An Optimization Algorithm Based on Active and Instance-Based Learning

Olac Fuentes and Thamar Solorio

Instituto Nacional de Astrofísica, Óptica y Electrónica
Luis Enrique Erro 1
Santa María Tonantzintla, Puebla, México 72840

Abstract. We present an optimization algorithm that combines active learning and locally-weighted regression to find extreme points of noisy and complex functions. We apply our algorithm to the problem of interferogram analysis, an important problem in optical engineering that is not solvable using traditional optimization schemes and that has received recent attention in the research community. Experimental results show that our method is faster than others previously presented in the literature and that it is very accurate for the case of noiseless interferograms, as well as for the case of interferograms with two types of noise: white noise and intensity gradients, which are due to slight missalignments in the system.

Keywords: Optimization, active learning, instance-based learning, locally-weighted regression

1 Introduction

Optimization in poorly modelled, noisy and complex domains is an important problem that is faced in many scientific and engineering areas. In such domains, traditional optimization algorithms, such as the Simplex method [1] or the Levenberg-Marquardt algorithm [2], do not yield satisfactory results and are usually very sensitive to the starting search points provided by the user. For these reasons, non-traditional optimization algorithms, including simulated annealing [3], genetic algorithms [4,5], evolution strategies [6,7] and hybrid evolutionary-classical algorithms [8], have been proposed. While good results have been reported, the running times of these algorithms are often high, and they are not well suited to all domains. Thus, more efficient and complementary algorithms are desirable.

In this paper we propose an efficient algorithm to perform optimization in complex domains. Our algorithm is based on the observation that the candidate solutions generated by an optimization algorithm, which are normally discarded by both traditional and non-traditional schemes, can be used as a training set for a learning algorithm, which in turn can predict the parameters of an optimal solution to the problem. An advantage of this approach is that, if we want to find the solutions to several similar problems, we can process them concurrently

R. Monroy et al. (Eds.): MICAI 2004, LNAI 2972, pp. 242–251, 2004.

and integrate all the candidate solutions in a single training set. Since the training set is continuously changed, we need a learning algorithm that requires a small training time. Instance-based learning algorithms, whose training consists of simply storing the training data, fulfill this requirement. In our work we used the locally-weighted regression algorithm [9], an instance-based learning algorithm that has been found to yield similar accuracy as neural networks in many application domains while preserving the short training times inherent to this class of methods.

We illustrate our method with an application to the problem of interferogram analysis, which has received recent attention in the literature. This is an interesting domain, as there are published attempted solutions using both traditional optimization schemes and evolutionary algorithms [10,11].

The organization of the remainder of this paper is as follows. In Section 2 we describe the proposed optimization algorithm, the main contribution of this paper. Section 3 presents background material about interferometry, the application area we use to illustrate the algorithm. Section 4 gives details about the adaptation of the algorithm to the problem. Section 5 shows the main results, and Section 6 presents conclusions and suggests directions for future work.

2 Outline of the Optimization Algorithm

We are interested in the problem of finding the parameters of a known analytic function that best match an observation. Let $\mathbf{o}$ be the observed (multidimensonal) variable, let $\mathbf{f}(\mathbf{x})$ be a function with the same dimensionality as $\mathbf{o}$. The goal of the optimization procedure is to obtain the value of $\mathbf{x}$ that minimizes $|\mathbf{o} - \mathbf{f}(\mathbf{x})|$. Typically, this problem is solved by an iterative process: in iteration i we generate $\mathbf{x_i}$, evaluate the target function $|\mathbf{o} - \mathbf{f}(\mathbf{x_i})|$ and based on the value of the target function, as well as its first and second derivatives (if they are available), we generate the next candidate value $\mathbf{x_{i+1}}$, which is expected to be closer to the optimum.

This work deals with the problem where we have several observations $\mathbf{o_1}, \ldots, \mathbf{o_n}$, and we want to find the vectors $\mathbf{x_1}, \ldots, \mathbf{x_n}$ that minimize the errors $e_i = |\mathbf{o_i} - \mathbf{f}(\mathbf{x_i})|$. Clearly, this can be solved by solving the n optimization problems separately. However, we propose a method to solve the problem more efficiently, posing it as a learning problem, where a learning algorithm learns the inverse function $\mathbf{f^{-1}}(\mathbf{x})$. The training set used by the algorithm is formed by the pairs of values $\langle \mathbf{f}(\mathbf{x_i}), \mathbf{x_i} \rangle$ previously generated in the search, its test set consists of the values $\mathbf{o_1}, \ldots, \mathbf{o_n}$ and it outputs an estimate of $\mathbf{x_1}, \ldots, \mathbf{x_n}$ that is expected to minimize $e_1, \ldots e_n$. When a new set of solutions $\mathbf{x_1}, \ldots, \mathbf{x_n}$ is proposed by the algorithm, we compute their corresponding $\mathbf{f}(\mathbf{x_1}), \ldots, \mathbf{f}(\mathbf{x_n})$ and use the new pairs $\langle \mathbf{f}(\mathbf{x_i}), \mathbf{x_i} \rangle$ to augment the training set, and continue this iterative process until convergence is attained. Since this type of active learning adds to the training set examples that are progressively closer to the points of interest, the errors are guaranteed to decrease in every iteration. The outline of the algorithm can be described by the following pseudocode:

1. Generate randomly an initial set of values $\mathbf{x_1}, \ldots, \mathbf{x_m}$ and compute their corresponding $\mathbf{f}(\mathbf{x_1}), \ldots, \mathbf{f}(\mathbf{x_m})$.
2. Let $R = \{\langle \mathbf{f}(\mathbf{x_1}), \mathbf{x_1} \rangle, \ldots, \langle \mathbf{f}(\mathbf{x_m}), \mathbf{x_m} \rangle\}$ be the initial training set.
3. Let $T = \mathbf{o_1}, \ldots, \mathbf{o_n}$ be the test set.
4. While T is not empty
   a) Train an approximator $\mathbf{A}$ using R as training set
   b) For each $\mathbf{o_i} \in T$
      i. Generate $\mathbf{A}(\mathbf{o_i})$
      ii. $R = R \cup \{\langle \mathbf{f}(\mathbf{A}(\mathbf{o_i})), \mathbf{A}(\mathbf{o_i}) \rangle\}$
      iii. If $|\mathbf{o_i} - \mathbf{f}(\mathbf{A}(\mathbf{o_i}))| < \mathit{threshold}$, remove $\mathbf{o_i}$ from T.

Here, $\mathbf{A}(\mathbf{o_i})$ can be seen as the best current guess for the value of $\mathbf{x}$ such that $\mathbf{f}(\mathbf{x}) = \mathbf{o_i}$. For the algorithm to be efficient, we need to minimize the time taken to train A. This can easily be done if we use an instance-based learning algorithm, such as the locally-weighted regression algorithm (explained in the next subsection). The other potentially time consuming step is the application of $\mathbf{A}$ to compute $\mathbf{A}(\mathbf{o_i})$, as it normally takes time proportional to the size of the training set to find the nearest neighbors of each example in the test set. However, since the algorithm is applied repeatedly to the same test set, we can cache the nearest neighbors of each example in the test set, and every time the training set is augmented (step 4.b.ii) we can check if the example added to the training set becomes a nearest-neighbor of any of them.

2.1 Locally-Weighted Regression

Locally-Weighted Regression (LWR) belongs to the family of instance-based learning algorithms. In contrast to most other learning algorithms, which use their training examples to construct explicit global representations of the target function, instance-based learning algorithms simply store some or all of the training examples and postpone any generalization effort until a new instance must be classified. They can thus build query-specific local models, which attempt to fit the training examples only in a region around the query point. In this work we use a linear model around the query point to approximate the target function.

Given a query point $\mathbf{x_q}$, to predict its output parameters $\mathbf{y_q}$, we find the k examples in the training set that are closest to it, and assign to each of them a weight given by the inverse of its distance to the query point: $w_i = \frac{1}{|\mathbf{x_q} - \mathbf{x_i}|}$. Let W, the weight matrix, be a diagonal matrix with entries $w_1, \ldots, w_n$. Let X be a matrix whose rows are the vectors $\mathbf{x_1}, \ldots, \mathbf{x_k}$, the input parameters of the examples in the training set that are closest to $\mathbf{x_q}$, with the addition of a "1" in the last column. Let Y be a matrix whose rows are the vectors $\mathbf{y_1}, \ldots, \mathbf{y_k}$, the output parameters of these examples. Then the weighted training data are given by $Z = WX$ and the weighted target function is $V = WY$. Then we use the estimator for the target function $\mathbf{y_q} = \mathbf{x_q}^T (Z^T Z)^{-1} Z^T V$.

Thus, locally weighted linear regression is very similar to least-squares linear regression, except that the error terms used to derive the best linear approximation are weighted by the inverse of their distance to the query point. Intuitively,

this yields much more accurate results than standard linear regression because the assumption that the target function is linear does not hold in general, but is a very good approximation when only a small neighborhood is considered.

3 Interferometry

Interferometry is a laboratory technique very commonly used to test the quality of optical systems. To perform interferometry, two beams, one passing through a reference surface and the other passing through the test surface, are combined and made to interfere, which results in a pattern, called interferogram, that characterizes the quality of the test surface. A schematic diagram of a simple interferometer is shown in figure 1. Experienced technicians can diagnose the flaws of the test surface by careful analysis of the interferogram, however, this is a time consuming task, and when there is a need to analyze more than a few interferograms, it becomes impractical. Thus, there is a need for techniques to automate this process.

The problem of automatically characterizing an interferogram has received recent attention in the literature. This is a difficult problem, and traditional optimization schemes based on the least-squares method often provide inconclusive results, specially in the presence of noisy data [12,13]. For this reason, non-traditional optimization schemes, such as evolutionary algorithms, have been proposed to solve this problem [11]. While evolutionary algorithms provide very accurate results in the case of both noiseless and noisy data, their running time is high, taking several minutes to analyze a single interferogram. Clearly, if we need to analyze a large number of interferograms, this approach becomes unfeasible.

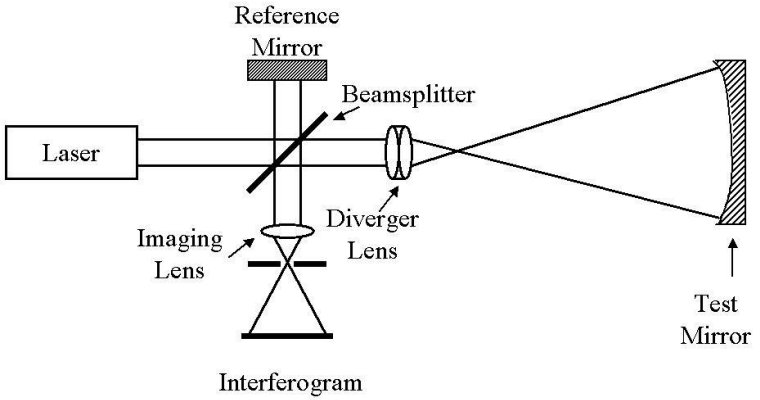


Fig. 1. A simple interferometer

3.1 Interferogram Simulation

To obtain the simulated interferograms we use Kingslake's formulation [14], where the intensity in the interferogrametric image is given by

$$I(x,y) = cos(\frac{2\pi}{\lambda} W(x,y)) + G(x,y) + N(x,y) \quad (1)$$

where λ is the wavelength of the light source, N is white noise which we represented as a random number obtained from a Gaussian distribution with zero mean, G is a noise term due to slight missalignments in the system and appears in the image as an intensity gradient and can be defined by three parameters γ_1, γ_2 and γ_3:

$$G(x,y) = \gamma_1 + \gamma_2 x + \gamma_3 y \quad (2)$$

$W(x,y)$ is the optical path difference (OPD) between the reference and test surfaces, and it is represented by a polynomial using the Seidel aberration formulation:

$$W(x,y) = A(x^2+y^2)^2 + By(x^2+y^2) + C(x^2+3y^2) + D(x^2+y^2) + Ey + Fx \quad (3)$$

where A is the spherical aberration coefficient, B, C and D are the comma coefficient, astigmatism and defocusing coefficients, respectively, E is the tilt about the y axis, and F is the tilt about the x axis.

Clearly, it is easy to obtain an interferogram I given the vector of aberration coefficients $v = [A, B, C, D, E, F]$ and the vector of intensity gradients $\gamma = [\gamma_1, \gamma_2, \gamma_3]$. However, we are interested in the inverse problem, that is, obtaining the vector of aberration coefficients and intensity gradient from the corresponding interferogram, which, as mentioned before, is a very difficult optimization problem. The following section will describe our proposed solution to this problem.

4 Automated Interferogram Analysis

The problem of finding the parameters that characterize an optical system is known as interferogram analysis. In this section we show the application of our proposed optimization method to the problem of interferogram analysis.

4.1 Preprocessing

The input to our system is a set of interferograms and the output is a vector of aberration and gradient coefficients that characterize them. Before we apply the optimization algorithm, we can greatly reduce the dimensionality of the learning task using a principal component analysis preprocessing stage to compress the high-dimensional interferograms into a more manageable size with minimal loss of information.

Principal Component Analysis. The formulation of standard PCA is as follows. Consider a set of m vectors $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_m}$, where the mean object of the set is defined by

$$\mu = \frac{1}{m} \sum_{i=1}^{m} \mathbf{v_i} \tag{4}$$

Each object differs from the mean by the vector

$$\theta_\mathbf{i} = \mathbf{v_i} - \mu \tag{5}$$

Let $A = [\theta_1, \theta_2, \ldots, \theta_m]$. C, the covariance matrix, is given by

$$C = \sum_{i=1}^{m} \sum_{j=1}^{m} \theta_i \theta_j^T = AA^T \tag{6}$$

The principal components are then the eigenvectors of C. If we sort these eigenvectors by decreasing order of their corresponding eigenvalues, a projection onto the space defined by the first k eigenvectors ($1 \leq k \leq m$) is optimal with respect to information loss. That is, let P be the matrix whose columns are the first k eigenvectors of C, then the optimal projection of $\mathbf{v_i}$ is given by

$$\mathbf{p_i} = P^T \theta_i \tag{7}$$

4.2 Optimization Algorithm

The algorithm to perform automated interferogram analysis can be described as follows. First we generate randomly k parameter vectors $\mathbf{x_o}, \ldots, \mathbf{x_k}$, where $\mathbf{x_i} = [A_i, B_i, C_i, D_i, E_i, F_i, \gamma_{1i}, \gamma_{2i}, \gamma_{3i}]$ contains aberration and gradient coefficients. For each $\mathbf{x_i}$ we construct the corresponding interferogram $I(\mathbf{x_i})$ applying equations 1, 2 and 3. Then we perform PCA on a matrix $J = [I(\mathbf{x_1}), \ldots, I(\mathbf{x_k})]$, obtaining P, the matrix of principal components, and μ, the mean vector, as described in 4.1. The projection of each interferogram into the eigenspace is then given by $\mathbf{p_i} = P^T(I(\mathbf{x_i}) - \mu)$. We can now use the set of pairs $R\langle \mathbf{p_i}, \mathbf{x_i} \rangle$ as initial training set to the algorithm.

Given a set of test $T_1, \ldots T_m$ interferograms, we first project them to the eigenspace created in the previous step, $\mathbf{t_i} = P^T(T_i - \mu)$, and we give these projections as the test set to the learning algorithm described in Section 2.

5 Experimental Results

In this section we describe the experiments performed with our optimization algorithm applied to the problem of predicting the vectors of aberration coefficients and intensity gradients. First, we generated a thousand aberration vectors, a thousand intensity gradient vectors and their corresponding interferograms, using an 81 by 81 resolution. For this experiment we dealt with noiseless interferograms (that is, the noise terms G and N were set to zero). Using principal

Table 1. Mean Absolute Errors for Simulated Interferograms

Coefficients	Mean Absolute Error	Standard Deviation
A	0.0011	0.0010
B	0.0015	0.0011
C	0.0006	0.0005
D	0.0010	0.0009
E	0.0002	0.0002
F	0.0004	0.0005

Table 2. Mean Absolute Errors for Noisy Simulated Interferograms

Coefficients	Mean Absolute Error	Standard Deviation
A	0.1109	0.1525
B	0.0887	0.0829
C	0.0398	0.0697
D	0.0591	0.0506
E	0.0250	0.0526
F	0.0463	0.0800
γ_1	0.0080	0.0023
γ_2	0.0100	0.0024
γ_3	0.0030	0.0010

component analysis we reduced the dimensionality of the task, keeping 47 eigenvectors, which preserve about 95% of the information in the original data. Then we randomly divided the data into ten equally sized subgroups, one group was used for testing and the remainder nine were considered the training set. Ten different experiments were performed, each one using a different group for testing. We repeated this procedure ten times, and the overall average are the results presented here. Table 1 shows averaged mean absolute errors and standard deviations for each aberration coefficient. As the experimental results show, our method is very accurate with the simulated interferograms. On average, each interferogram took 1.6 seconds to process, which is much faster than the results reported in recent works dealing with the same problem. For example, [11] reports that evolution strategies took about 3 minutes to find the parameters of each interferogram, using the same resolution and similar computing hardware.

Real data always pose the challenge of managing noise. In order to evaluate the noise sensibility of our method, we performed experiments on interferograms with simulated noise, using both Gaussian noise and an intensity gradient, as described in Section 3. Table 2 shows errors in aberration coefficients and intensity gradients. In Figure 2 we can see a visual comparison between noisy interferograms and the interferograms obtained from the predicted aberrations. It can be

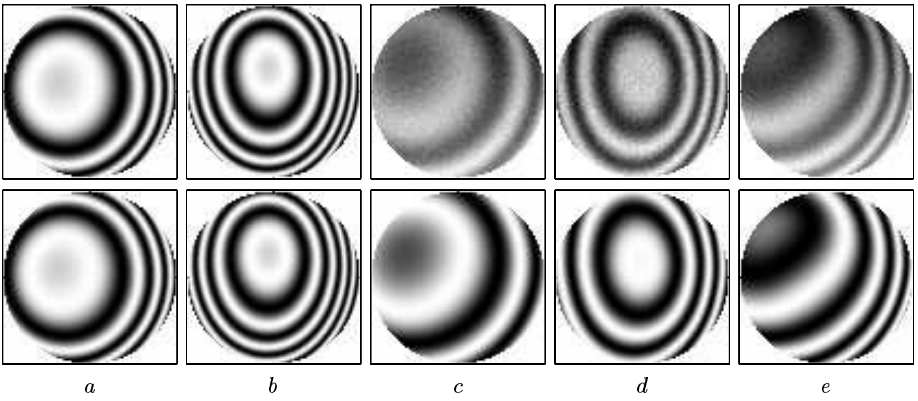


Fig. 2. The top row shows five of the interferograms used for testing and the bottom row shows the best matches found by the algorithm. Columns a and b show noiseless interferograms; it can be seen that the matches found by the algorithm are virtually undistinguishable from the test interferograms. Columns c, d and e show noisy interferograms; the matches found by the algorithm show almost identical interferograms, except that the noise has been removed.

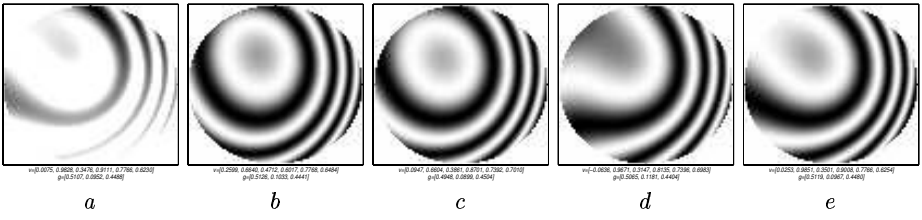
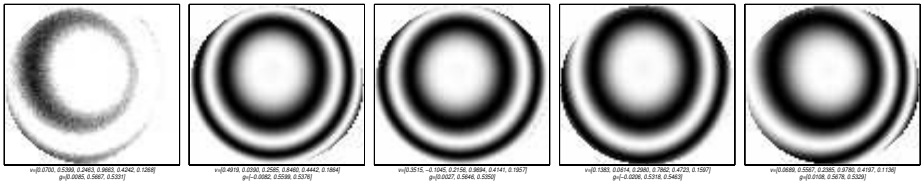


Fig. 3. A detailed trace of the optimization algorithm using a noiseless interferogram as input. The test interferogram is shown in a; b shows the result of applying LWR using only the original (randomly generated) training data. Figures c and d show successive approximations given by the algorithm as it iterates toward a solution, finally, the best result found is shown in e.

seen that our method's performance was not damaged by the noise in the test data. For this case, due to the fact that the parameter space is increased, the running time is also increased, taking 9 seconds to find the optimal parameters of each interferogram, on average. As this is, to the best of our knowledge, the first attempt to approximate interferograms using a noise model that is more complex than simple Gaussian noise, we cannot compare our results with previous approaches, however, the running time is still much smaller than that taken by the method that only deals with noiseless interferograms.

In Figure 3 we present a detailed execution trace of the algorithms, using a randomly chosen noiseless test example. We can see how the algorithm is gradually converging to a set of parameters that generate an interferogram that is



a b c d e

Fig. 4. A detailed trace of the optimization algorithm using a noisy interferogram as input. The test interferogram is shown in *a*; *b* shows the result of applying LWR using only the original (randomly generated) training data. Figures *c* and *d* show successive approximations given by the algorithm as it iterates toward a solution, finally, the best result found is shown in *e*.

virtually undistinguishable from the test interferogram. Figure 4 shows a similar trace, except that the input is now a noisy interferogram. The output of the algorithm is again an almost exact match to the test interferogram, except that the noisy has been eliminated.

6 Conclusions

In this paper we have presented an optimization algorithm that has a very strong feature: the ability of extending the training set automatically in order to best fit the target function for the test data. There is no need for manual intervention, and if new test instances need to be classified the algorithm will generate as many training examples as needed.

We have shown experimental results of the application of our method to solve the problem of, given a large set of interferograms, finding their corresponding vectors of aberration coefficients. The method yields very accurate results, even in the presence of noise, and also, it is faster by two orders of magnitude than other methods introduced earlier.

Present and future work includes:

- Testing the method using real interferograms.
- Extending the algorithm to handle higher-order aberrations.
- Testing the applicability of the method to other optimization problems in optics, as well as in other areas of science.

Acknowledgements. We would like to thank CONACYT for partially supporting this work and Sergio Vázquez y Montiel and Jaime Sánchez Escobar for stimulating discussions.

References

1. J. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.
2. K. Levenberg. A method for the solution of certain problems in least squares. *Quarterly Journal on Applied Mathematics*, 2:164–168, 1944.
3. S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.
4. John Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, MI, 1975.
5. David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, 1989.
6. Hans-Per Schwefel. *Numerical Optimization of Computer Models.* John Wiley & Sons, Ltd., 1981.
7. Thomas Bäck, Frank Hoffmeister, and Hans-Paul Schwefel. A survey of evolutionary strategies. In *Proceedings of the Fourth International Conference on Genetic Algorithms.* Morgan Kaufmann Publishers, Inc, 1991.
8. Nicholas J. Radcliffe and Patrick D. Surry. Formal memetic algorithms. In *Evolutionary Computing, AISB Workshop*, pages 1–16, 1994.
9. Christopher G. Atkeson, Andrew W. Moore, and Stefan Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11:11–73, 1997.
10. A. Cordero-Dávila, A. Cornejo-Rodríguez, and O. Cardona-Núñez. Polynomial fitting of interferograms with gaussian errors on fringe coordinates. I: Computer simulations. *Applied Optics*, 33:7343–7349, 1994.
11. S. Vázquez y Montiel, J. Sánchez, and O. Fuentes. Obtaining the phase of an interferogram using an evolution strategy, part I. *Applied Optics*, 41(17):3448–3452, June 2002.
12. D. Dutton, A. Cornejo, and M. Latta. A semiautomatic method for interpreting shearing interferograms. *Applied Optics*, 7:125–131, 1968.
13. J. Y. Wang and D. E. Silva. Wave-front interpretation with Zernike polynomials. *Applied Optics*, 19:1510–1518, 1980.
14. R. Kingslake. *Applied Optics and Optical Engineering.* Academic Press, 1979.