JMLR: Workshop and Conference Proceedings vol 26 (2012) 86–97 On-line Trading of Exploration and Exploitation 2

Stumping along a Summary for Exploration & Exploitation Challenge 2011

Christophe Salperwyck CHRISTOPHE.SALPERWYCK@ORANGE.COM
Orange Labs - Lannion, France
LIFL - Université de Lille 3,
Villeneuve d'Ascq, France

Tanguy Urvoy TANGUY.URVOY@ORANGE.COM
Orange Labs - Lannion, France

Editor: Dorota Głowacka, Louis Dorard and John Shawe-Taylor

Abstract

The *Pascal Exploration & Exploitation challenge 2011* seeks to evaluate algorithms for the online website content selection problem. This article presents the solution we used to achieve second place in this challenge and some side-experiments we performed. The methods we evaluated are all structured in three layers. The first layer provides an online summary of the data stream for continuous and nominal data. Continuous data are handled using an online quantile summary. Nominal data are summarized with a hash-based counting structure. With these techniques, we managed to build an accurate stream summary with a small memory footprint. The second layer uses the summary to build predictors. We exploited several kinds of trees from simple decision stumps to deep multivariate ones. For the last layer, we explored several combination strategies: online bagging, exponential weighting, linear ranker, and simple averaging.

Keywords: online learning, click through rate, content selection problem, online summary, probability decision tree, models averaging

1. Problem statement

The general *content selection problem* consists of selecting the best items to display for a given user profile in a given context to maximize the users' satisfaction (roughly represented by the number of clicks): this is a key problem of information retrieval. The considered items may be of any nature (news, ads, books, friends, etc.) and most of the time each couple (user profile, item) is described by a vector of features. Some features like *navigation history* or *cookies* are specific to the user, some features are item-specific, and some features like *textual* or *geographic distances* may involve both the user and the item.

The purpose of the *Pascal Exploration & Exploitation challenge 2011* was to evaluate online content selection algorithms by simulation from real data provided by *Adobe Test&Target* platform. The evaluation being done online, entrants had to submit a piece of Java code encoding their click prediction algorithm to the organizers. The experimental protocol is precisely described on the challenge website: http://explo.cs.ucl.ac.uk/description/. Each algorithm had to perform a sequence of iterations. For each iteration, a batch of six visitor-item pairs was extracted sequentially from the logs and given to the

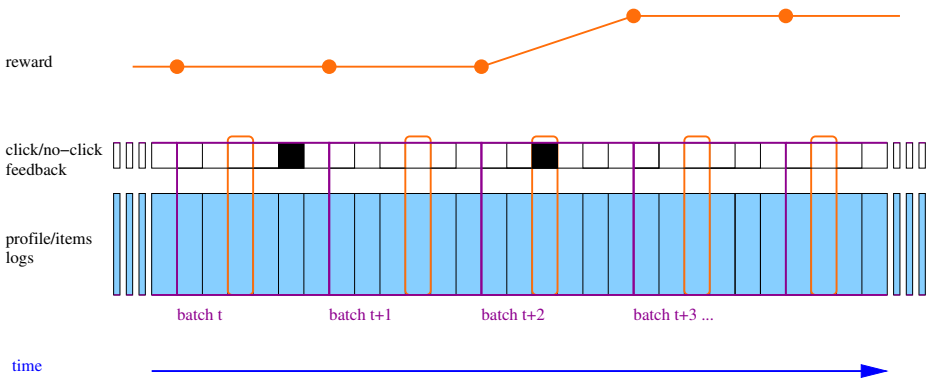

Figure 1: Challenge evaluation process

algorithm. The average click through rate being around 0.24% in the logs, most of the batches did not contain any click at all. The goal was to select the instance which was most likely to provoke a click, knowing that *click* or *no-click* feedback was only given for the selected instance (See Figure 1). The scarcity of clicks made the estimation of probabilities a difficult task. Fortunately, the goal was not to predict accurately the click probabilities of each instance but to respect their ordering according to these probabilities: the content selection problem is a ranking problem (Cortes and Mohri, 2004).

Other strong technical constraints of the challenge were the realistic time limit (100ms per iteration) and the limited space (1.7GB of memory) which motivated the use of stream summaries or incremental models.

In section 2, we detail our three-layers approach. The experiment results for the different layers configurations are given in section 3. In section 4, we discuss about the challenge offline evaluation protocol, and we conclude this article in section 5.

2. Our approach

The originality of our approach is to rely on a stream summary to build click predictions. Our methods are structured in three layers. The first layer, described in section 2.1, provides an online summary of the data stream for continuous and nominal data. The second layer, described in section 2.2, uses the summary to build incremental prediction trees, and the last layer, described in section 2.3, combines predictions trees in order to improve the final result. We experimented several methods for each layer.

2.1. Online summary

With only 100ms per round for predicting and learning and a bounded memory, the constraints of the challenge were rather tight (especially with Java garbage collector side effects). We first experimented a sliding reservoir of instances with standard batch prediction algorithms, but it consumed too much time and memory to fit the constraints of the challenge. Moreover, a basic sliding window is not appropriate for unbalanced data: a stratified (click/no-click) reservoir or an adapted stream summary seemed to be more promising options. We chose to start from MOA: Massive Online Analysis (Bifet et al., 2010). MOA

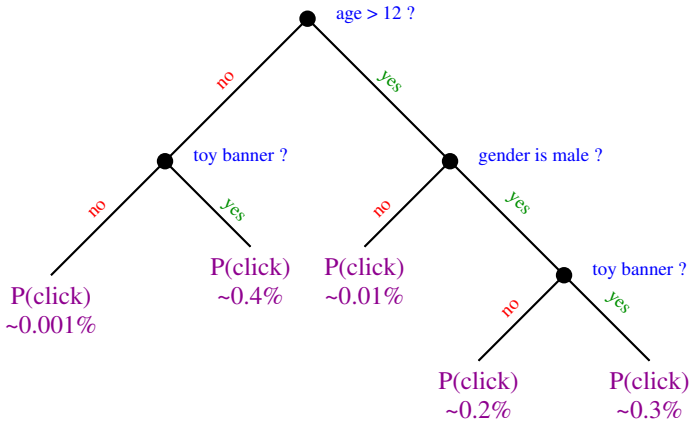

Figure 2: A toy illustration of multivariate probability estimation trees. Nodes are unfolded when there is enough data to estimate CTR accurately.

is developed by the Waikato University and is designed to work online and process data streams. We adapted their framework and classifiers to fit the challenge needs. Depending on the nature of the features: numerical or nominal, we built two kinds of summary.

Numerical features summary

In order to build probability estimation trees, we needed to keep the click/no-click counts by intervals for each numerical feature. This kind of summary is called a *quantile summary*. Greenwald and Khanna (2001) proposed an online merge-prune algorithm to maintain such a summary. Their algorithm is based on an array of tuples $< v_i, g_i, \Delta_i >$ where: v_i is a value seen in the stream, g_i is the number of values seen between v_{i-1} and v_i, and Δ_i is the maximum possible error on g_i. This algorithm has many interesting properties: it maintains online an approximate equal-frequency histogram with strong error guarantees (ϵ-approximate quantile summary for a given memory size), it uses a bounded memory space, and it is insensitive to data arrival order and distribution. We adapted the algorithm in order to store the counts per classes: the g_i counter was replaced by a counter per interval and per class: $g_{i,c}$.

Nominal features summary

A perfect nominal summary would keep click/no-click counts for all feature values. Unfortunately the number of nominal values is not supposed to be bounded. For instance some features can represent *client ids, cookies, city names, etc.* As we want a memory bounded summary we can only afford to focus on frequent values. The solution we used is based on hashing: nominal values are hashed and put into a fixed number of buckets, the click/no-click counts being stored for each bucket. We used a single hashing function, but in order to reduce errors several hashing functions may be combined as proposed in the count-min sketch algorithm (Cormode and Muthukrishnan, 2005).

2.2. Base predictors

The predictors we used were probability estimation trees, also known as PETs (Provost and Domingos, 2003). Their output is a click through rate estimate (CTR). We illustrate these

kinds of predictors in Figure 2. To estimate CTR in the leaves, we used an optimistically biased shrinkage-estimator:

$$\hat{P}(click) = \frac{n_{click} + m \cdot prior}{n_{total} + m}$$

where the prior was set to 0.0025 (a bit higher than global CTR) and the shrinkage parameter m was set to low values (less than 1000) in order to force a fast decreasing of the prior impact. The impact of these two parameters on performance seems to be slight. Depending on the feature types, numerical or nominal, we evaluated different strategies to estimate CTR.

Numerical features

We first started with decision stumps i.e. single level probability estimation trees. A decision stump is defined by two parameters: the features id i and the decision threshold θ. Figure 3 shows how the decision stump is built on a quantile summary.

$$\hat{P}(click \mid x_i) = \begin{cases} \hat{P}(click \mid x_i \leq \theta) & \text{if } x_i \leq \theta \\ \hat{P}(click \mid x_i > \theta) & \text{else} \end{cases}$$

The split criterion we used was the Gini impurity I_G (Entropy gives similar results). Given a split value θ on a feature i the Gini impurity is computed on two intervals (left: $x_i \leq \theta$ and right: $x_i > \theta$) using the previous estimates:

$$I_G(i) = \left(\hat{P}(click \mid x \leq \theta) - \hat{P}^2(click \mid x \leq \theta)\right) + \left(\hat{P}(click \mid x > \theta) - \hat{P}^2(click \mid x > \theta)\right)$$

The best split value is the value θ giving the lowest $I_G(i)$.

Then we explored deeper univariate trees by using a gradual unfolding. The tree is first a simple stump but new nodes are added when there is *'enough'* samples in the leaves for a correct CTR estimate (we fixed this amount to 1000 instances). To enhance the ranking quality, we used a bottom-up smoothing similar to the one proposed by Provost and Domingos (2003). This smoothing is defined recursively by:

$$\hat{P}(click \mid \text{node}) = \frac{n_{click} + m \cdot \hat{P}(click \mid \text{parent node})}{n_{total} + m}$$

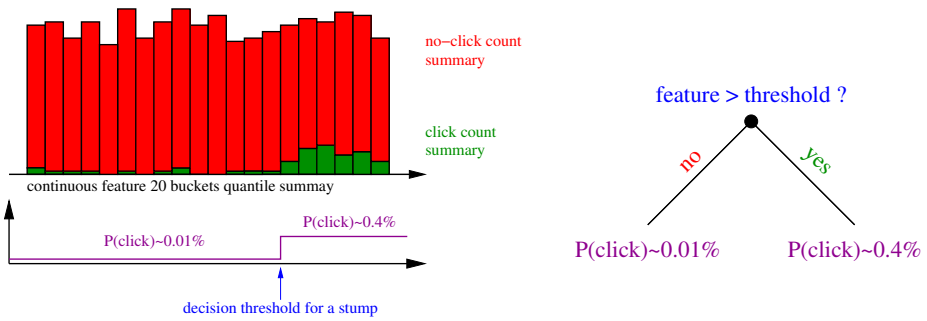

Figure 3: An illustration of how decision stumps are stacked on quantile summaries: the summary maintain a fine grained count of click/no-click for each bucket and the decision stump provides a rougher, but more reliable, CTR estimate.

Nominal features

We tried two different approaches. The first one transforms nominal values into numerical ones with a hash function and reuses the numerical-feature approach *(HStump)*. A combination of several hashing functions for the same feature avoids "unlucky orderings". The second option is to build a stump which contains one leaf for the most discriminant value and one leaf for all the other values *(1 vs. All)*.

Multivariate trees

Because it relies on a single feature, an univariate tree only requires a single summary: this simplifies the implementation of the system and reduces drastically the memory footprint. On the other hand, only multivariate models are able to catch interactions between features.

A multivariate tree needs to embed one summary per feature into each leaf. These summaries allow the algorithm to compute statistics to transform a leaf into a new decision node. The memory consumption (mainly taken by summaries) is proportional to the number of leaves. When a decision tree is built offline all the instances are available and the first node splits on the feature that maximizes the criterion. The next nodes are created in a similar way but only with the data in the subpartition induced by the parent node. In this challenge, instances arrive online and are collected by summaries. As a split (a node) is never reconsidered, we would like to guarantee that splits are made with a strong confidence. Domingos and Hulten proposed to use the Hoeffding bound to build trees online (Domingos and Hulten, 2000). These trees are called Hoeffding trees and are set up with a split confidence parameter δ.

2.3. Online combination of predictors

Without prior knowledge about features, we could not assume that all base predictors were accurate, especially the univariate ones based on poorly informative features. For this reason we looked for a combination system to favor the best predictors. This section describes the systems we experimented.

At each time step t each base predictor outputs a score $x_{i,t} \in [0,1]$. Let $\mathbf{x}_t = (x_{1,t}, \ldots, x_{N,t})$ be the predictors output vector at time t. Predictor's outputs are linearly combined in order to produce a score $\mathbf{w}_t \cdot \mathbf{x}_t \in [0,1]$ where $\mathbf{w_t} = (w_{1,t}, \ldots, w_{N,t})$ is a voting weight vector which verifies $|\mathbf{w_t}| = \sum_i |w_{i,t}| = 1$.

Baseline Our baseline was a simple averaging giving the same weight to all predictors.

Exponential weighting The main principle of this combination strategy is to reweight exponentially the predictors according to their past performance:

$$w_{i,t+1} \leftarrow w_{i,t} e^{\eta Reward_{t+1}(x_{i,t+1})} / Z_{t+1}$$

where Z_{t+1} is such that $|\mathbf{w_{t+1}}| = 1$. With a proper tuning of the parameter η, this combination strategy provides guarantees to converge to the best predictor performance (Cesa-Bianchi and Lugosi, 2006).

Online linear ranker Let $\mathcal{X}_0$, $\mathcal{X}_1$, be respectively the set of negative (no-click) and positive (click) instances. Set $\mathbf{w_t}$ to optimize *Area Under the roc Curve* (AUC) is equivalent

to minimize pairwise ranking loss:

$$RLoss(\mathbf{w}) = \sum_{(\mathbf{x_0},\mathbf{x_1})\in\mathcal{X}_0\times\mathcal{X}_1} [\mathbf{w}\cdot\mathbf{x_0} \geq \mathbf{w}\cdot\mathbf{x_1}]$$

Because the $\mathbf{x}_0 - \mathbf{x}_1$ are bounded, we can approximate the ranking loss with a linear function. This allows us to separate pairs:

$$RLoss(\mathbf{w}) \leq \sum_{(\mathbf{x_0},\mathbf{x_1})\in\mathcal{X}_0\times\mathcal{X}_1} 1+\mathbf{w}\cdot(\mathbf{x_0}-\mathbf{x_1}) = |\mathcal{X}_0|\cdot|\mathcal{X}_1|+\mathbf{w}\cdot\left(|\mathcal{X}_1|\cdot\sum_{\mathbf{x_0}\in\mathcal{X}_0}\mathbf{x_0} - |\mathcal{X}_0|\cdot\sum_{\mathbf{x_1}\in\mathcal{X}_1}\mathbf{x_1}\right)$$

The minimal linear loss is reached at $\lambda\mathbf{w}^*$ for any real λ with:

$$\mathbf{w}^* = \frac{1}{|\mathcal{X}_1|}\cdot\sum_{\mathbf{x_1}\in\mathcal{X}_1}\mathbf{x_1} - \frac{1}{|\mathcal{X}_0|}\cdot\sum_{\mathbf{x_0}\in\mathcal{X}_0}\mathbf{x_0}$$

Hence, for each t we simply maintain the centroids as in the Rocchio relevance feedback system (Rocchio, 1971):

- $\mathbf{w_{0,t}} = 1/|\mathcal{X}_0|\cdot\sum_{\mathbf{x}\in\mathcal{X}_0}\mathbf{x}$ (no click centroid)
- $\mathbf{w_{1,t}} = 1/|\mathcal{X}_1|\cdot\sum_{\mathbf{x}\in\mathcal{X}_1}\mathbf{x}$ (click centroid)
- $\mathbf{w_t} = (\mathbf{w_{1,t}} - \mathbf{w_{0,t}})/Z_t$ where Z_t such that $|\mathbf{w_t}| = 1$ ($w_{i,t}$ may be negative)

Online bagging Another kind of averaging that we experimented was online-bagging as proposed by (Oza, 2005). This bagging duplicates base classifiers several times and weights instances according to a Poisson law.

3. Experiments

Most of our experiments were done on the Adobe dataset but we did few experiments on our own advertising dataset (Orange web-mail). The characteristics of these two datasets are summarized in Table 1.

	Adobe (Challenge part 1)	**Orange web-mail**
Size	3M instances (i.e. 500K batches of 6)	6M instances (i.e. 1M batches of 6)
Features	99 continuous + 21 nominals *last nominal feature is "option id"*	33 nominals
Average CTR	~ 0.0024	~ 0.0005

Table 1: Datasets characteristics

submission	summary	base-predictors	combination	score
Random				1127 ± 41
LinRankP	GK-4/CMS-1	1R (proba. output)	LRank-120	~ 1600
LinRankB	GK-4/CMS-1	1R (0/1 output)	LRank-120	~ 1800
RHT	GK-100/CMS-1	200 trees (10 features)	OZA-200	~ 1900
Stumps1vAll	GK-100/CMS-1	Stump/1 vs. all	AVG-120	~ 2000
UTree	GK-100/CMS-1	UTree/HStump	AVG-120	~ 2030
Cnrs				$\sim$ **2040**
SimpleStumps	GK-100/CMS-1	Stump/HStump	AVG-120	~ 2050
Orange Best	GK-100/CMS-1	800 stumps + 200 RHT	OZA-1000	$\sim$ **2080**
Inria				$\sim$ **2200**
Perfect				6678

Table 2: Phase 1 results for the Adobe dataset
GK-100: Greenwald & Khanna (100 tuples), CMS-1: count-min sketch (1 hash),
1R: single rules $x_i \in \ ]a, b]$? 1 : 0*, HStump: hash then split like stump,*
RHT: random Hoeffding trees, UTree: univariate tree,
LRank-120: online linear ranker (120 predictors), AVG-120: simple averaging (120 pred.),
OZA-200: Oza bagging (instance weighting for 200 predictors)

3.1. Challenge results

Our experiments on Adobe dataset are detailed in Table 2. In this table, INRIA (1^{st} in this challenge), CNRS (3^{rd}), random and perfect predictor results are also presented.

We ran several random predictors with different seeds to estimate the standard reward deviation (± 41). The linear ranker stacked on probabilistic predictors (LinRankP) scored around 1600 while the one stacked on simple binary rules (LinRankB) scored around 1800. Similar results were obtained with exponential weighting. The combination strategies being difficult to tune, we backed up to a simple averaging and – surprisingly – obtained a significant improvement: around 2000 for decision stumps (Stumps1vAll), and 2030 for univariate trees (UTree). There seems to be a trade-off between the expressiveness of the base predictors and the power of the combination strategy.

Our best score using only univariate base classifiers was obtained with stumps for numerical features and HStump for the nominal ones (~ 2050 - SimpleStumps). Finally we have got our best result with a combination of 80% stumps and 20% Hoeffding trees with Oza bagging (~ 2080 - Orange Best). However, due to the multiplication of classifiers, the computing time of this model was much heavier than other submissions, especially the one with basic stumps. The 200 Hoeffding trees alone scored around 1900.

The evolution of the learning process is illustrated in Figure 4. This figure presents two sub-figures: the top one gives the evolution of the cumulated clicks and the bottom one the instantaneous CTR estimate with an exponential smoothing. Both curves are function of the batch number. The two random predictors' curves give an indication of the variability for a given strategy. The best entrant's CTR were around 2 times better than random.

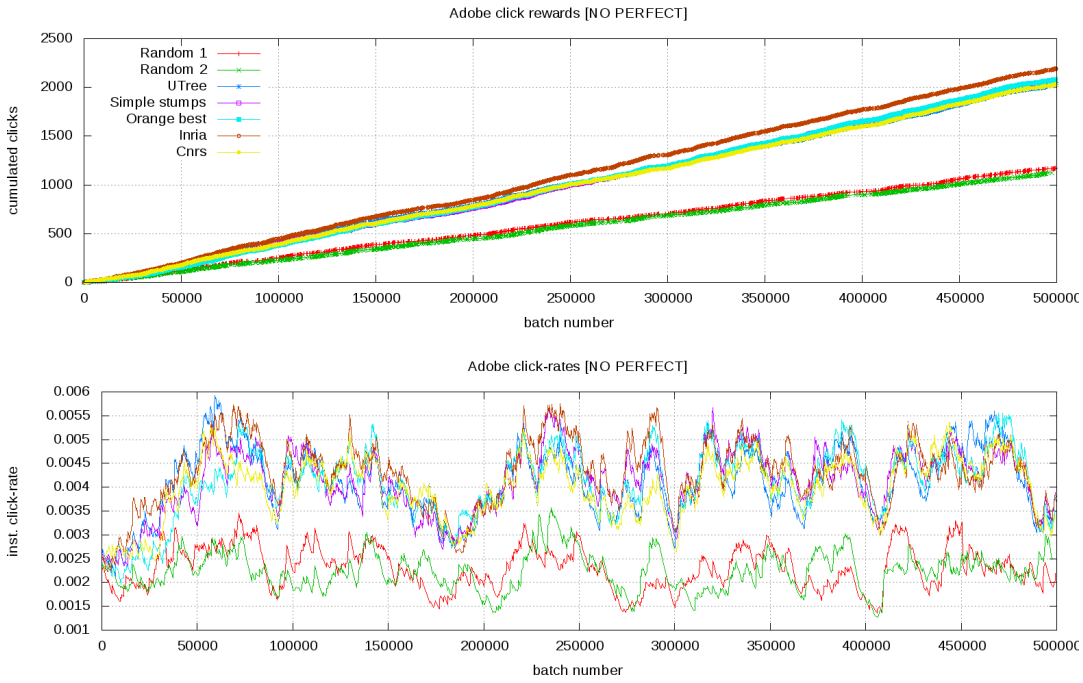

Figure 4: Phase 1 reward curves on Adobe dataset

3.2. Side Experiments

In order to deepen our understanding of the problem we did few side experiments on both Adobe and our own Orange web-mail dataset.

Offline analysis After the first phase of the challenge, a part of Adobe dataset was given to the entrants. Therefore we were able to refine our study of the predictors' behavior with an offline analysis on the whole dataset. This offline dataset analysis exhibited a maximum of 37 values for nominal features. For numerical features, we used the Khiops software (`www.khiops.com`) to perform an optimal discretization according to the MODL method (Boullé, 2006). This discretization resulted in one to three intervals depending on the feature. With 100 intervals for each of the 120 features our summary was small in memory (∼200kB) but fine enough to catch the data patterns.

To evaluate the quality of the online density estimation, we compared the discretization and click through rate estimates at different steps of the learning process to the MODL estimate on the whole dataset. Figure 5 illustrates this experiment. As expected, the online discretization converges to the MODL discretization and the average click through rate is higher.

Concept drift detection Another experiment was to check for *concept drifts* (i.e. abrupt changes of data distribution). Therefore we performed a comparison between a model which stops learning after 250,000 batches and the same model which keeps learning to the end. Figure 6 presents this experiment. There is no significant "drift" of the clicks distribution.

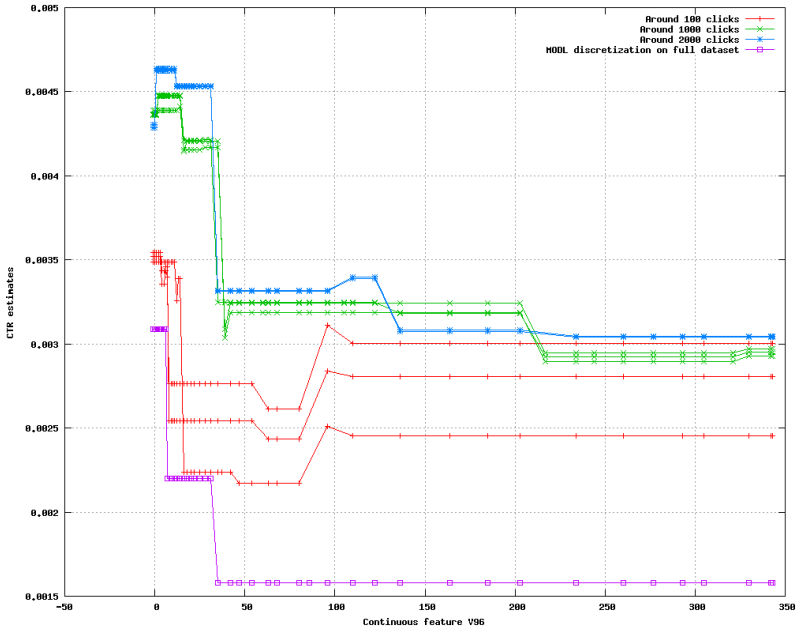

Figure 5: Discretization and CTR estimates at different steps for continuous feature 96 for different seeds.

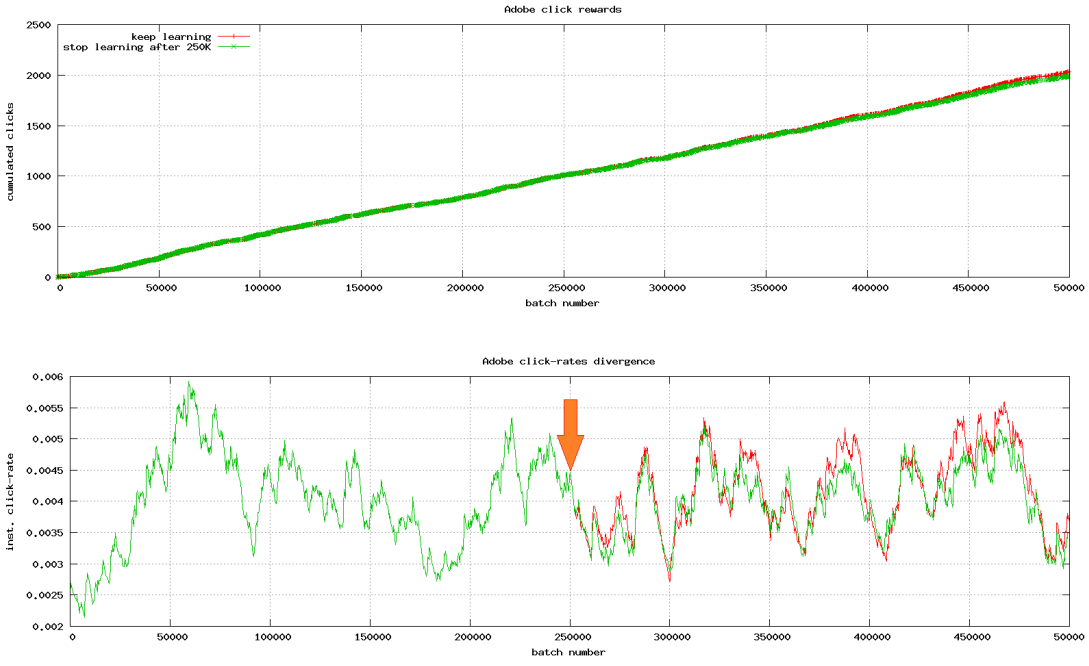

Figure 6: Deviation from a static model

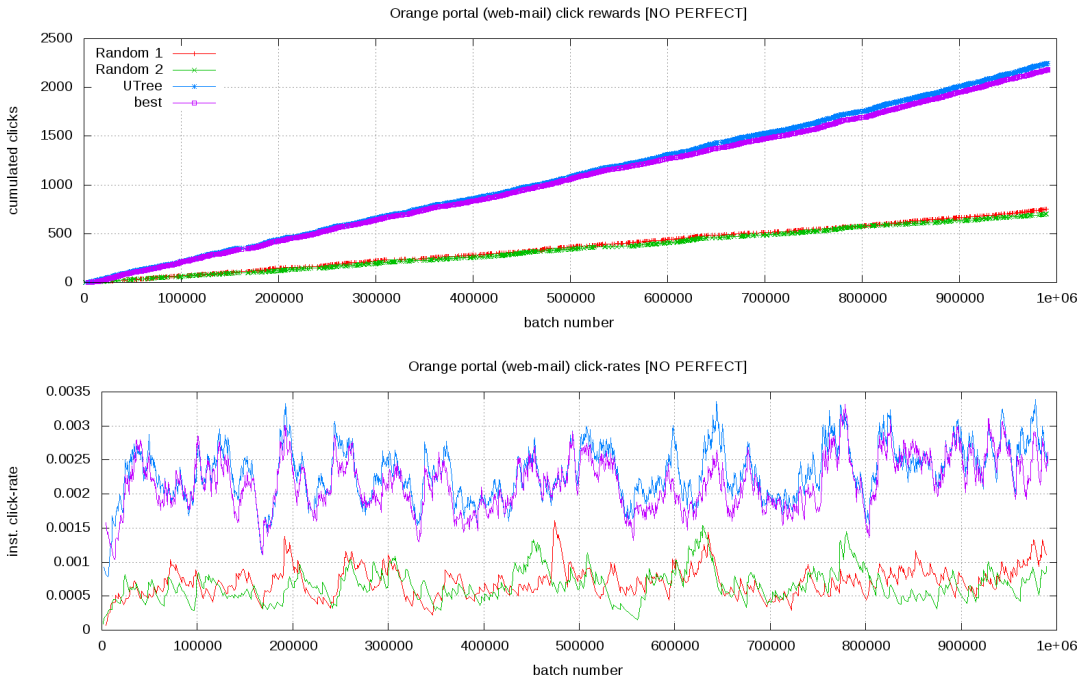

Figure 7: Challenge protocol applied on Orange dataset

Orange web-mail dataset The challenge protocol was also applied on our own – Orange web-mail – dataset. Figure 7 presents the results of two of our best predictors on this dataset. We obtain a click through rate around 3 times better than random policies.

3.3. Recommended algorithm

If we had to recommend a system to address a similar problem we would choose the following three layers because they performed well on both Adobe and Orange datasets with a small memory and time consumption:

1. Quantile summary for numerical features and Count-min Sketch for nominal ones;
2. Univariate PETs (one for each feature);
3. Averaging of the univariate predictors.

4. About Offline Evaluation Protocol

This challenge was one of the first to propose an evaluation of online content selection algorithms but the interaction between content selection and click feedback makes this evaluation difficult. The best solution is certainly to evaluate the different algorithms on similar subsets of real traffic but the cost of this method is prohibitive. A simulation from purely artificial model is not satisfying either: the most convenient solution is to evaluate offline from logged data.

The evaluation scheme proposed by this challenge was simple and made an efficient use of the logs: at each round a batch of 6 (profile, item, reward) instances was read from logs and submitted to the entrants' algorithms. The unrealistic aspect of this method was the possibility given for the algorithms to choose between different profiles: this is not the case for the real content selection problem where the profile is fixed. From our own experiments with Orange portal logs, some profile features are very discriminant. For instance the click through rate on Orange home page is five time higher than on other pages: the trained model only learns to select home page versus other pages. To reduce this side-effect, we only extracted logs from a single page (the web-mail).

For future challenges, if the log size is important and the number of items is limited, one may also use rejection sampling: at each round we take one log line and emulate all missing options features. If the selected option is the one of the log: we evaluate this instance based on known click/no-click results. If selected option is unknown we can reject (ignore) this instance. This method, well studied by (Langford et al., 2008; Li et al., 2010, 2011), rejects too many instances when the number of options is high. On the other hand, its great advantage is that it offers an unbiased estimate of the real algorithm performance when applied on uniform logs.

5. Conclusion

This challenge was difficult and interesting for both technical and scientific reasons. The need to embed the code into the challenge framework with limited resources required some technical investment from the entrants. On the other hand, the scarcity of clicks, the high level of noise, and the partial feedback which characterize the content selection problem required to explore innovative approaches.

The key ingredient of all our submissions was the elaboration of stream summaries: the Greenwald and Khanna stream summary for numerical features and the count-min sketch for nominal ones. From these summaries, we derived different kinds of click-predictors. Our best model was an online flavor of the *random forest* (although the term *"random tundra"* may appear more appropriate if we take into account the average tree size). A simpler model based only on decision stumps also proved to be remarkably efficient with lighter resource consumption. Some experiments on our own Orange dataset confirmed this phenomenon: the need for exploration, the high level of noise, and the click scarcity sharpen the need for very stable – hence simple, or strongly regularized – ranking models. We also observed a clear trade-off between the expressiveness of the base predictors and the power of the combination strategies.

Acknowledgments

We would like to thank the organizers for these very interesting challenge and workshop. We are already waiting for the next "Exploration/Exploitation" opus. We would also like to thank Sylvie Tricot for providing us the Orange advertising dataset and, Carine Hue and Vincent Lemaire for their comments. This work was partially supported by the FUI/FEDER Project DIFAC.

References

A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer. MOA: Massive online analysis. J. Mach. Learn. Res., 99:1601–1604, August 2010. ISSN 1532-4435.

M. Boullé. MODL: A Bayes optimal discretization method for continuous attributes. Machine Learning, 65(1):131–165, 2006.

N. Cesa-Bianchi and G. Lugosi. Prediction, Learning, and Games. Cambridge University Press, 2006.

G. Cormode and S. Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. Journal of Algorithms, 55(1):58–75, April 2005. ISSN 01966774.

C. Cortes and M. Mohri. AUC optimization vs. error rate minimization. In Advances in Neural Information Processing Systems. MIT Press, 2004.

P. Domingos and G. Hulten. Mining high-speed data streams. In Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 71–80. ACM New York, NY, USA, 2000.

M. Greenwald and S. Khanna. Space-efficient online computation of quantile summaries. ACM SIGMOD Record, 30(2):58–66, June 2001. ISSN 01635808.

J. Langford, A. Strehl, and J. Wortman. Exploration scavenging. In Proceedings of the 25th international conference on Machine learning, ICML '08, pages 528–535, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-205-4.

L. Li, W. Chu, J. Langford, and R.E. Schapire. A contextual-bandit approach to personalized news article recommendation. In Proceedings of the 19th international conference on World wide web, WWW '10, pages 661–670, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-799-8.

L. Li, W. Chu, J. Langford, and X. Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In Proceedings of the fourth ACM international conference on Web search and data mining, WSDM '11, pages 297–306, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0493-1.

N.C. Oza. Online Bagging and Boosting. 2005 IEEE International Conference on Systems, Man and Cybernetics, pages 2340–2345, 2005.

F. Provost and P. Domingos. Tree induction for probability-based ranking. Machine Learning, 52(3):199–215, 2003.

J. Rocchio. Relevance Feedback in Information Retrieval. In G Salton, editor, SMART Retrieval System Experiments in Automatic Document Processing, chapter 14, pages 313–323. Prentice Hall, 1971.