Submodular Functions: Finding influencers in networks and Detecting disease outbreaks

CS345a: Data Mining
Jure Leskovec and Anand Rajaraman
Stanford University
Feature selection:
- Given a set of features $X_1, \ldots, X_n$
- Want to predict $Y$ from a subset $A = (X_{i1}, \ldots, X_{ik})$
- What are the $k$ most informative features?

Active learning:
- Want to predict medical condition
- Each test has a cost (but also reveals information)
- Which tests should we perform to make most effective decisions?
Motivation (2)

- **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?
Given:
- finite set V
- A function $F: 2^V \rightarrow \mathbb{R}$

Want:
- $A^* = \arg\max_A F(A)$
- s.t. some constraints on A

For example:
- Influence maximization: $V = \ldots$ $F(A) = \ldots$
- Sensor placement: $V = \ldots$ $F(A) = \ldots$
- Feature selection: $V = \ldots$ $F(A) = \ldots$
Example: Feature selection

- Given random variables $Y, X_1, \ldots, X_n$
- Want to predict $Y$ from subset $A = (X_{i1}, \ldots, X_{ik})$

Naive Bayes Model:
$$P(Y, X_1, \ldots, X_n) = P(Y) \prod_i P(X_i | Y)$$

- Want $k$ most informative features:

$$A^* = \text{argmax } I(A; Y) \text{ s.t. } |A| \leq k$$

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

Uncertainty before knowing $A$

Uncertainty after knowing $A$
Given: finite set $V$ of features, utility function $F(A) = I(A; Y)$

Want: $A^* \subseteq V$ such that
$$A^* = \arg\max_{|A| \leq k} F(A)$$

Typically NP-hard!

Greedy hill-climbing:

Start with $A_0 = \{\}$
For $i = 1$ to $k$

$$s^* = \arg\max_s F(A \cup \{s\})$$

$$A_i = A_{i-1} \cup \{s^*\}$$

How well does this simple heuristic do?
Approximation guarantee

- Greedy hill climbing produces a solution $A$ where $F(A) \geq (1-1/e)$ of optimal value ($\approx 63\%$) [Hemhauser, Fisher, Wolsey ’78]

- Claim holds for functions $F$ with 2 properties:
  - $F$ is monotone:
    
    if $A \subseteq B$ then $F(A) \leq F(B)$ and $F(\emptyset) = 0$

  - $F$ is submodular:
    adding element to a set gives less improvement than adding to one of subsets
Definition:
- Set function $F$ on $V$ is called submodular if:
  For all $A, B \subseteq V$:
  $$F(A) + F(B) \geq F(A \cup B) + F(A \cap B)$$
Submodularity: Or equivalently

- **Diminishing returns characterization**
  
  **Definition:**
  
  Set function $F$ on $V$ is called **submodular** if:
  
  For all $A \subseteq B$, $s \notin B$:
  
  $\frac{F(A \cup \{s\}) - F(A)}{\text{Gain of adding } s \text{ to a small set}} \geq \frac{F(B \cup \{s\}) - F(B)}{\text{Gain of adding } s \text{ to a large set}}$

  
  ![Diagram](3/9/2010 Jure Leskovec & Anand Rajaraman, Stanford CS345a: Data Mining 9)
Example: Feature selection

- Given random variables $X_1, \ldots, X_n$
- **Mutual information:**
  \[
  F(A) = I(A; V \setminus A) = H(V \setminus A) - H(V \setminus A | A) \\
  = \sum_{y,A} P(A) [\log P(y|A) - \log P(y)] \\
  H(C|D) = H(C,D) - H(D)
  \]
- **Mutual information $F(A)$ is submodular**
  [Krause-Guestrin ‘05]
  \[
  F(A \cup \{s\}) - F(A) = H(s|A) - H(s| V \setminus (A \cup \{s\}))
  \]
  - $A \subseteq B \Rightarrow H(s|A) \geq H(s|B)$
  - “Information never hurts”
Example: Feature selection (2)

- Let $Y = \sum_i \alpha_i X_i + \varepsilon$, where $(X_1, \ldots, X_n, \varepsilon) \sim N(\cdot; \mu, \Sigma)$
- Want to pick a subset $A$ to predict $Y$
- $\text{Var}(Y | X_A = x_A)$: conditional var. of $Y$ given $X_A = x_A$
- Expected variance:
  \[ \text{Var}(Y | X_A) = \int p(x_A) \text{Var}(Y | X_A = x_A) \, dx_A \]
- Variance reduction:
  \[ F_{\text{V}}(A) = \text{Var}(Y) - \text{Var}(Y | X_A) \]
- Then [Das-Kempe 08]:
  - $F_{\text{V}}(A)$ is monotonic
  - $F_{\text{V}}(A)$ is submodular*
- Orthogonal matching pursuit [Tropp-Donoho] near optimal!

*under some conditions on $\Sigma$
Closedness properties

- $F_1, \ldots, F_m$ submodular functions on $V$ and $\lambda_1, \ldots, \lambda_m > 0$
- Then: $F(A) = \sum_i \lambda_i F_i(A)$ is submodular!
- Submodularity closed under nonnegative linear combinations
- Extremely useful fact:
  - $F_\theta(A)$ submodular $\Rightarrow \sum_\theta P(\theta) F_\theta(A)$ submodular!
  - Multicriterion optimization:
    - $F_1, \ldots, F_m$ submodular, $\lambda_i > 0$ $\Rightarrow \sum_i \lambda_i F_i(A)$ submodular
Example: Set cover

- Each element covers some area
- Observation: Diminishing returns

\[ \text{New element: } S' \]

\[ \text{Adding } S' \text{ helps a lot} \]

\[ A = \{S_1, S_2\} \]

\[ B = \{S_1, S_2, S_3, S_4\} \]

\[ \text{Adding } S' \text{ helps very little} \]
Example: Set cover

- **F is submodular:** \( A \subseteq B \)

\[
F(A \cup \{s\}) - F(A) \geq F(B \cup \{s\}) - F(B)
\]

Gain of adding a set \( s \) to a small solution

Gain of adding a set \( s \) to a large solution

- **Natural example:**
  - Sets \( s_1, s_2, ..., s_n \)
  - \( F(A) = \text{size of union of } s_i \)
    (size of covered area)
- Most influential set of size $k$: set $S$ of $k$ nodes producing largest expected cascade size $F(S)$ if activated [Domingos-Richardson ‘01]

- Optimization problem: $\max_{S \text{ of size } k} F(S)$
Fix outcome $i$ of coin flips
Let $F_i(S)$ be size of cascade from $S$ given these coin flips

- Let $F_i(v)$ = set of nodes reachable from $v$ on live-edge paths
- $F_i(S) = $ size of union $F_i(v) \rightarrow F_i$ is submodular
- $F= \sum F_i \rightarrow F$ is submodular [Kempe-Kleinberg-Tardos ‘03]
Example 2: Water Network

- Given a real city water distribution network
- And data on how contaminants spread in the network
- Problem posed by US Environmental Protection Agency
Water Network

- Real metropolitan area water network:
  - $V = 21,000$ nodes
  - $E = 25,000$ pipes

- Water flow simulator provided by EPA
- 3.6 million contamination events
- Multiple objectives:
  - Detection time, affected population, ...
- Place sensors that detect well “on average”
Water Network: Utility

- Utility of placing sensors
  - Water flow dynamics, demands of households, ...
  - For each subset $A \subseteq V$ compute utility $F(A)$

- Model predicts impact location
  - High impact
  - Medium impact
  - Low impact

- Set $V$ of all network junctions
  - High sensing quality $F(A) = 0.9$
  - Low sensing quality $F(A) = 0.01$

Sensor reduces impact through early detection!
Optimization problem

- **Given:**
  - Graph G(V,E), budget B
  - Data on how outbreaks o₁, ..., oᵢ, ..., oₖ spread over time

- **Select a set of nodes A maximizing the reward**

\[
\max_{A \subseteq V} \sum_{i} \text{Prob}(i) R_i(A)
\]

subject to \( cost(A) \leq B \)
Two parts to the problem

- **Cost:**
  - Cost of monitoring is node dependent

- **Reward:**
  - Minimize the number of affected nodes:
    - If $A$ are the monitored nodes, let $R(A)$ denote the number of nodes we save
**Reward function is submodular**

- **Claim:** [Krause et al. ’08]
  - Reward function is submodular

- **Consider cascade** $i$:
  - $R_i(u_k) =$ set of nodes saved from $u_k$
  - $R_i(A) =$ size of union $R_i(u_k)$, $u_k \in A$
  \[ \Rightarrow R_i \text{ is submodular} \]

- **Global optimization:**
  - $R(A) = \sum \text{Prob}(i) R_i(A)$
  \[ \Rightarrow R \text{ is submodular} \]
Solution quality: Nemhauser

(1-1/e) bound quite loose... can we get better bounds?
Solution quality: Better estimate

- Suppose $A$ is some solution to
  
  $\text{argmax}_A F(A)$ s.t. $|A| \leq k$

  and $A^* = \{s_1, \ldots, s_k\}$ is OPT solution

- Then:
  
  $F(A^*) \leq F(A \cup \{s\})$

  $\Rightarrow F(A) + \sum_{i=1}^k \delta_i$

  $\delta_i = F(A \cup \{s_i\}) - F(A)$

  $\delta_1 \geq \delta_2 \geq \ldots \geq \delta_n$

  $F(A^*) \leq F(A) + \sum_{i=1}^k \delta_i$

  $F(A) + \sum_{i=1}^k \delta_i$

  $F(A) + \sum_{i=1}^k \delta_i$
Submodularity gives **data-dependent** bounds on the performance of any algorithm.
13 participants

Performance measured in 30 different criteria

- G: Genetic algorithm
- D: Domain knowledge
- H: Other heuristic
- E: “Exact” method (MIP)

24% better performance than runner-up!
What was the trick?

- Simulated 3.6M contaminations on 40 machines for 2 weeks [Krause et al. ‘08]
  - 152 GB of simulation data
  - 16GB in RAM (compressed)

- Very accurate computation of F(A)

- Very slow evaluation of F(A):
  - Would take 6 weeks for all 30 settings
Hill-climbing algorithm is slow:
- At each iteration we need to re-evaluate gains of all sensors
- It scales as $O(n \cdot k)$

Add element with highest marginal gain

---

Hill-climbing

- reward

Add element with highest marginal gain
In round $i+1$:

- have so far picked $A_i = \{s_1, ..., s_i\}$
- pick $s_{i+1} = \arg\max_s F(A_i \cup \{s\}) - F(A_i)$
  i.e., maximize “marginal benefit” $\delta_s(A_i)$
  $$\delta_s(A_i) = F(A_i \cup \{s\}) - F(A_i)$$

**Observation:** Submodularity implies
$$i \leq j \implies \delta_s(A_i) \geq \delta_s(A_j)$$
$$A_i \subseteq A_j$$

Marginal benefits $\delta_s$ never increase!
Lazy hill climbing algorithm:

- First iteration as usual
- Keep an ordered list of marginal benefits $\delta_i$ from previous iteration
- Re-evaluate $\delta_i$ only for top element
- If $\delta_i$ stays on top, use it, otherwise re-sort
- Using “lazy evaluations” [Krause et al. ‘08]
  - 1 hour/20 sensors
  - Done in 2 days!
Non-constant cost functions

- For each \( s \in V \), let \( c(s) > 0 \) be its cost (e.g., feature acquisition costs, ...)
- Cost of a set \( C(A) = \sum_{s \in A} c(s) \) (modular function)
- Want to solve:
  \[
  A^* = \arg\max_A F(A) \text{ s.t. } C(A) \leq B \text{ (budget)}
  \]
- Cost-benefit greedy algorithm:
  Start with \( A = \{\} \)
  While there is an \( s \in V \setminus A \) s.t. \( C(A \cup \{s\}) \leq B \)
  \[
  s^* = \arg\max_{s: C(A \cup \{s\}) \leq B} \frac{F(A \cup \{s\}) - F(A)}{c(s)}
  \]
  \( A = A \cup \{s^*\} \)
Consider the following problem:

\[
\max_A F(A) \text{ s.t. } C(A) \leq 1
\]

Cost-benefit greedy picks \(a\).
Then cannot afford \(b\!\).

→ Cost-benefit greedy performs arbitrarily badly!

<table>
<thead>
<tr>
<th>Set A</th>
<th>F(A)</th>
<th>C(A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>{a}</td>
<td>2\varepsilon</td>
<td>\varepsilon</td>
</tr>
<tr>
<td>{b}</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Cost-benefit optimization

- **Theorem** [Leskovec-Krause et al. ‘07]:
  - $A_{CB}$: cost-benefit greedy solution and
  - $A_{UC}$: unit-cost greedy solution (i.e., ignore costs)

  Then:
  \[ \max \{ F(A_{CB}), F(A_{UC}) \} \geq \frac{1}{2} (1-1/e) \text{OPT} \]

- Can still compute **online bounds** and speed up using **lazy evaluations**

- **Note:** Can also get
  - $(1-1/e)$ approximation in time $O(n^4)$ [Sviridenko ’04]
  - Slightly better than $\frac{1}{2}(1-1/e)$ in $O(n^2)$ [Wolsey ‘82]
Question...

How Many Blogs Does the World Need?
By Michael Kinsley

I have 10 minutes. Which blogs should I read to be most up to date?
[Leskovec-Krause et al. ‘07]

Who are the most influential bloggers?

Thursday, Nov. 20, 2008
Detecting information outbreaks

Want to read things **before** others do.

Detect **blue** & **yellow** soon but miss **red**.

Detect all stories but **late**.
Performance on Blog selection

- Submodular formulation outperforms heuristics
- 700x speedup using lazy evaluations

blog selection ~45k blogs
Taking “attention” into account

- Naïve approach: Just pick 10 best blogs
- Selects big, well known blogs (Instapundit, etc.)
- These contain many posts, take long to read!

![Diagram showing cost/benefit analysis and ignoring cost over number of posts (time) allowed]
Maximization of submodular functions:
- NP hard
- But can use greedy hill climbing to get ~63% of OPT

Minimization of submodular functions:
- Polynomial time solvable
- Best known algorithm: $\Omega(n^5)$ function evaluations
Super- and Sub-modularity

- Set function $F$ on $V$ is called **submodular** if
  1) For all $A, B \subseteq V$:
     \[ F(A) + F(B) \geq F(A \cup B) + F(A \cap B) \]
  \[ \iff \]
  2) For all $A \subseteq B$, $s \notin B$:
     \[ F(A \cup \{s\}) - F(A) \geq F(B \cup \{s\}) - F(B) \]

- $F$ is called **supermodular** if $-F$ is submodular
- $F$ is called **modular** if $F$ is both sub- and supermodular:
  - E.g., for modular ("additive") $F$
     \[ F(A) = \sum_{i \in A} w(i) \]
Other settings:

- Optimize the worst case: 
  \[ A^* = \arg\max_{|A| \leq k} \min_i F_i(A) \]
  - [Krause et al. ’07]

- Online maximization of submodular functions:
  - [Golovin-Streeter ’08]

Pick sets: 
- \( A_1 \) 
- \( A_2 \) 
- \( A_3 \) 
- ... 
- \( A_T \)

SFs: 
- \( F_1 \) 
- \( F_2 \) 
- \( F_3 \) 
- ... 
- \( F_T \)

Reward: 
- \( r_1 = F_1(A_1) \) 
- \( r_2 \) 
- \( r_3 \) 
- ... 
- \( r_T \)

Total: \( \sum_t r_t \rightarrow \max \)

Time
Most of the slides borrowed from Andreas Krause

http://www.blogcascades.org
http://www.submodularity.org