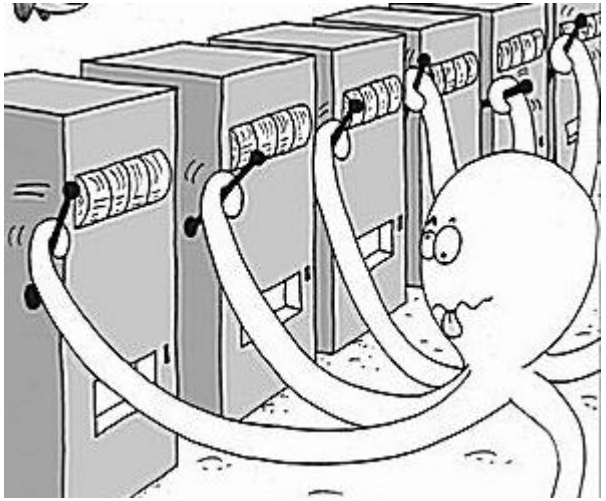
Bandits in Information Retrieval

Dorota Glowacka
University of Edinburgh
dorota.glowacka@ed.ac.uk

The Multi-Armed Bandit Problem

You are in a casino. There are many different slot machines (known as 'one-armed bandits', as they are know for robbing you), each with a lever (an arm). You think that some slot machines pay out more than others, so you'd like to maximize this. You only have a limited amount of resources – you can pull each arm a limited number of times. Your goal is to walk out of the casino with most money.

If you knew which lever would pay out the most, you would pull that lever all day.



The question is: *how do you learn which slot machine is the best and get the most money in the shortest amount of time?*

Explore – Exploit Dilemma

You have no initial knowledge about the payoff of the machines and so at each trial you face the following trade-off:

- Play the machine that has given you the highest reward so far (***exploit***)
- Play other machines in the hope of finding one that will give you a higher reward (***explore***)

Exploration – Exploitation Dilemma

- Online decision making involves a fundamental choice:

Exploitation: make the best decision given current information

Exploration: gather more information

- The best long-term strategy may involve short-term sacrifices
- Gather enough information to make the best overall decision

Examples

- Online Advertising

Exploit: Show the most successful advert

Explore: Show a new advert

- Restaurant Selection:

Exploit: Go to your favourite restaurant

Explore: Try a new restaurant

- Oil drilling

Exploit: Drill at the best known location

Explore: Drill at a new potential oil field

Practical Applications

- Dynamic allocation of resources (which project to work on given the uncertainty about the difficulty and payoff of each project)
- Clinical trials (investigating effects of different experimental treatments while minimizing patient loss)
- Financial portfolio design
- Adaptive routing (to minimize delays in the network)

The Multi-Armed Bandit

- A multi-armed bandit is a tuple $\langle A, R \rangle$
- *A* is a known set of *m* actions (or arms)
- $R^a(r) = \mathrm{P}[r \mid a]$ is an unknown probability distribution over rewards
- At each step *t*, agent selects an action $a_t \in A$
- The environment generates a reward r_t
- The goal is to maximize cumulative reward $\sum_{\tau=1}^{t} r_\tau$

Regret

The *regret* is the difference between the sum of *rewards r obtained so far* and the reward sum associated with *optimal strategy*.

Let $\mu_1, \ldots, \mu_m$ be the mean values associated with the rewards of each arm.

The regret after *t* rounds is:

$$\rho = t\mu^* - \sum_{\tau=1}^{t} r_\tau$$

where μ^* is the maximum reward mean.

Approaches to the Bandit Problem

- Regret is defined in terms of the average reward.
- If we can estimate average reward, then we can minimize regret.
- Let's take the action with the highest average reward:
  - Assume two actions (arms)
  - Action (arm) 1 has reward of 1 with probability 0.3 and otherwise the reward is 0
  - Action (arm) 2 has reward of 1 with probability 0.7 and otherwise has reward of 0
  - We play the first arm and get reward of 1
  - Next, we play the second arm and get reward 0
  - Now the average reward of arm 1 is higher than that of arm 2.

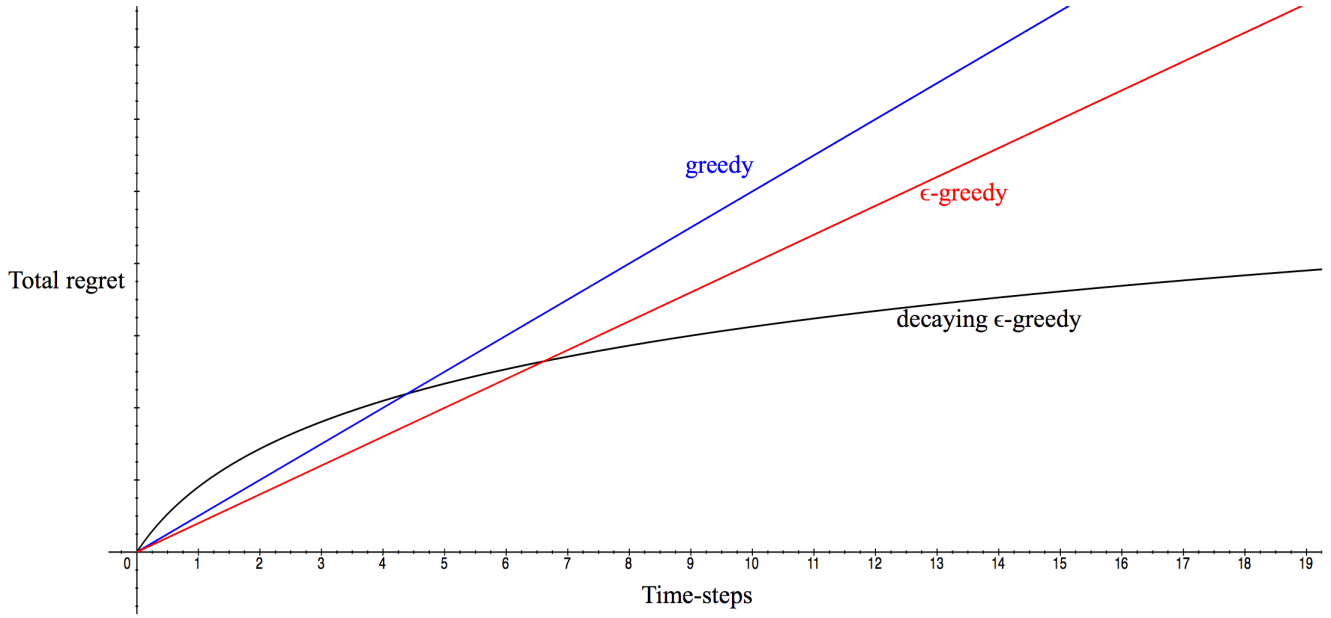
Greedy Algorithm

- After playing each arm once and observing the reward, we might conclude that the arm 1 gives us a better reward and so play for the rest of the game.
- The greedy algorithm selects arm with the highest value:

$$a_t^* = \underset{a \in A}{\operatorname{argmax}}\, \mu_t(a)$$

- The greedy algorithm can lock onto a suboptimal action forever and exploit it forever.

Regret



If you *only* explore, then the regret will be linear.

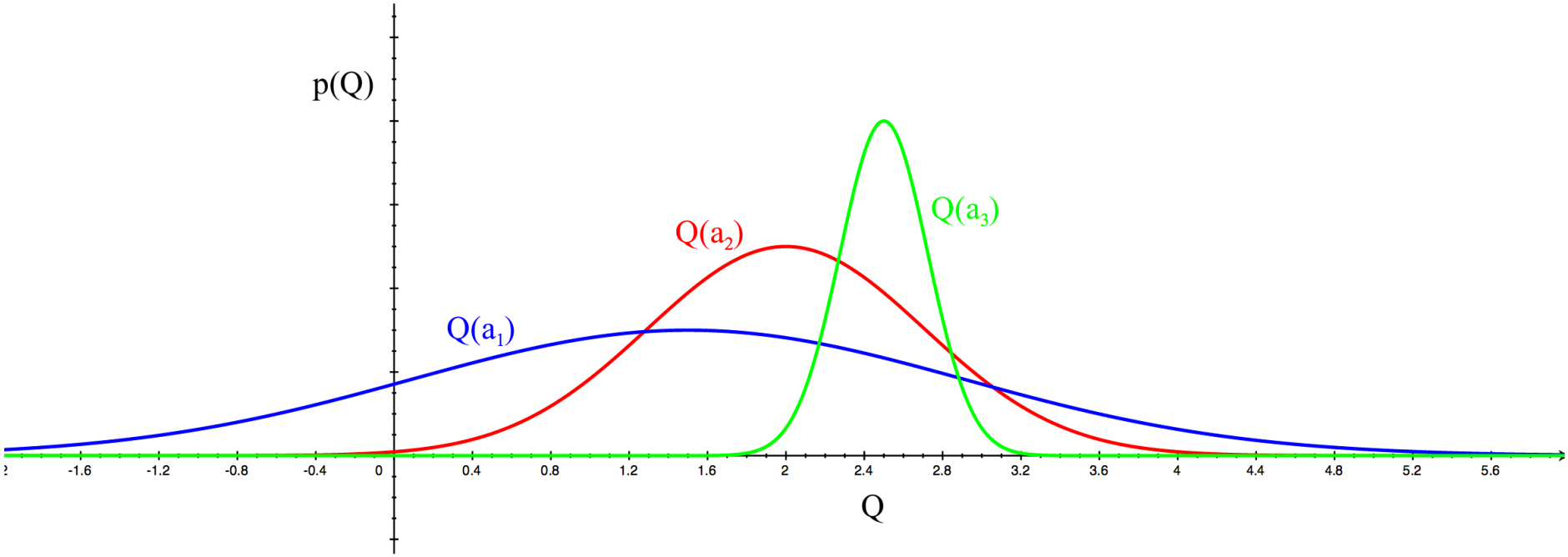
If you *never* explore, then the regret will be linear.

Is it possible to achieve sublinear total regret?

Optimism in Face of Uncertainty

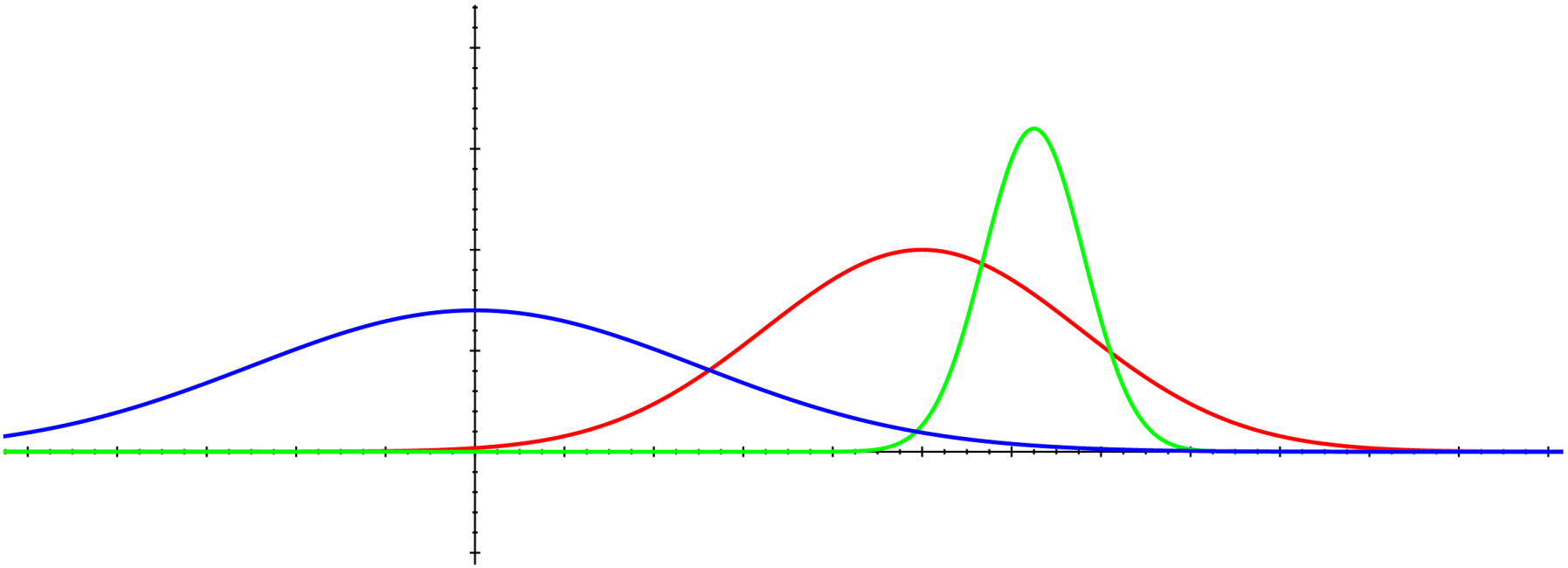
- The problem with the greedy algorithm is that it is *too certain* of its estimates – we should not conclude what the average reward of a given arm is based on one trial.
- The more *uncertain* we are about the reward of an action (arm), the more important it is to *explore* that action.
- It could turn out to be the best action!

Optimism in Face of Uncertainty



Which action should we choose?

Optimism in Face of Uncertainty



After observe reward of the blue action, we are less uncertain about its value.

UCB1

- For each action *a* record the average reward *μ(a)* and the number of times we have tried it *N(a)*. We write *t* for the total number of actions we have tried so far.
- Try the action that maximizes $\mu(a) + \sqrt{\frac{2 \log t}{N_t(a)}}$
- It is quick and easy to implement.

Linear Bandits

- What happens if the number of arms is very large and we cannot try all of them?
- Take advantage of similarity between arms, i.e. playing one arm will give you information about similar arms thus reducing the amount of exploration required.
- The assumption is that there is a similarity structure between arms.

Contextual Bandits

- In each round:

(1). **Observe** context vector x_a of each arm $a \in \mathcal{A}$

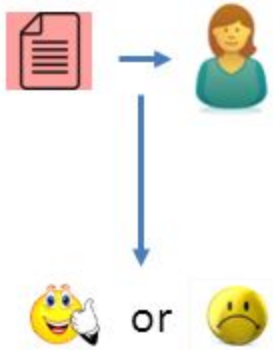
(2). **Picks** an arm a_t

Upper Confidence Bound

$$a_t = \arg max_a (\hat{r}_{a,t} + C_t(a))$$

Estimated reward (exploitation)

Confidence interval (exploration)



(3). **Receive** corresponding reward $r_{a_t,u}$

(4). **Update** model

Diverse Rankings with Bandits

- *Probabilistic ranking* advocates ranking documents in order of *decreasing relevance* to a *query*.
- Result: *similar documents* ranked at *similar positions*, while *users* may prefer *diverse* set of results.

- Problem: users click on few results (increased *abandonment*)

Diverse Ranking

- We want to learn an optimally diverse ranking of documents for a given query and maximize user satisfaction.
- In each round, a user arrives and an algorithm outputs a list of *k* results. The user scans the documents top-down and clicks on the first relevant one.
- Each slot *i* is examined by the user only if the documents in the higher slots are not clicked.

Rank 1: ✗
Rank 2: ✗
Rank 3: ✗
Rank 4: ✗
Rank 5: ✓
STOP
Rank 6
…
…
Rank k:

Ranked Bandits

- Run a bandit for each rank
- Each bandit maintains a value for every document in collection
- Bandits corresponding to each rank are treated independently
- If B_j and B_i select the same document, then a random document is selected for rank j

B_1 – d_1, d_2, d_3, d_4, … d_n
B_2 – d_1, d_2, d_3, d_4, … d_n
B_k – d_1, d_2, d_3, d_4, … d_n

Rank 1: d_2 ✗
Rank 2: d_4 ✓
Rank k: d_1 ✗

B_1 – 0, 0, 0, 0, …, 0
B_2 – 0, 0, 0, **1**, …, 0
B_k – 0, 0, 0, 0, …, 0

Ranked Bandits Algorithm

- No relevance judgments from experts required for training.
- Accounts for dependencies between documents.
- The algorithms learns a utility value for each document at each rank, maximizing the probability that a new user of the search system will find at least one relevant document within the top *k* positions.
- Equivalent to an online learning problem with no distinction between training and testing phases.

R.Kleinberg, F. Radlinski, T. Joachims: *Learning Diverse Rankings with Multi-Armed Bandits*. ICML 2008.

What if the Number of Documents is Large?

- Bandit algorithms are ideal for online settings with exploration/exploitation trade-offs but are impractical at web scales.
- Exploit document *similarity* and ranking *context* to optimize the convergence rate of the bandit algorithm.
- To exploit the *context*, we can factor in *conditional clickthrough rates* (user skips a set of documents) and *correlated clicks* (probability that two documents are ir/relevant to the user).

Ranked Bandits in Metric Spaces

- *Document model*: web documents are organised into *tree* hierarchies, where *closer pairs* are *more similar* and each document *x* is a leaf in the tree.
- The tree is a topic taxonomy on documents such that the *click event* on each *subtopic* is obtained from that on the *parent topic* via probability mutation (distance between child and parent).

Ranked Bandits in Metric Spaces

- The algorithm maintains a set of active strategies of the form $A(u, u_c)$, where u is a subtree in the document tree and u_c is a subtree in the context tree.
- We selectively refine the grid in promising areas and maintain a set of active strategies that partition the space of all (document, context) pairs.
- In each round, a *context h* arrives and the *active strategy* with h in u_c and with the maximal index is selected, and then a document from subtree u is selected at random.

- A. Slivkins, F. Radlinski, S. Gollapudi: *Learning Optimally Diverse Rankings Over Large Document Collections*. ICML 2010.
- A. Slivkins, F. Radlinski, S. Gollapudi: *Ranked Bandits in Metric Spaces: Learning Diverse Rankings over Large Document Collections*. JMLR 14 (2013).

Interactive System Optimization

- Conventional approaches to system optimization require optimizing a proxy measure (precision, recall, etc.), which requires *expensive manual relevance judgments*.
- These judgments often *ignore* the *user context* and so may not be very helpful with optimizing the search experience for a specific user.
- We can use *implicit feedback* gathered directly from the user.
- Users tend to be quite good at providing *judgments between two sets* of results, e.g. lists of retrieval results or product attractiveness.
- *Problem*: How can a learning algorithm access the utility a user sees in a set of results?

Dueling Bandits Problem

- Actions are restricted to noisy comparisons between pairs of strategies (bandits).
- Unlike MAB that requires an absolute reward, dueling bandit assumes only noisy binary feedback about relative reward of two strategies (bandits).
- Each iteration comprises a noisy comparison (duel) between two bandits.
- Regret corresponds to the fraction of users who prefer the best bandit over the selected ones.

Dueling Bandits Problem

Any single comparison between two bandits *b* and b_1 is determined independently of all other comparisons with probability:

$$P(b \succ b_1) = \frac{1}{2} + \varepsilon(b, b_1)$$

where $\varepsilon(b, b_1) \in \left[\frac{-1}{2}, \frac{1}{2}\right]$ is a measure of distinguishability between the two bandits. *P* refers to the fraction of users who prefer the results produced by *b* over those over b_1.

- The regret is:

$$R = \sum_{t=1}^{T} \varepsilon(b^*, b_t) + \varepsilon(b^*, b_t^1)$$

where b_t are bandits selected at time *t* and *b** is the best performing bandit.

Dueling Bandits Problem

How to select the next bandit?

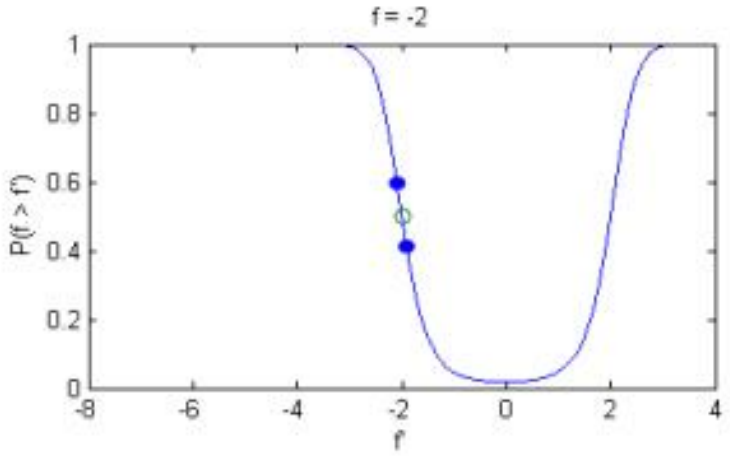
Gradient Descent – candidate b_t and compares it with the neighbouring bandit b'_t along a random direction u_t. If b'_t wins, then the update is taken along u_t and projected back into the space of retrieval functions (bandits).

Interleaved Filter – maintain candidate b and compare it with all the remaining bandits via round robin scheduling (interleaving). Repeat until only one bandit remains.

Y.Yisong, T. Joachims: Interactively Optimizing Information Retrieval Systems as a Dueling Bandit Probel. ICML 2009.

Y.Yue, J. Broder, R. Kleinberg, T. Joachims: The K-armed Dueling Bandits Problem. Journal of Computer and Systems Sciences 78 (2012).

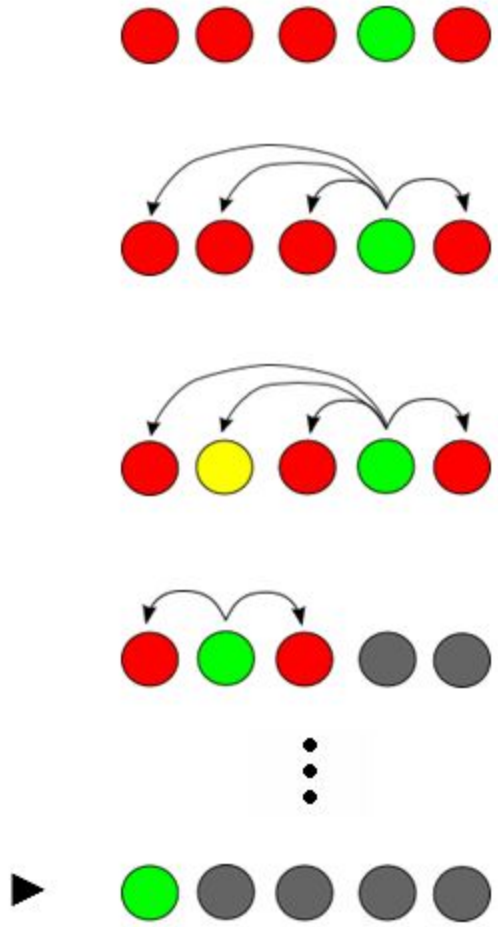
Dueling Bandit Gradient Descent



- Maintain f_t
  - Compare with f_t' (close to f_t -- defined by step size)
  - Update if f_t' wins comparison
- Expectation of update close to gradient of $P(f_t > f')$
  - Builds on Bandit Gradient Descent [Flaxman et al., 2005]

Interleaved Filter

- Choose candidate bandit at random
- Make noisy comparisons (Bernoulli trial) against all other bandits simultaneously
  - Maintain mean and confidence interval for each pair of bandits being compared
- …until another bandit is better
  - With confidence $1 - \delta$
- Repeat process with new candidate
  - Remove all empirically worse bandits
- Continue until 1 candidate left



Dueling Bandit Extensions

- *Beat the Mean Bandit* – relaxed setting where stochastic preferences can violate strong transitivity (Yue & Joachims 2011)
- *SAVAGE* – environmental parameters reflect the idiosyncratic preferences of a mixed crowed (Urvoy et al 2013)
- *RUCB* – more generic dueling bandit with no horizon specified (Zoghi et al. 2014)
- *MergeRUCB* – divide and conquer strategy to reduce the number of exploratory comparisons between rankers (Zoghi et al. 2015)

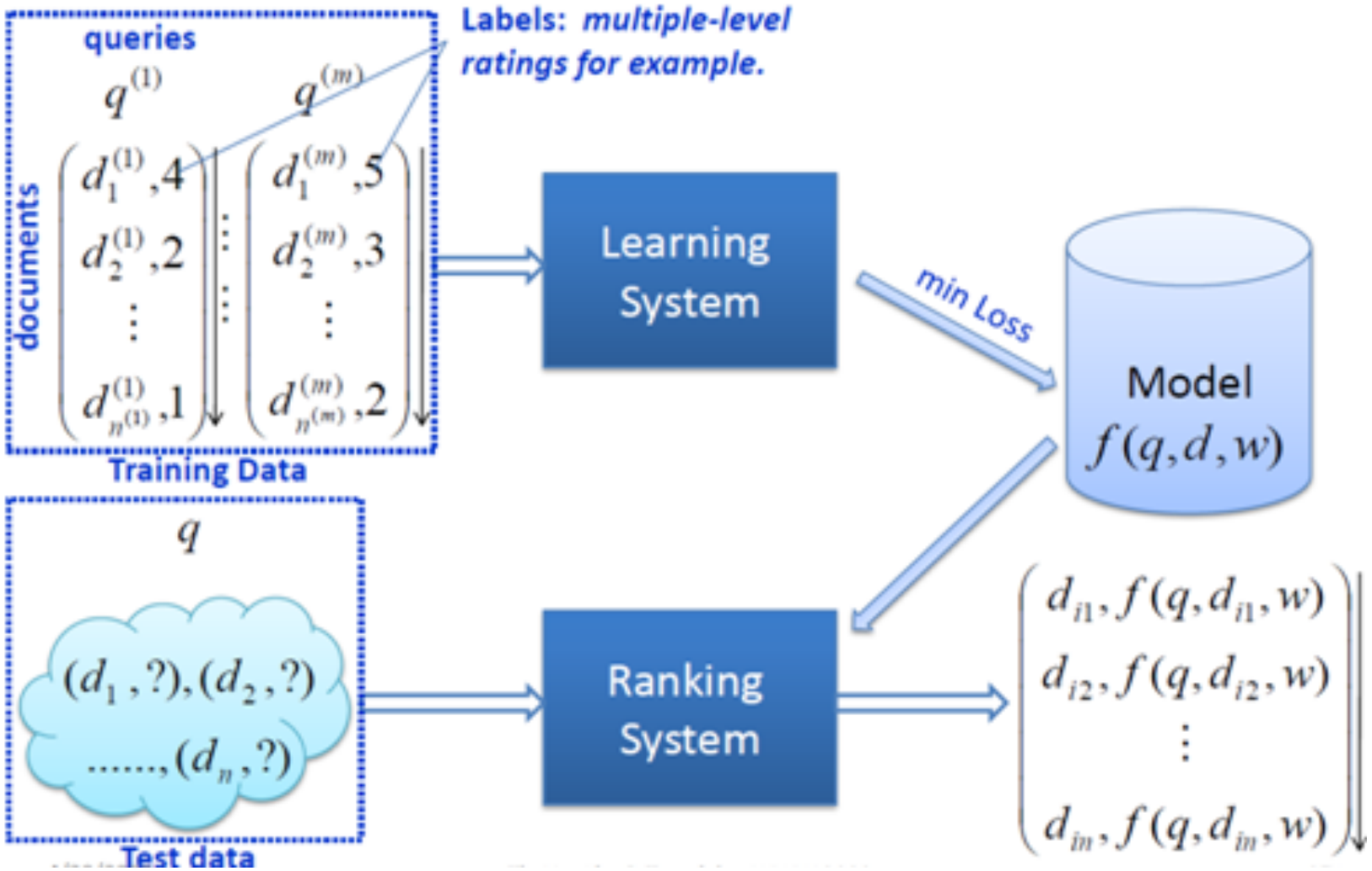
Y.Yue, T. Joachims: *Beat the Mean Bandit*. ICML 2011.

T. Urvoy, F. Clerot, R. Feraud, S. Naamane: *Generic Exploration and K-armed Voting Bandits*. ICML 2013.

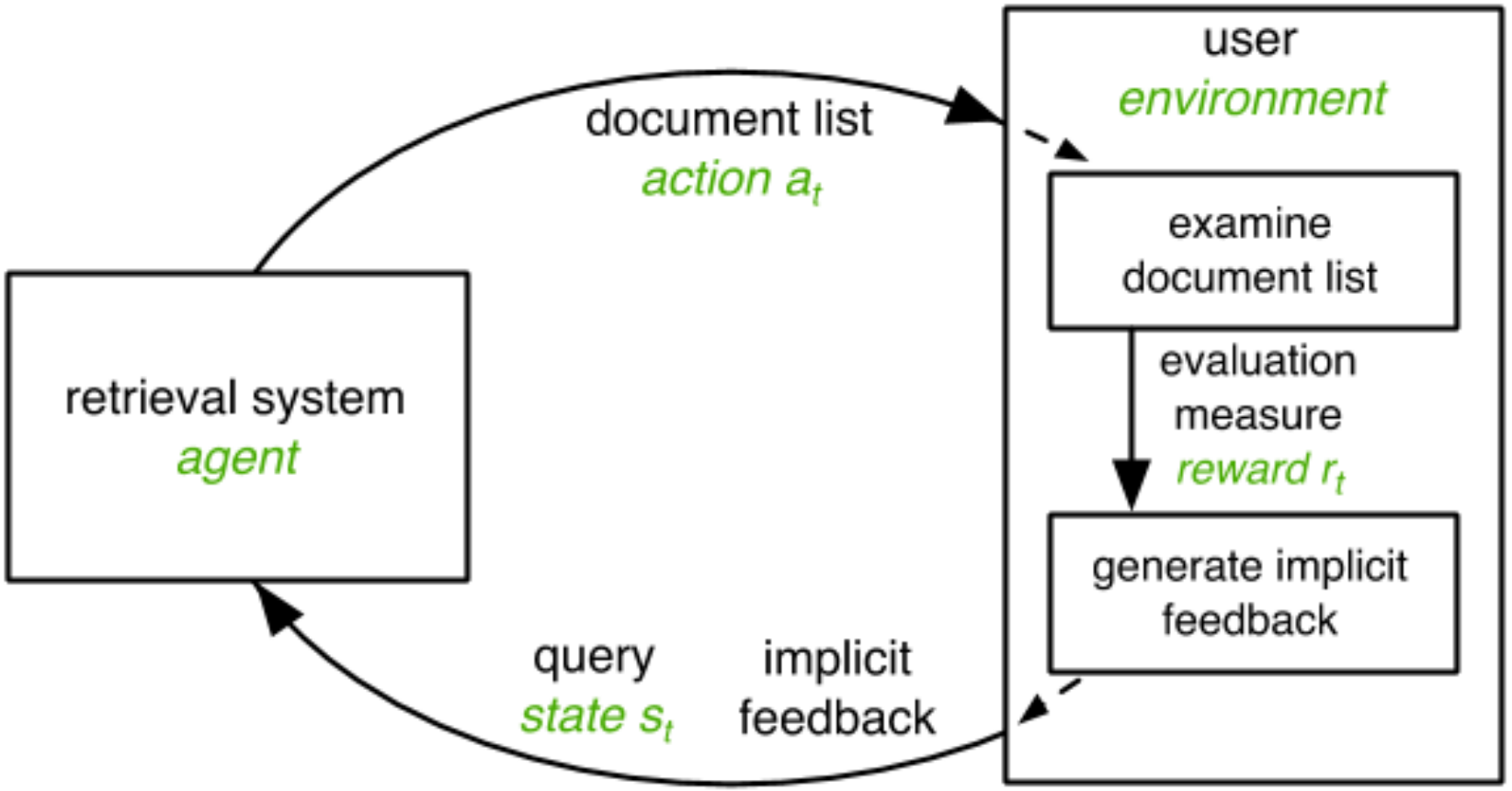
M. Zoghi, S. Whiteson, R. Munos, M. de Rijke: *Relative Upper Confidence Bound for the K-armed Dueling Bandit Problem*. ICML 2014.

M. Zoghi, S. Whiteson, M. de Rijke: *MergeUCB: A Method for Large-Scale Online Ranker Evaluation*. WSDM 2015.

Learning to Rank



Online Learning to Rank



Pairwise Learning to Rank

- Training data obtained from *queries* and *clicks* on a result list (Joachims 2002).
- Clicked documents displayed at lower ranks than a non-clicked document can be assumed to be preferred by the user:
  - Assume query q and a list of documents d_1, d_2, d_3 returned by the system
  - User clicks on documents d_2 and d_3, thus $d_2 > d_1$ and $d_3 > d_1$
  - We obtain labeled document pairs $(d_1, d_2, -1)$ and $(d_1, d_3, -1)$

Pairwise learning to rank

- Given a set of labeled document pairs, we apply stochastic gradient descent to find a weight vector *w*:

$$w = \underset{w}{\operatorname{argmin}} \left[\frac{1}{P} \sum_{i=0}^{P} L(w, x_i, y_i) + \frac{\lambda}{2} \|w\|_2^2 \right]$$

where the last term is a reguralization term, the hinge loss $L = \max(0, 1 - yw^T x)$, *x* is a document feature vector and *y* is a label.

- Documents are scored based on the weight vector *w*.
- After user feedback, *w* is updated.

Explore/Exploit in Pairwise Learning to Ranking

- Two document lists are defined:
  - *exploratory* by uniform *random sampling* of documents associated with the query
  - *exploitative* using baseline learning to rank algorithm
- The two lists are combined by a version of ε-greedy algorithm, where number of documents from each list is determined by ε in [0,1].
- For each rank, an exploitative action is selected with probability $1 - \varepsilon$ and an exploratory action with probability ε.

Listwise Learning to Rank

- Based on a query, two result lists are produced:
  - *Exploitative* from the current vector w
  - *Exploratory* from an exploratory vector w' generated from moving w in a random direction u by step-size δ.
- The two lists are compared using a function $f(l_1, l_2)$, e.g. dueling bandit
- If exploratory vector w' is judged to produce better results, then w moves towards it by step-size σ.

Explore/Exploit Listwise Learn to Rank

- Take two lists l_1 and l_2 and exploration rate k = [0.0, 0.5]
- For each rank, we randomly pick one of the results list biased by k.
- As k decreases, more exploitative documents are presented to the user.
- The resultant list is presented to the user, the clicks are collected and attributed to one of the lists.

Experimental Results

K	*0.5*	*0.4*	*0.3*	*0.2*	*0.1*
HP2003	102.58	109.78	**118.84**	116.38	117.52
HP2004	89.61	97.08	99.03	103.36	**105.69**
NP2003	90.32	100.94	105.03	108.15	**110.12**
NP2004	99.14	104.34	110.16	112.05	**116.00**
TD2003	70.93	75.20	**77.64**	77.54	75.70
TD2004	78.83	80.17	82.40	**83.54**	80.98
OHSUMED	125.35	126.92	127.37	**127.94**	127.21
MQ2007	95.50	94.99	95.70	**96.02**	94.94
MQ2008	89.39	90.55	91.24	**92.36**	92.25

Results for listwise approach. Cumulative NDCG (normalized discounted cumulative gain) for baseline (k = 0.5) and exploit (k = [0.1, 0.4]) runs. Best run per row highlighted in bold.

K. Hofmann, S. Whiteson, M. de Rijke: *Balancing exploration and exploitation in listwise and pairwise online learning to rank for information retrieval*. Inf. Retrieval (2013) 16: 63-90.

Relevance Judgments and Bandits

TREC7							
Num. of judgments	*100*	*300*	*500*	*700*	*900*	*1100*	*2000*
MTF	35	58.04	70.58	78.52	83.48	86.94	**92.7**
TS	34.62	56.4	70	78.18	83	86.36	92.4
TS-NS	**36.8**	**62.42**	**74.42**	**81.74**	**85.82**	**88.32**	92.58
TREC8							
MTF	34.6	58.48	71.78	79.22	84.5	87.58	93.22
TS	34.4	59.34	72.9	80.82	85.56	88.8	93.54
TS-NS	**36.96**	**64.62**	**77.3**	**82.5**	**86.34**	**89.2**	**93.6**

MoveToFront (MTF), stationary Thompson Sampling (TS) and Non-Stationary Thompson Sampling (TS-NS). Average number of relevant documents found at different number of judgments performed.

D. Losada, J. Parapar, A. Barreiro: *Feeling Lucky? Multi-armed Bandits for Ordering Judgments in Pooling-based Evaluation.* SAC 2016.

Additional references

1. D. Losada, J. Parapar, A. Barreiro: *Feeling Lucky? Multi-armed Bandits for Ordering Judgments in Pooling-based Evaluation.* SAC 2016.

2. L. Li et al. *A contextual-Bandit Approach to Personalized News Article Recommendation*. WWW 2010

3. O. Chapelle & L. Li. *An Empirical Evaluation of Thompson Sampling*. NIPS 2011

4. W. Li et al. *Exploration and Exploitation in a Performance based Contextual Advertising System.* KDD'10

5. L. Tang et al*. Personalized Recommendation via Parameter-Free Contextual Bandits*. SIGIR'15.

6. A. Medlar, K. Ilves, P. Wang, W. Buntine, D. Glowacka. *PULP: A System for Exploratory Search of Scientific Literature*. SIGIR 2016.

7. K. Ahukorala, A. Medlar, K. Ilves, D. Glowacka*. Balancing exploration and exploitation: Empirical parametrization of exploratory search systems*. CIKM 2015.

8. S. Hore, L. Tyrvainen, J. Pyykko, D. Glowacka*. A reinforcement learning approach to query-less image retrieval*. International Workshop on Symbiotic Interaction, 2014.

9. D. Glowacka, T. Ruotsalo, K. Konuyshkova, S. Kaski, G. Jacucci*. Directing exploratory search: Reinforcement learning from user interactions with keywords*. IUI 2013.

10. A. Medlar, J. Pyykko, D. Glowacka. *Towards Fine-Grained Adaptation of Exploration/ Exploitation in Information Retrieval.* IUI 2017.

11. B. Kveton, C. Szepesvari, Z. Wen, A. Ashkan: *Cascading Bandits: Learning to Rank in the Cascade Model*. ICML 2015.

12. S. Katariya, B. Kveton, C. Szepesvari, Z. Wen: *DCM Bandits: Learning to Rank with Multiple Clicks*. ICML 2016.

13. C. Li, P. Resnick, Q. Mei: *Multiple Queries as Bandit Arms*. CIKM 2016

… and more

- Music recommendation

Wang et al. *Exploration in Interactive Personalized Music Recommendation: A Reinforcement Learning Approach*. 2014. ACM Trans. Multimedia Comput. Vol. 11 (7)

- Comments recommendation

Mahajan et al*. LogUCB: An Explore-Exploit Algorithm for Comments Recommendation*. CIKM'12

- Collaborative Filtering

1. Wu et al. *Contextual Bandits in a Collaborative Environment.* SIGIR'16
2. Zhao et al. *Interactive Collaborative Filtering*. CIKM'13
3. A. Nakamura. *A UCB-Like Strategy of Collaborative Filtering*. ACML 2014.