

Learning piecewise Lipschitz fns in changing environments

Mar 3, 2020

Joint work with: N Balcan, T Dick

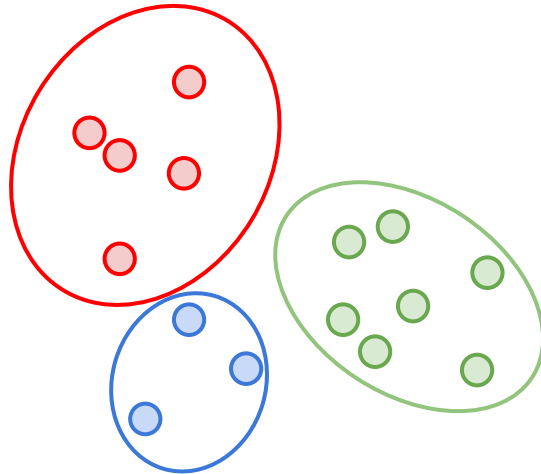


Presented by: Dravy Sharma
dravyans@andrew.cmu.edu
Grad Student, CSD, CMU

Motivation

Data-driven algorithm selection:

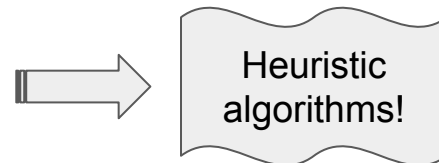
- Think of **hard** combinatorial problems
E.g. clustering, integer programming, subset selection



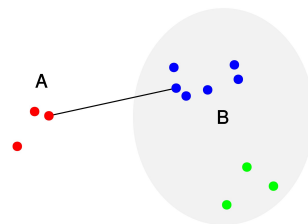
Motivation

Data-driven algorithm selection:

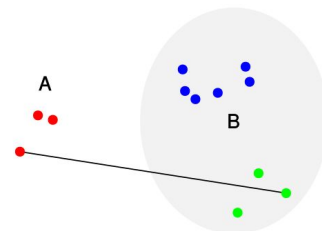
- Think of hard combinatorial problems
E.g. **clustering**, integer programming, subset selection



Single linkage



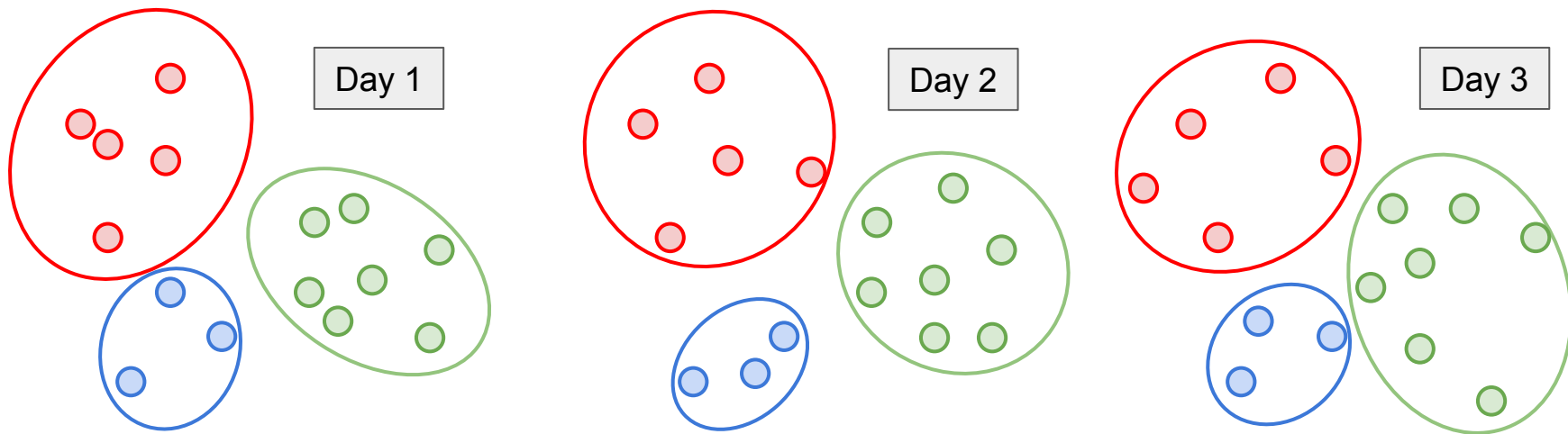
Complete linkage



Motivation

Data-driven algorithm selection:

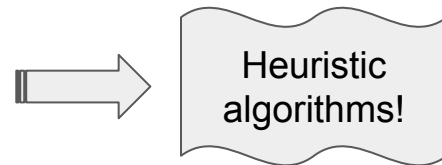
- Think of hard combinatorial problems
E.g. clustering, integer programming, subset selection
- Suppose you have to repeatedly solve instances
(drawn from unknown distribution)



Motivation

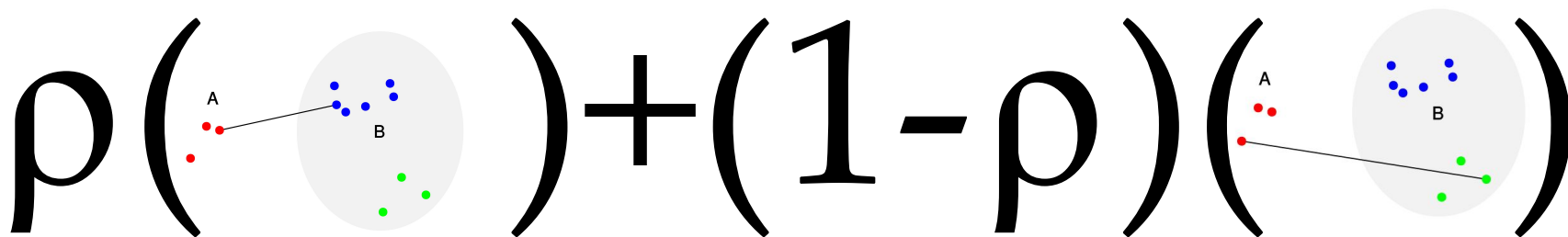
Data-driven algorithm selection:

- Think of hard combinatorial problems
E.g. clustering, integer programming, subset selection
- Suppose you have to repeatedly solve instances
(drawn from unknown distribution)



Interpolate between heuristics with parameter ρ

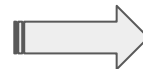
Learn data-specific optimal ρ

$$\rho \left(\begin{array}{c} \text{A} \\ \text{B} \end{array} \right) + (1 - \rho) \left(\begin{array}{c} \text{A} \\ \text{B} \end{array} \right)$$


Motivation

Data-driven algorithm selection:

- Think of hard combinatorial problems
E.g. clustering, integer programming, subset selection
- Suppose you have to repeatedly solve instances
(drawn from unknown distribution)



Heuristic
algorithms!



Interpolate between heuristics with parameter ρ
Learn data-specific optimal ρ



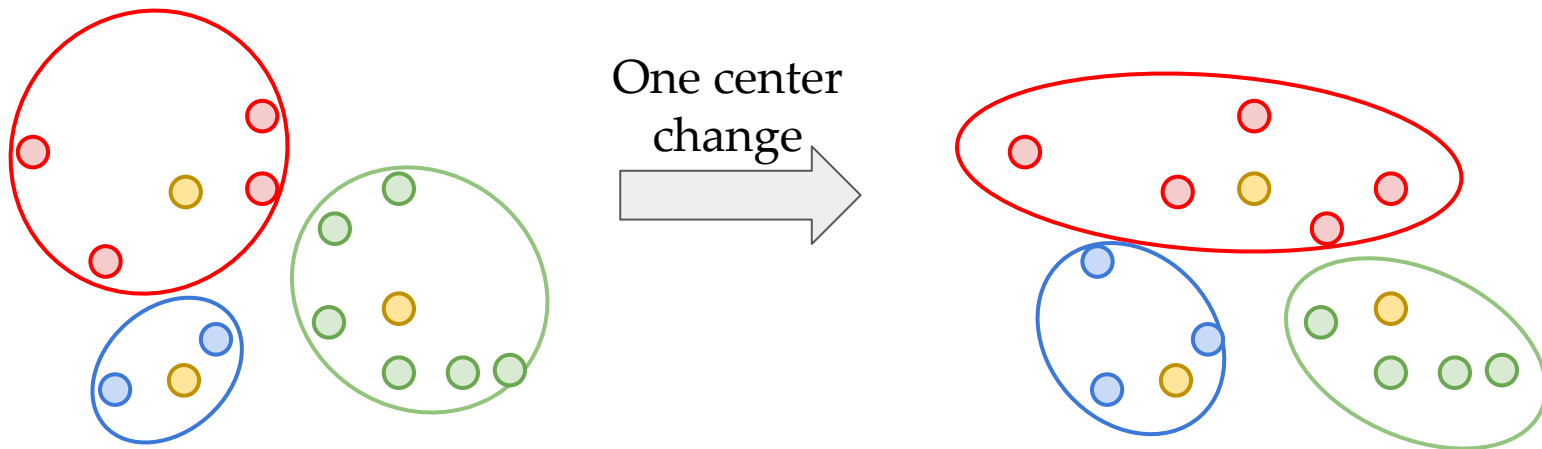
ML algorithms are often algo families
(d hyperparameters $\Rightarrow \rho \in \mathbb{R}^d$)

Motivation

How does algorithm payoff change with parameter ρ ?

- Can have sharp discontinuities!

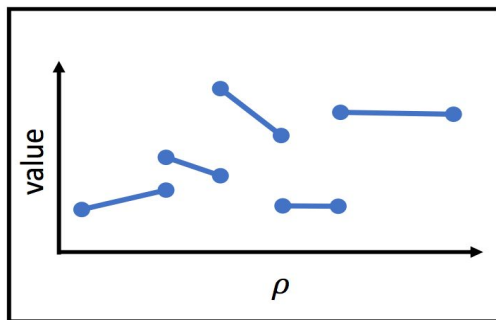
E.g. Choosing different initial centers in clustering can cascade into very different results



Motivation

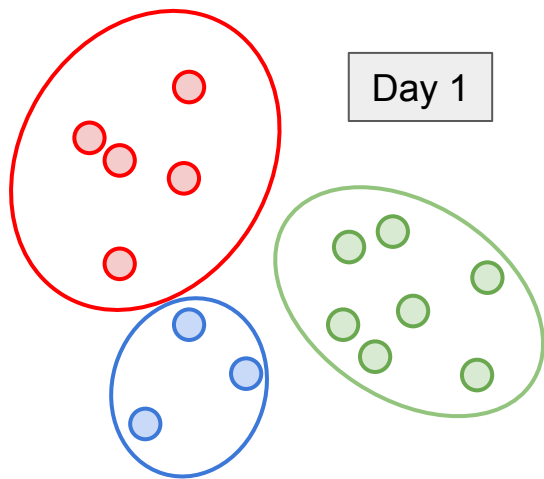
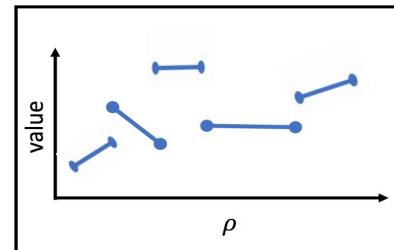
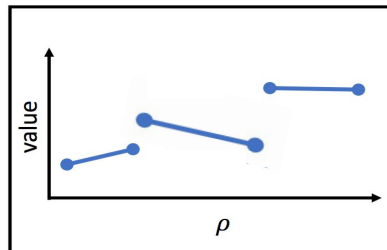
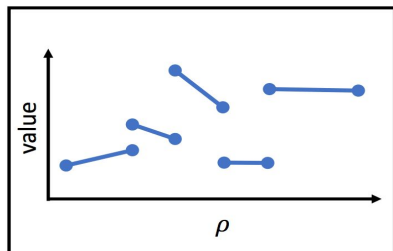
How does algorithm payoff change with parameter ρ ?

- Can have sharp discontinuities!
E.g. Choosing different initial centers in clustering can cascade into very different results
- Typically piecewise Lipschitz (discontinuous, but each piece has bounded slope)

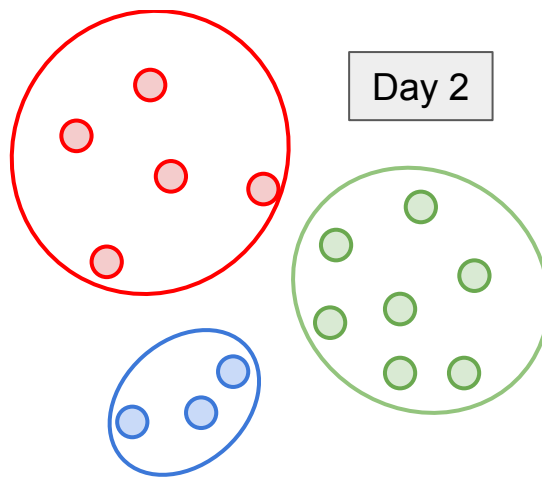


Motivation

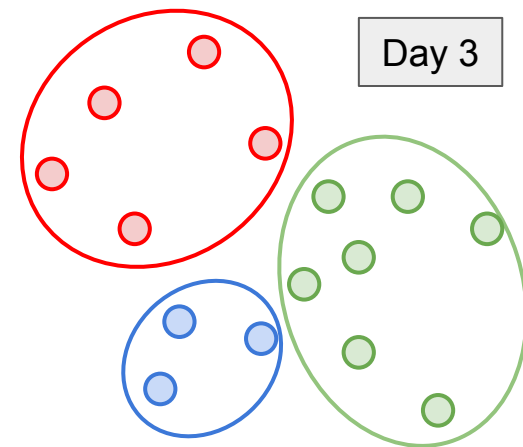
Choosing optimal ρ = online learning of piecewise Lipschitz fns



Day 1



Day 2



Day 3

Motivation

But why online?

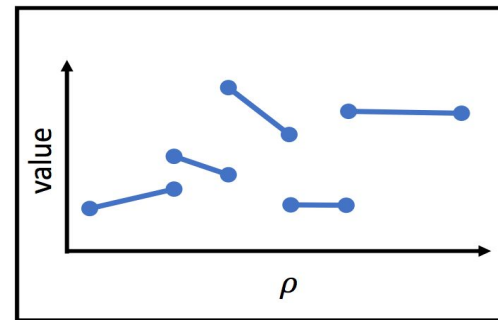
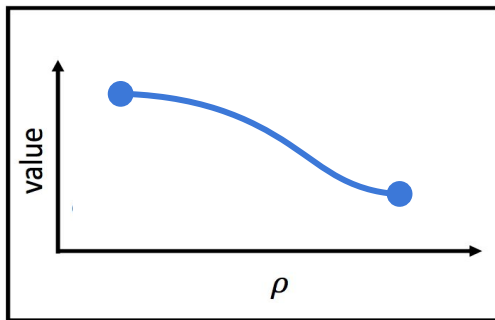
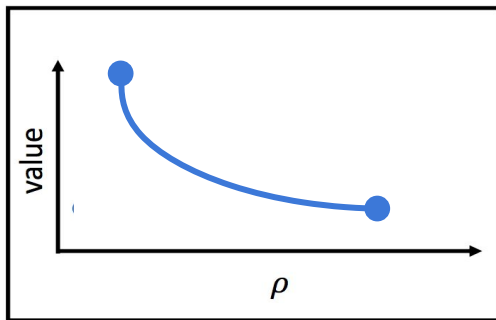
- Infeasible to use all data (computationally or otherwise)
- Dynamically adapt to new data patterns (e.g. changing user base)



Motivation (theoretical)

Also generalizes previously known studies in online learning:

- OCO - Online convex optimization
- Optimization of nonconvex but Lipschitz functions

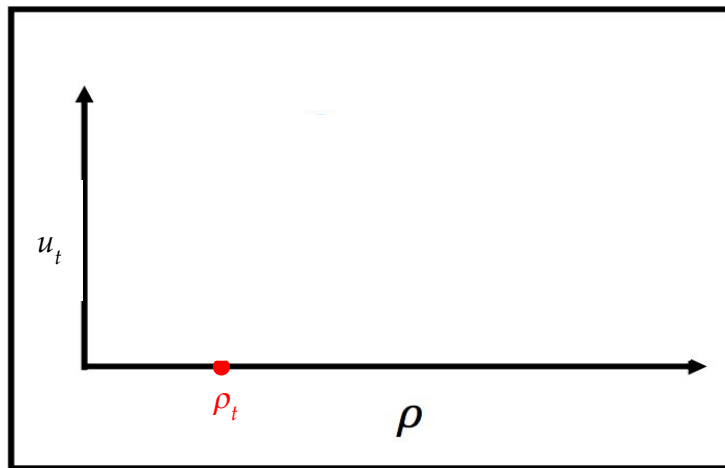


Online learning

- At each time $t = 1 \dots T$

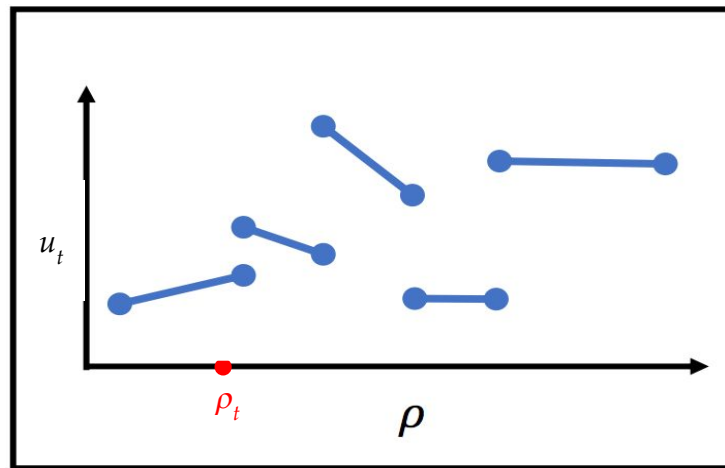
Online learning

- At each time $t = 1 \dots T$:
 - We need to pick a point ρ_t in domain



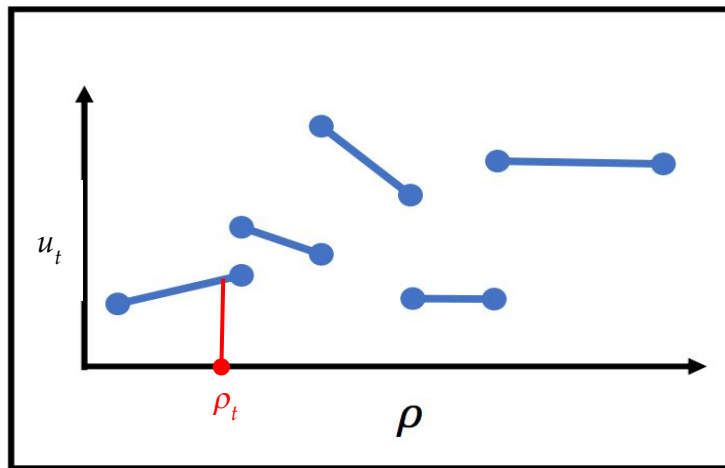
Online learning

- At each time $t = 1 \dots T$:
 - We need to pick a point ρ_t in domain
 - Payoff function $u_t(\cdot)$ is revealed



Online learning

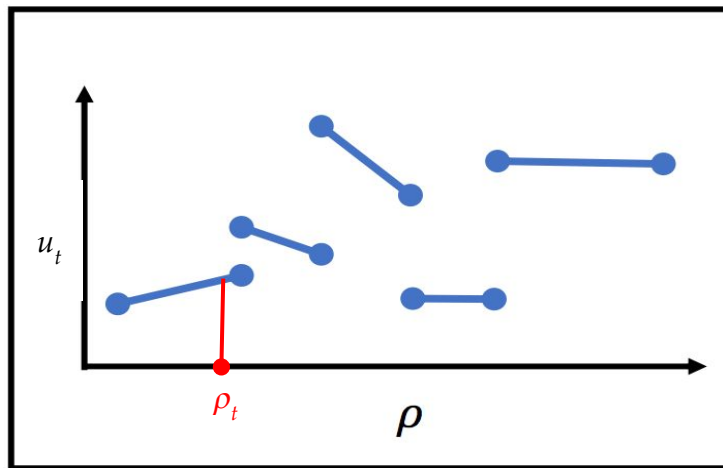
- At each time $t = 1 \dots T$:
 - We need to pick a point ρ_t in domain
 - Payoff function $u_t(\cdot)$ is revealed
 - We experience payoff $u_t(\rho_t)$



Online learning

- At each time $t = 1 \dots T$:
 - We need to pick a point ρ_t in domain
 - Payoff function $u_t(\cdot)$ is revealed
 - We experience payoff $u_t(\rho_t)$

Goal:
Maximize $\sum_t u_t(\rho_t)$



Online learning

- **Regret:** compares performance of an online algorithm with a somewhat more constrained *offline* optimal algorithm.



Online learning

- **Regret:** compares performance of an online algorithm with a somewhat more constrained *offline* optimal algorithm.
- Standard/static regret: Performance relative to best fixed point *in hindsight*

$$\mathbb{E} \left[\max_{\rho^* \in \mathcal{C}} \sum_{i=1}^T (u_t(\rho^*) - u_t(\rho_t)) \right]$$



Online learning

- **Regret:** compares performance of an online algorithm with a somewhat more constrained *offline* optimal algorithm.
- Standard/static regret: Performance relative to best fixed point *in hindsight*

$$\mathbb{E} \left[\max_{\rho^* \in \mathcal{C}} \sum_{i=1}^T (u_t(\rho^*) - u_t(\rho_t)) \right]$$



If regret is sublinear, average regret = $o(T)/T \rightarrow 0$ as T increases

Online learning

- **Regret:** compares performance of an online algorithm with a somewhat more constrained *offline* optimal algorithm.
- Standard/static regret: Performance relative to best fixed point *in hindsight*

$$\mathbb{E} \left[\max_{\rho^* \in \mathcal{C}} \sum_{i=1}^T (u_t(\rho^*) - u_t(\rho_t)) \right]$$

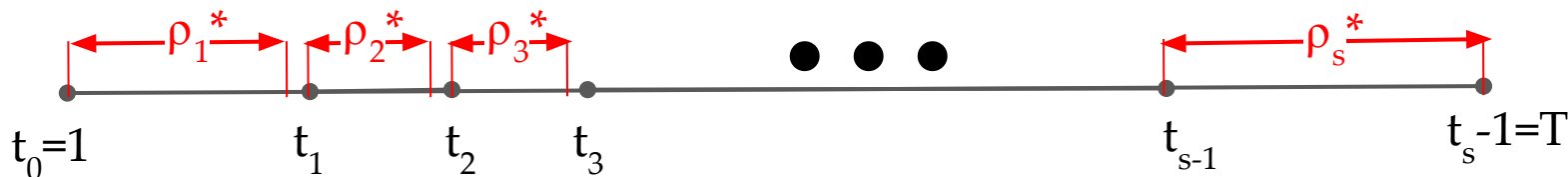
- Not suitable for changing environments!



Online learning

- '**s-shifted regret**' compares performance against offline algorithm which can use up to s experts by switching $s - 1$ times. [Herbster, Warmuth '98]

$$\mathbb{E} \left[\max_{\substack{\rho_i^* \in \mathcal{C}, \\ t_0=1 < t_1 < \dots < t_s=T+1}} \sum_{i=1}^s \sum_{t=t_{i-1}}^{t_i-1} (u_t(\rho_i^*) - u_t(\rho_t)) \right]$$



A mean adversary

- Lower bound for arbitrary piecewise Lipschitz functions: sublinear regret is impossible for any algorithm even for $s = 1$!

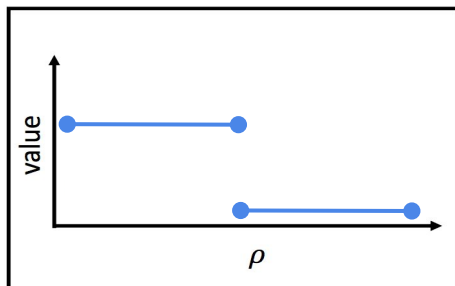


A mean adversary

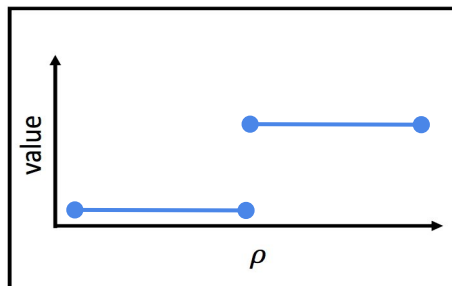
- Lower bound for arbitrary piecewise Lipschitz functions: sublinear regret is impossible for any algorithm even for $s = 1$!

Halving Adversary

$t = 1$



OR



*with
probability
 $\frac{1}{2}$ each*

Regret
=
 $\frac{1}{2}$

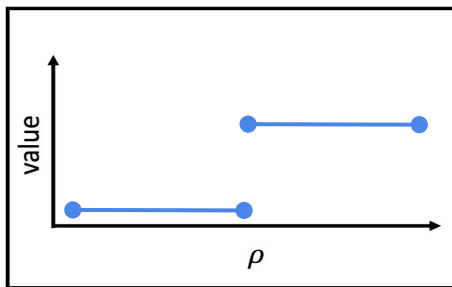


A mean adversary

- Lower bound for arbitrary piecewise Lipschitz functions: sublinear regret is impossible for any algorithm even for $s = 1$!

Halving Adversary

$t = 1$



Regret
=
 $\frac{1}{2}$

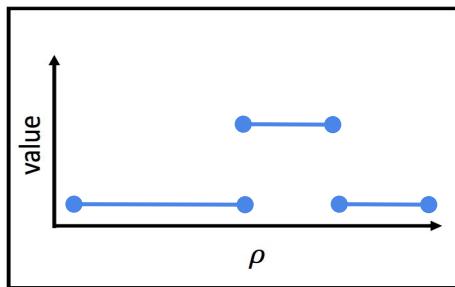


A mean adversary

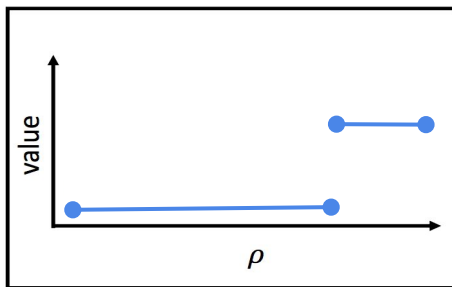
- Lower bound for arbitrary piecewise Lipschitz functions: sublinear regret is impossible for any algorithm even for $s = 1$!

Halving Adversary

$t = 2$



OR



*with
probability
 $\frac{1}{2}$ each*

Regret
=
 $\frac{1}{2}$

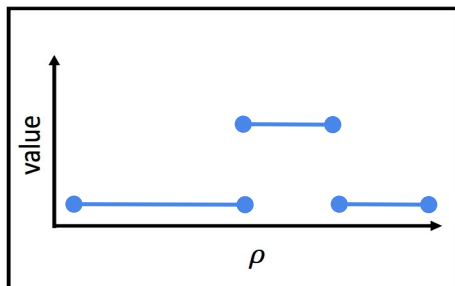


A mean adversary

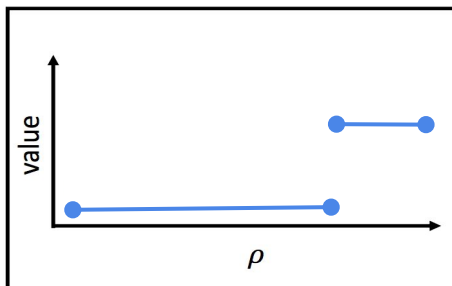
- Lower bound for arbitrary piecewise Lipschitz functions: sublinear regret is impossible for any algorithm even for $s = 1$!

Halving Adversary

$t = 2$



OR



*with
probability
 $\frac{1}{2}$ each*

Regret
=
 $\frac{1}{2}$

and so on ...



A mean adversary

- Adding up regret across all rounds, we get linear regret. Offline OPT can get all rounds right, we only get half in expectation.

Halving Adversary

$$\begin{aligned} \text{Regret} \\ = \\ T/2 \end{aligned}$$



Dispersion

- Is it ever possible to learn piecewise Lipschitz functions? Is so, when?

Turns out *dispersion* is necessary and sufficient!

- β -dispersed: if for all T and for all $\epsilon \leq T^{-\beta}$

$$\mathbb{E} \left[\max_{\rho \in \mathcal{C}} |\{1 \leq t \leq T \mid u_t \text{ is not } L\text{-Lipschitz in } \mathcal{B}(\rho, \epsilon)\}| \right] \leq \tilde{O}(\epsilon T)$$

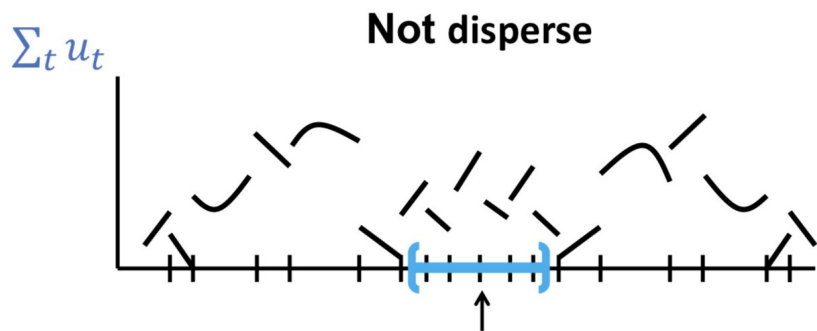


Intuitively, concentration of discontinuities in space, when averaged over time is bad for learning!

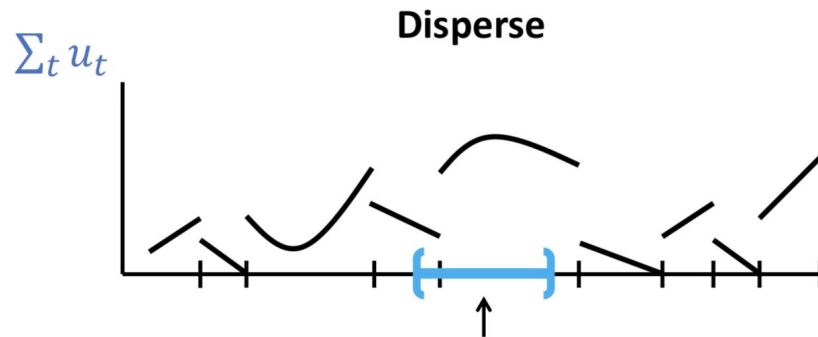
Dispersion

- β -dispersed: if for all T and for all $\epsilon \leq T^{-\beta}$

$$\mathbb{E} \left[\max_{\rho \in \mathcal{C}} |\{1 \leq t \leq T \mid u_t \text{ is not } L\text{-Lipschitz in } \mathcal{B}(\rho, \epsilon)\}| \right] \leq \tilde{O}(\epsilon T)$$



Many boundaries within interval



Few boundaries within interval

Main results

- Upper bound on ‘s-shifted regret’: There exists an efficient algorithm with regret bounded by

$$O(\sqrt{sdT \log T} + sT^{1-\beta})$$

- Lower bound
 - Matching modulo root-log(T) factor

For each $\beta > \frac{\log 3s}{\log T}$, there exist utility functions $u_1, \dots, u_T : [0, 1] \rightarrow [0, 1]$ which are β -dispersed, and regret of any online algorithm is $\Omega(\sqrt{sT} + sT^{1-\beta})$.

Dispersion gives a tight characterization!

Algorithm [Balcan Dick Vitercik, FOCS'18]

Algorithm

Exponential Forecaster

1. $w_1(\rho) = 1$ for all $\rho \in \mathcal{C}$
2. For each $t = 1, 2, \dots, T$:
 - i. $W_t := \int_{\mathcal{C}} w_t(\rho) d\rho$
 - ii. Sample ρ with probability proportional to $w_t(\rho)$, i.e. with probability $p_t(\rho) = \frac{w_t(\rho)}{W_t}$
 - iii. Update weights

$$w_{t+1}(\rho) = e^{\lambda u_t(\rho)} w_t(\rho)$$

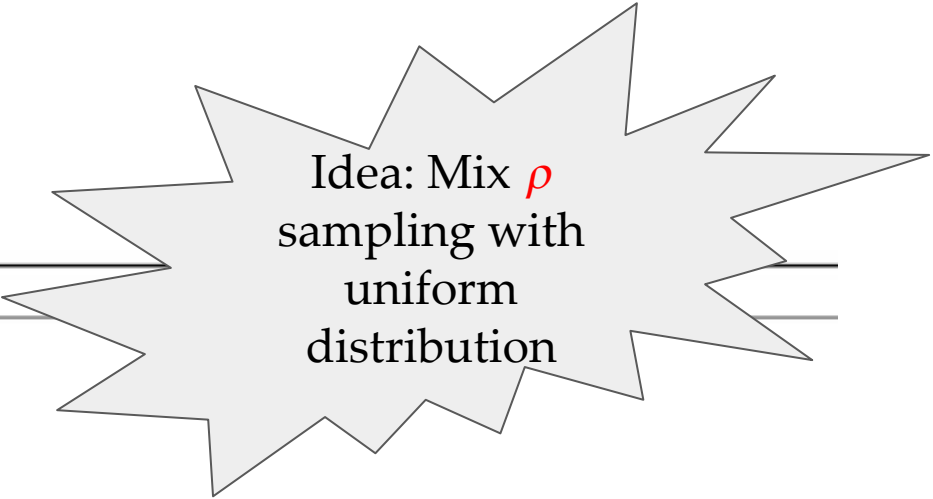
Good regular regret but can have bad s -shifted regret!

Algorithm

Algorithm **Fixed Share** Exponential Forecaster

1. $w_1(\rho) = 1$ for all $\rho \in \mathcal{C}$
2. For each $t = 1, 2, \dots, T$:
 - i. $W_t := \int_{\mathcal{C}} w_t(\rho) d\rho$
 - ii. Sample ρ with probability proportional to $w_t(\rho)$, i.e. with probability $p_t(\rho) = \frac{w_t(\rho)}{W_t}$
 - iii. Update weights

$$w_{t+1}(\rho) = (1 - \alpha) e^{\lambda u_t(\rho)} w_t(\rho) + \alpha Z_t$$



Idea: Mix ρ
sampling with
uniform
distribution

Good s -shifted regret!

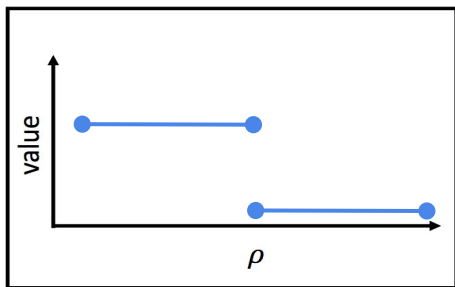
$$Z_t = \frac{\int_{\mathcal{C}} e^{\lambda u_t(\rho)} w_t(\rho) d\rho}{\text{VOL}(\mathcal{C})}$$

$$\alpha = \frac{s-1}{T-1}$$

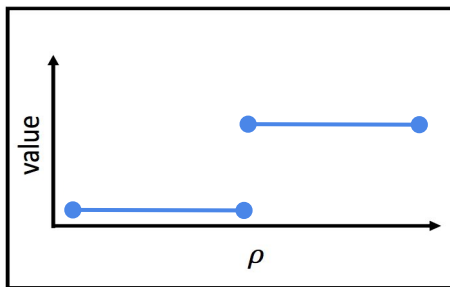
$$\lambda = \sqrt{s(d \log(RT^\beta) + \log(T/s)) / T/H}$$

Algorithm

- Mixing with uniform allows faster adaptation to changing 'best expert' (explore vs exploit!)



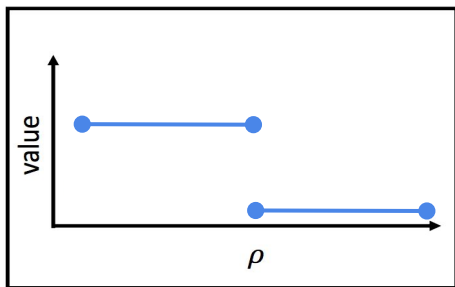
$t = 1 \dots T/2$



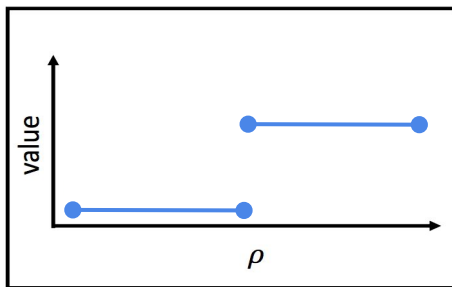
$t = T/2 \dots T$

Algorithm

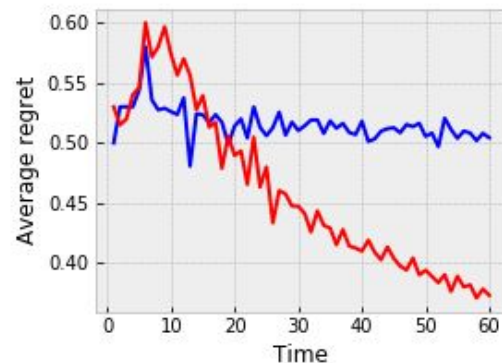
- Mixing with uniform allows faster adaptation to changing 'best expert'
(explore vs exploit!)



$t = 1 \dots T/2$



$t = T/2 \dots T$



Exponential Forecaster, Fixed Share EF

je ne regrette rien

Algorithm : Why it works?

- Ensures that the optimal solution, and its neighborhood, in hindsight have a large total density
- Achieve this by carefully setting the parameters, in particular the *exploration parameter* α which controls the rate at which we allow our confidence on 'good' experts to change
- Lipschitzness and dispersion are then used to ensure that solutions sufficiently close to the optimum are also good on average

Gets us the upper bound on regret: $O(\sqrt{sdT \log T} + sT^{1-\beta})$

Algorithm

- Implementation is tricky!
 - How to maintain weights for uncountably infinite points?
- Efficient implementation in continuous setting (infinite experts) with same asymptotic regret
 - Tricks:
 - Use approximate weights
 - Sample approximately without computing p_t explicitly
 - Use logconcave sampling and integration algorithms
 - Approximate weights and sampling in multi-dimensional case for piecewise concave utility functions - $O(\text{poly}(d, T))$
 - 1-D piecewise constant functions - $O(\log T)$ updates
- Polynomial time algorithm with same asymptotic expected regret!

Recurring environments

- s environment shifts may rotate among $m < s$ environments
- ‘ m -sparse, s -shifted’ regret to capture this, can we do better when $m \ll s$


$$\mathbb{E} \left[\max_{\substack{\rho_i^* \in \mathcal{C}, \\ t_0=1 < t_1 < \dots < t_s = T+1, \\ |\{\rho_i^* | 1 \leq i \leq s\}| \leq m}} \sum_{i=1}^s \sum_{t=t_{i-1}}^{t_i-1} (u_t(\rho_i^*) - u_t(\rho_t)) \right]$$



Mix with the past distributions!

Algorithm

' m -sparse, s -shifted' regret



Idea: Mix ρ
with all
previous
distributions

Algorithm Generalized Share Exponential Forecaster $[\alpha, \gamma]$

1. $w_1(\rho) = 1$ for all $\rho \in \mathcal{C}$
2. For each $t = 1, 2, \dots, T$:
 - i. $W_t := \int_{\mathcal{C}} w_t(\rho) d\rho$
 - ii. Sample ρ with probability proportional to $w_t(\rho)$, i.e. with probability $p_t(\rho) = \frac{w_t(\rho)}{W_t}$
 - iii. Update weights

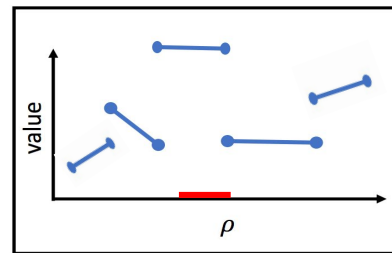
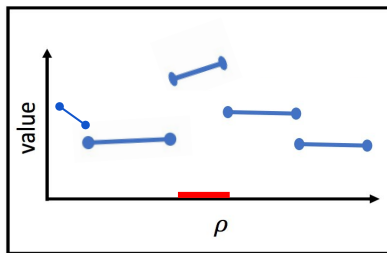
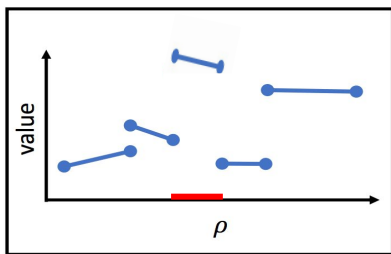
$$w_{t+1}(\rho) = (1 - \alpha)e^{\lambda u_t(\rho)} w_t(\rho) + \alpha \left(\int_{\mathcal{C}} e^{\lambda u_t(\rho)} w_t(\rho) d\rho \right) \sum_{i=1}^t \beta_{i,t} p_i(\rho)$$

where $\beta_{i,t} = \frac{e^{-\gamma(t-i)}}{\sum_{j=1}^t e^{-\gamma(t-j)}}$ $\lambda = \sqrt{(md \log(RT^\beta) + s \log(T/s))/T/H}$, $\alpha = s/T$ and $\gamma = s/mT$

$$R_T \leq O(H \sqrt{T(md \log(RT^\beta) + s \log(mT/s))} + (mH + L)T^{1-\beta})$$

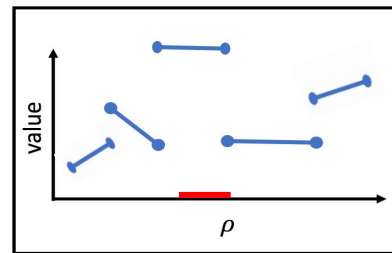
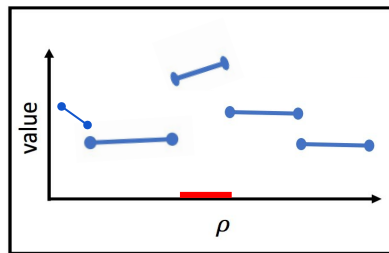
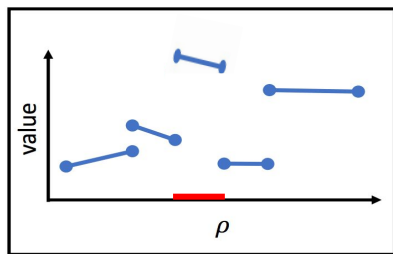
Algorithm

- Generalized Share helps in recurring environments
 - As long as the optimal region recurs, it can give better performance



Algorithm

- Generalized Share helps in recurring environments
 - As long as the optimal region recurs, it can give better performance

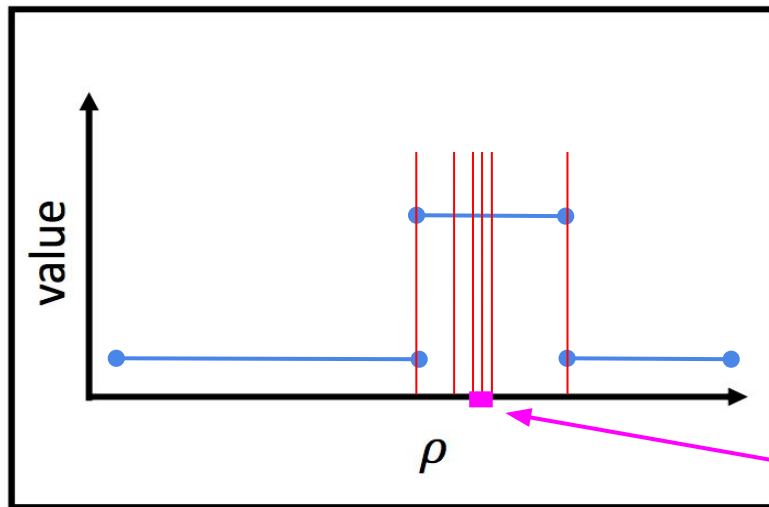


Regret bound:

$$O(\sqrt{\underline{s}dT \log T} + \underline{s}T^{1-\beta}) \quad \Rightarrow \quad O(\sqrt{(\underline{m}d + \underline{s})T \log T} + \underline{m}T^{1-\beta})$$

Lower bound

- Our 'mean adversary' doesn't work as it is not dispersed



Too concentrated!

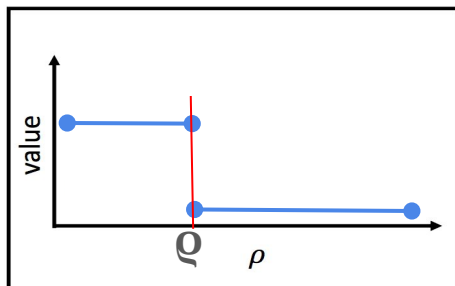
Lower bound

- Need a new, smarter way which ensures dispersion

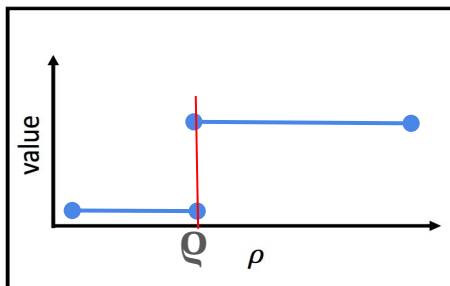


Lower bound

- Need a new, smarter way which ensures dispersion
 - Ingredient #1 $A(\varrho)$



OR

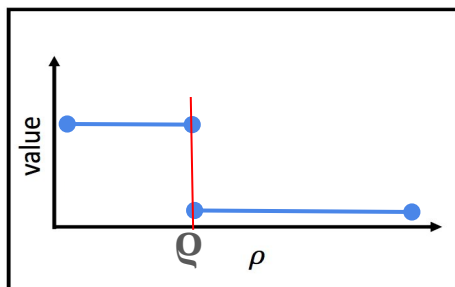


*with
probability
 $\frac{1}{2}$ each*

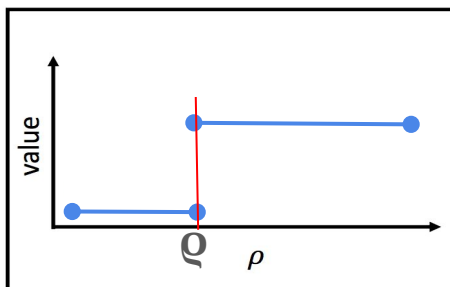
$t = 1 \dots T$

Lower bound

- Need a new, smarter way which ensures dispersion
 - Ingredient #1 $A(\varrho)$



OR



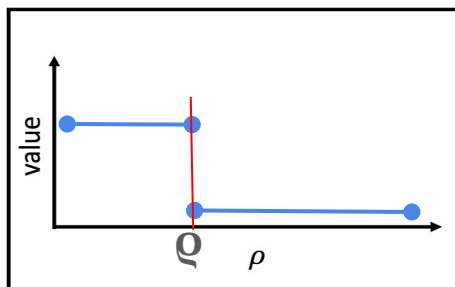
*with
probability
 $\frac{1}{2}$ each*

$t = 1 \dots T$

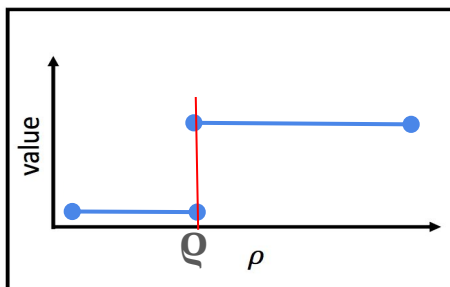
$$\begin{aligned} \text{Regret} &= E[\text{OPT} - \text{Any}] \\ &= \Omega(\sqrt{T}) \end{aligned}$$

Lower bound

- Need a new, smarter way which ensures dispersion
 - Ingredient #1 $A(\varrho)$



OR



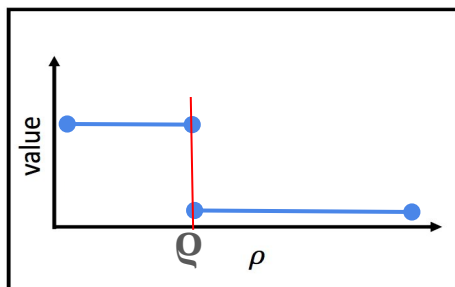
*with
probability
 $\frac{1}{2}$ each*

$t = 1 \dots T$

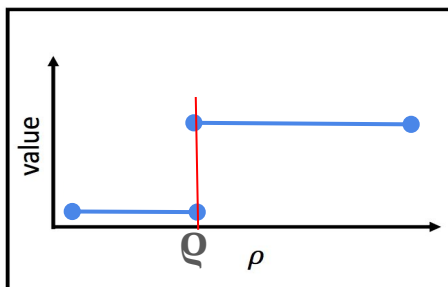
s-shifting Regret =
 $E[\text{OPT} - \text{Any}] = \Omega(\sqrt{sT})$

Lower bound

- Need a new, smarter way which ensures dispersion
 - Ingredient #1 $A(\varrho)$



OR



*with
probability
1/2 each*

**But still
NOT
dispersed!**

$t = 1 \dots T$

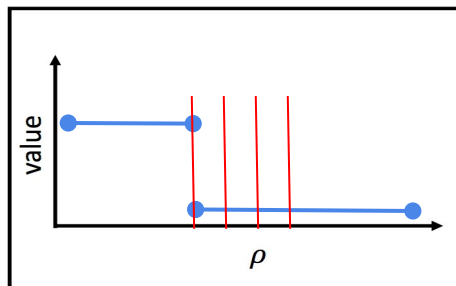
$$\text{s-shifting Regret} = \mathbb{E}[\text{OPT} - \text{Any}] = \Omega(\sqrt{sT})$$

Lower bound



- Need a new, smarter way which ensures dispersion
 - Ingredient #2: s phases with T/s fns each

Each phase has two parts:



First: Dispersed $A(\mathbf{q})$'s
in the center

$$t = 1 \dots T/s - T^{1-\beta}$$

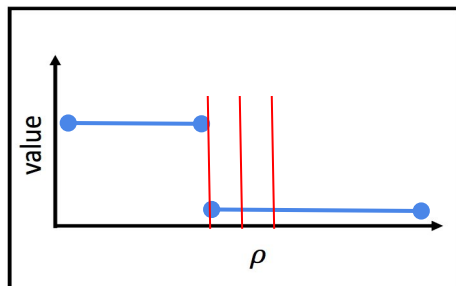
$$\text{Regret} = \Omega(\sqrt{T/s})$$

Lower bound



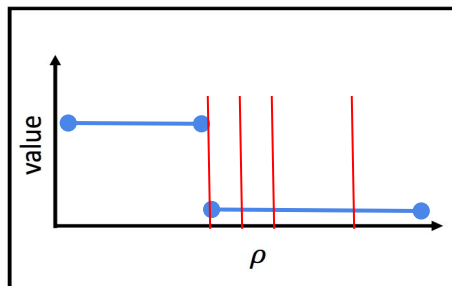
- Need a new, smarter way which ensures dispersion
 - Ingredient #2: s phases with T/s fns each

Each phase has two parts:



$$t = 1 \dots T/s - T^{1-\beta}$$

$$\text{Regret} = \Omega(\sqrt{T/s})$$



$$t = T/s - T^{1-\beta} \dots T/s$$

$$\text{Regret} = \Omega(T^{1-\beta})$$

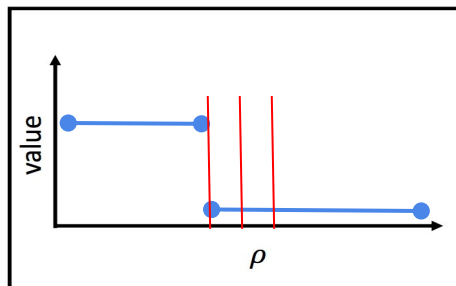


Second: 'Mean'
Halving Adversary

Lower bound

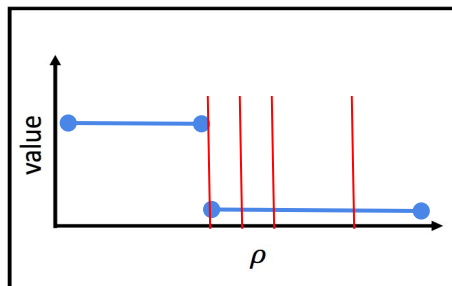
- Need a new, smarter way which ensures dispersion
 - Ingredient #2: s phases with T/s fns each

Each phase has two parts:



$$t = 1 \dots T/s - T^{1-\beta}$$

$$\text{Regret} = \Omega(\sqrt{T/s})$$



$$t = T/s - T^{1-\beta} \dots T/s$$

$$\text{Regret} = \Omega(T^{1-\beta})$$



Successive phases in largest 'unused' interval

$$\begin{aligned} \text{Total Regret} \\ = \\ \Omega(\sqrt{sT} + sT^{1-\beta}) \end{aligned}$$



Experiments

- α -Lloyd clustering [Balcan Dick White, NeurIPS'18]:
 - Way to initialize k-means centers
 - Pick successive centers randomly with probability proportional to d^α
 - Interpolates between random sampling ($\alpha = 0$), k-means++ ($\alpha = 2$) and farthest first traversal ($\alpha = \infty$)

Experiments

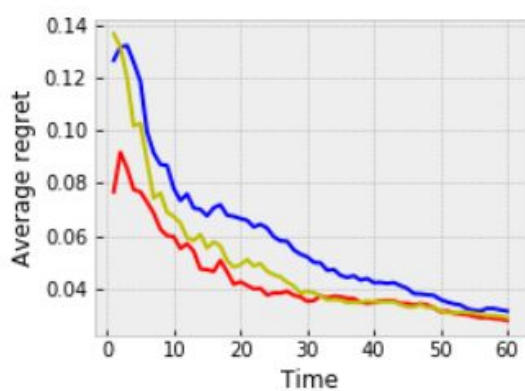
- α -Lloyd clustering [Balcan Dick White, NeurIPS'18]:
 - Way to initialize k-means centers
 - Pick successive centers randomly with probability proportional to d^α
 - Interpolates between random sampling ($\alpha = 0$), k-means++ ($\alpha = 2$) and farthest first traversal ($\alpha = \infty$)
- Quality of clusters is a piecewise constant function of α with potentially sharp changes cascading from initial choice of centers
 - Can we learn data-specific good α ?

Experiments

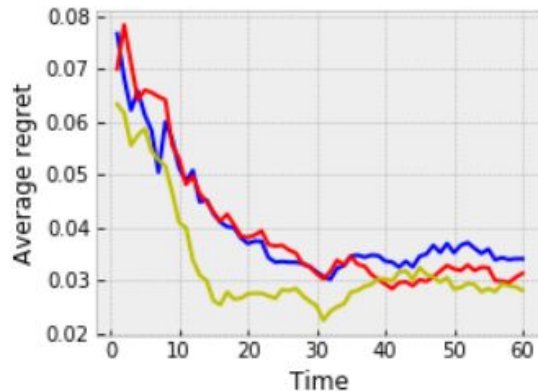
- For example, considering clustering the images of digits in MNIST.
 - $\rho = \alpha \in [0, 10]$
 - $u_t(\rho)$ = Hamming cost of clustering produced by α -Lloyd clustering
- How to simulate changing distributions?
 - We sample from different subsets of digits at different times
 - E.g. even digits for $t = 1 \dots T/2$ and odd digits for $t = T/2+1 \dots T$

Experiments

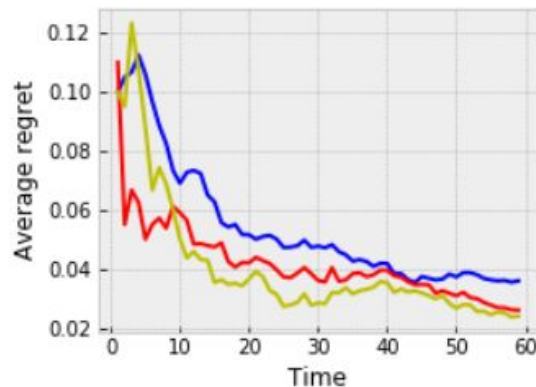
- Average 2-shifted regret for α -Lloyd clustering ($k = 2$):
 - Half the classes till $T/2$ and other half for other half.



(a) MNIST



(b) Omniglot_small_1

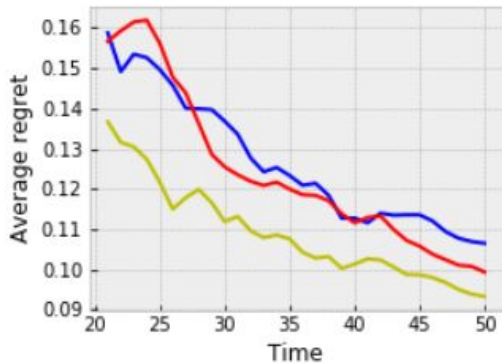


(c) Omniglot (full)

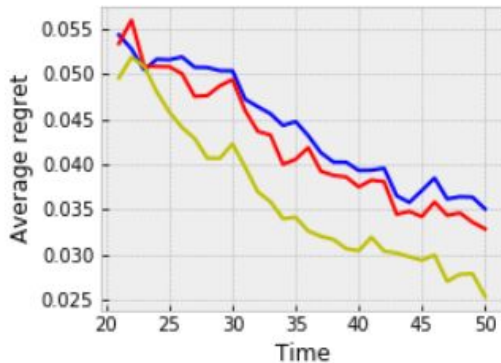
Figure 1: Average 2-shifted regret vs game duration T for online clustering against 2-shifted distributions. Color scheme: **Exponential Forecaster**, **Fixed Share EF**, **Generalized Share EF**

Experiments

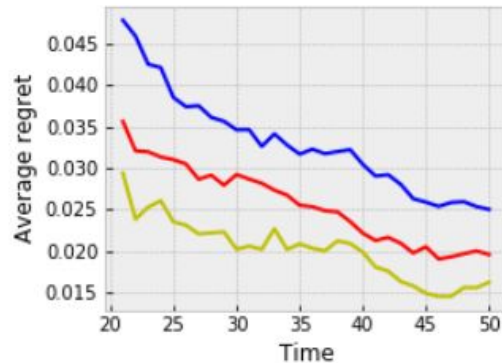
- Average 2-shifted regret for α -Lloyd clustering ($k = \# \text{classes}$):
 - All but one class presented in each phase.



(a) MNIST



(b) Omniglot_small_1

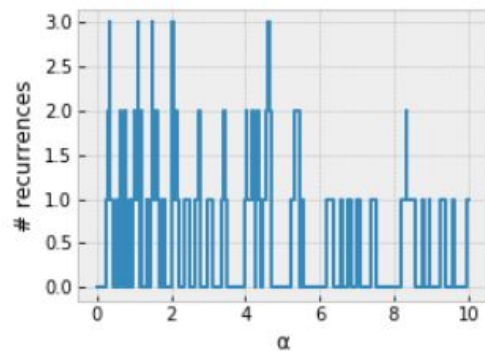


(c) Omniglot (full)

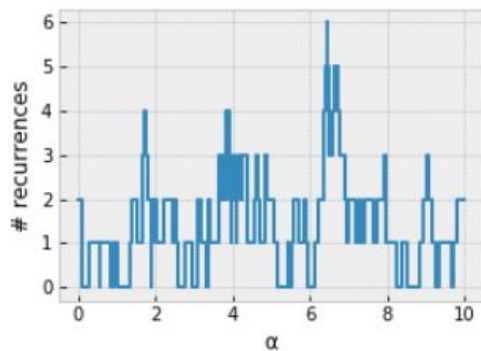
Figure 3: Average k -shifted regret vs game duration T for online clustering against k -shifted distributions. Color scheme: **Exponential Forecaster**, **Fixed Share EF**, **Generalized Share EF**

Experiments

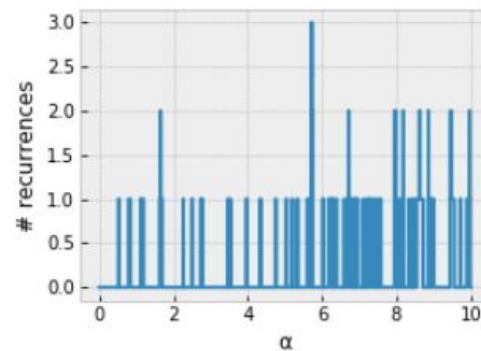
- Fixed Share vs Generalized Share: How to decide?



(a) MNIST



(b) Omniglot_small_1

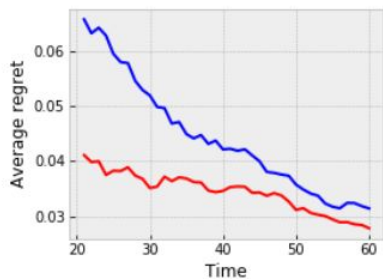


(c) Omniglot (full)

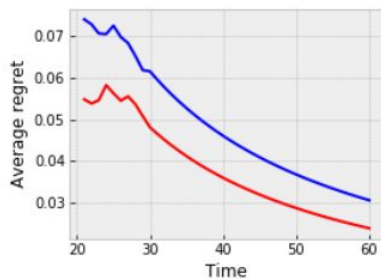
Figure 4: Number of recurrences of various values of α in the top decile across all rounds

Experiments

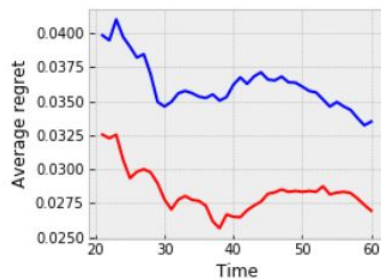
- Same dataset (MNIST), different clustering classes



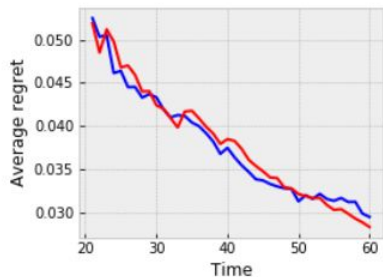
(a) {0, 2, 4, 6, 8}



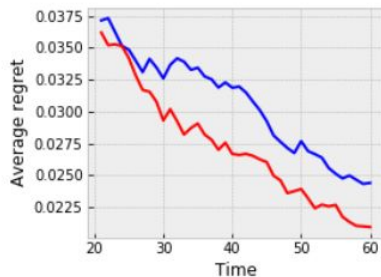
(b) {0, 1, 2, 3, 4}



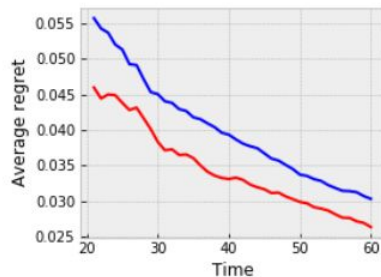
(c) {2, 3, 5, 6, 9}



(d) {1, 3, 4, 8, 9}



(e) {0, 4, 5, 7, 8}



(f) Average

Takeaways

- Hard problems have heuristics, no one heuristic may dominate all others.
- If you have to solve a problem several times, it pays to interpolate heuristics and learn data-specific algorithm/parameters.
- It's possible to adapt to sudden, sharp changes -- provided the data/distribution is 'nice' enough.
- Careful balance of exploring/revising and exploiting/reusing is key.
- It may be impossible to learn if data is not 'nice', so probably worth knowing if there is no hope.

Thank you!

QUESTIONS?