Algorithm Foundations of Data Science and Engineering

Lecture 11: Submodular and Its Applications

YANHAO WANG

DaSE @ ECNU
(for course related communications)
yhwang@dase.ecnu.edu.cn

Nov. 22, 2021

Outline

Motivation: set functions

Feature selection

Motivation: set functions

Feature selection

- Given a set of features $X_1, \cdots, X_n$;

Motivation: set functions

Feature selection

- Given a set of features $X_1, \cdots, X_n$;
- Want to predict Y from a subset $A = (X_{i1}, \cdots, X_{ik})$;

Motivation: set functions

Feature selection

- Given a set of features $X_1, \cdots, X_n$;
- Want to predict Y from a subset $A = (X_{i1}, \cdots, X_{ik})$;
- What are the k most informative features?

Motivation: set functions

Feature selection

- Given a set of features $X_1, \cdots, X_n$;
- Want to predict Y from a subset $A = (X_{i1}, \cdots, X_{ik})$;
- What are the k most informative features?

Influence maximization

Motivation: set functions

Feature selection

- Given a set of features $X_1, \cdots, X_n$;
- Want to predict Y from a subset $A = (X_{i1}, \cdots, X_{ik})$;
- What are the k most informative features?

Influence maximization

- In a social network, which nodes to advertise to?

Motivation: set functions

Feature selection

- Given a set of features $X_1, \cdots, X_n$;
- Want to predict Y from a subset $A = (X_{i1}, \cdots, X_{ik})$;
- What are the k most informative features?

Influence maximization

- In a social network, which nodes to advertise to?
- Which are the most influential blogs?

Motivation: set functions

Feature selection

- Given a set of features $X_1, \cdots, X_n$;
- Want to predict Y from a subset $A = (X_{i1}, \cdots, X_{ik})$;
- What are the k most informative features?

Influence maximization

- In a social network, which nodes to advertise to?
- Which are the most influential blogs?

Sensor placement

Motivation: set functions

Feature selection

- Given a set of features $X_1, \cdots, X_n$;
- Want to predict Y from a subset $A = (X_{i1}, \cdots, X_{ik})$;
- What are the k most informative features?

Influence maximization

- In a social network, which nodes to advertise to?
- Which are the most influential blogs?

Sensor placement

- Given a water distribution network;

Motivation: set functions

Feature selection

- Given a set of features $X_1, \cdots, X_n$;
- Want to predict Y from a subset $A = (X_{i1}, \cdots, X_{ik})$;
- What are the k most informative features?

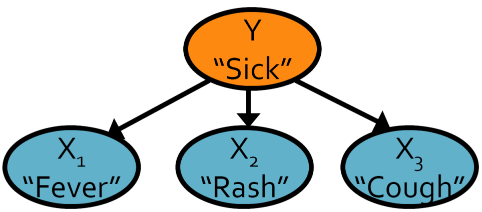
Influence maximization

- In a social network, which nodes to advertise to?
- Which are the most influential blogs?

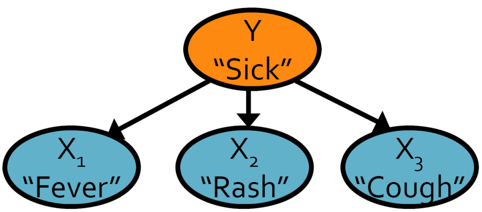
Sensor placement

- Given a water distribution network;
- Where should we place sensors to quickly detect contaminations?

Feature selection

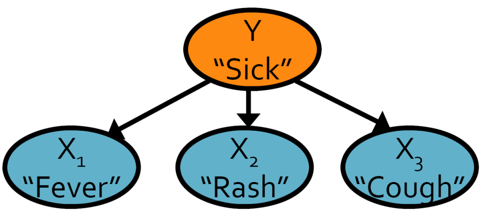


Feature selection



- Given r.v.s $Y, X_1, \cdots, X_n$, predict Y from a subset $A = (X_{i1}, \cdots, X_{ik})$;

Feature selection



- Given r.v.s $Y, X_1, \cdots, X_n$, predict Y from a subset $A = (X_{i1}, \cdots, X_{ik})$;
- Information gain:

$$I(A; Y) = H(Y) - H(Y|A),$$

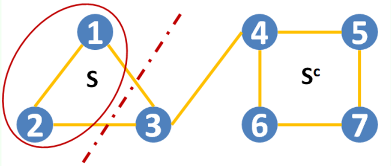
where $H(Y)$ is the conditional entropy, $I(A; Y)$ measures the difference of uncertainty before and after knowing A;

Cut function in a graph

Let $G = (V, E)$ be an undirected graph.

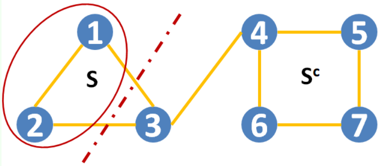
Cut function in a graph

Let $G = (V, E)$ be an undirected graph.



Cut function in a graph

Let $G = (V, E)$ be an undirected graph.

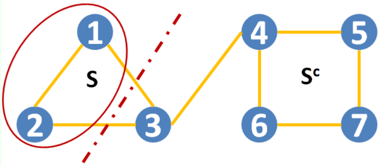


- The graph cut function is defined by

$$f(S) = |\{(u, v) | u \in S \subset V, v \in S^c\}|$$

Cut function in a graph

Let $G = (V, E)$ be an undirected graph.



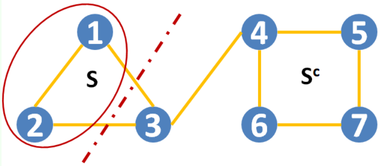
- The graph cut function is defined by

$$f(S) = |\{(u, v)|u \in S \subset V, v \in S^c\}|$$

  - For $S = \{1, 2, 3\}$, $f(S) = 1$;

Cut function in a graph

Let $G = (V, E)$ be an undirected graph.



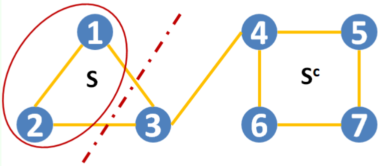
- The graph cut function is defined by

$$f(S) = |\{(u, v)|u \in S \subset V, v \in S^c\}|$$

  - For $S = \{1, 2, 3\}$, $f(S) = 1$;
  - For $S = \{1, 2\}$, $f(S) = 2$;

Cut function in a graph

Let $G = (V, E)$ be an undirected graph.

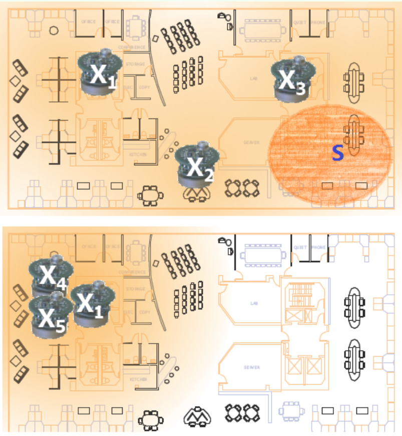


- The graph cut function is defined by

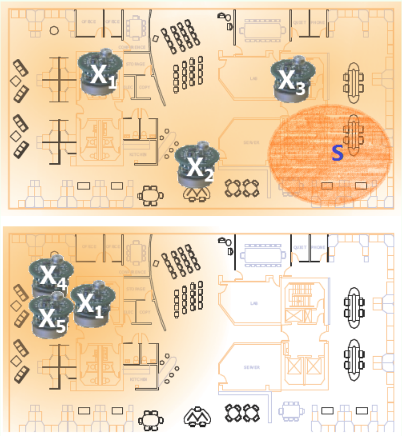
$$f(S) = |\{(u, v)|u \in S \subset V, v \in S^c\}|$$

  - For $S = \{1, 2, 3\}$, $f(S) = 1$;
  - For $S = \{1, 2\}$, $f(S) = 2$;
- The graph cut is a set function.

Sensor placement

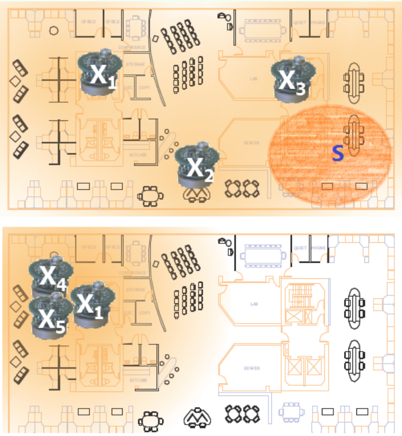


Sensor placement



Function $f(A)$ is the value of utility of having sensors at subset A of all locations.

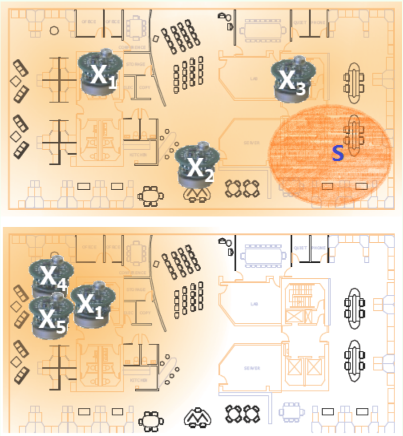
Sensor placement



Function $f(A)$ is the value of utility of having sensors at subset A of all locations.

- $A = \{1, 2, 3\}$ very informative (high value of $f(A)$).

Sensor placement



Function $f(A)$ is the value of utility of having sensors at subset A of all locations.

- $A = \{1, 2, 3\}$ very informative (high value of $f(A)$).
- $A = \{1, 4, 5\}$ redundant information (low value of $f(A)$).

Set function

A finite set $V = \{1, 2, \cdots, n\}$, set function

$$f : 2^V \to R,$$

where 2^V is the power set of V,

Set function

A finite set $V = \{1, 2, \cdots, n\}$, set function

$$f : 2^V \to R,$$

where 2^V is the power set of V, or

$$f : \{0, 1\}^n \to R.$$

Set function

A finite set $V = \{1, 2, \cdots, n\}$, set function

$$f : 2^V \to R,$$

where 2^V is the power set of V, or

$$f : \{0, 1\}^n \to R.$$

Other possibly useful properties a set function may have:

Set function

A finite set $V = \{1, 2, \cdots, n\}$, set function

$$f : 2^V \to R,$$

where 2^V is the power set of V, or

$$f : \{0, 1\}^n \to R.$$

Other possibly useful properties a set function may have:

- Monotone: if $A \subseteq B \subseteq X$, then $F(A) \leq F(B)$;

Set function

A finite set $V = \{1, 2, \cdots, n\}$, set function

$$f : 2^V \to R,$$

where 2^V is the power set of V, or

$$f : \{0, 1\}^n \to R.$$

Other possibly useful properties a set function may have:

- Monotone: if $A \subseteq B \subseteq X$, then $F(A) \leq F(B)$;
- Nonnegative: $F(A) \geq 0$ for all $S \subseteq X$;

Set function

A finite set $V = \{1, 2, \cdots, n\}$, set function

$$f : 2^V \to R,$$

where 2^V is the power set of V, or

$$f : \{0, 1\}^n \to R.$$

Other possibly useful properties a set function may have:

- Monotone: if $A \subseteq B \subseteq X$, then $F(A) \leq F(B)$;
- Nonnegative: $F(A) \geq 0$ for all $S \subseteq X$;
- Normalized: $F(\emptyset) = 0$.

Set function

A finite set $V = \{1, 2, \cdots, n\}$, set function

$$f : 2^V \to R,$$

where 2^V is the power set of V, or

$$f : \{0, 1\}^n \to R.$$

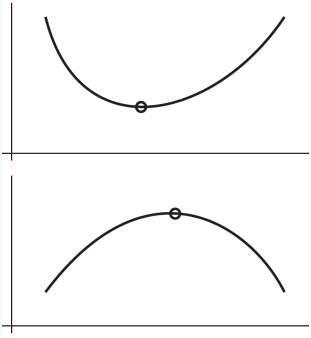
Other possibly useful properties a set function may have:

- Monotone: if $A \subseteq B \subseteq X$, then $F(A) \leq F(B)$;
- Nonnegative: $F(A) \geq 0$ for all $S \subseteq X$;
- Normalized: $F(\emptyset) = 0$.

There are many set functions, such as information gain, graph cut, and sensor utility, etc.

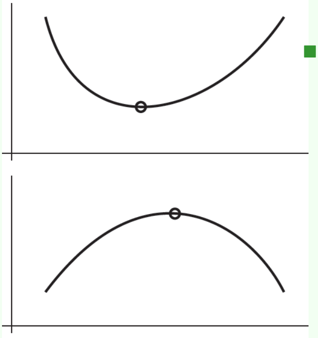
Continuous optimization

What makes continuous optimization tractable?



Continuous optimization

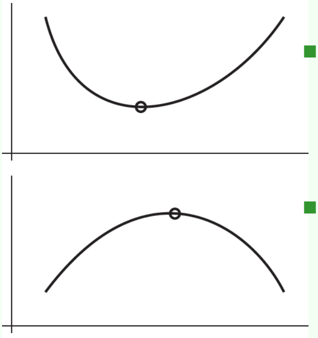
What makes continuous optimization tractable?



- A function $f : R^n \rightarrow R$ can be minimized efficiently, if it is convex.

Continuous optimization

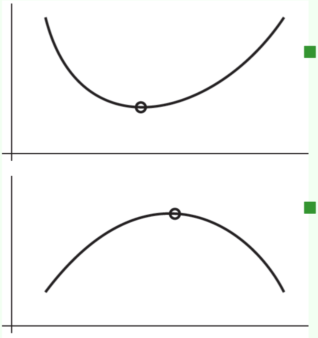
What makes continuous optimization tractable?



- A function $f : R^n \to R$ can be minimized efficiently, if it is convex.
- A function $f : R^n \to R$ can be maximized efficiently, if it is concave.

Continuous optimization

What makes continuous optimization tractable?

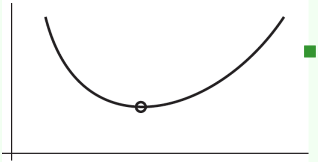


- A function $f : R^n \rightarrow R$ can be minimized efficiently, if it is convex.
- A function $f : R^n \rightarrow R$ can be maximized efficiently, if it is concave.

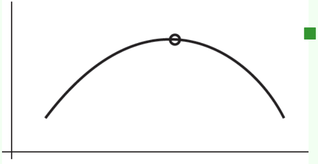
Discrete analogy?

Continuous optimization

What makes continuous optimization tractable?



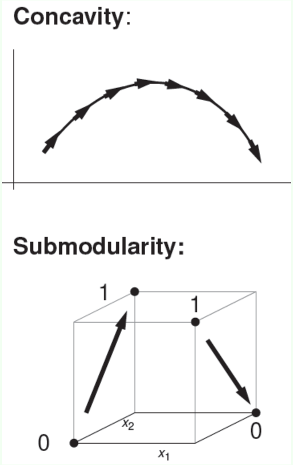
- A function $f : R^n \rightarrow R$ can be minimized efficiently, if it is convex.



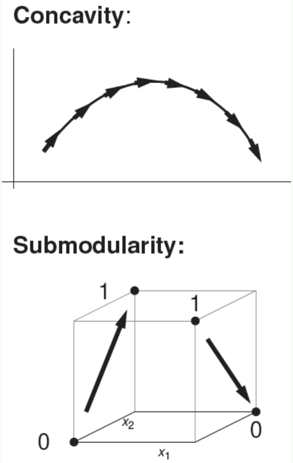
- A function $f : R^n \rightarrow R$ can be maximized efficiently, if it is concave.

Discrete analogy?
f is now a set function, or equivalently $f : 2^V \rightarrow R$ or $f : \{0, 1\}^n \rightarrow R$.

From concavity to submodularity

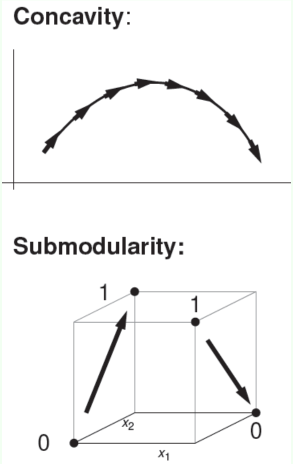


From concavity to submodularity



- $f : R \to R$ is concave, if the derivative $f'(x)$ is non-increasing in x.

From concavity to submodularity



- $f : R \to R$ is concave, if the derivative $f'(x)$ is non-increasing in x.

- $f : \{0,1\}^n \to R$ is submodular, if $\forall i$, the discrete derivative

$$\partial_i f(x) = f(x + e_i) - f(x)$$

is non-increasing in x.

Submodular

A functions $f : 2^V \to R$ is a submodular if for all $A, B \subseteq V$, we have that

Submodular

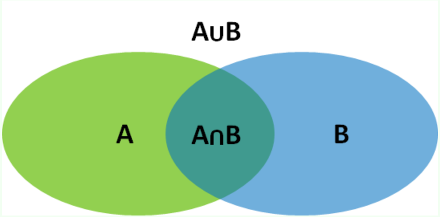
A functions $f : 2^V \to R$ is a submodular if for all $A, B \subseteq V$, we have that

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

Submodular

A functions $f : 2^V \to R$ is a submodular if for all $A, B \subseteq V$, we have that

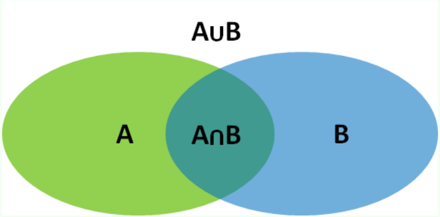
$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B).$$



Submodular

A functions $f : 2^V \to R$ is a submodular if for all $A, B \subseteq V$, we have that

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



i.e.,

$$f(A) - f(A \cap B) \geq f(A \cup B) - f(B).$$

Where do submodular functions appear?

- Algorithmic game theory:

Where do submodular functions appear?

- Algorithmic game theory:
  Submodular functions model valuation functions of agents with diminishing returns

Where do submodular functions appear?

- Algorithmic game theory:
  Submodular functions model valuation functions of agents with diminishing returns $\rightarrow$ algorithms and incentive-compatible mechanisms for problems like cost sharing, and marketing on social networks.

Where do submodular functions appear?

- Algorithmic game theory:
  Submodular functions model valuation functions of agents with diminishing returns $\rightarrow$ algorithms and incentive-compatible mechanisms for problems like cost sharing, and marketing on social networks.
- Machine learning:

Where do submodular functions appear?

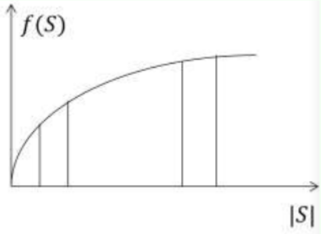
- Algorithmic game theory:
  Submodular functions model valuation functions of agents with diminishing returns $\rightarrow$ algorithms and incentive-compatible mechanisms for problems like cost sharing, and marketing on social networks.
- Machine learning:
  Submodular functions often appear as objective functions of machine learning tasks such as sensor placement, document summarization or feature selection

Where do submodular functions appear?

- Algorithmic game theory:
  Submodular functions model valuation functions of agents with diminishing returns $\rightarrow$ algorithms and incentive-compatible mechanisms for problems like cost sharing, and marketing on social networks.
- Machine learning:
  Submodular functions often appear as objective functions of machine learning tasks such as sensor placement, document summarization or feature selection $\rightarrow$ simple algorithms such as Greedy or local search work well.

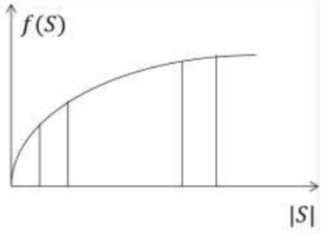
Submodularity: or equivalently

We have two equivalent definitions:



Submodularity: or equivalently

We have two equivalent definitions:

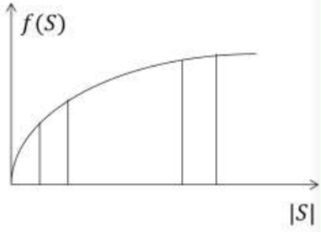


- Diminishing marginal return: for all $S \subseteq T \subseteq V$, all $v \in V \setminus T$,

  $$f(S \cup \{v\}) - f(S) \geq f(T \cup \{v\}) - f(T).$$

Submodularity: or equivalently

We have two equivalent definitions:



- Diminishing marginal return: for all $S \subseteq T \subseteq V$, all $v \in V \setminus T$,

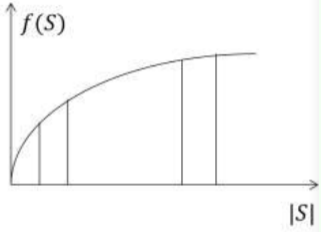
$$f(S \cup \{v\}) - f(S) \geq f(T \cup \{v\}) - f(T).$$

- Group diminishing returns: for all $S \subseteq T \subseteq V$, and $C \subseteq V \setminus T$,

$$f(S \cup C) - f(S) \geq f(T \cup C) - f(T).$$

Submodularity: or equivalently

We have two equivalent definitions:



- Diminishing marginal return: for all $S \subseteq T \subseteq V$, all $v \in V \setminus T$,

$$f(S \cup \{v\}) - f(S) \geq f(T \cup \{v\}) - f(T).$$

- Group diminishing returns: for all $S \subseteq T \subseteq V$, and $C \subseteq V \setminus T$,

$$f(S \cup C) - f(S) \geq f(T \cup C) - f(T).$$

Submodularity is the discrete analogue of concavity; in economics, known as diminishing returns.

Proof of equivalence

$f(S \cup T) + f(S \cap T) \leq f(S) + f(T) \Leftrightarrow f(S \cup \{v\}) - f(S) \geq f(T \cup \{v\}) - f(T)$.

Proof of equivalence

$f(S \cup T) + f(S \cap T) \leq f(S) + f(T) \Leftrightarrow f(S \cup \{v\}) - f(S) \geq f(T \cup \{v\}) - f(T)$.

- $\Rightarrow$: let $S \subset T$, consider two sets $S \cup \{v\}$ and T, if $v \notin T$, then
  $f(S \cup \{v\} \cup T) + f((S \cup \{v\}) \cap T) \leq f(S \cup \{v\}) + f(T)$.

Proof of equivalence

$f(S \cup T) + f(S \cap T) \leq f(S) + f(T) \Leftrightarrow f(S \cup \{v\}) - f(S) \geq f(T \cup \{v\}) - f(T)$.

- $\Rightarrow$: let $S \subset T$, consider two sets $S \cup \{v\}$ and T, if $v \notin T$, then
  $f(S \cup \{v\} \cup T) + f((S \cup \{v\}) \cap T) \leq f(S \cup \{v\}) + f(T)$.
  Note that $f(S \cup \{v\} \cup T) = f(T \cup \{v\})$ and
  $f((S \cup \{v\}) \cap T) = f(S)$.

Proof of equivalence

$f(S \cup T) + f(S \cap T) \leq f(S) + f(T) \Leftrightarrow f(S \cup \{v\}) - f(S) \geq f(T \cup \{v\}) - f(T)$.

- $\Rightarrow$: let $S \subset T$, consider two sets $S \cup \{v\}$ and T, if $v \notin T$, then
  $f(S \cup \{v\} \cup T) + f((S \cup \{v\}) \cap T) \leq f(S \cup \{v\}) + f(T)$.
  Note that $f(S \cup \{v\} \cup T) = f(T \cup \{v\})$ and $f((S \cup \{v\}) \cap T) = f(S)$.
  Thus, we have $f(S \cup \{v\}) - f(S) \geq f(T \cup \{v\}) - f(T)$.

Proof of equivalence

$f(S \cup T) + f(S \cap T) \leq f(S) + f(T) \Leftrightarrow f(S \cup \{v\}) - f(S) \geq f(T \cup \{v\}) - f(T)$.

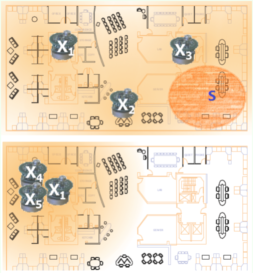
- $\Rightarrow$: let $S \subset T$, consider two sets $S \cup \{v\}$ and T, if $v \notin T$, then
  $f(S \cup \{v\} \cup T) + f((S \cup \{v\}) \cap T) \leq f(S \cup \{v\}) + f(T)$.
  Note that $f(S \cup \{v\} \cup T) = f(T \cup \{v\})$ and
  $f((S \cup \{v\}) \cap T) = f(S)$.
  Thus, we have $f(S \cup \{v\}) - f(S) \geq f(T \cup \{v\}) - f(T)$.
- $\Leftarrow$: let $T \backslash S = \{v_1, v_2, \cdots, v_k\}$, $T_j = \{v_1, v_2, \cdots, v_j\}$, $A_j = (S \cap T) \cup T_j$, and $B_j = S \cup T_j$, then we have $f(A_j \cup \{v_{j+1}\}) - f(A_j) \geq f(B_j \cup \{v_{j+1}\}) - f(B_j)$ for $j = 0, 1, 2, \cdots, k-1$.

Proof of equivalence

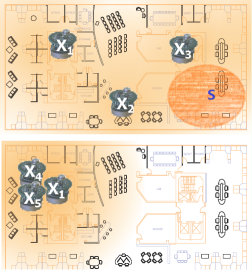
$f(S \cup T) + f(S \cap T) \leq f(S) + f(T) \Leftrightarrow f(S \cup \{v\}) - f(S) \geq f(T \cup \{v\}) - f(T)$.

- $\Rightarrow$: let $S \subset T$, consider two sets $S \cup \{v\}$ and T, if $v \notin T$, then
  $f(S \cup \{v\} \cup T) + f((S \cup \{v\}) \cap T) \leq f(S \cup \{v\}) + f(T)$.
  Note that $f(S \cup \{v\} \cup T) = f(T \cup \{v\})$ and
  $f((S \cup \{v\}) \cap T) = f(S)$.
  Thus, we have $f(S \cup \{v\}) - f(S) \geq f(T \cup \{v\}) - f(T)$.
- $\Leftarrow$: let $T \backslash S = \{v_1, v_2, \cdots, v_k\}$, $T_j = \{v_1, v_2, \cdots, v_j\}$, $A_j = (S \cap T) \cup T_j$, and $B_j = S \cup T_j$, then we have $f(A_j \cup \{v_{j+1}\}) - f(A_j) \geq f(B_j \cup \{v_{j+1}\}) - f(B_j)$ for $j = 0, 1, 2, \cdots, k-1$.
  Summing up all these equations, we have
  $f(S \cup T) + f(S \cap T) \leq f(S) + f(T)$.

Submodular example I: sensor placement

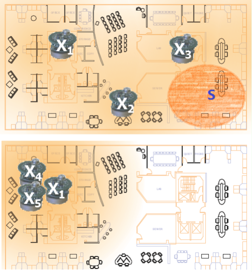


Submodular example I: sensor placement



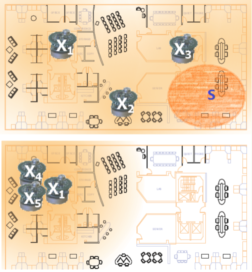
- Marginal gain: $\triangle_f(s|A) = f(A \cup \{s\}) - f(A)$ for $s \notin A$.

Submodular example I: sensor placement



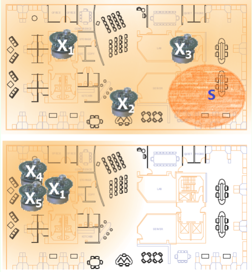
- Marginal gain: $\triangle_f(s|A) = f(A \cup \{s\}) - f(A)$ for $s \notin A$.
  - If $A = \{1, 2\}$, adding s will help a lot.

Submodular example I: sensor placement



- Marginal gain: $\triangle_f(s|A) = f(A \cup \{s\}) - f(A)$ for $s \notin A$.
  - If $A = \{1, 2\}$, adding s will help a lot.
  - If $A = \{1, 2, 3\}$, adding s does not help much.

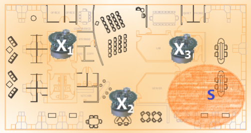
Submodular example I: sensor placement

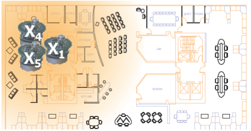


- Marginal gain: $\triangle_f(s|A) = f(A \cup \{s\}) - f(A)$ for $s \notin A$.
  - If $A = \{1, 2\}$, adding s will help a lot.
  - If $A = \{1, 2, 3\}$, adding s does not help much.
- Diminishing marginal return: $\forall A \subset B$ and $s \notin B$,

$$\begin{aligned}\triangle_f(s|A) &= f(A \cup \{s\}) - f(A) = f(\{s\} \setminus A) \\ &\geq f(\{s\} \setminus B) = f(B \cup \{s\}) - f(B)\end{aligned}$$

Submodular example I: sensor placement



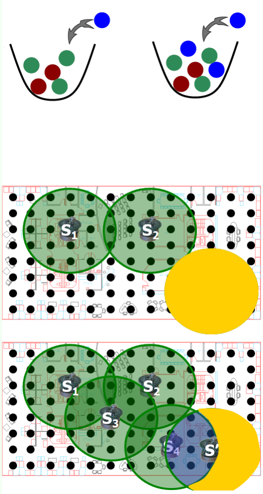


- Marginal gain: $\triangle_f(s|A) = f(A \cup \{s\}) - f(A)$ for $s \notin A$.
  - If $A = \{1, 2\}$, adding s will help a lot.
  - If $A = \{1, 2, 3\}$, adding s does not help much.
- Diminishing marginal return: $\forall A \subset B$ and $s \notin B$,

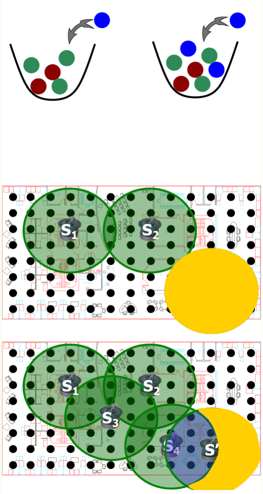
$$\begin{aligned} \triangle_f(s|A) &= f(A \cup \{s\}) - f(A) = f(\{s\} \setminus A) \\ &\geq f(\{s\} \setminus B) = f(B \cup \{s\}) - f(B) \end{aligned}$$

There are many similar applications, such as information cascade, document summarization, community detection, etc.

Submodular examples



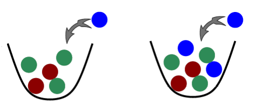
Submodular examples

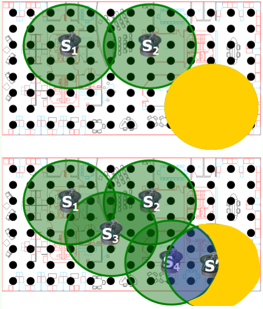


Count distinct colors

Given a set S of balls, $f(S)$ counts the number of distinct colors.

Submodular examples



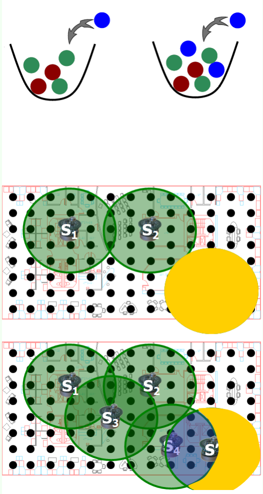


Count distinct colors

Given a set S of balls, $f(S)$ counts the number of distinct colors.

- Submodularity: incremental value of object diminishes in a larger context.

Submodular examples

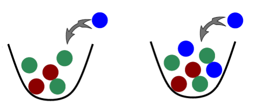


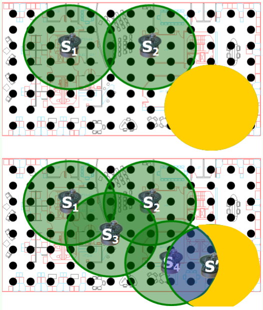
Count distinct colors

Given a set S of balls, $f(S)$ counts the number of distinct colors.

- Submodularity: incremental value of object diminishes in a larger context.
- Thus, f is a submodular.

Submodular examples





Count distinct colors

Given a set S of balls, $f(S)$ counts the number of distinct colors.

- Submodularity: incremental value of object diminishes in a larger context.
- Thus, f is a submodular.

Set covering

Assume that $A = \{S_1, S_2\}$ and $B = \{S_1, S_2, S_3, S_4\}$, then we have $f(A \cup \{S'\}) - f(A) \geq f(B \cup \{S'\}) - f(B)$.

Closedness property of submodularity

Submodularity has the closedness property under nonnegative linear combinations

Closedness property of submodularity

Submodularity has the closedness property under nonnegative linear combinations

Given a set V, let f_1 and f_2 be two submodular functions.

Closedness property of submodularity

Submodularity has the closedness property under nonnegative linear combinations

Given a set V, let f_1 and f_2 be two submodular functions.

- Let $a_1, a_2 \geq 0$, then $a_1 f_1 + a_2 f_2$ is a submodular.

Closedness property of submodularity

Submodularity has the closedness property under nonnegative linear combinations

Given a set V, let f_1 and f_2 be two submodular functions.

- Let $a_1, a_2 \geq 0$, then $a_1 f_1 + a_2 f_2$ is a submodular.
- Let $S \subset V$ be a fixed set, then $f^{'}(A) = f(A \cap S)$ and $\overline{f}(A) = f(A^c)$ are also submodulars.

Closedness property of submodularity

Submodularity has the closedness property under nonnegative linear combinations

Given a set V, let f_1 and f_2 be two submodular functions.

- Let $a_1, a_2 \geq 0$, then $a_1 f_1 + a_2 f_2$ is a submodular.
- Let $S \subset V$ be a fixed set, then $f^{'}(A) = f(A \cap S)$ and $\overline{f}(A) = f(A^c)$ are also submodulars.
- Let $S_1, S_2 \subset V$ be two fixed sets and $a_1, a_2 \geq 0$, then $f^{'}(A) = a_1 f(A \cap S_1) + a_2 f(A \cap S_2)$ is also a submodular.

Closedness property of submodularity

Submodularity has the closedness property under nonnegative linear combinations

Given a set V, let f_1 and f_2 be two submodular functions.

- Let $a_1, a_2 \geq 0$, then $a_1 f_1 + a_2 f_2$ is a submodular.
- Let $S \subset V$ be a fixed set, then $f^{'}(A) = f(A \cap S)$ and $\overline{f}(A) = f(A^c)$ are also submodulars.
- Let $S_1, S_2 \subset V$ be two fixed sets and $a_1, a_2 \geq 0$, then $f^{'}(A) = a_1 f(A \cap S_1) + a_2 f(A \cap S_2)$ is also a submodular.

Extremely useful fact

Closedness property of submodularity

Submodularity has the closedness property under nonnegative linear combinations

Given a set V, let f_1 and f_2 be two submodular functions.

- Let $a_1, a_2 \geq 0$, then $a_1 f_1 + a_2 f_2$ is a submodular.
- Let $S \subset V$ be a fixed set, then $f^{'}(A) = f(A \cap S)$ and $\overline{f}(A) = f(A^c)$ are also submodulars.
- Let $S_1, S_2 \subset V$ be two fixed sets and $a_1, a_2 \geq 0$, then $f^{'}(A) = a_1 f(A \cap S_1) + a_2 f(A \cap S_2)$ is also a submodular.

Extremely useful fact

- $f_\theta(A)$ is a submodular $\Rightarrow \sum_\theta P(\theta) f_\theta(A)$ is a submodular;

Closedness property of submodularity

Submodularity has the closedness property under nonnegative linear combinations

Given a set V, let f_1 and f_2 be two submodular functions.

- Let $a_1, a_2 \geq 0$, then $a_1 f_1 + a_2 f_2$ is a submodular.
- Let $S \subset V$ be a fixed set, then $f^{'}(A) = f(A \cap S)$ and $\overline{f}(A) = f(A^c)$ are also submodulars.
- Let $S_1, S_2 \subset V$ be two fixed sets and $a_1, a_2 \geq 0$, then $f^{'}(A) = a_1 f(A \cap S_1) + a_2 f(A \cap S_2)$ is also a submodular.

Extremely useful fact

- $f_\theta(A)$ is a submodular $\Rightarrow \sum_\theta P(\theta) f_\theta(A)$ is a submodular;
- Multicriterion optimization: $f_1, \cdots, f_m$ are submodulars, and $\lambda_i > 0 \Rightarrow \sum_i \lambda_i f_i(A)$ is a submodular;

Combinatorial optimization

There are many problems that we study in combinatorial optimization, such as min cut, max cut, max clique, vertex cover, set cover, etc.

Combinatorial optimization

There are many problems that we study in combinatorial optimization, such as min cut, max cut, max clique, vertex cover, set cover, etc.

- They are all problems in the forms

$$\arg\max_S\{f(S) : S \in \mathcal{F}\} \text{ or } \arg\min_S\{f(S) : S \in \mathcal{F}\},$$

where $\mathcal{F}$ is a discrete set of feasible solution.

Combinatorial optimization

There are many problems that we study in combinatorial optimization, such as min cut, max cut, max clique, vertex cover, set cover, etc.

- They are all problems in the forms

$$\arg\max_S \{f(S) : S \in \mathcal{F}\} \text{ or } \arg\min_S \{f(S) : S \in \mathcal{F}\},$$

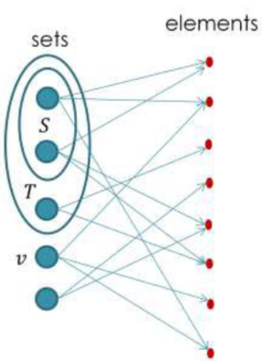
  where $\mathcal{F}$ is a discrete set of feasible solution.

- For set covering problem:

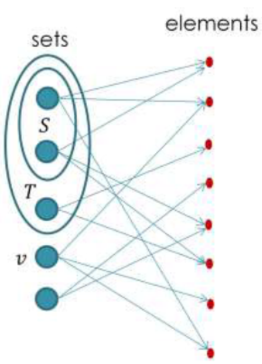
$$\begin{aligned}
&\text{minimize} \sum_{i=1}^{|S|} c_i x_i \\
&\text{s.t.} \sum_{i=1}^{|S|} x_i S_{ij} > 0, \text{ for } j = 1, 2, \cdots, |U| \\
&x_i \in \{0, 1\}
\end{aligned}$$

Outline

Set cover problem



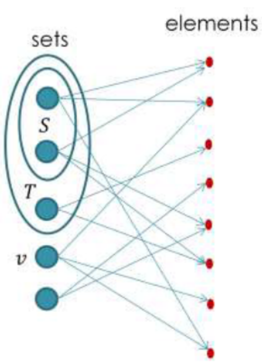
Set cover problem



Set coverage

- Each entry u is a subset of some base elements;

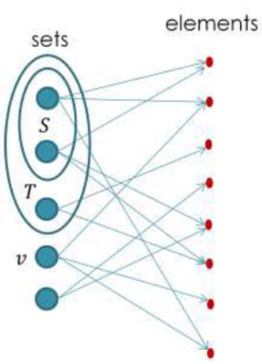
Set cover problem



Set coverage

- Each entry u is a subset of some base elements;
- Coverage $f(S) = | \cup_{u \in S} u|$;

Set cover problem



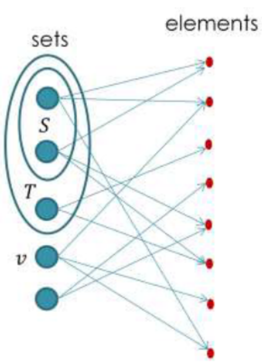
Set coverage

- Each entry u is a subset of some base elements;
- Coverage $f(S) = |\cup_{u \in S} u|$;
- $f(S \cup \{v\}) - f(S)$ is additional coverage of v on top of S.

$k-$max cover problem

- Find k subsets that maximizes their total coverage;

Set cover problem



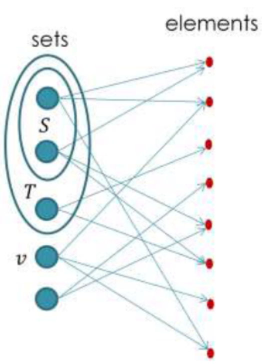
Set coverage

- Each entry u is a subset of some base elements;
- Coverage $f(S) = |\cup_{u \in S} u|$;
- $f(S \cup \{v\}) - f(S)$ is additional coverage of v on top of S.

$k-$max cover problem

- Find k subsets that maximizes their total coverage;
- It is NP-hard;

Set cover problem



Set coverage

- Each entry u is a subset of some base elements;
- Coverage $f(S) = |\cup_{u \in S} u|$;
- $f(S \cup \{v\}) - f(S)$ is additional coverage of v on top of S.

$k-$max cover problem

- Find k subsets that maximizes their total coverage;
- It is NP-hard;
- It is a special case of IM problems in IC model.

Example of 2$-$max cover

Ground set $\{a, b, c, d, e, f, g, h, i, j, k, l\}$

Example of 2−max cover

Ground set $\{a, b, c, d, e, f, g, h, i, j, k, l\}$

Subsets $A_1 = \{a, b, c, d\}, A_2 = \{e, f, g, h\}, A_3 = \{i, j, k, l\}$
$A_4 = \{a, e\}, A_5 = \{i, b, f, g\}, A_6 = \{c, d, g, h, k, l\}$
$A_7 = \{l\}$

Example of 2−max cover

Ground set $\{a, b, c, d, e, f, g, h, i, j, k, l\}$

Subsets $A_1 = \{a, b, c, d\}, A_2 = \{e, f, g, h\}, A_3 = \{i, j, k, l\}$
$A_4 = \{a, e\}, A_5 = \{i, b, f, g\}, A_6 = \{c, d, g, h, k, l\}$
$A_7 = \{l\}$

- A_6 has six elements, and A_1, A_2, A_3, A_5 have four elements;

Example of 2−max cover

Ground set $\{a, b, c, d, e, f, g, h, i, j, k, l\}$

Subsets $A_1 = \{a, b, c, d\}, A_2 = \{e, f, g, h\}, A_3 = \{i, j, k, l\}$
$A_4 = \{a, e\}, A_5 = \{i, b, f, g\}, A_6 = \{c, d, g, h, k, l\}$
$A_7 = \{l\}$

- A_6 has six elements, and A_1, A_2, A_3, A_5 have four elements;
- We find that

$$|A_1 \cup A_6| = 8, |A_2 \cup A_6| = 8$$
$$|A_3 \cup A_6| = 8, |A_5 \cup A_6| = 9.$$

Example of 2−max cover

Ground set $\{a, b, c, d, e, f, g, h, i, j, k, l\}$

Subsets $A_1 = \{a, b, c, d\}, A_2 = \{e, f, g, h\}, A_3 = \{i, j, k, l\}$
$A_4 = \{a, e\}, A_5 = \{i, b, f, g\}, A_6 = \{c, d, g, h, k, l\}$
$A_7 = \{l\}$

- A_6 has six elements, and A_1, A_2, A_3, A_5 have four elements;
- We find that

$$|A_1 \cup A_6| = 8, |A_2 \cup A_6| = 8$$
$$|A_3 \cup A_6| = 8, |A_5 \cup A_6| = 9.$$

- Collection $C = \{A_5, A_6\}$ is a 2−max cover since it covers nine elements.

Text summarization via extracting text

Definition

Given a keyword set denoted as $V = \{w_1, w_2, \cdots, w_n\}$, and sentence set $S = \{S_1, S_2, \cdots, S_m\}$, where $S_j = \{w_k | w_k \in V\}$, then text summarization is to find k sentences from C such that maximizes the coverage.

Text summarization via extracting text

Definition

Given a keyword set denoted as $V = \{w_1, w_2, \cdots, w_n\}$, and sentence set $S = \{S_1, S_2, \cdots, S_m\}$, where $S_j = \{w_k | w_k \in V\}$, then text summarization is to find k sentences from C such that maximizes the coverage.

- Let C be the set of k sentences;

Text summarization via extracting text

Definition

Given a keyword set denoted as $V = \{w_1, w_2, \cdots, w_n\}$, and sentence set $S = \{S_1, S_2, \cdots, S_m\}$, where $S_j = \{w_k | w_k \in V\}$, then text summarization is to find k sentences from C such that maximizes the coverage.

- Let C be the set of k sentences;
- Let $X_i = \begin{cases} 1, & S_i \in C \\ 0, & otherwise \end{cases}$, and $s_{ij} = \begin{cases} 1, & w_i \in S_j \\ 0, & otherwise \end{cases}$.

Text summarization via extracting text

Definition

Given a keyword set denoted as $V = \{w_1, w_2, \cdots, w_n\}$, and sentence set $S = \{S_1, S_2, \cdots, S_m\}$, where $S_j = \{w_k | w_k \in V\}$, then text summarization is to find k sentences from C such that maximizes the coverage.

- Let C be the set of k sentences;
- Let $X_i = \begin{cases} 1, & S_i \in C \\ 0, & otherwise \end{cases}$, and $s_{ij} = \begin{cases} 1, & w_i \in S_j \\ 0, & otherwise \end{cases}$.

Maximize: $\sum_{j=1}^{n} \bigvee_{i=1}^{m} X_i s_{ij}$

Text summarization via extracting text

Definition

Given a keyword set denoted as $V = \{w_1, w_2, \cdots, w_n\}$, and sentence set $S = \{S_1, S_2, \cdots, S_m\}$, where $S_j = \{w_k | w_k \in V\}$, then text summarization is to find k sentences from C such that maximizes the coverage.

- Let C be the set of k sentences;
- Let $X_i = \begin{cases} 1, & S_i \in C \\ 0, & otherwise \end{cases}$, and $s_{ij} = \begin{cases} 1, & w_i \in S_j \\ 0, & otherwise \end{cases}$.

Maximize: $\sum_{j=1}^{n} \bigvee_{i=1}^{m} X_i s_{ij}$

Subject to: $\sum_{i=1}^{m} X_i = k$

Text summarization via extracting text

Definition

Given a keyword set denoted as $V = \{w_1, w_2, \cdots, w_n\}$, and sentence set $S = \{S_1, S_2, \cdots, S_m\}$, where $S_j = \{w_k | w_k \in V\}$, then text summarization is to find k sentences from C such that maximizes the coverage.

- Let C be the set of k sentences;
- Let $X_i = \begin{cases} 1, & S_i \in C \\ 0, & otherwise \end{cases}$, and $s_{ij} = \begin{cases} 1, & w_i \in S_j \\ 0, & otherwise \end{cases}$.

$$\begin{array}{ll} \text{Maximize:} & \sum_{j=1}^{n} \bigvee_{i=1}^{m} X_i s_{ij} \\ \text{Subject to:} & \sum_{i=1}^{m} X_i = k \\ & X_i, s_{ij} \in \{0,1\} \text{ for } 1 \le i \le m \text{ and } 1 \le j \le n \end{array}$$

Submodularity of coverage

- Coverage of C can be defined as

$$f(C) = \left| \bigcup_{S_i \in C} S_i \right|.$$

Submodularity of coverage

- Coverage of C can be defined as

$$f(C) = \Big| \bigcup_{S_i \in C} S_i \Big|.$$

- Let $C \subset D$, and $S_k \in D$, we have

$$f(C \cup \{S_k\}) - f(C) = \Big| S_k - \bigcup_{S_i \in C} S_i \Big|$$

$$\geq \Big| S_k - \bigcup_{S_i \in D} S_i \Big| = f(D \cup \{S_k\}) - f(D).$$

In addition, since $\bigcup_{S_i \in C} S_i \subset \bigcup_{S_i \in D} S_i$, we therefore have

$$f(C) \leq f(D).$$

Outline

Hill-climbing algorithm

1: initialize $C = \emptyset$;
2: for $i = 1$ to k do

Hill-climbing algorithm

initialize $C = \emptyset$;
for $i = 1$ to k do
    select $c = \arg\max_{s \in S \setminus C} [f(C \cup \{s\}) - f(C)]$;

Hill-climbing algorithm

initialize $C = \emptyset$;
for $i = 1$ to k do
    select $c = \arg\max_{s \in S \setminus C} [f(C \cup \{s\}) - f(C)]$;
    $C = C \cup \{c\}$;
output C;

Hill-climbing algorithm

1: initialize $C = \emptyset$;
2: for $i = 1$ to k do
3: select $c = \arg\max_{s \in S \setminus C}[f(C \cup \{s\}) - f(C)]$;
4: $C = C \cup \{c\}$;
5: output C;

Theorem

If the set function f is monotone and submodular with $f(\emptyset) = 0$, then the greedy algorithm achieves $(1 - \frac{1}{e})$ approximation ratio,

Hill-climbing algorithm

```
initialize C = ∅;
for i = 1 to k do
    select c = arg max_{s ∈ S\C} [f(C ∪ {s}) − f(C)];
    C = C ∪ {c};
output C;
```

Theorem

If the set function f is monotone and submodular with $f(\emptyset) = 0$, then the greedy algorithm achieves $(1 - \frac{1}{e})$ approximation ratio, that is, the solution S found by the algorithm satisfies:

Hill-climbing algorithm

initialize $C = \emptyset$;
for $i = 1$ to k do
    select $c = \arg\max_{s \in S \setminus C}[f(C \cup \{s\}) - f(C)]$;
    $C = C \cup \{c\}$;
output C;

Theorem

If the set function f is monotone and submodular with $f(\emptyset) = 0$, then the greedy algorithm achieves $(1 - \frac{1}{e})$ approximation ratio, that is, the solution S found by the algorithm satisfies:

$$f(S) \geq (1 - \frac{1}{e}) \max_{S' \subseteq V, |S'| = k} f(S^{'}),$$

Hill-climbing algorithm

1: initialize $C = \emptyset$;
2: for $i = 1$ to k do
3: select $c = \arg\max_{s \in S \setminus C}[f(C \cup \{s\}) - f(C)]$;
4: $C = C \cup \{c\}$;
5: output C;

Theorem

If the set function f is monotone and submodular with $f(\emptyset) = 0$, then the greedy algorithm achieves $(1 - \frac{1}{e})$ approximation ratio, that is, the solution S found by the algorithm satisfies:

$$f(S) \geq (1 - \frac{1}{e}) \max_{S' \subseteq V, |S'| = k} f(S'),$$

where f is monotonicity if $f(S) \leq f(T)$ for all $S \subseteq T \subseteq V$.

Example of Hill-climbing algorithm: first iteration

keyword set	$W = \{w_1, w_2, \cdots, w_8\}$
sentences	$s_1 = \{w_1, w_2, w_8\}, s_2 = \{w_1, w_3, w_7\}, s_3 = \{w_1, w_6\}$
	$s_4 = \{w_1, w_3, w_7, w_8\}, s_5 = \{w_1, w_5, w_6\}, s_6 = \{w_1, w_5, w_8\}$
	$s_7 = \{w_5\}, s_8 = \{w_1, w_4, w_6\}, s_9 = \{w_2, w_8\}$

Example of Hill-climbing algorithm: first iteration

keyword set $W = \{w_1, w_2, \cdots, w_8\}$

sentences $s_1 = \{w_1, w_2, w_8\}, s_2 = \{w_1, w_3, w_7\}, s_3 = \{w_1, w_6\}$
$s_4 = \{w_1, w_3, w_7, w_8\}, s_5 = \{w_1, w_5, w_6\}, s_6 = \{w_1, w_5, w_8\}$
$s_7 = \{w_5\}, s_8 = \{w_1, w_4, w_6\}, s_9 = \{w_2, w_8\}$

Sentence	$f(C)$	$f(C \cup \{S_i\})$	$\Delta(S_i)$
s_1	0	3	3
s_2	0	3	3
s_3	0	2	2
s_4	0	4	4
s_5	0	3	3
s_6	0	3	3
s_7	0	1	1
s_8	0	3	3
s_9	0	2	2

Table: First iteration

Example of Hill-climbing algorithm: first iteration

keyword set $W = \{w_1, w_2, \cdots, w_8\}$

sentences $s_1 = \{w_1, w_2, w_8\}, s_2 = \{w_1, w_3, w_7\}, s_3 = \{w_1, w_6\}$
$s_4 = \{w_1, w_3, w_7, w_8\}, s_5 = \{w_1, w_5, w_6\}, s_6 = \{w_1, w_5, w_8\}$
$s_7 = \{w_5\}, s_8 = \{w_1, w_4, w_6\}, s_9 = \{w_2, w_8\}$

Sentence	$f(C)$	$f(C \cup \{S_i\})$	$\Delta(S_i)$
s_1	0	3	3
s_2	0	3	3
s_3	0	2	2
s_4	0	4	4
s_5	0	3	3
s_6	0	3	3
s_7	0	1	1
s_8	0	3	3
s_9	0	2	2

Table: First iteration

Sentence S_4 is selected in the first iteration since it has maximal coverage gain.

Example of Hill-climbing algorithm: second iteration

keyword set	$W = \{w_1, w_2, \cdots, w_8\}$
sentences	$s_1 = \{w_1, w_2, w_8\}, s_2 = \{w_1, w_3, w_7\}, s_3 = \{w_1, w_6\}$
	$s_5 = \{w_1, w_5, w_6\}, s_6 = \{w_1, w_5, w_8\}, s_7 = \{w_5\}$
	$s_8 = \{w_1, w_4, w_6\}, s_9 = \{w_2, w_8\}$

Example of Hill-climbing algorithm: second iteration

keyword set $W = \{w_1, w_2, \cdots, w_8\}$

sentences $s_1 = \{w_1, w_2, w_8\}, s_2 = \{w_1, w_3, w_7\}, s_3 = \{w_1, w_6\}$
$s_5 = \{w_1, w_5, w_6\}, s_6 = \{w_1, w_5, w_8\}, s_7 = \{w_5\}$
$s_8 = \{w_1, w_4, w_6\}, s_9 = \{w_2, w_8\}$

Sentence	$f(C)$	$f(C \cup \{S_i\})$	$\Delta(S_i)$
s_1	4	5	1
s_2	4	4	0
s_3	4	5	1
s_5	4	6	2
s_6	4	5	1
s_7	4	5	1
s_8	4	6	2
s_9	4	5	1

Table: The second iteration

Example of Hill-climbing algorithm: second iteration

keyword set $W = \{w_1, w_2, \cdots, w_8\}$

sentences $s_1 = \{w_1, w_2, w_8\}, s_2 = \{w_1, w_3, w_7\}, s_3 = \{w_1, w_6\}$
$s_5 = \{w_1, w_5, w_6\}, s_6 = \{w_1, w_5, w_8\}, s_7 = \{w_5\}$
$s_8 = \{w_1, w_4, w_6\}, s_9 = \{w_2, w_8\}$

Sentence	$f(C)$	$f(C \cup \{S_i\})$	$\Delta(S_i)$
s_1	4	5	1
s_2	4	4	0
s_3	4	5	1
s_5	4	6	2
s_6	4	5	1
s_7	4	5	1
s_8	4	6	2
s_9	4	5	1

Table: The second iteration

Sentence S_5 is selected in the second iteration since it has maximal coverage gain.

Example of Hill-climbing algorithm: third iteration

keyword set	$W = \{w_1, w_2, \cdots, w_8\}$
sentences	$s_1 = \{w_1, w_2, w_8\}, s_2 = \{w_1, w_3, w_7\}, s_3 = \{w_1, w_6\}$
	$s_6 = \{w_1, w_5, w_8\}, s_7 = \{w_5\}$
	$s_8 = \{w_1, w_4, w_6\}, s_9 = \{w_2, w_8\}$

Example of Hill-climbing algorithm: third iteration

keyword set	$W = \{w_1, w_2, \cdots, w_8\}$
sentences	$s_1 = \{w_1, w_2, w_8\}, s_2 = \{w_1, w_3, w_7\}, s_3 = \{w_1, w_6\}$
	$s_6 = \{w_1, w_5, w_8\}, s_7 = \{w_5\}$
	$s_8 = \{w_1, w_4, w_6\}, s_9 = \{w_2, w_8\}$

Sentence	$f(C)$	$f(C \cup \{S_i\})$	$\Delta(S_i)$
s_1	6	7	1
s_2	6	6	0
s_3	6	6	0
s_6	6	6	0
s_7	6	6	0
s_8	6	7	1
s_9	6	7	1

Table: The third iteration

Example of Hill-climbing algorithm: third iteration

keyword set $W = \{w_1, w_2, \cdots, w_8\}$

sentences $s_1 = \{w_1, w_2, w_8\}, s_2 = \{w_1, w_3, w_7\}, s_3 = \{w_1, w_6\}$

$s_6 = \{w_1, w_5, w_8\}, s_7 = \{w_5\}$

$s_8 = \{w_1, w_4, w_6\}, s_9 = \{w_2, w_8\}$

Sentence	$f(C)$	$f(C \cup \{S_i\})$	$\Delta(S_i)$
s_1	6	7	1
s_2	6	6	0
s_3	6	6	0
s_6	6	6	0
s_7	6	6	0
s_8	6	7	1
s_9	6	7	1

Table: The third iteration

Sentence S_1 is selected in the third iteration since it has maximal coverage gain.

Example of Hill-climbing algorithm: third iteration

keyword set $W = \{w_1, w_2, \cdots, w_8\}$

sentences $s_1 = \{w_1, w_2, w_8\}, s_2 = \{w_1, w_3, w_7\}, s_3 = \{w_1, w_6\}$

$s_6 = \{w_1, w_5, w_8\}, s_7 = \{w_5\}$

$s_8 = \{w_1, w_4, w_6\}, s_9 = \{w_2, w_8\}$

Sentence	$f(C)$	$f(C \cup \{S_i\})$	$\Delta(S_i)$
s_1	6	7	1
s_2	6	6	0
s_3	6	6	0
s_6	6	6	0
s_7	6	6	0
s_8	6	7	1
s_9	6	7	1

Table: The third iteration

Sentence S_1 is selected in the third iteration since it has maximal coverage gain.

Finally, it outputs text summarization $C = \{S_4, S_5, S_1\}$.

Project assignment: Only for undergraduate students

Task

Project assignment: Only for undergraduate students

Task

- Crawl a corpus, which contains some plain text in the Web under a topic;

Project assignment: Only for undergraduate students

Task

- Crawl a corpus, which contains some plain text in the Web under a topic;
- Extract keywords from them;

Project assignment: Only for undergraduate students

Task

- Crawl a corpus, which contains some plain text in the Web under a topic;
- Extract keywords from them;
- Implement an algorithm to generate 100 summarizations from the corpus;

Project assignment: Only for undergraduate students

Task

- Crawl a corpus, which contains some plain text in the Web under a topic;
- Extract keywords from them;
- Implement an algorithm to generate 100 summarizations from the corpus;

Submissions

Project assignment: Only for undergraduate students

Task

- Crawl a corpus, which contains some plain text in the Web under a topic;
- Extract keywords from them;
- Implement an algorithm to generate 100 summarizations from the corpus;

Submissions

- Crawled corpus;

Project assignment: Only for undergraduate students

Task

- Crawl a corpus, which contains some plain text in the Web under a topic;
- Extract keywords from them;
- Implement an algorithm to generate 100 summarizations from the corpus;

Submissions

- Crawled corpus;
- Source code and readme file;

Project assignment: Only for undergraduate students

Task

- Crawl a corpus, which contains some plain text in the Web under a topic;
- Extract keywords from them;
- Implement an algorithm to generate 100 summarizations from the corpus;

Submissions

- Crawled corpus;
- Source code and readme file;
- A technique report;

Project assignment: Only for undergraduate students

Task

- Crawl a corpus, which contains some plain text in the Web under a topic;
- Extract keywords from them;
- Implement an algorithm to generate 100 summarizations from the corpus;

Submissions

- Crawled corpus;
- Source code and readme file;
- A technique report;
- All data and documents submit to Baoli Gao, Email: 1760001992@qq.com;

Take-home messages

- Motivation of Submodular
- Submodular
- Set Covering Problem
  - Problem Formulation
  - Hill-climbing Algorithm