Fast boosting using adversarial bandits

Róbert Busa-Fekete[1,2] Balázs Kégl[1,3]

[1]Linear Accelerator Laboratory (LAL), University of Paris-Sud, CNRS

[2]Research Group on Artificial Intelligence of the Hungarian Academy of Sciences and University of Szeged (RGAI)

[3]Computer Science Laboratory (LRI), University of Paris-Sud, CNRS and INRIA-Saclay

ANR meeting
January 26, 2012

$\textsc{AdaBoost}(D_n = \{(\mathbf{x}_i, y_i)\}_{i=1}^n, \textsc{Base}(\cdot,\cdot), T)$

$\mathbf{w}^{(1)} \leftarrow (1/n, \dots, 1/n)$ ▷ *initial weights*

for $t \leftarrow 1$ **to** T

  $h^{(t)} \leftarrow \textsc{Base}(D_n, \mathbf{w}^{(t)})$ ▷ *calling the base learner*

  $\gamma^{(t)} \leftarrow \sum_{i=1}^{n} w_i^{(t)} h^{(t)}(\mathbf{x}_i) y_i$ ▷ *edge* $= 1 - 2 \times$ *error*

  $\alpha^{(t)} \leftarrow \frac{1}{2} \ln\left(\frac{1+\gamma^{(t)}}{1-\gamma^{(t)}}\right)$ ▷ *coefficient of* $h^{(t)}$

  for $i \leftarrow 1$ **to** n ▷ *re-weighting the points*

    if $h^{(t)}(\mathbf{x}_i) \neq y_i$ **then**

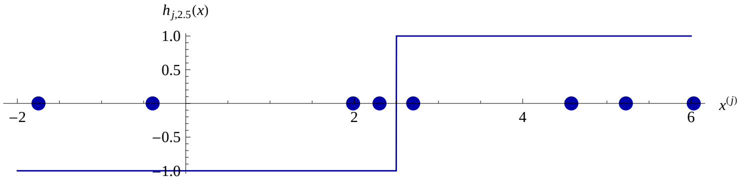
      $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{1-\gamma^{(t)}}$

    else

      $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{1+\gamma^{(t)}}$

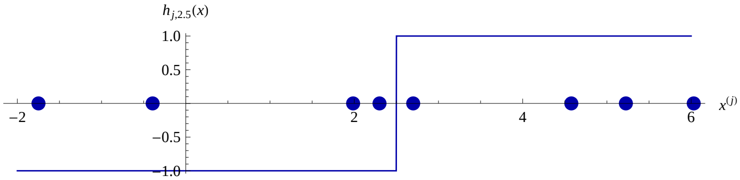
return $f^{(T)}(\cdot) = \sum_{t=1}^{T} \alpha^{(t)} h^{(t)}(\cdot)$

Decision stumps



$$h_{j,b}(\mathbf{x}) = \begin{cases} 1 & \text{if } x^{(j)} \geq b, \\ -1 & \text{otherwise,} \end{cases}$$

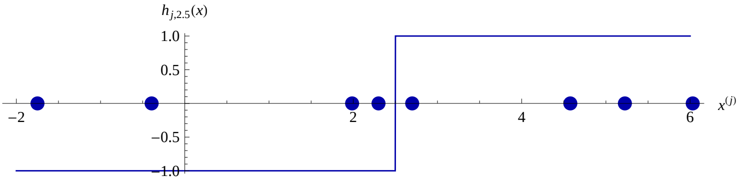
Decision stumps



$$h_{j,b}(\mathbf{x}) = \begin{cases} 1 & \text{if } x^{(j)} \geq b, \\ -1 & \text{otherwise,} \end{cases}$$

1. Can be learned in $\theta(ndK)$ time (if features are pre-sorted)

Decision stumps



$$h_{j,b}(\mathbf{x}) = \begin{cases} 1 & \text{if } x^{(j)} \geq b, \\ -1 & \text{otherwise,} \end{cases}$$

1. Can be learned in $\theta(ndK)$ time (if features are pre-sorted)
2. Looking at each feature in every boosting iterations

Accelerating the training of AdaBoost
Motivation, Related work

- Well boostable base learners
  - Decision stumps $O(ndK)$
  - Decision trees $O(ndK\log N)$
  - Decision product $O(ndKm)$
- Saving on the n factor
  - stochastic boosting, FilterBoost[1]
- Saving on the d factor
  - LazyBoost[2] (random selection)
  - our technique: learn the usefulness of the features in a sequential game using multi-armed bandits (MABs)

[1] J.K. Bradley and R.E. Schapire. FilterBoost: Regression and classification on large datasets. In *NIPS 2008*.

[2] G. Escudero, L. Màrquez, and G. Rigau. Boosting applied to word sense disambiguation. In *ECML 2000*.

The formal setup

- Partition the base classifier set $\mathcal{H}$ into $\{\mathcal{H}_1, \ldots, \mathcal{H}_M\}$
  - in each iteration t, use the Multi-armed Bandit (MAB) algorithm to select a subset $\mathcal{H}_{j^{(t)}}$
  - call the base learner to select $h^{(t)} \in \mathcal{H}_{j^{(t)}}$
  - compute the edge

$$\gamma^{(t)} = \sum_{i=1}^{n} w_i^{(t)} h^{(t)}(\mathbf{x}_i) y_i = 1 - 2 \times \text{error}$$

  - return the reward

$$r_j^{(t)} = \min\left(1, -\log\sqrt{1 - {\gamma^{(t)}}^2}\right).$$

  to the MAB

$\textsc{AdaBoost}(D_n = \{(\mathbf{x}_i, y_i)\}_{i=1}^n, \textsc{Base}(\cdot,\cdot), T)$

$\mathbf{w}^{(1)} \leftarrow (1/n, \ldots, 1/n)$ ▷ *initial weights*

for $t \leftarrow 1$ **to** T

  $h^{(t)} \leftarrow \textsc{Base}(D_n, \mathbf{w}^{(t)})$ ▷ *calling the base learner*

  $\gamma^{(t)} \leftarrow \sum_{i=1}^{n} w_i^{(t)} h^{(t)}(\mathbf{x}_i) y_i$ ▷ *edge* $= 1 - 2 \times$ *error*

  $\alpha^{(t)} \leftarrow \frac{1}{2} \ln\left(\frac{1+\gamma^{(t)}}{1-\gamma^{(t)}}\right)$ ▷ *coefficient of* $h^{(t)}$

  for $i \leftarrow 1$ **to** n ▷ *re-weighting the points*

    if $h^{(t)}(\mathbf{x}_i) \neq y_i$ **then**

      $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{1-\gamma^{(t)}}$

    else

      $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{1+\gamma^{(t)}}$

return $f^{(T)}(\cdot) = \sum_{t=1}^{T} \alpha^{(t)} h^{(t)}(\cdot)$

$\textsc{AdaBoost.BA}(D_n = \{(\mathbf{x}_i, y_i)\}_{i=1}^n, \textsc{Base}(\cdot,\cdot,\cdot), T, \mathcal{H}, \textsc{BanditAlgo})$

$\mathbf{w}^{(1)} \leftarrow (1/n, \ldots, 1/n)$ ▷ *initial weights*

for $t \leftarrow 1$ **to** T

  $j \leftarrow \textsc{BanditAlgo}.\mathit{getArm}()$

  $h^{(t)} \leftarrow \textsc{Base}\big(D_n, \mathbf{w}^{(t)}, \mathcal{H}_j\big)$ ▷ *calling the base learner*

  $\gamma^{(t)} \leftarrow \sum_{i=1}^{n} w_i^{(t)} h^{(t)}(\mathbf{x}_i) y_i$ ▷ *edge* $= 1 - 2 \times$ *error*

  $r_j^{(t)} = \min\left(1, -\log\sqrt{1 - {\gamma_{\mathcal{H}_j}^{(t)}}^2}\right)$ ▷ *calculate reward*

  $\textsc{BanditAlgo}.\mathit{receiveReward}(j, r_j^{(t)})$

  $\alpha^{(t)} \leftarrow \frac{1}{2}\ln\left(\frac{1+\gamma^{(t)}}{1-\gamma^{(t)}}\right)$ ▷ *coefficient of* $h^{(t)}$

▷ *re-weighting the points*

return $f^{(T)}(\cdot) = \sum_{t=1}^{T} \alpha^{(t)} h^{(t)}(\cdot)$

Multi-armed Bandit (MAB) Problem, Adversarial setup

- (, , , , ) indexed by $\{1, \ldots, M\}$

Multi-armed Bandit (MAB) Problem, Adversarial setup

- $\left(r_1^{(t)}, r_2^{(t)}, \ldots, r_{M-1}^{(t)}, r_M^{(t)} \right)$ indexed by $\{1, \ldots, M\}$
- An adversary chooses a reward vector $\mathbf{r}^{(t)} \in \mathbb{R}^M$

Multi-armed Bandit (MAB) Problem, Adversarial setup

- $\left(\blacksquare, \blacksquare, \blacksquare, r_{j^{(t)}}^{(t)}, \blacksquare\right)$ indexed by $\{1, \ldots, M\}$
- An adversary chooses a reward vector $\mathbf{r}^{(t)} \in \mathbb{R}^M$
- Decision maker chooses an arm $j^{(t)}$ and receives reward $r_{j^{(t)}}^{(t)}$ in each iteration step (bandit feedback)

Multi-armed Bandit (MAB) Problem, Adversarial setup

- $\left(\blacksquare, \blacksquare, \blacksquare, r_{j^{(t)}}^{(t)}, \blacksquare\right)$ indexed by $\{1, \ldots, M\}$
- An adversary chooses a reward vector $\mathbf{r}^{(t)} \in \mathbb{R}^M$
- Decision maker chooses an arm $j^{(t)}$ and receives reward $r_{j^{(t)}}^{(t)}$ in each iteration step (bandit feedback)
- The most common performance measure is the weak regret

$$R = \underbrace{\max_j \sum_{t=1}^{T} r_j^{(t)}}_{\text{Total reward of best arm}} - \underbrace{\sum_{t=1}^{T} r_{j^{(t)}}^{(t)}}_{G^{(T)} = \text{Total reward}}$$

Multi-armed Bandit (MAB) Problem, Adversarial setup

- ([redacted], [redacted], [redacted], $r^{(t)}_{j^{(t)}}$, [redacted]) indexed by $\{1, \ldots, M\}$
- An adversary chooses a reward vector $\mathbf{r}^{(t)} \in \mathbb{R}^M$
- Decision maker chooses an arm $j^{(t)}$ and receives reward $r^{(t)}_{j^{(t)}}$ in each iteration step (bandit feedback)
- The most common performance measure is the weak regret

$$R = \underbrace{\max_j \sum_{t=1}^{T} r^{(t)}_j}_{\text{Total reward of best arm}} - \underbrace{\sum_{t=1}^{T} r^{(t)}_{j^{(t)}}}_{G^{(T)} = \text{Total reward}}$$

- A theoretically well-founded algorithm: $\mathrm{EXP3.P}$

Multi-armed Bandit (MAB) Problem, Adversarial setup

- $\left(\blacksquare, \blacksquare, \blacksquare, r_{j^{(t)}}^{(t)}, \blacksquare\right)$ indexed by $\{1, \ldots, M\}$
- An adversary chooses a reward vector $\mathbf{r}^{(t)} \in \mathbb{R}^M$
- Decision maker chooses an arm $j^{(t)}$ and receives reward $r_{j^{(t)}}^{(t)}$ in each iteration step (bandit feedback)
- The most common performance measure is the weak regret

$$R = \underbrace{\max_j \sum_{t=1}^{T} r_j^{(t)}}_{\text{Total reward of best arm}} - \underbrace{\sum_{t=1}^{T} r_{j^{(t)}}^{(t)}}_{G^{(T)} = \text{Total reward}}$$

- A theoretically well-founded algorithm: $\mathrm{EXP3.P}$

Multi-armed Bandit (MAB) Problem, Adversarial setup

- ([redacted], [redacted], [redacted], $r_{j^{(t)}}^{(t)}$, [redacted]) indexed by $\{1, \ldots, M\}$
- An adversary chooses a reward vector $\mathbf{r}^{(t)} \in \mathbb{R}^M$
- Decision maker chooses an arm $j^{(t)}$ and receives reward $r_{j^{(t)}}^{(t)}$ in each iteration step (bandit feedback)
- The most common performance measure is the weak regret

$$R = \underbrace{\max_j \sum_{t=1}^{T} r_j^{(t)}}_{\text{Total reward of best arm}} - \underbrace{\sum_{t=1}^{T} r_{j^{(t)}}^{(t)}}_{G^{(T)}=\text{Total reward}}$$

- A theoretically well-founded algorithm: EXP3.P

Theorem (Auer et al.(1995))

With probability at least $1 - \delta$

$$R \leq 4\sqrt{MT \log \frac{MT}{\delta}} + 4\sqrt{\frac{5}{3} MT \log M} + 8 \log \frac{MT}{\delta}$$

Reward definition

Theorem (Schapire,Singer(1998))

For AdaBoost.MH

$$R\left(\mathbf{f}^{(T)}\right) = \frac{1}{n}\sum_{i=1}^{n}\mathbb{I}\left\{\ell(\mathbf{x}_i) \neq \widehat{\ell}(\mathbf{x}_i)\right\} \leq \sqrt{K-1}\prod_{t=1}^{T}\sqrt{1-\gamma^{(t)^2}}$$

Reward definition

Theorem (Schapire,Singer(1998))

For AdaBoost.MH

$$R\left(\mathbf{f}^{(T)}\right) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}\left\{\ell(\mathbf{x}_i) \neq \widehat{\ell}(\mathbf{x}_i)\right\} \leq \sqrt{K-1}\prod_{t=1}^{T}\sqrt{1-{\gamma^{(t)}}^2}$$

- reward should depend on $\sqrt{1-{\gamma^{(t)}}^2}$

Reward definition

Theorem (Schapire,Singer(1998))

For ADABOOST.MH

$$R\left(\mathbf{f}^{(T)}\right) = \frac{1}{n}\sum_{i=1}^{n}\mathbb{I}\left\{\ell(\mathbf{x}_i) \neq \widehat{\ell}(\mathbf{x}_i)\right\} \leq \sqrt{K-1}\prod_{t=1}^{T}\sqrt{1-\gamma^{(t)}{}^2}$$

- reward should depend on $\sqrt{1-\gamma^{(t)}{}^2}$
- sum of reward is optimized $\log\sqrt{1-\gamma^{(t)}{}^2}$

Reward definition

Theorem (Schapire,Singer(1998))

For ADABOOST.MH

$$R(\mathbf{f}^{(T)}) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}\left\{\ell(\mathbf{x}_i) \neq \widehat{\ell}(\mathbf{x}_i)\right\} \leq \sqrt{K-1}\prod_{t=1}^{T}\sqrt{1-{\gamma^{(t)}}^2}$$

- reward should depend on $\sqrt{1-{\gamma^{(t)}}^2}$
- sum of reward is optimized $\log\sqrt{1-{\gamma^{(t)}}^2}$
- actually, it is maximized $-\log\sqrt{1-{\gamma^{(t)}}^2}$

Reward definition

Theorem (Schapire,Singer(1998))

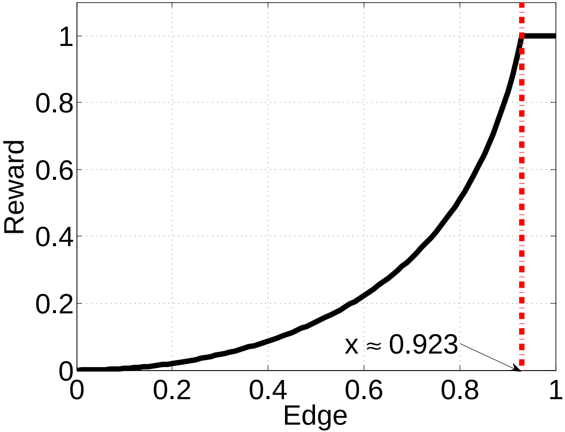
For AdaBoost.MH

$$R(\mathbf{f}^{(T)}) = \frac{1}{n}\sum_{i=1}^{n}\mathbb{I}\left\{\ell(\mathbf{x}_i) \neq \widehat{\ell}(\mathbf{x}_i)\right\} \leq \sqrt{K-1}\prod_{t=1}^{T}\sqrt{1-\gamma^{(t)^2}}$$

- reward should depend on $\sqrt{1-\gamma^{(t)^2}}$
- sum of reward is optimized $\log\sqrt{1-\gamma^{(t)^2}}$
- actually, it is maximized $-\log\sqrt{1-\gamma^{(t)^2}}$
- the rawrds must be bounded $\min\left(1, -\log\sqrt{1-\gamma^{(t)^2}}\right)$

Reward definition

$$r = \min\left(1, -\log\sqrt{1-\gamma^2}\right)$$



Convergence of AdaBoost.MH.Exp3.P

- Multiclass training error

$$R\left(\mathbf{f}^{(T)}\right) = \frac{1}{n}\sum_{i=1}^{n}\mathbb{I}\left\{\ell(\mathbf{x}_i) \neq \widehat{\ell}(\mathbf{x}_i)\right\}$$

Convergence of AdaBoost.MH.Exp3.P

- Multiclass training error

$$R(\mathbf{f}^{(T)}) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}\left\{\ell(\mathbf{x}_i) \neq \widehat{\ell}(\mathbf{x}_i)\right\}$$

- Weak learnability: $\gamma^{(t)} \geq \rho > 0$

Convergence of AdaBoost.MH.Exp3.P

- Multiclass training error

$$R\left(\mathbf{f}^{(T)}\right) = \frac{1}{n}\sum_{i=1}^{n}\mathbb{I}\left\{\ell(\mathbf{x}_i) \neq \widehat{\ell}(\mathbf{x}_i)\right\}$$

- Weak learnability: $\gamma^{(t)} \geq \rho > 0$

Convergence of AdaBoost.MH.Exp3.P

- Multiclass training error

$$R\left(\mathbf{f}^{(T)}\right) = \frac{1}{n}\sum_{i=1}^{n}\mathbb{I}\left\{\ell(\mathbf{x}_i) \neq \widehat{\ell}(\mathbf{x}_i)\right\}$$

- Weak learnability: $\gamma^{(t)} \geq \rho > 0$

Theorem

With probability at least $1-\delta$*:*
$R\left(\mathbf{f}^{(T)}\right) = 0$ *after*

$$T = \max\left(\log^2\frac{M}{\delta}, \left(\frac{4C}{\rho^2}\right)^4, \frac{4\log\left(n\sqrt{K-1}\right)}{\rho^2}\right)$$

iterations, where

$$C = \sqrt{32M} + \sqrt{27M\log M} + 16$$

Partitioning the base classifier set

- Decision stump: assign a subset to each feature:
  $\mathcal{H}_j = \{\varphi_{j,b}(\mathbf{x}) : b \in \mathbb{R}\}$
- For Decision trees and products, the naive solution is to assign a subset of features at size N to an arm $\Longrightarrow$
  Number of arms grows expoenentially
- Trees and products are modeled as sequences of decisions over the smaller partitioning used for stumps

$$\begin{tikzcd}
& & h_{j_1,b_1}(\mathbf{x}) \arrow[dl] \arrow[dr] & & \\
& h_{j_2,b_2}(\mathbf{x}) & & h_{j_3,b_3}(\mathbf{x}) & \\
\iddots & \ddots & & \iddots & \ddots
\end{tikzcd}$$

Experiments

- Stochastic bandit algorithms: assuming that the rewards are drawn from a stationary probability distribution, UCB[3], UCBV[4]
- Stochastic bandits does not fit to our setup since the *edge depends on the weights*

$$\gamma^{(t)} = \sum_{i=1}^{n} w_i^{(t)} h^{(t)}(\mathbf{x}_i) y_i$$

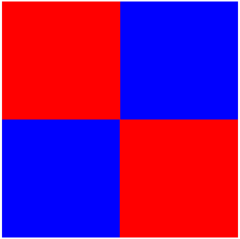
- Random feautre selection $\approx$ LazyBoost
- Synthetic data, UCI datasets and MNIST handwritten digits recognition

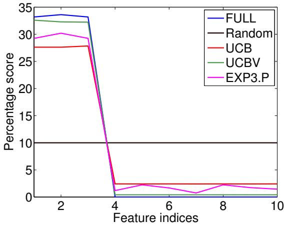
[3] Auer, P., Cesa-Bianchi, N., and Fischer, P.: Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47:235–256, 2002

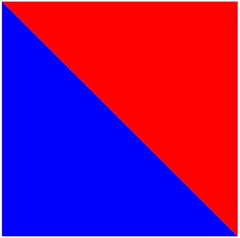
[4] Audibert, J.-Y., Munos, R., and Szepesvári, Cs.: Exploration-exploitation tradeoff using variance estimates in multi-armed bandits. *Theor. Comput. Sci.*, 410(19):1876–1902, 2009

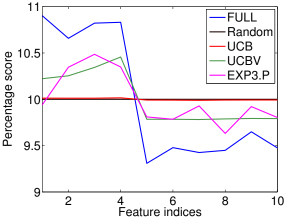
Synthetic datasets

- $d = 10$, number of useful feature $= 3$, stumps, $T = 1000$

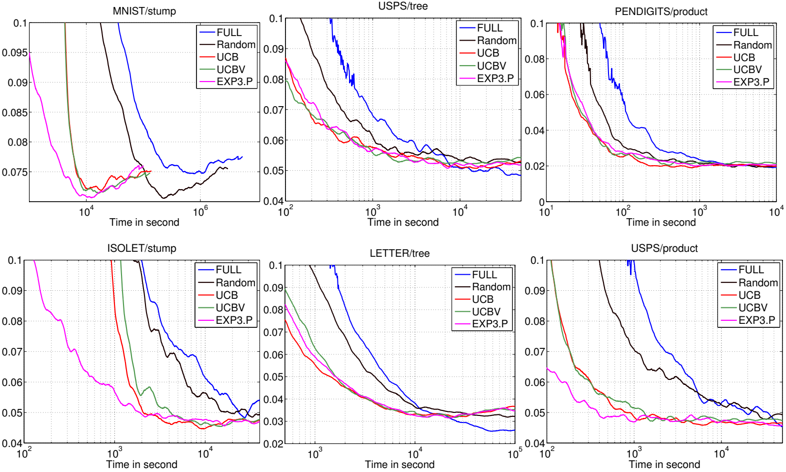








Test error vs. CPU time



Concluding remarks

- our multiboost implementation `multiboost.org` includes the bandit based setup
- ICML'10 Yahoo Learning to Rank Challenge
- top ten performance using regression-calibrated bandit boosting
- High-dimensional, structured feature spaces (linear, Haar): continuous, "metric" bandits?
- MABs are stateless, boosting is not $\rightarrow$ MDPs?

EXP3.P(η, λ, T)

for $j \leftarrow 1$ **to** M ▷ *initialization*

$\omega_j^{(1)} \leftarrow \exp\left(\frac{\eta\lambda}{3}\sqrt{\frac{T}{M}}\right)$

for $t \leftarrow 1$ **to** T

for $j \leftarrow 1$ **to** M

$p_j^{(t)} \leftarrow (1-\lambda)\frac{\omega_j^{(t)}}{\sum_{j'=1}^{M}\omega_{j'}^{(t)}} + \frac{\lambda}{M}$

$j^{(t)} \leftarrow \text{RANDOM}\left(p_1^{(t)}, \ldots, p_M^{(t)}\right)$

Receive reward $r_{j^{(t)}}^{(t)}$

for $j \leftarrow 1$ **to** M

$\widehat{r}_j^{(t)} \leftarrow \begin{cases} r_j^{(t)}/p_j^{(t)} & \text{if } j = j^{(t)} \\ 0 & \text{otherwise} \end{cases}$

$\omega_j^{(t+1)} \leftarrow \omega_j^{(t)} \exp\left(\frac{\lambda}{3M}\left(\widehat{r}_j^{(t)} + \frac{\eta}{p_j^{(t)}\sqrt{MT}}\right)\right)$