Gaussian Process Regression for Optimization

Daniel Lizotte **Tao Wang** **Michael Bowling** **Dale Schuurmans**
Department of Computing Science
University of Alberta
{dlizotte,trysi,bowling,dale}@cs.ualberta.ca

Abstract

We propose an optimization algorithm for non-convex, stochastic functions of several dimensions that are costly to evaluate. Such functions pose some of the most difficult optimization problems known; however, prior knowledge about the structure of the function can sometimes be leveraged to find acceptable solutions using a reasonable number of function evaluations. We show how structure can be encoded in an intuitive way using a Gaussian process model, and show how such a model can be used to direct the search for extrema of the function. We demonstrate the use of this technique on the problem of optimizing walk parameters for the Sony AIBO ERS-7, showing promising results using fewer function evaluations than other methods.

1 Introduction

Difficult (some might say 'hopeless') function optimization problems are at the heart of many (if not all) of the most interesting challenges in computing. Entire fields of study have emerged from the desire to solve these problems, and optimization algorithms with a wide variety of complexity, guarantees, and applicability are now known. Specifically, we are interested in non-convex, stochastic functions of several dimensions with computational cost on the order of seconds to minutes per function evaluation.

Theory [1] and practice suggest that we will never find a general, 'black-box' method that will work will in all cases. Therefore, when faced with these (near) 'hopeless' problems, it seems our best hope lies in using as much prior knowledge as possible about the problem at hand, while allowing our prior beliefs to be modified by observed data as they arrive. We propose a Bayesian method using Gaussian process regression [2, 3] that fits criteria we were concerned with: It is data-efficient, and it provides an intuitive mechanism for incorporating prior knowledge of the function to be optimized.

One challenging optimization problem that has gained attention in recent years is the task of maximizing forward velocity over the 'walk-parameter' space of the Sony AIBO robot dog. The optimization techniques applied to date in this space have, to our knowledge, been confined to various flavours of hill-climbing [4, 5] and genetic algorithms [6]. We show results that indicate our technique can achieve similar results to previous attempts using fewer function evaluations.

2 Background

Our proposed method is based on using Gaussian process regression on observed data points as they arrive.

Gaussian Process A *Gaussian process* (GP) [2, 7] is a (possibly uncountably infinite) collection of random variables for which any finite subset of the variables has a joint Gaussian distribution. The variables are typically indexed by some set X, so together the variables $f(x)$ are considered a (stochastic) function over the index set. For any finite subset of indices $x_1, x_2, ..., x_n$, we have

$$\begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_n) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu(x_1) \\ \mu(x_2) \\ \vdots \\ \mu(x_n) \end{bmatrix}, \begin{bmatrix} k(x_1,x_1) & k(x_1,x_2) & \dots & k(x_1,x_n) \\ k(x_2,x_1) & k(x_2,x_2) & \dots & k(x_2,x_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(x_n,x_1) & k(x_n,x_2) & \dots & k(x_n,x_n) \end{bmatrix} \right) \tag{1}$$

where $\mu(x)$ is the mean function, and $k(x_i, x_j)$ is a *kernel*. The kernel is a positive semi-definite function that specifies the covariance matrix Σ for any finite subset of variables. Note that each $f(x_i)$ is marginally Gaussian, with $\mu = \mu(x_i)$ and $\sigma^2 = k(x_i, x_i)$.

Furthermore, computing the conditional distribution of one subset of variables given another subset from this distribution is straightforward. For vectors $\boldsymbol{f}_1, \boldsymbol{f}_2$ of variables, if

$$\begin{bmatrix} \boldsymbol{f}_1(\boldsymbol{x}_1) \\ \boldsymbol{f}_2(\boldsymbol{x}_2) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{bmatrix}, \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{12}^T & \Sigma_{22} \end{bmatrix} \right) \tag{2}$$

then

$$\boldsymbol{f}_2 | \boldsymbol{f}_1 \sim \mathcal{N}\left(\boldsymbol{\mu}_2 + \Sigma_{12}^T \Sigma^{-1} \left(\begin{bmatrix} \boldsymbol{f}_1 \\ \boldsymbol{f}_2 \end{bmatrix} - \begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{bmatrix} \right), \Sigma_{11} - \Sigma_{12}^T \Sigma^{-1} \Sigma_{12} \right) \tag{3}$$

where $\Sigma_{mn}(i,j) = k(x_m(i), x_n(j))$. This follows immediately from the standard formula [7] for conditioning a marginal joint Gaussian distribution.

3 The model

Suppose we have an observable function $o(x) \sim \mathcal{N}\left(\mu_o(x), \sigma_o^2(x)\right)$, and we would like to find $\max_{x \in X} \mu_o(x)$. (In our experiments, we take σ_o^2 to be constant, but in general it could depend on x.) We intend to use information about the posterior distribution of $\mu_o(x)$ given observed data to select a point at which to evaluate o next. We let $\mu_o(x) = f(x)$, where f is drawn from a GP distribution with mean function μ_f and kernel k. The marginal distribution of o is therefore

$$\boldsymbol{o}(\boldsymbol{x}) \sim \mathcal{N}(\boldsymbol{\mu}_f(\boldsymbol{x}), \sigma_o^2 \mathrm{I} + \Sigma) \tag{4}$$

where again $\Sigma(i,j) = k(x_i, x_j)$, and the full joint of o and f is

$$\begin{bmatrix} \boldsymbol{o}(\boldsymbol{x}_o) \\ \boldsymbol{f}(\boldsymbol{x}_f) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_f(\boldsymbol{x}_o) \\ \boldsymbol{\mu}_f(\boldsymbol{x}_f) \end{bmatrix}, \quad \begin{bmatrix} \Sigma_{oo} + \sigma_o^2 \mathrm{I} & \Sigma_{of} \\ \Sigma_{of}^T & \Sigma_{ff} \end{bmatrix} \right) \tag{5}$$

and the posterior distribution of the mean function given a set of observed values is

$$\boldsymbol{f} | \boldsymbol{o} \sim \mathcal{N}\left(\boldsymbol{\mu}_f + \Sigma_{of}^T \Sigma^{-1} \left(\begin{bmatrix} \boldsymbol{o} \\ \boldsymbol{f} \end{bmatrix} - \begin{bmatrix} \boldsymbol{\mu}_f(\boldsymbol{x}_o) \\ \boldsymbol{\mu}_f(\boldsymbol{x}_f) \end{bmatrix} \right), \quad \Sigma_{oo} - \Sigma_{of}^T \Sigma^{-1} \Sigma_{of} \right) \tag{6}$$

Note that in our model, $\mu_o(x) = \mu_f(x)$, since $\boldsymbol{o}$ and $\boldsymbol{f}$ share the same (prior) mean. This is known as a *Gaussian process regression* (GPR) model [7], and is a generalization of least squares linear regression.[1]

[1]For $x \in \mathbb{R}^k$ and $k(x_i, x_j) = x_i \cdot x_j$, this formulation is equivalent to ridge regression.

4 Sampling strategies

Given a statistical model of the function we wish to optimize, our next consideration is what to do with the model. As we alluded earlier, we are interested in finding an optimum of f while trying to minimize the number of function evaluations. First, we note that if we were only allowed *one* further function evaluation, and our goal was to sample the function at its maximum point, the 'rational' or 'Bayes-optimal' [8] action would be to sample at the domain point x^* where $f(x^*)$ is the variable most likely to be the maximum out of all varibles $f(x)$ for $x \in X$. By 'Bayes-optimal' we mean that, according to the posterior, sampling at x^* gives the greatest expectation of our value function, which is in this case the probability of sampling the maximum of f.

If we knew we had a fixed horizon h of samples remaining before we must stop, the Bayes-optimal action could theoretically be computed by expanding all possible futures of length k beginning at sampling from each point $x \in X$, computing the probability of *eventually* sampling the maximum after sampling at x, and choosing x according to this criterion. (Discussion of this expansion in the context of cumulative reward optimization is discussed in [8].) Of course, this is intractable. In fact, even computing the one-step optimal point x^* is impossible over an infinite domain as it requires sampling from (or integrating over) a multivariate Gaussian distribution with an infinite number of variables. However, the question be answered at least in principle over any finite domain. In light of these computational issues, we restrict ourselves to heuristics geared toward finding x^*.

We now describe three potential strategies for choosing the next point x at which to evaluate $o(x)$. All of the sampling strategies below require the selection of a finite set $X_c \in X$ of candidate points to evaluate.

Greedy Sampling Sample o at the point $x_g \in X_c$ at which the posterior mean of f is greatest given the current observations.

If the posterior variance is uniform over X and $x^* \in X_c$, then $x_g = x^*$.

Thompson Sampling Sample a function $\hat{\boldsymbol{f}}(X_c)$ from the posterior model. Sample o at the point $x_t = \operatorname{argmax}_{x \in X_c} \hat{\boldsymbol{f}}(x)$. [9]

A Thompson sample is a sample from the distribution $P(x = \operatorname{argmax}_{x' \in X_c} x')$.

PMax Sampling Sample a function $\hat{\boldsymbol{f}}(X_c)$ from the posterior model. Note which of the $x \in X_c$ was the maximum. Repeat. Choose the point with the highest number of occurrences of being the max.

This is essentially a repeated form of Thompson sampling, and is intended to discover the point in X_c that has the highest probability (according to the posterior) of having the maximum true mean over all points in X_c, rather than generating a sample from $P(x = \operatorname{argmax}_{x' \in X_c} x')$.

Note that except for the choice of candidate points, the strategies described here require no knowledge of the geometry of the underlying space X, if such a thing even exists. They apply equally well to vectors in $\mathbb{R}^n$ as they do to strings or trees or documents, so long as a kernel can be defined between the objects concerned.[2] That said, it is possible to augment the Greedy and Thompson sampling strategies with hill-climbing in domains that admit a gradient in the underlying space.

Traditional hill climbing is sometimes applied to the kinds of functions we are discussing here (as in [5]), but when the objective function is stochastic, they require many function evaluations at the one point to reduce variance before making progress toward a potentially

[2] For an overview of a wide variety of kernels for different objects, see [10]

better point.

The class of algorithms most often associated with the type of problem consists of various types of Genetic Algorithms (GAs). One major difference between GPRs and GAs is data efficiency: When optimizing with GAs, samples from the objective function are 'thrown away' if they are shown to be poor, whereas in our method, this data is retained, and sampling methods will tend to avoid the points in the future. Although some GA methods, such as [11, 12] use GPR to aid optimization, they use them in conjunction with another optimization routine (in this case a genetic algorithm.) Furthermore, these methods use only the posterior mean, or the posterior mean and marginal posterior variance, while the methods described here (excluding greedy sampling) implicitly make use of the full distribution over f. (Posterior mean and posterior covariance.)

5 Experimental results

We now examine the application Gaussian processes to the task of optimizing robot walk velocity. At its most complex, a particular periodic walk could be described by listing every joint angle in the robot at every timestep over one walk period. Clearly this space is of far too high a dimension to achieve any kind of progress in terms of learning or even hand-coding a walk, and researchers have reduced this space to a more manageable parameterization in several ways. Two of the most popular are to specify a relatively small number of points through which the legs must pass [4] or to specify a set of 'neutral' positions for the legs and body, and generate a walk that leaves the exact path of the joints unspecified but requires they pass through this neutral position [6]. In practice, other parameters are introduced, such as the specified velocity with which a leg is lifted or set, that allow further control. For this work, we have used Carnegie Mellon University's Tekkotsu 2.2.2 [13] software to control the robots, which uses a parameterization similar to that in [6]. It is a parametrization of a 'trot' in which (by default) diagonally opposite legs lift and set at the same time. While 'state-of-the-art' walk speeds on the ERS-7 are reported to be in the 0.3000m/s to 0.4000m/s second range, it is widely accepted that the surface that the dogs walk on can have a significant impact on achievable velocity. We found that the felt surface in our lab had very low friction against the plastic of the Aibo, which resulted in many walks being unstable, and many 'catastrophic' failures (i.e. dogs flipping themselves over) that made experimentation extremely difficult. While we intend to investigate this problem further in the future, for now we have restricted our 'Target Velocity' parameter to a maximum of 0.250 m/s in order to make experimentation feasible.

Regardless of the parameterization, we now can treat the velocity function as stochastic map from $\mathbb{R}^k \to \mathbb{R}$. In our case, we optimize over the $k = 15$ dimensional parameter space shown in Table 1.

Using this parameterization, we define a Gaussian process as described in Section 3. In our case, we assume that at various points x, we will evaluate the noisy function o, which has as its mean the unobserved function f, and variance σ_o^2.

Because we had very little experience in this space, we consulted a helpful domain expert who suggested we confine our search to the bounds indicated in Table 1. We hypothesized that in general, parameter vectors that were close in terms of Euclidean distance were likely to generate similar walk velocities. We therefore chose to use the radial basis function kernel

$$k(x_i, x_j) = c \cdot \mathrm{e}^{-\frac{1}{2}(x_i - x_j)^T S (x_i - x_j)} \tag{7}$$

(Recall that $x_i, x_j \in \mathbb{R}^{15}$.) Where S is a scaling matrix that has along its diagonal the inverse of the range of each dimension, i.e. the distance from one end of one dimension's range to the other is equal to 1. (Assuming other dimensions are held constant.)

Table 1: Walk Parameters and Bounds

	BH	BA	Per	FX	FY	RX	RY	FLT	FST	RLT	RST
Min	95	-5	300	100	55	-115	60	-.05	-.05	-.05	-.05
Max	120	15	750	140	100	-90	110	.05	.05	.05	.05

	LV	SV	Hop	Target
Min	100	-250	-4	100
Max	250	-100	4	250

Legend BH and BA are the body height in mm and angle in degrees. FX, FY, RX, RY are the X and Y neutral positions of the front and rear legs in mm (mirrored appropriately for the left and right legs). FLT, FST, RLT, RST, are the front and rear lift and set times, expressed as a percentage deviation from the normal 'trot' pattern. LV and SV are lift and set velocities for all legs in mm/s, and Target is the requested target velocity, in mm/s.

We now need to determine the value of the prior mean of f, μ_f, the constant c in the kernel function, and the observed variance σ_o^2. First, we note that these are expectations over the domain X, and therefore can be easily sampled in our case. We used 50 random velocity samples to initialize c and μ_f, and 30 samples at a random point to initialize σ_o^2. That said, we would point out that these are values that could be elicited from a knowledgeable person: The parameters μ_f and c describe the marginal prior distribution of f at a point, and therefore can be computed easily from an intuition such as, 'I think f is probably on average about 0.15m/s, and within about 0.10m/s of that mean 99% of the time.' (This is approximately what the parameters we computed from the random data implied.) *A priori* ideas about the observation noise might be more difficult, but it is not unreasonable to have prior knowledge about this in certain situations; one thinks of specifications of sensors like sonar, laser rangefinders , etc.

In addition to being easily estimatable, it the effect of changing the value for each of these parameters has a predictable effect on our optimization method: Underestimating μ_f will cause the algorithm to be more 'greedy,' as all points above the mean will look more and more surprisingly good, and therefore more likely to be the actual maximum of the function. Conversely, setting μ_f too high makes everything appear mediocre or poor, thus causing excessive exploration. Altering c has an opposite effect; increasing it makes every point appear less extreme and therefore less likely to be the maximum, and decreasing it causes all points to seem more surprising.

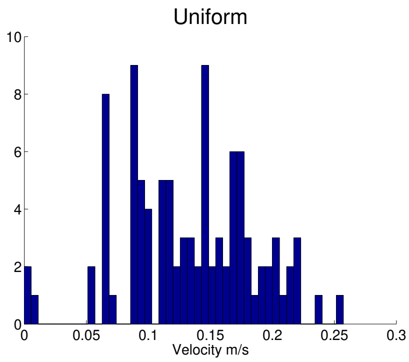
At this point we remark that we have chosen a very simplistic model. There is extensive literature, for example, on adapting parameterized kernels using data [2]. For example, we could easily adapt the scaling matrix S using a maximum likelihood criterion. Taken one step further in the spirit of being Bayesian, we could have also defined priors over these parameters. We have chosen not to do this in order to show how well even this primitive model can perform when initialized sensibly.

To evaluate the each parameter choice, we had the robot walk between two large orange markers. The robot determined its distance from the marker at different times based on the marker's apparent area in the robot's field of view in order to estimate its velocity. Each sampled measurement is over at least ten seconds of walking.

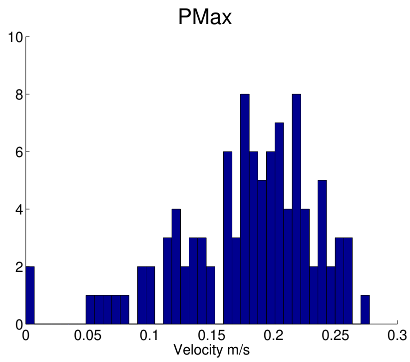
We ran our three sampling methods 100 times each, along with a run of points sampled uniformly within the above mentioned bounds, starting from the same random seed. Each of Greedy, Thompson, and PMax used a candidate set X_c of 1000 points drawn randomly from within the bounds to choose each sample. (A new set X_c was drawn for each sample choice.) PMax sampled a function from the posterior evaluated at X_c 1000 times to choose which point to sample next. Each sampling method took less than 4 seconds to run on a 1.5 GHz G4 machine, and a total run of 100 points took approximately 1 hour. Results are shown in Figure 5. A histogram of the resulting observations along with their minimum, median, mean, and maximum are reported.

The highest observations were recorded by the Greedy and PMax methods, however we sampled at the parameters with the best and second best observations from each method 30 times, and reported their mean and sample standard deviation in the figure also. We found that Greedy sampling had produced the best walk speed, while the other three top speeds were more or less equivalent. We also tested the default hand-coded Tekkotsu walk, and found it had a mean of 0.1920m/s and a sample standard deviation of 0.0169. (Here, we set the target velocity of the default walk to 0.250m/s.)

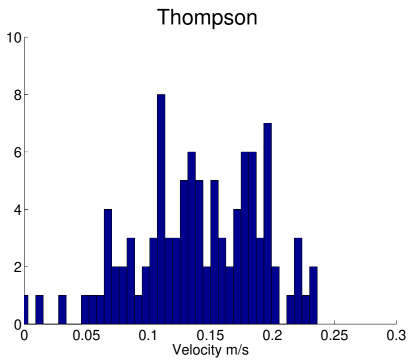
While the maxima found by the different methods during this run were more or less similar, the distributions of observations were significantly different. The median walk speed of PMax and Greedy is significantly higher than those of uniform and Thompson sampling, indicating that these methods are more likely to sample potentially good points.



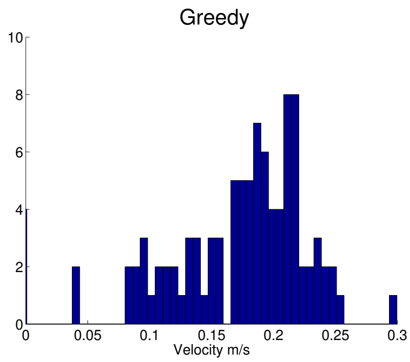
Min	Med.	Mean	Max
0.0000	0.1332	0.1299	0.2573
Best		2nd Best	
0.2569	(0.0191)	0.2074	(0.0161)



Min	Med.	Mean	Max
0.0000	0.1873	0.1778	0.2775
Best		2nd Best	
0.2501	(0.0192)	0.2360	(0.0199)



Min	Med.	Mean	Max
0.0000	0.1417	0.1389	0.2361
Best		Second	
0.2675	(0.0192)	0.2404	(0.0184)



Min	Med.	Mean	Max
0.0000	0.1838	0.1584	0.2997
Best		Second	
0.2701	(0.0233)	0.2291	(0.0161)

Figure 1: Sample figure caption

6 Conclusion

We have described an approach for optimization using Gaussian processes, and applied it to the problem of Aibo gait learning, resulting in a walk velocity exceeding that of a hand-coded walk by approximately 0.07m/s. While it is difficult to compare our results to those of others due to our experimental difficulties and the differences caused by walking surfaces, we have shown it is possible to find a walk parameter significantly better than a hand-coded default using only 100 function evaluations which took less than one hour using a single dog.

Acknowledgments

We would like to sincerely thank our domain expert Sonia Chernova, and our team of programmers: Eric Coulthard, Wesley Loh, James Neufeld, and Sam Vincent.

References

[1] David Wolpert and William G. Macready. No free lunch theorems for optimization. *No free lunch theorems for optimization*, 1(1):67–82, 1997.

[2] C. Williams. Prediction with Gaussian processes. In *Learning in Graphical Models*. MIT Press, 1999.

[3] R. Neal, editor. *Bayesian Learning for Neural Networks*. Springer, 1996.

[4] Min Sub Kim and William Uther. Automatic gait optimisation for quadruped robots. In *Australasian Conference on Robotics and Automation*, 2003.

[5] Nate Kohl and Peter Stone. Machine learning for fast quadrupedal locomotion. In *The Nineteenth National Conference on Artificial Intelligence*, pages 611–616, July 2004.

[6] Manuela Veloso Sonia Chernova. An evolutionary approach to gait learning for four-legged robots. In *Intelligent Robots and Systems*, 2004.

[7] C. E. Rasmussen. *Advanced Lectures in Machine Learning: ML Summer Schools 2003*, chapter Gaussian Processes in Machine Learning. Springer-Verlag, 2004.

[8] Tao Wang, Daniel Lizotte, Michael Bowling, and Dale Schuurmans. Bayesian sparse sampling for on-line reward optimization. In *ICML 2005*, Bonn, 2005. To appear.

[9] William R. Thompson. On the theory of apportionment. *Biometrika*, 25:285–294, 1933.

[10] John Shawe-Taylor and Nello Cristiani. *Kernel methods for pattern analysis*. Cambridge, 2004.

[11] H. Ulmer, F. Streichert, and A. Zell. Optimization by gaussian processes assisted evolution strategies. In *Selected Papers of the International Conference on Operations Research (OR 2003)*, pages 435–442, Heidelberg, 2003. Springer.

[12] Dirk Büche, Nicol N. Schraudolph, and Petros Koumoutsakos. Accelerating evolutionary algorithms with gaussian process fitness function models. *IEEE Transactions on Systems, Man and Cybernetics, Part C, Special Issue on Knowledge Extraction and Incorporation in Evolutionary Computation*, 32(4), 2004.

[13] Tekkotsu. `http://www-2.cs.cmu.edu/~tekkotsu/`.