Trading regret rate for computational efficiency in online learning with limited feedback

Shai Shalev-Shwartz

TTI-C  Hebrew University

On-line Learning with Limited Feedback Workshop, 2009

June 2009
Main Question

Given a runtime constraint $\tau$, horizon $T$, reference class $\mathcal{H}$: What is the achievable regret of an algorithm whose (amortized) runtime is $O(\tau)$?
Main Question

Given a runtime constraint $\tau$, horizon $T$, reference class $\mathcal{H}$: What is the achievable regret of an algorithm whose (amortized) runtime is $O(\tau)$?
Main Question

Given a runtime constraint $\tau$, horizon $T$, reference class $\mathcal{H}$: What is the achievable regret of an algorithm whose (amortized) runtime is $O(\tau)$?
The prediction problem

Arms: $A = \{1, \ldots, k\}$

For $t = 1, 2, \ldots, T$

- Learner receives side information $x_t \in \mathcal{X}$
- Environment chooses cost vector $c_t : A \rightarrow [0, 1]$ (unknown to learner)
- Learner chooses action $a_t \in A$
- Learner pay cost $c_t(a_t)$
Non-stochastic Multi-armed bandit with side information

The prediction problem

Arms: \( A = \{1, \ldots, k\} \)

For \( t = 1, 2, \ldots, T \)
  - Learner receives side information \( x_t \in \mathcal{X} \)
  - Environment chooses cost vector \( c_t : A \to [0, 1] \) (unknown to learner)
  - Learner chooses action \( a_t \in A \)
  - Learner pay cost \( c_t(a_t) \)

Goal

Low regret w.r.t. a reference hypothesis class \( \mathcal{H} \):

\[
\text{Regret} \overset{\text{def}}{=} \sum_{t=1}^{T} c_t(a_t) - \min_{h \in \mathcal{H}} \sum_{t=1}^{T} c_t(h(x_t))
\]
$\mathcal{H}$ is the class of linear hypotheses

- **Multiclass, separable**
  - Regret vs. $\tau$
  - Banditron
  - Halving

- **2 arms, non-separable**
  - Regret vs. $\tau$
  - IDPK
  - EXP4
First example: Bandit Multi-class Categorization

For $t = 1, 2, \ldots, T$

- Learner receives side information $x_t \in \mathcal{X}$
- Environment chooses 'correct' arm $y_t \in A$ (unknown to learner)
- Learner chooses action $a_t \in A$
- Learner pay cost $c_t(a_t) = 1[a_t \neq y_t]$
\[ \mathcal{H} = \{ x \mapsto \arg\max_r (W x)_r : W \in \mathbb{R}^{k,d}, \|W\|_F \leq 1 \} \]
Large margin assumption

Assumption: Data is separable with margin $\mu$:

$$\forall t, \forall r \neq y_t, (Wx_t)_{yt} - (Wx_t)_r \geq \mu$$
Halving for Bandit Multiclass categorization

Initialize: $V_1 = \mathcal{H}$
For $t = 1, 2, \ldots$

- Receive $x_t$
- For all $r \in [k]$ let $V_t(r) = \{h \in V_t : h(x_t) = r\}$
- Predict $\hat{y}_t \in \arg \max_r |V_t(r)|$
- If $1[\hat{y}_t \neq y_t]$ set $V_{t+1} = V_t \setminus V_t(\hat{y}_t)$
First approach – Halving

Halving for Bandit Multiclass categorization

Initialize: $V_1 = \mathcal{H}$

For $t = 1, 2, \ldots$

- Receive $x_t$
- For all $r \in [k]$ let $V_t(r) = \{ h \in V_t : h(x_t) = r \}$
- Predict $\hat{y}_t \in \arg \max_r |V_t(r)|$
- If $1[\hat{y}_t \neq y_t]$ set $V_{t+1} = V_t \setminus V_t(\hat{y}_t)$

Analysis:

- Whenever we err $|V_{t+1}| \leq (1 - \frac{1}{k}) |V_t| \leq \exp(-1/k) |V_t|$
- Therefore: $M \leq k \log(|\mathcal{H}|)$
Using Halving

- Step 1: Dimensionality reduction to \( d' = \tilde{O}\left(\frac{1}{\mu^2}\right) \)
- Step 2: Discretize \( \mathcal{H} \) to \((1/\mu)^{d'}\) hypotheses
- Apply Halving on the resulting finite set of hypotheses
Using Halving

- Step 1: Dimensionality reduction to $d' = \tilde{O}\left(\frac{1}{\mu^2}\right)$
- Step 2: Discretize $\mathcal{H}$ to $(1/\mu)^{d'}$ hypotheses
- Apply Halving on the resulting finite set of hypotheses

Analysis:
- Mistake bound is $\tilde{O}\left(\frac{k}{\mu^2}\right)$
- But runtime grows like $(1/\mu)^{1/\mu^2}$
How can we improve runtime?

- Halving is not efficient because it does not utilize the structure of $\mathcal{H}$.
- In the full information case: Halving can be made efficient because each version space $V_t$ can be made convex!
- The Perceptron is a related approach which utilizes convexity and works in the full information case.
- Next approach: Let's try to rely on the Perceptron.
For \( t = 1, 2, \ldots, T \)

- Receive \( x_t \in \mathbb{R}^d \)
- Predict \( \hat{y}_t = \arg \max_r (W^t x_t)_r \)
- Receive \( y_t \)
- If \( \hat{y}_t \neq y_t \) update: \( W^{t+1} = W^t + U^t \)
The Multiclass Perceptron

For $t = 1, 2, \ldots, T$

- Receive $x_t \in \mathbb{R}^d$
- Predict $\hat{y}_t = \arg\max_r (W^t x_t)_r$
- Receive $y_t$
- If $\hat{y}_t \neq y_t$ update: $W^{t+1} = W^t + U^t$

$$U^t = \begin{bmatrix}
0 & \ldots & 0 \\
\vdots \\
0 & \ldots & 0 \\
\ldots & x_t & \ldots \\
0 & \ldots & 0 \\
\vdots \\
0 & \ldots & 0 \\
\ldots & -x_t & \ldots \\
0 & \ldots & 0 \\
\vdots \\
0 & \ldots & 0
\end{bmatrix}$$

Problem: In the bandit case, we’re blind to value of $y_t$
The Banditron (Kakade, S, Tewari 08)

- **Explore**: From time to time, instead of predicting $\hat{y}_t$ guess some $\tilde{y}_t$
- Suppose we get the feedback 'correct', i.e. $\tilde{y}_t = y_t$
- Then, we have full information for Perceptron's update: $(x_t, \hat{y}_t, \tilde{y}_t = y_t)$
The Banditron (Kakade, S, Tewari 08)

- **Explore:** From time to time, instead of predicting $\hat{y}_t$, guess some $\tilde{y}_t$
- Suppose we get the feedback 'correct', i.e. $\tilde{y}_t = y_t$
- Then, we have full information for Perceptron’s update: $(x_t, \hat{y}_t, \tilde{y}_t = y_t)$

**Exploration-Exploitation Tradeoff:**
- When exploring we may have $\tilde{y}_t = y_t \neq \hat{y}_t$ and can learn from this
- When exploring we may have $\tilde{y}_t \neq y_t = \hat{y}_t$ and then we had the right answer in our hands but didn’t exploit it
The Banditron (Kakade, S, Tewari 08)

For $t = 1, 2, \ldots, T$

- Receive $x_t \in \mathbb{R}^d$
- Set $\hat{y}_t = \arg\max_r (W^t x_t)_r$
- Define: $P(r) = (1 - \gamma)1[r = \hat{y}_t] + \frac{\gamma}{k}$
- Randomly sample $\tilde{y}_t$ according to $P$
- Predict $\tilde{y}_t$
- Receive feedback $1[\tilde{y}_t = y_t]$
- Update: $W^{t+1} = W^t + \tilde{U}^t$
The Banditron (Kakade, S, Tewari 08)

For $t = 1, 2, \ldots, T$

- Receive $x_t \in \mathbb{R}^d$
- Set $\hat{y}_t = \arg\max_r (W^t x_t)_r$
- Define: $P(r) = (1 - \gamma)1[r = \hat{y}_t] + \frac{\gamma}{k}$
- Randomly sample $\tilde{y}_t$ according to $P$
- Predict $\tilde{y}_t$
- Receive feedback $1[\tilde{y}_t = y_t]$
- Update: $W^{t+1} = W^t + \tilde{U}^t$ where

$$\tilde{U}^t = \begin{bmatrix} 0 & \cdots & 0 \\ \vdots \\ 0 & \cdots & 0 \end{bmatrix}$$
The Banditron (Kakade, S, Tewari 08)

Theorem

- Banditron's regret is $O(\sqrt{\frac{kT}{\mu^2}})$
- Banditron’s runtime is $O(\frac{k}{\mu^2})$
The Banditron (Kakade, S, Tewari 08)

**Theorem**

- Banditron’s regret is $O(\sqrt{KT/\mu^2})$
- Banditron’s runtime is $O(k/\mu^2)$

The crux of difference between Halving and Banditron:

- Without having the full information, the version space is non-convex and therefore it is hard to utilize the structure of $\mathcal{H}$
- Because we relied on the Perceptron we did utilize the structure of $\mathcal{H}$ and got an efficient algorithm
- We managed to obtain 'full-information examples’ by using exploration
- The price of exploration is a higher regret
### Intermediate Summary – Trading Regret for Efficiency

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Regret</th>
<th>runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Halving</td>
<td>$\frac{k}{\mu^2}$</td>
<td>$\left(\frac{1}{\mu}\right)^{1/\mu^2}$</td>
</tr>
<tr>
<td>Banditron</td>
<td>$\frac{\sqrt{kT}}{\mu}$</td>
<td>$\frac{k}{\mu^2}$</td>
</tr>
</tbody>
</table>

![Graph showing regret vs. \(\tau\)](image-url)
Second example: general costs, non-separable, 2 arms

Action set $A = \{0, 1\}$

For $t = 1, 2, \ldots$

- Learner receives $x_t$
- Environment chooses cost vector $c_t : A \rightarrow [0, 1]$ (unknown to Learner)
- Learner chooses $\tilde{p}_t \in [0, 1]$
- Learner chooses action $a_t \in A$ according to $\Pr[a_t = 1] = \tilde{p}_t$
- Learner pays cost $c_t(a_t)$

Remark: Can be extended to $k$ arms using e.g. the offset tree (Beygelzimer and Langford '09)

Shai Shalev-Shwartz (TTI-C ⇒ Hebrew U)
Second example: general costs, non-separable, 2 arms

Action set $A = \{0, 1\}$

For $t = 1, 2, \ldots$

- Learner receives $x_t$
- Environment chooses cost vector $c_t : A \rightarrow [0, 1]$ (unknown to Learner)
- Learner chooses $\tilde{p}_t \in [0, 1]$
- Learner chooses action $a_t \in A$ according to $\Pr[a_t = 1] = \tilde{p}_t$
- Learner pays cost $c_t(a_t)$

Remark: Can be extended to $k$ arms using e.g. the offset tree (Beygelzimer and Langford ’09)
Hypothesis class and regret goal

\[ \mathcal{H} = \{ \mathbf{x} \mapsto \phi(\langle \mathbf{w}, \mathbf{x} \rangle) : \| \mathbf{w} \|_2 \leq 1 \}, \quad \phi(z) = \frac{1}{1 + \exp(-z/\mu)} \]

Goal: bounded regret

\[ \sum_{t=1}^{T} \mathbb{E}_{a \sim \tilde{p}_t} [c_t(a)] - \min_{h \in \mathcal{H}} \sum_{t=1}^{T} \mathbb{E}_{a \sim h(x_t)} [c_t(a)] \]
Hypothesis class and regret goal

\[ \mathcal{H} = \{ \mathbf{x} \mapsto \phi(\langle \mathbf{w}, \mathbf{x} \rangle) : \|\mathbf{w}\|_2 \leq 1 \}, \quad \phi(z) = \frac{1}{1+\exp(-z/\mu)} \]

Goal: bounded regret

Challenging: even with full information no known efficient algorithms
First approach – EXP4

- Step 1: Dimensionality reduction to $d' = \tilde{O}(\frac{1}{\mu^2})$
- Step 2: Discretize $\mathcal{H}$ to $(1/\mu)^{d'}$ hypotheses
- Apply EXP4 on the resulting finite set of hypotheses (Auer, Cesa-Bianchi, Freund, Schapire 2002)
First approach – EXP4

- Step 1: Dimensionality reduction to $d' = \tilde{O}(\frac{1}{\mu^2})$
- Step 2: Discretize $\mathcal{H}$ to $(1/\mu)^{d''}$ hypotheses
- Apply EXP4 on the resulting finite set of hypotheses
  (Auer, Cesa-Bianchi, Freund, Schapire 2002)

Analysis:

- Regret bound is $\tilde{O}\left(\sqrt{\frac{kT}{\mu^2}}\right)$
- Runtime grows like $(1/\mu)^{1/\mu^2}$
Second Approach – IDPK

1. Reduction to weighted binary classification
   Similar to Bianca Zadrozny 2003, Alina Beygelzimer and John Langford 2009

2. Learning fuzzy halfspaces using the Infinite-Dimensional-Polynomial-Kernel (S, Shamir, Sridharan 2009)
Reduction to weighted binary classification

- The expected cost of a strategy $p$ is:

$$E_{a \sim p}[c(a)] = pc(1) + (1 - p)c(0)$$
Reduction to weighted binary classification

- The expected cost of a strategy $p$ is:
  $$\mathbb{E}_{a \sim p}[c(a)] = p c(1) + (1 - p) c(0)$$

- **Limited feedback:** We don’t observe $c_t$ but only $c_t(a_t)$
Reduction to weighted binary classification

- The expected cost of a strategy $p$ is:
  \[ \mathbb{E}_{a \sim p}[c(a)] = pc(1) + (1 - p)c(0) \]

- **Limited feedback:** We don’t observe $c_t$ but only $c_t(a_t)$

- Define the following loss function:
  \[ \ell_t(p) = \nu_t|p - y_t| = \begin{cases} 
  \frac{c_t(1)}{\tilde{p}_t}|p - 0| & \text{if } a_t = 1 \\
  \frac{c_t(0)}{1 - \tilde{p}_t}|p - 1| & \text{if } a_t = 0 
\end{cases} \]
The expected cost of a strategy $p$ is:

$$E_{a \sim p}[c(a)] = pc(1) + (1 - p)c(0)$$

**Limited feedback:** We don’t observe $c_t$ but only $c_t(a_t)$

Define the following loss function:

$$\ell_t(p) = \nu_t |p - y_t| = \begin{cases} \frac{c_t(1)}{\tilde{p}_t} |p - 0| & \text{if } a_t = 1 \\ \frac{c_t(0)}{1 - \tilde{p}_t} |p - 1| & \text{if } a_t = 0 \end{cases}$$

Note that $\ell_t$ only depends on available information and that:

$$E_{a_t \sim \tilde{p}_t}[\ell_t(p)] = E_{a \sim p}[c_t(a)]$$
Reduction to weighted binary classification

- The expected cost of a strategy $p$ is:
  \[ \mathbb{E}_{a \sim p}[c(a)] = p \cdot c(1) + (1 - p) \cdot c(0) \]

- **Limited feedback:** We don’t observe $c_t$ but only $c_t(a_t)$

- Define the following loss function:
  \[ \ell_t(p) = \nu_t |p - y_t| = \begin{cases} 
  \frac{c_t(1)}{\tilde{p}_t} |p - 0| & \text{if } a_t = 1 \\
  \frac{c_t(0)}{1 - \tilde{p}_t} |p - 1| & \text{if } a_t = 0 
\end{cases} \]

- Note that $\ell_t$ only depends on available information and that:
  \[ \mathbb{E}_{a_t \sim \tilde{p}_t}[\ell_t(p)] = \mathbb{E}_{a \sim p}[c_t(a)] \]

- The above almost works – we should slightly change the probabilities so that $\nu_t$ will not explode.

  **Bottom line:** regret bound w.r.t. $\ell_t \Rightarrow$ regret bound w.r.t. $c_t$
Goal: regret bound w.r.t. class $\mathcal{H} = \{x \mapsto \phi(\langle w, x \rangle)\}$

Working with expected 0 – 1 loss: $|\phi(\langle w, x \rangle) - y|$
Step 2 – Learning fuzzy halfspaces with IDPK

- **Goal:** regret bound w.r.t. class $\mathcal{H} = \{ x \mapsto \phi(\langle w, x \rangle) \}$
  Working with expected 0 – 1 loss: $|\phi(\langle w, x \rangle) - y|$

- **Problem:** The above is non-convex w.r.t. $w$
Step 2 – Learning fuzzy halfspaces with IDPK

- **Goal**: regret bound w.r.t. class $\mathcal{H} = \{ x \mapsto \phi(\langle w, x \rangle) \}$
  
  Working with expected 0 – 1 loss: $|\phi(\langle w, x \rangle) - y|$

- **Problem**: The above is non-convex w.r.t. $w$

- **Main idea**: Work with a larger hypothesis class for which the loss becomes convex
Step 2 – Learning fuzzy halfspaces with IDPK

- **Original class:** $\mathcal{H} = \{ h_w(x) = \phi(\langle w, x \rangle) : \|w\| \leq 1 \}$

- **New class:** $\mathcal{H}' = \{ h_v(x) = \langle v, \psi(x) \rangle : \|v\| \leq B \}$ where $\psi : \mathcal{X} \rightarrow \mathbb{R}^N$
  
  s.t. $\forall j, \forall(i_1, \ldots, i_j), \psi(x)_{i_1,\ldots,i_j} = 2^{j/2} x_{i_1} \cdots x_{i_j}$

Lemma (S, Shamir, Sridharan 2009)

If $B = \exp(\tilde{O}(1/\mu))$ then for all $h \in \mathcal{H}$ exists $h' \in \mathcal{H}'$ s.t. for all $x$, $h(x) \approx h'(x)$.

Remark: The above is a pessimistic choice of $B$. In practice, smaller $B$s suffice. Is it tight? Even if it is, are there natural assumptions under which a better bound holds? (e.g. Kalai, Klivans, Mansour, Servedio 2005)
Step 2 – Learning fuzzy halfspaces with IDPK

- **Original class:** $\mathcal{H} = \{ h_w(x) = \phi(\langle w, x \rangle) : \|w\| \leq 1 \}$

- **New class:** $\mathcal{H}' = \{ h_v(x) = \langle v, \psi(x) \rangle : \|v\| \leq B \}$ where $\psi : \mathcal{X} \rightarrow \mathbb{R}^N$
  s.t. $\forall j, \forall (i_1, \ldots, i_j), \psi(x)_{i_1, \ldots, i_j} = 2^{j/2} x_{i_1} \cdots x_{i_j}$

**Lemma (S, Shamir, Sridharan 2009)**

If $B = \exp(\tilde{O}(1/\mu))$ then for all $h \in \mathcal{H}$ exists $h' \in \mathcal{H}'$ s.t. for all $x$, $h(x) \approx h'(x)$.
Step 2 – Learning fuzzy halfspaces with IDPK

- Original class: $\mathcal{H} = \{ h_w(x) = \phi(\langle w, x \rangle) : \|w\| \leq 1 \}$
- New class: $\mathcal{H'} = \{ h_v(x) = \langle v, \psi(x) \rangle : \|v\| \leq B \}$ where $\psi : \mathcal{X} \rightarrow \mathbb{R}^N$
  s.t. $\forall j, \forall (i_1, \ldots, i_j), \psi(x)(i_1, \ldots, i_j) = 2^{j/2} x_{i_1} \cdots x_{i_j}$

**Lemma (S, Shamir, Sridharan 2009)**

If $B = \exp(\tilde{O}(1/\mu))$ then for all $h \in \mathcal{H}$ exists $h' \in \mathcal{H'}$ s.t. for all $x$, $h(x) \approx h'(x)$.

**Remark:** The above is a pessimistic choice of $B$. In practice, smaller $B$ suffices. Is it tight? Even if it is, are there natural assumptions under which a better bound holds? (e.g. Kalai, Klivans, Mansour, Servedio 2005)
Proof idea

- Polynomial approximation: $\phi(z) \approx \sum_{j=0}^{\infty} \beta_j z^j$
Proof idea

- Polynomial approximation: $\phi(z) \approx \sum_{j=0}^{\infty} \beta_j z^j$
- Therefore:

$$\phi(\langle w, x \rangle) \approx \sum_{j=0}^{\infty} \beta_j (\langle w, x \rangle)^j$$

$$= \sum_{j=0}^{\infty} \sum_{k_1, \ldots, k_j} 2^{-j/2} \beta_j 2^{j/2} w_{k_1} \cdots w_{k_j} x_{k_1} \cdots x_{k_j}$$

$$= \langle v_w, \psi(x) \rangle$$
Proof idea

- Polynomial approximation: \( \phi(z) \approx \sum_{j=0}^{\infty} \beta_j z^j \)
- Therefore:

\[
\phi(\langle w, x \rangle) \approx \sum_{j=0}^{\infty} \beta_j (\langle w, x \rangle)^j
\]

\[
= \sum_{j=0}^{\infty} \sum_{k_1, \ldots, k_j} 2^{-j/2} \beta_j 2^{j/2} w_{k_1} \cdots w_{k_j} x_{k_1} \cdots x_{k_j}
\]

\[
= \langle v_w, \psi(x) \rangle
\]

- To obtain a concrete bound we use Chebyshev approximation technique: Family of orthogonal polynomials w.r.t. inner product:

\[
\langle f, g \rangle = \int_{x=-1}^{1} \frac{f(x)g(x)}{\sqrt{1-x^2}} \, dx
\]
Although the dimension is infinite, can be solved using the kernel trick

The corresponding kernel (a.k.a. Vovk’s infinite polynomial):

\[ \langle \psi(x), \psi(x') \rangle = K(x, x') = \frac{1}{1 - \frac{1}{2} \langle x, x' \rangle} \]

Algorithm boils down to online regression with the above kernel

Convex! Can be solved e.g. using Zinkevich’s OCP

Regret bound is \( B \sqrt{T} \)
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Regret</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXP4</td>
<td>$\sqrt{T/\mu^2}$</td>
<td>$\exp(\tilde{O}(1/\mu^2))$</td>
</tr>
<tr>
<td>IDPK</td>
<td>$T^{3/4}\exp(\tilde{O}(1/\mu))$</td>
<td>$T$</td>
</tr>
</tbody>
</table>
Trading regret rate for efficiency:

- Multiclass, separable
  - Banditron
  - Halving

- 2 arms, non-separable
  - IDPK
  - EXP4

Open questions:
- More points on the curve (new algorithms)
- Lower bounds...