Applications of Submodularity

Jeff A. Bilmes

bilmes@ee.washington.edu

http://ssli.ee.washington.edu/~bilmes

Department of Electrical Engineering

University of Washington, Seattle

Joint work with:

- Mukund Narasimhan
  - Former student at University of Washington, now at Microsoft Research

Outline

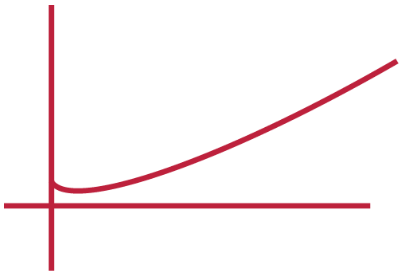
1. Submodularity
    1. Definition of Submodularity
    2. Examples of Submodularity
    3. Example of Submodularity in machine learning: bi-partite graph clustering, applied to words to improve language models.

Convexity and Convex Optimization

- Optimization is crucial for most machine learning problems.
- Objective function can vary: score, likelihood, probability. Goal is to find an extremum of this function.
- If the function is convex, any local extremum is a global extremum.
- Convexity allows efficient algorithms to find such an extremum even when the problem might be complex

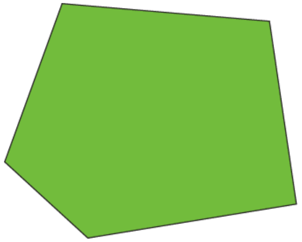
Convexity and Convex Optimization

- Convex Functions: f



$$f(\lambda_1 x_1 + \lambda_2 x_2) \leq \lambda_1 f(x_1) + \lambda_2 f(x_2)$$

- Convex Sets

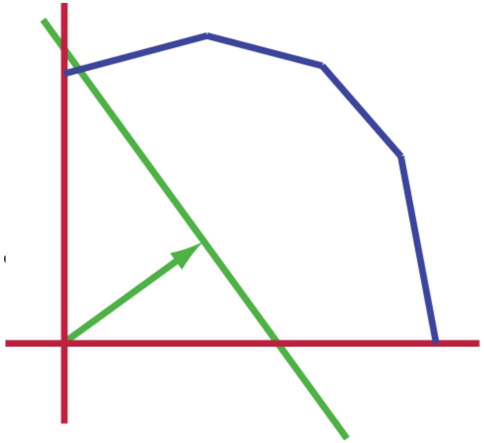


$$\forall x_1, x_2 \in \mathcal{V}, \forall 0 \leq \lambda \leq 1; \lambda_1 x_1 + \lambda_2 x_2 \in \mathcal{V}$$

- Convex Optimization

$$\min_{x \in \mathcal{V}} f(x)$$

  - unconstrained if $\mathcal{V}$ is everything.
  - Constraints on $\mathcal{V}$ lead to (LP, QP, SDP).



$$\min cx$$

$$\text{s.t. } Ax \leq b$$

Convexity and Machine Learning

- Some important and successful ML problems turn out to be convex
  - Support-vector machines
  - Kernel machines
  - One step of EM learning procedure

What about discrete optimization?

- Let $\mathcal{V}$ now be a set of n objects, and $f(\cdot)$ a function that maps from subsets of $\mathcal{V}$ to the reals. We wish to solve the following problem:

$$S^* = \operatorname*{argmin}_{\emptyset \subset S \subset \mathcal{V}} f(S)$$

- The problem seems hopelessly intractable because there are $2^{|V|}$-2 possibilities, so naïve enumeration won't work.
- This however captures many useful problems in speech: word clustering, structure learning, unit selection, language model selection, etc.
- A tractable solution is desirable.

Submodularity

- Submodularity is like a discrete form of convexity
- It formalizes the notion of diminishing returns.
- It characterizes many useful functions, such as entropy, mutual-information, and graph cuts.
- Some new machine learning procedures can arise when considering their potential submodularity.

Submodularity Defined

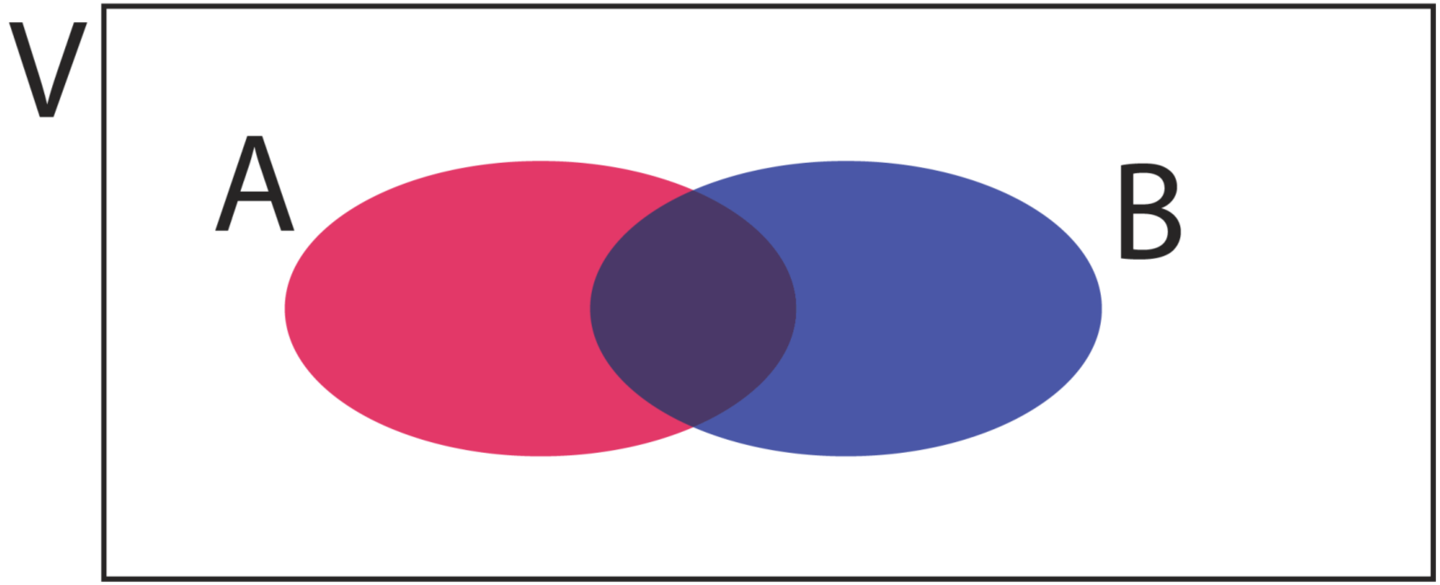
- Consider function $f : 2^{\mathcal{V}} \rightarrow \mathbf{R}$ defined on all subsets of $\mathcal{V}$
- Note, this is a function well defined on all subsets of some underlying set $\mathcal{V}$
- Function is said to be <u>submodular</u> if for all subsets $A, B \subseteq \mathcal{V}$, we have

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$$

Submodularity

- Function is said to be <u>submodular</u> if for all subsets $A, B \subseteq \mathcal{V}$, we have

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$$



Submodularity

- Submodularity captures the notions of diminishing returns which can serve as its definition:
  - The more you have, something new can have only potentially less value.
- For all $A \subseteq B \subset \mathcal{V}$ and for $x \in \mathcal{V} - B$

$$\underbrace{f(A \cup \{x\}) - f(A)}_{\text{Gain of adding } x \text{ to } A} \geq \underbrace{f(B \cup \{x\}) - f(B)}_{\text{Gain of adding } x \text{ to } B}$$

Submodularity

- Which definition to use?
  1. $f(A)+f(B)\geq f(A\cup B)+f(A\cap B)$
  2. $f(A\cup\{x\})-f(A)\geq f(B\cup\{x\})-f(B)$

- The first is more useful mathematically, but the second is more intuitive.

Example of Submodularity

- The set cardinality function is trivially submodular.

$$\forall A \subseteq \mathcal{V},\ f(A) = |A|$$

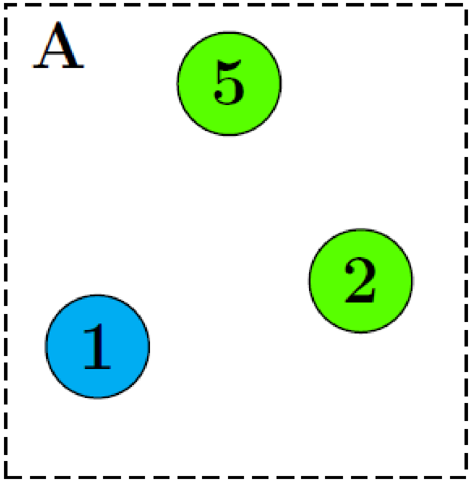
- Note that in this case, we always have equality in:

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$$

Example of Submodularity

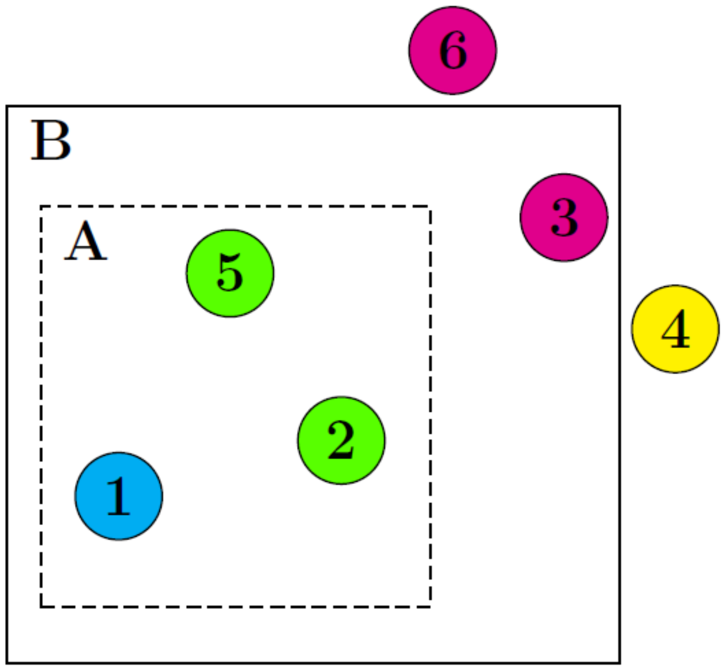
- Given an urn that may contain an unlimited number of single-color balls.
- For a given set A of balls, let $f(A)$ be the number of distinct ball colors in set A
- $f(A)$ will then be a submodular function.
- In the picture,

$$f(A) = 2$$



Example of Submodularity

- For any $A \subseteq \{1, 2, 3, 4, 5, 6\}$, $f(A)$ is the number of colors in A.



- $f\Big(\{1,2,5\} \cup \{6\}\Big) - f(\{1,2,5\}) = 1 > 0 = f\Big(\{1,2,3,5\} \cup \{6\}\Big) - f(\{1,2,3,5\})$

Examples of Submodularity

- $S = \{1, 2, 3, 4, 5, 6\}$ is the columns of a matrix.

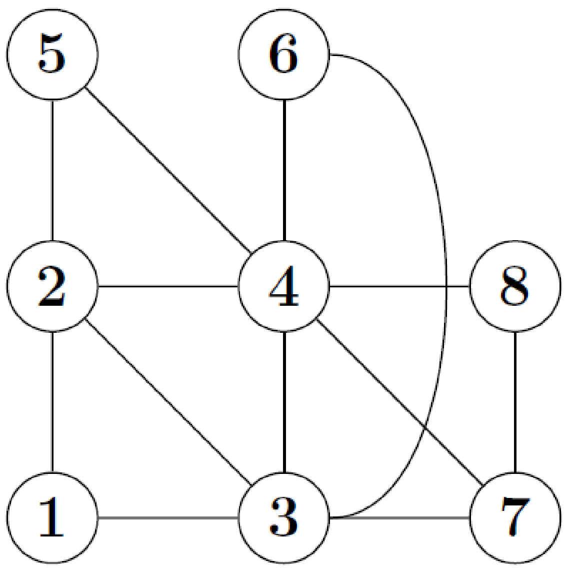
$$\begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \\ \end{array}$$

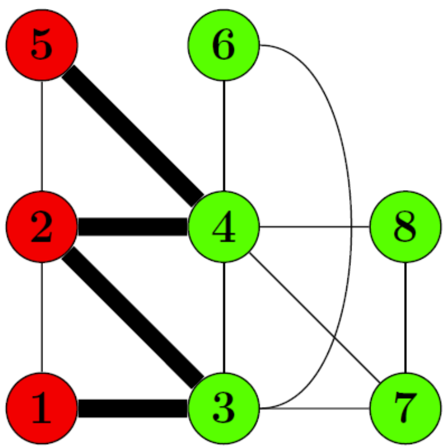
$$\begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

- $f(A)$ is the rank of the subspace spanned by A.
  - ⋆ $f\Big(\{1\}\Big) = f\Big(\{2\}\Big) = \cdots = f\Big(\{6\}\Big) = 1$
  - ⋆ $f\Big(\{1\} \cup \{4\}\Big) - f\Big(\{1\}\Big) = 1 \geq 0 = f\Big(\{1, 2\} \cup \{4\}\Big) - f\Big(\{1, 2\}\Big).$

Graph-cuts are submodular

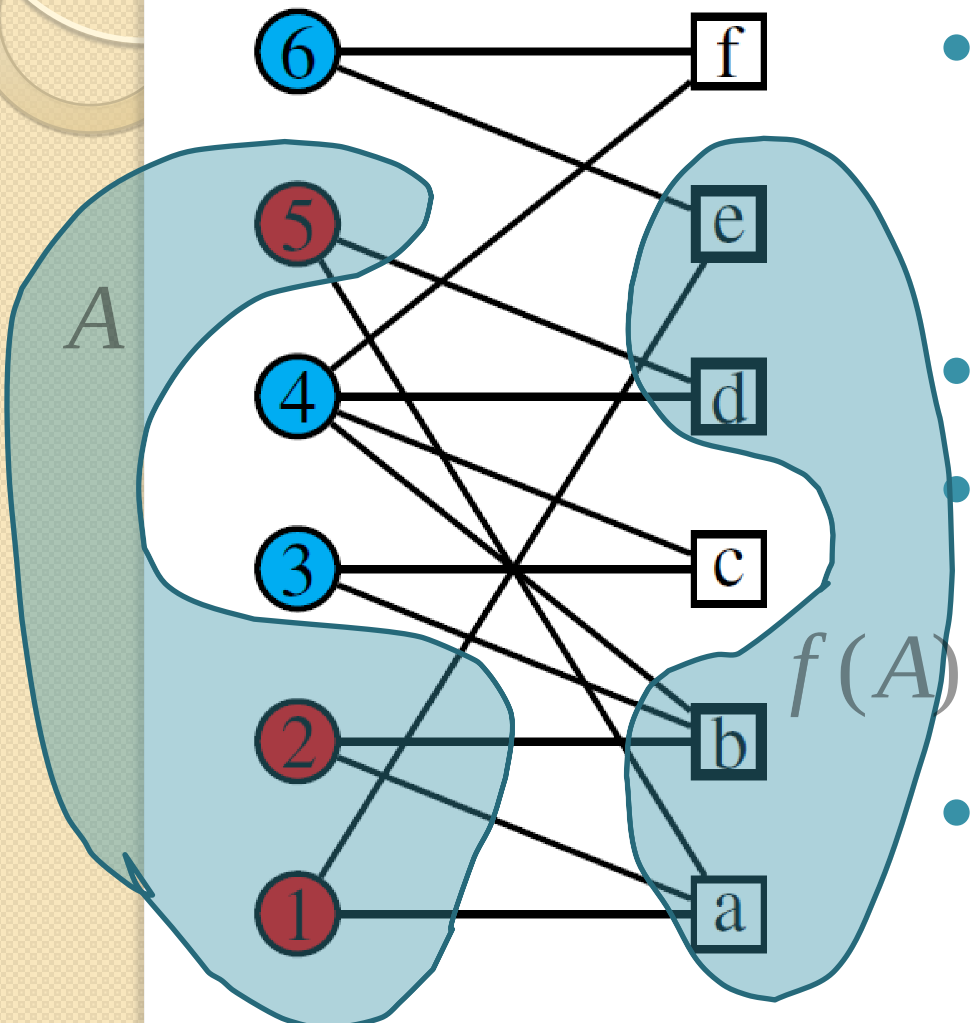
- $G = (S, E)$ a graph.
- If $A, B \subseteq S$ are disjoint, let $E(A; B)$ be edges adjacent to both A and B.
- Let $f(A) = |E(A; S \setminus A)|$. Then f is symmetric and submodular.





- $f(\{1, 2, 5\}) = f(\{3, 4, 6, 7, 8\}) = 4.$

Bi-partite graph-cuts are submodular



- Bi-partite graph has left part V and right part F.
- $G = (V, F, E)$
- Given $A \subseteq V$ then $f(A)$ is the number of neighbors of A
- Example: if $A = \{1, 2, 5\}$ then $f(A) = \{\text{a,b,d,e}\}$

Entropy is Submodular

- Let $(X_1, X_2, \ldots, X_n)$ be a collection of random variables and let $\mathcal{V} = \{1, 2, \ldots, n\}$
- For a given subset $A \subseteq V$ with $A = \{a_1, a_2, \ldots, a_{|A|}\}$ let $X_A = \{X_{a_1}, X_{a_2}, \ldots, X_{a_{|A|}}\}$
- We can define the entropy function as:

$$f(A) = H(X_A) = -\sum_{x_A} p(x_A) \log p(x_A)$$

- Then $f(A)$ is a submodular function!
- Intuition: Conditioning reduces uncertainty, uncertainty of a variable decreases as other RVs become known.

Symmetric Mutual Info. is also Submodular

- Let $(X_1, X_2, \ldots, X_n)$ be a collection of random variables and let $\mathcal{V} = \{1, 2, \ldots, n\}$
- For a given subset $A \subseteq V$ with $A = \{a_1, a_2, \ldots, a_{|A|})$ let $X_A = \{X_{a_1}, X_{a_2}, \ldots, X_{a_{|A|}}\}$
- We can define the symmetric mutual information function as:

$$f(A) = I(X_A; X_{\mathcal{V}-A}) = \sum_{x_{\mathcal{V}}} p(x_{\mathcal{V}}) \log \frac{p(x_A)\, p(x_{\mathcal{V}-A})}{p(x_{\mathcal{V}})}$$

- Generalizes graph cuts, but cut function is no longer just sum of edge weights.

Submodularity and Convexity

- Both generalize many problems in many fields (machine learning, economics, etc.)
- There is a 1-1 correspondence between certain submodular and certain convex functions (Lovász extension)
- Both are preserved under many transformations (addition, convolution, composition/addition with linear functions, etc.)
- Both exhibit tractable optimization.

Minimizing Submodular Functions

- Let $\mathcal{V}$ be a set of n objects, and f a function that maps from subsets of $\mathcal{V}$ to the reals.

$$S^* = \operatorname*{argmin}_{\emptyset \subset S \subset \mathcal{V}} f(S)$$

- Grötschel, Lovász, Schrijver: 1981 – first polynomial time algorithm for minimizing submodular functions (via ellipsoid method)
- Iwata, Fleischer, Fujishigi, and Schrijver: 2000, independently discovered first combinatorial polynomial algorithms for submodular function minimization.

Minimizing Submodular Functions

- Let $\mathcal{V}$ be a set of n objects, and f a function that maps from subsets of $\mathcal{V}$ to the reals.

$$S^* = \underset{\emptyset \subset S \subset \mathcal{V}}{\operatorname{argmin}} \; f(S)$$

- The function $f(\cdot)$ is symmetric if for $\emptyset \subseteq S \subseteq \mathcal{V}$:

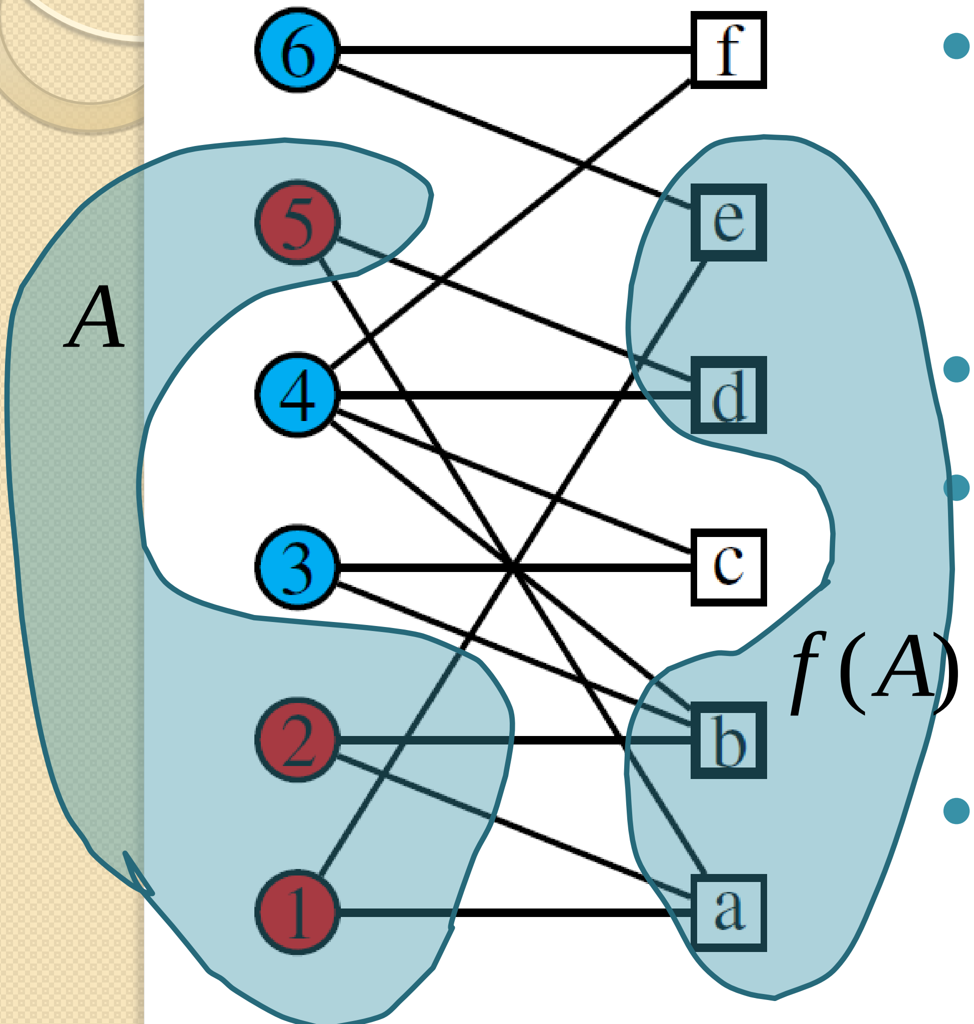
$$f(S) = f(\mathcal{V} - S)$$

- Queyranne: 1998 – first strongly-polynomial and O(n^3) algorithm for minimizing symmetric submodular functions.

Applications

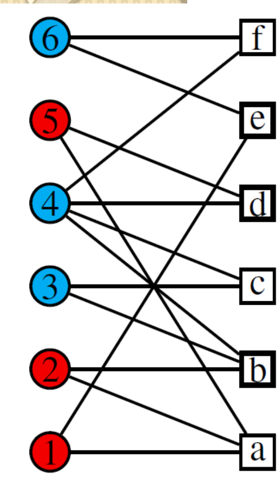
- Past few years, we've worked on many applications of submodularity
  - Learning graphical models (Narasimhan & Bilmes, 2004).
  - Subgraphical models (Narasimhan & Bilmes, 2005)
  - Submodular-supermodular procedure for discriminative structure (Narasimhan & Bilmes 2005)
  - Q-Clustering (Narasimhan & Jojic & Bilmes, 2005)
  - Balanced Word Clustering (Narasimhan & Bilmes, 2007)
- We here expand on only this last issue, finding balanced clusterings of words.

Bi-partite graph-cuts are submodular



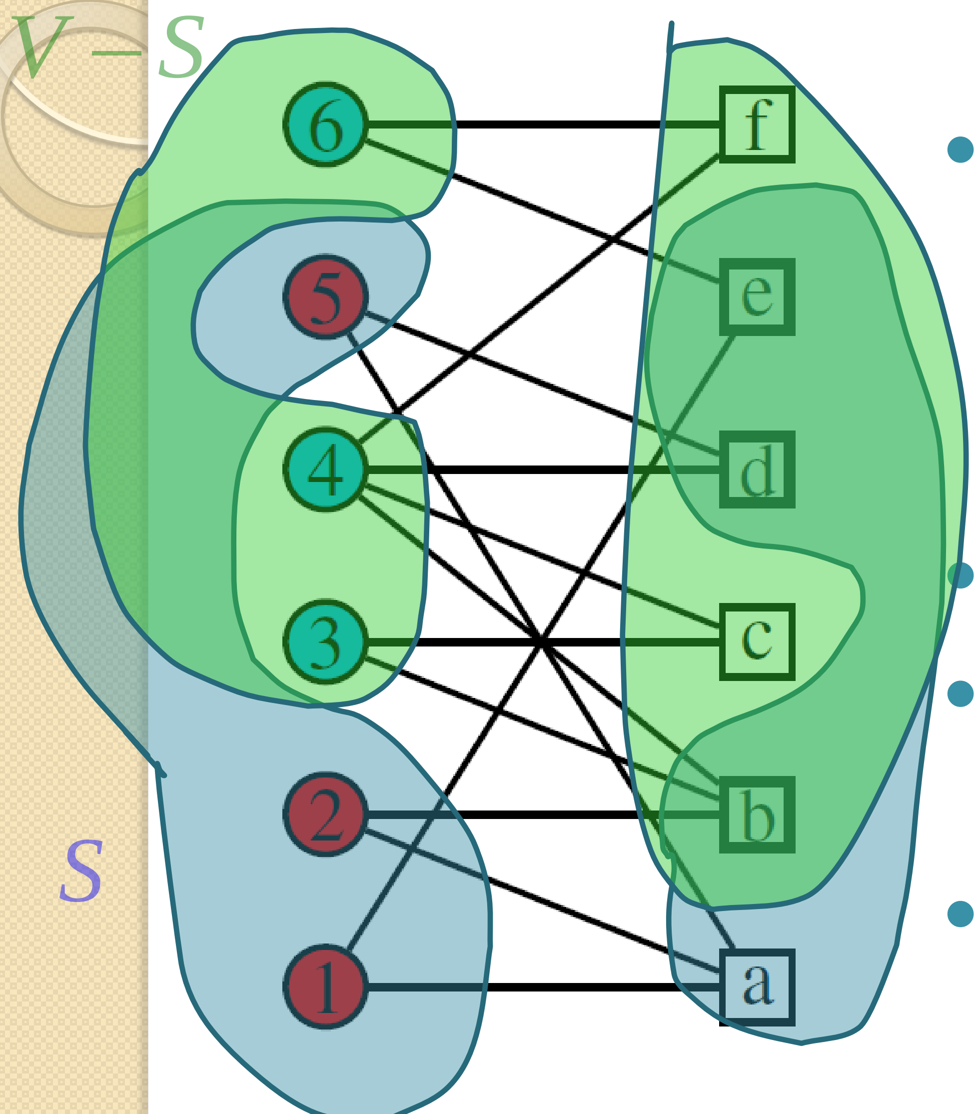
- Bi-partite graph has left part V and right part F.
- $G=(V,F,E)$
- Given $A \subseteq V$ then $f(A)$ is the number of neighbors of A
- Example: if $A=\{1,2,5\}$ then $f(A)=\{\text{a,b,d,e}\}$

Submodularity & bipartite cuts



- Same bipartite graph $G = (V, F, E)$
- V is a set of "objects", in our case it will be word (or types).
- F is a set of "features" (e.g., could be tags, or any other possible features of the words).
- A node v $\in$V connected to an f $\in$ F if object v has feature f.
- Goal: Partition V into clusters, to minimize the feature overlap, while keeping clusters balanced.

Bipartite cuts & submodularity



- For S⊆V, Γ_c(S) is the number of features common to both X and V - S.
- $\Gamma_c(\{1,2,5\}) = |\{b,d,e\}|$
- Γ_c(X) is symmetric submodular
- We can bi-partition V in polynomial time.

Maintaining Balanced Clusters

- We could minimize the above, but we also want to maintain balance. Let $S_1 \cup S_2 = \mathcal{V}$ be a partition of the objects. Criteria to minimize becomes:

$$\text{ratioCut}(S_1, S_2) = \frac{\Gamma_c(S_1)\Gamma_c(S_2)}{|S_1||S_2|}$$

- This criteria penalizes small clusters since we divide by the size of each one.
- Minimizing this criteria is unfortunately NP-complete (Shi & Malik).
- We define an iterative procedure guaranteed to find a form of "local" optima.

Local Split and Swap

- For any biparition $S_1 \cup S_2 = \mathcal{V}$ and $X \subseteq S_1$ let

$$\text{Gain}(X) = \Gamma_c(S_1) - \Gamma_c(S_1 - X)$$

$$\text{AvgGain}(X) = \frac{\Gamma_c(S_1) - \Gamma_c(S_1 - X)}{|X|}$$

$$\text{BestAvgGain} = \max_{\emptyset \subset X \subseteq S_1} \text{AvgGain(X)}$$

There exists a good swap

- Theorem (Narayanan 2003): Let (S_1, S_2) be a bipartition of $\mathcal{V}$ (so $\mathcal{V} = S_1 \cup S_2$ and $S_1 \cap S_2 = \emptyset$) and let $\emptyset \neq U \subset S_1$ be a proper subset of S_1 satisfying

$$\text{BestAvgGain} = \text{AvgGain(U)}$$

Then

$$\text{ratioCut}(S_1 - U, S_2 \cup U) < \text{ratioCut}(S_1, S_2)$$

- Therefore, if we find such a U, we can guarantee a local step in the right direction to reduce ratioCut.

Finding a good swap

- But how do we find such a U?
- Theorem (Narayanan 2003). It is the case that $\lambda = \mathrm{BestAveGain}$ iff there is a subset $U \subset S_1$ such that

$$\min_{X \subseteq S_1} \{\Gamma_c(X) - \lambda |X|\} = \Gamma_c(S_1) - \lambda |S_1|$$

$$= \Gamma_c(U) - \lambda |U|$$

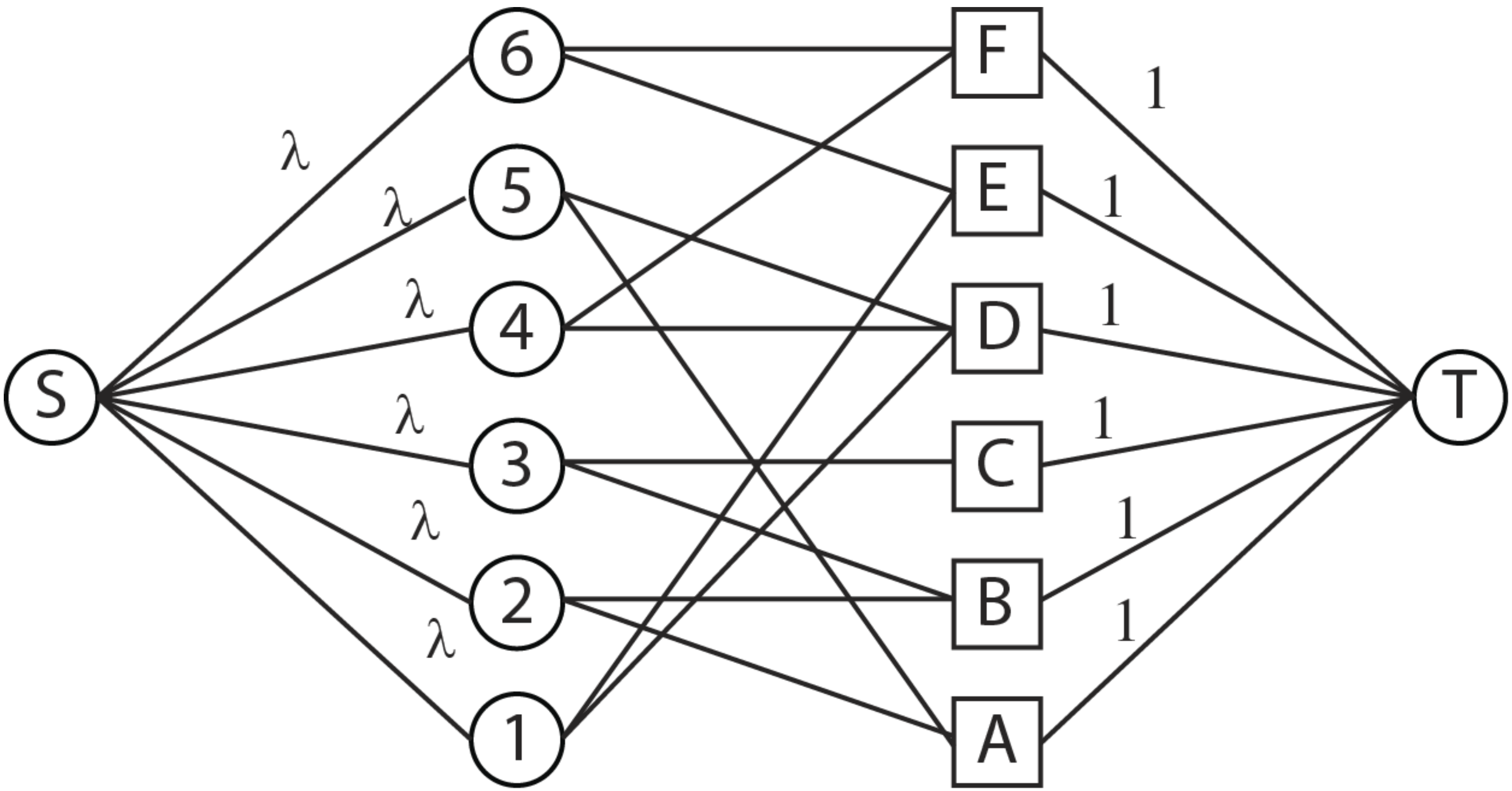
- So we need to be able to solve this minimization problem for all possible values of λ, and when we find it for all λ we try them all.

Finding a good swap

- How do we find the solution to the following
$$\min_{X \subseteq S_1} \{\Gamma_c(X) - \lambda |X|\}$$
for all values λ?
- Turns out there are no more than |V| distinct solutions, with |V| distinct values of λ and they can all be found simultaneously by an amazing algorithm by Tarjan (parametric flow) in $O(|V|^2|E\})$ time.

Partition sub-splitting & swap

- Represent as a flow, and use Tarjan's parametric max-flow (min-cut) algorithm (finds solutions for all λ).

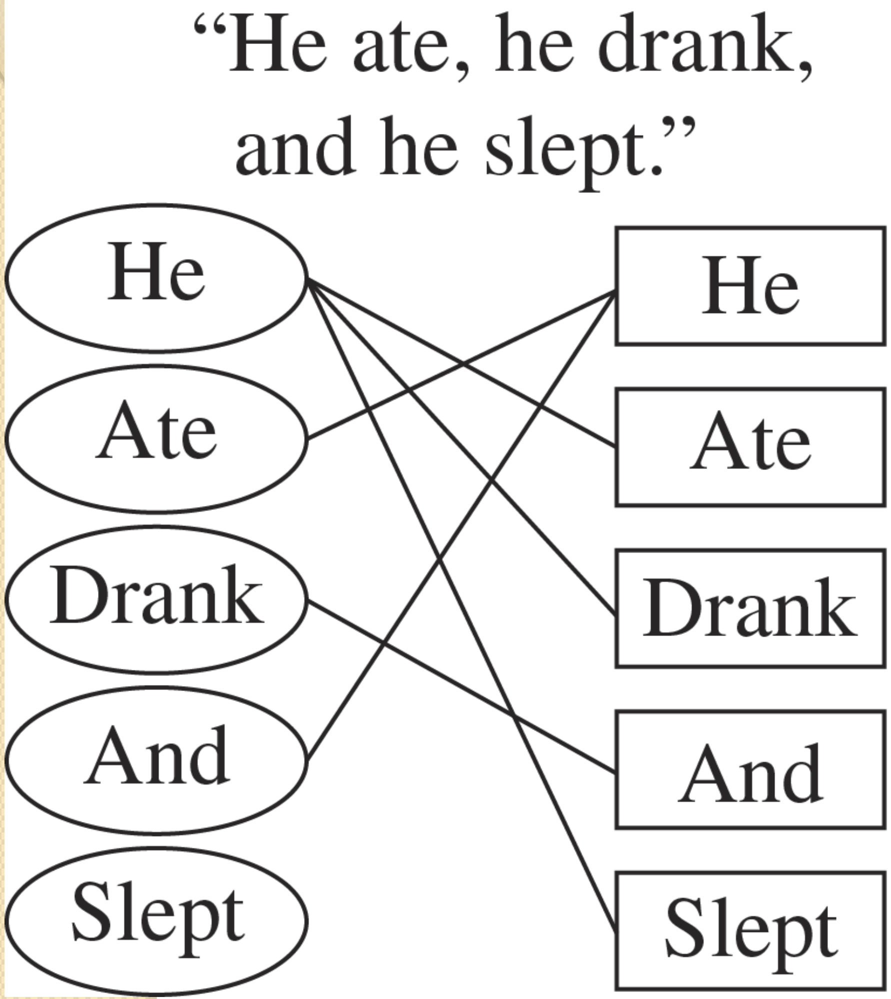


Iterative Algorithm To Improve RatioCut

1. Start with an arbitrary bi-partition
2. Compute a local move ($O(|V|^2|E|)$) using parametric max-flow procedure.
3. Try all $|V|$ possible improvements, and take the one that is best.
4. If an improvement was found, go to 1, otherwise stop.

- This gives a bi-partition: We partition the partitions to refine the clustering (top-down procedure) as needed.

Application: word clustering in language models



- Words on the left, their features on the right.
- The“features” of a word can be the words that occur to the right of it in a text corpus.
- Optimal for bi-gram LM, approximate for a tri-gram LM.

Application: word clustering in language models

- Clusters are used in a factored language model with generalized backoff (Bilmes & Kirchhoff, HLT03), (Kirchhoff'2004).
- In a bi-gram, we first backoff to the cluster of the previous word as in:

$$p(w_t \mid w_{t-1}, c(w_{t-1}))$$

where c(w) is the clustering of the previous word.

Language model perplexity results.

- 497 clusters on Wall Street Journal (WSJ) data from the Penn Treebank 2 tagged (88-89) WSJ collection. Factored language models using Kneser-Ney smoothing, all done using Stolcke's SRILM.
- Results for both bi-gram (for which algorithm is optimal) and tri-gram (for which algorithm is only an approximation).

Language model PPL results

- Why do we care about bigrams?
- If we can obtain good PPL results with a bigram, that can be used in a multi-pass ASR system to generate 1st pass lattices, but it a bigram still has small state space.

Language model PPL results

- Comparing results to manually-generated (cheating) POS tags, which potentially can look into the future (which is why it is cheating).

	Manually Generated	Bipartite Adjacency	Brown et al. ([3])
Bigram (Minimum)	276.229	263.616	279.867
Bigram (Average)	277.135	264.579	281.169
Bigram (Maximum)	278.837	266.335	283.710
Trigram (Minimum)	237.735	231.300	233.111
Trigram (Average)	239.189	233.239	234.887
Trigram (Maximum)	240.765	235.088	236.765

Conclusion

1. Submodularity is a powerful concept
2. Like convexity, it is sometimes possible to define tractable algorithms but in this case over discrete sets.
3. The polynomiality of submodular optimization pushes the boundary of the set of discrete problems that can be solved exactly.

The end