Date of acceptance Grade

Instructor

Designing interfaces for exploratory content based image retrieval systems

Sayantan Hore

Helsinki October 9, 2015

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
Sayantan Hore

Työn nimi — Arbetets titel — Title

Designing interfaces for exploratory content based image retrieval systems

Oppiaine — Läroämne — Subject
Computer Science

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
	October 9, 2015	0 pages + 57 appendices

Tiivistelmä — Referat — Abstract

Content Based Image Retrieval or CBIR systems have become the state of the art image retrieval technique over the past few years. They showed commendable retrieval performance over traditional annotation based retrieval. CBIR systems use relevance feedback as input query. CBIR systems developed so far did not put much effort to come up with suitable user interfaces for accepting relevance feedback efficiently i.e. by putting less cognitive load to the user and providing a higher amount of exploration in a limited amount of time. In this study we propose a new interface "FutureView" which allows peeking into the future providing access to more images in less time than traditional interfaces. This idea helps the user to choose more appropriate images without getting diverted. We used Gaussian process upper confidence bound algorithm for recommending images. We successfully compared this algorithm with Random and Exploitation algorithms with positive results.

ACM Computing Classification System (CCS):
A.1 [Introductory and Survey],
I.7.m [Document and text processing]

Avainsanat — Nyckelord — Keywords
layout, summary, list of references

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — övriga uppgifter — Additional information

Contents

1 Introduction **1**

2 Content Based Image Retrieval **2**

3 Related Work **3**

3.1 PicSOM 3

3.1.1 Inspirations from PicSOM 5

3.1.2 Differences from PicSOM 5

3.2 Pinview 6

3.3 Inspirations from Pinview 7

3.4 Differences from Pinview 7

4 Theoretical background **7**

4.1 Reinforcement Learning 7

4.1.1 Exploration and Exploitation 8

4.2 Bandit Problems 9

4.3 Bandit algorithms 10

4.3.1 Gittin's Indices 10

4.3.2 Upper Confidence Bound 11

4.4 UCB algorithms for independent armed bandits 12

4.4.1 UCB1 12

4.4.2 UCB1-Tuned 12

4.5 UCB algorithms for dependent armed bandits 13

4.5.1 LinRel 13

4.5.2 LinUCB 13

5 Gaussian Process Bandits **14**

5.1 Regression Problems 14

5.2 Gaussian Distribution 17

5.3 Bayesian Linear Regression Model 17
5.4 Bayes' Rule 18
5.5 The Standard Linear Model 19
5.6 The Basis Function Model 21
5.7 Gaussian Process 23
5.8 Applying Gaussian Process to Bandit Problems 26

6 System description **28**
6.1 Offline step 29
6.1.1 Feature extraction 29
6.1.2 The Kernel 29
6.1.3 Clustering images 30
6.2 The online step 30
6.3 The interface 31

7 Experiments **32**
7.1 Tasks and Task-Categories 32
7.2 Participants and task allocation 34

8 Results **35**

9 Discussion **39**

10 Corrective measures **40**
10.1 FutureView 41
10.2 Experimental setup 42
10.3 Participants 43
10.4 Tasks 43
10.5 Procedure 44

11 Results **45**

11.1 Excerpts from interviews 50

12 Changes suggested **50**

13 Problems with the system **51**

14 Future plan **51**

15 Conclusion **52**

References **53**

1 Introduction

With the advent of digital media, searching for media contents, especially images, became important for making applications involving image search and retrieval. Until recently, there had been no convenient technique to search for contents in images. Traditional image search systems allow users to enter texts only as a search query. This has been useful for text based information retrieval because, as the search query and the content being searched for are texts, the input can directly be mapped to the search space. This can be referred to as direct searching.

Images are not built with letters, so in case of a text based search, the keywords in the search query have to be mapped to the images first. Therefore, this is indirect searching. To achieve this mapping, all the available images have to be annotated or tagged. This annotation involves manual labour. Moreover, annotation can be wrong, also some images might not receive annotation at all. These lead to inappropriate search results. We will highlight three different such cases below.

- **False annotation:** Say, somebody has a pet dog named "Tiger". A photograph of the dog tagged as "Tiger" is uploaded to a social media image database. Now if a person is searching for a tiger in its true sense, it is very common that the system can include the photograph of that dog in the search result. As the system is not searching by the actual contents of the images, it cannot differentiate between two pictures with the same tag.

- **Missing annotation:** Say, the owner of "Tiger" forgot to attach a tag to the picture. Somebody, searching for dogs, would not receive the photograph of "Tiger".

- **Ambiguous annotation:** Say, somebody is searching for blue whales. The system will try to find different combinations of search keywords. Therefore it is common to get multiple images, some of them being whales, and the rest anything blue, like a blue shirt. The system cannot go inside an image to see whether the whale present there is blue or not. A blue whale can only be retrieved if either the whale in the picture is actually a blue one, or it is tagged as "blue whale".

To tackle the above mentioned issues, it is required to search for and by the actual contents of images. Contents of images can be represented on a computer as RGB

colour space, colour gradient, texture, shape etc. These representations are known as "features". If all the images are represented by features, appropriate pattern recognition and search algorithms can be applied to them. This image retrieval procedure is known as *content based image retrieval* or CBIR [DLW, KM05]. Researchers have been involved into building such systems over the past few years [SWS+00, CM00]. The first experiment on CBIR took place in 1992 [KKOH92] and after that several systems came out. Some of them involve multiple image features and apply machine learning techniques to retrieve images. Systems like [ZLZ01, TC01] use a supervised learning approach to train a multi-class classifier which classifies images based on available annotated training image set, but it is time consuming to train an agent before it can go live and perform. Some of the systems like [Vas13, Vas04] use unsupervised approach to cluster images based on the similarity in their contents, then first try to retrieve the most suitable cluster and from there pick up relevant images based on the search query. Systems like [LKLO00] use reinforcement learning where the learning agent collects relevance feedback [ZH03] from the users on the images they liked till the search session ends in an iterative process [CMM+00, LKLO00, VLP13]. There is no need to train it offline. These system have great potential in searching for abstract themes say "sorrow", because they let the user choose images based on what they have in mind. It reduces the chance of mis-classification or mis-clustering. Our system is modelled after this philosophy. We use *Gaussian process bandits* in reinforcement learning setting to retrieve relevant images. In the subsequent sections we include the necessary theoretical background, a brief description of similar but relevant systems, construction of our system following the theory given, experiment setup and conclusion.

2 Content Based Image Retrieval

In this section we will discuss what is content based image retrieval (CBIR) in general. Digital images are made of dots or pixels. In RGB (Red, Green, Blue) colour space each pixel is represented by a tuple holding three values like [12, 209, 99]. These three values represent the colours red, green and blue, respectively. Each value can be anywhere between 0 and 255. Combination of these three values generate all the colours with varying intensities. Adjacent pixels of various colours and intensities in a region form different textures and shapes. Therefore, any human identifiable visual object in a digital image can be broken down into a collection of pixels or a tuples with three integers. We can apply various mathematical opera-

tors or models on these tuples to retrieve objects, define edges or object boundaries [Can], detect interest points in images based on illumination changes (local maxima [Low99]) etc. We see here how we can analise the lower level building blocks of an image to come up with meaningful information about what visual objects are present in the image. Once we know an object is present in the image, we can retrieve that very image upon a query. We do not need the image to be annotated. This is the idea behind CBIR. CBIR systems extract information say, colour distribution, texture or shape from an image by representing the image as a feature vector which is a collection or vector of numeric values. Sometimes an entire image is represented by a vector or different regions of interest in an image can be represented by different feature vectors. These vectors change in values and length based on the feature they represent. For example, colour and shape descriptors for an image are different vectors. Once an image or parts of an image is coded into numbers, different mathematical tools can be applied on them. Different distance measures can be applied on these vectors to calculate the distances between various images. Here distances refer to similarity or dissimilarities among images based on overall colour presentation, presence of similar objects etc. Probabilistic measures can be applied to predict images based on a query and user's response on previous predictions. In the subsequent sections we discuss an overview of relevant CBIR systems, theoretical background and our system design in detail.

3 Related Work

We have seen many content based image retrieval systems coming up so far over the past few years. These include Query By Image Content (QBIC) by IBM [FSN95], Photobook by MIT [PPS94], Virage by Virage Technologies [BFG96], PicSOM [LKLO00] and Pinview [AHK$^+$10]. PicSOM arranges images in *Self Organizing Maps*(SOM) [Koh] and relies relevance feedback collected on example images shown to users. We have taken inspiration from PicSOM and Pinview in our study which we will discuss next.

3.1 PicSOM

PicSOM is a CBIR system that uses large unannotated image databases. It is the first system in its genre that uses SOM, especially it uses tree structured SOM (TS-

SOM) [KO90, Koi94]. The system can be configured to be used with multiple image databases and it can handle multiple features. New features can also be introduced to the system as long as the feature can be represented in fixed dimensions and Euclidean metric can be applied to measure the distances between images. The system uses five different features namely, Average Colour, Colour Moments, Texture Neighbourhood, Shape Histogram and Shape FFT (FFT stands for Fast Fourier Transform, which is Discrete Fourier Transform applied on a sequence of data. FFT reduces the computational complexity from $O(n^2)$ to $O(nlogn)$). For each of these features the system calculates the corresponding TS-SOM. The TS-SOMs are trained in a top-to-bottom manner by presenting the extracted feature vectors for each image to the corresponding TS-SOM multiple times. After the training, each node in the TS-SOM contains a model vector which roughly averages all the feature vectors mapped to that node down below. Also if the training is proper, each node at the bottom of a TS-SOM should uniquely represent one single image.

PicSOM starts with providing a choice of the image database to the user. Once the user selects the database, it shows a few images from that database to the user. The user selects a subset of the images from the shown set and continues. This subset is passed as a query to the system. The feature vectors for respective features for images in the selected subset are calculated and passed to the TS-SOMs to find similar images. Different TS-SOMs will return different images. The results need to be combined or only the TS-SOM, whose feature best represents the user's intention in the selection, have to be used. The user could be given a choice regarding which feature he wants to use for the current search. But features are not very straight forward always. For example, if the user is a layman, he might not be aware of Shape-FFT. Therefore, it is better if the system decides which feature would perform better by observing the selected subset. PicSOM has a very good solution to this problem. It passes the feature vectors for selected images to all the TS-SOMs and observes the spread of the images. The TS-SOM that represents the selected images the closest to each other or the least scattered, has the best performing feature for the current search.

The images in the selected subset are marked as positive and the images ignored by the user are marked as negative. The system uses a low pass filter mask to convolve the TS-SOM. This way the positive effect of the nodes, mainly associated with images marked so, are conveyed to the neighbouring nodes. In the next iteration, images from nodes with high positive values are selected and shown to the user and the process continues.

PicSOM restricts the amount of exploration in the lower level SOMs under the best match using a predefined value. This fixed value sometimes removes appropriate images and may go for unnecessary exploration some other time. This amount of exploration could be controlled depending on context and other parameters from the ongoing search sessions.

3.1.1 Inspirations from PicSOM

We have taken three major ideas from PicSOM for our CBIR implementation, which are,

- **QBPE** - QBPE stands for Query by Pictorial Example [CF80]. The idea is to replace text queries by images themselves. The query image can be an existing one from the image dataset being used or any random image choosen by the user. In our system we show random images from the image set to the user and the user selects the suitable ones. Those selected images are sent to the system as queries.
- **RF** - RF stands for relevance feedback [ZH03]. The user selects and rates images and those ratings are sent to the system as positive feedback or reward.
- **Hierarchical ordering of images** - PicSOM represents image features in SOM [Koh]. In our system we represent image features in hierarchical K-Means [HWd] clusters (in the first experiment only).

3.1.2 Differences from PicSOM

- PicSOM uses a lot of features in combination whereas we only use colour as a feature.
- PicSOM uses TS-SOM to organise images hierarchically. We used hierarchical clustering in the first experiment and later we removed that structure.
- PicSOM restricts the search space in a lower SOM by a predefined value, under the best match in the upper level SOM. We use a predefined exploration rate to do the same.

3.2 Pinview

Pinview uses PicSOM engine internally. It incorporates multiple input modalities for accepting relevance feedback namely click and eye movement. It uses multi-armed bandit algorithm, *Linrel* [Aue02], to calculate relevance score of an image as a linear combination of feature vectors of previously shown images. Say at iteration t, the feature vectors of all shown images over the past $t-1$ iterations are represented by the matrix of row vectors $\mathbf{X}_t = \{\mathbf{x}_1, ..., \mathbf{x}_{t-1}\}$. The collected relevance score so far is given by column vecror $y_t = \{y_1, ..., y_{t-1}\}$. The algorithm calculates an optimal weight vector $\hat{\mathbf{w}}_\mathbf{t}$ such that,

$$y_t = \mathbf{X}_t.\hat{\mathbf{w}}_\mathbf{t}$$

After calculating $\hat{\mathbf{w}}_\mathbf{t}$, for each new image $x_i \notin \mathbf{X}_t$, which has not been presented to the user so far, is given a relevance score calculated by,

$$y_i = x_i.\hat{\mathbf{w}}_\mathbf{t}$$

Once the relevance score is estimated, the system computes the upper confidence bound of that relevance as $y_i + k\sigma_i$ and the image with the largest upper confidence value is presented next. Here k is a constant that controls the exploration.

PicSOM selects the best performing image feature by examining the adjacency of selected images on TS-SOMs. But sometimes it is really required to combine features. Colour might work well on a query where the user is searching for blue skies, but to search blue sky with cloud formation, colour and texture together might play better. Therefore sometimes it is required to combine various kernels of different features. Pinview does the same by *multiple kernel learning* (MKL) [BLJ04]. If there are n different kernels, $K = \{k_1, k_2, ..., k_n\}$, for any two images i, j, the combined kernel function [AHK+10] would be,

$$k_\eta(i,j) = \sum_{i=1}^{n} \eta_i k_i(i,j)$$

Here $\eta = \{\eta_1, ...\eta_n\}$ are the weights given to kernel functions.

3.3 Inspirations from Pinview

- Pinview uses multiple input modalities like click and eye-movements as means of registering user feedback. We used a draggable slider (a modified click feedback mechanism) for giving feedback. We used the Tobii eye-tracker to capture user's gaze to understand how implicit and explicit feedbacks correlate. Though we did not go for any explicit study on that.
- Pinview modelled the system as a multi-armed-bandit [Git79]. We took the same approach.

3.4 Differences from Pinview

- PinView uses Linrel for recommendation. We used Gaussian Process Bandits (discussed in Section 5) which being a probabilistic approach, attains better performance.
- We created a custom interface for the system. In Pinview, there was no custom interface created purposefully for the tasks.

4 Theoretical background

Here, we discuss the theoretical background necessary for understanding the modelling of content based image retrieval, especially the model we went for. We discuss reinforcement learning, bandit problems, upper confidence bound, regression problems, Bayesian probability model, Gaussian process and application of Gaussian process in bandit problems one by one.

4.1 Reinforcement Learning

Reinforcement learning [SB98] enables a learner to learn to act from experience. It resembles mostly the form of learning human being follow from our childhood. The learner performs an act, observes the reaction of the surrounding and based on that reaction decides the next move or action. It is similar to play a game of chase. The reaction from the surrounding, often termed as "feedback" or "reward", either promotes or demotes the action taken. The possible set of values of the reward

changes based on the nature of the given problem. Sometimes it is a binary value where a 1 denotes a correct action and a 0 otherwise. Sometimes it is a number where a positive number denotes a correct action and a negative number denotes otherwise. The magnitude of the number reflects how much correct or incorrect the action was, i.e. the severity of the action taken.

The settings of reinforcement learning problems differ greatly from one another. One very common set of problems has an environment with a set of states S, at any given time the learning agent can be in any of the given states. As the agent performs an action, it changes the state. The target is to reach a goal state in the end. The agent is allowed to perform an action a from a set of allowed actions A. After performing an action, it receives an immediate reward r. The reward denotes whether the agent is nearing the goal state or going away. The target is to reach the goal state in a minimum number of moves or collecting the maximum cumulative total reward. An example of this setting can be a learner as a robotic car where it has a starting point (state) and a destination (goal state) and some stations to go through (intermediate states). The actions allowed are "move forward", "move backward", "move left" and "move right". The car has a limited amount of fuel to use. After performing an action, it changes the state and it gets the amount of fuel left as a feedback value (we assume one unit of fuel consumption for each move).

Another common class of problems do not allow state change. The agent remains in one state always. It collects rewards for each action and the goal is to collect the maximum amount of reward. One common example of this kind of problems is multi-armed-bandits [Git79].

Reinforcement learning is a form of machine learning. It is different from supervised and unsupervised learning in the sense that it learns immediately from rewards after each action performed. It is widely used in applications of game design, robotics, web advertisement, product recommendation in online stores etc. For example in product recommendation in an online store, the system, acting as an agent, recommends a product. The customer, acting as the environment, can either choose the product or not, thus generating positive or negative reward respectively.

4.1.1 Exploration and Exploitation

A reinforcement learning agent has multiple actions to choose from at each step. Some actions may have been explored and the rest not. For the actions already

explored, the agent knows the reward values. Therefore, the learner also knows which action produced the maximum reward so far and at each subsequent step chooses that action again and again to produce the maximum cumulative total reward. This phenomenon is known as exploitation. If there are unexplored actions which can produce even a greater reward than the highest known one, then certainly the agent will not be producing the highest possible maximum reward. Therefore we see that exploitation is not the best option always. The agent should explore the other actions. On the other hand, it may so happen that there is no unexplored action that produces a higher reward than the known ones. In that case exploration will be a loss and pure exploitation is beneficial. We see that the agent cannot clearly determine whether to go for exploitation or exploration. This situation is known as exploration and exploitation dilemma [SB98].

One solution to this problem is *epsilon-greedy* [SB98] or ϵ *-greedy* method. It provides ϵ as a measure of the amount of exploration allowed. This is a probabilistic quantity. Therefore, the amount of exploitation would be $1-\epsilon$. ϵ is set to achieve the optimally maximum total reward. In bandit problems, the amount of exploration is controlled by upper confidence bound [Agr95], a quantity probabilistically calculated.

4.2 Bandit Problems

Bandit problems [Git79] deal with sequential decision making where each step in the sequence has multiple actions to choose from. Each action is associated with a distribution of numerical rewards. Each action also leads to an observation which is a reward value from the associated distribution. The goal is to maximise the reward over a finite number of trials. The definition of the problem reflects the properties of a reinforcement learning problem as in both cases the learner performs an action and gets a reward in return. It learns the appropriateness of the performed action from the reward received. Therefore we can model bandit problems with reinforcement learning and train a bandit problem solver as a reinforcement learning agent.

The situation can be best described in terms of a slot machine in a casino. The machine has K arms each with a distribution of rewards. A player can play one arm at a time and gets an immediate reward. In this stochastic process the user has to find out which arm is giving the maximum reward. Thus, the user has to explore various arms in succession. This kind of problems are examples of bandit problems. Precisely, as the slot machine has K arms or the problem has K choices in each iteration, the problem is often termed as K - *armed* or *multi-armed* bandit. Once

the user gets an arm with a satisfactorily large reward, he can go on pulling that arm again and again. This is known as Exploitation. But the user might miss another unexplored arm with a bigger reward. Therefore the user might want to explore all the possible arms to find out that particular arm with the biggest reward (here we assume that K is uncountably large). This is known as Exploration. But in finite number of trials the user might not get that desired arm and in the mean time might not exploit the already found highest-reward-arm enough, generating a low overall feedback. Therefore, there is always an issue of optimisation between exploitation and exploration. The learning agent tries to learn an optimal combination of these two.

Bandit problems can be categorised as independent-arm bandits [LR85, ACBF02] and dependent-arm bandits [PCA07]. Independent-arm bandits are the simplest of the two where the success probabilities or the probabilities of getting a positive reward of the arms are independent of each other. On the other hand, dependent-arm bandit problems assume that these rewards influence each other. One example of dependent-arm bandits could be internet ad placement. We consider each separate ad as an arm. In each turn an ad or several ads are picked whose probability of getting selected by the user is the highest. Therefore, the probability of an ad to have had generated a positive reward is strongly dependent on the previous ads displayed and their generated rewards. If a previous ad had generated a higher reward, the probability of a similar ad producing another higher reward again is much higher.

4.3 Bandit algorithms

There are a few techniques that choose which action to choose in each step of a multi-armed-bandit problem. We discuss two of the most popular algorithms which provide an optimal policy to generate maximum cumulative total rewards.

4.3.1 Gittin's Indices

Bandit problems require us to find out a proper sequence of actions over the time steps to achieve the maximum discounted cumulative total reward. Therefore, at each time step, we should have a mechanism to decide which action to take. Gittins and Jones [GJ04] provided an indexing scheme for each available action, known as Gittins index. For each state s_i of an action i, the Gittins index for i is given as,

$$G_i(s_i) = \sup_{\tau>1} \frac{E\Big[\sum_{t=0}^{\tau-1} r_i(s_i(t))\beta^t | s_i(0) = s_i\Big]}{E\Big[\sum_{t=0}^{\tau-1} \beta^t | s_i(0) = s_i\Big]} \tag{1}$$

Here, r_i is the reward function for i, which generates immediate reward $r_i(s_i(t))$ if i is chosen at time t. $s_i(0)$ is the starting state. τ is a predefined stopping step. At each step t, the i with the highest $G_i(s_i)$ is chosen.

4.3.2 Upper Confidence Bound

In bandit problems our goal is to earn the maximum cumulative reward by selecting the highest reward generating action again and again. We do not know beforehand which action produces the highest reward. Therefore, we need a model to estimate the rewards over all possible slots and then select accordingly. We apply a simple method known as *action value* method [SB98] to formulate this problem. Say we have a set of actions, $A = \{a_1, a_2, ..., a_n\}_{n\to\infty}$, and associated rewards $R = \{R_1, R_2, ..., R_n \ | R_i \in \mathcal{R}\}$. Here, each R_i is a random variable denoting a probability distribution over rewards associated with the corresponding action a_i. The actual reward associated with a_i is taken to be the mean of R_i, which has to be estimated. Say after some finite number of trials t, we see that action a_1 has been selected k times, generating rewards $r_1^1, r_1^2, ..., r_1^k$. We estimate the mean reward for a_1, μ_{R_1} as the average of these rewards as $\mu_{R_1} = \frac{\sum_{j=1}^{k} r_i^j}{k}$. We can calculate μ_{R_i} for each a_i. Clearly, we target the a_i having the largest μ_{R_i}. We can also estimate the variance σ_{R_i} for each R_i. At some point we have estimated μ_{R_i} and σ_{R_i}, $\forall R_i$. The upper bound for the possible reward value for an R_i is $\mu_{R_i} + \sigma_{R_i}$, which means, if I take an action a_i, this is the highest possible reward it can generate. Let us denote $\mu_{R_i} + \sigma_{R_i}$ as U_i, the upper bound, which helps us to decide whether to explore or to exploit at any time t. Before taking an action at time t, we know the action a_j has been producing highest reward so far. We can exploit by performing that action. On the other hand we know all the U_i values. If any $U_k > U_j, k \neq j$, we can explore by selecting a_k. This is not random exploration, because we have some surity that we are exploring an action which probably will produce a higher reward, thus we are not in the danger of having a very low cumulative reward in the end. If we find all $U_k <= U_j, k \neq j$, we will certainly exploit a_j. Here, we are using the upper section of confidence interval for each R_i as the *upper confidence bound* or *UCB*. This simple solution for the exploitation-exploration dilemma was given by R. Agrawal in his upper confidence bound algorithm [Agr95].

This solution works quite well in a finite action space where, to estimate all the U_i values we need to perform each and every action at least a few times. For an infinitesimally large action space that will lead to too much random exploration initially. After discussing about gaussian process, we will see how we can apply that here to tackle this problem.

4.4 UCB algorithms for independent armed bandits

UCB algorithms like *UCB1* and *UCB1-tuned* target independent armed bandit problems. They are discussed below in brief.

4.4.1 UCB1

UCB1 [ACBF02] keeps the number of times each arm is played at each time step t, denoted by $n_i(t)$. The algorithm starts with playing each arm once. Say, it time t the ith arm has a mean $\hat{\mu}_i$. The algorithm picks arm j as,

$$j(t) = argmax_{i=1...k}\left(\hat{\mu}_i + \sqrt{\frac{2\ln t}{n_i}}\right) \tag{2}$$

4.4.2 UCB1-Tuned

The authors of UCB1 also proposed a second algorithm, UCB1-Tuned [**?**]. According to the authors, this algorithm performs better than UCB1 in practice. It takes the variance $\hat{\sigma}_i$ of each term into account also besides the mean $\hat{\mu}_i$. The algorithm picks arm j at time t as,

$$j(t) = argmax_{i=1...k}\left(\hat{\mu}_i + \sqrt{\frac{\ln t}{n_i} min\Big(\frac{1}{4}, V_i(n_i)\Big)}\right) \tag{3}$$

where,

$$V_i(t) = \hat{\sigma}^2(t) + \sqrt{\frac{2\ln t}{n_i}} \tag{4}$$

The variance $\hat{\sigma}^2(t)$ is calculated using an empirical sum of squares of reward values.

There are two popular algorithms based on the upper confidence bound technique. We discuss them one by one.

4.5 UCB algorithms for dependent armed bandits

Algorithms like *LinRel* and *LinUCB* target dependent armed bandit problems. They are discussed below.

4.5.1 LinRel

LinRel algorithm [Aue02] provides an alternative approach for obtaining the action that produces the highest reward in each step. It assumes that for any action i, the estimated reward $r_i(t)$ at iteration t is a linear function of all the rewards obtained in the last $t-1$ iterations. Therefore the algorithm tries to learn a weight vector $\mathbf{w}_i(t)$ such that if $\mathbf{r_i}$ represents the reward vector containing all the previous rewards for action i, then the expected reward at step t is calculated as,

$$E\big[r_i(t)\big] = \mathbf{r_i} \cdot \mathbf{w}_i(t)^T \tag{5}$$

Now to estimate $\mathbf{w}_i(t)$, let $\mathbf{x}_i(t)$ be the feature vector of the arm at time t and $\mathbf{X}_i(t)$ be all the previous $t-1$ feature vectors, it can be shown that $\mathbf{w}_i(t)^T$ is calculated as,

$$\mathbf{w}_i(t)^T = \mathbf{x}_i(t)^T \cdot (\mathbf{X}_i(t) \cdot \mathbf{X}_i(t)^T)^{-1} \cdot \mathbf{X}_i(t) \tag{6}$$

This expected reward is taken to be the mean reward. A confidence bound $\sigma_i(t)$ is calculated using the upper bound of $\mathbf{w}_i(t)$. The upper confidence bound ($UCB_i(t)$) [Agr95] (we discuss this in the next section) is generated by adding the positive confidence bound with the mean reward as,

$$UCB_i(t) = E\big[r_i(t)\big] + \sigma_i(t) \tag{7}$$

The action that produces the highest UCB value is considered.

4.5.2 LinUCB

LinUCB [LCLS10] is very similar to LinREL and is mostly used in personalized news article recommendation. It also describes the expected reward of an action as linear combination of state vectors or feature vectors of that action over the previous trials.

Say at time t, action a is in state $\mathbf{x}_{at}$ and it generates reward r_{at}, so the expected value of r_{at} is,

$$E\big[r_{at}|\mathbf{x}_{at}\big] = \mathbf{x}_{at}^T.\theta_a^* \tag{8}$$

$\mathbf{x}_{at}$ is a d dimensional vector. In each step we have a tuple $(\mathbf{X}_a, \mathbf{c}_a)$, consisting of a two dimensional matrix $\mathbf{X}_a$, whose rows correspond to feature vectors of action a for all the previous steps and the click feedbacks for the previous steps which is a vector $\mathbf{c}_a$. Applying ridge regression we calculate θ_a^* as

$$\theta_a^* = (\mathbf{X}_a^T\mathbf{X}_a + \mathbf{I}_d)^{-1}\mathbf{X}_a^T\mathbf{c}_a \tag{9}$$

Here $\mathbf{I}_d$ is the d dimensional identity matrix.

Now we calculate the upper confidence bound value as follows,

$$UCB(a_t) = \mathbf{x}_{at}^T.\theta_a^* + \alpha\sqrt{\mathbf{x}_{at}^T(\mathbf{X}_a^T\mathbf{X}_a + \mathbf{I}_d)^{-1})\mathbf{x}_{at}} \tag{10}$$

The action with highest UCB value is picked up.

5 Gaussian Process Bandits

Gaussian processes are nonparametric regression techniques. Before starting with Gaussian processes, we need to discuss regression problems, bandits, Gaussian distribution, Baye's rule, Bayesian linear regression models including the standard linear model and basis function models both. In the subsequent subsections we will address these topics briefly.

5.1 Regression Problems

In regression problems we have multiple random variables. At least one of those variables are dependent on a subset of the rest. Let us assume that we have to predict rainfall in a city for the coming monsoon based on average summer temperature. We take a dataset consisting of year by year rainfall and average summer temperature recorded over the past few years. We denote the temperature by x and rainfall by y. We assume x is an independent random variable and y is dependent on x. We

take a set $S = \{x_i, y_i | i = 0, 1, ...\}$ for the past years. The goal here is to learn the relation between x and y. We will apply the learned relation on unknown x values to get the corresponding y values. Here relation basically means a mathematical function.

In its simplest form the function could be a linear one like,

$$f(x_i) = \theta_1 x_i + \theta_0 \tag{11}$$

But often in real life scenarios we have more than one independent input variables. Besides temperature we can have amount of CO_2 emission, amount of deforestation (in square kilometres) and so forth. Therefore instead of having a single x_i we have a vector $\mathbf{x}_i = \{x_{i1}, x_{i2}, x_{i3}, ..., x_{im}\}$. We rewrite equation 1 as,

$$f(\mathbf{x}_i) = \sum_{j=1}^{m} \theta_j x_{ij} + \theta_0 \tag{12}$$

It is not possible to match each and every y_i because of the randomness of the data. We try to go as close as possible so that $f(x)$ can represent the pattern of the output. The closeness is measured by *least square* method. The goal is to minimize the distance between y_i and $f(x_i)$ over the entire dataset. Say d_i^2 represents the squared distance between y_i and $f(x_i)$.

$$d_i^2 = [y_i - f(\mathbf{x}_i)]^2 \tag{13}$$

Let d^2 denote the summation of all d_i^2,

$$\begin{aligned} d^2 &= \sum_i [y_i - f(\mathbf{x}_i)]^2 \\ d^2 &= \sum_i [y_i - \sum_{j=1}^{m} \theta_j x_{ij} - \theta_0]^2 \\ d^2 &= \sum_i [y_i - \theta_1 x_{i1} - \theta_2 x_{i2} - \theta_3 x_{i3} - ... - \theta_m x_{im} - \theta_0]^2 \end{aligned} \tag{14}$$

To obtain minimum d^2, we take partial derivative with respect to each θ_i and set those to zero. Therefore we get a set of partial differential equations as follows,

$$\begin{aligned}
\frac{\partial}{\partial\theta_0}(d^2) &= -2\sum_i[y_i - \theta_1 x_{i1} - \theta_2 x_{i2} - \theta_3 x_{i3} - ... - \theta_m x_{im} - \theta_0] = 0 \\
\frac{\partial}{\partial\theta_1}(d^2) &= -2\sum_i[y_i - \theta_1 x_{i1} - \theta_2 x_{i2} - \theta_3 x_{i3} - ... - \theta_m x_{im} - \theta_0]x_{i1} = 0 \\
&\vdots \\
\frac{\partial}{\partial\theta_m}(d^2) &= -2\sum_i[y_i - \theta_1 x_{i1} - \theta_2 x_{i2} - \theta_3 x_{i3} - ... - \theta_m x_{im} - \theta_0]x_{im} = 0 \qquad (15)
\end{aligned}$$

Assuming we have n observed data points, We can write this set of equations as,

$$\begin{aligned}
\theta_0 n + \theta_1\sum_{i=1}^n x_{i1} + \theta_2\sum_{i=1}^n x_{i2} + ... + \theta_m\sum_{i=1}^n x_{im} &= \sum_{i=1}^n y_i \\
\theta_0\sum_{i=1}^n x_{i1} + \theta_1\sum_{i=1}^n x_{i1}^2 + \theta_2\sum_{i=1}^n x_{i1}x_{i2} + ... + \theta_m\sum_{i=1}^n x_{i1}x_{im} &= \sum_{i=1}^n y_i x_1 \\
&\vdots \\
\theta_0\sum_{i=1}^n x_{im} + \theta_1\sum_{i=1}^n x_{i1}x_{im} + \theta_1\sum_{i=1}^n x_{i2}x_{im} + ... + \theta_m\sum_{i=1}^n x_{im}^2 &= \sum_{i=1}^n y_i x_m \qquad (16)
\end{aligned}$$

These set of equations can be written in matrix form,

$$\begin{pmatrix} \sum_{i=1}^n y_i \\ \sum_{i=1}^n y_i x_i \\ \vdots \\ \sum_{i=1}^n y_i x_m \end{pmatrix} = \begin{pmatrix} n & \sum_{i=1}^n x_{i1} & \sum_{i=1}^n x_{i2} \ldots \sum_{i=1}^n x_{im} \\ \sum_{i=1}^n x_{i1} & \sum_{i=1}^n x_{i1}^2 & \sum_{i=1}^n x_{i1}x_{i2} \ldots \sum_{i=1}^n x_{i1}x_{im} \\ \vdots & \vdots & \\ \sum_{i=1}^n x_{im} & \sum_{i=1}^n x_{i1}x_{im} & \sum_{i=1}^n x_{i2}x_{im} \ldots \sum_{i=1}^n x_{im}^2 \end{pmatrix} \begin{pmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_m \end{pmatrix}$$

To simplify the notation, let's denote the left hand side of Eqn. 5.1 as $\mathbf{B}$, the matrix at the right hand side as $\mathbf{X}$ and the vector θ as $\hat{\theta}$. Therefore Eqn. 5.1 can be written as,

$$\begin{aligned}
\mathbf{B} &= \mathbf{X}\hat{\theta} \\
\hat{\theta} &= \mathbf{X^{-1}B} \qquad (17)
\end{aligned}$$

We obtained the optimal $\hat{\theta} = \{\theta_1, \theta_2, \theta_3, ..., \theta_m\}$.

5.2 Gaussian Distribution

A one dimensional random variable x, which follows Gaussian distribution with mean μ and variance σ has a probability density function (PDF),

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\Big[- \frac{(x-\mu)^2}{2\sigma^2}\Big] \tag{18}$$

This is the equation of a bell curve as shown in Figure 1.

Here, μ is the expected value of a the random variable x which is the peak of the bell shaped curve, as shown in Figure 18 and σ is the maximum width of the distribution i.e. the width on both sides of μ in which most of the datapoints occur.

Eqn. 18 can be written as,

$$p(x) \sim \mathcal{N}(\mu, \sigma^2) \tag{19}$$

It means, x comes from a Gaussian distribution with mean μ and variance σ^2.

Sometimes we have a collection of random variables $\mathbf{x} = \{x_1, x_2, x_3, ..., x_m\}, m \in \mathcal{R}$, where each x_i where $i \in m$ follows Gaussian distribution, we write the PDF over $\mathbf{x}$ as,

$$p(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^m |\Sigma|}} exp\Big[- \frac{1}{2}(\mathbf{x}-\mu)^T \Sigma^{-1} (\mathbf{x}-\mu)\Big] \tag{20}$$

Here $\mu = \{\mu_1, \mu_2, ..., \mu_m\}$ is a mean vector where each μ_i represents the mean of x_i. Σ is a $m \times m$ covariance matrix where any $\Sigma[i, j]$ represents the co-variance between x_i and x_j.

Figure 2 shows the plot of a two dimensional Gaussian.

We can write Eqn. 20 as,

$$p(\mathbf{x}) \sim \mathcal{N}(\mu, \Sigma^2) \tag{21}$$

5.3 Bayesian Linear Regression Model

The traditional linear regression method is rigid in terms of learning the parameters. It gives fixed values for parameters for one observed datapoint. Therefore, for every

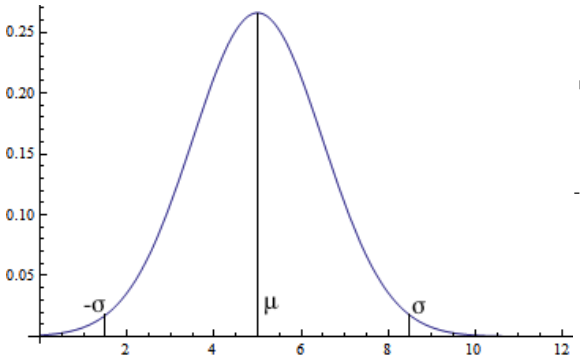


Figure 1: Univariate Gaussian

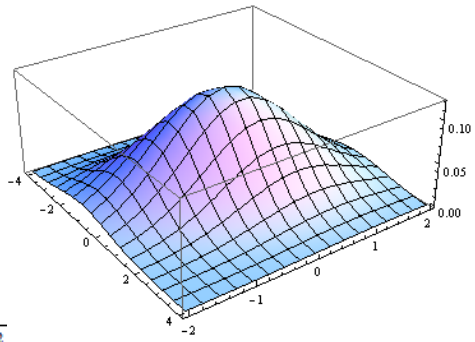


Figure 2: Multivariate Gaussian

new observed datapoint, the parameters are most likely to change. As always we only know numeric values of the parameters, it is hard to predict how much they can change for the next observed data point. It would be effective to learn a probability distribution over the parameters rather than fixed values. Having a distribution over parameters gives us the space where the parameters can move, which is convenient to understand a stochastic process. Bayesian probability model starts with a prior distribution over parameters and changes the distribution based on observations.

Bayesian probability model is built after *Bayes'* rule. It allows us to start with a prior belief on the data, which is an initial probability distribution associated with the data. Experiments generate evidence, which is used to change the prior belief, i.e. the initial distribution. The new distribution we get after incorporating the evidences is called posterior. The formulation is given below.

5.4 Bayes' Rule

We start with the expression of Bayes' rule. Say we have two random variables, x and y, where x is an independent variable but y depends on x. For the sake of regression problems we assume both x and y as continuous variables. The probability of x is given by $p(x)$, the joint probability of x and y is given by $p(y, x)$. We can break $p(y, x)$ in $p(y|x)p(x)$ or $p(x|y)p(y)$ (Here $p(x|y)$ cannot be written as $p(x)$ if x is conditionally dependent on y). Therefore,

$$\begin{aligned} p(y, x) = p(y|x)p(x) = p(x|y)p(y) \\ p(y|x) = \frac{p(x|y)p(y)}{p(x)} \end{aligned} \tag{22}$$

The term $p(x)$ is marginalized over all possible values of y, so we can write $p(x)$ as,

$$p(x) = \int_y p(x|y)p(y)dy \tag{23}$$

Combining Eqn. 17 and Eqn. 18,

$$p(y|x) = \frac{p(x|y)p(y)}{\int_y p(x|y)p(y)dy} \tag{24}$$

5.5 The Standard Linear Model

We start with the assumption that y_i differs from $f(\mathbf{x}_i)$ because of noise. In our discussion of regression problems in 5.1 we had a constant parameter θ_0. In this section we model that same θ_0 as the noise term which, given by ϵ_i, follows Gaussian i.i.d. $\mathcal{N}(0, \sigma^2)$. Therefore the i$^{\text{th}}$ observation can be written as,

$$\begin{aligned} y_i &= f(\mathbf{x}_i) + \epsilon_i \\ y_i &= \boldsymbol{\theta}^T \mathbf{x}_i + \epsilon_i \qquad \text{where } \boldsymbol{\theta}, \mathbf{x}_i \in \mathcal{R}^m \end{aligned} \tag{25}$$

The PDF associated with ϵ_i is,

$$p(\epsilon_i) = \frac{1}{\sqrt{2\pi}\sigma} exp\Big[- \frac{\epsilon_i^2}{2\sigma^2} \Big] \tag{26}$$

Replacing ϵ_i by $(y_i - \boldsymbol{\theta}^T \mathbf{x}_i)$ from Eqn. 25 ,

$$\begin{aligned} p(y_i - \boldsymbol{\theta}^T \mathbf{x}_i) &= \frac{1}{\sqrt{2\pi}\sigma} exp\Big[- \frac{(y_i - \boldsymbol{\theta}^T \mathbf{x}_i)^2}{2\sigma^2} \Big] \\ p(y_i | \mathbf{x}_i, \boldsymbol{\theta}) &= \frac{1}{\sqrt{2\pi}\sigma} exp\Big[- \frac{(y_i - \boldsymbol{\theta}^T \mathbf{x}_i)^2}{2\sigma^2} \Big] \end{aligned} \tag{27}$$

Thus $y_i | \boldsymbol{\theta}^T, \mathbf{x}_i$ is Gaussian distributed as $y_i | \boldsymbol{\theta}^T \mathbf{x}_i \sim \mathcal{N}(\boldsymbol{\theta}^T \mathbf{x}_i, \sigma^2)$

Over the entire training dataset $S = \{\mathbf{x}_i, y_i\}_{i=1}^n$, as y_i's are independent and identically distributed, we can write,

$$
\begin{aligned}
p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) &= \prod_{i=1}^{n} p(y_i|\boldsymbol{\theta}^T \mathbf{x}_i) \\
&= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} exp\Big[- \frac{(y_i - \boldsymbol{\theta}^T \mathbf{x}_i)^2}{2\sigma^2} \Big]
\end{aligned}
\tag{28}
$$

Therefore $p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta})$ is Gaussian distributed as $\mathbf{y}|\mathbf{X}, \boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\theta}^T \mathbf{X}, \sigma^2 \mathbf{I})$

Here $\mathbf{y}_i = \{y_1, y_2, ...y_n\}$ denotes the output vector and $X = \{\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_n}\}$ is the matrix comprising of all the input vectors, sometimes denoted as the design matrix.

Eqn 28 gives us the likelihood function w.r.t. $\boldsymbol{\theta}$. We could take a maximum likelihood estimate here and calculate the optimal $\boldsymbol{\theta}$. But instead of point estimation of individual parameters, we want to calculate the distribution over $\boldsymbol{\theta}$ based on the observed dataset S. We assume a prior over $\boldsymbol{\theta}$ as,

$$
\boldsymbol{\theta} \sim \mathcal{N}(\mathbf{0}, \Sigma_p) \tag{29}
$$

The posterior distribution over $\boldsymbol{\theta}$ given S would be,

$$
\begin{aligned}
p(\boldsymbol{\theta}|\mathbf{y}, \mathbf{X}) &= \frac{p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) p(\boldsymbol{\theta})}{p(\mathbf{y}|\mathbf{X})} \\
p(\boldsymbol{\theta}|\mathbf{y}, \mathbf{X}) &\propto p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) p(\boldsymbol{\theta})
\end{aligned}
\tag{30}
$$

We omit the marginal likelihood term, $p(\mathbf{y}|\mathbf{X}) = \int p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) p(\boldsymbol{\theta}) d\boldsymbol{\theta}$, being a constant and this posterior over $\boldsymbol{\theta}$ can be shown to follow Gaussian as,

$$
\boldsymbol{\theta}|\mathbf{y}, \mathbf{X} \sim \mathcal{N}(\frac{1}{\sigma^2} A^{-1} \mathbf{X}\mathbf{y}, A^{-1}) \tag{31}
$$

Here A is given by $\sigma^{-2}\mathbf{X}\mathbf{X}^\mathbf{T} + \mathbf{\Sigma_p^{-1}}$

Now for any next unobserved data point $S_* = \{y_*, x_*\}$, we can calculate the posterior distribution over y_*, given x_* and S, as below,

$$
\begin{aligned}
p(y_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) &= \int p(y_*|\mathbf{x}_*, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathbf{X}, \mathbf{y}) d\boldsymbol{\theta} \\
y_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y} &\sim \mathcal{N}(\frac{1}{\sigma^2} \mathbf{x}_*^T A^{-1} \mathbf{X}\mathbf{y}, \mathbf{x}_*^T A^{-1} \mathbf{x}_*)
\end{aligned}
\tag{32}
$$

The problem with this model is that it suffers from input dimensionality issues. If the input is of one or very few dimensions say $\mathbf{x}_i \in \mathcal{R}^2$, then the model might not be able to fit very complex non-linear functions, also if the input has a large dimension, say $\mathbf{x}_i \in \mathcal{R}^{200}$ then optimising that many parameters is cumbersome. Therefore we need a mechanism that transfers the input into another feature space of favourable dimensions.

5.6 The Basis Function Model

There are special functions called *basis functions*, which are used to transform the input vectors into another feature space of different dimension. We can define a collection of basis functions as below,

$$\Phi(.) = \{\phi_1(.), \phi_2(.), ..., \phi_n(.)\} \mid n \in \mathcal{R} \tag{33}$$

Say we have an input vector $\mathbf{x}_i \in \mathcal{R}^d$ as $\mathbf{x}_i = \{x_{i1}, x_{i2}, ..., x_{id}\}$, which changes when passed through the basis function. Each function from the basis function set is applied to each component of the input vector to generate d numbers which are then passed to another function to generate a single number. This way we generate $\Phi(\mathbf{x}_i) \in \mathcal{R}^n$

$$\begin{aligned}\Phi(\mathbf{x}_i) = \Bigg\{ & f\Big(\phi_1(x_{i1}), \phi_1(x_{i2}), ..., \phi_1(x_{id})\Big), \\ & f\Big(\phi_2(x_{i1}), \phi_2(x_{i2}), ..., \phi_2(x_{id})\Big), \\ & . \\ & . \\ & f\Big(\phi_n(x_{i1}), \phi_n(x_{i2}), ..., \phi_n(x_{id})\Big) \Bigg\}\end{aligned} \tag{34}$$

The standard linear model changes into,

$$y_i = \phi(\mathbf{x}_i)^T \boldsymbol{\theta} + \epsilon_i \tag{35}$$

The predictive distribution becomes,

$$y_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y} \sim \mathcal{N}(\boldsymbol{\phi}_*^T \Sigma_p \boldsymbol{\Phi}(\mathcal{K} + \sigma^2 \mathcal{I})^{-1}\mathbf{y}, \boldsymbol{\phi}_*^T \Sigma_p \boldsymbol{\phi}_* - \boldsymbol{\phi}_*^T \Sigma_p \boldsymbol{\Phi}(\mathcal{K} + \sigma^2 \mathcal{I})^{-1} \boldsymbol{\Phi}^T \Sigma_p \boldsymbol{\phi}_*) \tag{36}$$

Here $\mathcal{K} = \boldsymbol{\Phi}^T \Sigma_p \boldsymbol{\Phi}$ and we have used some shorthand. They are $\boldsymbol{\phi}_* = \boldsymbol{\phi}_*(\mathbf{x}_*)$ and $\boldsymbol{\Phi} = \boldsymbol{\Phi}(\mathbf{X})$ [RW06].

We see that some identical terms appear several times in Eqn. 36 as $\boldsymbol{\Phi}^T \Sigma_p \boldsymbol{\Phi}$, $\boldsymbol{\phi}_*^T \Sigma_p \boldsymbol{\Phi}$ and $\boldsymbol{\phi}_*^T \Sigma_p \boldsymbol{\phi}_*$. They must have been in the form $\boldsymbol{\phi}(\mathbf{x})^T \Sigma_p \boldsymbol{\phi}(\mathbf{x}_*)$ where $\mathbf{x}$ and $\mathbf{x}_*$ are training and test sets. This expression has a functional form called *covariance function or kernel function* as below,

$$k(\mathbf{x}, \mathbf{x}_*) = \boldsymbol{\phi}(\mathbf{x})^T \Sigma_p \boldsymbol{\phi}(\mathbf{x}_*) \tag{37}$$

This kernel has extreme importance in practical applications. We see from Eqn. 36 that the entire expression is made up of kernel functions. We can pre-calculate the entire kernel for all possible $\mathbf{x}$ and $\mathbf{x}_*$ and then we do not need the individual input feature vectors anymore. Here, we are achieving some kind of dimensionality reduction.

Here, the problem we need to deal with is to choose the appropriate amount of basis functions, their location centres, their width etc. As the kernel function is built with the inner product of input vectors elevated to the basis feature space, these parameters play an important role and if not chosen carefully they might not reflect the proper covariance between the given input. The most obvious drawback is the correlation between the inputs falls of quickly when they move far away from the basis function centres. We need an infinite amount of basis functions to reflect the true covariance between the inputs. This discussion is out of scope for this text. In the next section, we will see how we can build an efficient kernel function that does not have to deal with the setting of these basis function parameters, rather it directly relates the given inputs and instead of trying to get the distribution over parameters, it directly calculates the posterior over the function $f(x)$. In other words, instead of calculating posterior over $(X)^T\boldsymbol{\theta}$ or $\boldsymbol{\phi}(X)^T\boldsymbol{\theta}$, we directly calculate the posterior over $f(x)$. It frees us from dealing with $\boldsymbol{\theta}$ or $\boldsymbol{\phi}$. Here, comes the idea of *Gaussian Process.*

5.7 Gaussian Process

Gaussian process [RW06] is a mathematical framework that does not try to learn a finite set of parameters. It models the output as a function of covariances between the input data points. It tries to learn the function that gives that output over any input. In other words it tries to learn the distribution over the functions. As it expresses the function in terms of input covariances, it does not care about dimensions of input once covariances are calculated, thus it truly represents a non parametric multivariate Gaussian framework. This representation is known as Gaussian process.

In our example of rainfall prediction, we consider $x_i \in \mathcal{R}$ to be summer temperature. Here the independent random variable x_i is single dimensional. If we consider spring temperature also for that particular year then $\mathbf{x}_i = \{x_i, x_j\} \in \mathcal{R}^2$ becomes two dimensional. No matter how many different temperatures we consider, $\mathbf{x}_i \in \mathcal{R}^d$ remains a finite dimensional vector. Gaussian process helps in building a framework that actually allows us to fit any number of dimensions in. We do not have to fix d beforehand. Therefore we can make $\mathbf{x}_i$ represent various different temperatures at different months, weeks, days, hours and so forth.

Here, we show set of observations $\mathcal{X}$ on X- axis i.e. $\mathbf{x}_i$ values. As we can fit infinite number of points on X- axis, we can represent infinite amount of observations on it. If $\mathcal{X}$ is a set of three observations, we show three points on X-axis.

To represent the regression problem here, we have,

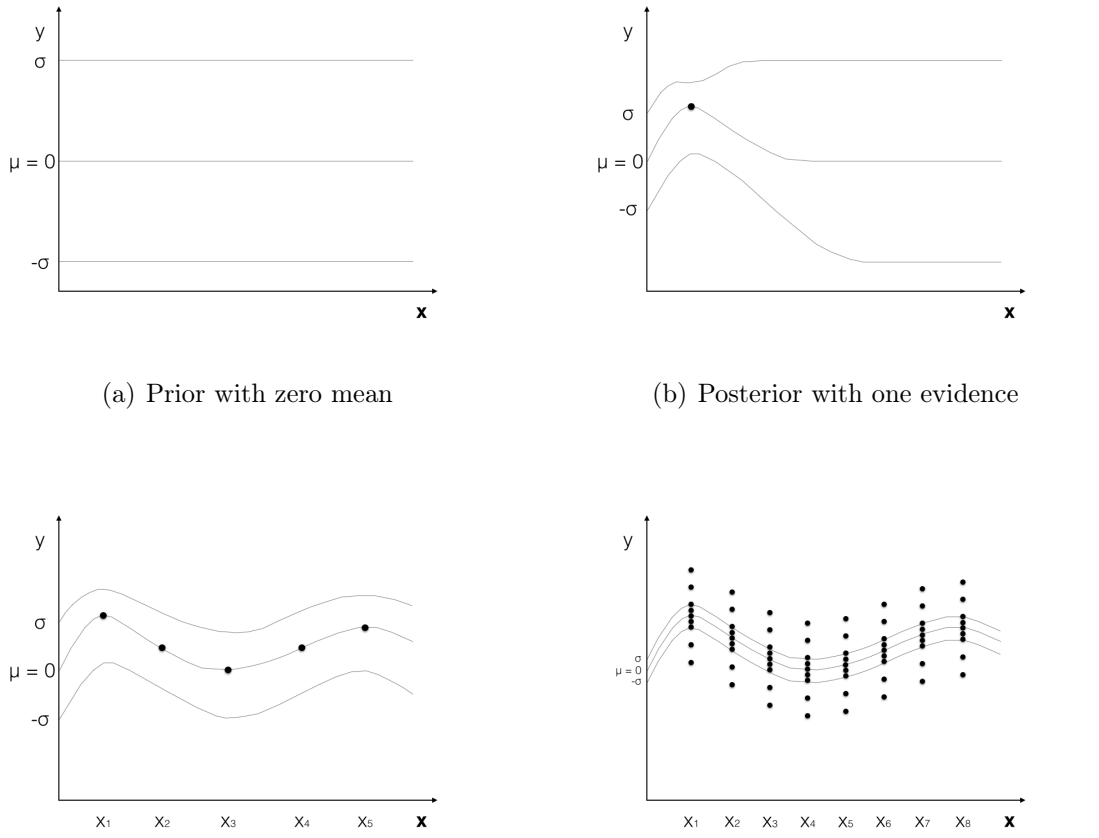
$$y_i = f(\mathbf{x}_i) + \epsilon_i \tag{38}$$

$\epsilon_i \sim (0, \sigma^2))$ is the standard Gaussian noise.

Without considering any parameter set, we try to learn the function f. We treat f as a dependent random variable on $\mathbf{x}_i$. A continuous function can represent an infinite number of points on it. Therefore, f can fit an infinite number of $\mathbf{x}_i$, where $i = 0, 1, ...\infty$. Thus, Gaussian process gives us an infinite dimensional multivariate Gaussian distribution or a distribution over functions.

Say, $\mathcal{F} = \{f_1, f_2, ..., f_n, f_{n+1}, ...\infty\}$ is a set of functions that can operate over $\mathbf{x}_i$. Any function f over an x can take any form like $f(x) = (x + 2)$ or $f(x) = x^2$ or $f(x) = \exp(\sin(x)^5)$, as long as $f(x) \in \mathcal{R}$. From this infinite set $\mathcal{F}$, we need to find the proper f that truly represents y i.e. truly fits the data.

To differentiate a function from a regular random variable we denote a function f



(a) Prior with zero mean

(b) Posterior with one evidence

(c) More evidence

(d) Learned function with entire training set

Figure 3: Visual representation of Gaussian process.

as $f(.)$ from now on. We start with a prior distribution over $\mathcal{F}$, given by,

$$f(.) \sim (m(.), k(.,.)) \tag{39}$$

Figure 3 presents a visual representation of Gaussian process. The prior is shown in 3(a) with zero mean. Before getting evidence, the prior distribution has uniform confidence interval around the mean for the entire input space. After getting one evidence point the space around that point shrank as shown in 3(b). We exclude all the functions that do not pass through our evidence point. The mean function has to pass through that point, therefore it is shifted and the confidence interval also narrowed down accordingly. 3(c) shows the same effect for more evidence points. 3(d) finally shows the shape of the posterior function when we have enough evidence.

As we are operating on a distribution over functions, the mean $m(.)$ and covariance $k(.,.)$ are also functions. The domain of $f(.)$ is a vector $\mathbf{x}_i \in \mathcal{R}^d$ where $d \to \infty$. $m(.)$ is the mean function which gives the mean of any input $\mathbf{x}_k \in \mathcal{X}$ as $m(\mathbf{x}_k) = \mu_{\mathbf{x}_k}$ and it generates a mean vector over $\mathcal{X}$ as,

$$m(\mathcal{X}) = \{m(\mathbf{x}_k) \mid k = 0, 1, 2, ...\} = \{\mu_{\mathbf{x}_k} \mid k = 0, 1, 2, ...\}$$

$k(.,.)$ takes two random vectors, $\mathbf{x}_k$ and $\mathbf{x}_l$ from $\mathcal{X}$ and gives the covariance between them as,

$$k(\mathbf{x}_k, \mathbf{x}_l) = Covariance(\mathbf{x}_k, \mathbf{x}_l)$$

$k(.,.)$ is also called the Kernel function as discussed before.

To represent the multivariate distribution over functions operating over any finite subset $\mathcal{X}_m \in \mathcal{X}$ we can write in matrix form as,

$$\begin{pmatrix} f(\mathbf{x}_1) \\ f(\mathbf{x}_2) \\ \vdots \\ f(\mathbf{x}_m) \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} m(\mathbf{x}_1) \\ m(\mathbf{x}_2) \\ \vdots \\ m(\mathbf{x}_m) \end{pmatrix}, \begin{pmatrix} k(\mathbf{x}_1, \mathbf{x}_1)k(\mathbf{x}_1, \mathbf{x}_2)\dots k(\mathbf{x}_1, \mathbf{x}_m) \\ k(\mathbf{x}_2, \mathbf{x}_1)k(\mathbf{x}_2, \mathbf{x}_2)\dots k(\mathbf{x}_2, \mathbf{x}_m) \\ \vdots \\ k(\mathbf{x}_m, \mathbf{x}_1)k(\mathbf{x}_m, \mathbf{x}_2)\dots k(\mathbf{x}_m, \mathbf{x}_m) \end{pmatrix}\right)$$

Here, the left hand side represents the output vector, the right hand side represents the multivariate Gaussian distribution over functions with a mean vector and a covariance matrix. Which can be written as,

$$f(\mathcal{X}_m) \sim \mathcal{N}(m(\mathcal{X}_m), K(\mathcal{X}_m)) \tag{40}$$

We see how easy Gaussian process is to customize, any time we can extract a finite subset out of the infinite distribution.

To simplify the calculation, we start with a zero mean prior as,

$$f(.) \sim \mathcal{N}(0, k(.,.)) \tag{41}$$

Combining Eqn. 40 and Eqn. 41 we get,

$$f(\mathcal{X}_m) \sim \mathcal{N}(0, K(\mathcal{X}_m)) \tag{42}$$

We take $\mathcal{X}_m$ as training set, so we have observed $\mathbf{y}_m$. Say we have another finite subset $\mathcal{X}_m^*$, which is the test set. We need to predict outputs $\mathbf{y}_m^*$ for $\mathcal{X}_m^*$. Both the subsets came from the same infinite Gaussian distribution. So for $\mathcal{X}_m^*$ we write,

$$f(\mathcal{X}_m^*) \sim \mathcal{N}(0, K(\mathcal{X}_m^*)) \tag{43}$$

We combine Eqn. 42 and Eqn. 43 together as,

$$\begin{pmatrix} f(\mathcal{X}_m) \\ f(\mathcal{X}_m^*) \end{pmatrix} \sim \mathcal{N}\left(0, \begin{pmatrix} K(\mathcal{X}_m, \mathcal{X}_m) & K(\mathcal{X}_m, \mathcal{X}_m^*) \\ K(\mathcal{X}_m^*, \mathcal{X}_m) & K(\mathcal{X}_m^*, \mathcal{X}_m^*) \end{pmatrix}\right) \tag{44}$$

We also have noise vectors $\varepsilon \in \mathcal{R}^m$ and $\varepsilon^* \in \mathcal{R}^m$. We get the distribution over $\mathbf{y}$ by adding noise to the distribution over $f(.)$. Addition of multiple Gaussians remains Gaussian, so we can write,

$$\begin{pmatrix} \mathbf{y}_m \\ \mathbf{y}_m^* \end{pmatrix} = \begin{pmatrix} f(\mathcal{X}_m) \\ f(\mathcal{X}_m^*) \end{pmatrix} + \begin{pmatrix} \varepsilon \\ \varepsilon^* \end{pmatrix} \sim \mathcal{N}\left(0, \begin{pmatrix} K(\mathcal{X}_m, \mathcal{X}_m) + \sigma^2\mathbf{I} & K(\mathcal{X}_m, \mathcal{X}_m^*) \\ K(\mathcal{X}_m^*, \mathcal{X}_m) & K(\mathcal{X}_m^*, \mathcal{X}_m^*) + \sigma^2\mathbf{I} \end{pmatrix}\right) \tag{45}$$

Using Bayesian posterior calculation it can be shown that,

$$\mathbf{y_m}^* | \mathbf{y_m}, \mathcal{X}_m, \mathcal{X}_m^* \sim (\mu^*, \Sigma^*) \tag{46}$$

This is the function represented by Figure 4.

where,

$$\mu^* = K(\mathcal{X}_m^*, \mathcal{X}_m)(K(\mathcal{X}_m, \mathcal{X}_m) + \sigma^2\mathbf{I})^{-1}\mathbf{y_m} \tag{47}$$

and,

$$\Sigma^* = K(\mathcal{X}_m^*, \mathcal{X}_m^*) + \sigma^2\mathbf{I} - K(\mathcal{X}_m^*, \mathcal{X}_m)(K(\mathcal{X}_m, \mathcal{X}_m) + \sigma^2\mathbf{I})^{-1}K(\mathcal{X}_m, \mathcal{X}_m^*) \tag{48}$$

5.8 Applying Gaussian Process to Bandit Problems

We ended 4.3.2 by mentioning that upper confidence bound works well in finite action spaces. Gaussian process helps us to determine upper confidence bound in

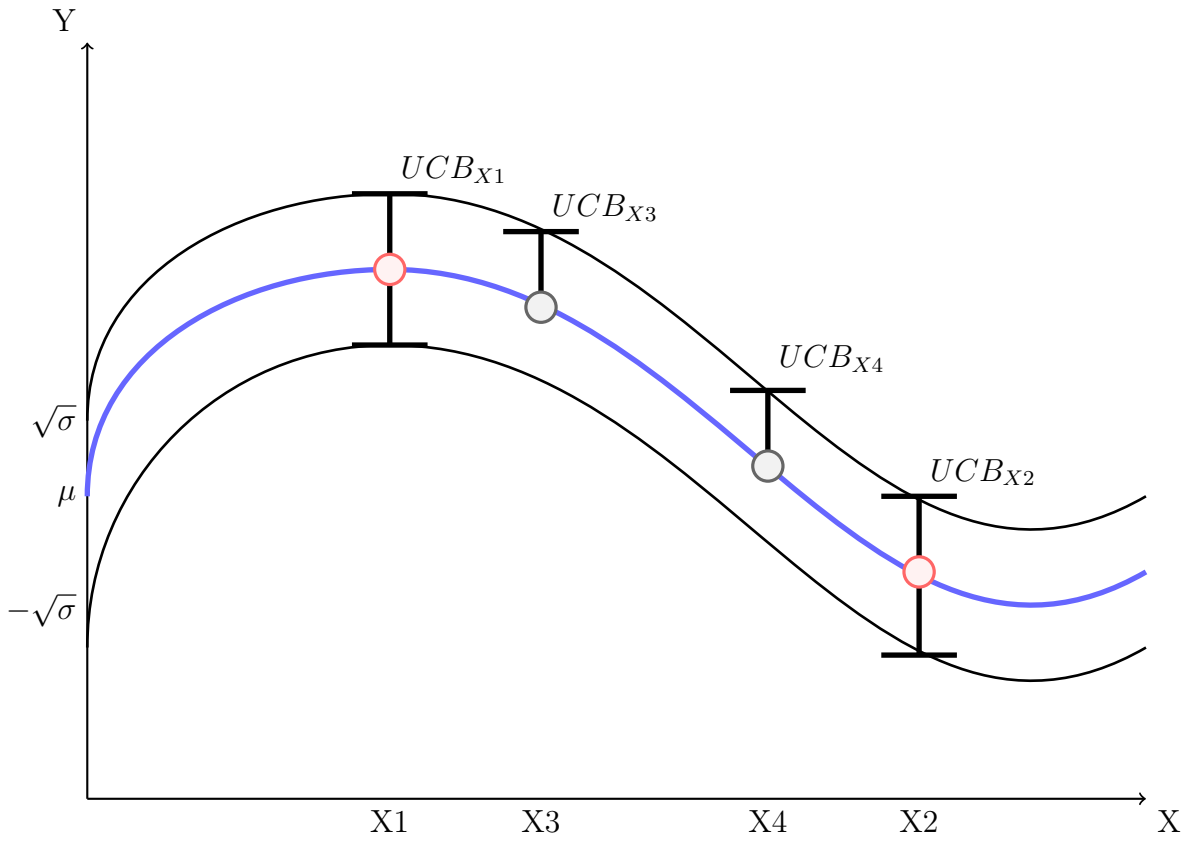


Figure 4: Gaussian process upper confidence bound

infinitely large action spaces, where estimating mean for each action distribution is practically impossible. We represent the actions A as our input space as before and we want to learn a function that gives mean rewards μ_{a_i} for each a_i. Theoretically the more a_is we explore, the closer we get to the actual reward function. But in real life experiments it is not always possible to try all or nearly all the actions from a very large action space.

By carefully observing how the posterior of the reward function changes given the observations so far and then choosing the next action, we can nearly estimate the actual reward function very quickly. This requires some optimisation over selection of actions which is beyond the scope of this discussion. Also the more we select each action, by the *law of large numbers*, the closer we go to the actual mean reward associated with that action. Figure 4 shows the upper confidence bound as the positive standard deviation (square root of variance) added to mean. UCB_{X1}, UCB_{X2}, UCB_{X3} and UCB_{X4} represent the upper confidence bounds of data points

$X1$, $X2$, $X3$ and $X4$ respectively.

Next we will see how we modelled our system based on these discussion.

6 System description

Our system has a collection of 25000 images. The system [HTPG14] shows a random set of a few images to the user and collects relevance feedback on each shown image. If the user does not give feedback to any of the shown images, the system assumes zero feedback on all of them and proceeds accordingly. The indices of the shown images along with the collected relevance feedbacks together form the collected evidence. This evidence is then fed to the Gaussian process framework. For the rest of the images which are not shown so far, the Gaussian process calculates the mean and variance and then the UCB value as discussed before. The image with the largest UCB value is picked by the system.

Bandit algorithms are designed to make one prediction at a time and each prediction takes the immediate reward received by the previous prediction into account. To select a set of images at each iteration, we needed to run the algorithm a few times. But, as before showing an image to the user, we do not have the relevance score for it ready, we use pseudo feedback to pick the next image. This pseudo feedback is nothing but the posterior mean for the corresponding image. Once we select the desired number of images, we show them to the user, receive relevance feedback on them, replace the pseudo feedbacks with the actual feedbacks given by our user [Kon13]. This way we nullify the long term effect of pseudo feedbacks in our system.

We run Gaussian process as many times as the number of images to be shown per iteration to get the next set of images.

We have discussed how our system uses relevance feedback from users as immediate rewards in each step to show more and more relevant images to the users in forthcoming iterations. But if it exploits too much, it could get stuck in a context trap. We needed to have a proper balance in the amount of exploration and exploitation to avoid the context trap [GRK$^+$13, GST10, SF12]. Faceted search [YSLH03, VGJ08] tried to avoid context trap using global image features but as global features are huge in number, the user had to traverse a vast area of the information space [KF06] leading to a cognitively demanding task. Also, initially when we have a small amount of evidence, Gaussian process can produce a very wide variance for some images far

away from shown images as shown in Figure 3(b). This can lead to a high UCB value for not so relevant images. This way the system will explore too much unnecessarily. To tackle this problem we introduce a term *exploration rate* which is nothing but a multiplying factor (< 1) that reduces the effective variance.

6.1 Offline step

We needed to process the images before starting the actual experiments. Therefore we broke the operation down into two parts as described below. These steps are performed only once before the actual experiment starts.

6.1.1 Feature extraction

We used MIRFLICKR 25000 [HL08] image set for the experiment. We consider only colour feature for this study. We represent each image as a 512 dimensional vector [Kon13]. We used *openCV* library to extract "rgb histogram" of each of the images and created an array of the histogram bins. Then, we normalised this array from 0 to 1. This array is the feature vector of the image concerned.

6.1.2 The Kernel

We use "Cityblock" or "Manhattan" distance to calculate pairwise distance between two images, to feature vectors precisely. Manhattan distance of two vectors is nothing but the sum of absolute values of the element wise distances of the vectors given. If $\mathbf{x}$ and $\mathbf{y}$ are two vectors of equal length then L, their Manhattan distance, would be $d_{Manhattan}(\mathbf{x}, \mathbf{y}) = \sum_{l}^{L} |x_l - y_l|$, where x_l and y_l are lth values of the two vectors, respectively. The distance is normalized by the number of dimensions i.e. 512. Therefore the number of dimensions are used as the length scale hyperparameter for the kernel. For any two images x_i and x_j, the kernel function is,

$$K(\mathbf{x}_i, \mathbf{x}_j) = \sum_{l=1}^{512} \frac{(|x_i^l - x_j^l|)}{512} \tag{49}$$

Here x_i^l and x_j^l represent the lth dimensional value in the corresponding feature vectors of images x_i and x_j respectively.

We calculate all the pairwise distances among the images and build a (25000×25000) distance matrix. This matrix serves as the entire co-variance matrix.

6.1.3 Clustering images

As we have a huge number of images, calculating UCB values for nearly all of them would be severely time consuming in each iteration. We can avoid doing so for very distant images. We apply K-means [HWd] clustering algorithm to cluster images as mentioned by [KG]. Each cluster centroid is the average of all the feature vectors allocated to it. To select an image, we first calculate the UCB values for the cluster centroids and select the nearest one. Then we calculate UCB values for images only in that cluster and find out the nearest image.

6.2 The online step

The user is presented with a set of random images. He finds one or more suitable images inside that set (there might not be one, we will discuss that later) and provides relevance feedback as shown in Figure 5. The system collects the shown image set with the given relevance feedback and sends the indices of the shown images and the set of feedbacks along with the cluster centroids to the Gaussian process. This is the first run of the Gaussian process. The process returns the nearest cluster to the shown images. Now the system sends the images assigned to the selected cluster along with the same set of shown images with their received feedback again into the Gaussian process as the second run. The process returns the nearest image to show to the user. This two phase procedure is run with proper inclusion of pseudo feedback discussed earlier to come up with the next set of suitable images until the user gets the desired image.

Say, till t iterations, we showed $X_t = \{x_1, x_2, ..., x_t\}$ images and received relevance feedback $Y_t = \{y_1, y_2, ..., y_t\}$.

According to Eqn. 47, for the rest of the images, the posterior mean function would be,

$$\mu(x) = K(x, X_t)[K(X_t, X_t) + \sigma^2_{noise} I]^{-1} Y_t \tag{50}$$

and, according to Eqn. 48, the posterior co-variance function would be,

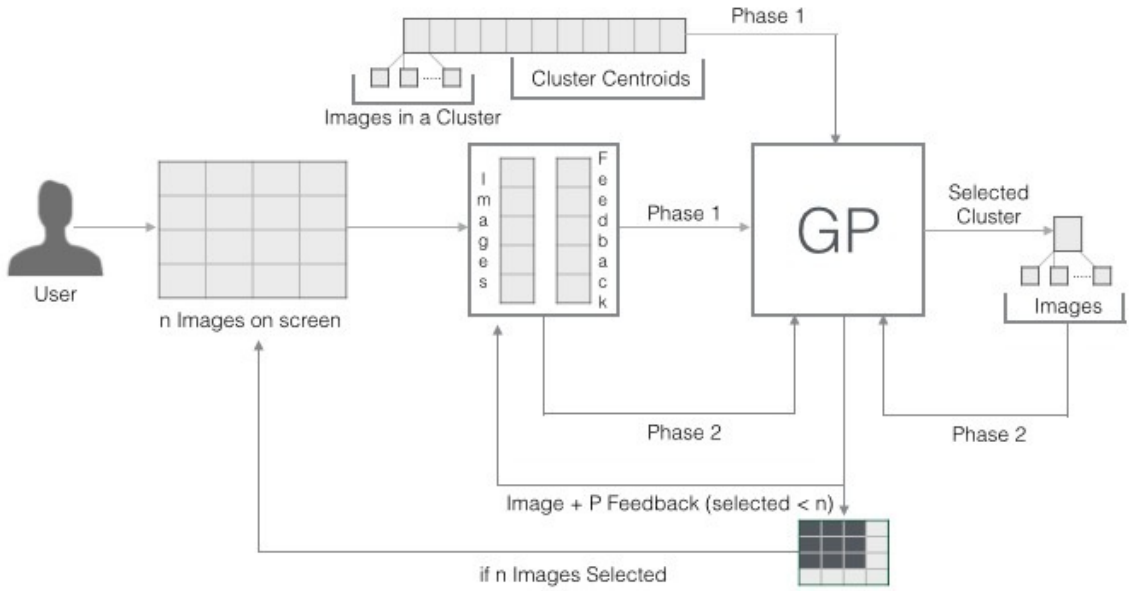


Figure 5: System diagram with hierarchical GP

$$\sigma^2(x) = [K(x,x) + \sigma^2_{noise} I] - K(x, X_t)[K(X_t, X_t) + \sigma^2_{noise} I]^{-1} K(x, X_t)^T \qquad (51)$$

At $t+1$ iteration, image x_{t+1} would be selected, satisfying,

$$x_{t+1} = argmax_x(\mu(x) + \beta\sqrt{\sigma^2(x)}) \qquad (52)$$

We saw in our discussion of Gaussian process that after observing a significant amount of datapoints, the posterior variance becomes very small. Therefore, the UCB value merely reflects the posterior mean, leading to an almost zero exploration. To keep the exploration alive, β is introduced, which is a fraction that grows with discrete time or iterations. At any iteration t, $\beta = \sqrt{\log t}$.

6.3 The interface

Our objective was to build a useful and easy to use interface which supports query less exploratory image search [CMM+00, LKLO00, SF12] while keeping the cognitive load of the user as low as possible. We went for a simple and clean interface, intuitive to understand, supporting visual guidance and feedback and with minimum textual instructions on screen. The main idea behind the design of the interface was to make it support query less exploratory image search while accommodating feedback

given by the users. The user navigates through images and gives ratings to images indicating how close they are to what the user is searching for. According to Cox et. al. [CMM+00] this type of interfaces are easy to use. The interface shows images in a grid like structure as shown in Figure 6. The user has to drag a sliding bar shown on-mouse-over each image to provide the feedback. The user can give feedback to any number of images and then click 'Next' at the top right corner. The display is refreshed with the next set of images.

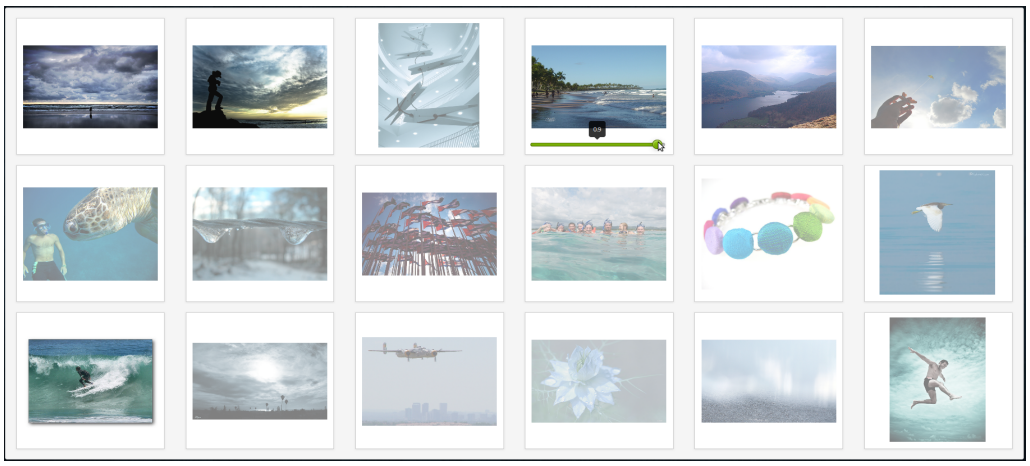


Figure 6: The interface

7 Experiments

We assign search tasks to users, say to find a "red rose". Each experiment consists of a single task. The user is asked to find a suitable image for the given task. Based on the nature of the task we sometime show sample images but not always. The aim of these experiments was to study 1) how good exploration-exploitation strategy performs compared to mere exploitative method, 2) applicability of exploration based search in different kinds of tasks with varying level of abstractness. Tasks are organised into categories as described below.

7.1 Tasks and Task-Categories

We could think of three different task types in terms of abstractness of the given task. Some tasks are very particular in nature. While having such a task, the user has to find that very image described by the task. No other similar image can

terminate the search. For example if the user has to find the Eiffel Tower or the Statue of Liberty etc. Some tasks are a bit generic than the ones mentioned before. For example if the user has to find out an image of sunset or seashore. They are not specific objects but are combination of objects in different colours and shapes with some similarities. There are a third kind of tasks which are very abstract in nature. Say the user has to find an image of happiness. We cannot attach any specific object to it. It is a state of mind and a matter of perception. For a user an image of a happy face can be happiness, for another user a rainbow can be happiness or children in a playground can be happiness. We followed Pichunter [CMM+00] to design three different categories of tasks for users as follows,

1. Target search - Look for a specific image
2. Category search - Look for a category of an image
3. Open ended search - Find a story, i.e. a set of images that gives a theme for an essay or an article

Target search was the most restrictive search, requiring the user to find an image say of a blue car.

Category search was a bit broader, say to find a cat.

Open ended search was the most broad one where the user was given no specific image to search for but a topic or theme like a visit to Paris, out of which several images could be listed for an article on the same.

Each of these searches had three separate sub tasks as follows,

for Target search

1. Red rose
2. City by night
3. Tall building

for Category search

1. Fruit
2. Sunset

3. Seashore

for Open ended search

1. A trip to Africa
2. Architecture
3. Gardening

Therefore, we had nine separate search tasks altogether. A user had to perform all these tasks but the order was different for each user. We took a set of the nine tasks, prepared twenty different random permutations out of them and assigned one to each user.

7.2 Participants and task allocation

We recruited twenty postgraduate students from university as participants. Before the actual experiments, we provided a short training to them on the search interface. Each participant was instructed to finish a search task whenever they felt it was done i.e. "red rose" is found or enough images gathered for writing an essay on "gardening". The participants were allowed to go for 25 iterations maximum even if they had not found suitable images.

As mentioned earlier, the goal of this study was to find out how well our algorithm performs over pure exploitation. For the sake of completeness, we also had one algorithm for random selection of images. This algorithm is the simplest, which does nothing but selects images randomly without taking relevance feedback into consideration. For pure exploitation, we devised a separate algorithm.

For pure exploitation we extract the set of rows corresponding to already shown images, from the kernel to a separate matrix. Each row in the new matrix represents one of the already shown images, i.e. first row represents the distance from the first image in the shown images set to the rest of the images. Then, we multiply each row with the corresponding feedback, thus all the distances get weighted. After that we perform element wise addition of all the weighted rows and generate an array. Next, we sort this array and pick highest n elements and their locations in the array. These locations give the indices of the next n images to be shown.

Therefore, we compared three different algorithms, Gaussian process upper confidence bound(GPUCB), Exploitation(EXPLOIT) and Random(RAND).

For each participant, the three subtask for each major task (target, category or open-ended) were a permutation of GPUCB, EXPLOIT and RAND. We also assigned random permutations of target, category and open-ended searches to different participants, so that they could not infer what is coming next.

The participants did not know what algorithm had been used behind each task they performed.

In total we had 20 participants, each running a separate experiment out of a randomly permuted set of 9 search tasks. We recorded the following parameters,

1. The username
2. The task type
3. The task name
4. The algorithm running behind the task
5. The start time
6. The indices of the images shown per iteration
7. The numeric value of the feedback collected for each image
8. The running time of an iteration
9. The end time

8 Results

We studied two things, firstly, how quickly i.e. in how many iterations, GPUCB can finish compared to EXPLOIT and RAND. Secondly, how many images get positive feedback on average in GPUCB compared to EXPLOIT and RAND [GH14]. This comparison measures how many more relevant images GPUCB produces compared to the other two over an entire experiment, i.e. the overall effectiveness of GPUCB.

Figure 7 clearly shows that GPUCB outperforms EXPLOIT and RAND in terms of average iterations needed to finish. In all the three major tasks, GPUCB could

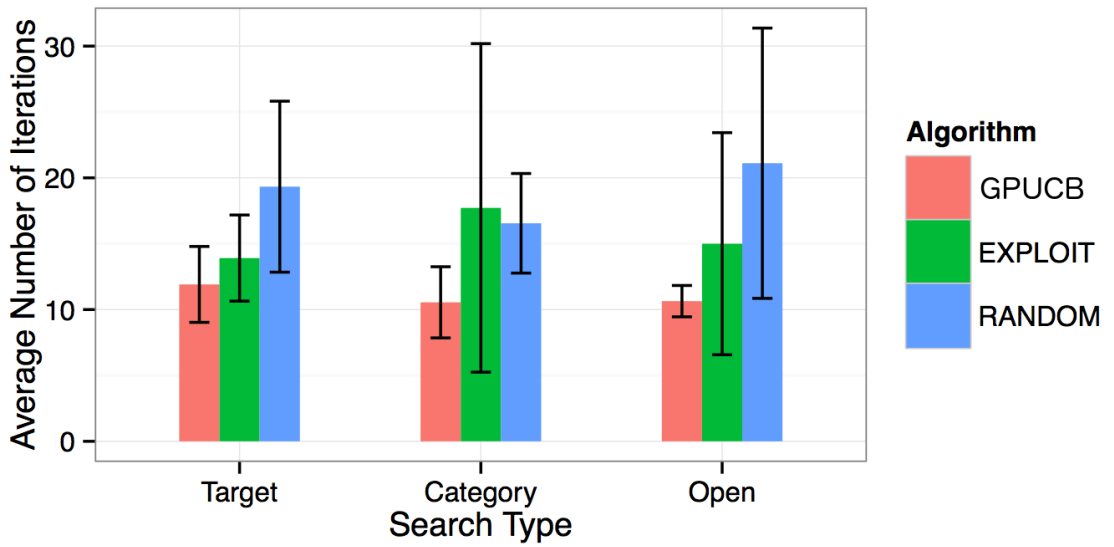


Figure 7: Average number of iterations along with 95% confidence intervals to complete each type of search with GP-SOM, EXPLOIT and RAND systems.

satisfy users in minimum number of iterations. Moreover, the confidence interval was consistent and small in case of GPUCB, while for EXPLOIT and RAND we see huge confidence interval leading to uncontrolled explorations.

EXPLOIT performed considerably better, almost as good as GPUCB, in target search. This is expected because in target search we specify fixed target, say "a red rose" to the user. If in the first iteration the user gets a red rose and provides maximum feedback possible to it, pure exploitation is the best way to conclude quickly rather than exploring which produces irrelevant images which will never receive feedbacks. Therefore, GPUCB might not be the best option in target search. Category and open ended searches do not have fixed targets, thus require more exploration and GPUCB performed better than EXPLOIT in those cases as seen from Figure 7.

Figure 8 shows the cumulative number of images getting positive feedback over the three tasks in all three algorithms. We use unannotated or untagged images, therefore the number of positive feedback over iterations are the only measure of how relevant the images were, presented to the user. As mentioned before, exploration did not affect target search to a great extent, so GPUCB and EXPLOIT generated more or less equal number of positive images. Both in category search and open ended search, GPUCB generated more positive feedback than the other two. In

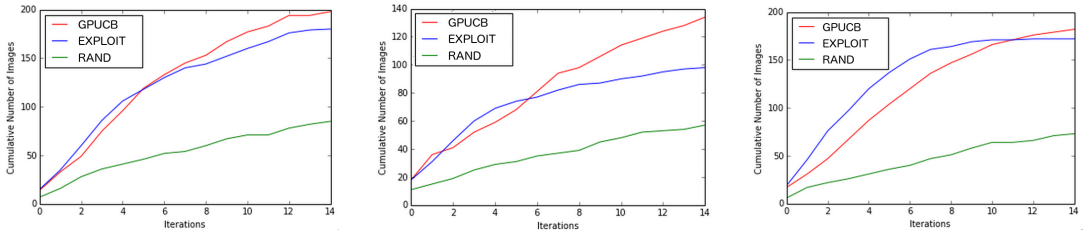


Figure 8: Cumulative number of images marked as positive by user over iterations for GP-SOM, EXPLOIT and RAND for different type of searches

all three tasks, EXPLOIT started producing more positive feedbacks in the initial iterations, as GPUCB produced some irrelevant images due to exploration, but in the end fell behind GPUCB. GPUCB performed well to finish around twenty iterations in target search and fifteen in open ended search. The other two continued. RAND was clearly outperformed by the other two everywhere.

Figure 9 and Figure 10 show the first two iterations of the experiment for red rose. Clearly the experiment started with images from random colour domains. Figure 11 shows how the experiment converged to mostly red dominated colour space with three red roses. Figure 12 to Figure 14 show the same for "city by night". It reflects how well GPUCB performed and how more relevant images appeared with later iterations.

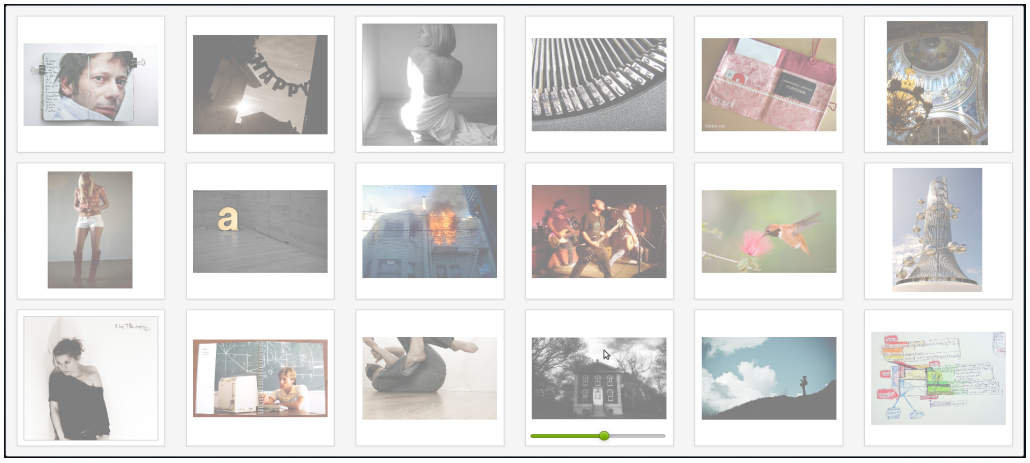


Figure 9: Red Rose Iteration 1

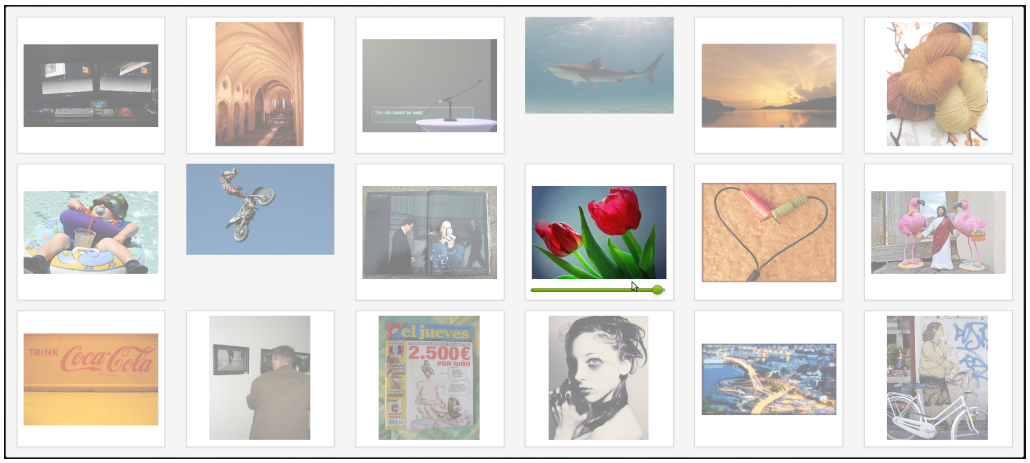


Figure 10: Red Rose Iteration 2

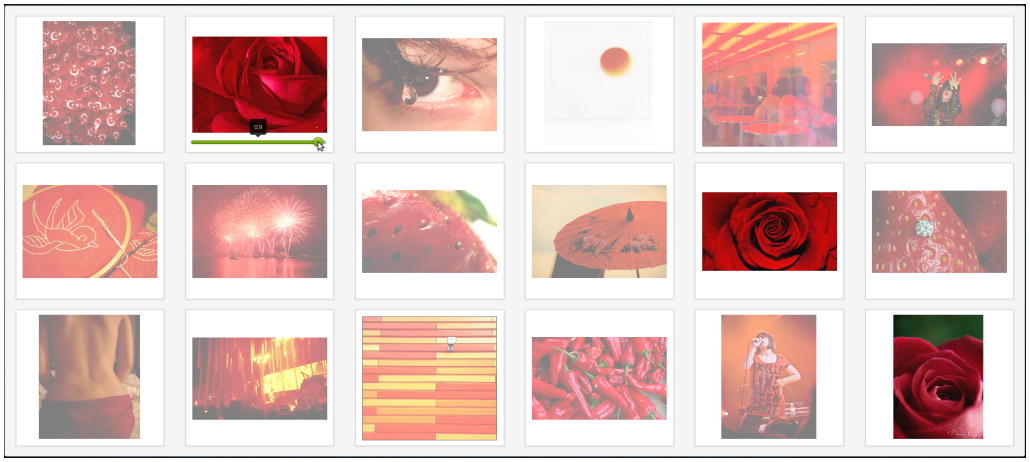


Figure 11: Red Rose Iteration 5

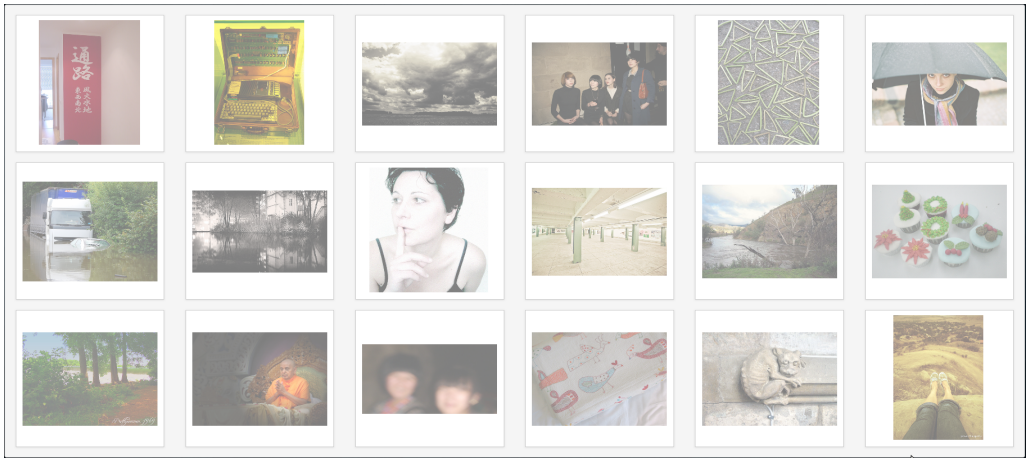


Figure 12: City by Night Iteration 1

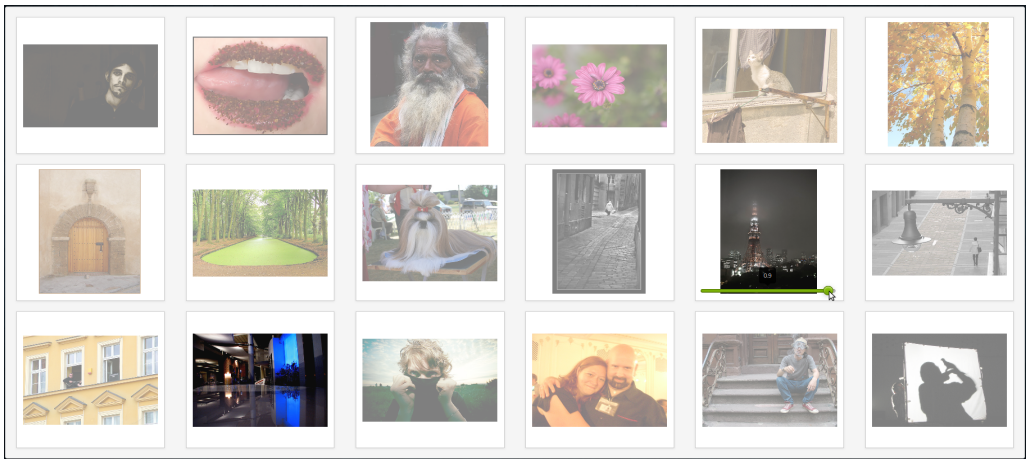


Figure 13: City by Night Iteration 2

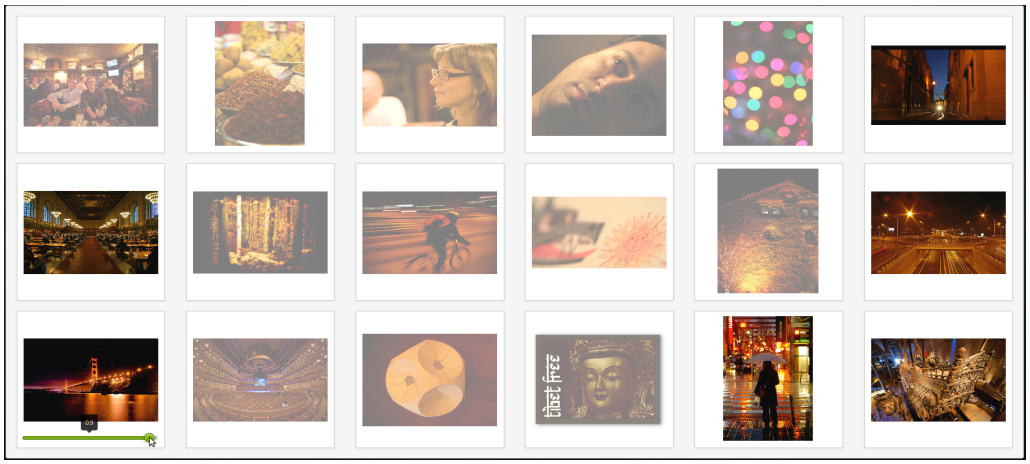


Figure 14: City by Night Iteration 5

9 Discussion

We noticed issues from the results. Next, we present them one by one.

1. One of the major issues is the applicability of GPUCB. It could not outperform EXPLOIT in target search. The reason is, once the user provides feedback to a suitable image, say a "red rose", EXPLOIT will bring more red roses only. The probability that the user will like one of them the most and finish the search is highest in case of EXPLOIT. The exploration in GPUCB here might introduce some irrelevant images, thus prolonging the process of getting the most suitable "red rose". On the contrary, sometimes it might be extremely useful to have exploration in target search. As we are showing random images initially, it might so happen that the user got no red, but a few reddish yellow

images while starting. Assuming a reddish yellow to be the closest available match, the user gave feedback to that. It is highly probable that EXPLOIT would bring up only reddish yellow images afterwards. Only exploration could bring up red images again to direct the search towards a positive end.

2. As the first iterations for both red rose (Figure 9) and city by night (Figure 12) show images randomly, it might so happen that completely irrelevant images had been shown requiring the experiment to restart .

3. The system was slow in fetching images because gaussian process took a long time to run on CPU .

4. Once the user gives all the feedback he wants and moves to the next iteration, it might so happen that the outcome is not very favourable and the user might want to change some of the previous feedback. But it is not possible to go back and change the feedback .

10 Corrective measures

We have taken corrective measures to fix problems 2, 3 and 4 mentioned above. They are described below.

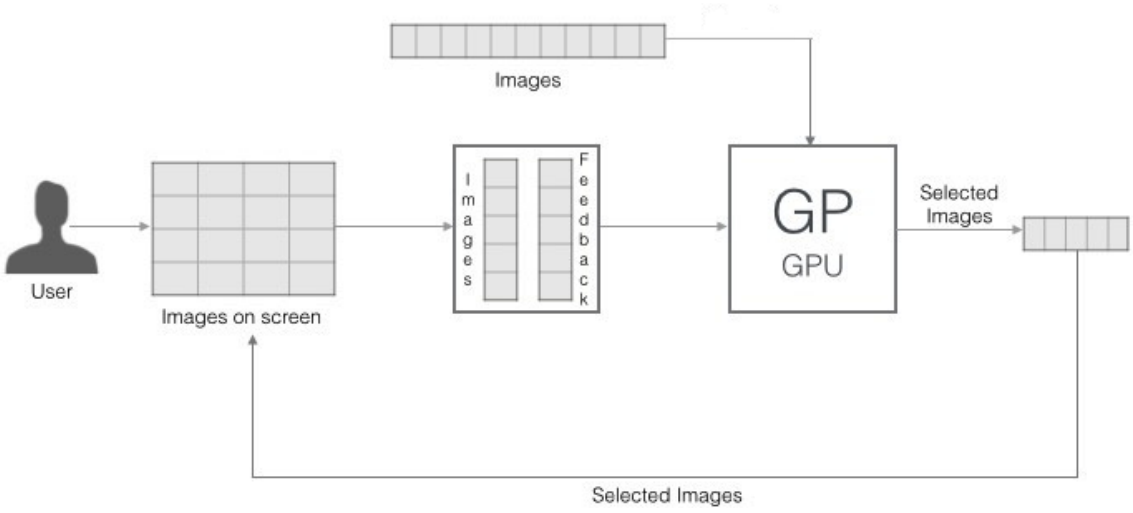


Figure 15: System diagram with GP on GPU

1. Problem 2 is fixed by setting the initial set of images fixed for a particular task. Say for red rose, whenever it runs irrespective of users, the first set of shown

images is the same always. This measure is taken for this set of experiments only. Otherwise in the ready system, for any given task, any user should start from a random set, as predefining a static set of image for each and every possible task the user can go for is not possible.

2. Problem 3 is solved by the introduction of GPU. We no longer use the two-phase Gaussian process model. Neither we have the image clusters anymore. Rather we run Gaussian process once on the entire set of images as shown in Figure 15. After calculating the UCB values over the test set of images, instead of picking one image at a time, we choose the first 12 images from the sorted UCB values. This way we save time from running the Gaussian process several times per iteration. Running Gaussian process over the entire image set (images not shown so far) takes an unaffordable amount of time on CPU but using GPU eliminates the problem. Also we do not pre-calculate the entire kernel, rather we push the input vectors (shown and not-shown image indices) into the GPU which calculates those required parts of the kernel matrix runtime. The GPU is fast enough to do that.

3. Problem 4 is solved by the introduction of a new interface called FutureView described below.

10.1 FutureView

In this experiment we introduce a new interface which helps eliminating the problem of users not being able to go back and change feedback if required. We split the screen into two sections, the left one having a width of 70% and the right one 30%. The left section has exactly the same structure of the display panel from the previous experiment but the right section is what we refer to as the FutureView. It also displays images but images here do not have any sliding-feedback controls and images appear a bit faded out. This view shows what images are going to appear in the next iteration if the user keeps the current feedback which is given to the images in the left section.

Therefore, the user sees an early view of the next iteration. If images there are not suitable, the user can change the feedback and get new images. Once the user feels that the images in the FutureView are suitable, he can go next and the images from the FutureView will replace the images in the left section. We will refer to this as current view from now on, and the same procedure follows. Initially each iteration

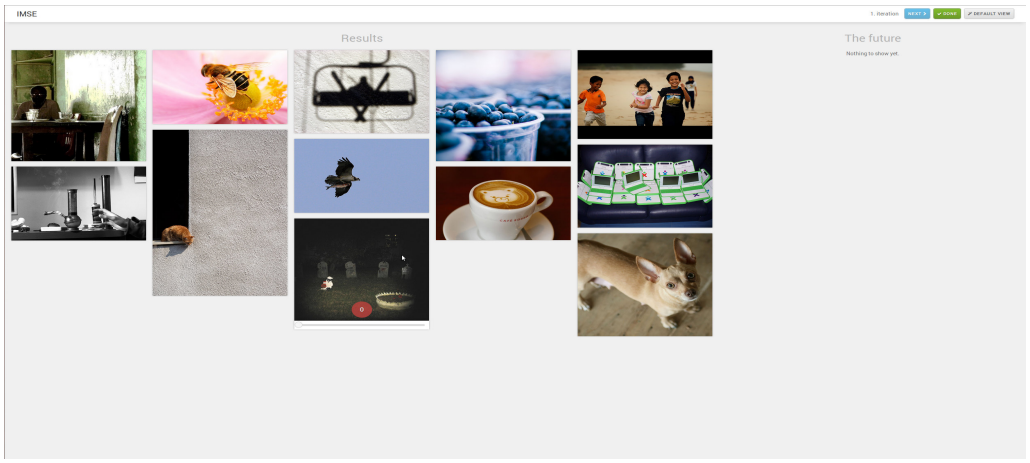


Figure 16: Empty FutureView

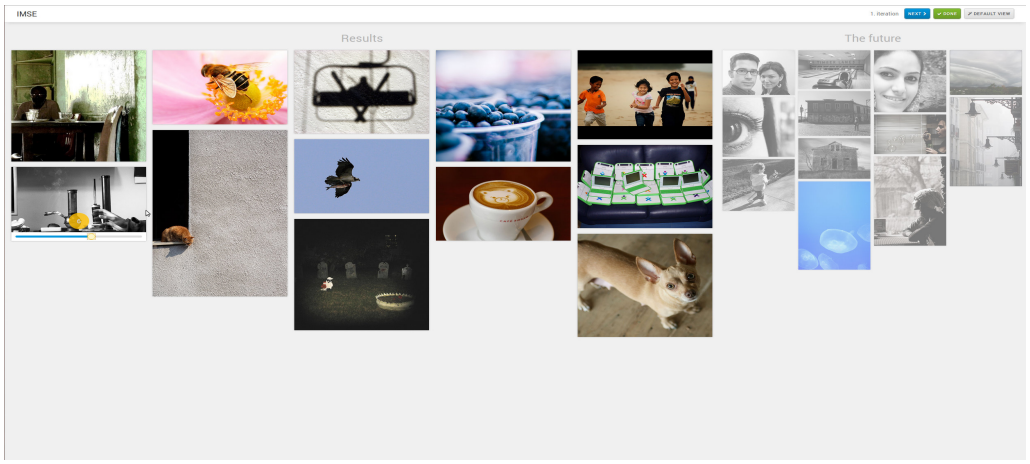


Figure 17: FutureView with images to be displayed in the next iteration

starts with an empty FutureView as shown in Figure 16 and as soon as the user gives one feedback, the FutureView fills up with images based on that feedback. Whenever the user gives feedback to any of the images in the current view, the FutureView gets refreshed with new images as shown in Figure 17.

10.2 Experimental setup

We aim to measure the effectiveness of FutureView in terms of 1) The amount of time taken to finish an experiment, 2) The number of positive feedback generated throughout the experiment, 3) The mental and physical stress of the user in performing the task and 4) Whether the user finishes a task successfully by finding the appropriate image. For a particular task we show a predefined static set of images

whenever the task is performed.

10.3 Participants

We recruited 12 participants from the university with average age 24 (maximum 30, minimum 20). Three of these participants were female. Most of these participants were students, researchers and professionals in the field of computer science, one of them a mathematician.

10.4 Tasks

The tasks were divided into three broad genres, target search, category search and open ended search, as before. We had two tasks in each of these genre,

- Target Search: Red Rose, Tall Building
- Category Search: City by Night, Seashore
- Open Ended Search: Happiness, Gardening

Our goal was to run a comparative study between the traditional view (current view) and FutureView over the same backend system to see which one performs better in terms of user satisfaction, task duration for an entire search session, amount of exploration over the image space etc.

We designed the experiments so that each participant had to perform 6 distinct tasks (Red Rose, Tall Building, City by Night, Seashore, Happiness and Gardening) altogether three being on the FutureView and the rest otherwise. We ensured each participant performed only one task from each genre on each system (we refer to the current view and FutureView as two distinct systems), thus any participant who performed the task of red rose on current view, had to perform the task of tall building on FutureView as these two are from the same genre (target search). These scheme was followed for all three genres. This way we ensured that each participant performed three task (one from each genre) on single view and the rest, the other three task on FutureView. This way we had $12 \times 3 = 36$ tasks performed on single view and 36 tasks performed on FutureView. This also ensured that each task was performed 6 times on each system.

In the previous experiment, described in 6.2, we had a random initial set to start with. But that led to having different impacts on the same task performed by different users. In this experiment, for each task, we kept a static initial set of images which could lead a user to the destination easily within 30 iterations. The task allocation is shown in Table 1. For each user and system combination, the three tasks were presented to the user one after another in a random permutation.

We used a Tobii X2-60 eye tracker with sampling rate of 60Hz, to measure how much time a user spends to look at the FutureView, to analyse how a particular search is affected by the user's approach to the FutureView. Here, we measure the amount of time the user spent looking at the FutureView. This measure gives us the idea, whether the user actually used the FutureView or not.

10.5 Procedure

Each participant was given an instruction sheet to go through before the study. We wanted to keep a record of how each individual user is familiar with searching for images on a computer, what they look for in general and which image search engines they use etc. for later reference and better understanding of their background in using such systems. Thus, we set up a Warm Up questionnaire in google form as follows,

- How often do you search for images?
- How often do you search for images for work?
- How often do you search for images for leisure?
- What kind of images do you often search for?
- What are the image search systems you have used so far? Regular search engines or something more specific?
- What difficulties did you find in using traditional image search systems you have used?
- What changes you would be happy to see in your favourite image search engines?

Each user was asked to complete the above mentioned questionnaire in the beginning. After a user filled in the questionnaire and read the instructions, they were given a consent form to sign. The user was then introduced to both the systems and led through two training tasks on each of the systems. Once the user felt familiar, we calibrated the eye tracker and started logging user's gaze. Next, we started the actual experiments. Six tasks were presented to the user in a random order. After finishing each task, the user had to fill up a task usability questionnaire (modelled after NASA TLX questionnaire [HS88]) and after every three tasks, i.e. after three experiments finished with each system, the user had to fill up a system usability questionnaire (modelled after SUS questionnaire [BKM]). After all the six tasks were finished, the eye tracker log was saved and we conducted a small interview with the user with mostly the following questions asked,

- How was your experience with this system?
- How interesting was the system to use?
- Did you notice anything different, exciting, worth mentioning in this system that you did not see in other commercial image search systems you have used?
- Was the FutureView useful?
- Was the FutureView easy to use?
- Did the FutureView help you to reach your destination more quickly than the single view?
- What prospect do you see in this system if we make it a commercial one?
- Will you personally use this system regularly if we make it live on the internet?
- Do you feel any major functionality that an image search system should have and present in commercial systems, missing in our system?
- What suggestions for modifications, improvements do you have for this system?

11 Results

For each experiment we saved the following,

- number of iterations to complete each task
- start time of experiment
- running time
- images shown
- feedbacks received
- feedback timestamps
- end time of experiment

From this information we calculated the average iterations per experiments, average running time per iteration of experiments, average total running time of an entire experiment and average number of feedback received per experiment. Table 2 shows genre wise mean and standard deviation of all the parameters we calculated. Sometimes due to outliers or sample sizes, mean and variance of two datasets differs but they follow the same data distribution. We need to show that two datasets have significant statistical differences to say that they are actually different. Here, we used *Wilcoxon signed-rank test* [Gib93] to determine whether the data generated by both FutureView and single view for each experiment and measurement have enough statistical differences or not. Table 3 shows the wilcoxon P value for each of the parameters comparing the FutureView and the single view. From Table 2 we see that the mean total running time is nearly equal in both the FutureView and single view but the average total images shown is much higher in FutureView than in single view across all the genres. We see from Table 3 that the P value is less than 0.05 in case of open and target searches, marginally over 0.05 in category search for total shown images and significantly greater than 0.05 for total duration. It supports the claim that total shown images is considerably higher in case of FutureView though it takes almost equal amount of time to finish. Therefore, the FutureView allows much higher exploration in a lesser amount of time. Figure 18 reflects the same.

In Table 4 and 5 we show the task-wise break up of the experiment parameters, respectively. From Table 5 we see that the total number of shown images is significantly higher in FutureView for "city by night", "gardening", "happiness" and "tall building". Though for "red rose" and "seashore" it did not pass the threshold of statistical significance ($p < 0.05$). It happens because maybe the user found the desired image quickly in FutureView before terminating the search, resulting a

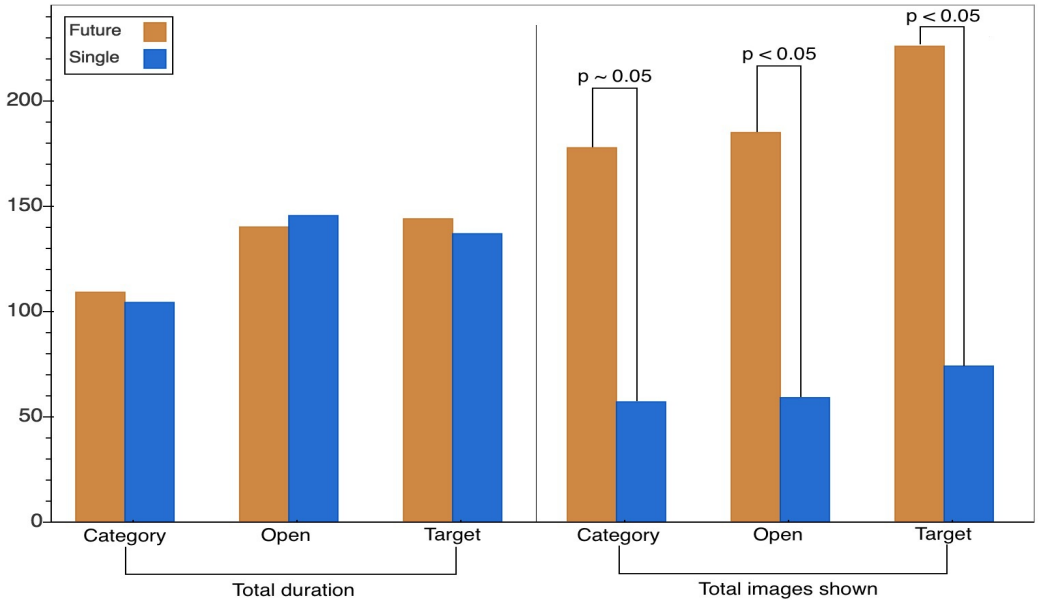


Figure 18: Total duration vs number of shown images (the Y-Axis denote 'Seconds' in 'Total duration' and 'Number of Images' in 'Total images shown')

limited number of shown images. On the other hand, in the single view may be the system explored a lot due to the presence of various colours in the images rated by the user, resulting a lot of iterations. Especially for "red rose" it is possible because the surrounding of a red rose can have multiple colours based on where it is. From Table 4 we see that the mean total shown images is much higher in FutureView for those two tasks thus resulting a cumulative higher number overall for each genre.

We also calculated the aggregates on usability parameters extracted from the NASA TLX questionnaire the users had to fill up after every task ended. These parameters were rated on a scale from 1 to 10. From Table 6 we see that for all the genres, the users felt more successful in the FutureView. It also shows that the amount of hard work needed to finish the tasks were more in single view in category search and target search. For open ended search it went otherwise. The reason could be that as the nature of the task did not give any fixed target, the users were diverted a bit when they were exposed to more exploration and could not frame a proper target in some cases thus explored a lot more than required. From Table 7 we see that for open ended search, the fact that the users felt more successful in accomplishing the tasks in FutureView, is statistically significant. Figure 19 shows a summary of the usability parameters.

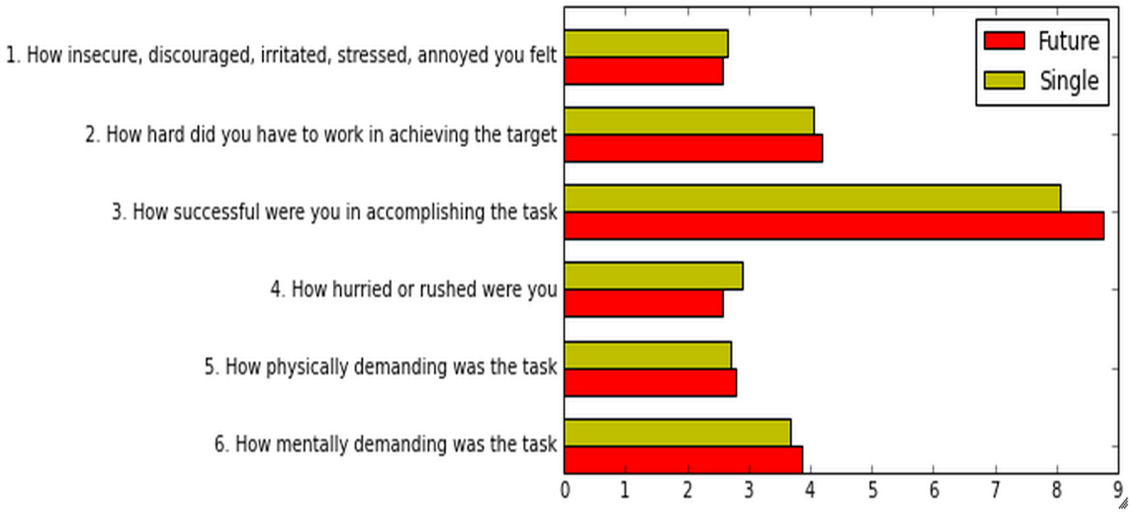


Figure 19: Comparison of usability parameters according to NASA TLX

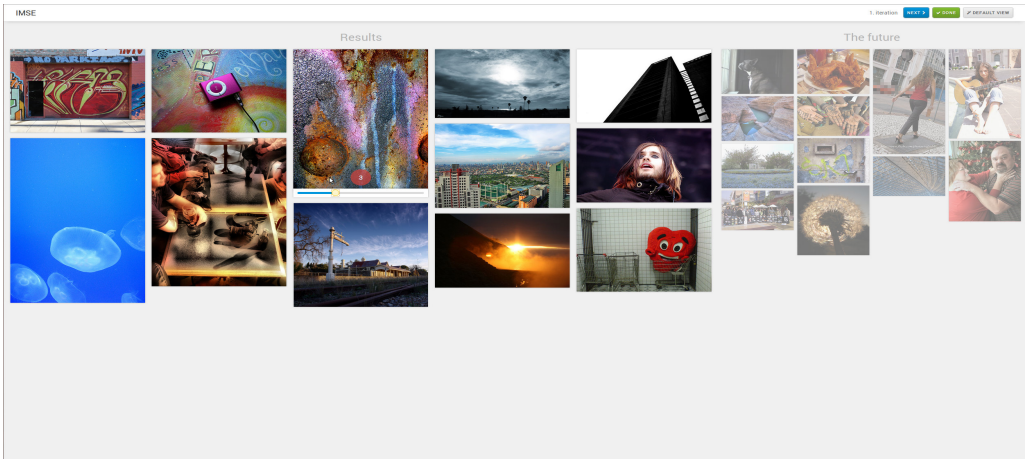


Figure 20: Gardening Iteration 1

From Table 8 we see that for tasks "city by night", "gardening", "happiness", "red rose" and "seashore", users felt more successful in the FutureView, while only for "tall building" both the views performed equally good. The reason for this could be that all the tasks other than "tall building" involves various combination of colours. Say for "red rose", the user rates a picture of a red rose in a flower vase in a room or a red rose in a garden with other flowers or plants in the surroundings. In these two cases we have two different global colour spaces. As the system considers only colour from the entire image rated, the recommended images can be very different from each other in case of these two red roses. Therefore, the user had to adjust the ratings to get the desired images in FutureView while single view did not produce

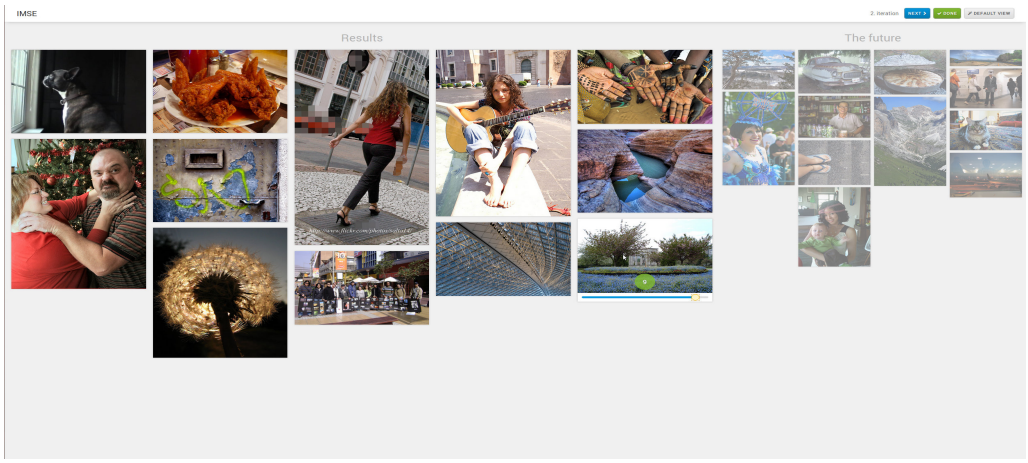


Figure 21: Gardening Iteration 2

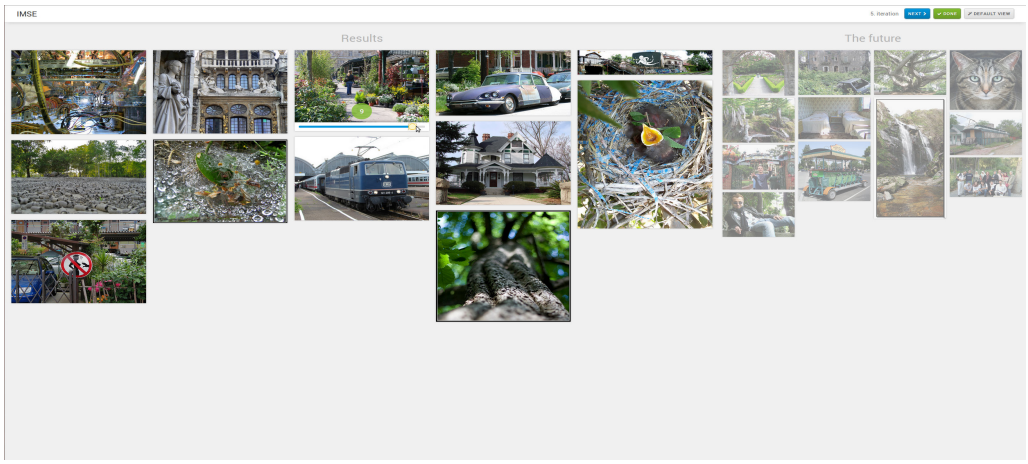


Figure 22: Gardening Iteration 5

the actual images always. The same can be said for example for "city by night". An example image can consist a lot of cars on road, or a public gathering or a deserted street at midnight with a lot or a few street lights or lights from nearby buildings. We see here also based on the images rated, the next set of recommended images can vary a lot. While for "tall building", there is a very little space for variation. An image representing so should consist one or a few "tall buildings" under the sky. We do not expect a lot of variations in the colours of buildings (at least modern high rise buildings are not made multicoloured) and the sky, so, whatever image the user chooses, there cannot be a wild variation in the possible outcome, leaving not much space for adjustment in the FutureView. Figure 20 and Figure 21 shows the first two iterations of the experiment with task Gardening in FutureView. They show how the experiment started with images in random colour spaces even in FutureView and

Figure 22 shows how it converged to a green heavy colour space with a few relevant images in the last iteration.

11.1 Excerpts from interviews

The users liked the concept of FutureView overall and they talked about that in the interviews. They said some positive things about the FutureView, like, the FutureView helped them reaching their target image more quickly than the single view, they felt secure in the navigation more in the FutureView as they could see what images are getting picked up in each step, they did not see a live view before in any of the image search systems they have used before, the FutureView is more useful to people whose job includes searching for images. The last statement supports our goal to make a system that does not just search for discrete individual images, but helps building up a story or help reaching near a story step by step. It helps searching for images which are hard to describe in a finite combination of tags.

12 Changes suggested

The users were asked for suggestions for changes or modification to the system. They provided a few good suggestions as,

- The system might include searching by text. We assume that as most image search systems include searching by text queries, the users felt that feature is missing but the system was not designed to accept text queries.
- The system does not track history. This was one of the major complains. When a user sees something interesting in the FutureView but still wants to explore a bit more, they demanded a way to save that instance like a checkpoint. Right now as the images are picked up randomly, if a user drags a slider and the images in the FutureView appear not so appealing and the user pushes the slider back to the previous state, the system does not get back to the exact same state, i.e. the system does not show the exact same set of images presented before dragging the slider. That is a major issue according to most of the users.
- The images in the FutureView were purposefully kept a bit blurred out so that the two views appear visually different. The users did not like it. They felt

that because of that blurring they could not see the actual colour of the images in the FutureView thus in a few occasions could not decide whether they had received the image they were looking for. As the system relies on colour, it was a major drawback from the usability point of view.

13 Problems with the system

Apart from what the users suggested we also felt some issues with the system while running the study. They are,

- The web server (Apache 2) needed to be restarted after each user finishes otherwise the system might crash. It crashed a couple of times resulting in a restart in the ongoing task. Actually, we show a lot of images which somehow clogs the browser or the memory in the backend. Restarting a server cannot be a solution. We should have a mechanism to clear images from previous experiments.
- The FutureView takes nearly a second to load. Sometimes users being too impatient, clicked on the next button right after dragging a slider without waiting for the FutureView to load. It led to a crash in the front end. We should disable the next button while the FutureView loads.
- One experiment had to be cancelled because a user used arrow keys on the keyboard to move the slider rather than dragging with mouse. We record mouse events to sync with eye tracker therefore arrow key press was useless. We should have had disabled those keys.

14 Future plan

The study gave rise to a few questions and place for improvements which we will try to address in the next study.

- One obvious place for improvement is to change the feedback system. Instead of dragging a tiny slider control, we can let the user rate the images quicker and with more convenience. This will bring the cognitive load down further.

- Enabling user to circle parts of images and let the system detect identifiable objects from the circled area and search for those objects is another area of research.

- Right now the system uses only colour as feature. We can put other features as texture, shape etc. and build separate kernels for them. We can then make the system search in all these different kernels and combine results from these multiple kernels in the end.

- The eye-tracker can be put to further use. We can record the implicit feedback from the eye-tracker and compare that with the actual recorded feedback to infer the mental state of the user. We can see how attentive they were during the experiment and whether they provided feedback keeping the task in mind or randomly.

15 Conclusion

The aim of this study was to create a novel interface that helps users in searching through contents of images while keeping the cognitive load low. From the results we claim that our interface performed quite well in exploring a lot of images within a comparable amount of time. The users were happy with the FutureView and according to the NASA TLX questionnaire they were more successful in completing the given tasks in FutureView. The study gave rise to a few questions and place for improvements which we will try to address in the next study as discussed in the last section.

For this study we used only colour as features for simplicity, but the overall colour histogram of an image is not sufficient to depict an image completely in terms of what objects the image contains, what theme the image could be assigned to etc. Therefore, if the user wants to look for a particular object he sees in an example image or a collection of objects depicting a context, we need to take a step further by extracting local features of an image in terms of shapes, textures etc. We need to identify presence of various objects in an image and extract them as features. Once this is done we can design various kinds of tasks for users. If we know what exactly an image contents, we can provide visual guidance to users who often tend to overlook, tiny objects in a corner in the target image. We try to extract all these local image features in a completely unsupervised manner, which is a very interesting

research problem. We can combine local features with colours also. If the system understands that a user is looking for cars and can recommend images containing so, it can also recommend red cars or blue cars whichever is required. Thus, we can combine different features together to produce more meaningful results.

References

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

Agr95 Agrawal, R., Sample mean based index policies with o(log n) regret for the multi-armed bandit problem. *Advances in Applied Probability*, volume 27. Applied Probability Trust, 1995, pages 1054 – 1078.

AHK+10 Auer, P., Hussain, J., Kaski, S., Klami, A., Kujala, J., Laaksonen, J., Leung, A., Pasupa, K. and Shawe-Taylor, J., Implicit feedback in content-based image retrieval. *Workshop on Applications of Pattern Analysis*, volume 21, Cumberland Lodge, UK, 2010, WAPA, pages 1199 – 1207.

Aue02 Auer, P., Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, volume 3. JMLR, 2002, pages 397–422.

BFG96 Bach, J., Fuller, C. and Gupta, A., The virage image search engine: an open framework for image management. *Storage and Retrieval for Image and Video Databases*, volume 2670, San Jose, CA, USA, 1996, SPIE.

BKM Bangor, A., Kortum, P. and Miller, J., The system usability scale (sus): An empirical evaluation. *International Journal of Human-Computer Interaction*, 6.

BLJ04 Bach, F. R., Lanckriet, G. R. G. and Jordan, M. I., Multiple kernel learning, conic duality, and the smo algorithm. *ICML'04 Proceedings of the twenty-first international conference on Machine learning*, New York, USA, 2004, ACM.

Can Canny, J., A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8, pages 679–698.

CF80 Chang, N. S. and Fu, K. S., Query by pictorial example. *IEEE transactions on Software Engineering.*

CM00 C., V. R. and M., T., Content-based image retrieval systems: A survey. Technical Report, Department of Computing Science, Utrecht University, 2000.

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. *IEEE Transactions on Image Processing*, ,1, pages 20–37.

DLW Datta, R., Li, J. and Wang, J., Content-based image retrieval: approaches and trends of the new age. *MIR'05, Proceedings of the 7th ACM SIGMM.*

FSN95 Flickner, M., Sawhney, H. and Niblack, W., Query by image and video content: the qbic system. IEEE, 1995.

GH14 Glowacka, D. and Hore, S., Balancing exploration-exploitation in image retrieval. UMAP, 2014.

Gib93 Gibbon, J. D., *Non Parametric Statistics, An Introduction.* Sage Publications, Newbury Park, California, 1993.

Git79 Gittins, J. C., Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society, Series B, 41*, 1979, pages 148–177.

GJ04 Gittins, J. C. and Jones, D. M., A dynamic allocation index for the sequential design of experiments. *ICML'04 Proceedings of the twenty-first international conference on Machine learning*, North-Holland, Amsterdam-London, 2004, Progress in Statistics (J. Gani, ed.), pages 241 – 266.

GRK+13 Glowacka, D., Ruotsalo, T., Konyushkova, K., Athukorala, K., K. S. and Jacucci, G., Directing exploratory search: Reinforcement learning from user interactions with keywords. IUI, 2013.

GST10 Glowacka, D. and Shawe-Taylor, J., Content-based image retrieval with multinomial relevance feedback. ACML, 2010, pages 111–125.

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*, New York, NY, USA, 2008, ACM.

HS88 Hart, S. G. and Staveland, L. E., Development of nasa-tlx (task load index): Results of empirical and theoretical research. *P. A. Hancock and N. Meshkati (Eds.) Human Mental Workload*, Amsterdam, 1988, North Holland Press.

HTPG14 Hore, S., Tyrvainen, L., Pykkö, J. and Glowacka, D., A reinforcement learning approach to query-less image retrieval. 2014.

HWd Hartigan, J. A., Wong, M. A., S. and de, R. M., A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, number 1, pages 100–108.

KF06 Kelly, D. and Fu, X., Elicitation of term relevance feedback: an investigation of term source and contex. SIGIR, 2006.

KG 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*, number 1, pages 100–108.

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. Computer Vision and Applications*, 1992, pages 530–533.

KM05 Kosch, H. and Maier, P., Content-based image retrieval systems-reviewing and benchmarking. *Multimedia information retrieval*, pages 253–262.

KO90 Koikkalainen, P. and Oja, E., Self organizing hierarchical feature maps. *Proceedings of 1990 International Joint Conference on Neural Networks*, volume 2, San Diego, CA, 1990, IEEE, INNS.

Koh Kohonen, T., The self organizing map. *Proceedings of the IEEE.*

Koi94 Koikkalainen, P., Progress with the tree-structured self- organizing map. *11th European Conference on Artificial Intelligence*, Wiley, New York, 1994, European Committee for Artificial Intelligence (ECCAI).

Kon13 Konyushkova, K., Imse: Instant interactive image retrieval system with exploration/exploitation trade-off. Master's thesis, Department of Computer Science, University of Helsinki, 2013.

KSd11 Katja, H., Shimon, W. and de, R. M., Contextual bandits for information retrieval. *NIPS 2011 Workshop on Bayesian Optimization, Experimental Design, and Bandits*, 2011.

LCLS10 Li, L., Chu, W., Langford, J. and Schapire, R., A contextual-bandit approach to personalised news article recommendation. *Proceedings of the 19th international conference on World wide web*, Raleigh, NC, USA, 2010, ACM, 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).

Low99 Lowe, D. G., Object recognition from local scale-invariant features. *Proc. 7th International Conference on Computer Vision (ICCV'99)*, pages 1150–1157.

LR85 Lai, T. L. and Robbins, H., Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics, 6*, 1985, pages 4–22.

PCA07 Pandey, S., Chakrabarti, D. and Agarwal, D., Multi-armed bandit problems with dependent arms. *Proceedings of the 24th International Conference on Machine Learning, Corvallis, OR*, 2007.

PPS94 Pentland, A., Picard, R. and Sclaroff, S., Photobook: tools for content-based manipulation of image databases. *Storage and Retrieval for Image and Video Databases*, volume 2185, San Jose, CA, USA, 1994, SPIE.

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

SB98 Sutton, R. S. and Burto, A. G., *Reinforcement Learning: An Introduction.* The MIT Press, Cambridge, Massachusetts, 1998.

SF12 Suditu, N. and Fleuret, F., Iterative relevance feedback with adaptive exploration/exploitation trade-off. *Storage and Retrieval for Image and Video Databases*. CIKM, 2012.

SWS+00 Smeulders, A., Worring, M., Santini, S., Gupta, A. and Jain, R., Content-based image retrieval at the end of the early years. *Pattern Anal. Mach. Intell.*, ,12, pages 1349–1380.

TC01 Tong, S. and Chang, E., Support vector machine active learning for image retrieval. *Proceedings of the ACM International Conference on Multimedia*, Ottawa, Canada, 2001.

Vas04 Vasconcelos, N., On the efficient evaluation of probabilistic similarity functions for image retrieval. *IEEE Trans. Inf. Theory*, Ottawa, Canada, 2004.

Vas13 Vasconcelos, N., The design of end-to-end optimal image retrieval systems. *Proceedings of the International Conference on ANN*, Istanbul, turkey, 2013.

VGJ08 Villa, R., Gildea, N. and Jose, J. M., A faceted interface for multimedia search. SIGIR, 2008.

VLP13 Villegas, M., Leiva, L. A. and Paredes, R., Interactive image retrieval based on relevance feedback. *Multimodal Interaction in Image and Video Application*, 2013, pages 83–109.

YSLH03 Yee, K. P., Swearingen, K., Li, K. and Hearst, M., Faceted metadata for image search and browsing. CHI, 2003, pages 401–408.

ZH03 Zhou, X. and Huang, T., Relevance feedback in image retrieval: A comprehensive review. *Multimedia Systems*, ,6, pages 536–544.

ZLZ01 Zhang, L., Liu, F. and Zhang, B., Support vector machine learning for image retrieval. *International Conference on Image Processing*, 2001.

User serial number	Task performed on Single View	Task performed on FutureView
1	Red Rose City by Night Happiness	Tall Building Seashore Gardening
2	Red Rose Seashore Gardening	Tall Building City by Night Happiness
3	Red Rose City by Night Happiness	Tall Building Seashore Gardening
4	Red Rose Seashore Gardening	Tall Building City by Night Happiness
5	Red Rose City by Night Happiness	Tall Building Seashore Gardening
6	Red Rose Seashore Gardening	Tall Building City by Night Happiness
7	Red Rose Seashore Gardening	Tall Building City by Night Happiness
8	Red Rose City by Night Happiness	Tall Building Seashore Gardening
9	Red Rose Seashore Gardening	Tall Building City by Night Happiness
10	Red Rose City by Night Happiness	Tall Building Seashore Gardening
11	Red Rose Seashore Gardening	Tall Building City by Night Happiness
12	Red Rose City by Night Happiness	Tall Building Seashore Gardening

Table 1: Task distribution per user

Genre wise Mean (Standard Deviation)						
	Category		Open		Target	
	Future	Single	Future	Single	Future	Single
Iterations	14.83 (17.96)	4.75 (4.75)	15.42 (10.23)	4.92 (4.48)	18.83 (14.36)	6.17 (4.71)
Avg. duration/Iteration (in Sec.)	8.86 (2.96)	24.39 (4.41)	10.04 (4.06)	31.26 (10.83)	8.71 (2.8)	24.44 (11.55)
Total duration (in Sec.)	109.08 (101.72)	104.25 (101.11)	140.08 (84.13)	145.5 (123.42)	144 (95.11)	136.92 (105.33)
Total shown images	178 (215.57)	57 (57.01)	185 (122.75)	59 (53.78)	226 (172.37)	74 (56.48)
Total feedback received	14.83 (17.96)	9.33 (8.05)	15.42 (10.23)	17.5 (18.26)	18.83 (14.36)	15.58 (17.68)

Table 2: Genre wise Mean and Standard Deviation of experiment parameters

Genre wise Wilcoxon P value			
	Category	Open	Target
Iterations	0.05405584757	0.003239921272	0.0206095954
Avg. duration/Iteration (in Sec.)	0.002217721464	0.002217721464	0.002217721464
Total duration (in Sec.)	0.8139453463	0.9687003809	0.6378701799
Total shown images	0.05405584757	0.003239921272	0.0206095954
Total feedback received	0.7891658874	0.7555407781	0.3461504513

Table 3: Genre wise Wilcoxon P value measure of experiment parameters (Future-View vs Single View)

Task wise Mean (Standard Deviation)						
	City by Night		Gardening		Happiness	
	Future	Single	Future	Single	Future	Single
Iterations	8.33 (5.05)	2.5 (2.35)	16.33 (8.45)	7 (5.76)	14.50 (12.52)	2.83 (0.75)
Avg. duration/Iteration (in Sec.)	9.87 (3.5)	24.7 (5.02)	8.83 (2.58)	31.24 (11.17)	11.25 (5.1)	31.27 (11.55)
Total duration (in Sec.)	82.83 (58.99)	54.17 (42.75)	136.83 (74.83)	209 (153.56)	143.33 (99.73)	82 (15.89)
Total shown images	100 (60.56)	30 (28.14)	196 (101.45)	84 (69.14)	174 (150.22)	34 (9.03)
Total feedback received	8.33 (5.05)	4.67 (3.56)	16.33 (8.45)	25.5 (23.97)	14.5 (12.52)	9.5 (2.35)
	Red Rose		Seashore		Tall Building	
	Future	Single	Future	Single	Future	Single
Iterations	23.83 (18.17)	7.67 (6.09)	21.33 (24.15)	7 (5.66)	13.83 (7.99)	4.67 (2.5)
Avg. duration/Iteration (in Sec.)	9.04 (3.67)	20.58 (2.8)	7.84 (2.13)	24.09 (4.18)	8.37 (1.87)	28.3 (15.81)
Total duration (in Sec.)	169.5 (112.02)	157 (137.22)	135.33 (132.78)	154.33 (121.01)	118.5 (76.11)	116.83 (67.91)
Total shown images	286 (218.05)	92 (73.06)	256 (289.76)	84 (67.88)	166 (95.82)	56 (30.04)
Total feedback received	23.83 (18.17)	22.33 (23.37)	21.33 (24.15)	14 (8.81)	13.83 (7.99)	8.83 (5.67)

Table 4: Task wise Mean and Standard Deviation of experiment parameters

Task wise Wilcoxon P value			
	City by Night	**Gardening**	**Happiness**
Iterations	0.07473549831	0.02770784936	0.02770784936
Avg. duration/Iteration (in Sec.)	0.02770784936	0.02770784936	0.04639946187
Total duration (in Sec.)	0.3454475305	0.3454475305	0.172954918
Total shown images	0.07473549831	0.02770784936	0.02770784936
Total feedback received	0.2071604489	0.3454475305	0.5282327769
	Red Rose	**Seashore**	**Tall Building**
Iterations	0.172954918	0.1380107376	0.02728117148
Avg. duration/Iteration (in Sec.)	0.02770784936	0.02770784936	0.02770784936
Total duration (in Sec.)	0.6001794871	0.6001794871	0.9165119079
Total shown images	0.172954918	0.1380107376	0.02728117148
Total feedback received	0.5001842571	0.752493649	0.1380107376

Table 5: Task wise Wilcoxon P value measure of experiment parameters (FutureView vs Single View)

Genre wise Mean (Standard Deviation)						
	Category		**Open**		**Target**	
	Future	**Single**	**Future**	**Single**	**Future**	**Single**
How mentally demanding was the task?	3.42 (2.27)	3 (1.6)	4.42 (2.23)	4.08 (1.73)	3.75 (1.66)	3.92 (1.78)
How physically demanding was the task?	2.75 (2.6)	2.5 (1.88)	2.67 (2.15)	2.67 (1.61)	2.92 (2.07)	3 (1.81)
How hurried or rushed you felt while performing the task?	2.75 (1.82)	2.67 (1.56)	2.75 (1.48)	3 (1.81)	2.25 (1.42)	3 (1.95)
How successful were you in accomplishing the task?	9.08 (1.68)	8.67 (1.3)	8.83 (1.11)	7.83 (1.27)	8.33 (1.61)	7.67 (2.57)
How hard did you have to work for accomplishing the task?	3.58 (2.61)	3.83 (2.33)	5.33 (2.67)	4.42 (2.35)	3.67 (1.83)	3.92 (2.15)
How insecure, discouraged, irritated, stressed and annoyed were you?	2.33 (2.1)	2.08 (1.16)	3 (2.45)	3.08 (1.68)	2.42 (1.73)	2.83 (2.21)

Table 6: Genre wise Mean and Standard Deviation of usability parameters according to NASA TLX questionnaire

Genre wise Wilcoxon P value			
	Category	**Open**	**Target**
How mentally demanding was the task?	0.5817769166	0.8767217432	0.7957199701
How physically demanding was the task?	0.5879367462	1	0.7054569861
How hurried or rushed you felt while performing the task?	0.7301661743	0.5890107045	0.04734500209
How successful were you in accomplishing the task?	0.4982248534	0.02192029034	0.5124416997
How hard did you have to work for accomplishing the task?	0.6846218727	0.4126908569	0.8582145731
How insecure, discouraged, irritated, stressed and annoyed were you?	0.864324782	0.8772900143	0.6666320553

Table 7: Genre wise Wilcoxon P value measure of usability parameters according to NASA TLX questionnaire (FutureView vs Single View)

Task wise Mean (Standard Deviation)						
	City by Night		Gardening		Happiness	
	Future	Single	Future	Single	Future	Single
How mentally demanding was the task?	3.17 (1.33)	2.67 (1.97)	5.33 (2.8)	3.83 (1.94)	3.5 (1.05)	4.33 (1.63)
How physically demanding was the task?	1.67 (0.82)	3.33 (2.42)	3.83 (2.56)	1.67 (0.82)	1.5 (0.55)	3.67 (1.63)
How hurried or rushed you felt while performing the task?	2.5 (1.38)	2.17 (1.47)	3.17 (1.47)	3.33 (1.86)	2.33 (1.51)	2.67 (1.86)
How successful were you in accomplishing the task?	9.33 (0.52)	9 (1.1)	8.67 (1.21)	8 (1.55)	9 (1.1)	7.67 (1.03)
How hard did you have to work for accomplishing the task?	3.33 (1.86)	3.33 (2.16)	5.83 (2.86)	4.5 (2.66)	4.83 (2.64)	4.33 (2.25)
How insecure, discouraged, irritated, stressed and annoyed were you?	1.5 (0.84)	2 (1.26)	4.33 (2.94)	3.17 (2.04)	1.67 (0.52)	3 (1.41)
	Red Rose		Seashore		Tall Building	
	Future	Single	Future	Single	Future	Single
How mentally demanding was the task?	3.67 (1.63)	4 (2.19)	3.67 (3.08)	3.33 (1.21)	3.83 (1.83)	3.83 (1.47)
How physically demanding was the task?	3 (2.19)	3.5 (2.26)	3.83 (3.37)	1.67 (0.52)	2.83 (2.14)	2.5 (1.22)
How hurried or rushed you felt while performing the task?	2.33 (1.51)	3.67 (2.07)	3 (2.28)	3.17 (1.6)	2.17 (1.47)	2.33 (1.75)
How successful were you in accomplishing the task?	9 (1.26)	7.67 (2.88)	8.83 (2.4)	8.33 (1.51)	7.67 (1.75)	7.67 (2.5)
How hard did you have to work for accomplishing the task?	3.67 (2.07)	4.67 (2.66)	3.83 (3.37)	4.33 (2.58)	3.67 (1.75)	3.17 (1.33)
How insecure, discouraged, irritated, stressed and annoyed were you?	1.67 (0.82)	3.5 (2.88)	3.17 (2.71)	2.17 (1.17)	3.17 (2.14)	2.17 (1.17)

Table 8: Task wise Mean and Standard Deviation of usability parameters according to NASA TLX questionnaire

Task wise Wilcoxon P value			
	City by Night	**Gardening**	**Happiness**
How mentally demanding was the task?	0.4795001222	0.3430278273	0.4605966187
How physically demanding was the task?	0.1755543028	0.07154597259	0.02559680539
How hurried or rushed you felt while performing the task?	0.6830913983	1	0.7854947471
How successful were you in accomplishing the task?	0.3173105079	0.4962424744	0.07414989774
How hard did you have to work for accomplishing the task?	0.8907458009	0.6732899797	0.670694381
How insecure, discouraged, irritated, stressed and annoyed were you?	0.256839258	0.4962424744	0.08446903225
	Red Rose	**Seashore**	**Tall Building**
How mentally demanding was the task?	0.4142161782	0.6724316013	1
How physically demanding was the task?	0.4962424744	0.1307970618	0.9153452831
How hurried or rushed you felt while performing the task?	0.10880943	1	0.7864570351
How successful were you in accomplishing the task?	0.4795001222	0.4614509878	0.684469821
How hard did you have to work for accomplishing the task?	0.4142161782	0.4614509878	0.8316408425
How insecure, discouraged, irritated, stressed and annoyed were you?	0.1307970618	0.5001842571	0.3362887904

Table 9: Task wise Wilcoxon P value measure of usability parameters according to NASA TLX questionnaire (FutureView vs Single View)