Microsoft Research

NIPS Thanks Its Sponsors

amazon.com

Microsoft Research

Google

facebook

SKYTREE. THE MACHINE LEARNING COMPANY

TWO SIGMA

United Technologies Research Center

D E Shaw & Co

xerox

YAHOO! LABS

IBM Research

TOYOTA

DRW TRADING GROUP

millionshort

criteo

PDT PARTNERS

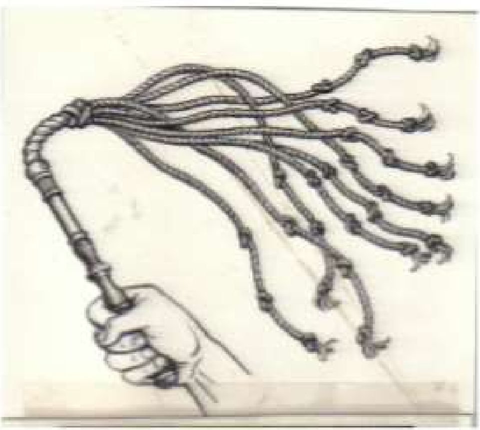
Springer Machine Learning Journal

Disney Research

From Bandits to Experts
A Tale of Domination and Independence

Nicolò Cesa-Bianchi

Università degli Studi di Milano, Italy



Joint work with:

Noga Alon
Claudio Gentile
Yishay Mansour

Nonstochastic sequential decision-making

Player repeateadly chooses actions from a set of K available actions

For $t = 1, 2, \dots$

1. Loss $\ell_t(a)$ is assigned to every action $a = 1, \dots, K$ (hidden from the player)

Nonstochastic sequential decision-making

Player repeateadly chooses actions from a set of K available actions

For $t = 1, 2, \ldots$

1. Loss $\ell_t(a)$ is assigned to every action $a = 1, \ldots, K$ (hidden from the player)
2. Player picks an action X_t (possibly using randomization) and incurs loss $\ell_t(X_t)$
3. Player gets feedback information

Nonstochastic sequential decision-making

Player repeateadly chooses actions from a set of K available actions

For $t = 1, 2, \ldots$

1. Loss $\ell_t(a)$ is assigned to every action $a = 1, \ldots, K$ (hidden from the player)
2. Player picks an action X_t (possibly using randomization) and incurs loss $\ell_t(X_t)$
3. Player gets feedback information
   - Bandit observation: Only $\ell_t(X_t)$ is revealed
   - Expert observation: $\ell_t(a)$ for each $a = 1, \ldots, K$ is revealed

Nonstochastic sequential decision-making

Player repeateadly chooses actions from a set of K available actions

7 3 2 4 1 6 7 4 9

For $t = 1, 2, \ldots$

1. Loss $\ell_t(a)$ is assigned to every action $a = 1, \ldots, K$ (hidden from the player)
2. Player picks an action X_t (possibly using randomization) and incurs loss $\ell_t(X_t)$
3. Player gets feedback information
   - Bandit observation: Only $\ell_t(X_t)$ is revealed
   - Expert observation: $\ell_t(a)$ for each $a = 1, \ldots, K$ is revealed

Goal: Player's total loss must be close to that of the single best action (no stochastic assumptions on losses)

Measuring player's performance

Regret (as a function of number T of plays)

$$R_T = \underbrace{\mathbb{E}\left[\sum_{t=1}^{T} \ell_t(X_t)\right]}_{\text{Total loss of player}} - \underbrace{\min_{a=1,\dots,K} \sum_{t=1}^{T} \ell_t(a)}_{\text{Total loss of single best action}}$$

Measuring player's performance

Regret (as a function of number T of plays)

$$R_T = \underbrace{\mathbb{E}\left[\sum_{t=1}^{T} \ell_t(X_t)\right]}_{\text{Total loss of player}} - \underbrace{\min_{a=1,\dots,K} \sum_{t=1}^{T} \ell_t(a)}_{\text{Total loss of single best action}}$$

Known results

- Hedge for experts: $R_T \leqslant \sqrt{T \ln K}$
- Exp3 for bandits: $R_T \leqslant \sqrt{TK \ln K}$

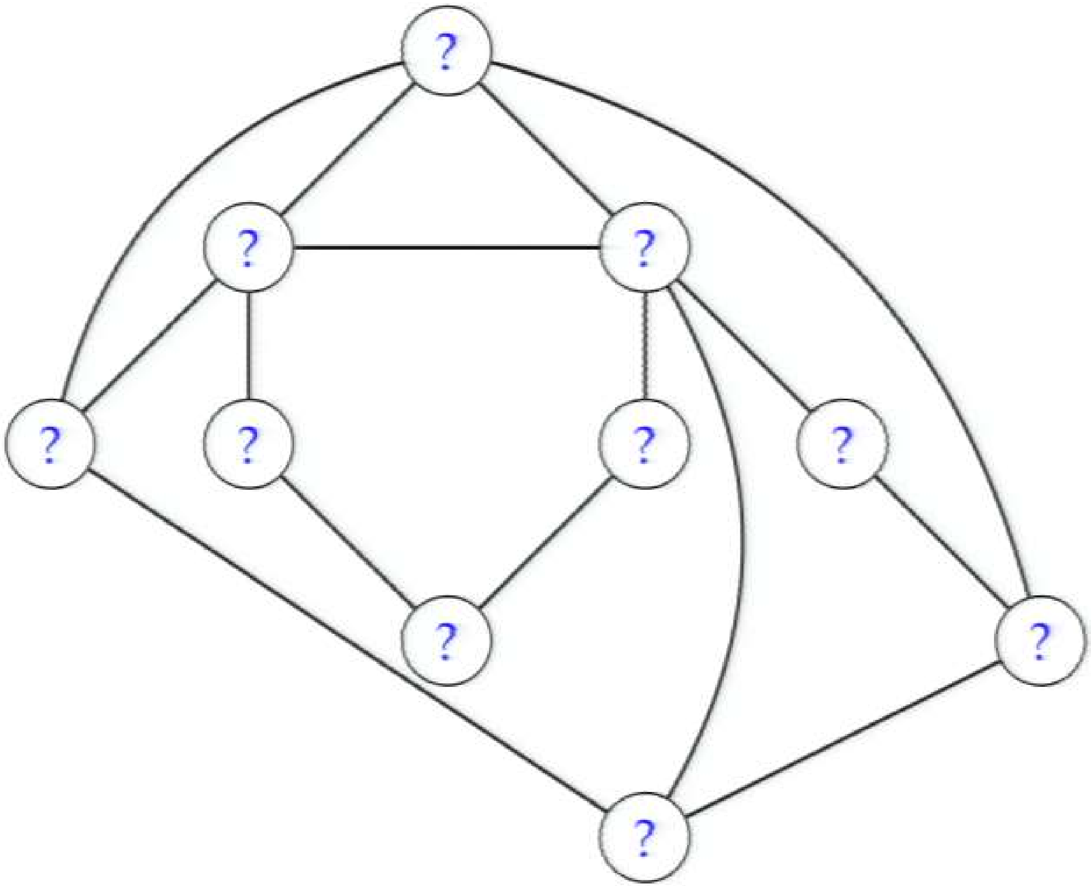
These bounds are tight (only $\ln K$ in the bandit bound is unnecessary)

Relationships between actions [Mannor and Shamir, 2011]

Undirected

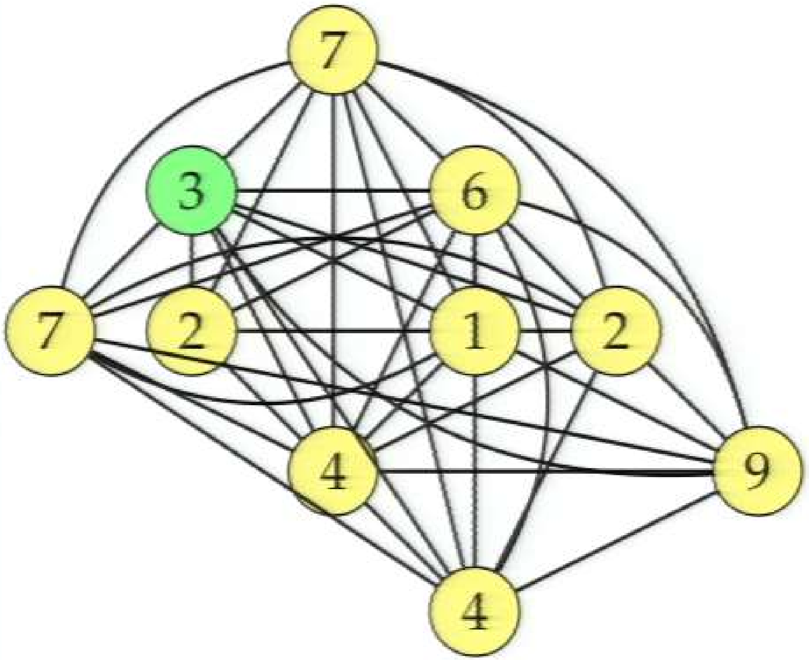
Directed

Undirected observation graph

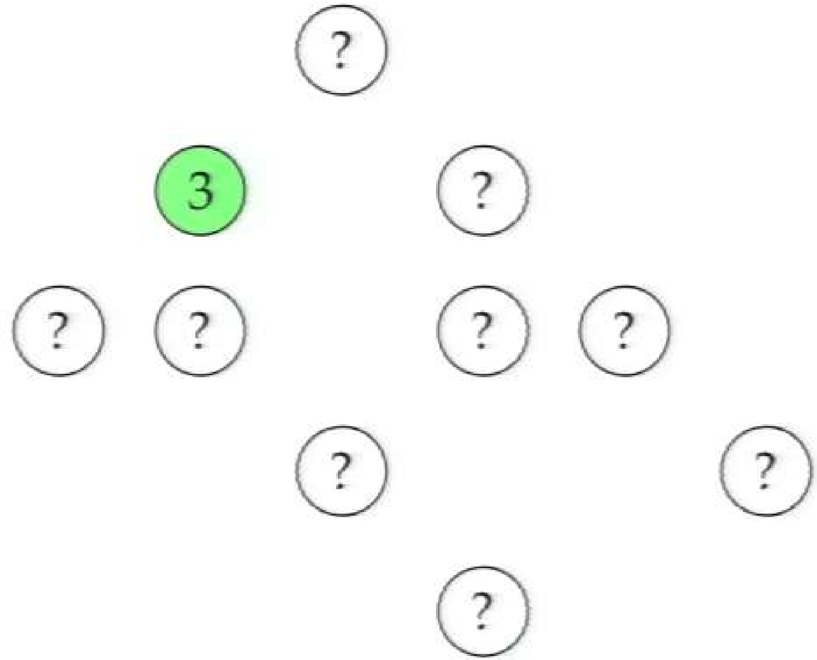


Recovering expert and bandit settings

Experts: clique

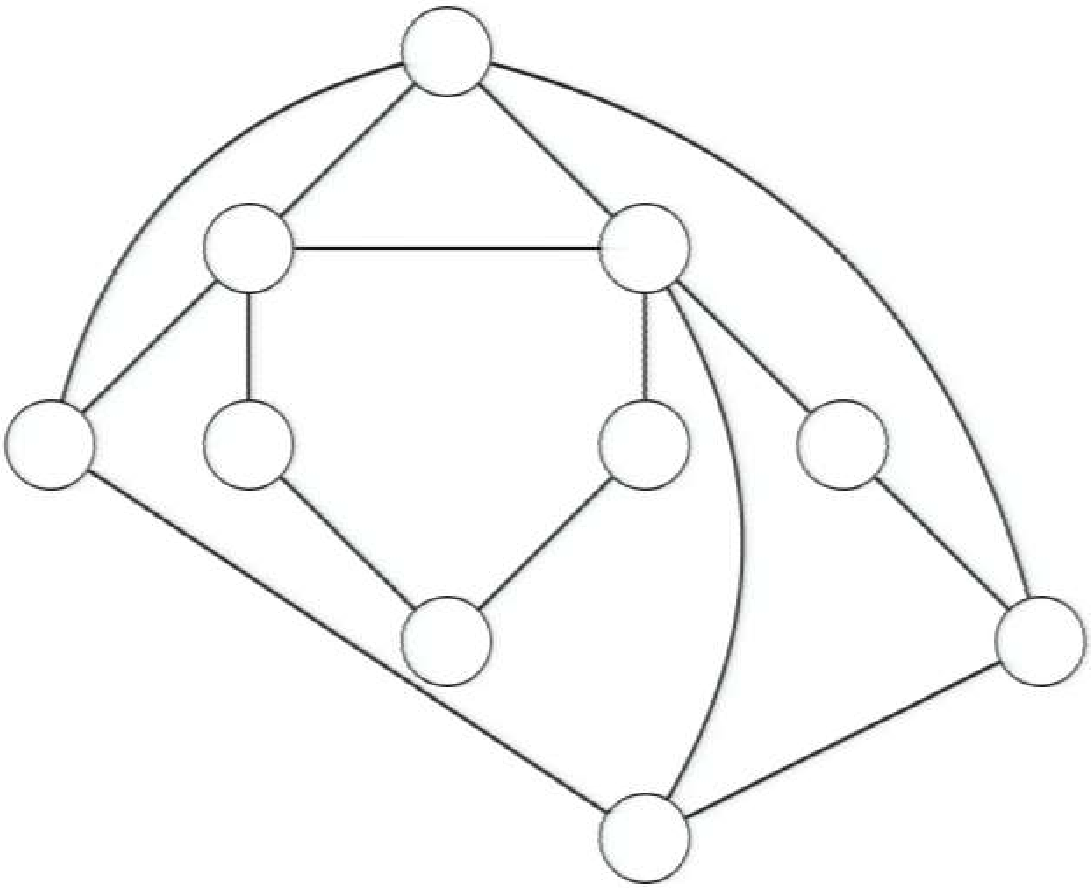


Bandits: edgeless graph



Independence number $\alpha(G)$

The size of the largest independent set



ELP algorithm

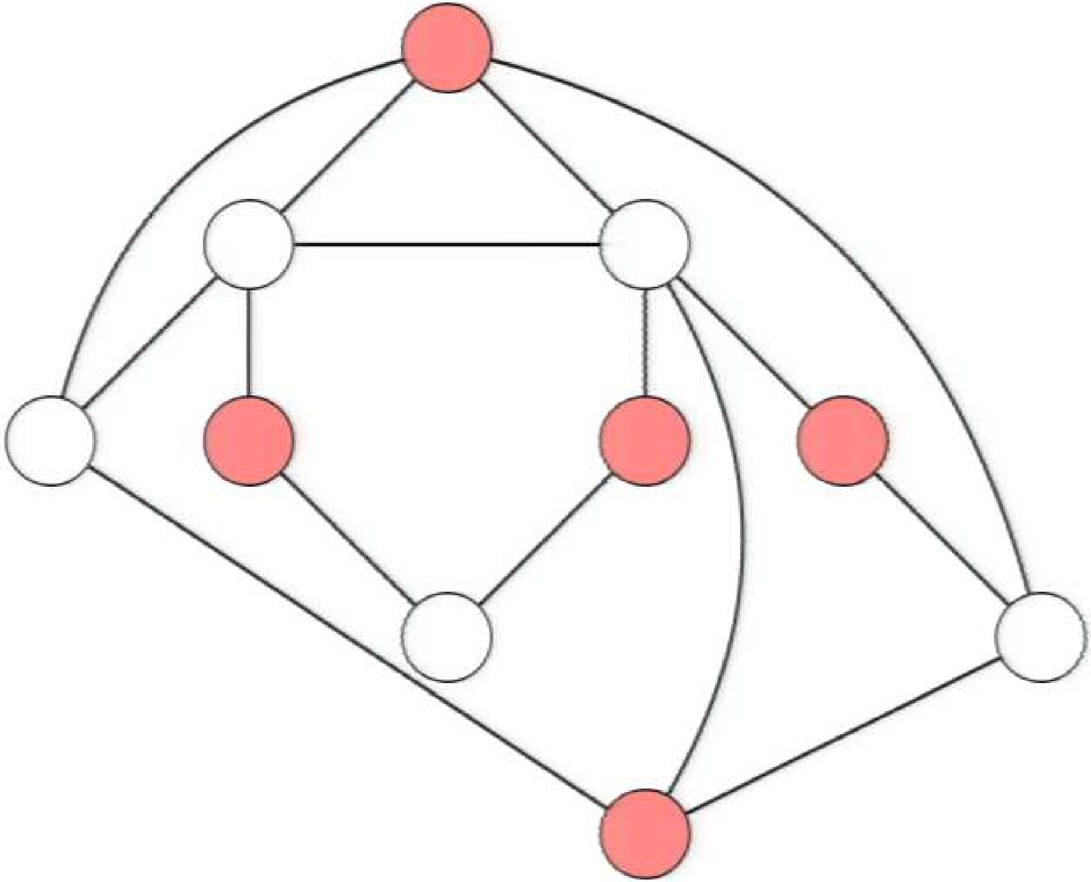
[Mannor and Shamir, 2011]

- Tight regret bound: $R_T \leqslant \sqrt{T\alpha(G)\ln K}$ $\alpha(G) \leqslant K$
- Experts ($G =$ clique): $\alpha(G) = 1$
- Bandits ($G =$ edgeless graph): $\alpha(G) = K$
- ELP must solve a linear program at each step
- Result holds also when G changes over time: $G_1, G_2, \ldots, G_T$

$$R_T \leqslant \sqrt{\sum_t \alpha(G_t) \ln K}$$

Independence number $\alpha(G)$

The size of the largest independent set



ELP algorithm [Mannor and Shamir, 2011]

- Tight regret bound: $R_T \leqslant \sqrt{T\alpha(G)\ln K}$ $\alpha(G) \leqslant K$
- Experts ($G =$ clique): $\alpha(G) = 1$
- Bandits ($G =$ edgeless graph): $\alpha(G) = K$
- ELP must solve a linear program at each step
- Result holds also when G changes over time: $G_1, G_2, \ldots, G_T$

$$R_T \leqslant \sqrt{\sum_t \alpha(G_t)\ln K}$$

Our results

Exp3-SET for undirected observation graphs

- Same regret bound as ELP
- No need of solving linear programs
- No need of knowing G_t before predicting!

Our results

Exp3-SET for undirected observation graphs

- Same regret bound as ELP
- No need of solving linear programs
- No need of knowing G_t before predicting!

Exp3-DOM for directed observation graphs

- Harder than the undirected case (less feedback for the player)
- Yet, regret worse than the undirected case only by log factors
- However, G_t must be known before predicting

Exp3-SET for undirected observation graphs

Player's strategy

$$\mathbb{P}(X_t = a) \quad \propto \quad \exp\left(-\eta \sum_{s=1}^{t-1} \widehat{\ell}_s(a)\right) \qquad a = 1, \dots, K$$

where $$\widehat{\ell}_t(a) = \begin{cases} \dfrac{\ell_t(a)}{\mathbb{P}(\ell_t(a) \text{ is observed})} & \text{if } \ell_t(a) \text{ is observed} \\ 0 & \text{otherwise} \end{cases}$$

Note: no exploration needed

Analysis

Regret bound

$$R_T \leqslant \frac{\ln K}{\eta} + \frac{\eta}{2} \sum_{t=1}^{T} \sum_{a} \mathbb{P}\big(X_t = a \mid \ell_t(a) \text{ is observed}\big) \leqslant \sqrt{T\alpha(G)\ln K}$$

Analysis

Regret bound

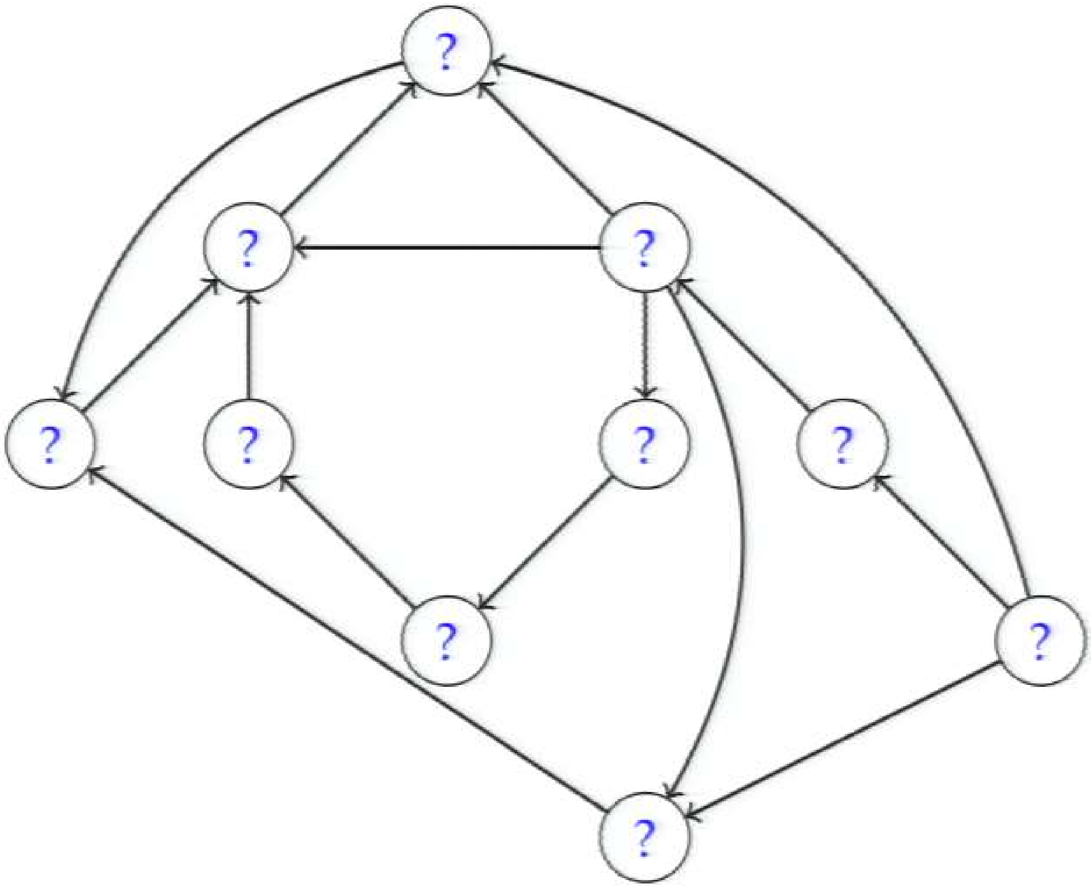
$$R_T \leqslant \frac{\ln K}{\eta} + \frac{\eta}{2} \sum_{t=1}^{T} \sum_{a} \mathbb{P}\big(X_t = a \mid \ell_t(a) \text{ is observed}\big) \leqslant \sqrt{T\alpha(G) \ln K}$$

Key lemma: $$\sum_{a} \mathbb{P}\big(X_t = a \mid \ell_t(a) \text{ is observed}\big) \leqslant \alpha(G)$$

Check special cases:

$$\mathbb{P}\big(X_t = a \mid \ell_t(a) \text{ is observed}\big) = \begin{cases} 1 & \text{bandits} \\ \mathbb{P}(X_t = a) & \text{experts} \end{cases}$$

Directed observation graph



Issues with directed observation graphs

Orientation of edges reduces feedback – regret will increase

$\sum_a \mathbb{P}(X_t = a \mid \ell_t(a) \text{ is observed})$ can be large even when $\alpha(G)$ is small

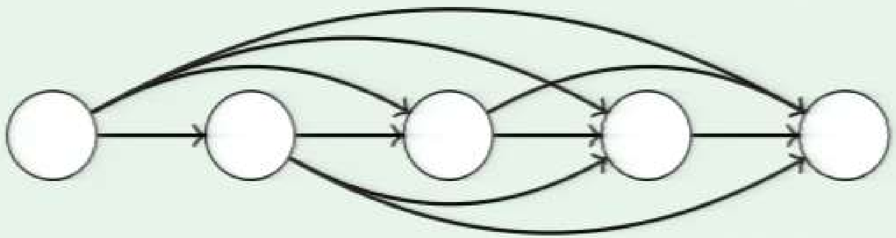
Issues with directed observation graphs

Orientation of edges reduces feedback – regret will increase

$\sum_a \mathbb{P}\big(X_t = a \mid \ell_t(a) \text{ is observed}\big)$ can be large even when $\alpha(G)$ is small

Example

$G =$ total order on K actions



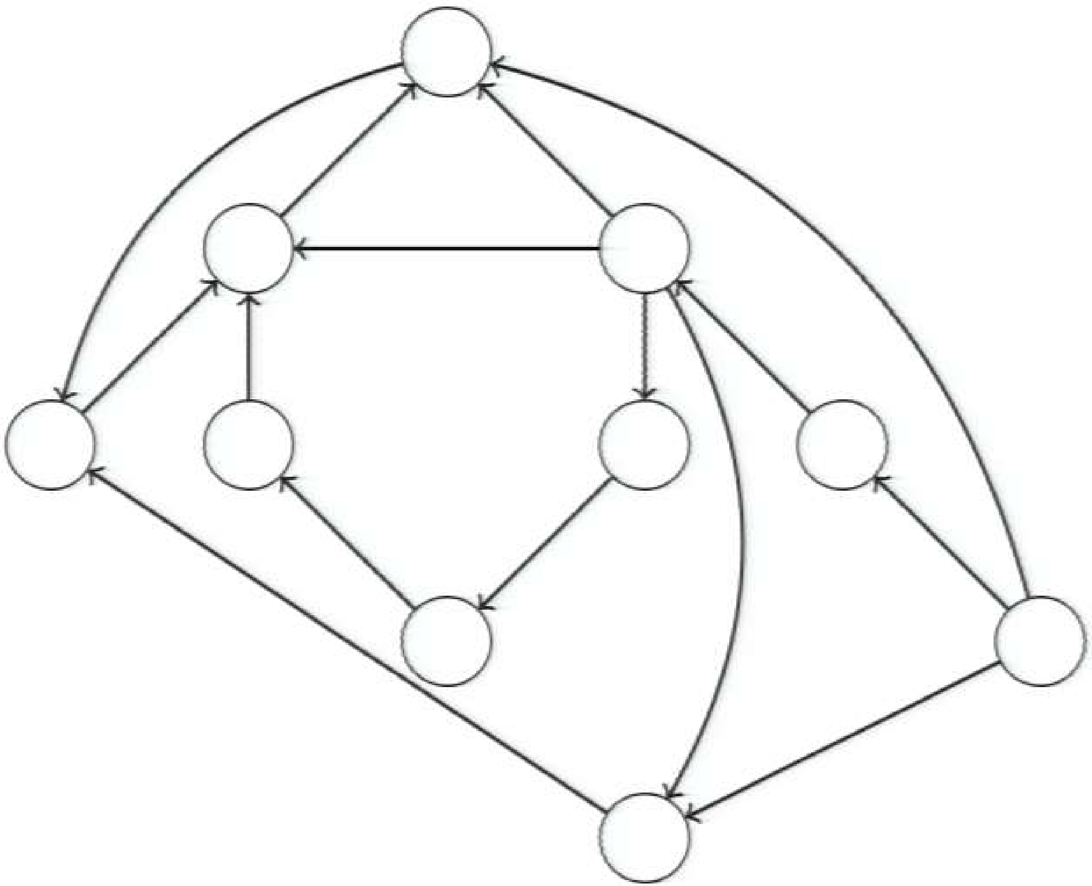
$\alpha(G) = 1$
ignoring orientation

There exists a distribution $\mathbb{P}(X_t = a) \quad a = 1, \dots, K$ such that

$$\sum_a \mathbb{P}\big(X_t = a \mid \ell_t(a) \text{ is observed}\big) = \frac{K+1}{2}$$

Domination number

The size of the smallest dominating set



Exp3-DOM for directed observation graphs

- $\mathbb{P}\big(X_t = a \mid \ell_t(a) \text{ is observed}\big)$ is controlled by mixing Exp3-SET with the uniform distribution over a dominating set of G
- Greedy approximation of dominating set is OK

Exp3-DOM for directed observation graphs

- $\mathbb{P}(X_t = a \mid \ell_t(a) \text{ is observed})$ is controlled by mixing Exp3-SET with the uniform distribution over a dominating set of G
- Greedy approximation of dominating set is OK

Key lemma for directed observation graphs

$$\sum_a \mathbb{P}(X_t = a \mid \ell_t(a) \text{ is observed}) = \mathcal{O}\big(\alpha(G)\ln(KT)\big)$$

Proof uses Turán's Theorem relating the independence number of a graph to its density

This gives regret $R_T = \mathcal{O}\Big((\ln K)\sqrt{T\alpha(G)\ln(KT)}\Big)$

Conclusions

- In the undirected case G_t can be revealed after predicting
- Lack of feedback caused by edge orientation costs only log factors in the regret
- Weaker result for directed case when G_t is only revealed after predicting. Is this inevitable?

NIPS Thanks Its Sponsors

AIR FORCE OFFICE OF SCIENTIFIC RESEARCH
UNITED STATES AIR FORCE

amazon.com

Microsoft Research

Google

facebook

SKYTREE
THE MACHINE LEARNING COMPANY

TWO SIGMA

United Technologies Research Center

D E Shaw & Co

xerox

YAHOO!
LABS

IBM Research

TOYOTA

DRW TRADING GROUP

millionshort

criteo

PDT PARTNERS

Springer
Machine Learning Journal

Disney Research

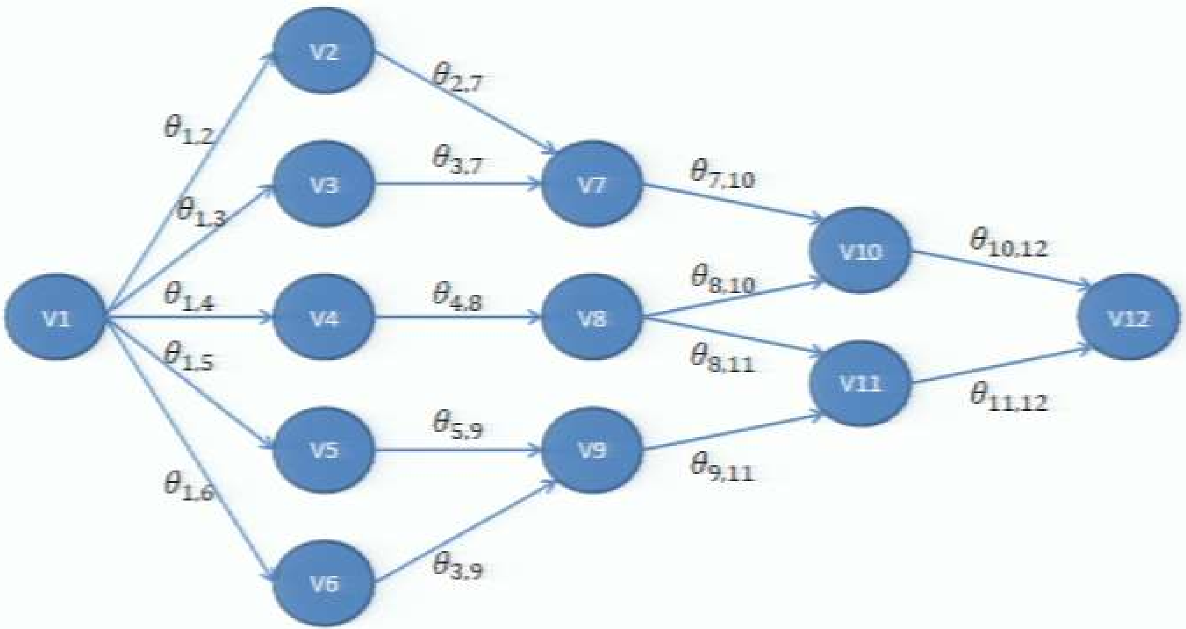
Eluder Dimension and the Sample Complexity of Optimistic Exploration

Daniel Russo
Joint Work with Prof. Benjamin Van Roy

Stanford University

NIPS 2013

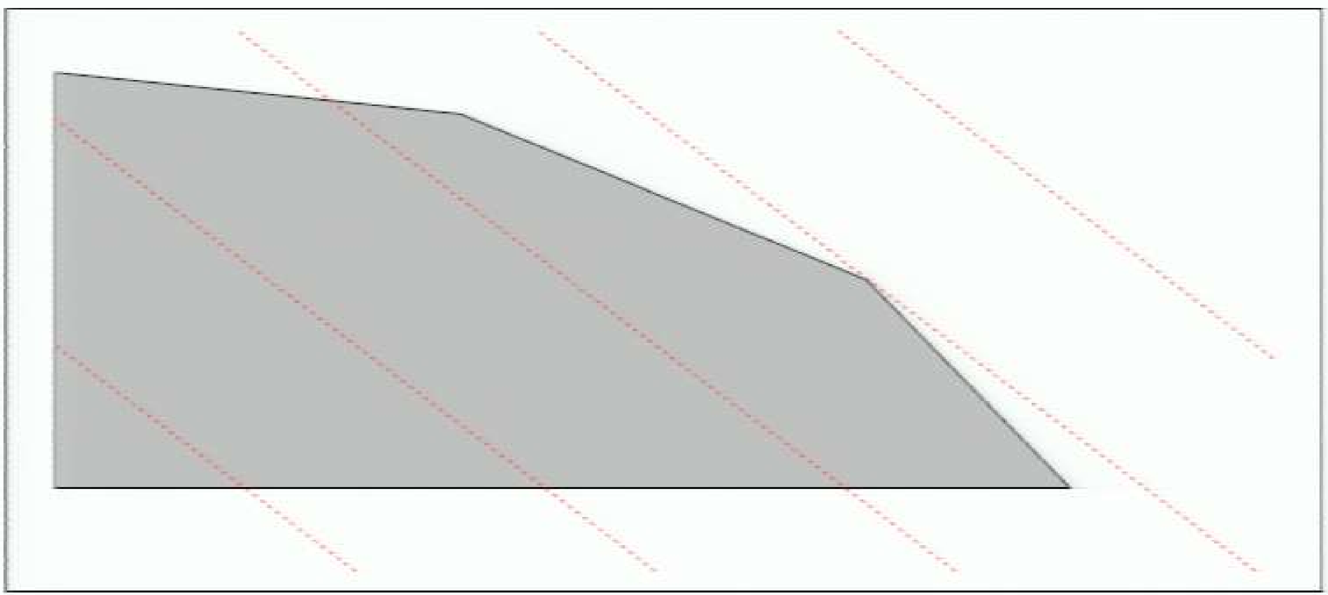
Online Shortest Path Problem with Bandit Feedback



- Repeatedly route packets from $V1$ to $V12$.
- Unknown $\theta_{i,j}$ specifies the mean time to travel between Vi and Vj.
- Observe the total routing time of each packet.
- Goal: Minimize the cumulative routing time of many packets.
- An example of a "linear bandit" problem.

Linear Bandit Problems

- Action space: $\mathcal{A}$
- Feature map: $\phi : \mathcal{A} \to \mathbb{R}^d$
- Mean reward of action $a \in \mathcal{A}$ is $\phi(a)^T\theta$
- $\theta \in \Theta \subset \mathbb{R}^d$ is unknown.
- Goal: Learn to solve $\max_{a \in \mathcal{A}} \phi(a)^T\theta$



Convergence to Optimality

- The agent can learn without exploring every possible action.

The work of Dani et al. (2008), Rusmevichientong and Tsitsiklis (2010), and Abbasi-Yadkori et al. (2011) yields *tight* regret bounds of order

$$d\sqrt{T}$$

- Bounds exhibit no dependence on the number of actions

Convergence to Optimality

- The agent can learn without exploring every possible action.

The work of Dani et al. (2008), Rusmevichientong and Tsitsiklis (2010), and Abbasi-Yadkori et al. (2011) yields *tight* regret bounds of order

$$d\sqrt{T}$$

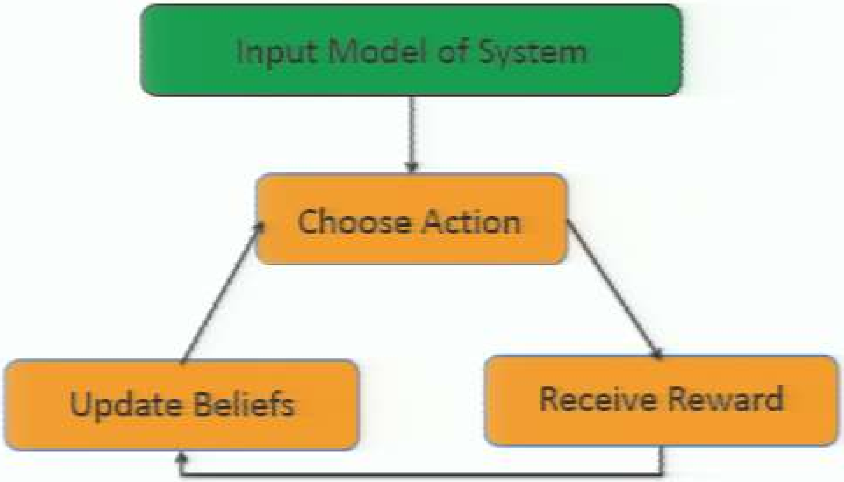
- Bounds exhibit no dependence on the number of actions
- What about more general model classes?

A General Multiarmed Bandit

- We want to solve

$$\max_{a \in \mathcal{A}} f_\theta(a)$$

- Know $f_\theta \in \mathcal{F} = \{f_\rho : \rho \in \Theta\}$
- Beliefs about $\theta \in \Theta$ may be encoded in terms of prior distribution.
- Agent sequentially chooses actions $A_1, A_2, \ldots$
- Choosing action A_t yields random reward with mean $f_\theta(A_t)$.



A General Multiarmed Bandit

- Evaluate the performance up to time T by regret:

$$\text{Regret}(T) = \sum_{t=1}^{T} \left[\underbrace{f_\theta(A^*)}_{\text{optimal action}} - \underbrace{f_\theta(A_t)}_{\text{selected action}} \right]$$

Theoretical Guarantees

Provide upper bounds on expected regret of order

$$\sqrt{\underbrace{\dim_E\left(\mathcal{F}, T^{-2}\right)}_{\text{Eluder dimension}} \underbrace{\log\left(N\left(\mathcal{F}, T^{-2}, \|\cdot\|_\infty\right)\right)}_{\text{log–covering number}} T}.$$

Theoretical Guarantees

Provide upper bounds on expected regret of order

$$\sqrt{\underbrace{\dim_E\left(\mathcal{F}, T^{-2}\right)}_{\text{Eluder dimension}} \underbrace{\log\left(N\left(\mathcal{F}, T^{-2}, \|\cdot\|_\infty\right)\right)}_{\text{log–covering number}} T}.$$

- Log–covering number:
  - Sensitivity to statistical over-fitting.
  - Closely related to concepts from statistical learning theory.

Theoretical Guarantees

Provide upper bounds on expected regret of order

$$\sqrt{\underbrace{\dim_E\left(\mathcal{F}, T^{-2}\right)}_{\text{Eluder dimension}} \underbrace{\log\left(N\left(\mathcal{F}, T^{-2}, \|\cdot\|_\infty\right)\right)}_{\text{log–covering number}} T}.$$

- Log–covering number:
  - Sensitivity to statistical over-fitting.
  - Closely related to concepts from statistical learning theory.
- Eluder dimension:
  - How does sampling one action reduce uncertainty about others?
  - A new notion we introduce.

Theoretical Guarantees

Provide upper bounds on expected regret of order

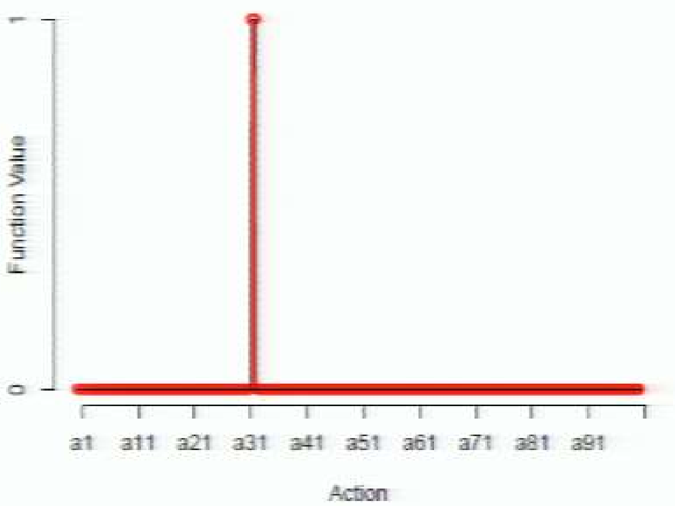
$$\sqrt{\underbrace{\dim_E\left(\mathcal{F}, T^{-2}\right)}_{\text{Eluder dimension}} \underbrace{\log\left(N\left(\mathcal{F}, T^{-2}, \|\cdot\|_\infty\right)\right)}_{\text{log–covering number}} T},$$

- Bound holds for *Thompson Sampling* and a general *UCB algorithm*.
- Matches the best bounds available for UCB algorithms when specialized to linear or generalized linear models.

What about VC Dimension?

Fix problem:

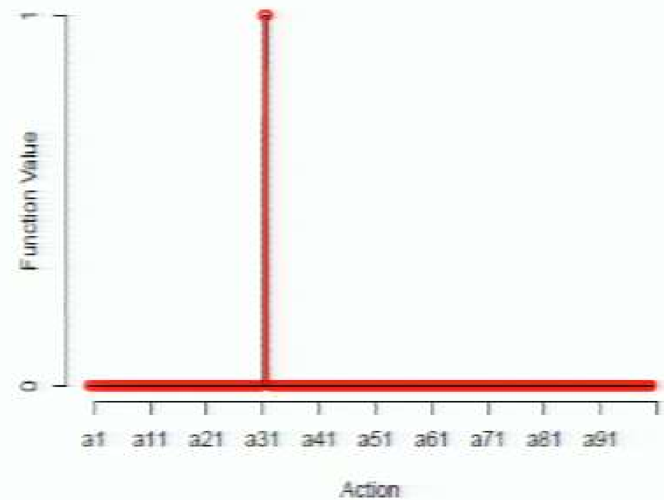
- $\mathcal{A} = \{a_1, ..., a_n\}$
- $\mathcal{F} = \{f_1, ..., f_n\}$
- $f_i(a) = \mathbf{1}_{\{a=a_i\}}$



What about VC Dimension?

Fix problem:

- $\mathcal{A} = \{a_1, ..., a_n\}$
- $\mathcal{F} = \{f_1, .., f_n\}$
- $f_i(a) = \mathbf{1}_{\{a=a_i\}}$



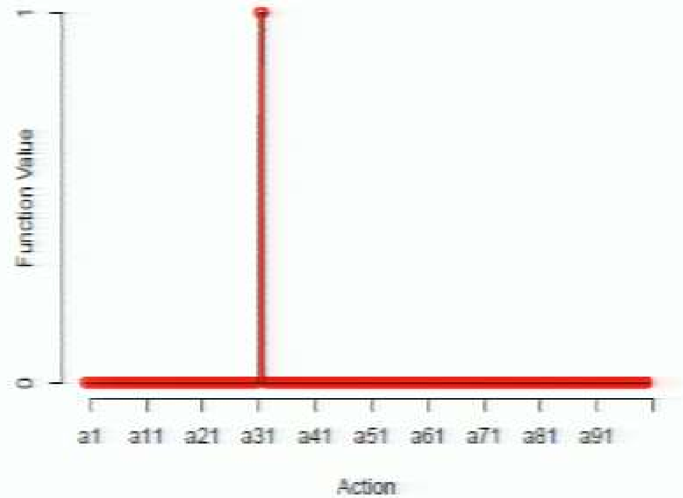
A noiseless prediction problem: Suppose A_t drawn uniformly from $\mathcal{A}$.

- $\mathrm{Dim}_{\mathrm{VC}}(\mathcal{F}) = 1$
- Always predicting $f(A_t) = 0$ already yields error rate of $1/n$.

What about VC Dimension?

Fix problem:

- $\mathcal{A} = \{a_1, ..., a_n\}$
- $\mathcal{F} = \{f_1, .., f_n\}$
- $f_i(a) = \mathbf{1}_{\{a=a_i\}}$



A multiarmed bandit problem: Suppose f_θ drawn uniformly from $\mathcal{F}$, then until the optimal action is identified,

1. Regret per round is 1
2. At most a single function is ruled out per round

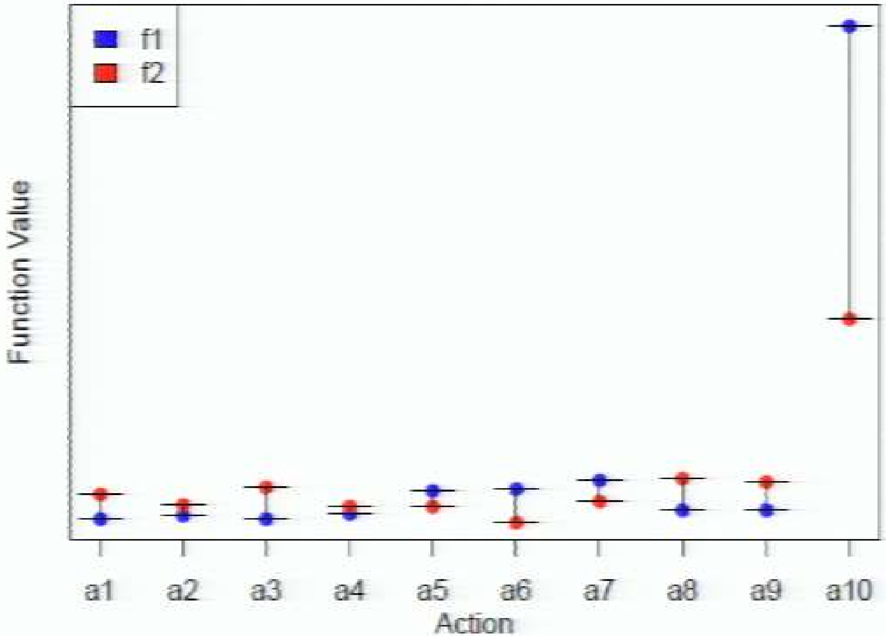
- Regret scales *linearly* with n.

Defining Eluder Dimension

- A politician sequentially presents information to reporters.
- But each piece of information must be *new*.
- How long can he continue?

Defining Eluder Dimension

An action a is independent of $\{a_1, ..., a_n\}$ if two functions that make similar predictions at $\{a_1, ..., a_n\}$ could differ significantly at a.

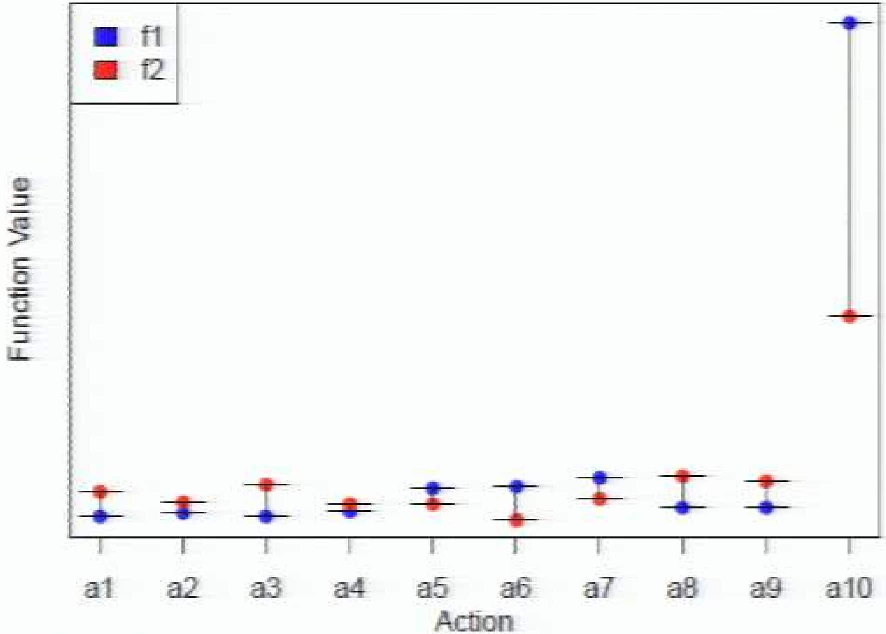


Defining Eluder Dimension

Definition

$a \in \mathcal{A}$ is ϵ-*independent* of $\{a_1, ..., a_n\} \subseteq \mathcal{A}$ with respect to $\mathcal{F}$ if

- there exist $f, \tilde{f} \in \mathcal{F}$ satisfying
  1. $\sqrt{\sum_{i=1}^{n}(f(a_i) - \tilde{f}(a_i))^2} \leq \epsilon$
  2. $f(a) - \tilde{f}(a) > \epsilon$.

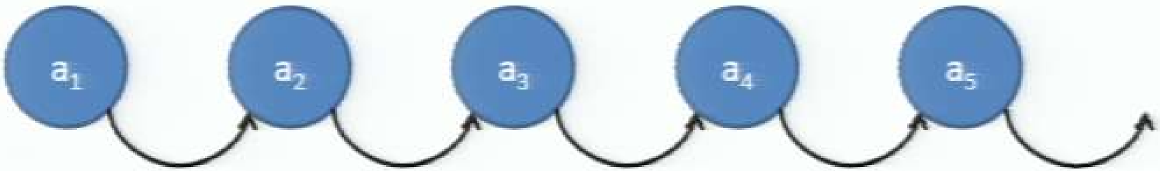


Defining Eluder Dimension

The eluder dimension is the length of the longest independent sequence.

Definition

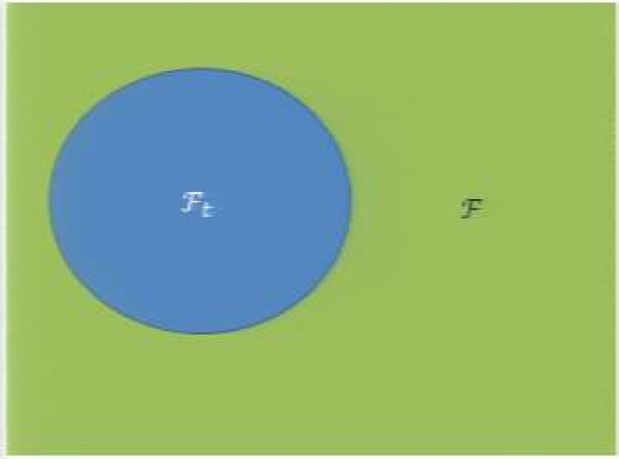
$\dim_E(\mathcal{F}, \epsilon)$ is the length of the longest sequence of elements in $\mathcal{A}$ such that, for some $\epsilon' \geq \epsilon$, every element is ϵ'-independent of its predecessors.

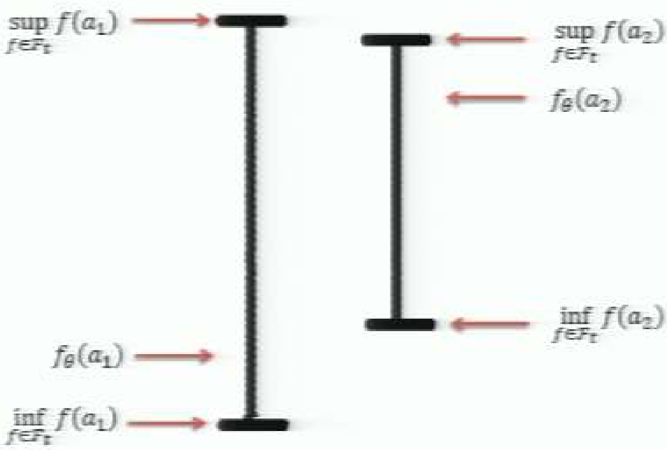


Optimism in the face of uncertainty

Act according to an "optimistic" model of the environment

1. $\mathcal{F}_t \leftarrow$ subset of $f \in \mathcal{F}$ that are statistically plausible given data.
2. Play $\overline{A}_t \in \arg\max_{a \in \mathcal{A}} \left\{ \sup_{f \in \mathcal{F}_t} f(a) \right\}$.





Optimism in the face of uncertainty

Act according to an "optimistic" model of the environment

1. $\mathcal{F}_t \leftarrow$ subset of $f \in \mathcal{F}$ that are statistically plausible given data.
2. Play $\overline{A}_t \in \arg\max_{a \in \mathcal{A}} \left\{ \sup_{f \in \mathcal{F}_t} f(a) \right\}$.

There is a huge literature on this approach:

- Bandit problems with independent arms
  - (Lai–Robins, 1985), (Lai, 1987), (Auer, 2002), (Audibert, 2009)...
- Bandit problems with dependent arms
  - (Rusmevichientong-Tsitsiklis 2010), (Filippi et. al, 2010), (Srinivas et. al, 2012)...
- Reinforcement Learning
  - (Kearns–Singh, 2002), (Bartlett–Terwari, 2009), (Jaksch et. al 2010)...
- Monte Carlo Tree Search
  - (Kocsis–Szepesvári, 2006)...

A posterior sampling strategy

"Thompson sampling" & "probability matching":

- Sample each action according to the posterior probability it is optimal.
- Generated a lot of recent interest.

Our paper *Learning to Optimize via Posterior Sampling*

- establishes a close connection with optimistic algorithms.
- implies our analysis also bounds the *Bayesian regret* of TS.

Proof sketch

$$\sqrt{\underbrace{\dim_E\left(\mathcal{F}, T^{-2}\right)}_{\text{Eluder dimension}}\underbrace{\log\left(N\left(\mathcal{F}, T^{-2}, \|\cdot\|_\infty\right)\right)}_{\text{log–covering number}} T}$$

1. Build generic confidence sets $\mathcal{F}_t \subset \mathcal{F}$
   - Size of $\mathcal{F}_t$ depends on the log–covering number of $\mathcal{F}$.
2. Measure the rate at which confidence intervals shrink.
   - Depends on the eluder dimension of $\mathcal{F}$.

Conclusion

- MABs require fundamentally different notions of model complexity.
- Huge value in having a unified conceptual understanding.
- *Much more work is needed*

This work:

- A step toward this goal.

NIPS Thanks Its Sponsors

AIR FORCE OFFICE OF SCIENTIFIC RESEARCH · UNITED STATES AIR FORCE

amazon.com

Microsoft Research

Google

facebook

SKYTREE. THE MACHINE LEARNING COMPANY

TWO SIGMA

United Technologies Research Center

D E Shaw & Co

xerox

YAHOO! LABS

IBM Research

DRW TRADING GROUP

TOYOTA

millionshort

criteo

PDT PARTNERS

Springer Machine Learning Journal

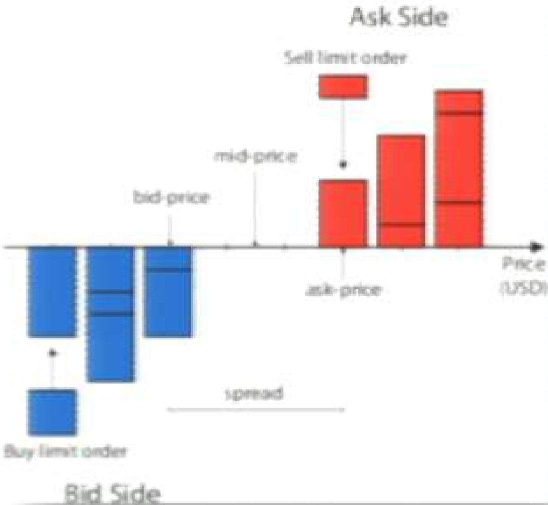
Disney Research

Adaptive Market-Making via Online Learning

Jacob Abernethy (U. Michigan Ann Arbor)
Satyen Kale (Yahoo! Labs)

In Stock Market, with Whom do you Trade?

- Generally, there's an *order book*
- Order book specifies at any time how many shares are up for bid and offer
- Traders can interact with order book via *market* and *limit* orders



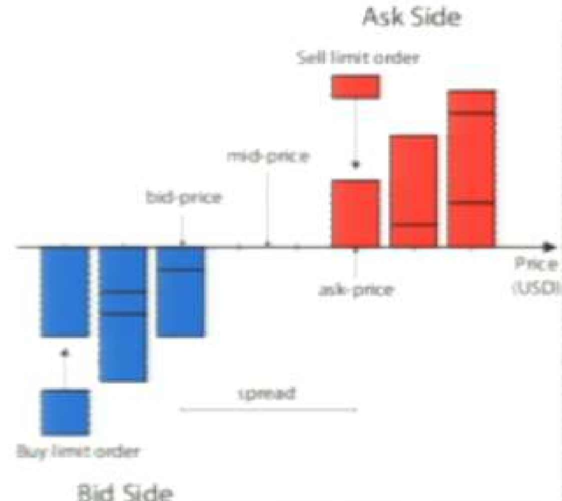
Bitcoin/USD order book on 12/6/2013 (MTGOX.com)

Buying			Selling		
Sum	Size	Bid	Ask	Size	Sum
0.1	0.056	903.76573	912.99	23.0939	23.1
0.1	0.0	902.0	913.99614	0.0375	23.1
0.1	0.056	901.96181	913.99999	10.2	33.3
1.7	1.6124	901.21	914.0	40.4699	73.8
11.1	9.337	901.2	914.74993	0.036	73.8
11.1	0.0432	901.01	915.0	46.9417	120.8
13.5	2.38	901.0	916.58309	0.026	120.8
13.5	0.01	900.9001	917.9	10.0	130.8
15.6	2.1	900.0	918.0	7.3681	138.2
15.6	0.01	898.58373	918.41992	0.054	138.2

Market Makers = Liquidity Providers

Market makers provide liquidity to financial markets:

- Quote both buy and sell prices
- Profit from *bid-ask spread*, i.e. difference in buy and sell prices
- Counterparty for transactions



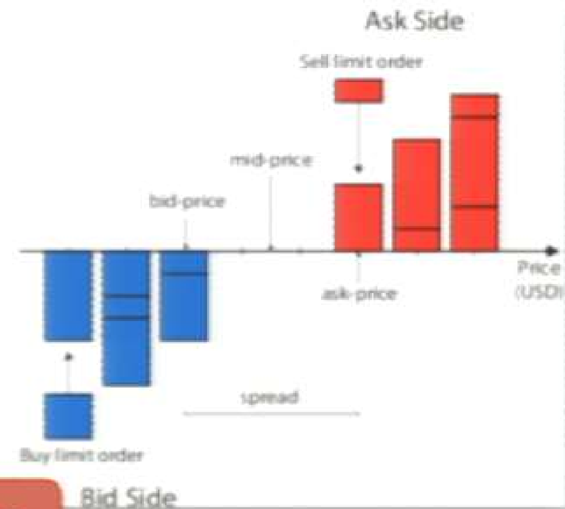
Bitcoin/USD order book on 12/6/2013 (MTGOX.com)

Buying				Selling	
Sum	Size	Bid	Ask	Size	Sum
0.1	0.056	903.76573	912.99	23.0939	23.1
0.1	0.0	902.0	913.99614	0.0375	23.1
0.1	0.056	901.96181	913.99999	10.2	33.3
1.7	1.6124	901.21	914.0	40.4699	73.8
11.1	9.337	901.2	914.74993	0.036	73.8
11.1	0.0432	901.01	915.0	46.9417	120.8
13.5	2.38	901.0	916.58309	0.026	120.8
13.5	0.01	900.9001	917.9	10.0	130.8
15.6	2.1	900.0	918.0	7.3681	138.2
15.6	0.01	898.58373	918.41992	0.054	138.2

Market Makers = Liquidity Providers

Market makers provide liquidity to financial markets:

- Quote both buy and sell prices
- Profit from *bid-ask spread*, i.e. difference in buy and sell prices
- Counterparty for transactions



Spread = $9.22

Bitcoin/USD order book o[illegible]/6/2013 (MTGOX.com)

Buying				Selling	
Sum	Size	Bid	Ask	Size	Sum
0.1	0.056	903.76573	912.99	23.0939	23.1
0.1	0.0	902.0	913.99614	0.0375	23.1
0.1	0.056	901.96181	913.99999	10.2	33.3
1.7	1.6124	901.21	914.0	40.4699	73.8
11.1	9.337	901.2	914.74993	0.036	73.8
11.1	0.0432	901.01	915.0	46.9417	120.8
13.5	2.38	901.0	916.58309	0.026	120.8
13.5	0.01	900.9001	917.9	10.0	130.8
15.6	2.1	900.0	918.0	7.3681	138.2
15.6	0.01	898.58373	918.41992	0.054	138.2

THIS TALK:
Designing Adaptive Market Makers

- We present and analyze "Spread-based Market Making"
- We ask, how can we set the critical parameter, the bid-ask spread, **adaptively**?
- We apply an experts (online learning) strategy. Problem: How to manage inventory switching costs?
- Theoretical results: switching costs are "not too bad"
- Empirical results: often our adaptive market maker does *better* than the best bid-ask spread.

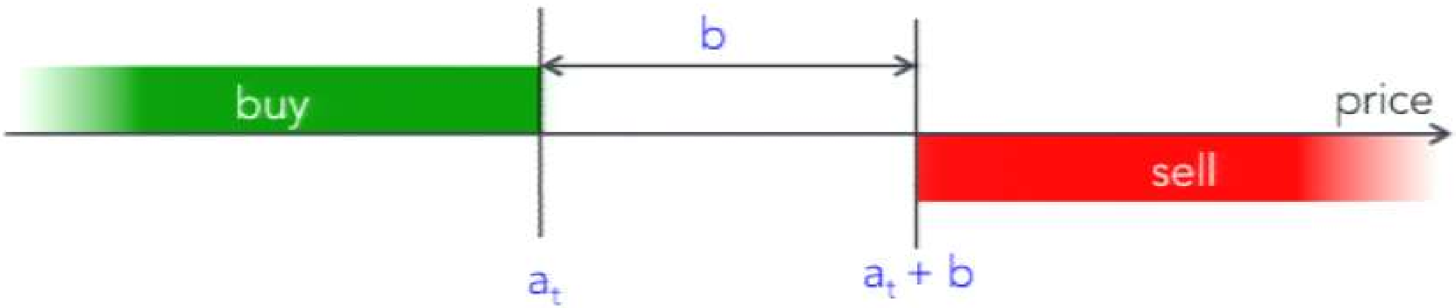
Online Market Making

At time $t = 1, 2, \ldots, T$

- Market maker places buy/sell orders
- Market maker observes price p_t (may be adversarially generated)
- Market maker executes applicable orders

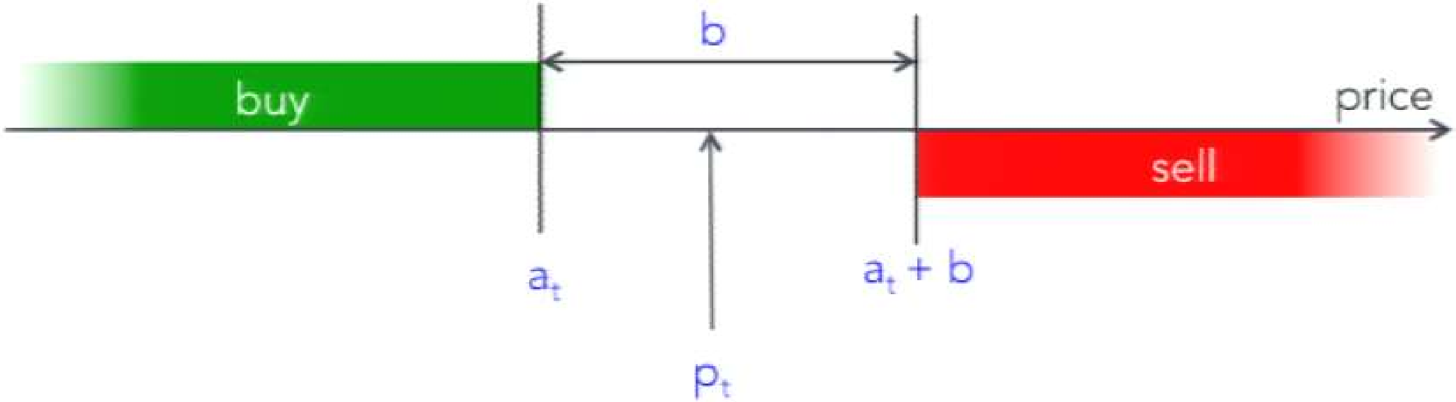
Spread-based strategies

Spread size parameter b
Window $[a_t, a_t + b]$



Spread-based strategies

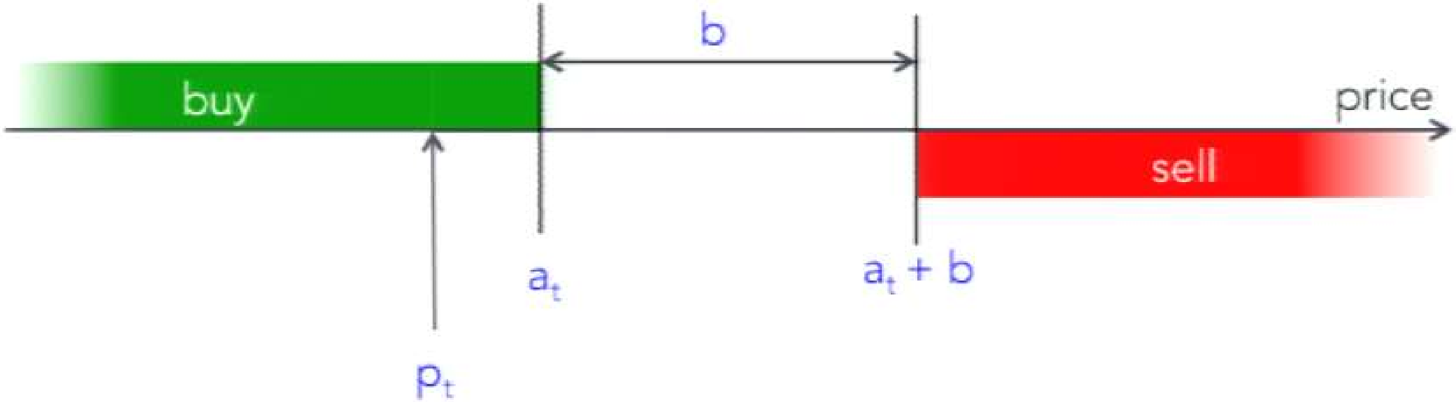
Spread size parameter b
Window $[a_t, a_t + b]$



Current price p_t in window:
no transactions, no change in window

Spread-based strategies

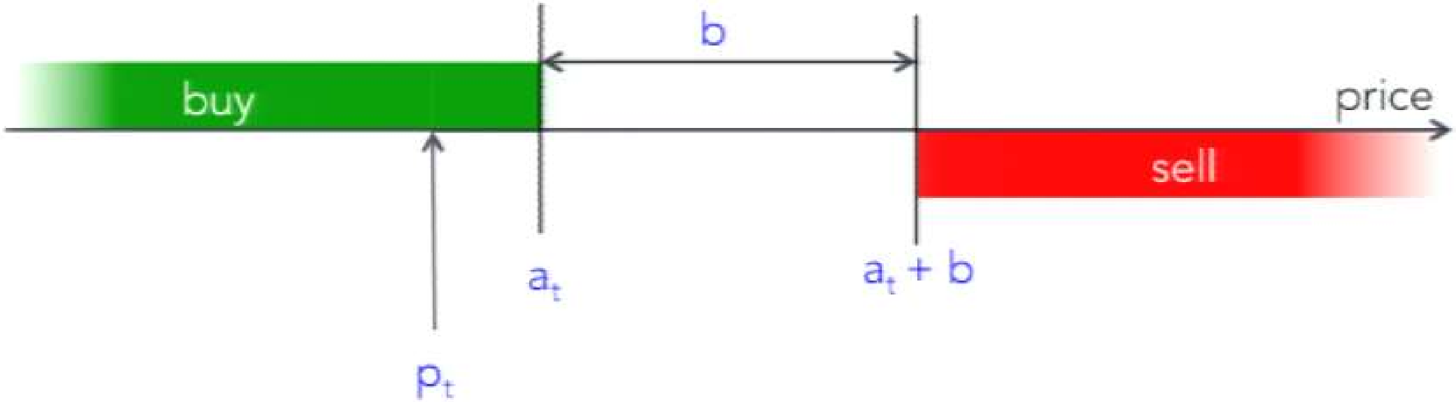
Spread size parameter b
Window $[a_t, a_t + b]$



$p_t < a_t$

Spread-based strategies

Spread size parameter b
Window $[a_t, a_t + b]$

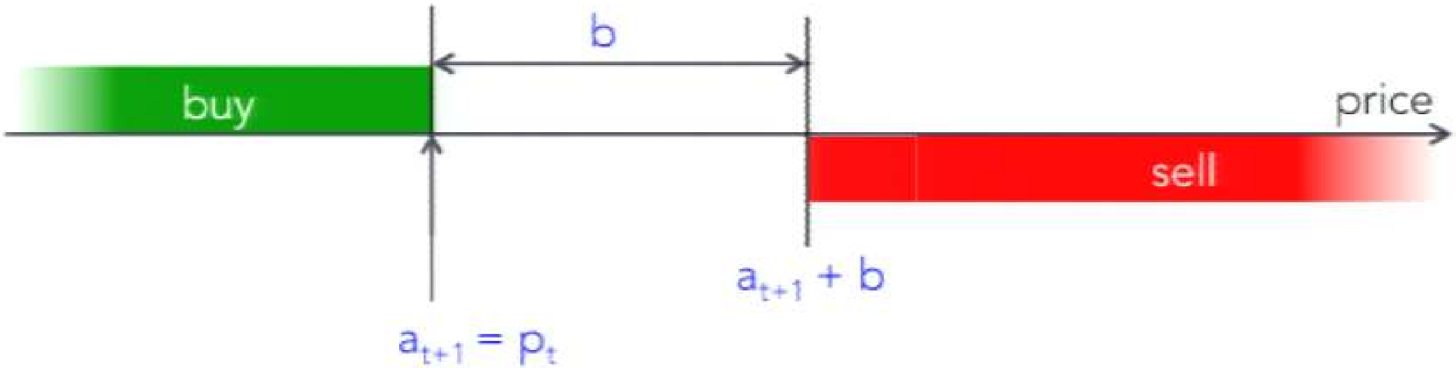


$p_t < a_t$

Window moved so that $a_{t+1} = p_t$
Buy $a_t - p_t$ shares

Spread-based strategies

Spread size parameter b
Window $[a_t, a_t + b]$

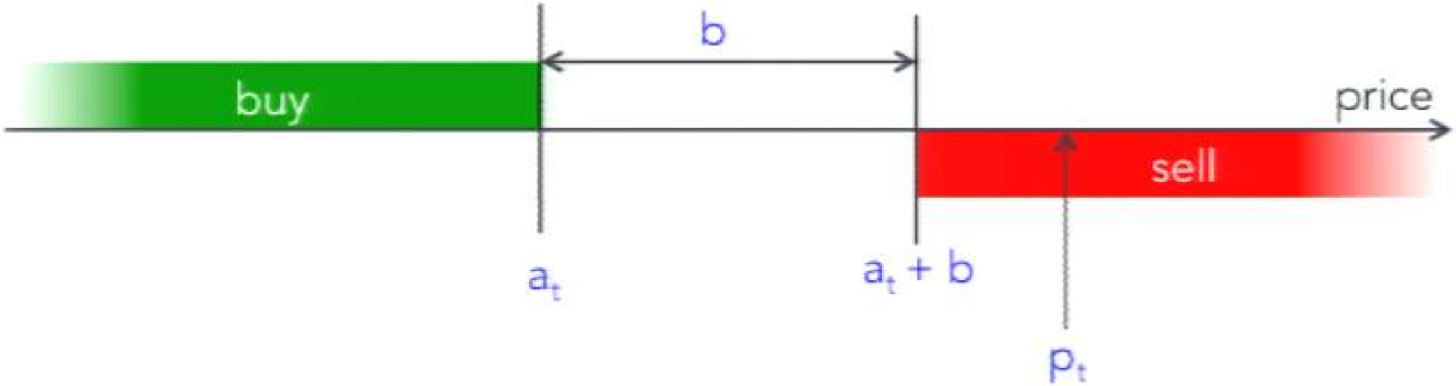


$p_t < a_t$

Window moved so that $a_{t+1} = p_t$
Buy $a_t - p_t$ shares

Spread-based strategies

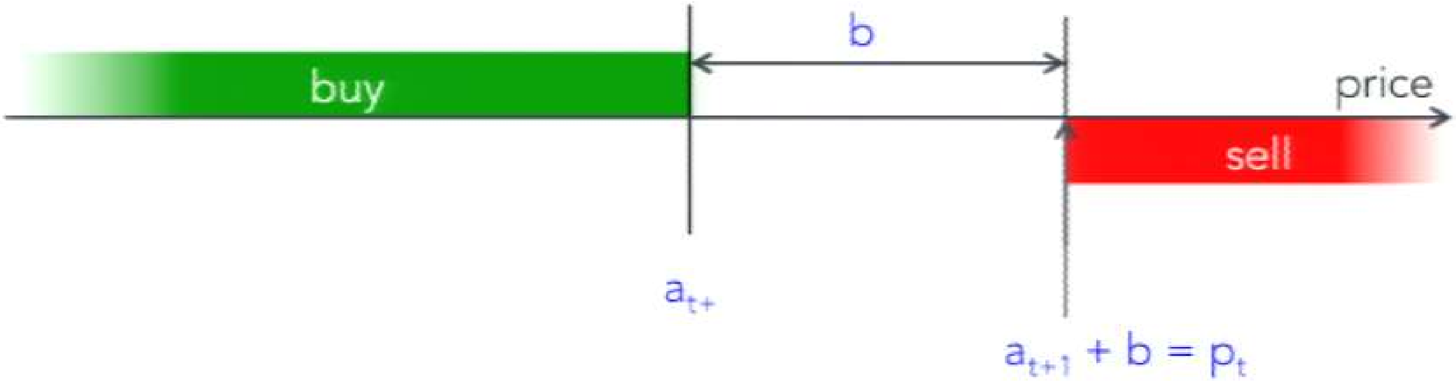
Spread size parameter b
Window $[a_t, a_t + b]$



$p_t > a_t + b$

Spread-based strategies

Spread size parameter b
Window $[a_t, a_t + b]$



$p_t > a_t + b$
Window moved so that $a_{t+1} + b = p_t$
Sell $p_t - (a_t + b)$ shares

Spread-Based Market Making

- Upside: Spread b implies buy and sell orders are matched to yield a profit of b
  - i.e. shares that are bought at some price are immediately offered for sale at a price b units higher
- Downside: price fluctuations within window yield no profit

Spread-Based Market Making

- **Upside:** Spread b implies buy and sell orders are matched to yield a profit of b
  - i.e. shares that are bought at some price are immediately offered for sale at a price b units higher
- **Downside:** price fluctuations within window yield no profit
- **Theorem:** spread b strategy payoff is at least

$$\sum_{t=1}^{T} \frac{b}{2}|a_{t+1} - a_t| - (|a_{T+1} - a_1| + b)^2$$

Adaptive Spread Selection

- How to adaptively choose the spread for market making?

Adaptive Spread Selection

- How to adaptively choose the spread for market making?
  - given a set B of different spread sizes, is it possible to adaptively choose spread to compete with the best spread in hindsight?
  - Regret = payoff(best strategy using spread in B) - payoff(algorithm)

Adaptive Spread Selection

- How to adaptively choose the spread for market making?
  - given a set B of different spread sizes, is it possible to adaptively choose spread to compete with the best spread in hindsight?
  - Regret = payoff(best strategy using spread in B) - payoff(algorithm)

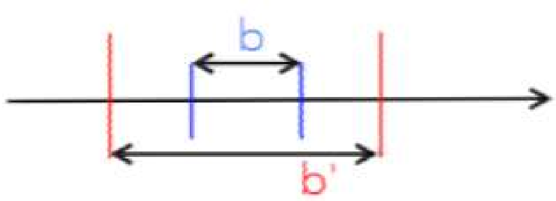
- Challenge: different *states*, positions in stock held by different spread-based strategies can be different
  - Typical online expert learning algorithms assume no state

Adaptive Spread Selection

- How to adaptively choose the spread for market making?
  - given a set B of different spread sizes, is it possible to adaptively choose spread to compete with the best spread in hindsight?
  - Regret = payoff(best strategy using spread in B) - payoff(algorithm)
- Challenge: different *states*, positions in stock held by different spread-based strategies can be different
  - Typical online expert learning algorithms assume no state
- Main Theorem: adaptive algorithm with $O(\sqrt{T})$ regret after T steps

How to Handle State

Nesting lemma: for two spreads b < b′, if initially the window for b is nested in that for b′, then it remains nested.



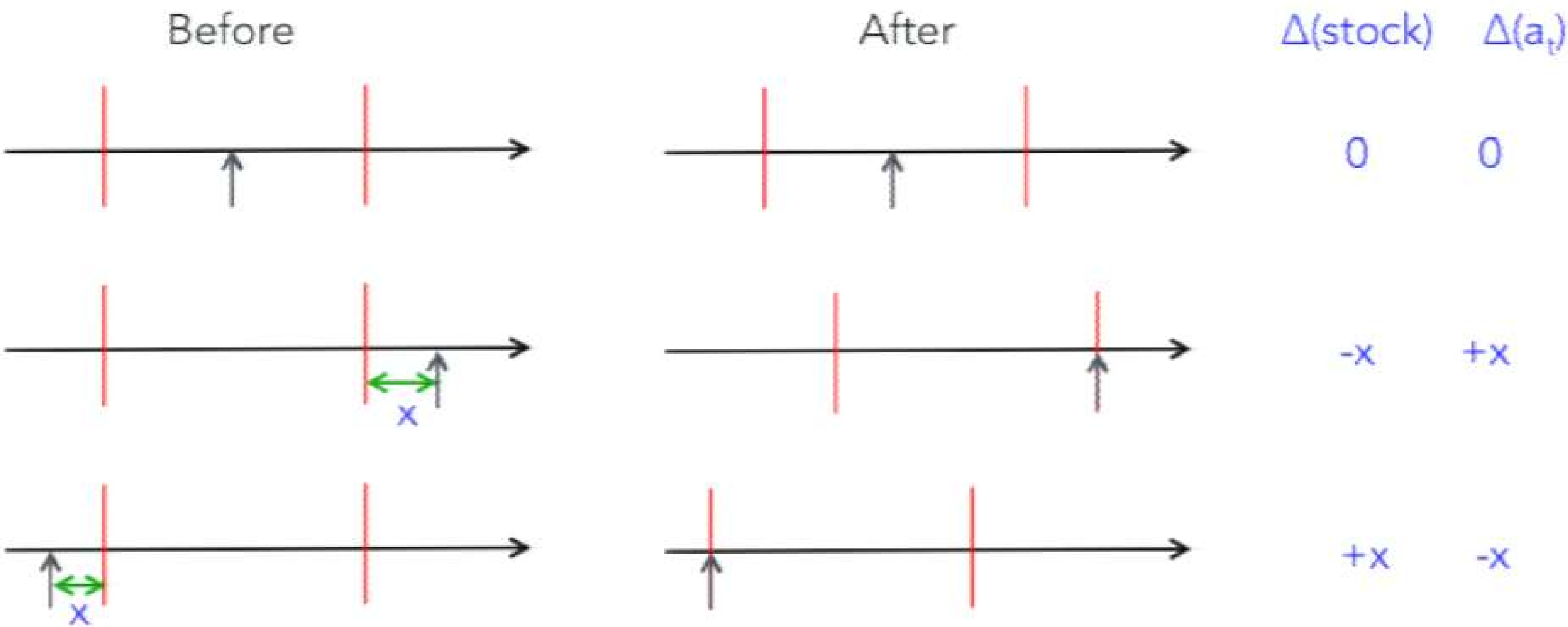
How to Handle State

Invariance lemma: for any strategy (stock position) + a_t is invariant over t.

How to Handle State

Invariance lemma: for any strategy (stock position) + a_t is invariant over t.

Proof by picture



How to Handle State

Nesting lemma: for two spreads $b < b'$, if initially the window for b is nested in that for b', then it remains nested.

Invariance lemma: for any strategy (stock position) + a_t is invariant over t.

How to Handle State

Nesting lemma: for two spreads $b < b'$, if initially the window for b is nested in that for b', then it remains nested.

Invariance lemma: for any strategy (stock position) + a_t is invariant over t.

Upshot: difference in state between strategies is bounded!

How to Handle State

Nesting lemma: for two spreads $b < b'$, if initially the window for b is nested in that for b', then it remains nested.

Invariance lemma: for any strategy (stock position) + a_t is invariant over t.

Upshot: difference in state between strategies is bounded!

Algorithm:

- Run an experts algorithm (eg. MW, FPL) over strategies
- For any t:
  - if strategy chosen at t is not the one from t-1, then buy/sell stock to match the new state
  - use same buy/sell orders as newly chosen strategy

Regret Minimizing Algorithm

Algorithm:

- Run an experts algorithm (eg. MW, FPL) over strategies
- For any t:
  - if strategy chosen at t is not the one from $t-1$, then buy/sell stock to match the new state
  - use same buy/sell orders as newly chosen strategy

Regret Minimizing Algorithm

Algorithm:

- Run an experts algorithm (eg. MW, FPL) over strategies
- For any t:
  - if strategy chosen at t is not the one from $t-1$, then buy/sell stock to match the new state
  - use same buy/sell orders as newly chosen strategy

Regret theorem: bounded cost of state change implies
(regret of algorithm) = (regret of experts alg) + (number of expert changes)

Regret Minimizing Algorithm

Algorithm:

- Run an experts algorithm (eg. MW, FPL) over strategies
- For any t:
  - if strategy chosen at t is not the one from t-1, then buy/sell stock to match the new state
  - use same buy/sell orders as newly chosen strategy

Regret theorem: bounded cost of state change implies
(regret of algorithm) = (regret of experts alg) + (number of expert changes)

For either MW or FPL, regret and number of expert changes both $O(\sqrt{T})$
Hence, regret of algorithm using MW or FPL is $O(\sqrt{T})$

Experiments

- Stock price data for MSFT, HPQ, and WMT downloaded from www.netfonds.no
  - For 5 days from May 6-10, 2013
  - 7,000 – 38,000 trades
  - Price quotes rounded to nearest cent
- Spread params (in cents) B = {1, 2, 3, 4, 5, 10, 20, 40, 80, 100}
- Implemented algorithm with MW, FPL; compared to simple uniform averaging, simple FTL, and best in hindsight

Results

Symbol	Date	T	Best	MW	FPL	FTL	Unif.
HPQ	5/7/13	13194	558	*620*	-42	19	101
HPQ	5/8/13	12016	186	*340*	-568	-242	-720
HPQ	5/9/13	14804	1058	891	327	214	591
MSFT	5/7/13	34017	1260	1157	1048	1247	64
MSFT	5/8/13	38664	2074	2064	1669	2074	939
MSFT	5/9/13	34386	1813	1803	1534	1811	656
WMT	5/7/13	11309	1333	580	995	918	535
WMT	5/8/13	12966	1372	1300	833	974	926
WMT	5/9/13	10431	2415	2330	1883	1991	1654

Red = best performance
Red italics = beats best in hindsight

Experiments

- Stock price data for MSFT, HPQ, and WMT downloaded from www.netfonds.no
  - For 5 days from May 6-10, 2013
  - 7,000 – 38,000 trades
  - Price quotes rounded to nearest cent
- Spread params (in cents) B = {1, 2, 3, 4, 5, 10, 20, 40, 80, 100}
- Implemented algorithm with MW, FPL; compared to simple uniform averaging, simple FTL, and best in hindsight

Results

Symbol	Date	T	Best	MW	FPL	FTL	Unif.
HPQ	5/7/13	13194	558	*620*	-42	19	101
HPQ	5/8/13	12016	186	*340*	-568	-242	-720
HPQ	5/9/13	14804	1058	891	327	214	591
MSFT	5/7/13	34017	1260	1157	1048	1247	64
MSFT	5/8/13	38664	2074	2064	1669	2074	939
MSFT	5/9/13	34386	1813	1803	1534	1811	656
WMT	5/7/13	11309	1333	580	995	918	535
WMT	5/8/13	12966	1372	1300	833	974	926
WMT	5/9/13	10431	2415	2330	1883	1991	1654

Red = best performance
Red italics = beats best in hindsight

Thank you!

Come by our poster: Sat11

NIPS Thanks Its Sponsors

amazon.com

Microsoft Research

Google

facebook

SKYTREE THE MACHINE LEARNING COMPANY

TWO SIGMA

United Technologies Research Center

D E Shaw & Co

xerox

IBM Research

YAHOO! LABS

TOYOTA

DRW TRADING GROUP

millionshort

criteo

PDT PARTNERS

Springer Machine Learning Journal

Disney Research

Submodular Optimization with Submodular Cover and Submodular Knapsack Constraints (SCSC/ SCSK)

Rishabh Iyer Jeff Bilmes

University of Washington, Seattle

NIPS-2013

MELODI
MachinE Learning, Optimization, & Data Interpretation @ UW

Outline

1. Introduction to Submodular Functions
2. Problem Formulation of SCSC/ SCSK
3. Algorithmic Framework
4. Empirical Results

Set functions $f : 2^V \to \mathbb{R}$

$V =$ 

- V is a finite "ground" set of objects.
- A set function $f : 2^V \to \mathbb{R}$ produces a value for any subset $A \subseteq V$.

Set functions $f : 2^V \to \mathbb{R}$

$$A = \{\ ,\ ,\ ,\ \}$$

- For example, $f(A) = 22$,

Submodular Set Functions

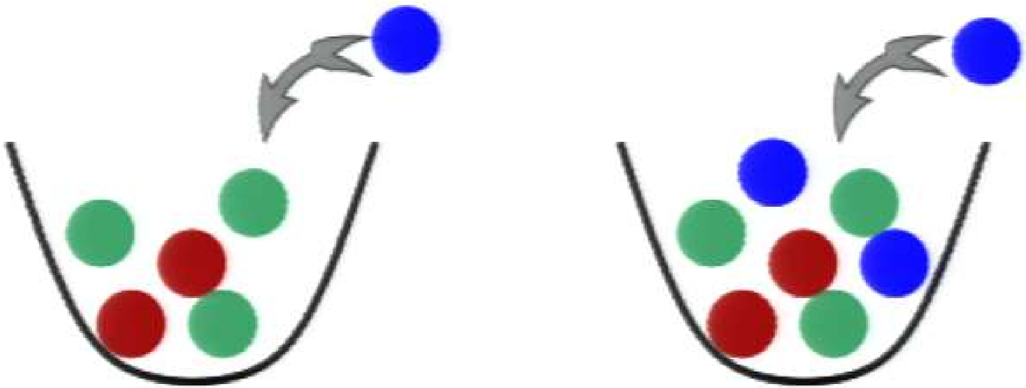
- Special class of set functions.

$$f(A \cup v) - f(A) \geq f(B \cup v) - f(B), \text{ if } A \subseteq B \tag{1}$$

Submodular Set Functions

- Special class of set functions.

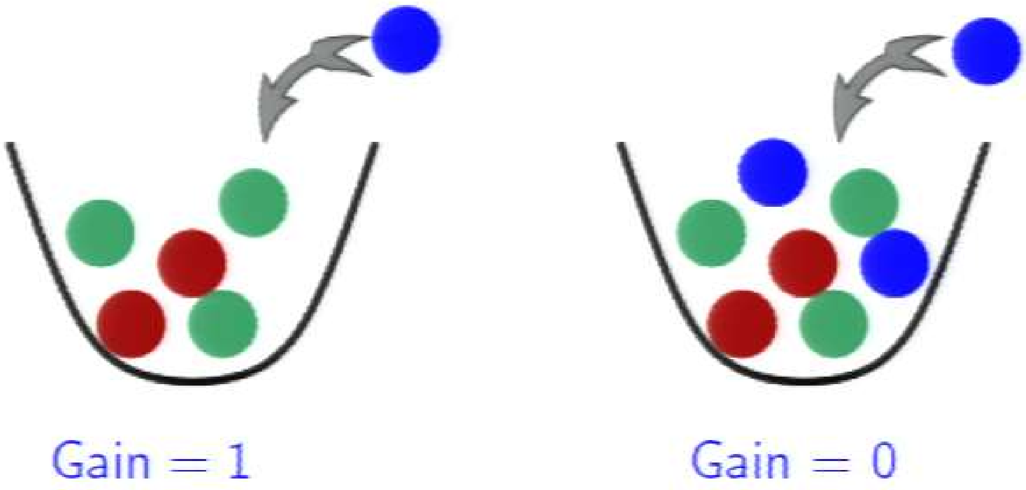
$$f(A \cup v) - f(A) \geq f(B \cup v) - f(B), \text{ if } A \subseteq B \tag{1}$$



Submodular Set Functions

- Special class of set functions.

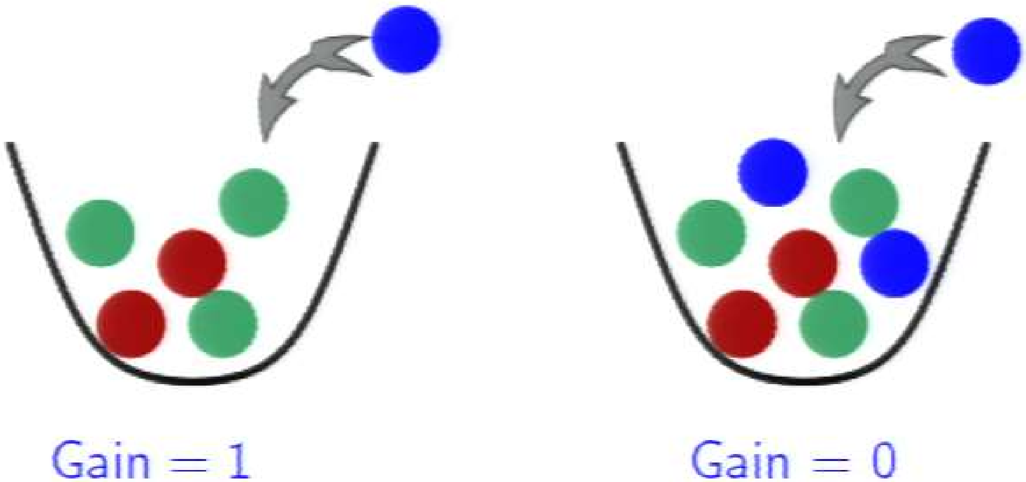
$$f(A \cup v) - f(A) \geq f(B \cup v) - f(B), \text{ if } A \subseteq B \qquad (1)$$



Submodular Set Functions

- Special class of set functions.

$$f(A \cup v) - f(A) \geq f(B \cup v) - f(B), \text{ if } A \subseteq B \quad (1)$$



- Monotonicity: $f(A) \leq f(B)$, if $A \subseteq B$.

Two Sides of Submodularity

Two Sides of Submodularity

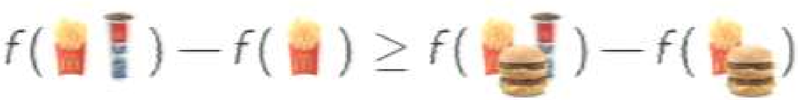
Submodular Minimization

- Solve $\min\{f(X) | X \subseteq V\}$.
- Polynomial-time.
- Relation to convexity.
- Models cooperation.

Two Sides of Submodularity

Submodular Minimization

- Solve $\min\{f(X) | X \subseteq V\}$.
- Polynomial-time.
- Relation to convexity.
- Models cooperation.



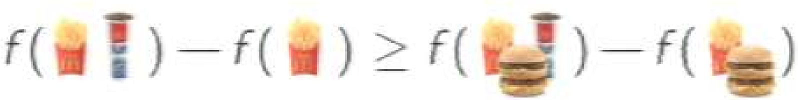
Submodular Maximization

- Solve $\max\{g(X) | X \subseteq V\}$.
- Constant-factor approximable.
- Relation to concavity.
- Models diversity and coverage.

Two Sides of Submodularity

Submodular Minimization

- Solve $\min\{f(X) | X \subseteq V\}$.
- Polynomial-time.
- Relation to convexity.
- Models cooperation.



Submodular Maximization

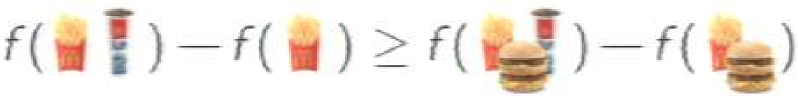
- Solve $\max\{g(X) | X \subseteq V\}$.
- Constant-factor approximable.
- Relation to concavity.
- Models diversity and coverage.

- Sometimes we want to simultaneously maximize coverage/ diversity (g) while minimizing cooperative costs (f).

Two Sides of Submodularity

Submodular Minimization

- Solve $\min\{f(X) | X \subseteq V\}$.
- Polynomial-time.
- Relation to convexity.
- Models cooperation.



Submodular Maximization

- Solve $\max\{g(X) | X \subseteq V\}$.
- Constant-factor approximable.
- Relation to concavity.
- Models diversity and coverage.

- Sometimes we want to simultaneously maximize coverage/ diversity (g) while minimizing cooperative costs (f).
- Often these naturally occur as budget or cover constraints (for example, maximize diversity subject to a budget constraint on the submodular cost).

Submodular Optimization with Submodular Constraints

- Historically: DS optimization

$$\min_{X \subseteq V} f(X) - \lambda g(X)$$

Submodular Optimization with Submodular Constraints

- Historically: DS optimization

Co-operative Costs → Coverage/ Diversity

$$\min_{X \subseteq V} f(X) - \lambda g(X)$$

Submodular Optimization with Submodular Constraints

- Historically: DS optimization

Co-operative Costs → $f(X)$, Coverage/ Diversity → $g(X)$

$$\min_{X \subseteq V} f(X) - \lambda g(X)$$

- Unfortunately, NP hard to approximate (Iyer-Bilmes'12).

Submodular Optimization with Submodular Constraints

- Historically: DS optimization

Co-operative Costs → $f(X)$; Coverage/ Diversity → $g(X)$

$$\min_{X \subseteq V} f(X) - \lambda g(X)$$

- Unfortunately, NP hard to approximate (Iyer-Bilmes'12).
- We introduce the following, which is often more natual anyway:

Submodular Optimization with Submodular Constraints

- Historically: DS optimization

Co-operative Costs → $f(X)$; Coverage/ Diversity → $g(X)$

$$\min_{X \subseteq V} f(X) - \lambda g(X)$$

- Unfortunately, NP hard to approximate (Iyer-Bilmes'12).
- We introduce the following, which is often more natual anyway:

$$\text{SCSC: } \min\{f(X) : g(X) \geq c\}, \quad \text{SCSK: } \max\{g(X) : f(X) \leq b\},$$

Submodular Optimization with Submodular Constraints

- Historically: DS optimization

Co-operative Costs   Coverage/ Diversity

$$\min_{X \subseteq V} f(X) - \lambda g(X)$$

- Unfortunately, NP hard to approximate (Iyer-Bilmes'12).
- We introduce the following, which is often more natual anyway:

Coverage/ Diversity

SCSC: $\min\{f(X) : g(X) \geq c\}$, SCSK: $\max\{g(X) : f(X) \leq b\}$,

Co-operative Costs

Submodular Optimization with Submodular Constraints

- Historically: DS optimization

Co-operative Costs   Coverage/ Diversity

$$\min_{X \subseteq V} f(X) - \lambda g(X)$$

- Unfortunately, NP hard to approximate (Iyer-Bilmes'12).
- We introduce the following, which is often more natual anyway:

Coverage/ Diversity

$$\text{SCSC: } \min\{f(X) : g(X) \geq c\}, \quad \text{SCSK: } \max\{g(X) : f(X) \leq b\},$$

Co-operative Costs

- While DS optimization is NP hard to approximate, SCSC and SCSK however, retain approximation guarantees!

Submodular Optimization with Submodular Constraints

- Historically: DS optimization

Co-operative Costs → Coverage/ Diversity

$$\min_{X \subseteq V} f(X) - \lambda g(X)$$

- Unfortunately, NP hard to approximate (Iyer-Bilmes'12).
- We introduce the following, which is often more natual anyway:

Coverage/ Diversity

SCSC: $\min\{f(X) : g(X) \geq c\}$, SCSK: $\max\{g(X) : f(X) \leq b\}$,

Co-operative Costs

- While DS optimization is NP hard to approximate, SCSC and SCSK however, retain approximation guarantees!
- Throughout this talk, assume f and g are monotone.

Our Main Contributions

Coverage/ Diversity

$$\text{SCSC: } \min\{f(X) : g(X) \geq c\}, \quad \text{SCSK: } \max\{g(X) : f(X) \leq b\},$$

Co-operative Costs

- Show how SCSC/SCSK subsume a number of important optimization problems.
- Provide a unifying algorithmic framework for these.
- Provide a complete characterization of the hardness of these problems.
- Emphasize the scalability and practicality of some of our algorithms!

I - Submodular Set Cover and Submodular Knapsack

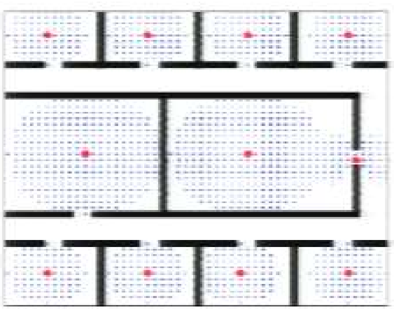
SSC: $\min\{w(X) : g(X) \geq c\}$, SK: $\max\{g(X) : w(X) \leq b\}$,

I - Submodular Set Cover and Submodular Knapsack

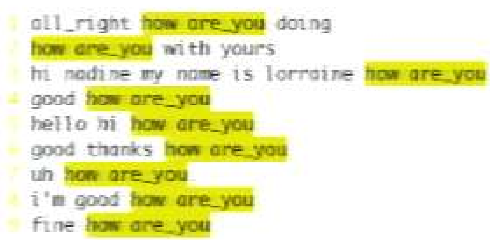
Coverage/ Diversity

$$\text{SSC: } \min\{w(X) : g(X) \geq c\}, \quad \text{SK: } \max\{g(X) : w(X) \leq b\},$$

Additive Costs



Sensor Placement (Krause et al'08)



Data Subset Selection (Wei et al'13)

Document Summarization (Lin-Bilmes'11)

II - Submodular Cost with Modular Constraints

$$\text{SML: } \min\{f(X) : w(X) \geq c\}, \quad \text{SS: } \max\{w(X) : f(X) \leq b\},$$

II - Submodular Cost with Modular Constraints

Additive functions

$$\text{SML: } \min\{f(X) : w(X) \geq c\}, \quad \text{SS: } \max\{w(X) : f(X) \leq b\},$$

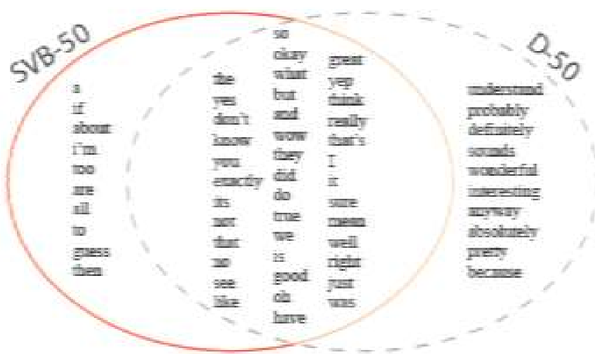
Co-operative Costs

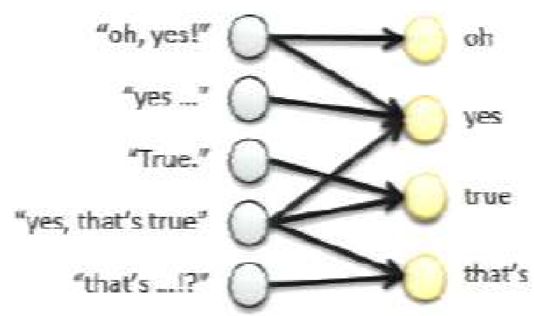
II - Submodular Cost with Modular Constraints

Additive functions

$$\text{SML: } \min\{f(X) : w(X) \geq c\}, \quad \text{SS: } \max\{w(X) : f(X) \leq b\},$$

Co-operative Costs





Limited vocabulary speech corpus selection (Lin-Bilmes'11)

III - Most General Case: SCSC and SCSK

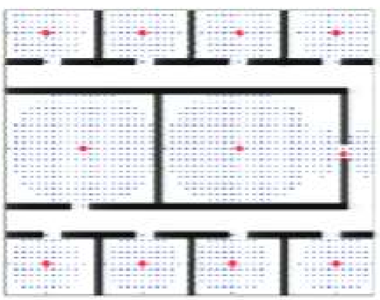
$$\text{SCSC: } \min\{f(X) : g(X) \geq c\}, \quad \text{SCSK: } \max\{g(X) : f(X) \leq b\},$$

III - Most General Case: SCSC and SCSK

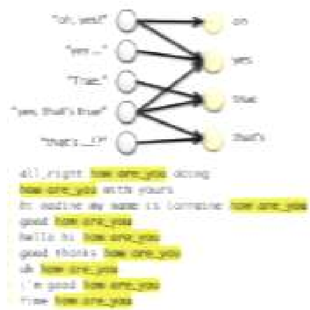
Coverage/ Diversity

SCSC: $\min\{f(X) : g(X) \geq c\}$, SCSK: $\max\{g(X) : f(X) \leq b\}$,

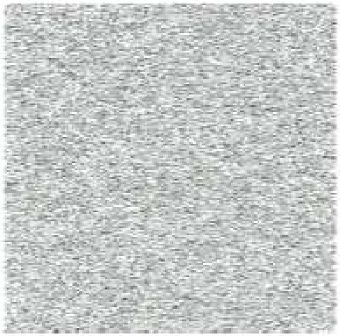
Co-operative Costs



Sensor Placement with Submodular Costs (I-Bilmes'12)



Limited vocabulary and accoustically diverse speech corpus selection (Lin-Bilmes'11, Wei et al'13)



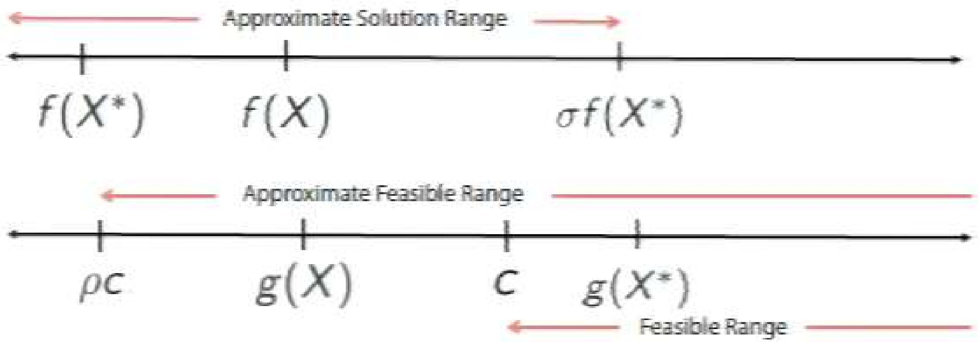
Privacy preserving communication (I-Bilmes'13)

Connections between SCSC and SCSK

- **Bi-criterion factors:**

Connections between SCSC and SCSK

- **Bi-criterion factors:** $[\sigma > 1, \rho < 1]$
- $\min\{f(X) : g(X) \geq c\}$: $[\sigma, \rho]$ approximation for SCSC is a set $X : f(X) \leq \sigma f(X^*)$ and $g(X) \geq \rho c$.

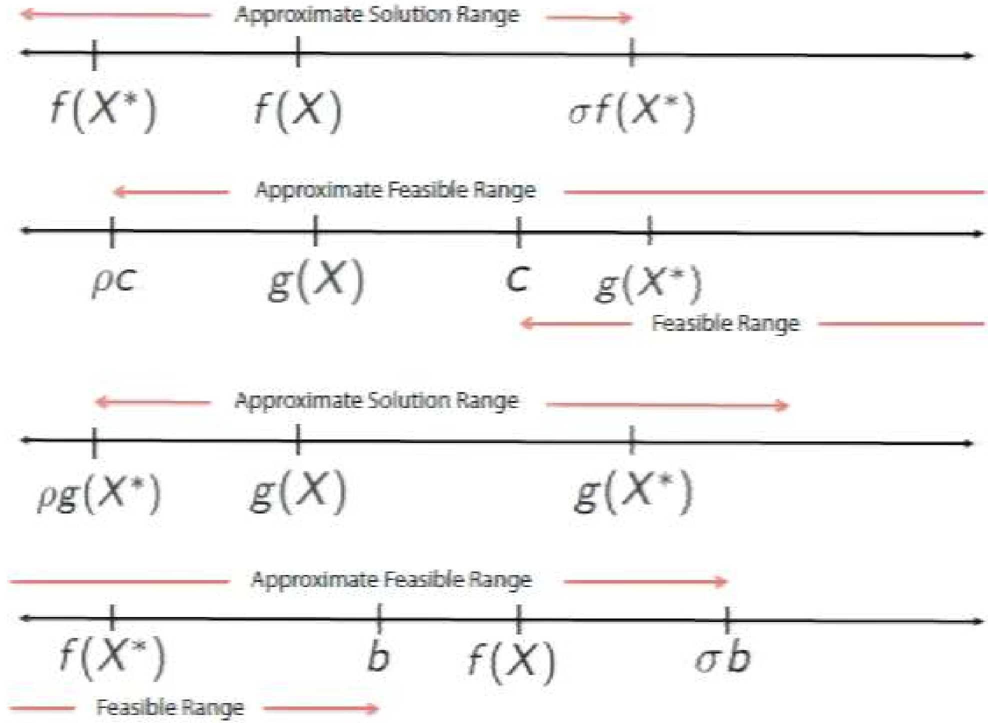


Connections between SCSC and SCSK

- **Bi-criterion factors:** $[\sigma > 1, \rho < 1]$

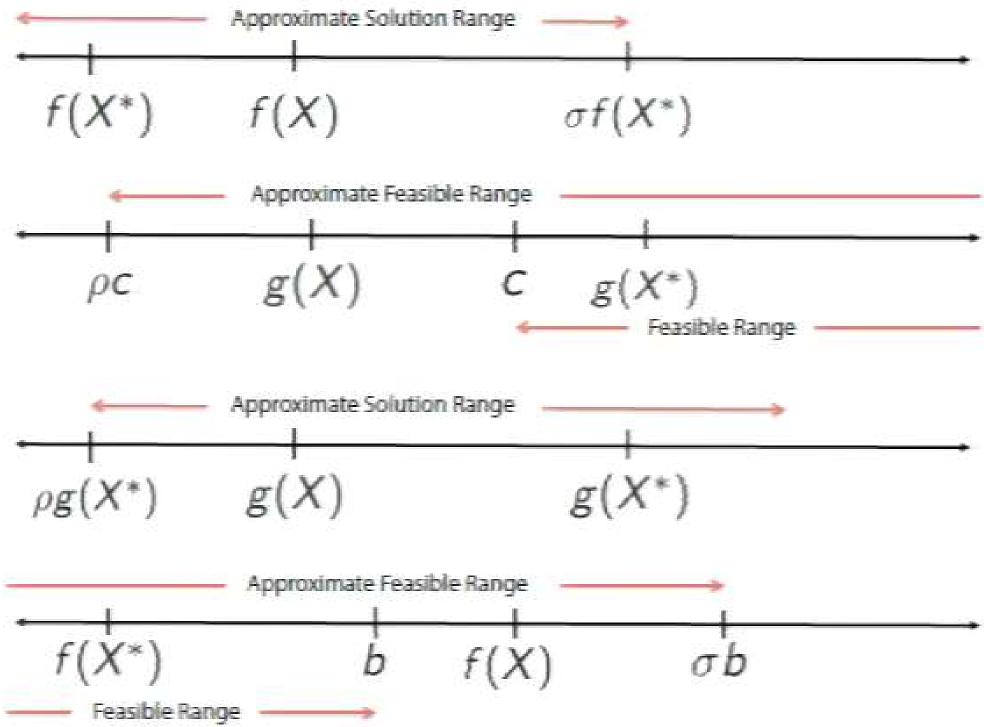
- $\min\{f(X) : g(X) \geq c\}$: $[\sigma, \rho]$ approximation for SCSC is a set $X : f(X) \leq \sigma f(X^*)$ and $g(X) \geq \rho c$.

- $\max\{g(X) : f(X) \leq b\}$: $[\rho, \sigma]$ approximation for SCSK is a set $X : g(X) \geq \rho g(X^*)$ and $f(X) \leq \sigma b$.



Connections between SCSC and SCSK

- **Bi-criterion factors:** $[\sigma > 1, \rho < 1]$
- $\min\{f(X) : g(X) \geq c\}$: $[\sigma, \rho]$ approximation for SCSC is a set $X : f(X) \leq \sigma f(X^*)$ and $g(X) \geq \rho c$.
- $\max\{g(X) : f(X) \leq b\}$: $[\rho, \sigma]$ approximation for SCSK is a set $X : g(X) \geq \rho g(X^*)$ and $f(X) \leq \sigma b$.

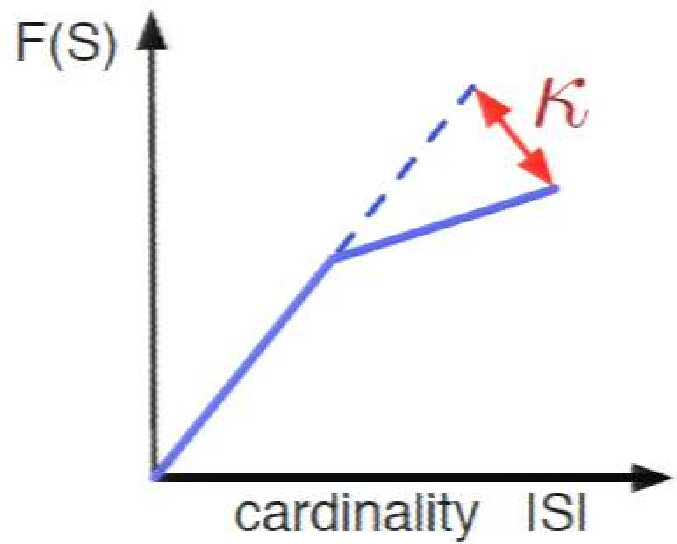


- **Theorem:** Given a $[\sigma, \rho]$ bi-criterion approx. algorithm for SCSC, we can obtain a $[(1+\epsilon)\rho, \sigma]$ bi-criterion approx. algorithm for SCSK, by running the algorithm for SCSC, $O(\log \frac{1}{\epsilon})$ times.
- The other direction also holds!

Curvature of a Submodular Function

- Curvature:

$$\kappa_f = 1 - \min_{j \in V} \frac{f(j|V \setminus j)}{f(j)} \quad \text{and} \quad \kappa_g = 1 - \min_{j \in V} \frac{g(j|V \setminus j)}{g(j)} \qquad (2)$$



- Curvature is a fundamental "complexity" parameter of a submodular function.

Hardness (Lower bounds) of the problems

	Modular g	Submodular g	
	($\kappa_g = 0$)	($0 < \kappa_g < 1$)	($\kappa_g = 1$)
Modular f ($\kappa_f = 0$)			
Submod f ($0 < \kappa_f < 1$)			
Submod f ($\kappa_f = 1$)			

Hardness (Lower bounds) of the problems

Knapsack

	Modular g	Submodular g	
	$(\kappa_g = 0)$	$(0 < \kappa_g < 1)$	$(\kappa_g = 1)$
Modular f $(\kappa_f = 0)$	FPTAS		
Submod f $(0 < \kappa_f < 1)$			
Submod f $(\kappa_f = 1)$			

Hardness (Lower bounds) of the problems

Knapsack SSC/SK

	Modular g	Submodular g	
	$(\kappa_g = 0)$	$(0 < \kappa_g < 1)$	$(\kappa_g = 1)$
Modular f $(\kappa_f = 0)$	FPTAS	$\frac{1}{\kappa_g}(1 - e^{-\kappa_g})$	$1 - 1/e$
Submod f $(0 < \kappa_f < 1)$			
Submod f $(\kappa_f = 1)$			

Hardness (Lower bounds) of the problems

Knapsack

SSC/SK

	Modular g	Submodular g	
	$(\kappa_g = 0)$	$(0 < \kappa_g < 1)$	$(\kappa_g = 1)$
Modular f $(\kappa_f = 0)$	FPTAS	$\frac{1}{\kappa_g}(1 - e^{-\kappa_g})$	$1 - 1/e$
Submod f $(0 < \kappa_f < 1)$	$\Omega(\frac{\sqrt{n}}{1+(\sqrt{n}-1)(1-\kappa_f)})$		
Submod f $(\kappa_f = 1)$	$\Omega(\sqrt{n})$		

SML/SS

Hardness (Lower bounds) of the problems

Knapsack

SSC/SK

	Modular g	Submodular g	
	$(\kappa_g = 0)$	$(0 < \kappa_g < 1)$	$(\kappa_g = 1)$
Modular f $(\kappa_f = 0)$	FPTAS	$\frac{1}{\kappa_g}(1 - e^{-\kappa_g})$	$1 - 1/e$
Submod f $(0 < \kappa_f < 1)$	$\Omega(\frac{\sqrt{n}}{1+(\sqrt{n}-1)(1-\kappa_f)})$	$\Omega(\frac{\sqrt{n}}{1+(\sqrt{n}-1)(1-\kappa_f)})$	$\Omega(\frac{\sqrt{n}}{1+(\sqrt{n}-1)(1-\kappa_f)})$
Submod f $(\kappa_f = 1)$	$\Omega(\sqrt{n})$	$\Omega(\sqrt{n})$	$\Omega(\sqrt{n})$

SML/SS

SCSC/SCSK

Algorithmic framework

Algorithm 1 General algorithmic framework to address both Problems 1 and 2

Algorithmic framework

Algorithm 1 General algorithmic framework to address both Problems 1 and 2

for $t = 1, 2, \cdots, T$ **do**
    Choose surrogate functions $\hat{f}_t$ and $\hat{g}_t$ for f and g respectively.

end for

Algorithmic framework

Algorithm 1 General algorithmic framework to address both Problems 1 and 2

1: **for** $t = 1, 2, \cdots, T$ **do**
2: Choose surrogate functions $\hat{f}_t$ and $\hat{g}_t$ for f and g respectively.
3: Obtain X^t as the optimizer of SCSC/SCSK with $\hat{f}_t$ and $\hat{g}_t$ instead of f and g.
4: **end for**

- Surrogate functions: modular upper/ lower bounds or Ellipsoidal Approximations.

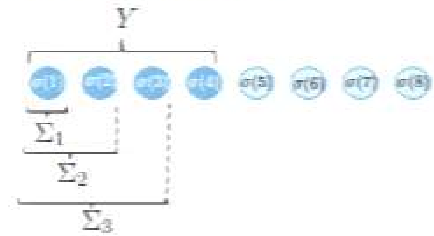
Surrogate functions

- **Modular Lower Bounds:** Induced via orderings of elements:

Surrogate functions

- **Modular Lower Bounds:** Induced via orderings of elements:

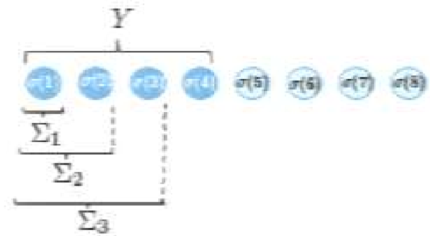
$f(X) \leq h_Y^\sigma(X)$, where $h_Y^\sigma(\sigma(i)) = f(\Sigma_i) - f(\Sigma_{i-1})$



Surrogate functions

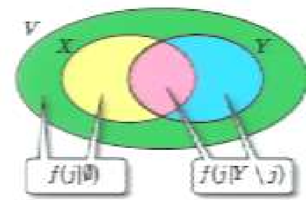
- **Modular Lower Bounds:** Induced via orderings of elements:

$$f(X) \leq h_Y^\sigma(X), \text{ where } h_Y^\sigma(\sigma(i)) = f(\Sigma_i) - f(\Sigma_{i-1})$$



- **Modular upper bounds:**
  Upper bound-I

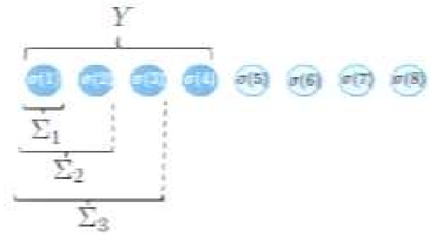
$$f(X) \leq m_{Y,1}(X) = f(Y) - \sum_{j \in Y \setminus X} f(j|Y \setminus j) + \sum_{j \in X \setminus Y} f(j|\emptyset)$$



Surrogate functions

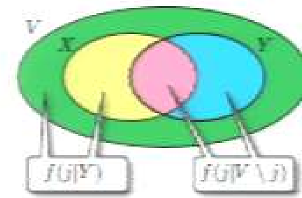
- **Modular Lower Bounds:** Induced via orderings of elements:

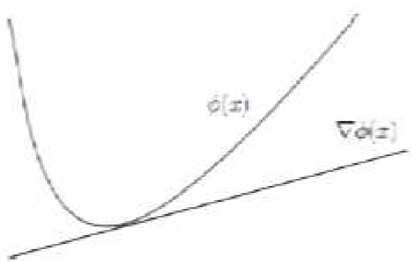
$$f(X) \leq h_Y^\sigma(X), \text{ where } h_Y^\sigma(\sigma(i)) = f(\Sigma_i) - f(\Sigma_{i-1})$$

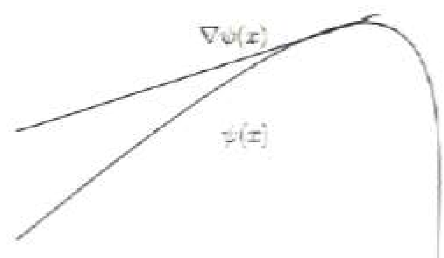


- **Modular upper bounds:**
  Upper bound-II

$$f(X) \leq m_{Y,2}(X) = f(Y) - \sum_{j \in Y \setminus X} f(j|V \setminus j) + \sum_{j \in X \setminus Y} f(j|Y)$$



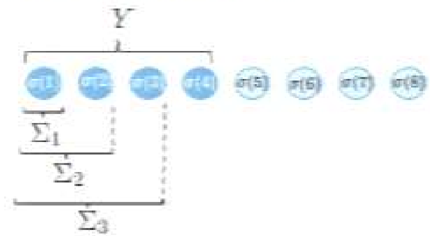




Surrogate functions

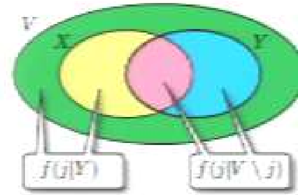
- **Modular Lower Bounds:** Induced via orderings of elements:

$$f(X) \leq h^{\sigma}_{Y}(X), \text{ where } h^{\sigma}_{Y}(\sigma(i)) = f(\Sigma_i) - f(\Sigma_{i-1})$$

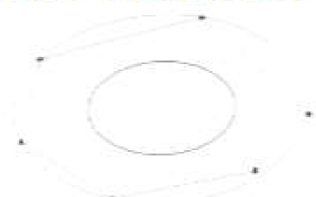


- **Modular upper bounds:**
  Upper bound-II

$$f(X) \leq m_{Y,2}(X) = f(Y) - \sum_{j \in Y \setminus X} f(j|V \setminus j) + \sum_{j \in X \setminus Y} f(j|Y)$$



- **Approximations:** Ellipsoidal Approximation gives the *tightest* approximation to a submodular function.



Submodular Set Cover (SSC) and Submodular Knapsack (SK)

Coverage/ Diversity

$$\text{SSC: } \min\{w(X) : g(X) \geq c\}, \quad \text{SK: } \max\{g(X) : w(X) \leq b\},$$

Additive Costs

Submodular Set Cover (SSC) and Submodular Knapsack (SK)

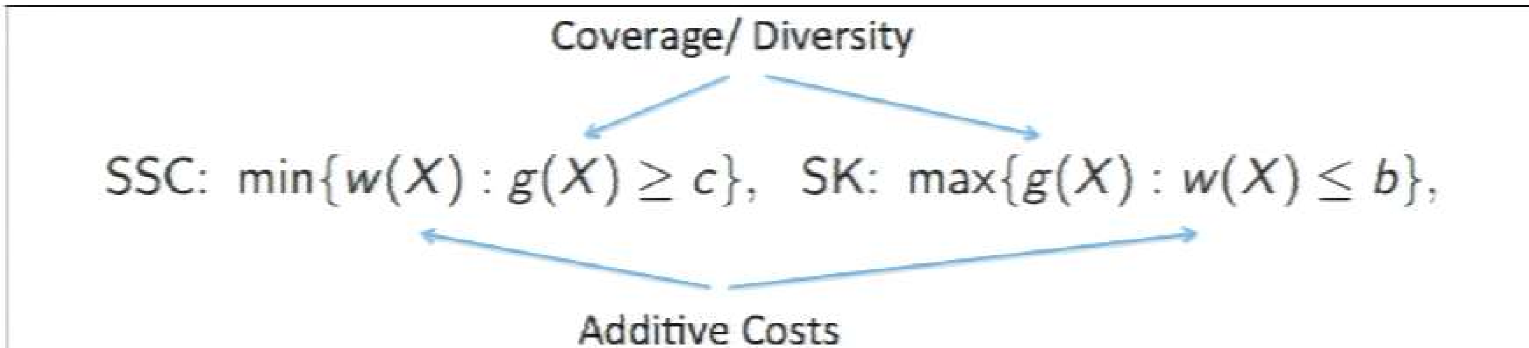
Coverage/ Diversity

$$\text{SSC: } \min\{w(X) : g(X) \geq c\}, \quad \text{SK: } \max\{g(X) : w(X) \leq b\},$$

Additive Costs

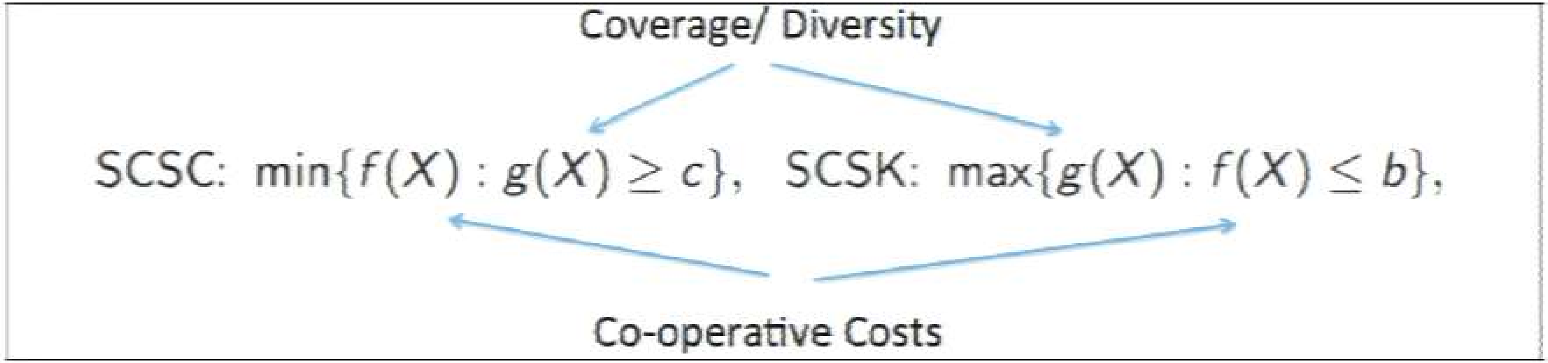
- Lemma: The greedy algorithm for SSC (Wolsey, 82) and SK (Nemhauser, 78) is special case of Algorithm 1 with g replaced by its modular lower bound.

Submodular Set Cover (SSC) and Submodular Knapsack (SK)



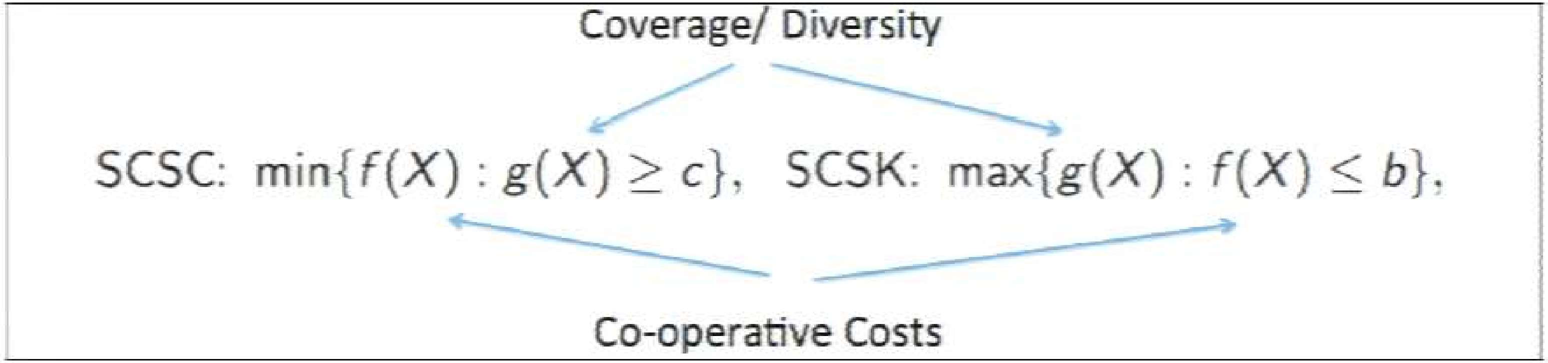
- Lemma: The greedy algorithm for SSC (Wolsey, 82) and SK (Nemhauser, 78) is special case of Algorithm 1 with g replaced by its modular lower bound.
- Approximation guarantees are constant factor – $1 - 1/e$ respectively.

Iterative Submodular Set Cover (ISSC)/Submodular Knapsack (ISK)



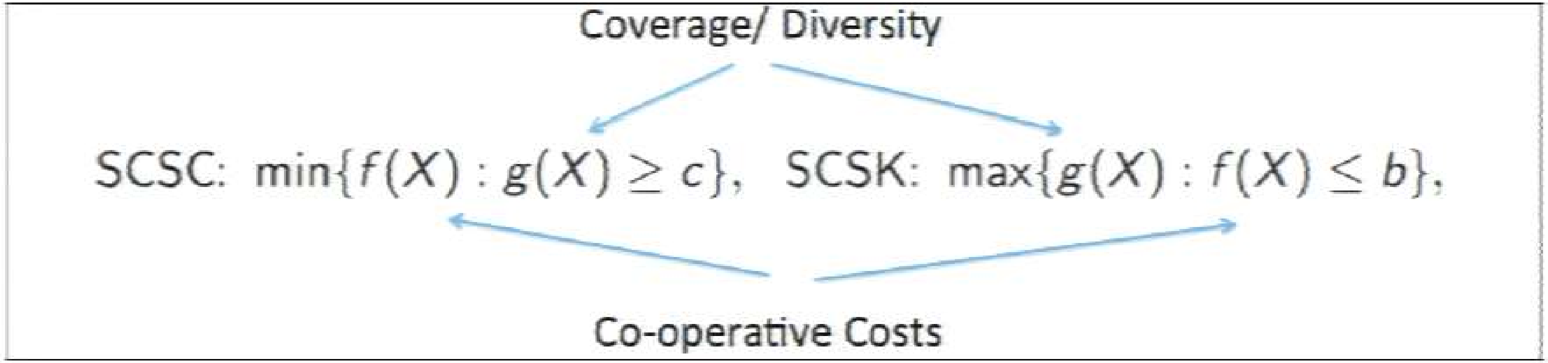
- Choose surrogate functions $\hat{f}_t$ as modular upper bounds.

Iterative Submodular Set Cover (ISSC)/Submodular Knapsack (ISK)



- Choose surrogate functions $\hat{f}_t$ as modular upper bounds.
- Fast iterative algorithms for SCSC and SCSK – Iteratively solve SSC or SK.

Iterative Submodular Set Cover (ISSC)/Submodular Knapsack (ISK)



- Choose surrogate functions $\hat{f}_t$ as modular upper bounds.
- Fast iterative algorithms for SCSC and SCSK – Iteratively solve SSC or SK.
- Theorem: ISSC and ISK obtain (bi-criterion) approximation factors $\frac{\sigma}{\rho} = O(\frac{n}{1+(n-1)(1-\kappa_f)})$.

Ellipsoidal Approximation Submodular Set Cover (EASSC)/ Submodular Knapsack (EASK)

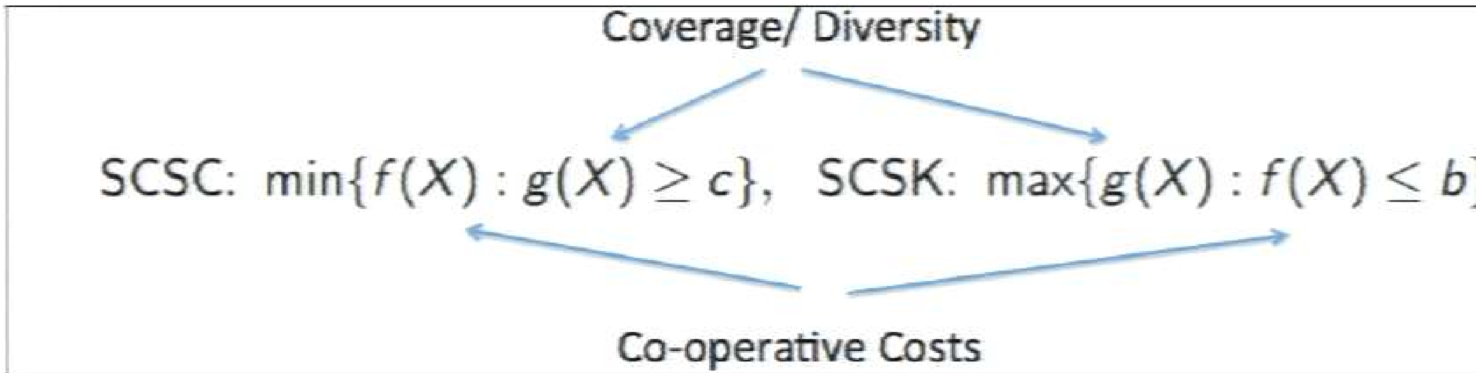
Coverage/ Diversity

SCSC: $\min\{f(X) : g(X) \geq c\}$, SCSK: $\max\{g(X) : f(X) \leq b$

Co-operative Costs

- Choose surrogate functions $\hat{f}_t$ as Ellipsoidal Approximation,
SCSC and SCSK.

Ellipsoidal Approximation Submodular Set Cover (EASSC)/ Submodular Knapsack (EASK)



- Choose surrogate functions $\hat{f}_t$ as Ellipsoidal Approximation, SCSC and SCSK.
- Theorem: EASSC and EASK obtain (bi-criterion) approxima factors of $\frac{\sigma}{\rho} = O(\frac{\sqrt{n}\log n}{1+(\sqrt{n}\log n - 1)(1-\kappa_f)})$.

Ellipsoidal Approximation Submodular Set Cover (EASSC)/ Submodular Knapsack (EASK)

Coverage/ Diversity

SCSC: $\min\{f(X) : g(X) \geq c\}$, SCSK: $\max\{g(X) : f(X) \leq b$

Co-operative Costs

- Choose surrogate functions $\hat{f}_t$ as Ellipsoidal Approximation, SCSC and SCSK.
- Theorem: EASSC and EASK obtain (bi-criterion) approxima factors of $\frac{\sigma}{\rho} = O(\frac{\sqrt{n}\log n}{1+(\sqrt{n}\log n-1)(1-\kappa_f)})$.
- These algorithms also extend to SML/SS.

Ellipsoidal Approximation Submodular Set Cover (EASSC)/ Submodular Knapsack (EASK)

Coverage/ Diversity

SCSC: $\min\{f(X) : g(X) \geq c\}$, SCSK: $\max\{g(X) : f(X) \leq b$

Co-operative Costs

- Choose surrogate functions $\hat{f}_t$ as Ellipsoidal Approximation, SCSC and SCSK.
- Theorem: EASSC and EASK obtain (bi-criterion) approxima factors of $\frac{\sigma}{\rho} = O(\frac{\sqrt{n}\log n}{1+(\sqrt{n}\log n - 1)(1-\kappa_f)})$.
- These algorithms also extend to SML/SS.
- This algorithm matches the hardness of this problem upto lo factors.

Limited Vocabulary data subset selection with Acco diversity

- **Accoustic Diversity:**

all_right how are_you do
how are_you with yours
hi nadine my name is lon
good how are_you
hello hi how are_you
good thanks how are_you
uh how are_you
i'm good how are_you
fine how are_you

Limited Vocabulary data subset selection with Acco diversity

- **Accoustic Diversity:**
  - Similarity matrix s_{ij} between utterances i and j (string kernel)

```
1 all_right how are_you do
2 how are_you with yours
3 hi nadine my name is lon
4 good how are_you
5 hello hi how are_you
6 good thanks how are_you
7 uh how are_you
8 i'm good how are_you
9 fine how are_you
```

Limited Vocabulary data subset selection with Acco diversity

- **Accoustic Diversity:**
  - Similarity matrix s_{ij} between utterances i and j (string kernel)
  - Submodular functions:
    1. Facility Location function: $g(X) = \sum_{i \in V} \max_{j \in X} s_{ij}$

```
all_right how are_you do
how are_you with yours
hi nadine my name is lon
good how are_you
hello hi how are_you
good thanks how are_you
uh how are_you
i'm good how are_you
fine how are_you
```

Limited Vocabulary data subset selection with Acco diversity

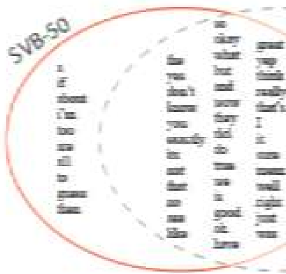
- **Accoustic Diversity:**
  - Similarity matrix s_{ij} between utterances i and j (string kernel)
  - Submodular functions:
    1. Facility Location function: $g(X) = \sum_{i \in V} \max_{j \in X} s_{ij}$
    2. Saturated coverage function $g(X) = \sum_{i \in V} \min\{\sum_{j \in X} s_{ij}, \beta \sum_{j \in V} s_{ij}\}$.

1 all_right how are_you dc
2 how are_you with yours
3 hi nadine my name is lo
4 good how are_you
5 hello hi how are_you
6 good thanks how are_you
7 uh how are_you
8 i'm good how are_you
9 fine how are_you

Limited Vocabulary data subset selection with Acco diversity

- **Accoustic Diversity:**
  - Similarity matrix s_{ij} between utterances i and j (string kernel)
  - Submodular functions:
    1. Facility Location function: $g(X) = \sum_{i \in V} \max_{j \in X} s_{ij}$
    2. Saturated coverage function $g(X) = \sum_{i \in V} \min\{\sum_{j \in X} s_{ij}, \beta \sum_{j \in V} s_{ij}\}$.
- **Limited Vocabulary:**

```
1 all_right how are_you do
2 how are_you with yours
3 hi nadine my name is lor
4 good how are_you
5 hello hi how are_you
6 good thanks how are_you
7 uh how are_you
8 i'm good how are_you
9 fine how are_you
```

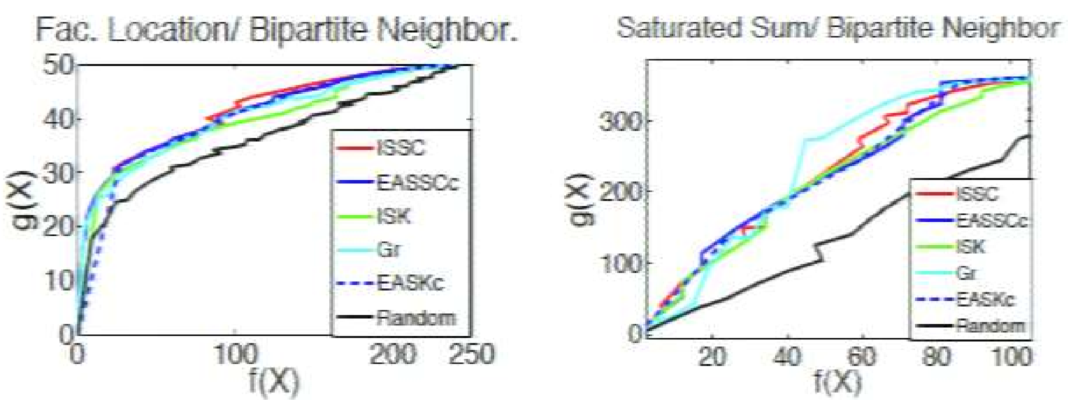


Results

- Compare our different algorithms on the TIMIT speech corp

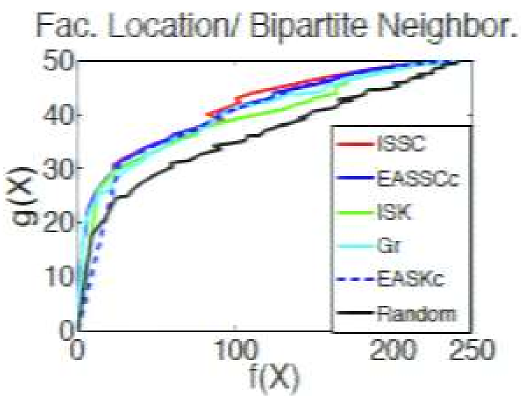
Results

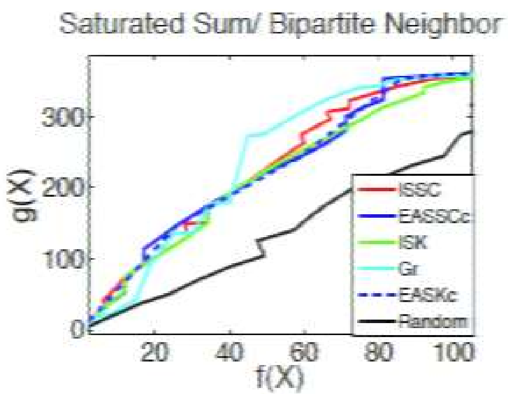
- Compare our different algorithms on the TIMIT speech corp
- Baseline is choosing random subsets.
- Observations:



Results

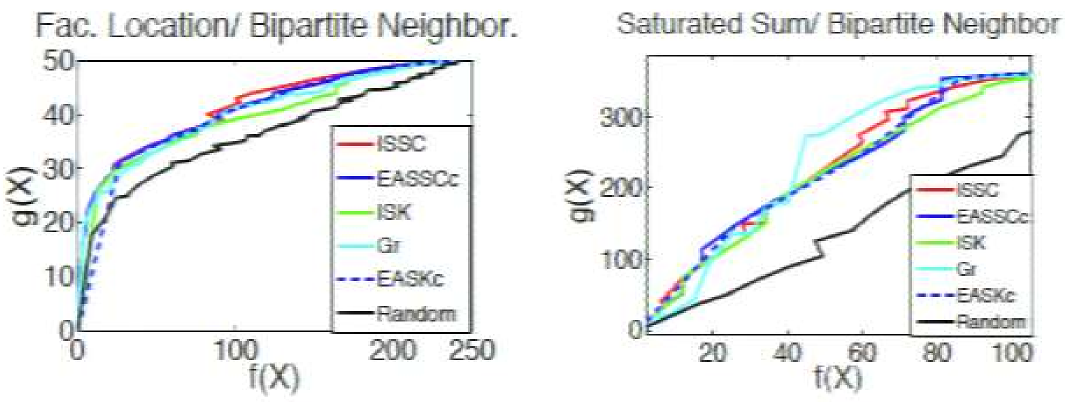
- Compare our different algorithms on the TIMIT speech corp
- Baseline is choosing random subsets.
- Observations:
  1. All the algorithms perform much better than random subset





Results

- Compare our different algorithms on the TIMIT speech corp
- Baseline is choosing random subsets.
- Observations:
  1. All the algorithms perform much better than random subset
  2. The iterative and much faster algorithms, perform comparab slower and tight Ellipsoidal Approximation based algorithms.



Conclusions/ Future Work

- We proposed some very efficient (scalable) algorithms and t
algorithms for submodular optimization under submodular constraints.
- In the paper: Extensions to handle multiple constraints, and non-monotone submodular functions.
- Future Work: Investigate our new algorithms on different re applications.

Thank You!