Submodular Functions, Optimization, and Applications to Machine Learning

— Spring Quarter, Lecture 2 —

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

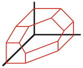
Prof. Jeff Bilmes

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

Mar 28th, 2018

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

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



Cumulative Outstanding Reading

- Read chapter 1 from Fujishige's book.

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):
- 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.

Two Equivalent Submodular Definitions

Definition 2.2.1 (submodular concave)

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

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B) \tag{2.8}$$

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

Definition 2.2.2 (diminishing returns)

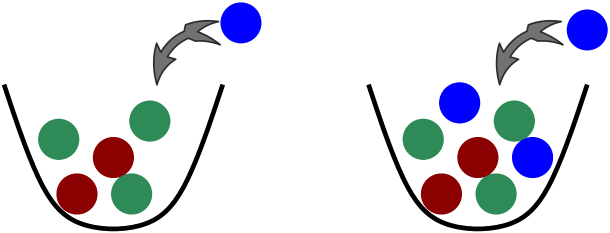
A function $f : 2^V \to \mathbb{R}$ is submodular if for any $A \subseteq B \subset V$, and $v \in V \setminus B$, we have that:

$$f(A \cup \{v\}) - f(A) \geq f(B \cup \{v\}) - f(B) \tag{2.9}$$

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.



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 2.2.1 (supermodular)

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

$$f(A) + f(B) \leq f(A \cup B) + f(A \cap B) \tag{2.8}$$

Definition 2.2.2 (supermodular (improving returns))

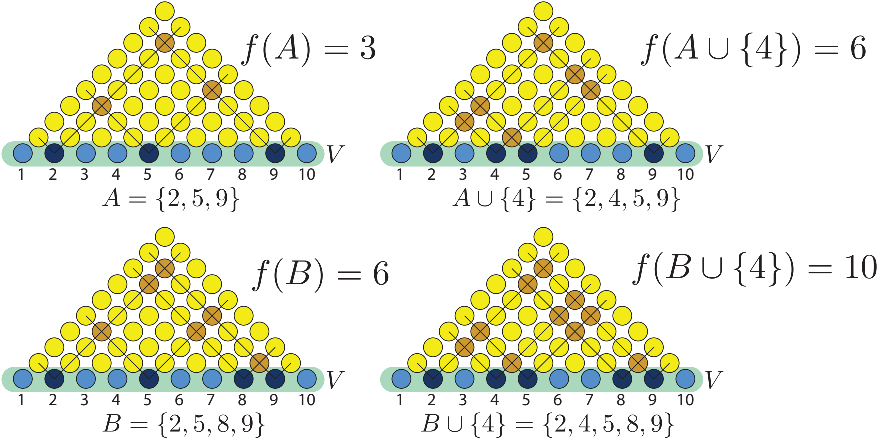
A function $f : 2^V \to \mathbb{R}$ is supermodular if for any $A \subseteq B \subset V$, and $v \in V \setminus B$, we have that:

$$f(A \cup \{v\}) - f(A) \leq f(B \cup \{v\}) - f(B) \tag{2.9}$$

- 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.



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:
  - 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.

Many different functions are submodular!

- We will see many applications of submodularity in machine learning.
- On next set of slides, we will state (without proof, for now) that many of the functions are submodular (or supermodular).
- In subsequent lectures, we will start showing how to prove submodularity.

Functions to Measure Diversity

Diversity is good, especially when it is high

- Quantitative measurement diversity in data science and ML. Goal of diversity: ensure small set properly represents the large.

Functions to Measure Diversity

Diversity is good, especially when it is high

- Quantitative measurement diversity in data science and ML. Goal of diversity: ensure small set properly represents the large.
- Web search: given ambiguous search term (e.g., "jaguar") with no other information, one wants articles more than just about cars.
  - Try google searching for words (e.g., "break") with many meanings (`http://muse.dillfrog.com/lists/ambiguous`), how well does google's diversity measure do?

Functions to Measure Diversity

Diversity is good, especially when it is high

- Quantitative measurement diversity in data science and ML. Goal of diversity: ensure small set properly represents the large.
- Web search: given ambiguous search term (e.g., "jaguar") with no other information, one wants articles more than just about cars.
  - Try google searching for words (e.g., "break") with many meanings (`http://muse.dillfrog.com/lists/ambiguous`), how well does google's diversity measure do?
  - Overall goal: user quickly finds informative, concise, accurate, relevant, comprehensive information ⇒ diversity

Functions to Measure Diversity

Diversity is good, especially when it is high

- Quantitative measurement diversity in data science and ML. Goal of diversity: ensure small set properly represents the large.
- Web search: given ambiguous search term (e.g., "jaguar") with no other information, one wants articles more than just about cars.
    - Try google searching for words (e.g., "break") with many meanings (`http://muse.dillfrog.com/lists/ambiguous`), how well does google's diversity measure do?
    - Overall goal: user quickly finds informative, concise, accurate, relevant, comprehensive information ⇒ diversity
- 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? Answer: submodular maximization.

Functions to Measure Diversity

Diversity is good, especially when it is high

- Quantitative measurement diversity in data science and ML. Goal of diversity: ensure small set properly represents the large.
- Web search: given ambiguous search term (e.g., "jaguar") with no other information, one wants articles more than just about cars.
  - Try google searching for words (e.g., "break") with many meanings (`http://muse.dillfrog.com/lists/ambiguous`), how well does google's diversity measure do?
  - Overall goal: user quickly finds informative, concise, accurate, relevant, comprehensive information $\Rightarrow$ diversity
- 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? Answer: submodular maximization.
- How do we choose the smallest set S that maintains a given degree of diversity? Constrained minimization (i.e., $\min |A|$ s.t. $f(A) \geq \alpha$).

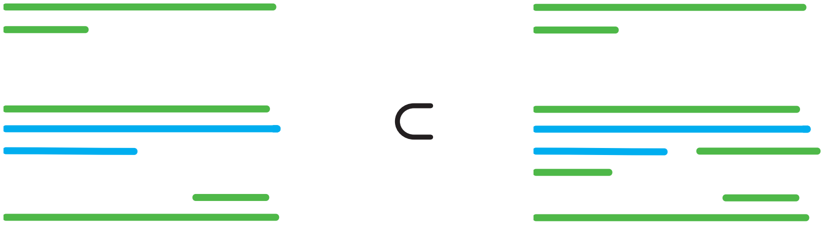
Functions to Measure Diversity

Diversity is good, especially when it is high

- Quantitative measurement diversity in data science and ML. Goal of diversity: ensure small set properly represents the large.
- Web search: given ambiguous search term (e.g., "jaguar") with no other information, one wants articles more than just about cars.
  - Try google searching for words (e.g., "break") with many meanings (`http://muse.dillfrog.com/lists/ambiguous`), how well does google's diversity measure do?
  - Overall goal: user quickly finds informative, concise, accurate, relevant, comprehensive information $\Rightarrow$ diversity
- 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? Answer: submodular maximization.
- How do we choose the smallest set S that maintains a given degree of diversity? Constrained minimization (i.e., $\min |A|$ s.t. $f(A) \geq \alpha$).
- Random sample has probability of poorly representing normally underrepresented groups.

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**

Large image collections need to be summarized

Many images, also that have a higher level gestalt than just a few, want a summary (subset of images) to represent the diversity in the large image set.

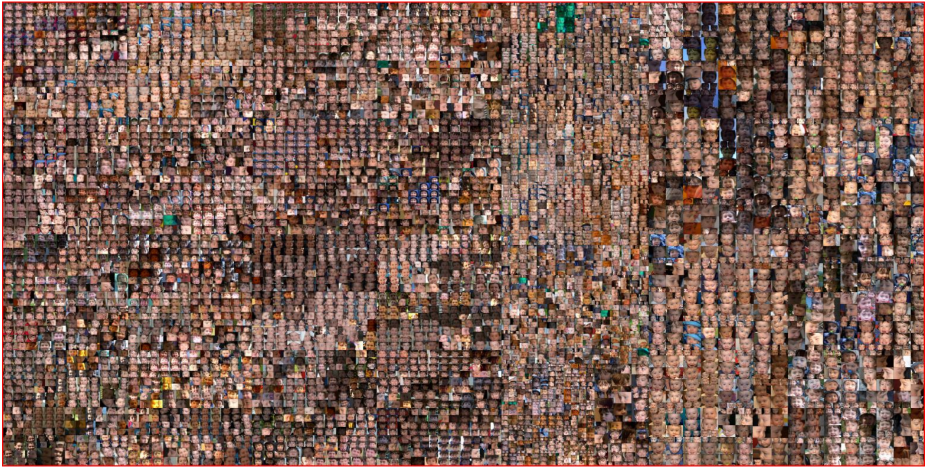


Image Summarization

10×10 image collection:

3 good summaries (diverse):

3 ok summaries:

3 poor summaries (redundant):

More Generally: Information and Summarization

- Let V be a set of information containing elements (V might say be any of words, sentences, documents, web pages, or blogs, sensor readings, etc.).
- Each $v \in V$ is one (or a set of) element(s). The total amount of information in V is measure by a function $f(V)$, and any given subset $S \subseteq V$ measures the amount of information in S, given by $f(S)$.
- How informative is any given item v in different sized contexts? Any such real-world information function would exhibit diminishing returns, i.e., the value of v decreases when it is considered in a larger context.
- A submodular function is likely a good model.

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 $(X_1, X_2, \dots, X_n) = X_V$ 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).

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:

$$I(Y; X_A) = \sum_{y, x_A} \Pr(y, x_A) \log \frac{\Pr(y, x_A)}{\Pr(y)\Pr(x_A)} = H(Y) - H(Y|X_A) \tag{2.1}$$

$$= H(X_A) - H(X_A|Y) = H(X_A) + H(Y) - H(X_A, Y) \tag{2.2}$$

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 $|A| \le k$. 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:

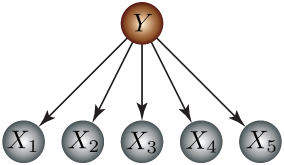
$$I(Y; X_A) = \sum_{y,x_A} \Pr(y, x_A) \log \frac{\Pr(y, x_A)}{\Pr(y)\Pr(x_A)} = H(Y) - H(Y|X_A) \qquad (2.1)$$

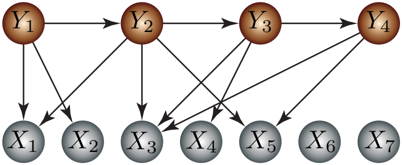
$$= H(X_A) - H(X_A|Y) = H(X_A) + H(Y) - H(X_A, Y) \qquad (2.2)$$

- 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

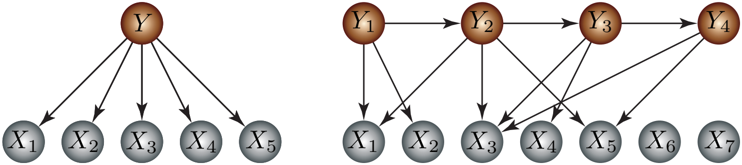
- Naïve Bayes property: $X_A \perp\!\!\!\perp X_B | Y$ for all A, B.





Information Gain and Feature Selection
in Pattern Classification: Naïve Bayes

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



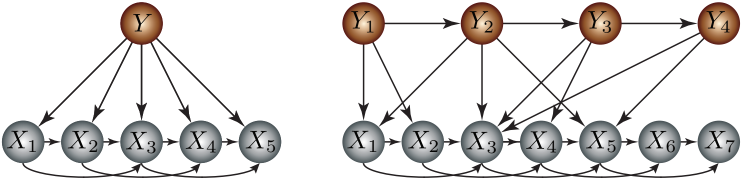
- When $X_A \perp\!\!\!\perp X_B | Y$ for all A, B (the Naïve Bayes assumption holds), then

$$f(A) = I(Y; X_A) = H(X_A) - H(X_A|Y) = H(X_A) - \sum_{a \in A} H(X_a|Y) \tag{2.3}$$

is submodular (submodular minus modular).

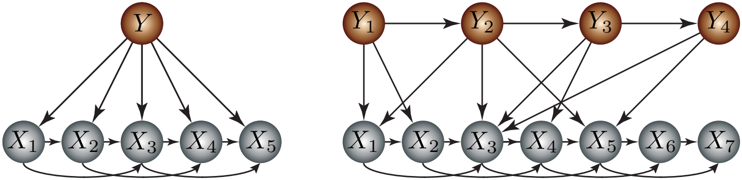
Variable Selection in Pattern Classification

- Naïve Bayes property fails:



Variable Selection in Pattern Classification

- Naïve Bayes property fails:



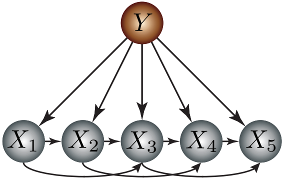
- $f(A)$ naturally expressed as a difference of two submodular functions

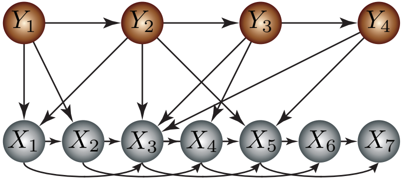
$$f(A) = I(Y; X_A) = H(X_A) - H(X_A|Y), \tag{2.4}$$

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

$$f(A) = I(Y; X_A) = H(X_A) - H(X_A|Y), \tag{2.4}$$

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:

$$f(A) = \sum_{a \in A} I(X_a; Y) - \lambda \sum_{a,a' \in A} I(X_a; X_{a'}|Y) \tag{2.5}$$

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

Variable Selection: Linear Regression Case

- Next, let Z be continuous. Predictor is linear $\tilde{Z}_A = \sum_{i \in A} \alpha_i X_i$.

Variable Selection: Linear Regression Case

- Next, let Z be continuous. Predictor is linear $\tilde{Z}_A = \sum_{i \in A} \alpha_i X_i$.
- Error measure is the residual variance

$$R^2_{Z,A} = \frac{\mathsf{Var}(Z) - E[(Z - \tilde{Z}_A)^2]}{\mathsf{Var}(Z)} \tag{2.6}$$

Variable Selection: Linear Regression Case

- Next, let Z be continuous. Predictor is linear $\tilde{Z}_A = \sum_{i \in A} \alpha_i X_i$.
- Error measure is the residual variance

$$R^2_{Z,A} = \frac{\text{Var}(Z) - E[(Z - \tilde{Z}_A)^2]}{\text{Var}(Z)} \tag{2.6}$$

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

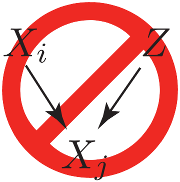
- Next, let Z be continuous. Predictor is linear $\tilde{Z}_A = \sum_{i \in A} \alpha_i X_i$.
- Error measure is the residual variance

$$R^2_{Z,A} = \frac{\text{Var}(Z) - E[(Z - \tilde{Z}_A)^2]}{\text{Var}(Z)} \tag{2.6}$$

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

$$f(A) = R^2_{Z,A} = b_A{}^\intercal (C_A^{-1})^\intercal b_A \tag{2.7}$$

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



Data Subset Selection

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

Data Subset Selection

- Suppose we are given a large data set $\mathcal{D} = \{x_i\}_{i=1}^n$ of n data items $V = \{v_1, v_2, \ldots, v_n\}$ and we wish to choose a subset $A \subset V$ of items that is good in some way (e.g., a summary).
- 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 "concepts", etc.) of each data item.

Data Subset Selection

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

$$g(m_u(A+v)) - g(m_u(A)) \geq g(m_u(B+v)) - g(m_u(B)) \tag{2.8}$$

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

$$g(m_u(A+v)) - g(m_u(A)) \geq g(m_u(B+v)) - g(m_u(B)) \tag{2.8}$$

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

$$f(X) = \sum_{u \in U} \alpha_u g_u(m_u(X)) \tag{2.9}$$

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

$$g(m_u(A+v)) - g(m_u(A)) \geq g(m_u(B+v)) - g(m_u(B)) \tag{2.8}$$

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

$$f(X) = \sum_{u \in U} \alpha_u g_u(m_u(X)) \tag{2.9}$$

  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 $u \in U$).
- Next, normalize the modular weights for each feature:

$$0 \leq \bar{m}_u(X) \triangleq \frac{m_u(X)}{\sum_{u' \in U} m_{u'}(X)} = \frac{m_u(X)}{m(X)} \leq 1 \tag{2.10}$$

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 U since $\bar{m}_u(X) \geq 0$ and $\sum_{u \in U} \bar{m}_u(X) = 1$ for any $X \subseteq V$.
- Consider the KL-divergence between these two distributions:

$$\begin{aligned} D(p || \{\bar{m}_u(X)\}_{u \in U}) &= \sum_{u \in U} p_u \log p_u - \sum_{u \in U} p_u \log(\bar{m}_u(X)) && (2.11) \\ &= \sum_{u \in U} p_u \log p_u - \sum_{u \in U} p_u \log(m_u(X)) + \log(m(X)) \\ &= -H(p) + \log m(X) - \sum_{u \in U} p_u \log(m_u(X)) && (2.12) \end{aligned}$$

Data Subset Selection, KL-divergence

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

$$D(p||\{\bar{m}_u(X)\}) = \text{const.} + \log m(X) - \sum_{u \in U} p_u \log(m_u(X)) \quad (2.13)$$

- 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)

$$g(X) \triangleq \log m(X) - D(p||\{\bar{m}_u(X)\}) = \sum_{u \in U} p_u \log(m_u(X)) \quad (2.14)$$

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., $D(p||\{\bar{m}_u(X)\})$).

Information Gain for Sensor Placement

- Given an environment, V is set of candidate locations for placement of a sensor (e.g., temperature, gas, audio, video, bacteria or other environmental contaminant, etc.).

Information Gain for Sensor Placement

- Given an environment, V is set 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. If a point is covered, we can answer a question about it (i.e., temperature, degree of contaminant).

Information Gain for Sensor Placement

- Given an environment, V is set 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. If a point is covered, we can answer a question about it (i.e., temperature, degree of contaminant).
- $f(V)$ is maximum coverage.

Information Gain for Sensor Placement

- Given an environment, V is set 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. If a point is covered, we can answer a question about it (i.e., temperature, degree of contaminant).
- $f(V)$ is maximum 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)

Information Gain for Sensor Placement

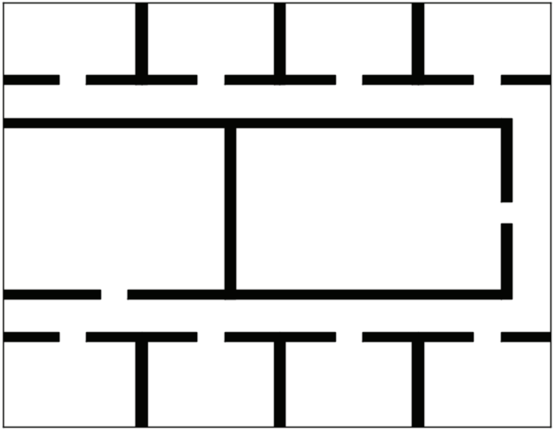
- Given an environment, V is set of candidate locations for placement of a sensor (e.g., temperature, gas, audio, video, bacteria or other environmental contaminant, etc.).
- We have a function $f(A)$ that measures the "coverage" of any given set A of sensor placement decisions. If a point is covered, we can answer a question about it (i.e., temperature, degree of contaminant).
- $f(V)$ is maximum 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.

Information Gain for Sensor Placement

- Given an environment, V is set 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. If a point is covered, we can answer a question about it (i.e., temperature, degree of contaminant).
- $f(V)$ is maximum 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.

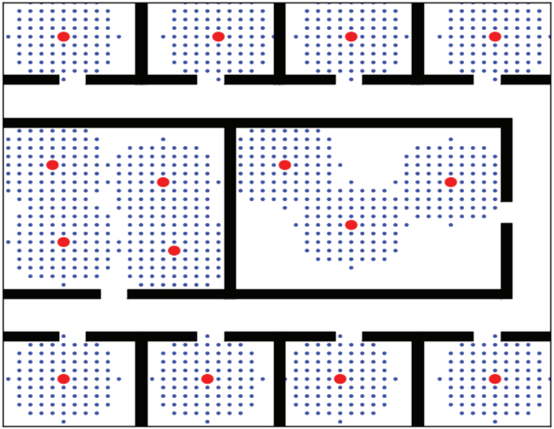
Sensor Placement within Buildings

- 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.



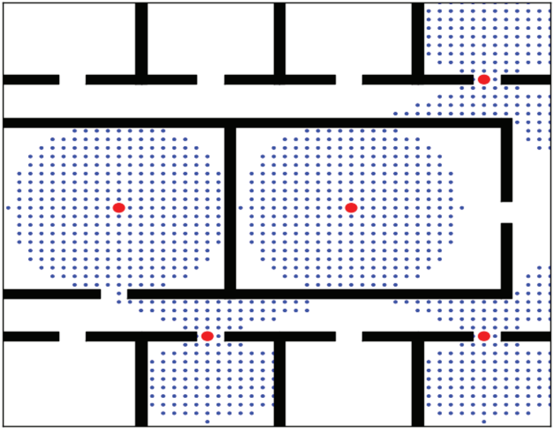
Sensor Placement within Buildings

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



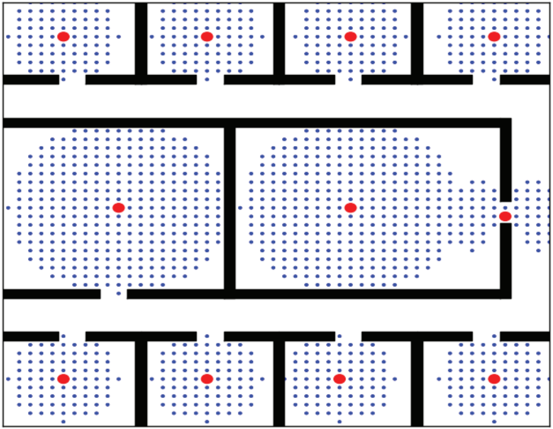
Sensor Placement within Buildings

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



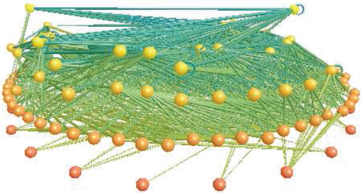
Sensor Placement within Buildings

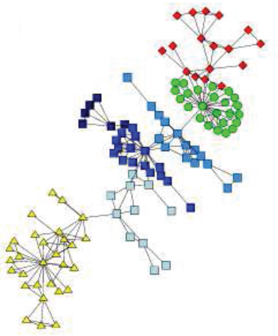
- Example sensor placement using mixed range sensors (located at red dots).

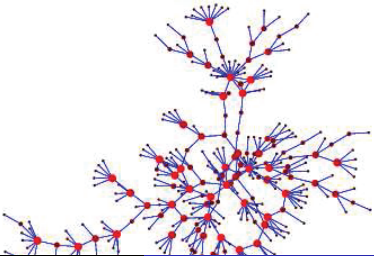


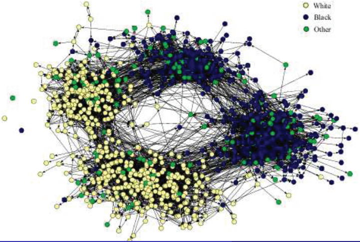
Social Networks

(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 in a network. How valuable is a given friend $v \in V$?

The value of a friend

- Let V be a set of individuals in a network. 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 in a network. 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 in a network. 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

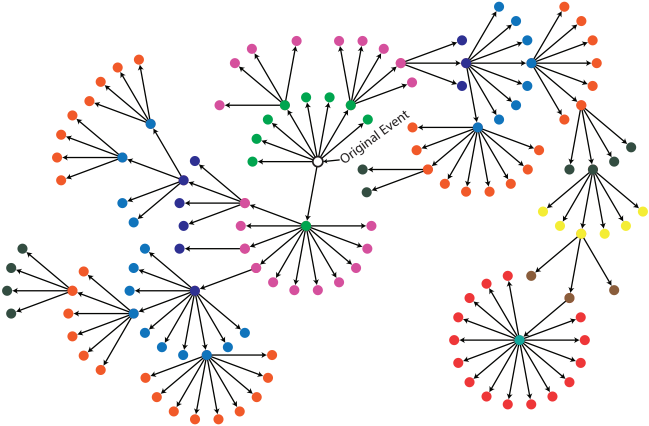
- Let V be a set of individuals in a network. How valuable is a given friend $v \in V$? It depends on how many friends you have.
- Valuate a group of friends $S \subseteq V$ via set function $f(S)$.
- A submodular model: a friend becomes less marginally valuable as your set of friends grows.
- Supermodular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

The value of a friend

- Let V be a set of individuals in a network. 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.
- Which is a better model?

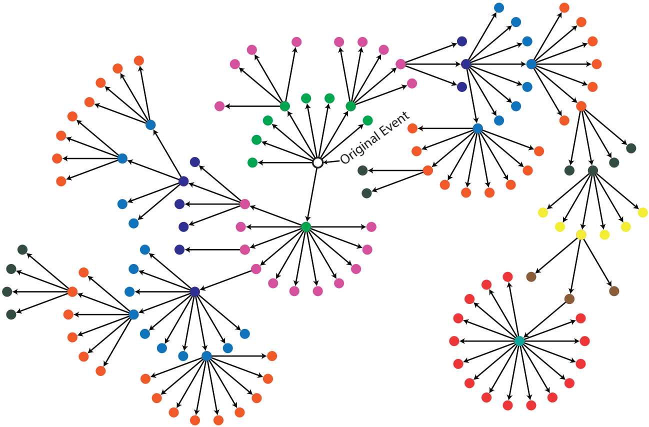
Information Cascades, Diffusion Networks

- 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

Where are they useful?

- **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 gets sick from whom? What is the infection network of such links? Given finite supply of vaccine, who to inoculate to protect overall population (cut the network)?
  - Infer the connectivity of a network (memes, purchase decisions, viruses, etc.) based only on diffusion traces (the time that each node is "infected")?
  - How to find the most likely tree or graph?

A model of influence in social networks

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

A model of influence in social networks

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

A model of influence in social networks

- Given a graph $G = (V, E)$, each $v \in V$ corresponds to a person, to each v we have an activation function $f_v : 2^V \to [0, 1]$ dependent only on its neighbors. I.e., $f_v(A) = f_v(A \cap \Gamma(v))$.
- Goal - Viral Marketing: find a small subset $S \subseteq V$ of individuals to directly influence, and thus indirectly influence the greatest number of possible other individuals (via the social network G).
- Define function $f : 2^V \to \mathbb{Z}^+$ to model the ultimate influence of an initial infected nodes S. Use following iterative process; at each step:
  - Given previous set of infected nodes S that have not yet had their chance to infect their neighbors,
  - activate new nodes $v \in V \setminus S$ if $f_v(S \cap \Gamma_v) \geq U[0, 1]$, where $U[0, 1]$ is a uniform random number between 0 and 1, and Γ_v are the neighbors of v.

A model of influence in social networks

- Given a graph $G = (V, E)$, each $v \in V$ corresponds to a person, to each v we have an activation function $f_v : 2^V \to [0, 1]$ dependent only on its neighbors. I.e., $f_v(A) = f_v(A \cap \Gamma(v))$.
- Goal - Viral Marketing: find a small subset $S \subseteq V$ of individuals to directly influence, and thus indirectly influence the greatest number of possible other individuals (via the social network G).
- Define function $f : 2^V \to \mathbb{Z}^+$ to model the ultimate influence of an initial infected nodes S. Use following iterative process; at each step:
  - Given previous set of infected nodes S that have not yet had their chance to infect their neighbors,
  - activate new nodes $v \in V \setminus S$ if $f_v(S \cap \Gamma_v) \geq U[0, 1]$, where $U[0, 1]$ is a uniform random number between 0 and 1, and Γ_v are the neighbors of v.
- For many f_v (including simple linear functions, and where f_v is submodular itself), we can show f is submodular (Kempe, Kleinberg, Tardos 1993).

Optimization Problem Involving Network Externalities

- (From Mirrokni, Roch, Sundararajan 2012): Let V be a set of users.

Optimization Problem Involving Network Externalities

- (From Mirrokni, Roch, Sundararajan 2012): Let V be a set of users.
- Let $v_i(S)$ be the value that user i has for a good if $S \subseteq V$ already own the good — e.g. $v_i(S) = \omega_i + f_i(\sum_{j \in S} w_{ij})$ where ω_i is inherent value, and f_i might be a concave function, and w_{ij} is how important $j \in S$ is to i (e.g., a network). Weights might be random.

Optimization Problem Involving Network Externalities

- (From Mirrokni, Roch, Sundararajan 2012): Let V be a set of users.
- Let $v_i(S)$ be the value that user i has for a good if $S \subseteq V$ already own the good — e.g. $v_i(S) = \omega_i + f_i(\sum_{j \in S} w_{ij})$ where ω_i is inherent value, and f_i might be a concave function, and w_{ij} is how important $j \in S$ is to i (e.g., a network). Weights might be random.
- Given price p for good, user i buys good if $v_i(S) \geq p$.

Optimization Problem Involving Network Externalities

- (From Mirrokni, Roch, Sundararajan 2012): Let V be a set of users.
- Let $v_i(S)$ be the value that user i has for a good if $S \subseteq V$ already own the good — e.g. $v_i(S) = \omega_i + f_i(\sum_{j \in S} w_{ij})$ where ω_i is inherent value, and f_i might be a concave function, and w_{ij} is how important $j \in S$ is to i (e.g., a network). Weights might be random.
- Given price p for good, user i buys good if $v_i(S) \geq p$.
- We choose initial price p and initial set of users $A \subseteq V$ who get the good for free.

Optimization Problem Involving Network Externalities

- (From Mirrokni, Roch, Sundararajan 2012): Let V be a set of users.
- Let $v_i(S)$ be the value that user i has for a good if $S \subseteq V$ already own the good — e.g. $v_i(S) = \omega_i + f_i(\sum_{j \in S} w_{ij})$ where ω_i is inherent value, and f_i might be a concave function, and w_{ij} is how important $j \in S$ is to i (e.g., a network). Weights might be random.
- Given price p for good, user i buys good if $v_i(S) \geq p$.
- We choose initial price p and initial set of users $A \subseteq V$ who get the good for free.
- Define $S_1 = \{i \notin A : v_i(A) \geq p\}$ initial set of buyers.

Optimization Problem Involving Network Externalities

- (From Mirrokni, Roch, Sundararajan 2012): Let V be a set of users.
- Let $v_i(S)$ be the value that user i has for a good if $S \subseteq V$ already own the good — e.g. $v_i(S) = \omega_i + f_i(\sum_{j \in S} w_{ij})$ where ω_i is inherent value, and f_i might be a concave function, and w_{ij} is how important $j \in S$ is to i (e.g., a network). Weights might be random.
- Given price p for good, user i buys good if $v_i(S) \geq p$.
- We choose initial price p and initial set of users $A \subseteq V$ who get the good for free.
- Define $S_1 = \{i \notin A : v_i(A) \geq p\}$ initial set of buyers.
- $S_2 = \{i \notin A \cup S_1 : v_i(A \cup S_1) \geq p\}$.

Optimization Problem Involving Network Externalities

- (From Mirrokni, Roch, Sundararajan 2012): Let V be a set of users.
- Let $v_i(S)$ be the value that user i has for a good if $S \subseteq V$ already own the good — e.g. $v_i(S) = \omega_i + f_i(\sum_{j \in S} w_{ij})$ where ω_i is inherent value, and f_i might be a concave function, and w_{ij} is how important $j \in S$ is to i (e.g., a network). Weights might be random.
- Given price p for good, user i buys good if $v_i(S) \geq p$.
- We choose initial price p and initial set of users $A \subseteq V$ who get the good for free.
- Define $S_1 = \{i \notin A : v_i(A) \geq p\}$ initial set of buyers.
- $S_2 = \{i \notin A \cup S_1 : v_i(A \cup S_1) \geq p\}$.
- This starts a cascade. Let $S_k = \{i \notin \cup_{j<k} S_j \cup A : v_j(\cup_{j<k} S_j \cup A) \geq p\}$,

Optimization Problem Involving Network Externalities

- (From Mirrokni, Roch, Sundararajan 2012): Let V be a set of users.
- Let $v_i(S)$ be the value that user i has for a good if $S \subseteq V$ already own the good — e.g. $v_i(S) = \omega_i + f_i(\sum_{j \in S} w_{ij})$ where ω_i is inherent value, and f_i might be a concave function, and w_{ij} is how important $j \in S$ is to i (e.g., a network). Weights might be random.
- Given price p for good, user i buys good if $v_i(S) \geq p$.
- We choose initial price p and initial set of users $A \subseteq V$ who get the good for free.
- Define $S_1 = \{i \notin A : v_i(A) \geq p\}$ initial set of buyers.
- $S_2 = \{i \notin A \cup S_1 : v_i(A \cup S_1) \geq p\}$.
- This starts a cascade. Let $S_k = \{i \notin \cup_{j<k} S_j \cup A : v_j(\cup_{j<k} S_j \cup A) \geq p\}$,
- and let S_{k^*} be the saturation point, lowest value of k such that $S_k = S_{k+1}$

Optimization Problem Involving Network Externalities

- (From Mirrokni, Roch, Sundararajan 2012): Let V be a set of users.
- Let $v_i(S)$ be the value that user i has for a good if $S \subseteq V$ already own the good — e.g. $v_i(S) = \omega_i + f_i(\sum_{j \in S} w_{ij})$ where ω_i is inherent value, and f_i might be a concave function, and w_{ij} is how important $j \in S$ is to i (e.g., a network). Weights might be random.
- Given price p for good, user i buys good if $v_i(S) \geq p$.
- We choose initial price p and initial set of users $A \subseteq V$ who get the good for free.
- Define $S_1 = \{i \notin A : v_i(A) \geq p\}$ initial set of buyers.
- $S_2 = \{i \notin A \cup S_1 : v_i(A \cup S_1) \geq p\}$.
- This starts a cascade. Let $S_k = \{i \notin \cup_{j<k} S_j \cup A : v_j(\cup_{j<k} S_j \cup A) \geq p\}$,
- and let S_{k^*} be the saturation point, lowest value of k such that $S_k = S_{k+1}$
- Goal: find A and p to maximize $f_p(A) = \mathbb{E}[p \times |S_{k^*}|]$.

Graphical Model Structure Learning

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

$$p(x) = \frac{1}{Z}\exp(-E(x)) \tag{2.15}$$

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

Graphical Model Structure Learning

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

$$p(x) = \frac{1}{Z} \exp(-E(x)) \tag{2.15}$$

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

$$p(x) = \frac{1}{Z} \exp(-E(x)) \tag{2.15}$$

  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:

$$E(x) = \sum_{c \in \mathcal{C}} E_c(x_c) \tag{2.16}$$

Graphical Model Structure Learning

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

$$p(x) = \frac{1}{Z}\exp(-E(x)) \tag{2.15}$$

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:

$$E(x) = \sum_{c \in \mathcal{C}} E_c(x_c) \tag{2.16}$$

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

$$p(x) = \frac{1}{Z}\exp(-E(x)) \tag{2.15}$$

  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:

$$E(x) = \sum_{c \in \mathcal{C}} E_c(x_c) \tag{2.16}$$

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

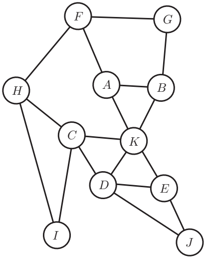
Graphical Models: Learning Tree Distributions

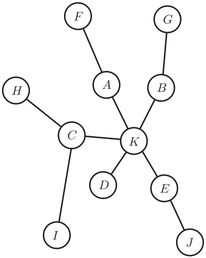
- Goal: find the closest distribution p_t to p subject to p_t factoring w.r.t. some tree $T = (V, F)$, i.e., $p_t \in \mathcal{F}(T, \mathcal{M})$.

Graphical Models: Learning Tree Distributions

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

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

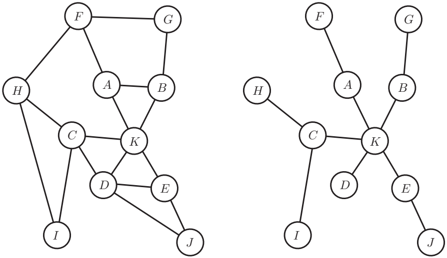




Graphical Models: Learning Tree Distributions

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

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

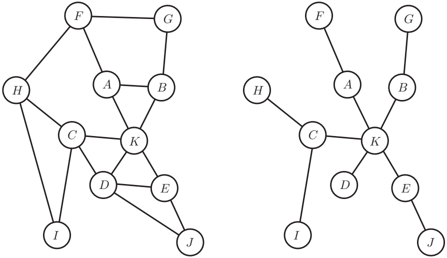


- Discrete problem: choose the optimal set of edges $A \subseteq E$ that constitute tree (i.e., find a spanning tree of G of best quality).

Graphical Models: Learning Tree Distributions

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

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

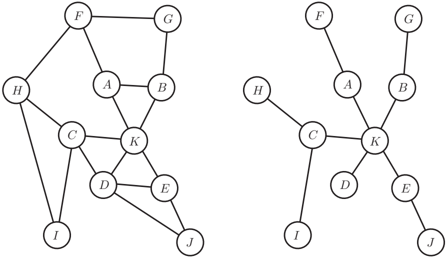


- Discrete problem: choose the optimal set of edges $A \subseteq E$ that constitute tree (i.e., find a spanning tree of G of best quality).
- Define $f : 2^E \to \mathbb{R}_+$ where f is a weighted cycle matroid rank function (a type of submodular function), with weights $w(e) = w(u, v) = I(X_u; X_v)$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

Graphical Models: Learning Tree Distributions

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

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



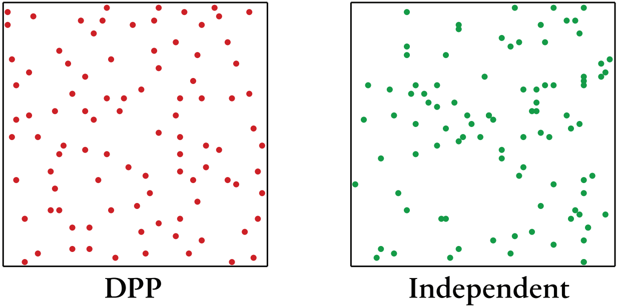
- Discrete problem: choose the optimal set of edges $A \subseteq E$ that constitute tree (i.e., find a spanning tree of G of best quality).
- Define $f : 2^E \to \mathbb{R}_+$ where f is a weighted cycle matroid rank function (a type of submodular function), with weights $w(e) = w(u, v) = I(X_u; X_v)$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

Determinantal Point Processes (DPPs)

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

Determinantal Point Processes (DPPs)

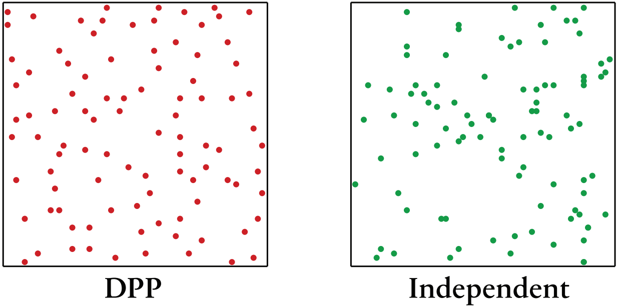
- Sometimes we wish not only to valuate subsets $A \subseteq V$ but to induce probability distributions over all subsets.
- We may wish to prefer samples where elements of A are diverse (i.e., given a sample A, for $a, b \in A$, we prefer a and b to be different).



(Kulesza, Gillen-water, & 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).

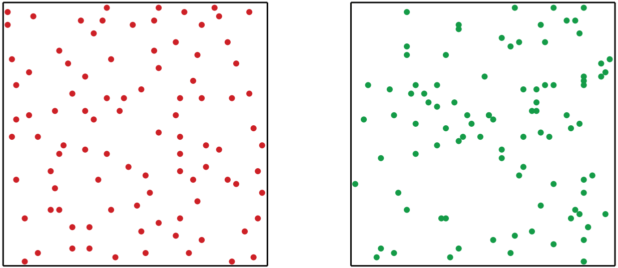


(Kulesza, Gillenwater, & Taskar, 2011)

- 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).



DPP **Independent**

(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, a subset $X \subseteq V$, let M_X be $|X| \times |X|$ principle submatrix, rows/columns specified by $X \subseteq V$.

DPPs and log-submodular probability distributions

- Given binary vectors $x, y \in \{0,1\}^V$, $y \leq x$ if $y(v) \leq x(v), \forall v \in V$.
- Given a positive-definite $n \times n$ matrix M, a subset $X \subseteq V$, let M_X be $|X| \times |X|$ principle submatrix, rows/columns specified by $X \subseteq V$.
- A Determinantal Point Process (DPP) is a distribution of the form:

$$\Pr(\mathbf{X} = x) = \frac{|M_{X(x)}|}{|M+I|} = \exp\left(\log\left(\frac{|M_{X(x)}|}{|M+I|}\right)\right) \propto \det(M_{X(x)}) \tag{2.17}$$

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, a subset $X \subseteq V$, let M_X be $|X| \times |X|$ principle submatrix, rows/columns specified by $X \subseteq V$.
- A Determinantal Point Process (DPP) is a distribution of the form:

$$\Pr(\mathbf{X} = x) = \frac{|M_{X(x)}|}{|M + I|} = \exp\left(\log\left(\frac{|M_{X(x)}|}{|M + I|}\right)\right) \propto \det(M_{X(x)}) \tag{2.17}$$

  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:

$$\sum_{x \in \{0,1\}^V : x \geq y} \Pr(\mathbf{X} = x) = \Pr(\mathbf{X} \geq y) = \exp\left(\log\left(|K_{Y(y)}|\right)\right) \tag{2.18}$$

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, a subset $X \subseteq V$, let M_X be $|X| \times |X|$ principle submatrix, rows/columns specified by $X \subseteq V$.
- A Determinantal Point Process (DPP) is a distribution of the form:

$$\Pr(\mathbf{X} = x) = \frac{|M_{X(x)}|}{|M + I|} = \exp\left(\log\left(\frac{|M_{X(x)}|}{|M + I|}\right)\right) \propto \det(M_{X(x)}) \tag{2.17}$$

  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:

$$\sum_{x \in \{0,1\}^V : x \geq y} \Pr(\mathbf{X} = x) = \Pr(\mathbf{X} \geq y) = \exp\left(\log\left(|K_{Y(y)}|\right)\right) \tag{2.18}$$

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

DPPs and log-submodular probability distributions

- Given binary vectors $x, y \in \{0,1\}^V$, $y \leq x$ if $y(v) \leq x(v), \forall v \in V$.
- Given a positive-definite $n \times n$ matrix M, a subset $X \subseteq V$, let M_X be $|X| \times |X|$ principle submatrix, rows/columns specified by $X \subseteq V$.
- A Determinantal Point Process (DPP) is a distribution of the form:

$$\Pr(\mathbf{X} = x) = \frac{|M_{X(x)}|}{|M + I|} = \exp\left(\log\left(\frac{|M_{X(x)}|}{|M + I|}\right)\right) \propto \det(M_{X(x)}) \tag{2.17}$$

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:

$$\sum_{x \in \{0,1\}^V : x \geq y} \Pr(\mathbf{X} = x) = \Pr(\mathbf{X} \geq y) = \exp\left(\log\left(|K_{Y(y)}|\right)\right) \tag{2.18}$$

- 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:

$$p(x) = \frac{1}{Z} \exp(-E(x)) \tag{2.19}$$

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:
$$p(x) = \frac{1}{Z} \exp(-E(x)) \tag{2.19}$$
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
$$x^* \in \operatorname*{argmax}_{x \in \{0,1\}^V} p(x) \tag{2.20}$$

Graphical Models and fast MAP Inference

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

$$p(x) = \frac{1}{Z} \exp(-E(x)) \tag{2.19}$$

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

$$x^* \in \operatorname*{argmax}_{x \in \{0,1\}^V} p(x) \tag{2.20}$$

- 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:

$$p(x) = \frac{1}{Z} \exp(-E(x)) \tag{2.19}$$

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

$$x^* \in \operatorname*{argmax}_{x \in \{0,1\}^V} p(x) \tag{2.20}$$

- 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:

$$p(x) = \frac{1}{Z}\exp(-E(x)) \tag{2.19}$$

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

$$x^* \in \operatorname*{argmax}_{x \in \{0,1\}^V} p(x) \tag{2.20}$$

- 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:

$$p(x) = \frac{1}{Z} \exp(-E(x)) \tag{2.19}$$

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

$$x^* \in \operatorname*{argmax}_{x \in \{0,1\}^V} p(x) \tag{2.20}$$

- 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:

$$p(x) = \frac{1}{Z} \exp(-E(x)) \tag{2.19}$$

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

$$x^* \in \operatorname*{argmax}_{x \in \{0,1\}^V} p(x) \tag{2.20}$$

- 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:

$$E(x) = \sum_{v \in V(G)} e_v(x_v) + \sum_{(i,j) \in E(G)} e_{ij}(x_i, x_j) \tag{2.21}$$

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:

$$E(x) = \sum_{v \in V(G)} e_v(x_v) + \sum_{(i,j) \in E(G)} e_{ij}(x_i, x_j) \tag{2.21}$$

- $e_v(x_v)$ 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:

$$E(x) = \sum_{v \in V(G)} e_v(x_v) + \sum_{(i,j) \in E(G)} e_{ij}(x_i, x_j) \tag{2.21}$$

- $e_v(x_v)$ and $e_{ij}(x_i, x_j)$ are like local energy potentials.
- Since $\log p(x) = -E(x) +$ 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:

$$E(x) = \sum_{v \in V(G)} e_v(x_v) + \sum_{(i,j) \in E(G)} e_{ij}(x_i, x_j) \tag{2.21}$$

- $e_v(x_v)$ 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:

$$E(x) = \sum_{v \in V(G)} e_v(x_v) + \sum_{(i,j) \in E(G)} e_{ij}(x_i, x_j) \tag{2.21}$$

- $e_v(x_v)$ 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.,

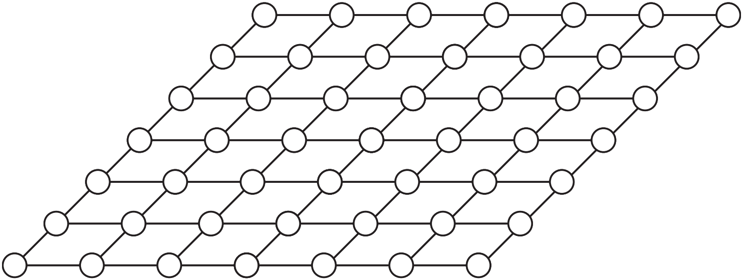
$$\min_{x \in \{0,1\}^V} E(x) \tag{2.22}$$

MRF example

Markov random field

$$\log p(x) \propto \sum_{v \in V(G)} e_v(x_v) + \sum_{(i,j) \in E(G)} e_{ij}(x_i, x_j) \tag{2.23}$$

When G is a 2D grid graph, we have

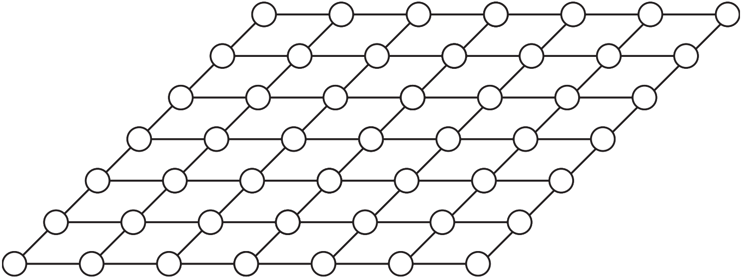


Create an auxiliary graph

- 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 $G_a = (V \cup \{s, t\}, E + \cup_{v \in V}((s, v) \cup (v, t)))$.

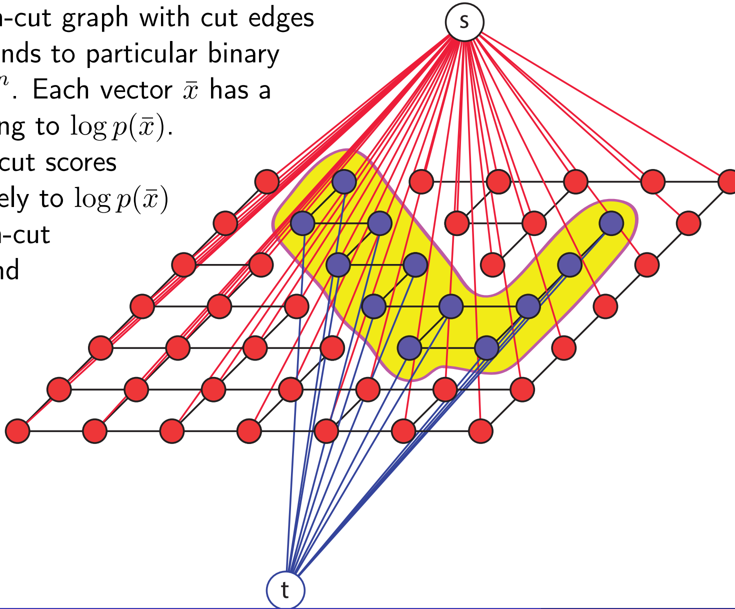
Transformation from graphical model to auxiliary graph

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 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 $\bar{x} \in \{0,1\}^n$.
- 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..
- 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.

Submodular potentials

submodularity is what allows graph cut to find exact solution

- 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:

$$e_{ij}(0,1) + e_{ij}(1,0) \geq e_{ij}(1,1) + e_{ij}(0,0) \tag{2.31}$$

Submodular potentials

submodularity is what allows graph cut to find exact solution

- 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:

$$e_{ij}(0,1) + e_{ij}(1,0) \geq e_{ij}(1,1) + e_{ij}(0,0) \tag{2.31}$$

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

Submodular potentials

submodularity is what allows graph cut to find exact solution

- 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:

$$e_{ij}(0,1) + e_{ij}(1,0) \geq e_{ij}(1,1) + e_{ij}(0,0) \tag{2.31}$$

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

$$f(X) = \sum_{\{i,j\} \in \mathcal{E}(G)} f_{i,j}(X \cap \{i,j\}) \tag{2.32}$$

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

Submodular potentials

submodularity is what allows graph cut to find exact solution

- 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:

$$e_{ij}(0,1) + e_{ij}(1,0) \geq e_{ij}(1,1) + e_{ij}(0,0) \tag{2.31}$$

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

$$f(X) = \sum_{\{i,j\} \in \mathcal{E}(G)} f_{i,j}(X \cap \{i,j\}) \tag{2.32}$$

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.

$$\log \Pr(x) = g(x) + \text{const.} = -E(x) + \text{const.} \tag{2.33}$$

where g is supermodular ($E(x) = -g(x)$ 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.

$$\log \Pr(x) = g(x) + \text{const.} = -E(x) + \text{const.} \tag{2.33}$$

where g is supermodular ($E(x) = -g(x)$ 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:

$$\log \Pr(x) = f(x) + \text{const.} \tag{2.34}$$

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

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




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.

$$f_w(X) = w\Big(\{(u, v) \in E : u \in X, v \in V \setminus X\}\Big) \tag{2.35}$$

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.

$$f_w(X) = w\Big(\{(u,v) \in E : u \in X, v \in V \setminus X\}\Big) \tag{2.35}$$

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

$$f_g(X) = g\Big(\{(u,v) \in E : u \in X, v \in V \setminus X\}\Big) \tag{2.36}$$

Addressing shrinking bias with edge submodularity

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

$$f_w(X) = w\Big(\{(u,v) \in E : u \in X, v \in V \setminus X\}\Big) \tag{2.35}$$

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

$$f_g(X) = g\Big(\{(u,v) \in E : u \in X, v \in V \setminus X\}\Big) \tag{2.36}$$

- Seen as a node function, $f_g : 2^V \rightarrow \mathbb{R}_+$ 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.

$$f_w(X) = w\Big(\{(u,v) \in E : u \in X, v \in V \setminus X\}\Big) \tag{2.35}$$

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

$$f_g(X) = g\Big(\{(u,v) \in E : u \in X, v \in V \setminus X\}\Big) \tag{2.36}$$

- 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).

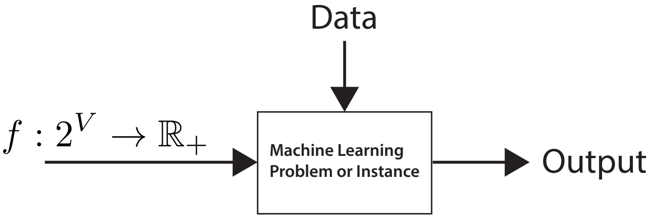
Graph-cut vs. cooperative-cut comparisons

Graph Cut | Cooperative Cut

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

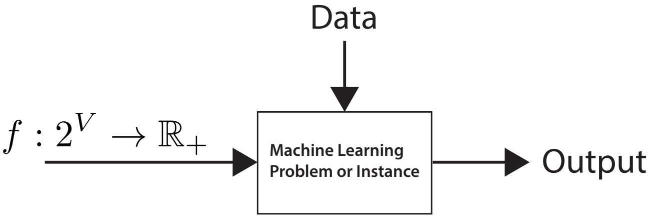
A submodular function as a parameter

- 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

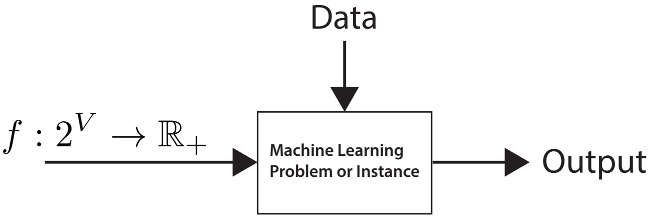
- In some cases, it may be useful to view a submodular function $f : 2^V \to \mathbb{R}$ as a input "parameter" to a machine learning algorithm.



- 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

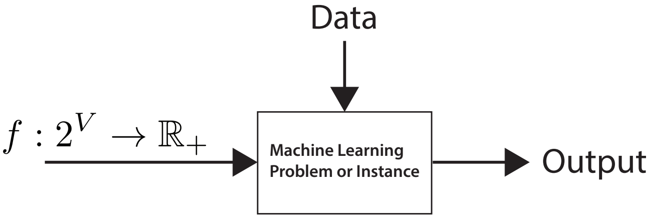
- In some cases, it may be useful to view a submodular function $f : 2^V \to \mathbb{R}$ as a input "parameter" to a machine learning algorithm.



- 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 $f + f_\epsilon \in \mathbb{S}$ (more on problem sets).

Supervised Machine Learning
From F. Bach

- We are given n samples of observed data $(x_i, y_i) \in \mathbb{R}^p \times \mathbb{R}$, $i \in [n]$.
  - Response vector $y = (y_1, \dots, y_n)^\intercal \in \mathbb{R}^n$
  - Design matrix $X = (x_1, \dots, x_n)^\intercal \in \mathbb{R}^{n \times p}$.

Supervised Machine Learning
From F. Bach

- We are given n samples of observed data $(x_i, y_i) \in \mathbb{R}^p \times \mathbb{R}$, $i \in [n]$.
  - Response vector $y = (y_1, \ldots, y_n)^\intercal \in \mathbb{R}^n$
  - Design matrix $X = (x_1, \ldots, x_n)^\intercal \in \mathbb{R}^{n \times p}$.
- Regularized empirical risk minimization:

$$\min_{w \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, w^\intercal x_i) + \lambda\Omega(w) = \min_{w \in \mathbb{R}^p} L(y, Xw) + \lambda\Omega(w) \qquad (2.37)$$

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

Supervised Machine Learning
From F. Bach

- We are given n samples of observed data $(x_i, y_i) \in \mathbb{R}^p \times \mathbb{R}$, $i \in [n]$.
  - Response vector $y = (y_1, \ldots, y_n)^\intercal \in \mathbb{R}^n$
  - Design matrix $X = (x_1, \ldots, x_n)^\intercal \in \mathbb{R}^{n \times p}$.
- Regularized empirical risk minimization:

$$\min_{w \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, w^\intercal x_i) + \lambda \Omega(w) = \min_{w \in \mathbb{R}^p} L(y, Xw) + \lambda \Omega(w) \tag{2.37}$$

  where $\ell(\cdot)$ is a loss function (e.g., squared error) and $\Omega(w)$ is a (perhaps sparse) norm.
- When data has multiple (k) responses, $y = (y^1, \ldots, y^k) \in R^{n \times k}$, we get:

$$\min_{w^1, \ldots, w^k \in \mathbb{R}^n} \sum_{j=1}^{k} \left\{ L(y^j, Xw^j) + \lambda \Omega(w^j) \right\} \tag{2.38}$$

Dictionary Learning and Selection

- When only the multiple responses $y = (y^1, \dots, y^k) \in \mathbb{R}^{n \times k}$ are observed, we get either dictionary learning

$$\min_{X=(x^1,\dots,x^p)\in\mathbb{R}^{n\times p}} \min_{w^1,\dots,w^k\in\mathbb{R}^p} \sum_{j=1}^{k} \left\{ L(y^j, Xw^j) + \lambda\Omega(w^j) \right\} \tag{2.39}$$

Dictionary Learning and Selection

- When only the multiple responses $y = (y^1, \dots, y^k) \in \mathbb{R}^{n \times k}$ are observed, we get either dictionary learning

$$\min_{X=(x^1,\dots,x^p)\in\mathbb{R}^{n\times p}} \min_{w^1,\dots,w^k\in\mathbb{R}^p} \sum_{j=1}^{k} \left\{ L(y^j, Xw^j) + \lambda\Omega(w^j) \right\} \tag{2.39}$$

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

$$f(D) = \min_{S\subseteq D, |S|\leq k} \min_{w_S^j\in\mathbb{R}^S} \sum_{j=1}^{k} \left\{ L(y^j, X_S w_S^j) + \lambda\Omega(w_S^j) \right\} \tag{2.40}$$

where D is the dictionary (allowed indices of X), and $X_S \in \mathbb{R}^{n\times|S|}$ is a column sub-matrix of X.

Dictionary Learning and Selection

- When only the multiple responses $y = (y^1, \dots, y^k) \in \mathbb{R}^{n \times k}$ are observed, we get either dictionary learning

$$\min_{X=(x^1,\dots,x^p)\in\mathbb{R}^{n\times p}} \min_{w^1,\dots,w^k\in\mathbb{R}^p} \sum_{j=1}^{k} \left\{ L(y^j, Xw^j) + \lambda\Omega(w^j) \right\} \qquad (2.39)$$

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

$$f(D) = \min_{S\subseteq D, |S|\leq k} \min_{w_S^j\in\mathbb{R}^S} \sum_{j=1}^{k} \left\{ L(y^j, X_S w_S^j) + \lambda\Omega(w_S^j) \right\} \qquad (2.40)$$

  where D is the dictionary (allowed indices of X), and $X_S \in \mathbb{R}^{n\times|S|}$ is a column sub-matrix of X.

- This is a subset selection problem, and the regularizer $\Omega(\cdot)$ is critical (could be structured sparse convex norm, via Lovász extension!).

Norms, sparse norms, and computer vision

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

$$\Omega(w) = \sum_{i=2}^{N} |w_i - w_{i-1}| \tag{2.41}$$

related to Lovász extension of a graph-cut submodular function.
- 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

$$\tilde{f}(w) = \sum_{i=1}^{n} w_{\sigma_i}(f(\sigma_1, \ldots, \sigma_i) - f(\sigma_1, \ldots, \sigma_{i-1})) \tag{2.42}$$

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

$$\tilde{f}(w) = \sum_{i=1}^{n} w_{\sigma_i}(f(\sigma_1, \ldots, \sigma_i) - f(\sigma_1, \ldots, \sigma_{i-1})) \tag{2.42}$$

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

Submodular Generalized Dependence

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

$$f(A \cup B) = f(A) + f(B), \tag{2.43}$$

Submodular Generalized Dependence

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

$$f(A \cup B) = f(A) + f(B), \tag{2.43}$$

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

$$f(A \cup B \cup C) + f(C) = f(A \cup C) + f(B \cup C) \tag{2.44}$$

Submodular Generalized Dependence

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

$$f(A \cup B) = f(A) + f(B), \tag{2.43}$$

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

$$f(A \cup B \cup C) + f(C) = f(A \cup C) + f(B \cup C) \tag{2.44}$$

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

$$f(A|B) \triangleq f(A \cup B) - f(B) < f(A), \tag{2.45}$$

Submodular Generalized Dependence

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

$$f(A \cup B) = f(A) + f(B), \tag{2.43}$$

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

$$f(A \cup B \cup C) + f(C) = f(A \cup C) + f(B \cup C) \tag{2.44}$$

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

$$f(A|B) \triangleq f(A \cup B) - f(B) < f(A), \tag{2.45}$$

- and a notion of "conditional mutual information"

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

Submodular Generalized Dependence

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

$$f(A \cup B) = f(A) + f(B), \tag{2.43}$$

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

$$f(A \cup B \cup C) + f(C) = f(A \cup C) + f(B \cup C) \tag{2.44}$$

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

$$f(A|B) \triangleq f(A \cup B) - f(B) < f(A), \tag{2.45}$$

- and a notion of "conditional mutual information"

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

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

$$I_f(S_1; S_2; \dots; S_k) = \sum_{i=1}^{k} f(S_k) - f(S_1 \cup S_2 \cup \dots \cup S_k) \tag{2.46}$$

$$I'_f(S_1; S_2; \dots; S_k) = \sum_{A \subseteq \{1,2,\dots,k\}} (-1)^{|A|+1} f(\bigcup_{j \in A} S_j) \tag{2.47}$$

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 $A_1^* \in \operatorname{argmin}_{A \subseteq V} I_f(A; V \setminus A)$ and $A_2^* = V \setminus A_1^*$.
- Then partition the partitions: $A_{11}^* \in \operatorname{argmin}_{A \subseteq A_1^*} I_f(A; A_1^* \setminus A)$, $A_{12}^* = A_1^* \setminus A_{11}^*$, and $A_{21}^* \in \operatorname{argmin}_{A \subseteq A_2^*} I_f(A; A_2^* \setminus A)$, etc.
- Recursively partition the partitions, we end up with a partition $V = V_1 \cup V_2 \cup \cdots \cup V_k$ that clusters the data.
- Each minimization can be done using Queyranne's algorithm (alternatively can construct a Gomory-Hu tree). This gives a partition no worse than factor 2 away from optimal partition. (Narasimhan&Bilmes, 2007).

Submodular Parameterized Clustering

- Given a submodular function $f : 2^V \to \mathbb{R}$, form the combinatorial dependence function $I_f(A; B) = f(A) + f(B) - f(A \cup B)$.
- Consider clustering algorithm: First find partition $A_1^* \in \operatorname{argmin}_{A \subseteq V} I_f(A; V \setminus A)$ and $A_2^* = V \setminus A_1^*$.
- Then partition the partitions: $A_{11}^* \in \operatorname{argmin}_{A \subseteq A_1^*} I_f(A; A_1^* \setminus A)$, $A_{12}^* = A_1^* \setminus A_{11}^*$, and $A_{21}^* \in \operatorname{argmin}_{A \subseteq A_2^*} I_f(A; A_2^* \setminus A)$, etc.
- Recursively partition the partitions, we end up with a partition $V = V_1 \cup V_2 \cup \cdots \cup V_k$ that clusters the data.
- Each minimization can be done using Queyranne's algorithm (alternatively can construct a Gomory-Hu tree). This gives a partition no worse than factor 2 away from optimal partition. (Narasimhan&Bilmes, 2007).
- Hence, family of clustering algorithms parameterized by f.

Is Submodular Maximization Just Clustering?

1. Clustering objectives often NP-hard and inapproximable, submodular maximization is approximable for any submodular function.
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

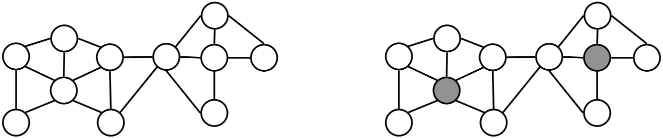
- Given training data $\mathcal{D}_V = \{(x_i, y_i)\}_{i \in V}$ of (x, y) pairs where x is a query (data item) and y is an answer (label), goal is to learn a good mapping $y = h(x)$.
- Often, getting y is time-consuming, expensive, and error prone (manual transcription, Amazon Turk, etc.)
- Batch active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

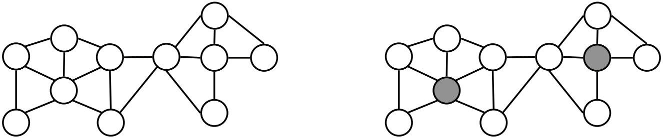
Active Transductive Semi-Supervised Learning

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

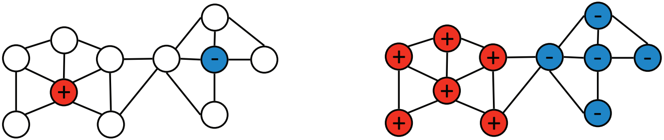


Active Transductive Semi-Supervised Learning

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

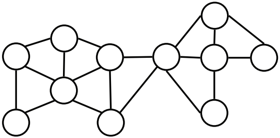
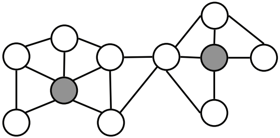


- Nature reveals labels $y_L \in \{0,1\}^L$, learner predicts labels $\hat{y} \in \{0,1\}^V$

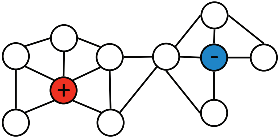
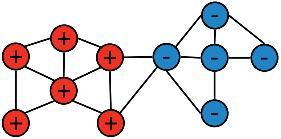


Active Transductive Semi-Supervised Learning

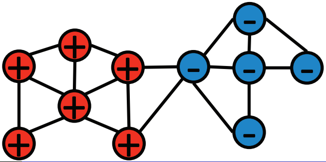
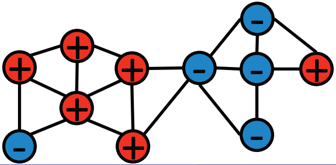
- Batch/Offline active learning: Given a set V of unlabeled data items, learner chooses subset $L \subseteq V$ of items to be labeled

- 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

$$\Psi(L) = \min_{T \subseteq V \setminus L : T \neq \emptyset} \frac{\Gamma(T)}{|T|} \tag{2.48}$$

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

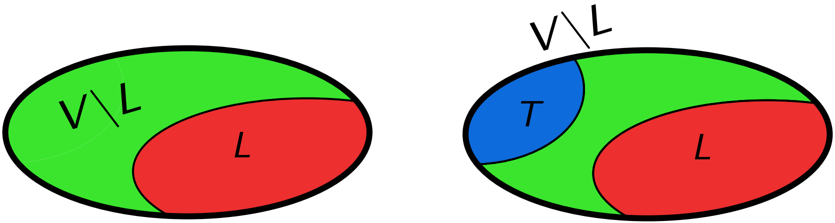
Choosing labels: how to select L

- Consider the following objective

$$\Psi(L) = \min_{T \subseteq V \setminus L : T \neq \emptyset} \frac{\Gamma(T)}{|T|} \tag{2.48}$$

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.



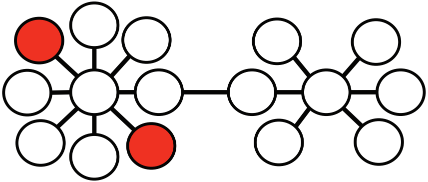
Choosing labels: how to select L

- Consider the following objective

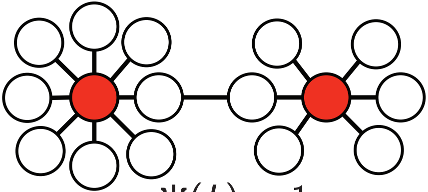
$$\Psi(L) = \min_{T \subseteq V \setminus L : T \neq \emptyset} \frac{\Gamma(T)}{|T|} \tag{2.48}$$

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

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



$\Psi(L) = 1/8$



$\Psi(L) = 1$

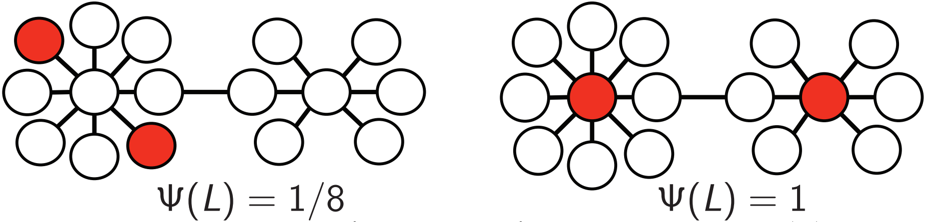
Choosing labels: how to select L

- Consider the following objective

$$\Psi(L) = \min_{T \subseteq V \setminus L : T \neq \emptyset} \frac{\Gamma(T)}{|T|} \tag{2.48}$$

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.



- 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 $A \subseteq V \setminus L$,

$$g(A) = \Gamma(A \cup \{v \in L : y_L(v) = 1\}) \tag{2.49}$$

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 $A \subseteq V \setminus L$,

$$g(A) = \Gamma(A \cup \{v \in L : y_L(v) = 1\}) \tag{2.49}$$

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

Generalized Error Bound

Theorem 2.6.1 (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

$$\|\hat{y} - y\|_1 \leq 2\frac{\Gamma(Y(y))}{\Psi(L)} \tag{2.50}$$

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:

$$\Psi(S) = \min_{T \subseteq V \setminus S : T \neq \emptyset} \frac{\Gamma(T)}{|T|} \tag{2.51}$$

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

$$d_\phi^{\mathcal{H}_\phi}(x, y) = \phi(x) - \phi(y) - \langle \mathcal{H}_\phi(y), x - y \rangle, \forall x, y \in \text{dom}(\phi) \quad (2.52)$$

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:

$$d_\phi^{\mathcal{H}_\phi}(x, y) = \phi(x) - \phi(y) - \langle \mathcal{H}_\phi(y), x - y \rangle, \forall x, y \in \mathrm{dom}(\phi) \tag{2.52}$$

- 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:

$$d_\phi^{\mathcal{H}_\phi}(x, y) = \phi(x) - \phi(y) - \langle \mathcal{H}_\phi(y), x - y \rangle, \forall x, y \in \mathrm{dom}(\phi) \quad (2.52)$$

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

$$d_f^{\mathcal{H}_f}(X, Y) = f(X) - f(Y) - \langle \mathcal{H}_f(Y), 1_X - 1_Y \rangle \quad (2.53)$$

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:

$$d_\phi^{\mathcal{H}_\phi}(x, y) = \phi(x) - \phi(y) - \langle \mathcal{H}_\phi(y), x - y \rangle, \forall x, y \in \text{dom}(\phi) \tag{2.52}$$

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

$$d_f^{\mathcal{H}_f}(X, Y) = f(X) - f(Y) - \langle \mathcal{H}_f(Y), 1_X - 1_Y \rangle \tag{2.53}$$

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:

$$d_\phi^{\mathcal{H}_\phi}(x, y) = \phi(x) - \phi(y) - \langle \mathcal{H}_\phi(y), x - y \rangle, \forall x, y \in \text{dom}(\phi) \quad (2.52)$$

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

$$d_f^{\mathcal{H}_f}(X, Y) = f(X) - f(Y) - \langle \mathcal{H}_f(Y), 1_X - 1_Y \rangle \quad (2.53)$$

  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 $g : \mathbb{N} \to \mathbb{R}$?" Many results, including that even with adaptive queries and monotone functions, can't do better than $\Omega(\sqrt{n}/\log n)$.
- *Balcan & Harvey (2011)*: submodular function learning problem from a learning theory perspective, given a distribution on subsets. Negative result is that can't approximate in this setting to within a constant factor.

Learning Submodular Functions

- Learning submodular functions is hard
- *Goemans et al. (2009)*: "can one make only polynomial number of queries to an unknown submodular function f and constructs a $\hat{f}$ such that $\hat{f}(S) \leq f(S) \leq g(n)\hat{f}(S)$ where $g : \mathbb{N} \to \mathbb{R}$?" Many results, including that even with adaptive queries and monotone functions, can't do better than $\Omega(\sqrt{n}/\log n)$.
- *Balcan & Harvey (2011)*: submodular function learning problem from a learning theory perspective, given a distribution on subsets. Negative result is that can't approximate in this setting to within a constant factor.
- *Feldman, Kothari, Vondrák (2013)*, shows in some learning settings, things are more helpful

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.
- *Feldman, Kothari, Vondrák (2013)*, shows in some learning settings, things are more helpful
- One example: can we learn a subclass, perhaps non-negative weighted mixtures of submodular components?

Structured Learning of Submodular Mixtures

- Constraints specified in inference form:

$$\underset{\mathbf{w},\xi_t}{\text{minimize}} \quad \frac{1}{T}\sum_t \xi_t + \frac{\lambda}{2}\|\mathbf{w}\|^2 \tag{2.54}$$

$$\text{subject to} \quad \mathbf{w}^\top \mathbf{f}_t(\mathbf{y}^{(t)}) \geq \max_{\mathbf{y}\in\mathcal{Y}_t}\Big(\mathbf{w}^\top \mathbf{f}_t(\mathbf{y}) + \ell_t(\mathbf{y})\Big) - \xi_t, \forall t \tag{2.55}$$

$$\xi_t \geq 0, \forall t. \tag{2.56}$$

Structured Learning of Submodular Mixtures

- Constraints specified in inference form:

$$\underset{\mathbf{w},\xi_t}{\text{minimize}} \quad \frac{1}{T}\sum_t \xi_t + \frac{\lambda}{2}\|\mathbf{w}\|^2 \tag{2.54}$$

$$\text{subject to} \quad \mathbf{w}^\top \mathbf{f}_t(\mathbf{y}^{(t)}) \geq \max_{\mathbf{y}\in\mathcal{Y}_t}\Big(\mathbf{w}^\top \mathbf{f}_t(\mathbf{y}) + \ell_t(\mathbf{y})\Big) - \xi_t, \forall t \tag{2.55}$$

$$\xi_t \geq 0, \forall t. \tag{2.56}$$

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

Structured Learning of Submodular Mixtures

- Constraints specified in inference form:

$$\underset{\mathbf{w},\xi_t}{\text{minimize}} \quad \frac{1}{T}\sum_t \xi_t + \frac{\lambda}{2}\|\mathbf{w}\|^2 \tag{2.54}$$

$$\text{subject to} \quad \mathbf{w}^\top \mathbf{f}_t(\mathbf{y}^{(t)}) \geq \max_{\mathbf{y}\in\mathcal{Y}_t}\Big(\mathbf{w}^\top \mathbf{f}_t(\mathbf{y}) + \ell_t(\mathbf{y})\Big) - \xi_t, \forall t \tag{2.55}$$

$$\xi_t \geq 0, \forall t. \tag{2.56}$$

- 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:

$$\underset{\mathbf{w},\xi_t}{\text{minimize}} \quad \frac{1}{T}\sum_t \xi_t + \frac{\lambda}{2}\|\mathbf{w}\|^2 \tag{2.54}$$

$$\text{subject to} \quad \mathbf{w}^\top \mathbf{f}_t(\mathbf{y}^{(t)}) \geq \max_{\mathbf{y}\in\mathcal{Y}_t}\left(\mathbf{w}^\top \mathbf{f}_t(\mathbf{y}) + \ell_t(\mathbf{y})\right) - \xi_t, \forall t \tag{2.55}$$

$$\xi_t \geq 0, \forall t. \tag{2.56}$$

- 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:

$$\underset{\mathbf{w},\xi_t}{\text{minimize}} \quad \frac{1}{T}\sum_t \xi_t + \frac{\lambda}{2}\|\mathbf{w}\|^2 \tag{2.54}$$

$$\text{subject to} \quad \mathbf{w}^\top \mathbf{f}_t(\mathbf{y}^{(t)}) \geq \max_{\mathbf{y}\in\mathcal{Y}_t}\Big(\mathbf{w}^\top \mathbf{f}_t(\mathbf{y}) + \ell_t(\mathbf{y})\Big) - \xi_t, \forall t \tag{2.55}$$

$$\xi_t \geq 0, \forall t. \tag{2.56}$$

- 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

$$\Pr(x) = \frac{1}{Z}\exp(-E(x)) \tag{2.57}$$

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

$$\Pr(x) = \frac{1}{Z} \exp(-E(x)) \tag{2.57}$$

  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

$$\Pr(x) = \frac{1}{Z}\exp(-E(x)) \tag{2.57}$$

  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):

$$\gamma_{U,k}(f) = \min_{L \subseteq U, S:|S| \leq k, S \cap L = \emptyset} \frac{\sum_{s \in S} f(x|L)}{f(S|L)} \tag{2.58}$$

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):

$$\gamma_{U,k}(f) = \min_{L \subseteq U, S: |S| \leq k, S \cap L = \emptyset} \frac{\sum_{s \in S} f(x|L)}{f(S|L)} \tag{2.58}$$

- 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):

$$\gamma_{U,k}(f) = \min_{L \subseteq U, S:|S| \leq k, S \cap L = \emptyset} \frac{\sum_{s \in S} f(x|L)}{f(S|L)} \tag{2.58}$$

- f is submodular if $\gamma_{U,k} \geq 1$ for all U and k.
- For some variable selection problems, can get bounds of the form:

$$\text{Solution} \geq \left(1 - \frac{1}{e^{\gamma_{U^*,k}}}\right)\text{OPT} \tag{2.59}$$

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):

$$\gamma_{U,k}(f) = \min_{L \subseteq U, S:|S| \leq k, S \cap L = \emptyset} \frac{\sum_{s \in S} f(x|L)}{f(S|L)} \tag{2.58}$$

- 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:

$$\text{Solution} \geq (1 - \frac{1}{e^{\gamma_{U^*,k}}})\text{OPT} \tag{2.59}$$

  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):

$$\gamma_{U,k}(f) = \min_{L \subseteq U, S:|S| \leq k, S \cap L = \emptyset} \frac{\sum_{s \in S} f(x|L)}{f(S|L)} \tag{2.58}$$

- 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:

$$\text{Solution} \geq (1 - \frac{1}{e^{\gamma_{U^*,k}}})\text{OPT} \tag{2.59}$$

  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.

Recall

The next page shows a slide from Lecture 1

Submodular-Supermodular Decomposition

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

Theorem 2.8.1 (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 $A \subseteq V$,*

$$h(A) = f(A) + g(A) \tag{2.8}$$

- 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).

Applications of DS functions

Any function $h : 2^V \to \mathbb{R}$ can be expressed as a difference between two submodular (DS) functions, $h = f - g$.

- Sensor placement with submodular costs. I.e., let V be a set of possible sensor locations, $f(A) = I(X_A; X_{V \setminus A})$ measures the quality of a subset A of placed sensors, and $c(A)$ the submodular cost. We have $f(A) - \lambda c(A)$ as the overall objective to maximize.

Applications of DS functions

Any function $h : 2^V \to \mathbb{R}$ can be expressed as a difference between two submodular (DS) functions, $h = f - g$.

- Sensor placement with submodular costs. I.e., let V be a set of possible sensor locations, $f(A) = I(X_A; X_{V \setminus A})$ measures the quality of a subset A of placed sensors, and $c(A)$ the submodular cost. We have $f(A) - \lambda c(A)$ as the overall objective to maximize.
- Discriminatively structured graphical models, EAR measure $I(X_A; X_{V \setminus A}) - I(X_A; X_{V \setminus A} | C)$, and synergy in neuroscience.

Applications of DS functions

Any function $h : 2^V \to \mathbb{R}$ can be expressed as a difference between two submodular (DS) functions, $h = f - g$.

- Sensor placement with submodular costs. I.e., let V be a set of possible sensor locations, $f(A) = I(X_A; X_{V \setminus A})$ measures the quality of a subset A of placed sensors, and $c(A)$ the submodular cost. We have $f(A) - \lambda c(A)$ as the overall objective to maximize.
- Discriminatively structured graphical models, EAR measure $I(X_A; X_{V \setminus A}) - I(X_A; X_{V \setminus A} | C)$, and synergy in neuroscience.
- Feature selection: a problem of maximizing $I(X_A; C) - \lambda c(A) = H(X_A) - [H(X_A | C) + \lambda c(A)]$, the difference between two submodular functions, where H is the entropy and c is a feature cost function.

Applications of DS functions

Any function $h : 2^V \to \mathbb{R}$ can be expressed as a difference between two submodular (DS) functions, $h = f - g$.

- Sensor placement with submodular costs. I.e., let V be a set of possible sensor locations, $f(A) = I(X_A; X_{V \setminus A})$ measures the quality of a subset A of placed sensors, and $c(A)$ the submodular cost. We have $f(A) - \lambda c(A)$ as the overall objective to maximize.
- Discriminatively structured graphical models, EAR measure $I(X_A; X_{V \setminus A}) - I(X_A; X_{V \setminus A} | C)$, and synergy in neuroscience.
- Feature selection: a problem of maximizing $I(X_A; C) - \lambda c(A) = H(X_A) - [H(X_A|C) + \lambda c(A)]$, the difference between two submodular functions, where H is the entropy and c is a feature cost function.
- Graphical Model Inference. Finding x that maximizes $p(x) \propto \exp(-v(x))$ where $x \in \{0, 1\}^n$ and v is a pseudo-Boolean function. When v is non-submodular, it can be represented as a difference between submodular functions.