Q2: Memory-based active learning for optimizing noisy continuous functions

Andrew W. Moore[12], **Jeff G. Schneider**[12], **Justin A. Boyan**[1], and **Mary S. Lee**[2]
CMU Computer Science & Robotics[1] Schenley Park Research Inc.[2]
Pittsburgh, PA 15213 6413 Howe Street
http://www.cs.cmu.edu/~AUTON Pittsburgh, PA 15206
{awm,schneide,jab,mslee}@cs.cmu.edu

Abstract

This paper introduces a new algorithm, Q2, for optimizing the expected output of a multi-input noisy continuous function. Q2 is designed to need only a few experiments, it avoids strong assumptions on the form of the function, and it is autonomous in that it requires little problem-specific tweaking.

These capabilities are directly applicable to industrial processes, and may become increasingly valuable elsewhere as the machine learning field expands beyond prediction and function identification, and into embedded active learning subsystems in robots, vehicles and consumer products.

Four existing approaches to this problem (response surface methods, numerical optimization, supervised learning, and evolutionary methods) all have inadequacies when the requirement of "black box" behavior is combined with the need for few experiments. Q2 uses instance-based determination of a convex region of interest for performing experiments. In conventional instance-based approaches to learning, a neighborhood was defined by proximity to a query point. In contrast, Q2 defines the neighborhood by a new geometric procedure that captures the size and shape of the zone of possible optimum locations. Q2 also optimizes weighted combinations of outputs, and finds inputs to produce target outputs.

We compare Q2 with other optimizers of noisy functions on several problems, including a simulated noisy process with both non-linear continuous dynamics and discrete-event queueing components. Results are encouraging in terms of both speed and autonomy.

1 ACTIVE LEARNING FOR OPTIMIZATION

The apparently humble task of parameter tweaking for noisy systems is of great importance whether the parameters being tweaked are for an algorithm, a real manufacturing process, a simulation, or a scientific experiment. The purpose of this paper is two-fold. First, we wish to highlight the potential importance of machine learning as an as-yet underexploited tool in this domain. Second, we will introduce Q2, a new algorithm designed for this domain.

We consider a generalized noisy optimization task in which a vector $\mathbf{x}$ of real-valued inputs produces a scalar output y that is a noisy function of $\mathbf{x}$:

$$y = g(\mathbf{x}) + noise \qquad (1)$$

Given a constrained space of legal inputs, the task is to find the input vector $\mathbf{x}_{\text{opt}}$ that maximizes g, using only a small number of experiments.

In both industrial settings and in algorithm-tuning, this task often demands considerable human intervention and insight. A factory manager who wants to optimize a process can:

- Buy a computer, statistics software, and hire a professional statistician to solve the problem using insight and experiment design.
- Save money and try to "wing it" by manually tuning the parameters.

For highly expensive or safety-critical processes, the first option is always preferable, leaving only the question of which are the best analysis and experiment design tools for the statistician to use. This area is heavily investigated by the academic statistics community.

But there are also many situations in which it is impractical to enlist human-aided analysis during optimization, for example if a vehicle engine self-tunes during driving. And there are many other situations in which the potential benefit from optimization is too small to justify paying for expert professional analysis. In such cases, it is tempting to ask: Can "black box" automated methods optimize noisy systems? If practical black box methods are found, they could be widely used. Somewhat fancifully, this could lead to the eventual inclusion of Black Box Optimizer chips within a huge range of consumer products, from vehicle engines and industrial equipment down to refrigerators, toasters, and toys.

In the next section we discuss variants of the Black Box Noisy Optimization task. Then in Section 3 we discuss existing approaches. After that we present and evaluate Q2, a new algorithm.

2 VARIANTS OF NOISY OPTIMIZATION

The generalized noisy optimization task summarized by Equation 1 has many variants. For instance, in some domains each experiment is a lengthy procedure, and so there is ample computation time between experiments. In other domains, experiments are very quick, leaving an optimizer little time to make its recommendations. The specifics of the domain determine which methods are appropriate. The following factors need to be considered:

- **Minimize regret or the number of experiments?** Do we pay a constant cost per experiment, or do experiments with poor results cost us more? In scenarios such as tuning the parameters for an algorithm, or optimizing a test plant in which all products will be discarded, the cost per experiment may be constant. But in a task such as minimizing the fuel consumption of a running engine, some experiments cost more than others. Here, we focus on simply minimizing the number of experiments. Note that this presumes that we are not risk-averse: there is no penalty for performing highly unpredictable experiments.

- **How much computer time is available to choose experiments?** If experiments are very cheap and very quick, then an algorithm that needs extensive CPU time to select the ideal next experiment could still be inferior to one that requires only a fraction of a second to suggest a reasonable-but-less-than-ideal experiment. Here, we assume that experiments are costly enough (in time or money) that it pays to choose them carefully. But the Q2 algorithm can be adjusted to satisfy any desired tradeoff between the speed and the quality of proposed experiments.

- **Are we doing local or global optimization?** Unless we have strong prior knowledge, global optimization of a function of more than a couple of inputs requires a very large number of experiments. Q2 is only designed to find a local optimum, though empirically it appears to be good at discovering the global optimum.

- **Can we re-use old data?** Many algorithms have a "current location" or "current set of k recent evaluations" but otherwise disregard earlier evaluations. Q2, however, can exploit any existing data, including previous evaluations obtained by other experimental methods.

In this paper we also assume that there are no long term dynamics, i.e. the output of the n'th experiment depends only on the n'th chosen $\mathbf{x}$, not on previous $\mathbf{x}$ values or the time. Unlike [2, 6] we only try to find the optimum, not to model the g function.

3 POSSIBLE APPROACHES

Many disciplines have methods that are relevant to noisy optimization. Space permits only a brief survey.

Numerical analysis: Numerical methods such as Newton-Raphson or Levenberg-Marquardt [11] have fast convergence properties, but they must be applied carefully to prevent oscillations or divergence to infinity, which violates our desire for black box autonomy. Furthermore, current numerical methods cannot survive noise.

Stochastic approximation: The algorithm of [12] finds roots without the use of derivative estimates. Keifer-Wolfowitz (KW) [5] is a related algorithm for noisy optimization. It estimates the gradient by performing experiments in both directions along each dimension of the input space. Based on the estimate, it moves its experiment center and repeats. It uses decreasing step sizes to ensure convergence. KW's strengths are its aggressive exploration, its simplicity, and that it comes with convergence guarantees. However, it can attempt wild experiments if there is noise, and discards the data it collects after each gradient estimate is made. Amoeba (see below) is a similar approach, but in our experience is superior to KW.

Amoeba search: Amoeba [11] searches k-d space using a simplex (i.e. a k-dimensional tetrahedron). The function is evaluated at each vertex. The worst-performing vertex is reflected through the hyperplane defined by the remaining vertices to produce a new simplex that has moved up the estimated gradient. Ingenious simplex transformations let the simplex shrink near the optimum, grow in large linear zones, and ooze along ridges.

Experiment design & response surface methods: Current RSM practice is described in the classic reference [1]. It proceeds by cautious steepest ascent hill-climbing. A region of interest (ROI) is established at a starting point and experiments are made at positions that can best be used to identify local function properties with low-order polynomial regression. Much of the RSM literature concerns *experimental design*—deciding where to take data in order to acquire the lowest variance estimate of the polynomial coefficients in a fixed number of experiments. When the gradient is estimated confidently, the ROI is moved accordingly. Quadratic regression locates optima within the ROI, and diagnoses ridge systems and saddle points. The strength of RSM is that it avoids changing operating conditions based on inadequate evidence, but moves once the data justifies it. A weakness of RSM is that human judgment is needed: it is not an algorithm, but a manufacturing methodology.

Evolutionary computation and learning automata: Methods such as genetic algorithms begin by sampling uniformly, but then bias later samples in favor of the experiments that had good outcomes. There is a vast literature of refinements of such methods. These approaches need thousands, sometimes millions, of evaluations, because they attack a different problem: Global Optimization, usually for noise-free, cheap-to-evaluate criteria.

PMAX: PMAX is a simple, effective algorithm. Based on the data from the experiments so far, it uses a non-linear function approximator to estimate the underlying function $g(\mathbf{x})$. The next experiment is taken at the point that maximizes the estimate of g. This approach has been used with a decision-tree approximator [13], with neural nets (in many commercial products), and with locally weighted regression [9]. Variations of PMAX include taking the next experiment not at the predicted optimum, but instead where the confidence intervals are widest [6], or where the top of the confidence interval is maximized [9], or in accordance with the Interval Estimation heuristic [4] or similar criteria [13].

Empirically, we have found that PMAX using locally weighted regression as the function approximator is often faster than more sophisticated alternatives [9]. However it has some serious drawbacks:

- In conventional function approximation one must solve the bias-variance tradeoff. This is often determined automatically using cross-validation [8], but this proves difficult with a set of very few, weirdly distributed datapoints obtained during optimization. Empirically we have observed dismal performance when attempting this. In addition, conventional approaches search for the best model over the whole data range, whereas we only need our model to be accurate in the vicinity of the optimum.
- PMAX is very expensive. It needs to train a function approximator each time an experiment is made, and then the approximate function must be numerically optimized to produce the suggested experiment.
- PMAX can get stuck in hallucinated optima since it is not choosing experiments to give the most information (in the way that RSM does).

4 THE *Q2* ALGORITHM

The Q2 algorithm is an attempt to combine the strengths of Newton's method (superlinear convergence), RSM (using estimates of significance in the face of noise), and PMAX (exploiting all available data). Let us first outline the structure of the Q2 algorithm, before discussing its details:

1. Input a set of previous experimental results
$$(\mathbf{x}_1 \rightarrow y_1), (\mathbf{x}_2 \rightarrow y_2), \ldots, (\mathbf{x}_n \rightarrow y_n) \qquad (2)$$
and HR: a hyper-rectangular portion of input space over which the optimization is constrained to take place.
2. Select a convex Region Of Interest (ROI) within HR such that:
   - The constrained optimum within HR is expected to lie within ROI.
   - There is no evidence to contradict the assumption that the function is well-approximated by a quadratic within ROI.
3. Select a useful experiment to take within ROI.
4. Return the experiment, the estimated location of the optimum, and (optionally) other information such as the ROI and a regression analysis of the local quadratic.

In typical operation, the suggested experiment will be performed, we will add the new datapoint to the dataset, and return to Step 2.

Step 2: Selecting the ROI

Step 2 begins by generating a sequence of candidate Regions Of Interest, $ROI_1, ROI_2, \ldots, ROI_j, \ldots$ from which the final ROI will be selected. The generated sequence has the properties that

$$ROI_1 := HR \text{ and } ROI_j \supseteq ROI_{j+1} \qquad (3)$$

where ROI_{j+1} is determined by cutting away an unpromising subregion of ROI_j. How is the cut determined? Let us consider an example.

Figure 1 shows a Gaussian function of two inputs. Suppose HR is set to be the full square region depicted in the figure, and suppose we have available the thirty noisy datapoints that are also shown. Call this dataset DS_1. We can fit a quadratic to DS_1. Write

$$\hat{y}_k = c + \mathbf{b}^T \mathbf{x}_k + \tfrac{1}{2}\mathbf{x}_k^T \mathbf{A} \mathbf{x}_k \tag{4}$$

where A is symmetric, or, equivalently,

$$\hat{y}_k = c + b_1 x_{k1} + b_2 x_{k2} + \tfrac{1}{2} a_{11} x_{k1}^2 + a_{12} x_{k1} x_{k2} + \tfrac{1}{2} a_{22} x_{k2}^2 \tag{5}$$

The regression is a matter of simple matrix manipulation. Write $\mathbf{z}_k$ = the vector of polynomial terms for the kth input point, $\mathbf{x}_k$.

$$\mathbf{z}_k = (1, x_{k1}, x_{k2}, x_{k1}^2, x_{k1}x_{k2}, x_{k2}^2) \tag{6}$$

Write $\mathbf{Z}$ = a matrix whose kth row is $\mathbf{z}_k$, and write $\mathbf{Y}$ = a vector whose kth element is y_k. Finally define

$$\boldsymbol{\beta} = (c, b_1, b_2, \tfrac{1}{2}a_{11}, a_{12}, \tfrac{1}{2}a_{22})^T \tag{7}$$

as the regressed coefficients. Then using Bayesian regression with non-informative priors on $\boldsymbol{\beta}$ and σ^2 (the estimated Gaussian noise), we have the MAP of $\boldsymbol{\beta}$ (also the maximum likelihood value in this case) as

$$\boldsymbol{\beta} = (\mathbf{Z}^T \mathbf{Z})^{-1} \mathbf{Z}^T \mathbf{Y} \tag{8}$$

In practice, if the information is known, we can put Gaussian priors on the coefficients and an inverse-Gamma prior on the noise. For our dataset the resulting quadratic approximation is shown in Figure 2. Note that because the underlying function is so far from quadratic, this is a poor fit.

Q2 evaluates each of the datapoints in DS_1 using the quadratic, producing the values of Equation 5. Let $(\mathbf{x}_{k(1)}, y_{k(1)})$ be the datapoint that is predicted to be the worst, i.e. $k(1) = \text{argmin}_k \hat{y}_k$. It will be used to define a cut of ROI_1. We look at the direction of the steepest gradient, $\nabla \hat{y}$, of the quadratic at $\mathbf{x}_{k(1)}$, and we cut using the half-plane perpendicular to this direction so that

$$ROI_2 = ROI_1 \cap \{\mathbf{x} \mid (\mathbf{x} - \mathbf{x}_{k(1)}).\mathbf{d}_1 \geq 0\} \tag{9}$$

where $\mathbf{d}_1 = \nabla \hat{y}$ evaluated at $\mathbf{x}_{k(1)}$.

In Figure 2, the worst point according to the quadratic is at the top left, and with some effort the resulting cut-plane can be seen.

Why do we use the above approach? We want to use our unreliable (probably biased) quadratic to tell us how to reduce the *ROI*. We assume that even if the quadratic is a poor model for g, it will be adequate to predict an *unpromising* location for the optimum. Why pick the point with the *predicted* worst value instead of the actual worst value? Because the actual values are noisy, meaning that an unlucky datapoint could be misleadingly removed.

We have described how ROI_2 is constructed from ROI_1. In general, ROI_{j+1} is constructed from ROI_j using a similar recipe: set $DS_{j+1} = DS_j - (\mathbf{x}_{k(j)}, y_{k(j)})$, do a regression using dataset DS_{j+1} (which will be less biased than using DS_j), and cut using the point that the new regression predicts will be worst. Figure 3 shows the approximation that results after the first cut has been made (giving a less biased fit than Figure 2), and also shows the second cut. Figure 4 shows what remains after the twelfth cut: the fit is now good, because it is only based on datapoints near the quadratic-shaped optimum. Figures 5–7 use a bigger dataset and an extreme ridge system.

At this point Q2 has generated a series of candidate regions, $ROI_1, ROI_2 \ldots$. To decide which to select, we perform regression analysis on the quadratics in each of the *ROI*s. As j increases, ROI_j shrinks and is based on fewer datapoints. So, as j increases, ROI_j's bias decreases and its variance increases. We select the ROI_j with the best tradeoff using the criterion: *Choose the smallest ROI for which Bayesian regression analysis is confident about the location of the optimum, and for which the optimum is, with high probability, inside the ROI.*[1]

The results of this criterion are shown in Figures 8–13. With fewer or noisier datapoints, larger *ROI*s are chosen. The shape of the chosen *ROI*s nicely reflects the shape of the local ridge system (Figure 7). If irrelevant inputs are included, the *ROI* chosen by Q2 tends to stretch to ignore irrelevant dimensions (pictures omitted because of space constraints).

Step 3: Choosing the experiment

Once the *ROI* is determined, the estimated optimum is easily obtained as

$$\hat{\mathbf{x}}_{\text{opt}} = -\mathbf{A}^{-1}\mathbf{b} \tag{10}$$

(assuming the quadratic fit has revealed a maximum, meaning $\mathbf{A}$ is negative-definite). $\hat{\mathbf{x}}_{\text{opt}}$ is not necessarily the best place to experiment in order to gain useful new information. Instead, we investigated these options:

1. Put experiment at $\hat{\mathbf{x}}_{\text{opt}}$.
2. Choose a random point within *ROI*.
3. Choose the point in *ROI* that is predicted to most reduce the uncertainty about the location of the

[1] This is achieved by taking the joint posterior distribution (normal-gamma) on the noise and the coefficients of the quadratic form, and then (via Monte Carlo sampling) seeing whether at least $\tau = 98\%$ of the samples lie in the *ROI* and whether the expected regret of committing to the optimum is below a threshold (2% of the range of output values). Empirically these threshold choices are not performance-critical.

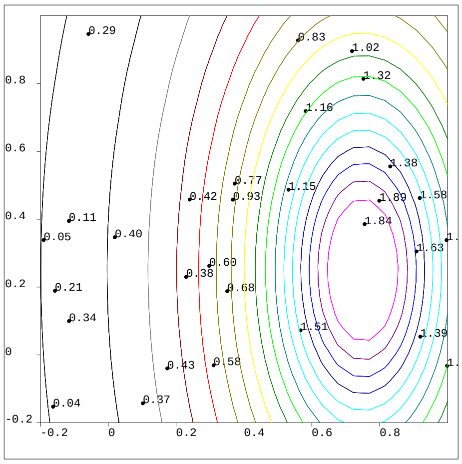

Figure 1: A function of two inputs. The optimum is at (0.75,0.25). It is a Gaussian bump, and hence very flat more than about 0.4 units of distance from the optimum. Also shown are 30 noisy datapoints. These were generated with uniformly random (x,y) coordinates, with z (height) set to f(x,y) plus Gaussian noise with standard deviation 0.1.

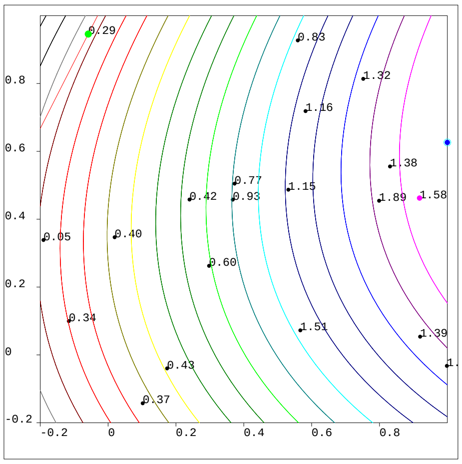

Figure 2: The best-fitting global quadratic regression approximation obtained by least squares regression on the 30 datapoints. The worst-scoring datapoint is in the top left.

optimum.

4. Choose the point in *ROI* that keeps the regression as orthogonal [1] as possible, mimicking established RSM practice.
5. Choose the point in *ROI* as far away from any previous datapoints (in or out of *ROI*) as possible.

Option 5 is best empirically. This is because options 3 and 4, despite their elegance, usually choose experiments at the edge of the *ROI*, reducing the opportunity for future cuts to shrink future *ROI*s. Option 1 quickly becomes stuck, and option 2 frequently wastes experiments.

Details

In this short paper, many details have been omitted. Some regressions predict a minimum or a saddlepoint, instead of a maximum. We have special-purpose techniques to deal with this. The Bayesian analysis is largely standard, and also omitted: see [3] for more details. Some confidence measures require Monte Carlo integration. These details will be discussed in a forthcoming technical report [10].

5 RESULTS

We begin by comparing Q2 with four versions of Amoeba and three versions of PMAX on the function f_1 from Figure 1 with noise of 0.3 added to each evaluation[2]. Amoeba is the classic search algorithm from [11]. Amoeba2 is the same except it is made resistant to noise by doing two evaluations and taking their average at each simplex vertex. Amoeba4 and Amoeba8 similarly average four and eight evaluations at each vertex. All the Amoebas begin with a medium-sized simplex started randomly in input space.

The results are in Figure 14. In this (and all subsequent experiments) we performed 25 independent runs of each optimizer, with each run consisting of 60 experiments. As well as selecting the datapoints for the experiments, at every stage the optimizers also gave their estimate of the location of the optimum. To assess the various optimizers, we wish to compare how good they are at estimating the optimum, and so we look at the true value of the underlying function at these estimates of the optimum. For the ith run of a particular optimizer, let s_i denote the mean of the true values at the estimates of the optimum. The figures in the left hand column are the mean s_i value of the optimizer over all 25 runs (i.e. $(\sum_i s_i)/25$). These values are also drawn graphically in the same column: the further to the right the dot lies, the better the mean score. The horizontal lines depict the 95% confidence intervals on the mean. The right hand column shows the mean performance of the optimizer on the final 15 of the 60 experiments. Unsurprisingly, all methods do better in later experiments, so the right hand means are higher.

Figure 14 shows that Q2 outperforms all the other methods on this problem. Amoeba4 is the best of the Amoebas; it is less affected by noise than Amoeba and Amoeba2, but it makes better progress than Amoeba8, which wastes 8 evaluations on every vertex.

[2] These tasks are available from http://www.cs.cmu.edu/~AUTON

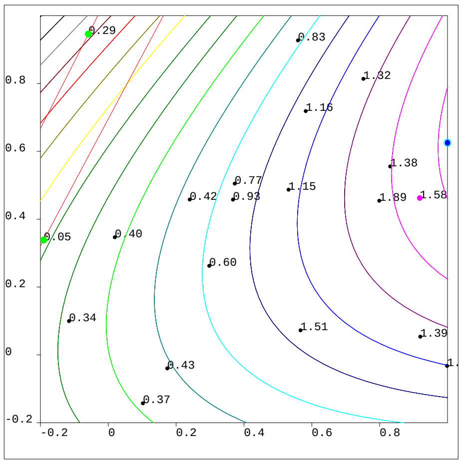

Figure 3: After the worst-scoring point is removed from the regression, we have the following fit to the remaining 29 datapoints. The worst predicted point among these is halfway up along the left edge. Note the cut that it causes.

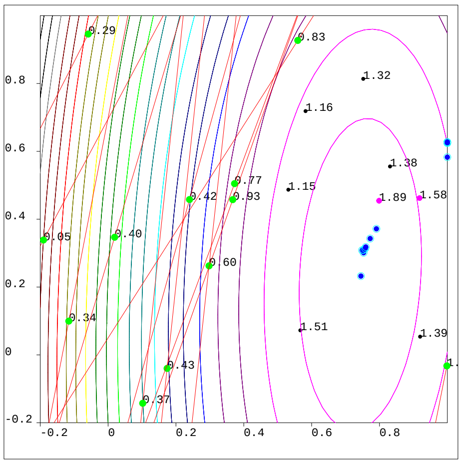

Figure 4: After 12 cuts the remaining datapoints (those inside the convex region defined by the cuts) are relatively close to the optimum, and the resulting local quadratic regression is an excellent local approximation.

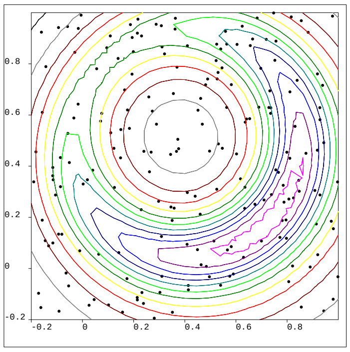

Figure 5: Another function of two inputs. The optimum is on the banana-shaped ridge at (0.75,0.2). 200 datapoints are shown (their heights omitted).

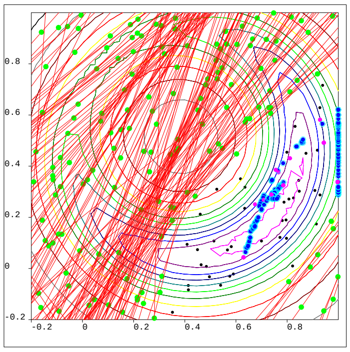

Figure 6: After the first 150 cuts, the region of interest nicely surrounds the ridge.

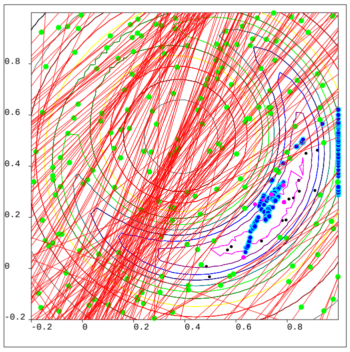

Figure 7: After the first 180 cuts, the region of interest is smaller still, yet continues to surround the true optimum.

Table 1 (shown later) gives results for the 2d-functions of Figures 5, 15, and 16 for noise levels of 0 and 0.3. With no noise, the one-evaluation-per-step version is always the best Amoeba. With noise, the best Amoeba is problem specific. The best PMAX is also problem specific. Q2 adapts well to noise and to differing levels of function complexity. Q2 is beaten by the Global and mediumly local PMAX for the noisy pure quadratic $f_3(x_1, x_2)$. In all other cases Q2 wins, but its main strength is autonomy: unlike Amoeba and PMAX no problem specific parameter needs to be chosen to make Q2 perform well.

Figure 17 shows a simulated, sanitized version of a real industrial process. Liquids enter a tank at a certain rate (a parameter) and a certain mix-ratio (a parameter) unless the tank is above a certain level (a parameter). They react causing a color dependent on the tank mix-ratio and the time spent in the tank. Thickener is added at a certain rate (a parameter), and the output passes through a cooling tunnel to wait on a holding belt. While waiting, color may change. When the belt fills beyond a certain level (a parameter), production halts. Customer demand randomly consumes material on the holding belt. The yield is the amount of material that reaches the customer with color lying in an acceptable tolerance range. This is a very noisy task. The yield is a highly non-quadratic function; one input is almost irrelevant, the others are all important, and two of the inputs must run to their maximum legal value for best performance. The results are given in Figure 18, and show a significant win for Q2. Q2 and the PMAX's also have far more repeatable results than the Amoebas.

We also applied conventional RSM to this task, using a star design prescribed by [1]. The star occupied the hyperrectangle defined by the legal ranges of values for

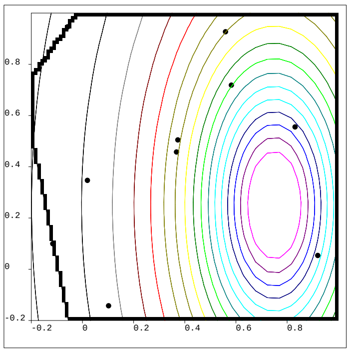

Figure 8: The region of interest selected for the function of Figure 1 given a dataset of only 10 points.

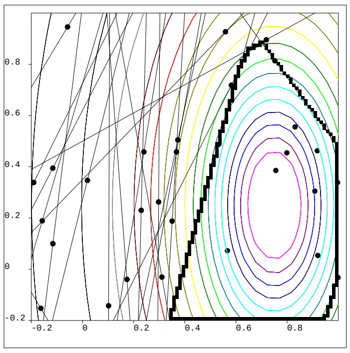

Figure 9: The region of interest when given 30 datapoints.

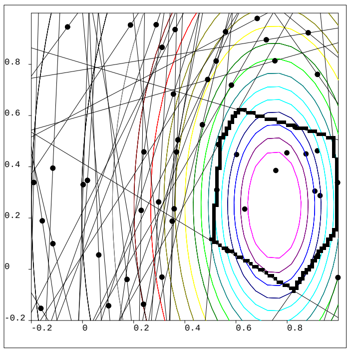

Figure 10: The region of interest when given 50 datapoints.

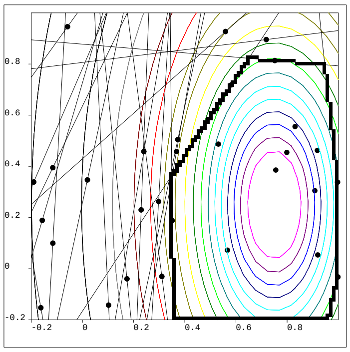

Figure 11: The region of interest selected for the function of Figure 1 given a dataset of 30 points, with no noise.

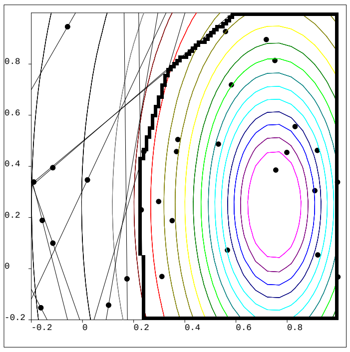

Figure 12: The region of interest when noise with std. dev. $\sigma = 0.5$ is added to the observations.

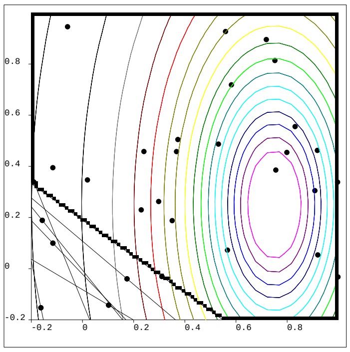

Figure 13: The region of interest when $\sigma = 2.0$.

each input. It needed 76 evaluations, but the chosen optimum had a yield below 10 units: worse than all the other methods, indicating that the assumption of a global quadratic is inadequate in this domain.

Next, we examine a domain where experiments are time-consuming. Figure 19 shows a generalization of the multi-buffer machine task described in [7] (this makes 10 products instead of 5). There are two inputs defining a simple parameterized policy for when to service the machine. Services are costly, but unscheduled breakdown is much worse. This task is evaluated by a computationally expensive simulation; for each setting of the two inputs, we perform 10000 simulation steps to evaluate the performance. Evaluations are very stochastic (with highly non-Gaussian noise). The results are shown for runs of only 24 experiments. Q2 learns a good policy in these 24 experiments, i.e. a total of only 24×10000 simulation steps. This compares favorably with the tens of millions of simulation steps needed for reinforcement learning in [7], but Q2 is unlikely to find as good a policy as their semi-MDP formulation.

The final results show Q2 being used for root-finding instead of optimization. The hand position in Figure 21 is a noisy function of θ_1 and θ_2. The task requires us to achieve the goal hand position. Although space permits no details, the version of Q2 for root (or target) finding uses linear instead of quadratic regression in its ROIs. The results are shown in Figure 22. Figure 23 shows the results when, on each experiment, the target position is varied randomly within the workspace. Amoeba, a pure optimization method for a fixed goal, is no longer applicable here, but PMAX and Q2 can still be used because their decision making simply requires a dataset of previous experiences. Q2's ability to tune its regions of interest decisively beats all PMAXs.

	Mean over all 100 trials	Mean over last 25 trials
PmaxGlobal	-0.417	-0.368
PmaxLocal	-0.402	-0.342
PmaxVLocal	-0.475	-0.418
Q2 (Linear)	-0.042	-0.021

Figure 23: Performance on kinematics when the target varies during each experiment.

	Mean over all 60 trials	Mean over last 15 trials
Amoeba	0.999	1.040
Amoeba2	1.130	1.234
Amoeba4	1.181	1.445
Amoeba8	0.950	1.209
PmaxGlobal	1.667	1.812
PmaxLocal	1.681	1.846
PmaxVLocal	1.517	1.691
Q2	1.716	1.894

Figure 14: Performance on $f_1(x_1, x_2)$ from Figure 1.

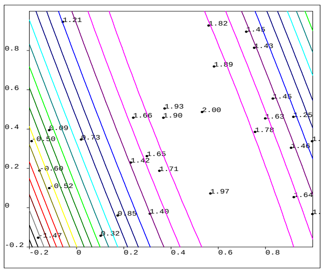

Figure 15: $f_3(x_1, x_2)$: a simple (pure quadratic) two-input function with an optimum at (0.5,0.5).

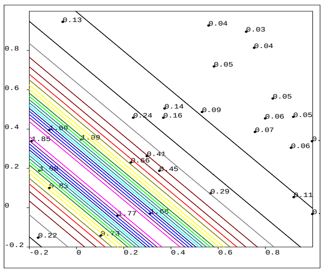

Figure 16: $f_4(x_1, x_2)$: a function in which the only relevant direction is $x + y$. The optima lie along a diagonal ridge.

Noise 0.0

$f_2(x_1, x_2)$

	Mean over all 60 trials	Mean over last 15 trials
Amoeba	2.168	2.509
Amoeba2	1.879	2.250
Amoeba4	1.576	1.993
Amoeba8	1.202	1.844
PmaxGlobal	1.735	1.859
PmaxLocal	1.866	2.116
PmaxVLocal	1.938	2.268
Q2	1.968	2.476

$f_3(x_1, x_2)$

	Mean over all 60 trials	Mean over last 15 trials
Amoeba	1.985	2.000
Amoeba2	1.970	2.000
Amoeba4	1.940	1.996
Amoeba8	1.885	1.982
PmaxGlobal	1.953	1.998
PmaxLocal	1.953	1.998
PmaxVLocal	1.938	1.994
Q2	1.959	2.000

$f_4(x_1, x_2)$

	Mean over all 60 trials	Mean over last 15 trials
Amoeba	1.816	2.000
Amoeba2	1.631	1.999
Amoeba4	1.275	1.944
Amoeba8	0.700	1.158
PmaxGlobal	1.618	1.803
PmaxLocal	1.691	1.870
PmaxVLocal	1.663	1.908
Q2	1.730	1.999

Noise 0.3

$f_2(x_1, x_2)$

	Mean over all 60 trials	Mean over last 15 trials
Amoeba	1.633	1.728
Amoeba2	1.656	1.839
Amoeba4	1.479	1.849
Amoeba8	1.182	1.802
PmaxGlobal	1.769	1.909
PmaxLocal	1.861	2.092
PmaxVLocal	1.835	2.117
Q2	1.859	2.388

$f_3(x_1, x_2)$

	Mean over all 60 trials	Mean over last 15 trials
Amoeba	1.769	1.771
Amoeba2	1.871	1.890
Amoeba4	1.904	1.954
Amoeba8	1.865	1.936
PmaxGlobal	1.910	1.979
PmaxLocal	1.911	1.984
PmaxVLocal	1.787	1.868
Q2	1.892	1.944

$f_4(x_1, x_2)$

	Mean over all 60 trials	Mean over last 15 trials
Amoeba	0.808	0.861
Amoeba2	0.875	1.009
Amoeba4	0.962	1.388
Amoeba8	0.637	0.956
PmaxGlobal	1.549	1.738
PmaxLocal	1.619	1.843
PmaxVLocal	1.489	1.756
Q2	1.675	1.947

Table 1: Optimization results for seven optimizers on three problems at two noise levels.

6 CONCLUSION

This paper has highlighted the importance of Black Box Noisy Optimization, surveyed possible approaches, and then introduced a new algorithm: Q2.

Algorithms like Newton's method, golden ratio search and conjugate gradient [11] maintain a region expected to contain an optimum and in which future experiments will occur. Q2 tries to do the same thing with two innovations. First, it can derive a ROI from a previous dataset irrespective of how that dataset was collected. Second, Q2 can survive noise. Q2 is also related to RSM and traditional instance-based learning. Future Q2 work will include trials on real processes, batching experiments, semi-quadratic regression for high dimensions, and survival of slowly time-varying systems.

Future work: This algorithm only finds local optima: what can be done to encourage further exploration for alternative optima? We also hope to produce a formal characterization of when this approach will best work. The main limitation is that the computational cost grows rapidly with the number of inputs, and the current Q2 is unlikely to be useful above 10 inputs. We have begun investigation into versions applicable to hundreds of inputs.

Acknowledgements

This work was supported by an NSF Career Award to Andrew Moore, and Justin Boyan's NASA GSRP Fellowship.

References

[1] G. E. P. Box and N. R. Draper. *Empirical Model-Building and Response Surfaces.* Wiley, 1987.

[2] D. A. Cohn, Z. Ghahramani, and M. I. Jordan. Active learning with statistical models. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems 7.* MIT Press, 1995.

[3] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian Data Analysis.* Chapman and Hall, 1995.

[4] L. P. Kaelbling. Learning in Embedded Systems. PhD. Thesis; Technical Report No. TR-90-04, Stanford University, Department of Computer Science, June 1990.

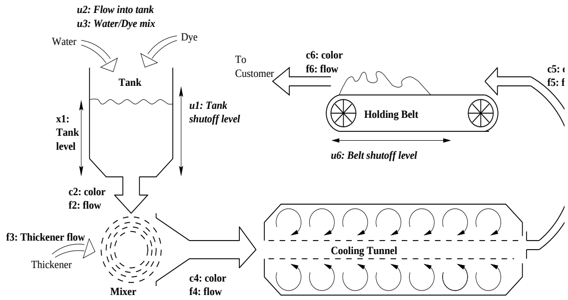

Figure 17: A simulated production process described in the text.

	Mean over all 60 trials	Mean over last 15 trials
Amoeba	25.297	27.130
Amoeba2	27.412	33.216
Amoeba4	22.302	26.534
Amoeba8	19.858	21.411
PmaxGlobal	28.006	37.237
PmaxLocal	27.634	37.952
PmaxVLocal	23.012	27.105
Q2	36.334	45.589

Figure 18: Performance on the simulated production process.

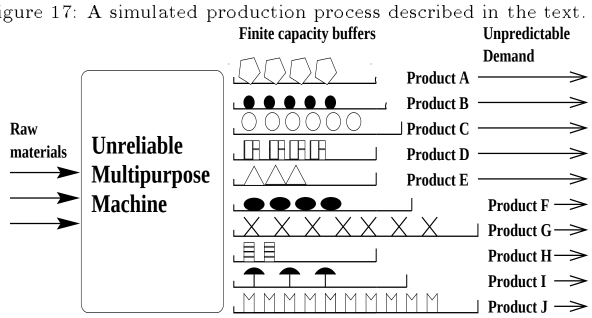

Figure 19: A multi-buffer servicing task similar to those described in [7].

	Mean over all 24 trials	Mean over last 6 trials
Amoeba	1.553	1.897
Amoeba2	1.711	2.859
Amoeba4	0.940	2.179
PmaxGlobal	-1.489	-1.380
PmaxLocal	-1.521	-1.454
PmaxVLocal	-1.565	-1.518
Q2	0.535	3.352

Figure 20: Performance at the multi-buffer task.

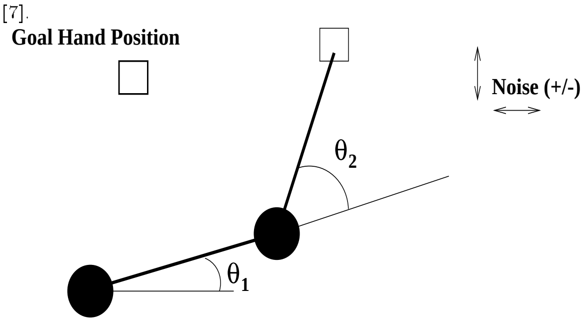

Figure 21: A 2-input, 2-output kinematics task.

	Mean over all 40 trials	Mean over last 10 trials
Amoeba	-0.084	-0.082
Amoeba2	-0.092	-0.087
Amoeba4	-0.096	-0.096
Amoeba8	-0.084	-0.071
PmaxGlobal	-0.051	-0.046
PmaxLocal	-0.050	-0.042
PmaxVLocal	-0.057	-0.054
Q2 (Linear)	-0.062	-0.023

Figure 22: Performance on the kinematics task.

[5] H. Kushner and D. Clark. *Stochastic Approximation Methods for Constrained and Unconstrained Systems.* Springer-Verlag, 1978.

[6] D. J. C. MacKay. Bayesian Model Comparison and Backprop Nets. In J. E. Moody, S. J. Hanson, and R. P. Lippman, editors, *Advances in Neural Information Processing Systems 4*. Morgan Kaufmann, April 1992.

[7] S. Mahadevan, N. Marchalleck, T. Das, and A. Gosavi. Self-Improving Factory Simulation using Continuous-Time Average-Reward Reinforcement Learning. In *Proceedings of the 14th International Conference on Machine Learning (ICML '97), Nashville, TN.* Morgan Kaufmann, July 1997.

[8] A. W. Moore, D. J. Hill, and M. P. Johnson. An Empirical Investigation of Brute Force to choose Features, Smoothers and Function Approximators. In S. Hanson, S. Judd, and T. Petsche, editors, *Computational Learning Theory and Natural Learning Systems, Volume 3*. MIT Press, 1992.

[9] A. W. Moore and J. Schneider. Memory-based Stochastic Optimization. In D. Touretzky, M. Mozer, and M. Hasselmo, editors, *Neural Information Processing Systems 8*, 1996.

[10] A. W. Moore, J. Schneider, J. Boyan, and M. S. Lee. Q2: A memory-based active learning algorithm for Blackbox Noisy Optimization. In preparation, 1998.

[11] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipes in C.* Cambridge University Press, 1992.

[12] H. Robbins and S. Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.

[13] M. Salganicoff and L. H. Ungar. Active Exploration and Learning in Real-Valued Spaces using Multi-Armed Bandit Allocation Indices. In *Proceedings of the 12th International Conference on Machine Learning.* Morgan Kaufmann, 1995.