Optimal Design of Experiment for Adaptive Sampling

1 Objectives of Tutorial

After reading this tutorial, we hope that you:

1. Gain insight and deeper understanding of material covered in the lecture
2. Understand the overall adaptive sampling pipeline and various options available to an adaptive sampler designer
3. Gain intuition about the explore-exploit problem

This tutorial in-part mirrors the structure of algorithm architecture we presented in the lecture, and aims to present formulas that may be helpful for the completion of the problem set (PSet). Additional information is made available to you in order to complete the information delivery started by the lecture. For any questions or clarifications on this document, please contact [vpreston, jetodd] at mit.edu.

2 Introduction

Adaptive sampling is a fairly far-reaching set of concepts with implications in seemingly disparate research communities: active learning, SLAM, informative path planning, passive irrevocable sampling, multi-armed bandit optimization, and others. For the purposes of this tutorial we will be drawing on two of these communities: multi-armed bandit optimization and informative path planning.

The multi-armed bandit problem, first given serious attention in the 1950s, draws from the image of a slots player at a row of slot machines (one-armed bandits colloquially) who must decide how to play the row to (which machines to select, and in what order) to maximize his/her winnings. The general approach for these "bandit" problems is to trade-off exploration (trying out new machines in the row) and exploitation (playing high-earning machines over-and-over). In adaptive sampling, one might consider this a network sensor selection problem: imagine temperature sensors distributed around a building. It is too expensive and computationally challenging to monitor all the sensors at once, so we would like to select sensors which will both give us a sense of the building's temperature, and target rooms which are too warm and thus will require someone to turn on the cooling system.

This trade-off between exploration and exploitation is also found in adaptive sampling literature in robotics, labeled as "informative path planning" which is the foundation of work related to robotic navigation through an unknown environment in order to characterize an underlying function of a target variable of interest. Some examples of this include active-SLAM [1], which attempts to optimize map building by choosing strategic locations and poses in an *a priori* unknown environment in order to build the best spatial map. Another canonical example would be a robot seeking to find and characterize a variable source (imagine an odor plume or an oil spill) in a convex environment in which the distribution of the target of interest is a priori unknown. Informative path planning algorithms consider the trade-off between exploration and exploitation; the idea being that in order to find the "best" areas to take samples from, the robot needs to move between local maxima by exploring the rest of the world which may have a lower reward.

By uniting concepts in multi-bandit literature to some approaches of informative path planning, we can discuss this exploration-exploitation trade-off rigorously and characterize robot performance using information-theoretic methods. For this tutorial, we will first walk you through the fundamental elements of the informative path planning problem: belief building, characterizing reward, and navigation schemes. Then, we will present two algorithms in the context of foundational literature in both multi-bandit planning and informative path planning. We will conclude with a discussion of practical implications of this field of research and current open areas.

3 World Modeling: Building Belief

To perform informative path planning it is necessary to create some representation of the environment in which you are operating. This is called your *world model* and it is important to realize that in the case of adaptive sampling, your world model is both your input to the sampler, and the overall goal of the sampler. We are continuously generating a plan based on this world model, and as new information is collected by the sensors of our system, this world model improves, with the goal to achieve the most suitable model for your mission.

There are many ways to formulate this world model. Bayesian nets are sometimes used when your planner is being used to perform a sequence of tasks, each possible state of the system represented as nodes in your net over which to plan. In the case of SLAM formulation, where our planner is trying to generate a good map of the area, this generally requires the construction of an occupancy grid, representing the environment as a grid of cells defined as either occupied (impassable) or unoccupied (passable). Sensor data is used to define the state of these cells.

This state space can be paired with a probabilistic framework in order to estimate state values at unobserved states. A key challenge in motion planning with high uncertainty (either noisy sensor measurements or partially observed state vector) is that it is often the case that a robot cannot observe its current state directly and must instead estimate a distribution over the set of all possible states based on sensor measurements. The decision process based on these probabilities and a set of observations is referred to as the Partially Observable Markov Decision Process (POMDP).

In the case of a mapping problem, such as that described in [2] in which maps are constructed by classifying some sensor input, it is computationally intractable to sample every point in your map. Rather a sparse sample can be taken and a *belief* space can be constructed such that you can predict the value of a potential future measurement at an unsampled point in the map.

This state space over which your path planning operates is referred to as the *belief state* or *belief model*, and is essentially a state of probability distributions which are used to infer values at unobserved states. There are numerous ways to construct this belief state. We invite the inquisitive reader to read through the literature provided at the end of this tutorial for more information on alternative belief models. For this tutorial we will be focusing on the popular Gaussian Process (GP) for constructing a belief model.

3.1 Gaussian Processes

Recall, the Gaussian Process is a generalization of the Gaussian probability distribution. A Gaussian process generates data located throughout some domain such that any finite subset of the range follows a multivariate Gaussian distribution. A Gaussian Process $GP(\mu(x), k(x, x')$ is defined by its mean $\mu(x)$ and its covariance $k(x, x')$. We can construct a GP belief model based on some initial sparse sample set for our domain and thus predict unobserved values between these sample points based on the probability distributions.

In many adaptive sampling cases it is assumed that the mean of this GP which we partner to the data is zero everywhere, therefore what relates one observation to another is the covariance function $k(x, x')$. The reliability of a given GP in predicting unsampled states is entirely dependent on the choice of covariance function, or kernel. The adaptive sampling examples discussed below utilize two of the more popular kernels.

- The Matérn class of covariance functions is given by

$$k(\mathbf{x}, \mathbf{x}') = \left(\frac{2^{1-\nu}}{\Gamma(\nu)}\right) r^{\nu} B_{\nu}(r) \tag{1}$$

  where $r = \left(\frac{\sqrt{2\nu}}{l}\right) \|\mathbf{x} - \mathbf{x}'\|$ where ν controls the smoothness of the sample paths and B_ν is a modified Bessel function

- The squared exponential covariance function is given by

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 exp\left(-\frac{1}{2l^2}\|\mathbf{x} - \mathbf{x}'\|^2\right) \tag{2}$$

  where l is the length scale that determines the strength of correlation between two points.

Our problem set also examines the use of the smoothing Radial Basis Function (RBF) kernel, described as

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right) \tag{3}$$

where $\|\mathbf{x} - \mathbf{x}'\|^2$ is the squared Euclidean distance and $\frac{-1}{2\sigma^2}$ sets the spread of the kernel.

3.2 Updating the Belief - Bayes Theorem

The important thing to note is that this belief model does not stay fixed. As the vehicle executes an action (see section 3) the belief model now needs to incorporate new data based on this action. In the case of habitat mapping, once the robot moves to a new location it will take some observation of its surroundings, and so the belief model needs to be updated to take this new observed information into account. For Gaussian processes, once the Gaussian distributions have been constructed, they can be propagated as new information is attained using a Bayes filter. In the case of activeSLAM, as the robot takes new observations, the occupancy grid can be updated using a Bayesian filter called an extended Kalman filter (EKF).

A Bayesian filter is a probabilistic state estimation framework that updates the vehicle's posterior belief based on its prior belief. Bayes Theorem is given by

$$P(H|D) = \frac{P(D|H)P(H)}{P(D)} \tag{4}$$

where $P(H)$ is the prior belief, $P(H|D)$ is the posterior belief to be calculated and $P(D|H)$ is the *likelihood* function, i.e. the probability of the data assuming the probability hypothesis is true,. Note, the likelihood function is not a probability function.

For the Gaussian Process, we have

$$P(f|D) = \frac{p(f)p(D|f)}{p(D)} \tag{5}$$

where the Gaussian process defines a distribution over functions $p(f)$. Essentially we have a known distribution, and we are making some prediction of the future state based on this known distribution. Once we incorporate new data we must updated this prediction of future states with this new data in mind.

Please Note! In practice (i.e. in code implementation) generally one is not expected to perform this Bayesian update by hand. For the PSet, we strongly recommend that you look into the API for the GP model function. *Hint:* It may be better to retrain a model from scratch.

For more information about GPs, please see References [3–5].

4 Information as Reward

Now that we have established our world model and a method of updating this model, we come to the central process of adaptive sampling. The purpose of incorporating adaptive sampling into an informative planning algorithm is to direct the sampling of your world model as you seek to improve the model. Instead of randomly sampling locations within your world model, you are trying to select new states for your system to sample based on which states will be most rewarding to the overall mission goal, without exceeding the constraints of the system. In the case of a habitat mapping scenario, you may be trying to gather a diverse set of information about a variety of habitats, so you want to go to areas of interest (or high information gain) within your world model.

As described in the introduction, adaptive sampling has wide-reaching applications and the objectives differs in each case, however it can be broadly grouped into two major classes; *exploitative* sampling and *explorative* sampling.

4.1 Exploitative

In exploitative sampling, the information planner is selecting new sample points with the intention of focusing on high value regions of the world model that it knows will satisfy the mission goal. To put this in context, in the multi-armed bandit problem described in the introduction the goal is to maximize winnings, thus the sampler selects machines it knows will deliver high earnings every time.

There are numerous possible reward functions for exploitative algorithm, each depending on the context of your problem. Some examples include maximizing your expected value, seeking the highest mean value within your sample space, or following the highest probability of finding a high value target.

4.2 Explorative

Explorative adaptive sampling has the goal of gathering the highest possible coverage and thus reducing the uncertainty in the belief model. Your planner is trying to build a better understanding of the world model and maximize some information metric. A simple way of understanding this is to think about ActiveSLAM. Adaptive sampling is used by the robot in conjunction with a SLAM formulation to reduce the uncertainty in its own internal map by seeking out sample points with high uncertainty values.

Possible reward functions for explorative sampling include reducing the expected variance in your map, and minimizing entropy, where entropy is a measure of information gain. Lets review this concept in a little more detail.

4.2.1 Entropy as Information Gain

Information entropy is defined as the average amount of information produced at a given point x, calculated as

$$H = -\sum p(x)\log_2 p(x) \tag{6}$$

where x represents some point in our two dimensional world and $p(x)$ is our probabilistic value at x in our belief model. The higher your entropy value, the more uncertain you are about the given point. Therefore in the context of an informative planning problem, we want to seek out points in our map that will help us reduce our overall entropy (i.e. reduce the uncertainty we have in our map). This entropy formulation is the method implemented in the problem set.

4.3 Trade-off

Many adaptive sampling problems actually aim to find a balance between these two goals, a trade-off between exploring the world model and exploiting our reward metric. To do this, our reward function becomes an addition between both our exploitation reward and some weighted exploration reward function, weighted using parameter β. This β term can be fixed over the course of your mission, or may be time-dependent so the priority of your mission changes with time, shifting from exploitative to exploration.

A popular trade-off relationship is the Upper Confidence Bound (UCB) rule, which clearly illustrates this relative weighting of the respective reward functions

4.3.1 Upper Confidence Bound

The UCB reward rule is given by

$$\boldsymbol{x}_t = \arg\max_{\boldsymbol{x}\in D}\left[\mu_{t-1}(\boldsymbol{x}) + \beta_t^{1/2}\sigma_{t-1}(\boldsymbol{x})\right] \tag{7}$$

Where

- x represents the current state of the system
- $\mu_{t-1}(\boldsymbol{x})$ represents the projected reward (mean) of a position $\boldsymbol{x}$ based upon the current model

- $\sigma_{t-1}(\boldsymbol{x})$ represents the projected uncertainty of a position $\boldsymbol{x}$ based upon the current model
- β_t is a time-varying parameter with the definition $\beta_t = 2log(\|D\|\pi_t/\delta)$ where $\sum_{t\geq 1}\pi_t^{-1} = 1, \pi_t > 0$
- D is the set of all sensor entities in the world and $\delta \in (0, 1)$

Notice the very specific definition of β. We invite the curious reader to check out the literature referenced [6] for the detailed proof about why this entity is defined this way (hint: it has to do with some specific properties of Gaussian distributions), but it is important to note the following: β increases with time. As such, we can interpret the UCB decision rule as something which actually favors exploration (gaining uncertainty reward) more as time passes.

5 Navigation

In informative path planning it is not sufficient to simply define a reward function and allow your system to propagate to satisfy that reward. The constraints of your system must also be taken into consideration. Informative path planners must select the new point in their sample space whilst ensuring it satisfies the system's budget constraints (fuel, time, energy etc). The cost of moving to a new sample point must be considered. Essentially you are trying to solve this objective

$$\arg\max_{\boldsymbol{r}\in S} I(r) \text{ s.t. } c(r) \leq B \tag{8}$$

where S is the set of all possible trajectories, B is your budget, $I(r)$ is the function representing the information you gather along the trajectory r and $c(r)$ is the cost of that trajectory.

There are numerous ways to represent actions and their costs. In the problem set we focus on the simplest case, where your actions are represented by a set of primitive actions based on the dynamics of our robot (we can move forward, backwards, left and right). We take the cost to be simply that of the euclidean distance between the current pose of the vehicle and the location of the sample site. This also assumes each action is discrete, i.e. we are sampling at waypoints along our trajectory. Computationally this is much simpler than continuously sampling along your trajectory.

While our problem set presents a highly simplified problem, in reality an informative path planner normally operates over a complex world model and is required to search through a large set of possible trajectories when selecting the most rewarding. The Rapidly-exploring Information Gathering (RIG) algorithm is an incremental sampling-based motion planning algorithm which can be used to generate these optimized trajectories when attempting to maximize some reward metric in your planning [7] .

5.1 Rapidly-exploring Information Gathering

You may be familiar with Rapidly-exploring Random Tree (RRT) as a search algorithm that can generate open-loop trajectories by quickly and randomly sampling a search space and building a branched tree. RIG combines concepts of RRTs with branch and bound optimization in order to both minimize cost and maximize some reward metric. The belief space of the system is sampled and each potential trajectory is extended towards those sample points to build up a set of all possible trajectories.

Breaking that down, from a starting node, points are randomly sampled in your belief space and for each of these points, all nearby nodes are extended towards this new point to create a graph (set of possible trajectories). The reward and cost at each new node are calculated and the reward and cost are updated along the new edges of your graph, generating new nodes to store these updated reward and cost values. New nodes are generated at the neighbors of these newly added nodes and this process continues until all eligible nodes are expanded. Optionally you can incorporate pruning of the graph to limit the number of nodes generated. The resulting graph should present possible trajectories that satisfy the budget constraints of your system.

Unlike other path planning problems, there is no fixed destination in informative path planning but rather the goal is to maximize some reward metric whilst being constrained by your budget. Your budget often determines the stopping criteria for your planning, as the system will continue to seek reward until the budget is exhausted. To optimize the trajectory generation in RIG, any candidate trajectory that will exceed that budget is never extended, we only explore completed trajectories.

For interested students, we have included the basic algorithm structure for the RIG below.

```
Algorithm 1: Rapidly-exploring Information Gathering (RIG)
  Input: Input: Step size $\Delta$; Budget $B$; Workspace $\mathcal{X}_{all}$, Free space $\mathcal{X}_{free}$, World $\varepsilon$, Starting state $\mathbf{x}_{start}$
  %Initialize cost, information and starting node
  $I_{init} \leftarrow InitialInformation(\mathbf{x}_{start}, \varepsilon$
  $C_{init} \leftarrow 0$, $n \leftarrow (\mathbf{x}_s tart, C_{init}, I_{init})$
  % Initialize vertex list, edge list and graph
  $V \leftarrow \{n\}, V_{closed} \leftarrow \emptyset, E_{closed} \leftarrow \emptyset, \mathcal{G} \leftarrow (V, E)$
  while processing time remains do
      % Sample configuration space of system
      $\mathbf{x}_{samp} \leftarrow Sample(\mathcal{X}_{all})$
      % Find near points to be extended
      $N_{near} \leftarrow Near(\mathbf{x}_{samp}, V \setminus V_{closed})$
      for all $n_{near} \in N_{near}$ do
          % Extend towards new point
          $\mathbf{x}_{new} \leftarrow Steer(\mathbf{x}_{near}, \mathbf{x}_{samp}, \Delta)$
          if NoCollision($\mathbf{x}_{near}, \mathbf{x}_{new}, \mathcal{X}_{free}$ then
              % Calculate new information and cost
              $I_{new} \leftarrow Information(I_{n_{near}}, \mathbf{x}_{new}, \varepsilon)$
              $C(\mathbf{x}_{new}) \leftarrow EvaluateCost(\mathbf{x}_{n_{near}}, \mathbf{x}_{new})$
              $C_{new} \leftarrow C_{n_{near}} + c(\mathbf{x}_{new})$
              $n_{new} \leftarrow (\mathbf{x}_{new}, C_{new}, I_{new})$
              if PRUNE($n_{new}$) then
                  Delete $n_{new}$
              else
                  % Add edges and vertex
                  $E \leftarrow E \cup \{(n_{new}, n_{near}), (n_{near}, n_{new})\}$
                  $V \leftarrow V \cup \{n_{new}\}$
                  % Add to closed list if budget exceeded
                  if $C_{new} > B$ then
                      $V_{closed} \leftarrow V_{closed} \cup \{n_{new}\}$
                  end
                  $\mathcal{G} \leftarrow (V, E)$
                  % Recursively add nodes along new edges
                  $\mathcal{G} \leftarrow UpdateInfoAndCost(n_{new}, \mathcal{G})$
              end
          end
      end
  end
  $r \leftarrow MaxInformationPlan(\mathcal{G})$
```

6 A Closer Look: Myopic Waypoint Selection

Now that you've gotten a sense of the core of informative path planning, let's put it all together in one of the most classic applications of informative path planning: myopic greedy waypoint selection.

Myopic: a "short-sighted" plan; in practice a one-step look-ahead

Greedy: the highest valued action is selected

In this scheme, "local" decisions are made, and multiple local decisions form a "global" plan. Let's take a closer look at two key pieces of literature which discuss this type of planning.

6.1 Network Sensor Selection

Within the context of the multi-armed bandit problem, Srinivas et al. [6] present a myopic-greedy algorithm for sensor selection in a network in their paper *Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design.* The formulation of the problem they use is as follows:

- GP belief model for the underlying function in which the sensor network is distributed. They demonstrate that their approach works with linear, radial-basis (RBF) and Matérn kernels.
- The UCB decision (reward) rule of the form $\boldsymbol{x_t} = \arg\max_{\boldsymbol{x} \in D} \mu_{t-1}(\boldsymbol{x}) + \beta_t^{1/2} \sigma_{t-1}(\boldsymbol{x})$ where $\beta_t = 2log(\|D\| \pi_t / \delta)$ and $\sum_{t \geq 1} \pi_t^{-1} = 1, \pi_t > 0$
- No navigation scheme (in a sensor selection problem like this, any sensor can be sampled at any time)

Thus, the algorithm which is implemented in this paper is as follows:

```
Algorithm 2: GP-UCB Algorithm
  Input: Input space $D$; GP Prior $\mu_0 = 0, \sigma_0, k$
  for $t = 1, 2, \ldots$ do
      Choose $\boldsymbol{x_t} = \arg\max_{\boldsymbol{x} \in D} \mu_{t-1}(\boldsymbol{x}) + \beta_t^{1/2} \sigma_{t-1}(\boldsymbol{x})$
      Sample $y_t = f(\boldsymbol{x_t}) + \epsilon_t$
      Perform Bayesian update to obtain $\mu_t$ and $\sigma_t$
  end
```

where y_t is the actual sensor reading from the function f at location $\boldsymbol{x_t}$ with noise ϵ_t. The beauty of the algorithm is its general simplicity; but the theoretical beauty of it comes when we consider it's performance. First, let's introduce the concept of regret.

Regret: the loss in reward between the optimal selections and the actual selections.

In this paper, the instantaneous regret, the regret of a single decision, is expressed as $r_t = f(\boldsymbol{x*}) - f(\boldsymbol{x_t})$ where $\boldsymbol{x*}$ is the optimal decision with perfect knowledge of the function f and the cumulative regret of all actions in time T is $R_t = \sum_{t=1}^{T} r_t$. For a planner to be considered no-regret, the asymptotic quality of the average cumulative regret should follow: $\lim_{T \to \infty} R_T / T = 0$.

When the UCB decision rule is used with the definition of β presented, then this algorithm has a no-regret quality. What's more, this definition of β allows for the finite average regret to be bounded based upon how much exploration is conducted. This makes some intuitive sense. Look at the regret definition closely: notice that the regret formulation values exploitation of high values – the optimal planner will always select for the highest valued sensor. But the UCB decision rule balances both exploitation and exploration. Exploration leads to better exploitation (generally) in time, but the act of exploration itself has a low-value. However, since the value of exploration is controlled in time by our β term, we can bound how well we exploit by how much we tend to explore.

This myopic greedy planner with performance guarantees has some attractive practical implications: there is some confidence by the user that the algorithm will perform within some expectation given some number of samples taken; and myopic greedy planners are computationally inexpensive, since only a one-step look-ahead is performed. Unfortunately, sensor selection is certainly a different problem than robot navigation: the ability to "jump" anywhere in the world is a nice luxury.

6.2 Extending to Paths

Luckily, Sun et al. [8] recently extended the work of Srinivas shown here into the informative path planning domain. In their paper *No-Regret Replanning under Uncertainty*, they formulate their problem as follows:

- GP belief model for underlying function in which the vehicle can traverse spatially where they consider a linear and squared exponential kernel.
- The UCB reward function of the same formulation as in Srinivas, expressed as $b_k = \{\mu_{t-1}(x^k_{t,j})\}^L_{j=0} + l\beta^{1/2}_t \sum^L_{j=0} \sigma_{t-1}(x^k_{t,j})$ where the notation follows that described below and l is the Lipschitz constant.
- A navigation scheme of pre-computed paths for each possible query location in the environment, where along each pre-computed path there are several measurement locations. This is notationally expressed as:
  - τ_i is the trajectory i from K pre-computed trajectories
  - L is the number of segments in trajectory τ_i
  - x_t is the robot's state at time t, and the trajectory starting from this point is $\{x^i_{t,0}, x^i_{t,1}, ..., x^i_{t,L}\}$ where $i \in \{1, 2...K\}$
  - at any time-step, the robot must pick the index of the best trajectory, I_t and execute that trajectory. At the end of the trajectory, replanning occurs.

Note that in their formulation they consider the fact that the underlying function is Lipshitz constant with respect to the l_1 norm with Lipschitz constant l. This is a critical part to their claim that their method of summing over points along a trajectory to get a trajectory's reward is valid in their proof that no-regret properties hold.

Their formulation for cumulative and instantaneous regret is the same as in Srinivas et al. but with the new notation applied. However, let's take a moment to consider the implication of this within the context of path planning. No-regret in this case means that as time goes on, the greedy selection of a pre-computed path made at x_t based upon the belief model, will eventually match the decision an optimal planner with complete knowledge would make. This makes some intuitive sense: the longer we explore/exploit an environment, the better our estimation of the underlying model should get, and the better the decisions should become. But this also means that we're only comparing the optimal planner to our planner *locally* - at each location that our actual planner selects; we are not comparing the actual global trajectory against the optimal global trajectory. While this is more-or-less accepted in the literature as being a satisfactory standard, there are potentially important ramifications when considering finite horizons or budgetary constraints which beg for global paths to be made as "optimal" as possible as a whole.

The algorithm applied takes this form:

Algorithm 3: UCB-Replanning Algorithm

```
Input: A library of $K$ trajectories $\{\tau_k\})k = 1^K$, sequence of parameters $\beta_t$, a GP $(\mu_0, \sigma_0)$ that models the
       function $v$ over the state space
for $t = 1$ to $T$ do
    for $k = 1$ to $K$ do
        Compute the sequence of means of belief $g$ on the points along trajectory $\tau_k$ as $\mu_{t-1}(x^k_{t,j})^L_{j=0}$
        Compute the sequence of standard deviations of belief $g$ on the points along the trajectory as
         $\sigma_{t-1}(x^k_{t,j})^L_{j=0}$
        Compute the UCB $b_k = \mu_{t-1}(x^k_{t,j})^L_{j=0} + l\beta_t^{1/2} \sum_{j=0}^{L} \sigma_{t-1}(x^k_{t,j})$
    end
    Choose index $I_t = \arg\max_{k \in \{1,\dots,K\}} b_k$ and execute trajectory $\tau_{I_k}$
    Observe samples and update to obtain $\mu_t$ and $\sigma_t$
end
```

Great! Now we've got an algorithm which has performance guarantees for pre-computed trajectories of multiple sampling points. This has some exciting theoretical and practical implications for myopic greedy planners for adaptive sampling. In the PSet, we ask you to implement something a little bit more simple. How would you modify this algorithm to consider only point-to-point sampling? What if you set a constant parameter to apply to the uncertainty term in the UCB-algorithm? Consider this during your work!

7 A Closer Look: Nonmyopic Action Selection

Rather than only consider local decisions, we can also consider nonmyopic greedy planners.

Nonmyopic: a "far-sighted" plan; in practice an n-step look-ahead where n is greater than 1

This could be considered a more "global" planning regime where even if greedy selections are still made, they are made greedily within the context of several-steps rather than one step. There are two variations on this theme of nonmyopic planning in the literature, respectively demonstrated by Hitz et al. [9] and Arora et al. [10].

7.1 Variations on a Global Plan

In the paper *Adaptive Continuous-space Informative Path Planning for Online Environmental Monitoring* Hitz et al. present an algorithm which creates an initially random global path in an unknown environment. As the robot begins to traverse the path, points along the global path are designated "replanning" points which allow for the global path to be altered locally to better utilize the information that has been gathered so far in the mission. The local manipulation is greedily selected in the context of the entire global plan. Their formulation can be summarized as follows:

- GP belief model for underlying function in which the vehicle can traverse spatially and the kernel used is the squared exponential.
- The UCB reward function as formulated by Srinivas but with constant β.

- Spline path representation (in which the parameters for the spline are adapted) and there are sampling points along the path. Note that at each local replanning step, an evolutionary algorithm, known as CMA-ES, is used to generate candidate splines which are randomly sampled to be evaluated by the UCB rule.

Hitz et al. also consider a budget for their plan – that is, they create a hard stopping point for the algorithm based upon the distance that the vehicle as traveled. This limits local replanning to feasible trajectories based upon remaining budget globally.

A simplified version of the algorithm developed in their paper is provided.

```
Algorithm 4: Local-Adaptation Nonmyopic Algorithm
Input: Global path $P_s$, planning horizon $h$, travel distance $d_{tr}$, budget factor $\lambda$, environment $\varepsilon$
Assign a local budget allowance based upon $P_s$, $h$, and $\lambda$ to apply towards $d_{tr}$
Initialize iteration count $i = 0$, spent budget $B_{spent} = 0$, initial local goal point along the global path $c_0$, and
  initial parameterization to $P_s$
while True do
    Update the remaining local budget for this iteration
    Set $c_i$ for the local planning along $P_s$ for this iteration
    Set $c_{i+1}$ for the next iteration
    From current position to $c_i$ call CMA-ES algorithm to generate possible local trajectories to $c_i$
    Select the best local trajectory based upon the UCB-criteria considering each local trajectory to $c_i$ and
      the global path to $c_{i+1}$
    Execute best trajectory
    Sample and update GP model
    Update $B_{spent}$
    Update $i+ = 1$
    if remaining local budget is depleted then
        break
    end
end
```

Through empirical results, they demonstrate through simulation that their approach generally performs well in recreating the underlying function of a variable of interest; however because of the nature of the random initialized plan, the unpredictability and local budget limitation one the spline evolutionary algorithm, and the use of a non-adaptive β term in their UCB rule, the performance can be "arbitrarily poor" at times. While for theoretical and some practical purposes this is not necessarily a problem, for high-stakes field deployments the risk of an arbitrarily poor result is unacceptable. Further, the budget setting, and the parameter searching for β can be difficult or non-trivial to intuitively set for all testing scenarios. This paper, however, it among one of few informative path planning papers which rigorously present field trials. Using a working marine surface vehicle to map algae and plankton blooms in a lake, the authors make some interesting scientific claims about a strain of plankton; a significant win for proponents of adaptive sampling algorithms as a useful tool for environmental inquiry.

7.2 Forward Simulation using Monte Carlo Tree Search

In the second type of nonmyopic planning found in literature, we see a small group of researchers using forward simulation techniques to plan long-horizons. We highlight here the work of Arora et al, however a number of papers are provided in the related works which further expound upon the concepts presented here and we invite the curious reader to check those out.

Arora et al. in their paper *Multi-Modal Active Perception for Information Gathering in Science Missions* are interested in performing geological environmental inquiry in an extraterrestrial world in which a ground vehicle must select the best sensor it is equipped with to take measurements, and choose where to navigate to next. To model this problem, they use an entirely different model for belief – a Bayes Net. Without getting into too much detail, a Bayes Net can represent states in a world, and overlay variable relationships onto states; these relationships being update-able using Bayesian techniques. What is perhaps particularly powerful about this choice of model is the ability to a priori encode inter-relatedness of different quantities in the environment (as in, they can encode the relationship between hardness of a rock specimen to other qualities like chemical composition).

Arora et al. have the same objective as any informative path planning problem: to maximize the expected informativeness of any action sequence. This action sequence, in their formulation, is nonmyopic; it includes several actions in a row in the plan. To chain individual actions together, they employ a Monte Carlo Tree Search (MCTS), which is a forward-simulation technique which uses back-propagation to allow a planner to select the most potentially rewarding branch (action) in the global scale.

Their algorithm takes the following form:

Algorithm 5: Nonmyopic MCTS Algorithm

```
Input: Sensing budget $S$, Belief space $Bel$, Domain knowledge Bayes net $K$, Remaining budget $R$
function MAIN is
    $R \leftarrow S$
    while $R > 0$ do
        $robotPose \leftarrow getLocalization()$
        $a_{opt} \leftarrow planner(robotPose, R, Bel, K)$
        $Z \leftarrow takeObservation(a_{opt})$
        $Bel \leftarrow updateBeliefSpace(Z, Bel, K)$
        $R \leftarrow R - cost(a_{opt})$
    end
end
function PLANNER($robotPose, R, Bel, K$) is
    $T \leftarrow initializeTree(robotPose, R)$
    $currentNode \leftarrow T.rootNode$
    while within computational budget do
        $currentNode \leftarrow treePolicy(T)$
        $sequence \leftarrow rolloutPolicy(currentNode, R)$
        $reward \leftarrow getReward(sequence, Bel, K)$
        $T \leftarrow updateTree(T, reward)$
    end
    return $bestChild(T)$
end
function ROLLOUTPOLICY(currentNode,R) is
    $sequence \leftarrow currentNode$
    while $R > 0$ do
        $nextNode \leftarrow defaultPolicy(currentNode)$
        $currentNode \leftarrow nextNode$
        $sequence \leftarrow sequence + currentNode$
        $R \leftarrow currentNode.R$
    end
    return $sequence$
end
function GETREWARD($sequence, B, K$) is
    $reward \leftarrow 0$
    for $i = 1 : length(sequence)$ do
        $currentAction \leftarrow sequence(i)$
        $Z = sampleObs(currentAction, Bel, K)$
        $Bel_{new} = updateBelief(Z, Bel, K)$
        $infoGain = calcInfoGain(Bel_{new}, Bel)$
        $reward \leftarrow rewrd + infoGain$
        $Bel \leftarrow Bel_{new}$
    end
    return $reward$
end
```

Let's break this down a little and talk about how the MCTS is operating here. In $MAIN$ we see that the robot's pose is taken, and then the MCTS is executed in $PLANNER$. Starting from the robot's position and belief, the MCTS branches to new locations (new nodes) by applying some action along the branch (the actions applied follow some policy which is referenced in the algorithm – this makes sense of the number of actions is perhaps extremely large, and some random sampling of these actions should be taken to prevent the branching factor from exploding). Until the computational limit is reached, the start node is expanded into several actions, each of those actions are also expanded following the same policy and so on. Each path is a sequence. forward simulation is done by following a sequence and applying the $GETREWARD$ function - based upon the current belief, each action in the sequence is used to calculate the reward of that potential action, and then update a simulated belief model as though what is the predicted outcome of that action is what happens in reality, and so on. Back propagation happens by assigning that sequence's reward to the initial action expansion.

MCTS is a particularly popular method for nonmyopic planning in a number of planning problems outside of adaptive sampling. The implementation here is powerful as it leverages the model that is used, a Bayes Net, to make computationally efficient long-term plans.

8 Final Thoughts

Both myopic and nonmyopic planners have their merit and have been demonstrated in theory and in the field for adaptive sampling regimes. However, not all is solved. In each of the core algorithmic areas and at the big-picture-level of the problem there are the following open areas:

Modeling: a GP model is great for targeting single value quantities, and quantities which have measurable gradients instead of hard fronts; but if we wanted to sample things with hard or binary fronts (like plankton blooms for instance) a GP may be inappropriate. What is more, if we wanted to encode a priori known information about the distribution about quantities related to the distribution (science-relationships for instance) this would be difficult in a GP. Work on Bayes nets was presented here, but overall has been a fairly small body of work. Sequential Bayesian optimization techniques, and simplifying assumptions for partially-observable Markov decision processes (POMDPs) to make them extensible for practical computation could be more fully explored to inspire other ways of optimizing for information gain.

All of the models presented here assume a static environment – but what if the environment changes in time and space? There is some work in GP modeling to develop spatio-temporal kernels which could capture changing phenomena, but this work has not yet been applied significantly to informative path planning problems.

Reward and Performance Metrics: We largely focused here on the implementation of the UCB reward rule, however a number of other functions exist including expected mean, variance reduction, probability of information, etc. There are not yet any rigorous proofs or empirical studies which convincingly demonstrate the performance of these other metrics in domain specific applications. We consider the definition of regret used in this tutorial: regret is generally an exploitative evaluation. But what it coverage is also important? Can one incur "coverage regret" or something similar? Exploring different reward formulations or regret formulations may lead to more interesting or varied behaviors desirable in different contexts.

Navigation and Path Optimization: Literature tends to focus on either getting from sampling location A to sampling location B, with limited work which considers the utility of sampling points between a initial and destination node (Hitz et al. and Sun et al. being exceptions presented here). There is some literature which actually formulates informative path planning as a traveling salesman problem, though generally little has been done to push the boundaries of optimizing paths in general.

Big Picture: For practical implementations, it is common practice to "budget" the exploration by setting an arbitrary amount of time or distance that the vehicle is allowed to operate (Hitz et al. presented here budgeted travel distance). While this is important in the case of considering physical limitations of the vehicle (battery life for instance), in reality, the aim of the mission is to characterize something well. For particularly conservative budgets, it may be the case that the robot develops a model that is "good enough" well before the termination point is met (imagine a robot exploring a lake for a chemical plume, only to find fairly quickly that the lake is "mixed" and there is no plume. We would certainly like the robot to come back and tell us that rather than continue exploring for nothing for several more hours). This is critical to consider in resource-limited deployments (where the very act of deploying is extremely expensive), or long-term multi-mission robotic systems such as extraterrestrial explorers which may need to transition between missions and characterize its overall performance without human input. "Stopping criteria" as it is known, is nearly completely unexplored in IPP with promising concepts in Bayesian Optimization literature which could be applied.

In general, adaptive sampling is a key requirement of using robotic explorers in a number of contexts; particularly environmental exploration and sampling to intelligently optimize over time towards a mission goal for the end-user. Open-loop exploration, like performing lawnmowing surveys, while useful, fails to exploit appropriately in a single deployment and is an inefficient use of time for a human user. Widely in practice today is the method of deploying a vehicle on a broad survey, which is then reviewed by a human user to identify interesting phenomenon, and then a new open-loop mission is replanned which surveys those interesting locations. If the robot is equipped however with informative path planning algorithms and a sense of what is "interesting" a single deployment may be sufficient. In the PSet we leave it as an exercise to reflect on and discuss ways in which adaptive sampling as presented here could become more robust.

9 Further Reading

References

[1] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age," *IEEE Trans. Robot.*, vol. 32, pp. 1309–1332, dec 2016.

[2] P. Rigby, O. Pizarro, and S. B. Williams, "Toward adaptive benthic habitat mapping using gaussian process classification," *J. F. Robot.*, vol. 27, pp. 741–758, nov 2010.

[3] M. Ebden, "Gaussian processes for regression: A quick introduction," *Website Robot. Res. Gr. . . .*, no. August, 2008.

[4] S. Patil, G. Kahn, M. Laskey, J. Schulman, K. Goldberg, and P. Abbeel, "Scaling up Gaussian belief space planning through covariance-free trajectory optimization and automatic differentiation," in *Springer Tracts Adv. Robot.*, vol. 107, pp. 515–533, 2015.

[5] J. Bloom and J. Orloff, "Bayesian Updating with Discrete Priors."

[6] N. Srinivas, A. Krause, S. M. Kakade, and M. W. Seeger, "Information-theoretic regret bounds for Gaussian process optimization in the bandit setting," *IEEE Trans. Inf. Theory*, vol. 58, no. 5, pp. 3250–3265, 2012.

[7] G. A. Hollinger and G. S. Sukhatme, “Sampling-based robotic information gathering algorithms,” *Int. J. Rob. Res.*, vol. 33, pp. 1271–1287, aug 2014.

[8] W. Sun, N. Sood, D. Dey, G. Ranade, S. Prakash, and A. Kapoor, “No-regret replanning under uncertainty,” in *Proc. - IEEE Int. Conf. Robot. Autom.*, pp. 6420–6427, sep 2017.

[9] G. Hitz, E. Galceran, M.-È. Garneau, F. Pomerleau, and R. Siegwart, “Adaptive continuous-space informative path planning for online environmental monitoring,” *J. F. Robot.*, vol. 34, pp. 1427–1449, dec 2017.

[10] A. Arora, P. M. Furlong, R. Fitch, S. Sukkarieh, and T. Fong, “Multi-Modal Active Perception for Information Gathering in Science Missions,” dec 2017.