Thompson Sampling in Switching Environments with Bayesian Online Change Point Detection

Mellor, Joseph
University of Manchester

Shapiro, Jonathan
University of Manchester

February 18, 2013

Abstract

Thompson Sampling has recently been shown to be optimal in the Bernoulli Multi-Armed Bandit setting [Kaufmann et al., 2012]. This bandit problem assumes stationary distributions for the rewards. It is often unrealistic to model the real world as a stationary distribution. In this paper we derive and evaluate algorithms using Thompson Sampling for a Switching Multi-Armed Bandit Problem.

We propose a Thompson Sampling strategy equipped with a Bayesian change point mechanism to tackle this problem. We develop algorithms for a variety of cases with constant switching rate: when switching occurs all arms change (Global Switching), switching occurs independently for each arm (Per-Arm Switching), when the switching rate is known and when it must be inferred from data. This leads to a family of algorithms we collectively term Change-Point Thompson Sampling (CTS).

We show empirical results of the algorithm in 4 artificial environments, and 2 derived from real world data; news click-through [Yahoo!, 2011] and foreign exchange data [Dukascopy, 2012], comparing them to some other bandit algorithms. In real world data CTS is the most effective.

1 Introduction

Thompson Sampling has recently been shown to be optimal in the Bernoulli Multi-Armed Bandit setting [Kaufmann et al., 2012]. This bandit problem assumes stationary distributions for the rewards. It is often unrealistic to model the real world as a stationary distribution and algorithms such AdaptEvE have been proposed to solve bandit problems in this environment. In this paper we concern ourselves with a Switching Multi-Armed Bandit Problem.

We first review Thompson Sampling, before describing the non-stationary environment we are concerned with. We then propose a method to solve this problem and review the techniques we employ. We then test our algorithms on a variety of environments.
1.1 Thompson Sampling

Thompson Sampling is effectively a probability matching algorithm. The desired strategy is to pull an arm with the probability that the arm is the best arm. This probability can be written as

\[ P(a_i = a^*) = \int_\theta I(a_i = a^* | \theta) P(\theta | D) d\theta, \]

where \( \theta \) is a model of our arms, \( a^* \) is the optimal arm, \( a_i \) is the \( i \)th arm and \( D \) is the history of past rewards. \( I(x) \) is the indicator function, which is 1 when \( x \) is true, and 0 otherwise. The strategy can thus be reduced to sampling from the model distribution \( P(\theta | D) \) and then picking the arm that is maximal given this model.

This paper considers multi-armed bandits with Bernoulli arms. Each arm, \( j \), delivers a reward of 1 with probability \( \theta_j \) and 0 otherwise. In the stationary case the parameter is \( \theta = (\theta_1, \ldots, \theta_k) \) where \( k \) is the number of arms. The arms are assumed independent and so \( P(\theta | D) \) is a product of terms \( P(\theta_j | D_j) \), where \( D_j \) are past rewards for arm \( j \). Further, since the arms are Bernoulli, for which the Beta distribution is a conjugate prior, we can write \( P(\theta) \) as a product of Beta distributions:

\[ P(\theta | D) = \prod_{j=1}^k P(\theta_j | \alpha_j, \beta_j, D_j) \]

where \( \alpha_j = \# \{ \text{reward} = 1 \} + \alpha_0 \) and \( \beta_j = \# \{ \text{reward} = 0 \} + \beta_0 \).

We can sample from \( P(\theta | D) \) by sampling from all \( P(\theta_j | \alpha_j, \beta_j, D_j) \) and choosing the arm, \( j \), with largest \( \theta_j \).

1.2 Model of Dynamic Environment

In this paper we assume that the environment changes over time. We assume abrupt switching defined by a hazard function, \( h(t) \), such that,

\[ \theta_i(t) = \begin{cases} \theta_i(t-1) & \text{with probability } h(t) \\ \theta_{\text{new}} \sim U(0,1) & 1 - h(t). \end{cases} \]

The algorithms presented are designed with 2 such models in mind. The first model we will refer to as the Global Switching model. This model switches at a constant rate, when a change point happens all arms change their expected rewards. The second model will be referred to as Per-Arm Switching. In this model change points occur independently for each arm, such that when the expected reward switches for one arm, it is uncorrelated to when all other arms switch.

Examples of this form of changing environment might include stock market data where stock prices can change their statistical nature very quickly subject to external events. Switching behaviour has been studied in Financial Markets [Preis et al., 2011], and multi-armed bandits have been applied to this field before [Sorensen, 2007].
1.3 Regret for Switching Environments

In a stationary multi-armed bandit the regret measure we use is taken to be the expected difference in reward between our strategy and that of a policy which always chose the arm with highest expected payoff, \( \mu^* \). In a switching system the arm which is considered optimal changes. In this case a different form of regret is often used. Garivier and Moulines [2008] use the following definition of regret.

\[
R(T) = \langle \sum_{t=0}^{T} (\mu^*_t - \mu_{i_t}) \rangle,
\]

(4)

where \( \mu^*_t \) is the highest expected payoff of an arm at time \( t \) and where \( \langle .. \rangle \) denotes expectation over possible sequences of \( \{\mu_{i_t}\} \).

Unless otherwise stated our experiments will report a related quantity,

\[
R_n(T) = \langle \sum_{t=0}^{T} I(\mu^*_t \neq \mu_{i_t}) \rangle,
\]

(5)

which is the expected number of times a suboptimal arm is pulled. This corresponds to the results Hartland et al. [2007] report.

2 Switching Thompson Sampling

In order to perform Thompson Sampling we wish to sample from \( P(\theta|D_{t-1}) \), which is the probability of the arm model given the data so far. In a switching system the arms model \( \theta \) is only dependent on the data since the last switching occurred, but we do not know when this happened. If we did we could just do the same Bayesian update as with the standard Bernoulli case to arrive at the distribution of our model. Since we do not know the runlength \( r_t \) we can introduce it as a latent variable and marginalise it out. Taking \( D_{t-1} \) as the history of rewards and arm pulls seen so far, we can write this as

\[
P(\theta|D_{t-1}) = \sum_{r_t} P(\theta|D_{t-1}, r_t) \cdot P(r_t|D_{t-1}).
\]

(6)

Now to sample from \( P(\theta|D_{t-1}) \) we just need to sample from the \( P(r_t|D_{t-1}) \) (the runlength distribution) and then given that runlength, sample from \( P(\theta|D_{t-1}, r_t) \) to arrive at our arm model \( \theta \). We can select the arm to pull that in expectation maximises the reward given this model.
3 Bayesian Online Change Detection

Fearnhead and Liu [2007] as well as Adams and MacKay [2007] have independently done work on calculating the online posterior of the runlength. They show exact inference on the runlength can be achieved by a simple message passing algorithm. Letting $x_t$ be the reward at time $t$ so that $D_t = x_t \cup D_{t-1}$. The inference procedure can be easily derived as follows.

\[
P(r_t | x_{t-1}, D_{t-2}) = \frac{P(r_t, x_{t-1}, D_{t-2})}{P(x_{t-1}, D_{t-2})}
\]  

(7)

The numerator can then be expressed as

\[
P(r_t, x_{t-1}, D_{t-2})
= \sum_{r_{t-1}} P(r_t, r_{t-1}, x_{t-1}, D_{t-2})
= \sum_{r_{t-1}} P(r_t, x_{t-1} | r_{t-1}, D_{t-2}) P(r_{t-1}, D_{t-2})
= \sum_{r_{t-1}} P(r_t | r_{t-1}) P(x_{t-1} | r_{t-1}, D_{t-2}) P(r_{t-1}, D_{t-2}).
\]

(8)

(9)

(10)

The derivation just applies the rules of probability up to and including equation (9). One assumption is made in equation (10) that the runlength is only dependent on the previous time steps runlength. This forms a simple message passing algorithm because $r_t$ can only take values depending on $r_{t-1}$. In fact $r_t = r_{t-1} + 1$ when switching does not occur and $r_t = 0$ when it does. $P(r_t | r_{t-1})$ is defined by a hazard function $h(t)$. For simplicity in our case this is a constant switching rate $\gamma$.

Unfortunately the exact inference has space and time requirements that grow linearly in time. The space requirements are linear because at each time step the support set of the posterior runlength distribution increases by one, which means we have to store information for an extra value of the runlength at every step. The update is also linear in time, as the message passing algorithm requires an update to each runlength in the support. Adams and MacKay suggest a simple thresholding technique to eliminate runlengths with small probability mass associated with them. As we can only know in expectation how much memory this algorithm will require, an alternative with hard guarantees on memory requirements is desirable. Fearnhead and Liu suggest a much more sophisticated particle filter resampling step to maintain a finite sample of the runlength distribution, which has the benefit that we can be certain on the upper limit of space the algorithm requires, this approach is the one taken in this paper.
3.1 Particle Filters

A particle filter is a Monte-Carlo method for approximately estimating a sequential Bayesian model. Particles are used to represent points in the distribution to be estimated and are assigned weights that correspond to their approximate probabilities. The number of particles can grow at each time step and so occasionally some particles need to be thrown away. This leaves us to assign new weights to the remaining particles. This procedure is called resampling.

3.1.1 Stratified Optimal Resampling

[Fearnhead and Clifford 2003] originally proposed optimal resampling. We wish to reduce a discrete probability distribution with a support of \( N \) discrete points down to a stochastic distribution of \( M \) discrete points, where the set of \( M \) points is a subset of the original \( N \). The original \( N \) points each have probability mass \( p_i \) associated with them, and the procedure finds a reweighting of these probabilities, \( q_i \), such that \( N - M \) of the probabilities are 0. The idea is that we wish there to be no bias in the sampling procedure, which means that the expected value of \( q_i \) should be the original probability mass \( p_i \). The algorithm is optimal in the sense that the expected squared difference between the original probabilities \( p_i \) and the new weights \( q_i \) are minimised.

This can be done by the following procedure;

1. Find \( \kappa \) such that \( M = \sum_{i=1}^{N} \min(1, p_i / \kappa) \)
2. Sample \( u \) from uniform distribution, \( U(0, \kappa) \)
3. Iterate through all \( p_i \)
   a. If \( p_i > \kappa \) Then \( q_i = p_i \)
   b. Otherwise
      i. \( u = u - p_i \)
      ii. If \( u < 0 \) Then \( q_i = \kappa \) and \( u = u + \kappa \)
      iii. Otherwise \( q_i = 0 \)

The particles where \( p_i = q_i \) are kept with probability 1. The remaining particles are such that \( q_i = \kappa \) with probability \( p_i / \kappa \) and \( q_i = 0 \) otherwise. Thus their expectation remains the same.

The worstcase time complexity of this algorithm is \( O(N \log N) \), but it has an amortised cost of \( O(N) \) [Fearnhead and Clifford 2003].

4 Proposed Inference Models

We have shown we can perform Thompson Sampling in a switching system by splitting the procedure into a stage that samples the runlength since a switch occurred and a stage that samples from the arm model given this runlength.
The Global Switching and Per-Arm Switching models are appealing due to their simplicity. Only one runlength distribution needs to be inferred for Global Switching, which does not depend on the number of arms the bandit has. The Per-Arm model can store the runlengths for each arm independent, and the space requirements grow linearly with respect to the number of arms. Other models with more complicated dependencies between arms can quickly become intractable.

4.1 Global switching
In global switching there is a single change point process across all of the arms since when one arm switches distribution so do all other arms. This means that the data from every arm pull contributes to the posterior of the single runlength distribution. Effectively to sample from the posterior of the full bandit model, we first need to sample from the runlength distribution, this gives us an estimate of the runlength, which tells us how much data from the past our arms can use. Once the global runlength is sampled, we then proceed by sampling individually from the posterior distributions of the arms, given only the data since the last changepoint (determined by the runlength). The arm with the corresponding maximum sample is then pulled. We only need to store the posterior probabilities of the given runlengths and the hyperparameters for the arm posteriors associated with those runlengths. We will call the runlength distribution the Change Point model, and the set of hyperparameters associated with each runlength for a given action the Arm model.

The Change Point model is an approximation of the runlength distribution storing a probability for at most $N$ runlengths. Let $w^t_i$ be the probability of having a runlength of $i$ at time $t$. In this paper the arm rewards are assumed to come from a Bernoulli distribution so the hyperparameters stored are the 2 parameters for the Beta distribution. Let $\alpha^t_{i,j}$ and $\beta^t_{i,j}$ be the hyperparameters for a runlength of $i$ at time $t$ for arm $j$. At any point in time $t$ there is a set of runlengths $R_t \subset \mathbb{N}$, $|R_t| \leq N$, where for every $r \in R_t$ there exists quantities $w^t_r$, $\alpha^t_{r,j}$ and $\beta^t_{r,j}$. When $|R_t| = N$ then a resampling step is performed in order to reduce the number of runlengths stored. For ease of notation let $\{w\}^t$ be the set of runlength probabilities at time $t$ and let $\{\alpha\}^t$ and $\{\beta\}^t$ be the sets of hyperparameters for arm $j$ at time $t$. Similarly let $\{\alpha\}$ and $\{\beta\}$ be the set of all hyperparameters at time $t$. The algorithm is presented in pseudocode in figure 1.

We will refer to this algorithm as Global Change-Point Thompson Sampling (Global-CTS).

4.2 Per-arm switching
The difference in implementation with respect to global switching is that now there is a runlength distribution for each arm. That is, for each arm $j$ we have a different set of runlength probabilities $w^t_{i,j} \in \{w\}_j^t$. In the per-arm switching model at a timestep $t$ we update the Change Point model associated with the
Algorithm 1 Global Change-Point Thompson Sampling

```
procedure GLOBAL-CTS(N, γ, α₀ = 1, β₀ = 1)
    t ← 0
    w₀ ↑ = 1, and add to \{w\}^t  \triangleright Initialise time
    for all arms j, α₀,j ← α₀
    for all arms j, β₀,j ← β₀
    while Interacting do
        a ← SELECTACTION(\{w\}^t, \{α\}^t, \{β\}^t)
        r ← PULLARM(a)
        \{w\}^{t+1} ← UPDATECHANGEMODEL(\{w\}^t, \{α\}^t, \{β\}^t, a, r, γ)
        \{α\}^t, \{β\}^t ← UPDATEARMMODELS(\{α\}^t, \{β\}^t, a, r)
        if |\{w\}^{t+1}| = N then
            PARTICLERESAMPLE(\{w\}^{t+1}, \{α\}^{t+1}, \{β\}^{t+1})
        end if
        t ← t + 1
    end while
end procedure
```

```
procedure UPDATECHANGEMODEL(\{w\}^t, \{α\}^t, \{β\}^t, a, r, γ)
    if r = 1 then
        likelihoodα ← \frac{α_i,α}{α_i,α+\beta_i,α}, For all i s.t. w_i ∈ \{w\}^t
    else
        likelihoodα ← \frac{β_i,α}{α_i,α+\beta_i,α}, For all i s.t. w_i ∈ \{w\}^t
    end if
    w_{t+1} ← (1 − γ) \* likelihood, \* s_i, For all i s.t. w_i ∈ \{w\}^t
    w_{t+1} ← N \* likelihood, \* s_i
    Normalise \{w\}^{t+1}
    return \{w\}^{t+1}
end procedure
```

```
procedure UPDATEARMMODELS(\{α\}^t, \{β\}^t, a, r)
    if r=1 then
        α_{i+1,a} ← α_{i,a} + 1, For all i s.t. α_{i,a} ∈ \{α\}^t
    else
        β_{i+1,a} ← β_{i,a} + 1, For all i s.t. β_{i,a} ∈ \{β\}^t
    end if
    α_{β,j} ← α_0, For all arms j
    β_{β,j} ← β_0, For all arms j
    return \{α\}^{t+1}, \{β\}^{t+1}
end procedure
```

```
procedure PARTICLERESAMPLE(\{w\}^{t+1}, \{α\}^{t+1}, \{β\}^{t+1})
    Find set to discard d ∈ D using Stratified Optimal Resampling on \{w\}^{t+1}
    Discard all \{w\}^{t+1}, \{α\}^{t+1}, \{β\}^{t+1}
end procedure
```

```
procedure SELECTACTION(\{w\}^t, \{α\}^t, \{β\}^t)
    Pick i with probability w_i
t    for each arm j do
        sample_j ← Beta(α_{i,j}, β_{i,j})
    end for
    return max_j sample_j
end procedure
```

arm that was pulled at \(t\) much like via the update equations sketched in [10].
The Change Point models associated with arms not pulled at \(t\) are updated differently since the runlength for these arms is independent of the reward we received for the arm we actually pulled. The reward likelihood term disappears in the update equations for the runlength distribution of unpulled arms. This
is shown in equation 13. Since we normalise the distribution at each step we can ignore the factor $P(x_{t-1})$. This is shown as follows,

$$P(r_t, x_{t-1}, D_{t-2}) = P(x_{t-1})P(r_t, x_{t-1}, D_{t-2})$$  \hspace{1cm} (11)

\[\propto \sum_{r_{t-1}} P(r_t, r_{t-1}, D_{t-2}) \]  \hspace{1cm} (12)

\[\propto \sum_{r_{t-1}} P(r_t | r_{t-1}, D_{t-2})P(r_{t-1}, D_{t-2}). \]  \hspace{1cm} (13)

We will refer to this algorithm as Per-Arm Change-Point Thompson Sampling (PA-CTS).

### 5 Learning the Switching Rate

Both [Wilson et al. 2010](#) and [Turner et al. 2009](#) have proposed methods for learning the hazard function from the data. Wilson et al. method can learn a hazard function that is piecewise constant via a hierarchical generative model. Turner et al. can learn any parametric hazard rate via gradient descent, but from initial investigations appeared to not perform particularly well if the hazard rate is adapted at every time step. For the purposes of this paper, a constant switching rate was assumed which was learned using the approach of Wilson et al.

For the simplest case where we consider a single constant switch rate, Wilson et al. model whether a change point occurred as a Bernoulli variable. The hyperparameters of this switching rate are those of a Beta distribution and can be thought of as the number of times the system has switched, $a_t$ and the number of times it not switched $b_t$. We now compute the joint distribution $P(r_t, a_t | x_{t-1}, D_{t-2})$ as oppose to the original distribution $P(r_t | x_{t-1}, D_{t-2})$. The message passing proceeds in a very similar fashion as before, except now the number of particles also grows quadratically rather than linearly.

In the global switching model the algorithm now keeps track of sets of particles $w_{r,\alpha}^f$, $\alpha_{r,\alpha}^f$, and $\beta_{r,\alpha}^f$ associated with a runlength $r$, learning rate hyperparam-
eter $a$, arm $i$ and time $t$. The updates are as follows.

$$w_{0,0}^{t+1} \leftarrow 1$$

$$w_{r+1,a}^{t+1} \leftarrow \frac{t - a + 1}{t + 2} \frac{\alpha_{r,i,a}^t}{\alpha_{r,i,a}^t + \beta_{r,i,a}^t} w_{r,a}^t \text{if reward } = 1$$

$$w_{r,a}^{t+1} \leftarrow \frac{t - a + 1}{t + 2} \frac{\beta_{r,i,a}^t}{\alpha_{r,i,a}^t + \beta_{r,i,a}^t} w_{r,a}^t \text{if reward } = 0$$

$$w_{r,a+1}^{t+1} \leftarrow \frac{a + 1}{t + 2} \frac{\alpha_{r,i,a}^t}{\alpha_{r,i,a}^t + \beta_{r,i,a}^t} w_{r,a}^t \text{if reward } = 1$$

$$w_{r,a+1}^{t+1} \leftarrow \frac{a + 1}{t + 2} \frac{\beta_{r,i,a}^t}{\alpha_{r,i,a}^t + \beta_{r,i,a}^t} w_{r,a}^t \text{if reward } = 0$$

We again use the resampling algorithm of Fearnhead to manage the space requirements of the algorithm.

In the Global Switching model there is only 1 runlength distribution, and so only 1 switching rate to learn, this leads naturally to an algorithm Non-Parametric Global Change-Point Thompson Sampling (NP Global-CTS). With Per-Arms there are many possibilities, there could be a single switching rate for each of the independent arms, or each arm could have a separate switching rate. In this paper we assume each arm has a separate switching rate and call this algorithm Non-Parametric Per-Arm Change-Point Thompson Sampling (NP PA-CTS).

6 Tracking Changes In The Best Arm

The algorithms presented so far attempt to track changes in all arms, irrespective of whether they are pulled. The distribution of $\theta_i$ for an arm not pulled will slowly become flatter and thus have higher variance. One of the stated assumptions of Adapt-EvE, was that it was only important to track whether the perceived best arm has changed distribution. We can modify the algorithms to better replicate this assumption.

The model for the Per-Arm Switching is adapted most simply. Since both the Change Point models and the Arm models are independent for each arm, we can track the change of the best arm by only updating the Change Point and Arm models for the arm that was pulled. For the Global Switching model the change is not so clear since all arms share a Change Point model. In this paper we updated the shared Change Point model and only updated the Arm model for the arm pulled. The question then arises what should the hyperparameters for the unpulled arms be that are associated with the new runlength of zero. The approach taken here was to set them to the hyperparameters associated with a random other runlength in the distribution.

We can apply the sample method from Wilson to infer the switching rate for
these architectures as well. The algorithms described in this section with be denoted by having a “2” appended to the algorithm name.

7 Experiments

6 different non-stationary environments were used to evaluate our bandit model. 4 are based on purely synthetic data, and 2 use data collected from the real world. The parameters for all experiments shown were tuned based on the PASCAL challenge. Bold denotes best results in all tables.

7.1 Global Switching Environment

We first compare the algorithms in an environment with a constant global switching rate. Global-CTS and NP Global-CTS were designed for this environment and so a-priori we would expect them to perform the best.

The first set of experiments were a single run of the algorithms working in an instance of this environment type with 2 arms. Figures 1 and 2 plot example heatmaps of the runlength distributions of some of the algorithms. At a particular time, the graphs show the runlength distribution. In the case of the PA-CTS and NP PA-CTS there are 2 plots for each algorithm, corresponding to the runlength distribution for each arm. The pay off of the 2 arms has been superimposed over the top of the plots so that it can be seen how the runlength distribution matches up with the changes in the environment. From the heatmap figures we can see the change point prediction works when applied to a bandit problem. As expected the change point distribution looks to be more accurate for the Global-CTS and NP Global-CTS algorithms which use the Global Switching model, this is because each data point can contribute to the posterior runlength distribution. The PA-CTS also performs reasonably well even though the amount of data that has influence on each posterior is reduced. For the NP PA-CTS algorithm, learning the separate switching rates appears to significantly decrease the certainty for a particular runlength.

An experiment comparing the algorithms in this setting was performed. Each run was over a period of $10^6$ time steps and the experiment was repeated 100 times. The results are displayed in table 1. All parameters were set as for the PASCAL challenge test run. The environment constant switching rate was $10^{-4}$.

Global-CTS performs the best in the environment, which is not surprising since the environment fits the algorithms model. NP Global-CTS performs well in this too, which suggests that learning the hazard rate for this model may be feasible.
7.2 Per-Arm Switching Environment

The next environment was a switching system where the switching for each arm was independent of every other arm. PA-CTS and NP PA-CTS were designed with this situation in mind and again a-priori may be expected to perform better.

An experiment comparing the algorithms was performed with $10^6$ iterations and then repeated 100 times. The results are shown in table 2. As expected the PA-CTS algorithm performs best in this environment. NP PA-CTS, the algorithm corresponding to PA-CTS that learns the hazard rate suffers much more regret, which would appear to indicate for the particular model the parameters are not being learned quickly enough. The algorithms designed for a Global Switching environment also perform reasonably in this sort of environment.
Figure 2: Runlength Distribution for PA-CTS and NP PA-CTS in Global Switching Environment. The mean payoffs of the arms are super-imposed over the distribution.

### 7.3 Bernoulli Armed Bandit with Random Normal Walk

The PASCAL challenge environments were found to be periodic, which is not the sort of environment our algorithms were intended for. Another simulated environment was investigated. In this environment at time $t$, each arm, $i$, was Bernoulli with probability of success $\theta_i(t)$. At each time step the success rate of the arm was allowed to drift as a truncated normal walk. That is, the probability of success for an arm $\theta_i(t) \in [0, 1]$ conditional on $\theta_i(t-1) \in [0, 1]$ is,

$$P(\theta_i(t)|\theta_i(t-1)) = \frac{e^{-(\theta_i(t-1)-\theta_i(t))^2}}{\sqrt{2\pi\sigma^2}} \int_0^1 e^{-\frac{(\theta_i(t-1)-x)^2}{2\sigma^2}} dx.$$

Table 3 shows a comparison of the algorithms. The experiment was run 100 times, where each run had a period of $10^6$. The variance of the random walk was set to $\sigma^2 = 0.03$.

In this sort of environment it appears that our algorithms perform better than the benchmark algorithms with NP PA-CTS2 achieving the smallest regret.

12
Table 1: Results against Global Switching Environment (given as number of mistakes \(\times 10^{-3}\) ± Std. Error)

| Name          | Regret | Name          | Regret |
|---------------|--------|---------------|--------|
| Global-CTS    | 5.9 ± 0.07 | Global-CTS2  | 30.5 ± 1.07 |
| PA-CTS        | 12.1 ± 0.10 | PA-CTS2      | 49.6 ± 1.70 |
| NP Global-CTS | 6.7 ± 0.08  | NP PA-CTS    | 29.4 ± 0.95 |
| NP Global-CTS2| 10.3 ± 0.20 | NP PA-CTS2   | 25.6 ± 0.86 |
| UCB           | 178.3 ± 8.20 | DiscountedUCB | 15.5 ± 0.27 |
| Random        | 333.1 ± 2.09 |             |        |

Table 2: Results against Per-Arm Switching Environment (given as number of mistakes \(\times 10^{-3}\) ± Std. Error)

| Name          | Regret | Name          | Regret |
|---------------|--------|---------------|--------|
| Global-CTS    | 13.8 ± 0.20 | Global-CTS2  | 37.9 ± 1.02 |
| PA-CTS        | 13.0 ± 0.11 | PA-CTS2      | 67.1 ± 1.23 |
| NP Global-CTS | 13.8 ± 0.17 | NP PA-CTS    | 30.8 ± 0.79 |
| NP Global-CTS2| 15.8 ± 0.28 | NP PA-CTS2   | 38.1 ± 0.83 |
| UCB           | 176.1 ± 7.44 | DiscountedUCB | 16.8 ± 0.28 |
| Random        | 336.4 ± 1.85 |             |        |

7.4 PASCAL Challenge 2006

The PASCAL Exploration vs. Exploitation Challenge 2006 was a competition in a multi-armed bandit problem [Hussain et al., 2006]. The challenge revolved around website content optimisation, whereby the options available corresponded to different content to present to a user on a website. The challenge is a good general test for the algorithms presented in this paper as to perform well it was required for the bandit algorithms to be able to work in non-stationary environments. The challenge had 6 separate environments in which the algorithms needed to perform; Frequent Swap (FS), Long Gaussians (LG), Weekly Variation (WV), Daily Variation (DV), Weekly Close Variation (WCV) and Constant (C). These environments are artificially generated, where the dynamics of the expected payoffs resemble either periodic Gaussian, Sinusoidal or constant signals.

Hartland et al. [2007] won this competition with the Adapt-EvE algorithm. The Adapt-EvE algorithms most prominent feature is its use of a change-point detection mechanism. Since the algorithms presented in this paper also use a change-point mechanism it is interesting to compare their performance. The challenge also provides an environment for which the algorithm was not directly designed for and so will hopefully indicate some robustness in their strategy. We were unable to implement a version of Adapt-EvE that replicated the performance reported, so here we are simply replicating the results published. To avoid an unfair comparison in other environments we did not run our own implementation of Adapt-EvE in those environments.

Table 4 shows a comparison of the Change-Point Thompson Sampling algorithms (Global-CTS, PA-CTS, NP Global-CTS NP PA-CTS) against Adapt-EvE Meta-Bandit and Meta-p-Bandit [Hartland et al., 2007]. The comparison also features the algorithm “DiscountedUCB”, which was submitted by Thomas
Table 3: Results against Bernoulli Bandit with Truncated Normal Walk (given as number of mistakes×10^{−3}± Std. Error)

| Name          | Regret   | Name          | Regret   |
|---------------|----------|---------------|----------|
| Global-CTS    | 97.9 ± 0.10 | Global-CTS2  | 134.1 ± 0.23 |
| PA-CTS        | 107.1 ± 0.16 | PA-CTS2      | 148.9 ± 0.35 |
| NP Global-CTS | 116.6 ± 0.13 | NP PA-CTS    | 117.0 ± 0.11 |
| NP Global-CTS2| 100.9 ± 0.10 | NP PA-CTS2  | 94.8 ± 0.13  |
| UCB           | 194.5 ± 3.78 | DiscountedUCB| 162.4 ± 0.47 |
| Random        | 325.9 ± 0.24 |              |           |

Jakobs to the same competition and performed comparably to Adapt-EvE. The code for this algorithm was available and so has been included for comparison in all other environments.

Table 4: Results against PASCAL EvE Challenge 2006 (given as number of mistakes×10^{−3})

|                | Global-CTS | Global-CTS2 | Adapt-EvE Meta p |
|----------------|------------|-------------|-----------------|
| WCV            | 8.9 ± 0.4  | 6.9 ± 0.4   | 5.5 ± 0.9       |
| FS             | 27.9 ± 2.4 | 12.5 ± 1.3  | 10.6 ± 1.3      |
| C              | 0.6 ± 0.1  | 1.0 ± 0.2   | 3.2 ± 0.3       |
| DV             | 17.1 ± 0.3 | 6.6 ± 0.3   | 6.1 ± 0.7       |
| LG             | 4.4 ± 0.4  | 3.4 ± 0.5   | 4.3 ± 1.4       |
| WV             | 8.2 ± 0.3  | 5.3 ± 0.5   | 5.1 ± 0.9       |
| Total          | 97.2 ± 35.8| 33.8        | 34.7            |

|                | PA-CTS | PA-CTS2 | Adapt-EvE Meta |
|----------------|--------|--------|----------------|
| WCV            | 4.2 ± 0.8 | 6.4 ± 0.8 | 5.4 ± 0.8 |
| FS             | 13.7 ± 1.6 | 15.1 ± 1.7 | 14.0 ± 1.9 |
| C              | 3.2 ± 0.4  | 2.0 ± 0.3  | 2.5 ± 0.5    |
| DV             | 4.5 ± 1.5  | 4.9 ± 0.5   | 6.2 ± 0.7    |
| LG             | 9.4 ± 2.9  | 3.7 ± 0.7   | 4.8 ± 1.6    |
| WV             | 4.7 ± 1.7  | 5.4 ± 0.5   | 4.8 ± 0.8    |
| Total          | 39.6 ± 37.4| 37.4      | 37.7          |

|               | NP Global-CTS | NP Global-CTS2 | DiscountedUCB |
|---------------|---------------|----------------|---------------|
| WCV           | 8.9 ± 0.4     | 9.0 ± 0.3      | 9.2 ± 0.5     |
| FS            | 28.2 ± 2.7    | 14.8 ± 1.2     | 10.1 ± 1.1    |
| C             | 0.3 ± 0.2     | 0.8 ± 0.2      | 5.5 ± 0.5     |
| DV            | 17.6 ± 0.3    | 16.0 ± 0.3     | 7.9 ± 0.9     |
| LG            | 4.4 ± 0.4     | 4.0 ± 0.3      | 2.9 ± 0.4     |
| WV            | 8.5 ± 0.3     | 8.4 ± 0.3      | 4.0 ± 0.4     |
| Total         | 97.9 ± 35.1   | 53.1           | 35.7          |

|                | NP PA-CTS | NP PA-CTS2 | Random |
|----------------|----------|-----------|--------|
| WCV            | 12.8 ± 0.7 | 10.4 ± 0.4 | 25.3 ± 0.5 |
| FS             | 23.1 ± 1.2 | 23.0 ± 1.9 | 49.1 ± 0.5 |
| C              | 15.8 ± 0.4 | 1.9 ± 0.2  | 20.0 ± 0.1 |
| DV             | 15.1 ± 1.0 | 24.2 ± 0.3 | 57.2 ± 0.3 |
| LG             | 14.4 ± 2.1 | 8.2 ± 0.5  | 112.1 ± 9.1 |
| WV             | 12.1 ± 1.1 | 11.7 ± 0.4 | 57.2 ± 0.3 |
| Total          | 93.2 ± 79.3| 79.3       | 321.3 |

7.5 Yahoo! Front Page Click Log Dataset

[Yahoo! (2011)] have produced a bandit algorithm dataset. The dataset provides information about the top story presented to a user on the front page of Yahoo. Each entry in the dataset gives information about a single article presented, the time it was presented, contextual information about the user and whether the
user “clicked-through” to the article or not. The dataset was designed for the contextual bandit problem. Given context of a user the goal is to select an article to present to the user as to maximise the expected rate at which users click on the article to read more (click-through). The articles also change during the dataset, and so bandit algorithms designed specifically for this environment also need the ability to modify the number of arms they can select from.

For the purposes of our experiments we do not concern ourselves with the contextual case, nor do we try to incorporate new articles as they arrive. Instead we ignore the context, and we only pick from a set number of articles. This reduces the problem to a conventional multi-armed bandit problem. To maximise the amount of data used, for each run we randomly selected the set of articles (in our case 5 articles) from a list of 100 permutations of possible articles which overlapped in time the most. The click-through rates were estimated from the data by taking the mean of an articles click-through rate every 1000 time ticks. The simulation then proceeded as described by Li et al. [2011], the results are presented in table 5. The regret for each run was normalised by the number of arm pulls, since this was different in each run of the simulation. Parameters were set as for the PASCAL challenge dataset.

Table 5: Results against Yahoo! Front Page Click Log Dataset (± Std. Error)

| Name        | Regret       | Name        | Regret       |
|-------------|--------------|-------------|--------------|
| Global-CTS  | 0.489 ± 0.035| Global-CTS2 | 0.443 ± 0.031|
| PA-CTS      | 0.522 ± 0.029| PA-CTS2     | 0.505 ± 0.028|
| NP Global-CTS| 0.490 ± 0.029| NP PA-CTS   | 0.590 ± 0.018|
| NP Global-CTS2| 0.530 ± 0.026| NP PA-CTS2  | 0.563 ± 0.018|
| UCB         | 0.526 ± 0.040| DiscountedUCB| 0.568 ± 0.022|
| Random      | 0.800 ± 0.001|

7.6 Foreign Exchange Rate Data

We constructed a final test environment from Foreign Exchange Rate data [Dukascopy, 2012]. Ask prices for 4 currency exchange rates (GBP-USD, USD-JPY, NZD-CHF, EUR-CAD) at a resolution of 2 minutes spanning 7 years were used. This amounted to approximately $10^6$ datapoints per exchange rate pair. The bandit problem using this data was set up as follows. Each exchange rate was thought of as a 2-armed bandit. It was imagined that the agent could make fictitious trades, and could either decide to buy a long call option (if they believe the rate will increase) and a short call option (if they believe the rate will go down). To turn this into a Bernoulli bandit problem, we ignore the scale of the change and provide a reward of 1 if the bandit predicted correctly the rate going up/down and 0 otherwise. When the rate remains the same, the agent receives a reward of 0 irrespective of their decision. For the purpose of the experiment we imagine the option length is 100 time ticks, so that the agent has to decide if the exchange rate will increase or decrease in 100 time ticks. Although this bandit scenario is not true to life, we believe that the underlying data should exhibit...
some of the characteristics of a switching system for which the algorithms were
designed [Preis et al., 2011]. We can not estimate a “true” average payoff at
each timestep, and so can not measure the regret of these algorithms, instead
we report the error. The results are shown in table 6.

Table 6: Results against Foreign Exchange Bandit Environment (number of
mistakes $\times 10^{-3}$±Std. Error)

| Name             | Error   | Name             | Error   |
|------------------|---------|------------------|---------|
| Global-CTS       | 351.9 ± 14.1 | Global-CTS2     | 358.0 ± 13.95 |
| PA-CTS           | 370.4 ± 13.7  | PA-CTS2         | 380.9 ± 12.5   |
| NP Global-CTS    | 348.2 ± 13.7  | NP PA-CTS       | 353.5 ± 13.8   |
| NP Global-CTS2   | 353.2 ± 13.4  | NP PA-CTS2      | 352.0 ± 13.9   |
| UCB              | 613.9 ± 17.1  | DiscountedUCB   | 606.3 ± 16.9   |
| Random           | 623.3 ± 14.1  |                  |             |

8 Conclusion

This paper has explored several algorithms using Thompson Sampling in con-
junction with Change Point detection. We have shown that they perform well
in the environments for which they are designed. Bandit scenarios based on
real-world data such as the Yahoo! dataset and the Foreign Exchange also
demonstrate their performance. They are shown not to perform as well as ap-
propriately tuned competing algorithms in the PASCAL challenge. However
our results suggest that a strategy that just tracks changes in the perceived
best arm (Global-CTS2,PA-CTS2), similar to Adapt-EvE, works well.
Since the model is extremely modular it is hoped that further assumptions can
be incorporated into the model to improve performance. It is also worth noting
that non-Bernoulli payoffs can just as easily be used, e.g. Normal distributed
payoffs. A Bayesian approach also avoids difficulties that arise with handling
false alarms in change point detection schemes. The algorithms have been de-
rived from simple models and so are theoretically motivated, however steps still
need to be taken to provide any theoretic justification for their performance.

References

Ryan Prescott Adams and David J.C. MacKay. Bayesian online changepoint
detection. Cambridge, UK, 2007.

Dukascopy. Dukascopy historical data feed. http://www.dukascopy.com/
swiss/english/marketwatch/historical/, 2012. [Online; accessed 05-Oct-
2012].

Paul Fearnhead and Peter Clifford. On-line inference for hidden Markov models
via particle filters. Journal of the Royal Statistical Society: Series B (Statis-
tical Methodology), 65(4):887–899, 2003. doi: 10.1111/1467-9868.00421. URL
http://dx.doi.org/10.1111/1467-9868.00421.
Paul Fearnhead and Zhen Liu. On-line inference for multiple change points problems. *Journal of the Royal Statistical Society B*, 69:589–605, 2007.

A. Garivier and E. Moulines. On Upper-Confidence Bound Policies for Non-Stationary Bandit Problems. *ArXiv e-prints*, May 2008.

Cédric Hartland, Nicolas Baskiotis, Sylvain Gelly, Michèle Sebag, and Olivier Teytaud. Change Point Detection and Meta-Bandits for Online Learning in Dynamic Environments. *CAp*, pages 237–250, 2007. URL [http://hal.inria.fr/inria-00164033](http://hal.inria.fr/inria-00164033).

Z. Hussain, P. Auer, N. Cesa-Bianchi, L. Newnham, and J. Shawe-Taylor. Exploration vs. exploitation pascal challenge. http://pascallin.ecs.soton.ac.uk/Challenges/EEC/, 2006.

Emilie Kaufmann, Nathaniel Korda, and Rmi Munos. Thompson sampling: An optimal finite time analysis. *CoRR*, abs/1205.4217, 2012. URL [http://dblp.uni-trier.de/db/journals/corr/corr1205.html#abs-1205-4217](http://dblp.uni-trier.de/db/journals/corr/corr1205.html#abs-1205-4217).

Lihong Li, Wei Chu, John Langford, and Xuanhui Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In Irwin King, Wolfgang Nejdl, and Hang Li, editors, *WSDM*, pages 297–306. ACM, 2011. ISBN 978-1-4503-0493-1. URL [http://dblp.uni-trier.de/db/conf/wsdm/wsdm2011.html#LiCLW11](http://dblp.uni-trier.de/db/conf/wsdm/wsdm2011.html#LiCLW11).

T. Preis, J. J. Schneider, and H. E. Stanley. Switching processes in financial markets. *Proceedings of the National Academy of Sciences*, 108(19):7674–7678, April 2011. ISSN 1091-6490. doi: 10.1073/pnas.1019484108. URL [http://dx.doi.org/10.1073/pnas.1019484108](http://dx.doi.org/10.1073/pnas.1019484108).

Morten Sorensen. Learning by investing: Evidence from venture capital. SIFR Research Report Series 53, Institute for Financial Research, May 2007. URL [http://ideas.repec.org/p/hhs/sifrwp/0053.html](http://ideas.repec.org/p/hhs/sifrwp/0053.html).

Ryan Turner, Yunus Saatci, and Carl Edward Rasmussen. Adaptive sequential Bayesian change point detection. In *Advances in Neural Information Processing Systems (NIPS): Temporal Segmentation Workshop*, 2009.

Robert C. Wilson, Matthew R. Nassar, and Joshua I. Gold. Bayesian online learning of the hazard rate in change-point problems. *Neural Comput.*, 22 (9):2452–2476, 2010. ISSN 0899-7667. doi: 10.1162/NECO_a_00007. URL [http://dx.doi.org/10.1162/NECO_a_00007](http://dx.doi.org/10.1162/NECO_a_00007).

Yahoo! Yahoo! webscope dataset ydata-frontpage-todaymodule-clicks-v1_0. [http://labs.yahoo.com/Academic_Relations](http://labs.yahoo.com/Academic_Relations), 2011. [Online; accessed 05-Oct-2012].

17