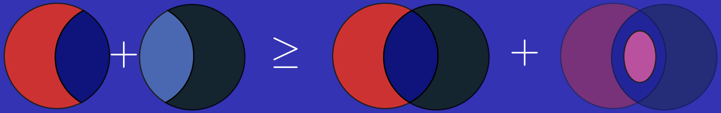


Submodularity in Machine Learning

Scalable Algorithms for Submodular Optimization

Rishabh Iyer and Jeffrey A. Bilmes

Departments of Electrical Engineering
& Computer Science and Engineering
University of Washington, Seattle

June 18th, 2014

Outline

Acknowledgments

Special Thanks to my colleagues and collaborators, Jeff Bilmes, Stefanie Jegelka, Kai Wei, Jennifer Gillenwater, Ganesh Ramakrishnan, Sebastian Tschiatschek, Yoshinobu Kawahara, Ganesh Ramakrishnan, Matthai Phillipose, Bethany Herwaldt, Shengjie Wang, Wenruo Bai.

Outline

Recap: Submodular Functions

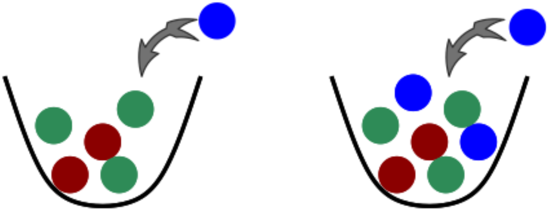
- Special class of set functions.

$$f(A \cup v) - f(A) \geq f(B \cup v) - f(B), \text{ if } A \subseteq B \tag{1}$$

Recap: Submodular Functions

- Special class of set functions.

$$f(A \cup v) - f(A) \geq f(B \cup v) - f(B), \text{ if } A \subseteq B \quad (1)$$

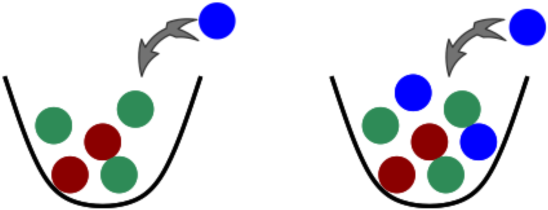


$f = \#$ of distinct colors of balls in the urn.

Recap: Submodular Functions

- Special class of set functions.

$$f(A \cup v) - f(A) \geq f(B \cup v) - f(B), \text{ if } A \subseteq B \qquad (1)$$



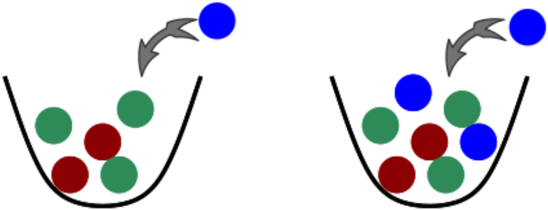
$f = \#$ of distinct colors of balls in the urn.

Gain $= 1$

Recap: Submodular Functions

- Special class of set functions.

$$f(A \cup v) - f(A) \geq f(B \cup v) - f(B), \text{ if } A \subseteq B \quad (1)$$



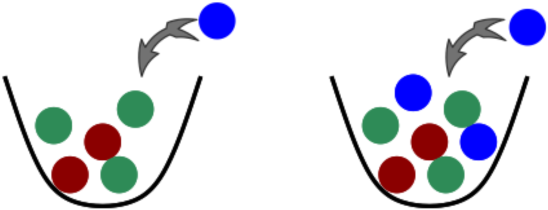
$f = \#$ of distinct colors of balls in the urn.

Gain $= 1$ Gain $= 0$

Recap: Submodular Functions

- Special class of set functions.

$$f(A \cup v) - f(A) \geq f(B \cup v) - f(B), \text{ if } A \subseteq B \quad (1)$$



$f = \#$ of distinct colors of balls in the urn.

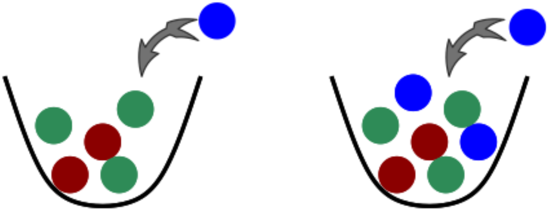
Gain $= 1$ Gain $= 0$

- Monotonicity: $\mathbf{f}(\mathbf{A}) \leq \mathbf{f}(\mathbf{B})$, if $\mathbf{A} \subseteq \mathbf{B}$.

Recap: Submodular Functions

- Special class of set functions.

$$f(A \cup v) - f(A) \geq f(B \cup v) - f(B), \text{ if } A \subseteq B \qquad (1)$$



$f = \#$ of distinct colors of balls in the urn.

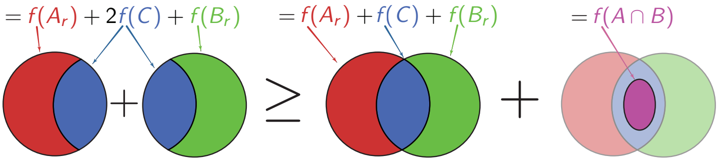
Gain $= 1$ Gain $= 0$

- Monotonicity: $\mathbf{f}(\mathbf{A}) \leq \mathbf{f}(\mathbf{B})$, if $\mathbf{A} \subseteq \mathbf{B}$.
- Modular function $\mathbf{f}(\mathbf{X}) = \sum_{\mathbf{i} \in \mathbf{X}} \mathbf{f}(\mathbf{i})$ analogous to linear functions.

Recap: Alternate definition – Submodular Functions

- A function $f : 2^V \to \mathbb{R}$ is submodular if:

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

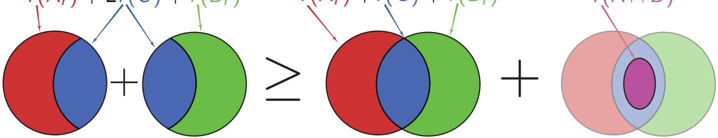


Recap: Alternate definition – Submodular Functions

- A function $f : 2^V \to \mathbb{R}$ is submodular if:

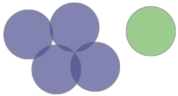
$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$$

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

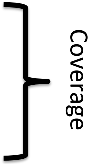


- Submodularity has been widely used: non-additive measure theory, economics, game theory, statistical physics and thermodynamics, electrical networks, and operations research.

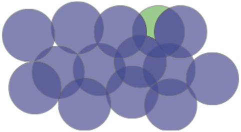
Facets of Submodular functions (models in maximization)



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



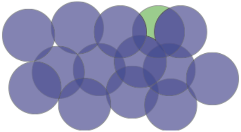
Facets of Submodular functions (models in maximization)



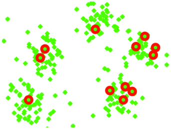
$f(S) = \bigcup_{s \in S} \text{area}(s)$

Coverage

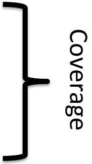
Facets of Submodular functions (models in maximization)



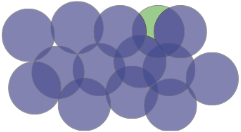
$f(S) = \bigcup_{s \in S} \text{area}(s)$



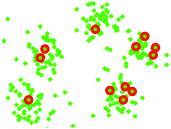
$F(S) = \sum_{i \in V} \max_{j \in S} s_{ij}$



Facets of Submodular functions (models in maximization)

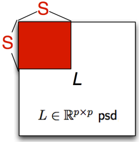


$f(S) = \bigcup_{s \in S} \text{area}(s)$



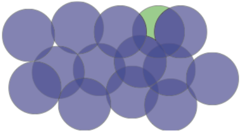
$F(S) = \sum_{i \in V} \max_{j \in S} s_{ij}$

Coverage

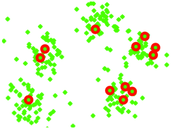


Diversity

Facets of Submodular functions (models in maximization)

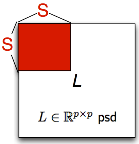


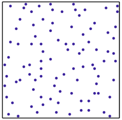
$f(S) = \bigcup_{s \in S} \text{area}(s)$



$F(S) = \sum_{i \in V} \max_{j \in S} s_{ij}$

Coverage

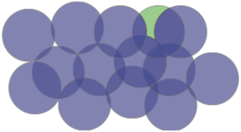




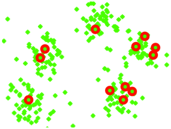
$F(S) = \log \det(L_S)$

Diversity

Facets of Submodular functions (models in maximization)

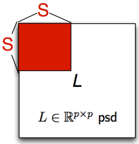


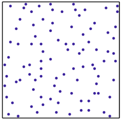
$f(S) = \bigcup_{s \in S} \text{area}(s)$



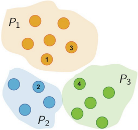
$F(S) = \sum_{i \in V} \max_{j \in S} s_{ij}$

Coverage





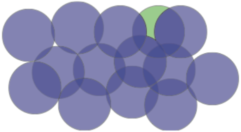
$$F(S) = \log\det(L_S)$$



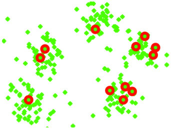
$F(S) = \sum_{k=1}^{3} \sqrt{\sum_{i \in S \cap P_i} r_i}$

Diversity

Facets of Submodular functions (models in maximization)

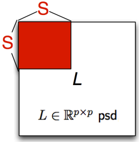


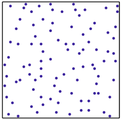
$f(S) = \bigcup_{s \in S} \text{area}(s)$



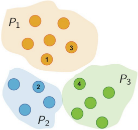
$F(S) = \sum_{i \in V} \max_{j \in S} s_{ij}$

Coverage





$F(S) = \log\det(L_S)$



$F(S) = \sum_{k=1}^{3} \sqrt{\sum_{i \in S \cap P_i} r_i}$

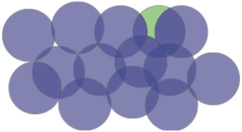
Diversity

$F(S) = I(X_A; Y) = H(X_A) - H(X_A|Y)$

Information

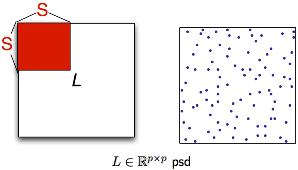
Facets of Submodular functions (models in maximization)

$$F(A) = \cup_{s \in A} \text{area}(s)$$



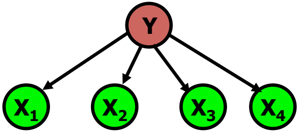
Coverage

$$F(A) = \log det(L_A)$$



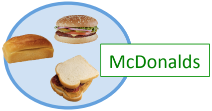
Diversity

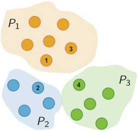
$$F(A) = H(X_A)$$



Information

Facets of Submodular functions (models in minimization)

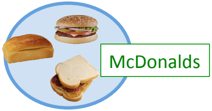


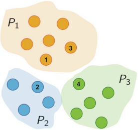


$$F(S) = \sum_{k=1}^{3} \sqrt{\sum_{i \in S \cap P_i} r_i}$$

Cooperative Costs

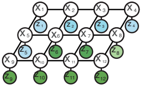
Facets of Submodular functions (models in minimization)

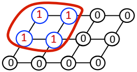




$$F(A) = \sum_{i=1}^{3} \sqrt{\sum_{j \in A \cap P_i} r_i}$$

Cooperative Costs

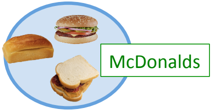


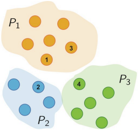


$$E(\mathbf{x};\mathbf{z}) = \sum_i E_i(x_i) + \sum_{ij} E_{ij}(x_i, x_j)$$

Attractive Potentials

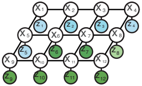
Facets of Submodular functions (models in minimization)

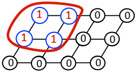




$$F(A) = \sum_{i=1}^{3} \sqrt{\sum_{j \in A \cap P_i} r_i}$$

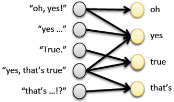
Cooperative Costs



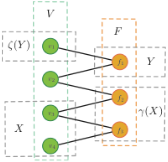


$$E(\mathbf{x};\mathbf{z}) = \sum_i E_i(x_i) + \sum_{ij} E_{ij}(x_i, x_j)$$

Attractive Potentials



$$F(A) = \gamma(A)$$



(a) Bipartite graph

Complexity

Recap: Submodular Optimization

- Submodular Optimization Problems:

$$\begin{aligned} \underset{S \subseteq V}{\text{maximize}} \quad & f(S) \\ \text{subject to} \quad & S \in \mathcal{C} \end{aligned} \tag{2}$$

$$\begin{aligned} \underset{S \subseteq V}{\text{minimize}} \quad & f(S) \\ \text{subject to} \quad & S \in \mathcal{C} \end{aligned} \tag{3}$$

Recap: Submodular Optimization

- Submodular Optimization Problems:

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

$$\begin{array}{ll} \underset{S \subseteq V}{\text{minimize}} & f(S) \\ \text{subject to} & S \in \mathcal{C} \end{array} \quad (3)$$

- Unconstrained: $\mathcal{C} = 2^V$, Constrained: $\mathcal{C} \subseteq 2^V$.

Recap: Submodular Optimization

- Submodular Optimization Problems:

$$\begin{array}{ll} \underset{S \subseteq V}{\text{maximize}} & f(S) \\ \text{subject to} & S \in \mathcal{C} \end{array} \quad (2) \qquad \begin{array}{ll} \underset{S \subseteq V}{\text{minimize}} & f(S) \\ \text{subject to} & S \in \mathcal{C} \end{array} \quad (3)$$

- Unconstrained: $\mathcal{C} = 2^V$, Constrained: $\mathcal{C} \subseteq 2^V$.
- Bounded size $\mathcal{C} = \{S \subseteq V : |S| \leq k\}$, knapsack bounded budget $\{S \subseteq V : \sum_{s \in S} w(s) \leq b\}$

Recap: Submodular Optimization

- Submodular Optimization Problems:

$$\begin{array}{ll} \underset{S \subseteq V}{\text{maximize}} & f(S) \\ \text{subject to} & S \in \mathcal{C} \end{array} \quad (2) \qquad \begin{array}{ll} \underset{S \subseteq V}{\text{minimize}} & f(S) \\ \text{subject to} & S \in \mathcal{C} \end{array} \quad (3)$$

- Unconstrained: $\mathcal{C} = 2^V$, Constrained: $\mathcal{C} \subseteq 2^V$.
- Bounded size $\mathcal{C} = \{S \subseteq V : |S| \leq k\}$, knapsack bounded budget $\{S \subseteq V : \sum_{s \in S} w(s) \leq b\}$
- Matroid independence constraints, or independence in multiple matroids, or matroids + knapsack.

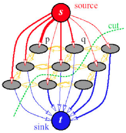
Recap: Submodular Optimization

- Submodular Optimization Problems:

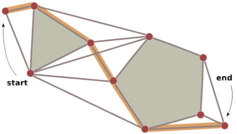
$$\begin{array}{ll} \underset{S \subseteq V}{\text{maximize}} & f(S) \\ \text{subject to} & S \in \mathcal{C} \end{array} \quad (2)$$

$$\begin{array}{ll} \underset{S \subseteq V}{\text{minimize}} & f(S) \\ \text{subject to} & S \in \mathcal{C} \end{array} \quad (3)$$

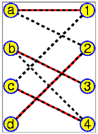
- Combinatorial constraint (i.e., feasible $\mathcal{C}$ might be trees, matchings, paths, vertex covers, or cuts).



cuts



paths



matchings

- Sub-level sets of g, $\mathcal{C} = \{S \subseteq V : g(S) \leq \alpha\}$, sup-level sets $\mathcal{C} = \{S \subseteq V : g(S) \geq \alpha\}$).
- When f (and g) are submodular, there is hope to solve these problems with guarantees, practically and scalably!

Recap: Submodular Optimization

$f \to$ cooperative cost or algorithmic complexity.
$g \to$ coverage/ diversity or information.

- Submodular Minimization:

$$\min_{X \in \mathcal{C}} f(X) \quad \Longrightarrow \quad \text{Minimizing Cooperative Costs}$$

- Submodular Maximization:

$$\max_{X \in \mathcal{C}} g(X) \quad \Longrightarrow \quad \text{Maximizing Coverage/ Diversity}$$

- Difference of Submodular (DS) optimization:

$$\min_{X} f(X) - \lambda g(X)$$

- Submodular optimization under submodular constraints,

$$\min\{f(X) : g(X) \geq c\}, \quad \max\{g(X) : f(X) \leq b\}$$

(DS optimization and submodular constraints) → Minimizing cooperative costs and Maximizing Coverage/ Diversity

⇓

Unifying Algorithmic Framework

Approximation Algorithms

Most of these problems are NP-hard!

Use the notion of approximation algorithms!

Minimization: $\mathrm{OPT} \leq f(X) \leq \rho\mathrm{OPT}, \rho > 1$

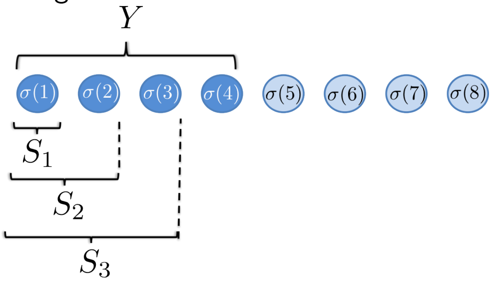
Maximization: $\rho\mathrm{OPT} \leq f(X) \leq \mathrm{OPT}, \rho < 1$

Recap: Submodular Subgradients (Fujishige 2005)

- Like convex functions, a submodular function g has subgradients. Defined at any $Y \subseteq V$.

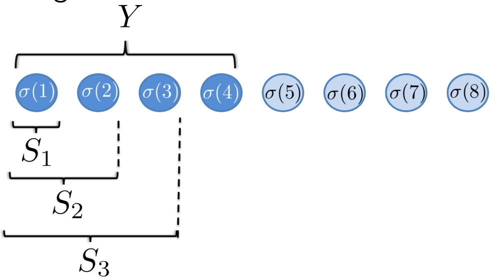
Recap: Submodular Subgradients (Fujishige 2005)

- Like convex functions, a submodular function g has subgradients. Defined at any $Y \subseteq V$.
- An ordering σ of the ground set.



Recap: Submodular Subgradients (Fujishige 2005)

- Like convex functions, a submodular function g has subgradients. Defined at any $Y \subseteq V$.
- An ordering σ of the ground set.

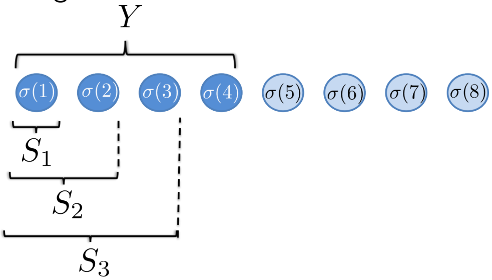


- Corresponding subgradient h_Y^σ is:

$$h_Y^\sigma(\sigma(i)) = g(\sigma(i)|S_{i-1})$$

Recap: Submodular Subgradients (Fujishige 2005)

- Like convex functions, a submodular function g has subgradients. Defined at any $Y \subseteq V$.
- An ordering σ of the ground set.



- Corresponding subgradient h_Y^σ is:

$$h_Y^\sigma(\sigma(i)) = g(\sigma(i) | S_{i-1})$$

- Modular lower bound: $m_{h_Y}(X) = g(Y) + h_Y(X) - h_Y(Y) \leq g(X)$.

Recap: Submodular Supergradients (Iyer et al, 2013)

- Define gain of j in context of A: $f(j|A) \triangleq f(A \cup j) - f(A)$

Recap: Submodular Supergradients (Iyer et al, 2013)

- Define gain of j in context of A: $f(j|A) \triangleq f(A \cup j) - f(A)$
- Unlike convex functions, surprisingly, we show that submodular functions also have super-gradients. Defined at any $Y \subseteq V$.

Recap: Submodular Supergradients (Iyer et al, 2013)

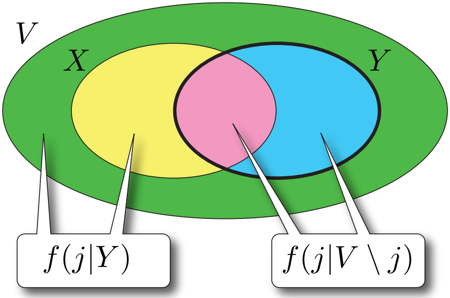
- Define gain of j in context of A: $f(j|A) \triangleq f(A \cup j) - f(A)$
- Unlike convex functions, surprisingly, we show that submodular functions also have super-gradients. Defined at any $Y \subseteq V$.
- Three of these supergradients (which we call grow, shrink, and bar) are in fact easy to obtain.

Recap: Submodular Supergradients (Iyer et al, 2013)

- Define gain of j in context of A: $f(j|A) \triangleq f(A \cup j) - f(A)$
- Unlike convex functions, surprisingly, we show that submodular functions also have super-gradients. Defined at any $Y \subseteq V$.
- Three of these supergradients (which we call grow, shrink, and bar) are in fact easy to obtain.

Grow:

$$\hat{g}_Y(j) = \begin{cases} f(j|Y) & \text{for } j \notin Y \\ f(j|V\backslash\{j\}) & \text{for } j \in Y \end{cases}$$

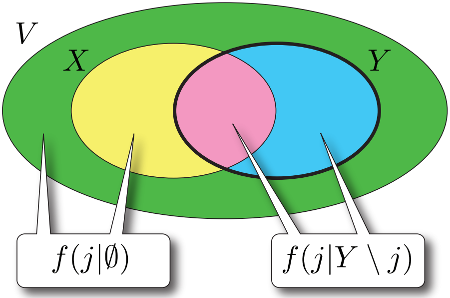


Recap: Submodular Supergradients (Iyer et al, 2013)

- Define gain of j in context of A: $f(j|A) \triangleq f(A \cup j) - f(A)$
- Unlike convex functions, surprisingly, we show that submodular functions also have super-gradients. Defined at any $Y \subseteq V$.
- Three of these supergradients (which we call grow, shrink, and bar) are in fact easy to obtain.

Shrink:

$$\check{g}_Y(j) = \begin{cases} f(j|\emptyset) & \text{for } j \notin Y \\ f(j|Y \backslash \{j\}) & \text{for } j \in Y \end{cases}$$

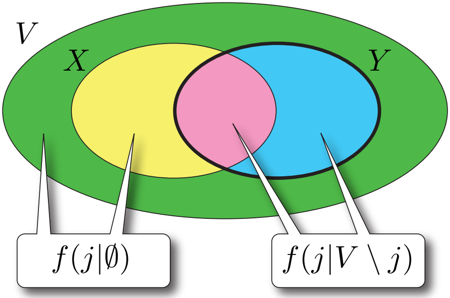


Recap: Submodular Supergradients (Iyer et al, 2013)

- Define gain of j in context of A: $f(j|A) \triangleq f(A \cup j) - f(A)$
- Unlike convex functions, surprisingly, we show that submodular functions also have super-gradients. Defined at any $Y \subseteq V$.
- Three of these supergradients (which we call grow, shrink, and bar) are in fact easy to obtain.

Bar:

$$\bar{g}_Y(j) = \begin{cases} f(j|\emptyset) & \text{for } j \notin Y \\ f(j|V \backslash \{j\}) & \text{for } j \in Y \end{cases}$$

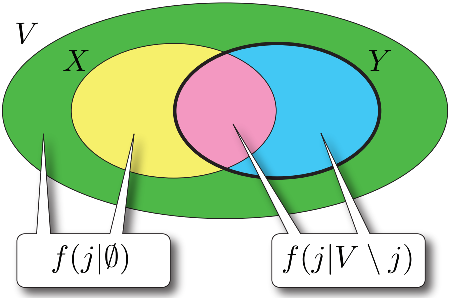


Recap: Submodular Supergradients (Iyer et al, 2013)

- Define gain of j in context of A: $f(j|A) \triangleq f(A \cup j) - f(A)$
- Unlike convex functions, surprisingly, we show that submodular functions also have super-gradients. Defined at any $Y \subseteq V$.
- Three of these supergradients (which we call grow, shrink, and bar) are in fact easy to obtain.

Bar:

$$\bar{g}_Y(j) = \begin{cases} f(j|\emptyset) & \text{for } j \notin Y \\ f(j|V \backslash \{j\}) & \text{for } j \in Y \end{cases}$$



- Modular upper bound: $m^{g_Y}(X) = f(Y) + g_Y(X) - g_Y(Y) \leq f(X)$.

Recap: Continuous Extensions of Submodular Functions

- A natural convex extension of a submodular function is the Lovász extension (Lovász, 1983).
- This extension is easy to evaluate and optimize! ☺
- A natural (near-concave) extension is the multilinear extension!

$$\tilde{f}(x) = \sum_{X \subseteq V} f(X) \prod_{i \in X} x_i \prod_{i \notin X} (1 - x_i) \tag{4}$$

- Requires an exponential sum ☹, but can be approximated through sampling (Vondrak, 2007).
- For subclasses of submodular functions, one can compute the exact multilinear extension! (I-Jegelka-Bilmes, 2014) ☺

Submodular Optimization Survey

Combinatorial Algorithms

- Ellipsoidal Approximation and Exact SFM Algorithms

  Slow but tight

Continuous Relaxations

Submodular Optimization Survey

Combinatorial Algorithms

- Ellipsoidal Approximation and Exact SFM Algorithms

  Slow but tight

- Majorization-Minimization Framework

  Fast and Scalable, and includes techniques like the greedy algorithm.

Continuous Relaxations

Submodular Optimization Survey

Combinatorial Algorithms

- Ellipsoidal Approximation and Exact SFM Algorithms

  Slow but tight

- Majorization-Minimization Framework

  Fast and Scalable, and includes techniques like the greedy algorithm.

Continuous Relaxations

- Use the convex or multilinear relaxations and rounding

Fast and Scalable

Unconstrained Submodular Minimization (USMin) Survey

- Unconstrained setting ($\mathcal{C} = 2^V$) is poly-time.

Unconstrained Submodular Minimization (USMin) Survey

- Unconstrained setting ($\mathcal{C} = 2^V$) is poly-time.
- Combinatorial Algorithms Iwata/Fleischer/Fujishige'00, Schrijver'00, Orlin'07, strongly polynomial $O(n^5\gamma + n^6)$ but high order (we shall very briefly touch upon these in this tutorial)

Unconstrained Submodular Minimization (USMin) Survey

- Unconstrained setting ($\mathcal{C} = 2^V$) is poly-time.
- Combinatorial Algorithms Iwata/Fleischer/Fujishige'00, Schrijver'00, Orlin'07, strongly polynomial $O(n^5\gamma + n^6)$ but high order (we shall very briefly touch upon these in this tutorial)
- Relaxations: Subgradient Descent using Lovász extension, and Minimum Norm point algorithm (Fujishige/Isotani'11, Wolfe'76, Chakrabarty'14, Bach'13, I-Jegelka-Bilmes, 2014) – We shall cover this in this tutorial.

Unconstrained Submodular Minimization (USMin) Survey

- Unconstrained setting ($\mathcal{C} = 2^V$) is poly-time.
- Combinatorial Algorithms Iwata/Fleischer/Fujishige'00, Schrijver'00, Orlin'07, strongly polynomial $O(n^5\gamma + n^6)$ but high order (we shall very briefly touch upon these in this tutorial)
- Relaxations: Subgradient Descent using Lovász extension, and Minimum Norm point algorithm (Fujishige/Isotani'11, Wolfe'76, Chakrabarty'14, Bach'13, I-Jegelka-Bilmes, 2014) – We shall cover this in this tutorial.
- Special cases: graph cuts/low-order (Kolmogorov), decomposable case (Stobbe & Krause, Jegelka et. al. 2011), etc.

Constrained Submodular Minimization

- Combinatorial Algorithms: Ellipsoidal Algorithms (Goemans, 2009) based on approximating the submodular function (only briefly touch on this)

Constrained Submodular Minimization

- Combinatorial Algorithms: Ellipsoidal Algorithms (Goemans, 2009) based on approximating the submodular function (only briefly touch on this)
- Majorization-Minimization, MMin (I-Jegelka-Bilmes 2013, I-Bilmes, 2013) – we shall cover this extensively.

Constrained Submodular Minimization

- Combinatorial Algorithms: Ellipsoidal Algorithms (Goemans, 2009) based on approximating the submodular function (only briefly touch on this)
- Majorization-Minimization, MMin (I-Jegelka-Bilmes 2013, I-Bilmes, 2013) – we shall cover this extensively.
- Relaxation based algorithms, using the Lovász extension (Nagano & Iwata, 2009, I-Jegelka-Bilmes, 2014). We shall also cover this extensively.

Submodular Maximization

- Submodular maximization: discrete algorithms, greedy, accelerated greedy, bidirectional greedy, randomized local search etc. (Nemhauser et al 1978, Buchbinder et al, 2012, 2014)

Submodular Maximization

- Submodular maximization: discrete algorithms, greedy, accelerated greedy, bidirectional greedy, randomized local search etc. (Nemhauser et al 1978, Buchbinder et al, 2012, 2014)
- Relaxations, using the Multilinear extension: continuous greedy, accelerated cont. greedy, continuous greedy with multiplicative weight updates, etc. in constrained settings (see Vondrák et al 2008 etc.),

Submodular Maximization

- Submodular maximization: discrete algorithms, greedy, accelerated greedy, bidirectional greedy, randomized local search etc. (Nemhauser et al 1978, Buchbinder et al, 2012, 2014)
- Relaxations, using the Multilinear extension: continuous greedy, accelerated cont. greedy, continuous greedy with multiplicative weight updates, etc. in constrained settings (see Vondrák et al 2008 etc.),
- The combinatorial algorithms for submodular maximization, can be unified within a Minorization-Maximization framework.

Overview of this part of the tutorial

- Relaxation based algorithms for submodular minimization and maximization.

Overview of this part of the tutorial

- Relaxation based algorithms for submodular minimization and maximization.
- Majorization-Minimization (and Minorization-Maximization) framework.

Overview of this part of the tutorial

- Relaxation based algorithms for submodular minimization and maximization.
- Majorization-Minimization (and Minorization-Maximization) framework.
- Greedy and Local search based techniques for submodular maximization.

Outline

Relaxation Algorithms

- Relaxation algorithms for minimization and maximization follow the two stage protocol:

Relaxation Algorithms

- Relaxation algorithms for minimization and maximization follow the two stage protocol:
  1. Find the optimal (or approximate) solution $\hat{x}$ to the problem $\min_{x \in \mathcal{P}_{\mathcal{C}}} \breve{f}(x)$ (or $\max_{x \in \mathcal{P}_{\mathcal{C}}} \tilde{f}(x)$).

Relaxation Algorithms

- Relaxation algorithms for minimization and maximization follow the two stage protocol:
    1. Find the optimal (or approximate) solution $\hat{x}$ to the problem $\min_{x \in \mathcal{P}_\mathcal{C}} \breve{f}(x)$ (or $\max_{x \in \mathcal{P}_\mathcal{C}} \tilde{f}(x)$).
    2. Round the continuous solution $\hat{x}$ to obtain the discrete indicator vector of set $\hat{X}$.

Relaxation Algorithms

- Relaxation algorithms for minimization and maximization follow the two stage protocol:
  1. Find the optimal (or approximate) solution $\hat{x}$ to the problem $\min_{x \in \mathcal{P}_{\mathcal{C}}} \breve{f}(x)$ (or $\max_{x \in \mathcal{P}_{\mathcal{C}}} \tilde{f}(x)$).
  2. Round the continuous solution $\hat{x}$ to obtain the discrete indicator vector of set $\hat{X}$.
- $\mathcal{P}_{\mathcal{C}}$ denotes the polytope corresponding to the family $\mathcal{C}$ of feasible set

Combinatorial Algorithms

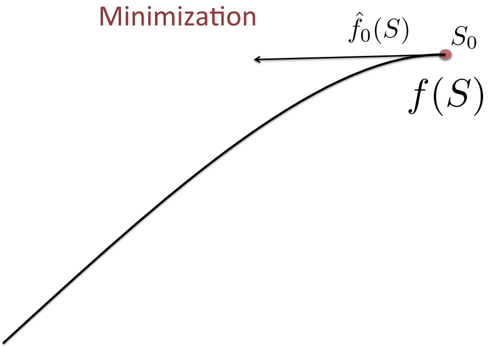
- In this tutorial, we shall study a class of majorization-minimization based algorithms which use the supergradients of a submodular function (for submodular minimization), and minorization-maximization algorithms which use the subgradients (for submodular maximization).

Combinatorial Algorithms

- In this tutorial, we shall study a class of majorization-minimization based algorithms which use the supergradients of a submodular function (for submodular minimization), and minorization-maximization algorithms which use the subgradients (for submodular maximization).
- The latter class of algorithms for submodular maximization subsume several greedy and local search algorithms.

Majorization-Minimization (MMin) Framework

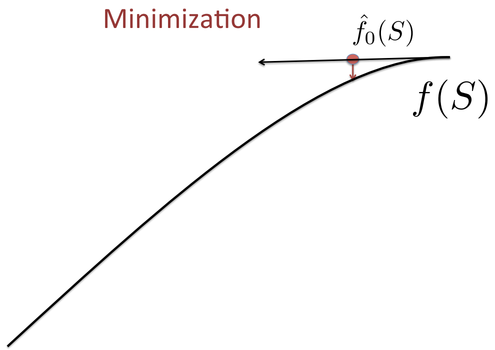
Idea: For minimization problems, use an upper bound of the objective function.



Always improves the objective value at every iteration!

Majorization-Minimization (MMin) Framework

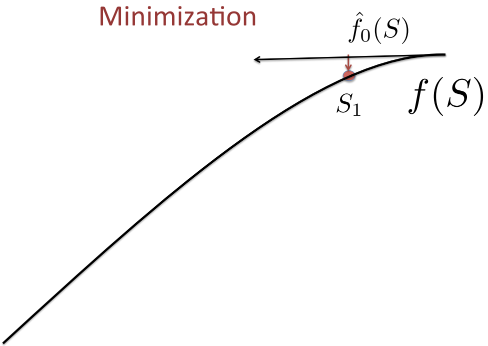
Idea: For minimization problems, use an upper bound of the objective function.



Always improves the objective value at every iteration!

Majorization-Minimization (MMin) Framework

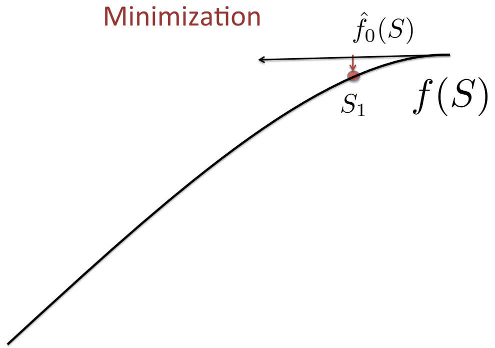
Idea: For minimization problems, use an upper bound of the objective function.



Always improves the objective value at every iteration!

Majorization-Minimization (MMin) Framework

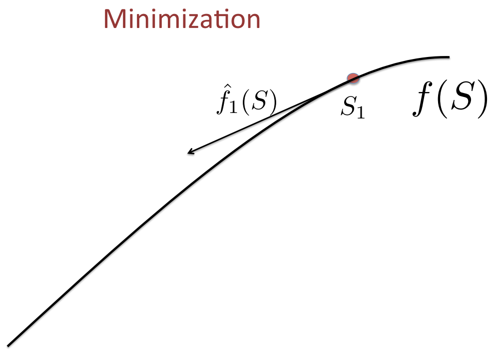
Idea: For minimization problems, use an upper bound of the objective function.



Always improves the objective value at every iteration!

Majorization-Minimization (MMin) Framework

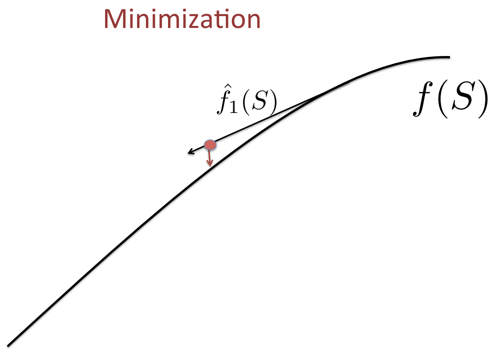
Idea: For minimization problems, use an upper bound of the objective function.



Always improves the objective value at every iteration!

Majorization-Minimization (MMin) Framework

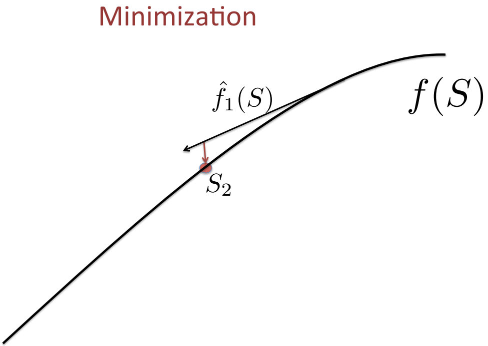
Idea: For minimization problems, use an upper bound of the objective function.



Always improves the objective value at every iteration!

Majorization-Minimization (MMin) Framework

Idea: For minimization problems, use an upper bound of the objective function.

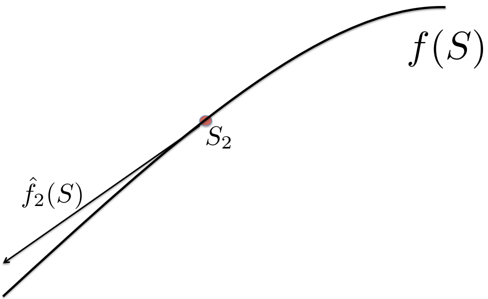


Always improves the objective value at every iteration!

Majorization-Minimization (MMin) Framework

Idea: For minimization problems, use an upper bound of the objective function.

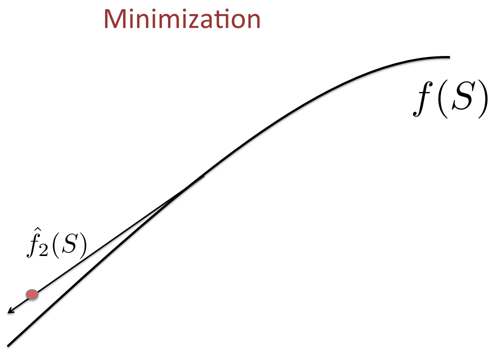
Minimization



Always improves the objective value at every iteration!

Majorization-Minimization (MMin) Framework

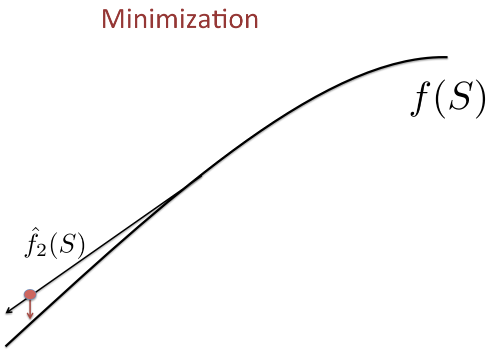
Idea: For minimization problems, use an upper bound of the objective function.



Always improves the objective value at every iteration!

Majorization-Minimization (MMin) Framework

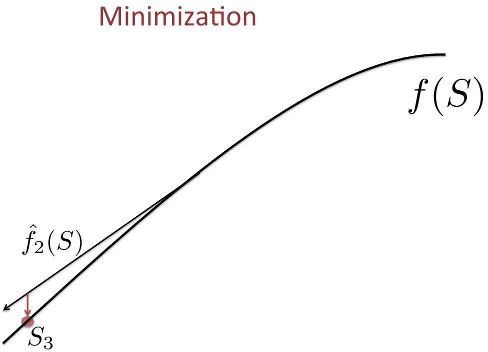
Idea: For minimization problems, use an upper bound of the objective function.



Always improves the objective value at every iteration!

Majorization-Minimization (MMin) Framework

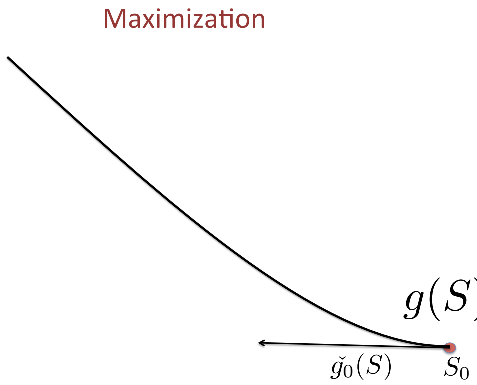
Idea: For minimization problems, use an upper bound of the objective function.



Always improves the objective value at every iteration!

Minorization-Maximization

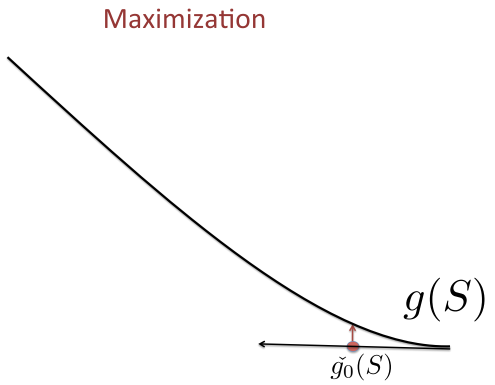
Idea: For maximization problems, use a lower bound of the objective function.



Always improves the objective value at every iteration!

Minorization-Maximization

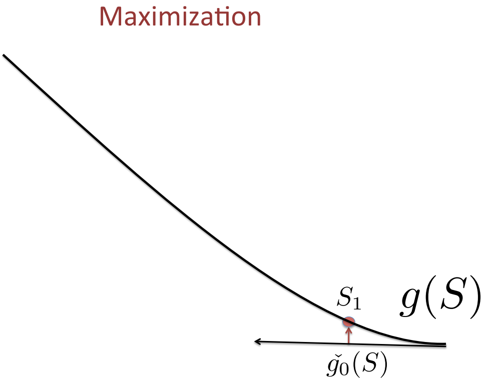
Idea: For maximization problems, use a lower bound of the objective function.



Always improves the objective value at every iteration!

Minorization-Maximization

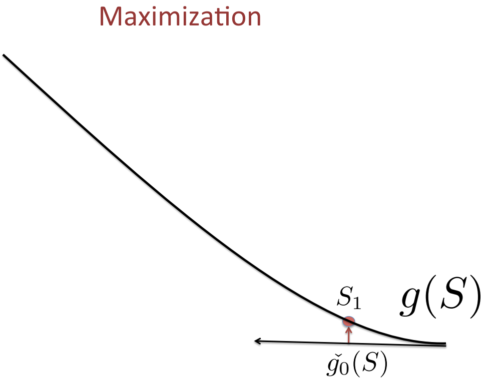
Idea: For maximization problems, use a lower bound of the objective function.



Always improves the objective value at every iteration!

Minorization-Maximization

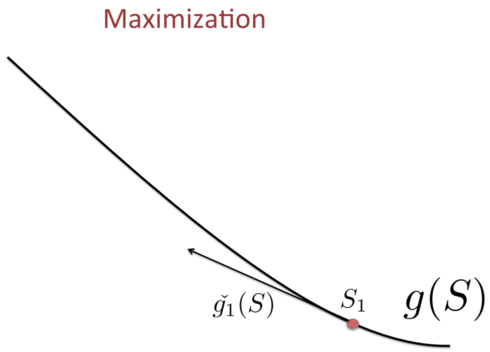
Idea: For maximization problems, use a lower bound of the objective function.



Always improves the objective value at every iteration!

Minorization-Maximization

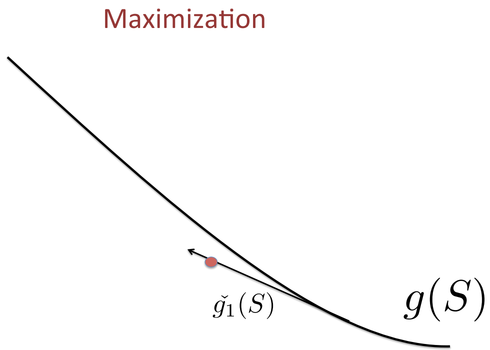
Idea: For maximization problems, use a lower bound of the objective function.



Always improves the objective value at every iteration!

Minorization-Maximization

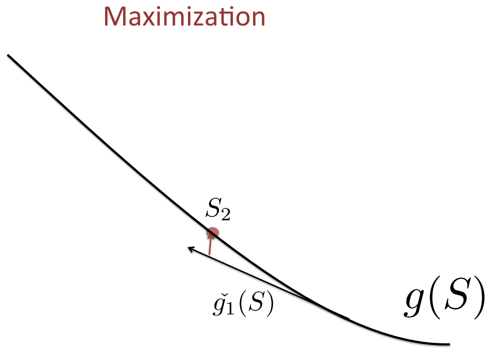
Idea: For maximization problems, use a lower bound of the objective function.



Always improves the objective value at every iteration!

Minorization-Maximization

Idea: For maximization problems, use a lower bound of the objective function.

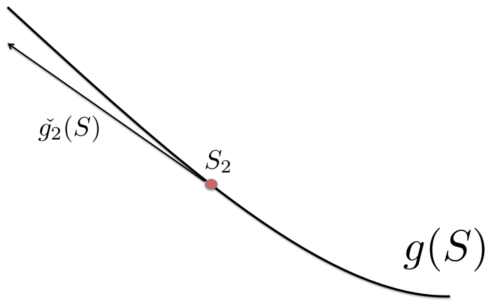


Always improves the objective value at every iteration!

Minorization-Maximization

Idea: For maximization problems, use a lower bound of the objective function.

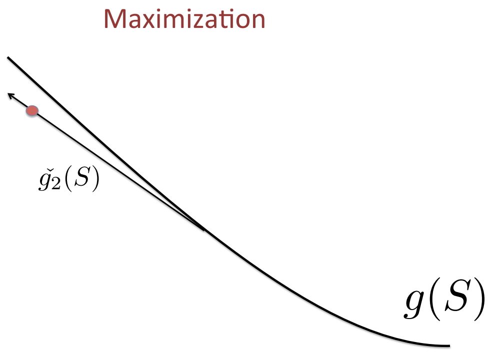
Maximization



Always improves the objective value at every iteration!

Minorization-Maximization

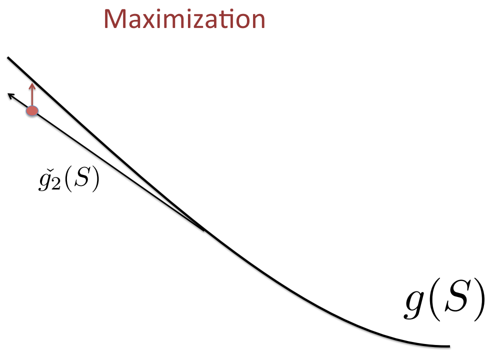
Idea: For maximization problems, use a lower bound of the objective function.



Always improves the objective value at every iteration!

Minorization-Maximization

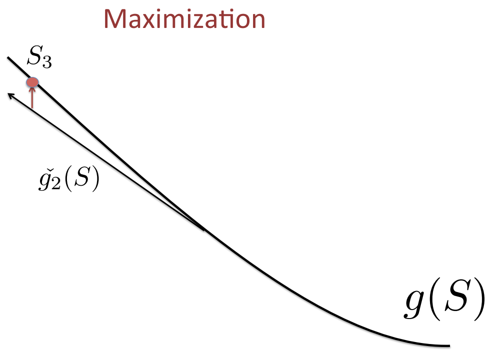
Idea: For maximization problems, use a lower bound of the objective function.



Always improves the objective value at every iteration!

Minorization-Maximization

Idea: For maximization problems, use a lower bound of the objective function.



Always improves the objective value at every iteration!

Outline

Submodular Function Minimization (SFM)

- This is an important problem concerning submodular functions, and can be defined as:

$$\min_{X \subseteq V} f(X) \tag{5}$$

Submodular Function Minimization (SFM)

- This is an important problem concerning submodular functions, and can be defined as:

$$\min_{X \subseteq V} f(X) \tag{5}$$

- A number of combinatorial algorithms exist which can solve this problem in polynomial time. However the complexity of the best known combinatorial algorithm is $O(n^6)$.

Submodular Function Minimization (SFM)

- This is an important problem concerning submodular functions, and can be defined as:
$$\min_{X \subseteq V} f(X) \tag{5}$$
- A number of combinatorial algorithms exist which can solve this problem in polynomial time. However the complexity of the best known combinatorial algorithm is $O(n^6)$.
- These techniques are slow and hard to implement ☹

Submodular Function Minimization (SFM)

- This is an important problem concerning submodular functions, and can be defined as:

$$\min_{X \subseteq V} f(X) \tag{5}$$

- A number of combinatorial algorithms exist which can solve this problem in polynomial time. However the complexity of the best known combinatorial algorithm is $O(n^6)$.
- These techniques are slow and hard to implement ☹
- In this tutorial, we shall study algorithms based on convex relaxations.

Submodular Function Minimization (SFM)

- This is an important problem concerning submodular functions, and can be defined as:

$$\min_{X \subseteq V} f(X) \tag{5}$$

- A number of combinatorial algorithms exist which can solve this problem in polynomial time. However the complexity of the best known combinatorial algorithm is $O(n^6)$.
- These techniques are slow and hard to implement ☹
- In this tutorial, we shall study algorithms based on convex relaxations.
- Uses a lot of the nice connections between convexity and submodularity.

Outline

SFM and convex programming

- The problem of minimizing submodular functions on the boolean hypercube (without constraints) is equivalent to minimizing the Lovász extention.

SFM and convex programming

- The problem of minimizing submodular functions on the boolean hypercube (without constraints) is equivalent to minimizing the Lovász extention.

Theorem 1

$$\min_{X \subseteq V} f(X) = \min_{x \in [0,1]^n} \hat{f}(x) \tag{6}$$

Furthermore, from a minimizer w of the Lovász extention, the minimizers of the submodular function f can be obtained as the support sets $S_i : w(S_i) - w(S_{i-1}) \neq 0$. Hence each minimizer of the Lovász extention produces a chain of minimizers of the corresponding submodular function.

Algorithms in this section

1. Minimum Norm Point Algorithm.
2. Sub-gradient descent algorithm.
3. Conditional gradient descent algorithm.
4. Smoothing in special cases of decomposable functions.

The Minimum Norm Point Algorithm (Fujishige, 2005)

- We have the followng duality relationship:

Theorem 2

For a submodular function f,

$$\min_{X \subseteq V} f(X) = \max_{y \in P_f, y \leq 0} y(E) \tag{7}$$

Further consider the following quadratic program over the base polytope.

$$\min_{x \in B_f} \|x\|_2^2 \tag{8}$$

Let x^* be the minimizer of equation (8), then we can obtain the minimizer of the right hand side of equation (7) by defining

$$y^*(j) = \min\{x^*(j), 0\}, \forall j \in V. \tag{9}$$

Further define:

$$\begin{aligned} A_- &= \{j \in V : x^*(j) < 0 \\ A_0 &= \{j \in V : x^*(j) \leq 0. \end{aligned} \tag{10}$$

Then A_0 and A_- are the unique maximal and minimal minimizers of the left hand side of equation (7).

The Minimum Norm Point Algorithm (cont)...

- From the above theorem we see that SFM is equivalent to finding the minimum norm point in the base polytope.

The Minimum Norm Point Algorithm (cont)...

- From the above theorem we see that SFM is equivalent to finding the minimum norm point in the base polytope.
- Further from the minimizers of the minimum norm problem we can obtain the lattice of minimizers of the submodular program.

The Minimum Norm Point Algorithm (cont)...

- From the above theorem we see that SFM is equivalent to finding the minimum norm point in the base polytope.
- Further from the minimizers of the minimum norm problem we can obtain the lattice of minimizers of the submodular program.
- We can use the minimum norm point algorithm (Wolfe, 1976), to find the minimum norm point on a polytope.

The Minimum Norm Point Algorithm (cont)...

- From the above theorem we see that SFM is equivalent to finding the minimum norm point in the base polytope.
- Further from the minimizers of the minimum norm problem we can obtain the lattice of minimizers of the submodular program.
- We can use the minimum norm point algorithm (Wolfe, 1976), to find the minimum norm point on a polytope.
- This is an exact algorithm and for the base polytope every iteration can be computed efficiently.

The Minimum Norm Point Algorithm (cont)...

- From the above theorem we see that SFM is equivalent to finding the minimum norm point in the base polytope.
- Further from the minimizers of the minimum norm problem we can obtain the lattice of minimizers of the submodular program.
- We can use the minimum norm point algorithm (Wolfe, 1976), to find the minimum norm point on a polytope.
- This is an exact algorithm and for the base polytope every iteration can be computed efficiently.
- Though this algorithm is known to converge in a finite number of iterations, its convergence rate is still an open question.

Sub-gradient Descent (Bach, 13)

- Recall that the Lovász extention of a submodular function is a convex function but is non-smooth.

Sub-gradient Descent (Bach, 13)

- Recall that the Lovász extention of a submodular function is a convex function but is non-smooth.
- However the sub-gradient can directly be evaluated. In particular at a point $w \in [0,1]^n$, the sub-gradient h is exactly the maximizer $h = \text{argmax}_{s \in \mathcal{P}_f} s^\top w$.

Sub-gradient Descent (Bach, 13)

- Recall that the Lovász extention of a submodular function is a convex function but is non-smooth.
- However the sub-gradient can directly be evaluated. In particular at a point $w \in [0,1]^n$, the sub-gradient h is exactly the maximizer $h = \text{argmax}_{s \in \mathcal{P}_f} s^\top w$.
- Furthermore, the sub-gradients h satisfies $\forall k, f(V) - f(V \backslash k) \leq h_k \leq f(k)$. This directly follows from the submodularity of f.

Sub-gradient Descent (Bach, 13)

- Recall that the Lovász extention of a submodular function is a convex function but is non-smooth.
- However the sub-gradient can directly be evaluated. In particular at a point $w \in [0,1]^n$, the sub-gradient h is exactly the maximizer $h = \text{argmax}_{s \in \mathcal{P}_f} s^\top w$.
- Furthermore, the sub-gradients h satisfies $\forall k, f(V) - f(V \backslash k) \leq h_k \leq f(k)$. This directly follows from the submodularity of f.
- It is easy to show that $\hat{f}$ is Lipschitz continuous with constant $L = \sum_{k \in V} \alpha_k^2$ where $\alpha_k = f(k) - f(V) + f(V \backslash k)$.

Sub-gradient Descent Analysis (Bach, 2013)

- The hypercube $[0, 1]^n$ is contained in an l_2 ball of radius $D = \sqrt{n}/2$.

Sub-gradient Descent Analysis (Bach, 2013)

- The hypercube $[0,1]^n$ is contained in an l_2 ball of radius $D = \sqrt{n}/2$.
- Correspondingly we can use projected sub-gradient descent, to minimize $\hat{f}(w)$ on the boolean hypercube and use a step size $\gamma_t = \frac{D\sqrt{2}}{\sqrt{nt}}$.

Sub-gradient Descent Analysis (Bach, 2013)

- The hypercube $[0,1]^n$ is contained in an l_2 ball of radius $D = \sqrt{n}/2$.
- Correspondingly we can use projected sub-gradient descent, to minimize $\hat{f}(w)$ on the boolean hypercube and use a step size $\gamma_t = \frac{D\sqrt{2}}{\sqrt{nt}}$.
- The algorithm basically takes the sub-gradient step and projects the point to $[0,1]^n$. Further the projection is simple and can be done by component wise rounding.

Sub-gradient Descent Analysis (Bach, 2013)

- The hypercube $[0,1]^n$ is contained in an l_2 ball of radius $D = \sqrt{n}/2$.
- Correspondingly we can use projected sub-gradient descent, to minimize $\hat{f}(w)$ on the boolean hypercube and use a step size $\gamma_t = \frac{D\sqrt{2}}{\sqrt{nt}}$.
- The algorithm basically takes the sub-gradient step and projects the point to $[0,1]^n$. Further the projection is simple and can be done by component wise rounding.

Theorem 3

After t steps of projected subgradient descent, among the p sup-level sets of w_t, there is a set B such that $F(B) - \min F(A) \leq \frac{D\sqrt{n}}{\sqrt{2t}}$.

Conditional gradient descent (Bach, 2013)

- Note that the minimum norm point problem is equivalent to the proximal problem using an l_2 regularizer.

Conditional gradient descent (Bach, 2013)

- Note that the minimum norm point problem is equivalent to the proximal problem using an l_2 regularizer.

Lemma 4

For a submodular function f,

$$\min_{y \in \mathcal{B}_f} \|y\| = \min_{y \in \mathbb{R}^n} \|y\| + \hat{f}(y) \quad (11)$$

Conditional gradient descent (Bach, 2013)

- Recall that the minimum norm point problem is equivalent to SFM in that from the solution of the minimum norm problem, we can obtain the minimizers of SFM.

Conditional gradient descent (Bach, 2013)

- Recall that the minimum norm point problem is equivalent to SFM in that from the solution of the minimum norm problem, we can obtain the minimizers of SFM.
- However from the solution of the minimum norm point problem we can obtain the solutions of a much larger family of combinatorial problems.

Conditional gradient descent (Bach, 2013)

- Recall that the minimum norm point problem is equivalent to SFM in that from the solution of the minimum norm problem, we can obtain the minimizers of SFM.
- However from the solution of the minimum norm point problem we can obtain the solutions of a much larger family of combinatorial problems.
- We can solve the minimum norm problem using the conditional gradient algorithm.

Conditional gradient descent (Bach, 2013)

- Recall that the minimum norm point problem is equivalent to SFM in that from the solution of the minimum norm problem, we can obtain the minimizers of SFM.
- However from the solution of the minimum norm point problem we can obtain the solutions of a much larger family of combinatorial problems.
- We can solve the minimum norm problem using the conditional gradient algorithm.
- Start with any point w_0 in the base and iterate-

Conditional gradient descent (Bach, 2013)

- Recall that the minimum norm point problem is equivalent to SFM in that from the solution of the minimum norm problem, we can obtain the minimizers of SFM.
- However from the solution of the minimum norm point problem we can obtain the solutions of a much larger family of combinatorial problems.
- We can solve the minimum norm problem using the conditional gradient algorithm.
- Start with any point w_0 in the base and iterate-
- To find the next iterate, we minimize the linear lower bound at w_{t-1} on the base polytope, which is equivalent to finding $w_t = \text{argmin}_{s \in \mathcal{B}_f} \langle s, w_{t-1} \rangle$ which can easily be performed through the greedy algorithm

Conditional gradient descent (Bach, 2013)

- Recall that the minimum norm point problem is equivalent to SFM in that from the solution of the minimum norm problem, we can obtain the minimizers of SFM.
- However from the solution of the minimum norm point problem we can obtain the solutions of a much larger family of combinatorial problems.
- We can solve the minimum norm problem using the conditional gradient algorithm.
- Start with any point w_0 in the base and iterate-
- To find the next iterate, we minimize the linear lower bound at w_{t-1} on the base polytope, which is equivalent to finding $w_t = \text{argmin}_{s \in \mathcal{B}_f} \langle s, w_{t-1} \rangle$ which can easily be performed through the greedy algorithm
- We perform line search with respect to a weight β. In other words define $w(\beta) = w_t\beta + w_{t-1}(1-\beta)$. Further find $\min_{\beta \in [0,1]} w(\beta)^\top w(\beta)$.

Conditional gradient descent analysis

- Correspondingly we can give the convergence rates of the conditional gradient descent algorithm for submodular function minimization.

Theorem 5

After t steps of the conditional gradient method described above, among the p sub-level sets of w_t, there is a set B such that $F(B) - minF(A) \leq \frac{1}{\sqrt{t}} \sqrt{\sum_{k=1}^{p} \alpha_k^2}$.

- The conditional gradient descent itself has an error proportional to $\frac{1}{t+1}$. However due to the rounding an additional factor is added, and the error rates of the submodular functions are the same order as sub-gradient descent methods.

Smoothing in a special case of decomposable functions

- A smoothing based method was proposed by (Stobbe and Krause, 2010) for a class of functions known as decomposable functions.
- The class of decomposable functions are submodular functions which can be expressed as a sum of concave over modular functions.
- For this class of functions smoothing the Lovász extention followed by the optimal algorithm of Nesterov gives convergence rates of $\frac{1}{\epsilon}$.
- Recall that the convergence rates of the sub-gradient descent and conditional gradient descent have convergence rates of $\frac{1}{\epsilon^2}$.

Outline

Majorization-Minimization (MMin) for $\min_{S \subseteq V} f(S)$

$S_0 = \emptyset$;

for $i = 0, 1, \ldots$ **do**

  compute modular upper bound $\hat{f}_i(S) = m^f_{S_i}(S) \geq f(S)$ based on S_i;

  Set $S_{i+1} = \operatorname{argmin}_{S \in \mathcal{C}} \hat{f}_i(S)$ - modular minimization;

(I-Jegelka-Bilmes (2013a))

Majorization-Minimization (MMin) for $\min_{S \subseteq V} f(S)$

$S_0 = \emptyset$;
for $i = 0, 1, \ldots$ **do**

compute modular upper bound $\hat{f}_i(S) = m^f_{S_i}(S) \geq f(S)$ based on S_i;

Set $S_{i+1} = \text{argmin}_{S \in \mathcal{C}} \hat{f}_i(S)$ - modular minimization;

; `// only need to solve linear-cost problem!` ☺

(I-Jegelka-Bilmes (2013a))

Majorization-Minimization (MMin) for $\min_{S \subseteq V} f(S)$

$S_0 = \emptyset$;
for $i = 0, 1, \ldots$ **do**

compute modular upper bound $\hat{f}_i(S) = m^f_{S_i}(S) \geq f(S)$ based on S_i;

Set $S_{i+1} = \text{argmin}_{S \in \mathcal{C}} \hat{f}_i(S)$ - modular minimization;

; // only need to solve linear-cost problem! ☺

Always improves the objective value!

(I-Jegelka-Bilmes (2013a))

Majorization-Minimization (MMin) for $\min_{S \subseteq V} f(S)$

$S_0 = \emptyset$;
for $i = 0, 1, \ldots$ **do**

compute modular upper bound $\hat{f}_i(S) = m^f_{S_i}(S) \geq f(S)$ based on S_i;

Set $S_{i+1} = \text{argmin}_{S \in \mathcal{C}} \hat{f}_i(S)$ - modular minimization;

; // only need to solve linear-cost problem! ☺

Always improves the objective value!

Highly scalable and practical!

(I-Jegelka-Bilmes (2013a))

Unconstrained Minimization

	MMin-IIIa	MMin-IIIb	MMin-I	MMin-II
g	$\bar{g}$	$\bar{g}$	$\hat{g}$	$\check{g}$
S_0	$\emptyset$	V	$\emptyset$	V
S_c	A	B	A_+	B_+

- MMin-IIIa and IIIb are first iterations of MMin-I and MMin-II.

Unconstrained Minimization

	MMin-IIIa	MMin-IIIb	MMin-I	MMin-II
g	$\bar{g}$	$\bar{g}$	$\hat{g}$	$\check{g}$
S_0	$\emptyset$	V	$\emptyset$	V
S_c	A	B	A_+	B_+

- MMin-IIIa and IIIb are first iterations of MMin-I and MMin-II.
- A and B obtainable in $O(n)$ oracle calls.

Unconstrained Minimization

	MMin-IIIa	MMin-IIIb	MMin-I	MMin-II
g	$\bar{g}$	$\bar{g}$	$\hat{g}$	$\check{g}$
S_0	$\emptyset$	V	$\emptyset$	V
S_c	A	B	A_+	B_+

- MMin-IIIa and IIIb are first iterations of MMin-I and MMin-II.
- A and B obtainable in $O(n)$ oracle calls.
- A_+ and B_+ are local minimizers obtainable in $O(n^2)$ calls.

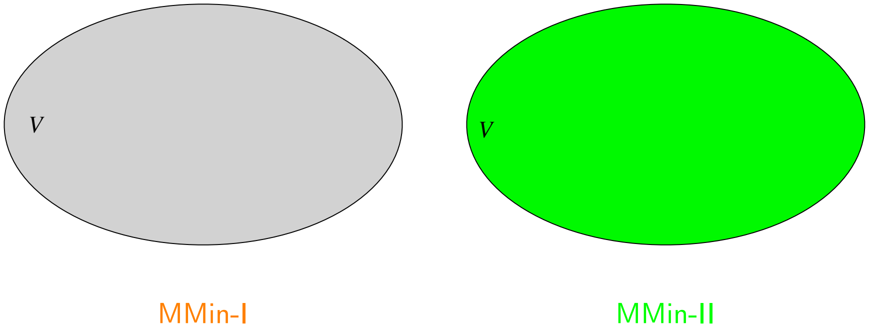
Unconstrained Minimization

	MMin-IIIa	MMin-IIIb	MMin-I	MMin-II
g	$\bar{g}$	$\bar{g}$	$\hat{g}$	$\check{g}$
S_0	$\emptyset$	V	$\emptyset$	V
S_c	A	B	A_+	B_+

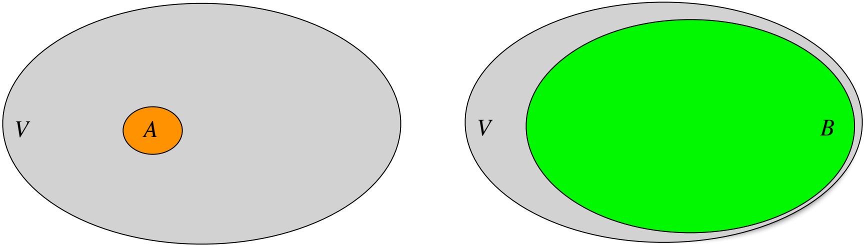
- MMin-IIIa and IIIb are first iterations of MMin-I and MMin-II.
- A and B obtainable in $O(n)$ oracle calls.
- A_+ and B_+ are local minimizers obtainable in $O(n^2)$ calls.

$$A \subseteq A_+ \subseteq X^* \subseteq B_+ \subseteq B$$

Illustrating Unconstrained Minimization



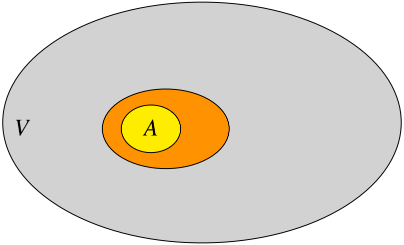
Illustrating Unconstrained Minimization



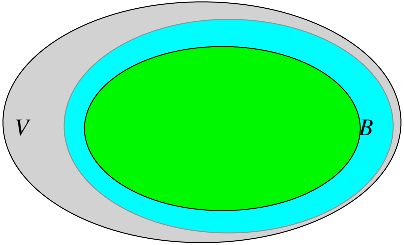
MMin-I

MMin-II

Illustrating Unconstrained Minimization

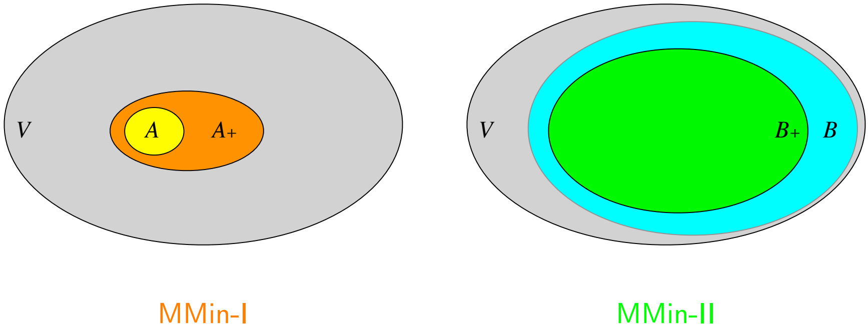


MMin-I

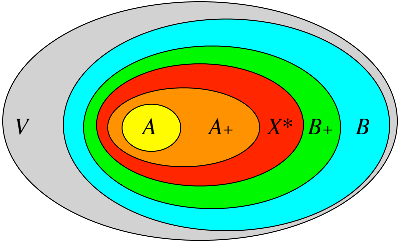


MMin-II

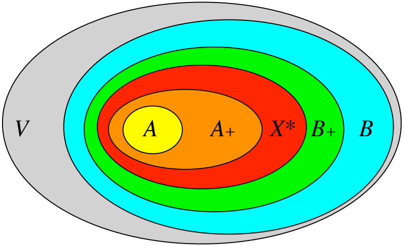
Illustrating Unconstrained Minimization



Illustrating Unconstrained Minimization



MMin-I



MMin-II

Outline

Constrained Submodular Minimization

$$\text{compute } S^* \in \underset{S \in \mathcal{C}}{\operatorname{argmin}} f(S)$$

- Constraints include cardinality constraint,

$$\text{Card: } \mathcal{C} = \{S \subseteq V : |S| \geq k\}$$

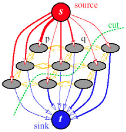
Constrained Submodular Minimization

$$\text{compute } S^* \in \operatorname*{argmin}_{S \in \mathcal{C}} f(S)$$

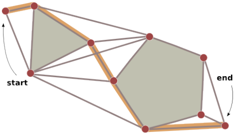
- Constraints include cardinality constraint,

$$\text{Card: } \mathcal{C} = \{S \subseteq V : |S| \geq k\}$$

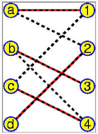
or combinatorial...



cuts



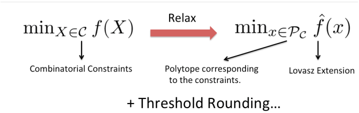
paths



matchings

Outline

Relaxation based Algorithm (I-Jegelka-Bilmes, 2014)



- Define a class of constraints: $\mathcal{C} = \{X \mid |X \cap W| \geq b_W, \text{ for all } W \in \mathcal{W}\}$.
- A large class of constraints including matroid spans, covers, paths, matchings and cuts.
- Resultant polytope: $\hat{\mathcal{P}}_\mathcal{C} = \left\{ x \in [0,1]^n \;\middle|\; \sum_{i \in W} x_i \geq b_W \text{ for all } W \in \mathcal{W} \right\}$
- **Algorithm:** Solve a convex optimization problem, using generic convex solvers (e.g ADMM etc.)
- **Rounding:** Round using threshold rounding: $X_\theta = \{i : x(i) \geq \theta\}$.

Relaxation Based Algorithm (I-Jegelka-Bilmes, 2014)

Theorem: The θ-rounding scheme for constraints $\mathcal{C} = \{X \mid |X \cap W| \geq b_W, \text{ for all } W \in \mathcal{W}\}$ achieves a worst case approximation bound of $\max_{W \in \mathcal{W}} |W| - b_W + 1$.

- Unifies a number of results for matroid spans, covers cuts, matchings etc.

	Matroid Constraints		Set Covers		Paths, Cuts and Matchings		
	Cardinality	Trees	Vertex Covers	Edge Covers	Cuts	Paths	Matchings
CR.	$n-k+1$	$m-n+1$	2	$deg(G) \leq n$	$P_{max} \leq n$	$C_{max} \leq m$	$O(n)$
IG	$\Omega(n-k+1)$	$\Omega(m-n+1)$	2	$\Omega(n)$	$\Omega(n)$	$\Omega(m)$	$\Omega(n)$
Hard	$\Omega(\sqrt{n})$	$\Omega(n)$	$2-\epsilon$	$\Omega(n)$	$\Omega(\sqrt{m})$	$\Omega(n^{2/3})$	$\Omega(n)$

Table: Comparison of the results of the Continuous Relaxations (CR), the hardness, and the integrality gaps (IG) of the corresponding constrained submodular minimization problems.

Outline

Majorization-Minimization, MMin (I-Jegelka-Bilmes, 2013)

$S_0 = \emptyset$;

for $i = 0, 1, \ldots$ **do**

compute modular upper bound $\hat{f}_i(S) = m^f_{S_i}(S) \geq f(S)$ based on S_i;

Set $S_{i+1} = \text{argmin}_{S \in \mathcal{C}} \hat{f}_i(S)$ - find best cut/path/matching. . . ;

(I-Jegelka-Bilmes (2013a))

Majorization-Minimization, MMin (I-Jegelka-Bilmes, 2013)

$S_0 = \emptyset$;
for $i = 0, 1, \ldots$ **do**

compute modular upper bound $\hat{f}_i(S) = m^f_{S_i}(S) \geq f(S)$ based on S_i;

Set $S_{i+1} = \text{argmin}_{S \in \mathcal{C}} \hat{f}_i(S)$ - find best cut/path/matching...;

; `// only need to solve linear-cost problem!` ☺

(I-Jegelka-Bilmes (2013a))

Majorization-Minimization, MMin (I-Jegelka-Bilmes, 2013)

$S_0 = \emptyset$;
for $i = 0, 1, \ldots$ **do**
compute modular upper bound $\hat{f}_i(S) = m^f_{S_i}(S) \geq f(S)$ based on S_i;
Set $S_{i+1} = \text{argmin}_{S \in \mathcal{C}} \hat{f}_i(S)$ - find best cut/path/matching. . . ;
; `// only need to solve linear-cost problem!` ☺

Always improves the objective value!

(I-Jegelka-Bilmes (2013a))

Majorization-Minimization, MMin (I-Jegelka-Bilmes, 2013)

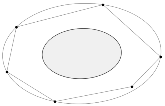
$S_0 = \emptyset$;
for $i = 0, 1, \ldots$ **do**
  compute modular upper bound $\hat{f}_i(S) = m^f_{S_i}(S) \geq f(S)$ based on S_i;
  Set $S_{i+1} = \text{argmin}_{S \in \mathcal{C}} \hat{f}_i(S)$ - find best cut/path/matching. . . ;
  ; // only need to solve linear-cost problem! ☺

Always improves the objective value!

Highly scalable and practical!

(I-Jegelka-Bilmes (2013a))

Ellipsoidal Approximation (Goemans et al, 2009)



- Ellipsoidal Approximation gives the tightest approximation to a submodular function.
- Based on a neat idea, that one can express a submodular function as a linear program over the submodular polyhedron

$$f(X) = \max_{x \in \mathcal{P}_f} x(X) \tag{12}$$

- The idea is then to approximate $\mathcal{P}_f$ by an inner and outer John's ellipsoid.
- This construction gives a $O(\sqrt{n} \log n)$ approximation.
- This approximation $\hat{f}$ is of the form, $\hat{f}(X) = \sqrt{w_f(X)}$, where w_f is a modular function constructed using f.

Theoretical Results

$$\text{minimize} \;\; F(S) : S \in \mathcal{C} = \text{cut/path/matching/cardinality constraint...}$$

For graph based problems, m = number of edges, n = number of vertices.

(Goel et al (09), Goemans et al (2009), Jegelka-Bilmes (11) ...)

Theoretical Results

minimize $F(S) : S \in \mathcal{C} =$ cut/path/matching/cardinality constraint...

For graph based problems, $m =$ number of edges, $n =$ number of vertices.

(Goel et al (09), Goemans et al (2009), Jegelka-Bilmes (11) ...)

Theoretical Results

minimize $F(S) : S \in \mathcal{C} =$ cut/path/matching/cardinality constraint...

For graph based problems, $m =$ number of edges, $n =$ number of vertices.

How good are these algorithms? $f(S) \leq \alpha f(S^*)$

(Goel et al (09), Goemans et al (2009), Jegelka-Bilmes (11) ...)

Theoretical Results

minimize $F(S) : S \in \mathcal{C} =$ cut/path/matching/cardinality constraint...

For graph based problems, $m =$ number of edges, $n =$ number of vertices.

How good are these algorithms? $f(S) \leq \alpha f(S^*)$

Constraint:	**MMin**	**EA**	**Lower bound**
trees/matchings	n	$O(\sqrt{m})$	$\Omega(n)$
cuts	m	$O(\sqrt{m})$	$\Omega(\sqrt{m})$
paths	n	$O(\sqrt{m})$	$\Omega(n^{2/3})$
cardinality(k)	k	$O(\sqrt{n})$	$\Omega(\sqrt{n})$

(Goel et al (09), Goemans et al (2009), Jegelka-Bilmes (11) ...)

Theoretical Results

minimize $F(S) : S \in \mathcal{C} =$ cut/path/matching/cardinality constraint...

For graph based problems, $m =$ number of edges, $n =$ number of vertices.

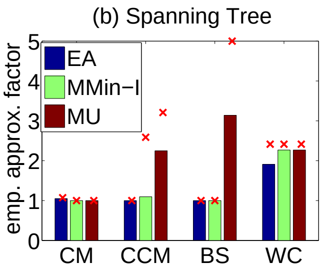
How good are these algorithms? $f(S) \leq \alpha f(S^*)$

Constraint:	MMin	EA	Lower bound
trees/matchings	n	$O(\sqrt{m})$	$\Omega(n)$
cuts	m	$O(\sqrt{m})$	$\Omega(\sqrt{m})$
paths	n	$O(\sqrt{m})$	$\Omega(n^{2/3})$
cardinality(k)	k	$O(\sqrt{n})$	$\Omega(\sqrt{n})$

Worst case polynomial upper/lower bounds ☹

(Goel et al (09), Goemans et al (2009), Jegelka-Bilmes (11) ...)

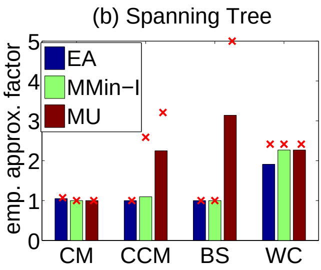
Empirical Results



(MU = Mod. Upper bound Heuristic, and the first iteration of MMin)

(I-Jegelka-Bilmes (2013a))

Empirical Results



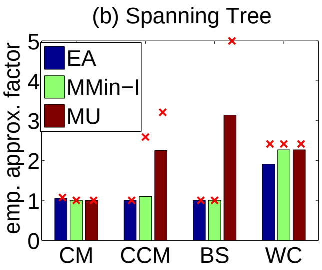
(MU = Mod. Upper bound Heuristic, and the first iteration of MMin)

Observations:

- Empirical Results always better than worst case bounds
- MMin performs comparably to the more complicated EA!

(I-Jegelka-Bilmes (2013a))

Empirical Results



(MU = Mod. Upper bound Heuristic, and the first iteration of MMin)

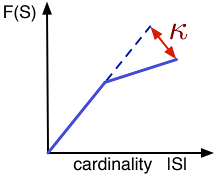
Observations:

- Empirical Results always better than worst case bounds
- MMin performs comparably to the more complicated EA!

Can we say more?

(I-Jegelka-Bilmes (2013a))

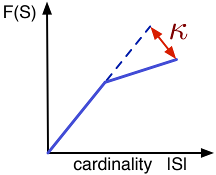
Theory meets Practice (Curvature)



$$\kappa_f = 1 - \min_j \frac{f(j|V\backslash j)}{f(j)}$$

- tighter analysis for submodular max *(Vondrák 08)*

Theory meets Practice (Curvature)



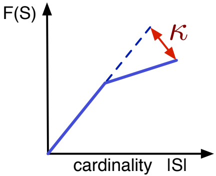
$$\kappa_f = 1 - \min_j \frac{f(j|V\backslash j)}{f(j)}$$

- tighter analysis for submodular max *(Vondrák 08)*

Lemma (I-Jegelka-Bilmes'13) Tightened bounds for submodular min:

Upper: α_n Lower: β_n.

Theory meets Practice (Curvature)



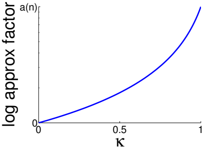
$$\kappa_f = 1 - \min_j \frac{f(j|V\setminus j)}{f(j)}$$

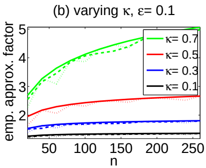
- tighter analysis for submodular max *(Vondrák 08)*

Lemma (I-Jegelka-Bilmes'13) Tightened bounds for submodular min:

$$\text{Upper: } \frac{\alpha_n}{1+(\alpha_n-1)(1-\kappa_f)} \quad \text{Lower: } \frac{\beta_n}{1+(\beta_n-1)(1-\kappa_f)}.$$

For large n, both EA and MMin are $O(1/(1-\kappa))$.





Improved Curvature Based Bounds

minimize $F(S) : S \in \mathcal{C} =$ cut/path/matching/cardinality constraint...

For graph based problems, $m =$ number of edges, $n =$ number of vertices.

How good are these algorithms? $f(S) \leq \alpha f(S^*)$

Constraint:	**MMin**	**EA**	**Lower bound**
trees/matchings	$\frac{n}{1+(n-1)(1-\kappa_f)}$	$O(\frac{\sqrt{m}}{1+\sqrt{m}-1)(1-\kappa_f)})$	$\Omega(\frac{n}{1+(n-1)(1-\kappa_f)})$
cuts	$\frac{m}{1+(m-1)(1-\kappa_f)}$	$O(\frac{\sqrt{m}}{1+\sqrt{m}-1)(1-\kappa_f)})$	$\Omega(\frac{\sqrt{m}}{1+\sqrt{m}-1)(1-\kappa_f)})$
paths	$\frac{n}{1+(n-1)(1-\kappa_f)}$	$O(\frac{\sqrt{m}}{1+\sqrt{m}-1)(1-\kappa_f)})$	$\Omega(\frac{n^{2/3}}{1+(n^{2/3}-1)(1-\kappa_f)})$
cardinality(k)	$\frac{k}{1+(k-1)(1-\kappa_f)}$	$O(\frac{\sqrt{n}}{1+\sqrt{n}-1)(1-\kappa_f)})$	$\Omega(\frac{\sqrt{n}}{1+\sqrt{n}-1)(1-\kappa_f)})$

Worst case upper/lower bounds bounded by $O(\frac{1}{(1-\kappa_f)})$ ☺

((I-Jegelka-Bilmes, 2013))

Comparison of the Relaxation based techniques, and the Combinatorial Algorithms

Unlike the Ellipsoidal Approximation based algorithm, and MMin, the Continuous Relaxation based algorithms do not seem to admit curvature dependent approximation guarantees.

	Matroid Constraints		Set Covers		Paths, Cuts and Matchings		
	Cardinality	Trees	Vertex Covers	Edge Covers	Cuts	Paths	Matchings
CR.	$n-k+1$	$m-n+1$	2	$deg(G) \leq n$	$P_{max} \leq n$	$C_{max} \leq m$	$O(n)$
MMin	k	n	$\|VC\| \leq n$	$\|EC\| \leq n$	$C_{max} \leq m$	$P_{max} \leq n$	$O(n)$
EA	$\sqrt{n}$	$\sqrt{m}$	$\sqrt{n}$	$\sqrt{m}$	$\sqrt{m}$	$\sqrt{m}$	$\sqrt{m}$
IG	$\Omega(n-k+1)$	$\Omega(m-n+1)$	2	$\Omega(n)$	$\Omega(n)$	$\Omega(m)$	$\Omega(n)$
Hard	$\Omega(\sqrt{n})$	$\Omega(n)$	$2-\epsilon$	$\Omega(n)$	$\Omega(\sqrt{m})$	$\Omega(n^{2/3})$	$\Omega(n)$

Table: Comparison of the results of the Continuous Relaxation, with the semigradient framework (MMin), the Ellipsoidal Approximation (EA) algorithm, hardness, and the integrality gaps (IG) of the corresponding constrained submodular minimization problems.

Outline

Submodular Function Maximization

$$\text{compute } S^* \in \operatorname*{argmax}_{S \in \mathcal{C}} g(S)$$

- Unconstrained submodular maximization, $\mathcal{C} = 2^V$.

Submodular Function Maximization

$$\text{compute } S^* \in \operatorname*{argmax}_{S \in \mathcal{C}} g(S)$$

- Unconstrained submodular maximization, $\mathcal{C} = 2^V$.
- Other constraints include cardinality or knapsack constraints,

$$\text{Cardinality: } \mathcal{C} = \{S \subseteq V : |S| \leq k\}, \text{ Knapsack: } \mathcal{C} = \{S \subseteq V : w(S) \leq b\}$$

Submodular Function Maximization

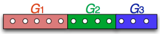
$$\text{compute } S^* \in \operatorname*{argmax}_{S \in \mathcal{C}} g(S)$$

- Unconstrained submodular maximization, $\mathcal{C} = 2^V$.
- Other constraints include cardinality or knapsack constraints,

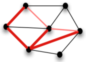
$$\text{Cardinality: } \mathcal{C} = \{S \subseteq V : |S| \leq k\}, \text{ Knapsack: } \mathcal{C} = \{S \subseteq V : w(S) \leq b\}$$

or matroid constraints...

partition

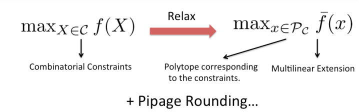


graphic



Outline

Multilinear Extension and Rounding



- Use the Multilinear extension:

$$\bar{f}(x) = \sum_{X \subseteq V} f(X) \prod_{i \in X} x_i \prod_{i \notin X} (1 - x_i), \tag{13}$$

- **Algorithm:** Solve a continuous optimization problem, using continuous greedy algorithms (akin to the conditional gradient algorithm).
- **Rounding:** Round using a pipage rounding scheme (Vondrak-2007).
- These techniques work for both constrained and unconstrained maximization.

Multilinear Extension and Rounding

- The main challenge in using these algorithms in real world problems is the complexity of evaluating the multilinear extension.
- Requires repeated sampling of the submodular function!
- However, the multilinear extension can be efficiently computed for several subclasses of submodular functions, including Facility Location, Set Covers, Log-Determinants etc.

Outline

Bidirectional Greedy for Unconstrained Maximization (Buchbinder, 2012)

Algorithm 1: Bidirectional Greedy Algorithm

Start with $A_0 = \emptyset, B_0 = V$, and an initial ordering of $V : \tau = \{\tau_1, \tau_2, \cdots, \tau_n\}$

for $i = 1$ to n **do**

  $a_i \leftarrow f(A_{i-1} \cup \tau_i) - f(A_{i-1})$

  $b_i \leftarrow f(B_{i-1} \backslash \tau_i) - f(B_{i-1})$

  if $a_i \geq b_i$ **then**

    $A_i = A_{i-1} \cup \tau_i, B_i = B_{i-1}$

  else

    $A_i = A_{i-1}, B_i = B_{i-1} \backslash \tau_i$

return A_n (or B_n)

- This is a deterministic algorithm, and provides a $1/3$ approximation for unconstrained submodular maximization.

Local Search for Unconstrained Maximization (Vondrak, 2007)

Algorithm 2: Deterministic Local Search Algorithm

Start with $Y_0 = \emptyset, n \leftarrow 0$

repeat

  $Y_1 = \text{argmax}_{v \in V} f(v)$;

  while $f(Y_{n+1}) \geq (1+\eta) f(Y_n)$ **do**

    $y = \text{argmax}_{v \in V \setminus Y_n} f(v|Y_n),\ Y_{n+1} = Y_n \cup y$;

    $n \leftarrow n+1$;

  while $f(Y_{n+1}) \geq (1+\eta) f(Y_n)$ **do**

    $y = \text{argmax}_{v \in Y_n} f(v|Y_n \setminus v),\ Y_{n+1} = Y_n \setminus y$;

    $n \leftarrow n+1$;

until convergence $(Y_n = Y_{n-1})$;

return the better amongst Y_n and $V \setminus Y_n$.

- Again, a deterministic algorithm, and a 1/3 approximation.

Randomized Bidirectional Greedy for Unconstrained Submodular Maximization (Buchbinder, 2012)

Algorithm 3: Randomized Bidirectional Greedy Algorithm

Start with $A_0 = \emptyset, B_0 = V$, and an initial ordering of $V : \tau = \{\tau_1, \tau_2, \cdots, \tau_n\}$

for $i = 1$ to n **do**

$a_i \leftarrow f(A_{i-1} \cup \tau_i) - f(A_{i-1})$

$b_i \leftarrow f(B_{i-1} \backslash \tau_i) - f(B_{i-1})$

$a'_i \leftarrow \max\{a_i, 0\}, b'_i \leftarrow \max\{b_i, 0\}$

with probability $\frac{a'_i}{a'_i + b'_i}$: $A_i = A_{i-1} \cup \tau_i, B_i = B_{i-1}$

else with probability $\frac{b'_i}{a'_i + b'_i}$: $A_i = A_{i-1}, B_i = B_{i-1} \backslash \tau_i$

return A_n (or B_n)

- A randomized algorithm, and a $1/2$ approximation in expectation.

Greedy Algorithm for Constrained Submodular Maximization (Nemhauser, 1978)

Algorithm 4: Greedy Algorithm for $\max_{X \in \mathcal{C}} f(X)$

Start with $Y_0 = \emptyset, n \leftarrow 0$

repeat

$y = \text{argmax}_{v \in V \setminus Y_n} f(v|Y_n)$;

$Y_{n+1} = Y_n \cup y$;

$n \leftarrow n + 1$;

until $\underline{Y_n \notin \mathcal{C}}$;

return Y_{n-1}.

- Under cardinality constraints, this is a $1 - 1/e$ approximation for monotone submodular functions.
- Variants of this also extend to other constraints like knapsack and Matroid constraints.

Outline

Minorization-Maximization Subgradient Ascent

$S_0 = \emptyset$;
for $i = 0, 1, \ldots$ **do**
  compute modular lower bound $\check{g}_i = h^g_{S_i} \leq g$ based on S_i;
  Set $S_{i+1} = \operatorname{argmax}_{S \in \mathcal{C}} \check{g}_i(S)$;

(I-Jegelka-Bilmes (2013a))

Minorization-Maximization Subgradient Ascent

$S_0 = \emptyset$;
for $i = 0, 1, \ldots$ **do**
  compute modular lower bound $\breve{g}_i = h^g_{S_i} \leq g$ based on S_i;
  Set $S_{i+1} = \text{argmax}_{S \in \mathcal{C}} \breve{g}_i(S)$;
  ; // only need to solve linear-cost problem! ☺

(I-Jegelka-Bilmes (2013a))

Minorization-Maximization Subgradient Ascent

$S_0 = \emptyset$;
for $i = 0, 1, \ldots$ **do**
  compute modular lower bound $\check{g}_i = h^g_{S_i} \leq g$ based on S_i;
  Set $S_{i+1} = \text{argmax}_{S \in \mathcal{C}} \check{g}_i(S)$;
  ; `// only need to solve linear-cost problem!` ☺

Always improve the objective value at every iteration!

(I-Jegelka-Bilmes (2013a))

Minorization-Maximization Subgradient Ascent

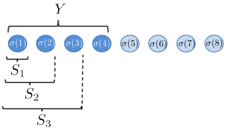
$S_0 = \emptyset$;
for $i = 0, 1, \ldots$ **do**
  compute modular lower bound $\check{g}_i = h^g_{S_i} \leq g$ based on S_i;
  Set $S_{i+1} = \text{argmax}_{S \in \mathcal{C}} \check{g}_i(S)$;
  ; // only need to solve linear-cost problem! ☺

Always improve the objective value at every iteration!

A number of maximization algorithms can be unified with this framework!

(I-Jegelka-Bilmes (2013a))

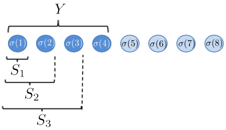
Unconstrained Maximization (I-Jegelka-Bilmes, 2013)



different subgradients ... yield known algorithms ☺

(I-Jegelka-Bilmes (2013))

Unconstrained Maximization (I-Jegelka-Bilmes, 2013)

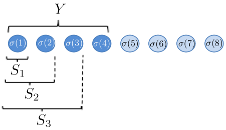


different subgradients ... yield known algorithms ☺

- Random Subgradient ⇒ 1/4 Approx!

(I-Jegelka-Bilmes (2013))

Unconstrained Maximization (I-Jegelka-Bilmes, 2013)

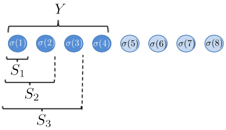


different subgradients ... yield known algorithms ☺

- Random Subgradient ⇒ 1/4 Approx!
- Randomized / Deterministic local search based subgradient ⇒ 1/3 Approx (FMV'07)!

(I-Jegelka-Bilmes (2013))

Unconstrained Maximization (I-Jegelka-Bilmes, 2013)

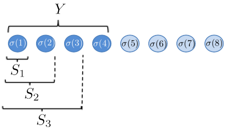


different subgradients ... yield known algorithms ☺

- Random Subgradient ⇒ 1/4 Approx!
- Randomized / Deterministic local search based subgradient ⇒ 1/3 Approx (FMV'07)!
- Bi-directional Greedy subgradient ⇒ 1/3 Approx (BFNS'12)!

(I-Jegelka-Bilmes (2013))

Unconstrained Maximization (I-Jegelka-Bilmes, 2013)

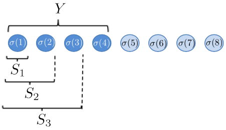


different subgradients ... yield known algorithms ☺

- Random Subgradient ⇒ 1/4 Approx!
- Randomized / Deterministic local search based subgradient ⇒ 1/3 Approx (FMV'07)!
- Bi-directional Greedy subgradient ⇒ 1/3 Approx (BFNS'12)!
- Randomized Greedy subgradient ⇒ 1/2 Approx! (BFNS'12)!

(I-Jegelka-Bilmes (2013))

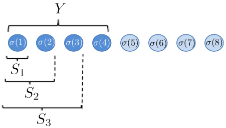
Constrained Maximization ($\max\{f(X) : |X| \leq k\}$)



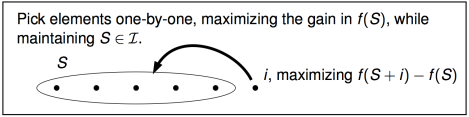
- Random subgradient $\Rightarrow$ k/n Approx (Filmus'13)!

(Nemhauser et al (78), Minoux (82), I-Jegelka-Bilmes (2013), Wei-I-Bilmes (14))

Constrained Maximization ($\max\{f(X) : |X| \leq k\}$)



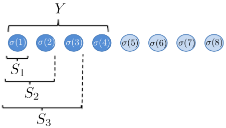
- Random subgradient $\Rightarrow$ k/n Approx (Filmus'13)!



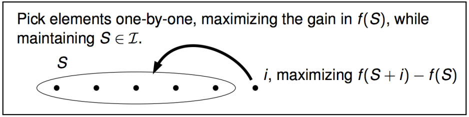
- Greedy and variants: Pick next, what looks best

(Nemhauser et al (78), Minoux (82), I-Jegelka-Bilmes (2013), Wei-I-Bilmes (14))

Constrained Maximization ($\max\{f(X) : |X| \leq k\}$)



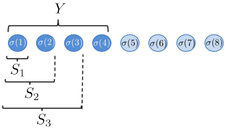
- Random subgradient $\Rightarrow$ k/n Approx (Filmus'13)!



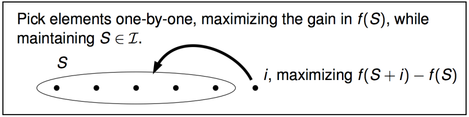
- Greedy and variants: Pick next, what looks best
  - Monotone submodular functions - $\Rightarrow$ $1 - 1/e$ Approx. (NWF'78)!

(Nemhauser et al (78), Minoux (82), I-Jegelka-Bilmes (2013), Wei-I-Bilmes (14))

Constrained Maximization ($\max\{f(X) : |X| \leq k\}$)



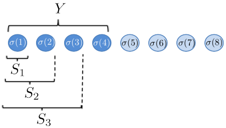
- Random subgradient $\Rightarrow$ k/n Approx (Filmus'13)!



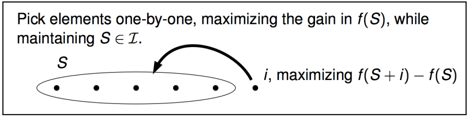
- Greedy and variants: Pick next, what looks best
  - Monotone submodular functions - $\Rightarrow$ $1 - 1/e$ Approx. (NWF'78)!
  - Randomized Greedy (non-monotone) $\Rightarrow$ $1/e$ Approx. (BFNS'14)!
  - Simple variants extend to Matroid, knapsack constraints.

(Nemhauser et al (78), Minoux (82), I-Jegelka-Bilmes (2013), Wei-I-Bilmes (14))

Constrained Maximization ($\max\{f(X) : |X| \leq k\}$)



- Random subgradient ⇒ k/n Approx (Filmus'13)!



- Greedy and variants: Pick next, what looks best
  - Monotone submodular functions - ⇒ $1 - 1/e$ Approx. (NWF'78)!
  - Randomized Greedy (non-monotone) ⇒ $1/e$ Approx. (BFNS'14)!
  - Simple variants extend to Matroid, knapsack constraints.
  - Possible to scale greedy to massive datasets through acceleration/ approximations!

(Nemhauser et al (78), Minoux (82), I-Jegelka-Bilmes (2013), Wei-I-Bilmes (14))

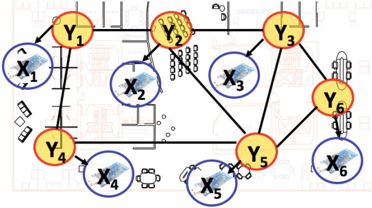
Outline

Difference of submodular functions

Co-operative Costs → $f(X)$ Coverage/ Diversity → $g(X)$

$$\min_{X \subseteq V} f(X) - \lambda g(X)$$

E.g:



Feature Subset selection

Difference of submodular functions

Co-operative Costs → Coverage/ Diversity

$$\min_{X \subseteq V} f(X) - \lambda g(X)$$

Unfortunately this NP hard and inapproximable ☹

Theorem (I-Bilmes, 2012, 2015) DS minimization is NP hard to approximate, and DS maximization is information theoretic hard to approximate upto any poly-factor.

Resort to heuristics: Majorize-Minimize style algorithms!

(Narasimhan-Bilmes (2005), I-Bilmes (2012))

Majorization-Minimization semigradient alg. for DS opt.

$S_0 = \emptyset$;

for $i = 0, 1, \ldots$ **do**

compute modular lower bound of g: $\breve{g}_i = h^g_{S_i} \leq g$ and modular upper bound of f: $\hat{f}_i = m^f_{S_i} \geq f$ based on S_i;

(Narasimhan-Bilmes (2005), I-Bilmes (2012))

Majorization-Minimization semigradient alg. for DS opt.

$S_0 = \emptyset$;

for $i = 0, 1, \ldots$ **do**

compute modular lower bound of g: $\breve{g}_i = h^g_{S_i} \leq g$ and modular upper bound of f: $\hat{f}_i = m^f_{S_i} \geq f$ based on S_i;

SubSup: $S_{i+1} = \text{argmin}_{S \in \mathcal{C}} f(S) - \breve{g}_i(S)$;

SupSub: $S_{i+1} = \text{argmin}_{S \in \mathcal{C}} \hat{f}_i(S) - g(S)$;

ModMod: $S_{i+1} = \text{argmin}_{S \in \mathcal{C}} \hat{f}_i(S) - \breve{g}_i(S)$;

(Narasimhan-Bilmes (2005), I-Bilmes (2012))

Majorization-Minimization semigradient alg. for DS opt.

$S_0 = \emptyset$;
for $i = 0, 1, \ldots$ **do**

compute modular lower bound of g: $\breve{g}_i = h^g_{S_i} \leq g$ and modular upper bound of f: $\hat{f}_i = m^f_{S_i} \geq f$ based on S_i;

SubSup: $S_{i+1} = \text{argmin}_{S \in \mathcal{C}} f(S) - \breve{g}_i(S)$;

SupSub: $S_{i+1} = \text{argmin}_{S \in \mathcal{C}} \hat{f}_i(S) - g(S)$;

ModMod: $S_{i+1} = \text{argmin}_{S \in \mathcal{C}} \hat{f}_i(S) - \breve{g}_i(S)$;

; // Every iteration is submodular min, submodular max or modular min! ☺

(Narasimhan-Bilmes (2005), I-Bilmes (2012))

Majorization-Minimization semigradient alg. for DS opt.

$S_0 = \emptyset$;
for $i = 0, 1, \ldots$ **do**

compute modular lower bound of g: $\breve{g}_i = h^g_{S_i} \leq g$ and modular upper bound of f: $\hat{f}_i = m^f_{S_i} \geq f$ based on S_i;

SubSup: $S_{i+1} = \text{argmin}_{S \in \mathcal{C}} f(S) - \breve{g}_i(S)$;

SupSub: $S_{i+1} = \text{argmin}_{S \in \mathcal{C}} \hat{f}_i(S) - g(S)$;

ModMod: $S_{i+1} = \text{argmin}_{S \in \mathcal{C}} \hat{f}_i(S) - \breve{g}_i(S)$;

; // Every iteration is submodular min, submodular max or modular min! ☺

Improve the objective value at every iteration!

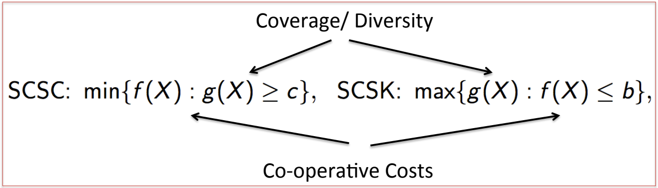
(Narasimhan-Bilmes (2005), I-Bilmes (2012))

Outline

Submodular optimization with submodular cover and submodular knapsack constraints

$$\text{SCSC: } \min\{f(X) : g(X) \geq c\}, \quad \text{SCSK: } \max\{g(X) : f(X) \leq b\},$$

Submodular optimization with submodular cover and submodular knapsack constraints



Submodular optimization with submodular cover and submodular knapsack constraints

Coverage/ Diversity

SCSC: $\min\{f(X) : g(X) \geq c\}$, SCSK: $\max\{g(X) : f(X) \leq b\}$,

Co-operative Costs

Optimize one of the functions, the other one occurs as constraints

More natural in many applications!

Submodular optimization with submodular cover and submodular knapsack constraints

Coverage/ Diversity

$$\text{SCSC: } \min\{f(X) : g(X) \geq c\}, \quad \text{SCSK: } \max\{g(X) : f(X) \leq b\},$$

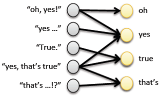
Co-operative Costs

Optimize one of the functions, the other one occurs as constraints

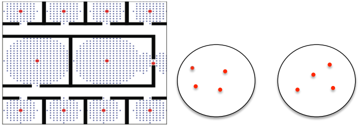
More natural in many applications!

E.g:

1 all_right how are_you doing
2 how are_you with yours
3 hi nadine my name is lorraine how are_you
4 good how are_you
5 hello hi how are_you
6 good thanks how are_you
7 uh how are_you
8 i'm good how are_you
9 fine how are_you



Limited vocabulary and diverse corpus selection



Sensor Placement with Cooperative costs

Special Case-I (Modular f and Submodular g)

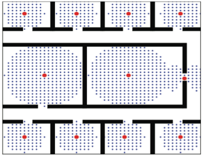
Coverage/ Diversity

$$\text{SSC: } \min\{w(X) : g(X) \geq c\}, \quad \text{SK: } \max\{g(X) : w(X) \leq b\},$$

Additive Costs

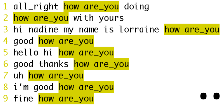
Maximize coverage/ diversity but with additive costs on items

E.g:



Sensor Placement

Summarization



....

Data Subset Selection

Special Case-II: Submodular f and Modular g

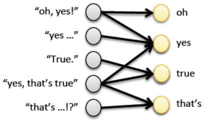
Additive functions

$$\text{SML: } \min\{f(X) : w(X) \geq c\}, \quad \text{SS: } \max\{w(X) : f(X) \leq b\},$$

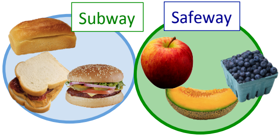
Co-operative Costs

Minimize cooperative costs but with additive cover

E.g:



Limited vocabulary corpus selection



Modeling supermarket purchasing

Connections between SCSC and SCSK

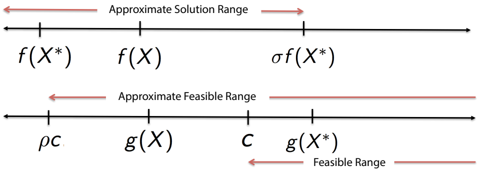
Bi-criterion factors:

Connections between SCSC and SCSK

Bi-criterion factors:

- $\min\{f(X) : g(X) \geq c\}$: $[\sigma, \rho]$ approximation for SCSC is a set $X : f(X) \leq \sigma f(X^*)$ and $g(X) \geq \rho c$.

$[\sigma > 1, \rho < 1]$



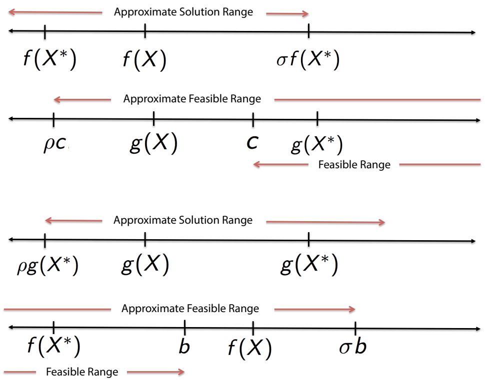
Connections between SCSC and SCSK

Bi-criterion factors:

- $\min\{f(X) : g(X) \geq c\}$: $[\sigma, \rho]$ approximation for SCSC is a set $X : f(X) \leq \sigma f(X^*)$ and $g(X) \geq \rho c$.

- $\max\{g(X) : f(X) \leq b\}$: $[\rho, \sigma]$ approximation for SCSK is a set $X : g(X) \geq \rho g(X^*)$ and $f(X) \leq \sigma b$.

$[\sigma > 1, \rho < 1]$

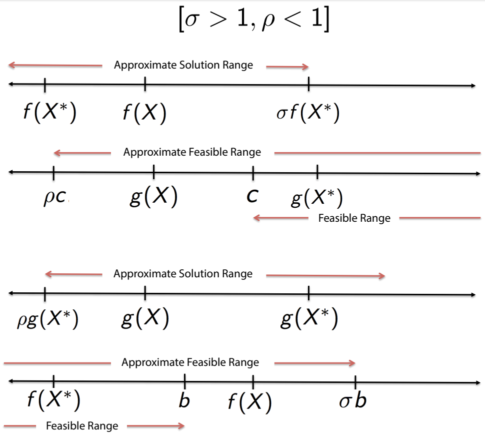


Connections between SCSC and SCSK

Bi-criterion factors:

- $\min\{f(X) : g(X) \geq c\}$: $[\sigma, \rho]$ approximation for SCSC is a set $X : f(X) \leq \sigma f(X^*)$ and $g(X) \geq \rho c$.

- $\max\{g(X) : f(X) \leq b\}$: $[\rho, \sigma]$ approximation for SCSK is a set $X : g(X) \geq \rho g(X^*)$ and $f(X) \leq \sigma b$.



Theorem: Given a $[\sigma, \rho]$ bi-criterion approx. algorithm for SCSC (or SCSK), we can obtain a $[(1+\epsilon)\rho, \sigma]$ bi-criterion approx. algorithm for the other, by running the given algorithm, $O(\log \frac{1}{\epsilon})$ times.

Majorization-Minimization algorithm

$$\text{SCSC: } \min\{f(X) : g(X) \geq c\}, \quad \text{SCSK: } \max\{g(X) : f(X) \leq b\},$$

(I-Bilmes (2013))

Majorization-Minimization algorithm

$$\text{SCSC: } \min\{f(X) : g(X) \geq c\}, \quad \text{SCSK: } \max\{g(X) : f(X) \leq b\},$$

$S_0 = \emptyset$;
for $i = 0, 1, \ldots$ **do**
  compute modular lower bound of g: $\check{g}_i = h^g_{S_i} \leq g$ and modular upper bound of f: $\hat{f}_i = m^f_{S_i} \geq f$ based on S_i;

(I-Bilmes (2013))

Majorization-Minimization algorithm

$$\text{SCSC: } \min\{f(X) : g(X) \geq c\}, \quad \text{SCSK: } \max\{g(X) : f(X) \leq b\},$$

$S_0 = \emptyset$;
for $i = 0, 1, \ldots$ **do**

compute modular lower bound of g: $\breve{g}_i = h^g_{S_i} \leq g$ and modular upper bound of f: $\hat{f}_i = m^f_{S_i} \geq f$ based on S_i;

SCSC: $S_{i+1} = \text{argmin}\{\hat{f}_i(S) : \breve{g}_i(S) \geq c\}$;
SCSK: $S_{i+1} = \text{argmax}\{\breve{g}_i(S) : \hat{f}_i(S) \leq b\}$;

(I-Bilmes (2013))

Majorization-Minimization algorithm

$$\text{SCSC: } \min\{f(X) : g(X) \geq c\}, \quad \text{SCSK: } \max\{g(X) : f(X) \leq b\},$$

$S_0 = \emptyset$;

for $i = 0, 1, \ldots$ **do**

compute modular lower bound of g: $\breve{g}_i = h^g_{S_i} \leq g$ and modular upper bound of f: $\hat{f}_i = m^f_{S_i} \geq f$ based on S_i;

SCSC: $S_{i+1} = \text{argmin}\{\hat{f}_i(S) : \breve{g}_i(S) \geq c\}$;

SCSK: $S_{i+1} = \text{argmax}\{\breve{g}_i(S) : \hat{f}_i(S) \leq b\}$;

; // Every iteration is knapsack problem! ☺

(I-Bilmes (2013))

Majorization-Minimization algorithm

$$\text{SCSC: } \min\{f(X) : g(X) \geq c\}, \quad \text{SCSK: } \max\{g(X) : f(X) \leq b\},$$

$S_0 = \emptyset$;
for $i = 0, 1, \ldots$ **do**
  compute modular lower bound of g: $\breve{g}_i = h^g_{S_i} \leq g$ and modular upper bound of f: $\hat{f}_i = m^f_{S_i} \geq f$ based on S_i;
  SCSC: $S_{i+1} = \text{argmin}\{\hat{f}_i(S) : \breve{g}_i(S) \geq c\}$;
  SCSK: $S_{i+1} = \text{argmax}\{\breve{g}_i(S) : \hat{f}_i(S) \leq b\}$;
  ; // Every iteration is knapsack problem! ☺

Highly scalable and practical!

(I-Bilmes (2013))

Theoretical Results (I-Bilmes (2013))

$$\text{SCSC: } \min\{f(X) : g(X) \geq c\}, \quad \text{SCSK: } \max\{g(X) : f(X) \leq b\},$$

- Submodular Set Cover and Submodular Knapsack – Modular f, submodular g

  Majorize-Minimize $\Rightarrow$ Greedy Algorithm $\Rightarrow$ $1 - 1/e$ Approx!

Theoretical Results (I-Bilmes (2013))

$$\text{SCSC: } \min\{f(X) : g(X) \geq c\}, \quad \text{SCSK: } \max\{g(X) : f(X) \leq b\},$$

- Submodular Set Cover and Submodular Knapsack – Modular f, submodular g

  Majorize-Minimize $\Rightarrow$ Greedy Algorithm $\Rightarrow$ $1 - 1/e$ Approx!

- Submod. Cost Submod. Cover (SCSC) and Submod. Cost Submod. Knapsack (SCSK) – Submodular f, Modular/Submodular g

  Majorize-Minimize (MMin) $\Rightarrow$ $\frac{\sigma}{\rho} = \frac{n}{1+(n-1)(1-\kappa_f)}$ Approx!

Theoretical Results (I-Bilmes (2013))

$$\text{SCSC: } \min\{f(X) : g(X) \geq c\}, \quad \text{SCSK: } \max\{g(X) : f(X) \leq b\},$$

- Submodular Set Cover and Submodular Knapsack – Modular f, submodular g

  Majorize-Minimize $\Rightarrow$ Greedy Algorithm $\Rightarrow$ $1 - 1/e$ Approx!

- Submod. Cost Submod. Cover (SCSC) and Submod. Cost Submod. Knapsack (SCSK) – Submodular f, Modular/Submodular g

  Majorize-Minimize (MMin) $\Rightarrow$ $\frac{\sigma}{\rho} = \frac{n}{1+(n-1)(1-\kappa_f)}$ Approx!

  Ellipsoidal Approx. (EA) $\Rightarrow$ $\frac{\sigma}{\rho} = O(\frac{\sqrt{n}}{1+(\sqrt{n}-1)(1-\kappa_f)})$ Approx!

Hardness (Lower bounds) of the problems

	Modular g	Submodular g	
	$(\kappa_g = 0)$	$(0 < \kappa_g < 1)$	$(\kappa_g = 1)$
Modular f $(\kappa_f = 0)$			
Submod f $(0 < \kappa_f < 1)$			
Submod f $(\kappa_f = 1)$			

Hardness (Lower bounds) of the problems

Knapsack

	Modular g	Submodular g	
	$(\kappa_g = 0)$	$(0 < \kappa_g < 1)$	$(\kappa_g = 1)$
Modular f $(\kappa_f = 0)$	FPTAS		
Submod f $(0 < \kappa_f < 1)$			
Submod f $(\kappa_f = 1)$			

Hardness (Lower bounds) of the problems

Knapsack SSC/SK

	Modular g	Submodular g	
	$(\kappa_g = 0)$	$(0 < \kappa_g < 1)$	$(\kappa_g = 1)$
Modular f $(\kappa_f = 0)$	FPTAS	$\frac{1}{\kappa_g}(1 - e^{-\kappa_g})$	$1 - 1/e$
Submod f $(0 < \kappa_f < 1)$			
Submod f $(\kappa_f = 1)$			

Hardness (Lower bounds) of the problems

Knapsack

SSC/SK

	Modular g	Submodular g	
	$(\kappa_g = 0)$	$(0 < \kappa_g < 1)$	$(\kappa_g = 1)$
Modular f $(\kappa_f = 0)$	FPTAS	$\frac{1}{\kappa_g}(1 - e^{-\kappa_g})$	$1 - 1/e$
Submod f $(0 < \kappa_f < 1)$	$\Omega(\frac{\sqrt{n}}{1+(\sqrt{n}-1)(1-\kappa_f)})$		
Submod f $(\kappa_f = 1)$	$\Omega(\sqrt{n})$		

SML/SS

Hardness (Lower bounds) of the problems

Knapsack

SSC/SK

	Modular g	Submodular g	
	$(\kappa_g = 0)$	$(0 < \kappa_g < 1)$	$(\kappa_g = 1)$
Modular f $(\kappa_f = 0)$	FPTAS	$\frac{1}{\kappa_g}(1 - e^{-\kappa_g})$	$1 - 1/e$
Submod f $(0 < \kappa_f < 1)$	$\Omega(\frac{\sqrt{n}}{1+(\sqrt{n}-1)(1-\kappa_f)})$	$\Omega(\frac{\sqrt{n}}{1+(\sqrt{n}-1)(1-\kappa_f)})$	$\Omega(\frac{\sqrt{n}}{1+(\sqrt{n}-1)(1-\kappa_f)})$
Submod f $(\kappa_f = 1)$	$\Omega(\sqrt{n})$	$\Omega(\sqrt{n})$	$\Omega(\sqrt{n})$

SML/SS

SCSC/SCSK

Hardness (Lower bounds) of the problems

Knapsack

SSC/SK

	Modular g	Submodular g	
	$(\kappa_g = 0)$	$(0 < \kappa_g < 1)$	$(\kappa_g = 1)$
Modular f $(\kappa_f = 0)$	FPTAS	$\frac{1}{\kappa_g}(1 - e^{-\kappa_g})$	$1 - 1/e$
Submod f $(0 < \kappa_f < 1)$	$\Omega(\frac{\sqrt{n}}{1+(\sqrt{n}-1)(1-\kappa_f)})$	$\Omega(\frac{\sqrt{n}}{1+(\sqrt{n}-1)(1-\kappa_f)})$	$\Omega(\frac{\sqrt{n}}{1+(\sqrt{n}-1)(1-\kappa_f)})$
Submod f $(\kappa_f = 1)$	$\Omega(\sqrt{n})$	$\Omega(\sqrt{n})$	$\Omega(\sqrt{n})$

SML/SS

SCSC/SCSK

- Hardness depends (mainly) on κ_f and not (so much) on that of κ_g.

Conclusions/ Future Work

- A review of algorithms for submodular minimization, submodular maximization, DS optimization and submodular optimization subject to submodular constraints.
- Scalable framework of algorithms.
- Theoretical guarantees and hardness results.

Thank You!