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

Shai Shalev-Shwartz

TTI-C $\Longrightarrow$ Hebrew University

On-line Learning with Limited Feedback Workshop, 2009

June 2009

Online Learning with restricted computational power

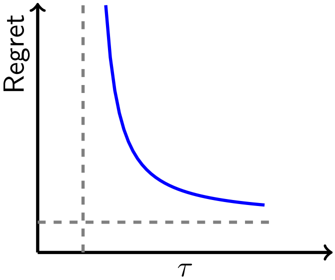
Main Question

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

Online Learning with restricted computational power

Main Question

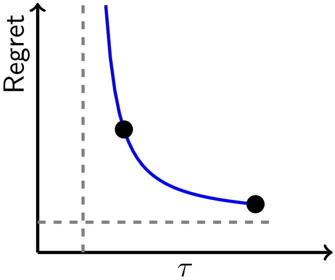
Given a runtime constraint τ, horizon T, reference class $\mathcal{H}$:
What is the achievable regret of an algorithm whose (amortized) runtime is $O(\tau)$?



Online Learning with restricted computational power

Main Question

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



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 $\mathbf{x}_t \in \mathcal{X}$
- Environment chooses cost vector $c_t : A \to [0, 1]$ (unknown to learner)
- Learner chooses action $a_t \in A$
- Learner pay cost $c_t(a_t)$

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 $\mathbf{x}_t \in \mathcal{X}$
- 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

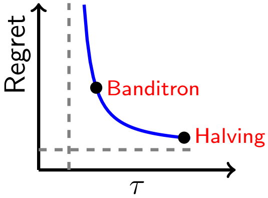
Low regret w.r.t. a reference hypothesis class $\mathcal{H}$:

$$\text{Regret} \stackrel{\text{def}}{=} \sum_{t=1}^{T} c_t(a_t) - \min_{h \in \mathcal{H}} \sum_{t=1}^{T} c_t(h(\mathbf{x}_t))$$

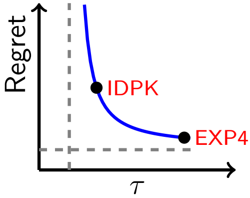
Outline

$\mathcal{H}$ is the class of linear hypotheses

Multiclass, separable



2 arms, non-separable



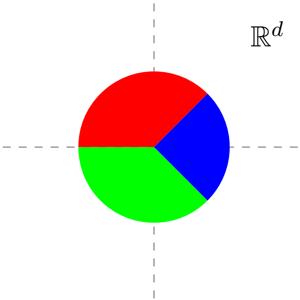
First example: Bandit Multi-class Categorization

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

- Learner receives side information $\mathbf{x}_t \in \mathcal{X}$
- Environment chooses 'correct' arm $y_t \in A$ (unknown to learner)
- Learner chooses action $a_t \in A$
- Learner pay cost $c_t(a_t) = \mathbf{1}[a_t \neq y_t]$

Linear Hypotheses

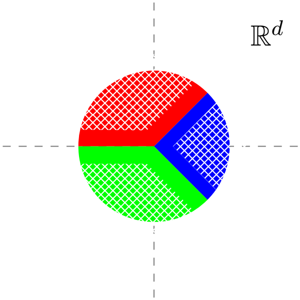
$$\mathcal{H} = \{\mathbf{x} \mapsto \underset{r}{\operatorname{argmax}}\, (W\,\mathbf{x})_r \;:\; W \in \mathbb{R}^{k,d},\ \|W\|_F \le 1\}$$



Large margin assumption

Assumption: Data is separable with margin μ:

$$\forall t,\ \forall r \neq y_t,\ (W\mathbf{x}_t)_{y_t} - (W\mathbf{x}_t)_r \geq \mu$$



First approach – Halving

Halving for Bandit Multiclass categorization

Initialize: $V_1 = \mathcal{H}$

For $t = 1, 2, \ldots$

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

Analysis:

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

Using Halving

- Step 1: Dimensionality reduction to $d' = \tilde{O}(\frac{1}{\mu^2})$
- 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 $d' = \tilde{O}(\frac{1}{\mu^2})$
- 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: Lets try to rely on the Perceptron

The Mutliclass Perceptron

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

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

The Mutliclass Perceptron

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

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

$$
U^t = \left[\begin{array}{ccc}
0 & \ldots & 0 \\
 & \vdots & \\
0 & \ldots & 0 \\
\ldots & \mathbf{x}_t & \ldots \\
0 & \ldots & 0 \\
 & \vdots & \\
0 & \ldots & 0 \\
\ldots & -\mathbf{x}_t & \ldots \\
0 & \ldots & 0 \\
 & \vdots & \\
0 & \ldots & 0
\end{array}\right]
\begin{array}{l}
\\
\\
\\
\text{Row } y_t \\
\\
\\
\\
\text{Row } \hat{y}_t \\
\\
\\
\\
\end{array}
$$

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: $(\mathbf{x}_t, \hat{y}_t, \tilde{y}_t = y_t)$

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

The Banditron (Kakade, S, Tewari 08)

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

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

$$\tilde{U}^t = \begin{array}{cl} \left[\begin{array}{ccc} 0 & \cdots & 0 \\ & \vdots & \\ 0 & \cdots & 0 \\ \cdots & \frac{\mathbf{1}[y_t = \tilde{y}_t]}{P(\tilde{y}_t)}\,\mathbf{x}_t & \cdots \\ 0 & \cdots & 0 \\ & \vdots & \\ 0 & \cdots & 0 \\ \cdots & -\mathbf{x}_t & \cdots \\ 0 & \cdots & 0 \\ & \vdots & \\ 0 & \cdots & 0 \end{array}\right] & \begin{array}{l} \\ \\ \\ \text{Row } \tilde{y}_t \\ \\ \\ \\ \text{Row } \hat{y}_t \\ \\ \\ \\ \end{array} \end{array}$$

The Banditron (Kakade, S, Tewari 08)

Theorem

- *Banditron's regret is* $O(\sqrt{k\,T/\mu^2})$
- *Banditron's runtime is* $O(k/\mu^2)$

The Banditron (Kakade, S, Tewari 08)

Theorem

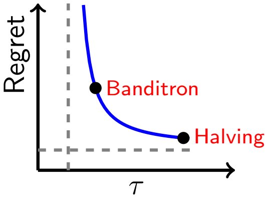
- *Banditron's regret is* $O(\sqrt{k\,T/\mu^2})$
- *Banditron's runtime is* $O(k/\mu^2)$

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

Algorithm	Regret	runtime
Halving	$\frac{k}{\mu^2}$	$\left(\frac{1}{\mu}\right)^{1/\mu^2}$
Banditron	$\frac{\sqrt{kT}}{\mu}$	$\frac{k}{\mu^2}$



Second example: general costs, non-separable, 2 arms

Action set $A = \{0, 1\}$
For $t = 1, 2, \ldots$

- Learner receives $\mathbf{x}_t$
- Environment chooses cost vector $c_t : A \to [0, 1]$ (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 $c_t(a_t)$

Second example: general costs, non-separable, 2 arms

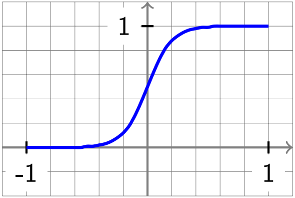
Action set $A = \{0, 1\}$
For $t = 1, 2, \ldots$

- Learner receives $\mathbf{x}_t$
- Environment chooses cost vector $c_t : A \to [0, 1]$ (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

$$\mathcal{H} = \{\mathbf{x} \mapsto \phi(\langle \mathbf{w}, \mathbf{x} \rangle) : \|\mathbf{w}\|_2 \le 1\}, \qquad \phi(z) = \frac{1}{1+\exp(-z/\mu)}$$

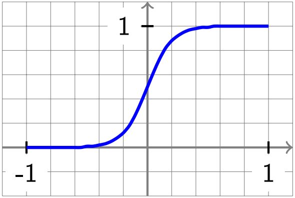


Goal: bounded regret

$$\sum_{t=1}^{T} \mathbb{E}_{a \sim \tilde{p}_t}[c_t(a)] - \min_{h \in \mathcal{H}} \sum_{t=1}^{T} \mathbb{E}_{a \sim h(\mathbf{x}_t)}[c_t(a)]$$

Hypothesis class and regret goal

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



Goal: bounded regret

$$\sum_{t=1}^{T} \mathbb{E}_{a \sim \tilde{p}_t}[c_t(a)] - \min_{h \in \mathcal{H}} \sum_{t=1}^{T} \mathbb{E}_{a \sim h(\mathbf{x}_t)}[c_t(a)]$$

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 $\tilde{O}\left(\sqrt{\frac{k\,T}{\mu^2}}\right)$
- 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-**D**imensional-**P**olynomial-**K**ernel (S, Shamir, Sridharan 2009)

Reduction to weighted binary classification

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

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:
  $\mathbb{E}_{a\sim p}[c(a)] \; = \; p\,c(1) + (1-p)\,c(0)$
- Limited feedback: We don't observe c_t but only $c_t(a_t)$
- Define the following loss function:

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

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 $c_t(a_t)$
- 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 ℓ_t only depends on available information and that:

$$\mathbb{E}_{a_t\sim\tilde{p}_t}[\ell_t(p)] \;=\; \mathbb{E}_{a\sim p}[c_t(a)]$$

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 $c_t(a_t)$
- 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 ℓ_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 ν_t will not explode.
  Bottom line: regret bound w.r.t. $\ell_t \Rightarrow$ regret bound w.r.t. c_t

Step 2 – Learning fuzzy halfspaces with IDPK

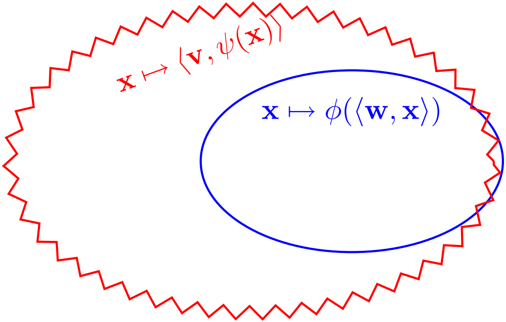
- Goal: regret bound w.r.t. class $\mathcal{H} = \{\mathbf{x} \mapsto \phi(\langle \mathbf{w}, \mathbf{x} \rangle)\}$
  Working with expected $0-1$ loss: $|\phi(\langle \mathbf{w}, \mathbf{x} \rangle) - y|$

Step 2 – Learning fuzzy halfspaces with IDPK

- Goal: regret bound w.r.t. class $\mathcal{H} = \{\mathbf{x} \mapsto \phi(\langle \mathbf{w}, \mathbf{x} \rangle)\}$
  Working with expected $0-1$ loss: $|\phi(\langle \mathbf{w}, \mathbf{x} \rangle) - y|$
- Problem: The above is non-convex w.r.t. $\mathbf{w}$

Step 2 – Learning fuzzy halfspaces with IDPK

- Goal: regret bound w.r.t. class $\mathcal{H} = \{\mathbf{x} \mapsto \phi(\langle \mathbf{w}, \mathbf{x} \rangle)\}$
  Working with expected $0-1$ loss: $|\phi(\langle \mathbf{w}, \mathbf{x} \rangle) - y|$
- Problem: The above is non-convex w.r.t. $\mathbf{w}$
- Main idea: Work with a larger hypothesis class for which the loss becomes convex



Step 2 – Learning fuzzy halfspaces with IDPK

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

Step 2 – Learning fuzzy halfspaces with IDPK

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

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 $\mathbf{x}$, $h(\mathbf{x}) \approx h'(\mathbf{x})$.

Step 2 – Learning fuzzy halfspaces with IDPK

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

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 $\mathbf{x}$, $h(\mathbf{x}) \approx h'(\mathbf{x})$.

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:

$$\begin{aligned}
\phi(\langle \mathbf{w}, \mathbf{x} \rangle) &\approx \sum_{j=0}^{\infty} \beta_j (\langle \mathbf{w}, \mathbf{x} \rangle)^j \\
&= \sum_{j=0}^{\infty} \sum_{k_1,\ldots,k_j} 2^{-j/2} \beta_j 2^{j/2} w_{k_1} \cdots w_{k_j} x_{k_1} \cdot \cdots \cdot x_{k_j} \\
&= \langle \mathbf{v}_{\mathbf{w}}, \psi(\mathbf{x}) \rangle
\end{aligned}$$

Proof idea

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

$$\begin{aligned}
\phi(\langle \mathbf{w}, \mathbf{x} \rangle) &\approx \sum_{j=0}^{\infty} \beta_j (\langle \mathbf{w}, \mathbf{x} \rangle)^j \\
&= \sum_{j=0}^{\infty} \sum_{k_1,\ldots,k_j} 2^{-j/2} \beta_j 2^{j/2} w_{k_1} \cdots w_{k_j} x_{k_1} \cdots \cdot x_{k_j} \\
&= \langle \mathbf{v}_{\mathbf{w}}, \psi(\mathbf{x}) \rangle
\end{aligned}$$

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

$$\langle f, g \rangle = \int_{x=-1}^{1} \frac{f(x) g(x)}{\sqrt{1-x^2}} dx$$

Infinite-Dimensional-Polynomial-Kernel

- Although the dimension is infinite, can be solved using the kernel trick
- The corresponding kernel (a.k.a. Vovk's infinite polynomial):

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

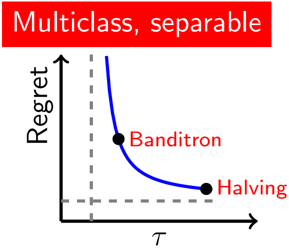
- Algorithm boils down to online regression with the above kernel
- Convex! Can be solved e.g. using Zinkevich's OCP
- Regret bound is $B\sqrt{T}$

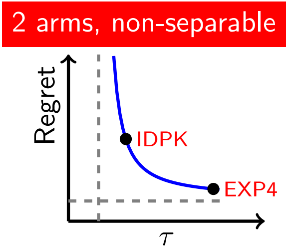
Trading Regret for Efficiency

Algorithm	Regret	runtime
EXP4	$\sqrt{T/\mu^2}$	$\exp(\tilde{O}(1/\mu^2))$
	$\wedge$	$\vee$
IDPK	$T^{3/4}\exp(\tilde{O}(1/\mu))$	T

Summary

- Trading regret rate for efficiency:





Open questions:

- More points on the curve (new algorithms)
- Lower bounds ???