Non-linear Contextual Bandits

by

John Chia

B. Electrical Engineering, Ryerson University, 2009

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE STUDIES
(Computer Science)

The University Of British Columbia
(Vancouver)

April 2012

Abstract

The multi-armed bandit framework can be motivated by any problem where there is an abundance of choice and the utility of trying something new must be balanced with that of going with the status quo. This is a trade-off that is present in the everyday problem of where and what to eat: should I try a new restaurant or go to that Chinese place on the corner? In this work, a multi-armed bandit algorithm is presented which uses a non-parametric non-linear data model (a Gaussian process) to solve problems of this sort. The advantages of this method over existing work is highlighted through experiments. The method is also capable of modelling correlations between separate instances of problems, e.g., between similar dishes at similar restaurants. To demonstrate this, a few experiments are performed. The first, a synthetic example where the reward function is actually sampled from a Gaussian process, begs the question but helps pin down the properties of the algorithm in a controlled environment. The second, a problem where the objective is to aim a cannon at a distant target, shows how a well-defined objective, i.e., hit the target, can be used to speed up convergence. Finally, the third, an experiment with photographic post-processing, shows how the algorithm can learn from experience. The experiments demonstrate both the flexibility and the computational complexity of the model. This complexity means that problems such as the aforementioned restaurant problem, among others, are still future work.

Revision: 13

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

From whom to bank with, to what to wear in the morning, to the decision to be a law-abiding citizen or a criminal, life is about choice. Every choice we make involves a trade-off which helps us decide what choices to make. There are many types of trade-offs; for instance, there are trade-offs that balance two properties of an object, for example, the trade-off between durability and cost in consumer goods. This work is *not* concerned with such qualitative trade-offs; rather, it is concerned about a behavioural trade-off between risk and reward; specifically, it is concerned with the trade-off between behaving informed by *past experience*, i.e., exploitation, and behaving informed by *potential gains*, i.e., exploration.

This so-called *exploitation-exploration dilemma*, between choosing favorites and breaking new ground is a crucial consideration in this work. It appears nearly everywhere one looks; for instance, it appears in the choice of where and what to eat: should one try a new restaurant (explore) or go to the one that is known to be good (exploit)? Another example is in relationships: should one stay with a partner (exploit) or should one try a new partner (explore)? The trade-off is not even confined to human interaction – it has also been observed in nature. For instance, colonies of ants have been observed in distinct exploration and exploitation phases when searching for new nesting sites [22].

Obviously, what is typically sought in problems of choice is the best choice among possible alternatives. Hence, the aforementioned problems (e.g., choosing what to eat) are problems in optimization. It turns out that the optimal behavior

in *finding* the best choice is some balance of exploration and exploitation. Further, such an optimal strategy needs to minimize regret, i.e., that notion which tells us, in hindsight, we could have been better off with another choice.

The rest of this work concerns itself with one particular framework for processes involving exploration and exploitation. This framework is commonly called *multi-armed bandits* or just *bandits*, for short. It has been widely studied and applied to a diverse set of fields. For instance, some of the oldest work appears in the context of sequential experimental design where bandits were used to model the process of selecting a sequence of experiments [e.g., 12].

There has been a recent resurgence of interest in bandit problems perhaps due to its applicability to some problems on the Internet [e.g., 14, 20]; in particular, the field of recommendation systems, that often deals with problems of too much choice, may be set to hugely benefit from advances in bandit algorithms. This work is concerned with with applying the framework of *contextual* multi-armed bandits to problems where there may be a very large, perhaps infinite, number of choices.

The contextual bandit problem differs from the regular bandit problem by adding to each action *context*. This is similar in some ways to *features* in the field of supervised learning: they allow one to exploit dependencies between objects of interest. However, in bandits, features are used to model dependencies between choices rather than objects to classify.

In this work, an algorithm to solve the contextual bandit problem is introduced using a non-parametric statistical regression method named Gaussian process regression. Further, two applications of this algorithm are presented: an application to learn the targeting function of a cannon and an application to learn the white balance profile of a digital image sensor. It should be noted that the contextual bandit formulation presented here is very similar to recent work by Krause and Ong [16]. However, at the time this work was being developed, Krause and Ong had not yet published [16]; that is, this work was developed independently of Krause and Ong [16]. This work also contributes a note and experiments on the benefits of when the maximum reward value is known in advance.

What follows is organized into four chapters. The first describes the statistical data model used, i.e., Gaussian process regression, which is described both in a form that assumes real-valued observations and one that assumes binary obser-

vations. The next chapter concerns itself with defining the bandit problem more precisely than laid out above, and discusses a Gaussian process bandit algorithm. In the next chapter, several experiments with bandit problems are presented and discussed. Finally, a concluding chapter summarizes the work and lays out a few directions for future work.

Chapter 2

Data Model

In this chapter, a non-parametric non-linear regression framework is introduced based on a Gaussian process (GP). This will be used to model observed data and also, through kernels, to model correlations between inputs. A probit regression framework built on top of GP regression, from [5], is also described.

2.1 Notation

Scalar variables are denoted by symbols in italics while vectors are bold faced. To refer to the reward function itself, $r(\mathbf{x})$ is used. Context features, which are often multidimensional and include features of the choices (also called arms or actions), are represented by $\mathbf{x}$. To denote reward *observations* of this function, the symbol $\mathbf{r}$ is used. Subscript t is used as an iterator while subscript T is used to denote the current iteration.

2.2 Gaussian Process Model

The reward function is distributed according to a GP prior defined by a mean function $m(\mathbf{x})$ defined on an action/context $\mathbf{x}$ and a kernel function $k(\mathbf{x},\mathbf{x}')$ defined on an action/context $\mathbf{x}$ with respect to another action/context $\mathbf{x}'$.

$$r(\mathbf{x}) \quad \sim \quad \mathrm{GP}(m(\mathbf{x}), k(\mathbf{x},\mathbf{x}')). \tag{2.1}$$

The GP prior encodes certain global smoothness assumptions the function through the choice of covariance kernel.

The prior mean function may be set to zero without loss of generality. The kernel function is a chosen in order to capture correlations between input points $\mathbf{x}$ and $\mathbf{x}'$. Popular choices include the Gaussian kernel, the Matern kernel or even a linear kernel. In this work, the Gaussian kernel is used exclusively. It is given by:

$$k(\mathbf{x},\mathbf{x}') \quad = \quad \sigma_A^2 \exp\left(-\tfrac{1}{2}(\mathbf{x}-\mathbf{x}')^T\mathbf{D}^{-1}(\mathbf{x}-\mathbf{x}')\right) \tag{2.2}$$

where $\mathbf{D} \in \mathbb{R}^{D\times D}$ is a symmetric positive definite matrix, usually diagonal or at least diagonally dominant and $\sigma_A^2 \in \mathbb{R}$ is an amplitude parameter. Its purpose is to control the scale of the underlying input space, in a global sense. In other words, it determines which and to what extent each of the D input features is considered in norm of the difference $(\mathbf{x}-\mathbf{x}')$. These hyperparameters, i.e., $\mathbf{D}$ and σ_A can be learned by maximum likelihood [e.g., 23] or integrated out via, e.g., Bayesian Monte Carlo [5].

The kernel may be used to define a covariance matrix $\mathbf{K}(\mathbf{X},\mathbf{Y})$, or a kernel matrix, on a set of points. The value at the ith row and jth column of this matrix is given by evaluating the kernel function on the $\{i,j\}-th$ input points. That is, where the i-th column of, e.g., $\mathbf{X}$ is denoted $\mathbf{x}_i$,

$$[\mathbf{K}(\mathbf{X},\mathbf{Y})]_{i,j} = k(\mathbf{x}_i,\mathbf{y}_j) \tag{2.3}$$

A GP is defined as a collection of normal random variables. Usually, some of these random variables have been observed at known inputs $\{\mathbf{x}_i\}_{i=1}^T$; that is, $r(\mathbf{x})$ was sampled at $\mathbf{X}_{1:T} = \{\mathbf{x}_i\}_{i=1}^T$ to form $\mathbf{r}_{1:T}$. One may also wish to predict $r(\mathbf{x})$ for some set of inputs $\mathbf{X}$. This implies the following jointly normal model:[1]

$$\begin{bmatrix}\mathbf{r}\\ \mathbf{r}_{1:T}\end{bmatrix} \quad \sim \quad \mathrm{N}\left(0, \begin{bmatrix}\mathbf{K}(\mathbf{X},\mathbf{X}) & \mathbf{K}(\mathbf{X},\mathbf{X}_{1:T})\\ \mathbf{K}(\mathbf{X}_{1:T},\mathbf{X}) & \mathbf{K}(\mathbf{X}_{1:T},\mathbf{X}_{1:T})\end{bmatrix}\right) \tag{2.4}$$

[1] Note: $\mathbf{r}$ refers to a vector of observations of $r(\mathbf{x})$ rather than the function itself.

The predictive density is then formed by conditioning on $\mathbf{r}_{1:T}$:

$$\begin{aligned}
\mathbf{r}|\mathbf{r}_{1:T} &\sim \mathrm{N}\left(\mu_T(\mathbf{X}), \sigma_T^2(\mathbf{X})\right) && (2.5)\\
\mu_T(\mathbf{X}) &= \mathbf{K}(\mathbf{X}_{1:T}, \mathbf{X})\mathbf{K}(\mathbf{X}, \mathbf{X})^{-1}\mathbf{r}_{1:T} && (2.6)\\
\sigma_T^2(\mathbf{X}) &= \mathbf{K}(\mathbf{X}, \mathbf{X}) - \mathbf{K}(\mathbf{X}, \mathbf{X}_{1:T})\mathbf{K}(\mathbf{X}_{1:T}, \mathbf{X}_{1:T})^{-1}\mathbf{K}(\mathbf{X}_{1:T}, \mathbf{X}) && (2.7)
\end{aligned}$$

This density distribution predicts both the value of rewards, i.e., $\mu_T(\mathbf{X})$ and the uncertainty in this prediction, i.e., $\sigma_T^2(\mathbf{X})$.

2.3 Probit Regression

The probit regression model given here can be found described in more detail in [e.g., 5, 8]. The problem concerns inferring a latent function $r(\mathbf{x})$ from binary observations. These observations impose an ordering on the values of $r(\mathbf{x})$; that is, the dataset consists of pair comparisons of the form "$\mathbf{x}$ is preferred to $\mathbf{x}'$", which is written as $\mathbf{x} \succ \mathbf{x}'$ and is taken to be a binary random variable taking value 1 when true and 0 otherwise. Further, these observations are considered to be noisy and are said to imply a function $u(\mathbf{x})$ corrupted by Gaussian noise given by ε.

$$u(\mathbf{x}) = r(\mathbf{x}) + \varepsilon \quad (2.8)$$

Under this model, one may write the probability of $\mathbf{x} \succ \mathbf{x}'$ as,

$$\begin{aligned}
P\left(\mathbf{x} \succ \mathbf{x}' | r(\mathbf{x}), r(\mathbf{x}')\right) &= P\left(u(\mathbf{x}) > u(\mathbf{x}') | r(\mathbf{x}), r(\mathbf{x}')\right) && (2.9)\\
&= \Phi\left(\frac{r(\mathbf{x}) - r(\mathbf{x}')}{\sqrt{2}\sigma}\right) && (2.10)
\end{aligned}$$

The latent function $r(\mathbf{x})$ is assumed to come from a GP, that is $r(\mathbf{x}) \sim \mathrm{GP}(0, k(\mathbf{x}, \mathbf{x}'))$. The prior distribution on the vector $\mathbf{r} = \{r(\mathbf{x}_i)\}_{i=1}^T$ is given by:

$$P(\mathbf{r}) = |2\pi\mathbf{K}|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}\mathbf{r}^T\mathbf{K}^{-1}\mathbf{r}\right) \quad (2.11)$$

The MAP estimate of $\mathbf{r}$ is given by maximizing the following unnormalized posterior distribution on $\mathbf{r}$.

$$p(\mathbf{r}|\mathcal{D}) \quad \propto \quad p(\mathbf{r}) \prod_{t=1}^{T} p(\mathbf{x} \succ \mathbf{x}' | r(\mathbf{x}_t), r(\mathbf{x}'_t)) \tag{2.12}$$

This can be done in a number of ways. The Laplace approximation is one that is convenient and easily derived. It is also computationally efficient. The downside is that it can introduce huge approximation errors when the posterior distribution (in log space) is not approximated well by the first and second terms of its Taylor series expansion.

The Hessian and gradient of the approximated log posterior are given in [5] and can be used with some kind of iterative method (e.g. LBFGS) to find the mode $\mathbf{r}_{MAP}$. The details of the derivation may be found in [5].

Chapter 3

Non-linear Contextual Bandits

Contextual bandit problems are simply collections of multi-armed bandit problems where correlations between both problems and choices (or actions) are learned and exploited to improve performance.

In multi-armed bandit problems, an agent is concerned with acting in a way informed by the outcome of its past actions. A typical example of a multi-armed bandit problem is that of playing an array slot machines with the intention to maximize returns [2]. In this setting, the agent starts with no prior knowledge and must play one of k slot machine "arms" over T rounds. Each slot machine is modelled by a random variable r_k and each play of a machine samples from a distribution D on $\{r_1, \cdots, r_k\}$. However, only one r_k is revealed at each round. The agent must learn to favor the highly rewarding distributions over time to be successful.

In the original contextual bandits extension, also called associative reinforcement learning [1] or bandits with side information [10], features are assigned to each action and correlations between actions are exploited to improve performance. More recently, the notion of context was extended by introducing features, which are themselves distributed randomly, that correlate between bandit problems [19].

This chapter is mainly concerned with presenting a non-linear contextual multi-armed bandit algorithm. First, the contextual multi-armed bandit problem, the contextual multi-armed algorithm and measures of success in bandit settings are discussed. Then, the concept of utility functions is introduced and three utility functions are discussed. Finally, the non-linear contextual multi-armed bandit al-

gorithm is presented and discussed.

Abuses of Notation

While reading the rest of this work, one may find references to seemingly multiple versions of the reward function defined in the preceding model section, e.g., $r(\mathbf{x})$ or $r(a,\mathbf{x})$. This is merely an abuse of notation as the reward model is defined generally enough to subsume any additional arguments into additional dimensions of the single input vector $\mathbf{x}$.

3.1 Definitions

The context space is defined over any arbitrary set $\mathbb{X}$. Actions $a \in \{1,\dots,k\}$ are assigned features from the context space.

Definition 1. (General Multi-armed Bandit Problem) *The problem proceeds in repeated rounds. At each round, a sample is drawn from a distribution P over $\{\mathbf{x}_1,\cdots,\mathbf{x}_k,r_1,\cdots,r_k\}$. All contexts $\{\mathbf{x}_i \in \mathbb{X}\}_{i=1}^{k}$ are observable but only one reward $r_a \in \mathbb{R}$ is revealed based on the action $a \in \{1,...,k\}$ chosen.*

The non-contextual problem, i.e., that used by Auer et al. [2], is specified by setting the contexts to the action indices, i.e., $\mathbf{x}_a = a$ and $p(\mathbf{x}_a = a) = 1$. In this variant, which is the simplest possible variant, actions are identifiable only by an arbitrary index variable. By adding features to each action, e.g., as considered by Auer [1], the resulting bandit problem closely resembles the form of Bayesian optimization considered by, e.g., Jones [15]. Still, at this point, all stochasticity is confined to the reward distributions; that is, action features $\mathbf{x}_i$ are pre-determined.

The fully-random contextual bandit problem allows both the rewards and the context to vary randomly. That is, between iterations, the full context of the bandit problem may change, including actions available and the effect of those actions on the world. It is important to note that, in this variant, the context is given, i.e., provided by the world, and includes features providing environmental information as well as information about each action. That is, the context can be explicitly composed by concatenating an action's features $\mathbf{a}_i$ and additional environmental context $\mathbf{c}$ such that $\mathbf{x}_i = [\mathbf{a}_i\ \mathbf{c}]^T$. Note that for a given $\mathbf{c}$ and assuming that action

features $\mathbf{a}_i$ are pre-determined, the problem is exactly the same as a deterministic contextual bandit problem. Hence, one can view the fully-random contextual bandit problem as a collection of dependent deterministic bandit problems.

Definition 2. (Multi-armed Bandit Algorithm) *A multi-armed bandit algorithm plays rounds $t \in \{1,...,T\}$ of the general multi-armed bandit problem (or a subcase thereof) by choosing an action a and collecting a reward $r_{a,t}$ based on the current context $\{\mathbf{x}_{a,t}\}_{a=1}^{k}$ and past observations $\{(\mathbf{x}_1, a_1, r_1), \cdots, (\mathbf{x}_{t-1}, a_{t-1}, r_{t-1})\}$.*

3.1.1 Measures of Success

The general idea is to quantify the notion of regret, i.e., that one could have chosen a better action with perfect foresight, and then to develop some quantifiable optimality criterion around that notion. In the bandits literature, there are two main forms of regret: so-called simple regret and cumulative regret.

The former is motivated by applications in global optimization where the goal is to find the single best point after a fixed number of iterations. While both require a balance between exploration and exploitation, in the case of simple regret, the maximum need only be hit once to achieve optimal performance. The definition of simple regret is omitted but may be found in [e.g., **?**].

Cumulative regret is motivated by problems in online optimization where the goal is to maximize the reward at every iteration. Unlike in the case of simple regret, where the cost of exploration is hidden by the most successful round, the cost of all exploration is included in this measure.

Definition 3. (Cumulative Regret) *Given a general multi-armed bandit problem and a multi-armed bandit algorithm, the cumulative regret of the latter on the former at round T is:*

$$R_T = \sum_{t=1}^{T} \left(\max_a r(a, \mathbf{x}_t) - r(a_t, \mathbf{x}_t) \right) \tag{3.1}$$

3.2 Utility Functions

In regret minimization, especially in the cumulative case, the core problem is handling the trade-off between exploitation of past knowledge, i.e., making known profitable actions, and exploration of options unexplored, i.e., trying new actions for the first time. The following approach is presented as an iterated optimization of a function $u(\mathbf{a}|\mathcal{D},\mathbf{x})$ that changes as the observations $\mathcal{D}$ grow. This function is designed to model the exploration versus exploitation dilemma and combine the value of each into a single quantity called utility; hence, the function is referred to as the utility function. Three utility functions are given below: upper-confidence bound (UCB), probability of improvement (PI) and expected improvement (EI).

In discrete actions spaces, Auer et al. and its extensions [e.g., 1] can be said to be similar to the above iterated utility function optimization approach. The model they use, predominantly, is least squares which limits them to cases where the rewards are linear with respect to context. In this section, an extension to that family of work is presented which uses a non-linear data model in a non-parametric formulation (section 2.2). This allows it to handle general problems involving arbitrary smooth reward functions.

3.2.1 Upper Confidence Bound

The upper confidence bound (UCB) function (equation 3.2) is a probabilistic upper bound of the reward function, i.e., it gives a probabilistic estimate for the maximum reward for the entire input domain of the reward function. It is a linear combination of two quantities one each for exploitation and exploration. When the model is normal, these are given by the mean $\mu(\mathbf{x})$ and variance $\sigma(\mathbf{x})$ of the posterior predictive distribution which may be found in equation 2.6 and equation 2.7 respectively.

$$\mathrm{UCB}(\mathbf{x}) = \mu(\mathbf{x}) + \alpha\sigma(\mathbf{x}) \tag{3.2}$$

This choice of utility encapsulates information about past rewards and future potential rewards very distinctly into two terms. The trade-off between exploitation of past rewards and the exploration of promising potential is made through the

parameter α. Intuitively, by setting $\alpha = 0$ the algorithm can be made to be purely exploitative, i.e., always going with the action which gave the most reward. By selecting α very large, the algorithm will be purely explorative, i.e., always selecting actions that it has never seen before.[1]

The setting of α can have a serious effect on the performance of the algorithm. A choice too low can cause excessive exploitation; conversely, a choice too high can cause excessive exploration. Note that fixing, e.g., $\alpha = 2$ does not result in a 95% bound. To guarantee a high-probability bound, one must set α to be an increasing sequence, e.g., Srinivas et al. [25] proves an asymptotic high-probability no-regret bound for $\alpha_t = 2\log\left(|D|t^2\pi^2/6\delta\right)$ where δ may be arbitrarily set to the desired error rate of the bound.

3.2.2 Probability of Improvement

The probability of improvement utility function was originally considered by Kushner [17] and extended by Mockus [21] to higher dimensions. A target reward ζ is carefully chosen; this is somewhat isomorphic to UCB's α parameter [15]. The function is then defined as the probability that choosing $\mathbf{x}$, i.e., sampling $r(\mathbf{x})$, will give an observation of *at least* ζ. In the case of normally distributed observations, it is given by the following where $\Phi(x)$ refers to the Normal cumulative distribution function.

$$\text{PI}(\mathbf{x}) \quad = \quad P\left[r(\mathbf{x}) \geq \zeta\right] = 1 - \Phi\left(\frac{\zeta - \mu(\mathbf{x})}{\sigma(\mathbf{x})}\right) \tag{3.3}$$

Intuitively, this function will initially assign high utility to actions close to the current best actions and the quantity $\frac{\zeta - \mu(\mathbf{x})}{\sigma(\mathbf{x})}$ will grow. As the algorithm continues to choose points in an area around a local maximum, the uncertainty in this area will reduce accordingly. Eventually, the quantity will be dominated by a minuscule $\sigma(\mathbf{x})$; at this point, the algorithm should proceed elsewhere.

Again, the actual behavior of the algorithm using this utility function is highly dependent on the selection of the target value, ζ. There are cases where the selection of ζ is easy: for instance, when the maximum reward (but not the action that

[1] In a discrete action space, it will eventually try all actions. In this case, ties will be broken arbitrarily, e.g., by floating point error.

gives it) is known then one may set $\zeta = r_{max}$. In general, one does not know such information. For these cases, one might choose some margin δ over the best reward previously seen $r^+ = \max_{r \in \mathcal{D}} r$. One might also wish to use a percentage margin, that is $\delta = \varepsilon |r^+|$ where $\varepsilon < 1$. The latter is stated below.

$$\zeta = r^+ + \varepsilon |r^+| \tag{3.4}$$

There is an isomorphism between ζ in the probability of improvement utility and α in the upper confidence bound utility. Consider the following optimization problem,

$$\alpha \quad = \quad \max_{\mathbf{x}} \frac{\zeta - \mu(\mathbf{x})}{\sigma(\mathbf{x})} = \frac{\zeta - \mu(\mathbf{x}^*)}{\sigma(\mathbf{x}^*)} \tag{3.5}$$

One can show that the maximum of the equivalent UCB problem is given at a^*, i.e., $\zeta = \max_{\mathbf{x}} \mu(\mathbf{x}) + \alpha\sigma(\mathbf{x})$. The two utility functions remain distinct for any single choice of α or ζ since the mapping $\zeta = f_t(\alpha)$ depends on the iteration t [15].

3.2.3 Expected Improvement

The expected improvement function builds on the probability of improvement heuristic. It can be derived by taking the expectation of some improvement δ on the best reward seen r^+ with respect to the distribution $\mathrm{N}(\delta | r^+ - \mu(\mathbf{x}), \sigma^2(\mathbf{x}))$ [15]. It is stated in terms of $u = -\frac{r^+ - \mu(\mathbf{x})}{\sigma(\mathbf{x})}$. Note, $\Phi(u)$ and $\phi(u)$ refer to the normal cumulative distribution function and normal probability density functions respectively.

$$\mathrm{EI}(\mathbf{x}) \quad = \quad \sigma(\mathbf{x}) \left(u\Phi(u) + \phi(u) \right) \tag{3.6}$$

There is no need to know the optimum beforehand as the function implicitly considers all possible improvements, i.e., the derivation includes an integration over all improvements δ. Still, some authors, [e.g., 5], recommend an additional term δ' be added to increase exploration behavior; that is:

$$u \quad = \quad \frac{r^+ + \delta' - \mu(\mathbf{x})}{\sigma(\mathbf{x})} \tag{3.7}$$

3.3 The Algorithm

The "utility optimization multi-armed bandit algorithm" (UOMABA, Algorithm 1) proceeds in iterations. Each iteration begins with an optimization of one of the aforementioned utility functions to select a new sampling point. At that point, a new sample of the application-dependent reward function, i.e. $r(\mathbf{a}_t|\mathbf{x}_t)$, is taken. The model is updated with the new observation and the algorithm continues ad infinitum. A pseudo-code listing is given in algorithm 1 for the general case given no assumptions about the data model or utility function. In experiments, three utility functions and three models are compared in three applications.

Algorithm 1 UOMABA

Require: $\mathcal{M}, u(a), r(a)$
  for $t = 1 \ldots T$ **do**
    Observe $\mathbf{x}_t$
    Set $a_t = \text{argmax}_{a \in \mathcal{A}} u(a|\mathbf{x}_t, \mathcal{D}, \mathcal{M})$
    Sample r_t, e.g., evaluate $r(a_t|\mathbf{x}_t)$
    Update model $\mathcal{M}$
  end for

3.4 Related Work

The multi-armed bandit problem has its roots in the Second World War where it was considered impossible to solve [13]. It remained as such until the 50s when Robbins [24] proved an optimal strategy exists for the multi-armed bandit problem as originally formulated. The strategy, which was based on calculating a set of indices for each possible option and choosing the option with the highest index, remains at the core of algorithms today.

The work of Auer et al. [2] introduced *Hedge* and *EXP3*, a pair of related algorithms that use a stochastic strategy of arm-pulling such that an arm was pulled with probability related to its proportion of the gains accumulated at any point in time. Derivative works improved the basic *Hedge* algorithm, e.g., *NormalHedge* [7] outperforms *Hedge* when the number of arms is large. Others were more concerned with theory, e.g., EXP4.P [4] has high-probability guarantees on performance.

The so-called "UCB strategy", which uses a statistical upper bound on each

arm, was first introduced by Lai and Robbins [18]. Auer et al. considered normally distributed rewards and provides an upper confidence bound approach in [3]. This lead to the use of reward functions which were defined on either discrete sets of arms [1] or a continuum of arms [11] with features defined over a linear space.

The ε-greedy approach used a single-parameter ε control the amount of time the agent spends being greedy, i.e., where the agent makes purely exploitative choices. The remainder of the agent's behavior would be exploratory. Sutton and Barto [26] considers this approach. The phrase "contextual bandits", referring to the fully-random contextual bandit problem, was originally coined in a paper concerning an ε-greedy type algorithm by Langford and Zhang [19]. Since then, Li et al. [20] have considered the fully-random contextual multi-armed bandit problem in a linear space. Although, in that work, a non-linear projection is performed as a pre-processing step.

The published work that is most similar to this work consider smooth reward functions distributed according to a Gaussian process. Srinivas et al. consider the problem in a discrete choice setting using the UCB utility function in [25] and also includes theoretical regret bounds for several kernels. Bull considers the use of the expected improvement utility function in [6] and also provides theoretical bounds on asymptotic performance.

Chapter 4

Experiments

This chapter consists of experiments that aim to demonstrate the properties of non-linear contextual bandits and also present some examples of problems that may be solved with them. The first section contains experiments which were performed with synthetic data – data generated from the model itself. These experiments serve as illustrative evidence of the properties of the contextual bandit algorithm under various conditions. The next section contains an application involving the firing of a cannon at a target. Given the target's location, the contextual bandit algorithm learns to choose actions that hit the target. Finally, the last section contains an application to image processing. Given some color information about the image, the contextual bandit algorithm learns to pick the correct white balance settings for images. Each section proceeds in a similar manner: from the simplest formulation of the problem, i.e., a single pre-determined context with action features, to the fully-random context formulation of the problem. A summary of the experiments,

§	Problem	Pure context features	Action features
4.1	Synthetic	Synthetic	Synthetic
4.2	Cannon aim	Target coordinates (2)	Cannon inclination & velocity
4.3	White balance	Average color (2)	Color channel scaling (2)

Table 4.1: Summary of the pure context and action features for each experiment that was performed.

including the nature of their contexts, is given in Table 4.1.

4.1 Samples from a Gaussian Process

The objective of the experiments in this section were primarily to demonstrate properties of the algorithm, e.g., when using each of the three utility functions (PI, EI and UCB). Each experiment runs off data that is given by a function $r(\mathbf{x})$ that is sampled from a Gaussian process with known hyperparameters. Examples of function samples are given in Figure 4.1. UOMABA (algorithm 1) is then configured with some particular choice of hyperparameter learning (including an oracle learner), some utility function $u(\mathbf{x})$ and some choice of model $\mathcal{M}$. The algorithm is then run for T iterations, restarted then run again for R runs. The results (i.e., cumulative regret) are reported as averages over runs to reduce variability.

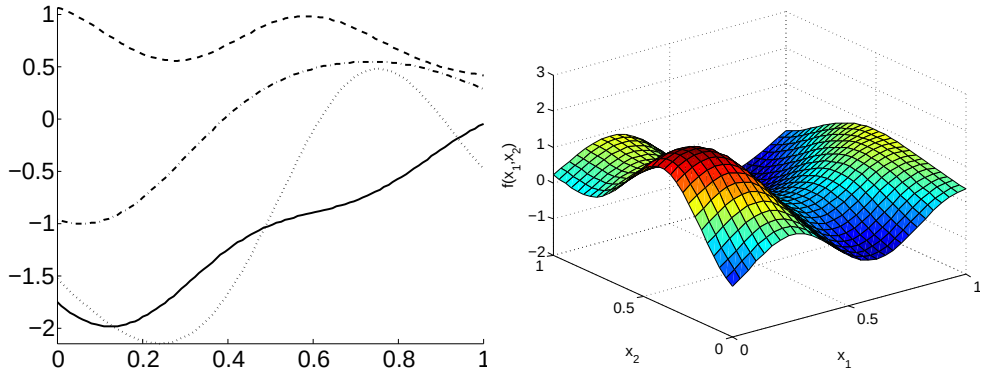

Figure 4.1: Appearing above are examples of reward functions that were drawn as Gaussian process samples. On the left are samples in 1D with length scale $1/4$ while on the right is a sample in 2D with length scale $1/4$ in both directions (isotropic).

The rest of the section proceeds in subsections covering a particular area of focus. First, the performance in terms of regret of three utility functions (EI, PI and UCB – as discussed in section 3.2) are compared in one and two dimensions. Next, the performance of the Gaussian process model is compared to that of the parametric model. Then, several methods to learn the hyperparameters of the GP

are compared. Finally, the full contextual bandit problem is considered where each iteration may involve an entirely new randomly chosen context.

4.1.1 Utility Functions: UCB vs PI vs EI

This experiment was conducted to compare the performance of the probability of improvement (PI), expected improvement (EI), and upper confidence bound (UCB) utility functions. The experiment is conducted as described previously, in independent runs with regret results reported in averages across runs. This is captured in Figure 4.2 where at some iteration t_0, the average regret across all runs at $t = t_0$ is plotted. (Note that the standard deviation of this estimate could also have been given but this was omitted so as not to clutter the plots.) In all runs, the algorithm uses the GP model.

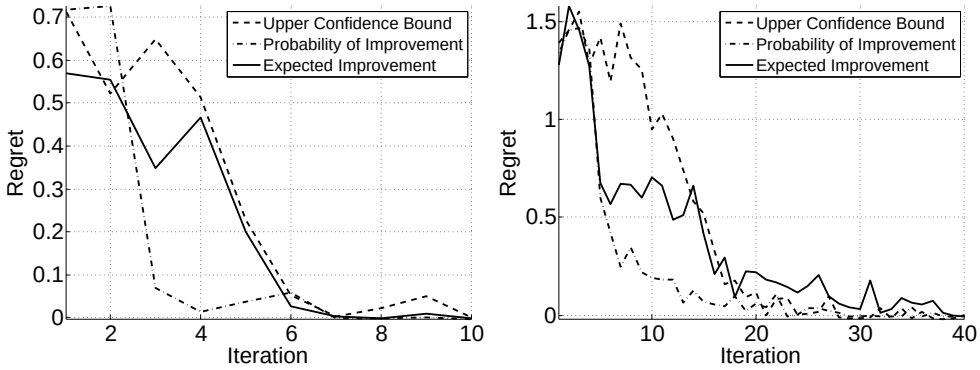

Figure 4.2: Experiments in one-dimension and two-dimensions where the reward surface is a noise-less function sampled from a Gaussian process. In this case, PI is allowed to use the known maximum value of the reward – this helps it beat both of its competitors. UCB was run with a parameter taken from [9]. EI was run without an extra exploration parameter.

The UCB Parameter

In Figure 4.2, the UCB heuristic was not able to outperform either EI or PI (at least when the maximum reward is known). It is known that with the choice of UCB pa-

rameter that was used (from [9]) that the asymptotic performance is guaranteed to be no-regret – but what about its finite-time performance? This experiment aims to demonstrate that the choice by Chu et al. [9] may be empirically defended by comparing the performance of a number of choices of the UCB parameter. Specifically, the choices that are considered are: two constant values (2 and 5), a parameter sampled from the Beta$(2,3)$ distribution, the choice given by Srinivas et al. [25] and the choice given by Chu et al. [9]. The results are given as an averaged-over-runs regret plot Figure 4.3.

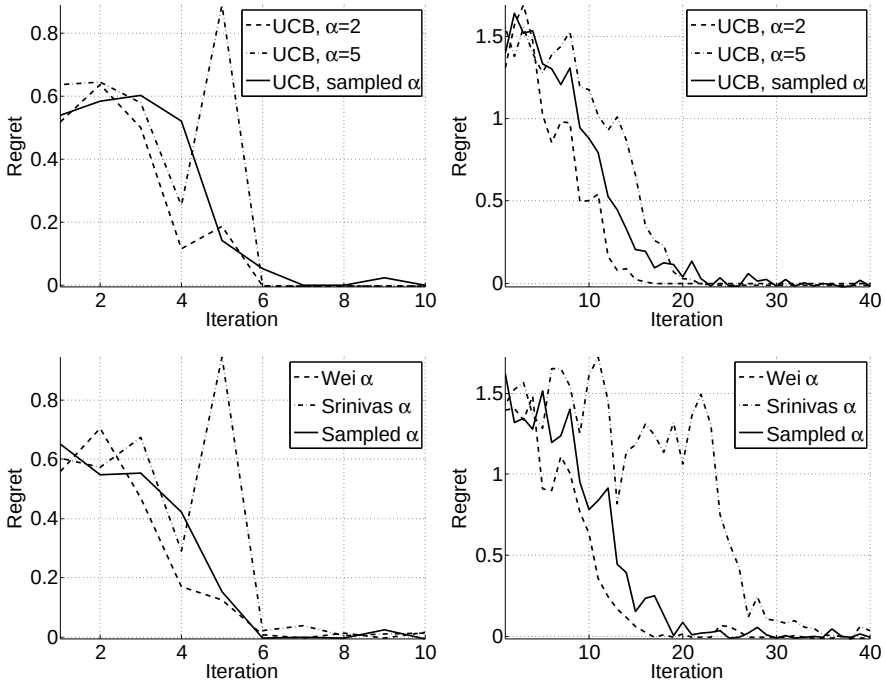

Figure 4.3: Experiments to quantify the performance of choice in UCB parameter, in one (left) and two (right) dimensions. Wei's choice in [9] seems comparable to the static choice $\alpha = 2$ at least in this low-dimensional finite-time experiment.

The PI Advantage when the Target is Known

The probability of improvement utility function allows the easy integration of information related to the maximum of the reward function. Using this information, it is able to form a more accurate balance of exploitation and exploration. For instance, it can stop exploring totally when the target has been reached. This leads to it consistently outperforming both UCB and EI in this experiment and also every upcoming experiment. This experiment aims to highlight what benefit that extra information has for PI in an environment where everything is synthetically controlled. The results are given again as an averaged-over-runs regret plot (Figure 4.4).

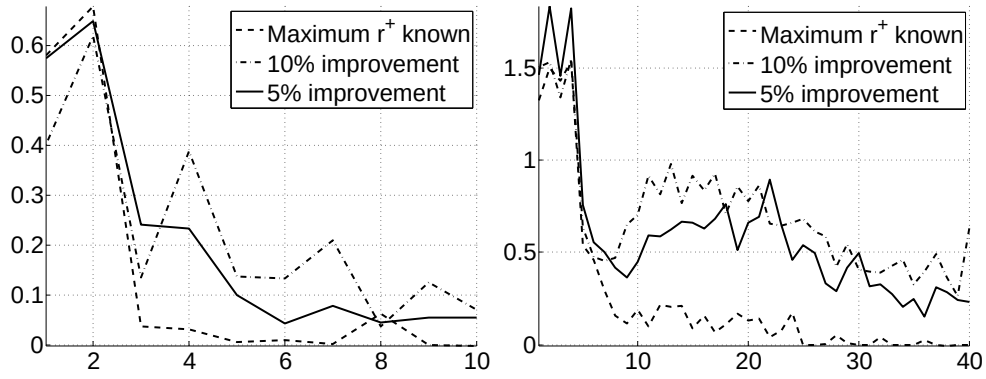

Figure 4.4: Experiments illustrating the effect of a known reward maximum in one and two dimensions respectively. Note that only the value of the maximum is known, not its location. The extra information is extremely helpful at achieving zero regret sooner. The intuitive explanation is that this extra information allows PI to "know" when to stop exploring. This end of exploration, which all these utility optimization bandits suffer from eventually, may not be the desired behavior if, for instance, one desires information about *all* the maximum points of the reward function.

4.1.2 Hyperparameter Learning

This experiment focuses on comparing two methods (MAP learning and a simple heuristic) for estimating the hyperparameters, i.e., length scale and magnitude, of the Gaussian process from which the reward function is assumed to be sampled. Also, the case where the hyperparameters are known is used as an optimal reference.

In the case of MAP learning, an inverse gamma prior is placed on the length scale hyperparameter $\mathbf{D}$ (in Equation 2.2) which was set to be isotropic (i.e., $\mathbf{D} = \lambda\mathbf{I}$). The uniform distribution was used as a prior on the magnitude σ_A. Gradient methods may be used to find the mode of the posterior distribution on the hyperparameters. The book of Rasmussen and Williams [23] may be consulted for the explicit details.

This experiment introduces a heuristic that can be used to set the length scale of a GP in an efficient manner with, as will be shown, competitive performance. It can be inspired by reasoning about the geometric density of kernels. There do not appear to be any existing work in the literature that has presented this heuristic thus far.

In a few words, the heuristic says to set the length scale inversely proportional to the global density of samples. Recall that GP regression places a kernel at each sample. With just a few samples, the length scale is set rather large so that the large regions of unexplored space will take on values inferred from data rather than the value of the GP prior mean function. As more samples come in, the length scale may be shrunk.

The results are presented as averaged-over-runs regret plots given in Figure 4.5 and Figure 4.6.

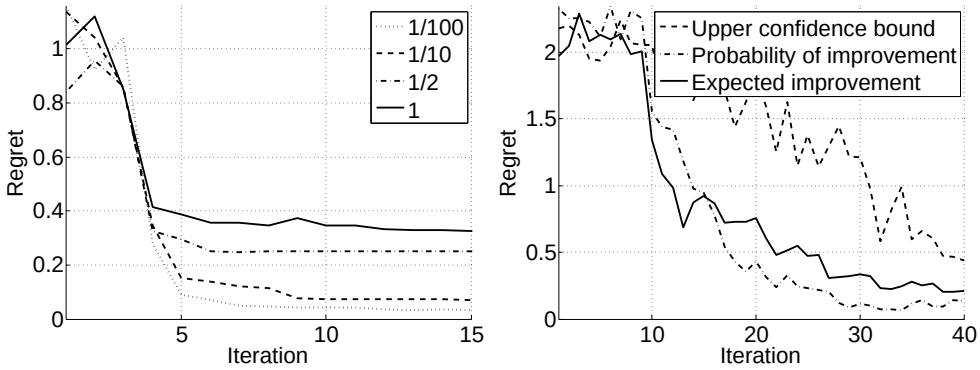

Figure 4.5: (Left) Regularization was needed to stabilize the numerical problem. Too much regularization resulted in much poorer performance. A regularization constant of 1/100 was eventually used. (Scales less than 1/100 were not possible in general.) (Right) MAP estimation did not seem to favour any particular utility function.

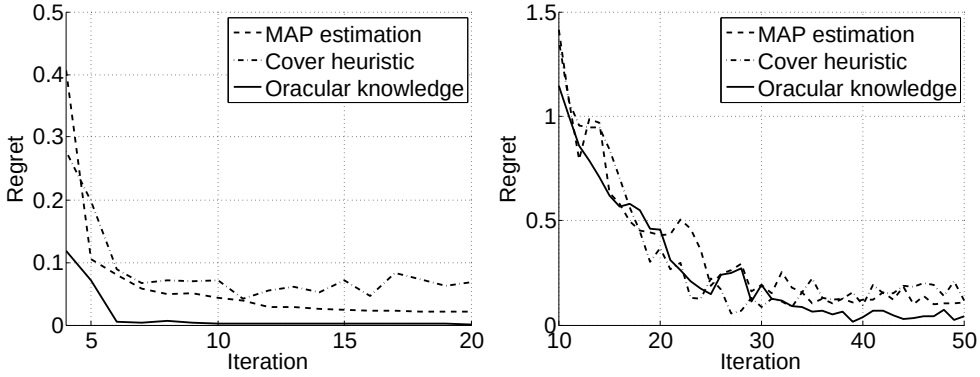

Figure 4.6: Experiments comparing MAP estimation to a heuristic ("cover") and oracle knowledge. The PI utility function was used. Without oracle knowledge of hyperparameters, zero instantaneous regret may take awhile to achieve as both MAP estimation and the simple heuristic both fail in this short but simple test run.

4.1.3 Effects of Random Context

While many of the properties of the deterministic context problem apply to the random context problem, there are significant differences that can be highlighted with numerical experiments. The first and most important difference is in the difficulty of the problem. Since the context is random and sampled from an unknown distribution, the algorithm must solve a potentially wildly different deterministic bandit problem at every iteration. In the worst case, an algorithm must solve an infinite number of deterministic bandit problems to solve the random context problem completely. For this experiment, the GP assumption gives us constraints on smoothness such that the problem should be much easier than the worst case.

The setup is as follows. The reward function remains sampled from a GP. Its domain is expanded to comprise both action features and at least one dimension representing a global random context. As before, at each round, contexts $\mathbf{x}$ are sampled uniformly. The bandit algorithm may then choose an action, but its access to the reward function is even more limited, that is, it may only evaluate the function in the "slice" implied by the sampled context $\mathbf{x}$. For instance, in a 2D problem (e.g., Figure 4.1 on the right letting $\mathbf{x} = [x_1 \; x_2]^T$) the random context would be a single number (e.g. $x_1 = 0.5$) and the resulting ordinary bandit problem (e.g. choosing x_2) would be the 1D "slice" at, e.g., $x_1 = 0.5$.

The results are presented yet again as averaged-over-runs regret plots given in Figure 4.7. PI maintains an impressive performance margin over both EI and UCB in these experiments. Again, PI was able to use information about the optimum that EI and UCB couldn't which better informed its exploration-exploitation balance.

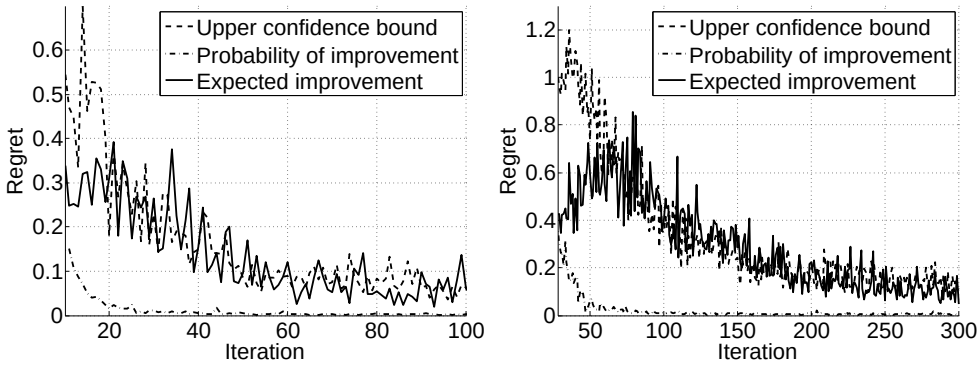

Figure 4.7: Experiments with a one-dimensional random context. Actions have features in one dimension (left) and two dimensions (right). PI was yet again given information about the maximum and managed to increasingly outperform EI and UCB as the dimension of the problem scales.

4.2 The Cannon Problem

This set of experiments attempts to highlight the bandit algorithm's performance at solving the cannon problem. The cannon problem is a basic targeting problem involving projectile motion. It is a basic element of popular games like *Angry Birds* or *Worms*. In the cannon problem, the objective is to aim a cannon, or any other projectile launching device, at a distant fixed target. In the bandit formulation, the aiming of the cannon is treated as an action while the target's location is treated as additional random context. A pictorial summary of the problem is given in Figure 4.8.

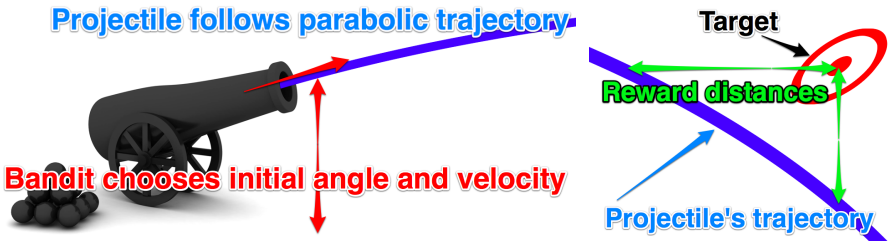

Figure 4.8: Illustration showing the basic set up of the cannon problem at the cannon end (left) and the target end (right). The bandit chooses an action by selecting an initial angle and velocity, illustrated in red. The projectile is launched and follows a parabolic trajectory, illustrated in blue, that may come arbitrarily close to the target, shown as a red bulls-eye. The reward is given as the *minimum* of the squared reward distances, illustrated in green.

The rest of the section proceeds first by discussing the construction of the reward function which is used to give feedback to the algorithm about how close the projectile came to the target. Next, constraints on the distribution of targets are discussed. Hyperparameter learning and cross-validation are discussed next. Finally, two experiments are performed: the first considers a simpler "fixed targets" variation where the target location is fixed between iterations while the second considers the fully random variation where target locations may change at each iteration.

4.2.1 Rewards

The cannon problem can reframed in terms of a contextual bandit problem. For a given target location, i.e., $\mathbf{c} = [x_0, y_0]^T$, an agent must choose an angle α and a velocity v to hit a target at $\mathbf{c}$ or equivalently to minimize the following cost function, i.e., a negative reward function, with respect to a $\mathbf{c} = [x_0, y_0]^T$. Note that in the fully random case, the target location, i.e., $\mathbf{c}$, is randomly chosen.

$$
\begin{aligned}
r(v, \alpha, x_0, y_0) &= -\min\{d_x(v, \alpha, x_0, y_0), d_y(v, \alpha, x_0, y_0)\} && (4.1)\\
d_x(v, \alpha, x_0, y_0) &= \left(x_0 - \frac{v_0^2}{g}\sin\alpha\cos\alpha\right)^2 && (4.2)\\
d_y(v, \alpha, x_0, y_0) &= \left(y_0 - \left(\frac{g x_0^2}{v_0^2\cos^2\alpha} + x_0\tan\alpha\right)\right)^2 && (4.3)
\end{aligned}
$$

Only evaluations of the reward function above are given to the agent. The agent learns to solve each problem by choosing an action and observing thereafter its reward. In the case of random contexts, it also learns to correlate target positions, which are given as additional context, so that its past experience may be exploited.

It is notable that this reward function is not strictly smooth: in particular, it has singularities at the intersection of the parabolas. Hence, it cannot be entirely modelled by GP regression. However, the important parts, i.e., the regions around the maximum, *are* in fact smooth.

4.2.2 Targets

The targets one is able to hit depend on the domain of actions. For these experiments, the domain of actions was limited to $(v, \alpha) \in [0, 1] \times [0, \pi/2]$. Correspondingly, this limited the range of targets. One can show, e.g., by numerical simulation, that the range of targets for this action domain is accurately bounded by a quadratic function. See figure 4.9 for details.

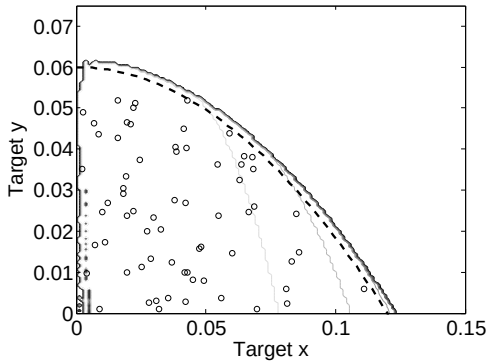

Figure 4.9: The region of targets able to be hit with actions $(v, \alpha) \in [0,1] \times [0, \pi/2]$ lies under the solid curve. The dotted line is the actual boundary used. Target locations, shown as circles, were generated from the uniform distribution and rejected outside the region.

4.2.3 Hyperparameters

As in the previous experiment with a synthetic function, several methods for learning hyperparameters were considered. MAP estimation and the heuristic given in Section 4.1.2 were performed in an identical manner as the previous experiment.

Cross-validation was performed for the GP model's length scale. A model was trained a set of randomly selected objective samples and tested on a separate set of sample points. The result is given in Figure 4.10 where the cross-validated mean squared error of the regression model on the test points is given. The function was determined to be isotropic in the action dimensions and target location dimensions separately; hence, they appear simultaneously in the plots below.

To avoid over-fitting, the simplest model is desired that still performs well. For this experiment, the simplest model is considered to be the one with the largest length scale. Further, the length scale was chosen to be the largest whose cross-validation error bars (representing standard error) still includes that with lowest estimated mean squared error.

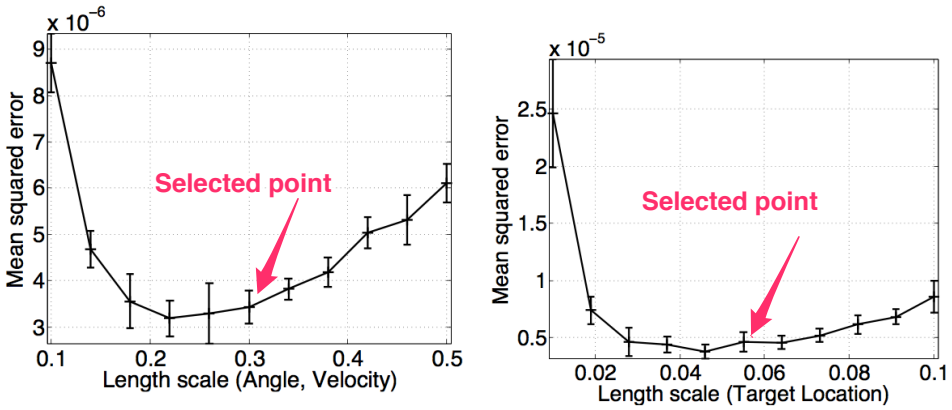

Figure 4.10: Cross-validation for the length scale in the action dimensions (left) and the target location dimensions (right). The function within each of these two dimensional subspaces is assumed to be isotropic. (This assumption was validated by fixing three of the four parameters and cross-validating on the remaining one.) The selected point was chosen according to Occam's razor to minimize over-fitting.

4.2.4 Fixed Targets

In the deterministic version of this experiment, the target is randomly chosen at the start of each problem and fixed for the remainder. As in the previous set of experiments, to reduce the variance of the sampling distribution, a number of independent runs with different randomly chosen targets are solved. The results are reported as averaged-over-runs regret.

Each independent problem is initialized by sampling a target location (x_0, y_0), i.e., as described in Section 4.2.2. The bandit algorithm is initialized with a set of no-cost actions: nine initial points were used in these experiments. At each iteration thereafter, the algorithm chooses an action and receives a reward related to the distance away from the target, i.e., it evaluates the function given in Equation 4.1.

Figure 4.11 compares the performance of three hyperparameter learning methods: the kernel covering heuristic, MAP estimation and cross-validation estimates. Utility functions were also compared: as in the previous experiments, PI was able to soundly outperform both EI and UCB by knowing information about the maxi-

mum; see Figure 4.12.

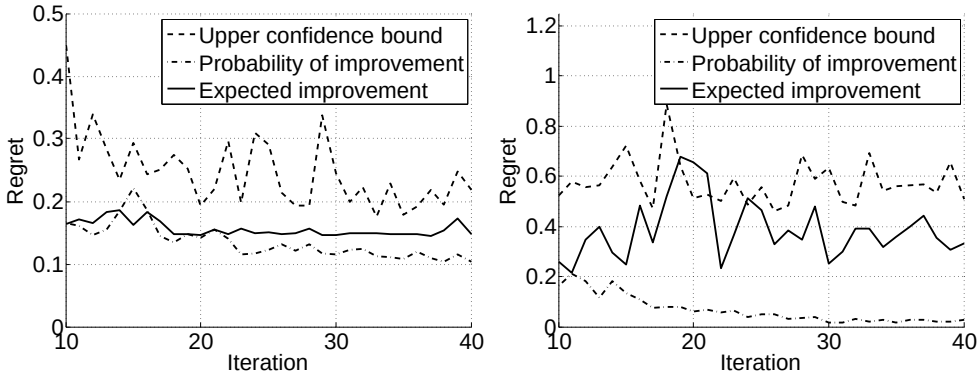

Figure 4.11: Deterministic cannon experiment with MAP estimation (left) and estimation with the cover heuristic (right). While the cover heuristic does not perform as well as the length scale estimate from cross-validation in Figure 4.12, it does manage to outperform MAP estimation making it a reasonable alternative when cross-validation is not feasible. For instance, this may be the case in high dimension or when the problem is truly online.

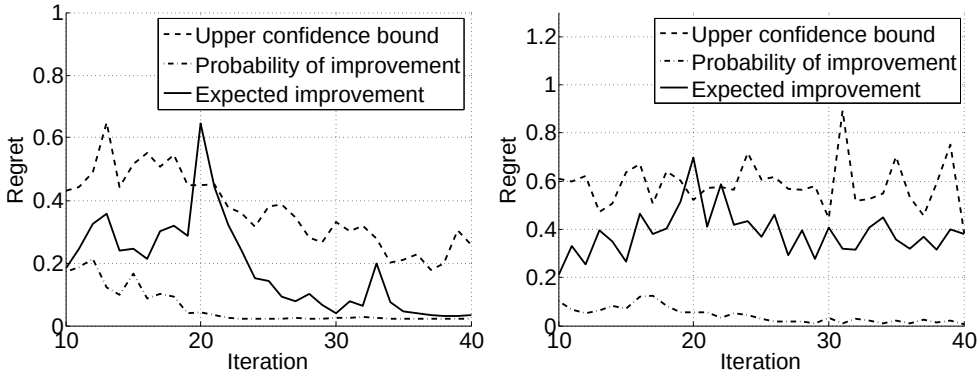

Figure 4.12: (Left) Deterministic cannon experiment using the length scale determined by cross-validation (i.e., 0.3). The three utility functions (PI, UCB and EI) are compared. The result is very similar to the synthetic experiment with PI using its knowledge of the maximum to its advantage and both EI & UCB turning in performance similar to each other. (Right) Same experiment with the length scale halved (i.e. set to 0.15). PI still managed to converge while neither EI or UCB even show indications of convergence after 40 iterations. Hence, when the maximum is known, PI may a choice that is more robust to inaccurate data modelling.

4.2.5 Random Targets

In the random version of this experiment, the target's location is re-sampled at each iteration. The target is still explicitly given to the algorithm, but its location changes every iteration. To be effective at this solving this variation, the algorithm must associate target locations in addition to learning the target-specific reward function. The rest of the problem proceeds as before, i.e., to reduce the variance of the results, a number of independent runs are made.

Figure 4.13 compares the performance of three utility functions at solving the problem with random targets. PI was yet again observed to soundly outperform both EI and UCB by knowing information about the maximum.

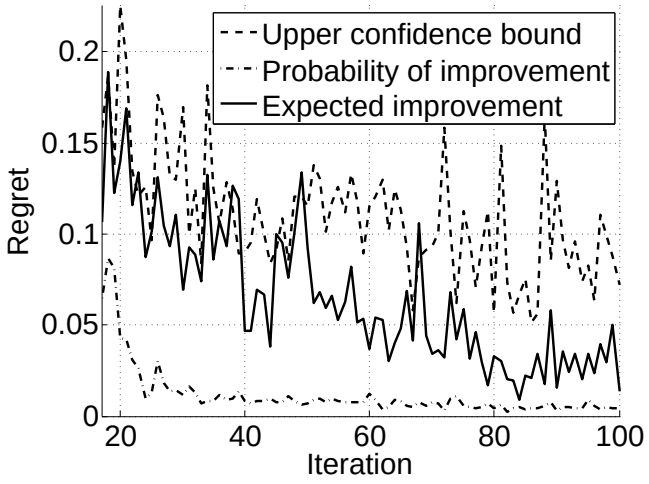

Figure 4.13: Stochastic cannon experiment is performed on the full four-dimensional reward (Equation 4.1) but where two dimensions are randomly chosen at each iteration. Hyperparameters were chosen according to the results of cross-validation, see Section 4.2.3.

4.2.6 Comparison to Supervised Learning

This problem could be framed as a supervised learning problem, i.e., regression for a function that maps targets to actions. One could get this mapping by allowing the learner to observe actual points on the trajectory of the projectile, i.e., the closest point to the target on the trajectory. The learner would then select actions based

on the recommendations of the function learned. However, without prior knowledge on the structure of the mapping (e.g., that it is parabolic) or some other way to initialize the mapping (e.g., with a sufficiently large set of randomly selected actions), the learner could spend an extraneous amount of time trying actions that have already been observed to fail. For instance, if GP regression were used with a constant zero mean function, the learner would not be able to try anything but the $\{0, 0\}$ action assuming no other initialization took place. So, while the problem could be framed as a supervised learning problem, it would need some way to gather an initial comprehensive training dataset. This consideration is exactly where the bandit algorithm fits in.

4.3 Parameter Optimization with Human Feedback

In this section, the following image processing problem is considered: given an image captured under the illumination of an unknown light source, choose an appropriate scaling for the red, green and blue channels of the digital sensor such that what was perceived as white during capture is reproduced as white in the output image. This is a task known in digital photography as choosing the "white balance". The white balance is usually chosen from a set of light source presets described as, e.g., incandescent, fluorescent, sunlight and cloudy. Choosing the wrong preset can make an image appear with unnatural colours. An example of the same scene shot under two presets, of which only one is correct, is given in Figure 4.14 to highlight these effects.

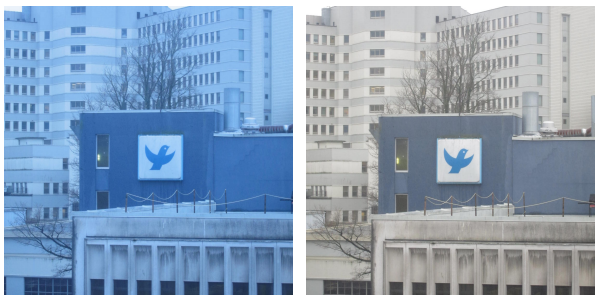

Figure 4.14: An example that highlights the importance of correctly choosing white balance. Shown is the same scene shot under two different white balance settings: on the left, the incorrect setting was chosen and the image has an overall blue tint. The correct setting appears on the right.

There exist more compelling image processing problems that one might have selected instead of the mundane task of white balancing. The white balance problem has two properties that can motivate its use in this experiment. The first is that the optimum for the problem is clearly and objectively defined; that is, white is well known and easily identified. Contrast this with, e.g., the problem of learning image exposure, tone or saturation; in these cases, there is more than one optimum, the optimums are subjective and the optimums vary across users. The white balance problem allows one to do multiple session learning across potentially multiple

users without explicit user modelling. This saves complexity which in turn feeds into the second desirable property of the white balance problem: its simplicity. The problem, for the great variety of real world light sources, can be solved with just two dimensions. (The general problem is three dimensional.) This simplicity keeps the computational cost down and allows the use of a complex non-parametric model such as Gaussian process regression.

The section begins with a short description of the user preference learning framework that was used. It is based on user ratings of so-called pair comparisons. The white balance problem is then described in more detail. Next, the independent form of the bandit problem is considered and experimental results are presented. Finally, the dependencies are introduced between each previously independent problem and experimental results for this are presented and discussed.

4.3.1 Implied Reward Function

Analogous to the reward functions of previous experiments, the reward function for this experiment is considered to be defined over actions and additional problem dependent context. In this case, the bandit chooses how to scale each color channel while image color features provide the additional context.

Unlike in previous experiments, there is no explicit reward function. Instead, the reward function is implied by human behavior. At each iteration of the algorithm, one action is chosen by the algorithm and the result, e.g., the processed image, is presented to a user through an interface depicted in Figure 4.15. The user is asked to evaluate a pair comparison that indicates whether the action is better or worse than the best historical action. That is, a user is presented with two images side-by-side. One represents the best historical thus far while the other represents what the bandit has determined has the most current utility in trying. The user then indicates which of the two is, e.g., the better representation of white. In this case, the comparison should be objective but there is no reason why this same mechanism couldn't be used to collect preferences rather than beliefs. An example session for a single image is given in Figure 4.16.

The framework that was used to model user beliefs is the same as that of Brochu [5] that was used to model user preferences and was described in Section 2.3. It

Figure 4.15: An annotated screenshot from the user interface. The historically best action was used to render the image on the left while the candidate action proposed by the bandit algorithm was used on the right. The user may select between the left and right images, or may indicate that the settings are too close to call by selecting the middle button.

can be loosely described as a probit GP that is fit from a set of pair comparisons. It should be noted that the preference operator $\succ$ is not in general transitive. That is, if we have $a \succ b \succ c$ then it doesn't follow that $a \succ c$. This is often the case, in particular, when the preferences are weak. In some sense, weak preferences exist in a state of uncertainty like quantum particles. This inspired the development of random weak preferences; that is, if a is weakly preferred to b then only sometimes $a \succ b$ holds true. The rest of the time the converse holds true, i.e., $b \succ a$. Hence, the user has actually three choices for each pair comparison: the left one, the right one or neither has the "whitest whites" .

4.3.2 Image White Balancing

White balancing is a calibration problem. The full response of digital image sensors is pre-calibrated at the factory using media with known reference color values.

Figure 4.16: An example session in the pair comparison interface showing four iterations of the white balance problem on a single image. At each iteration, represented by a pair of images, the historically best action is displayed on the left and is compared to the action proposed by the bandit algorithm on the right. The user indicates his preference (or in this case, which one he believes is more accurate) by pressing either the left arrow or the right arrow. He can also indicate uncertainty by choosing the symbol $\simeq$.

However, it is impossible to use one calibration setting to support multiple light sources with vastly different emission spectra. For example, sunlight is full spectrum while incandescent light is strongest in red and green wavelengths resulting in a "yellowish" light source. If you've ever taken a picture indoors with the white balance setting on your camera off, you will have seen an example of this miscalibration. The goal of white balancing is to scale the response of the digital

sensor's red, green and blue receptors to match our visual cortex's so that what we perceive as white is reproduced as white.

While white balance problems are optimizations over three parameters in general, in most image processing software the problem is simplified to a two dimensional one involving the "color temperature" and the "greenness" of a light source. Although this is a drastic assumption, it works well for the vast majority of light sources, with some exceptions, e.g., exotic situations such as infrared photography. The assumption that two parameters suffices is adopted for the purposes of this work.

The 3D problem seeks to find scaling coefficients $\{k_r, k_g, k_b\}$ such that a color value $c_0 = \{r_0, g_0, b_0\}$ is transformed by a linear scaling:

$$c_0^b = \begin{pmatrix} k_r & 0 & 0 \\ 0 & k_g & 0 \\ 0 & 0 & k_b \end{pmatrix} c_0 \tag{4.4}$$

The 2D problem formulation that is used in this work ties the red and blue scaling coefficients together and scales the green channel with the remaining parameter independently resulting in an optimization over the "temperature" and "greenness" respectively. That is, the 2D white balance problem uses the transformation:

$$c_0^b = \begin{pmatrix} T & 0 & 0 \\ 0 & G & 0 \\ 0 & 0 & (1-T) \end{pmatrix} c_0. \tag{4.5}$$

4.3.3 Experiments with Independent Problems

The independent problem setting is the same as that of [5]; that is, information that provides (in the language of Li et al. [20]) image-dependent "context" is not used, i.e., information about the particulars of the current image with respect to others previously seen is not used. The proceeding section, on the use of such information to improve performance between problems, is what really concerns

this work; however, the current section is important to do well as everything in the next section depends on the performance of the independent form of a bandit problem.

Implementation Notes

For these experiments, the interactive bayesian optimization framework of Brochu [5] was used with some customizations and a custom front-end. In particular, the DIRECT global optimizer was not used as it did not improve the efficiency of each iteration or converge consistently. Instead, a brute force search in a slightly perturbed enumerated grid of points was used. The random perturbations, which were Gaussian, allowed the selection of actions in between enumerations which effectively increases the resolution of the method.

Although framework that was used (i.e. Brochu [5]) includes support for learning hyperparameters, not nearly enough data was collected to make this feasible. Hence, the hyperparameters were chosen by hand. Length scale changes resulted in a trade-off between the mean time to convergence and the variance in the same quantity. It was found that $\frac{1}{4}$ was a liveable compromise. The magnitude and noise scale were left at the default 1.0 and 1×10^{-2} respectively. Cross-validation was not performed due to the cost and time involved in having a human evaluate pair comparisons.[1]

In previous experiments, the clear winner in terms of utility function was PI. In this experiment, since the objective maximum is not known in advance, the EI utility function was used. See Section 4.1, Figure 4.4.

Experimental Results

Three images (Figure 4.17 and Figure 4.18) in two lighting conditions (incandescent light and sun light, respectively) were used and six independent trials were run for each of the six combinations. In each trial, a user evaluates pairs of possible white balance settings until convergence, that is, until he or she is satisfied that the image has accurate colors. The number of pairs until convergence is used as the main measure of performance. In table 4.2, the iterations to converge for each

[1] Even one as cheap to employ as a grad student.

	CAT	Garbage Truck	Blue doll
Indoors	17	7	24
	2	7	2
Mean: 9.72	10	4	4
Std. dev: 6.79	10	16	8
	25	6	4
	8	11	10
Outdoors	2	9	3
	20	3	8
Mean: 8.00	6	5	10
Std. dev: 6.35	25	12	6
	2	6	2
	4	7	14

Table 4.2: Iterations to convergence for six trials per combination of three images in two lighting conditions without information sharing between trials.

of the 36 trials performed as well as the mean iterations to convergence for both lighting conditions.

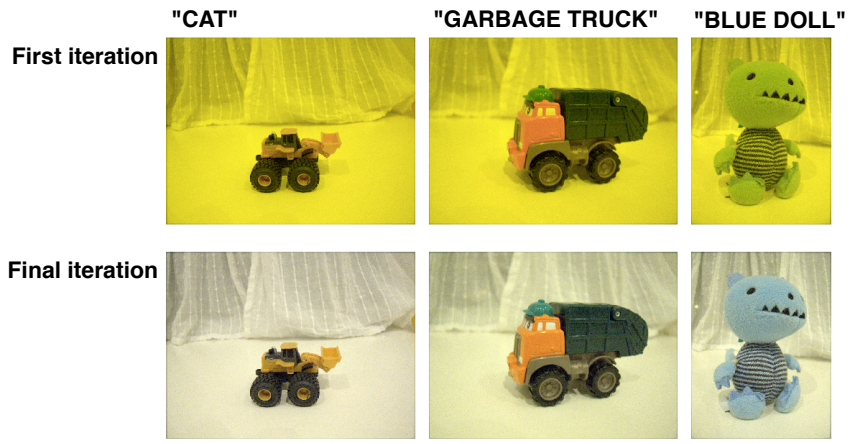

Figure 4.17: Before and after results. Each column represents an independent run. The images on top were processed using a default white balance setting while the images on the bottom were processed using the white balance setting learned by the bandit algorithm in conjunction with the user evaluating a sequence of paired comparisons. These images were all captured under incandescent light.

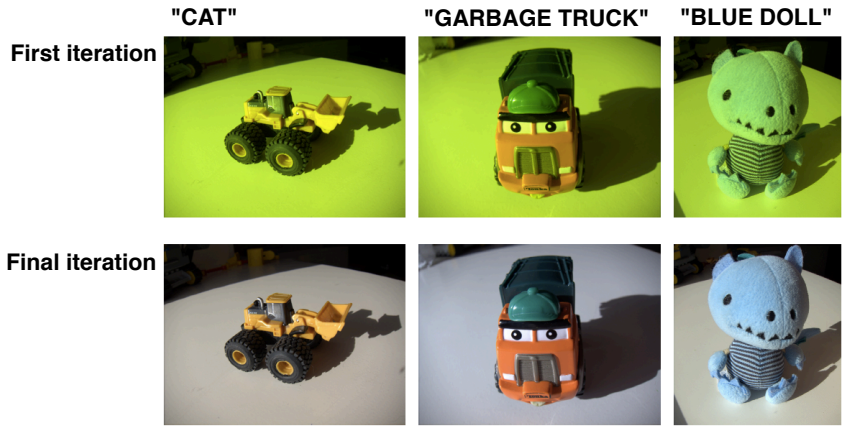

Figure 4.18: As figure 4.17 but captured under sun light. Top row: at iteration one; bottom row: at the final iteration.

4.3.4 Image Features

To model dependencies between images (which were previously treated as independent problems), information is sought that will help correlate between similar actions. Since the problem is concerned with balancing the color of the images, the image's geometry can safely be ignored and the color information can be considered alone. This is a clear advantage as color is much easier to measure and quantify than geometry.

Features for each image are constructed from a reference processed with a common white balance setting. The image is considered in the hue-saturation-value color space. To increase the sensitivity of the image features to the hue rather than saturation and value, only points in the upper subsection of the HSV cone are used; that is, points below a certain value threshold are ignored. The justification for this comes from observing that color sensitivity of human vision is substantially reduced when considering dark colors.

4.3.5 Experiments with Dependent Problems

For these experiments, the data model was expanded by adding image features. These were used to augment action features where each augmented feature vector was the result of concatenation of the existing two-dimensional action (i.e., the white balancing scale parameters) with a fixed feature vector that was specific to each subject/lighting condition combination. This setting differs from the "contextual bandit" setting of Krause and Ong [16] or Li et al. [20] in one major way. Instead of a new problem ("context") arriving at each iteration, the problem is allowed to remain the same for as many iterations as needed before proceeding to the next problem. This is a result of the practical difference between the objectives of this work and that of [16] or Li et al. [20]: whereas this work considers bandits as a tool for helping users with some task (e.g. white balancing an image), the work of, e.g., [20] considers bandits as a tool for an organization to improve its click-through-rate by serving users relevant content. In the latter, the number of iterations is unconstrained and multiple users may be interacting with the system at once; while in the former, there are only a few iterations that may be done before the user loses patience and only a single user at a time is considered. An illustrative

summary of the experiment is given in Figure 4.19.

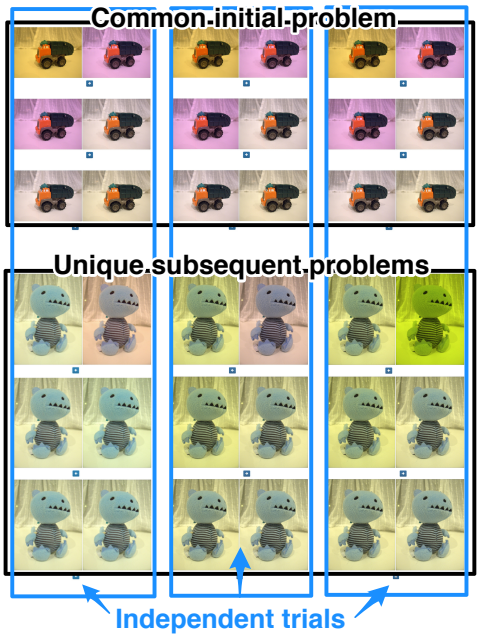

Figure 4.19: An illustration of the method used to conduct experiments with dependent problems. The columns of paired images represent independent trials each of which shares a common initial problem. The number of iterations until convergence for the subsequent problem is recorded separately from the initial problem. Note that this example was contrived for illustration purposes and the number of iterations shown is not consistent with actual performance.

Experimental Results

Two of the three images used in the previous independent setting experiment were selected based on their proximity in feature space. Likewise, two sources of light were considered for each image: incandescent light and sun light. The experiment proceeds in four "configurations" covering each image preceding and proceeding from the other under the same lighting. The first image is processed from an otherwise unintialized state. After the user finds the right white balance for this image,

Configuration	Doll (in) Truck (in)	Truck (in) Doll (in)	Truck (out) Doll (out)	Doll (out) Truck (out)
Initial problem	10	7	11	15
Trial 1	1	4	1	2
Trial 2	2	2	7	1
Trial 3	6	6	4	3
Trial 4	1	5	4	4
Trial 5	2	9	5	2
Mean	2.40	5.20	4.20	2.40
Std	2.07	2.59	2.17	1.14

Table 4.3: Iterations to "convergence" (defined as when a "similar" pair is presented) for five trials of four configurations in a dependent problems setting. In each configuration, both subjects were considered under the same source of lighting ("in" for indoors incandescent light, "out" for outdoor sunlight). Two orderings per configuration were considered in a number of trials.

the state of the algorithm is saved. From this saved state, a number of separate trials are run and the performance of these runs are used to measure performance.

The configurations, as well as convergence results, are summarized in table 4.3. Convergence is defined differently than in the previous experiment: this experiment considers convergence as when two similar images are presented. The reason for this peculiar definition of convergence is that if one were to use the convergence criterion from the previous experiment (i.e., when the white in an image appears perceptually white) then every trial would have converged at one iteration. The first iteration of each image in each configuration is given in figure 4.20.

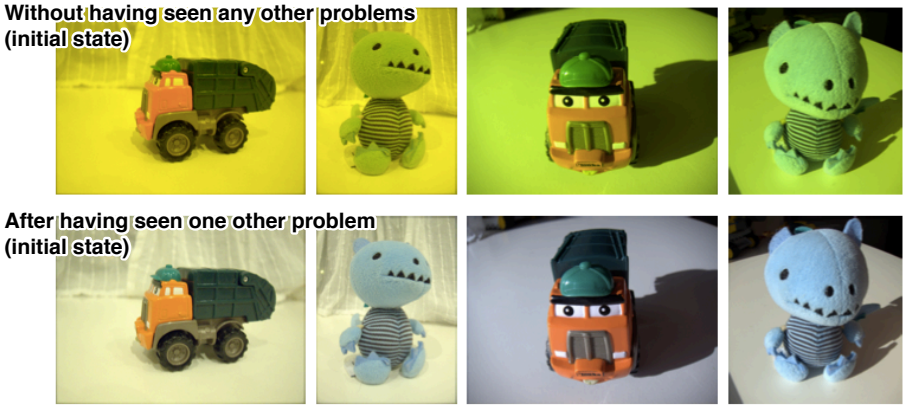

Figure 4.20: Illustrative example of the performance improvement when the algorithm is initialized on related problems. The algorithm used information from the iterations of the first problem to get better results at the first iteration of the second problem; that is, it used information learned from a different but related image shot under the same lighting conditions to improve performance on the second image. In the first row, the images from the first iteration of the algorithm, i.e., without pre-learning on a different image is presented. In the second row, the first iteration after pre-learning on a separate image is shown. The pair of columns on the left were shot under indoor light; on the right, they were shot under outdoor light.

Chapter 5

Conclusion

In this work, a contextual bandit algorithm based on the use of Gaussian process regression was presented[1] and several applications were used to illustrate its properties in experiments. In addition to the novelty of applying a bandit algorithm based on Gaussian processes to the contextual problem, this work also includes results that show how, in certain applications where the maximum possible reward value is known, the use of the probability of improvement utility function can improve performance by an incredible amount. Further, a simple heuristic for hyperparameter selection in situations where data is scarce was shown in experiments to be competitive with MAP estimation, in terms of regret performance, but at a much lower computational cost.

5.1 Future Work

There are many more compelling applications of the bandit framework than could be considered here. This work has been constrained by a number of factors, some that were technical and some that were legal. Of the technical constraints, the most damning was the cubic complexity of Gaussian process regression. Of the legal constraints, perhaps not the most damning but certainly the most annoying was the lack of large, publicly available datasets. These are certainly, among others, areas for future work.

[1] At the time that this work was being done, the very similar work of [16] was unknown and unpublished.

5.1.1 Applications with Big Data and Parametric Models

Huge data has huge potential for the development of knowledge-based learning. This is true not just for bandits but for the entire machine learning community. The privacy obstacles to releasing such a dataset are not trivial and, with the de-anonymized Netflix dataset still fresh in mind, may not be resolved any time soon. Still, a standard dataset would have immense utility to this field of research and is perhaps what prevents a great many potential applications of bandits from being explored.

5.1.2 Regret Bounds when Using PI in Favourable Conditions

This work managed to show that when the target is known, i.e., the value of the reward function at its maximum is known, the probability of improvement utility function outperformed the expected-improvement and upper-confidence-bound utility functions by what seems to be a significant margin. Further, the experimental results (e.g., Section 4.2.5 or Section 4.1.3) suggest that the performance difference becomes more significant as the problem complexity increases, i.e., the dimensions of the problem increase. This suggests that the result given by [25], i.e., there is an $O\left(\sqrt{t\left(\log t\right)^{d+1}}\right)$ bound on regret, may be improved if it is assumed that the target is known[2].

5.1.3 Robust Regression with Heavy-tailed Distributions

Under the UCB utility function, when using a constant parameter, [15] showed that entire regions of the action space could be ruled out prematurely early. This lead to an algorithm which never converged to the optimal action. This could be attributed to a failure of the model to accurately represent uncertainty when the number of samples is quite low. It would be interesting to see if and how this changes by using a regression model with fatter tails. For instance, a Bayesian might use the T-distribution for the data likelihood in a kernelized regression framework. In regression, this is known to be more robust to outliers. In bandits, this property

[2] Actually, to do this, one may be able to exploit the isomorphism between UCB and PI to adapt an existing UCB proof.

of accounting for outliers may make it less susceptible to prematurely ruling out promising areas of the action space.

5.1.4 User Modelling and Preference Learning

One key limitation of the experiment on the use of bandits to white balance images (Section 4.3) was its inability to combine information about *preferences* rather than *beliefs*. That is, in the parlance of photography, this would mean collecting information about preferences on the temperature of the image rather than on the user's opinion about how neutral its tones are as Section 4.3 does. Much of the bandit algorithm used in Section 4.3 *does* support the learning of preferences. If the user wishes to pursue what he considers the perfect color temperature for a scene, he may do that with the current system. Further, the system is context-dependent so it may learn image-dependent preferences. What is missing is to integrate user features into context such that the system becomes capable of learning user-dependent preferences across *multiple users*. This would be useful, e.g., assuming that scaling issues can be resolved, in a new kind of image editor that recommends image settings based on opinions of "the crowd".

5.1.5 Connection with Classification

The white balance bandit problem (Section 4.3) is, in general, a contextual bandit problem defined on a continuous action space. For all practical intents and purposes, there are only a finite number of light source spectra that need to be accounted for. Hence, it could be framed in terms of a discrete choice contextual bandit problem. One can show that a discrete choice contextual bandit problem is a generalization of online classification that includes features on the classes (or, in bandit terminology, actions) as well as the ability for classes to be defined in terms of their features. Future work may want to explore this connection further.

Bibliography

[1] P. Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3:397–422, 2003. → pages 8, 9, 11, 15

[2] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. Gambling in a rigged casino: The adversarial multi-armed bandit problem. Technical Report NC2-TR-1998-025, 1998. → pages 8, 9, 11, 14

[3] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Journal of Machine Learning Research*, 47: 235–256, May 2002. → pages 15

[4] A. Beygelzimer, J. Langford, L. Li, L. Reyzin, and R. E. Schapire. Contextual bandit algorithms with supervised learning guarantees. *Journal of Machine Learning Research*, 15:19–26, 2011. → pages 14

[5] E. Brochu. *Interactive Bayesian Optimization: Learning User Preferences for Graphics and Animation*. PhD thesis, University of British Columbia, 2010. → pages 4, 5, 6, 7, 13, 34, 37, 38

[6] A. D. Bull. Convergence rates of efficient global optimization algorithms. Technical Report arXiv:1101.3501v2, 2011. → pages 15

[7] K. Chaudhuri, Y. Freund, and D. Hsu. A parameter-free hedging algorithm. In *Advances in Neural Information Processing Systems*, pages 297–305, 2009. → pages 14

[8] W. Chu and Z. Ghahramani. Preference learning with gaussian processes. In *International Conference on Machine Learning*, pages 137–144, 2005. → pages 6

[9] W. Chu, L. Li, L. Reyzin, and R. E. Schapire. Contextual bandits with linear payoff functions. *Journal of Machine Learning Research*, 15:208–214, 2011. → pages vi, 18, 19

[10] C. chun Wang, S. R. Kulkarni, and H. V. Poor. Bandit problems with side observations. *IEEE Transactions on Automatic Control*, 50:338–355, 2005. → pages 8

[11] V. Dani, T. P. Hayes, and S. M. Kakade. Stochastic linear optimization under bandit feedback. In *Conference on Learning Theory*, pages 355–366, 2008. → pages 15

[12] J. Gittins and D. Jones. A dynamic allocation index for the sequential design of experiments. In J. Gani, editor, *Progress in Statistics*, pages 241–266. North-Holland, Amsterdam, NL, 1974. → pages 2

[13] J. Gittins, K. Glazebrook, and R. Weber. *Multi-armed Bandit Allocation Indices*. John Wiley & Sons, Ltd., 2 edition, 2011. → pages 14

[14] T. Graepel, J. Q. Candela, T. Borchert, and R. Herbrich. Web-scale Bayesian click-through rate prediction for sponsored search advertising in Microsoft Bing search engine. In *International Conference on Machine Learning*, pages 493–498, 2010. → pages 2

[15] D. R. Jones. A taxonomy of global optimization methods based on response surfaces. *J. of Global Optimization*, 21:345–383, December 2001. → pages 9, 12, 13, 46

[16] A. Krause and C. S. Ong. Contextual gaussian process bandit optimization. In *Advances in Neural Information Processing Systems 24*, pages 2447–2455. 2011. → pages 2, 41, 45

[17] H. J. Kushner. A new method of locating the maximum of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, 86: 97–106, 1964. → pages 12

[18] T. L. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6:4–22, 1985. → pages 15

[19] J. Langford and T. Zhang. The epoch-greedy algorithm for multi-armed bandits with side information. In *Advances in Neural Information Processing Systems*, 2007. → pages 8, 15

[20] L. Li, W. Chu, J. Langford, and R. E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *World Wide Web*, pages 661–670, 2010. → pages 2, 15, 37, 41

[21] J. Mockus. Application of bayesian approach to numerical methods of global and stochastic optimization. *Journal of Global Optimizatin*, 4(4):347 365, 1994. → pages 12

[22] S. C. Pratt and D. J. Sumpter. A tunable algorithm for collective decision-making. *Proc Natl Acad Sci U S A*, 103(43):15906–15910, Oct. 2006. ISSN 0027-8424. → pages 1

[23] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006. → pages 5, 21

[24] H. Robbins. Some aspects of the sequential design of experiments. *Bulletin of the AMS*, 58:527–535, 1952. → pages 14

[25] N. Srinivas, A. Krause, S. M. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *International Conference on Machine Learning*, 2010. → pages 12, 15, 19, 46

[26] R. S. Sutton and A. G. Barto. Reinforcement learning: An introduction. *IEEE Transactions on Neural Networks*, 9(5):1054–1054, 1998. → pages 15