbf{f}_t(\mathbf{y}) + \ell_t(\mathbf{y})\Big) - \xi_t, \forall t \tag{1.72}$$

$$\xi_t \geq 0, \forall t. \tag{1.73}$$

- 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.

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 1: Subgradient descent learning

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

$w_0 = 0$;

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

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

  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 $\frac{1}{T} \sum_t \mathbf{w}_t$.

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

$$\Pr(x) = \frac{1}{Z} \exp(-E(x)) \tag{1.74}$$

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

$$\Pr(x) = \frac{1}{Z} \exp(-E(x)) \tag{1.74}$$

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

$$\Pr(x) = \frac{1}{Z} \exp(-E(x)) \tag{1.74}$$

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):

$$\gamma_{U,k}(f) = \min_{L \subseteq U, S: |S| \leq k, S \cap L = \emptyset} \frac{\sum_{s \in S} f(x|L)}{f(S|L)} \tag{1.75}$$

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):

$$\gamma_{U,k}(f) = \min_{L \subseteq U, S : |S| \leq k, S \cap L = \emptyset} \frac{\sum_{s \in S} f(x|L)}{f(S|L)} \tag{1.75}$$

- 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):

$$\gamma_{U,k}(f) = \min_{L \subseteq U, S:|S| \leq k, S \cap L = \emptyset} \frac{\sum_{s \in S} f(x|L)}{f(S|L)} \tag{1.75}$$

- 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:

$$\text{Solution} \geq (1 - \frac{1}{e^{\gamma_{U^*,k}}})\text{OPT} \tag{1.76}$$

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):

$$\gamma_{U,k}(f) = \min_{L \subseteq U, S: |S| \leq k, S \cap L = \emptyset} \frac{\sum_{s \in S} f(x|L)}{f(S|L)} \tag{1.75}$$

- 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:

$$\text{Solution} \geq (1 - \frac{1}{e^{\gamma_{U^*,k}}})\text{OPT} \tag{1.76}$$

  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).

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):

$$\gamma_{U,k}(f) = \min_{L \subseteq U, S : |S| \leq k, S \cap L = \emptyset} \frac{\sum_{s \in S} f(x|L)}{f(S|L)} \tag{1.75}$$

- 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:

$$\text{Solution} \geq (1 - \frac{1}{e^{\gamma_{U^*,k}}})\text{OPT} \tag{1.76}$$

  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).
- Other analogous concepts: curvature of a submodular function, and also the submodular degree.