Date of acceptance Grade

Instructor

IMSE, A content based image retrieval system

Sayantan Hore

Helsinki July 20, 2014

UNIVERSITY OF HELSINKI
Department of Computer Science

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

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

Tekijä — Författare — Author
Sayantan Hore

Työn nimi — Arbetets titel — Title

IMSE, A content based image retrieval system

Oppiaine — Läroämne — Subject
Computer Science

Työn laji — Arbetets art — Level	Aika — Datum — Month and year July 20, 2014	Sivumäärä — Sidoantal — Number of pages 19 pages + 1 appendices

Tiivistelmä — Referat — Abstract

Content based image retrieval systems are out there for a few years now. We present a system, IMSE, here which searches out images using Gaussian Process Upper Confidence Bound algorithm. This algorithm has performed better in comparison to random search and plain exploration. We present a comparison of how the algorithm performs on CPU as well as GPU. We also evaluate the user experience and with the search interface.

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

Content based image retrieval systems have been proven to fit better in retrieval performance. Text or tag based matching systems rely only on texts, thus the images themselves are secondary in the search procedure. The user sees the image and selects one by the objects and color the image represents. The combination of those objects and colors should actually be searched for instead of tags. Tags hardly represent an entire image. An image portraying a cloudy afternoon can be tagged as "Sad", based on the mood and understanding of the person creating the tags. Another image of the same genre, tagged by different persons, might miss the tag. If a user is in search for "Sad" images and finds one with the tag, selects it. The other "Sad" image would not appear in the search results as it lacks the tag. Had features and content of the image been searched for, we could get better and more appropriate search results.

IMSE is a content based image retrieval system, using Gaussian Process Upper Confidence Bound (GP-UCB) as a retrieval algorithm. We are using MIRFLICKR 25000 image set for testing. As this produces a huge kernel, GP-UCB is slow on CPU. We have written a GPU version of the algorithm and made a comparative study of running time over CPU and GPU. The system is written in Python (and Django for the web interface). Any system has to have a pleasing and assisting user interface. As our system is targeted towards web users, we cared for having a decent web interface, written in Twitter Bootstrap (v3.0) framework, jQuery (v1.11.1), CSS and javascript. We have followed Human Computer Interaction guidelines for user interaction.

In subsequent section, we will present the theoretical background of Reinforcement Learning, Gaussian Process, Upper Confidence Bound, Image Retrieval, GPU implementation details, user experience and performance measures.

2 The need for content based retrieval in detail

...

2.1 Shortcomings of traditional image search

...

2.2 Why content based retrieval is better

...

2.3 Approaches

...

3 Theoretical background

...

3.1 Reinforcement Learning

Reinforcent learning [SB98] is a form of machine learning in which the learning agent interacts with its surroundings, by performing an action. The surroundings or the environment reacts by giving some feedback to the agent. The feedback, perceived to be either positive or negative, helps the agent to understand whether the action was correct or incorrect respectively. The feedback is considered as a reward associated with the action performed. The agent has a goal to reach which is quantified as receiving the maximum cumulative reward after a finite sequence of actions. A system modelled after reinforcement learning, has a finite number of states $S = \{s_1, s_2, ..., s_n\}$. The agen starts from a state s_i and tries to reach a *goal state* $s_j; j \neq i$. Any state $s_k \in S$ has a set of actions $A = \{a_1, a_2, ..., a_m\}$ associated with it. The agent while being in s_k can perform any one of the available actions. It receives a reward r_k, which is positive if the action takes the agent closer to the goal state s_j, otherwise negative. Clearly the same action (moving forward or backward for example), associated to more than one different states, can produce different rewards depending on starting state and goal state. The task of learning here is to predict the reward for a particular action given the state. As the agent goes through trials, it gathers knowledge or inference about the outcome of an action for a state in terms of reward, as it receives an immediate reward for each action performed. The knowledge of the agent about different combination of state-action pairs is reinforced real time.

Reinforcement learning is fundamentally different from two other major form of learnings, supervised and unsupervised learning. Neither it tries to classify a new unlabeled dataset based on already labeled dataset, nor it tries to find patterns in a new dataset. It rather tries to understand an environment which can react on its action. It is therefore useful in applications of game theory, robotics, web advertisement, product prediction in online stores etc. In all these cases there is an active party providing rewards on an action. For example in product recommendation in an online store, the system, acting as an agent, recommends a product and the customer, acting as the environment, can either choose the product or not, thus generating positive or negative reward resprectively.

3.1.1 Exploration and Exploitation

There are problems where there is an infinitely large state-action pairs but the agent has to reach the goal state in finite steps. Clearly, the shortest path generates the maximum reward. As the step size is finite, the agent cannot try each available state-action pair. It would rather try to repeat the action in each state it visits, that has generated the maximum reward so far. Then it might miss an action in a particular state that would generate a bigger reward. Again one action might generate a lower immediate reward in a state but it might be a keypoint that follows a shorter path than the known one to the goal state thereafter, thus contributing to a bigger overall gain. The agent has two problems here, firstly, as it has to achieve the goal in finite time, it cannot go through all possible combinations (i.e. *exploration*) much. Secondly, always performing the best known action (i.e. *exploitation*) might miss a better action. This situation is known as exploration-exploitation [SB98] dilemma. There is common solution to this problem called *epsilon-greedy* or ϵ *-greedy.*

3.2 Bandit Problems

Bandit problems [HWdR11] are sequential decision problems where each iteration in the sequence has multiple choices of actions. 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 situation can be best described in terms of a slot machine in a casino. The machine has K arms each with a distribution of reward. 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. To find that out the user has to explore various arms in succession. This kind of problems are known as 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 satisfactory 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. So 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 big-reward-arm enough, generating a low overall feedback. Therefore there is always a tussle between exploitation and exploration. The learning agent tries to learn an optimal combination of these two.

Reinforcement learning fits well with bandit problems where the user, who pulls the arms of slot machines, is the agent and the machine or the collection of machines as a whole is the environment provides an immediate reward.

3.2.1 Upper Confidence Bound

In typical 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 $R = \{R_1, R_2, ..., R_n \mid R_i \in \mathcal{R}\}$. 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 we do not know, so need to estimate. 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,

$$\mu_{R_1} = \frac{\sum_{j=1}^{k} r_i^j}{k} \tag{1}$$

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. This U_i 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, therefore we call U_i the *upper confidence bound*. This simple solution for the exploitation-exploration dilemma was given by R. Agrawal in his upper confidence bound algorithm [R.A95].

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

3.3 Regression Problems

Regression problems are supervised learning problems where we have multiple random variables. At least one of those variables are dependant on a subset of the rest. Lets' assume that we have to predict rainfall prediction in a city for the coming monsoon based on average summer temperature. We first 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. Clearly x is an independent random variable and y is dependent on x. We take a set $S = \{x_i, y_i\}$ for the past years. The goal here is to learn the relation between x and y. We will apply the 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,

$$f(x_i) = \theta_1 x_i + \theta_0 \tag{2}$$

But often in real life scenarios we have more than one independent variables. Besides temperature we can have amount of CO_2 emission, amount of deforestation (in square kilometres) and so forth. Therefore in these cases 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,

$$f(\mathbf{x}_i) = \sum_{j=1}^{m} \theta_j x_{ij} + \theta_0 \tag{3}$$

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 $f(x_i)$.

$$d_i^2 = [y_i - f(\mathbf{x}_i)]^2 \tag{4}$$

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

$$\begin{aligned} d^2 &= \sum_i [y_i - f(\mathbf{x}_i)]^2 \\ d^2 &= \sum_i [y_i - \sum_{j=1}^{m} \theta_j x_{ij} - \theta_0]^2 \\ d^2 &= \sum_i [y_i - \theta_1 x_{i1} - \theta_2 x_{i2} - \theta_3 x_{i3} - ... - \theta_m x_{im} - \theta_0]^2 \end{aligned} \tag{5}$$

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,

$$\begin{aligned} \frac{\partial}{\partial \theta_0}(d^2) &= -2 \sum_i [y_i - \theta_1 x_{i1} - \theta_2 x_{i2} - \theta_3 x_{i3} - ... - \theta_m x_{im} - \theta_0] = 0 \\ \frac{\partial}{\partial \theta_1}(d^2) &= -2 \sum_i [y_i - \theta_1 x_{i1} - \theta_2 x_{i2} - \theta_3 x_{i3} - ... - \theta_m x_{im} - \theta_0] x_{i1} = 0 \\ \frac{\partial}{\partial \theta_2}(d^2) &= -2 \sum_i [y_i - \theta_1 x_{i1} - \theta_2 x_{i2} - \theta_3 x_{i3} - ... - \theta_m x_{im} - \theta_0] x_{i2} = 0 \\ &\vdots \\ \frac{\partial}{\partial \theta_m}(d^2) &= -2 \sum_i [y_i - \theta_1 x_{i1} - \theta_2 x_{i2} - \theta_3 x_{i3} - ... - \theta_m x_{im} - \theta_0] x_{im} = 0 \end{aligned} \tag{6}$$

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

$$\begin{aligned}
\theta_0 n + \theta_1 \sum_{i=1}^{n} x_{i1} + \theta_2 \sum_{i=1}^{n} x_{i2} + ... + \theta_m \sum_{i=1}^{n} x_{im} &= \sum_{i=1}^{n} y_i \\
\theta_0 \sum_{i=1}^{n} x_{i1} + \theta_1 \sum_{i=1}^{n} x_{i1}^2 + \theta_2 \sum_{i=1}^{n} x_{i1}x_{i2} + ... + \theta_m \sum_{i=1}^{n} x_{i1}x_{im} &= \sum_{i=1}^{n} y_i x_1 \\
\theta_0 \sum_{i=1}^{n} x_{i2} + \theta_1 \sum_{i=1}^{n} x_{i1}x_{i2} + \theta_1 \sum_{i=1}^{n} x_{i2}^2 + ... + \theta_m \sum_{i=1}^{n} x_{i2}x_{im} &= \sum_{i=1}^{n} y_i x_2 \\
&\vdots \\
\theta_0 \sum_{i=1}^{n} x_{im} + \theta_1 \sum_{i=1}^{n} x_{i1}x_{im} + \theta_1 \sum_{i=1}^{n} x_{i2}x_{im} + ... + \theta_m \sum_{i=1}^{n} x_{im}^2 &= \sum_{i=1}^{n} y_i x_m \qquad (7)
\end{aligned}$$

This can be written in matrix form,

$$\begin{pmatrix} \sum_{i=1}^n y_i \\ \sum_{i=1}^n y_i x_i \\ \sum_{i=1}^n y_i x_2 \\ \vdots \\ \sum_{i=1}^n y_i x_m \end{pmatrix} = \begin{pmatrix} n & \sum_{i=1}^n x_{i1} & \sum_{i=1}^n x_{i2} \dots \sum_{i=1}^n x_{im} \\ \sum_{i=1}^n x_{i1} & \sum_{i=1}^n x_{i1}^2 & \sum_{i=1}^n x_{i1}x_{i2} \dots \sum_{i=1}^n x_{i1}x_{im} \\ \sum_{i=1}^n x_{i2} & \sum_{i=1}^n x_{i1}x_{i2} & \sum_{i=1}^n x_{i2}^2 \dots \sum_{i=1}^n x_{i2}x_{im} \\ \vdots & \vdots & \\ \sum_{i=1}^n x_{im} & \sum_{i=1}^n x_{i1}x_{im} & \sum_{i=1}^n x_{i2}x_{im} \dots \sum_{i=1}^n x_{im}^2 \end{pmatrix} \begin{pmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_m \end{pmatrix}$$

In simplified notation,

$$\begin{aligned}
\mathbf{Y} &= \mathbf{X}\theta \\
\mathbf{X^T Y} &= \mathbf{X^T X}\theta \\
\theta &= (\mathbf{X^T X})^{\mathbf{-1}}\mathbf{X^T Y} \qquad (8)
\end{aligned}$$

We obtained the optimal θ.

3.4 Gaussian Distribution

A one dimensional random variable x, which Gaussian distribution with mean μ and variance σ has a PDF,

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp(\frac{(x-\mu)^2}{2\sigma^2}) \qquad (9)$$

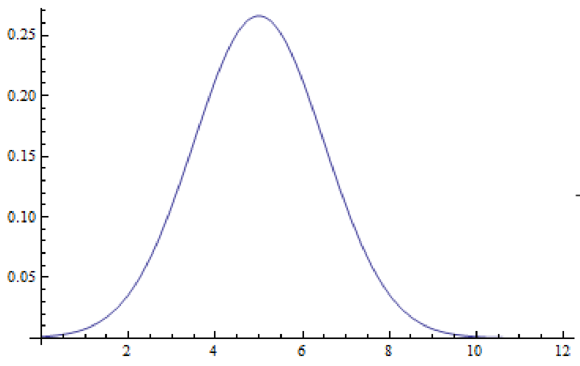

Figure 1: Univariate Gaussian

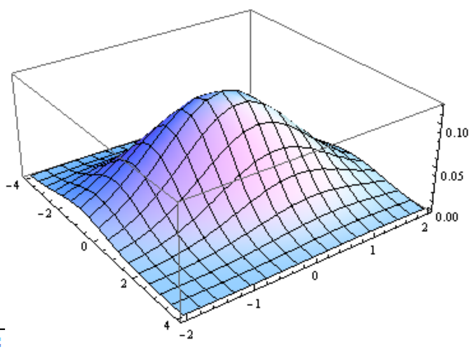

Figure 2: Multivariate Gaussian

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

Eqn. 8 can be written as,

$$p(x) \sim \mathcal{N}(\mu, \sigma^2) \tag{10}$$

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 $x_i, i \in m$ follows Gaussian distribution, we write the PDF over $\mathbf{x}$ as,

$$p(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^m |\Sigma|}} exp(-\frac{1}{2}(\mathbf{x} - \mu)^T \Sigma^{-1}(\mathbf{x} - \mu)) \tag{11}$$

Here $\mu = \{\mu_1, \mu_2, ..., \mu_m\}$ is a mean vector where each μ_i represents the mean of x_i where $i \in m$. Σ is a mXm 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. 10 as,

$$p(\mathbf{x}) \sim \mathcal{N}(\mu, \Sigma^2) \tag{12}$$

3.5 Bayesian Probability Model

The traditional linear regression method is rigid in terms of learning the parameters. It gives fixed values for parameters for one observed dataset. Therefore for every new observed datapoint, the parameters are bound to change. It would be effective to learn a probability distribution over the parameters rather than fixed values. Having a distribution over parameters gives us 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 to the data. Experiments generate evidences, which are 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.

3.5.1 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. 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 dependant on y). Therefore,

$$\begin{aligned} p(y,x) = p(y|x)p(x) &= p(x|y)p(y) \\ p(y|x) &= \frac{p(x|y)p(y)}{p(x)} \end{aligned} \tag{13}$$

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

$$p(x) = \sum_i p(x|y_i)p(y_i) \tag{14}$$

Combining Eqn. 8 and Eqn. 9,

$$p(y|x) = \frac{p(x|y)p(y)}{\sum_i p(x|y_i)p(y_i)} \tag{15}$$

In case y is a continuous variable,

$$p(y|x) = \frac{p(x|y)p(y)}{\int_y p(x|y)p(y)dy} \tag{16}$$

We start with the assumption that y_i differs from $f(\mathbf{x}_i)$ because of noise, also Each noise term ϵ_i follows Gaussian i.i.d. $\mathcal{N}(0, \sigma^2)$. Therefore the i^th^ can be written as,

$$\begin{aligned} y_i &= f(\mathbf{x}_i) + \epsilon_i \\ y_i &= \theta^T \mathbf{x}_i + \epsilon_i \qquad \text{where } \theta, \mathbf{x}_i \in \mathcal{R}^m \end{aligned} \tag{17}$$

The PDF associated with ϵ_i is,

$$p(\epsilon_i) = \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{\epsilon^2}{2\sigma^2}) \tag{18}$$

It follows from Eqn. 9 that $(y_i - \theta^T x_i) \sim \mathcal{N}(0, \sigma^2)$, so by combining Eqn. 9 and Eqn. 10,

$$p(y_i - \theta^T x_i) = \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{(y_i - \theta^T x_i)^2}{2\sigma^2}) \tag{19}$$

From Eqn. 11 we conclude,

$$\begin{aligned} p(y_i - \theta^T \mathbf{x}_i) &\sim \mathcal{N}(0, \sigma^2) \\ p(y_i | \mathbf{x}_i, \theta) &\sim \mathcal{N}(\theta^T \mathbf{x}_i, \sigma^2) \end{aligned}$$

We need to calculate the distribution over θ based on observed dataset $S = \{\mathbf{x}_i, y_i\}_{i=1}^n$ consisting of n data points. The conditional PDF over θ given S would be,

$$p(\theta|S) = \frac{p(\theta)p(S|\theta)}{p(S)}$$

$$p(\theta|S) = \frac{p(\theta)p(S|\theta)}{\int p(\theta)p(S|\theta)d\theta}$$

$$p(\theta|S) = \frac{p(\theta)\prod_{i=1}^n p(y_i|\mathbf{x}_i, \theta)}{\int p(\theta)\prod_{i=1}^n p(y_i|\mathbf{x}_i, \theta)d\theta}$$

We need to predict y_* for the next unobserved data point $S_* = y_*, x_*$ for some x_* given S which can be formulated as below,

$$p(y_*|x_*, S) = \int p(y_*|x_*, \theta)p(\theta|S)d\theta \tag{20}$$

3.6 Gaussian Process

The problem with the multivariate Gaussian distribution is that we are confined to the number of operating variables or the dimension of the input $\mathbf{x_i} \in \mathcal{R}^m$. Therefore to calculate the posterior distribution for y_*, we had to go for a two step solution. Firstly we learned a multivariate Gaussian posterior $p(\theta|S)$ over the set of parameters $\theta \in \mathcal{R}^m$ and secondly used it to calculate the posterior over y_*.

It would be nice to have a mathematical framework that does not try to learn a finite set of parameters. Therefore it does not care about dimensions of input, thus truly represents an infinite dimensional multivariate Gaussian distribution. 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 $x_i = \{x_i, x_j\} \in \mathcal{R}^2$ becomes two dimensional. No matter how many different temperatures we consider, $x_i \in \mathcal{R}^m$ 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 m beforehand. Therefore we can make x_i represent various different temperatures at different months, weeks, days, hours and so forth.

Previously we showed different random variables over different co-ordinate axis to represent multivariate distribution. We cannot see beyond three dimensions, so to draw infinite dimensional distributions we do not represent one random variable on a single axis. For example X-axis does not represent any particular random variable in x_i anymore, rather the entire x_i. As we can fit infinite number of points on X-axis, we can represent infinite dimensions on it. If x_i is a three dimensional vector, we show three points on X-axis.

To represent the regression problem here, we have,

$$y_i = f(\mathbf{x}_i) + \epsilon_i \tag{21}$$

$\epsilon_i \sim (0, \sigma^2))$ 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 x_i. A continuous function can represent infinite number of points on it. Therefore f can fit over an infinite dimensional x_i. Thus Gaussian process gives us a multivariate distribution over functions.

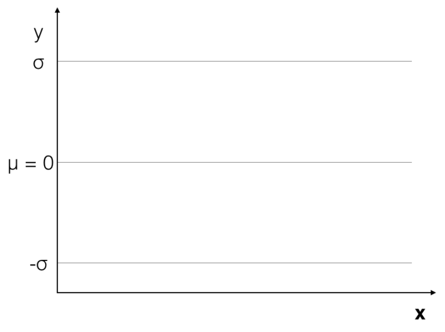

Figure 3: Prior with zero mean

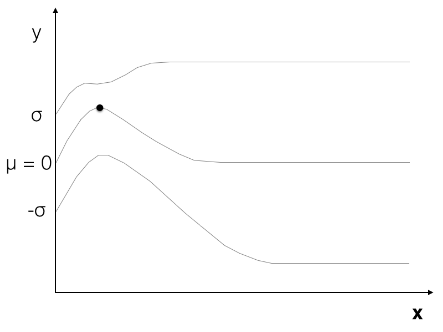

Figure 4: Posterior with one evidence

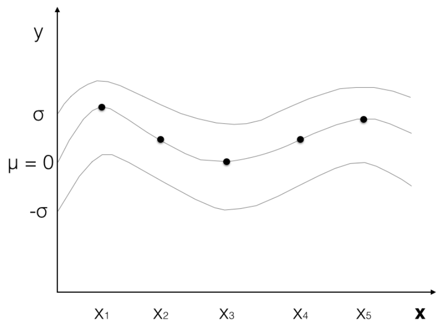

Figure 5: More evidences

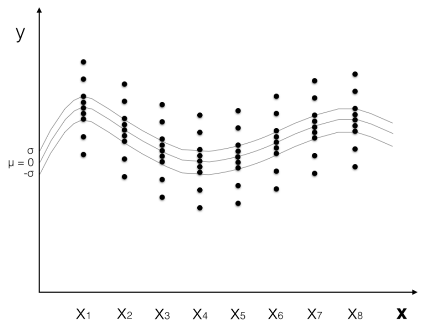

Figure 6: Learned function with entire training set

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

To differentiate a function from a regular random variable we denote a function f as $f(.)$ from now on. We start with a prior distribution over $\mathcal{F}$, given by,

$$f(.) \sim (m(.), k(., .)) \tag{22}$$

The prior is shown in Figure 1 with zero mean. Before getting evidences, the prior distribution has uniform confidence interval around the mean for the entire space. Figure 2 shows that after getting one evidence point the space around that point shrinked. We exclude all the functions that do not pass through our evidence point. The mean function has to pass through that point, therefore it shifted and the confidence interval also narrowed down accordingly. Figure 3 shows the same effect for more evidence points. Figure 4 finally shows the shape of the posterior function when we have enough evidences.

As we are operating on a distribution over functions, the mean $m(.)$ and covariance matrix $k(., .)$ are also functions. The domain of $f(.)$ is a vector $x_i \in \mathcal{R}^M$ where $M \to \infty$. $m(.)$ is the mean function which gives the mean of any random variable $x_{ip}(x_{ip} \in \mathbf{x}_i, p \in M)$,

$$m(x_{ip}) = \mu_{ip}$$

$k(., .)$ takes two random variables, x_{ip} and x_{iq} and gives the covariance between them as,

$$k(x_{ip}, x_{iq}) = Covariance(x_{ip}, x_{ip})$$

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

To represent the multivariate distribution over functions operating over any finite subset $\mathbf{x}_i^m \in \mathbf{x}_i$ we can write in matrix form as,

$$\begin{pmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_m) \end{pmatrix} \sim \left(\begin{pmatrix} m(x_1) \\ m(x_2) \\ \vdots \\ m(x_m) \end{pmatrix}, \begin{pmatrix} k(x_1,x_1)k(x_1,x_2)\ldots k(x_1,x_m) \\ k(x_2,x_1)k(x_2,x_2)\ldots k(x_2,x_m) \\ \vdots \\ k(x_m,x_1)k(x_m,x_2)\ldots k(x_m,x_m) \end{pmatrix} \right)$$

Which can be written as,

$$f(\mathbf{x}_m) \sim (m(\mathbf{x}_m), K(\mathbf{x}_m)) \tag{23}$$

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,

$$f(.) \sim (0, k(.,.)) \tag{24}$$

Combining Eqn. 18 and Eqn. 19 we get,

$$f(\mathbf{x}_m) \sim (0, K(\mathbf{x}_m)) \tag{25}$$

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

$$f(\mathbf{x}_m^*) \sim (0, K(\mathbf{x}_m^*)) \tag{26}$$

We combine Eqn. 20 and Eqn. 21 together as,

$$\begin{pmatrix} f(\mathbf{x}_m) \\ f(\mathbf{x}_m^*) \end{pmatrix} \sim \left(0, \begin{pmatrix} K(\mathbf{x}_m, \mathbf{x}_m) & K(\mathbf{x}_m, \mathbf{x}_m^*) \\ K(\mathbf{x}_m^*, \mathbf{x}_m) & K(\mathbf{x}_m^*, \mathbf{x}_m^*) \end{pmatrix} \right) \tag{27}$$

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,

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

$$\begin{pmatrix}\mathbf{y} \\ \mathbf{y}^*\end{pmatrix}\Bigg|_{\mathbf{x}_m, \mathbf{x}_m^*} = \begin{pmatrix}f(.) \\ f(.)^*\end{pmatrix} + \begin{pmatrix}\varepsilon \\ \varepsilon^*\end{pmatrix} \sim \left(0, \begin{pmatrix}K(\mathbf{x}_m, \mathbf{x}_m) + \sigma^2\mathbf{I} & K(\mathbf{x}_m, \mathbf{x}_m^*) \\ K(\mathbf{x}_m^*, \mathbf{x}_m) & K(\mathbf{x}_m^*, \mathbf{x}_m^*) + \sigma^2\mathbf{I}\end{pmatrix}\right) \quad (28)$$

Using Bayesian posterior calculation it can be shown that,

$$\mathbf{y_m}^*|\mathbf{y_m}, \mathbf{x}_m, \mathbf{x}_m^* \sim (\mu^*, \Sigma^*) \quad (29)$$

This is the function represented by Figure 4.

where,

$$\mu^* = K(\mathbf{x}_m^*, \mathbf{x}_m)(K(\mathbf{x}_m, \mathbf{x}_m) + \sigma^2\mathbf{I})^{-1}\mathbf{y_m} \quad (30)$$

and,

$$\Sigma^* = K(\mathbf{x}_m^*, \mathbf{x}_m^*) + \sigma^2\mathbf{I} - K(\mathbf{x}_m^*, \mathbf{x}_m)(K(\mathbf{x}_m, \mathbf{x}_m) + \sigma^2\mathbf{I})^{-1}K(\mathbf{x}_m, \mathbf{x}_m^*) \quad (31)$$

3.7 Upper Confidence Bound

...

3.7.1 Applying Gaussian Process to Bandit Problems

Gaussian process regression is one of the most suitable approach for solving multi armed bandit problems. We represent the actions A as our input space and we want to learn a function that gives mean rewards μ_{a_i} for each a_i. We can have an infinitely large input space and gaussian process easily fits that with reward function. 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 ovservations so far and then choosing the next action, we can nearly estimate the actual reward function very quickly. This requires some optimization 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.

3.7.2 Contextual Bandit

...

3.8 Neural Networks

...

3.8.1 Multi-Layered Neural Network

...

4 Dissecting an Image

...

4.1 Extract objects from an Image

...

4.2 Extract color from an image

...

4.3 Combining objects and color features

...

5 CPU vs GPU

...

5.1 Why CPU is not enough

...

5.2 How does GPU help

...

6 Programming architecture of GPU

...

7 System design

...

7.1 On CPU

...

7.2 On GPU

...

8 Experiments and Results

...

8.1 System setup

...

8.2 Experiments

...

8.3 Performance evaluation on CPU

...

8.4 Performance evaluation On GPU

...

8.5 Performance comparison on CPU and GPU

...

9 User Interaction and Usability

Design Principles -

Gestalt Laws

Norman’s principles

Shneiderman’s Golden Rules

Evaluation Techniques -

GOMS

KLM

Fitts Law

References

HWdR11 Hofmann, K., Whiteson, S. and de Rijke, M., Contextual bandits for information retrieval. *NIPS 2011 Workshop on Bayesian Optimization, Experimental Design, and Bandits*, 2011.

R.A95 R.Agrawal, 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.

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

Appendix 1. Model ABC

The appendices here are just models of the table of contents and the presentation. Each appendix usually starts on its own page, with the name and number of the appendix at the top. Each appendix is paginated separately.

In addition to complementing the main document, each appendix is also its own, independent entity. This means that an appendix cannot be just an image or a piece of programming, but the appendix must explain its contents and meaning.