Learning with Global Cost in Stochastic Environments

Eyal Even-dar, Shie Mannor and Yishay Mansour

Technion

COLT, June 2010

(You haven't heard it last year.)

Learning with Global Cost in Stochastic Environments

Eyal Even-dar, Shie Mannor and Yishay Mansour

Technion

COLT, June 2010

(You haven't heard it last year.) Really.

Table of contents

Regret Minimization

- Let L be a sequence of losses of length T, then $R(T, L) = \mathbb{E}[\max(Cost(alg, L) - Cost(\text{opt in hindsight}, L), 0)]$
- $R(T) = \max_L R(T, L)$

Regret Minimization

- Let L be a sequence of losses of length T, then $R(T, L) = \mathbb{E}[\max(Cost(alg, L) - Cost(\text{opt in hindsight}, L), 0)]$
- $R(T) = \max_L R(T, L)$
- An algorithm is **no-regret** if $R(T)$ is sublinear in T.

Regret Minimization

- Let L be a sequence of losses of length T, then $R(T, L) = \mathbb{E}[\max(Cost(alg, L) - Cost(\text{opt in hindsight}, L), 0)]$
- $R(T) = \max_L R(T, L)$
- An algorithm is **no-regret** if $R(T)$ is sublinear in T.

Cost is in general not additive

Regret Minimization (biased view)

So we have come a long way

- N experts (full and partial information)
- Shortest path (full and partial information)
- Strongly convex functions (better bounds)
- Many more... (40% of papers this year).

Regret Minimization (biased view)

So we have come a long way

- N experts (full and partial information)
- Shortest path (full and partial information)
- Strongly convex functions (better bounds)
- Many more... (40% of papers this year).

But some room to grow

Regret Minimization (biased view)

So we have come a long way

- N experts (full and partial information)
- Shortest path (full and partial information)
- Strongly convex functions (better bounds)
- Many more... (40% of papers this year).

But some room to grow

- There is no memory/state (in most works).

Regret Minimization (biased view)

So we have come a long way

- N experts (full and partial information)
- Shortest path (full and partial information)
- Strongly convex functions (better bounds)
- Many more... (40% of papers this year).

But some room to grow

- There is no memory/state (in most works).
- Losses are assumed to be additive across time (in almost all works).

Regret Minimization (biased view)

So we have come a long way

- N experts (full and partial information)
- Shortest path (full and partial information)
- Strongly convex functions (better bounds)
- Many more... (40% of papers this year).

But some room to grow

- There is no memory/state (in most works).
- Losses are assumed to be additive across time (in almost all works).
- Most algorithms are essentially greedy (bad for job talks).

Regret Minimization with State

- Routing [AK ....]
- MDPs [EKM, YMS]
- Paging [BBK]
- Data structures [BCK]
- **Load balancing – this talk**

Are we optimizing the true loss function?

- Predicting click through rates (calibration)
- Handwriting recognition (calibration)
- Relevant documents, viral marketing (sub modular function)

Are we optimizing the true loss function?

- Predicting click through rates (calibration)
- Handwriting recognition (calibration)
- Relevant documents, viral marketing (sub modular function)
- **Load balancing**

Model

- N alternatives
- Algorithm chooses a distribution $\bar{p}_t$ over the alternatives and then observes loss vector $\bar{\ell}_t$.
- Algorithm accumulated loss: $\bar{L}_t^A = \sum_{\tau=1}^t \bar{\ell}_\tau \cdot \bar{p}_\tau$
- Overall loss: $\bar{L}_t = \sum_{\tau=1}^t \bar{\ell}_\tau$
- Algorithm cost: $C(\bar{L}_t^A)$, where C is a **global cost function**.
- OPT cost: $C^*(\bar{L}_t) = \min_{\alpha \in \Delta(N)} C(\alpha \cdot \bar{L}_t)$.
- Regret: $\max\{C(\bar{L}_t^A) - C^*(\bar{L}_t), 0\}$.

Assume C is L_d norm ($d \geq 1 \implies$ C is convex and C^* concave).

Model - load balancing with makespan

Assume makespan: $C = \|\cdot\|_\infty$.

Time	loss	Dist.	Alg Accu.	$C(Alg)$	Over loss	C^*
1	(1,1)	(.5,.5)	(.5,.5)	.5	(1,1)	.5

Model - load balancing with makespan

Assume makespan: $C = \|\cdot\|_\infty$.

Time	loss	Dist.	Alg Accu.	$C(Alg)$	Over loss	C^*
1	(1,1)	(.5,.5)	(.5,.5)	.5	(1,1)	.5
2	(1,0)	(.5,.5)	(1,.5)	1	(2,1)	.66

Model - load balancing with makespan

Assume makespan: $C = \|\cdot\|_\infty$.

Time	loss	Dist.	Alg Accu.	$C(Alg)$	Over loss	C^*
1	(1,1)	(.5,.5)	(.5,.5)	.5	(1,1)	.5
2	(1,0)	(.5,.5)	(1,.5)	1	(2,1)	.66
3	(1,0)	(.33,.66)	(1.33,.5)	1.33	(3,1)	.75

Model - load balancing with makespan

Assume makespan: $C = \|\cdot\|_\infty$.

Time	loss	Dist.	Alg Accu.	$C(Alg)$	Over loss	C^*
1	(1,1)	(.5,.5)	(.5,.5)	.5	(1,1)	.5
2	(1,0)	(.5,.5)	(1,.5)	1	(2,1)	.66
3	(1,0)	(.33,.66)	(1.33,.5)	1.33	(3,1)	.75
4	(0,1)	(.25,.75)	(1.33,1.25)	1.33	(3,2)	1.2

Model - load balancing with makespan

Assume makespan: $C = \|\cdot\|_\infty$.

Time	loss	Dist.	Alg Accu.	$C(Alg)$	Over loss	C^*
1	(1,1)	(.5,.5)	(.5,.5)	.5	(1,1)	.5
2	(1,0)	(.5,.5)	(1,.5)	1	(2,1)	.66
3	(1,0)	(.33,.66)	(1.33,.5)	1.33	(3,1)	.75
4	(0,1)	(.25,.75)	(1.33,1.25)	1.33	(3,2)	1.2

Minimizing the sum of losses does not minimize C^* and vice versa

Model - load balancing with makespan

Let's focus on the makespan (L_∞) for now.

Optimal policy in hindsight the load vector $\bar{L}$ is

$$p_i = \frac{1/L_i}{\sum_{j=1}^{N} 1/L_j}$$

Cost of the optimal policy is

$$C^*(\bar{L}) = \frac{1}{\sum_{j=1}^{N} 1/L_j} = \frac{\prod_{j=1}^{N} L_j}{\sum_{j=1}^{N} \prod_{i \neq j} L_i}$$

The Loss Model

The loss sequence is generated by a stochastic source. In the talk we consider a very simple case, however the results hold in general.

The Loss Model

The loss sequence is generated by a stochastic source. In the talk we consider a very simple case, however the results hold in general.

The loss vector allows correlation between the arms: some measure D **provided IID loss vectors.** (Note: arms are possibly correlated.)

The Loss Model

The loss sequence is generated by a stochastic source. In the talk we consider a very simple case, however the results hold in general.

The loss vector allows correlation between the arms: some measure D provided IID loss vectors. (Note: arms are possibly correlated.)

Known D and unknown D are both interesting.

(We thought known D would be easy - how hard can the stochastic case be if you solved the adversarial case and you know the source?)

Known source - a simple example

- Consider two machines:
- Each time w.p $1/2$ load vector is $(1,0)$ and w.p $1/2$ load vector is $(0,1)$
- W.h.p the cost of the best fixed policy in hindsight is $T/4 - O(1)$
- **What is the optimal policy?**

"Naive model based"

Standard technique in control/machine learning:

1. Learn the model
2. Compute optimal policy for the learned model

AKA "certainty equivalence"

"Naive model based"

Standard technique in control/machine learning:

1. Learn the model
2. Compute optimal policy for the learned model

AKA "certainty equivalence"

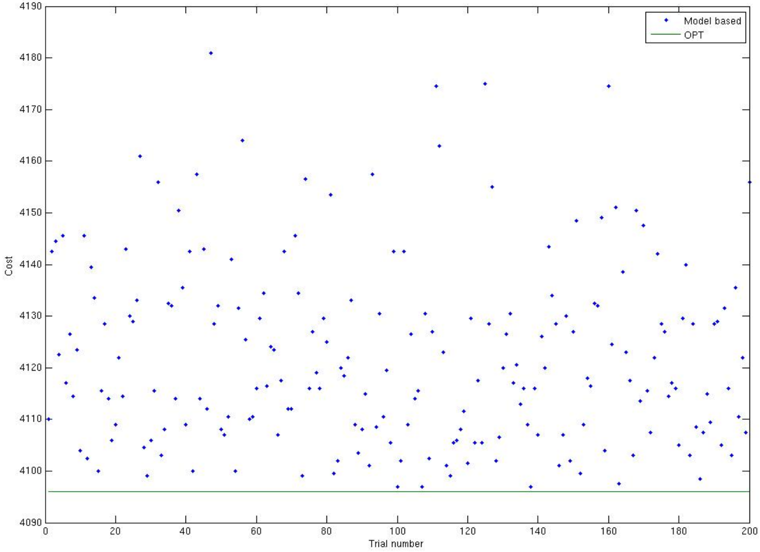
We already know the model, so let's do the following:

Naive model based algorithm

At each time-step assign $1/2$ of the job to machine 1 and half to machine 2

How good is it? is it optimal?

"Naive model based" - Simulation



Performance analysis

Cost ingredients

- Sum of actual loads on two machines
- Difference between the machines

$$\max(x, y) = \frac{x+y}{2} + \frac{|x-y|}{2}$$

"Naive model based" - Analysis

Analysis

- **Expected sum:** $T/2$ (like every algorithm...)
- **Difference:** W.h.p Load on one machine is at least $T/4 + \sqrt{T}/2$ and on the second machine is $T/4 - \sqrt{T}/2$. Thus difference is $\sqrt{T}$.
- **Cost:** At least $T/4 + \sqrt{T}/2$.
- **Regret:** $O(\sqrt{T})$

"Naive model based" - Analysis

Analysis

- **Expected sum:** $T/2$ (like every algorithm...)
- **Difference:** W.h.p Load on one machine is at least $T/4 + \sqrt{T}/2$ and on the second machine is $T/4 - \sqrt{T}/2$. Thus difference is $\sqrt{T}$.
- **Cost:** At least $T/4 + \sqrt{T}/2$.
- **Regret:** $O(\sqrt{T})$

Can we do better?

Least loaded machine

The algorithm

At every time-step assign the next job to the least loaded machine

Least loaded machine

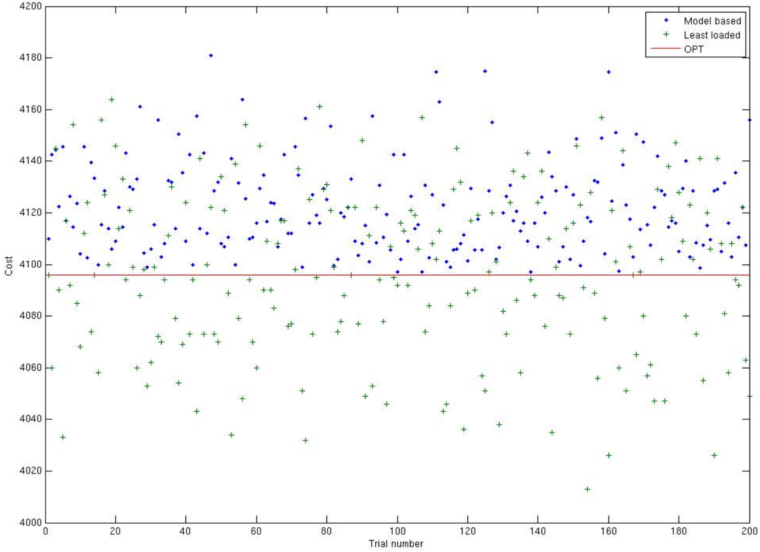
The algorithm

At every time-step assign the next job to the least loaded machine

Analysis

- **Expected sum:** $T/2$
- **Expected difference:** < 1
- **Expected cost:** $T/4$
- **Expected regret:** $O(\sqrt{T})$

Comparison - Simulation



Regret and bias-variance tradeoffs

Least loaded machine

- In terms of expected loss, the algorithm is optimal
- Regret is still $O(\sqrt{T})$.
- The regret measures the variance and the bias for this setting!

Regret and bias-variance tradeoffs

Least loaded machine

- In terms of expected loss, the algorithm is optimal
- Regret is still $O(\sqrt{T})$.
- The regret measures the variance and the bias for this setting!

Can we lower the regret while maintaining the optimal expected loss?

Balance at the end

End balance

- Until $T - 4\sqrt{T}$: play at random $(.5, .5)$
- After time $T - 4\sqrt{T}$: use least loaded machine algorithm

Balance at the end

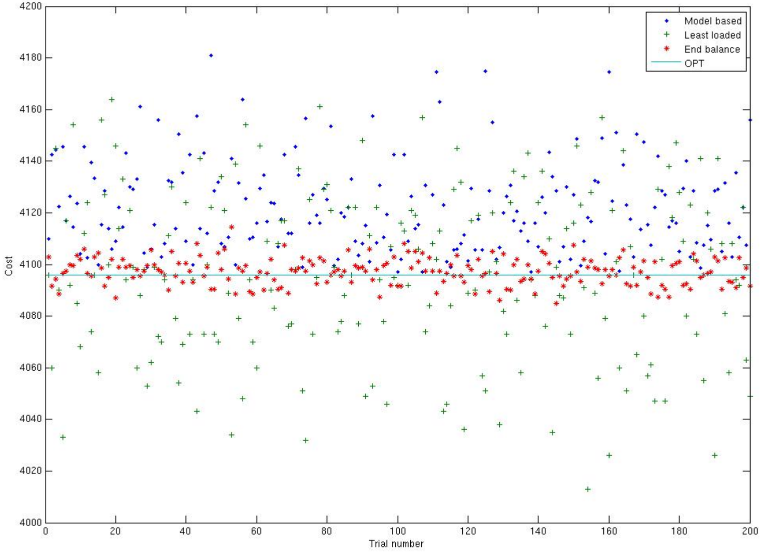
End balance

- Until $T - 4\sqrt{T}$: play at random $(.5, .5)$
- After time $T - 4\sqrt{T}$: use least loaded machine algorithm

Analysis

- **Expected sum:** $T/2$
- **Expected difference:** < 1
- **Expected cost:** $T/4$
- **Expected regret:** $O(T^{1/4})$

Comparison - Simulation



Improved algorithms

Recursive balance

- Partition time into blocks of size, $T/2, T/4, T/8, \ldots 1$. (Yes: blocks become smaller.)
- At every block set play $\frac{1}{2}+\epsilon$ to balance the “offset” from the previous block.
    - “offset” - the deviation of the process from its true probability (not influenced by the algorithm)

Improved algorithms

Recursive balance

- Partition time into blocks of size, $T/2, T/4, T/8, \ldots 1$. (Yes: blocks become smaller.)
- At every block set play $\frac{1}{2} + \epsilon$ to balance the "offset" from the previous block.
    - "offset" - the deviation of the process from its true probability (not influenced by the algorithm)
- Regret of the algorithm is $O(\log T)$

Improved algorithms

Recursive balance

- Partition time into blocks of size, $T/2, T/4, T/8, \ldots 1$. (Yes: blocks become smaller.)
- At every block set play $\frac{1}{2} + \epsilon$ to balance the "offset" from the previous block.
  - "offset" - the deviation of the process from its true probability (not influenced by the algorithm)
- Regret of the algorithm is $O(\log T)$

Anytime

- Set $\epsilon = 1/T^{1/4}$
- At every time step assign weight $\frac{1}{2} + \epsilon$ on the least loaded machine.

Improved algorithms

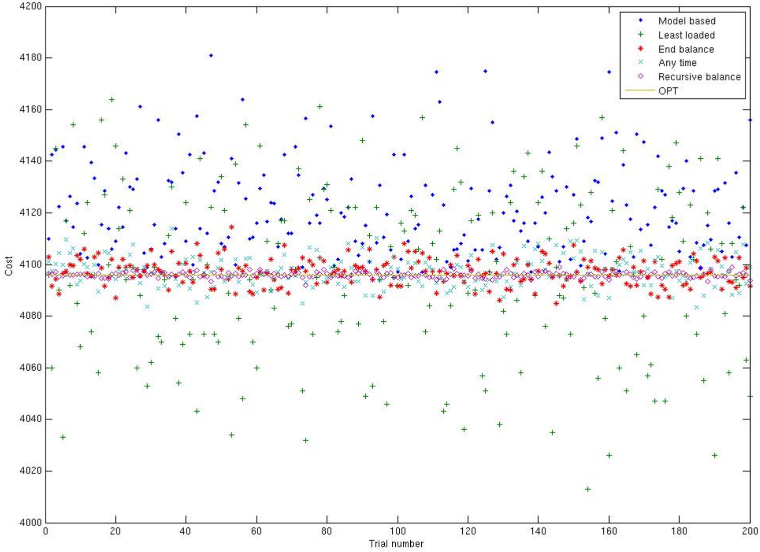
Recursive balance

- Partition time into blocks of size, $T/2, T/4, T/8, \ldots 1$. (Yes: blocks become smaller.)
- At every block set play $\frac{1}{2} + \epsilon$ to balance the "offset" from the previous block.
  - "offset" - the deviation of the process from its true probability (not influenced by the algorithm)
- Regret of the algorithm is $O(\log T)$

Anytime

- Set $\epsilon = 1/T^{1/4}$
- At every time step assign weight $\frac{1}{2} + \epsilon$ on the least loaded machine.
- Regret of the algorithm $O(T^{1/4})$ **any time**.

Comparison - Simulation



Analysis

Define generic properties of a desired phased algorithm.

- P1 The empirical frequencies of the losses are close their true expectations.
- P2 The base weights are close to the optimal weight for all actions.
- P3 The phase length does not shrink too fast.

We analyze a generic algorithm with the above properties:
$\Longrightarrow$ Regret is small if properties hold for most phases with high probability.
Define a generic algorithm: Use a base weight vector w^*. During phase k the weight of action i the algorithm does not change, and it equals

$$w^k(i) = w^*(i) + \frac{T^{k-1}}{T^k}(opt^{k-1}(i) - w^*(i))$$

Makespan: known probabilities

Theorem

For known probabilities the regret is at most $O(\log T)$

- Set $w^*(i) = \frac{1/p(i)}{P}$, where $P = \sum_{i=1}^{n} 1/p(i)$, i.e., the optimal allocation for p.
- Set the number of phases $m = \log(T)$.
- Set the length of phase k to be $T^k = T/2^k$ for $k \in [1, m]$.

Makespan: Unknown probabilities

Theorem

For unknown probabilities w.h.p the regret is at most $O(\log^2 T)$.

Don't have true p: estimate entire past leads to difficult analysis. We couldn't solve it.

Makespan: Unknown probabilities

Theorem

For unknown probabilities w.h.p the regret is at most $O(\log^2 T)$.

Don't have true p: estimate entire past leads to difficult analysis. We couldn't solve it.

Instead we divide the time horizon to blocks and each block to phases.

Partition T to $\log(T/2)$ blocks, where the r-th block, B^r, has 2^r time steps.

- Set reference $w^{r,*}(i)$ using the observed probabilities in block B^{r-1} as follows.
- In block B^r we have $m = r$ phases, where the duration of phase k is $T^{r,k} = |B^r|/2^k = 2^{r-k}$. (Decreasing phase lengths.)

Not known if tight.

Conclusions and open problems

- Stochastic vs adversarial model: different rates. (Not really surprising.)
- Information model is specific - other information models are possible Next COLT?
- Calibration without calibrating.

Conclusions and open problems

- Stochastic vs adversarial model: different rates. (Not really surprising.)
- Information model is specific - other information models are possible Next COLT?
- Calibration without calibrating.

Still open:

- Lower bounds Looks really hard
- For which other global functions no regret is possible?
- Relaxed goals for global functions.

Thank you!