Proceedings of Machine Learning Research 147:289–298, 2021 ISIPTA 2021

Stochastic Optimization for Numerical Evaluation of Imprecise Probabilities

Nicholas Syring NSYRING@IASTATE.EDU
Department of Statistics, Iowa State University, USA
Ryan Martin RGMARTI3@NCSU.EDU
Department of Statistics, North Carolina State University, USA

Abstract

In applications of imprecise probability, analysts must compute lower (or upper) expectations, defined as the infimum of an expectation over a set of parameter values. Monte Carlo methods consistently approximate expectations at fixed parameter values, but can be costly to implement in grid search to locate minima over large subsets of the parameter space. We investigate the use of stochastic iterative root-finding methods for efficiently computing lower expectations. In two examples we illustrate the use of various stochastic approximation methods, and demonstrate their superior performance in comparison to grid search.

Keywords: inferential model; logistic regression; lower expectation; Monte Carlo; plausibility function; stochastic approximation

1. Introduction

Applications of imprecise probability require the evaluation of lower and upper expectations. That is, let $\mathscr{P}$ denote a collection of probability distributions P and, if g is a P-integrable function for each $P \in \mathscr{P}$, then let

$$\underline{P}g = \inf_{P\in\mathscr{P}} Pg \quad \text{and} \quad \overline{P}g = \sup_{P\in\mathscr{P}} Pg$$

denote the lower and upper expectation of g, respectively, where $Pg = \int g(y)\,P(dy)$ is the ordinary expectation. Often times the convex hull of $\mathscr{P}$ defines the set of statistical models under consideration. However, since expectation is a linear operator $\underline{P}g$ and $\overline{P}g$ are also the infimum and supremum of g over the convex hull of $\mathscr{P}$, so it is sufficient to minimize and/or maximize g over $\mathscr{P}$. In practical problems, the collection $\mathscr{P}$ typically will be indexed by a finite-dimensional parameter $\theta \in \Theta \subseteq \mathbb{R}^q$, $q \geq 1$, which provides both simplicity and flexibility. That is, let

$$M(\theta) = \int g_\theta(y)\,P_\theta(dy),$$

now for a class of functions $\{g_\theta : \theta \in \Theta\}$, indexed by Θ, and consider the optimization problem

$$\inf_\theta M(\theta) \quad \text{and/or} \quad \sup_\theta M(\theta).$$

Robbins and Monro (1951) provided a classical example of this problem. Consider a linear regression model $M(x_i) = x_i^\top \beta + \varepsilon_i$ where $y_i = M(x_i)$ is a response, x_i is a vector of covariates, and ε_i, $i = 1,\ldots,n$, are independent, mean-zero random variables. Usually, the goal is to estimate the parameter β given observations (y_i, x_i), $i = 1,\ldots,n$. However, the real interest is often to find the optimal input x to produce a desired response y. For example, farmers would be interested in maximizing crop yield by varying the amounts of fertilizers and pesticides applied to their fields. In such a case, it is not necessary to assume linearity or any particular form of the regression function. To maximize $M(x)$, one could implement a fixed design, which is akin to a (stochastic) grid search. Alternatively, Robbins and Monro (1951) provide an iterative method to maximize $M(x)$, discussed in Section 2, which can be understood as a method of optimal sequential design in regression models.

Another general example arises in the context of statistical inference, and the so-called inferential model framework presented in Liu and Martin (2015) and Liu and Martin (2020). Those methods rely on nested random sets or, alternatively, possibility measures, which require maximizing the expectations of certain (data- and) θ-dependent functions. For example, one relatively simple inferential model construction (see, e.g., Martin, 2015, 2018) defines a plausibility contour

$$\pi(\theta; y_{\text{obs}}) = P_\theta\{T(Y,\theta) > T(y_{\text{obs}},\theta)\},$$

where y_{obs} denotes the observed data, Y is a random element having distribution P_θ, and $T(y,\theta)$ is a suitably chosen scalar-valued function. See Section 4.2 for an example. If A is some assertion or hypothesis about the unknown value of θ, the inferential model's upper probability, given the observed data y_{obs}, is the supremum

$$\overline{\Pi}(A; y_{\text{obs}}) = \sup_{\theta\in A} \pi(\theta; y_{\text{obs}}),$$

which is a special case of the general problem presented above, with $g_\theta(y) = 1\{T(y,\theta) > T(y_{\text{obs}},\theta)\}$.

Recently, several authors explored Monte Carlo sampling methods for evaluating a lower or upper expectation. The basic idea is Monte Carlo with a grid search. That is, let θ_j,

$j \in 1, \ldots, J$ denote a finite subset of Θ and, for each j, let

$$\widehat{M}(\theta_j) = \frac{1}{N} \sum_{n=1}^{N} g_{\theta_j}(Y_n^{(j)}), \quad Y_n^{(j)} \sim P_{\theta_j},$$

be a Monte Carlo estimate of $M(\theta_j)$. Then the lower and upper expectations can be readily approximated by $\min_j \widehat{M}(\theta_j)$ and $\max_j \widehat{M}(\theta_j)$, respectively.

Decadt et al. (2019) show such Monte Carlo estimates of lower expectations are consistent. In practice, however, the accuracy of these Monte Carlo approximations depends on the smoothness of the function and the fineness of the grid. As a consequence, it may be computationally prohibitive to produce enough Monte Carlo samples for a good approximation, especially when the parameter is multi-dimensional. To lighten the computational burden, Fetz and Oberguggenberger (2016) and Fetz (2019) propose reusing Monte Carlo samples by weighted resampling. Zhang and Shields (2018) also uses importance resampling to sample imprecise Bayesian posterior distributions. Troffaes (2018) proves consistency of importance sampling estimators of lower expectations using empirical process theory and provides examples using more efficient importance sampling algorithms.

In contrast, we propose to replace grid search by an iterative procedure. The familiar deterministic setting provides some useful background. Suppose the function $M(\theta)$ is known, sufficiently smooth, and convex. Then, Newton's method with updates

$$\theta_{t+1} = \theta_t - \{\ddot{M}(\theta_t)\}^{-1}\dot{M}(\theta_t)$$

converges quadratically to the minimizer $\theta^\star$, where the dots denote differentiation with respect to θ. Alternatively, the gradient descent update

$$\theta_{t+1} = \theta_t - \varepsilon \dot{M}(\theta_t)$$

does not require the second derivative, and converges for a small enough step size $\varepsilon > 0$ and an initial point θ_0 close enough to $\theta^\star$. An important advantage of this approach is that it is less sensitive to the dimension of the optimization problem than a grid search. However, in our present context, the function we seek to optimize, and hence its gradient, are not available in closed-form, so Newton's method cannot be applied directly. Is there a stochastic version?

Robbins and Monro (1951) proposed a stochastic analog of Newton's method and showed that, under certain conditions, its sequence of iterates converges in probability to $\theta^\star$. Roughly speaking, the Robbins–Monro procedure is a gradient descent algorithm but with the known derivative replaced by a (crude) Monte Carlo approximation thereof. In this paper, we consider the use of the Robbins–Monro algorithm and its variants for numerical evaluation of lower and upper expectations.

In Section 2 we discuss existing results that characterize problems suitable for stochastic optimization. Section 3 reviews a number of variations on the classical stochastic optimization algorithm and discusses practical considerations such as the choice of step size (ε_t). Section 4 presents two relevant examples comparing the performance of various stochastic optimization methods with that of grid search. Some concluding remarks are given in Section 5.

2. Algorithm and its Properties

2.1. Robbins–Monro Algorithm

For the analysis in this section, suppose that the optimization problem is sufficiently regular that it can be recast as a root-finding problem. First, let P_θ have a density p_θ with respect to, say, Lebesgue measure, and that $\theta \mapsto p_\theta(x)$ is differentiable for all x except perhaps in a set of Lebesgue measure zero. Second, suppose that M is differentiable and that the order of differentiation and expectation can be interchanged, i.e.,

$$R(\theta) := \dot{M}(\theta) = \int h_\theta(x)\, p_\theta(x)\, dx,$$

where $h_\theta(x) = \dot{g}_\theta(x) + g_\theta(x)\dot{\ell}_\theta(x)$ and $\ell_\theta(x) = \log p_\theta(x)$. Then minimizing M corresponds to finding a root of R. Keep in mind that R is a vector-valued function.

For root-finding, the basic Robbins–Monro algorithm proceeds as follows. For an arbitrary θ_0, define the updates

$$\theta_{t+1} = \theta_t - \varepsilon_{t+1}X_{t+1}, \quad t \geq 0, \tag{1}$$

where (ε_t) is a vanishing, deterministic step size sequence and $(X_t : t \geq 1)$ is a sequence of random vectors such that

$$E(X_{t+1} \mid X_1, \ldots, X_t) = R(\theta_t).$$

For some intuition as to why the Robbins–Monro algorithm works, consider the following heuristics. For simplicity, let $q = 1$. First, try to minimize $M(\theta)$ by iterating gradient descent: $\theta_{t+1} = \theta_t - Y_t$ where $Y_t = \dot{M}(\theta_t) + \delta_t$ represents a noisy estimate of $\dot{M}(\theta_t)$. Then, by substitution, the gradient descent update satisfies $\theta_{t+1} = \theta_t - \dot{M}(\theta_t) - \delta_t$. In general, the noise term δ_t has approximately constant variance, so the iterates θ_t cannot converge to $\theta^\star$. On the other hand, if we apply the Robbins-Monro update the noise term δ_t is multiplied by the vanishing step-size and can safely be ignored for large enough t. Taking the argument one step further, suppose $\dot{M}(\theta_t) < 0$ for $\theta_t < \theta^\star$ and $\dot{M}(\theta_t) > 0$ for $\theta_t > \theta^\star$. For $\theta_t < \theta^\star$ the iterates satisfy

$$\begin{aligned} E(\theta_{t+1} \mid \theta_t) &= \theta_t - \varepsilon_t \dot{M}(\theta_t) - \varepsilon_t E(\delta_t) \\ &= \theta_t - \varepsilon_t \dot{M}(\theta_t) \\ &> \theta_t; \end{aligned}$$

the iterates satisfy the reverse inequality when $\theta_t > \theta^\star$. Therefore, the *expected direction* of the next iteration is towards the minimizer $\theta^\star$.

2.2. Convergence Properties

The claim is that, under certain conditions, the sequence (θ_t) defined by the Robbins–Monro algorithm converges almost surely to the root $\theta^\star$ of R as $t \to \infty$. A particularly elegant proof of this convergence property is based on the following almost supermartingale convergence theorem of Robbins and Siegmund (1971). Given a probability space and an increasing sequence $\{\mathscr{A}_t : t \geq 1\}$ of sub-σ-algebras on that space, they define a sequence of random variables $\{Z_t : t \geq 1\}$ to be an *almost supermartingale* if there exists non-negative random variables ω_t, ξ_t, and ζ_t such that

$$E(Z_{t+1} \mid \mathscr{A}_t) \leq (1+\omega_t)Z_t + \xi_t - \zeta_t, \quad t \geq 1. \tag{2}$$

Then their Theorem 1 states that, if both $\sum_t \omega_t$ and $\sum_t \xi_t$ converge almost surely, then

$$\lim_{t\to\infty} Z_t \text{ exists} \quad \text{and} \quad \sum_{t=1}^{\infty} \zeta_t < \infty \quad \text{almost surely.}$$

To relate this to the Robbins–Monro algorithm, write $Z_t = \|\theta_t - \theta^\star\|^2$, where $\|\cdot\|$ denotes the usual ℓ_2-norm on $\mathbb{R}^q$. Then we set $\mathscr{A}_t$ to be the σ-algebra generated by $\{Z_1, \ldots, Z_t\}$, take conditional expectation, and simplify. The following theorem is a slight generalization of Application 2 in Robbins and Siegmund (1971).

Theorem 1 *Suppose the function R and its root $\theta^\star$ satisfy*

$$(\theta - \theta^\star)^\top R(\theta) \geq 0, \quad \text{for all } \theta \in \mathbb{R}^q, \tag{3}$$

with equality if and only if $\theta = \theta^\star$. Moreover, assume that there exists positive a and b such that

$$E(\|X_{t+1}\|^2 \mid \mathscr{A}_t) \leq a + b\|\theta_t\|^2. \tag{4}$$

If the positive step size sequence $\{\varepsilon_t\}$ satisfies

$$\sum_{t=1}^{\infty} \varepsilon_t = \infty \quad \text{and} \quad \sum_{t=1}^{\infty} \varepsilon_t^2 < \infty, \tag{5}$$

then the Robbins–Monro sequence in (1) satisfies $\theta_t \to \theta^\star$ almost surely as $t \to \infty$.

Proof First, it is easy to see that

$$\begin{aligned} Z_{t+1} &= \|\theta_t - \varepsilon_{t+1}X_{t+1} - \theta^\star\|^2 \\ &= Z_t + \varepsilon_{t+1}^2\|X_{t+1}\|^2 - 2\varepsilon_{t+1}(\theta_t - \theta^\star)^\top X_{t+1}. \end{aligned}$$

Taking conditional expectation gives

$$\begin{aligned} E(Z_{t+1} \mid \mathscr{A}_t) = Z_t &+ \varepsilon_{t+1}^2 E(\|X_{t+1}\|^2 \mid \mathscr{A}_t) \\ &- 2\varepsilon_{t+1}(\theta_t - \theta^\star)^\top R(\theta_t). \end{aligned}$$

By (4) and the inequality $(a+b)^2 \leq 2(a^2+b^2)$,

$$E(\|X_{t+1}\|^2 \mid \mathscr{A}_t) \leq a + 2bZ_t + 2b\|\theta^\star\|^2.$$

Therefore, if we set

$$\begin{aligned} \omega_t &= 2b\varepsilon_{t+1}^2 \\ \xi_t &= 2b\|\theta^\star\|^2\varepsilon_{t+1}^2 \\ \zeta_t &= 2\varepsilon_{t+1}(\theta_t - \theta^\star)^\top R(\theta_t), \end{aligned}$$

then they are all positive and inequality (2) holds. By the second condition in (5), it follows that both ω_t and ξ_t are summable, so it follows from Theorem 1 in Robbins and Siegmund (1971) that Z_t has a limit and $\sum_t \zeta_t$ is finite almost surely. To see that the Z_t limit must be 0, by the first condition in (5), the only way $\sum_t \zeta_t$ could be finite is if $(\theta_t - \theta^\star)^\top R(\theta_t)$ vanishes almost surely, at least on a subsequence. Since Z_t has a limit, it is clear that the only way this latter claim holds is if the Z_t limit is 0. ■

Stronger and more modern results for almost sure convergence of the Robbins–Monro process are available. For a glimpse of what modern proofs entail, consider rewriting update (1) by adding and subtracting $\varepsilon_{t+1}R(\theta_t)$ to get

$$\theta_{t+1} = \theta_t - \varepsilon_{t+1}R(\theta_t) - \varepsilon_{t+1}\{X_{t+1} - R(\theta_t)\}.$$

The latter term is a zero-mean martingale difference sequence and, under mild conditions, would converge to 0 almost surely. Ignoring the negligible martingale term and rewriting the above relationship, we get

$$\varepsilon_{t+1}^{-1}(\theta_{t+1} - \theta_t) \approx R(\theta_t).$$

The left-hand side resembles a derivative of "$t \mapsto \theta_t$," so there is a close connection between the asymptotic properties of the Robbins–Monro process and that of solutions to the ordinary differential equation "$\frac{d}{dt}\theta_t = R(\theta_t)$." See Martin and Ghosh (2008) for an overview, and Kushner and Yin (2003) for a comprehensive account.

Besides almost sure convergence there are results characterizing the random behavior of $(\theta_t - \theta^\star)$ for large t. An early reference is Sacks (1951), which shows that under certain regularity conditions $t^{1/2}(\theta_t - \theta^\star)$ converges in distribution to a normal random variable with variance proportional to ε_0^2 where the step size is given by $\varepsilon_t = \varepsilon_0 t^{-1}$.

3. Variations on Robbins–Monro

3.1. Finite Differences

A key ingredient of the update in (1) is the identification of an unbiased estimator of the gradient $\dot{M}(\theta)$. In general, it may be hard to find such an estimator. Instead, the update can be modified to use a finite-difference approximation. Let e_i denote the unit vector in direction i and define

$$Y_{t,i} = \frac{g(X_i^+) - g(X_i^-)}{2c_t},$$

for $c_t > 0$ fixed or vanishing and where

$$X_i^+ \sim P_{\theta_t + c_t e_i} \quad \text{and} \quad X_i^- \sim P_{\theta_t - c_t e_i}.$$

Setting $Y_t = (Y_{t,1}, \ldots, Y_{t,q})$ the modified update is

$$\theta_{t+1} = \theta_t - \varepsilon_t Y_t. \tag{6}$$

Kiefer and Wolfowitz (2019) originated the update in (6) and proved consistency of θ_t.

To see how the finite-difference approximation may affect the behavior of the algorithm we'll rewrite the update in (6). Define the random Monte Carlo errors

$$\psi_{t,i} = M(\theta_t + c_t e_i) - g(X_i^+) - \{M(\theta_t - c_t e_i) - g(X_i^-)\},$$

which are simply differences between the true function values and noisy estimates obtained from Monte Carlo sampling. Let

$$\dot{M}_i(\theta_t) - \nu_{t,i} = \frac{M(\theta_t + c_t e_i) - M(\theta_t - c_t e_i)}{2c_t},$$

so that $\nu_{t,i}$ denotes the error from approximating the derivative by the finite difference. Then, the update in (6) can be written

$$\theta_{t+1} = \theta_t + \varepsilon_t \dot{M}(\theta_t) - \varepsilon_t(\psi_t/2c_t) - \varepsilon_t \nu_t$$

where $\psi_t = (\psi_{t,1}, \ldots, \psi_{t,q})$ and $\nu_t = (\nu_{t,1}, \ldots, \nu_{t,q})$. As with the Robbins–Monro update, the asymptotic behavior of the Kiefer–Wolfowitz algorithm follows the behavior of the differential equation $d\theta/dt = \dot{M}(\theta_t)$ provided the error terms $\varepsilon_t \psi_t/(2c_t)$ and $\varepsilon_t \nu_t$ vanish. According to Kushner and Yin (2003), $\nu_t = O(c_t)$, so the choice of c_t presents a kind of bias-variance trade-off. A large c_t results in worse approximation of the derivative $\dot{M}(\theta_t)$, but lowers the Monte Carlo noise term $\psi_t/(2c_t)$.

The variance of the Monte Carlo noise term can be minimized by maximizing the covariance of $g(X_i^+)$ and $g(X_i^-)$. One method that may improve the practical performance of the Kiefer–Wolfowitz update is to use correlated Monte-Carlo samples. For example, if P_θ can be sampled by an inverse-CDF transform, then we could sample $U \sim \text{Unif}(0,1)$ and let $X_i^\pm = P^{-1}_{\theta \pm c_t e_i}(U)$.

Sacks (1951) also showed asymptotic normality of the Kiefer–Wolfowitz iterates under regularity conditions. If the step size is taken to be $\varepsilon_t = \varepsilon_0 t^{-1}$, then $t^{1/2} c_t(\theta_t - \theta^\star)$ converges in distribution to a normal random variable with variance proportional to ε_0^2. Sacks (1951) specifies $c_t \to 0$, which implies the Kiefer–Wolfowitz procedure converges more slowly than the Robbins–Monro procedure; in other words, there is a significant cost to approximating derivatives by finite differences.

3.2. Parameter Constraints

When estimating lower expectations it is natural to confine the parameter space to compact subsets; and see the example in Section 4.1. Updates (1) and (6) offer no guarantee the iterates θ_t will remain within any finite neighborhood of the initial point θ_0. A simple fix projects the algorithm to the constraint space. Let $\Theta_C \subset \Theta$ denote a compact constraint space, usually a rectangle in $\mathbb{R}^q$, and let $d : \mathbb{R}^q \times \mathbb{R}^q \mapsto \mathbb{R}^+$ denote a metric on $\mathbb{R}^q$. Let $\mathsf{proj}_d(\theta)$ equal the value of θ' minimizing $d(\theta, \theta')$ over $\theta' \in \Theta_C$. Then, the projected update has the form

$$\theta_{t+1} = \mathsf{proj}_d\{\theta_t - \varepsilon_t Y_t\}.$$

3.3. Averaging Iterates

In practice the analyst applies update (1) or (6) until they meet some pre-specified convergence criteria, at which point they report the most up-to-date iterate θ_t. It turns out that it may be advantageous instead to report the average of the iterates $\bar{\theta} = t^{-1}\sum_{s=1}^t \theta_s$ as the final estimate. In practice, the updates generally converge more quickly with larger step sizes ε_t. But, large step sizes also increase iterate variability, and averaging the iterates naturally reduces this variability. Furthermore, the practical benefit of iterate averaging holds up in theory. When $\varepsilon_t = O(t^{-1})$ it can be shown the mean squared error $E\|\theta_t - \theta^\star\|_2^2$ behaves like t^{-1}, but the corresponding mean squared error for the averaged iterates $\bar{\theta}_t$ vanishes like t^{-1} even when $\varepsilon_t = O(t^{-1/2})$; see Polyak and Juditsky (1992) and Kushner and Yin (2003).

3.4. Averaging or Bounding Monte Carlo Samples

In addition to averaging iterates, it may be helpful to average over $M \geq 1$ Monte Carlo samples at each iteration. One reason to take $M > 1$ samples is when the function $M(\theta)$ likely may be zero, for example, when $M(\theta)$ is the probability of a rare event. In that case, the estimate of the gradient could vanish, which may trigger the convergence criteria or simply cause the iterates to get stuck at a constant value. For the Robbins–Monro update (1), replace Y_t by $\overline{Y}_t = M^{-1}\sum_{j=1}^M Y_t^j$ where each Y_t^j is an independent sample with mean $\dot{M}(\theta_t)$. For the Kiefer–Wolfowitz update (6), the finite difference approximation can be computed by the Monte Carlo average $\overline{Y}_t = M^{-1}\sum_{j=1}^M Y_t^j$, where $Y_t^j = (Y_{t,1}^j, \ldots, Y_{t,q}^j)$,

$$Y_{t,i}^j = \frac{g(X_i^{+,j}) - g(X_i^{-,j})}{2c_t},$$

and $X_i^{\pm,j} \sim P_{\theta_t \pm c_t e_i}$, for $j = 1, \ldots, M$. The obvious drawback to observation averaging is that it requires many more Monte Carlo samples.

In other cases, the function $M(\theta)$ may be highly sensitive to θ so that there is a chance to generate an extreme update. Kushner and Yin (2003) suggest upper-bounding the absolute value of Monte Carlo samples, or, equivalently, upper-bounding by a constant the absolute change in subsequent iterate values.

3.5. Second-Order Methods

The convergence rate of iterative methods for optimizing deterministic functions usually improves when those methods employ second-order information, like a Hessian matrix. Perhaps surprisingly, stochastic second-order methods can achieve faster rates than their first-order counterparts only by a constant multiple (Agarwal et al., 2012). The stochastic or Robbins–Monro analog of the classical Newton method updates via

$$\theta_{t+1} = \theta_t - \varepsilon_t Z_t Y_t, \tag{7}$$

where Z_t is an unbiased estimator of $[\ddot{M}(\theta_t)]^{-1}$. As with the first-order Robbins–Monro algorithm, it may be challenging to find an unbiased estimator of the inverse Hessian. Similar to the Kiefer–Wolfowitz approach, the Hessian can be estimated by finite differences, but there is a more efficient approach. In deterministic function optimization the BFGS methods (e.g., Fletcher, 1987, Chap. 3.4) are computationally efficient alternatives to computing matrix inverses. These methods iteratively update Z_t by

$$\begin{aligned}
\delta_t &= \theta_{t+1} - \theta_t \\
\gamma_t &= Y_{t+1} - Y_t \\
Z_{t+1} &= \left(I - \frac{\gamma_t \delta_t^\top}{\delta_t^\top \gamma_t}\right)^\top Z_t \left(I - \frac{\gamma_t \delta_t^\top}{\delta_t^\top \gamma_t}\right) + \frac{\delta_t \delta_t^\top}{\delta_t^\top \gamma_t}.
\end{aligned}$$

Byrd et al. (2016) developed a stochastic version of the BFGS algorithm and demonstrated its performance in machine learning problems.

3.6. ADADELTA and Choice of Step Size

All three updates discussed above rely on a user-specified step size or learning rate ε_t. When an average of iterates will be reported often $\varepsilon_t = \varepsilon_0 t^{-\tau}$ where $\tau \in (1/2, 3/4)$. In practice, the leading constant ε_0 can have a surprisingly strong impact on the speed of convergence of the iterates. When ε_0 is too small the sequence $\{\theta_t\}$ may change very little and practically fail to converge because the user's maximum number of iterations is exceeded. On the other hand, when ε_0 is very large there is excessive variation early in the sequence $\{\theta_t\}$, and this may cause the sequence of iterate averages to converge slowly.

Finding the optimal value ε_0 is a challenge. One strategy is to run a small number of iterations, say 10, for several values of ε_0 and choose the smallest value of ε_0 such that a measure of variation (e.g., range or variance) of the corresponding 10 iterates is sufficiently large.

Alternative updating formulas, such as the ADADELTA method in Zeiler (2012), do not require a user-specified step size at all. Instead, ADADELTA iteratively and adaptively computes a step size from the change in successive iterates and a running average of the gradient. Select an averaging parameter close to one, e.g., $\rho = 0.995$. Define the root mean square function $RMS(x) = (x^2 + \eta)^{1/2}$ for some small stabilizing constant, e.g., $\eta = 0.995$. The ADADELTA algorithm makes an initial update based only on the gradient, as in (1), and for $t > 1$ updates according to:

1. Compute gradient estimate Y_t
2. Accumulate gradient $S_t = \rho S_{t-1} + (1-\rho) Y_t^2$
3. Compute update $\delta\theta_t = -\frac{RMS(D_{t-1})}{RMS(S_t)} Y_t$
4. Accumulate update $D_t = \rho D_{t-1} + (1-\rho)(\delta\theta_t)^2$
5. Apply update $\theta_{t+1} = \theta_t + \delta\theta_t$

4. Examples

4.1. A Gaussian Probability

The first example is taken from Fetz (2019). Consider estimating the lower expectation $\min_\theta M(\theta)$ where $M(\theta) = E[1\{X \notin (-2,2)\}]$ and where $X \sim N(\theta, \sigma = 2)$. The minimum of $M(\theta)$ occurs at $\theta = 0$. Stochastic optimization techniques experience the greatest gains over grid search Monte Carlo estimation when the parameter space is multidimensional. Nevertheless, this simple example illustrates the practical differences between the variations of stochastic optimization described in Section 3.

First, consider the Robbins–Monro update in (1) run for 1000 iterations. Figure 1 displays the iterates and average iterates for two learning rates: $t^{-1/2}$ and $5t^{-1/2}$. There are two important features of the plot. The larger learning rate produces much more variation in the iterates, which helps the sequence to quickly find the minimum and then randomly vary around that minimum. The bias in the iterates coming from the initial point θ_0 quickly dissipates and the average of iterates quickly settles down near $\theta = 0$. In contrast, the sequence of iterates with learning rate $t^{-1/2}$ move very slowly towards $\theta = 0$; so slowly, in fact, that there is no benefit to averaging the iterates since the average remains biased towards the initial point $\theta_0 = 6$.

Next, we consider whether averaging over $M > 1$ Monte Carlo samples may improve the performance of the Robbins–Monro update. For the basic method, consider running the Robbins–Monro update in (1) and with $\varepsilon_t = 20t^{-1/2}$ for 1000 iterations, and in each step using only $M = 1$ Monte Carlo sample Y_t to estimate the gradient. For comparison, run the Robbins–Monro update only 100 but use $M = 10$ Monte Carlo samples in each iteration, and estimate the gradient by $\overline{Y}_t = \frac{1}{10}\sum_{j=1}^{10} Y_t^j$ for Monte Carlo samples $Y_t^1, \ldots, Y_t^{10}$. The idea is to compare the performance of the two variations of (1) for the same number of Monte Carlo samples. Each method was run 1000 times, and 100 of those corresponding paths of iterate averages are displayed in Figures 2–3. The figures do not suggest averaging Monte Carlo samples at each iteration speeds up

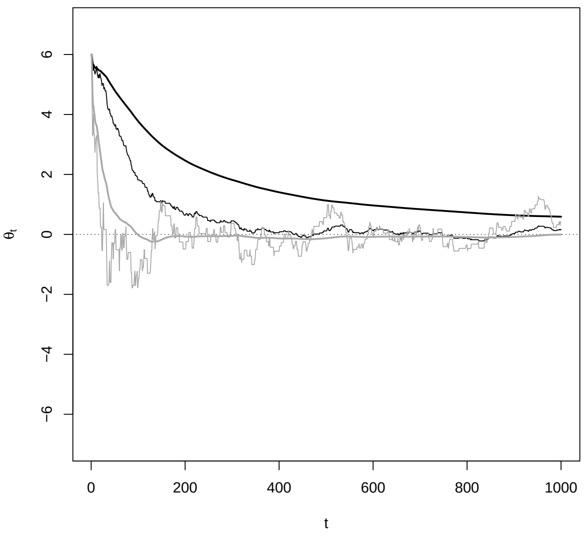

Figure 1: Sequences of iterates θ_t (fine lines) and their averages $\overline{\theta}_t$ (bold lines) for the Robbins–Monro update (1). Black lines indicate learning rate $\varepsilon_t = t^{-1/2}$ while gray lines correspond to $\varepsilon_t = 5t^{-1/2}$.

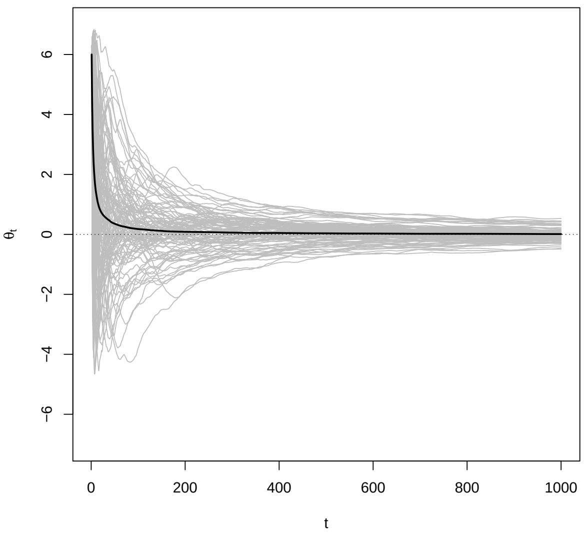

Figure 2: Averages of iterates $\bar{\theta}_t$ (gray lines) and their overall average (black line) over 1000 repetitions of the Robbins–Monro update (1) with no Monte Carlo averaging of samples.

convergence. The last iterate average $\bar{\theta}_{100}$ for the averaging method was about 0.34 on average and with standard deviation 0.44 over 1000 repetitions. In contrast, the last iterate average $\bar{\theta}_{1000}$ for the method that drew only one Monte Carlo sample per iteration has average value 0.01 with standard deviation 0.19 over 1000 repetitions. For reference, a simple grid search performing 10 Monte Carlo samples at each of 100 grid points performed worse than both methods, with an average solution of -0.44 and a standard deviation of 0.82 over 1000 repetitions. The takeaway is that averaging multiple Monte Carlo samples at each iteration does not speed up convergence.

Besides Monte Carlo averaging, it is possible incorporating second derivative information into the Robbins–Monro update might improve its practical performance even if it provides no substantial benefit according to convergence theory. However, in this example the Newton style of update in (7) is very inefficient. The trouble is that the update is highly sensitive to the second derivative, at least for the first several hundred iterations, and this sensitivity causes the algorithm to behave erratically. One way to dampen the effect of high variation in the estimate of the second derivative is, of course, to use $M \gg 1$ Monte Carlo samples per iteration. We used $M = 50$ Monte Carlo samples to produce Figure 4, yet the iterate paths are still prone to excessive variation early in the sequence. Figure 5 illustrates a disadvantage of iterate averaging is that iterate paths are sensitive to early steps and slow to converge. An alternative is to compute a moving average of the last k iterates, rather than all the previous iterates. Over 100 repetitions the second-order updating method produced an average solution within ± 0.01 with a standard deviation of 0.08, better than the first order methods tried above. But, this is based on 50 times the Monte Carlo samples, so does not represent an improvement over the Robbins–Monro method.

So far, our investigation suggests the original Robbins–Monro update (1) without Monte Carlo averaging of samples, but with averaging of iterates performs best provided a good choice of learning rate can be made. Since it is not straightforward to choose a good learning rate in practice, we last consider the ADADELTA update of Section 3.6 that makes an “automatic" choice of learning rate. Figure 6 displays iterate and average iterate trajectories for the ADADELTA update. After 10000 iterations the average solution using the average of iterates is 0.19 with a standard deviation of 0.50; the median solution is 0.07. ADADELTA practically converges much more slowly than the Robbins–Monro update with a good choice of learning rate, but it may be an attractive option if the cost of using many iterations is not too high.

4.2. Inferential Models in Logistic Regression

Suppose we have binary data $Y = (Y_1, \ldots, Y_n)^\top$, independent, where Y_i is a Bernoulli random variable with parame-

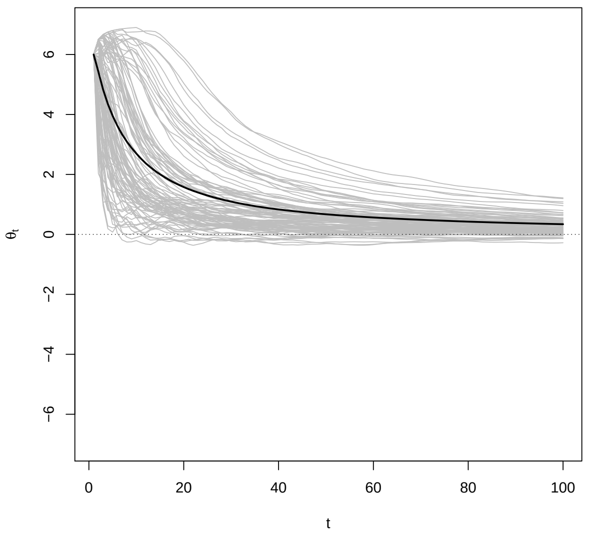

Figure 3: Averages of iterates $\bar{\theta}_t$ (gray lines) and their overall average (black line) over 1000 repetitions of the Robbins–Monro update (1) with averaging of $M = 10$ Monte Carlo samples per iteration.

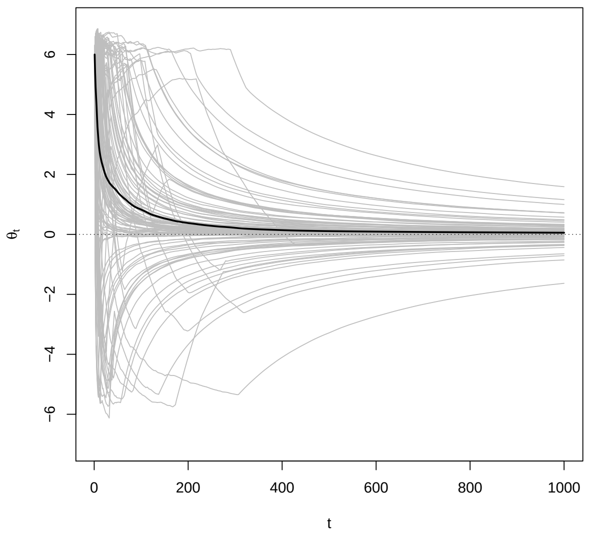

Figure 5: Averages of iterates $\bar{\theta}_t$ (gray lines) and their overall average (black line) over 1000 repetitions of the second-order update (7) with averaging of $M = 50$ Monte Carlo samples per iteration.

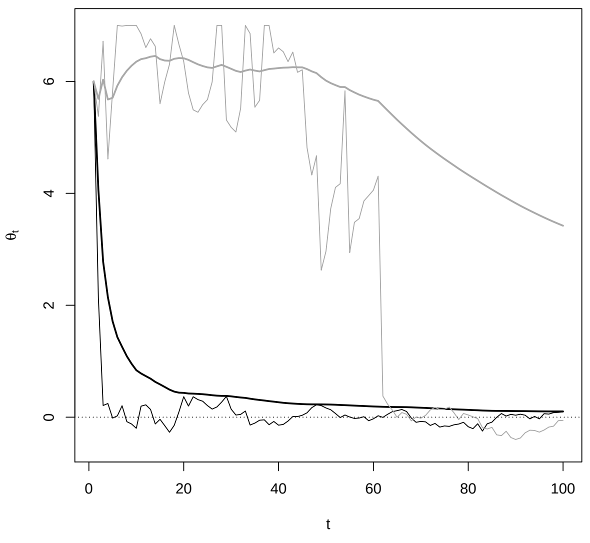

Figure 4: Two sequences of iterates θ_t (fine lines) and their averages $\bar{\theta}_t$ (bold lines) for the second-order update (7).

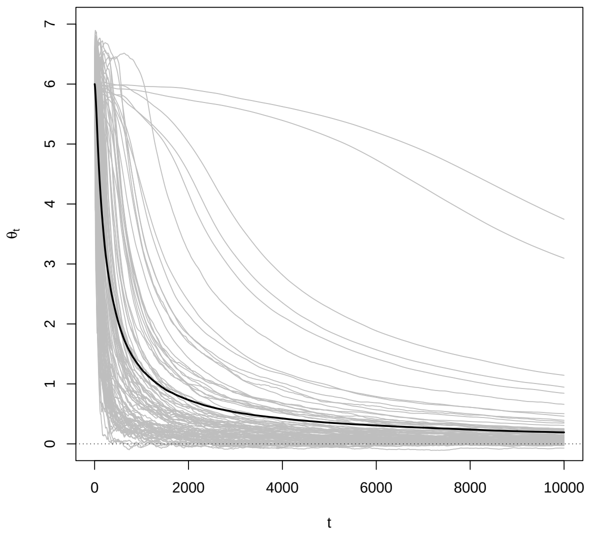

Figure 6: Averages of iterates $\bar{\theta}_t$ (gray lines) and their overall average (black line) over 1000 repetitions of ADADELTA.

ter $p_i = F(x_i^\top \theta)$, where x_i is a fixed/deterministic q-vector of explanatory variables, θ is a q-vector of coefficients, and F is the logistic distribution function, $F(z) = (1+e^{-z})^{-1}$. One reason the logistic distribution is preferred for binary regression is its connection to the *odds* of $Y = 1$, which satisfy

$$\log\left(\frac{p_i}{1-p_i}\right) = x_i^\top \theta.$$

Consequently, logistic regression parameters have a convenient interpretation similar to linear regression slope parameters: for a unit increase in x_{ij} the logarithm of the odds increases by θ_j. Relevant questions like "does predictor x_j affect/increase/decrease the odds?" can be investigated by evaluating suitable lower and upper probabilities of the respective assertions $A = \{\theta : \theta_j = 0\}$, $A = \{\theta : \theta_j > 0\}$, and $A = \{\theta : \theta_j < 0\}$.

As discussed in Section 1, if the goal is inference on θ based on observed data $Y = y_{\text{obs}}$, we can construct a generalized inferential model as follows. Define

$$T(y,\theta) = -\log\{L_y(\theta)/L_y(\hat{\theta}_y)\},$$

where L_y and $\hat{\theta}_y$ denote the likelihood function and the maximum likelihood estimator, respectively, based on a data set y, which depends implicitly on F and the explanatory variables $x_1,\ldots,x_n$. And, let P_θ denote the joint distribution of Y, which depends explicitly on the parameter θ and implicitly on the (deterministic) explanatory variables $x_1,\ldots,x_n$. Then the plausibility contour is given by

$$\pi(\theta; y_{\text{obs}}) = P_\theta\{T(Y,\theta) > T(y_{\text{obs}},\theta)\},$$

and to evaluate the upper probability of an assertion A about θ requires solving the optimization problem

$$\overline{\Pi}(A; y_{\text{obs}}) = \sup_{\theta \in A} \pi(\theta; y_{\text{obs}}).$$

One reason to compute this upper probability is to use it for evaluating a hypothesis test. In the inferential modeling framework a level-α test of $H_0 : \theta \in \Theta_0$ rejects if the inferential model upper probability $\overline{\Pi}(\Theta_0; y_{\text{obs}}) < \alpha$, which is similar to a p-value-based rejection rule. The general theory ensures, among other things, that the aforementioned test controls the frequentist Type I error at level α.

Next, we describe two simulation experiments to evaluate the performance of stochastic optimization in computing inferential model upper probabilities for logistic regression. Our simulations mimic experiments with fixed, complete designs. In our first simulation we define $q = 3$ predictors x_j, $j = 1,2,3$, each taking values in $\{0,1/3,2/3,1\}$ and we form the 64×3 fixed design matrix with one row for every combination of predictor values. Given predictor vector $x_i = (x_{i,j}, x_{i,2}, x_{i,3})^\top$, response Y_i is sampled from a Bernoulli distribution with success probability $p_i = F(x_i^\top \theta^\star)$ for $\theta^\star = (-2,-1,2)^\top$ and where $F(z)$ denotes the logistic distribution function. Let $A := \{\theta = (\theta_1,\theta_2,\theta_3)^\top : \theta_2 > 0\}$. To quantify our uncertainty about the answer to the question "does predictor x_2 increase the odds of success?," we compute $\overline{\Pi}(A; y_{\text{obs}})$.

For the second simulation we again consider a logistic regression model but now with $q = 4$ predictors, each taking values in the set $\{0,1/2,1\}$, so that the complete, fixed design matrix has $3^4 = 81$ rows. The true coefficients are $\theta^\star = (-2,-1,2,1)^\top$, and our assertion of interest is $A := \{\theta : \theta_2 > 0\}$.

For the stochastic optimization approach we utilize the Kiefer–Wolfowitz update in (6), which requires approximating the gradient of $\pi(\theta_t; y_{\text{obs}})$ by finite differences. A Monte Carlo approximation of $\pi(\theta_t; y_{\text{obs}})$ is given by

$$\tilde{\pi}_N(\theta_t) = N^{-1} \sum_{i=1}^{N} 1\{T(Y_n^{(j)}, \theta_t) > T(y_{\text{obs}}, \theta_t)\},$$

where $Y_n^{(j)} = (Y_{1,n}^{(j)}, \ldots, Y_{n,n}^{(j)})^\top$ and

$$Y_{i,n}^{(j)} \overset{\text{ind}}{\sim} \text{Ber}\{F(x_i^\top \theta_t)\}, \quad i = 1,\ldots,n.$$

Then, the i^{th} component of the gradient of $\pi(\theta_t; y_{\text{obs}})$ is approximated by the following difference of Monte Carlo approximations,

$$\frac{\tilde{\pi}_N(\theta_t + c_t e_i) - \tilde{\pi}_N(\theta_t - c_t e_i)}{2c_t},$$

where e_i denotes the unit vector in direction i. We run two different variations of the Kiefer–Wolfowitz algorithm, both with a step-size of $30t^{-0.5}$ and a finite-difference radius of $c_t = t^{-0.5}$, but with different numbers of samples N used in the finite difference approximation and different numbers of iterations M. For the first run we set $N = 16$ and $M = 1250$, and for the second run we set $N = 40$ and $M = 500$. We do not use any stopping rule for early termination of the updates; they run until reaching M iterations. For each Kiefer–Wolfowitz algorithm we compare the approximations of $\overline{\Pi}(A; y_{\text{obs}})$ using averaged iterates versus the final iterate. Stochastic optimization theory implies we should average iterates since we use a large step size satisfying $\varepsilon_t = O(t^{-0.5})$. For comparison to stochastic optimization, we also approximate $\overline{\Pi}(A; y_{\text{obs}})$ by a grid search method, which we describe below.

We run each simulation on a total of 200 randomly generated response vectors. For simulation 1 the total number of Monte Carlo samples used is $N \times M \times n \times 6 = 7{,}680{,}000$, and for simulation 2 the total is $N \times M \times n \times 8 = 12{,}960{,}000$. Note that $2 \times N \times n$ Monte Carlo samples are needed to approximate each element of the gradient vector. We use nearly the same number of Monte Carlo samples for grid search. In the first simulation we fix a grid with $15^3 = 3375$ points on the set $\theta \in [-3,3] \times [0,3] \times [-3,3]$ and approximate $\overline{\Pi}(A; y_{\text{obs}})$ using 36 Monte Carlo samples at each point for a total of $15^3 \times 36 \times n = 7{,}760{,}000$ samples. For the second simulation we reduce the density on the grid to 10^4 points in the set $\theta \in [-3,3] \times [0,3] \times [-3,3] \times [-3,3]$ and reduce the number of Monte Carlo samples used to approximate $\overline{\Pi}(A; y_{\text{obs}})$ at each point to 16 for a total of $10^4 \times 16 \times n = 12{,}960{,}000$ Monte Carlo samples.

In both simulations, we compare the various estimates of the upper probability $\overline{\Pi}(A; y_{\text{obs}})$ to the plausibility contour $\pi(\tilde{\theta}; y_{\text{obs}})$ evaluated at $\tilde{\theta}$, the constrained maximum likelihood estimator (MLE) under the constraint $\theta_2 \geq 0$. The reason for this is two-fold. First, we know that $\tilde{\theta}$ maximizes $\theta \mapsto T(y_{\text{obs}}, \theta)$ over A so there is good reason to think the maximizer of the θ-dependent probability that defines $\pi(\theta; y_{\text{obs}})$ would also be maximized, at least approximately, at $\tilde{\theta}$. Second, even if $\pi(\tilde{\theta}; y_{\text{obs}})$ is not an especially accurate approximation of $\overline{\Pi}(A; y_{\text{obs}})$, we do know that $\overline{\Pi}(A; y_{\text{obs}}) \geq \pi(\tilde{\theta}; y_{\text{obs}})$. Therefore, between two estimates of $\overline{\Pi}(A; y_{\text{obs}})$, say, Est_1 and Est_2, if

$$\pi(\tilde{\theta}; y_{\text{obs}}) \approx \text{Est}_1 \gg \text{Est}_2,$$

then Est_1 must be better than Est_2.

Table 1 displays the average of estimates of $\overline{\Pi}(A; y_{\text{obs}})$ over 200 simulated response vectors under Settings 1 and 2 with 3 and 4 predictor variables, respectively. The main takeaway is that grid search loses approximation accuracy as the dimension increases while the stochastic optimization approaches do not appear to lose accuracy. Grid search declines in efficiency compared to the constrained MLE when increasing from 3 to 4 predictors due to taking a coarser grid and using fewer Monte Carlo samples to approximate $\pi(\theta; y_{\text{obs}})$ at each grid point. We see little difference in performance between different variations of Kiefer–Wolfowitz updating. The approaches using averaged iterates produce better results than without averaging, which may suggest substantial variability in iterates remained after 500 iterations in the second simulation under the second run of Kiefer–Wolfowitz updating.

Stochastic optimization's performance was as good or better than grid search, arguably even better than the results indicate. First, in these simulations, the grid for grid search was chosen favorably as a subset that contains the constrained MLE with high probability. In practice, the choice of grid is challenging, and may not be guaranteed to contain the maximizer. Second, in order to compare the algorithms, we used no stopping criteria for stochastic optimization. As the example in Section 4.1 illustrates, the number of iterations needed for practical convergence of stochastic optimization can vary considerably. So, the number of Monte Carlo samples needed to produce the stochastic optimization results in Table 1 is generally less than the number of samples actually used.

	Estimate of $\overline{\Pi}(A; y_{\text{obs}})$	
Method	Sim 1	Sim 2
Constrained MLE	0.55 (0.35)	0.52 (0.35)
Grid Search	0.49 (0.34)	0.41 (0.33)
K–W $M = 16$, averaging	0.48 (0.38)	0.51 (0.36)
no averaging	0.48 (0.38)	0.49 (0.34)
K–W $M = 40$, averaging	0.50 (0.37)	0.51 (0.36)
no averaging	0.50 (0.37)	0.22 (0.14)

Table 1: Average estimates (and standard deviations) of $\overline{\Pi}(A; y_{\text{obs}})$ for two variations of Kiefer–Wolfowitz updating, both with and without averaging iterates. A grid search approximation of $\overline{\Pi}(A; y_{\text{obs}})$ and a Monte Carlo approximation of $\pi(\tilde{\theta}; y_{\text{obs}})$ at the constrained MLE are displayed for comparison.

5. Conclusion

Several recent works have studied Monte Carlo approximation of upper and lower expectations by grid search methods. In theory these methods are guaranteed to provide accurate approximations, but in practice accuracy is limited by specification of a subset of the parameter space in which to conduct grid search. It is possible the analyst may specify a subset not containing the true optimal $\theta^\star$, in which case grid search is inconsistent, regardless of the fineness of the grid. However, even if the analyst makes a good choice of subset, accuracy is limited by the fineness of the grid for a fixed computational cost. This "curse of dimensionality" causes practical accuracy to decline sharply as the number of parameters increases.

Stochastic optimization offers provably accurate approximation of upper and lower expectations along with better practical performance than grid search in multi-dimensional problems. It can be challenging to decide which variation of stochastic optimization to use and how to choose values of tuning parameters, but these choices are no more challenging for the analyst than subset selection for grid search. In our examples, the Robbins–Monro and Kiefer–Wolfowitz procedures with iterate averaging and relatively large step-sizes worked best.

In light of recent observations in Martin (2021), much of what is considered "frequentist statistical inference" can be formulated using notions of imprecise probability, in particular, possibility theory. This means that problems like the one presented in Section 4.2, where lower and upper probabilities are evaluated based on optimization, are of fundamental importance for statisticians and data scientists. Therefore, it is important to be able to solve these problems as accurately and efficiently as possible. The stochastic optimization tools presented here seem quite promising, but more work is needed to develop (a) general rules for tuning the algorithms' parameters and (b) software that is easy to use.

References

A. Agarwal, P. L. Bartlett, P. Ravikumar, and M. J. Wainwright. Information-theoretic lower bounds on the oracle complexity of stochastic convex optimization. *IEEE Trans Inf Theory*, 58(5):3235–3249, 2012.

R. H. Byrd, S. L. Hansen, J. Nocedal, and Y. Singer. A stochastic quasi-Newton method for large-scale optimization. *SIAM J Optim*, 26(2):1008–1031, 2016.

A. Decadt, G. de Cooman, and J. De Bock. Monte Carlo estimation for imprecise probabilities: Basic properties. In *Proceedings of the Eleventh International Symposium on Imprecise Probabilities: Theories and Applications*, volume 103 of *Proc Mach Learn Res*, pages 134–144, Ghent, Belgium, 2019.

T. Fetz. Improving convergence of iterative importance sampling for computing upper and lower expectations. In *Proceedings of the Eleventh International Symposium on Imprecise Probabilities: Theories and Applications*, volume 103 of *Proc Mach Learn Res*, pages 185–193, Ghent, Belgium, 2019.

T. Fetz and M. Oberguggenberger. Imprecise random variables, random sets, and Monte Carlo simulation. *Int J Approx Reason*, 78:252–264, 2016.

R. Fletcher. *Practical Methods of Optimization, 2nd ed.* New York: John Wiley & Sons, 1987.

J. Kiefer and J. Wolfowitz. Stochastic estimation of the maximum of a regression function. *Ann Math Statist*, 23 (3):462–466, 2019.

H. J. Kushner and G. G. Yin. *Stochastic Approximation and Recursive Algorithms and Applications*. New York: Springer, 2003.

C. Liu and R. Martin. *Inferential Models: Reasoning with Uncertainty*. Monographs in Statistics and Applied Probability Series, Chapman & Hall/CRC Press, 2015.

C. Liu and R. Martin. Inferential models and possibility measures. 2020. https://researchers.one/articles/20.08.00004.

R. Martin. Plausibility functions and exact frequentist inference. *J Amer Stat Assoc*, 110:1552–1561, 2015.

R. Martin. On an inferential model construction using generalized associations. *J Stat Plan Infer*, 195:105–115, 2018.

R. Martin. An imprecise-probabilistic characterization of frequentist statistical inference. 2021. https://researchers.one/articles/21.01.00002.

R. Martin and J. K. Ghosh. Stochastic approximation and Newton's estimate of a mixing distribution. *Stat Sci*, 23 (3):365–382, 2008.

B. T. Polyak and A. B. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM J Control Optim*, 30: 838–855, 1992.

H. Robbins and S. Monro. A stochastic approximation method. *Ann Math Statist*, 22(3):400–407, 1951.

H. Robbins and D. Siegmund. A convergence theorem for non negative almost supermartingales and some applications. In J. S. Rustagi, editor, *Optimizing Methods in Statistics*, pages 233–257. Academic Press, 1971.

J. Sacks. Asymptotic distribution of stochastic approximation procedures. *Ann Math Statist*, 29(2):373–405, 1951.

M. Troffaes. Imprecise Monte Carlo simulation and iterative importance sampling for the estimation of lower previsions. *Int J Approx Reason*, 101:31–48, 2018.

M. D. Zeiler. ADADELTA: An adaptive learning rate method. 2012. https://arxiv.org/abs/1212.5701.

H. Zhang and M. D. Shields. On the quantification and efficient propagation of imprecise probabilities resulting from small datasets. *Mech Syst Signal Process*, 98:465–483, 2018.