Submodular functions cont.

Optimization - 10725
Carlos Guestrin
Carnegie Mellon University
April 30th, 2008

Submodularity

- Formalizes notion of diminishing returns

\( \forall A, B \quad F(A \cup \{x\}) - F(A) \geq F(B \cup \{x\}) - F(B) \quad A \subseteq B \)

- Equivalent definition:

\( \forall A, B \quad F(A) + F(B) \geq F(A \cup B) + F(A \cap B) \)
What do we get from submodularity

- Submodularity is a general property of set functions

- Submodular function can be minimized in polynomial time!

- But our problem is

Another example…

- Maximum cover
  - Ground elements $v \in V$
  - Set of sets $S_i \subseteq V$
  - Pick $k$ sets, maximize number of covered elements

  $$F(A) = \sum_v I(v \text{ is } U S_i)$$

  $A$ is a set of sets $A = \{S_1, ..., S_k\}$

  $$F(A \cup \{x\}) - F(A) \geq F(B \cup \{x\}) - F(B) \quad A \subseteq B$$

  Consider element $v$

  $$\{\text{ covered } A, \text{ covered } B \text{ but not } B \cup A, \text{ covered } B \text{ but not } A\}$$
Maximizing submodular functions – cardinality constraint

- Given
  - Submodular function
  - Normalized $F(\emptyset) = 0$
  - Non-decreasing $F(A) \leq F(B)$ for $A \subseteq B$

- Greedy algorithm guarantees
  $$F(A_{\text{greedy}}) \geq (1 - \frac{1}{e}) F(A_{\text{opt}})$$

- Can you get better algorithm?
  
  **NO**, unless $P = NP$.

Online bounds

- Submodularity provides bounds on the quality of the solution $A$ obtained by any algorithm
  - For normalized, non-decreasing functions

- Advantage of adding elements to $A$:

- Bound on the quality of any set $A$:

- Tighter bound:
Battle of the Water Sensor Networks Competition

- Real metropolitan area network (12,527 nodes)
- Water flow simulator provided by EPA
- 3.6 million contamination events
- Multiple objectives: Detection time, affected population, ...
- Place sensors that detect well “on average”

BWSN Competition results

- 13 participants
- Performance measured in 30 different criteria
  - G: Genetic algorithm
  - D: Domain knowledge
  - H: Other heuristic
  - E: “Exact” method (MIP)
What was the trick?

Simulated all 3.6M contaminations on 2 weeks / 40 processors
152 GB data on disk, 16 GB in main memory (compressed)

- Very accurate sensing quality 😊
- Very slow evaluation of F(A) 😞

30 hours/20 sensors
6 weeks for all
30 settings 😞

Using “lazy evaluations”:
1 hour/20 sensors
Done after 2 days! 😊

Lazy evaluations

- Naïve implementation of greedy:
  - Advantage of an element never increases:
    - Advantage:
    - What if you already picked a larger set:
      - Set after picking i elements: A_i

- Lazy evaluations:
  - Keep a priority queue over elements:
    - Initialize with advantage of each element
  - Pick element on top, recompute priority
  - If element remains on top
Other maximization settings

- Non-monotone submodular functions
- Non-unit costs
- Complex constraints
  - Paths
  - Spanning trees
- Worst-case optimization

Announcements

- University Course Assessments
  - Please, please, please, please, please, please, please, please, please, please, please, please, please...
- Project:
  - Poster session: Tomorrow 3-6pm, NSH Atrium
    - please arrive a 15mins early to set up
    - Don’t wait until the last minute to print
  - Paper: May 5th by 3pm
    - electronic submission by email to instructors list
    - maximum of 8 pages, NIPS format
    - no late days allowed
- Final:
  - Out: Monday, May 5
  - Due: Friday, May 9
Submodularity and concavity

- Consider set function $F(A)$ that only depends $|A|$:  

  - Recall defn of submodular functions:

    - In fact, $F(|A|)$ submodular if and only if

Submodular polyhedron

- $|V| = n$, for $x$ in $R^n$
  - Define $X(A) =$

  - Submodular polyhedron

  - For positive costs $c$, suppose we maximize:
Maximizing over submodular polyhedron

- Want:
  - Complex polyhedron, but very simple solution
    - Order nodes in increasing order of cost:
    - OPT x:
    - Prove optimality using duality

Extension of a set function

- For any set function F, define extension of F by:
  - Easy to compute for submodular functions
  - Amazing Theorem:
What can we do with convexity of extension?

- Suppose $c_A$ is a 01-vector for set $A$:
  - $c(i) =$

- Formulate maximization:

- At optimum:
  - By telescopic sum using OPT $x$

Minimizing submodular functions

- We know that
- Integer program:

- Convex relaxation

  - At optimum,
  - Can be solved using
- Thus, submodular function minimization:
Minimizing symmetric submodular fns

- Minimizing general submodular fns, not practical, because of ellipsoid algorithm
  - Symmetric submodular fns:
    - If submodular function symmetric:
      - Want non-trivial minimum:
        - Queryanne’s Algorithm for minimizing symmetric submodular fns:
          - Very simple to implement
          - Only $O(n^3)$

Application of minimizing symmetric submodular fns

- Given set $V$ of random variables
  - Split into two sets that are as independent as possible
- Submodular function:
  - Can be optimized using Queyranne’s algorithm
    - Useful, e.g., structure learning in graphical models
Submodular fns overview

- Minimized in polytime
- Approximate maximization
  - Under many different constrained settings

- Many many applications in AI/ML
  - Structure learning in graphical models
  - Clustering
  - Active learning
  - Sensor placement
  - Viral marketing
  - What blogs should we read to stay in touch with important stories
  - ...

- Not explored enough, plenty of opportunities!!