Submodular Optimization with Submodular Cover and Submodular Knapsack Constraints (SCSC/SCSK)

Rishabh Iyer    Jeff Bilmes

University of Washington, Seattle

NIPS-2013
Outline

1. Introduction to Submodular Functions
2. Problem Formulation of SCSC/SCSK
3. Algorithmic Framework
4. Empirical Results
Set functions $f : 2^V \rightarrow \mathbb{R}$

- $V$ is a finite “ground” set of objects.
- A set function $f : 2^V \rightarrow \mathbb{R}$ produces a value for any subset $A \subseteq V$. 
Set functions $f : 2^V \rightarrow \mathbb{R}$

For example, $f(A) = 22$, 
Submodular Set Functions

- Special class of set functions.

\[
f(A \cup v) - f(A) \geq f(B \cup v) - f(B), \text{ if } A \subseteq B \quad (1)
\]
Submodular Set Functions

- Special class of set functions.

\[ f(A \cup v) - f(A) \geq f(B \cup v) - f(B), \text{ if } A \subseteq B \]  

(1)
Submodular Set Functions

- Special class of set functions.

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

Gain $= 1$
Submodular Set Functions

- Special class of set functions.

\[ f(A \cup v) - f(A) \geq f(B \cup v) - f(B), \text{ if } A \subseteq B \]  

Gain = 1  

Gain = 0
Submodular Set Functions

- Special class of set functions.

\[
f(A \cup v) - f(A) \geq f(B \cup v) - f(B), \text{ if } A \subseteq B
\]  

- Monotonicity: \( f(A) \leq f(B), \text{ if } A \subseteq B. \)
Submodular Set Functions

- Special class of set functions.

\[ f(A \cup v) - f(A) \geq f(B \cup v) - f(B), \text{ if } A \subseteq B \]  

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

Gain = 1  

Gain = 0
Two Sides of Submodularity
Two Sides of Submodularity

Submodular Minimization

- Solve \( \min \{ f(X) | X \subseteq V \} \).
- Polynomial-time.
- Relation to convexity.
- Models cooperation.

\[
f(\text{Pizza}) - f(\text{Soda}) \geq f(\text{Pizza + Soda}) - f(\text{Soda})
\]
Two Sides of Submodularity

**Submodular Minimization**
- Solve $\min \{ f(X) | X \subseteq V \}$.
- Polynomial-time.
- Relation to convexity.
- Models cooperation.

$$f(\text{🍎} \text{🍔} ) - f(\text{🍎}) \geq f(\text{🍔}) - f(\text{🍎})$$

**Submodular Maximization**
- Solve $\max \{ g(X) | X \subseteq V \}$.
- Constant-factor approximable.
- Relation to concavity.
- Models diversity and coverage.

Sometimes we want to simultaneously maximize coverage/diversity ($g(X)$) while minimizing cooperative costs ($f(X)$). Often these naturally occur as budget or cover constraints (for example, maximize diversity subject to a budget constraint on the submodular cost).
Two Sides of Submodularity

### Submodular Minimization
- Solve $\min \{ f(X) | X \subseteq V \}$.
- Polynomial-time.
- Relation to convexity.
- Models cooperation.

\[
 f(\text{️}) - f(\text{️}) \geq f(\text{️️}) - f(\text{️})
\]

### Submodular Maximization
- Solve $\max \{ g(X) | X \subseteq V \}$.
- Constant-factor approximable.
- Relation to concavity.
- Models diversity and coverage.

Sometimes we want to simultaneously maximize coverage/diversity ($g$) while minimizing cooperative costs ($f$).
Two Sides of Submodularity

Submodular Minimization
- Solve $\min\{f(X) | X \subseteq V\}$.
- Polynomial-time.
- Relation to convexity.
- Models cooperation.

Submodular Maximization
- Solve $\max\{g(X) | X \subseteq V\}$.
- Constant-factor approximable.
- Relation to concavity.
- Models diversity and coverage.

Sometimes we want to simultaneously maximize coverage/diversity ($g$) while minimizing cooperative costs ($f$).

Often these naturally occur as budget or cover constraints (for example, maximize diversity subject to a budget constraint on the submodular cost).
Submodular Optimization with Submodular Constraints

Historically: DS optimization

\[
\min_{X \subseteq V} f(X) - \lambda g(X)
\]
Historically: DS optimization

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

Co-operative Costs       Coverage/ Diversity

Historically: DS optimization

Unfortunately, NP hard to approximate (Iyer-Bilmes'12).

We introduce the following, which is often more natural anyway:

While DS optimization is NP hard to approximate, SCSC and SCSK however, retain approximation guarantees!
Submodular Optimization with Submodular Constraints

- Historically: DS optimization

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

- Co-operative Costs  Coverage/ Diversity

- Unfortunately, NP hard to approximate (Iyer-Bilmes’12).
Submodular Optimization with Submodular Constraints

Historically: DS optimization

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

Co-operative Costs Coverage/Diversity

Unfortunately, NP hard to approximate (Iyer-Bilmes’12).

We introduce the following, which is often more natural anyway:
Historically: DS optimization

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

Unfortunately, NP hard to approximate (Iyer-Bilmes’12).

We introduce the following, which is often more natural anyway:

**SCSC:** \(\min\{f(X) : g(X) \geq c\}\),  **SCSK:** \(\max\{g(X) : f(X) \leq b\}\),
Historically: DS optimization

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

Unfortunately, NP hard to approximate (Iyer-Bilmes’12).

We introduce the following, which is often more natural anyway:

SCSC: \( \min \{ f(X) : g(X) \geq c \} \),  
SCSK: \( \max \{ g(X) : f(X) \leq b \} \),
Submodular Optimization with Submodular Constraints

- Historically: DS optimization

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

- Unfortunately, NP hard to approximate (Iyer-Bilmes’12).

- We introduce the following, which is often more natural anyway:

\[
\begin{align*}
\text{SCSC: } & \min \{ f(X) : g(X) \geq c \}, \\
\text{SCSK: } & \max \{ g(X) : f(X) \leq b \},
\end{align*}
\]

- While DS optimization is NP hard to approximate, SCSC and SCSK however, retain approximation guarantees!
Submodular Optimization with Submodular Constraints

- Historically: DS optimization

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

- Unfortunately, NP hard to approximate (Iyer-Bilmes’12).

- We introduce the following, which is often more natural anyway:

\[
\begin{align*}
& \text{SCSC: } \min \{ f(X) : g(X) \geq c \}, \\
& \text{SCSK: } \max \{ g(X) : f(X) \leq b \},
\end{align*}
\]

- While DS optimization is NP hard to approximate, SCSC and SCSK however, retain approximation guarantees!

- Throughout this talk, assume \( f \) and \( g \) are monotone.

Iyer & Bilmes, 2013 (UW, Seattle)
Our Main Contributions

- Show how SCSC/SCSK subsume a number of important optimization problems.
- Provide a unifying algorithmic framework for these.
- Provide a complete characterization of the hardness of these problems.
- Emphasize the scalability and practicality of some of our algorithms!
I - Submodular Set Cover and Submodular Knapsack

\[
\text{SSC: } \min \{w(X) : g(X) \geq c\}, \quad \text{SK: } \max \{g(X) : w(X) \leq b\},
\]
I - Submodular Set Cover and Submodular Knapsack

Coverage/Diversity

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

Additive Costs
I - Submodular Set Cover and Submodular Knapsack

Coverage/Diversity

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

Additive Costs

Sensor Placement  
(Krause et al’08)

Data Subset Selection  
(Wei et al’13)

Document Summarization  
(Lin-Bilmes’11)

Iyer & Bilmes, 2013  (UW, Seattle)
II - Submodular Cost with Modular Constraints

\[ \text{SML: } \min \{ f(X) : \ w(X) \geq c \}, \hspace{0.5cm} \text{SS: } \max \{ w(X) : f(X) \leq b \}, \]
II - Submodular Cost with Modular Constraints

Additive functions

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

Co-operative Costs
II - Submodular Cost with Modular Constraints

Additive functions

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

Co-operative Costs

Limited vocabulary speech corpus selection (Lin-Bilmes’11)

Iyer & Bilmes, 2013 (UW, Seattle)
III - Most General Case: SCSC and SCSK

SCSC: \( \min \{ f(X) : g(X) \geq c \} \),  
SCSK: \( \max \{ g(X) : f(X) \leq b \} \),
III - Most General Case: SCSC and SCSK

Coverage/Diversity

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

Co-operative Costs
III - Most General Case: SCSC and SCSK

Coverage/ Diversity

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

Co-operative Costs

Sensor Placement with Submodular Costs (I-Bilmes’12)

Limited vocabulary and accoustically diverse speech corpus selection (Lin-Bilmes’11, Wei et al’13)

Privacy preserving communication (I-Bilmes’13)
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
    - \( f(X) \leq \sigma b \).

Theorem: Given a \([\sigma, \rho]\) bi-criterion approx. algorithm for SCSC, we can obtain a \([(1 + \epsilon)\rho, \sigma]\) bi-criterion approx. algorithm for SCSK, by running the algorithm for SCSC, \(O(\log \frac{1}{\epsilon})\) times. The other direction also holds!
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, we can obtain a \([(1 + \epsilon)\rho, \sigma]\) bi-criterion approx. algorithm for SCSK, by running the algorithm for SCSC, \(O(\log \frac{1}{\epsilon})\) times.
  - The other direction also holds!
Curvature of a Submodular Function

- Curvature:
  \[ \kappa_f = 1 - \min_{j \in V} \frac{f(j | V \setminus j)}{f(j)} \quad \text{and} \quad \kappa_g = 1 - \min_{j \in V} \frac{g(j | V \setminus j)}{g(j)} \]  

- Curvature is a fundamental “complexity” parameter of a submodular function.
## Hardness (Lower bounds) of the problems

<table>
<thead>
<tr>
<th></th>
<th>Modular $g$</th>
<th>Submodular $g$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(\kappa_g = 0)$</td>
<td>(0 &lt; $\kappa_g$ &lt; 1)</td>
<td>$(\kappa_g = 1)$</td>
</tr>
</tbody>
</table>

- **Modular $f$** ($\kappa_f = 0$)
- **Submod $f$** ($0 < \kappa_f < 1$)
- **Submod $f$** ($\kappa_f = 1$)
Hardness (Lower bounds) of the problems

<table>
<thead>
<tr>
<th></th>
<th>Modular $g$</th>
<th>Submodular $g$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$(\kappa_g = 0)$</td>
<td>$(0 &lt; \kappa_g &lt; 1)$</td>
</tr>
<tr>
<td>Modular $f$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$(\kappa_f = 0)$</td>
<td>FPTAS</td>
<td></td>
</tr>
<tr>
<td>Submod $f$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$(0 &lt; \kappa_f &lt; 1)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Submod $f$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$(\kappa_f = 1)$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Hardness (Lower bounds) of the problems

<table>
<thead>
<tr>
<th>Modular $g$</th>
<th>Submodular $g$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(\kappa_g = 0)$</td>
<td>$(0 &lt; \kappa_g &lt; 1)$</td>
</tr>
<tr>
<td><strong>Modular $f$</strong> $(\kappa_f = 0)$</td>
<td>FPTAS</td>
</tr>
<tr>
<td><strong>Submod $f$</strong> $(0 &lt; \kappa_f &lt; 1)$</td>
<td></td>
</tr>
<tr>
<td><strong>Submod $f$</strong> $(\kappa_f = 1)$</td>
<td></td>
</tr>
</tbody>
</table>

Hardness depends (mainly) on $\kappa_f$ and not (so much) on that of $\kappa_g$.  

Iyer & Bilmes, 2013 (UW, Seattle)
### Hardness (Lower bounds) of the problems

<table>
<thead>
<tr>
<th></th>
<th>Modular $g$</th>
<th>Submodular $g$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(\kappa_g = 0)$</td>
<td>FPTAS</td>
<td>$\frac{1}{\kappa_g}(1 - e^{-\kappa_g})$</td>
</tr>
<tr>
<td>$(0 &lt; \kappa_g &lt; 1)$</td>
<td></td>
<td>$1 - 1/e$</td>
</tr>
<tr>
<td>$(\kappa_g = 1)$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Knapsack**
  - Modular $f$ ([$\kappa_f = 0$])
  - Submod $f$ ([$0 < \kappa_f < 1$])
  - Submod $f$ ([$\kappa_f = 1$])

- **SSC/SK**
  - SML/SS

Hardness depends (mainly) on $\kappa_f$ and not (so much) on that of $\kappa_g$.

Iyer & Bilmes, 2013 (UW, Seattle)
### Hardness (Lower bounds) of the problems

<table>
<thead>
<tr>
<th></th>
<th>Modular $g$</th>
<th>Submodular $g$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(\kappa_g = 0)$</td>
<td>FPTAS</td>
<td>$1 - \frac{1}{e}$</td>
</tr>
<tr>
<td>Modular $f$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$(\kappa_f = 0)$</td>
<td>FPTAS</td>
<td>$\frac{1}{\kappa_g}(1 - e^{-\kappa_g})$</td>
</tr>
<tr>
<td>$(0 &lt; \kappa_f &lt; 1)$</td>
<td>$\Omega\left(\frac{\sqrt{n}}{1+(\sqrt{n}-1)(1-\kappa_f)}\right)$</td>
<td>$\Omega\left(\frac{\sqrt{n}}{1+(\sqrt{n}-1)(1-\kappa_f)}\right)$</td>
</tr>
<tr>
<td>Submod $f$</td>
<td>$\Omega(\sqrt{n})$</td>
<td>$\Omega(\sqrt{n})$</td>
</tr>
<tr>
<td>$(0 &lt; \kappa_f &lt; 1)$</td>
<td>$\Omega(\sqrt{n})$</td>
<td>$\Omega(\sqrt{n})$</td>
</tr>
<tr>
<td>Submod $f$</td>
<td>$\Omega(\sqrt{n})$</td>
<td>$\Omega(\sqrt{n})$</td>
</tr>
<tr>
<td>$(\kappa_f = 1)$</td>
<td>$\Omega(\sqrt{n})$</td>
<td>$\Omega(\sqrt{n})$</td>
</tr>
</tbody>
</table>

Iyer & Bilmes, 2013 (UW, Seattle)

Submodular Functions  | Problem Formulation  | Algorithmic Framework  | Empirical Results
---|---|---|---
Knapsack  | SSC/SCSK  | SML/SS  | SCSC/SCSK

Hardness depends (mainly) on $\kappa_f$ and not (so much) on that of $\kappa_g$. 
Hardness (Lower bounds) of the problems

<table>
<thead>
<tr>
<th></th>
<th>Modular ( g ) ((\kappa_g = 0))</th>
<th>Submodular ( g ) ((0 &lt; \kappa_g &lt; 1))</th>
<th>Submodular ( g ) ((\kappa_g = 1))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modular ( f ) ((\kappa_f = 0))</td>
<td>FPTAS ( \frac{1}{\kappa_g}(1 - e^{-\kappa_g}) )</td>
<td>( 1 - 1/e )</td>
<td></td>
</tr>
<tr>
<td>Submod ( f ) ((0 &lt; \kappa_f &lt; 1))</td>
<td>( \Omega\left(\frac{\sqrt{n}}{1+(\sqrt{n}-1)(1-\kappa_f)}\right) )</td>
<td>( \Omega\left(\frac{\sqrt{n}}{1+(\sqrt{n}-1)(1-\kappa_f)}\right) )</td>
<td>( \Omega\left(\frac{\sqrt{n}}{1+(\sqrt{n}-1)(1-\kappa_f)}\right) )</td>
</tr>
<tr>
<td>Submod ( f ) ((\kappa_f = 1))</td>
<td>( \Omega(\sqrt{n}) )</td>
<td>( \Omega(\sqrt{n}) )</td>
<td>( \Omega(\sqrt{n}) )</td>
</tr>
</tbody>
</table>

- Hardness depends (mainly) on \( \kappa_f \) and not (so much) on that of \( \kappa_g \).
Algorithmic framework

**Algorithm 1** General algorithmic framework to address both Problems 1 and 2
Algorithmic framework

Algorithm 1 General algorithmic framework to address both Problems 1 and 2

1: for $t = 1, 2, \cdots, T$ do

4: end for
Algorithmic framework

**Algorithm 1** General algorithmic framework to address both Problems 1 and 2

1. **for** \( t = 1, 2, \cdots, T \) **do**
2. Choose surrogate functions \( \hat{f}_t \) and \( \hat{g}_t \) for \( f \) and \( g \) respectively.

4. **end for**
Algorithmic framework

**Algorithm 1** General algorithmic framework to address both Problems 1 and 2

1. for $t = 1, 2, \cdots, T$ do
2. Choose surrogate functions $\hat{f}_t$ and $\hat{g}_t$ for $f$ and $g$ respectively.
3. Obtain $X^t$ as the optimizer of SCSC/SCSK with $\hat{f}_t$ and $\hat{g}_t$ instead of $f$ and $g$.
4. end for
Algorithmic framework

Algorithm 1  General algorithmic framework to address both Problems 1 and 2

1: for $t = 1, 2, \cdots, T$ do
2: Choose surrogate functions $\hat{f}_t$ and $\hat{g}_t$ for $f$ and $g$ respectively.
3: Obtain $X^t$ as the optimizer of SCSC/SCSK with $\hat{f}_t$ and $\hat{g}_t$ instead of $f$ and $g$.
4: end for

- Surrogate functions: modular upper/ lower bounds or Ellipsoidal Approximations.
Surrogate functions

- **Modular Lower Bounds:** Induced via orderings of elements:

\[
\text{Modular Lower Bounds: } \quad \text{Induced via orderings of elements:}
\]

\[
\begin{align*}
\text{Induced via orderings of elements:} \\
&\text{Modular Lower Bounds:} \\
&\text{Induced via orderings of elements:}
\end{align*}
\]
Surrogate functions

- **Modular Lower Bounds:** Induced via orderings of elements:

\[ f(X) \leq h^\sigma_Y(X), \text{ where } h^\sigma_Y(\sigma(i)) = f(\Sigma_i) - f(\Sigma_{i-1}) \]
Surrogate functions

- **Modular Lower Bounds:** Induced via orderings of elements:

  \[ f(X) \leq h^\sigma_Y(X), \text{ where } h^\sigma_Y(\sigma(i)) = f(\Sigma_i) - f(\Sigma_{i-1}) \]

- **Modular upper bounds:**

  ![Diagram of modular upper bounds]
Surrogate functions

- **Modular Lower Bounds:** Induced via orderings of elements:

  \[ f(X) \leq h^\sigma_Y(X), \quad \text{where} \quad h^\sigma_Y(\sigma(i)) = f(\Sigma_i) - f(\Sigma_{i-1}) \]

- **Modular upper bounds:**
  
  Upper bound-I

  \[ f(X) \leq m_{Y,1}(X) = f(Y) - \sum_{j \in Y \setminus X} f(j|Y \setminus j) + \sum_{j \in X \setminus Y} f(j|\emptyset) \]
Surrogate functions

- **Modular Lower Bounds**: Induced via orderings of elements:

\[
f(X) \leq h_Y^\sigma(X), \text{ where } h_Y^\sigma(\sigma(i)) = f(\Sigma_i) - f(\Sigma_{i-1})
\]

- **Modular upper bounds**:

Upper bound-II

\[
f(X) \leq m_{Y,2}(X) = f(Y) - \sum_{j \in Y \setminus X} f(j|V \setminus j) + \sum_{j \in X \setminus Y} f(j|Y)
\]
Surrogate functions

- **Modular Lower Bounds:** Induced via orderings of elements:

\[ f(X) \leq h^\sigma_Y(X), \text{ where } h^\sigma_Y(\sigma(i)) = f(\Sigma_i) - f(\Sigma_{i-1}) \]

- **Modular upper bounds:**
  Upper bound-II

\[ f(X) \leq m_{Y,2}(X) = f(Y) - \sum_{j \in Y \setminus X} f(j \mid V \setminus j) + \sum_{j \in X \setminus Y} f(j \mid Y) \]
Surrogate functions

- **Modular Lower Bounds:** Induced via orderings of elements:
  \[ f(X) \leq h_Y^\sigma(X), \text{ where } h_Y^\sigma(\sigma(i)) = f(\Sigma_i) - f(\Sigma_{i-1}) \]

- **Modular upper bounds:**
  Upper bound-II
  \[ f(X) \leq m_{Y,2}(X) = f(Y) - \sum_{j \in Y \setminus X} f(j|V \setminus j) + \sum_{j \in X \setminus Y} f(j|Y) \]

- **Approximations:** Ellipsoidal Approximation gives the *tightest* approximation to a submodular function.
Submodular Set Cover (SSC) and Submodular Knapsack (SK)

- **Coverage/Diversity**
  - SSC: \( \min \{ w(X) : g(X) \geq c \} \)
  - SK: \( \max \{ g(X) : w(X) \leq b \} \)

- **Additive Costs**

---

Iyer & Bilmes, 2013 (UW, Seattle)
Submodular Set Cover (SSC) and Submodular Knapsack (SK)

Lemma: The greedy algorithm for SSC (Wolsey, 82) and SK (Nemhauser, 78) is a special case of Algorithm 1 with $g$ replaced by its modular lower bound.

Coverage/Diversity

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

Additive Costs
Submodular Set Cover (SSC) and Submodular Knapsack (SK)

Lemma: The greedy algorithm for SSC (Wolsey, 82) and SK (Nemhauser, 78) is special case of Algorithm 1 with $g$ replaced by its modular lower bound.

Approximation guarantees are constant factor $1 - 1/e$ respectively.
Iterative Submodular Set Cover (ISSC)/Submodular Knapsack (ISK)

Choose surrogate functions $\hat{f}_t$ as modular upper bounds.
Iterative Submodular Set Cover (ISSC)/Submodular Knapsack (ISK)

- Choose surrogate functions $\hat{f}_t$ as modular upper bounds.
- Fast iterative algorithms for SCSC and SCSK – Iteratively solve SSC or SK.

\[ \text{SCSC: } \min\{f(X) : g(X) \geq c\}, \quad \text{SCSK: } \max\{g(X) : f(X) \leq b\}, \]
Iterative Submodular Set Cover (ISSC)/Submodular Knapsack (ISK)

Choose surrogate functions $\hat{f}_t$ as modular upper bounds.
Fast iterative algorithms for SCSC and SCSK – Iteratively solve SSC or SK.
Theorem: ISSC and ISK obtain (bi-criterion) approximation factors
$$\frac{\sigma}{\rho} = O\left(\frac{n}{1+(n-1)(1-\kappa_f)}\right).$$
Iterative Submodular Set Cover (ISSC)/Submodular Knapsack (ISK)

- Choose surrogate functions $\hat{f}_t$ as modular upper bounds.
- Fast iterative algorithms for SCSC and SCSK – Iteratively solve SSC or SK.
- Theorem: ISSC and ISK obtain (bi-criterion) approximation factors $\frac{\sigma}{\rho} = O\left(\frac{n}{1+(n-1)(1-\kappa_f)}\right)$.
- These algorithms also extend to SML/SS.
Choose surrogate functions $\hat{f}_t$ as Ellipsoidal Approximation, in both SCSC and SCSK.
Choose surrogate functions $\hat{f}_t$ as Ellipsoidal Approximation, in both SCSC and SCSK.

Theorem: EASSC and EASK obtain (bi-criterion) approximation factors of $\frac{\sigma}{\rho} = O\left(\frac{\sqrt{n \log n}}{1+\left(\sqrt{n \log n} - 1\right)(1-\kappa_f)}\right)$. 

Theorem: EASSC and EASK obtain (bi-criterion) approximation factors of $\frac{\sigma}{\rho} = O\left(\frac{\sqrt{n \log n}}{1+\left(\sqrt{n \log n} - 1\right)(1-\kappa_f)}\right)$. 

These algorithms also extend to SML/SS. This algorithm matches the hardness of this problem up to log factors.
Ellipsoidal Approximation Submodular Set Cover (EASSC)/ Submodular Knapsack (EASK)

Choose surrogate functions $\hat{f}_t$ as Ellipsoidal Approximation, in both SCSC and SCSK.

Theorem: EASSC and EASK obtain (bi-criterion) approximation factors of $\frac{\sigma}{\rho} = O\left(\frac{\sqrt{n}\log n}{1+\left(\sqrt{n}\log n - 1\right)(1-\kappa_f)}\right)$.

These algorithms also extend to SML/SS.
Choose surrogate functions $\hat{f}_t$ as Ellipsoidal Approximation, in both SCSC and SCSK.

Theorem: EASSC and EASK obtain (bi-criterion) approximation factors of 

$$\frac{\sigma}{\rho} = O\left(\frac{\sqrt{n} \log n}{1 + (\sqrt{n} \log n - 1)(1 - \kappa_f)}\right).$$

These algorithms also extend to SML/SS.

This algorithm matches the hardness of this problem upto log factors.
Limited Vocabulary data subset selection with Acoustic diversity

- **Acoustic Diversity:**

```
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 data subset selection with Acoustic diversity

- **Accoustic Diversity:**
  - Similarity matrix $s_{ij}$ between utterances $i$ and $j$ (string kernel)
Limited Vocabulary data subset selection with Acoustic diversity

- **Accoustic Diversity:**
- Similarity matrix $s_{ij}$ between utterances $i$ and $j$ (string kernel)
- Submodular functions:

```plaintext
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 data subset selection with Acoustic diversity

**Acoustic Diversity:**

- Similarity matrix $s_{ij}$ between utterances $i$ and $j$ (string kernel)
- Submodular functions:
  - Facility Location function:
    \[ g(X) = \sum_{i \in V} \max_{j \in X} s_{ij} \]
Limited Vocabulary data subset selection with Acoustic diversity

- **Acoustic Diversity:**
  - Similarity matrix $s_{ij}$ between utterances $i$ and $j$ (string kernel)
  - Submodular functions:
    1. Facility Location function:
       $$g(X) = \sum_{i \in V} \max_{j \in X} s_{ij}$$
    2. Saturated coverage function
       $$g(X) = \sum_{i \in V} \min\{\sum_{j \in X} s_{ij}, \beta \sum_{j \in V} s_{ij}\}.$$
Limited Vocabulary data subset selection with Acoustic diversity

- **Accoustic Diversity:**
  - Similarity matrix $s_{ij}$ between utterances $i$ and $j$ (string kernel)
  - Submodular functions:
    1. Facility Location function:
       $$ g(X) = \sum_{i \in V} \max_{j \in X} s_{ij} $$
    2. Saturated coverage function
       $$ g(X) = \sum_{i \in V} \min\{\sum_{j \in X} s_{ij}, \beta \sum_{j \in V} s_{ij}\} $$

- **Limited Vocabulary:**

---

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

---

Iyer & Bilmes, 2013 (UW, Seattle)
Limited Vocabulary data subset selection with Acoustic diversity

- **Acoustic Diversity:**
  - Similarity matrix $s_{ij}$ between utterances $i$ and $j$ (string kernel)
  - Submodular functions:
    1. Facility Location function:
       $$g(X) = \sum_{i \in V} \max_{j \in X} s_{ij}$$
    2. Saturated coverage function
       $$g(X) = \sum_{i \in V} \min\{\sum_{j \in X} s_{ij}, \beta \sum_{j \in V} s_{ij}\}.$$ 

- **Limited Vocabulary:**

![Bipartite graph](image-url)
Limited Vocabulary data subset selection with Acoustic diversity

- **Acoustic Diversity:**
  - Similarity matrix $s_{ij}$ between utterances $i$ and $j$ (string kernel)
  - Submodular functions:
    1. Facility Location function:
       $$g(X) = \sum_{i \in V} \max_{j \in X} s_{ij}$$
    2. Saturated coverage function
       $$g(X) = \sum_{i \in V} \min\{\sum_{j \in X} s_{ij}, \beta \sum_{j \in V} s_{ij}\}.$$  

- **Limited Vocabulary:**

![Bipartite graph](image)

Bipartite Neighborhood function: $|\gamma(X)|$.  

Iyer & Bilmes, 2013 (UW, Seattle)
Results

- Compare our different algorithms on the TIMIT speech corpus.
Results

- Compare our different algorithms on the TIMIT speech corpus.
- Baseline is choosing random subsets.
Results

- Compare our different algorithms on the TIMIT speech corpus.
- Baseline is choosing random subsets.
- Observations:

  ![Graph - Fac. Location/ Bipartite Neighbor.](image1)

  ![Graph - Saturated Sum/ Bipartite Neighbor](image2)

Iyer & Bilmes, 2013 (UW, Seattle)
Results

- Compare our different algorithms on the TIMIT speech corpus.
- Baseline is choosing random subsets.
- Observations:
  1. All the algorithms perform much better than random subset selection.

![Graph 1](image1.png)

Fac. Location/ Bipartite Neighbor.

![Graph 2](image2.png)

Saturated Sum/ Bipartite Neighbor
Compare our different algorithms on the TIMIT speech corpus.
Baseline is choosing random subsets.
Observations:
1. All the algorithms perform much better than random subset selection.
2. The iterative and much faster algorithms, perform comparably to the slower and tight Ellipsoidal Approximation based algorithms.
Conclusions/ Future Work

- We proposed some very efficient (scalable) algorithms and two tight algorithms for submodular optimization under submodular constraints.
- In the paper: Extensions to handle multiple constraints, and non-monotone submodular functions.
- Future Work: Investigate our new algorithms on different real world applications.

Thank You!
Connections between SCSC and SCSK

- SCSC and SCSK are closely related, and can be transformed into one another!
Connections between SCSC and SCSK

- SCSC and SCSK are closely related, and can be transformed into one another!

- **Bi-criterion factor:** $[\sigma, \rho]$ approximation for (1) $\implies$ a set $X : f(X) \leq \sigma f(X^*)$ and $g(X) \geq \rho c$ [$\sigma > 1, \rho < 1$].

\[\text{Algorithm 2}
\]

\begin{enumerate}
\item \textbf{Input:} An SCSC instance, $c$, $[\sigma, \rho]$ algorithm for SCSK, $\epsilon > 0$.
\item \textbf{Output:} $[1 + \epsilon \sigma, \rho]$ approx. for SCSC.
\item $b \leftarrow \arg\min_j f(j), \hat{X}_b \leftarrow \emptyset$.
\item \textbf{while} $g(\hat{X}_b) < \rho$ \textbf{do}
\item $b \leftarrow (1 + \epsilon \sigma)$
\item $\hat{X}_b \leftarrow [\rho, \sigma]$ approx. for SCSK using $b$.
\item \textbf{end while}
\item \textbf{Return} $\hat{X}_b$.
\end{enumerate}
Connections between SCSC and SCSK

- SCSC and SCSK are closely related, and can be transformed into one another!

  **Bi-criterion factor:** $[\sigma,\rho]$ approximation for (1) $\implies$ a set $X: f(X) \leq \sigma f(X^*)$ and $g(X) \geq \rho c$. A $[\rho,\sigma]$ approximation for (2) $\implies$ a set $X: g(X) \geq \rho g(X^*)$ and $f(X) \leq \sigma b$ $[\sigma > 1, \rho < 1]$. 

Algorithm 2

1: Input: An SCSC instance, $c$, $[\rho,\sigma]$ algorithm for SCSK, $\epsilon > 0$.
2: Output: $[1 + \epsilon \sigma, \rho]$ approx. for SCSC.
3: $b \leftarrow \arg\min_j f(j)$, $\hat{X}_b \leftarrow \emptyset$.
4: while $g(\hat{X}_b) < \rho c$ do
5:   $b \leftarrow (1 + \epsilon) b$
6:   $\hat{X}_b \leftarrow [\rho,\sigma]$ approx. for SCSK using $b$.
7: end while
8: Return $\hat{X}_b$.
Connections between SCSC and SCSK

- SCSC and SCSK are closely related, and can be transformed into one another!
- **Bi-criterion factor:** $[\sigma, \rho]$ approximation for (1) $\implies$ a set $X : f(X) \leq \sigma f(X^*)$ and $g(X) \geq \rho c$. A $[\rho, \sigma]$ approximation for (2) $\implies$ a set $X : g(X) \geq \rho g(X^*)$ and $f(X) \leq \sigma b$ $[\sigma > 1, \rho < 1]$.

**Algorithm 2** Algorithm for SCSC using an algorithm for SCSK
Connections between SCSC and SCSK

- SCSC and SCSK are closely related, and can be transformed into one another!
- **Bi-criterion factor**: \([\sigma, \rho]\) approximation for (1) \(\Rightarrow\) a set \(X : f(X) \leq \sigma f(X^*)\) and \(g(X) \geq \rho c\). A \([\rho, \sigma]\) approximation for (2) \(\Rightarrow\) a set \(X : g(X) \geq \rho g(X^*)\) and \(f(X) \leq \sigma b\) \([\sigma > 1, \rho < 1]\).

**Algorithm 2** Algorithm for SCSC using an algorithm for SCSK

1. **Input**: An SCSC instance, \(c\), \([\rho, \sigma]\) algorithm for SCSK, \(\epsilon > 0\).
SCSC and SCSK are closely related, and can be transformed into one another!

**Bi-criterion factor:** \([\sigma, \rho]\) approximation for (1) \(\Rightarrow\) a set \(X:\ f(X) \leq \sigma f(X^*)\) and \(g(X) \geq \rho c\). A \([\rho, \sigma]\) approximation for (2) \(\Rightarrow\) a set \(X:\ g(X) \geq \rho g(X^*)\) and \(f(X) \leq \sigma b\) \([\sigma > 1, \rho < 1]\).

---

**Algorithm 2** Algorithm for SCSC using an algorithm for SCSK

1. **Input:** An SCSC instance, \(c\), \([\rho, \sigma]\) algorithm for SCSK, \(\epsilon > 0\).
2. **Output:** \([(1 + \epsilon)\sigma, \rho]\) approx. for SCSC.
Connections between SCSC and SCSK

- SCSC and SCSK are closely related, and can be transformed into one another!

- Bi-criterion factor: \([\sigma, \rho]\) approximation for (1) \(\Rightarrow\) a set \(X : f(X) \leq \sigma f(X^*)\) and \(g(X) \geq \rho c\). A \([\rho, \sigma]\) approximation for (2) \(\Rightarrow\) a set \(X : g(X) \geq \rho g(X^*)\) and \(f(X) \leq \sigma b\) \([\sigma > 1, \rho < 1]\).

**Algorithm 2** Algorithm for SCSC using an algorithm for SCSK

1. **Input:** An SCSC instance, \(c\), \([\rho, \sigma]\) algorithm for SCSK, \(\epsilon > 0\).
2. **Output:** \([(1 + \epsilon)\sigma, \rho]\) approx. for SCSC.
3. \(b \leftarrow \text{argmin}_j f(j), \hat{X}_b \leftarrow \emptyset\).
Connections between SCSC and SCSK

- SCSC and SCSK are closely related, and can be transformed into one another!
- **Bi-criterion factor:** \([\sigma, \rho]\) approximation for (1) \(\implies\) a set \(X : f(X) \leq \sigma f(X^*)\) and \(g(X) \geq \rho c\). A \([\rho, \sigma]\) approximation for (2) \(\implies\) a set \(X : g(X) \geq \rho g(X^*)\) and \(f(X) \leq \sigma b\) \([\sigma > 1, \rho < 1]\).

### Algorithm 2
**Algorithm for SCSC using an algorithm for SCSK**

1. **Input:** An SCSC instance, \(c\), \([\rho, \sigma]\) algorithm for SCSK, \(\epsilon > 0\).
2. **Output:** \([(1 + \epsilon)\sigma, \rho]\) approx. for SCSC.
3. \(b \leftarrow \text{argmin}_j f(j), \hat{X}_b \leftarrow \emptyset\).
4. **while** \(g(\hat{X}_b) < \rho c\) **do**

7. **end while**
Connections between SCSC and SCSK

- SCSC and SCSK are closely related, and can be transformed into one another!
- **Bi-criterion factor:** $[\sigma, \rho]$ approximation for $(1) \implies$ a set $X : f(X) \leq \sigma f(X^*)$ and $g(X) \geq \rho c$. A $[\rho, \sigma]$ approximation for $(2) \implies$ a set $X : g(X) \geq \rho g(X^*)$ and $f(X) \leq \sigma b \ [\sigma > 1, \rho < 1]$. 

**Algorithm 2** Algorithm for SCSC using an algorithm for SCSK

1. **Input:** An SCSC instance, $c$, $[\rho, \sigma]$ algorithm for SCSK, $\epsilon > 0$.
2. **Output:** $[(1 + \epsilon)\sigma, \rho]$ approx. for SCSC.
3. $b \leftarrow \arg\min_j f(j)$, $\hat{X}_b \leftarrow \emptyset$.
4. **while** $g(\hat{X}_b) < \rho c$ **do**
5. $b \leftarrow (1 + \epsilon) b$
6. $\hat{X}_b \leftarrow [\rho, \sigma]$ approx. for SCSK using $b$.
7. **end while**
Connections between SCSC and SCSK

- SCSC and SCSK are closely related, and can be transformed into one another!

- **Bi-criterion factor:** \([\sigma, \rho]\) approximation for (1) \(\Rightarrow\) a set \(X : f(X) \leq \sigma f(X^*)\) and \(g(X) \geq \rho c\). A \([\rho, \sigma]\) approximation for (2) \(\Rightarrow\) a set \(X : g(X) \geq \rho g(X^*)\) and \(f(X) \leq \sigma b\) \([\sigma > 1, \rho < 1]\).

**Algorithm 2** Algorithm for SCSC using an algorithm for SCSK

1: **Input:** An SCSC instance, \(c\), \([\rho, \sigma]\) algorithm for SCSK, \(\epsilon > 0\).
2: **Output:** \([(1 + \epsilon)\sigma, \rho]\) approx. for SCSC.
3: \(b \leftarrow \arg\min_j f(j), \hat{X}_b \leftarrow \emptyset\).
4: **while** \(g(\hat{X}_b) < \rho c\) **do**
5: \(b \leftarrow (1 + \epsilon)b\)
6: \(\hat{X}_b \leftarrow [\rho, \sigma]\) approx. for SCSK using \(b\).
7: **end while**
8: Return \(\hat{X}_b\).
Hardness Theorem

- Theorem: For any $\kappa > 0$, there exists submodular function $f$ with curvature $\kappa_f = \kappa$ such that no polynomial time algorithm for SCSC and SCSK $\frac{\sigma}{\rho} = \frac{n^{1/2-\epsilon}}{1+(n^{1/2-\epsilon}-1)(1-\kappa)}$ for any $\epsilon > 0$. 

Hardness depends on the curvature of the submodular function $f$ and not on that of $g$. 

Table: Summary of Hardness results for SCSC/SCSK
Theorem: For any $\kappa > 0$, there exists submodular function $f$ with curvature $\kappa_f = \kappa$ such that no polynomial time algorithm for SCSC and SCSK $\frac{\sigma}{\rho} = \frac{n^{1/2-\epsilon}}{1+(n^{1/2-\epsilon}-1)(1-\kappa)}$ for any $\epsilon > 0$.

Hardness depends on the curvature of the submodular function $f$ and not on that of $g$. 
Hardness Theorem

- Theorem: For any $\kappa > 0$, there exists submodular function $f$ with curvature $\kappa_f = \kappa$ such that no polynomial time algorithm for SCSC and SCSK $\frac{\sigma}{\rho} = \frac{n^{1/2-\epsilon}}{1+(n^{1/2-\epsilon}-1)(1-\kappa)}$ for any $\epsilon > 0$.

- Hardness depends on the curvature of the submodular function $f$ and not on that of $g$.

<table>
<thead>
<tr>
<th></th>
<th>Modular $g$</th>
<th>Submodular $g$</th>
</tr>
</thead>
<tbody>
<tr>
<td>($\kappa_g = 0$)</td>
<td>FPTAS</td>
<td>$\frac{1}{\kappa_g}(1 - e^{-\kappa_g})$</td>
</tr>
<tr>
<td>Modular $f$ ($\kappa_f = 0$)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Submod $f$ ($0 &lt; \kappa_f &lt; 1$)</td>
<td>$\Omega(\frac{\sqrt{n}}{1+(\sqrt{n}-1)(1-\kappa_f)})$</td>
<td>$\Omega(\frac{\sqrt{n}}{1+(\sqrt{n}-1)(1-\kappa_f)})$</td>
</tr>
<tr>
<td>Submod $f$ ($\kappa_f = 1$)</td>
<td>$\Omega(\sqrt{n})$</td>
<td>$\Omega(\sqrt{n})$</td>
</tr>
</tbody>
</table>

Table: Summary of Hardness results for SCSC/SCSK