Submodular Functions, Optimization, and Applications to Machine Learning

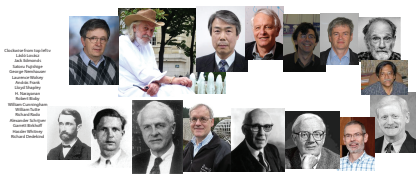
— Spring Quarter, Lecture 1 —

http://www.ee.washington.edu/people/faculty/bilmes/classes/ee563_spring_2018/

Prof. Jeff Bilmes

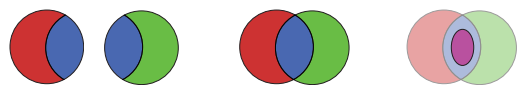
University of Washington, Seattle
Department of Electrical Engineering
http://melodi.ee.washington.edu/~bilmes

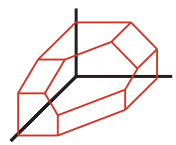
Mar 26th, 2018



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

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





Announcements

- Welcome to: Submodular Functions, Optimization, and Applications to Machine Learning, EE563.
- 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 Virtual Office Hours: Mondays, 10:00-11:00pm, via zoom (link posted on canvas).
- EEB 042, class web page is at our web page (`http://www.ee.washington.edu/people/faculty/bilmes/classes/ee563_spring_2018/`).
- Use our discussion board (`https://canvas.uw.edu/courses/1216339/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?
- Submodularity is an ideal model for cooperation, complexity, and attractiveness as well as for diversity, coverage, & information
- 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.

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 video 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 handouts and 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/1216339/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://www.ee.washington.edu/people/faculty/bilmes/classes/ee596b_spring_2016/`.

Cumulative Outstanding Reading

- Read chapter 1 from Fujishige's book.

Class Road Map - EE595 Spring 2016

- L1(3/28): Motivation, Applications, & Basic Definitions
- L2(3/30): Machine Learning Apps (diversity, complexity, parameter, learning target, surrogate).
- L3(4/4): Info theory exs, more apps, definitions, graph/combinatorial examples, matrix rank example, visualization
- L4(4/6): Graph and Combinatorial Examples, matrix rank, Venn diagrams, examples of proofs of submodularity, some useful properties
- L5(4/11): Examples & Properties, Other Defs., Independence
- L6(4/13): Independence, Matroids, Matroid Examples, matroid rank is submodular
- L7(4/18): Matroid Rank, More on Partition Matroid, System of Distinct Reps, Transversals, Transversal Matroid,
- L8(4/20): Transversals, Matroid and representation, Dual Matroids,
- L9(4/25): Dual Matroids, Properties, Combinatorial Geometries, Matroid and Greedy
- L10(4/27): Matroid and Greedy, Polyhedra, Matroid Polytopes,
- L11(5/2): From Matroids to Polymatroids, Polymatroids
- L12(5/4): Polymatroids, Polymatroids and Greedy
- L13(5/9): Polymatroids and Greedy; Possible Polytopes; Extreme Points; Polymatroids, Greedy, and Cardinality Constrained Maximization
- L14(5/11): Cardinality Constrained Maximization; Curvature; Submodular Max w. Other Constraints
- L15(5/16): Submodular Max w. Other Constraints, Most Violated $\leq$, Matroids cont., Closure/Sat,
- L16(5/18): Closure/Sat, Fund. Circuit/Dep,
- L17(5/23): Min-Norm Point and SFM, Min-Norm Point Algorithm,
- L18(5/25): Proof that min-norm gives optimal, Lovász extension.
- L19(6/1):
- L20(6/6): Final Presentations maximization.

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

Class Road Map - EE563

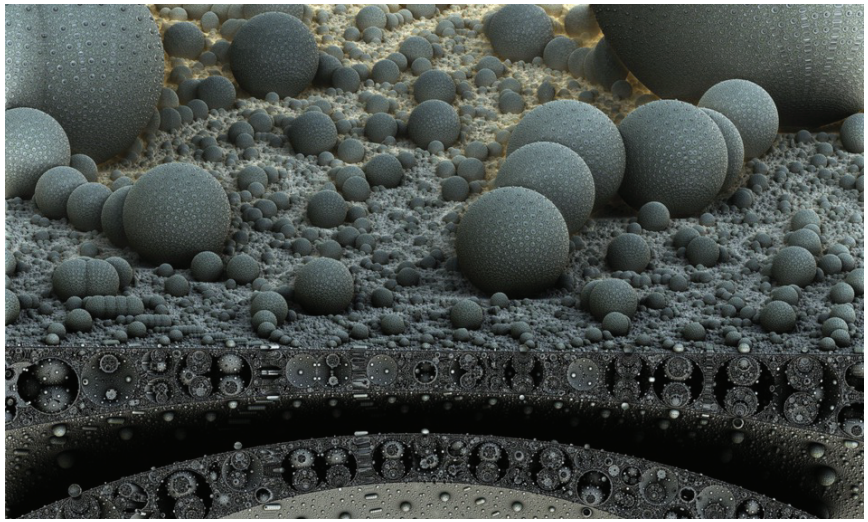
- L1(3/26): Motivation, Applications, & Basic Definitions,
- L2(3/28): Machine Learning Apps (diversity, complexity, parameter, learning target, surrogate).
- L3(4/2): Info theory exs, more apps, definitions, graph/combinatorial examples
- L4(4/4):
- L5(4/9):
- L6(4/11):
- L7(4/16):
- L8(4/18):
- L9(4/23):
- L10(4/25):
- L11(4/30):
- L12(5/2):
- L13(5/7):
- L14(5/9):
- L15(5/14):
- L16(5/16):
- L17(5/21):
- L18(5/23):
- L–(5/28): Memorial Day (holiday)
- L19(5/30):
- L21(6/4): Final Presentations maximization.

Last day of instruction, June 1st. Finals Week: June 2-8, 2018.

The Ideal Machine Learning Methods

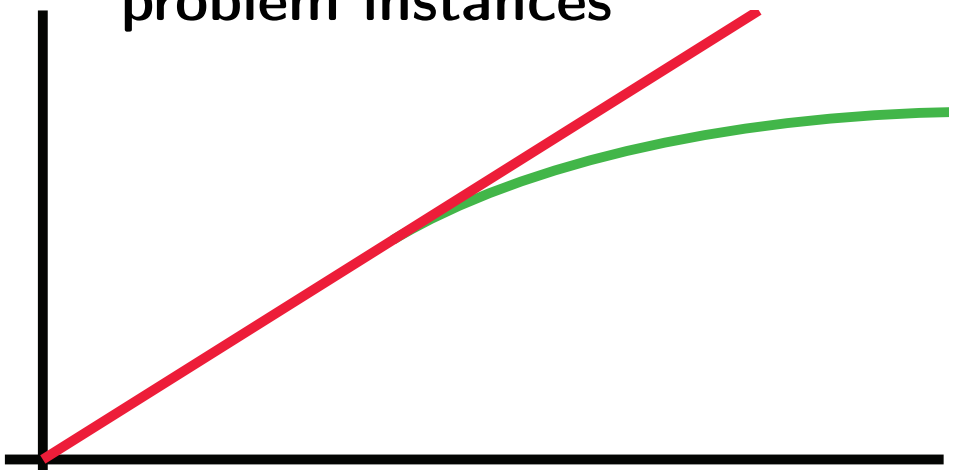
- Simple to define

- Mathematically rich

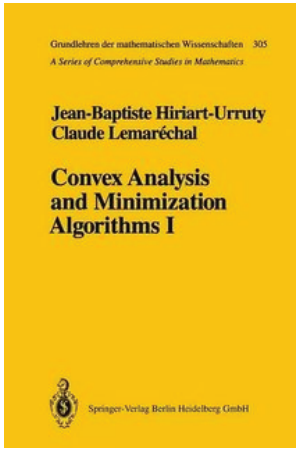


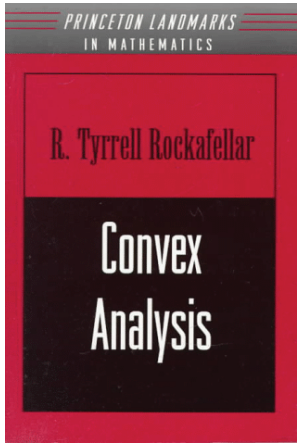
- Naturally suited to many real-world applications

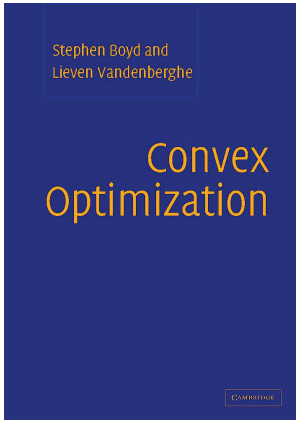
- Efficient & scalable to large problem instances

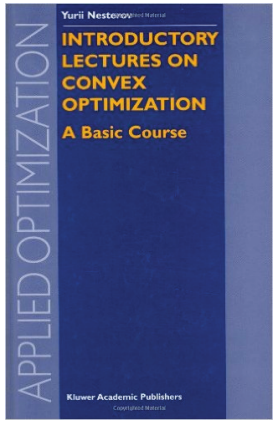


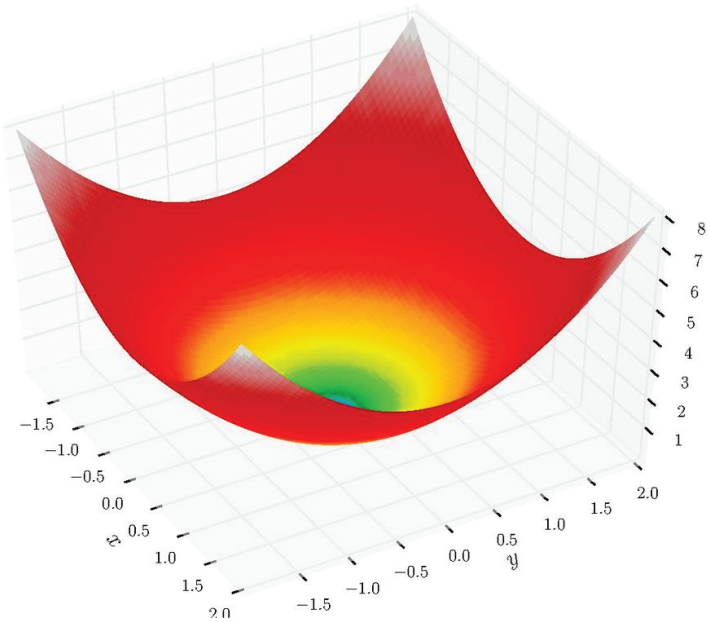
Convex Analysis in Machine Learning

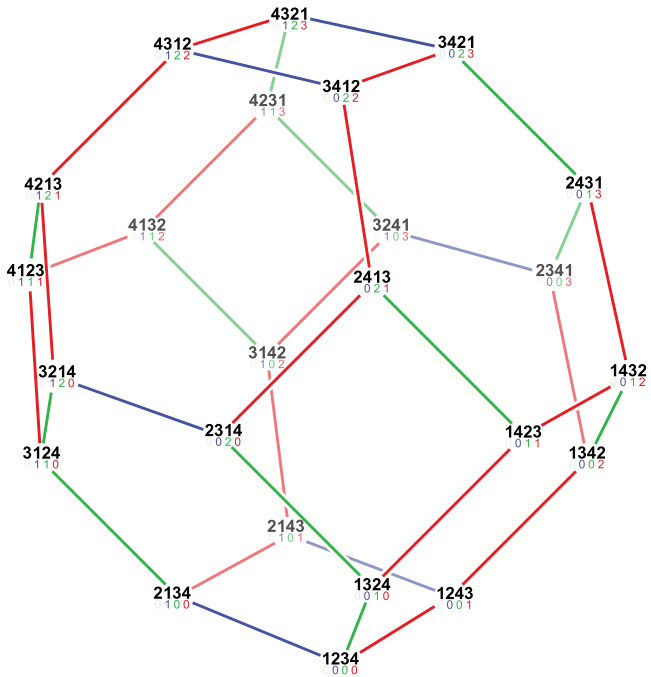












Successful Convexity in Machine Learning

- 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). Convex analysis for non-convex problems.

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, summarization; the science of data management.
- Sparse modeling, compressed sensing, low-rank approximation.
- Probabilistic models: structure learning in graphical models and neural networks. Non-graphical global potentials.
- Variable and feature 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 peptides, proteins, drug trial participants
- Genomics: cell-type or assay selection, genomic summarization
- Social networks, influence, viral marketing, information cascades, diffusion networks

Classic Discrete Optimization Problems

- **Operations Research/Industrial Engineering**: facility and factory location, packing and covering.
- **Sensor placement** where to optimally place sensors?
- **Information:** Information theory, sets of random variables.
- **Geometry**: Polytopes and polyhedra
- **Mathematics:** e.g., monge matrices, efficient dynamic programming, Birkhoff lattice theory
- **Combinatorial Problems**: e.g., sets, graphs, graph cuts, max k coverage, packings, coverings, partitions, paths, flows, matchings, colorings,
- **Algorithms**: Algorithms, and time/space complexity
- **Economics**: markets, economies of scale, mathematics of supply & demand

General Integer Programming (e.g., Integer Linear Programming (ILP), Integer Quadratic Programming (IQP), etc). General case can ignore 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

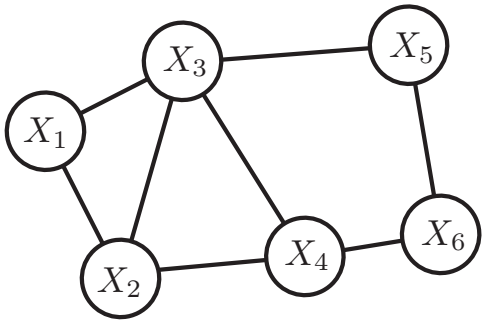
- In this course, we wish to demonstrate that submodular and supermodular functions also possess attractions of these four sorts as well.

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 $x \in \{0,1\}^n$

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

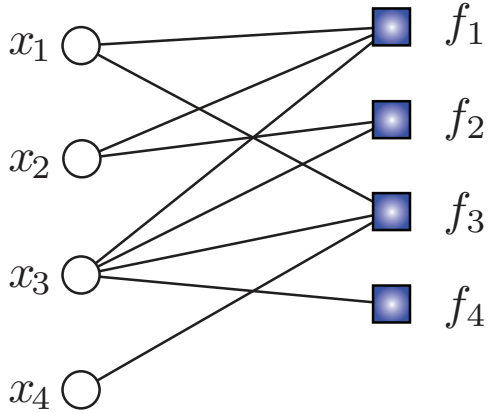
Example: Undirected Graphs



$$\begin{aligned} f(x_{1:6}) &= f(x_1, x_2, x_3) + f(x_2, x_3, x_4) \\ &\quad + f(x_3, x_5) + f(x_5, x_6) + f(x_4, x_6) \end{aligned}$$

$$\begin{aligned} f(x_{1:6}) &= f(x_1, x_2) + f(x_2, x_3) + f(x_3, x_1) \\ &\quad + f(x_2, x_3) + f(x_3, x_4) + f(x_4, x_2) \\ &\quad + f(x_3, x_5) + f(x_5, x_6) + f(x_4, x_6) \end{aligned}$$

Example: Factor/Hyper Graphs



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

Graphical Models/Decomposition: Real-Object Example

- How to valuate a set of items?

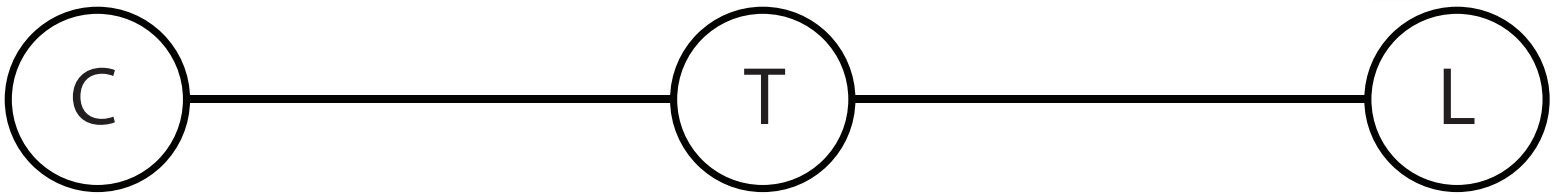
Graphical Models/Decomposition: Real-Object Example

- 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).



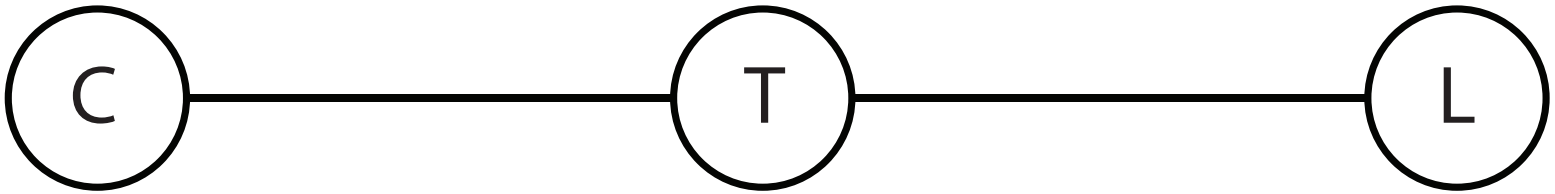


$$\text{value}(C, T, L) = \text{value}(C, T) + \text{value}(T, L) \tag{1.2}$$

Graphical Decomposition Limitation: Manner of Interaction

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



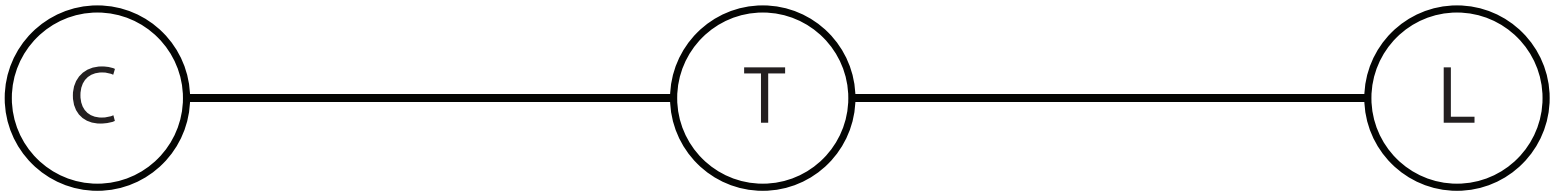


$$\text{value}(C,T,L) = \text{value}(C,T) + \text{value}(T,L) \tag{1.3}$$

Graphical Decomposition Limitation: Manner of Interaction

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





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

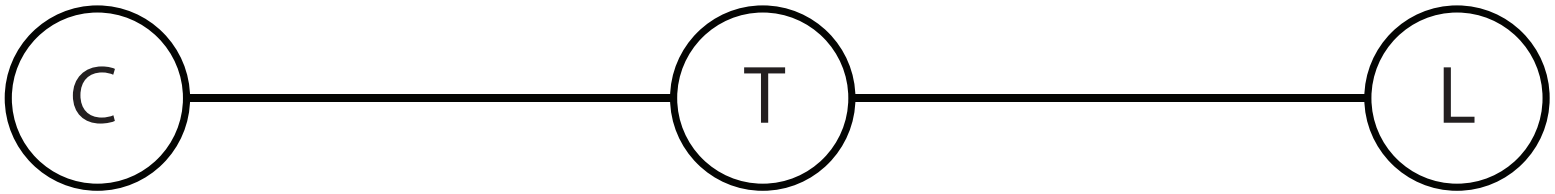
- Coffee and Tea are "substitutive"

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

Graphical Decomposition Limitation: Manner of Interaction

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





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

- Coffee and Tea are "substitutive"

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

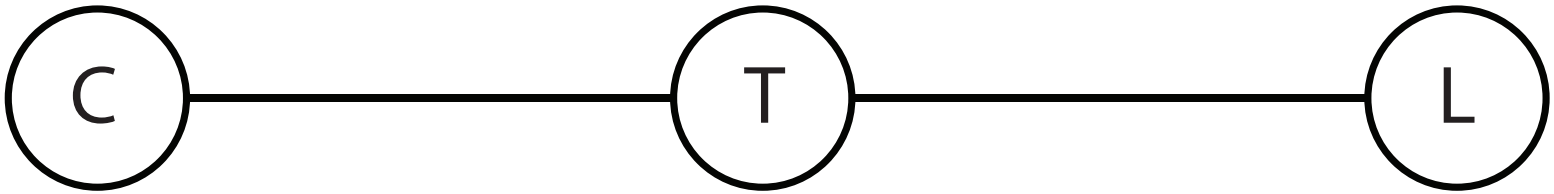
- Tea and Lemon are "complementary"

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

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

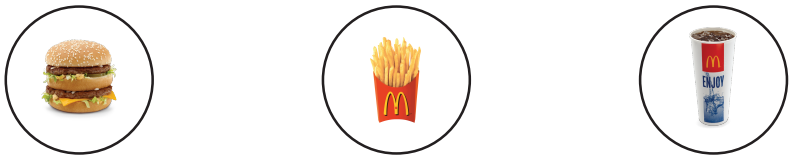
- Three items. Hamburger (H), Fries (F), Soda (S)

Options for Cost Models via Graphical Decomposition

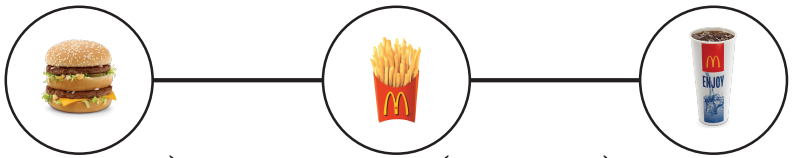
- Three items. Hamburger (H), Fries (F), Soda (S)



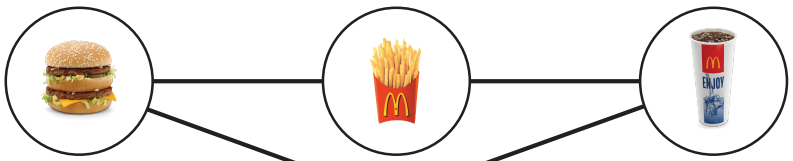
- Some graphical model options for costs(H, F, S):



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



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



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

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

$$\text{ccs}\left(\text{🍔🥤}\right) - \text{ccs}\left(\text{🍔}\right) \geq \text{ccs}\left(\text{🍟🥤}\right) - \text{ccs}\left(\text{🍟🍔}\right)$$

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

$$\text{ccs}(\,\text{  }\,) - \text{ccs}(\,\text{  }\,) \geq \text{ccs}(\,\text{  }\,) - \text{ccs}(\,\text{  }\,)$$

- Health costs

$$\text{hcs}(\,\text{  }\,) - \text{hcs}(\,\text{  }\,) \leq \text{hcs}(\,\text{  }\,) - \text{hcs}(\,\text{  }\,)$$

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

$$\text{ccs}(\,\_\,) - \text{ccs}(\,\_\,) \geq \text{ccs}(\,\_\,) - \text{ccs}(\,\_\,)$$

- Health costs

$$\text{hcs}(\,\_\,) - \text{hcs}(\,\_\,) \leq \text{hcs}(\,\_\,) - \text{hcs}(\,\_\,)$$

- In both cases, graphical-only decompositions fail!

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

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

$V =$ 

Also given a set function $f : 2^V \rightarrow \mathbb{R}$ 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:

$$A = \left\{ \right.$$

$$\left. \right\}$$

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

Subset $B \subseteq V$ of objects:

$$B = \left\{ \right.$$




$\left. \right\}$

Also given a set function $f : 2^V \rightarrow \mathbb{R}$ 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 $x(X) \stackrel{\Delta}{=} \mathbf{1}_X$.
- $f : \{0,1\}^V \to \{0,1\}$ 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:

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B) \tag{1.8}$$

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:

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

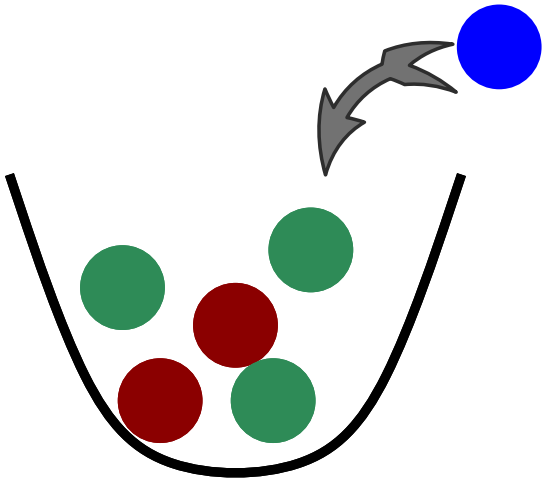
The incremental "value", "gain", or "cost" of v decreases (diminishes) as the context in which v is considered grows from A to B.

Example Submodular: Number of Colors of Balls in Urns

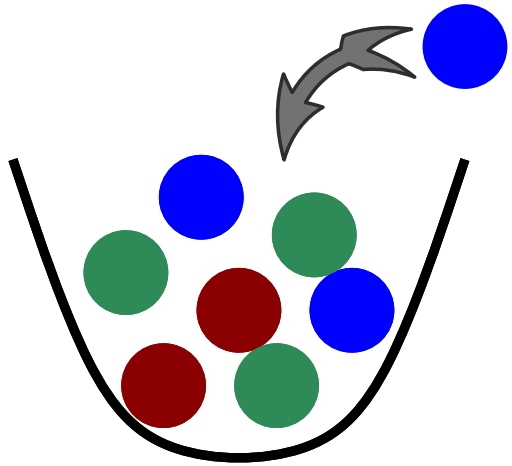
- Consider an urn containing colored balls. Given a set S of balls, $f(S)$ counts the number of distinct colors in S.

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.



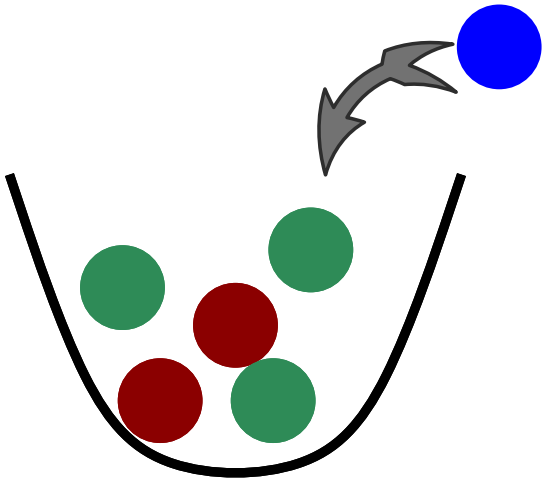
Initial value: 2 (colors in urn).
New value with added blue ball: 3



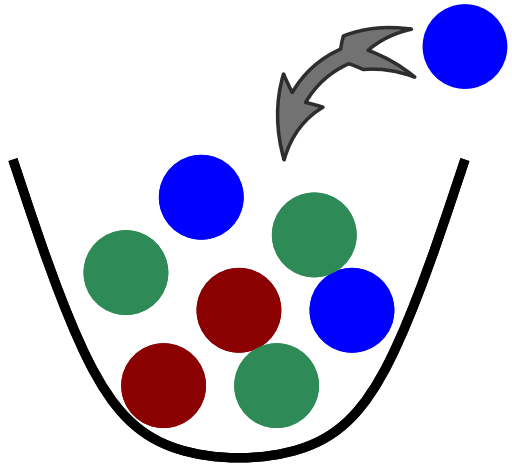
Initial value: 3 (colors in urn).
New value with added blue ball: 3

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

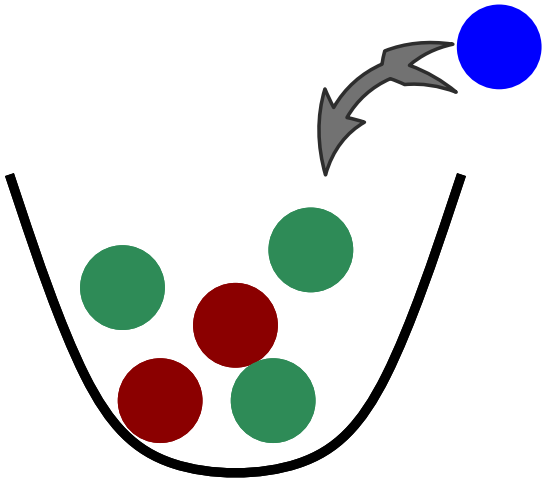


Initial value: 3 (colors in urn).
New value with added blue ball: 3

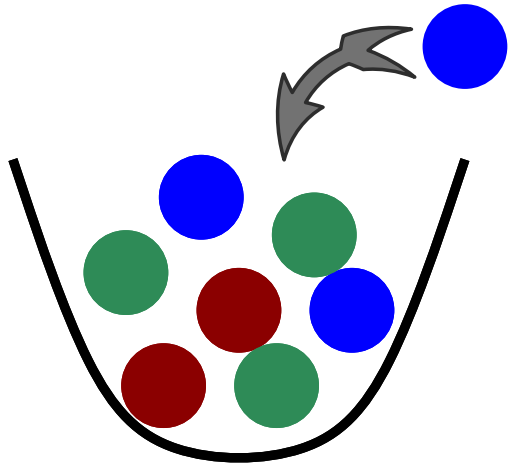
- Submodularity: Incremental Value of Object Diminishes in a Larger Context (diminishing returns).

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:

$$f(A) + f(B) \leq f(A \cup B) + f(A \cap B) \tag{1.10}$$

Definition 1.3.4 (supermodular (improving returns))

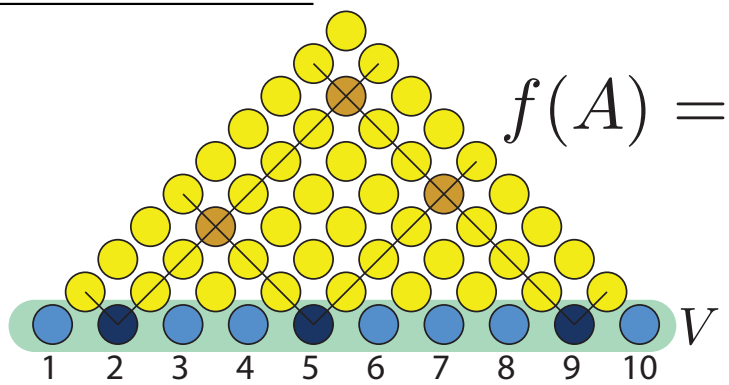
A function $f : 2^V \to \mathbb{R}$ is supermodular if for any $A \subseteq B \subset V$, and $v \in V \setminus B$, we have that:

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

- 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 $c = 0$).

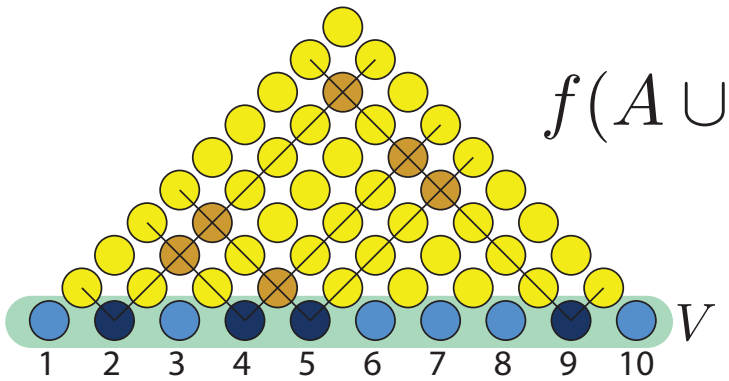
Example Supermodular: Number of Balls with Two Lines

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.



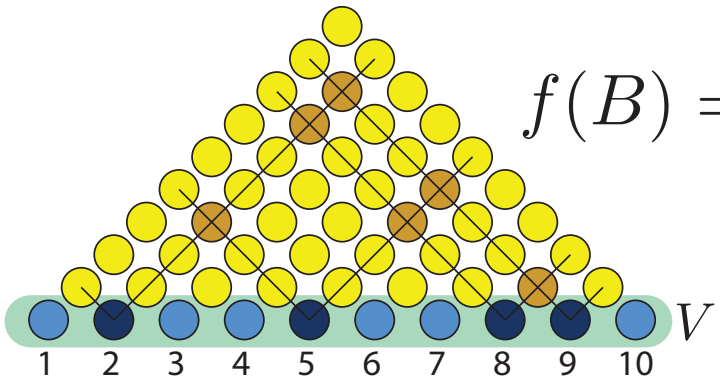
$f(A) = 3$

$A = \{2, 5, 9\}$



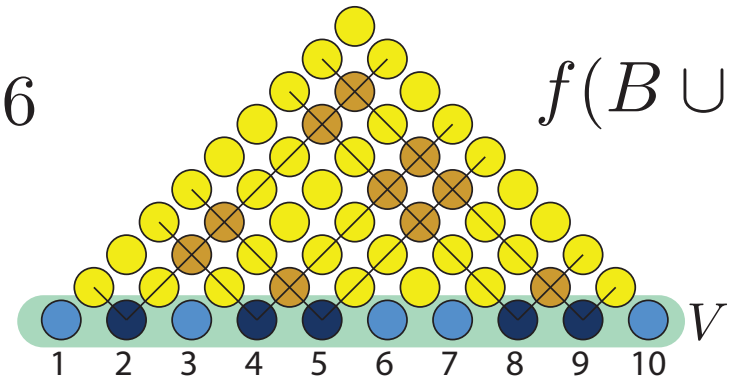
$f(A \cup \{4\}) = 6$

$A \cup \{4\} = \{2, 4, 5, 9\}$



$f(B) = 6$

$B = \{2, 5, 8, 9\}$



$f(B \cup \{4\}) = 10$

$B \cup \{4\} = \{2, 4, 5, 8, 9\}$

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 $A \subseteq V$,*

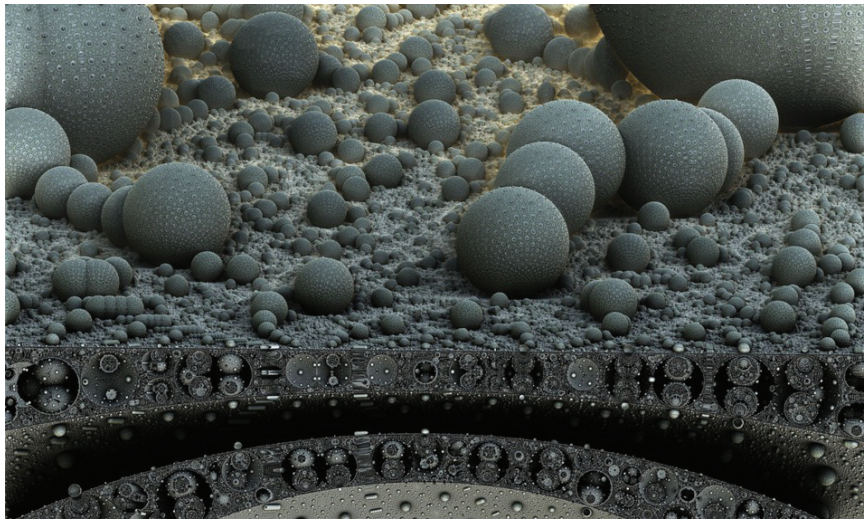
$$h(A) = f(A) + g(A) \tag{1.12}$$

- 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

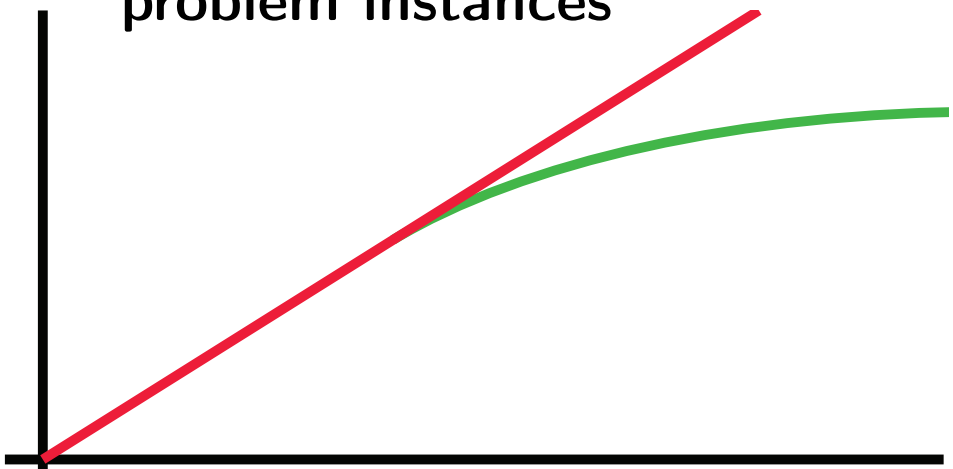
- Simple to define

- Mathematically rich



- Naturally suited to many real-world applications

- Efficient & scalable to large problem instances



Discrete Optimization

- Unconstrained minimization and maximization:

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

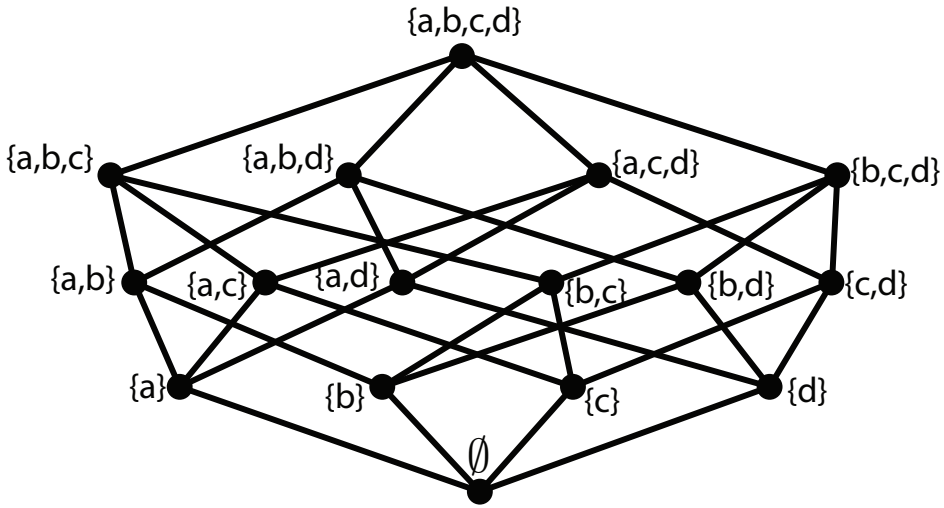
$$\max_{X \subseteq V} f(X) \tag{1.14}$$

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!!

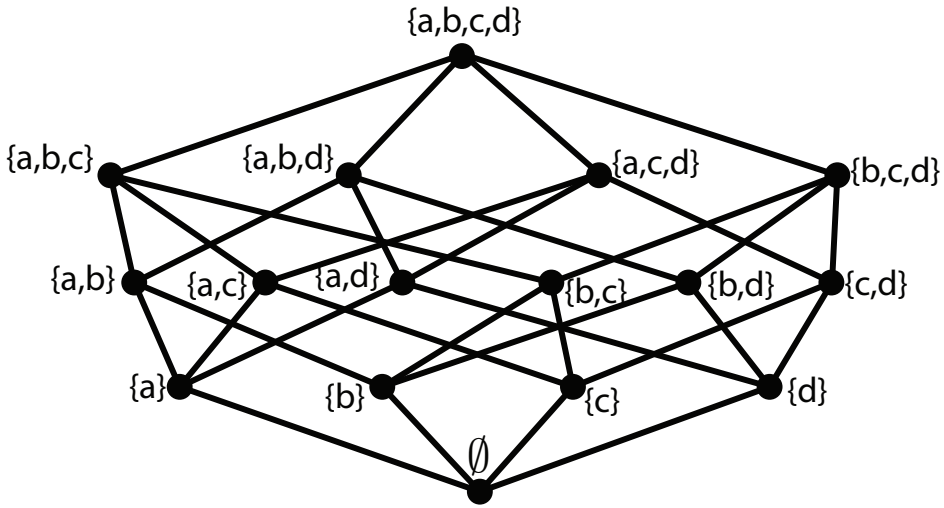


Discrete Optimization

- 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!!



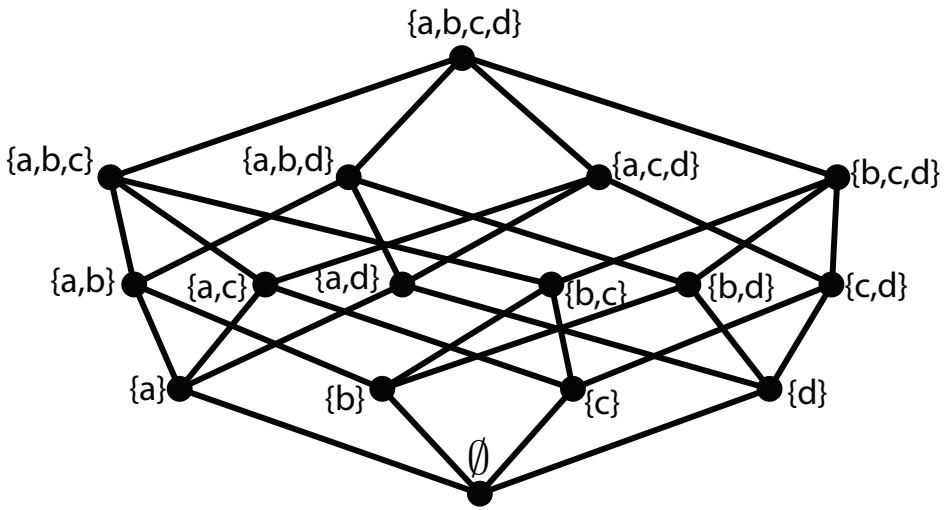
- Alternatively, we may partition V into (necessarily disjoint) blocks $\{V_1, V_2, \ldots\}$ that collectively are good in some way.

Discrete Optimization

- Unconstrained minimization and maximization:

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

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



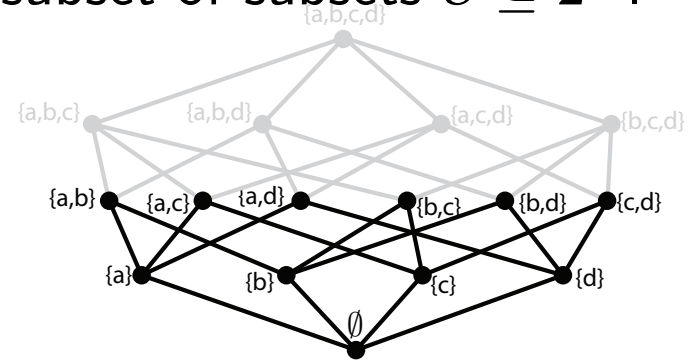
- Alternatively, we may partition V into (necessarily disjoint) blocks $\{V_1, V_2, \ldots\}$ that collectively are good in some way.
- When f is submodular, however, Eq. (1.13) is polytime, and Eq. (1.14) is constant-factor approximable. Partitionings are also approximable!

Constrained Discrete Optimization

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

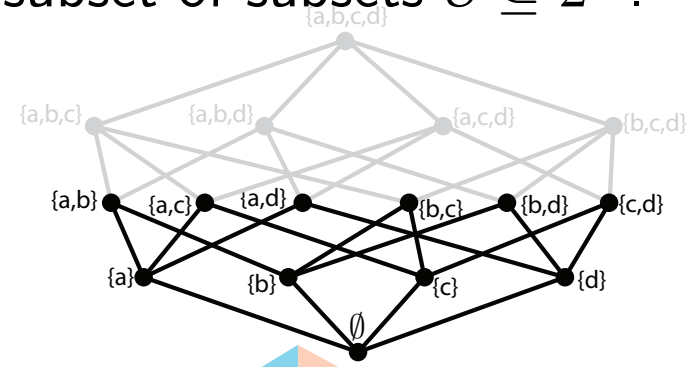
Constrained Discrete Optimization

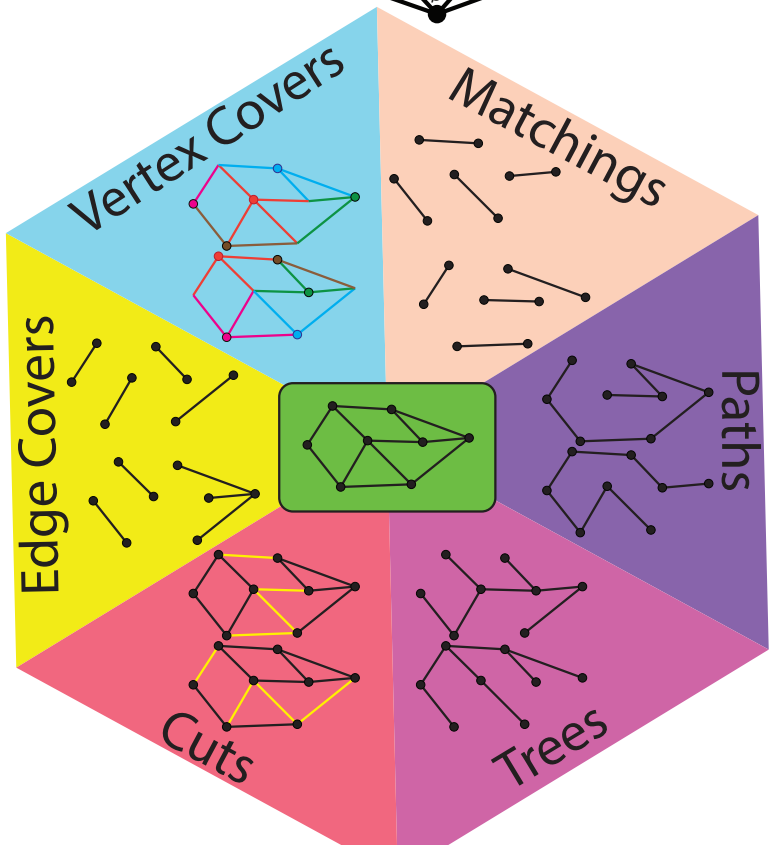
- Constrained case: interested only in a subset of subsets $\mathcal{S} \subseteq 2^V$.
- Ex: Bounded size $\mathcal{S} = \{S \subseteq V : |S| \leq k\}$, or given cost vector w and budget, bounded cost $\left\{S \subseteq V : \sum_{s \in S} w(s) \leq b\right\}$.



Constrained Discrete Optimization

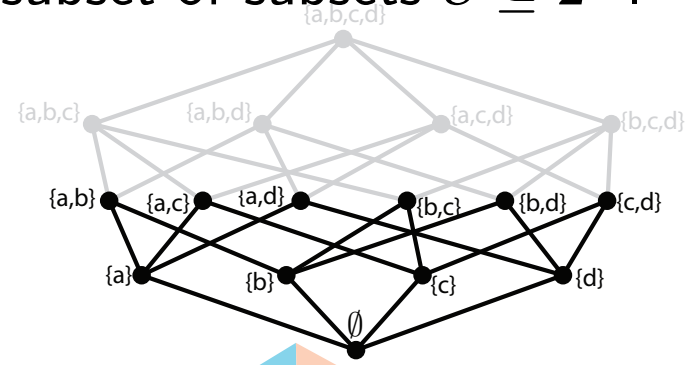
- Constrained case: interested only in a subset of subsets $\mathcal{S} \subseteq 2^V$.
- Ex: Bounded size $\mathcal{S} = \{S \subseteq V : |S| \leq k\}$, or given cost vector w and budget, bounded cost $\left\{S \subseteq V : \sum_{s \in S} w(s) \leq b\right\}$.
- Ex: feasible sets $\mathcal{S}$ as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

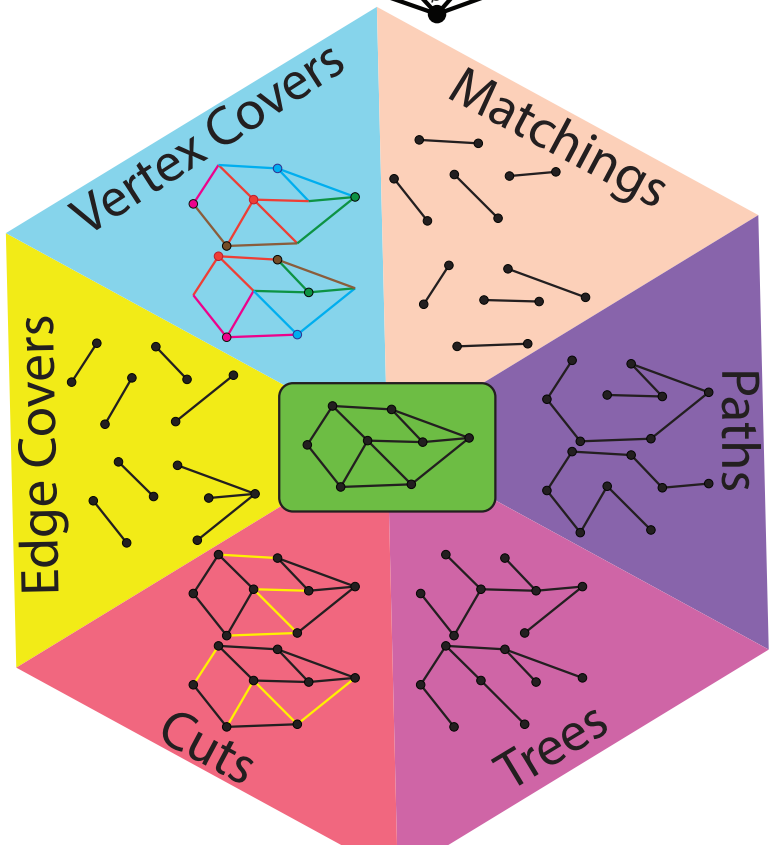




Constrained Discrete Optimization

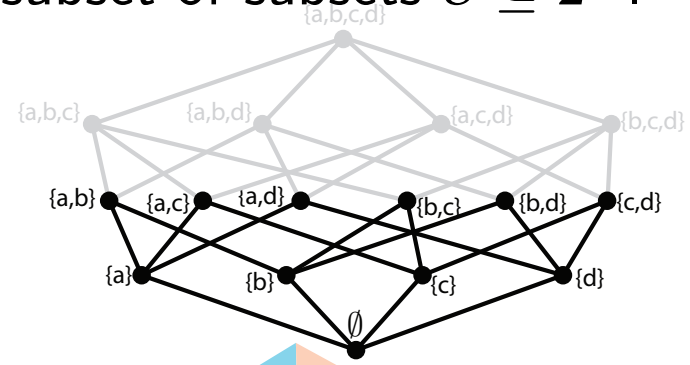
- Constrained case: interested only in a subset of subsets $\mathcal{S} \subseteq 2^V$.
- Ex: Bounded size $\mathcal{S} = \{S \subseteq V : |S| \leq k\}$, or given cost vector w and budget, bounded cost $\left\{S \subseteq V : \sum_{s \in S} w(s) \leq b\right\}$.
- Ex: feasible sets $\mathcal{S}$ as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



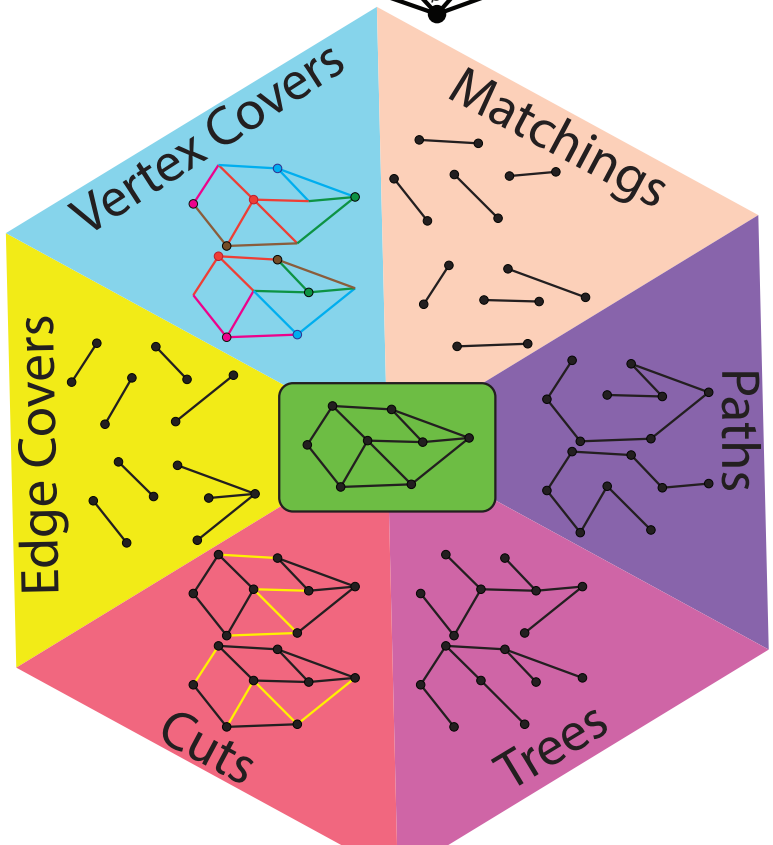


Constrained Discrete Optimization

- 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 $\left\{S \subseteq V : \sum_{s \in S} w(s) \leq b\right\}$.



- 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!

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.

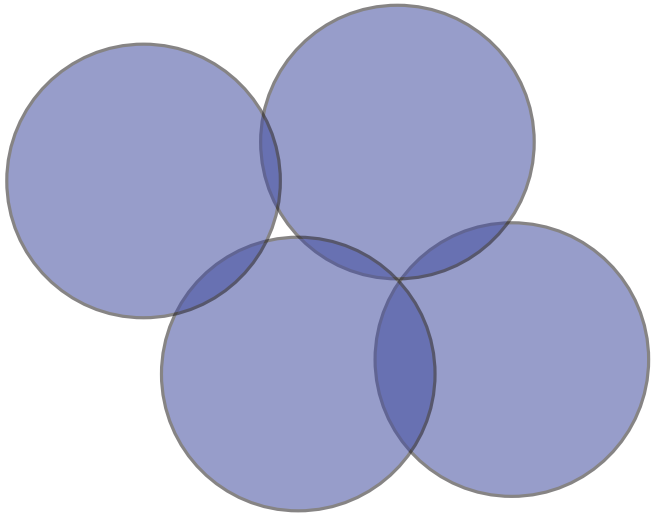
Continuous Set Cover

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 $f(S) = \bigcup_{s \in S} \text{area}(s)$ 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

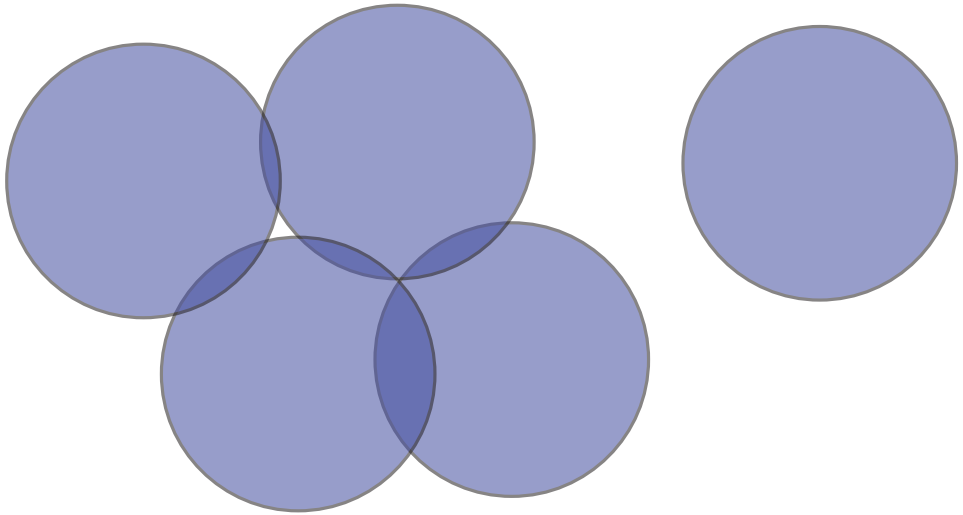


Union of areas of elements of A is given by:

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

Continuous Set Cover

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

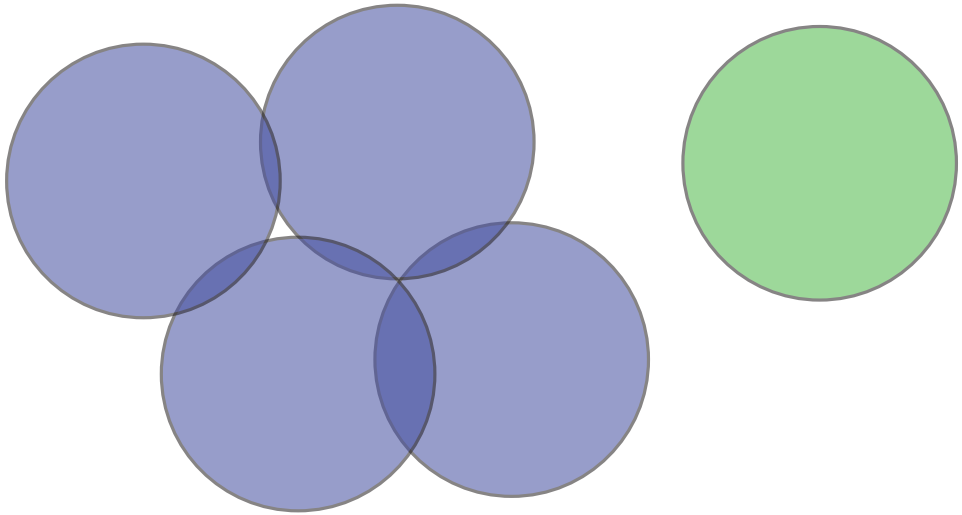


Area of A along with with v:

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

Continuous Set Cover

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



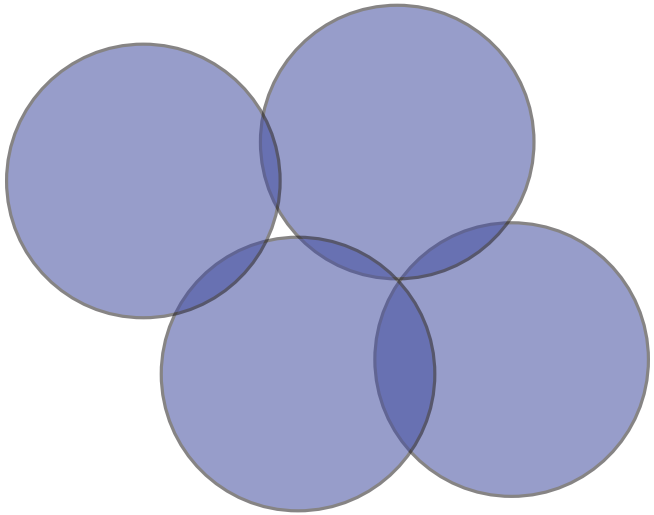
Gain (value) of v in context of A:

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

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

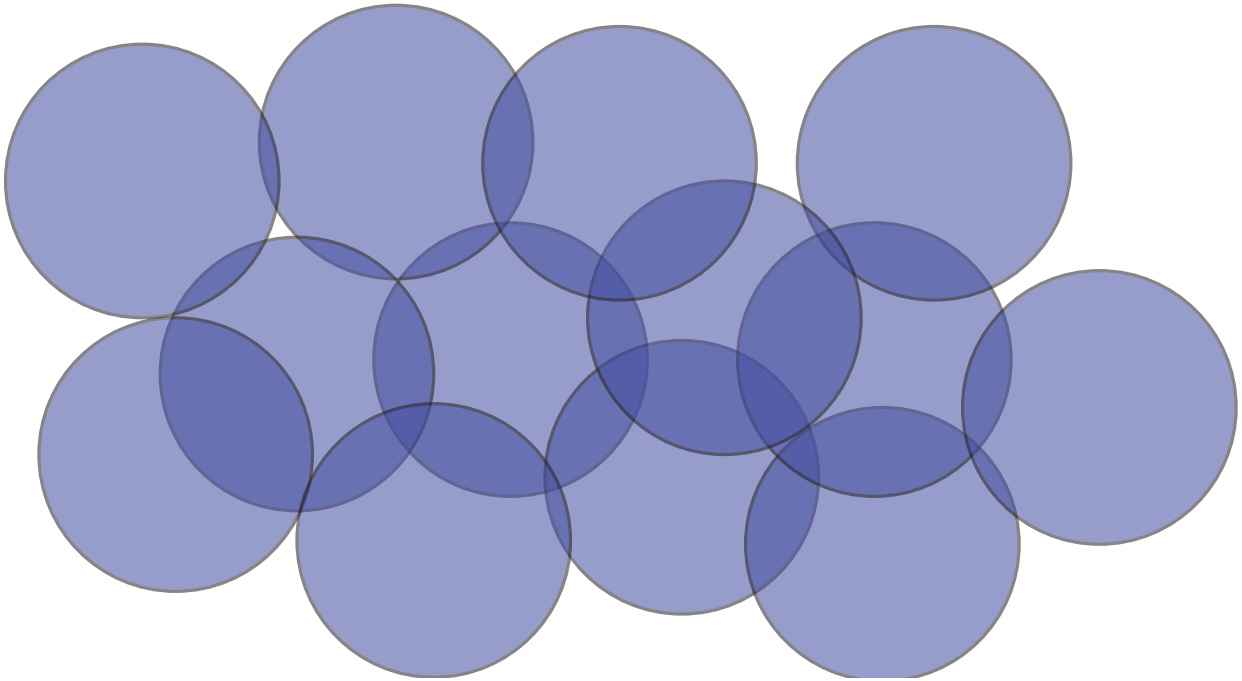


Area of A once again.

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

Continuous Set Cover

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

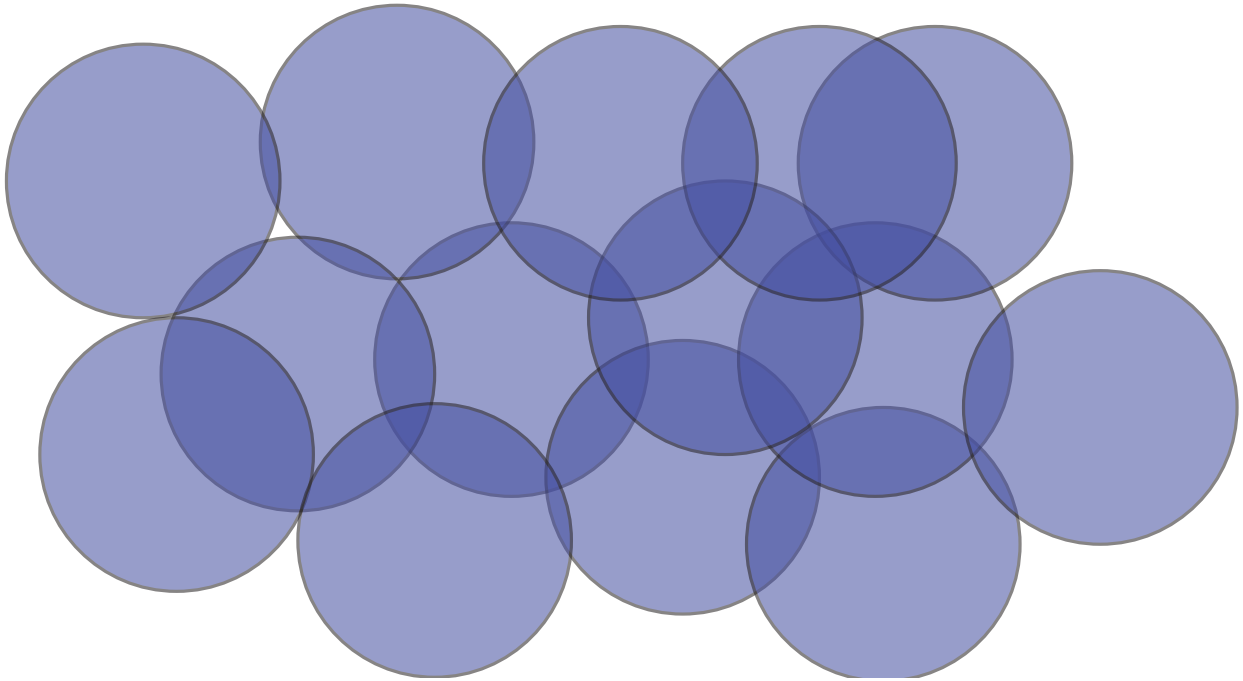


Union of areas of elements of $B \supset A$, where v is not included:

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

Continuous Set Cover

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

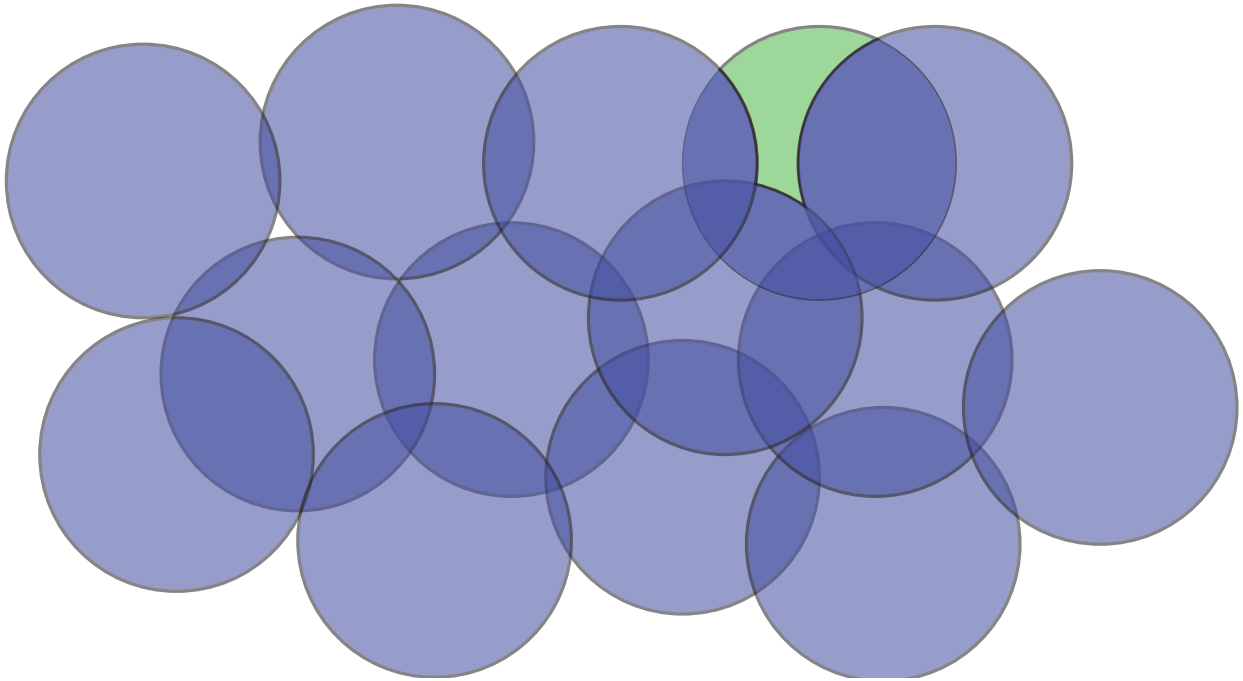


Area of B now also including v:

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

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

- 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

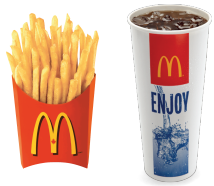
- Grocery store: finite set of items V that one can purchase.
- Each item $v \in V$ has a price $m(v)$.
- Basket of groceries $A \subseteq V$ costs:

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

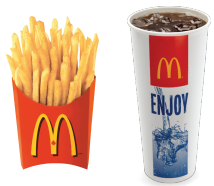
  the sum of individual item costs (no two-for-one discounts).
- This is known as a modular function.

Discounted Consumer Costs (as we saw earlier)

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

$f($  $) - f($  $) \geq f($  $) - f($  $)$

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

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

- 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

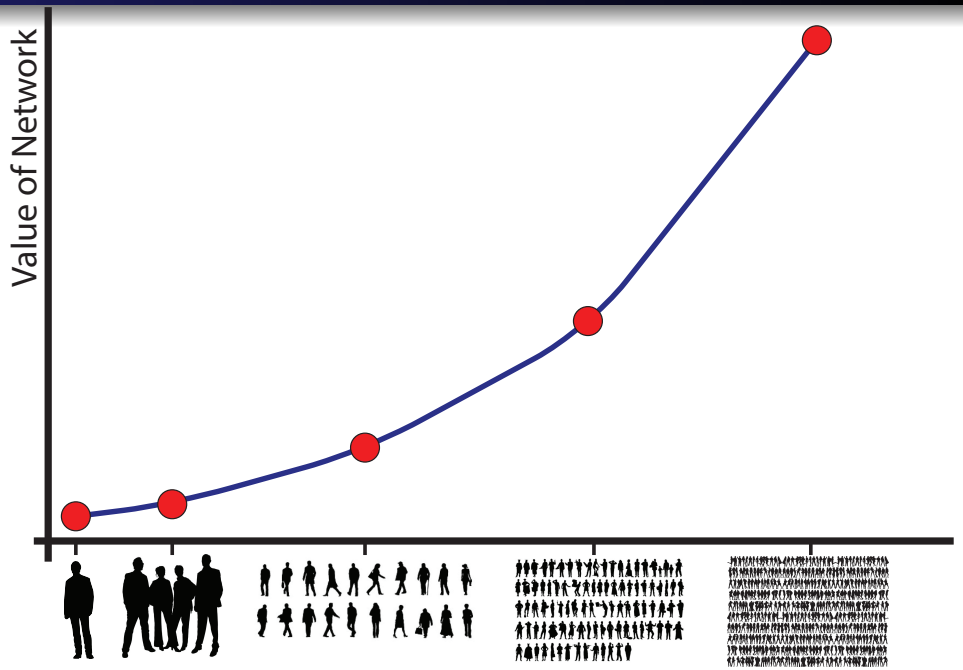
- What is a good model of the cost of manufacturing a set of items?
- Let V be a set of possible items to manufacture, and let $f(S)$ for $S \subseteq V$ be the manufacture costs of items in the subset S.
- Ex: V might be paint colors to produce: green, red, blue, yellow, white, etc.
- Producing green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

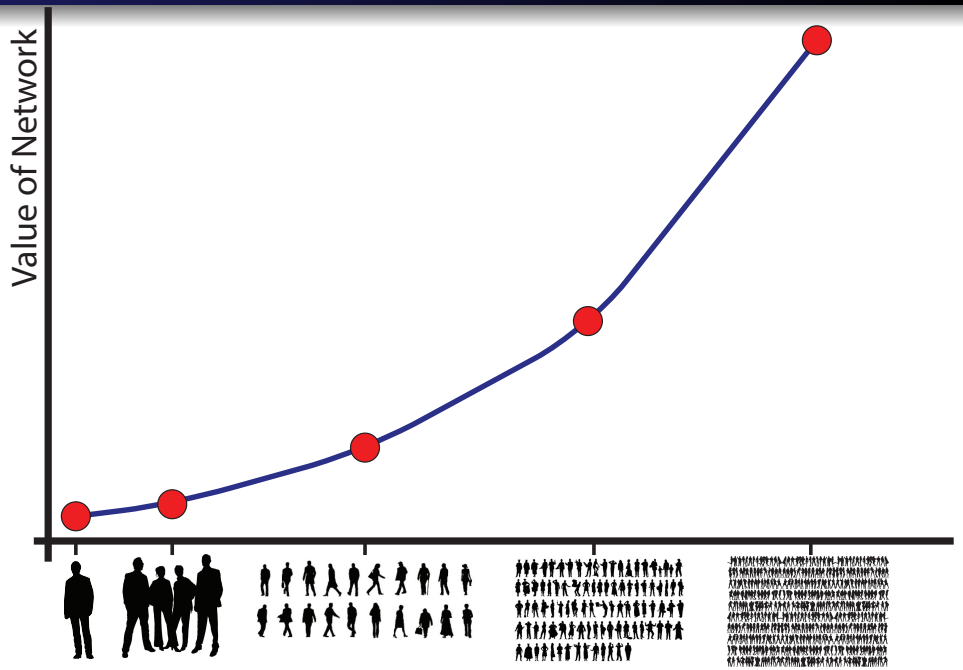
Demand side Economies of Scale: Network Externalities

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



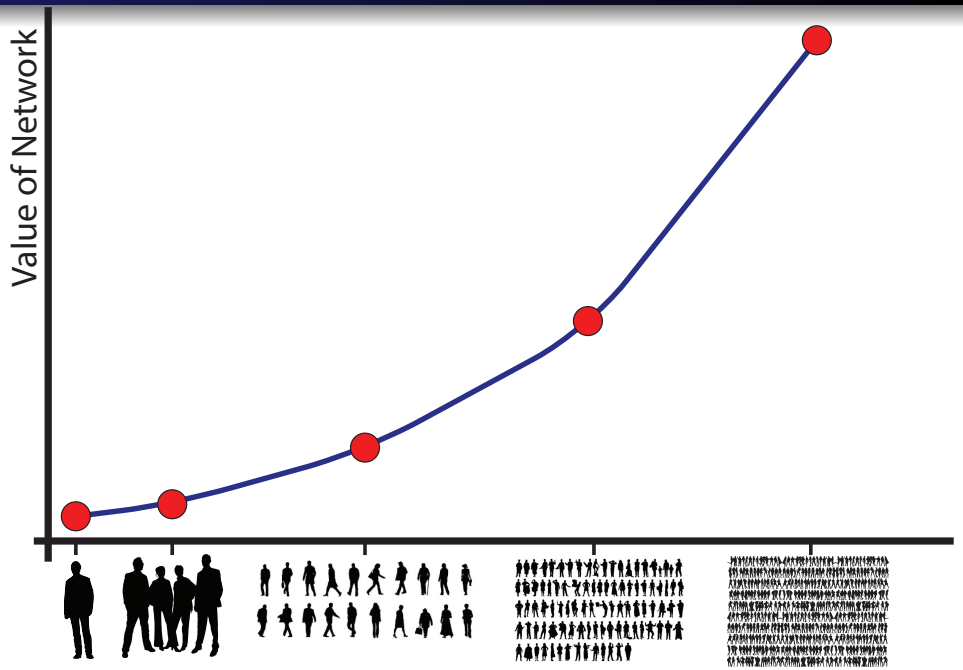
Demand side Economies of Scale: Network Externalities

- 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

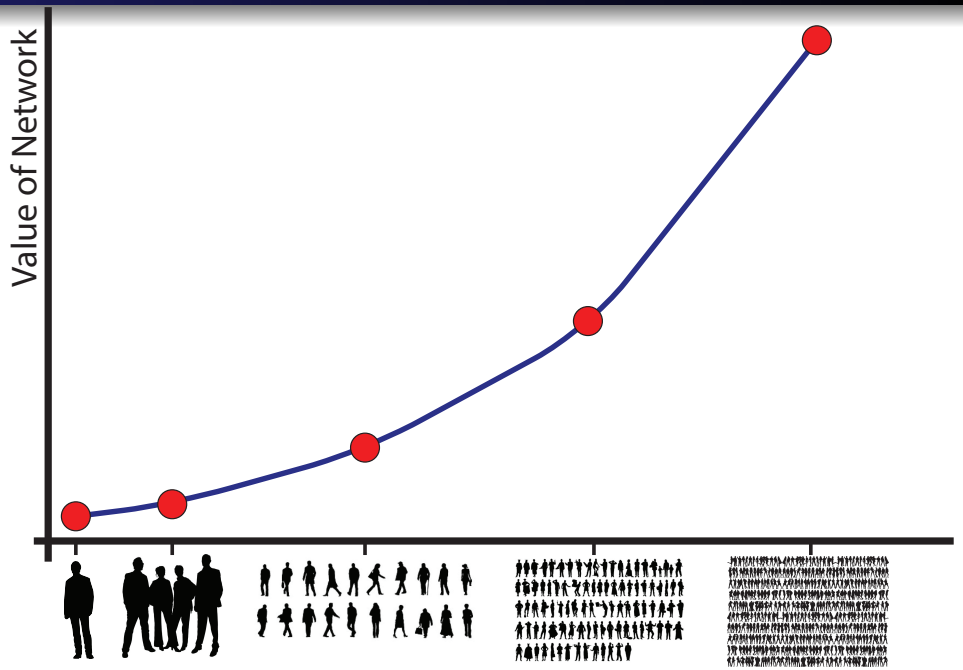
- Value of a network to a user depends on the number of other users in that network. External use benefits internal use.
- Consumers derive positive incremental value when size of the market for that good increases.



- 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

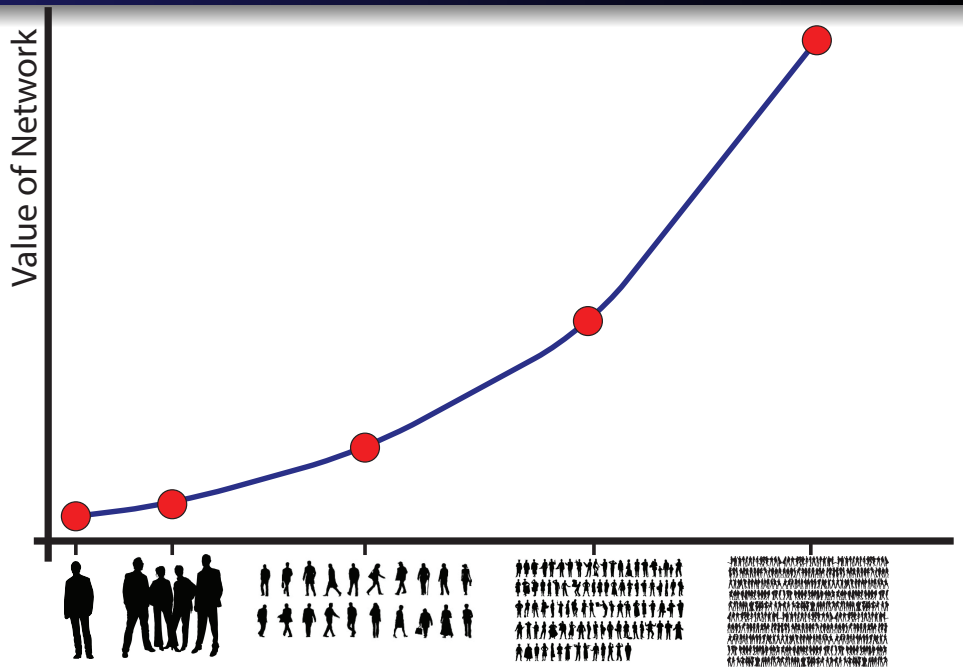
- Value of a network to a user depends on the number of other users in that network. External use benefits internal use.
- Consumers derive positive incremental value when size of the market for that good increases.



- 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?

Review So far

- Machine learning paradigms should be: easy to define, mathematically rich, naturally applicable, and efficient/scalable.
- Convexity (continuous structures) and graphical models (based on factorization or additive separation) are two such modeling paradigms.
- Submodularity/supermodularity offer a distinct mathematically rich paradigm over discrete space that neither need be continous nor be additively additively separable,
- submodularity offers forms of structural decomposition, e.g., $h = f + g$, into potentially global (manner of interaction) terms.
- Set cover, supply and demand side economies of scale,

Submodularity's utility in ML

- A model of a physical process:

Submodularity's utility in ML

- A model of a physical process:
  - When maximizing, submodularity naturally models: diversity, coverage, span, and information.

Submodularity's utility in ML

- A model of a physical process:
  - When maximizing, submodularity naturally models: diversity, coverage, span, and information.
  - When minimizing, submodularity naturally models: cooperative costs, complexity, roughness, and irregularity.

Submodularity's utility in ML

- A model of a physical process:
  - When maximizing, submodularity naturally models: diversity, coverage, span, and information.
  - When minimizing, submodularity naturally models: cooperative costs, complexity, roughness, and irregularity.
  - vice-versa for supermodularity.

Submodularity's utility in ML

- A model of a physical process:
  - When maximizing, submodularity naturally models: diversity, coverage, span, and information.
  - When minimizing, submodularity naturally models: cooperative costs, complexity, roughness, and irregularity.
  - vice-versa for supermodularity.
- 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:
  - When maximizing, submodularity naturally models: diversity, coverage, span, and information.
  - When minimizing, submodularity naturally models: cooperative costs, complexity, roughness, and irregularity.
  - vice-versa for supermodularity.
- 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:
  - When maximizing, submodularity naturally models: diversity, coverage, span, and information.
  - When minimizing, submodularity naturally models: cooperative costs, complexity, roughness, and irregularity.
  - vice-versa for supermodularity.
- 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:
  - When maximizing, submodularity naturally models: diversity, coverage, span, and information.
  - When minimizing, submodularity naturally models: cooperative costs, complexity, roughness, and irregularity.
  - vice-versa for supermodularity.
- 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:
  - When maximizing, submodularity naturally models: diversity, coverage, span, and information.
  - When minimizing, submodularity naturally models: cooperative costs, complexity, roughness, and irregularity.
  - vice-versa for supermodularity.
- 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:
  - When maximizing, submodularity naturally models: diversity, coverage, span, and information.
  - When minimizing, submodularity naturally models: cooperative costs, complexity, roughness, and irregularity.
  - vice-versa for supermodularity.
- 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.