Date of acceptance Grade

Instructor

ImSe: Instant Interactive Image Retrieval System with Exploration/Exploitation trade-off

Ksenia Konyushkova

Helsinki May 17, 2013

UNIVERSITY OF HELSINKI
Department of Computer Science

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Laitos — Institution — Department
Faculty of Science	Department of Computer Science

Tekijä — Författare — Author
Ksenia Konyushkova

Työn nimi — Arbetets titel — Title

ImSe: Instant Interactive Image Retrieval Systemwith Exploration/Exploitation trade-off

Oppiaine — Läroämne — Subject
Computer Science

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
	May 17, 2013	59 pages + 0 appendices

Tiivistelmä — Referat — Abstract

Imagine a journalist looking for an illustration to his article about patriotism in a database of unannotated images. The idea of a suitable image is very vague and the best way to navigate through the database is to provide feedback to the images proposed by an Image Retrieval system in order to enable the system to learn what the ideal target image of the user is. Thus, at each search iteration a set of n images is displayed and the user must indicate how relevant they are to his/her target. When considering real-life problems we must also take into account the system's time-complexity and scalability to work with Big Data. To tackle this issue we utilize hierarchical Gaussian Process Bandits with visual Self-Organizing Map as a preprocessing technique. A prototype system called *ImSe* was developed and tested in experiments with real users in different types of tasks. The experiments show favorable results and indicate the benefits of proposed algorithms in different types of tasks.

ACM Computing Classification System (CCS):
Information systems - Information retrieval - Specialized information retrieval - Multimedia and multimodal retrieval - Image search
Theory of computation - Theory and algorithms for application domains - Machine learning theory - Reinforcement learning - Sequential decision making
Information systems - Information retrieval - Retrieval tasks and goals - Recommender systems

Avainsanat — Nyckelord — Keywords
Relevance Feedback, Exploration/Exploitation, Content-Based Image Retrieval

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — övriga uppgifter — Additional information

Acknowledgements

I would like to express my gratitude to everyone who supported me in my thesis work. I am grateful to my advisor Dr Dorota Głowacka for her invaluable guidance and constant supervision throughout the research. I would like to thank our group leader Professor Petri Myllymäki for his support and inspiration. A big thanks goes to the SciNet project group members, who helped me to acquire practical knowledge and skills in conducting the research. I also want to thank all the participants in the usability testing of the ImSe system.

Thank you to the University of Helsinki and Finland for the opportunity to pursue Master's degree. Thanks to my fellow classmates in the Algorithms and Machine Learning program and the international students' community in general for the warm atmosphere and motivation during classes and project work. I would like to thank Pirjo Moen for her advice during my studies and all the teachers at the department for their enthusiasm and involvement. I would like to extend my appreciation to my university in Russia, Nizhny Novgorod – Higher School of Economics and especially to the dean of the faculty of Business Informatics and Applied Mathematics, Valery Kalyagin, for the solid background that made my Master's studies possible.

Last, but not the least, I would like to extend my genuine gratitude to my family – to my parents and grandmother for moral support and encouragement, and to my brother for criticism and inspiration.

Contents

1 Introduction

We consider a problem of Image Retrieval from a database of images without any associated metadata such as keywords, tags or natural language text. In a situation when the user is unable to query the system because there are no labels available or they are not appropriate for the type of search, we employ the mechanism of Relevance Feedback to specify the user needs. For instance, imagine a journalist looking for an illustration to his or her article about patriotism in the database of unannotated photos. The idea of a suitable image may be very vague and the best way to navigate through the database is to provide feedback to the images proposed by the system. The user plays a key role in the process of learning as he or she must indicate relevance of n displayed images. We propose a system that operates through a sequence of rounds, where n images are displayed at each round and the user provides Relevance Feedback by indicating how close the displayed images are to his or her ideal target image.

With the growth of the Internet and the associated amount of available images, there has been a steadily increasing interest in new tools for managing and analyzing them [SWS+00]. New Image Retrieval techniques are required in many domains, such as medicine, photography, advertising or design. Traditional Image Retrieval tools (e.g. *Google Image Search*, *AltaVista*) rely heavily on the metadata associated with images, such as caption and keywords but with the rapid growth of the amount of digital images, keyword annotation becomes infeasible [ZH03]. To overcome this problem, some systems employ crowd sourcing for label collecting. For example, a game for collecting tags was designed and the collected labels were later used in Google Image Retrieval [goo11]. The game soon became popular and it was possible to collect a reliable database of labels due to the large number of participants.

In Content-Based Image Retrieval (CBIR) no image metadata is used but instead CBIR systems rely on visual features automatically extracted from images, such as color, shape or texture. The first CBIR experiments date back to 1992 [KKOH92], where color and shape features were used for automatic Image Retrieval. Since then, many CBIR Image Retrieval systems have been developed, such as *Superfish* [SF12] or *CIRES* [IqA02]. The main problem of applying such features to intelligent tasks in Computer Vision is the semantic gap between automatic feature extraction and human interpretation of these features. Image Retrieval systems can only provide information on feature similarity, but they lack semantic understanding and interpretation of images that guide humans in their image selection [SWS+00].

However, despite these problems, the process of Image Retrieval can be greatly improved by actively involving the user into the search loop and utilizing his knowledge in the iterative search process for better target modeling and personalized search (for example, *GNU Image-Finding Tool* [MSP03]). The most popular way of involving the user in the retrieval process is through Relevance Feedback [ZH03]. Relevance Feedback was first used in document retrieval but soon it was applied in multimedia search, including Image Retrieval. For example, *PicHunter* [CMM+00]) is a Bayesian approach to Image Retrieval with user interactions. The other Relevance Feedback Image Retrieval system *PicSOM* [LKLO00] utilizes Self-Organizing Maps in search for suitable images.

Relevance Feedback mechanisms are also used as input for Exploration/ Exploitation algorithms which attempt to predict "the most informative" and "the most positive" image at the same time [ZH03]. Multi-Armed Bandits problems are discussed in association with casino slot machines – bandits. In a stochastic decision making situation a gambler tries to learn what reward each of the bandit arms can give. In an Image Retrieval problem we consider showing an image to a user to be pulling an arm and the feedback we get back from the user is the reward. For example, *PinView* [AHK+10] is based on contextual Multi-Arm Bandit algorithm *LinRel* [Aue02]. In order to be applied in real-life systems the algorithms used in Image Retrieval systems must be time-efficient and easily scalable to large datasets. Most work on Exploration/Exploitation algorithms concentrates on analyzing their theoretical guaranties but often ignores their potential usability in real life applications. To allow the users to interact with a retrieval system we have to ensure that it is able to provide an immediate response to user's manipulations. The more elaborate an algorithm is, the more precise result it tends to provide, but at the same time the more computational power it tends to require. To overcome scalability limitations one can use primarily retrieved images [MV11] or precomputed clusters to choose the most representative samples in an Active Learning scenario [NS04]. Wu, 2001 [WM01] tackles, for example, the problem of high-dimensional computation in nearest neighborhood Relevance Feedback search.

In this work, we introduce *ImSe*, an exploratory Image Retrieval system based on Multi-Armed Bandits. We propose a novel approach based on a hierarchical Gaussian Process (GP) bandits combined with Self-Organizing Maps (SOM) to speed up the computation. The proposed algorithm is called *GP-SOM* and forms an integral part of the *ImSe* system. At every iteration, the algorithm balances between exploiting available knowledge to make the best current prediction and exploring other

possibilities to decrease uncertainty given only a limited feedback from the user. Classical Exploration/Exploitation trade-off algorithms are designed to predict one object per iteration. In hierarchical settings the question of selecting multiple objects per round becomes even harder and we tackle this problem by utilizing the Gaussian Process belief on the potential feedback as a temporal pseudo feedback.

This document is organized as follows. In the next section, we briefly describe issues associated with Image Retrieval. In Section 3, we give a theoretical overview of Reinforcement Learning with particular emphasis on trading off between exploration and exploitation. A detailed description of the proposed system and the *GP-SOM* algorithm in Section 4 is followed by its complexity analysis in Section 5. The experimental design and results of the system evaluation are presented in Section 6. Finally, we discuss the results and future plans in Section 7.

2 Image Retrieval

Interest in Image Retrieval was motivated by the growth of the Internet and the popularity of digital imaging. The amount of available images has increased dramatically within last years and so tools for managing them have become a growing field of research. Image Retrieval is an active field of research since it is used in many domains, such as medicine, photography, advertising, design etc.

With the increasing presence of digital imaging in modern life, the question of storing, monitoring, analyzing and searching images becomes vital. The database type defines what type of user and algorithm are the best suited for a given retrieval task. Smeulders, 1998 [SKG98] identifies *narrow* and *broad domain* image databases. *Narrow* databases, such as medical images of blood cells or frontal photos of faces in the same light and background, are characterized by predictive variability. On one hand, it is easier to define the similarity measure for this type of database, but on the other, the similarity measure cannot be applied to other domains. *Broad* domain databases include collections of different images; it could be photo and image collections of groups of people or images extracted from the Web. It is harder to work with this kind of data. It is impossible to restrict the variability of the database and in consequence, features and similarity measures are trickier to define. Feature extraction is the subfield of Computer Vision and there is still no perfect solution to this problem. Usually, the results based on *narrow* databases, where domain-specific knowledge is used, are higher than on *broad* databases.

Two main fields of Image Retrieval are *Context*-Based Image Retrieval (for example, Yee, 2003 [YSLH03]), where textual data and other metadata are used, and *Content*-Based Image Retrieval (for example, Rui, 1999 [RHC99]) that relies only on features extracted from the images. They emerged from two different research areas – Database Management and Computer Vision.

In Context-Based Retrieval, filenames, image captions, HTML titles and "ALT=" text, hyperlinks, keywords and other texts are used [FSA96]. Context-Based Retrieval is sometimes called metadata Image Retrieval. This approach was established in late 1970s (for instance, see [CF80]). The examples of Context-Based Image Retrieval systems include many industrial Image Retrieval systems such as *Google Image Search*, *AltaVista*, *Corbis*, *Live Search* from Microsoft and *Webseer* [FSA96]. Methods adapted from information retrieval can be used in metadata Image Retrieval. For instance, the ranking technique PageRank may be applied not only to

textual data but also to image ranking [JB08].

In our work we will concentrate on Content-Based Image Retrieval and in particular we will analyze the benefits of engaging the user in the search process.

2.1 Content-Based Image Retrieval

Unfortunately, in most most cases the information needed to adapt text retrieval techniques to Image Retrieval (as it is done in metadata Image Retrieval) is not available or not reliable. With the growing amount of digital images, keyword annotation becomes infeasible [ZH03]. When people assign labels to images, there is a lot of room for subjectivity and mistakes in the annotation procedure. Studies have shown that users performing the labeling task independently and without communication create and use very personalized hierarchies of concepts. To overcome these problems, systems of crowd sourced label collecting were designed. For example, in [goo11] a game for collecting tags was created. Two random people who enter the system are coupled together and the same image is shown to them. They neither have any information about their partner nor any communication tool. Their task is to give a label describing the image that will co-occur with the label of the partner. This task formulation results in a situation where people type many relevant tags. The tag that is the same for both players is usually a very precise descriptor of the shown image. This game soon became very popular and due to its large scale a reliable database of labels was collected for the first versions of *Google image search* in the early days of Image Retrieval.

However, even when image metadata and tags are available, some tasks require other types of knowledge. For example, suppose a person saw an image of an exotic animal and now he or she would like to find an image of this animal and learn its name. In this scenario, the user would guide an image search with queries like "long ears", "small fuzzy tail", "white fur", that are unlikely to be present in the labeled database.

Another problem occurs when the user's retrieval target is am abstract concept. For example, the task may be to illustrate "patriotism", "madness" or "boredom". Labels usually describe the visual content of images, but not their meaning. That is why these words are not common labels in the database, which means that the user has to define a concrete target in his mind and only then look for the image. For instance, the user may decide that a good illustration for "patriotism" is an image of

a battle from the Second World War in Stalingrad. However, there is no guarantee that such an image exists at all in the database.

Now, let us consider a search of an image of a "cat" in a huge database, where there is likely to be plenty of such images. The user may have a particular requirement what the cat should look like: for instance, it should sit in the garden, be white and red and it should be cleaning itself. Most likely, thousands of different cat images in the database will all have only the tag "cat" without the level of details needed in this task. This brings us to the question of how detailed the labels should be to answer such queries and how to get them without neglecting their quality.

In order to overcome all these difficulties in the labeled search, the researchers came with the concept of Content-Based Images Retrieval which utilizes features from images themselves [RHC99]. There is no laborious part in this process and no need to assign labels manually because the features are extracted automatically. There is no problem of detailed label assignments as all the information is present in the images and can be extracted differently for different purposes without human labeler involved.

Thus, Content-Based Image Retrieval (CBIR) stands for search of an image in a database without using metadata such as labels or associated text, but only by means of automatically derived visual features such as color, shape and texture. The first attempt to use CBIR was in 1992 [KKOH92], where color and shape features were used for automatic Image Retrieval. Since then, a continuous development of the field has led to many Image Retrieval systems of different nature.

2.1.1 Feature Extraction

Research in CBIR is a broad and expanding discipline nowadays and there is a great variety of systems based on CBIR. Many of them involve elaborate Computer Vision techniques for analyzing images and their structure in order to find similar images, for example *CIRES* [IqA02] or *SuperFish* [SF12]. *SuperFish* uses the classic idea of visual similarity but operates in the specific domain of commercial products. Image Retrieval systems, like *img(Rummager)*, may combine traditional text-based Informational Retrieval with feature extraction [CBL09].

Content-Based Image Retrieval consist of three parts: Feature Extraction, Multi-dimensional Indexing and Retrieval System Design [RHC99]. Feature extraction is of separate interest in itself and many methods have been developed in the field of

Computer Vision. We will not go into details of these techniques but just mention briefly those that we refer to later on in this work.

- *Color.* This is the most popular feature used in retrieval tasks. It is easy and fast to extract, easy to interpret and independent of the image size and orientation. Usually, when talking about color feature, we mean color histograms in RGB or HSV spaces.
- *Texture.* The next popular and simple choice for visual feature is texture. It describes the structure of surface and its relationship to the surrounding environment. When talking about texture, we usually mention such concepts as contrast, entropy and regularity. Among the extraction methods we should mention wavelet transform and Gabor wavelet transform.
- *Shape.* Another popular visual feature is shape, that is often required to be rotation, scaling and translation invariant. Shape feature may be 2D or 3D. The basic technique to extract it is through different kinds of Fourier transforms.

Other popular visual features in Computer Vision include color layout, segmentation, different kinds of edge detectors, corner detectors, points of interest detectors and many others.

The second aspect related to Image Retrieval is High Dimensional Indexing [GCP11]. It is essential because we have to work with huge dimensions and non-Euclidean distance measures in visual features. Very often we have to utilize either indexing or dimensionality reduction techniques to allow efficient browsing of the database.

Finally, to develop a full system, we need to design the retrieval part itself. In many systems, retrieval is done with search by example, search by sketch or navigating in image categories. In this work, we are mostly interested in random browsing systems. For example, this principle is used in *PicSOM* [LKLO00] and *PinView* [AHK+10] CBIR sysems.

2.1.2 Semantic Gap

Even though the field of feature extraction is well-developed and many researchers are actively working in it, there are still some unsolved problems. The main issue is applying extracted features to intelligent tasks in Computer Vision such as classification, clustering or search. What remains unsolved is the visual information

interpretation – the so-called Semantic Gap. It is often not clear at all how to judge the similarity of images just by the extracted features. It is clear that high-level concepts are not straightforwardly encoded in color, texture and shape. It is very easy for humans to recognize an objects on the screen even though they may vary a lot in position, rotation and lighting. On the contrary, the machine that only relies on pixels of images cannot easily deduct what object it is. Image Retrieval systems can provide only feature similarity information but they lack semantic understanding and image interpretation that guides humans in their behavior [SWS+00].

The Semantic Gap between image content and the context led to a huge disappointment in the early Computer Vision applications. Even though vision tasks are tackled by many researcher and a lot of progress has been made since the field was established, the accuracy of the predictions is still far from being perfect even in well understood tasks such as object classification [SWS+00].

Many problems in Computer Vision are not addressed, because it is impossible to solve them without image understanding. For example, search by association (open-ended) search is rarely tackled in classical Computer Vision applications. However, we still believe that even without complete image understanding the task of Images Retrieval can be tackled by including interactive components. This will be discussed in the next section on the Relevance Feedback mechanism.

2.2 Relevance Feedback

Initially in Image Retrieval tasks the user was considered to be just a "labeler" of the image database. When Content-Based retrieval was developed, its aim was to dispose of the user in the retrieval loop and make the process fully automated. But none of these approaches reached high performance and researchers in the field decided to bring the user back to the retrieval process in a more intelligent way.

Relevance Feedback was first developed in document retrieval but it was soon extended to multimedia retrieval. It is common knowledge that there is no problem of Semantic Gap for humans, that is so hard to overcome in traditional Content-Based Retrieval systems. Thus, we need to utilize user knowledge in the interactive search process for better personalized and targeted modeling of each retrieval session. In Relevance Feedback Content-Based Image Retrieval we bring the user back to the retrieval loop, where we are interested more in an interactive image understanding rather than completely automatic methods for retrieval.

There are many systems that require interactions and feedback from the user in different forms. For example, *Fire* system allows the user to define a part of an image to use in search instead of the whole canvas area [DKN04]. *PicsLikeThat* [CCP] relies on the image similarities as in any classical CBIR system, but learns image relationships from user's interactions.

However, the most popular way of utilizing the user in the retrieval process is through Relevance Feedback mechanism as it is done, for example, in *GNU Image-Finding Tool*. The user is presented with a set of images and he provides feedback on how relevant these images are to his search. The system then makes the next prediction which is consistent with the received feedback [MSP03]. The user is engaged in the loop of active learning, where he or she iteratively refines the original query by providing the feedback for the current system prediction. We should also mention two other Relevance Feedback Image Retrieval systems relying on the same principle of interactive query refining: *PinView* [AHK+10] and *PicSOM* [LKLO00]. These two systems are the most similar to the system presented in this work and so we will describe them in more detail later on.

2.3 Scalability Limitations

New challenges occur in the Image Retrieval field all the time, even though previous ones, such as Semantic Gap and the role of the user in the Image Retrieval, still remain unsolved [LSDJ06]. For example, nowadays sizes of actual databases are so big that researchers need to consider how to tackle the task in a scalable manner [GCP11]. The fashionable concept of "Big Data" is mentioned in Image Retrieval more and more often. This issue is closely related to the challenge of providing Image Retrieval services in an instant interactive manner in the modern digital world. The difference between training stage and actual performance stage diminishes and the problems in Image Retrieval have a strong on-line oriented character. There were attempts to make image search systems scalable as, for instance in SALSAS [GCP11], where the complexity is almost constant for any database size thanks to locality sensitivity hashing. Pre-clustering can also be incorporated to take data distribution into account during the search [NS04].

3 Reinforcement Learning and Exploration/Exploitation trade-off

This chapter introduces the concept of Reinforcement Learning (RL), what kind of problems it solves and what techniques it uses. We start with the definition and characteristics of RL problems in Section 3.1. Then, in Section 3.2 we explain in more details the Exploration/Exploitation trade-off problems and how a variety of Bandit algorithms help to tackle this problem. We start our description of Bandit policies with classical UCB like algorithms and then move into problem formulation with dependent arms and discuss algorithms like *LinRel*, *LinUCB* and *GB-UCB* Bandits. Finally, we conclude the section on Reinforcement Learning with a list of open challenges.

3.1 Reinforcement Learning

Reinforcement Learning (RL) is motivated by physiological approach. It appeared at the intersection of computer science and statistics on one side and psychology and neuroscience on the other side. Nowadays, it attracts a lot of attention from the Artificial Intelligence and Machine Learning communities. RL is mainly characterized by the nature of its problems rather than particular methods to solve them. The basic problem in Reinforcement Learning is decision making by an agent in situations of uncertainty in a stochastic environment with perception-action links between an agent and environment [SB98]. An agent is acting in a stochastic non-deterministic environment and learns the behavior policy that maps states to actions without specifying directly how to reach the target but by getting reward and punishment from the environment (which, however, may have delayed character).

The usual way to describe a RL problem is through action-perception loop [SB98]. An agent is acting in a stochastic environment and his aim is to increase a long-term reward obtained from the environment, which may often be delayed. The agent must discover what actions lead to a better reward in a set of trial-and-error interactions. In a general framework of Reinforcement Learning problems, an agent chooses at each iteration an action a based on indication of the current state s of the environment, which changes the environment state s that is communicated back to a decision maker through a reinforcement signal r – reward or punishment [SB98]. In the trial-and-error interaction, an agent learns an optimal behavior B

in the dynamic environment without explicit instructions of how the target may be achieved. Thus, a RL problem is characterized by a set of environment states s, a set of actions a that an agent can take in states and a reinforcement signal r.

Reinforcement Learning problems are often encountered in the fields of robotics, distributed control, multiagent learning, economics, telecommunication, decision support systems, resource management etc. ([BBDS08], [KLM96]). It differs a lot from Supervised Machine Learning problem formulation and adaptation of Supervised methods does not lead to good results. First of all, there are no training input/output pairs in RL and algorithm has to learn in an on-line manner. As a consequence, there are no training/testing stages in the learning process which also means that the performance is evaluated in an on-line fashion simultaneously with learning. Another important feature of RL is that an agent never knows the optimal action. After making a decision and choosing an action, an agent receives the reward, but it never knows how good the reward is compared to other possible actions and what the optimal strategy is in that situation and the long-term best action. PAC models in Supervised Learning may seem to be similar to RL but their aim is to learn the policy with high expected future predicted accuracy, while in RL we are interested in minimizing the total loss, even the part of it that is obtained while learning. The closest RL framework is established in Q-learning [WD92] that can be viewed as a transfer point between Supervised ML and Reinforcement ML. Reinforcement Learning is often compared with search and planning problems.

There are two main classes of algorithms used to solve Reinforcement Learning problems [KLM96].

- The fist type of strategy is to search for the best behavior in the environment in the space of possible behavior strategies. An example of an algorithm following this strategy is a genetic algorithm.
- The second approach uses statistics and dynamic programming in estimating the utility of different actions in the states of the environment

We often mention optimal behavior in the definition of RL. Optimal behavior may be defined differently and it determines the way the problem is solved. Some popular optimality criteria according to [KLM96] are listed below. Let us denote the reward obtained at the iteration i as r_i.

- *Finit time horizon* optimality is used when the lifetime for algorithm h is known and in this case it is clear that we aim to maximize the reward R obtained

within h iterations that is equal to:

$$E(\sum_{t=0}^{h} r_t). \tag{1}$$

- *Infinite horizon* optimality is applied when we do not know when the algorithm should stop and in this situation we take into consideration all future rewards but discount them over time with a factor $0 \leq \gamma < 1$. The total reward R is:

$$E(\sum_{t=0}^{\infty} \gamma^t r_t). \tag{2}$$

- Average reward may also be used for judging the optimality. It is very similar to the case of infinite horizon with γ approaching 1. Then the reward R is equal to:

$$E(\frac{1}{h}\sum_{t=0}^{\infty} r_t). \tag{3}$$

Having defined an optimality criteria for a policy, we now need to decide what criteria should be used for evaluating the learning performance. It may be one of the following:

- *Convergence*: asymptotic convergence to an optimal behavior. This evaluation of the learning performance is the closest to Supervised Machine Learning policy evaluation but it may not be the best in practice.

- *Speed of convergence*: how fast the proposed algorithm finds an optimal or sub-optimal strategy.

- *Level of performance*: how good the strategy found after a given time is.

- *Regret*: this is a very popular choice in RL problems, especially in Bandit algorithms. It penalizes the mistakes when they occur and aims to find the best policy with the least regret despite the fact that we did not know and could not follow the best policy from the beginning. Regret consists of the expected difference between total reward of the algorithm at time T and the best possible reward. Many Multi-Armed Bandit problems are analyzed in terms of regret bounds (see [ACBF02] for an illustration).

One of the central issues in Reinforcement Learning is trading off between Exploration and Exploitation strategies in the uncertain stochastic environment. A decision maker must choose between following the best current predicted strategy (exploit the available knowledge) and testing other alternatives (exploring to reduce the uncertainty about the environment) to discover a strategy that beat the current best choice. These problems are often addressed by Multi-Armed Bandit algorithms.

3.2 Multi-Armed Bandits

Multi-Armed Bandit problem formulation deals with the decision making process in the face of uncertainty [SB98]. It has been an area of research since 1950 and its definition goes back to 1979 [Git79]. The intuitive idea of Multi-Armed Bandit comes from the example of a gambler (agent) playing a row of slot machines (arms of a bandit) with some unknown prizes (rewards). At every iteration, the agent faces a decision-making problem which arm to play and receives some reward for it. The problem is stochastic but stationary and reward in arms may be modeled as a sample from a fixed probability distribution. The agent is aiming to maximize the reward but as he does not know how rewards are generated, he has to not only exploit the most promising arms but also explore different potential actions. The challenge in Multi-Armed problems is to balance between exploration and exploitation. As in any Reinforcement Learning problem, the evaluative feedback that we get from an agent tells us only how good the action is, but we have no idea if the selected arm is the best of the worse among other possibilities.

A close area of research to Exploration/Exploitation policies is Active Learning and the arm selection process may be considered as an Active Learning process. The aim of Active Learning algorithm may be, for example, to build a classifier as quick and precise as possible by choosing the most informative example from the input space to label [Mac92].

Examples of Multi-Armed Bandits problems include placing advertisements on the web, where placing an add corresponds to pulling an arm and user clicks are a reward. For instance, this is used by Yahoo for news articles recommendation [LCLS10]. Another example of Multi-Armed Bandits approach to a recommendation task is shown for Information Retrieval of scientific articles in [GRK+13]. In this work, the authors used Multi-Armed Bandit formulation for directing the users in the concept space and predicting his intent from keyword manipulations.

In this subsection, we present a number of different Exploration/Exploitation Bandit algorithms under independent arms assumption. We start with a Greedy algorithm and then show that even the simplest active exploration strategy ϵ-Greedy is more probable to converge to the optimal solution. Then, we present the idea of Upper Confidence Bound in Bandits algorithms. Finally, we explain why algorithms with distinct arms cannot be applied in many practical problems and we introduce another class of algorithms with dependency in arms such as *LinRel*, *LinUCB* and a non-parametric Gaussian Process Bandits.

3.2.1 Greedy Strategy

Let us denote the actual value of an action a as $r^*(a)$ and if we play an arm we keep its estimate as a sample average of obtained rewards – $\mu_t(a)$. The more each arm is played, the more precise the estimation of the true value becomes. The simplest strategy is a Greedy one, where after obtaining the first estimates we stick to the arm a with the highest reward $\mu_y(a)$ – this is so-called Greedy strategy [SB98]. If the environment was deterministic, the strategy to exploit the available knowledge would always be optimal, but as the environment is stochastic, the estimate might be inaccurate and it brings us to the question of balancing exploration and exploitation.

3.2.2 Epsilon Greedy

The main idea of non-greedy strategies is to improve estimates of true arm rewards by exploring different, sometimes not optimal actions to maximize the reward in the long run. It is not possible to explore and exploit by pulling arm in the same iteration and that is why the question of balancing between exploring and exploiting in Bandits algorithms is of a great importance.

A very simple but often a very good way to bring exploration into the algorithm is to introduce a random action into the behavior of an agent with probability ϵ. Such a strategy is called Epsilon Greedy and in practice it may be a strong benchmark and often hard to outperform [SB98]. The more each arm is selected, the closer its estimate is to the actual value and the probability of selecting an optimal arm converges to $1 - \epsilon$, while Greedy behavior tries to maximize the reward only in one iteration. The benefit of utilizing Epsilon Greedy strategy depends on the amount of variance in the system. The bigger the variance in the reward, the better Epsilon Greedy strategy is compared to a Greedy one.

3.2.3 UCB Policy

When we aim to get asymptotically optimal regret in a long term, we do not need to explore the whole space equally. For example, after a few runs many of the arms are believed to be far from the optimal solution and there is no need to explore them in a stationary environment, while some of the arms that are close to the current best solution deserve more attention. The Upper Confidence Bound (UCB) algorithms implement this idea [ACBF02]. They establish confidence intervals for all arms and continue playing an arm if and only if its upper bound gives some probability on belief that this might be the best arm.

The general idea is to maximize the Upper Confidence Bound of the arm selected at each iteration that is a combination of predicted mean and variance of an arm [ACBF02]. This is Optimism in the Face of Uncertainty (OFU) principle. For each arm, we keep reward estimate $\mu_t(a)$ at iteration t as a predicted mean of the relevance score and uncertainty measure $\sigma_i(a)$ as a standard deviation of the relevance. β_t is a parameter to adjust the confidence level and thus balance between exploration and exploitation. It is a positive monotonic factor and may be a constant or a function of time. The predicted mean, the uncertainty and the parameter β_t together define a confidence interval for each arm a at iteration i:

$$[\mu_t(a) - \sqrt(\beta_t) \cdot \sigma_t(a); \mu_t(a) + \sqrt(\beta_t) \cdot \sigma_t(a)]. \tag{4}$$

We believe that true reward for an arm is in the interval of the Equation 4. The arm is played if it has the highest upper bound in the confidence interval. This is how exploration is achieved – Upper Confidence Bound includes our uncertainty in the current estimate and it allows us to try promising arms. Thus, at each iteration i we choose the best arm to pull that maximizes the Upper Confidence Bound:

$$argmax\{\mu_t(a) + \sqrt(\beta_t) \cdot \sigma_t(a)\}. \tag{5}$$

Thus, UCB-like policies aim to estimate the function of reward accurately when its potential value is high but do no care about the accuracy of the estimation as soon as Upper Confidence Bound on the reward is low.

The difference in policies is in how the predicted mean and the variance are calculated and each realization of UCB principle specifies the way how $\mu_t(a)$, $\sigma_t(a)$ and β_t could be obtained. Classical UCB1 algorithm sets $\mu_i(a)$ to be the sample average of the arm and variance $\sigma_t(a)$ is calculated as:

$$\sigma_t(a) = \sqrt{\frac{2 \cdot \ln t}{n_t(a)}}, \tag{6}$$

where t is the iteration number and $n_t(a)$ is the number of times arm a was played so far.

A popular choice for β_t is an increasing function of time – $2 \cdot \ln(t)$. When β_t grows with time, it ensures that asymptotically even the worst arms would be sometimes played and it allows the algorithm to deal even with not completely stationary problems. Factor β_t may be adjusted to balance Exploration/Exploitation level. The theoretical guarantees on asymptotic convergence of UCB1 policy were proven in [ACBF02] and the regret bound is shown to be $\mathcal{O}(\sqrt{N})$.

3.2.4 UCB-tuned Algorithm

The algorithms based on the Upper Confidence Bound principle are gaining their popularity. For example, UCB-Normal deals with a problem where the rewards are known to be normally distributed and UCB-tuned is a modification of UCB algorithm that performs better than UCB in all the experiments, but a theoretical guarantee is not proven for it [ACBF02]. Many other strategies rely on the same principle, sometimes not in a direct way. In this section we will present an UCB-tuned version of the UCB policy.

We substitute $\sigma_t(a)$ from Equation 6 with:

$$\sqrt{\frac{\ln t}{n_t(a)} \cdot \min\{\frac{1}{4}; V(n_t(a))\}}, \tag{7}$$

where $V(n_t(a))$ is the upper confidence bound on the variance of an arm a:

$$V(n_t(a)) = (\frac{1}{n_t(a)} \cdot \sum_{t=1}^{n_t(a)} r_t^2(a)) - \hat{r}^2(a) + \sqrt{\frac{2 \cdot \ln t}{n_t(a)}}. \tag{8}$$

It was shown in the experiments in [ACBF02] that the UCB-tuned version of UCB-like policy performs quite well. However, in many real situations the number of bandit arms may be greater than the number of the experiments we can conduct, and as a consequence we cannot pull arms multiple times to get better estimates of the rewards. In this situation, the simple and efficient algorithms for exploring arms described above are not applicable any more and in order to tackle the problem we need to take into consideration the correlation between arms, which brings us to the stochastic Multi-Armed Bandit problems with dependent arms. The dependency structure may be captured, for example, by clustering [PCA07], linear regression (Section 3.2.5) or Gaussian Processes (Section 3.2.7).

3.2.5 LinRel Algorithm

The algorithm used in *PinView* system [AHK+10] is *LinRel* [Aue02]. It relies on linear regression model that captures functional relationship in correlation of arm inputs and reward. In this section, we consider an extension of a standard *LinRel* algorithm and its kernelized version that uses non-linear feature transformation by utilizing a kernel. Basically, the algorithm adopts the UCB main idea but sets its own way of calculating the belief on the reward and uncertainty with a linear regression estimation.

At each iteration, *LinRel* suggests arms to be pulled based on the feedback obtained in previous iterations. In each iteration, the *LinRel* algorithm obtains an estimate of weight vector $\hat{w}$ by solving a linear regression problem in Equation 9. Suppose we have matrix X, where each row x_a is a feature vector of arms pulled so far. Let $r = (r_1, r_2...r_h)^\top$ be the column vector of relevance scores received so far from the user, where h is the number of iterations. Thus, *LinRel* tries to estimate $\hat{w}$ by solving the linear regression problem:

$$r = X \cdot w. \tag{9}$$

Based on the estimated weight vector $\hat{w}$, *LinRel* calculates an estimated relevance score $\mu_t(a) = x_a \cdot \hat{w}$ for each arm a that has not already been presented to the user. As mentioned earlier, in order to deal with the Exploration/Exploitation trade-off, we choose arms to present not with the highest score but with the largest Upper Confidence Bound for the relevance score. If σ_t is an upper bound on standard deviation of relevance estimate $\mu_t(a)$, the upper confidence bound of arm a is calculated as:

$$\mu_t(a) + \beta_t \sigma_t, \tag{10}$$

where $\beta_t > 0$ is a constant used to adjust the confidence level of the upper confidence bound.

In the regularized version of the algorithm, we add the regularization factor λ to the linear regression and the relevance score m_t of arm x_a is calculated as:

$$m_a = x_a \cdot (X^\top \cdot X + \lambda I)^{-1} X^\top, \tag{11}$$

and the arm that maximizes

$$m_a \cdot r + \beta_t \|m_a\| \tag{12}$$

is selected for presentation. As we can see, *LinRel* relevance estimation of the reward is based on the linear regression with regularization to avoid overfitting. The variance σ_t in this case is defined as:

$$\sigma_t = diag(\|s_a\|). \tag{13}$$

To deal with non-linear transformation of the input space we utilize the kernel trick [STC04] and map our original feature space into another one described as dot products of the original features. Kernel trick allows us to work with non-linear projections that improve the quality of classification, while keeping the complexity limited.

3.2.6 LinUCB Algorithm

Another contextual bandit algorithm very close to *LinRel* was introduced in [LCLS10]. It was developed for personalized news article recommendation but the algorithm itself is a general enough to be applied in different settings. It extends *LinRel* idea and their performance is very similar but *LinUCB* was shown in the experiments on personalized web recommendations to be more effective when the data is sparse and the number of arms is huge. The algorithm is also motivated by the fact that is applicable to dynamic datasets, which is the case, for example, in news articles recommendation [LCLS10].

LinUCB is also based on ridge linear regression, that was used in Equation 11, but without regularization:

$$m_a = (X^\top \cdot X + I)^{-1} X^\top \cdot r_t, \tag{14}$$

and we define A_t as:

$$A_t = X^\top \cdot X + I. \tag{15}$$

We select an arm a to pull that maximizes the upper confidence bound:

$$argmax\{x_a^\top \cdot m_a + \alpha\sqrt{x_a^\top \cdot A_t^\top \cdot x_a}\}, \tag{16}$$

where the constant α is defined as:

$$1 + \sqrt{\frac{\ln \frac{2}{\delta}}{2}}, \tag{17}$$

for $\delta > 0$.

This algorithm has time complexity linear in the number of arms and cubic in the number of features d, but at the same time its regret bound is no different from other state of art algorithms, which is $\mathcal{O}(\sqrt{KdN})$.

3.2.7 Gaussian Process Upper Confidence Bound Algorithm

The last policy that we discuss in this chapter is used as a basis for the proposed algorithm and it is a non-parametric policy called Gaussian Process Bandits Upper Confidence Bound (GP-UCB) [SKKS09]. We start with introducing the Gaussian Processes framework in general, which originated in 1996 [WR96]. Traditional ways to solve regression or classification problems usually include parametric methods that are intuitive and easy to understand. When the problem is more complected, traditional parametric methods loose the advantage of their interpretability and with complex datasets they do not have enough expressive power [Ras06]. Another disadvantage of parametric methods is that they impose some structure of the input data, which means that we have to agree about the form of the model before fitting the data. It can be either linear, quadratic, polynomial, etc. but we have to specify it in advance. Gaussian Process gives the data freedom to speak for itself and this is less parametric tool.

Gaussian Process can be used to set a Bayesian framework for regression and can be seen as an infinite-dimensional multivariate Gaussian distribution. By definition, Gaussian Process is a set of random variables such that any finite subset of it follows multivariate joint Gaussian distribution. As one-dimensional Gaussian distribution (over vectors) is completely defined by its mean and variance, Gaussian Process (which defines a distribution over functions) is fully specified by its mean and covariance functions [RW06]. Often, mean μ_a is assumed to be zero and a popular choice for covariance function is "squared exponential":

$$k(x_a, x_{a'}) = \sigma_f^2 \exp[\frac{-(x_a - x_{a'})^2}{2l^2}], \tag{18}$$

where x_a and $x_{a'}$ are two arms, σ_f^2 is the highest value that the covariance can reach and l is a length parameter that is responsible for making distinct observations neglectable. If we want to take the noise into account, we define kernel as a combination of the covariance function defined in the Equation 18 and add Gaussian noise to it:

$$k(x_a, x_{a'}) = \sigma_f^2 \exp[\frac{-(x_a - x_{a'})^2}{2l^2}] + \sigma_n^2 \delta(x_a, x_{a'}), \tag{19}$$

where $\delta(x_a, x_{a'})$ is the Kronecker delta function and σ_n characterizes the amount of noise we add. With the mean and covariance functions we can define prior on the Gaussian Process. Suppose x is the training set with known reward values and x^* is the test set values with mean values μ and μ^* corresponding to them. Σ is the covariance for a training dataset, Σ_* is the covariance between training x and test data x_* and Σ_{**} is the covariance of the test data. Then, a prior distribution on the joint distribution can be written as [Ras06]:

$$\begin{bmatrix} r \\ r_* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu \\ \mu_* \end{bmatrix}, \begin{bmatrix} \Sigma & \Sigma_* \\ \Sigma_*^\top & \Sigma_{**} \end{bmatrix} \right). \tag{20}$$

The posterior condition probability for $p(f_*|f)$ can be derived to be:

$$p(f_*|f) \sim \mathcal{N}(\Sigma_* \Sigma^{-1} f, \Sigma_{**} - \Sigma_* \Sigma^{-1} \Sigma_*^T). \tag{21}$$

We can define Gaussian Process Bandits in the same way as UCB policies. Assume that K is kernel representation of arms from X (and in particular K, K_* and K_{**} are parts of the kernel matrix, where K correspond to pairwise kernel function between all shown datapoints, K_* – between shown datapoints and those we have to predict, and K_{**} – between datapoints to predict), r is a vector of Relevance Feedback we received so far. We have a probabilistic measures of posterior mean and uncertainty and then as usual in UCB-like policies we pull the arm with the highest score:

$$K_* K^{-1} r + K_{**} - K_* K^{-1} K_*^T. \tag{22}$$

3.2.8 Open Challenges

One of the important characteristics and a challenge in all the Bandits algorithms is their sensitivity to the amount of computation in the on-line step of the decision making procedure. We have to limit the time an agent might spend between actual interactions with the environment to keep it to be interactive.

Developing Bandit algorithms on trees is an active field of research nowadays. For instance, Upper Confidence Tree (UCT) algorithm based on standard UCB technique created a lot of interest thanks to its success in building intelligent agents for playing Go game [KS06]. UCT algorithms are often considered in combination with Monte Carlo Tree Search.

A lot of effort is spent on building policies for non-stationary problems, where the distribution of the rewards are changing with time, but still this problem formulation remains not completely solved.

All the Multi-Armed Bandit policies are designed to pull one arm of a bandit at a time. However, some of the applications require multiple selection per iteration ([GRK+13], [LCLS10]). In most of the cases, the problem is solved by sorting UCB and selecting top n arms to pull. The problem is even more complicated when we deal with hierarchical settings. Moreover, new challenges occur in this scenario. For instance, we might want to ensure the diversity in the arm selection and this problem is almost ignored in most Bandit settings.

4 ImSe System Design

In this section, we introduce Relevance Feedback Content-Based Image Retrieval system *ImSe* based on a time efficient Exploration/Exploitation strategy *GP-SOM*. First, we explain why there is a need for a system of this type and what problems related Content-Based Image Retrieval and Multi-Armed Bandit algorithms the system attempts to address. Next, in Section 4.2, we provide an overview of the data flow and computational methods used in *ImSe*. Sections 4.3 and 4.4 explain the data preprocessing stage and the online stages of the computation. The preprocessing techniques include feature extraction, Self-Organizing Map and kernel computation. The online step is based on a time-efficient algorithm *GP-SOM*, which also demonstrates an effective way of ensuring diversity in the arm selection process.

4.1 Motivation

In order to be applied in real-life systems, the algorithms for Image Retrieval need to be time-efficient and easily scalable to large datasets. Many researchers in Exploration/Exploitation algorithms analyze theoretical guaranties, but often ignore their potential usability in real life applications. To keep users interacting with a retrieval system we have to ensure that it is able to provide an immediate result in response to user's manipulations. The more elaborate an algorithm is, the more precise result it can usually provide, but at the same time the more computational power it requires.

In this section, we describe the design of the *ImSe* system. We introduce a time-efficient reinforcement learning algorithm applied to Content-Based Image Retrieval. We employ hierarchical Gaussian Process (GP) bandits [PACJ07] to balance between exploration and exploitation, where Self-Organizing Maps (SOM) [Koh01] of image features discretize the input space and are used as layers in the bandit hierarchy. We call our algorithm *GP-SOM*. By employing Self-Organizing Map as a discretization of a search space we allow state-of-art computational techniques to be applies to huge spaces. Like all Exploration/Exploitation algorithms, *GP-SOM* balances between presenting images that are most similar to the current user's selection and collecting a diverse feedback from the user to make more confident predictions in future iterations.

4.2 Algorithm Overview

The system operates through a sequence of rounds, when a set of n images is displayed at each iteration and the user must indicate which images are the closest to the ideal target image and which are the most different from the ideal target. The user provides feedback ranging from -1 to 1 (with 0 by default), where -1 indicates that a given image is very different from the ideal target and 1 indicates that a given image is very similar to the ideal target, i.e. it shares a lot of features with the ideal target.

The predictions are done by Gaussian Process techniques. We employ an idea of hierarchical Multi-Armed Bandits [PACJ07] to balance between exploration and exploitation to sample images and SOM serves as levels in the hierarchy. The main flow of the algorithm is presented in Algorithm 1 and a graphical presentation of the data flow in the system is illustrated in Figure 1. A detailed description of each of the steps is presented in the next sections. Several techniques are used in the proposed *GP-SOM* algorithm. First, we extract features $H = h_1, h_2, ..., h_N$ from a database of N images $P = p_1, p_2, ..., p_N$ and compute the distances between them for digital processing of visual information. Distances are presented in a matrix D_P of dimensions $N \times N$. Feature extraction and distances calculation are described in details in Section 4.3.1. Instead of straight optimization in image space, we utilize hierarchical Multi-Armed Bandits. In Section 4.3.2, we build a SOM $M = \{V, A_V\}$ based on the extracted features, which results in an image hierarchy with m model vectors $V = \{v_1, ..., v_m\}$. A_V shows image assignments to model vectors and $A_V = \{a_1, ..., a_m\}$, where $a_i = \{p_j : p_j \in cluster(v_i)\}$. Next, we calculate a kernel $K_P \subset \mathbb{R}^{N \times N}$ for images and model vectors (Section 4.3.3). The preprocessing techniques are crucial for the system to the able to perform retrieval in on-line manner. In Section 4.4, we describe how the pre-build SOM is used as levels of hierarchy in GP bandits. In order to ensure diversity of the images presented at each iteration we apply pseudo Relevance Feedback when sampling the images.

Algorithm 1 *GP-SOM* overview

Require: database of image $P = p_1, ..., p_N$

calculate features $H \leftarrow ExtractFeatures(P)$

calculate distances $D_P \leftarrow CalculateDistances(H)$

pre-compute Self-Organizing Map $M \leftarrow calculateSOM(9H)$

calculate kernel $K_P \leftarrow CalculateKernel(D_P, M)$

repeat

predict n images $p^1, p^2, ..., p^n \leftarrow HierarchicalMAB(H, K_P, M)$

User provides feedback $F = \{f^1, f^2, ..., f^n\}$, where $-1 \leq f^i \leq 1$

until user terminates the search

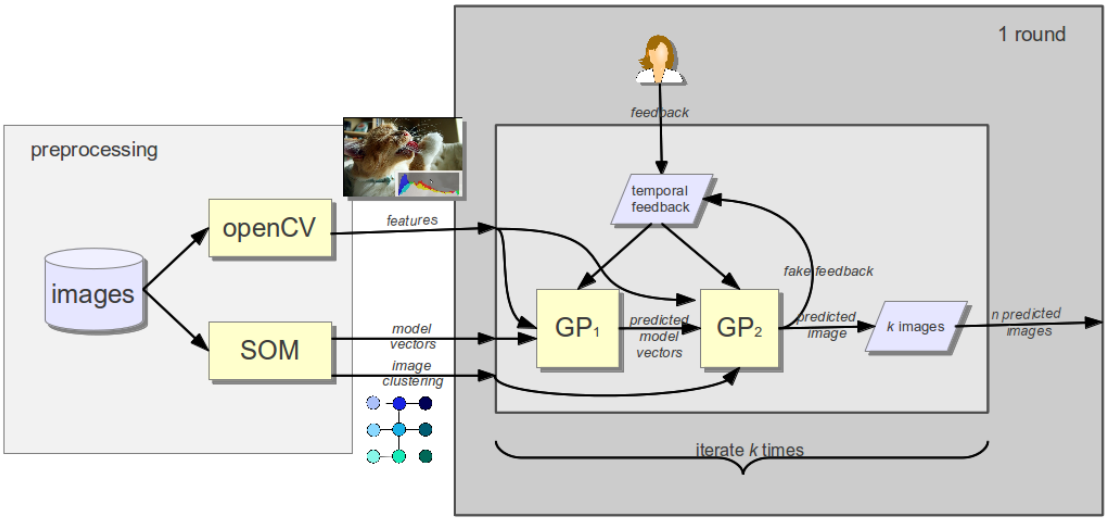

Figure 1: Dataflow overview of *ImSe* system

4.3 Preprocessing

In order to speed up the time it takes to perform one iteration using our interactive retrieval system, we first pre-compute the features of all the images in a given database. We also pre-compute the distances between the images, the SOM discretization and the kernel for the GP bandits. This step is done off-line. The aim of the pre-processing step is to keep the time needed for the on-line calculations almost independent of the linear increase in the size of the database.

4.3.1 Feature Extraction

As a first step, we extract features from the images. In our experiments reported below, we used the color feature as it is easy to understand and interpret for humans as well as fast and easy to extract and process. Color has a high discriminating power compared to texture or shape and many studies have shown that the color feature dominates users' decisions when comparing images [HLP+10]. Plain histogram representation with 64 bins will saturate the feature comparison for datasets with more than 25000 images that is why a joint histogram was chosen. This kind of comparison reaches its discriminative limit with size of dataset more than 250 000 and this is enough for this stage of development of *ImSe* system ([PZ99], [SS94]). OpenCV [Ope12] library from Intel with bindings to Python was chosen as a tool for extracting color information. OpenCV stands for Open Source Computer Vision Library and it's main focus is on real-time image processing. It is written is C++ but has an extensive interface to Python. Having developed OpenCV together with Integrated Performance Primitives Intel made these routines very fast and effective.

The color processing works in the following way. First we extract 3 color planes from images – red (R), green (G) and blue (B) planes. Then we create a 3 dimensional histogram with 64 bins in each direction and color depth from 0 to 255 for the extracted feature planes and normalize it to sum all bins to 1. Color histogram represents colors distribution in an image and the way it is contacted is no different from any kind of histogram, where pixels are random vectors and their values are amount of colors in each color space. There is no information about the way OpenCV processes the images apart from plain code and user documentation, but we can guess the extracting information about RGB values and counting the amount of pixels with the same values is straightforward.

The next question that we face is feature interpretation. In particular we have to decide how to compare color histograms with each other. We need to define a distance metrics for image color histograms. Our experiments showed that classic distance measures like pure Euclidean distance is not a good intuitive measure of human similarity perception. For example, this measure doesn't work well when histograms are sparse. OpenCV library has a few options for histogram comparison. It may be correlation similarity measure, Chi-Square, Intersection or Hellinger distance. A more suitable distance interpretation is probability density distance metrics [Hel09]. It was decided to use Hellinger distance from openCV library for this purpose because as a probability density metrics it has a nice probabilistic interpretation and

it performed well in intuitive understanding. Hellinger distance is a measure of similarity of probability distributions that is closely related to Bhattacharyya distance. If H^i and H^j are histograms of images i and j, N_b is a number of histogram bins and H_k^i is the value of kth bin in the histogram of image i, then Hellinger coefficient $d(H^i, H^j)$ for discrete distribution in bins can be calculated as:

$$d(H^i, H^j) = \sqrt{1 - \frac{1}{\sqrt{H^i \cdot H^j \cdot (N_b)^2}} \cdot \sum_{k=1}^{N_b} \sqrt{H_k^i \cdot H_k^j}}. \tag{23}$$

4.3.2 Self-Organizing Map (SOM)

Next in the preprocessing stage we create a discretization of the input space topology through a Self-Organizing Map (SOM) [Koh01]. It represents a non-linear projection of high dimensional space into a lower dimension space. SOM is an unsupervised method for dimensionality reduction by constructing an artificial neural network of instances that reflects their topological order. SOM provides the so-called model vectors that are treated in our algorithm as clustering of the input space.

Non-linear projection methods are not widely-used as a preprocessing technique due to their complexity and specificity of applicability. We choose using them because they seem to be a very natural and intuitive way of organizing images in terms of color features. One of popular ways to obtain SOM is through Expectation Maximization algorithm.

In our case, the number of clusters m is determined to be $\sqrt{N}$, where N is the number of images in the database. This choice is done in order to minimize computations that will be explained in Section 5. First, we create models vectors (which correspond to centroids in traditional clustering) and calculate the initial distances between them based on a 2-dimensional grid (see Figure 2).

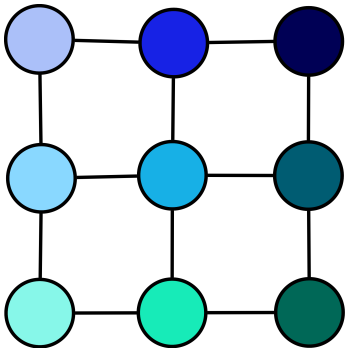

Figure 2: An example of a grid for a Self-Organizing Map

SOM may be calculated using the EM algorithm. At the beginning model vectors are assigned randomly. To avoid empty initializations, models vectors are chosen to be random images from a set of images in the database. The main difference between SOM and traditional Expectation-Maximization techniques (as K-means) is in the Maximization step. In the SOM Maximization step, when recalculating a model vector, the images assigned to other model vectors are also considered and their influence depends on the neighborhood function between model vectors. The neighborhood function in our system is the Gaussian kernel:

$$D_t = \exp(\frac{-D_0}{2 \cdot (\sigma_0 \cdot \exp(\frac{-t}{\lambda_t}))^2}), \tag{24}$$

where D_0 represents the initial distances in model vector grids, t is time, σ_0 is the degree of dependency between model vectors at the beginning (the bigger σ_0, the more dependent they are) and λ_t is the rate of dependency change with time (the bigger λ_t, the slower the dependence decreases). In the *ImSe* system the values of these parameters were set to $\sigma_0 = 5$ and $\lambda_t = 4$.

Consider an image i that is currently assigned to a model vector m_i^* with distance d_i^* and model vector m^k for which we want to calculate influence of image i. The neighborhood function D_t determines distance from model vector m_i^* to model vector m^k as d_{*k}. Then the weight of image i for cluster centroid m^k can be determined by:

$$d_i^* \cdot d_{*k}. \tag{25}$$

The Expectation step is similar to the classic K-means: datapoints are assigned to the closest model vectors. The preprocessing step results in an objects hierarchy which serves as an input to the hierarchical GP bandits algorithm. The Expectation step is similar to the classic K-means: datapoints are assigned to the closest model vectors. The procedure described above is presented in Algorithm 2.

Algorithm 2 BuildSOM

Require: histogram features $H = \{h_1, h_2, ..., h_N\}$, where $h_i \subset \mathbb{R}^{3 \cdot 64 \times 1}$ and $0 \leq h_i \leq 255$;

number of clusters m;

constants σ_0 and λ_t.

iteration $t \leftarrow 0$

model vectors $V = \{v_1, ..., v_m\}, v_i \subset \mathbb{R}^{3 \cdot 64 \times 1}$ are initialized randomly

locate model vectors V in a grid in the order of matrix G:

$$G \leftarrow \begin{vmatrix} v_1 & ... & v_{\sqrt{m}} \\ ... & ... & ... \\ v_{m-\sqrt{m}} & ... & v_m \end{vmatrix}$$

initialize distances D_0 in grid G: $D_0 \subset \mathbb{R}^{m \times m} : d_{ij} \leftarrow (mod_{\sqrt{m}}(i) - mod_{\sqrt{m}}(j))^2 + (div_{\sqrt{m}}(i) - div_{\sqrt{m}}(j))^2$

repeat

Calculate distances D_t at iteration t: $D_t \leftarrow \exp(\frac{-D_{t-1}}{2 \cdot (\sigma_0 \cdot \exp(\frac{-t}{\lambda_t}))^2)})$

Calculate distance D_{VH} between histograms of datapoints H and model vectors V: $D_{VH} \leftarrow$ EuclideanDistance(V,H)

for image $p_i \in P$ **do**

find the closest model vector v_{pi}: $v_{pi} \leftarrow \min_v\{D_V H(p_i)\}$

end for

for model vector $v_i \in V$ **do**

collect datapoints p_j into assignments a_i in SOM: $a_i \leftarrow \{p_j : v_{pj} = v_i\}$

recalculate model vectors v_i as means of histograms H of datapoints from assignments a_i: $v_i \leftarrow mean(H[a_i])$

end for

increase time: $t \leftarrow t + 1$

until converged

return model vectors V and image assignments to them A_V: $M = \{V, A_V\}$

The preprocessing step results in an object hierarchy which serves as an input to the hierarchical GP bandits algorithm. An example SOM for 1000 images and 36 model vectors is presented in Figure 3. For the illustration purpose, instead of model vectors we show images that are the closest to them. We can clearly see that images close to each other on the 2-dimensional grid share common features.

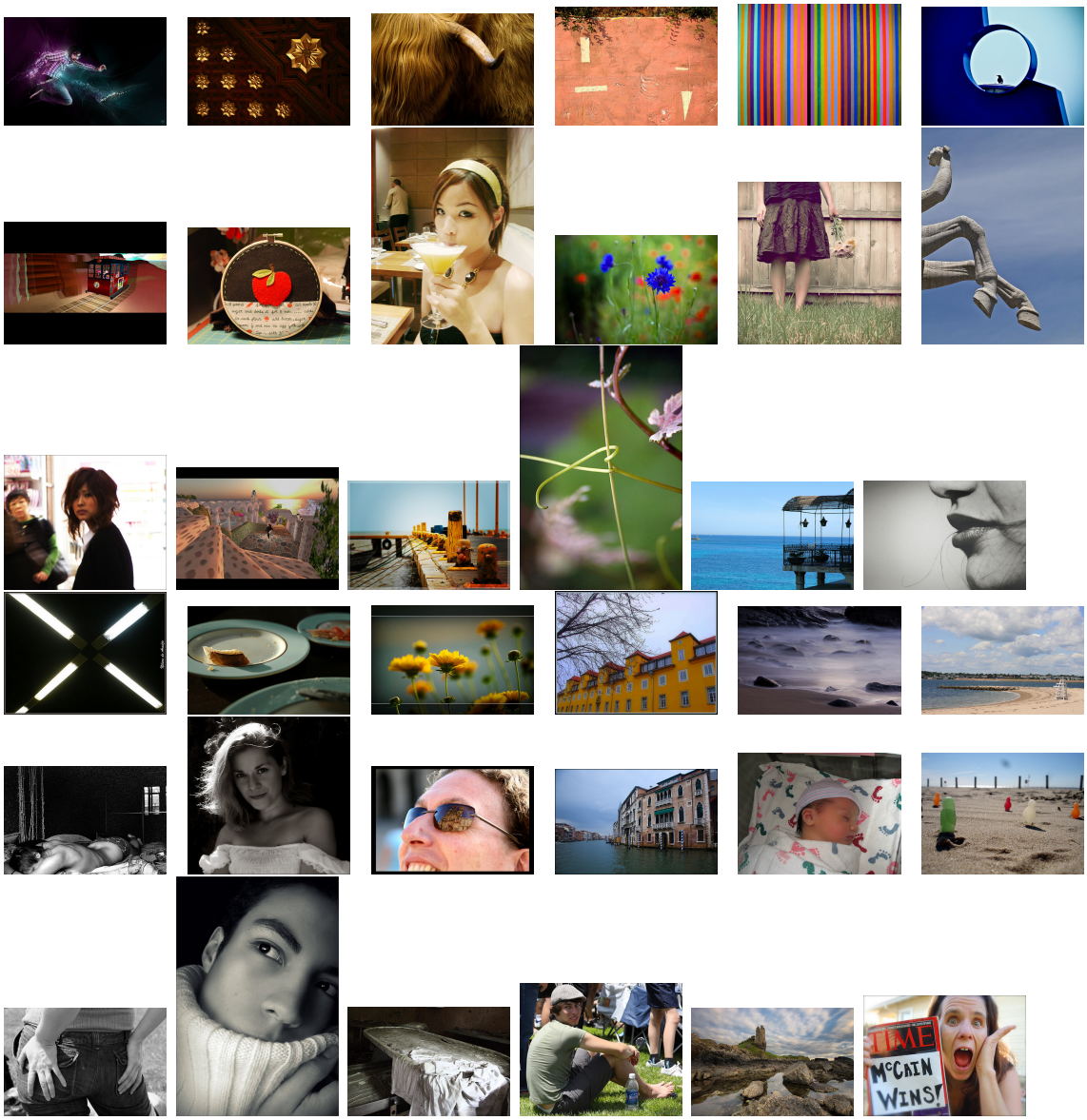

Figure 3: An example of a Self-Organizing Map for an image dataset.

4.3.3 Kernel

Hellinger distances from a random image to all the other images are shown in the Figure 4. We can see that there are almost no images very close to the particular image (no points withing distance less than 0.5) and at the same time there are not very many images that are completely different from it (distance more than 0.9). Most of the images in the dataset are located withing distances from 0.7 to 0.9. This can give us a feeling of how the data distribution looks like.

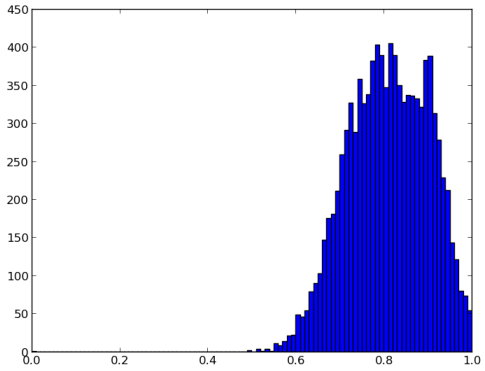

Figure 4: Distances from a random image to all others

We decided to use a well known kernel trick to make non-linear projection of our data [STC04]. As the next preprocessing step, we calculate a single kernel for both images and model vectors. The kernel is based on the Hellinger coefficient presented in Equation 26:

$$d(H^i, H^j) = \sqrt{1 - \frac{1}{\sqrt{H^i \cdot H^j \cdot (N_b)^2}} \cdot \sum_{k=1}^{N_b} \sqrt{H_k^i \cdot H_k^j}}, \tag{26}$$

for which we calculate a Gaussian kernel as in Equation 27:

$$D_t = \exp\left(\frac{-D_g}{2 \cdot (\sigma_0 \cdot \exp(\frac{-t}{\lambda_t}))^2}\right). \tag{27}$$

We set the constant values at $\sigma_0 = 0.5$ and $\lambda_t = 0.8$.

The final kernel matrix includes distances from images to images, from cluster centers to images and from clusters to clusters. This kernel matrix together with cluster assignments from SOM is passed on to the on-line step of *GP-SOM* algorithm as a prior covariance function in GP.

4.4 On-line step in *GP-SOM*

The basic idea behind the *GP-SOM* algorithm is to apply Bandit problem formulation to the image search. Multi-Armed Bandits are used here in order to predict not only the most relevant images according to the current selection of Gaussian Process, but also those images, that would help to decrease uncertainty in the feature space. An intuitive idea is that arms of a Bandit are images, rewards – user feedback. At

every iteration, *GP-SOM* faces a decision-making problem – which image to show to the user and receives some reward for it – user feedback. The challenge here is to balance between exploration to find images that beat our current best prediction and exploitation of the feedback in the system.

Instead of straightforward optimization in the image space, we utilize hierarchical Multi-Armed Bandits [PACJ07]. Model vectors that are produced as a result of clustering become the first level of instances for hierarchical Multi-Armed Bandits and images associated with the model vectors become the second level. We apply a 2-layer bandit settings: first we select a model vector and then an image from a pool of images associated with a chosen model vector.

Thus, in the first layer, the arms are considered to be model vectors and we select one model vector by *GP-UCB* algorithm. In the next step, arms are images associated with the chosen model vector and we select one image. There might be two causes for an image to be selected – it might have high predicted user interest or we might want to get feedback on it to reduce our uncertainty about it. When we need to obtain multiple images in a single iteration, we want to ensure that they are not only different but also diverse. For example, if we get feedback on a particular location l in the image space, it will be propagated on the neighboring datapoints by GP and there is no need to show multiple images from location l. In other words, an optimal selection of images should be spread out in the space and thus an algorithm should try to reduce a gap in 'unexplored' parts of the map.

Exploration/Exploitation algorithms aim to achieve this diversity, however, they are designed to select one image per iteration, then receive feedback on it and predict the next image. In our system, we present multiple images at each iteration and thus have to design a procedure that will ensure optimal selection of n images which cannot be assumed to be independent. Thus, when sampling the ith image among the n images, we need to take into consideration the influence of $i-1$th image that has been selected for presentation before receiving any Relevance Feedback from the user. Thus, when sampling the ith image, we assume that we received pseudo feedback for the previously sampled images $1, 2, ..., i-1$. As we do not know the user's real opinion on the selected images, we take the GP belief on $1, 2, ..., i-1$th images as our pseudo feedback. Again, we use hierarchical *GP-UCB* procedure to sample the next image, taking into consideration user's feedback from previous iterations and pseudo feedback from the GP in the current iteration. We repeat the selection procedure n times in order to obtain n images to present to the user. The

procedure is illustrated in the Figure 4.4. The sampling procedure is described in Algorithm 3.

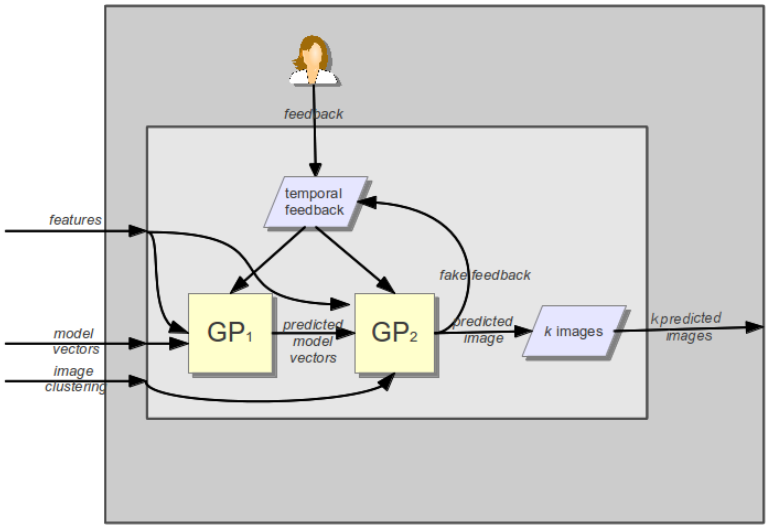

Figure 5: Dataflow in the on-line step of *GP-SOM*

Algorithm 3 HierarchicalMAB

Require: SOM $M = \{V', A_V\}$, where $V' = \{v_1', ..., v_m'\}$ is a set of kernelized model vectors, $v_i' \subset \mathbb{R}^{N \times 1}$ and $A_V = \{a_1, ..., a_m\}$ is a set of image assignments to model vectors, $a_i = \{p_j : p_j \in cluster(v_i)\}$, where p_j is an image;

set of shown images p_{t-1}^i up to iteration $t-1$: $S = \{p_{t-1}^1, ..., p_{t-1}^n\}$;

feedback F on S: $F = \{f_{t-1}^1, ..., f_{t-1}^n\}$, where $-1 \leq f_{t-1}^i \leq 1$.

set of selected images: $R \leftarrow \{\}$

repeat

Arms are model vectors $V' = \{v_1', ..., v_m'\}$, choose an arm v^*: $v^* \leftarrow \max\{\text{GP-UCB}(V', S, F)\}$

Arms are images associate with v^* - $A_{V^*} = \{p_j : p_j \in cluster(v^*)\}$, choose an image p^* and get its predicted relevance p_f^*: $p_f^*, p^* \leftarrow \max\{\text{GP-UCB}(A_{V^*}, S, F)\}$

Update the set of shown images S with the selected image p^*: $S \leftarrow S \cup p^*$

Update the feedback F with pseudo feedback p_f^*: $F \leftarrow F \cup p_f^*$

Add the selected image p^* to the result set R: $R \leftarrow R \cup p^*$

until n images are sampled: $|R| = n$

return set of n selected images R

---

5 Complexity analysis

We benchmark our system against two Exploration/Exploitation algorithms already presented in the section about contextual Multi-Armed Bandits: kernelized version of *LinRel* [Aue02], which forms an integral part of the *PinView* system, and the Gaussian Processes Bandit algorithm [SKKS09]. At each iteration, they suggest new images to be presented based on the feedback from the user obtained in previous iterations. The system exploratory selects not only current best predicted images, but also images for which the user feedback improves the accuracy of the estimate by decreasing the uncertainty in the feature space enabling better image selections in subsequent iterations. We selected these two algorithms as they are the closest in terms of design to *GP-SOM*.

When designing our algorithm, we consider not only theoretical guarantees of the Exploration/Exploitation methods but also their applicability to real-time systems. Usually theoretical analysis of bandit-style algorithms concentrate on regret bounds, but the time required for computing every prediction is neglected. The application we are considering is very practical, user-oriented and interactive and usability plays an important role. We can even sacrifice a bit of quality if we can obtain the result much faster. Below, we compare the computational complexity of *LinRel*, *GP-UCB* bandits and *GP-SOM* and find out how the complexity depends on the size of database and number of images presented at each step.

Let N denote the number of images in the dataset. At each iteration, we present n images, where $n \ll N$. We assume that the search can only last as long as all images have been displayed, so that $i \cdot n < N$. In most situations we are interested in the situation when $i \cdot n \ll N$, otherwise the user can just browse the whole database.

Let us denote the constant in multiplication complexity as c_m, summation as c_s and inversion as c_i. We will consider the basic linear algebra operations, where matrix multiplication of $[m \times k]$ by $[k \times p]$ takes time $c_m \cdot \mathcal{O}(mkp)$, summation of two matrices of the size $[m \times n]$ takes time $c_s \cdot \mathcal{O}(mk)$ and inversion of an $[k \times k]$ matrix requires $c_i \cdot \mathcal{O}(k^3)$ steps.

5.1 LinRel

Let us consider the $i+1$th iteration. In each iteration i, *LinRel* calculates estimated relevance score μ_i of each image:

$$\mu_i = s_i \cdot f_i, \tag{28}$$

where s_i is defined as:

$$s_i = X_p \cdot (X_{s_i}^\top \cdot X_{s_i} + \lambda I)^{-1} X_{s_i}^\top, \tag{29}$$

and the variance is:

$$\sigma_i = diag(\|s_i\|). \tag{30}$$

First, we calculate the complexity of one iteration of Kernelized *LinRel*. Remember that the X_p matrix dimensionality is $[N \times N]$, X_{s_i} is $[i \cdot n \times N]$ and f_i is a vector of $[1 \times i \cdot n]$. Thus, the computation of s_i from Equation 29 requires:

$$\begin{aligned} &c_m \cdot \mathcal{O}(N^2 in) + c_s \cdot \mathcal{O}(N^2) + c_i \cdot \mathcal{O}(N^3) + c_m \cdot \mathcal{O}(N^3) + c_m \cdot \mathcal{O}(N^2 in) \\ &\quad = c_m \cdot \mathcal{O}(N^3) + c_s \cdot \mathcal{O}(N^2) + c_i \cdot \mathcal{O}(N^3) \\ &\quad = \mathcal{O}(N^3). \end{aligned} \tag{31}$$

Computing μ from Equation 28 takes just $c_m \cdot \mathcal{O}(N^2) = \mathcal{O}(N^2)$ and σ from Equation 30 takes $c_m \cdot \mathcal{O}(N^2)$ as we need only values from the diagonal. To sum up, the complexity of each step of *LinRel* does not depend on the number of iterations, but on the number of images in the dataset as $\mathcal{O}(N^3)$.

5.2 GP Bandits

At each iteration i, we present to the user the image that maximizes $argmax\{\mu_i + \sqrt{\beta} \cdot \sigma_i\}$, where μ_i is a predicted mean of the relevance score, σ_i is a standard deviation. In GP-UCB, we define μ as:

$$\mu = K_* K^{-1} r, \tag{32}$$

and variance as:

$$K_{**} - K_* K^{-1} K_*^T. \tag{33}$$

When analyzing the complexity of GP, we should notice that $K_* K^{-1}$ is used in both Equations 32 and 33 and so can be precomputed as v. K, K_* and K_{**} have

dimensionality $[in \times in]$, $[N \times in]$ and $[N \times N]$ respectively. Computing v takes $c_m \cdot \mathcal{O}(Ni^2n^2) + c_i \cdot \mathcal{O}(i^3n^3)$, μ from Equation 32 takes $c_m \cdot \mathcal{O}(Nin)$ and variance from formula 33 takes $c_m \cdot \mathcal{O}(Nin) + c_s \cdot \mathcal{O}(N)$ (because we only need values from the diagonal). The total complexity of one iteration of GP bandits takes only $\mathcal{O}(Ni^2n^2)$ compared to $\mathcal{O}(N^3)$ of *LinRel*. At the same time, we avoid inverting a huge $[N \times N]$ matrix and do it only with a smaller $[in \times in]$ matrix.

5.3 GP-SOM

When building the Self-Organizing Map, we choose the number of points it contains such that the complexity of the algorithm at each level of hierarchy is approximately the same, which means that the number of model vectors in the map is chosen to be approximately $\sqrt{N}$. We have already shown that the complexity of GP bandits is $\mathcal{O}(Ni^2n^2)$. In *GP-SOM*, we perform the same calculations but on 2 levels with $\sqrt{N}$ objects in the set and n times (as after selecting each of the images we provide a pseudo feedback). Thus, the complexity of *GP-SOM* is $2 \cdot n \cdot \mathcal{O}(\sqrt{N}i^2n^2)$ and $2 \cdot n$ is much smaller than $\sqrt{N}$ in any realistic situation. Moreover, Self-Organizing Map approach can be generalized into an l-level hierarchical bandit by introducing additional levels in the map and increasing the complexity only by a scalar factor. If we fix the number of arms in each run to be P and allow an l level hierarchy, we can process P^l images with the complexity $l \cdot \mathcal{O}(Pi^2n^3)$.

Table 1 summarizes the complexity of the three algorithms [KG13]. We can easily see the benefit of using hierarchical GP bandits in Exploration/Exploitation trade-off. Additionally, it was shown that the SOM approach can be generalized into l level hierarchical bandits by introducing additional levels in the map and increasing the complexity only by a scalar factor.

LinRel	$\mathcal{O}(N^3)$
GP	$\mathcal{O}(Ni^2n^2)$
GP-SOM	$\mathcal{O}(\sqrt{N}i^2n^3)$

Table 1: Time complexity of the on-line part of *LinRel*, GP bandits and *GP-SOM*.

6 Experiment Design and Results

We tested the performance of the *ImSe* system in simulations (Section 6.1) comparing it to *LinRel* and *GP-UCB* and in real-life experiments (Section 6.2), where we couldn't afford comparing to computationally expensive methods and involved real users only for *GP-SOM*, Random and only Exploitation settings. Each of the following sections presents the design of the experiments and the results obtained in them.

6.1 Simulations

Due to the computational complexity of *LinRel* and GP bandits, we did not test these algorithms in real-life settings with large dataset but instead we tested them in simulations. In order to compare the performance of the three algorithms, we ran a set of simulation experiments.

We used the MIRFLICKR-25000 dataset [HL08] with 3 sets of visual descriptors: texture, shape and color [MJHL10]. We consider 3 visual aspects of each image when constructing a map and the closest image is determined by a harmonic mean of similarities in visual aspects:

$$\frac{1}{(1/d_t + 1/d_s + 1/d_c),} \tag{34}$$

where d_t is distance from a datapoint to the model vector in texture space, d_s in shape space, and d_c in color space.

6.1.1 The User Model

We assume that the choice of one of the presented images is a random process, where more relevant images are more likely to be chosen. In our simulation experiments, we will rely on the user model proposed in [AGL+11], which has been shown to be a close approximation of real user behavior. We assume a similarity measure $S(x_1, x_2)$ between images x_1, x_2, which also measures the relevance of an image x compared to an ideal target image t by $S(x, t)$. Let $0 \leq \lambda \leq 1$ be the uniform noise in the user's choice. The probability of choosing image $x_{i,j}$ is given by:

$$D\{x_i^t = x_{i,j} \mid x_{i,1}, \ldots, x_{i,k}; t\} = (1 - \lambda)\frac{S(x_{i,j}, t)}{\sum_{j=1}^{k} S(x_{i,j}, t)} + \frac{\lambda}{n}. \tag{35}$$

Assuming a distance function $d(\cdot,\cdot)$, a possible choice for the similarity measure is $S(x,t) = d(x,t)^{-a}$ with parameter $a > 0$. With the polynomial similarity measure, the user's response depends on the relative size of the image distances to the ideal target image. We use Euclidean norm as the distance measure between image x and the target image t. In all the experiments, the values of a and λ were kept constant at 4 and 0.1, respectively (the optimal values based on [AGL+11]).

6.1.2 Simulations Results

The reported results are averaged over 100 searches for randomly selected target images from the dataset. We also tested the influence of n, i.e. the number of images displayed at each iteration, on the performance of the algorithms. In our experiments n was set to 5, 10 and 20. The results are summarized in Figure 6, where we show the average number of iterations to find the target image with respect to the size of n.

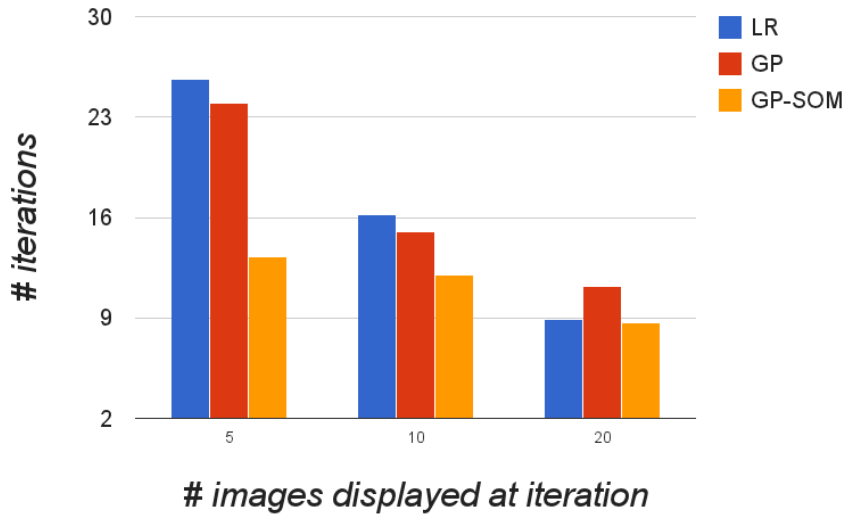

Figure 6: Comparison of the performance of *LinRel*, GB UCB and *GB-SOM*.

GP-SOM significantly outperforms *LinRel* and GP bandits for smaller values of n. For large values of n, there is no significant difference between *LinRel* and *GP-SOM*, however *GP-SOM* is more computationally efficient than *LinRel*, i.e. *LinRel* is much slower and does not scale up to large datasets of images. We believe this result is

due to our enhanced multiple image selection policy with temporal pseudo feedback that allows us to select diverse images at every iteration and it has the most effect with small n.

6.2 Real-life Experiments

The Image Retrieval system *ImSe* was implemented in *Python* programming language with scientific libraries *NumPy* and *SciPy* and deployed with *Django* framework and *Apache* at the university server. The system is available under the url `<http://hand.hiit.fi/imse/start/?user='thesis'>`.

6.2.1 Task Design

Image Retrieval is sometimes classified in the literature [CMM+00] by the type of target. According to this criteria image search consists of the following categories.

- *Target search.* Target search is the situation when the user is looking for a particular image in the database. For instance, the user might want to find a very specific image of himself/herself with a favorite cat at the age of 4 or some well-known historical photography of a famous cathedral in the city.
- *Category search.* In this situation the user doesn't have a concrete target in mind, but he or she will be satisfied with any of the images from the desired category. In this case the user might want to find a photo of a 4-year-old girl or a building in the baroque style.
- *Search by association.* In this type of task the user is just browsing a big collection of images without knowing how the target may look like, but only with a concept in mind. For example, the user may be willing to illustrate such concept as 'youth' or illustrate an essay about the history of some city a century ago.

Other studies may identify more categories of search, but in our work we will concentrate on the tasks identified by [CMM+00] and try to investigate what kind of algorithms supports users in performing these tasks the best.

We designed three subtasks for each of these types. In the "Target search" task, we asked the users to find a particular image randomly chosen from a database. We

used images of: a) water, b) insect, c) city. In "Category search", the users were asked to find images from a certain category. The selected categories were: a) night, b) flowers, c) sunset. In the "Open search" task, the users were given a description of a situation that requires a set of images as an answer. The three subtasks were: a) to find and image to illustrate an essay about wild life, b) to find an image to attach to an invitation to a mountain trip, c) to help a journalist to find an illustration to an article about spring.

Task 1. Target search

Please find the image that is shown in the top left corner of the screen. If you find an exact image, finish the experiment by pressing the 'Finish' button, otherwise terminate when you don't feel that continuing the experiment will bring you closer to the target. You will perform 3 subtasks in this task. We ask you to mark (as Excellent, Good or Satisfactory) as many images as you can during each of the session, but at least 3 images in any of the subtasks. If you don't like the target and/or first images, you can update the page.

a) Follow the link to look for an image with *water*

b) Follow the link to look for an image with an *insect*

c) Follow the link to look for an image with *city*

Task 2. Category search

In this task we ask you to find as many images of a particular category as you can. You will perform 3 subtasks in this task. As previously, we ask you to mark images as Excellent, Good and Satisfactory. Please find as many images of the category of each subtask as you can, but at least 5 images in every subtask.

a) Follow the link to look for images of category NIGHT

b) Follow the link to look for images of category FLOWERS

c) Follow the link to look for images of category SUNSET

Task 3. Open-ended search

In the last task we ask you to perform an open-ended search, where you have to find suitable images for 3 different scenarios. As usual, mark as many images as you find relevant with marks Excellent, Good or Satisfactory. Mark at least 5 images in each search. You are welcome to be creative in this task!

a) You are looking for an image to accompany your essay on wild life. Use this link to perform this task.

b) You want to invite a friend for a mountain trip and looking for an illustration to your invitation letter. Use this link to perform this task.

c) You are a journalist writing an article about spring. Find an appropriate illustration for it. Use this link to perform this task.

6.2.2 System Interface.

ImSe has a very simple user interface designed to be used without any initial training. The basic interface of the system is presented in Figure 7. In the "Target search" task, the target image is displayed in the top left corner of the screen. At each iteration, the users were presented with 10 images.

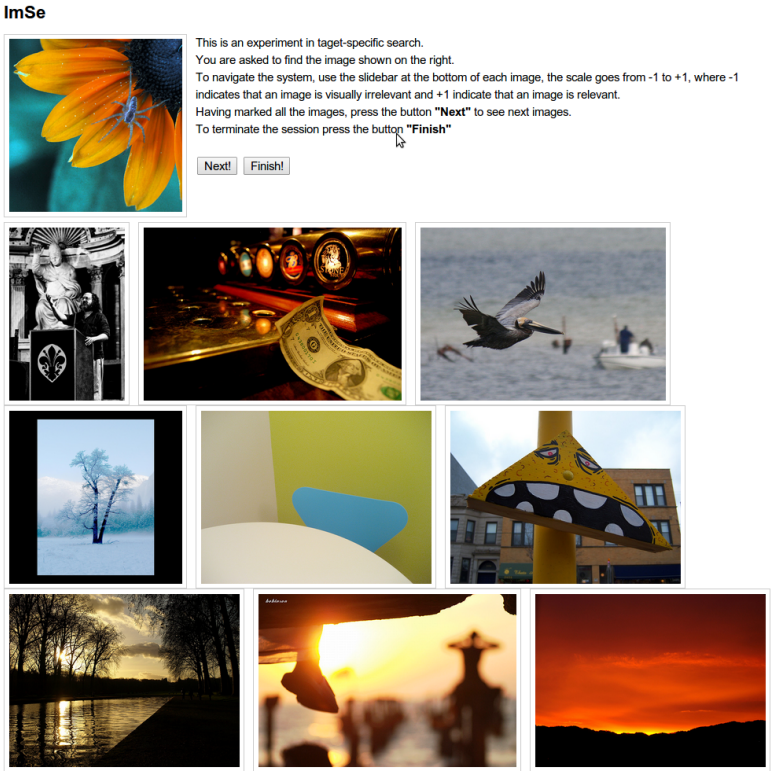

Figure 7: *ImSe* system interface

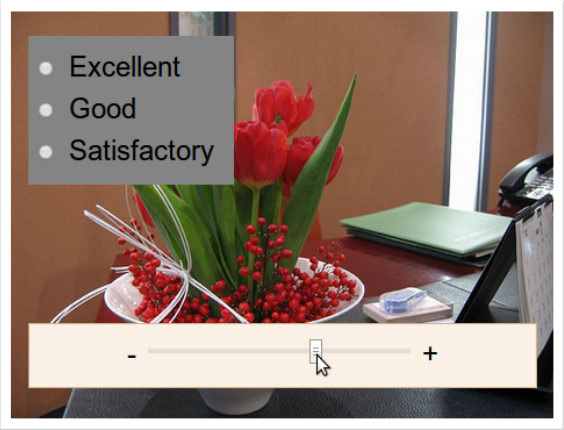

Figure 8: ImSe image control panel

When a mouse hover around an image, the control panels appear. The displayed images include two control panels (Figure 8). To navigate the system, the slidebar at the bottom of each image is used. The scale runs from -1 to +1, where -1 indicates that an image is irrelevant and +1 indicates that an image is relevant. Through this slidebar, the users provide their relevance feedback on the displayed images. In the top left corner, there are 3 radio buttons that were added only for testing, the input from which is used for the system evaluation. The users were asked to mark all the appropriate images either as "Satisfactory" (sharing some of the features, but not sufficient to substitute the target image), "Good" (close to the target, but not exact) or "Excellent" (almost identical to the target). The users were not required to grade all the images, but only those they find useful for prediction. To navigate the system, the "Next" button is used. Having completed the task, the user presses the "Finish" button to terminate the search (Figure 7). We collect the information about the images presented to the users, the feedback and all the marks ("Excellent", "Good" and "Satisfactory") provided by the users. We did not ask the users to run the search for a specific number of iterations but instead we allowed them to finish the search when they felt satisfied with the results. Below we show the instructions that were provided to the users.

System usage

We ask you perform several tasks in different system configurations. Please watch the video attached about system usage.

To navigate the system, use the slidebar at the bottom of each image, the scale goes from -1 to +1, where -1 indicates that an image is visually irrelevant and +1 indicate that an image is relevant, press 'Next' to get next prediction. You do not need to grade all the

images, but only those you find useful for navigation (both relevant or irrelevant). Mark all appropriate images that you find during the search as

- Excellent (almost identical to the target),
- Good (close, but not exact) and
- Satisfactory (sharing the some of mains features, but not sufficient to substitute the target).

Marks from slidebar go to the system for navigating and marks Excellent, Good and Satisfactory go to us to evaluate system's success. Please note that no text is used in the system, only visual similarity between images, it will help you to navigate in the space. Mark all the appropriate images before terminating, press 'Finish' to terminate the search. Please note that it is not possible to go back to the previous step, so you have to mark the images as they are presented to you (but you don't need to mark all of the presented images)! Do not forget about possibility to give negative feedback! Use Chrome browser!

Now you can try it out here.

6.2.3 Logging

During the interactions with the system some information about the system usage was recorded and specially for this purposes a structure of relational database was constructed. There are 3 instances that are recorded, they are Image, Experiment and Iteration.

Image is used mostly inside the system and it allows to refer any image stored there. It has the following fields:

field	**type**	**meaning**
index	integer	position of the image in the kernel
filename	string	path and name of the file containing the image

Table 2: Image database structure

The most important for log analysis is the Experiment table in the database, it contains all the basic information about the task execution. In order to analyze and evaluate the performance of different algorithms we record such information as the time when the task started and ended and how may images of each type close to the target were marked.

field	**type**	**meaning**
session id	string	the session id in the cookies
user name	string	the name assigned to users during the experiment
algorithm	string	the name of the algorithm utilized in the experiment, possible values are GP-SOM, Exploitation and Random
target type	string	the type of the target in the task, can be either 'target', 'category' or 'open' search
category	string	the description of image that was a target in the search, for example, 'water', 'insect', 'sunset', 'wild life', 'mountain trip'
target	Image	the link to an Image object, if 'target' search was used, otherwise None
interactions	integer	the number of interactions performed during the search session
excellents	integer	the number of excellent images marker during the search
goods	integer	the number of good images marker during the search
satisfactories	integer	the number of satisfactory images marker during the search
images number total	integer	total number of images in the database retrieved
images number iteration	integer	number of images shown to the user at each iteration
finished	boolean	true is user completed the search by pressing the button to terminate the search
time start	integer	time stamp for the beginning of the experiment
time finish	integer	time stamp when the user completed the task

Table 3: Experiment database structure

For more detailed look at the experiment data we can refer to the Iteration table that contains information about every step from the experiment. In this table we can find information about what images were shown to the user, what feedback for them was received, when each of the relevant images was marked.

field	type	meaning
experiment	Experiment	the reference to the corresponding experiment from Experiment table
iteration	integer	the number of this iteration
images shown	string	indexes of images shown at the iteration
feedback	string	feedback from slidebar for the shown images
marks	string	the marks that were given to the images shown if any
time	integer	time stamp at which the iteration was done

Table 4: Experiment database structure

6.2.4 Experiments Configuration

We recruited 18 participants (9 male and 9 female) with different backgrounds (linguistics, arts, biology, computer science, physics etc.) and of different ages (22-55) to participate in the experiments. We used the same MIRFLICKR-25000 dataset [HL08] as in the simulation experiments. We used different permutations for different algorithms in order to minimize the influence of order in which different settings were performed by the users. However, each subtask was performed an equal number of times with each of the tested settings, i.e. *GP-SOM*, exploitation only (Expl) and random (Rand).

Users were performing each type of task (Target, Category and Open search) in 3 different system configurations: a) algorithm used for image prediction was *GP-SOM* b) strategy was 'only Exploitation' c) images at each iteration were shown randomly. Trying to minimize the influence of order in which the systems are tried by users, everyone used the systems in different order and the amount of trials in different configurations were equal when aggregated for all the users. The configuration for the experiment can be found in Table 5.

task	target	target	target	category	category	category	open	open	open
user	water	insect	city	night	flowers	sunset	wildlife	mountain	spring
1	gp-som	Expl	Rand	Expl	gp-som	Rand	Rand	gp-som	Expl
2	Expl	gp-som	Rand	Rand	gp-som	Expl	gp-som	Rand	Expl
3	Rand	gp-som	Expl	gp-som	Rand	Expl	Expl	Rand	gp-som
4	gp-som	Rand	Expl	Expl	Rand	gp-som	Rand	Expl	gp-som
5	Expl	Rand	gp-som	Rand	Expl	gp-som	gp-som	Expl	Rand
6	Rand	Expl	gp-som	gp-som	Expl	Rand	Expl	gp-som	Rand
7	Expl	gp-som	Rand	gp-som	Expl	Rand	Rand	gp-som	Expl
8	Rand	gp-som	Expl	Expl	gp-som	Rand	gp-som	Rand	Expl
9	gp-som	Rand	Expl	Rand	gp-som	Expl	Expl	Rand	gp-som
10	Expl	Rand	gp-som	gp-som	Rand	Expl	Rand	Expl	gp-som
11	Rand	Expl	gp-som	Expl	Rand	gp-som	gp-som	Expl	Rand
12	gp-som	Expl	Rand	Rand	Expl	gp-som	Expl	gp-som	Rand
13	Rand	gp-som	Expl	Expl	gp-som	Rand	gp-som	Expl	Rand
14	gp-som	Rand	Expl	Rand	gp-som	Expl	Expl	gp-som	Rand
15	Expl	Rand	gp-som	gp-som	Rand	Expl	Rand	gp-som	Expl
16	Rand	Expl	gp-som	Expl	Rand	gp-som	gp-som	Rand	Expl
17	gp-som	Expl	Rand	Rand	Expl	gp-som	Expl	Rand	gp-som
18	Expl	gp-som	Rand	gp-som	Expl	Rand	Rand	Expl	gp-som

Table 5: Combinatorial design of the experiment for 18 subjects, 3 types of tasks and 3 algorithms.

6.2.5 Real-life Experimental Results.

Having conducted all the experiments we collected the data from logs to analyze the performance of different settings. Table 6 summarizes the average number of iterations that it took the users to complete each type of task. In all types of tasks, the average number of iterations for *GP-SOM* and Exploitation algorithm was approximately the same, but clearly smaller than for the Random algorithm. In general, the Target search took the longest reflecting the fact that searching for a specific target takes more iterations than a less specific type of search.

Algorithm	*Target search*	*Category search*	*Open search*
Random	20.8	14.4	19.1
Exploitation	16.1	12.2	13.8
GP-SOM	15.4	12.6	13.8

Table 6: Number of iterations for *GP-SOM*, Exploitation and Random algorithms for different types of tasks.

In Table 7, we report the average time spent on each iteration for different types of settings. The results show that the users need to spend more time at every iteration of *GP-SOM*, observing how the results have changed, compared to Exploitation, where the next iteration is very similar to the previous one and there is very little variation in the type of images presented.

Algorithm	**Target search**	**Category search**	**Open search**
Random	0.43	0.43	0.37
Exploitation	0.37	0.28	0.25
GP-SOM	0.43	0.42	0.31

Table 7: Time in minutes spent at each iteration for *GP-SOM*, Exploitation and Random algorithms for different types of tasks.

Taking into consideration the images bookmarked as 'Excellent', 'Good' and 'Satisfactory', we define the following measure of weighted average mark as a quality of search that is used in the further analysis of the results:

$$(3 \cdot \#excellent + 2 \cdot \#good + 1 \cdot \#satisfactory)/\#iterations. \quad (36)$$

Table 8 compares the performance for all the three algorithms in terms of the defined measure for different categories of search. We report the average cumulative score per iteration. The proposed algorithm *GP-SOM* outperforms Exploitative and Random algorithms for all types of searches. Judging by the weighted number of bookmarked images by the users, the Target search is the most difficult due to the specific nature of the task. The best result by *GP-SOM* was obtained in Category search. As expected, the random algorithm performs the worst. The pure exploitation algorithm performs worse than *GP-SOM* in all tasks indicating that *GP-SOM* helps the user to find more suitable images.

Search type	**Algorithm**	**Weighted average of bookmarked images per iteration**
Target	Random	0.95
	Exploitation	1.56
	GP-SOM	1.64
Category	Random	1.51
	Exploitation	2.06
	GP-SOM	2.85
Open	Random	1.34
	Exploitation	2.06
	GP-SOM	2.58

Table 8: Comparison of the performance of *GP-SOM*, Exploitation and Random algorithms in different types of tasks.

Below, we discuss the results for all types of search separately.

Target search. In Target search, the users were shown a particular target image to find, but as it may take a long time to find a very specific image, we allowed the users to terminate the search if they felt they found enough close substitutes for the target image. In our case, images marked as "Excellent" were close substitutes for the target image. In Figure 9, we can see that the number of "excellent" images found by the users grows with the number of iterations in *GP-SOM*, Exploitation and Random settings. We can clearly see that the strategy with Exploitation only has converged after 12 iterations and the users could not find any more "excellent" images as they are stuck in a local maximum. *GP-SOM*, where exploration is also present, continues to find suitable images throughout the whole search session.

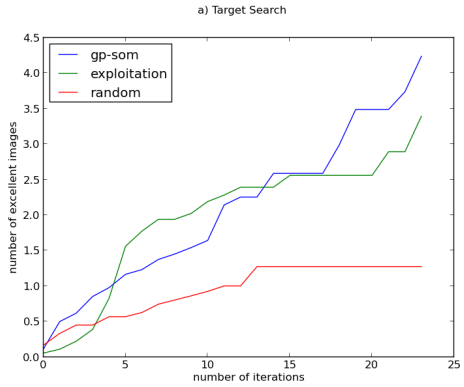

Figure 9: Cumulative number of "excellent" bookmarked images in target search.

Table 9 shows detailed information on how many images of each type were bookmarked by users on average per one iteration. Users of *GP-SOM* were more successful than those of the other algorithms in finding "Excellent" and "Good" images. For Target search, "Excellent" images are of great importance as they are similar enough to the target image to substitute it.

Target search			
Algorithm	#excellent/#iterations	#good/#iterations	#satisfactory/#iterations
Random	0.057	0.175	0.431
Exploitation	0.134	0.317	0.521
GP-SOM	0.151	0.346	0.493
Category search			
Random	0.30	0.197	0.213
Exploitation	0.4	0.318	0.209
GP-SOM	0.68	0.297	0.212
Open-ended search			
Random	0.137	0.282	0.363
Exploitation	0.341	0.361	0.309
GP-SOM	0.484	0.38	0.364

Table 9: The average number of bookmarked images per iteration for *GP-SOM*, Exploitation and Random algorithms in Target, Category and Open-ended searches.

Category search. We obtained category annotations for some of the image classes from MIRFLICKR dataset and we used this information for additional evaluation of the performance of the "Category search". We calculated the precision for all three algorithms and got on average 12% precision for Random browsing, 23% for Exploitation search and 26% for the *GP-SOM* based system (Table 10).

Algorithm	**Precision**
Random	12%
Exploitation	23%
GP-SOM	26%

Table 10: Precision in Category search in *GP-SOM*, Exploitation and Random algorithm.

The graph of cumulative quality score was obtained using Equation 36 for Category search and it is shown in Figure 10. It took only 5 iterations for *GP-SOM* to outperform the Exploitation setting. Table 9 shows the average number of images of every type per iteration. *GP-SOM* obtains the highest score for "Excellent" images but it is slightly worse at finding "Good" images, while the amount of "Satisfactory" images retrieved is approximately the same in all settings.

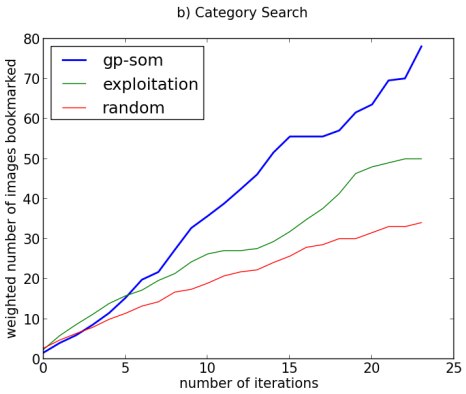

Figure 10: Weighted cumulative number of bookmarked images in category search

Open-ended Search. When analyzing the Open-ended search we looked at the weighted average of number of bookmarked images over time and total number of images of each category per iteration. The results are illustrated in Figure 11. As we can see in Table 9, *GP-SOM* performs well in all categories but its performance is significantly high in finding "Excellent" images.

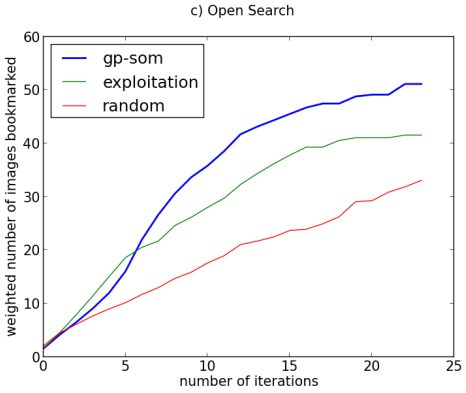

Figure 11: Weighted cumulative number of bookmarked images in open ended search.

6.2.6 Detailed Discussion

In Figure 12 we can see weighted average number of bookmarked images in each of all the tasks separately that were offered to the users. The difference in the performance of different algorithms depends on the task. For example, in the Target search of insects, category search of flowers and sunset and open-ended search of illustration

for a wild-life, the *GP-SOM* algorithms is performing better and better after a few iterations and outperforms Exploitation and Random algorithms that are converged too fast and do not help to find better results with time. The task of finding an insect image in the database is not a trivial as finding, for example, water image, especially as there are not that many insect images in the database. We are glad to demonstrate that the *GP-SOM* algorithm was successful in supporting the user in this type of a task.

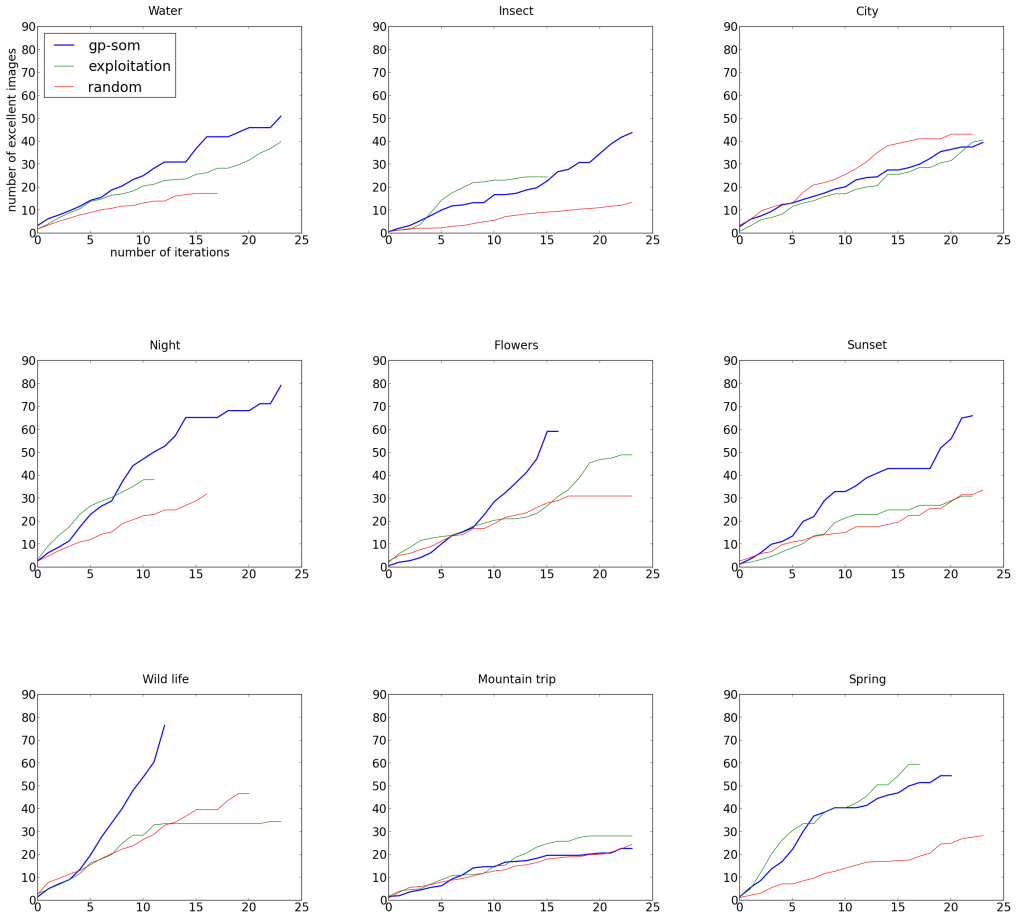

Figure 12: Weighted cumulative number of bookmarked images in category search

In general, we can notice that people managed with category search better that with the other types and managed to find the largest number of relevant images, even though open-ended tasks gave them more freedom to define what the appropriate target is.

There are 2 tasks in which none of the settings performed well – they are target search for city and open-ended search for image for a invitation for a mountain trip.

Besides, mountain trip search performed the worst among other tasks and we believe that users failed to complete the task because there were not enough images of this content in the database. We explain no difference in the results reached by the users performing the task in different settings in the search of a particular image of a city by lack of appropriate image features descriptors distinguishing this type of images. It is actually almost impossible to see the difference in city images judging only by 3-D color histogram color and more sophisticated feature extraction techniques are needed. In the task on finding images to illustrate an article about spring we see that Exploitation and *GP-SOM* algorithms are performing almost identical, but much better than a Random algorithm. It shows that color features used to solve this task helped a lot to distinguish them from other images, but the task was easy and flexible enough to cope with it even without using Exploration.

7 Conclusions and Discussion

We proposed a model of time-efficient Relevance Feedback mechanism for Content-Based Image Retrieval with an Exploration/ Exploitation algorithm that balances between presenting images about which the system is uncertain and images about which the system has high confidence. The model uses clusters of images obtained through Self-Organizing Map to serve as the first layer in GP bandits in order to overcome scalability issues in large unannotated databases.

We performed simulations and real-life experiments to test the performance of the proposed algorithm and it was found to outperform the competing algorithms in all the criteria: convergence, time-efficiency and usability. Simulation results demonstrated that the proposed mechanism of selecting multiple images ensures their diversity and achieves better performance when a small number of images are presented at each iteration.

The hierarchical technique used in *GP-SOM* facilitates running image searches much faster and can be easily used in real-life systems. We created an Image Retrieval system *ImSe* incorporating the *GP-SOM* algorithm and tested its performance in different types of searches. The experimental results show that in all types of search, users of the *GP-SOM* algorithm were able to find more relevant images in a fewer number of iterations compared to other algorithms.

There are a number of interesting observations related to the experimental results. First, there we observed a clear difference between male and female behavior in Image Retrieval that we did not analyze in details: female participants tend to spend more time per iteration selecting images compared to male participants. Another interesting observation concerns performance of the *GP-SOM* algorithm in tasks with a varying level of "complexity". For instance, when the task is more difficult (few similar images in the database or the image contains very specific features), *GP-SOM* gives us a large improvement over a strategy based only on exploitation. We will conduct more tests in the future to fully confirm these observations.

An obvious direction for ongoing research is to learn individual feature kernel combinations instead of relying only on one feature and incorporate this into a hierarchical scenario. We would also like to deploy even bigger variants of the system with more sophisticated visual features. Another interesting process we would like to investigate is ensuring the diversity among multiple selected objects as an optimization problem. It also could be interesting to model the stochastic environment with non-

parametric methods, where the distribution of the rewards is changing with time. Moreover, we are curious to investigate other types of feedback and manipulations from the user.

References

ACBF02 Auer, P., Cesa-Bianchi, N. and Fischer, P., Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47,2-3(2002), pages 235–256.

AGL+11 Auer, P., Głowacka, D., Leung, A., Lim, S. H., Medlar, A. and Shawe-Taylor, J., Study of exploration-exploitation trade-offs with delayed feedback, fp7–216529 pinview. Technical Report, European Community's Seventh Framework Programme, 2011.

AHK+10 Auer, P., Hussain, Z., Kaski, S., Klami, A., Kujala, J., Laaksonen, J., Leung, A., Pasupa, K. and Shawe-Taylor, J., Pinview: Implicit feedback in content-based image retrieval. *JMLR: Workshop and Conference Proceedings: Workshop on Applications of Pattern Analysis*, 11, pages 51 – 57.

Aue02 Auer, P., Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3, pages 397 – 422.

BBDS08 Busoniu, L., Babuska, R. and De Schutter, B., A comprehensive survey of multiagent reinforcement learning. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 38,2(2008), pages 156–172.

CBL09 Chatzichristofis, S. A., Boutalis, Y. S. and Lux, M., Img(rummager): An interactive content based image retrieval system. *2nd International Conference on Similarity Search and Applications (SISAP)*, Prague, Czech Republic, 2009, pages 151–153.

CCP Corinth, R., Crome, D., Palm, L., Ukhanova, N. and Wieczorek, P., WWW-picslikethat. URL `http://www.picslikethat.com/`.

CF80 Chang, N.-S. and Fu, K.-S., Query-by-pictorial-example. *Software Engineering, IEEE Transactions on*, ,6, pages 519–524.

CMM+00 Cox, I. J., Miller, M. L., Minka, T. P., Papathomas, T. V. and Yianilos, P. N., The bayesian image retrieval system, pichunter: theory, implementation, and psychophysical experiments. *Image Processing, IEEE Transactions on*, 9,1(2000), pages 20–37.

DKN04 Deselaers, T., Keysers, D. and Ney, H., Classification error rate for quantitative evaluation of content-based image retrieval systems. *International Conference on Pattern Recognition*, Cambridge, UK, 2004, pages 505–508.

FSA96 Frankel, C., Swain, M. J. and Athitsos, V., Webseer: An image search engine for the world wide web.

GCP11 Gorisse, D., Cord, M. and Precioso, F., Salsas: Sub-linear active learning strategy with approximate k-nn search. *Pattern Recognition*, 44,10(2011), pages 2343–2357.

Git79 Gittins, J. C., Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 148–177.

goo11 Google image labeler, 2011. URL `http://images.google.com/imagelabeler/`.

GRK+13 Glowacka, D., Ruotsalo, T., Konuyshkova, K., Kaski, S., Jacucci, G. et al., Directing exploratory search: Reinforcement learning from user interactions with keywords. *Proceedings of the 2013 international conference on Intelligent user interfaces*. ACM, 2013, pages 117–128.

Hel09 Hellinger, E., Neue begründung der theorie quadratischer formen von unendlichvielen veränderlichen. *Journal für die reine und angewandte Mathematik*, 136, pages 210–271.

HL08 Huiskes, M. J. and Lew, M. S., The mir flickr retrieval evaluation. *MIR '08: Proceedings of the 2008 ACM International Conference on Multimedia Information Retrieval*, 2008.

HLP+10 Hussain, Z., Leung, A. P., Pasupa, K., Hardoon, D. R., Auer, P. and Shawe-Taylor, J., Exploration-exploitation of eye movement enriched multiple feature spaces for content-based image retrieval. In *Machine Learning and Knowledge Discovery in Databases*, Springer, 2010, pages 554–569.

IqA02 Iqbal, Q. and Aggarwal, J. K., Cires: A system for content-based retrieval in digital image libraries. *Invited session on Content Based Image Retrieval: Techniques and Applications International Conference*

on Control, Automation, Robotics and Vision (ICARCV), Singapore, 2002, pages 205–210.

JB08 Jing, Y. and Baluja, S., Visualrank: Applying pagerank to large-scale image search. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30,11(2008), pages 1877–1890.

KG13 Konyushkova, K. and Glowacka, D., Content-based image retrieval with hierarchical gaussian process bandits with self-organizing maps. *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2013.

KKOH92 Kato, T., Kurita, T., Otsu, N. and Hirata, K., A sketch retrieval method for full color image database-query by visual example. *Pattern Recognition, 1992. Vol. I. Conference A: Computer Vision and Applications, Proceedings., 11th IAPR International Conference on*. IEEE, 1992, pages 530–533.

KLM96 Kaelbling, L. P., Littman, M. L. and Moore, A. W., Reinforcement learning: A survey. *arXiv preprint cs/9605103*.

Koh01 Kohonen, T., *Self-organizing maps*, volume 30. Springer Verlag, 2001.

KS06 Kocsis, L. and Szepesvári, C., Bandit based monte-carlo planning. In *Machine Learning: ECML 2006*, Springer, 2006, pages 282–293.

LCLS10 Li, L., Chu, W., Langford, J. and Schapire, R. E., A contextual-bandit approach to personalized news article recommendation. *Proceedings of the 19th international conference on World wide web*. ACM, 2010, pages 661–670.

LKLO00 Laaksonen, J., Koskela, M., Laakso, S. and Oja, E., Picsom–content-based image retrieval with self-organizing maps. *Pattern Recognition Letters*, 21,13(2000), pages 1199–1207.

LSDJ06 Lew, M. S., Sebe, N., Djeraba, C. and Jain, R., Content-based multimedia information retrieval: State of the art and challenges. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, 2,1(2006), pages 1–19.

Mac92 MacKay, D. J., Information-based objective functions for active data selection. *Neural computation*, 4,4(1992), pages 590–604.

MJHL10 Mark J. Huiskes, B. T. and Lew, M. S., New trends and ideas in visual concept detection: The mir flickr retrieval evaluation initiative. *MIR '10: Proceedings of the 2010 ACM International Conference on Multimedia Information Retrieval*, New York, NY, USA, 2010, ACM, pages 527–536.

MSP03 Müller, H., Squire, D. M. and Pun, T., Learning from user behavior in image retrieval: application of the market basket analysis. *Int. J. of Comp. Vision, Special Issue on Content-Based Image Retrieval, to appear.*

MV11 Mironica, I. and Vertan, C., An adaptive hierarchical clustering approach for relevance feedback in content-based image retrieval systems. *Signals, Circuits and Systems (ISSCS), 2011 10th International Symposium on*, 30 2011-july 1 2011, pages 1 –4.

NS04 Nguyen, H. T. and Smeulders, A., Active learning using pre-clustering. *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004, page 79.

Ope12 WWW-opencv, 2012. URL `http://opencv.org/`.

PACJ07 Pandey, S., Agarwal, D., Chakrabarti, D. and Josifovski, V., Bandits for taxonomies: A model-based approach. *SIAM Intl. Conf. on Data Mining (SDM)*, 2007.

PCA07 Pandey, S., Chakrabarti, D. and Agarwal, D., Multi-armed bandit problems with dependent arms. *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pages 721–728.

PZ99 Pass, G. and Zabih, R., Comparing images using joint histograms. *Multimedia systems*, 7,3(1999), pages 234–240.

Ras06 Rasmussen, C. E., Gaussian processes for machine learning.

RHC99 Rui, Y., Huang, T. S. and Chang, S.-F., Image retrieval: Current techniques, promising directions, and open issues. *Journal of visual communication and image representation*, 10,1(1999), pages 39–62.

RW06 Rasmussen, C. E. and Williams, C. K. I., *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

SB98 Sutton, R. S. and Barto, A. G., *Reinforcement learning: An introduction*, volume 1. Cambridge Univ Press, 1998.

SKG98 Smeulders, A. W., Kersten, M. L. and Gevers, T., Crossing the divide between computer vision and data bases in search of image databases. In *Visual Database Systems 4 (VDB4)*, Springer, 1998, pages 223–239.

SKKS09 Srinivas, N., Krause, A., Kakade, S. M. and Seeger, M., Gaussian process bandits without regret: An experimental design approach. *CoRR*, abs/0912.3995.

SS94 Stricker, M. and Swain, M., The capacity of color histogram indexing. *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*. IEEE, 1994, pages 704–708.

STC04 Shawe-Taylor, J. and Cristianini, N., *Kernel methods for pattern analysis*. Cambridge university press, 2004.

SF12 WWW-superfish, 2012. URL `http://wwws.superfish.com/`.

SWS+00 Smeulders, A. W. M., Worring, M., Santini, S., Gupta, A. and Jain, R., Content-based image retrieval at the end of the early years. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22,12(2000), pages 1349–1380.

WD92 Watkins, C. J. and Dayan, P., Q-learning. *Machine learning*, 8,3-4(1992), pages 279–292.

WM01 Wu, P. and Manjunath, B., Adaptive nearest neighbor search for relevance feedback in large image databases. *Proceedings of the ninth ACM international conference on Multimedia*. ACM, 2001, pages 89–97.

WR96 Williams, C. K. and Rasmussen, C. E., Gaussian processes for regression.

YSLH03 Yee, K.-P., Swearingen, K., Li, K. and Hearst, M., Faceted metadata for image search and browsing. *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 2003, pages 401–408.

ZH03 Zhou, X. S. and Huang, T. S., Relevance feedback in image retrieval: A comprehensive review. *Multimedia systems*, 8,6(2003), pages 536–544.