

Submodular functions cont.

Optimization - 10725

Carlos Guestrin

Carnegie Mellon University

April 30th, 2008

©2008 Carlos Guestrin

1

Submodularity

supermodular $F(A \cup x) - F(A) \leq F(B \cup x) - F(B)$
 $A \subseteq B$
modular $F(A \cup x) - F(A) = F(B \cup x) - F(B)$
 $(\Rightarrow) F(A) = \sum_{A_i \in A} F(A_i)$

- Formalizes notion of diminishing returns

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




- Equivalent definition:

$$\forall A, B \quad \underbrace{F(A) + F(B)} \geq \underbrace{F(A \cup B) + F(A \cap B)}$$

©2008 Carlos Guestrin

2

What do we get from submodularity

max $f(x)$
 x
 e.g. $x \in$ 
 Hand 

- Submodularity is a general property of set functions
- Submodular function can be minimized in polynomial time!

$$\min_{A \subseteq V} F(A)$$

F is submodular
 polytime

- But our problem is water problem

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

F is submodular

Another example...

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



$$F(A) = \sum_v I(v \text{ is } \bigcup_{i \in A} S_i)$$

A is a set of sets

$$A = \{S_1, S_2, \dots, S_k\}$$

$$F(A \cup S_i) - F(A) \geq F(B \cup S_i) - F(B) \quad A \subseteq B$$

consider element v

\hookrightarrow $\begin{cases} \text{covered } A \\ \text{covered } S_i, \text{ but not } B \text{ or } A \\ \text{covered } B, \text{ but not } A \end{cases}$

Maximizing submodular functions – cardinality constraint

Given

- Submodular function $F(A)$
 - Normalized $F(\emptyset) = 0$
 - Non-decreasing $F(A) \leq F(B) \quad A \subseteq B$
- e.g.) max cover

Greedy algorithm guarantees

$$F(A_{\text{greedy}}) \geq \underbrace{(1 - \frac{1}{e})}_{63\%} F(A_{\text{opt}})$$

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

Can you get better algorithm?

NO, unless $P=NP$, if you could solve max-cover better than $(1 - \frac{1}{e})$ then $P=NP$

Online bounds

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

- Submodularity provides bounds on the quality of the solution \hat{A} obtained by any algorithm $|\hat{A}| \leq k$

- For normalized, non-decreasing functions

$$F(\emptyset) = 0 \quad F(A) \leq F(B) \quad A \subseteq B$$

- Advantage of adding elements to \hat{A} :

$$\delta_{\hat{A}}(v) = F(\hat{A} \cup \{v\}) - F(\hat{A})$$



- Bound on the quality of any set A:

$$v^* = \max_v \delta_{\hat{A}}(v)$$

$$F(\hat{A}) \geq F(A_{\text{opt}}) - k \delta_{\hat{A}}(v^*) \quad F(A_{\text{opt}}) = \max_{A: |A| \leq k} F(A)$$

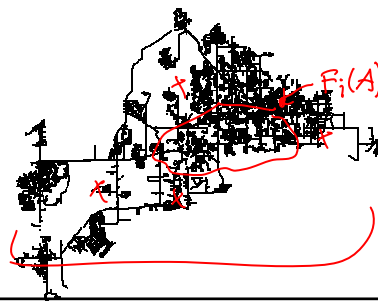
- Tighter bound:

$$\delta_{\hat{A}}(v_1) \geq \delta_{\hat{A}}(v_2) \geq \dots \geq \delta_{\hat{A}}(v_k)$$

$$F(\hat{A}) \geq F(A_{\text{opt}}) - \sum_{i=1}^k \delta_{\hat{A}}(v_i)$$

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"



$$F(A) = \frac{1}{n} \sum_{i=1}^n F_i(A)$$

Sum of Submodular fns is submodular

©2008 Carlos Guestrin

7

BWSN Competition results

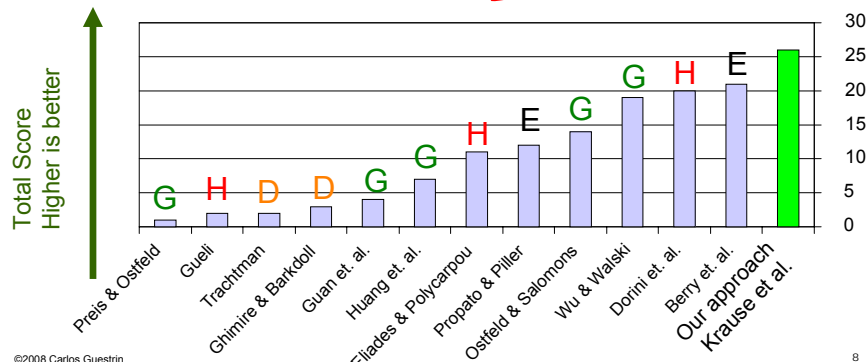
- 13 participants
- Performance measured in 30 different criteria

G: Genetic algorithm

D: Domain knowledge

H: Other heuristic

E: "Exact" method (MIP)



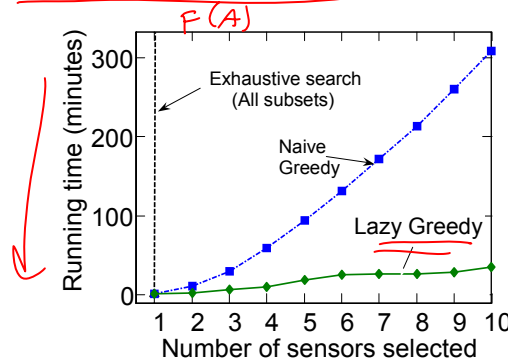
©2008 Carlos Guestrin

8

What was the trick?

Simulated alB.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 ☹



submodularity
to the rescue:

Using "lazy evaluations":
1 hour/20 sensors
Done after 2 days! ☺

©2008 Carlos Guestrin

Lazy evaluations

- Naïve implementation of greedy:

$$O(n \cdot k) \leq O(n^2)$$

$$A_0 = \emptyset$$

$$\text{For } i = 1 : k$$

$$s_i^* = \max_v F(A_{i-1} \cup \{v\}) - F(A_{i-1})$$

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

- Advantage of an element never increases:

$$\square \text{ Advantage: } d_A(v) = F(A \cup \{v\}) - F(A) \geq F(B \cup \{v\}) - F(B) = d_B(v)$$

- What if you already picked a larger set:

- Set after picking i elements: A_i

$$d_{A_i}(v) \leq d_{A_{i-1}}(v) \leq d_{A_{i-2}}(v) \leq \dots$$

- Lazy evaluations:

- Keep a priority queue over elements:

- Initialize with advantage of each element $\leftarrow d_{A_0}(v)$

- Pick element on top, recompute priority $\leftarrow d_{A_i}(v)$

- If element remains on top

don't have to
recompute rest

$$\begin{matrix} \text{time } 0 \\ d_{A_0}(v_1) \\ \vdots \\ d_{A_0}(v_k) \end{matrix}$$

$$\begin{matrix} \text{time } i \\ d_{A_i}(v_1) \\ d_{A_i}(v_2) \\ \vdots \\ d_{A_i}(v_k) \end{matrix}$$

©2008 Carlos Guestrin

10

Other maximization settings

- Non-monotone submodular functions

NOT: $F(A) \leq F(B) \quad A \subseteq B$

- Non-unit costs

$c(v_1) = 0.2 \quad c(v_2) = 10 \quad c(v_3) = 5 \dots$

- Complex constraints

- ☐ Paths
- ☐ Spanning trees



- Worst-case optimization

Competition
 $\max_{|A| \leq k} \sum_i F_i(A)$

$\max_A \min_i F_i(A)$
 $|A| \leq k$

©2008 Carlos Guestrin

11

Announcements

- University Course Assessments

- ☐ Please, please, 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

⚠ Please write your project # very visibly on your poster

- ☐ 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

NO late days on final

12

Submodularity and concavity

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

$$F(A) = F(|A|)$$



- Recall defn of submodular functions:

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

$$F(k+1) - F(k) \geq F(m+1) - F(m) \quad |A|=k \quad k \leq m \quad |B|=m$$

discrete derivative at $F(k)$ discrete derivative at $F(m)$

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

$F(n)$ is concave

function on integers n

Submodular polyhedron

- $|V|=n$, for x in \mathbb{R}^n

Define $X(A) = \sum_{i \in A} x(v_i)$

if $A = \{1, 7, 14\}$
 $X(A) = x(v_1) + x(v_7) + x(v_{14})$
 $x_A = x_1 + x_7 + x_{14}$

$$X = \begin{pmatrix} x(v_1) \\ \vdots \\ x(v_n) \end{pmatrix}$$

- Submodular polyhedron for set function F

$$P_F = \{x \in \mathbb{R}^n \mid X(A) \leq F(A), \forall A \subseteq V\}$$

- For positive costs $c, c \in \mathbb{R}_+^n$, suppose we maximize:

$$\max_x c'x \quad x \in P_F$$

$$x \in \{x \in \mathbb{R}^n \mid X(A) \leq F(A), \forall A \subseteq V\}$$

Linear Program 2^n constraints

Maximizing over submodular polyhedron

- Want:

$$\max_x c'x \quad x \in P_F \quad \} \quad LP$$

- Complex polyhedron, but very simple solution

- Order nodes in increasing order of cost:

$$v_1, v_2, v_3, \dots, v_n, \quad c(v_1) \leq c(v_2) \leq c(v_3) \leq \dots$$

- OPT x :

if F is submodular

$$\begin{cases} x^*(v_1) = F(v_1) \\ \forall k \in \{2, \dots, n\} \quad x^*(v_k) = F(v_1, \dots, v_k) - F(v_1, \dots, v_{k-1}) \end{cases}$$

- Prove optimality using duality

What can we do with convexity of extension?

- Suppose c_A is a 01-vector for set A :

$$c(i) = \begin{cases} 1 & \text{if } i \in A \\ 0 & \text{otherwise} \end{cases}$$

- Formulate maximization:

$$\max_x c_A' x \quad x \in P_F \quad \{ \mathbb{R}^n \mid x(A) \leq F(A), \forall A \subseteq V \}$$

- At optimum:

- By telescopic sum using OPT x

$$\begin{aligned} x^*(v_1) &= F(v_1) \\ x^*(v_k) &= F(v_1, \dots, v_k) - F(v_1, \dots, v_{k-1}) \\ c_A' x^* &= \sum_i c(v_i) x^*(v_i) = F(A) \end{aligned}$$

$\underbrace{c(v_1) \geq c(v_2) \geq \dots}_{i \in A} \quad \underbrace{\dots}_{i \notin A}$

Extension of a set function 1: (corrected)

- For 01-cost vector c_A , then:

Solving $\max_x c_A'x$ returns x^* s.t. $c_A'x^* = F(A)$
 for 01 costs c_A define $\hat{f}(c_A) = \max_x c_A'x$
 $x \in P_F$

- Define extension for general cost vector c :

Can write c as: $c = \sum_i \lambda_i c_{A_i}$

c_{A_i} is a 0-1 vector, $A_1 \supset A_2 \supset A_3 \supset \dots$

- Obtain sets recursively:

- Find the smallest value in vector, choose λ_i to take this value:

$c = \begin{pmatrix} 0.1 \\ 0 \\ 0.5 \\ 1 \end{pmatrix}$ $c_{A_1} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix}$ $\lambda_1 = 0.1$, $c_{A_2} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}$ $\lambda_2 = 0.4$, $c_{A_3} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$ $\lambda_3 = 0.5$

Extension: $\hat{f}(c) = \sum \lambda_i \hat{f}(c_{A_i}) = \sum \lambda_i F(A_i)$

©2008 Carlos Guestrin

17

Extension of a set function 2: (corrected)

- For any set function F , define the extension of F to real valued vectors (rather than 01) as:

- Rewrite c using positive combination of 01-vectors:

$$c = \sum_i \lambda_i c_{A_i}$$

- Extension: $\hat{f}(c) = \sum_i \lambda_i \hat{f}(c_{A_i})$

- Amazing Theorem:

$\hat{f}(c)$ is convex in $c \in \mathbb{R}_+^n$ if and only if F is submodular

©2008 Carlos Guestrin

18

Minimizing submodular functions

- We know that $\hat{f}(C) = F(A)$
- Integer program: $\min_C \hat{f}(C)$ $\left\{ \begin{array}{l} C \\ \forall i, C(i) \in \{0,1\} \end{array} \right\} \quad \min_{A \subseteq V} F(A)$
- Convex relaxation $\left\{ \begin{array}{l} \min_C \hat{f}(C) \\ C \\ \forall i, C(i) \in [0,1] \end{array} \right\}$
 - At optimum, will return integer $C \Rightarrow$ exact for
 - Can be solved using Ellipsoid algorithm
- Thus, submodular function minimization: $\min_{A \subseteq V} F(A) \leftarrow$ can be solved in polytime !!!

©2008 Carlos Guestrin

19

Minimizing symmetric submodular fns

- Minimizing general submodular fns, not practical, because of ellipsoid algorithm
- Symmetric submodular fns: $F(A) = F(V/A)$; $F(\emptyset) = 0$ (wlog)
- If submodular function symmetric: $F(A) + F(A^c) = F(A) + F(V/A) \geq F(V) + F(\emptyset) = F(\emptyset) + F(\emptyset)$
 $\forall A \quad F(A) \geq F(\emptyset)$
- Want non-trivial minimum: $\min_{A \subseteq V} F(A)$
 $1 \leq |A| \leq n$
- Queriyanne's Algorithm for minimizing symmetric submodular fns:
 - Very simple to implement
 - Only $O(n^3)$ calls of $F(A)$

©2008 Carlos Guestrin

20

Application of minimizing symmetric submodular fns

- Given set V of random variables
 - Split into two sets that are as independent as possible



$$I(A, B) = I(B, A)$$

$$I(A, B) = 0 \Leftrightarrow A \perp B$$

- Submodular function:

$$F(A) = I(A; V/A)$$

submodular
symmetric

$F_s(A) = I(A; V/A|S)$
is submodular in A
not in S

- Can be optimized using Queyranne's algorithm
 - Useful, e.g., structure learning in graphical models

©2008 Carlos Guestrin

21

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!!

©2008 Carlos Guestrin

22

Closing....

23

What you have learned this semester

- linear programs and QPs
- duality of LPs and QPs
- Lagrange multipliers
- semi-infinite (or just big) programs
- constraint generation, column generation, separation oracles
- combining dynamic programming and LPs
- maximum margin Markov networks
- ellipsoid algorithm
- gradient and subgradient descent
- convex sets, functions and programs
- duality of functions and sets
- maxent vs MLE in Gibbs
- SOCP
- SDP
- Interior point methods, Newton's method
- Mixed integer linear programs
- cutting planes, branch & bound, branch & cut
- approximation algorithms
- Submodularity
- ...

look at
problem
find
appropriate
structure

24

But we thought we could cover
even more... 😊

- More types of optimization problems
- More algorithms
- More theory
- More applications
- ...

- There are many other classes that may interest you:
 - Approximation algorithms
 - Advanced Approximation algorithms
 - Randomized Algorithms
 - Combinatorial Optimization (Tepper)
 - Optimization and Decision Making (Tepper)
 - Methods of Optimization (Math)
 - ...

25

You have done a lot!!!

**Thank You for the
Hard Work!!!**

**And for being the
guinea pigs for
a new course**

26