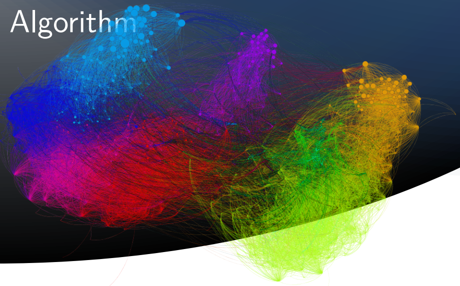
Graphs in Machine Learning

Submodularity: Theory

Definition, Properties, and Greedy Algorithm

Michal Valko

Inria & ENS Paris-Saclay, MVA



Partially based on material by: Andreas Krause, Branislav Kveton, Michael Kearns

Submodularity: modeling diminishing returns

Example: $S = \{\text{stuff}\} = \{\text{bread, apple, tomato, ...}\}$

$f(V) =$ cost of getting products V

Submodularity: modeling diminishing returns

Example: $S = \{\text{stuff}\} = \{\text{bread, apple, tomato, ...}\}$

$f(V) =$ cost of getting products V

$f(\{\text{bread}\}) = c(\text{bakery}) + c(\text{bread})$

$f(\{\text{bread}, \text{apple}\}) = c(\text{bakery}) + c(\text{bread}) + c(\text{market}) + c(\text{apple}\})$

$f(\{\text{bread}, \text{tomato}\}) = c(\text{bakery}) + c(\text{bread}) + c(\text{market}) + c(\text{tomato})$

$f(\{\text{bread}, \text{tomato}, \text{apple}\}) = c(\text{bakery}) + c(\text{bread}) + c(\text{market}) + c(\text{tomato}) + c(\text{apple})$

Submodularity: modeling diminishing returns

Example: $S = \{\text{stuff}\} = \{\text{bread, apple, tomato, ...}\}$

$f(V) = $ cost of getting products V

$f(\{\text{bread}\}) = c(\text{bakery}) + c(\text{bread})$

$f(\{\text{bread}, \text{apple}\}) = c(\text{bakery}) + c(\text{bread}) + c(\text{market}) + c(\text{apple}\})$

$f(\{\text{bread}, \text{tomato}\}) = c(\text{bakery}) + c(\text{bread}) + c(\text{market}) + c(\text{tomato})$

$f(\{\text{bread}, \text{tomato}, \text{apple}\}) = c(\text{bakery}) + c(\text{bread}) + c(\text{market}) + c(\text{tomato}) + c(\text{apple})$

Adding an apple to the smaller set costs more!

Submodularity: modeling diminishing returns

Example: $S = \{\text{stuff}\} = \{\text{bread, apple, tomato, ...}\}$

$f(V) =$ cost of getting products V

$f(\{\text{bread}\}) = c(\text{bakery}) + c(\text{bread})$
$f(\{\text{bread}, \text{apple}\}) = c(\text{bakery}) + c(\text{bread}) + c(\text{market}) + c(\text{apple}\})$
$f(\{\text{bread}, \text{tomato}\}) = c(\text{bakery}) + c(\text{bread}) + c(\text{market}) + c(\text{tomato})$
$f(\{\text{bread}, \text{tomato}, \text{apple}\}) = c(\text{bakery}) + c(\text{bread}) + c(\text{market}) + c(\text{tomato}) + c(\text{apple})$

Adding an apple to the smaller set costs more!

$$\{\text{bread}\} \subseteq \{\text{bread}, \text{tomato}\}$$

$$f(\{\text{bread}, \text{apple}\}) - f(\{\text{bread}\}) > f(\{\text{bread}, \text{tomato}, \text{apple}\}) - f(\{\text{tomato}, \text{bread}\})$$

Submodularity: modeling diminishing returns

Example: $S = \{\text{stuff}\} = \{\text{bread, apple, tomato, ...}\}$

$f(V) =$ cost of getting products V

$f(\{\text{bread}\}) = c(\text{bakery}) + c(\text{bread})$
$f(\{\text{bread, apple}\}) = c(\text{bakery}) + c(\text{bread}) + c(\text{market}) + c(\text{apple}\})$
$f(\{\text{bread}, \text{tomato}\}) = c(\text{bakery}) + c(\text{bread}) + c(\text{market}) + c(\text{tomato})$
$f(\{\text{bread}, \text{tomato}, \text{apple}\}) = c(\text{bakery}) + c(\text{bread}) + c(\text{market}) + c(\text{tomato}) + c(\text{apple})$

Adding an apple to the smaller set costs more!

$$\{\text{bread}\} \subseteq \{\text{bread}, \text{tomato}\}$$

$$f(\{\text{bread}, \text{apple}\}) - f(\{\text{bread}\}) > f(\{\text{bread}, \text{tomato}, \text{apple}\}) - f(\{\text{tomato}, \text{bread}\})$$

Diminishing returns: Buying in bulk is cheaper!

Submodularity: modeling diminishing returns

Example: $S = \{\text{stuff}\} = \{\text{bread, apple, tomato, ...}\}$

$f(V) =$ cost of getting products V

$f(\{\text{bread}\}) = c(\text{bakery}) + c(\text{bread})$

$f(\{\text{bread}, \text{apple}\}) = c(\text{bakery}) + c(\text{bread}) + c(\text{market}) + c(\text{apple}\})$

$f(\{\text{bread}, \text{tomato}\}) = c(\text{bakery}) + c(\text{bread}) + c(\text{market}) + c(\text{tomato})$

$f(\{\text{bread}, \text{tomato}, \text{apple}\}) = c(\text{bakery}) + c(\text{bread}) + c(\text{market}) + c(\text{tomato}) + c(\text{apple})$

Adding an apple to the smaller set costs more!

$$\{\text{bread}\} \subseteq \{\text{bread}, \text{tomato}\}$$

$$f(\{\text{bread}, \text{apple}\}) - f(\{\text{bread}\}) > f(\{\text{bread}, \text{tomato}, \text{apple}\}) - f(\{\text{tomato}, \text{bread}\})$$

Diminishing returns: Buying in bulk is cheaper!

A **set function** on a discrete set A is **submodular** if for any $S \subseteq T \subseteq A$ and for any $e \in A \setminus T$

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

Submodularity: Application

Link to our **product placement** problem on a **social network graph?**

Objective: Find $\arg\max_{S \subseteq A, |S| \leq k} f(S)$

Submodularity: Application

Link to our product placement problem on a social network graph?

Objective: Find $\arg\max_{S \subseteq A, |S| \leq k} f(S)$

Property: NP-hard in general

Special case: f is **nonnegative**, **submodular** and **monotone**.

Submodularity: Application

Link to our **product placement** problem on a **social network graph?**

Objective: Find $\arg\max_{S \subseteq A, |S| \leq k} f(S)$

Property: NP-hard in general

Special case: f is **nonnegative**, **submodular** and **monotone**.

http://thibaut.horel.org/submodularity/papers/nemhauser1978.pdf

Let $S^\star = \arg\max_{S \subseteq A, |S| \leq k} f(S)$ where f is monotonic and submodular set function and let $S_{\texttt{Greedy}}$ be a **greedy solution**.

$$\text{Then} \quad f(S_{\texttt{Greedy}}) \geq \left(1 - \frac{1}{e}\right) \cdot f(S^\star).$$

Submodularity: Application

Link to our **product placement** problem on a **social network graph?**

Objective: Find $\arg\max_{S \subseteq A, |S| \leq k} f(S)$

Property: NP-hard in general

Special case: f is **nonnegative**, **submodular** and **monotone**.

http://thibaut.horel.org/submodularity/papers/nemhauser1978.pdf

Let $S^\star = \arg\max_{S \subseteq A, |S| \leq k} f(S)$ where f is monotonic and submodular set function and let $S_{\texttt{Greedy}}$ be a **greedy solution**.

$$\text{Then} \quad f(S_{\texttt{Greedy}}) \geq \left(1 - \frac{1}{e}\right) \cdot f(S^\star).$$

Other applications: information, graph cuts, covering, ...

Submodularity: `Greedy` algorithm

```
Input:
    k: the maximum allowed cardinality of the output
    V: a ground set
    f: a monotone, non-negative, and submodular function
Run:
S_0 = ∅
for i = 1 to k do
    S_i ← S_{i-1} ∪ { argmax_{a ∈ V \ S_{i-1}} [f({a} ∪ S_{i-1}) − f(S_{i-1})] }
end for
Output:
    Return S_Greedy = S_k
```

Let $S^\star = \arg\max_{S \subseteq A, |S| \leq k} f(S)$ where f is monotonic and submodular set function and let $S_{\texttt{Greedy}}$ be a **greedy solution**.

$$\text{Then} \quad f(S_{\texttt{Greedy}}) \geq \left(1 - \frac{1}{e}\right) \cdot f(S^\star).$$

Submodularity: Approximation guarantee of `Greedy`

Let S_i be the i-th set selected by `Greedy`. We show

$$f\left(S^\star\right) - f\left(S_{i-1}\right) \leq f\left(S^\star \cup S_{i-1}\right) - f\left(S_{i-1}\right)$$

Submodularity: Approximation guarantee of `Greedy`

Let S_i be the i-th set selected by `Greedy`. We show

$$
\begin{aligned}
f\left(S^{\star}\right)-f\left(S_{i-1}\right) &\leq f\left(S^{\star} \cup S_{i-1}\right)-f\left(S_{i-1}\right) \\
&\leq f(a \cup S_{i-1})-f(S_{i-1})+f(S^{\star}/a \cup S_{i-1})-f(S_{i-1})
\end{aligned}
$$

Submodularity: Approximation guarantee of Greedy

Let S_i be the i-th set selected by Greedy. We show

$$
\begin{aligned}
f\left(S^\star\right) - f\left(S_{i-1}\right) &\leq f\left(S^\star \cup S_{i-1}\right) - f\left(S_{i-1}\right) \\
&\leq f(a \cup S_{i-1}) - f(S_{i-1}) + f(S^\star / a \cup S_{i-1}) - f(S_{i-1}) \\
&\leq \sum_{a \in S^\star \setminus S_{i-1}} \left(f\left(\{a\} \cup S_{i-1}\right) - f\left(S_{i-1}\right)\right)
\end{aligned}
$$

Submodularity: Approximation guarantee of `Greedy`

Let S_i be the i-th set selected by `Greedy`. We show

$$
\begin{aligned}
& f\left(S^{\star}\right)-f\left(S_{i-1}\right) \leq f\left(S^{\star} \cup S_{i-1}\right)-f\left(S_{i-1}\right) \\
& \leq f(a \cup S_{i-1})-f(S_{i-1})+f(S^{\star}/a \cup S_{i-1})-f(S_{i-1}) \\
& \leq \sum_{a \in S^{\star} \backslash S_{i-1}}\left(f\left(\{a\} \cup S_{i-1}\right)-f\left(S_{i-1}\right)\right) \\
& \leq \sum_{a \in S^{\star} \backslash S_{i-1}}\left(f\left(S_i\right)-f\left(S_{i-1}\right)\right) \leq k\left(f\left(S_i\right)-f\left(S_{i-1}\right)\right)
\end{aligned}
$$

Submodularity: Approximation guarantee of `Greedy`

Let S_i be the i-th set selected by `Greedy`. We show

$$
\begin{aligned}
f\left(S^\star\right) - f\left(S_{i-1}\right) &\le f\left(S^\star \cup S_{i-1}\right) - f\left(S_{i-1}\right) \\
&\le f(a \cup S_{i-1}) - f(S_{i-1}) + f(S^\star / a \cup S_{i-1}) - f(S_{i-1}) \\
&\le \sum_{a \in S^\star \setminus S_{i-1}} \left(f\left(\{a\} \cup S_{i-1}\right) - f\left(S_{i-1}\right)\right) \\
&\le \sum_{a \in S^\star \setminus S_{i-1}} \left(f\left(S_i\right) - f\left(S_{i-1}\right)\right) \le k\left(f\left(S_i\right) - f\left(S_{i-1}\right)\right)
\end{aligned}
$$

Difference from the optimum of $S_{\texttt{Greedy}} = S_k$ after the k-th step

...

$$
\begin{aligned}
f(S^\star) - f(S_k) &= f(S^\star) - f(S_{k-1}) - \left(f\left(S_k\right) - f\left(S_{k-1}\right)\right) \\
&\le f(S^\star) - f(S_{k-1}) - \frac{f(S^\star) - f(S_{k-1})}{k} \\
&\le \left(1 - \tfrac{1}{k}\right) \cdot \left(f(S^\star) - f(S_{k-1})\right) \le \left(1 - \tfrac{1}{k}\right)^k \cdot f(S^\star)
\end{aligned}
$$

Submodularity: Graph-related examples

- Influence maximization on networks (current example)

Submodularity: Graph-related examples

- Influence maximization on networks (current example)
- Maximum-weight spanning trees

Submodularity: Graph-related examples

- Influence maximization on networks (current example)
- Maximum-weight spanning trees
- Graph cuts

Submodularity: Graph-related examples

- Influence maximization on networks (current example)
- Maximum-weight spanning trees
- Graph cuts
- Structure learning in graphical models (PGM course)

Submodularity: Graph-related examples

- Influence maximization on networks (current example)
- Maximum-weight spanning trees
- Graph cuts
- Structure learning in graphical models (PGM course)
- More examples http://people.math.gatech.edu/~tetali/LINKS/IWATA/SFGT.pdf

Submodularity: Graph-related examples

- Influence maximization on networks (current example)
- Maximum-weight spanning trees
- Graph cuts
- Structure learning in graphical models (PGM course)
- More examples http://people.math.gatech.edu/~tetali/LINKS/IWATA/SFGT.pdf
- Deep Submodular Functions (2017) https://arxiv.org/pdf/1701.08939.pdf

Submodularity: Graph-related examples

- Influence maximization on networks (current example)
- Maximum-weight spanning trees
- Graph cuts
- Structure learning in graphical models (PGM course)
- More examples http://people.math.gatech.edu/~tetali/LINKS/IWATA/SFGT.pdf
- Deep Submodular Functions (2017) https://arxiv.org/pdf/1701.08939.pdf

Submodularity: Graph-related examples

- Influence maximization on networks (current example)
- Maximum-weight spanning trees
- Graph cuts
- Structure learning in graphical models (PGM course)
- More examples http://people.math.gatech.edu/~tetali/LINKS/IWATA/SFGT.pdf
- Deep Submodular Functions (2017) https://arxiv.org/pdf/1701.08939.pdf

back to the influence-maximization example ...

Michal Valko

michal.valko@inria.fr

Inria & ENS Paris-Saclay, MVA

https://misovalko.github.io/mva-ml-graphs.html