Optimizing Submodular Functions

CS246: Mining Massive Datasets
Jure Leskovec, Stanford University
http://cs246.stanford.edu
Announcement: Final Exam Logistics
Final: Logistics

- **Date:**
  - Tuesday, March 19, 3:30-6:30 PM PDT

- **Location:**
  - if SUNetID[0] in ['a', .. 'l'] then 420-040
  - if SUNetID[0] in ['m', .. 'z'] then Bishop Auditorium

- **Alternate Date:**
  - Monday, March 18, 6:30-9:30 PM PDT

- **Location:**
  - Gates 104
  - There is still SOME SPACE LEFT!

- **TAs will NOT answer questions during the final**
You may come to Stanford to take the exam, or...

- **Date:**
  - From Tue, Mar 19, 3:30 PM to Wed, Mar 20, 3:30 PM (all hours in PDT)
  - **Agree with your exam monitor** on the most convenient 3-hour slot in that window of time
- **Exam monitors will receive an email from SCPD with the final exam,** which they will in turn forward to you right before the beginning of your 3-hour slot
- Once you completed the exam, make sure to send the file **back to your exam monitor** (high-quality scanned copy)
- **Exam monitors will NOT answer questions during the final**
Final: Instructions

- Final exam is open book and open notes
- A calculator or computer is REQUIRED
  - You may only use your computer to do arithmetic calculations (i.e., the buttons found on a standard scientific calculator)
  - You may also use your computer to read course notes or the textbook
  - But no Internet/Google/Python access is allowed
- Practice finals are posted on Piazza!
- We recommend bringing a power strip
Optimizing Submodular Functions
Recommendations: Diversity

- Redundancy leads to a bad user experience

  Obama Calls for Broad Action on Guns

  Obama unveils 23 executive actions, calls for assault weapons ban

  Obama seeks assault weapons ban, background checks on all gun sales

- Uncertainty around information need => don’t put all eggs in one basket

- How do we optimize for diversity directly?
Monday, January 14, 2013

France intervenes
Chuck for Defense
Argo wins big
Hagel expects fight
Covering the day’s news

Monday, January 14, 2013

France intervenes
Chuck for Defense
Argo wins big
New gun proposals
Encode Diversity as Coverage

- **Idea:** Encode diversity as coverage problem
- **Example:** Word cloud of news for a single day
  - Want to select articles so that most words are “covered”
Diversity as Coverage
What is being covered?

- **Q:** What is being covered?
- **A:** **Concepts** (In our case: Named entities)

France  Mali  Hagel  Pentagon  Obama  Romney  Zero Dark Thirty  Argo  NFL

Hagel expects fight

- **Q:** Who is doing the covering?
- **A:** **Documents**
Suppose we are given a set of documents $D$
- Each document $d$ covers a set $X_d$ of words/topics/named entities $W$
- For a set of documents $A \subseteq D$ we define

$$F(A) = \left| \bigcup_{i \in A} X_i \right|$$

Goal: We want to

$$\max_{|A| \leq k} F(A)$$

Note: $F(A)$ is a set function: $F(A): \text{Sets} \to \mathbb{N}$
Maximum Coverage Problem

- Given universe of elements $W = \{w_1, \ldots, w_n\}$ and sets $X_1, \ldots, X_m \subseteq W$

- Goal: Find $k$ sets $X_i$ that cover the most of $W$
  - More precisely: Find $k$ sets $X_i$ whose size of the union is the largest
  - Bad news: A known NP-complete problem
Simple Greedy Heuristic

Simple Heuristic: Greedy Algorithm:
- Start with \( A_0 = \{ \} \)
- For \( i = 1 \ldots k \)
  - Find set \( d \) that \( \max F(A_{i-1} \cup \{d\}) \)
  - Let \( A_i = A_{i-1} \cup \{d\} \)

- Example:
  - Eval. \( F(\{d_1\}), \ldots, F(\{d_m\}) \), pick best (say \( d_1 \))
  - Eval. \( F(\{d_1\} \cup \{d_2\}), \ldots, F(\{d_1\} \cup \{d_m\}) \), pick best (say \( d_2 \))
  - Eval. \( F(\{d_1, d_2\} \cup \{d_3\}), \ldots, F(\{d_1, d_2\} \cup \{d_m\}) \), pick best
  - And so on...

\[ F(A) = \bigcup_{d \in A} X_d \]
Simple Greedy Heuristic

- Goal: Maximize the covered area
Simple Greedy Heuristic

- Goal: Maximize the covered area
Simple Greedy Heuristic

- Goal: Maximize the covered area
Simple Greedy Heuristic

- Goal: Maximize the covered area
Simple Greedy Heuristic

- **Goal:** Maximize the covered area
Goal: Maximize the size of the covered area
Greedy first picks A and then C
But the optimal way would be to pick B and C
**Approximation Guarantee**

- **Greedy** produces a solution $A$ where: $F(A) \geq (1-1/e)\cdot OPT$  \hspace{1cm} ($F(A) \geq 0.63\cdot OPT$)

  [Nemhauser, Fisher, Wolsey ’78]

- **Claim holds for functions $F(\cdot)$ with 2 properties:**
  - **$F$ is monotone:** (adding more docs doesn’t decrease coverage)
    if $A \subseteq B$ then $F(A) \leq F(B)$ and $F(\{\})=0$
  - **$F$ is submodular:**
    adding an element to a set gives less improvement than adding it to one of its subsets
**Submodularity: Definition**

**Definition:**
- Set function $F(\cdot)$ is called **submodular** if:
  - For all $A,B \subseteq W$:
    $$F(A) + F(B) \geq F(A \cup B) + F(A \cap B)$$
Submodularity: Or equivalently

- **Diminishing returns characterization**

Equivalent definition:
- Set function $F(\cdot)$ is called **submodular** if:

  For all $A \subseteq B$:

  $$F(A \cup \{d\}) - F(A) \geq F(B \cup \{d\}) - F(B)$$

  Gain of adding $d$ to a small set ≥ Gain of adding $d$ to a large set

  + • $d$  
  + • $d$  

  Large improvement  
  Small improvement
Example: Set Cover

- $F(\cdot)$ is submodular: $A \subseteq B$

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

Gain of adding $d$ to a small set

Gain of adding $d$ to a large set

- **Natural example:**
  - Sets $d_1, \ldots, d_m$
  - $F(A) = |\bigcup_{i \in A} d_i|$ (size of the covered area)

- **Claim:**
  - $F(A)$ is submodular!
Submodularity—Diminishing returns

- Submodularity is discrete analogue of concavity

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

Gain of adding $X_d$ to a small set \( \geq \) Gain of adding $X_d$ to a large set

Adding $d$ to $B$ helps less than adding it to $A$!
Submodularity & Concavity

- **Marginal gain:**
  \[ \Delta_F(d|A) = F(A \cup \{d\}) - F(A) \]

- **Submodular:**
  \[ F(A \cup \{d\}) - F(A) \geq F(B \cup \{d\}) - F(B) \] \( A \subseteq B \)

- **Concavity:**
  \[ f(a + d) - f(a) \geq f(b + d) - f(b) \] \( a \leq b \)
Submodularity: Useful Fact

- Let $F_1 \ldots F_m$ be submodular and $\lambda_1 \ldots \lambda_m > 0$ then $F(A) = \sum_{i=1}^{m} \lambda_i F_i(A)$ is submodular.
  - Submodularity is closed under non-negative linear combinations!

- This is an extremely useful fact:
  - Average of submodular functions is submodular:
    $F(A) = \sum_i P(i) \cdot F_i(A)$
  - Multicriterion optimization: $F(A) = \sum_i \lambda_i F_i(A)$
Back to our problem

- **Q:** What is being covered?
- **A:** Concepts (In our case: Named entities)

![Diagram showing relationships between concepts and documents]

**Hagel expects fight**

- **Q:** Who is doing the covering?
- **A:** Documents
Objective: pick \( k \) docs that cover most concepts

\[ F(A) : \text{the number of concepts covered by } A \]

- Elements...concepts, Sets ... concepts in docs
- \( F(A) \) is submodular and monotone!
- We can use greedy algorithm to optimize \( F \)
Objective: pick $k$ docs that cover most concepts

- France
- Mali
- Hagel
- Pentagon
- Obama
- Romney
- Zero Dark Thirty
- Argo
- NFL

Enthusiasm for Inauguration wanes

Inauguration weekend

The good:
- Penalizes redundancy
- Submodular

The bad:
- Concept importance?
- All-or-nothing too harsh
Probabilistic Set Cover
**Objective:** pick $k$ docs that cover most concepts

- Each concept $c$ has importance weight $w_c$
All-or-nothing too harsh

- **Document coverage function**

\[ \text{cover}_d(c) = \text{probability document } d \text{ covers concept } c \]

[e.g., how strongly \(d\) covers \(c\)]

Obama  Romney

Enthusiasm for Inauguration wanes
Probabilistic Set Cover

- **Document coverage function:**
  \[
  \text{cover}_d(c) = \text{probability document } d \text{ covers concept } c
  \]
  - \( \text{Cover}_d(c) \) can also model how relevant is concept \( c \) for user \( u \)

- **Set coverage function:**
  \[
  \text{cover}_A(c) = 1 - \prod_{d \in A} (1 - \text{cover}_d(c))
  \]
  - Prob. that at least one document in \( A \) covers \( c \)

- **Objective:**
  \[
  \max_{A: |A| \leq k} F(A) = \sum_{c} w_c \text{ cover}_A(c)
  \]
The objective function is also submodular

- Intuitively, it has a diminishing returns property
- Greedy algorithm leads to a \((1 - \frac{1}{e}) \approx 63\%\) approximation, i.e., a near-optimal solution
Objective: pick $k$ docs that cover most concepts

- France
- Mali
- Hagel
- Pentagon
- Obama
- Romney
- Zero Dark Thirty
- Argo
- NFL

Enthusiasm for Inauguration wanes

Inauguration weekend

- Each concept $c$ has importance weight $w_c$
- Documents partially cover concepts: $\text{cover}_d(c)$
Lazy Optimization of Submodular Functions
Submodular Functions

Greedy

Marginal gain: 
\[ F(A \cup x) - F(A) \]

- **Greedy algorithm is slow!**
  - At each iteration we need to re-evaluate marginal gains of all remaining documents
  - Runtime \( O(|D| \cdot K) \) for selecting \( K \) documents out of the set of \( D \) of them

Add document with highest marginal gain
Speeding up Greedy

- **In round $i$:** So far we have $A_{i-1} = \{d_1, \ldots, d_{i-1}\}$
  - Now we pick $d_i = \arg \max_{d \in V} F(A_{i-1} \cup \{d\}) - F(A_{i-1})$
  - Greedy algorithm maximizes the “marginal benefit”
    $$\Delta_i(d) = F(A_{i-1} \cup \{d\}) - F(A_{i-1})$$

- **By submodularity property:**
  $$F(A_i \cup \{d\}) - F(A_i) \geq F(A_j \cup \{d\}) - F(A_j) \text{ for } i < j$$

- **Observation:** By submodularity:
  For every $d \in D$
  $$\Delta_i(d) \geq \Delta_j(d) \text{ for } i < j \text{ since } A_i \subseteq A_j$$

- **Marginal benefits $\Delta_i(d)$ only shrink!**
  (as $i$ grows)

Selecting document $d$ in step $i$ covers more words than selecting $d$ at step $j$ ($j > i$)
Lazy Greedy

- **Idea:**
  - Use $\Delta_i$ as upper-bound on $\Delta_j$ ($j > i$)

- **Lazy Greedy:**
  - Keep an ordered list of marginal benefits $\Delta_i$ from previous iteration
  - Re-evaluate $\Delta_i$ only for top element
  - Re-sort and prune

$$F(A \cup \{d\}) - F(A) \geq F(B \cup \{d\}) - F(B) \quad \text{for } A \subseteq B$$

(Upper bound on)
Marginal gain $\Delta_i$

![Diagram]

Leskovec et al., KDD ’07
Lazy Greedy

**Idea:**

- Use $\Delta_i$ as upper-bound on $\Delta_j$ ($j > i$)

**Lazy Greedy:**

- Keep an ordered list of marginal benefits $\Delta_i$ from previous iteration
- Re-evaluate $\Delta_i$ only for top element
- Re-sort and prune

\[
F(A \cup \{d\}) - F(A) \geq F(B \cup \{d\}) - F(B) \quad A \subseteq B
\]
Lazy Greedy

- **Idea:**
  - Use $\Delta_i$ as upper-bound on $\Delta_j$ $(j > i)$

- **Lazy Greedy:**
  - Keep an ordered list of marginal benefits $\Delta_i$ from previous iteration
  - Re-evaluate $\Delta_i$ **only** for top element
  - Re-sort and prune

$$F(A \cup \{d\}) - F(A) \geq F(B \cup \{d\}) - F(B) \quad A \subseteq B$$
Summary so far:

- Diversity can be formulated as a set cover
- Set cover is submodular optimization problem
- Can be (approximately) solved using greedy algorithm
- Lazy-greedy gives significant speedup

![Graph showing running time vs. number of blogs selected for different methods]

- Lower is better
- Running time (seconds)
- Number of blogs selected
- Exhaustive search (all subsets)
- Naive greedy
- Lazy
But what about personalization?

- Election trouble
- Songs of Syria
- Sandy delays

Recommendations
We assumed same concept weighting for all users
Each user has **different** preferences over concepts.

**politico**

**movie buff**

- France
- Mali
- Hagel
- Pentagon
- Obama
- Romney
- Zero Dark Thirty
- Argo
- NFL
Personal concept weights

- Assume each user $u$ has different preference vector $w_c^{(u)}$ over concepts $c$

$$\max_{A:|A|\leq k} F(A) = \sum_{c} w_c \ \text{cover}_A(c)$$

- **Goal:** Learn personal concept weights from user feedback
France intervenes
Chuck for Defense
Argo wins big
Multiplicative Weights (MW)

- **Multiplicative Weights algorithm**
  - Assume each concept $c$ has weight $w_c$
  - We recommend document $d$ and receive feedback, say $r = +1$ or $-1$

- **Update the weights:**
  - For each $c \in X_d$ set $w_c = \beta^r w_c$
    - If concept $c$ appears in doc $d$ and we received positive feedback $r=+1$ then we increase the weight $w_c$ by multiplying it by $\beta$ ($\beta > 1$)
    - otherwise we decrease the weight (divide by $\beta$)
  - Normalize weights so that $\sum_c w_c = 1$
Summary of the Algorithm

- **Steps of the algorithm:**
  1. Identify **items** to recommend from
  2. Identify **concepts** [what makes items redundant?]
  3. **Weigh** concepts by general importance
  4. Define **item-concept coverage function**
  5. **Select** items using probabilistic set cover
  6. Obtain **feedback**, **update** weights