Tuning Optimizers for Time-Constrained Problems using Reinforcement Learning

Paul Ruvolo
Department of Computer Science
University of California San Diego
La Jolla, CA 92093
pruvolo@cs.ucsd.edu

Ian Fasel
Department of Computer Sciences
University of Texas at Austin
ianfasel@cs.utexas.edu

Javier Movellan
Machine Perception Laboratory
University of California San Diego
movellan@mplab.ucsd.edu

Abstract

Many popular optimization algorithms, like the Levenberg-Marquardt algorithm (LMA), use heuristic-based "controllers" that modulate the behavior of the optimizer during the optimization process. For example, in the LMA a damping parameter λ is dynamically modified based on a set of rules that were developed using various heuristic arguments. Here we show that a modern reinforcement learning technique utilizing a very simple state space can dramatically improve the performance of general purpose optimizers, like the LMA, on problems where the number of function evaluations allowed is constrained by a budget. Results are given on both classical non-linear optimization problems as well as a difficult computer vision task. Interestingly the controllers learned for a particular optimization domain work well on other optimization domains. Thus, the controller appeared to have extracted optimization rules that were not just domain specific but generalized across a range of optimization domains.

1 Introduction

Most popular optimization algorithms, like the Levenberg-Marquardt algorithm (LMA) use simple "controllers" that modulate the behavior of the optimization algorithm based on the state of the optimization process. For example, in the LMA a damping factor λ smoothly modifies the algorithm to make it behave more like standard gradient descent versus more like Gauss-Newton optimization [1, 2]. The LMA uses the following heuristic for controlling λ: If an iteration of the LMA with the current damping factor λ_t reduces the error then the new parameters produced by the LMA iteration are accepted and the damping factor is divided by a constant term $\eta > 0$, i.e., $\lambda_{t+1} = \lambda_t/\eta$. Otherwise, if the error is not reduced, the new parameters are not accepted, the damping factor is multiplied by η, and the LMA iteration is repeated with the new damping parameter. While various heuristic arguments have been used to justify this particular way of controlling the damping factor, it is not clear whether this "controller" is optimal in any way or whether it can be significantly improved.

Improving the performance of off-the-shelf optimizers is particularly important for time-constrained optimization problems. For example the LMA algorithm has become popular for many real-time computer vision problems [3, 4, 5], such as object tracking and feature tracking in video where only

a fixed amount of time can be allocated to the optimizer on each incoming video frame. Time constrained optimization is in fact becoming an increasingly important problem in applications such as operations research, robotics, and machine perception. Given the special properties of time constrained optimization problems it is likely that the heuristic-based controllers used in off-the-shelf optimizers may not be particularly efficient. Additionally, standard techniques for non-linear optimization like the LMA do not address issues such as when to stop a fruitless local search or when to revisit a previously visited part of the parameter space.

Reinforcement learning (RL) is a machine learning approach to learn optimal controllers by examples and thus is an obvious candidate to improve the heuristic-based controllers used in the most popular and heavily used optimization algorithms. An advantage of RL methods over other approaches to optimal control is that they do not require prior knowledge of the underlying system dynamics and the system designer is free to choose reward metrics that best match the desiderata for controller performance (e.g., minimum loss within a fixed amount of time).

2 Related Work

The idea of using RL in optimization problems is not new [6, 7, 8, 9]. However, previous approaches have focused on using RL methods to develop problem-specific optimizers for NP-complete problems. Here our focus is on using RL methods to modify the controllers implicit in the most popular and heavily used optimization algorithms. In particular our goal is to make these algorithms more efficient for optimization on time budget problems. As we will soon show, a simple RL approach can result in dramatic improvements in performance over these popular optimization packages.

3 Learning to Optimize Non-linear Least Squares Functions with a Budget

The value of a given optimizer is defined as the expected percentage reduction in overall loss from the initial point to the best point the optimizer finds within a fixed budget of optimization steps. This criterion suggests a natural one-step reward function where L is a loss function we are trying to minimize, B is the budget of function evaluations, I is the indicator function, x_0 is the initial point visited in the current optimization episode, and x_{opt} is the point with the lowest loss visited in the current optimization episode:

$$r_k \quad = \quad I(k < B) \times I(L(x_k) < L(x_{opt})) \times (L(x_{opt}) - L(x_k)) \times \frac{1}{L(x_0)} \tag{1}$$

While there are a number of possible RL algorithms for learning an optimizer control policy, we chose to employ Least-Squares Policy Iteration (LSPI) [10]. LSPI is an iterative reinforcement learning procedure that repeatedly applies the following two steps until convergence: approximating the action-value function as a linear combination of a fixed set of basis functions and then improving the current policy greedily over the approximate action-value function. The bases are arbitrary functions of the state and action. The method is efficient in terms of the number of interactions required with the dynamical system and can reuse the same set of samples to evaluate multiple policies, which is a crucial difference between LSPI and earlier methods like LSTD. The output of the LSPI procedure is a weight vector that defines the action-value function of the optimal policy as a linear combination of the basis vectors.

In this paper, we focus on a set of classical non-linear optimization problems, including examples such as the Kowalik and Osborne function, Scaled Meyer function, as well as a "smile-detection" computer vision problem. Each optimization problem takes the form of a non-linear objective function with a least squares error criterion and thus is well-suited to Levenberg-Marquardt style optimization. The action space we consider in our experiments consists of adjustments to the damping factor (maintain, decrease by a multiplicative factor, or increase by a multiplicative factor) used in the LMA, the decision of whether or not to throw away the last descent step, along with two actions that are not available to the LMA. These additional actions include moving to a new random point in the domain of the objective function and also returning to the best point found so far and performing one descent step using the LMA (using the current damping factor). The number of actions available at each step is 8 (6 for various combinations of adjustments to λ and returning the previous iterate along with the 2 additional actions just described). Our goal for choosing the state space

Initialize a policy π_0 that explores randomly
$S \leftarrow \{\}$
for $i = 1$ to n **do**
  Generate a random optimization problem U
  Optimize U for T time steps using policy π_0 and generate samples $V \in (s, a, r, s')^T$
  $S \leftarrow S \cup V$
end for
repeat
  Construct the approximate action-value function $Q_t^{\pi_t}$ using the samples S
  Set π_{t+1} to be the one step policy improvement of π_t using $Q_t^{\pi_t}$
  $t \leftarrow t + 1$
until $Q_{t-1}^{\pi} \approx Q_t^{\pi}$
return π_t

Figure 1: Our algorithm for learning controllers for optimization on a budget. The construction of the approximate action-value function and the policy improvement step are performed using the techniques outlined in [10].

was to allow the controller to select actions from a set of possible actions that have been seen to be useful when used heuristically in existing optimization algorithms. The RL algorithm can then learn an optimization strategy that essentially blends or switches between different types of optimization strategies in a manner best suited to the class of problems it is trained on and its remaining optimization budget. In this case, the first four actions are based on the existing LM heuristics, and the other two are sometimes used in other optimization methods. We could also have added actions that appear in optimization routines such as conjugate-gradient steps or parameter mutations based on genetic-algorithms.

The state space used to make the action decision includes a fixed-length window of history that encodes whether a particular step in the past increased or decreased the residual error from the previous iterate. This window is set to size 2 for most of our experiments, however, we did evaluate windows of size 1, 2, and 3 (see Figure 3). Also included in the state space is the amount of function evaluations left in our budget and a problem-specific state feature described in Section 4.2.

The state and action space are mapped through a collection of fixed basis functions which the LSPI algorithm combines linearly to approximate the optimal action-value function. For most applications of LSPI these functions consist of radial-basis functions distributed throughout the continuous state and action space. The basis we use in our problem treats each action independently and thus constructs a tuple of basis functions for each action. To encode the number of evaluations left in the optimization episode, we use a collection of radial-basis functions centered at different values of budget remaining (specifically we use basis functions spaced at 4 step intervals with a bandwidth of .3). The history window of whether the loss went up or down during recent iterations of the algorithm is represented as a d-dimensional binary vector where d is the length of history window considered. For the facial expression recognition task the tuple includes an additional basis described in Section 4.2.

Our method for learning an optimization controller consists of two phases. In the first phase samples are collected through interactions between a random optimization controller and an optimization problem in a series of fixed length optimization episodes. These samples are tuples of the form (s, a, r, s') where s' denotes the state arrived at when action a was executed starting from state s and reward r was received. The second phase of our algorithm applies LSPI to learn an action-value function and implicitly an optimal policy (which is given by the greedy maximization of the action-value function over actions for a given state). A sketch of our algorithm is given in Figure 1.

4 Experiments

We demonstrate the ability of our method to both achieve superior performance to off the shelf non-linear optimization techniques as well as provide insight into the specific policies and action-value functions learned.

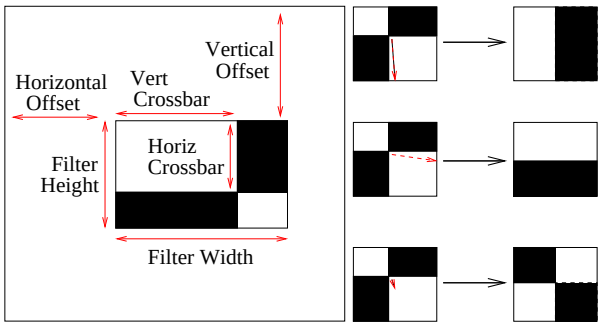


Figure 2: The parameterization of the feature space. The position of the cross-hairs in the middle of the box filter can freely float. This added generality allows for the features proposed in [12] to be generated as special cases (**right**). The weighting coefficients for the four boxes (depicted in a checkerboard pattern) are determined by linear regression between filter outputs of each box and the labels of the training set.

4.1 Classical Nonlinear Least Squares Problems

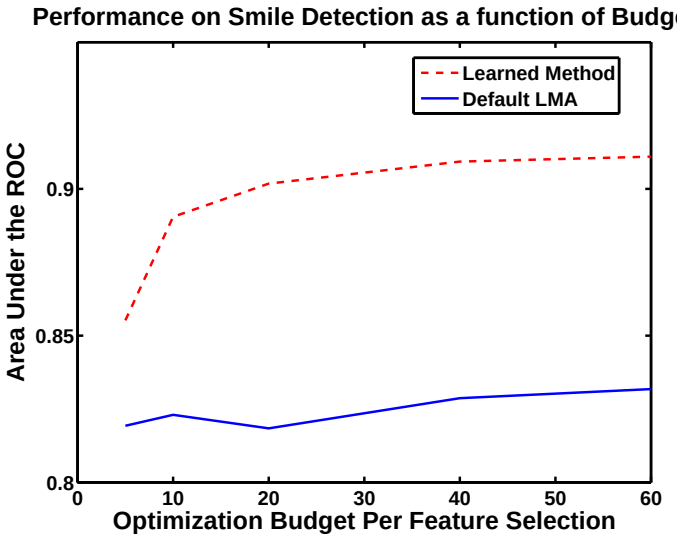
In order to validate our approach we apply it to a dataset of classical non-linear optimization problems [11]. This dataset of problems includes famous optimization problems that cover a wide variety of non-linear behavior. Examples include the Kowalik and Osborne function and the Scaled Meyer function. Our method is able to learn a policy which results in a 6% gain in performance (measured in total reduction in loss from the starting point) when compared to the LMA when the budget of function evaluations is set to 5.

4.2 Learning to Classify Facial Expressions

We learn a detector for the presence or absence of a smile using the pixel intensities of an image patch containing a face. We accomplish this by employing the sequential regression procedure L2-boost [13]. L2-boost creates a strong classifier by iteratively fitting the residuals of the current model over a collection of weak-learners (in this case a set of parameterized box-filter features described below) which are combined linearly in the final classifier. The basis space for LSPI is augmented for this task by included a basis that specifies the number of features already selected by the L2-boost procedure.

The filter response of a box filter is computed by taking the sum of the pixel intensities in a particular box, multiplying this sum by a weighting coefficient, and subsequently summing over the outputs of each box. These filters have been successful for face detection in [12] and have also shown promise for recognizing facial expressions when combined using boosting methods. We frame the problem of feature selection as an optimization procedure over a continuous parameter space. The 6 dimensional parameter space defines an infinite set of box-filters that includes many of those proposed in [12] as a special case (see Figure 2). The 6 dimensions of the feature space specify the horizontal offset, the filter height, the vertical cross bar position, the vertical offset, the horizontal crossbar, and the filter width of the box filter. The weighting coefficients are determined by standard linear regression between the outputs of the four-boxes for a particular filter and the residuals for each image.

For each trial a subset of $1,000$ images from the GENKI dataset (which is a collection of $60,000$ faces from the web) are selected as a training set while the rest of the images are used for validation. In each trial 3 box filters are selected using the L2-boosting procedure. Within each round of feature selection a total of 20 feature evaluations are allowed. The default version of the LMA is used as a mode of comparison. After collecting samples from 100 episodes of optimization, LSPI is able to learn a policy that achieves a 2.66 fold greater reduction in total loss than the LMA on a test set of faces from the GENKI dataset (see Figure 3). Since the LMA does not move to a new random part of the state space a more fair comparison can be drawn to our method without access to this action which still results in a 20% greater reduction in total loss than the LMA. Figure 3 shows that



Controller Type	Average Reduction in Loss Relative to the LMA
Learned (history window = 1)	2.3
Learned (history window = 2)	2.66
Learned (no random restarts)	1.20
Learned on Classical (no random restarts)	1.19
Default LMA	1.0

Figure 3: **Top**: The performance on detecting smile versus not smile is substantially better when using an optimization controller learned with our algorithm than using the default LMA. In each run 3 features are selected by the L2-boost procedure. The number of feature evaluations per feature (the budget) varies along the x-axis. **Bottom:** This table describes the relative improvement in total loss reduction for policies learned using our method.

the policies learned using our method achieve a substantial gain in performance for classification on a validation set of test images (between .036 and .083 better A'). Additionally our results show that adding additional information to the state space improves performance. Learning a policy that uses a history window of error changes on the last two time steps is able to achieve a 15% greater reduction in total loss than a policy learned with a history window of size 1. Using the error changes over the last three time steps does not appear to yield significant improvement over using the last two time steps.

The policies learned exhibit the following general trends. During early stages of feature selection the learned policies either sample a new point in the feature space (if the error has increased from the last iteration) or do an Levenberg-Marquardt step on the best point visited up until now (if the error has gone down at the last iteration). Later in the optimization, the policy performs a Levenberg-Marquardt step on the current best point no matter what the change in loss; this strategy reflects the diminished utility of sampling a new part of the state space at a later stage in the feature selection process. By examining the basis weights learned by LSPI we can see that the learned policy favors discarding the last iterate versus keeping (similar to the LMA) and also that the policy favors increasing the damping parameter when the error has increased on the last iteration and decreasing the damping factor when the error has decreased (also similar to the LMA).

The state space formulation is general enough to allow policies learned on one type of optimization problem to be applied to other optimization domains. The optimization controllers learned in the classical least squares minimization task achieve a 19% improvement over the standard LMA on the smile detection task. Applying the controllers learned on the smile detection task to the classical least squares problem yields a more modest 6% improvement. These results support the claim that our method is extracting useful structure for optimizing under a fixed budget and not simply learning a controller that is amenable to a particular problem domain.

5 Conclusion

We have presented a novel approach to the problem of learning optimization procedures for optimization on a fixed budget. While we have shown that our approach achieves dramatically better performance than ubiquitous methods for non-linear least squares optimization on the task of optimizing within a fixed budget of function evaluations there is likely a great deal of room for improvement. For instance, by incorporating domain specific features into the state space richer policies might be learned. We also want to apply this technique to other problems in machine perception such as finding feature point locations on a face that have high likelihood but also respect global constraints on the relative position of these feature points. The hard real-time constraints of this problem make it a particularly appropriate target for the methods presented in this document.

References

[1] K. Levenberg, "A method for the solution of certain problems in least squares," *Applied Math Quarterly*, 1944.

[2] D. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *SIAM Journal of Applied Mathematics*, 1963.

[3] D. Cristinacce and T. F. Cootes, "Feature detection and tracking with constrained local models," *BMVC*, pp. 929–938, 2006.

[4] M. Pollefeys, L. V. Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch, "Visual modeling with a hand-held camera," *IJCV*, vol. 59, no. 3, pp. 207–232, 2004.

[5] P. Beardsley, P. Torr, and A. Zisserman, "3d model acquisition from extended image sequences." *Proceedings of ECCV*, pp. 683–695, 1996.

[6] V. V. Miagkikh and W. F. P. III, "Global search in combinatorial optimization using reinforcement learning algorithms," in *Proceedings of the Congress on Evolutionary Computation*, vol. 1. IEEE Press, 6-9 1999, pp. 189–196.

[7] L. M. Gambardella and M. Dorigo, "Ant-q: A reinforcement learning approach to the traveling salesman problem," in *International Conference on Machine Learning*, 1995, pp. 252–260.

[8] J. A. Boyan and A. W. Moore, "Learning evaluation functions for global optimization and boolean satisfiability," in *AAAI/IAAI*, 1998, pp. 3–10.

[9] R. Moll, T. J. Perkins, and A. G. Barto, "Machine learning for subproblem selection," in *ICML '00: Proceedings of the Seventeenth International Conference on Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, pp. 615–622.

[10] M. Lagoudakis and R. Parr, "Least-squares policy iteration," *Journal of Machine Learning Research*, 2003.

[11] H. B. Nielsen, "Uctp problems for unconstrained optimization," *Technical Report, Technical University of Denmark*, 2000.

[12] P. Viola and M. Jones, "Robust real-time object detection," *International Journal of Computer Vision*, 2002.

[13] P. Buhlmann and B. Yu, "Boosting with the l2 loss: Regression and classification," *Journal of the American Statistical Association*, 2003.