Active Pointillistic Pattern Search

Yifei Ma Carnegie Mellon
Dougal J. Sutherland Carnegie Mellon
Roman Garnett University of Bonn
Jeff Schneider[*] Carnegie Mellon

Abstract

We introduce the problem of *active pointillistic pattern search* (APPS), which seeks to discover regions of a domain exhibiting desired behavior with limited observations. Unusually, the patterns we consider are defined by large-scale properties of an underlying function that we can only observe at a limited number of points. Given a description of the desired patterns (in the form of a classifier taking functional inputs), we sequentially decide where to query function values to identify as many regions matching the pattern as possible. We demonstrate the proposed algorithm by identifying vortices in a fluid flow simulation.

1 Introduction

Consider a function containing interesting patterns that are defined only over a region of space. For example, if you view the direction of wind as a function of geographical location, it defines fronts, vortices, and other weather patterns, but those patterns are defined only in the aggregate. If we can only measure the direction and strength of the wind at point locations, we then need to infer the presence of patterns over broader spatial regions. This bears some resemblance to the artistic technique known as pointillism, where the painter creates small and distinct dots each of a single color, but when viewed as a whole they reveal a scene. Pointillist paintings typically use a fairly dense covering of the canvas, but in our setting, "observing a dot" is expensive. Where should we make these observations in order to uncover interesting regions as quickly as possible?

We propose a probabilistic solution to this problem, known as *active pointillistic pattern search* (APPS). We assume we are given a predefined list of candidate regions and a classifier that estimates the probability that a given region fits the desired pattern. Our goal is then to find as many regions that are highly likely to match the pattern as we can. We accomplish this by sequentially selecting point locations to observe so as to approximately maximize expected reward.

1.1 Related Work

Our concept of active pattern search falls under the broad category of *active learning* [8], where we seek to sequentially build a training set to achieve some goal as fast as possible. Our focus solely on finding positive ("interesting") regions, rather than attempting to learn to discriminate accurately between positives and negatives, is similar to the problem previously described as *active search* [2]. In previous work on active search, it has been assumed that the labels of interest can be revealed directly. In active pattern search, on the other hand, the labels are never revealed but must be inferred via a provided classifier. This indirection increases the difficulty of the search task considerably.

In *Bayesian optimization* [1, 6], we seek to find the global optimum of an expensive black-box function. Bayesian optimization provides a model-based approach where a Gaussian process (GP) prior is placed on the objective function, from which a simpler acquisition function is derived and optimized to drive the selection procedure. In [10], the authors extend this idea to optimizing a latent function from binary observations. Our proposed active pattern search also uses a Gaussian process prior to model the unknown underlying function and derives an acquisition function from it, but differs in that we seek to identify entire *regions* of interest, rather than finding a single optimal value.

[*]{yifeim,dsutherl}@cs.cmu.edu, rgarnett@uni-bonn.de, schneide@cs.cmu.edu .

Level set estimation [3, 4], rather than finding optima of a function, seeks to select observations so as to best discriminate the portions of a function above and below a given threshold. This goal considers the point values of the function and cannot be used to find complex patterns.

APPS can be viewed as a generalization of *active area search* (AAS) [5], which is a considerably simpler version of active search for region-based labels. In AAS, the label of a region is only determined by whether its mean value exceeds some threshold. APPS allows for arbitrary classifiers rather than simple thresholds. This extends the usefulness of this class of algorithms considerably.

2 Problem Formulation

There are three key components of the APPS framework: a function f which maps input covariates to data observations, a predetermined set of regions wherein instances of function patterns are expected, and a classifier that evaluates the salience of the pattern of function values in each region. We define $f\colon \mathbb{R}^m \to \mathbb{R}$ to be the function of interest,[1] which can be observed at any location $x \in \mathbb{R}^m$ to reveal a noisy observation z. We assume the observation model $z = f(x) + \varepsilon$, where $\varepsilon \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2)$. We suppose that a set of regions where matching patterns might be found is predefined, and will denote these $\{g_1, \ldots, g_k\}$; $g_i \subset \mathbb{R}^m$. Finally, for each region g, we assume a classifier h_g which evaluates f on g and returns the probability that it matches the target pattern: $h_g(f) \in [0, 1]$, where the mathematical interpretation of h_g is similar to a functional of f. Classifier forms are typically the same for all regions, but with different parameters.

Unfortunately, in general, we will have little knowledge about f other than the limited observations made at our selected set of points. Classifiers which take functional inputs (such as our assumed h_g) generally do not account for uncertainty in their inputs, which should be inversely related to the number of observed data points. We thus must consider the probability that $h_g(f)$ is high enough, marginalized across the range of functions f that might match our observations. As is common in nonparametric Bayesian modeling, we model f with a Gaussian process (GP) prior; we assume hyperparameters, including prior mean and covariance functions, are set by domain experts. Given a dataset $\mathcal{D} = (X, z)$, we define

$$f \sim \mathcal{GP}(\mu, \kappa); \qquad f \mid \mathcal{D} \sim \mathcal{GP}(\mu_{f|\mathcal{D}}, \kappa_{f|\mathcal{D}}), \tag{1}$$

to be a given GP prior and its posterior conditioned on $\mathcal{D}$, respectively. Thus, since f is a random variable, we can obtain the marginal probability that g matches the desired pattern,

$$T_g(\mathcal{D}) = \mathbb{E}_f\big[h_g(f) \mid \mathcal{D}\big], \tag{2}$$

which then gives rise to the definition of a matching region as one whose marginal probability passes a given threshold θ (typically $\theta > 0.5$). A reward is assigned to each region g as an indicator function

$$r_g(\mathcal{D}) = \mathbb{1}\big\{T_g(\mathcal{D}) > \theta\big\}. \tag{3}$$

We make two assumptions regarding the interactive procedure. The first is that once a region is flagged as potentially matching (i.e., its marginal probability exceeds θ), it will be immediately flagged for further review and no longer considered during the run. The second is that the data resulting from this investigation will not be made immediately available during the course of the algorithm; rather the classifiers h_g will be trained offline. We consider both of these assumptions to be reasonable when the cost of investigation is relatively high and the investigation collects different types of data. For example, if the algorithm is being used to run autonomous sensors and scientists collect separate data to follow up on a matching region, these assumptions allow the autonomous sensors to continue in parallel with the human intervention, and avoid the substantial complexity of incorporating a completely different modality of data into the modeling process. Making different assumptions would lead to interesting extensions to our algorithms that we do not consider here.

Garnett et al. [2] attempt to maximize their reward at the end of a fixed number of queries. Directly optimizing that goal involves an exponential lookahead process. However, this can be approximated by a greedy search like the one we perform. Similarly, one could attempt to maximize the area under the recall curve through the search process. This also requires an intractable amount of computation which is often replaced with a greedy search.

[1]For clarity, in this and the next sections we will focus on scalar-valued functions f. The extension to vector-valued functions is straightforward, and indeed we will consider such cases in our experiments.

We now write down the greedy criterion our algorithm seeks to optimize. Define $\mathcal{D}_t$ to be data collected before time step t and $\mathcal{G}_t = \{g : T_g(\mathcal{D}_\tau) \leq \theta, \forall \tau \leq t\}$ to be the set of remaining search subjects; we aim to greedily maximize the sum of rewards over all the regions in $\mathcal{G}_t$ in expectation,

$$\max_{x_*} \mathbb{E} \sum\nolimits_{g \in \mathcal{G}_t} \big[r_g(\mathcal{D}_*) \mid x_*, \mathcal{D}_t \big], \tag{4}$$

where $\mathcal{D}_*$ is the (random) dataset at the next time step including a new unobserved data point.

3 Method

For the aim of maximizing the greedy expected reward of finding matching patterns, (4), a more careful examination of the GP model can yield a straight-forward sampling method. This method, in the following, turns out to be quite useful in APPS problems with rather complex classifiers.

At each step, given $\mathcal{D}_t = (X, z)$ as the set of any already collected (noisy) observations of f and x_* as any potential input location, we can assume the distribution of possible observations z_* as,

$$z_* \mid x_*, \mathcal{D}_t \sim \mathcal{N}\big(\mu_{f|\mathcal{D}_t}(x_*),\ \kappa_{f|\mathcal{D}_t}(x_*, x_*) + \sigma^2\big). \tag{5}$$

Conditioned on an observation value z_*, we can update our GP model to include the new observation (x_*, z_*), which further affects the marginal distribution of region classifier outputs and thus the probability that this region matches. With $\mathcal{D}_* = \mathcal{D}_t \cup \big\{(x_*, z_*)\big\}$ as the updated dataset, we use $r_g(\mathcal{D}_*)$ to be the updated reward of region g. The utility of this proposed location x_* for region g is thus measured by the *expected* reward function, marginalizing out the unknown observation value z_*:

$$u_g(x_*, \mathcal{D}_t) = \mathbb{E}_{z_*}\big[r_g(\mathcal{D}_*) \mid x_*, \mathcal{D}_t\big] = \Pr\big\{T_g(\mathcal{D}_*) > \theta \mid x_*, \mathcal{D}_t\big\}. \tag{6}$$

Finally, in APPS, we select the next observation location x_* by maximizing the sum of expected rewards over all remaining search (region) subjects, reproducing (4), by linearity of expectation:

$$x_* = \arg\max_x u(x, \mathcal{D}_t) = \arg\max_x \sum\nolimits_{g \in \mathcal{G}_t} u_g(x, \mathcal{D}_t). \tag{7}$$

For the most general definition of the region classifier h_g, the basic algorithm is to compute (6) and thus (7) via sampling at two stages:

1. Sample the outer variable z_* in (6) according to (5).
2. For every draw of z_*, sample enough of $(f \mid \mathcal{D}_*)$ to compute the marginal reward $T_g(\mathcal{D}_*)$ in (2), in order to obtain one sample for (6).

When $h_g(f) = \Phi(\langle w, f \rangle + b)$ is a generalized linear model with a probit link function, we can in fact compute both (2) and (6) analytically, allowing us to efficiently perform exact searches. If we further assume that the regions are independent (a nontrivial simplification of the true problem), we can also analyze how APPS trades off between the current marginal probability of salience and the amount of possible variance reduction of each region. We defer this analysis to a future version of the paper.

4 Empirical Evaluation

We now study the task of identifying vortices in a vector field based on limited observations of flow vectors. We consider the results of a large-scale simulation of a turbulent fluid in three dimensions over time in the Johns Hopkins Turbulence Databases[2] [7]. Following Sutherland et al. [9], we aim to recognize vortices in two-dimensional slices of the data at a single timestep, based on the same small training set of 11 vortices and 20 non-vortices, partially shown in Figure 1(a).

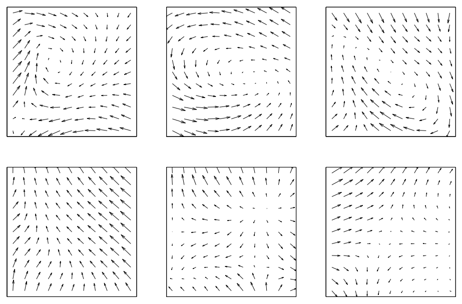
Recall that h_g assigns probability estimates to the entire function class $\mathcal{F}$ confined to region g. We consider the average flow across sectors (angular slices from the center) of our region as building blocks in detecting vortices. We count how many sectors have clockwise/counter-clockwise flows:

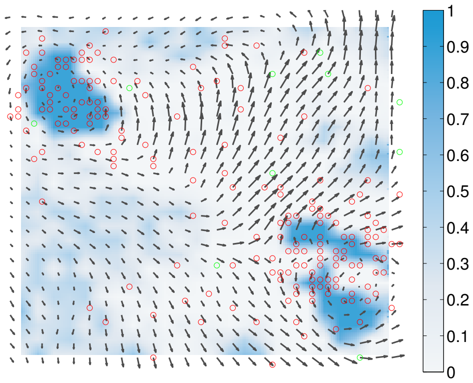
1. First, we divide a region into K sectors. In each sector, we take a linear transformation of the fluid flow vector as the integral of the inner product between the actual flow vectors and a template. The template is an "ideal" vortex, but with larger weights in the center than the periphery. This produces a K-dimensional summary statistic $L_g(f)$ for each region.

[2] http://turbulence.pha.jhu.edu

2. Next, we improve robustness against different flow speeds in the data by scaling $L_g(f)$ to have maximum-magnitude entry 1, and flip its sign if its mean is negative. Call the result $\tilde{L}_g(f)$.
3. Finally, we feed the normalized $\tilde{L}_g(f)$ vector through a 2-layer neural network of the form $h_g(f) = \sigma\left(w_{\text{out}} \sum_{i=1}^{K} \sigma\left(w_{\text{in}} \tilde{L}_g(f)_i + b_{\text{in}}\right) + b_{\text{out}}\right)$, where σ is the logistic sigmoid function.

$L_g(f) \mid \mathcal{D}$ obeys a K-dimensional multivariate normal distribution, from which we can sample many possible $L_g(f)$, which we then normalize and pass through the neural network as described above. This gives samples of probabilities h_g, whose mean is a Monte Carlo estimate of (2).


(a)



(b)

Figure 1: (a): Positive (top) and negative (bottom) training examples for the classifier. (b): The velocity field used; each arrow is the average of a 2×2 square of actual data points. Background color shows the probability obtained by each region classifier on the 200 circled points; red circles mark points selected by one run initialized at the green circles.

We used $K = 4$ sectors, and the weights in the template were fixed such that the length scale matches the distance from the center to an edge. The network was optimized for classification accuracy on the training set. We then identified a 50×50-pixel slice of the data that contains two vortices, some other "interesting" regions, and some "boring" regions, mostly overlapping with Figure 11 of [9]; the region is shown in Figure 1(b). We then ran APPS, initialized with 10 uniformly random points, for 200 steps. The velocity vectors f were modeled using a GP with a squared-exponential covariance of length scale 1.8 on each dimension (with the two dimensions independent of one another). We defined the regions as 11×11 squares and spaced them every 2 points along the grid, for 400 total regions. We thresholded at $\theta = 0.7$. We evaluated (2) via a Monte Carlo approximation: first we took 4 samples of z_*, and then 15 samples from the posterior of f over the window for each z_*. At each step we evaluated a random subset of 80 possible candidates x_*.

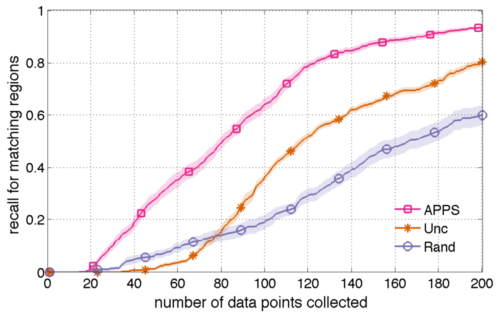


Figure 2: Recall for vortex regions (means and standard errors; 15 runs).

Figure 2 shows recall curves of active pattern search, uncertainty sampling (selecting observations to minimize variance in the underlying function f), and random selection, where we call the true label the output of the classifier when all data is known, and the proposed label is true if $T_g > \theta$ at that point of the search (evaluated using more Monte Carlo samples than in the search process, to gain assurance in our evaluation, but without increasing the time required for the search). We can see that active pattern search substantially outperforms uncertainty sampling and random selection. RAND was initially better than UNC, because RAND could be lucky whereas UNC initially attempts to cover the space without gaining detailed knowledge of any region. Later, RAND suffered from selecting redundant points. Precision is governed primarily by the classifier and θ, which is held constant for all methods.

References

[1] E. Brochu, V. M. Cora, and N. de Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning, 2010. `arXiv:1012.2599 [cs.LG]`.

[2] R. Garnett, Y. Krishnamurthy, X. Xiong, J. Schneider, and R. P. Mann. Bayesian Optimal Active Search and Surveying. In *ICML*, 2012.

[3] A. Gotovos, N. Casati, G. Hitz, and A. Krause. Active Learning for Level Set Estimation. In *IJCAI*, 2013.

[4] K. H. Low, J. Chen, J. M. Dolan, S. Chien, and D. R. Thompson. Decentralized active robotic exploration and mapping for probabilistic field classification in environmental sensing. AAMAS '12, 2012.

[5] Y. Ma, R. Garnett, and J. Schneider. Active Area Search via Bayesian Quadrature. In *AISTATS*, 2014.

[6] M. A. Osborne, R. Garnett, and S. J. Roberts. Gaussian Processes for Global Optimization. In *LION*, 2009.

[7] E. Perlman, R. Burns, Y. Li, and C. Meneveau. Data exploration of turbulence simulations using a database cluster. In *Proceedings of the 2007 ACM/IEEE Conference on Supercomupting*, 2007.

[8] B. Settles. *Active Learning*. Morgan & Claypool, 2012.

[9] D. J. Sutherland, L. Xiong, B. Póczos, and J. Schneider. Kernels on Sample Sets via Nonparametric Divergence Estimates, 2012. `arXiv:1202.0302 [cs.LG]`.

[10] M. Tesch, J. Schneider, and H. Choset. Expensive function optimization with stochastic binary outcomes. In *ICML*, 2013.