Optimization and Decision Theory under Uncertainty

Rex Ying*
zy26@cs.duke.edu

April 18, 2016

1 Stochastic Optimization using Greedy Algorithm

1.1 Sensor allocation problem

01/20/2016

Given n locations, and the joint distribution over $\{X_i\}_n$ Find subset of size $\leq k$ that maximizes $H(S)$. This is an NP-hard problem.

Properties of entropy:

1. $f(p) = p \log p$ is concave
2. $H(X) \geq 0$
3. monotonicity f is monotone if $f(S \cup \{t\}) \geq f(S) \forall S \subseteq U, t \notin S$. Entropy is monotone:

$$
\begin{aligned}
&H(S \cup \{t\}) - H(S) = \sum_{x_s, x_t} p(x_s, x_t) \log \frac{1}{p(x_s, x_t} - \sum_{x_s} p(x_s) \log \frac{1}{x_s} \\
=&\sum_{x_s} p(x_s) \sum_{x_t} p(x_t|x_s) \log \frac{1}{p(x_t|x_s)} \\
=&\sum_{x_s} p(x_s) H(x_t|x_s) \\
=&\mathbf{E}\left[H(x_t|x_s)\right] \\
=&H(X_t|S)
\end{aligned}
$$

4. Submodularity:

*Undergraduate student, Department of Computer Science, Duke University, Durham, NC 27708.

A function $f : S \to \mathbb{R}^+$ is submodular if for all $A \subseteq B \subseteq U$ and $t \notin B$,

$$f(A \cup \{t\}) - f(A) \geq f(B \cup \{t\}) - f(B)$$

This is similar to the concept of decreasing marginal benefit.

Lemma 1.1. *Entropy function is submodular.*

Proof. WTS: $H(X_t|W) \geq H(X_t|S)$.

Given $W \subset S$, $t \in S$, We want to show WLOG $W = \emptyset$, $H(X_t) \geq H(X_t|S)$. Conditioning does not increase entropy.

Let $g(x) = x \log x$

$$RHS = \mathbf{E}\left[H(X_t|X_s)\right]$$

$$LHS = \sum_{X_t} p(X_t) \log \frac{1}{p(X_t)}$$

where $p(X_t) = \mathbf{E}\left[p(X_t|X_S)\right]$

$$LHS = \sum_{X_t} g(p(X_t)) = \sum_{X_t} g(\mathbf{E}\left[p(X_t|X_S)\right] \geq \sum_{X_t} \mathbf{E}\left[g(p(X_t|X_S))\right] = RHS$$

Last step is by Jensen's inequality, since $g(x)$ is concave. □

Most but not all submodular functions are monotonic.

We use the greedy algorithm to find a set that approximates the optimal set for maximizing entropy. At each step, we find an additional location that increases the joint entropy the most.

Theorem 1.2. *Let S^* be the optimal solution of size k, and S_g be the solution gien by the greedy algorithm, then $f(S_g) \geq (1 - \frac{1}{e}) f(S^*)$. (1974)*

For generic case of monotone, submodular functions, greedy is the best one can do in poly time.

Proof. Let $G_0, G_1, \ldots, G_k$ be the sets found by greedy. Consider some stage i, $|G_i \cup S^*| \geq k$. The function is monotone: $f(G_i \cup S^*) \geq f(S^*)$.

Add all elements of S^* not in G_i, let them be $y_1, y_2, \ldots$. Let $z_j = f(G_i \cup \{y_1, y_2, \ldots, y_j\}) - f(G_i \cup \{y_1, y_2, \ldots, y_{j-1})$. Then $\sum_{j=1}^{k} z_j = f(S^* \cup G_i) - f(G_i) \geq f(S^*) - f(G_i)$. Therefore, $\exists y_r$, such that $z_r \geq (f(S^*) - f(G_i))/k$.

By submodularity, $(f(S^*) - f(G_i))/k \leq z_r \leq f(G_i \cup y_r) - f(G_i) \leq f(G_{i+1}) - f(G_i)$.

Unroll the recurrence relation:

$$\begin{aligned} f(G_{i+1}) &= f(G_i) + f(G_{i+1}) - f(G_i) \\ \geq & f(G_i)(1 - \frac{1}{k}) + \frac{1}{k} f(S^*) \geq \ldots \geq \left[1 - (1 - \frac{1}{k})^k\right] f(S^*) \geq (1 - \frac{1}{e}) f(S^*) \end{aligned}$$

□

1.2 0122

Sensor allocation problem continued

Assumption: given X_S, there is some oracle that gives you $p(X_j|X_S)$. (eg. provided by graphical models)

There are still exponential number of possibilities for the given X_S since each r.v. can take a discrete set of values. To compute expectation, we use the Monte Carlo algorithm.

Claim: For any ϵ, $\delta > 0$, let $N = \frac{1}{2}\left(\frac{\log|\chi_j|}{\epsilon}\right)\log\frac{2}{\delta}$. Then with probability $1 - \delta$, $|H - \hat{H}| \leq \epsilon$.

Proof. This is a result of the Hoeffding's Inequality:

Set $\delta = 2e^{2N\frac{\epsilon^2}{U^2}}$, and $U = \log|\chi_j|$. □

Theorem 1.3 (Hoeffding's Inequality)**.** *Let $Z_1, Z_2, \ldots, Z_n$ be iid r.v. in $[0, U]$ and let $S = \sum_{i=1}^N Z_i$, then* $\Pr\left[|\frac{S}{n} - \mathbf{E}[Z_1]| \geq \epsilon\right] \leq 2e^{-2n\frac{\epsilon^2}{U^2}}$.

At each step, greedy makes an error of ϵS^*.

$$f(S_g) = (1 - \frac{1}{\epsilon}) f(S^*) - k\epsilon$$

Probability that the greedy messed up: $nk\delta$

Proof of Hoeffding's Inequality using Markov's inequality: Given r.v. $Y \geq 0$, $\Pr[Y \geq a] \leq \frac{\mathbf{E}[Y]}{a}$.

$Y = e^{tS}$, $\Pr[S \geq a] = \Pr\left[e^{tS} \geq e^{ta}\right] \leq \frac{\mathbf{E}\left[e^{tS}\right]}{e^{ta}} = \frac{(\mathbf{E}\left[e^{tZ_1}\right])^N}{e^{ta}}$

The worst case is when Z taks the Bernoulli distribution.

Other submodular functions:

- cardinality (modular function)

- $f(S) = \max_{s \in S} s$
- $f(S) = \mathbf{E}[\max_{i \in S} X_i]$ (first show submodularity sample by sample, then use linearity of expectation).

1.3 Diffusion

Independent cascade model: For every pair of vertices u, v, there is an edge from u to v with weight p_{uv} indicating the probability of spread.

Given $\{p\}$ and budget k, find the set of nodes of size k to maximize the expected reach:

$$f(S) = \mathbf{E}[\#\text{ nodes reached if process starts at } S]$$

WTS this function is submodular.

Deterministic version: adding an additional element to larger set could potentially reach targets already covered. (Set cover)

Sets: $Q_1, Q_2, \ldots, Q_n$, $f(S) = |\bigcup_{i \in [1..n])Q_i}|$ is submodular.

0126 $f(S) = \mathbf{E}[\#\text{activevertices}|S] = \sum_\sigma \Pr[\sigma] f_\sigma$ where f_σ is the number of vertices reachable from S. This is subbmodular.

1.4 Stochastic set cover (query processing)

Given filters $F_1, F_2, \ldots, F_n$ which return true with probability p_i, and cost of evaluating the filter c_i. Assume evaluation of F_i is independent of evaluation of F_j for any i, j.

Filters are used in conjunctive queries $Q_1, Q_2, \ldots, Q_m$. Each Query $Q_j = F_{j1} \wedge F_{j2} \wedge \ldots \wedge F_{jk}$.

Objective: find ordering of evaluation to minimize expected cost. Evaluate filter with high failure probabillity first so with higher probability one can have early stop. We also want to evaluate filter with smaller cost first.

Greedy algorithm for one query: sort in increasing $\frac{c_i}{1-\pi}$.

Proof. Use exchange argument. Suppose F_i is followed by F_j. If we were to exchange them, all other terms will be thee same except $c_i + p_i c_j$, versus $c_j + p_j c_i$. □

The problem with multiple queries is NP-hard.

Proof. It can be reduced to set cover. With probability arbitrarily close to 1, all filters evaluate to false. Now each filter (almost surely) covers a set of elements. □

Use non-adaptive decision trees (degenerate linear decision tree): Fix an order f_{σ_i}. Evaluate filters in this order. Skip f_{σ_i} if there is no unanswered query containing it. Now the size of each decision tree is linear (search space is still exponential).

1.5 Lower bound

let μ be the max number of queries that any filter appears in a query (up to m); β be the max number of filters in any query (up to n).

Claim: Any fixed ordering is $\Omega(\mu)$ approximate of the optimal adaptive ordering.

Proof. Existence: Construct an instance where $m = n^2$, $\mu = n$, $\beta = 3$. For filters F_i, each $c_i = 0$, $p_i = \frac{1}{n}$. For filters H_i, each $c_i = 1$, $p_i \to 0$.

Queries: $Q_{ij} = F_i \wedge H_i \wedge H_j$. Group the queries into n groups: $G_i = \{Q_{i1}, Q_{i2}, \ldots, Q_{in}\}$ for $i = 1, 2, \ldots, n$.

OPT strategy: evaluate all F_i's since they have no cost. If F_i is evaluated to false, all queries in G_i are answered (probability $1 - \frac{1}{n}$). If G_i is left, we only need to evaluate H_i, which is common for all queries in G_i. It will evaluate to false with overwhelming probability and all queries in the group is done. The expected group left is 1 and the expected cost (just cost of H_i) is 1.

Fixed ordering. All F orders are put in front, and then some ordering of H filters.

$$\Pr[\text{some group } G_i \text{ is not done}] \geq 1 - e^{-1}$$

If it is not done, it is random which group is not done. In expectation, half of H filters are evaluated. The expected cost is $\Omega(n)$. □

It is also an upper bound. Hence if μ is small, fixed ordering is not too bad.

Claim: There exists a fixed ordering that is a $O(\mu)$ approximation.

Algorithm: order in increasing $\frac{c_i}{1-p_i}$.

Proof. Consider any query Q_i.

$$\mathbf{E}[\text{cost spent by OPT to answer } Q_j] \geq \mathbf{E}[\text{cost spent by greedy to answer } Q_j]$$

Cost of greedy $\leq \sum_{j=1}^{m} \mathbf{E}$ [cost spent by greedy to answer Q_j] $\leq \sum_{j=1}^{m} \mathbf{E}$ [cost spent by OPT to answer Q_j] $\leq \mu \sum_{i=1}^{n} \mathbf{E}$ [costofOPTtoevaluateF_i] $= \mu OPT$.

It is an overestimate since one filter can resolve more than 1 queries. Last step: change sum over queries to sum over filters. But a filter can resolve up to μ queries. □

To do better, one needs adaptive strategy.

1.6 0128

Greedy for set cover: each time pick a set with largest $c(S)$/#newelemcovered.

Proof: look at j-th element that the greedy alg covers using set S_i. Claim: $price(e_j) \leq \frac{OPT}{m-j+1}$

$OPT \geq \sum_{S \in F \ \Phi} c(S) 1_{S \in OPT} \ m - j + 1 \leq \sum_{S \in F \ \Phi} n_s^*$ Divide first by second.

Greedy $\leq \sum_{j=1}^{m} \frac{OPT}{m-j+1} \approx OPT \log m$

Stochastic: if query evaluates to true, it covers edges connected; if evaluates to false, it covers edges 2 hops away. The algo finishes when all edges are covered.

Can't seem to analyze coverage of elements.

Greedy: each step chose filter with highest price: $c(F)/\mathbf{E}$ [#edgescovered]. Analysis is more messy as this strategy is adaptive.

Condition on $F_1, \dots F_{i-1}$.

Claim1: $\mathbf{E}[Greedy] = \sum_e \mathbf{E}[pricee]$

$\mathbf{E}[\mathsf{OPT}|F_1, \dots, F_{i-1}] \geq c(F_i) 1_{F_i \in \mathsf{OPT}} + \sum_{F \notin \Phi \cup \{F_i\}} c(F) \Pr[F \in \mathsf{OPT}] \ m - j + 1 \leq |F_i| 1_{F_i \in \mathsf{OPT}} + \sum_{F \notin \Phi \cup \{F_i\}} n_F \Pr[F \in \mathsf{OPT}]$

Trouble when OPT evaluates F_i.

Define OC(F) as price(F) $\times$ actual # of edges covered by F in greedy

Account for OPT's cost as follows 1. if both evaluate filter, cost(F) = OC(F) 2. otherwise cost(F) = c(F) Event of OPT evaluating F_i is independent of outcome of F_i.

2 Linear Programming for Stochastic Optimization

2.1 Prophet Inequalities

Gamblers know the distributions $X_1, \ldots, X_n$, but does not know the order by which the prophet presents.

Prophet gets $\mathbf{E}[\max_i X_i]$ (optimal).

2.1.1 Threshold policy

Choose threshold w. Discard if below w.

Theorem 2.1. *Prophet Inequality [Krengel, Sucheston, and Garling '77] The threshold policy achieves the best possible: there always exists a threshold such that the expected value is half OPT.*

Proof. Account for reward (charging)

If $\max_i X_i > w$ then some box chosen gives gambler fixed reward w

If box j is encountered, "excess payoff" $= (X_j - w)^+$. If excess > 0, policy stops.

Let $x^* = \max_i X_i$.

$$\text{payoff} \geq \Pr[x^* \geq w]\, w + \sum_j \Pr[j \text{ encountered}]\, \mathbf{E}\left[(X_j - w)^+\right] \tag{2.1}$$

$$\geq \Pr[x^* \geq w]\, w + \Pr[x^* < w] \sum_{j=1}^{n} \mathbf{E}\left[(X_j - w)^+\right] \tag{2.2}$$

Choose $w = \sum_{j=1}^n \mathbf{E}[(X_j - w)^+]$. RHS decreases as LHS increases. The desired w can be found via binary search. With this w, payoff $\geq w$.

WTS: w is at least half of $\mathbf{E}[x^*]$.

$2w = w + \mathbf{E}\left[\sum_{j=1}^n \mathbf{E}[(X_j - w)^+]\right] \geq w + \mathbf{E}[(x^* - w)^+] \geq \mathbf{E}[x^*]$ □

2.1.2 Linear Program Relaxation in Succinct Form

Let $z_{jv} = \Pr[\text{prophet chooses j and } X_j = v]$.

Constraints are:

$$\sum_{j,v} z_j v = 1$$

$$\forall j, v, z_{j,v} \leq \Pr[X_j = v] = f_j(v)$$

Objective:

$$\max \mathbf{E}[\text{Reward of prophet}] = \sum_{j,v} v z_{jv}$$

The constraints are true for any decision policy of prophet. Here we consider behavior of prophet. Not well defined for gamblers since their policies are not fixed and they don't know the order of X_j.

An example: $X_a = 2$ w.p. $\frac{1}{2}$, and 0 otherwise. $X_b = 1$ w.p. $\frac{1}{2}$, and 0 otherwise. Variable $z_{a,2}$ can be thought of as $z_{a,2} = f_a(2)\Pr[X_a \text{ choosen} | X_a = 2]$

$$\max 2z_{a,2} + z_{b,1}$$

Subject to

$$z_{a,2} \leq \frac{1}{2}$$

$$z_{b,1} \leq \frac{1}{2}$$

$$z_{a,2} + z_{b,1} \leq 1$$

The linear program solves to $\max 2z_{a,2} + z_{b,1} = 0.5 \times 3 = 1.5$.

However, $\mathbf{E}[\max_j X_j] = \mathbf{E}[\max\{X_a, X_b\}] = \frac{5}{4}$.

In different senarios this LP may choose variable number of boxes, but in expectation it chooses 1. That is, the resulting policy may not be feasible for prophet.

In general, LP gives policy P_j for box j. Policy P_j: if $X_j = v$, choose j with probability $\frac{z_{j,v}}{f_j(v)}$.

This is a weakly coupled decision system, since there is only 1 constraint, namely $\sum_j \Pr[P_j \text{ chosen}] \leq 1$, that couples all the otherwise independent policies.

2.1.3 Conversion to a Feasible Policy

02/04/2016 We use the notations

$$R(P_i) = \mathbf{E}[\text{reward of } P_i]$$

$$C(P_i) = \mathbf{E}[P_i \text{ chooses box } i]$$

The LP finds one policy P_i for each box i, satisfying the constraint

$$\sum_{i=1}^{m} c(P_i) \leq 1$$

The objective

$$\max \sum_{i=1}^{n} R(P_i) = \text{LP-OPT}$$

This is a much weaker constraint. A feasible solution of the LP relaxation is not necessarily feasible for gambler. Reward of prophet $\mathsf{OPT} \leq \text{LP} - \text{OPT}$.

Let $p^*_{iv} = \Pr[\text{prophet chooses i } | X_i = v]$.

Since this LP has only 1 constraint, we can use the Lagrange multiplier. Define the dual variable (Lagrange multiplier) as w.

$$\text{LAG}(w) = \max \sum_{i=1}^{n} R(P_i) + w(1 - \sum_{i=1}^{n} C(P_i)) = w + \max \sum_{i=1}^{n} (R(P_i) - wC(P_i)) \tag{2.3}$$

Take any feasible solution, second term is non-negative, LAG is at least the first term. This implies the weak duality: $\text{LAG}(w) \geq \text{LP} - \text{OPT}$ This implies the weak duality: $\text{LAG}(w) \geq \text{LP} - \text{OPT}$

We can solve for each box separately. For each box i, if chosen, get reward, and pay the penalty w. This implies the threshold policy. Hence $\text{LAG}(w)$ outputs for each box i, the threshold policy for that box.

In fact $\text{LAG}(w) = w + \sum_{i=1}^{n} \mathbf{E}[(X_i - w)^+] \geq \text{LP} - \text{OPT}$ (won't need this).

$\sum_{i=1}^{n} \mathbf{E}[(X_i - w)^+]$ is the sum of values in all boxes if w (the penalty) is 0, i.e. every box is opened. It reaches 0 when w is very big. One can find w^* such that the sum is 1 (open 1 box in expectation).

At w^*, $\text{LAG}(w^*) = \sum_{i=1}^{n} R(P^*_i) \geq \text{LP} - \text{OPT}$.

How to make use of the solution?

If box i encountered, w.p. $\frac{1}{2}$ skip it; w.p. $\frac{1}{2}$ execute P^*_i (threshold policy for w^* in this case).

1. If i encountered, $\mathbf{E}[\text{Reward from } i | i \text{ reached}] = \frac{1}{2} R(P^*_i)$.
2. Let $X_i = \#$ boxes chosen before i.

By union bound,

$$\mathbf{E}[X_i] \leq \sum_{j=1}^{n} \Pr[j \text{ chosen }] \, \text{LAG}(w^*) = \sum_{j=1}^{n} R(P^*_i) + w^*(1 - \sum C(P^*_i)) \leq \sum_{j=1}^{n} \frac{1}{2} C(P^*_i) \leq \frac{1}{2}$$

By Markov's inequality,

$$\Pr[X_i \geq 1] \leq \frac{\mathbf{E}[X_i]}{1} \leq \frac{1}{2}$$

$$\Pr[i \text{ encountered }] \geq 1 - \frac{1}{2}$$

$\mathbf{E}[\text{Reward from } i] = \Pr[i \text{ encountered}]\,\mathbf{E}[\text{Reward from } i | i \text{ encountered}] \geq \frac{1}{2}\frac{1}{2}R(P_i^*) = \frac{1}{4}R(P_i^*)$

gambler's reward $\geq \frac{1}{4}$ LP-OPT.

This analysis doesn't assume threshold policy.

Proof. Alternative Proof of Optimality with Better Constant

For a policy $P_i(w)$, let

$$LAG(w) = w + \sum_{i=1}^{n}(R_i(w) - wC_i(w)) = w + \sum \phi_i(w)$$

Then $R_i(w) = \phi_i(w) + wC_i(w)$

$$\mathbf{E}[\text{ Reward of } P_i(w)] = \phi_i(w) + w\Pr[P_i(w) \text{ chooses } i]$$

Change the way that the reward is accounted for. $\phi_i(w)$ is given upfront. In addition, give w if the box is chosen. In expectation one gets the same reward as the original game.

Choose $w = \sum_{i=1}^{n} \phi_i(w) \geq \frac{1}{2}\text{LP} - \text{OPT}$.

Final policy: if box i encountered, run P_i.

Analysis: One of the 2 cases has to happen

1. Case 1: Some box chosen, and the reward is $w \geq \frac{1}{2}\text{LP} - \text{OPT}$.
2. Case 2: All boxes encountered (no box chosen). The reward is $\sum \phi_i \geq \frac{1}{2}\text{LP} - \text{OPT}$.

□

2.2 Stochastic Matching

02/09/2016

Given a graph $G(V, E)$. Each edge e has probability p_e of match (independent), with reward r_e if matched. Vertex v has "patience" t_v for trials. If an edge trial fails, there is no need to re-try, as the outcome is deterministic. The goal is to find a maximal matching.

Let $M(e)$ be the event that edge e is tried at some point. Let $y_e = \Pr[M(e) = 1]$.
Then by definition $x_e = \Pr[\text{edge } e \text{ is matched}] = p_e y_e$.

Constraints:

1. any vertex v participates in at most t_v trials.

$$\forall v, \sum_{e \in \delta(v)} y_e \leq t_v$$

2. any vertex v is matched at most once. Hence it must be true in expectation.

$$\forall v, \sum_{e \in \delta(v)} x_e \leq 1$$

3. $y_e \leq 1$. All variables are non-negative.

Objective:

$$\max \sum_e r_e x_e$$

Since the LP is a relaxation of the problem, the resulting LP-OPT is at least OPT (optimal strategy for this problem).

It generalizes the maximal matching in bipartite graph.

The algorithm:

1. Solve the LP relaxation
2. Policy: consider edges in random order. On reaching edge e, if e can be tried, i.e. both of its vertices are not yet matched, try e with probability $\frac{y_e}{2\alpha}$.

Random order gives smaller α value.

Open Question: is there a non-LP approach to solving this problem?

Conditioned on reaching e and e can be tried, $\mathbf{E}[\text{reward from } e] = \frac{x_e r_e}{2\alpha}$.

Suppose $e = (u, v)$. There are 4 conditions that prevent e from being tried.

$$\Pr[\#\text{ of tries at } u = t_u]$$

$$\Pr[\#\text{ of tries at } v = t_v]$$

$$\Pr[\#\text{ matches at } u \geq 1]$$

$$\Pr\left[\#\text{ matches at } v \geq 1\right]$$

Bound the probability of the first condition. Let $U = \#$ tries at u before reaching e.

$$\mathbf{E}\left[U\right] = \sum_{e \in \delta(u)} \Pr\left[e' \text{ is before } e \text{ in the random order and } e' \text{ has been tried}\right] \leq \sum_{e \in \delta(u)} \frac{1}{2} \frac{y_{e'}}{2\alpha} \leq \frac{t_u}{4\alpha}$$

The last step is by the second constraint of LP and the random ordering.

By Markov's Inequality,

$$\Pr\left[U \geq t_u\right] \leq \frac{1}{4\alpha}$$

Similarly one can show that all 4 conditions have probability $\frac{1}{4\alpha}$. By union bound,

$$\Pr\left[e \text{ is eligible for trial}\right] \geq 1 - \frac{1}{\alpha}$$

Therefore

$$\mathbf{E}\left[\text{reward from } e\right] \geq \sum_{e} (1 - \frac{1}{\alpha}) \frac{x_e r_e}{2\alpha} = OPT(1 - \frac{1}{\alpha}) \frac{1}{2\alpha} \geq \frac{OPT}{8}$$

The optimal bound is $\frac{OPT}{8}$ achieved when $\alpha = 2$.

This is a *local bound*: every edge has a set of 4 conditions. In the analysis we use Markov's inequality locally.

2.3 Return to Stochastic Set Cover

Consider the special case: every Q_j is the AND of 2 filters. In this case, we represent filters as vertices, and queries as edges.

Any decision policy must evaluate ≥ 1 filter for every edge on a decision paths. Hence $OPT \geq$ cost of the cheapest vertex cover. Min cost vertex cover has 2-approx.

The ILP formulation of vertex cover:

$$\min \sum_{v} c_v x_v$$

Subject to:

$$\forall e = (u, v), \quad x_u + x_v \geq 1$$

$$\forall v, \quad x_v \in \{0, 1\}$$

LP-relaxation: $x_v \in [0, 1]$.

2.3.1 SSC Approximation by Solving LP

Algo 1: solve LP. At least one of $u, v \geq \frac{1}{2}$ for every edge by constraint. Round x_v.

Claim: Suppose LP-OPT is the optimal solution to the LP relaxation, and ALG is the solution given by the algorithm, then

$$\text{LP-OPT} \leq \mathsf{OPT} \leq 2\text{LP-OPT} = \text{ALG}$$

Algorithm for $\beta = 2$ stochastic set cover:

1. Find 2-approx to VC
2. Evaluate these filters, with total cost $\leq 2\mathsf{OPT}$
3. for all queries Q_j not yet resolved, resolve Q_j. This costs $\leq$ OPT, because conditioned on a filter in the vertex cover being true, the other has to be evaluated by any valid decision policy (argue sample by sample).

Therefore the algorithm gives 3-approximation.

2.3.2 VC Approximation without solving LP

02/11/2016

Solve 2-approx vertex cover by writing its dual and use greedy algorithm.

The dual of the VC LP relaxation:

$$\max \sum_e \beta_e$$

Subject to

$$\sum_{e \sim v} \beta_e \leq c_v$$

Increase β_e until get stuck. Output vertices whose constraint is tight.

This can be done in linear time.

$$Cost = \sum_{v \text{ tight}} c_v = \sum_{v \text{ tight}} (\sum_{v \in \partial(e)} \beta_e) \leq 2 \left(\sum_e \beta_e \right) \leq 2 \text{ PRIMAL-LP-OPT}$$

2.3.3 Better SSC Approximation

Let $x_v = \Pr[\text{ policy evaluates filter } v]$; The objective function is:

$$C = \min \sum_v c_v x_v$$

Subject to:

$$\forall Q = (u, v), \quad x_u + x_v \geq \Pr[u \text{ evaluated or } v \text{ evaluated}] = 1$$

Define $g(v) = \Pr[\text{any neighbors of } v \text{ is true}] = 1 - \prod_{u \sim v}(1 - p_u)$. It satisfies $\forall v$, $x_v \geq g(v)$.

1. Solve the LP.
2. If $x_v \geq \frac{1}{2}$, evaluate v.
3. For all queries not resolved in (2), resolve them.

Let A be the vertex cover found in (2), B be the remaining independent set. $\Pr[\text{algo evaluates } v \in B] = g(v)$ by definition.

$$\mathbf{E}[\text{ALG}] = \sum_{v \in A} c_v + \sum_{v \in B} c_v g(v)$$

$$\mathbf{E}[\text{LP-OPT}] \geq \sum_{v \notin B} c_v x_v + \sum_{v \in B} c_v g(v)$$

3 Markov Decision Processes

We are given a set of states $\{S\}$, actions $\{a\}$. Rewards at each step depends on (S, a). We are also given $p_a(S, S')$, which is the probability that under the action a the state transitions from S to S'.

Let time constraint T be a random variable drawn from an exponential distribution (memoryless).

Discount factor $\gamma = 1 - \frac{1}{T}$. R is the reward at $t = 0$. Time Reward $= R + \gamma R + \gamma^2 R + \ldots$

We can write equations that imply dynamic programming.

Let $v(S) = \mathbf{E}[\text{ total reward } | \text{ start state } = S]$.

A policy specifies that if you are in state S', what action to take. Time no longer matters because of the memorylessness property.

$$v(S) = \max_a R_{S,a} + \gamma \sum_{S'} p_a(S, S')v(S')$$

However, there could be loops. States do not form a DAG.

3.1 Value Iteration

Start with arbitrary value of v's. Define an operator

$$Tv(S) = \max_a R_{S,a} + \gamma \sum_{S'} p_a(S, S')v(S')$$

Iterator over $v \leftarrow Tv$.

Claim: Value iteration converges.

Proof. Let $d(v, v') = \max_S |v(S) - v'(S)|$. WTS: $d(Tv, Tv') \leq \gamma d(v, v')$.

Fix an S, suppose a^* is the optimal action for state S.

$$Tv(S) = R_{S,a^*} + \gamma \sum_{S'} p_{a^*}(S, S')v(S')$$

$$Tv'(S) \geq R_{S,a^*} + \gamma \sum_{S'} p_{a^*}(S, S')v'(S')$$

$$|Tv(S) - Tv'(S)| \leq \gamma \sum_{S'} p_{a^*}(S, S')|v(s) - v'(S')| \leq \gamma \max_{S'} |v(S') - v(S)| = \gamma d(v, v')$$

□

3.2 Multi-armed Bandits

02/16/2016

Famous problem in stochastic control.

Given n separate Markov processes (arms), each having state space S_i. If arm is in state $s_i \in S_i$, there are 2 available actions: play and not play. If arm i is played at state s_i, the reward is $R_i(A_i)$(random variable). If arm i in state $s_i \in S_i$ is played, it moves its state to s_i' with probability $p_i(s_i, s_i')$.

Constraint: can only play 1 arm at each time step.

There exists a greedy algorithm that gives OPT in linear time.

We pretend that after playing, the arm stays at s_i for random $T_i(s_i)$ time before transiting.

Let

$$r_i(s_i) = \frac{\mathbf{E}\left[R_i(s_i)\right]}{\mathbf{E}\left[\sum_{t=0}^{T_i(s_i)} \gamma^t\right]} \Rightarrow \mathbf{E}\left[R_i(s_i)\right] = \mathbf{E}\left[\sum_{t=0}^{T_i(s_i)} r_i(s_i)\gamma^t\right]$$

Here we r eplace expectation of product with product of expectation, by independence.

Let $(i^*, s^*) = \underset{i, s_i}{\operatorname{argmax}}\{r_i(s_i)\}$

Claim: If arm i^* is in state s^*, then the optimal policy plays i^*.

Proof. Use exchange argument. Suppose the optimum policy does not play i^* in the next step. Modify the policy f as follows, play i^*, then execute the optimum policy. If opt plays i^*, skip this play. The new policy gives better reward since the action on i^* is not as heavily discounted.

Let τ be time at which old policy plays i^*. The difference between the 2 policies, in expectation, looks like:

- Old
$$\mathbf{E}\left[\sum_{t=0}^{\tau} \bar{r}(t)\gamma^t + \gamma^{\tau+1}\sum r(s^*)\gamma^t + \dots\right]$$
- New:
$$\mathbf{E}\left[\sum_{t=0}^{T(s^*)} r(s^*)\gamma^t + \gamma^{T(s^*)+1}\sum_{t=0}^{\tau} \bar{r}(t)\gamma^t + \dots\right]$$

We can see that the new policy gives better reward. □

When i^* is in s, play until the state is no longer s or s^*. Make R, T the time to reach this state and reward.

$\tilde{T}$: expectation of time starting from s till we hit a state that is neither s nor s^*.

$\tilde{S}$: discounted reward per step till state s^*.

Calculate by dynamic programming

Now we eliminate s^*, i.e. $\forall x \to s^* \to y$ with probability p, we have $x \to y$ with probability p in the new arm. Iteratively eliminate 1 state each time.

Computing policy: compute ordering of all (i, s), $\forall i, s$ inductively.

Executing policy: play arms with highest (i, s) pair in the ordering.

3.2.1 Gittins Index Policy

02/18/2016

Policy: for each arm i and $s \in S_i$, compute $\alpha_i(s)$, the Gittins index. At any time step, play the arm with largest Gittins index.

α_i is the maximum that λ can take so that $V_i(s, \lambda)$ decides to play at state s. This can be done by binary search on λ. $\alpha_i(s) = \frac{\mathbf{E}\left[\tilde{R}(s)\right]}{\mathbf{E}\left[\sum_{t=0}^{T(s)} \gamma^t\right]}$

Let i^*, s^* be state with largest Gittins Index. Then $\alpha_{i^*}(s^*) = \mathbf{E}\left[R(s^*)\right]$. From the exchange argument last time, this is the correct choice. For the state with the largest reward, the Gittins Index is the reward.

The removal of (i^*, s^*) during induction does not change the Gittins Index. (s^* has larger Gittins Index so the policy will keep playing).

4 Online Prediction

02/18/2016

4.1 Experts Problem

There are T time periods. At each time step t, expert i incurs loss l_i^t. At step t, algorithm picks onoe expert i_t and incurs its loss. After an expert is picked, all losses (from all experts) are revealed.

We want to minimize $loss(\text{ALG}) = \sum_{t=1}^{T} l_{i_t}^t$

This can be viewed as the adversarial multi-arm bandits problem.

$\mathbf{E}\left[\mathsf{OPT}\right] = \min_i T\mathbf{E}\left[D_i\right]$. The OPT knows all distributions, but is restricted to picking 1 expert. Our policy can pick different experts.

Let $i^* = \min_i \sum_{t=1}^T l_i^t$; REGRET $= \sum_{t=1}^T l_{i_t}^t - \sum_{t=1}^T l_{i^*}^t$.

4.1.1 Follow the leader

At step t_i, choose $i_t = \underset{i}{\operatorname{argmin}} \sum_{t'=1}^{i-1} l_i^{t'}$. This deterministic algorithm is bad, because of ties.

Let S_t be the set f min experts at time t. The adversary can ensure that we only pick the best expert among the f experts after f tries.

No deterministic algorithm can do better than n times OPT. This algorithm is "the best" deterministic algorithm.

FTL has loss 1 but l_{min} did not change. The size of S_t decreases by 1.

This can happen for at most n times.

After n steps where FTL makes loss 1, l_{min} must increase. $l_{FTL}^T = n l_{min}^T + (n-1)$

4.1.2 Randomized version RFTL

Pick a random expert among S_t each step.

$|S_{t-1}| = m$; $|S_t| = m - k$, $\mathbf{E}[RFTL] = \frac{k}{m}$ Loss of RFTL in worst case: $\sum_{i=1}^n \frac{1}{i} = \log n$

$l_{RFTL}^T \leq \log n l_{min}^t + \log n$

Goal: $l_{alg} - l_{min} = O(\sqrt{T})$

4.1.3 Randomized weighted majority

02/23/2016

The description of the randomized weighted majority algorithm for the expert problem, RWM(E), is as follows.

Initialization: we assign $w_i' = 1$ to for all expert $i \in [1:n]$. The decision maker picks each expert with probability $p_i' = \frac{1}{n}$.

At any time t, if the loss at $(t-1)$-th step, $l_i^{t-1} = 1$, then $w_i^t := w_i^{t-1}(1-\epsilon)$, else $w_i^t := w_i^{t-1}$. Set $p^t = \frac{w_i^t}{\sum_{j=1}^n w_j^t}$. The decision maker then chooses each expert i with probability p_i^t.

Theorem 4.1.

$$L_{\text{RWM}}^T \leq (1+\epsilon) L_{min}^T + \frac{\log n}{\epsilon}$$

where L_{min}^T is the minimum loss made by choosing a single optimal expert.

Using this theorem, REGRET $= \epsilon L_{min}^T + \frac{\log n}{\epsilon} \leq \epsilon T + \frac{\log n}{\epsilon}$. Set $\epsilon = \sqrt{\frac{\log n}{T}}$ to minimize the regret.

Regret per time step is $O(T^{-\frac{1}{2}})$, and the ratio of optimal solution to the solution given by RWM goes to 0 as $T \to \infty$.

Proof. Define a potential function for total weight.

Let $w^t = \sum_{j=1}^n w_j^t$. Each weight w_j^t decreases over time but cannot be too small.

Define the expected loss of RWM:

$$F^t = \frac{\sum_{i=1}^n w_i^t l_i^t}{w^t} = \frac{\sum_{i:l_i^t=1} w_i^t}{w^t} \tag{4.1}$$

For each i such that $l_i^{t+1} = 1$, $w_i^{t+1} = w_i^t - \epsilon w_i^t$.

$$w^{t+1} = w^t - \epsilon \sum_{i:l_i^t=1} w_i^t = w^t - \epsilon F^t w^t = w^t(1 - \epsilon F^t) \tag{4.2}$$

Sum over all steps:

$$w^{T+1} = w^0 \prod_{t=1}^T (1 - \epsilon F^t) = n \prod_{t=1}^T (1 - \epsilon F^t) \tag{4.3}$$

But $w^{T+1} \geq w_i^{T+1} \forall i$, in particular

$$w^{T+1} \geq (1 - \epsilon)^{L_{min}} \tag{4.4}$$

For F^t close to 0, $\ln(1 - \epsilon F^t) \leq -\epsilon F^t$. Take natural log on both sides of 4.4 and substitute 4.3,

$$L_{min} \ln(1 - \epsilon) \leq \ln N - \sum_{t=1}^T \epsilon F^t = \ln N - \epsilon L_{RWM} \tag{4.5}$$

Where L_{RWM} is the total loss of the RWM algorithm. We use another approximation: $\ln(1 - \epsilon) \geq -\epsilon - \epsilon^2$.

$$L_{\text{RWM}}^T \leq \frac{1}{\epsilon}(-L_{min} \ln(1 - \epsilon) + \ln N) \leq (1 + \epsilon) L_{min} + \frac{\ln N}{\epsilon} \tag{4.6}$$

□

4.1.4 Generalization of the Expert Problem

The space of decision vectors is:

$$P = \left\{ \mathbf{x} | \mathbf{x} \geq \mathbf{0}, \sum_{i=1}^{N} \mathbf{x}_i = 1 \right\}$$

Choose a point in P (in the case of the expert problem, it is the probability distribution to choose expert at time t).

More generalization: P is a convex set. At time t, an algorithm chooses $\mathbf{x}_t \in P$, and then is presented with loss vector $\mathbf{l}_t$. The loss is $\mathbf{l}_t \cdot \mathbf{x}_t$.

Define

$$\text{REGRET} = \sum_{t=1}^{T} \mathbf{l}_t \cdot \mathbf{x}_t - \min_{\mathbf{x} \in P} \mathbf{l}_t \cdot \mathbf{x} \tag{4.7}$$

This can be applied to learning distributions. $\mathbf{l_t}$ can be seen as the feedback of an attempt of classification.

Follow the leader algorithm (FTL) does the following: at each step t,

$$\mathbf{x}_t = \operatorname*{argmin}_{\mathbf{x} \in P} \sum_{t'=1}^{t-1} \mathbf{l}_{t'} \mathbf{x}$$

The algorithm is not stable. It could keep changing its decision based on the previous outcome. We want a way to improve the FTL algorithm such that it does not change decisions too often and obtain a lower regret.

The solution is to use regularization.

$$\mathbf{x}_t = \operatorname*{argmin}_{\mathbf{x} \in P} \sum_{t'=1}^{t-1} \mathbf{l}_{t'} \mathbf{x} - \frac{1}{\epsilon} H(\mathbf{x})$$

$\frac{1}{\epsilon}$ is the weight for the regularizer. H is the entropy function. This implies the multiplicative update used in the RWM algorithm.

The regularization function $R : P \to \mathbb{R}$, should be strongly convex and smooth. $R(\mathbf{x}) = -H(\mathbf{x})$ satisfies the condition.

Minimize this function to get $\mathbf{x}_i = (1 - \epsilon)$.

An alternative regularizer is $R(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x}$ where $\mathbf{w}$ is drawn uniformly from $(0, \frac{1}{\epsilon})$. This implies the online gradient descent.

In fact we will show that as long as loss function is linear, this algorithm works for any space P.

5 Online Linear Optimization

5.1 Follow the regularized leader

02/25/2016

Assume $\langle \mathbf{l}, \mathbf{x} \rangle \leq R$; $\|\mathbf{x} - \mathbf{x}'\| \leq D$.

Algorithm (1957, Hannan; 2005, Kalai, Vempala): Regularized follow the leader (RFTL). Also called "follow the perturbed leader" in Kalai, Vampala.

To simplify the notation, let $M(T) = \underset{\mathbf{x} \in D}{\operatorname{argmin}} \mathbf{l} \cdot \mathbf{x}$. We assume this is efficiently computable. Let $\mathbf{l}_{1:t} = \sum_{i=1}^{t} \mathbf{l}_i$.

The algorithm is as follows:

1. Choose $\mathbf{p} \in [0, \frac{1}{\epsilon}]^n$ uniformly at random. We will see how to pick ϵ later.
2. $\mathbf{x}_t = M(\mathbf{l}_{1:t-1} + \mathbf{p})$.

Theorem 5.1. $\mathbf{E}\left[\textit{loss of } FTRL(\epsilon)\right] \leq M(\mathbf{l}_{1:T}) \cdot \mathbf{l}_{1:T} + REGRET\epsilon T + \frac{D}{\epsilon}$.

We use the analysis of Kalai and Vampala, which is considerably cleaner.

1. Define the hypothetical strategy, be the leader: use $M(\mathbf{l}_{1:t})$ at each step t.

   Claim: Be-the-leader Beats OPT

   $$\sum_{t=1}^{T} M(\mathbf{l}_{1:t}) \cdot \mathbf{l}_t \leq M(\mathbf{l}_{1:T}) \cdot \mathbf{l}_{1:T}$$

   Proof. We use induction on time step. Supoose the proposition is true at time $T - 1$.

   $$\langle M(\mathbf{l}_{1:T-1})\mathbf{l}_{1:T-1} \rangle \leq \langle M(\mathbf{l}_{1:T})\mathbf{l}_{1:T-1} \rangle$$

   $$\sum_{t=1}^{T} M(\mathbf{l}_{1:T}) \leq M(\mathbf{l}_{1:T-1})\mathbf{l}_{1:T-1} + M(\mathbf{l}_{1:T})\mathbf{l}_T \leq M(\mathbf{l}_{1:T})l_{1:T-1} + M(\mathbf{l}_{1:T})\mathbf{l}_T = M(\mathbf{l}_{1:T})l_{1:T}$$

   □

2. BTL with noise.

Claim: for all $\mathbf{p}_0 = \mathbf{0}, \mathbf{p}_1, \ldots, \mathbf{p}_t \in \mathbb{R}^n$,

$$\sum_{t=1}^{T} M(\mathbf{l}_{1:t} + \mathbf{p}_t) \cdot \mathbf{l}_t \leq M(\mathbf{l}_{1:T}) \cdot \mathbf{l}_{1:T} + D \sum_{t=1}^{T} \|\mathbf{p}_t - \mathbf{p}_{t-1}\|_\infty$$

Proof. Let loss at time t be $\mathbf{q}_t = \mathbf{l}_t + (\mathbf{p}_t - \mathbf{p}_{t-1})$. Define $\mathbf{q}_{1:t} = \mathbf{l}_{1:t} + \mathbf{p}_t$. Apply the previous claim 1,

$$\sum_{t=1}^{T} M(\mathbf{l}_{1:t})\mathbf{q}_t \leq M(\mathbf{l}_{1:T})\mathbf{q}_{1:T} \leq M(l_{1:T}) \cdot (\mathbf{l}_{1:T} + \mathbf{p}_T)$$

□

In this case $p_1 = [0, 1/\epsilon]^n$, $p_2 = p_3 = \ldots = p_n = p_1$. The last term is $\frac{D}{\epsilon}$.

3. Look at distribution of $\mathbf{l}_{1:t-1} + \mathbf{p}$ and $\mathbf{l}_{1:t} + \mathbf{p}$. Both are distributions over n-dimensional cube of size $\frac{1}{\epsilon}$ Suppose cubes ovelap at f fraction of volumn.

   f fraction of the time they are the same. In expectation,

   $$\mathbf{E}\left[M(\mathbf{l}_{1:t-1} + \mathbf{p}) \cdot \mathbf{l}_t\right] \leq \mathbf{E}\left[M(\mathbf{l}_{1:t} + \mathbf{p}) \cdot \mathbf{l}_t\right] + (1 - f)R$$

   even though for each realization of $\mathbf{p}$, both quantities are different. When $\mathbf{p}$ is large, the overlap fraction is large.

4. Claim: $[0, \frac{1}{\epsilon}]^n$ and $[v, v + \frac{1}{\epsilon}]^n$ differ in $f \geq 1 - \epsilon|v|$.

   Proof. Fix $x \in [0, \frac{1}{\epsilon}]^n$ on some dimension i, $x_i \notin v_i + \left[0, \frac{1}{\epsilon}\right]$. Under uniform distribution,

   $$\Pr\left[x_i \notin v_i + \left[0, \frac{1}{\epsilon}\right]\right] = v\epsilon$$

   .

   By union bound,

   $$\Pr\left[\mathbf{x} \notin \left[v, v + \frac{1}{\epsilon}\right]^n\right] \leq \epsilon \|\mathbf{v}\|_1$$

   □

   Therefore

$$\mathbf{E}\left[\sum_{t=1}^{T} M(\mathbf{l}_{1:t-1} + p_t)\mathbf{l}_t\right] \leq \mathbf{E}\left[\sum_{t=1}^{T} M(\mathbf{l}_{1:t} + p_t)\mathbf{l}_t\right] + T\epsilon \|\mathbf{l}\|_1 R \leq M(\mathbf{l}_{1:T}) \cdot \mathbf{l}_{1:T} + \frac{D}{\epsilon} + T\epsilon \|\mathbf{l}\|_1 R$$

On one side, it has to be small for step 2 of adding noise, but it cannot be too small so that the third step can be bounded.

5.2 Gradient Descent

03/01/2016

To get ϵ regret, one needs $O(\frac{1}{\epsilon^2})$ step, using first order information (gradient).

5.2.1 Lower Bound

D = diameter of P; G = max norm of gradient.

Theorem 5.2. *Any algorithm for online prediction incurs* $\Omega(DG\sqrt{T})$ *regret.*

$$P = [-1, 1]^n$$

At step t, $\mathbf{g}_t$ is randomly chosen with each dimension 1 w.p. $\frac{1}{2}$ and -1 w.p. $\frac{1}{2}$.

Suppose alg is at point $\mathbf{x}_t$,

$$\mathbf{E}\left[\text{loss at time } t\right] = \mathbf{E}\left[\mathbf{g}_t \cdot \mathbf{x}_t\right] = 0$$

OPT: for each dimension i, pick 1 if sum of g_{it} is greater than 0; otherwise pick 0.

$$cost(OPT) = \mathbf{E}\left[\min_{x\in P} \sum_{t=1}^{T} g_t x\right] = n\mathbf{E}\left[-|\sum_{t=1}^{T} g_t|\right] = -\Omega(n\sqrt{T})$$

One observes this lower bound behavior in practice.

Improvement: Nesterov's accelerated gradient descent

5.3 Stochastic

Noisy gradient: ∇_x s.t. $\mathbf{E}\left[\nabla_{\mathbf{x}}\right] = \nabla f(\mathbf{x})$, $\mathbf{E}\left[\|\nabla_{\mathbf{x}}\|^2\right] \leq G^2$.

The SGD algorithm:

Input $x_1 \in P$, stepsize $\epsilon_t = \frac{1}{\sqrt{t}}$.

```
for t = 1 to T
  \nabla_t := sampled gradient at x_t
  x_{t+1} := x_t − ε_t ∇_t
```

```
end for
```
$\bar{\mathbf{x}} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{x_t}$

$$\frac{1}{T}\sum_{t=1}^{T} \nabla_t(x_t - x^*) \leq O(\frac{\sqrt{T}}{T})$$

$$\frac{1}{T}\sum_{t=1}^{T} \mathbf{E}\left[\nabla_t x_t\right] - \mathbf{E}\left[\nabla_t x^*\right] = \frac{1}{T}\sum_{t=1}^{T} \mathbf{E}\left[\nabla f(x_t) x_t\right] - \mathbf{E}\left[\nabla f(x_t) x^*\right] \leq O(\frac{1}{\sqrt{T}})$$

Use convexity

6 Online Algorithms

6.1 Move-To-Front Algorithm

03/22/2016

The first result of online algorithms.

Claim: Move To Front is 2-competitive.

Potential function $\phi_t = |\{(x, y)|x, y \text{ are in different order in } \mathsf{OPT} \text{ and } \mathsf{MTF}\}|$, i.e. number of inversions in MTF with respect to OPT.

Amortized cost of MTF: $a_t = m_t + \phi_{t+1} - \phi_t$. We will show $a_t \leq 2s_t - 1$, where s_t is the optimal cost. If so,

$$\begin{aligned}
&\sum_{t=1}^{T} a_t \leq 2OPT - T \\
\Rightarrow &\sum_{t=1}^{T} m_t + \phi_{T+1} - \phi_1 \leq 2OPT - T \\
\Rightarrow &\mathsf{MTF} \leq 2\mathsf{OPT} - T + (\phi_1 - \phi_{T+1}) \leq 2\mathsf{OPT} - T + \binom{m}{2}
\end{aligned}$$

The term $\binom{m}{2}$ represents the initial effect: the initial list could be bad. But when T is sufficiently large, it compensates the initial costs and make the total cost $\leq 2\mathsf{OPT}$.

To show $a_t \leq 2s_t$,

Suppose $\sigma_t = x$. Define $S_t(y) =$ cost spent by OPT on y (y before x in OPT). $m_t(x) =$ cost spent by MTF on y.

$S_t = \sum_y S_t(y)$. $m_t = \sum_y m_t(y)$.

Define indicators $\phi_{t+1}(y) = 1$ if (x, y) is an inversion after x is requested. $\phi_t(y) = 1$ if (x, y) is an inversion before x is requested.

$$\phi_{t+1} - \phi_t = \sum_y (\phi_{t+1}(y) - \phi_t(y))$$

$$a_t = \sum_t [m_t(y) + \phi_{t+1}(y) - \phi_t(y)]$$

$$s_t = \sum_y S_t(y)$$

Now prove case by case:

$$\forall y,\ m_t(y) + \phi_{t+1}(y) - \phi_t(y) \leq 2S_t(y)$$

$S_t(y)$ is 0 if in OPT x is before y.

We analyze case by case: Eg. $m_t(y) = 1$, $S_t(y) = 0$, MTF moves y before x, $\phi_{t+1}(y) = 0$, $\phi_t(y) = -1$. The factor of 2 is needed when $S_t(y) = m_t(y) = 1$. y appears before x. But MTF moves x before y, and OPT does not. $\phi_{t+1}(y) = 1$. The inequality is still satisfied.

There is also one case where $m_t(y) = S_t(y) = 1$ and potentials are 0, which accounts for the -1 in $2S_t(y) - 1$.

Claim: There is no deterministic algorithm whose competitive ratio is better than that of MTF.

Proof. Adversary: at each step, give the item at the end of the current list. It is always possible because the algorithm is deterministic.

Cost of any online algorithm is $O(nT)$.

If the OPT knows the request sequence, it chooses random permutation of items.

$$\mathbf{E}\left[\text{cost of serving}\sigma_t\right] = \frac{n}{2}$$

$$\mathbf{E}\left[\text{cost of OPT}\right] = \frac{nT}{2}$$

Best fixed ordering knowing $\sigma_1 \dots \sigma_T$ has cost $\leq \frac{nT}{2}$. Therefore exist an ordering such that the competitive ratio ≥ 2.

□

Remark 1. *This lower bound proof crucially needs the assumption of deterministic algorithm.*

6.2 Paging

03/31/2016

Cache has size $k << n$, where n is the number of unit-sized pages. Given a sequence of pages we define the cost as the number of evictions needed.

Longest Forward Distance (LFD): If one knows the entire request sequence (not online), the optimal decision is to evict the page whose request is furthest away in the future. One can extend to to a sequence modeled my Markov chains.

Claim: LFD is optimal.

Proof. Use exchange argument.

Suppose ALG is the opt algorithm. Suppose ALG and LFD agree till time t. Alg throws u, LFD throws v.

LFD simulates ALG going forward as follows,

1. Make the same evictions as ALG (cache state is the same)
2. If ALG evicts v, LFD evicts u. Cache state coincides
3. If v is requested and ALG has v in cache, then LFD has u in cache. Since request of u is before request v, ALG must have already incurred a page fault while LFD did not.
4. When u is requested, ALG faults.

The number of page faults of ALG and LFD is the same. Therefore LFD is as good as any optimal algorithm. □

6.3 Online Paging

fLFD, FIFO, FWF (evict every algorithm), FREQ, LRU (least recently used)

Marking algorithm ALG: run in phases

At start of a phase, all pages "unmarked"

On page request: Mark page If eviction needed, evict some unmarked page If all pages marked, unmark, start a new phase

Any marking algorithm is k-competitive (k is the size of cache).

Proof. We argue phase by phase the algorithm is k-competitive.

Number of evictions per phase is $\leq k$. Number of distinct page requests in phase is $k+1$ (last one cannot fit). By pigeonhole, at least one page fault occurs (for OPTs).

□

Claim: Any deterministic algorithm is $\geq k$-competitive.

Proof. At every step, ask for page not in the cache.

Given the sequence, the OPT can use LFD. At fault, it evicts the page that is furtherest in the future. There are at least k distinct requests between current request and that request in the future. It won't fault for the next $k-1$ distinct pages.

□

Remark 2. *FIFO, which is not a marking algorithm, is also k-competitive.*

To improve bound, we can use randomized algorithms. This also explains why marking works well in practice (stochastic inputs).

Claim: Randomized marking algorithm is $2\log k$-competitive.

Proof. New: pages requested in phase i, but not $i-1$. Let $M_i = |\text{new}|$. Old: pages requested in phase i and $i-1$.

Let x_j be j-th distinct old request. m_j is the number of new pages requested before x_j. They must be in cache and marked. There are $j-1$ old requests in cache. Let S_j be the rest old pages that are unmarked. $|S_j| = k - m_j - j + 1$. The randomized marking algorithm randomly evicts a page in S_j. S_j is a random subset of $k-j+1$ old pages.

x_j incurs eviction if $x_j \notin S_j$. $\Pr[x_j \notin S_j] = \frac{m_j}{k-j+1} \leq \frac{M_i}{k-j+1}$.

$$\mathbf{E}[Costofalginphasei] = \sum_{j=1}^{k-M_i} \frac{M_i}{k-j+1} + M_i \leq M_i H_k$$

Where H_k is the sum of harmonic sequence up to $\frac{1}{k}$.

Opt must have made at least M_i page faults in phase i and phase $i-1$. Sum over all phases.

$$\mathsf{OPT} \geq \frac{1}{2}\sum_i M_i$$

Therefore $\mathbf{E}[ALG] \leq 2H_k \mathbf{E}[\mathsf{OPT}]$. □

Claim: Yao's Theorem Any randomized algorithm must be $\Omega(\log k)$-competitive.

Proof. Best possible randomized algorithm:

$$\min_D \max_{\sigma_j} \mathbf{E}_{i \in D} \left[\frac{[Alg_i(\sigma_j)]}{\mathsf{OPT}(\sigma_j)} \right]$$

Let distribution over sequence be P. The minimax theorem gives

$$\min_D \max_{\sigma_j} \mathbf{E}_{i \in D} \left[\frac{[Alg_i(\sigma_j)]}{\mathsf{OPT}(\sigma_j)} \right] \geq \max_P \min_i \frac{[Alg_i(\sigma_j)]}{\mathbf{E}_{\sigma_j \in P}[\mathsf{OPT}(\sigma_j)]}$$

This gives a lower bound on the competitive ratio: performance of the best algorithm over any distribution of inputs.

Idea: pick a distribution. Show that the best algorithm for this distribution is still lower bounded by $\log k$ if it doesn't know the sequence. □

There are $k+1$ pages. At each step, random page is requested.

ALG does not know the future. For a new request, $\mathbf{E}[costofalg] = \frac{T}{k+1}$.

The time OPT faults is the problem of coupon collector: kH_k. $\mathbf{E}[Durationbetweenfaults] = kH_k$. If the events are iid, the number of faults is $\frac{T}{kH_k}$. Vand's identity

6.4 Online Maximal Matching

04/06/2016

Dual fitting

We know one side of the graph, L, initially. Vertices of R arrive online. For each vertex that arrives, learn all its incident edges, we need to decide which one to match online.

Applications: adword matching search queries and advertisements.

Greedy: choose one of the edges whose other end point is not matched.

No deterministic algorithm can do better than a factor of 2. Bad example: $v_1 \to u_1$, $v_2 \to u_1$. WOLG pick (v_1, u_1), then $v_1 \to u_2$.

LP

Let $x_e = 1$ if the edge is in the matching

$$\max \sum_e x_e$$

$$\sum_{e \in \delta(v)} x_e \leq 1, \forall v \in L \cup R$$

$$x_e \geq 0 \forall e$$

Dual (vertex cover):

$$\min_{v \in L \cup R} p_v$$

$$p_v + p_w \geq 1 \forall (v, w) \in E$$

$$p_v \geq 0 \forall v \in L \cup R$$

Idea: find a feasible dual solution whose value is twice as greedy. $\mathsf{OPT} = DUAL^* \leq DUAL \leq 2Greedy$.

If greedy adds edge (v, w), $q_v = \frac{1}{2}$, $q_w = \frac{1}{2}$. It won't assignment more than $\frac{1}{2}$ to any vertex.

$$\sum_{v \in L \cup R} q_v = Greedy$$

For any $(v, w) \in L \times R$, $q_v + q_w \geq 0.5$. w should have matched to v if v is not yet matched.

Hence if we define $p_v = 2q_v$, these variables satisfies the dual constraints.

$$\sum_{v \in L \cup R} p_v = 2Greedy$$

By weak duality, greedy algorithm is a 2-approximation.

In fact for every edge added, the opt could have created at most 2 edges.

6.4.1 Fractional Matching

Let y_v be the fraction of matching: $y_v = \sum_{e=(u,v)} x_v$.

Waterfilling

When v arrives, for edges (u_i, v) with each u_i having value y_{u_i}, do water-filling to make them as equal as possible.

If we assign y_e to q_v and q_u values equally, we can prove better than a factor of 2. There will always exist an edge that has $q_u + q_v = \frac{1}{2}$.

Idea: increase q_v as a higher rate if y_v is approaching 1.

When pushing $\mathrm{d}z$ amount of flow from w to v,

$$\frac{\mathrm{d}q_v}{\mathrm{d}z} = g(y_v)$$

$$\frac{\mathrm{d}q_w}{\mathrm{d}z} = 1 - g(y_v)$$

We want g to be convex.

Take some edge $(w, v) \in E$.

1. Case: At the end of the algorithm, $y_v = 1$, i.e. v is saturated.

$$q_v + q_w \geq q_v = \int_0^1 g(y)\mathrm{d}y$$

2. Case: $y_v < 1$ at the end of the algorithm. w pushes all water to v when w arrives. If w pushed water to m, $y_m \leq y_v \Rightarrow 1 - g(y_m) \leq 1 - g(y_v)$. (waterfilling properties) q_w increases at rate $\geq 1 - g(y_v)$. $q_w \geq 1 - g(y_v)$. $q_v = \int_0^{y_v} g(y)\mathrm{d}y$.

$$q_v + q_w \geq 1 - g(y_v) + \int_0^{y_v} g(y)\mathrm{d}y \tag{6.1}$$

We want to find the min value of the 2 cases, among all edges. We want q_v be constant w.r.t. y_v. Take derivative and set it to 0 gives $g(y_v) = ce^{y_v}$.

Substitute it into the expressions for $q_v + q_w$ for both cases, the min is $\min\{k(e-1), 1-k\}a$. Find k maximizes the min. $k = e^{-1}$, and $q_v + q_w \geq \frac{e-1}{e}$.

When pushing water to v, q_v increases exponentially. $p_e = \frac{e}{e-1}q_v$.

$$\frac{e}{e-1}WF \geq OPT \Rightarrow WF \geq \frac{e-1}{e}\mathsf{OPT}$$

6.4.2 Randomized Algorithm for Matching

04/07/2016

When $w \in R$ arrives, match to available $v \in L$, with smallest y_v.

Suppose (v, w) is matched, set $q_v \leftarrow g(y_v) = e^{y_e - 1}$; $q_v \leftarrow 1 - g(y_v)$. y_v is a random variable between 0 and 1.

$\sum q_v =$ size of matching.

Take (v, w), can we bound $\mathbf{E}[q_v] + \mathbf{E}[q_w]$? Fix all $y_u \in L \backslash \{v\}$. Bound $\mathbf{E}[q_v + q_w | L \backslash \{v\}]$.

Pretend: v does not exist. Suppose w gets matched to t (or is not matched). If t does not exist, pretend $y_t = 1$.

Original: if $y_v < y_t$, v is matched (otherwise w would match v when w arrives). w is matched to a vertex of value $\leq y_t$.

From 2, $\mathbf{E}[q_w | y - v] \geq 1 - g(y_t)$.

From 1, y_v is chosen uniformly, if $y_v < y_t$, we get $g(y_v)$. $\mathbf{E}[q_v | y - v] \geq \int_0^{y_t} g(y) \mathrm{d}y$.

$\mathbf{E}\left[q_v + q_w | L \backslash \{v\} = \int_0^{y_t} g(y) \mathrm{d}y + 1 - g(y_t)\right]$. This equation is the same as **??**. $\mathbf{E}[q_v + q_w | L \backslash \{v\}] \geq 1 - e^{-1}$ is true for every realization of $y(L \backslash \{v\})$, which means $\mathbf{E}[q_v + q_w] \geq 1 - e^{-1}$.

Therefore $\mathbf{E}[\sum_v q_v] \geq \mathsf{OPT}\left(\frac{e-1}{e}\right)$.

(This is somewhat magical)

6.5 Primal-Dual Method

Covering problem:

$$\min \sum_{c=1}^{n} c_i x_i$$

$$\sum_{i=1}^{n} a_{ij} x_i \geq b_j, \forall j = 1, 2, \ldots, m$$

The dual is the packing problem:

$$\max \sum_{j=1}^{m} b_j y_j$$

$$\sum_{j=1}^{m} a_{ij} y_j \leq c_j, \forall i = 1, 2, \ldots, m$$

We have complementary slackness and weak duality.

Approximate complementary slackness:

$$x_i > 0 \Rightarrow \mathbf{a}_i \cdot \mathbf{y} \in [\frac{c_i}{\alpha}, c_i]$$

$$y_i > 0 \Rightarrow \mathbf{a}_j \cdot \mathbf{x} \in [b_j, \frac{b_j}{\beta}]$$

(approximate solution)

Then by weak duality, $\mathbf{b} \cdot \mathbf{y}$ and $\mathbf{c} \cdot \mathbf{x}$ are close to OPT.

$$\mathbf{b} \cdot \mathbf{y} \leq \mathbf{c} \cdot \mathbf{x} \leq \alpha\beta \mathbf{b} \cdot \mathbf{y}$$

Proof.

$$\mathbf{c} \cdot \mathbf{x} = \sum c_i x_i \leq \sum (\alpha \mathbf{a}_i \cdot \mathbf{y}) x_i = \alpha \sum (\mathbf{a}_j \cdot \mathbf{x}) y_i \leq \alpha \sum_j \beta b_j y_j = \alpha\beta \mathbf{b} \cdot \mathbf{y}$$

□

Ski rental: buy: B dollars, rent: 1 dollar.

Deterministic 2-competitive: rent for the first B times, and buy after that.

x is 1 if OPT buy, 0 if rent. $z_j = 1$ if OPT rent on j-th trip, 0 otherwise.

$$\min Bx + \sum_{j=1}^{k} z_j$$

$$x + z_j \geq 1, \forall j$$

Online: On day j, one gets 1 additional constraint and variable, and we need to maintain a feasible solution. x has to be monotone. The feasible solution we maintain has to be integral.

Dual:

$$\max \sum_{j=1}^{k} y_j$$

$$\sum_{j=1}^{k} y_j \leq B$$

$$y_j \leq 1, \forall j$$

On day j, if $+z_j \geq 1$, do nothing. Else increase y_j as much as possible:

- If $y_j = 1$ (constraint corresponding to z_j is tight), set $z_j = 1$.
- If $\sum_{y_j} = B$ (constraint corresponding to x is tight), set $x = 1$.

$x > 0 \Rightarrow \sum_j y_j = B$; $z_j > 0 \Rightarrow y_j = 1$. $y_j > 0 \Rightarrow 1 \leq x + z_j \leq 2$. Therefore

$$2(D) \geq (P) \geq (D)$$

Use relaxation, and find fractional solution. There exist a rounding to randomized algorithm to integer solution with better that 2 bound.

Fractional alg: $x \leftarrow 0$

On each day j, if $x < 1$ then $z_j \leftarrow 1 - x$. $\Delta_x = \frac{x}{B} + \frac{1}{c}\frac{1}{B}$. $y_j \leftarrow 1$. End if

Claim: Both primal and dual solutions are feasible.

Proof. $(x, \mathbf{z})$ is clearly feasible. To show $\mathbf{y}$ is feasible, we want to make sure the number of time x increases is less than B ($\sum_j y_j$ increases by 1 after each incrases of x).

$$x_B = \frac{(1 + \frac{1}{B})^B - 1}{c} = 1$$

To make it true, choose $c = (1 + \frac{1}{B})^B - 1 \approx e - 1$.

□

Approximate C.S.:

Proof. If $x < 1$, $\Delta Dual = 1$. $\Delta Primal = B\Delta x + z_j = B\Delta x + (1 - x) = 1 + \frac{1}{c} \approx \frac{e}{e-1}$.

At the beginning, primal and dual are 0. We are increasing the the primal and dual at roughly constant rate. In the end,

$$Dual \leq Primal \leq \frac{e}{e-1} Dual \leq \frac{e}{e-1} \mathsf{OPT}$$

The optimal solution lies somewhere between

$$\Rightarrow Bx + \sum z_j \leq \frac{e}{e-1} OPT$$

□

Rounding: Choose $\alpha \in [0, 1]$ at random. If $x > \alpha \Rightarrow \hat{x} = 1, \hat{z}_j = 0$. If $x < \alpha \Rightarrow \hat{x} = 0, \hat{z}_j = 1$.

$\mathbf{E}[\hat{x}] = x$, $\mathbf{E}[\hat{z}_j] = z_j$. In expectation it has same cost as the fractional solution.

6.6 Budget Allocation

04/13/2016

More on Primal-dual

Keyword j arrives Match j to advertiser i Charge i amount b_{ij}, deduct b_{ij} from its budget B_i. Goal: maximize total revenue, sum of b_{ij} charged.

Dual (consistent with ski rental)

$$\max \sum_{i,j} b_{ij} y_{ij}$$

$$\sum_i y_{ij} \leq 1, \ \forall \text{ keyword } j$$

$$\sum_j b_{ij} y_{ij} \leq B_i \ \forall \text{ advertiser } i$$

Primal

$$\min_i B_i x_i + \sum_j z_j$$

$$z_j \geq b_{ij}(1 - x_i) \ \forall i, j$$

Algorithm:

```
init z_i ← 0, ∀i
when j arrives, find i = argmax_{i'} b_{i'j}(1 − x_i)
y_ij ← 1
price = min{B_i, b_ij}
z_j = b_ij(1 − z_i)
Δx_i = x_i b_ij/B_i + b_ij/B_i · 1/(c−1)
```

Theorem 6.1. *Let $R_{max} = \max \frac{b_{ij}}{B_i}$, a small number. $c = (1 + R_{max})^{R_{max}^{-1}} \to e$. Competitive ratio is $(1 - \frac{1}{e})(1 - R_{max}) \to 1 - \frac{1}{e}$ as $R_{max} \to 0$.*

Proof. primal is always feasible, since z_j only decreases over time as x_i increases.

Δ DUAL $= b_{ij}$ if $y_{ij} \leftarrow 1$. Δ PRIMAL $= B_i \Delta x_i + z_j = B_i(x_i \frac{b_{ij}}{B_i} + \frac{b_{ij}}{B_i} \frac{1}{c-1}) + b_{ij}(1 - x_i) = b_{ij} \frac{c}{c-1}$.

PRIMAL $\geq$ DUAL $\geq \frac{c-1}{c}$ PRIMAL

But we need to show that DUAL is feasible. For the second constraint, Violate by at most b_{ij} for every i.

We will show that $\sum b_{ij}y_{ij} \geq B_i \Rightarrow x \geq 1$. We will show by induction, $x_i \geq \frac{1}{c-1}(c^{\sum b_{ij}y_{ij}B_i} - 1)$. the power is the fraction of i's budget spent.

suppose k is matched to i. $x_i^{new} = x_i^{old}(1 + \frac{b_{ij}}{B_i}) + \frac{b_{ik}}{B_i}\frac{1}{c-1} \geq \frac{1}{c-1}(c^{\sum b_{ij}y_{ij}B_i} - 1)$

We choose c such that $(1 + \frac{b_{ik}}{B_i}) \geq [(1 + R_{max})^{R_{max}^{-1}}]^{\frac{b_{ik}}{B_i}} = c^{\frac{b_{ik}}{B_i}}$.

□

6.7 Secretary Problem

04/15/2016

What guarantee can we obtain when input is not adversarial? What other models are there?

Secretaries come to interview in an online fashion. If the employer decides to move on, he cannot come back to that secretary. Stops when a secretary is hired. Get reward only if one gets the BEST secretary.

There is nothing we can do if the input is adversarial. We can assume that the order is random. The adversary chooses a ranking of the secretaries, and a random permutation of it is presented to the employer.

Model as a Markov decision process. The system has 4 types of states. Let k be the current time.

- $(\Delta, k, 1)$: secretary already picked and is best so far
- $(\Delta, k, 0)$: secretary already picked and is not the best
- $(k, 1)$: secretary not yet picked and the current secretary is the best so far
- $(k, 0)$: secretary not yet picked and the current secretary is not the best

$(k, 1)$ goes to $(k + 1, 1)$ w.p. $\frac{1}{k+1}$; $(k, 0)$ goes to $(k + 1, 1)$ w.p. $\frac{1}{k+1}$.

The real choice is at $(k, 1)$. For state $(k, 0)$, we won't hire the current secretary since we won't get any reward for hiring him.

At $(k, 1)$, if not hire,

$$\frac{1}{k+1}J(k+1,1) + \frac{k}{k+1}J(k+1,0)$$

If hire,

$$\Pr[k \text{ is max }] = \frac{k}{k+1}\frac{k+1}{k+2}\cdots = \frac{k}{M}$$

We pick $\max\{\frac{k}{M}, \frac{1}{k+1}J(k+1,1) + \frac{k}{k+1}J(k+1,0)\}$.

At $(k, 0)$, if hire we get reward 0; if not hire, it is the same as not hiring at state $(k, 1)$:

$$\frac{1}{k+1}J(k+1,1) + \frac{k}{k+1}J(k+1,0)$$

Claim: If $J(k,1) > \frac{k}{M}$, then $J(k-1,1) > \frac{k-1}{M}$.

Proof.

$$J(k,1) > \frac{k}{M} \Rightarrow J(k,0) > \frac{k}{M}$$

$\frac{1}{k}J(k,1) + \frac{k-1}{k}J(k,0) > \frac{k}{M} \Rightarrow J(k-1,1) \geq \frac{k}{M} > \frac{k-1}{M}$. □

This claim means there is a transition point after which we start to hire.

Policy: $(k, 1)$ for $k < r$, not hire; $(k, 1)$ for $k \geq r$, hire. Ignore the first r secretaries.

Let best secretary be n. if the algorithm choose n, then $n > r$, and the best in $[1 : n]$ is in $[1 : r]$.

Given n

$$\Pr[n+1 \text{ is max}] = \frac{1}{M}$$

$$\Pr[\text{best in } [1:n] \text{ is in} [1:r]] = \frac{r}{n}$$

$$\Pr[n+1 \text{ is max and chosen}] = \frac{r}{Mn}$$

$$\Pr[\text{Alg gets reward}] = \frac{r}{M}\sum_{n \geq r}\frac{1}{n}$$

Let $x = \frac{r}{M}$, approximate summation by integral,

$$Rew(r) = x\int_x^1 \frac{1}{t}\mathrm{d}t = x\ln\frac{1}{x}$$

When x is maximized, $\frac{1}{e}$. The optimal policy throws away first $\frac{1}{e} \approx 0.37$ of the secretaries, and w.p. $\frac{1}{e}$ the algorithm chooses the best.

Can we assume something more about input and reduce the ratio of $\frac{1}{e}$?

Input is i.i,d. from distribution D.

Definition 1. *Monotone Hazard Rate Let $F(x)$ be the survival function, $f(x)$ be the pdf. Monotone hazard rate implies $\frac{f(x)}{F(x)}$ increases in x.*

In the device failure senario, the failure rate increases over time conditioned on not failing so far. It is the derivative on $\ln F(x)$. The MHR implies that $\ln F(x)$ is concave.

Most distributions satisfy the property.

Exception: heavy tail distributions such as $f(x) = \frac{1}{x}$.

One can show that $r = \frac{M}{\log M}$ in the strategy achieves the optimal if one assumes MHR. This proof is complicated.

We can show a weaker result: the max in the first $\frac{M}{\log M}$ is very close to the global max.

Throw t dots on the graph of $F(x)$, with reasonable probability, max $\leq F^{-1}(\frac{1}{T \log T})$. Throw $\frac{T}{\log T}$, max $\geq F^{-1}(\frac{\log T}{T})$.

Claim: For MHR distributions, $x_1 = F^{-1}(\frac{1}{T \log^s T})$, $x_2 = F^{-1}(\frac{\log^r T}{T})$. Then

$$x_1 \leq x_2[1 + (r + s + o(1))\frac{\log \log T}{\log T}]$$

Proof. Let $\Delta = x_1 - x_2$. Let $h(x) = -\log F(x)$ which is increasing and convex (MHR).

$$h(x_1) - h(x_2) = (r + s) \log \log T$$

By convexity,

$$h(x_1) - h(x_2) \geq \Delta h'(x_2) \Rightarrow h'(x_2) \leq \frac{(r + s) \log \log T}{\Delta}$$

But

$$h(x_2) \leq h(0) + x_2 h'(x_2) \Rightarrow \log T - r \log \log T \leq x_2 \frac{(r + s) \log \log T}{\Delta} \Rightarrow \Delta = x_1 - x_2 \leq x_2[1 + (r + s + o(1))\frac{\log \log T}{\log T}]$$

□