Reliability Engineering and System Safety 79 (2003) 245–252

RELIABILITY ENGINEERING & SYSTEM SAFETY

www.elsevier.com/locate/ress

An optimization algorithm based on stochastic sensitivity analysis for noisy objective landscapes

H. Okano[a], M. Koda[b,*]

[a]IBM Research, Tokyo Research Laboratory, 1623-14 Shimotsuruma, Yamato, Kanagawa 242-8502, Japan

[b]Institute of Policy and Planning Sciences, University of Tsukuba, 1-1-1 Tennoudai, Tsukuba Science City, Tsukuba 305-8573, Japan

Abstract

A function minimization algorithm that updates solutions based on approximated derivative information is proposed. The algorithm generates sample points with Gaussian white noise, and approximates derivatives based on stochastic sensitivity analysis. Unlike standard trust region methods which calculate gradients with n or more sample points, where n is the number of variables, the proposed algorithm allows the number of sample points M to be less than n. Furthermore, it ignores small amounts of noise within a trust region. This paper addresses the following two questions: how does the derivative approximation become worse when the number of sample points is small? Can the algorithm find good solutions with inexact derivative information when the objective landscape is noisy? Through intensive numerical experiments using quadratic functions, the algorithm is shown to be able to approximate derivatives when M is about $n/10$ or more. The experiments using a formulation of the traveling salesman problem show that the algorithm can find reasonably good solutions for noisy objective landscapes with inexact derivatives information.

Keywords: Optimization algorithm; Stochastic sensitivity; Gradient method; Traveling salesman problem

1. Introduction

Optimization problems that seek for minimization (or equivalently maximization) of an objective function have practical importance in various areas. Once a task is modeled as an optimization problem, general optimization techniques become applicable; e.g. linear programming, gradient methods, etc. One may encounter difficulties, however, in applying these techniques when the objective function is non-differentiable or when it is defined as a procedure. The example problem considered in this paper involving such a function is a parametric local search for the *traveling salesman problem* (TSP), a representative combinatorial optimization problem, in which an objective function of a parameter vector is defined as a heuristic procedure. Another example, which has been studied by the authors [1] but which is not covered in this paper, is a model involving a step function. For both cases, the associated objective landscapes are noisy in the sense that they include many non-differentiable points and many local minima.

In this paper, an unconstrained function minimization algorithm that updates solutions based on derivative information approximated with sample points is proposed for such noisy landscapes. The assumption of this study is that the number of sample points may be less than the dimension n of the objective function. Note that standard trust region methods require n or more sample points (see Refs. [2–5] and references therein). For example, direct search methods that approximate derivatives by linear interpolation maintain $n+1$ non-degenerate points within a trust region, and search methods that use quadratic polynomial interpolation require $(n+1)(n+2)/2$ non-degenerate points. To evaluate practicality of the proposed algorithm, the following questions may be helpful:

- How does the derivative approximation become worse when the number of sample points is small?
- Can the algorithm find good solutions with inexact derivative information when the objective landscape is noisy?

This paper aims to answer these questions through numerical experiments.

The optimization algorithm proposed in this paper assumes that the objective function is scaled such that an area formed by the Gaussian distribution of unit variance can be used as a trust region, and uses stochastic sensitivity

* Corresponding author. Tel.: +81-298-535222; fax: +81-298-553849.
E-mail address: koda@shako.sk.tsukuba.ac.jp (M. Koda).

0951-8320/03/$ - see front matter
PII: S0951-8320(02)00236-3

analysis to approximate derivatives. The method is to inject Gaussian white noise into each of the variables in the current solution, and approximate the first derivatives using the injected noise. A mathematical framework of the derivative approximation based on stochastic sensitivity analysis is also presented.

The paper is organized as follows. In Section 2, the optimization algorithm is described, and the mathematical framework of the derivative approximation is presented. Section 3 shows numerical experiments to answer the first question. In Section 4, the algorithm is applied to the TSP with a noisy objective landscape to answer the second question. Finally, Section 5 summarizes the paper.

2. Stochastic gradient method

Consider an unconstrained optimization problem

$$\min_{x\in\mathbb{R}^n} f(x) \in \mathbb{R}. \tag{1}$$

Gradient methods iteratively update the current solution x as

$$x \leftarrow x + \mu\frac{\delta x}{|\delta x|}, \qquad \delta x = -\nabla f(x), \tag{2}$$

where μ is a step width determined by line search, δx is a descent direction, and $\nabla f(x)$ is a gradient vector defined by

$$\nabla f(x) = \left(\frac{\partial f(x)}{\partial x_1}, \frac{\partial f(x)}{\partial x_2}, \ldots, \frac{\partial f(x)}{\partial x_n}\right)^{\mathrm{T}}. \tag{3}$$

To avoid the exact gradient calculation of Eq. (3), and to cope with noisy objective landscapes, Koda and Okano proposed a noise-based gradient method for artificial neural network learning [1], and further modified the method for function minimization [6]. The algorithm, called *stochastic noise reaction* (SNR), injects a Gaussian white noise, $\xi_i \in N(0,1)$, into a variable x_i as

$$x_i(j) = x_i + \xi_i(j), \tag{4}$$

where $\xi_i(j)$ denotes the jth realization of the noise injected into the ith variable. Each component of a derivative is approximated without explicitly differentiating the objective function by using

$$\frac{\partial f(x)}{\partial x_i} \approx \frac{1}{M}\sum_{j=1}^{M} f(x(j))\xi_i(j), \tag{5}$$

where M is a loop count for taking the average. Note that, in Eq. (5), all the components in the gradient $\nabla f(x)$, i.e. $\partial f(x)/\partial x_i$, $i = 1,2,\ldots,n$, are computed at the same time, which means the gradient approximation requires the objective function to be evaluated M times, so that the dimension n does not explicitly dominate the computational overhead. The value of M was set to 100 in all of the numerical experiments here.

In this paper, SNR is used within the algorithmic framework described in Fig. 1. A noise $\xi_i(j)$ for each variable x_i is formally generated in Step 6, while, in actual implementations, all the realizations of the noise should be generated and normalized in advance to ensure $\langle\xi_i\rangle = 0$ and $\mathrm{var}(\xi_i) = 1$, where $\langle\cdot\rangle$ denotes the expectation operator. Gaussian noise $\xi_i \in N(0,1)$ can be generated and normalized as $\xi_i(j) := \tilde{\xi}_i(j) - (1/M)\sum_{j=1}^{M}\tilde{\xi}_i(j)$, where $\tilde{\xi}_i(j) := \cos(2\pi a)\sqrt{-2\log b}$, and $a, b \in (0,1) \subset \mathbb{R}$. Noise sequences with $\mathrm{var}(\xi_i) \not\approx 1$ should be discarded and regenerated.

In Step 12, the next solution is searched for using line search by sampling, in which the maximum displacement of the sampling point farthest from the current solution is 1. When more than two solutions on the line $x + 0.01s(\delta x/w)$, $s = 1,2,\ldots,100$, share the same minimum value, the one with larger value of s is selected so that the search does not stay within a small region.

Algorithmic framework of SNR

1: Initialize the current solution $x := x^0$.
2: Initialize the best solution $x^{best} := x$.
3: **For** $k := 1,2,\ldots,N$ **do begin** // N iterations.
4: Initialize descent direction $\delta x := 0$.
5: **For** $j := 1,2,\ldots,M(=100)$ **do begin** // Derivative approximation.
6: Generate a noise vector $\xi(j)$. // $\xi_i(j) \in N(0,1)$, $i = 1,2,\ldots,n$.
7: $x_i(j) := x_i + \xi_i(j)$. // Inject the noise into the solution.
8: $\delta x_i := \delta x_i - f(x(j))\xi_i(j)$. // $\langle f(x+\xi)\xi_i\rangle \approx \frac{\partial f(x)}{\partial x_i}$.
9: **If** $f(x^{best}) > f(x(j))$ **then** $x^{best} := x(j)$. // Save the best solution.
10: **end;**
11: $w := \max_i |\delta x_i|$. // Find max component in δx.
12: $s := \arg\min_{s=1,2,\ldots,100} f(x + 0.01s\frac{\delta x}{w})$. // Line search by sampling.
13: $x := x + \mu\frac{\delta x}{|\delta x|}$ where $\mu = 0.01s\frac{|\delta x|}{w}$. // Update the current solution.
14: **If** $f(x^{best}) > f(x)$ **then** $x^{best} := x$. // Save the best solution.
15: **If** terminal condition is met **then goto** 17.
16: **end;**
17: Output x^{best}.

Fig. 1. Algorithmic framework of SNR.