Stochastic Search with an Observable State Variable

Lauren A. Hannah* Warren B. Powell† David M. Blei‡

June 23, 2010

Abstract

In this paper we study convex stochastic search problems where a noisy objective function value is observed after a decision is made. There are many stochastic search problems whose behavior depends on an exogenous state variable which affects the shape of the objective function. Currently, there is no general purpose algorithm to solve this class of problems. We use nonparametric density estimation to take observations from the joint state-outcome distribution and use them to infer the optimal decision for a given query state. We propose two solution methods that depend on the problem characteristics: function-based and gradient-based optimization. We examine two weighting schemes, kernel-based weights and Dirichlet process-based weights, for use with the solution methods. The weights and solution methods are tested on a synthetic multi-product newsvendor problem and the hour-ahead wind commitment problem. Our results show that in some cases Dirichlet process weights offer substantial benefits over kernel based weights and more generally that nonparametric estimation methods provide good solutions to otherwise intractable problems.

1 Introduction

Stochastic search is a class of stochastic optimization problem problem where we have to find a deterministic parameter to minimize the expectation of a function of uncertain quantities, where the expectation is hard to compute, requiring instead the use of Monte Carlo samples or direct observation of exogenous quantities. The problem is typically written

$$\min_{x \in \mathcal{X}} \mathbb{E}\left[F(x, Z)\right], \tag{1}$$

where $x \in \mathbb{R}^d$ is the decision, $\mathcal{X}$ is a decision set, $Z : \Omega \to \Psi$ is a random outcome and $F : \mathbb{R}^d \times \Psi \to \mathbb{R}$ is an objective function. A classic example is the newsvendor problem where we have to stock a quantity of product to serve an uncertain demand with an unknown distribution. Each iteration, we can only observe how much we sold, after which we make adjustments. A rich theory has evolved to address problems of this type (see [57]).

In this paper, we introduce an important variation of the stochastic search problem. Assume that we are first allowed to observe a state variable S (such as whether it is raining or sunny) which changes our belief about the distribution of the random vector Z. After observing S, we then choose x, and only then do we observe Z, or we may only observe $F(x, S, Z)$ (or its derivative). Each iteration starts with a new state, after which we choose an action and then observe the results.

*Department of Operations Research and Financial Engineering, Princeton University, Princeton, NJ, 08544, lhannah@princeton.edu

†Department of Operations Research and Financial Engineering, Princeton University, Princeton, NJ, 08544

‡Department of Computer Science, Princeton University, Princeton, NJ 08544

Since information from the current state, decision and observation is used to update beliefs for future decisions, we have two challenges: 1) assembling information from previous state-decision-outcome pairs into something that can be used to make a decision for the current state and 2) finding the optimal decision given a state.

If the state space is small (say, rainy or sunny) we can use classical methods from stochastic search by simply conditioning on the state when we do our updates. But it is often the case that S is a vector, frequently with continuous elements. The hour ahead wind commitment problem is an example: a wind farm manager must pledge how much energy she will provide to a utility company an hour in the future. If too much energy is pledged, the difference must be bought; if too little is pledged, the difference is lost. The objective function depends on the future wind and market price, both unknown. The last 24 hours of wind and market prices, time of day and time of year all contain information about the objective function. This problem cannot be solved using standard techniques from stochastic search and stochastic optimization.

To combat the first problem, sharing information across observations, we propose using nonparametric density estimation for the joint state and outcome distribution to group observations from "similar" states with weights. To combat the second problem, making a decision given an observed state, we use the weighted observations to construct convex, deterministic approximations of the conditional expectation of the objective function. Care is taken to ensure that the resulting optimization problems are computationally feasible.

With this high level summary in mind, we turn to a more formal description of the problem setting. When we include the state variable, the stochastic search problem of Equation (1) becomes

$$\min_{x\in\mathcal{X}} \mathbb{E}\left[F(x,s,Z)|S=s\right]. \tag{2}$$

Note that the function F itself may change with the state. Conventional stochastic search techniques require us to sample from the conditional distribution $p(Z|S=s)$, treating each state observation independently [57]. We use nonparametric density estimation for the joint distribution of (S,Z) to weight the states because similar values of S usually affect Z and F in a similar way.

We propose a new model-free method to solve the stochastic optimization problem with an observable state variable. Our problem is motivated by online applications where we are given a state, and then after making a decision, we are given the realization of Z which depends on the state. For this reason, we index estimates and random variables, such as S_n, with a subscript that indicates at which iteration the value can be used. We use a nonparametric density estimate of (S,Z) to weight previous observations $(S_i,Z_i)_{i=0}^{n-1}$. Given an observed state, we generate an estimate of $\mathbb{E}[F(x,s,Z)]$, called the approximate function $\bar{F}_n(x|s)$, based on the weighted previous observations. For this paper, we are concerned with two classes of stochastic search solution methods for convex problems:

- **Function-Based Optimization:** given a state $S_n=s$ and an outcome $Z(\omega_{n+1})$ with $\omega_{n+1}\in\Omega$, the entire response function $F(x,s,Z(\omega_{n+1}))$ is known.
- **Gradient-Based Optimization:** given a state $S_n=s$, a decision x_N and an outcome ω_{n+1}, we only observe the stochastic gradient $\hat{\beta}(x_n,s,Z(\omega_{n+1}))=\nabla_x F(x_n,s,Z(\omega_{n+1}))$.

The wind commitment problem can be solved by function-based optimization; once the wind speed at time $n+1$ is known, the value for all possible commitment levels at time n is also known. In many problems the entire objective function is too expensive too compute or cannot be explicitly computed; however, it is often possible to observe or estimate derivatives around a decision value. For example, many resource allocation problems involve solving a linear program; the dual variables provide the gradients. For more complicated problems, function-based optimization may be

unstable or produce functions too complicated for use in a solver. Gradient-based optimization eliminates these problems by restricting the form of the approximate function $\bar{F}_n(x|s)$ to piecewise linear, separable and convex. In both methods, however, we use weights derived from a joint distribution of (S, Z) with previous observations to form an approximate function, $\bar{F}_n(x|s)$.

A large class of function-based methods currently exist for problems without a state variable [46, 55, 54]. We craft a function-based optimization algorithm by extending the existing methods to weighted observations. We give conditions for almost sure convergence of this algorithm to a global optimum.

Our approach to gradient-based optimization is less straightforward. Stochastic approximation [45, 30] is the most popular gradient search method for problems without a state variable, but for reasons explained in Section 4, it cannot be extended to problems with a state variable. Instead, we try to construct a function that has the same behavior as the original objective function around the optimum. Like [41], we use a separable, convex, piecewise linear approximation to do this, except that our previous observations are weighted according to the current state so that they produce appropriate slopes for the piecewise linear approximation. This algorithm is designed to be more robust to noise and high-dimensional decision spaces than function-based optimization. We show that the resulting algorithm converges to an arbitrarily small neighborhood around the optimum with probability one when the true objective function is itself separable.

Both methods rely heavily on weighting functions. We give two methods to generate weights: kernels and Dirichlet process mixture models. Kernels are easy to implement and often give good results. They can develop problems, however, when the state variable is moderate to high dimensional by giving all but a few observations weights that are effectively zero. This can lead to unstable results. As an alternative in these situations, we propose using weights generated by Dirichlet process mixture models. Dirichlet process mixture models are Bayesian nonparametric models that produce a distribution over data partitions. In effect, they cluster data in a Bayesian manner. We derive weights from this model by placing equal weight on all previous observations that are in the same cluster as the current observation. Then we approximate the average of these weights by taking a Monte Carlo sampling of clusterings. This method requires more work, but it is far more stable than kernel methods. We give conditions for when kernel and Dirichlet process weights satisfy the convergence criteria for both optimization algorithms.

We test our methods on two problems, a two-product newsvendor problem and the hour ahead wind commitment problem. In the two-product newsvendor problem, we use synthetic data and compare both optimization methods under each weighting function. We demonstrate that gradient-based optimization is relatively more robust than function-based optimization when a large amount of noise is present. In the hour ahead wind commitment problem, we use synthetic price data and wind data from the North American Land Data Assimilation System. Due to the computational difficulties of computing weights and testing solutions every iteration, we only compare function-based optimization under the different weighting schemes. Dirichlet process weights produce better results for this problem.

We contribute novel algorithms to include state variables in function-based optimization and gradient-based optimization problems. We study two methods to do this: kernel weights and Dirichlet process weights. This is a new use of Dirichlet process mixture models. We give empirical analysis for these methods where we show promising results on test problems.

The paper is organized as follows. In Section 2, we review the treatment of search and optimization problems in the presence of a state variable in different communities. In Section 3, we review established function-based optimization methods, propose an algorithm that incorporates a state variable and prove convergence of that algorithm. In Section 4, we review current gradient-based optimization methods, propose an algorithm that incorporates a state variable and prove conver-

gence of the algorithm under certain conditions. In Section 5, we present two weighting schemes, kernel and Dirichlet based weights. We present an empirical analysis of our methods for synthetic newsvendor data and the hour ahead wind commitment problem in Section 6 and a discussion in Section 7.

2 Literature review

Several communities, including operations research, optimization and machine learning, have studied problems with forms similar to the stochastic search problem with a state variable of Equation (2). The problems and solution methods are diverse; even within communities, optimization problems with a state variable are never treated as an entire problem class. We briefly outline the resulting hodgepodge of problems and methods.

Most commonly, stochastic search problems with a state variable are considered individually rather than as an entire problem class. The prevailing approach is to construct a model for Z and use the information in S to supply the parameters. In the case of wind farms, [56] constructs a model for the wind and use a state of the world to determine parameters. They also select which state variables are needed to parameterize the wind distribution. Collectively, creating a model and selecting state variables to generate parameters for that model can require substantial time and domain knowledge.

In some areas, problems with state variables have become their own classes. In the statistics and operations research community, there has been some study of portfolio and bandit problems in the presence of a state variable (called a "covariate" in the bandit literature and "side information" in the learning theory community). [9] and [25] studied portfolio optimization with a finite state variable, but handled it in a manner that amounts to treating each value as a separate problem.

Bandit problems with a state variable are another established area of study. A bandit problem is a sequential decision problem with small, finite set of statistical populations (arms). At each iteration, only one arm can be sampled; a random reward R_i is obtained with probability p_i, where i denotes the arm number. When a state variable $S = s$ is included, the probability of a reward is $p_i(s)$. The goal is to maximize the average reward. This bandit variant was first introduced by [64] and has been studied when the distributions are assumed to have a parametric form [50, 62, 19]. They have also been studied where the mean function has been estimated in a nonparametric fashion [65, 44]; however, decision-making mechanisms vary widely due to the non-convex decision set.

In the optimization community, parametric nonlinear programming includes what can be viewed as a state variable in a math programming setup. The basic problem has the form

$$\min_{x \in \mathcal{X}} F(x, s),$$

where s "parameterizes" the program. Such problems have been used for sensitivity analysis [29, 43] and have been the focus of renewed interest in the model predictive control community [3]. Parametric nonlinear programming, however, is deterministic and it assumes that F is known for a given s.

The machine learning community solves problems with state variables more than any other community. Machine learning is a catch-all term for a large set of subfields, including learning theory, reinforcement learning, classification, Bayesian nonparametrics and many others. Problems with a state variable arise in some of these subfields, such as reinforcement learning and learning theory.

State variables arise in a general way in dynamic programming and reinforcement learning. In these problems, we might be in state S and take an action a, which determines, or influences, the next state S' that we visit. The choice of the action a, then, needs to consider the expected value of being in state S'. A host of algorithms have been proposed to solve this problem class (see, for example, [42, 5, 58, 60, 40, 4]), where the major complication is that we do not know, and have to approximate, the value of being in state S'. Our problem class is closest to the field of stochastic search [57], where we often have to face the challenge of vector-valued (and possibly continuous) decision vectors x. Of all the algorithms in reinforcement learning, our problem is most similar to Q-learning which requires estimating the value of $Q(S, a)$ (in our notation, $Q(S, x)$). Although we do not have to deal with the value of the downstream state, we do have to address the challenge of complex state variables and vector-valued decisions, which the Q-learning literature has not addressed.

State variables are also common in learning theory, where they are called "side information." The portfolio references and many of the bandit references are written from a learning theory perspective. The work closest to ours is [23]; they incorporated a state variable into an online convex optimization problem. In this setting, each iteration a player selects a decision from a convex set and an adversary selects a loss function from a finite set of options. The state variable contains some information about the set of loss functions. They construct an algorithm that minimizes regret, a notion of loss under a worst-case scenario, rather than expected loss. They propose using a combination of a nearest-neighbor and ϵ-net mapping from the state space to the decision space under smoothness constraints. Values are updated by gradient observations. Their algorithm does not converge to a fixed decision for a given state with more than one possible loss function.

We now turn to the first of our methods, function-based optimization.

3 Function-based optimization with an observable state variable

We use function-based optimization when a single outcome ω can tell us the value of all decisions given that outcome [24, 46, 55, 54]. For example, in the hour ahead wind commitment problem, if the wind is known, then the value of all commitment levels is known. Function-based optimization relies on sampling a set of scenarios, $\omega_1, \ldots, \omega_n$ from Ω, to approximate the expectation:

$$\min_{x \in \mathcal{X}} \frac{1}{n} \sum_{i=1}^{n} F(x, Z(\omega_i)). \tag{3}$$

Equation (3) is deterministic and deterministic methods can be used. They can accommodate complex constraint sets but require that the entire function is known for an outcome ω. The following asymptotic results hold for function-based optimization. Let x^* be the true solution and x_n^* be the solution to Equation (3). Under sufficient conditions, $x_n^* \to x^*$ almost surely as $n \to \infty$ [46, 39, 20]. Moreover, [53] and [48] show asymptotic normality under stricter conditions than those required for convergence.

Function-based optimization has been well studied, but under a variety of names. [46], [55], and [20] call it sample path optimization; [49] use likelihood ratios to approximate the optimization problem, calling it the stochastic counterpart method; [24] studied this method and call it retrospective optimization; within the stochastic programming literature, it is often known as sample average approximation [54, 31] and scenario optimization [6]. A similar method has been used in discrete event systems and gradient estimation, called perturbation analysis [26, 17].

3.1 Algorithm for function-based optimization

Extensions of existing algorithms to include a state variable are fairly straightforward. New observations of F have the form $F(x, S_i, Z(\omega_{i+1}))$ where S_i is a random state variable. It is difficult to place a prior over the space of all convex functions or create a joint distribution over states and the space of convex functions. Therefore, we develop a locally weighted average of the observations to approximate the true mean function at a given state.

Let $s \in \mathcal{S}$ be a fixed query state. We would like to minimize $\mathbb{E}[F(x, s, Z)]$ with respect to x. Let $(S_i, Z(\omega_{i+1}))_{i=0}^{n-1}$ be a set of n observations and $(w_n(s, S_i))_{i=0}^{n-1}$ be a set of weights based on the query state and the observed states where

$$\sum_{i=0}^{n-1} w_n(s, S_i) = 1.$$

The weight functions may change with the number of observations n. Set

$$\bar{F}_n(x|s) = \sum_{i=0}^{n-1} w_n(s, S_i) F(x, S_i, Z(\omega_{i+1})). \tag{4}$$

The optimization problem becomes

$$\min_{x \in \mathcal{X}} \bar{F}_n(x|s). \tag{5}$$

This produces an average of observations weighted by how close the observed states are to the current state. $F(x, S_i, Z(\omega_{i+1}))$ is convex in x for every S_i and ω_{i+1}, so $\bar{F}_n(x|s)$ is convex. This is particularly helpful because Equation (5) can then be solved with any number of deterministic solvers. We give the general procedure in Algorithm 1.

Algorithm 1: Function-based optimization with an observable state variable

Require: Query state s.

1: **for** $i = 0$ to $n - 1$ **do**
2: Observe random state S_i.
3: Observe random function $F(x, S_i, Z(\omega_{i+1}))$.
4: **end for**
5: Generate weights $(w_n(s, S_i))_{i=0}^{n-1}$.
6: Set

$$\bar{F}_n(x|s) = \sum_{i=0}^{n-1} w_n(s, S_i) F(x, S_i, Z(\omega_{i+1})).$$

7: Solve

$$x_n^*(s) = \min_{x \in \mathcal{X}} \bar{F}_n(x|s).$$

Define $x_n^*(s)$ as

$$x_n^*(s) = \arg\min_{x \in \mathcal{X}} \bar{F}_n(x|s),$$

and the true optimal decision $x^*(s)$ as

$$x^*(s) = \arg\min_{x \in \mathcal{X}} \mathbb{E}\left[F(x, s, Z) \,|\, S = s\right].$$

Let $\mathbb{E}\left[F(x,s,Z)\right] = F(x|s)$. We would like the approximation to have the following property,

$$\lim_{n\to\infty} x_n^*(s) = \lim_{n\to\infty} \arg\min_{x\in\mathcal{X}} \bar{F}_n(x|s) = \arg\min_{x\in\mathcal{X}} F(x|s) = x^*(s), \text{ almost surely.}$$

We discuss convergence properties in the following subsection.

3.2 Convergence analysis

Let $x_n^*(s)$ be the solution to Equation (5) and let $x^*(s)$ be the true solution. We would like $x_n^*(s) \to x^*(s)$ almost surely, pointwise in s, for s in a compact subset of the state space, denoted $\mathcal{S}_D$. In Theorem 3.1, we give conditions under which almost sure convergence occurs.

Before we state the main theorem, we give a set of assumptions.

(A3.1) $\mathcal{X}$ is a convex subset of $\mathbb{R}^d$ and $\mathcal{S}_D$ is a compact subset of $\mathcal{S}$.

(A3.2) The function $F(x,s,Z(\omega))$ is almost surely convex and continuous in $x \in \mathcal{X}$ for every $s \in \mathcal{S}_D$.

(A3.3) For every fixed $s \in \mathcal{S}_D$ and $x \in \mathcal{X}$, let $(w_n(s,S_i)_{0:n})_{n=0}^{\infty}$ and the distribution of (Z,S) be such that

$$\lim_{n\to\infty} \sum_{i=0}^{n-1} w_n(s,S_i) F(x,S_i,Z(\omega_{i+1})) = F(x|s).$$

(A3.4) For every fixed $s \in \mathcal{S}_D$, the function $\mathbb{E}[F(x,s,Z(\omega)]$ has a unique minimizer.

Assumptions (A3.1)–(A3.2) places bounds on the decision set and state set. Assumption (A3.3) places a pointwise convergence condition on the functional estimator; this places restrictions on both the weighting functions $w_n(s,S_i)$ and the distribution of $Z|S=s$. See Section 5 for a discussion on this assumption. Assumption (A3.4) assures that there is only one optimal decision per state. The main convergence theorem for function-based optimization is as follows.

Theorem 3.1. *Let $\bar{F}_n(x|s) = \sum_{i=1}^{n} w_n(s,S_i)F(x,S_i,Z(\omega_{i+1}))$. Suppose that assumptions (A3.1)–(A3.4) hold. Then, $x_n^*(s) \to x^*(s)$ almost surely, pointwise for every $s \in \mathcal{S}_D$.*

The proof of Theorem 3.1 relies heavily on Proposition 2.4 and Theorem 3.7 of [46]. We state them now, modified for our setting.

Proposition 3.2 (Proposition 2.4 of [46])**.** *If, for every $s \in \mathcal{S}_D$, $\bar{F}_n(x|s)$ converges uniformly to $F(x|s)$ on all compact, non-empty subsets of $\mathcal{X}$, then $\bar{F}_n(x|s)$ epiconverges to $F(x|s)$.*

Theorem 3.3 (Corollary 3.11 of [46])**.** *Suppose that $\bar{F}_n(x|s)$ epiconverges to $F(x|s)$ and that $F(x|s)$ has a unique minimizer for a fixed $s \in \mathcal{S}_D$. Then $x_n^*(s)$ converges almost surely to $x^*(s)$.*

Proof of Theorem 3.1. Fix s in $\mathcal{S}_D$. We show almost sure convergence by satisfying the conditions of Corollary 3.11 in [46]. First, we show that $\bar{F}_n(x|s) \to F(x|s)$ uniformly for $x \in \mathcal{X}_D$, where $\mathcal{X}_D$ is a compact subset of $\mathcal{X}$. Because $\bar{F}_n(x|s)$ is bounded and continuous, it is equicontinuous; because it is equicontinuous and converges pointwise in x to $F(x|s)$, $\bar{F}_n(x|s)$ converges uniformly to $F(x|s)$. Continuity of $F_n(x|s)$ and uniform convergence to $F(x|s)$ satisfy the conditions of Proposition 2.4 of [46], which in turn satisfies the conditions of Corollary 3.11.

We discuss the choice of weight functions in Section 5. Before that, however, we present an algorithm and theoretical results for gradient-based optimization with a state variable.

4 Gradient-based optimization with an observable state variable

In gradient-based optimization, we no longer observe an entire function $F(x, S_n, Z(\omega_{n+1}))$, but only a derivative taken at x_i,

$$\hat{\beta}(x_n, S_n, \omega_{n+1}) = \nabla_x F(x_n, S_n, Z(\omega_{n+1})).$$

Stochastic approximation is the most popular way to solve stochastic search problems using a gradient; it modifies gradient descent algorithms to account for random gradients [45, 30]. The general idea is to optimize x by iterating,

$$x_{n+1} = \Gamma_{\mathcal{X}} \left(x_n - a_n \nabla_x F(x_n, Z(\omega_{n+1})) \right), \tag{6}$$

where $\Gamma_{\mathcal{X}}$ is a projection back into the constraint set $\mathcal{X}$, $\nabla_x F(x_n, Z(\omega_{n+1}))$ is the stochastic gradient at x_n and a_n is a stepsize. Another approach to gradient-based optimization uses construction of piecewise linear, convex functions to approximate $F(x)$ [18, 41]; we will follow the second approach.

Including a state variable into gradient-based optimization is less straightforward than it is for function-based optimization. We encounter difficulties because we choose x_n given S_n; Equation (6) works because *only* x_n changes. When we include an observed state S_n, the decision x_n is based on the state S_n. Therefore, it cannot be chosen in an iterative manner directly from x_{n-1}, which is based on the state S_{n-1}. Additionally, constructing the approximate function $\bar{F}_n(x|s)$ in a convex manner is not trivial because the gradient observations are based on *both* x_n and S_n. In this section, we give an algorithm for gradient-based optimization with a state variable, along with convergence analysis for that algorithm.

4.1 Algorithm for gradient-based optimization

We propose modeling $F(x|s)$ with a piecewise linear, convex, separable approximation. Even if $F(x|s)$ is not itself separable, we aim to approximate it with a simpler (separable) function that has the same minimum for every fixed s. Approximating the minimum well is easier than approximating the entire function [8, 41]. Moreover, convex interpolation is easier in one dimension than multiple dimensions. We approximate $\mathbb{E}[F(x, s, Z)]$ by a series of separable functions,

$$\bar{F}_n(x|s) = \sum_{k=1}^{d} f_n^k(x^k|s),$$

where x^k is the k^{th} component of x and $f_n^k(x|s)$ is a univariate, piecewise linear function in x. We enforce convexity restrictions on marginal functions $f_n^k(x|s)$ for every $s \in \mathcal{S}$. We assume the existence of stochastic gradients, $\hat{\beta}(x, s, \omega) = \nabla_x F(x, s, Z(\omega))$, which are obtained as a response instead of $F(x, s, Z(\omega))$.

The observations $(x_i, S_i, \hat{\beta}(x_i, S_i, \omega_{i+1}))_{i=0}^{n-1}$ are used to update $\bar{F}_n(x|s)$ sequentially. We want to assemble a set of d piecewise linear marginal functions $f_n^k(x|s)$ by constructing a series of slopes, $v_{0:n-1}^k(S_n)$, based on $\hat{\beta}_{1:n}$. We use weights to group the gradients from states "similar" to S_n. We outline the algorithm as follows.

Step 1: Observe S_n and generate $(w_n(S_n, S_i))_{i=0}^{n-1}$ This is discussed in Section 5.

Step 2: Construct slopes for $f_n^k(x|S_n)$ given $\hat{\beta}_{1:n}$, $x_{0:n-1}$ and $(w_n(S_n, S_i))_{i=0}^{n-1}$ Fix k. We begin by placing the observed decisions in ascending order:

$$x_{[0]}^k \leq x_{[1]}^k \leq \cdots \leq x_{[n-1]}^k,$$

where $[0], \ldots, [n-1]$ is the ordered numbering. A necessary and sufficient condition for $f_n^k(x|S_n)$ to be convex is for the slopes to be nondecreasing; that is,

$$\frac{d}{dx} f_n^k(x|S_n) \leq \frac{d}{dy} f_n^k(y|S_n)$$

for every $x \leq y$. We find a set of slopes $v_{n,[0]}^k(S_n) \leq \cdots \leq v_{n,[n-1]}^k(S_n)$ corresponding to the ordered decisions $x_{[0]}^k, \ldots, x_{[n-1]}^k$ using weighted least squares minimization, which is a quadratic program,

$$v_n^k(S_n) = \arg\min_v \sum_{i=0}^{n-1} w_n\left(S_n, S_{[i]}\right)\left(\hat{\beta}(x_{[i]}^k, S_{[i]}, \omega_{[i+1]}) - v_{[i]}\right)^2, \tag{7}$$
$$\text{subject to : } v_{[i-1]} \leq v_{[i]}, \quad i = 1, \ldots, n-1.$$

Step 3: Reconstruct $f_n^k(x|S_n)$, $\bar{F}_n(x|s)$ given $v_n^k(S_n)$ Suppose that $\mathcal{X}$ is compact; there exists a minimum value x_{min}^k and a maximum value x_{max}^k for each dimension k. Set $x_{[-1]}^k = x_{min}^k$ and $x_{[n]}^k = x_{max}^k$. Define $f_n^k(x|S_n)$ as follows,

$$f_n^k(x|S_n) = \sum_{i=0}^{\ell} v_{n,[i]}^k(S_n)\left(x_{[i]}^k - x_{[i-1]}^k\right) + v_{n,[\ell]}^k(S_n)\left(x - x_{[\ell]}^k\right), \tag{8}$$

where ℓ is the smallest index such that $x_{[\ell]}^k \leq x < x_{[\ell+1]}^k$. Set

$$\bar{F}_n(x|S_n) = \sum_{k=1}^{d} f_n^k(x|S_n).$$

Note that the reconstruction is the same as the original up to a constant, which does not affect the optimal decision.

Step 4: Choose x_n given $\bar{F}_n(x|S_n)$ We want to choose an x_n so that we learn as much as possible for an arbitrary s. This is done by picking x_n as follows,

$$x_n = \arg\min_{x \in \mathcal{X}} \bar{F}_n(x|S_n). \tag{9}$$

Note that $\bar{F}_n$ is a piecewise linear function; if $\mathcal{X}$ is a linear constraint set, the minimum can be found with a linear program.

A full overview of this procedure is given in Algorithm 2. Notice that while the function-based optimization of Algorithm 1 can essentially be performed in a *post hoc* batch setting, Algorithm 2 can only be performed in an online setting. We discuss a grid-based extension of Algorithm 2 in the following subsection.

Algorithm 2: Gradient-based optimization with an observable state variable

Require: Query state s, initial slopes v_0.

1: **for** $i = 0$ to $n-1$ **do**

2: Observe random state S_i.

3: Generate weights $(w_i(S_i, S_j))_{j=0}^{i-1}$. (See Section 5.)

4: **for** $k = 1$ to d **do**

5: Place decision observations in ascending order: $x^k_{[0]} \leq \cdots \leq x^k_{[i-1]}$.

6: Compute slopes $v_i^k(S_i)$ by

$$v_i^k(S_i) = \arg\min_v \sum_{j=0}^{i-1} w_i\left(S_i, S_{[j]}\right)\left(\hat{\beta}(x^k_{[j]}, S_{[j]}, \omega_{[j+1]}) - v_{[j]}\right)^2,$$
$$\text{subject to : } v_{[j-1]} \leq v_{[j]}, \quad j = 1, \ldots, i-1.$$

7: Reconstruct marginal function $f_i^k(x^k|S_i)$ using slopes $v_i^k(S_i)$ as per Equation (8).

8: **end for**

9: Set

$$x_i = \arg\min_{x\in\mathcal{X}} \sum_{k=1}^{d} f_n^k(x^k|S_i).$$

10: Observe random gradient $\hat{\beta}(x_i, S_i, \omega_{i+1}) = \nabla_x F(x_i, S_i, Z(\omega_{i+1}))$.

11: **end for**

12: Compute $v_n^k(s)$, $k = 1, \ldots, d$ as in Step 6.

13: Compute $f_n^k(x^k|s)$, $k = 1, \ldots, d$ using $v_n^k(s)$ as in Step 7.

14: Set

$$x_n^*(s) = \arg\min_{x\in\mathcal{X}} \sum_{k=1}^{d} f_n^k(x^k|s).$$

4.2 Grid-based decisions

One of the more computationally heavy parts of Algorithm 2 is Step 6, the projection of the slopes to an ordered space via a quadratic program. The number of parameters and constraints grows linearly with the number of observations. In some numerical work, we have found it easier to make decisions on a grid format.

Fix k. If $\mathcal{X}$ is compact, we can create an arbitrarily fine grid on the k^{th} dimension with a finite number of points,

$$a_1^k \leq \cdots \leq a_{N(k)}^k.$$

Suppose that they are evenly spaced with distance α and let the intersection of this set of points with $\mathcal{X}$ be denoted $\mathcal{X}^G$. If all decisions are selected from $\mathcal{X}^G$, the parameter and constraint set for Equation (7) never grows.

The inclusion of a grid changes the way that we select x_n. Let $\hat{x}_n$ be the solution to the original approximated problem,

$$\hat{x}_n = \min_{x\in\mathcal{X}} \bar{F}_n(x|S_n).$$

We can generate x_n from $\hat{x}_n$ in one of two ways: 1) projection to the nearest feasible point in $\mathcal{X}^G$,

or 2) random selection of a neighboring point in $\mathcal{X}^G$. The second option is computationally simple and can be guaranteed to break the constraints by at most an arbitrarily small amount through grid construction. We use the second method in our numerical examples.

4.3 Convergence analysis

We now give conditions under which Algorithm 2 converges in probability to the global optimum pointwise for every state s is a set of query states $\mathcal{S}_D$. The observed state S_n is likely not the same as our query state s, but we often care about what the approximation says is the best decision for s after n observations, defined as $x_n^*(s)$. Set

$$x_n^*(s) = \arg\min_{x\in\mathcal{X}} \bar{F}_n(x|s).$$

We would like $x_n^*(s)$ to approach the true optimal decision, $x^*(s)$, as $n \to \infty$, where

$$x^*(s) = \arg\min_{x\in\mathcal{X}} \mathbb{E}\left[F(x, s, Z) \mid S = s\right] = \arg\min_{x\in\mathcal{X}} F(x|s).$$

The outline of the proof is to show that the decisions sampled for a sufficiently small neighborhood of states around s accumulates in an arbitrarily small neighborhood around $x^*(s)$. This is done by verifying optimality conditions in the accumulation regions. First, however, we need to define a set of assumptions.

Suppose that there are a finite set of functions $g_i(x)$, $i = 1, \ldots, p$ and $h_j(x)$, $j = 1, \ldots, q$ such that the constraint set $\mathcal{X}$ can be written as

$$\mathcal{X} = \{x \,:\, g_i(x) \leq 0,\ h_j(x) = 0,\ i = 1, \ldots, d,\ j = 1, \ldots, q\}.$$

The following conditions require separability and strong convexity of the objective function, along with differentiability of the objective function and constraints.

(A4.1) For every $s \in \mathcal{S}$, the function $F(x|s)$ is separable in x,

$$F(x|s) = \sum_{k=1}^{d} f^k(x^k|s).$$

Define the gradient function

$$v^k(x^k, s) = \frac{\partial}{\partial x^k} f^k(x^k|s).$$

(A4.2) Let $F(x|s)$ be strongly convex in x for every $s \in \mathcal{S}$ with parameter m; that is,

$$\left(\nabla_x F(x|s) - \nabla_y F(y|s)\right)^T (x - y) \geq m||x - y||_2^2.$$

(A4.3) $F(x|s)$ is twice continuously differentiable in x and s for every $x \in \mathcal{X}$ and $s \in \mathcal{S}$.

(A4.4) If the state space and decision space are well sampled around the point (x, s) and every observation is unbiased, that is,

$$\mathbb{E}\left[\hat{\beta}(x', s')\right] = v(x', s'),$$

with $v(x, s)$ as in (A4.1), then

$$\lim_{n\to\infty} v_n(x, s) = v(x, s),$$

where $v_n(x, s)$ is defined by the projection in Equation (7).

Now we discuss the optimality conditions. Define $N_{\mathcal{X}}(x)$, the normal cone to $\mathcal{X}$ at x,

$$N_{\mathcal{X}}(x) = \left\{ y \in \mathbb{R}^d \mid \langle y, x - v \rangle \geq 0, \forall v \in \mathcal{X} \right\}.$$

Define the subgradient of $F(x|s)$ at x as $\partial F(x|s)$. Then, the point x^* is a global minimizer of $F(x|s)$ for a fixed s if and only if

$$0 \in \partial F(x^*|s) + N_{\mathcal{X}}(x^*). \tag{10}$$

We aim to show that as $n \to \infty$, $x_n^*(s)$ and only $x_n^*(s)$ satisfies Equation (10).

Theorem 4.1. *Let $\mathcal{S}_D$ be a compact subset of $\mathcal{S}$ and assumptions (A4.1)–(A4.4) hold. Then, for every $s \in \mathcal{S}_D$ and every $\epsilon > 0$,*

$$\mathbb{P}\left\{ |x_n^*(s) - x^*(s)| > \epsilon \right\} \to 0$$

as $n \to \infty$.

Proof. Fix $s \in \mathcal{S}_D$ and $\epsilon > 0$. Consider the k^{th} component of the decision variable. Since $\mathcal{X}$ is compact, there exists an x_{min}^k and x_{max}^k such that $x_{min}^k \leq x^k \leq x_{max}^k$ for all $x \in \mathcal{X}$. We place an $\epsilon/4md$-net on this axis such that $x^{*,k}(s)$ is in the center of one of the partitions, where m is the strong convexity constant. Label the regions $a_1^k, \ldots, a_{M(k)}^k$. This is done for all components of the decision variable, $k = 1, \ldots, d$.

Note that for every iteration, the decision x_n and the calculated optimum $x_n^*(s)$ are random variables. Let $p_n^k(a_i, s)$ be the probability that $x_n^{*,k}(s) \in a_i^k$,

$$p_n^k(a_i, s) = \mathbb{P}\left\{ x_n^{*,k}(s) \in a_i^k \right\}, \quad i = 1, \ldots M(k).$$

Fix $\gamma > 0$. Let $\mathcal{L}^k(s)$ be the set of partitions a_i^k where there exists a subsequence $(n_j)_{j=1}^{\infty}$ such that

$$\mathcal{L}^k(s) = \left\{ a_i^k \mid \liminf_{n_j \to \infty} p_{n_j}^k(a_i^k, s) > \gamma \right\}.$$

That is, $\mathcal{L}^k$ is the set of all decisions sampled infinitely often with at least limiting probability γ. Let

$$A_\epsilon^k(s) = \left\{ a_i^k \mid \left| a_i^k - x^{*,k}(s) \right| < \epsilon/d \right\}.$$

We will show that $\mathcal{L}^k(s)$ is non-empty and that $\mathcal{L}^k(s) \subset A_\epsilon^k(s)$; this is done in two phases.

Claim 1: $\mathcal{L}^k(s) \neq \emptyset$. If $\gamma \leq 1/M(k)$, then at least one decision will be sampled infinitely often with at least probability γ.

Claim 2: $\mathcal{L}^k(s) \subset A_\epsilon^k(s)$. Suppose $b \in \mathcal{L}^k(s)$ and $b \notin A_\epsilon^k(s)$. Then, there exists an infinite subsequence $(n_i)_{i=1}^{\infty}$ such that $x_{n_i}^*(s) \in b$. Since the marginal values, $v_n(x, s)$ are also continuous in s, there exists a set of s' such that $\liminf p_{n_j}^k(b, s') > \gamma$ and $|v^k(x, s) - v^k(x, s')| < \epsilon/4md$ for every $x \in b$. Denote this set by $B(s, b)$. The states in set $B(s, b)$ will be sampled infinitely often, so by (A4.2) and (A4.4), there exists an N such that for every $n \geq N$,

$$\left| v_n^k(x, s) - v(x, s) \right| < m * \epsilon/2md = \epsilon/2d.$$

But by (A4.2) and (A4.3), the function $x^*(s)$ is uniformly continuous in s over $\mathcal{S}_D$. Combining this fact with (A4.1), for sufficiently small ϵ,

$$0 \notin v_n(x, s) + N_{\mathcal{X}}(x)$$

for all $n \geq N$ and all $x \in b$. Therefore, $b \notin \mathcal{L}^k(s)$, so Claim 2 is true and therefore the theorem holds. □

We now discuss the choice of weight functions.

5 Weight functions

The choice of weight functions determines whether assumptions (A3.3) and (A4.4) are satisfied and which distributions of $F(x, s, \omega)$ satisfy them. More importantly, however, the weight functions also determine how well $F_n(x|s)$ approximates $F(x|s)$ with finite sample sizes. Before we discuss the specifics of individual weighting functions, let us discuss how weighting functions are constructed.

Weighting functions rely on density estimation procedures to approximate the conditional density $f(y|s)$, where s is the state and y is the response. Then the mean conditional response $\mathbb{E}[Y|s] = \int yf(y|s)dy$ is calculated, which is the object of interest. The conditional density is approximated by a weighted sum of observations, $\hat{f}_n(y|s)$; it is either expressly composed by observational weights, as in the case of kernel regression, or can be decomposed into observational weights, as in the case of Dirichlet process regression.

In this section, we discuss two weighting schemes in detail: kernels and the Dirichlet process similarity measure. Kernels been well studied and are easy to implement as a weighting scheme. However, they often do not perform well with more complicated problems, such as those with a moderate or large number of covariates. Therefore, we propose a Dirichlet process similarity measure when a richer class of weighting functions is required. We discuss kernel weights in Subsection 5.1 and the Dirichlet process similarity measure along with relevant background material in Subsection 5.2.

5.1 Kernel weights

Kernel weights rely on kernel functions, $K(s)$, to be evaluated at each observation to approximate the conditional density. A common choice for K with continuous covariates is the Gaussian kernel,

$$K_h(s) = (2\pi h)^{-1/2} \exp\{-s^2/2h\},$$

where the variance h is called the bandwidth. Kernel weights have the advantage of being simple and easy to implement. The simplest and most universally applicable weighting scheme is based on the Nadaraya-Watson estimator [35, 63]. If $K(s)$ is the kernel and h_n is the bandwidth after n observations, define

$$w_n(s, S_i) = \frac{K_{h_n}(s - S_i)}{\sum_{j=0}^{n-1} K_{h_n}(s - S_j)}.$$

In the case of function-based optimization, the function estimate $\bar{F}_n(x|s)$ is

$$\bar{F}_n(x|s) = \frac{\sum_{i=0}^{n-1} K_{h_n}(s - S_i) F(x, S_i, Z(\omega_{i+1}))}{\sum_{j=0}^{n-1} K_{h_n}(s - S_j)}.$$

Kernel methods have few requirements to satisfy assumptions (A3.3) and (A4.4). For function-based optimization, assumption (A3.3) is satisfied if:

1. $F(x, s, Z)$ has finite variance for every $x \in \mathcal{X}$, $s \in \mathcal{S}$.
2. $F(x|s) = \mathbb{E}[F(x, s, Z)]$ is continuous in s for every $x \in \mathcal{X}$.

Sufficient conditions for gradient-based assumption (A4.4) are similar:

1. $\hat{\beta}(x, s, Z)$ has finite variance for every $x \in \mathcal{X}$, $s \in \mathcal{S}$.
2. $v(x, s) = \nabla_x \mathbb{E}[F(x, s, Z)]$ is continuous in s and x.

The first condition is assured by assumption (A4.3), so only the second condition must be checked.

Despite ease of use and a guarantee of convergence, kernel estimators require a well-sampled space, are poor in higher dimensions and are highly sensitive to bandwidth size. There is a large literature on bandwidth selection [12, 28], but it is usually chosen by cross-validation. To overcome many of these difficulties, we propose using a Dirichlet process mixture model over the states as an alternative weighting scheme.

5.2 Dirichlet process weights

One of the curses of dimensionality is sparseness of data. As the number of dimensions grows, the distance between observations grows exponentially. In kernel regression, this means that only a handful of observations have weights that are effectively non-zero. Instead of basing weights on a distance that grows quickly with the number of dimensions, we would like to average responses for "similar" observations. Regions of "similarity" can be defined with a clustering algorithm. Dirichlet process mixture models (DPMMs) are Bayesian nonparametric models that produce a distribution over data partitions [38, 27]. They were introduced in the 1970's [13, 7, 1], but have gained popularity in the last fifteen years as more powerful computers have allowed posterior computation [32, 11]. DPMMs have been used for classification [52], clustering [34, 10] and density estimation [11, 33], but we will use the partitioning feature directly.

Dirichlet process mixture models are easiest to understand in the context of density estimaiton. The idea is that complicated distributions, such as the distribution of the state variable, can be modeled as a countable mixture of simpler distributions, such as Gaussians. To make the model more flexible, the number of components is allowed to be countably infinite. The Dirichlet process is a Bayesian model that places a prior on the mixing proportions, leading to a small number of components with non-trivial proportions. Because an infinite number of components are allowed, the Dirichlet process circumvents the issue of determining the "correct" number of components, as is necessary in algorithms like k-means.

The clustering/partitioning property of the Dirichlet process is derived from the mixture assumption. Two observations are in the same cluster or partition if they are generated by the same mixture component. The query state s is also placed in a cluster; the estimated response for s is simply the average of all the responses associated with states in that cluster. However, because the data labels, component locations and mixing proportions are not known, the Dirichlet process produces a distribution over clusterings. We use Monte Carlo methods to integrate over the clusterings.

In this subsection, we discuss the basic properties of Dirichlet process mixture models, how they can be used to generate a weighting function, how it can be approximated, and finally what is required for it to satisfy the optimization algorithm assumptions.

5.2.1 Dirichlet process mixture models

A mixture model represents a distribution, $g_0(s)$, as a weighted sum of simpler distributions, $g(s \mid \theta_i)$, which are parameterized by θ_i,

$$g_0(s) = \sum_{i=1}^{\infty} p_i g(s \mid \theta_i).$$

Here, p_i is the mixing proportion for component i. For example, if $g_0(s)$ is a univariate, continuous distribution, we may wish to represent it as a sum of Gaussian densities,

$$g_0(s) = \sum_{i=1}^{K} p_i \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{1}{2\sigma_i^2}(s-\mu_i)^2}. \tag{11}$$

In Equation (11), g is the Gaussian density; it is parameterized by $\theta_i = (\mu_i, \sigma_i^2)$, the mean and variance for component i.

The difficulties of using a mixture model are 1) determining parameters $(p_i, \theta_i)_{i=1}^K$, and 2) determining K. There are many optimization based algorithms to find (p_i, θ_i) (see [21] for a review), but fewer ways to find a good value for K. However, if we assume $K = \infty$ with only a finite number of components with weights p_i that are effectively non-zero, then we effectively do not have to choose K.

We can use a Dirichlet process (DP) with base measure $\mathbb{G}_0$ and concentration parameter α to place a distribution over the joint distribution of (p_i, θ_i), the mixture proportion and location of component i [13, 1]. A Dirichlet process effectively places a discrete probability distribution over the parameter (θ) space; the θ's that are given positive probability in that space are components of the mixture model. The probability associated with θ is the component weight p. We shall call this distribution over the parameter space P. Note that a Dirichlet process prior does not tell us deterministically what (p_i, θ_i) *will* be; instead it places a distribution over what it *could* be. For that reason, P is actually a random measure. A Dirichlet process mixture model is constructed as follows.

Assume that data $S_1, \ldots, S_n$ are drawn from the same distribution, which is modeled by a mixture over distribution $G(\theta)$. We let $g(\cdot \mid \theta)$ be the density, while $G(\theta)$ is the distribution, for example $N(\mu, \sigma^2)$. Observation S_i is drawn from a component of that model, $G(\theta_i)$, with parameter θ_i. Conditioned on θ_i, S_i has the distribution $G(\theta_i)$. Now, let P be the mixing distribution over θ; we give P a Dirichlet process prior with base distribution $\mathbb{G}_0$ and concentration parameter α. In sum, this produces a Dirichlet process mixture model. We use the following hierarchical model for the DPMM,

$$\begin{aligned} P &\sim DP(\alpha, \mathbb{G}_0), \\ \theta_i | P &\sim P, \\ S_i | \theta_i &\sim G(\theta_i). \end{aligned} \tag{12}$$

Here, "X $\sim$ Y" means "X has the distribution of Y." Note the conditional independence at every level of the Model (12); for example, given θ_i, S_i is independent of P and the other S_j. Distributions F and $\mathbb{G}_0$ often depend on additional hyperparameters; these will be explained in context later.

A Dirichlet process is used as a prior on P because it produces an almost surely discrete distribution over parameters. This is demonstrated when we integrate out P from Model (12) to obtain a conditional distribution of $\theta_n | \theta_{1:n-1}$ [7]

$$\theta_n \,|\, \theta_1, \ldots, \theta_{n-1} \sim \frac{1}{\alpha + n - 1} \sum_{i=1}^{n-1} \delta_{\theta_i} + \frac{\alpha}{\alpha + n - 1} \mathbb{G}_0. \tag{13}$$

Here, δ_θ is the Dirac measure with mass at θ. Equation (13) is known as a *Polya urn posterior*. Note that θ_n has positive probability of assuming the value of one of the previously observed θ_i, but it also can take a completely new value drawn from $\mathbb{G}_0$ with positive probability. The parameter α controls how likely θ_n is to take a new value.

We now give an example of a DPMM. Suppose that $g_0(s)$ is univariate and continuous. An infinite Gaussian mixture model is a good approximation, parameterized by $\theta_i = (\mu_i, \sigma_i^2)$. Let $S_1, \ldots, S_n$ be drawn from this distribution. The mixture model can be written as,

$$\begin{aligned} P &\sim DP(\alpha, \mathbb{G}_0), \\ \theta_i = (\mu_i, \sigma_i^2) \,|\, P &\sim P, \\ S_i \,|\, \theta_i &\sim N(\mu_i, \sigma_i^2). \end{aligned} \tag{14}$$

Often, $\mathbb{G}_0$ is chosen to be conjugate to G to ease posterior sampling; in this case, the conjugate $\mathbb{G}_0$ is Normal-Inverse-Gamma with hyperparameters $(\lambda_0, \nu_0, \alpha_0, \beta_0)$. α is also a hyperparameter; it is usually given a Gamma prior or set equal to 1. We now discuss DPMM weights.

5.2.2 The Dirichlet process weights

Dirichlet process mixture models intrinsically produce a partition structure [38, 27], that is, a clustering of the observed data. We can see this in the Polya urn posterior of Equation (13); each hidden parameter has positive probability of taking the same value as another parameter. If two parameters have the same value, the associated observations are in the same partition/cluster. Because a Dirichlet process places a distribution over component parameters and probabilities, which form a partition of the data, it also places a distribution over all partition structures of the data. We use the partition structure to induce weights on the observations by giving equal weights to all observations that are in the same partition as the current observed state.

Let the cluster/partition C_i be defined as the set of all observations that have the same parameter, $C_i = \{j : \theta_j = \theta_i^*\}$. Let $\mathbf{p} = \{C_1, \ldots, C_{n(\mathbf{p})}\}$ be the partition of the observations $\{1, \ldots, n\}$. Given a partition $\mathbf{p}$, there are $n(\mathbf{p})$ clusters, generating $n(\mathbf{p})$ unique parameter values, $\theta_1^*, \ldots, \theta_{n(\mathbf{p})}^*$. Now suppose that we know the partition $\mathbf{p}$. Given this partition, we wish to place our observed state s into one of the partitions. We do not know its partition, but we can generate a probability that it is in cluster C_i,

$$\begin{aligned} p_s(C_i|\mathbf{p}) &= \mathbb{P}(s \in C_i \,|\, \mathbf{p}, S_{1:n}) \\ &\propto |C_i| \int g(s \,|\, \theta^*) dH_{C_i}(\theta^*), \end{aligned} \tag{15}$$

where $|C_i|$ is the number of elements in C_i, $H_{C_i}(\theta^*)$ is the posterior distribution of θ^* conditioned on the base measure $\mathbb{G}_0$ and the set of observations $\{S_j : S_j \in C_i\}$. Sometimes it is impossible to compute the integral in Equation (15) and it is approximated by Monte Carlo integration conditioned on θ^*.

The probability is calculated for each cluster C_i, $i = 1, \ldots, n(\mathbf{p})$. Given the probabilities $(p_s(C_i|\mathbf{p}))_{i=1}^{n(\mathbf{p})}$, the weighting function is defined by

$$w_n(s, S_i) \,|\, \mathbf{p} = \sum_{j=1}^{n(\mathbf{p})} \frac{p_s(C_j \,|\, \mathbf{p})}{|C_j|} \mathbf{1}_{\{S_i \in C_j\}}. \tag{16}$$

Equation (16) is different from the conditional Polya urn posterior of Equation (13): unlike the observed states, the query state is not allowed to be in a cluster by itself. We do this because we often do not have prior information on the response distribution.

Equation (16) is conditioned on a partition structure, but the Dirichlet process produces a distribution over partition structures. Let $\pi(\mathbf{p})$ be the partition probability function, which is a

prior distribution for partitions $\mathbf{p}$,

$$\pi(\mathbf{p}) = \frac{\alpha^{n(\mathbf{p})-1} \prod_{j=1}^{n(\mathbf{p})} (|C_j| - 1)!}{\prod_{j=1}^{n-1} (\alpha + j)}. \tag{17}$$

The posterior distribution, $\pi(\mathbf{p}|S_{1:n})$ has the form

$$\pi(\mathbf{p}|S_{1:n}) \propto \pi(\mathbf{p}) \prod_{j=1}^{n(\mathbf{p})} \int_{\mathcal{T}} \prod_{i \in C_j} g(S_i|\theta)\, \mathbb{G}_0(d\theta). \tag{18}$$

We can combine Equations (16) and (18) to obtain unconditional weights,

$$w_n(s, S_i) = \sum_{\mathbf{p}} \pi(\mathbf{p}|S_{1:n}) \left(\sum_{j=1}^{n(\mathbf{p})} \frac{p_s(C_j \,|\, \mathbf{p})}{|C_j|} \mathbf{1}_{\{S_i \in C_j\}} \right). \tag{19}$$

The conditional weights in Equation (16) are easy to compute, but it is nearly impossible to compute $\pi(\mathbf{p}|S_{1:n})$ or even enumerate all possible partitions, $\mathbf{p}$. Therefore, we approximate Equation (19) by performing a Monte Carlo integration over the partitions. We obtain M i.i.d. posterior partition samples, $(\mathbf{p}^{(m)})_{m=1}^{M}$ and set

$$w_n(s, S_i) \approx \frac{1}{M} \sum_{m=1}^{M} \sum_{j=1}^{n(\mathbf{p}^{(m)})} \frac{p_s(C_j \,|\, \mathbf{p}^{(m)})}{|C_j|} \mathbf{1}_{\{S_i \in C_j\}}. \tag{20}$$

We now show how to obtain $(\mathbf{p}^{(m)})_{m=1}^{M}$ given $S_1, \ldots, S_n$ using Gibbs sampling.

5.2.3 Gibbs Sampler for the State Variable

Markov Chain Monte Carlo (MCMC) [36] is the most popular and simple way to obtain partition structure samples, $(\mathbf{p}^{(m)})_{m=1}^{M}$. MCMC is based on constructing a Markov chain that has a limiting distribution equal to the partition structure posterior distribution. The most common way to implement MCMC is by Gibbs sampling. In Gibbs sampling, the partition $\mathbf{p} = \{C_1, \ldots, C_{n(\mathbf{p})}\}$ and possibly the parameters $\theta^* = (\theta_1^*, \ldots, \theta_{n(\mathbf{p})}^*)$ form the state of the Markov chain. If $\mathbb{G}_0$ is a conjugate prior for G, then θ^* is not needed and the sampler is called "collapsed"; otherwise, θ^* is included. Every iteration we choose an i sequentially, with $1 \leq i \leq n$, and remove S_i from the clustering. Then, we randomly assign it to either 1) one of the existing clusters, or 2) to a new cluster. The assignment probabilities are chosen such that the limiting distribution of the Markov chain is the posterior distribution $\pi(\mathbf{p} \,|\, S_{1:n})$. We follow Algorithm 3 of [36]; this algorithm is designed for problems with base measure $\mathbb{G}_0$ conjugate to the state conditional distribution G. However, efficient algorithms for non-conjugate base measures can also be found in [36].

Let c_i be the partition number for observed state S_i and let n_c be the number of observations in cluster/partition c. The partition $\mathbf{p}$ can readily be reconstructed from $c_1, \ldots, c_n$; $c_1, \ldots, c_n$ have the same relation to $C_1, \ldots, C_{n(\mathbf{p})}$ that $\theta_1, \ldots, \theta_n$ have to $\theta_1^*, \ldots, \theta_{n(\mathbf{p})}^*$. That is, $C_i = \{j : c_j = i\}$.

Gibbs sampling repeatedly samples a Markov chain where the limiting distribution is the posterior distribution of Equation (12). The state of the chain is $\mathbf{c} = (c_1, \ldots, c_n)$. We move around the space by changing the partition for one observed state S_i probabilistically while holding the partitions of all the other states fixed. Let

$$\mathbf{c}_{-i} = (c_1, \ldots, c_{i-1}, c_{i+1}, \ldots, c_n),$$

Algorithm 3: Gibbs sampler for $\mathbf{p}|S_{1:n}$ with a conjugate base measure

Require: Observed states $S_1, \ldots, S_n$.

Initialize $\mathbf{c}$, set $m = 1$.
while $m \leq M$ **do**
  for $i = 1$ to n **do**
    Sample $c_i|\mathbf{c}_{-i}$ using transition probabilities

$$\mathbb{P}(c_i = c|\mathbf{c}_{-i}, S_i) = b\frac{n_{-i,c}}{n+\alpha}\int g(S_i|\theta^*)dH_{-i,c}(\theta^*),\ c = c_j,\ j \neq i,$$
$$\mathbb{P}(c_i \neq c_j\,\forall\, j|\mathbf{c}_{-i}, S_i) = b\frac{\alpha}{n+\alpha}\int g(S_i|\theta^*)d\mathbb{G}_0(\theta^*), \qquad \text{otherwise}$$

  end for
  if Convergence criteria are satisfied **then**
    Construct $\mathbf{p}$ from $\mathbf{c}$,

$$C_i = \{j\,:\,c_j = i\},\quad i = 1, \ldots, \max(\mathbf{c}).$$

    Set $\mathbf{p}^{(m)} = \mathbf{p}$.
    Set $m = m + 1$.
  end if
end while

and $n_{-i,c}$ be the number of observations in partition c when i has been removed. Fixing $\mathbf{c}_{-i}$, we compute the transition probabilities for c_i as follows:

$$\mathbb{P}(c_i = c|\mathbf{c}_{-i}, S_i) = b\frac{n_{-i,c}}{n+\alpha}\int g(S_i|\theta^*)dH_{-i,c}(\theta^*), \qquad \text{if } c = c_j \text{ for } j \neq i,$$
$$\mathbb{P}(c_i \neq c_j\ \forall\, j \neq i|\mathbf{c}_{-i}, S_i) = b\frac{\alpha}{n+\alpha}\int g(S_i|\theta^*)d\mathbb{G}_0(\theta^*), \qquad \text{otherwise.}$$

Here, b is a normalizing constant, $H_{-i,c}(\theta^*)$ is the posterior distribution conditioned on the base measure $\mathbb{G}_0$ and set of observations $\{S_j : \theta_j = \theta_c^*,\ j \neq i\}$. Because $\mathbb{G}_0$ is conjugate a conjugate prior for g, $H_{-i,c}(\theta^*)$ has a closed form solution (see [14] for a comprehensive list of posterior forms). The chain is run until some convergence criteria are met. Convergence is notoriously hard to diagnose, but general rules of thumb include setting a very large number of "burn-in" iterations and discarding all observations before the burn-in or running a number of chains and comparing the within and between sequence parameter variance or posterior probability. See [14] for a more thorough discussion of convergence criteria. The Gibbs sampler for $(\mathbf{p}^{(m)})_{m=1}^M$ is given in Algorithm 3.

We now discuss under which conditions Dirichlet process weights satisfy the convergence conditions for stochastic search with a state variable.

5.2.4 Convergence conditions

Assumptions (A3.3) and (A4.4) concern the consistency of the regression estimator, which in turn depends on the underlying observation distribution and weights. Dirichlet process weights also produce a density estimate; weak consistency of this density estimate is enough to satisfy the assumptions.

Posterior consistency is the notion that the posterior distribution of the DP mixture model in Equation (12) accumulates in neighborhoods "close" to the true distribution of the observations. For weak consistency, we would like it to accumulate in weak neighborhoods.

Weak consistency for DPMMs depends on both the model and the base measure; it has been examined in numerous articles [2, 15, 16, 61, 59, 47]. Gaussian DPMMs with a conjugate base measure, here the Normal-Inverse Wishart, are weakly consistent for many continuous densities. See [15], [61], [59] and [47] for conditions. [16] also show that DPMMs are consistent for finite distributions provided that the base measure $\mathbb{G}_0$ gives full support to the required probability simplex.

Assumption (A3.3) is satisfied if:

1. The DPMM and base measures are weakly consistent for the true state distribution $g_0(s)$.
2. The distribution of $F(x, s, Z)$ is weakly convergent in s for every $x \in \mathcal{X}$. That is, $F(x, s', Z) \Rightarrow F(x, s, Z)$ as $s' \to s$, where "$\Rightarrow$" denotes weak convergence.
3. $F(x, s, Z)$ is almost surely bounded and continuous in x and s.

These three conditions combine to produce a weakly convergent Bayes estimate of the conditional density, which is generated by the Dirichlet process weights on the observations. These are much heavier conditions than those for kernel-based weights.

Because random sampling is required in all current weak consistency results for DPMMs, it is an open question under which conditions Dirichlet process weights satisfy assumption (A4.4). We now study function-based optimization, gradient-based optimization and the weighting functions empirically.

6 Empirical analysis

We analyzed the performance of function-based and gradient-based optimization algorithms in conjunction with kernel- and Dirichlet-based weights on the hour ahead wind commitment problem and the two-product newsvendor problem.

6.1 Multi-product constrained newsvendor problem

A multi-product newsvendor problem is a classic operations research inventory management problem [37]. In the two product problem, a newsvendor is selling products A and B. She must decide how much of each product to stock in the face of random demand, D_A and D_B. A and B can be be bought for (c_A, c_B) and sold for (p_A, p_B), respectively. Any inventory not sold is lost. Let (x_A, x_B) be the stocking decisions for A and B respectively; it is subject to a budget constraint, $b_A\, x_A + b_B\, x_B \leq b$, and a storage constraint, $r_A\, x_A + r_B\, x_B \leq r$. An observable state $S = (S_1, S_2)$ contains information about D_A and D_B. The problem is,

$$\begin{aligned} \max_{x_A,\, x_B} \; & -c_A\, x_A - c_B\, x_B + \mathbb{E}\left[p_A \min\left(x_A, D_A\right) + p_B \min\left(x_B, D_B\right) | S = s\right] \\ \text{subject to}: \; & b_A\, x_A + b_B\, x_B \leq b, \\ & r_A\, x_A + r_B\, x_B \leq r. \end{aligned} \tag{21}$$

We generated data for Problem (21) in the following way. Demand and two state variables were generated in a jointly trimodal Gaussian mixture with parameters $D_A = 1/3 * [N(10, 4) + N(28, 5) + N(30, 5)]$, $D_B = 1/3 * [N(10, 3) + N(22, 9) + N(35, 12)]$; there were two state variables,

S_1 and S_2; parameters were also included in the Gaussian trimodal mixture; all parameters were generated as follows: $\mu_{a,i} \sim N(0,3)$, $\sigma^2_{a,i}) \sim Inverse\,Gamma(1,1)$, $a = A, B$, $i = 1, 2, 3$.

6.1.1 Newsvendor Competitors

The following methods were compared:

1. *Function-based with kernel.* Bandwidth is selected according to the "rule of thumb" method of the *np* package for R,
$$h_j = 1.06\sigma_j n^{-1/(4+d)},$$
where σ_j is defined as min(sd, interquartile range/1.349) [22].

2. *Gradient-based with kernel.* Bandwidth selection was the same as in the function-based case; decisions were made online after an initialization period of 5 random decisions.

3. *Function-based with Dirichlet weights.* We used the following hierarchical model,
$$\begin{aligned} P &\sim DP(\alpha, \mathbb{G}_0), \\ \theta_i = (\mu_i, \sigma_i^2)|P &\sim P, \\ S_{i,j}|\theta_i &\sim N(\mu_{i,j}, \sigma^2_{i,j}),\ j = 1, 2. \end{aligned} \tag{22}$$
Conjugate base measures were used. Posterior samples were drawn using Gibbs sampling with a fully collapsed sampler run for 500 iterations with a 200 iteration burn-in with samples taken every 5 iterations.

4. *Gradient-based with Dirichlet weights.* Dirichlet process mixture model was as in Model 22; base measures and sampling procedures were the same. Decisions were made online after an initialization period of 5 random decisions.

5. *Optimal.* These are the optimal decisions with known mixing parameters and unknown components. That is, we know that the distribution of D_A is $1/3*[N(10,4)+N(28,5)+N(30,5)]$, we have similar knowledge for D_B, S_1 and S_2 and we know their joint, but we do not know from which component (1, 2 or 3) the observation was drawn. Decisions were made by solving the unconstrained newsvendor problem, projecting onto the constraint set (if necessary) and then performing a boundary search until the the optimal decision was reached.

6.1.2 Newsvendor Results

Decisions were made under each regime over eight sample paths; 100 test state/demand pairs were fixed and decisions were made for these problems given the observed states/decisions in the sample path for each method. Results are given in Figure 6.1. The kernel and Dirichlet process weights performed approximately equally for each method. Gradient-based optimization produced more stable results than function-based optimization. This is due to a combination of factors. First, because gradient-based methods only update values in one location at a time, they are more stable than function-based methods, which update entire functions. Hence function-based methods are more sensitive to noise. Second, the relatively small training sample size did not lead to a well-defined clustering. The low-demand cluster was situated between the medium- and high-demand clusters. This created greater variance among the function-based estimates, which are already susceptible to noise.

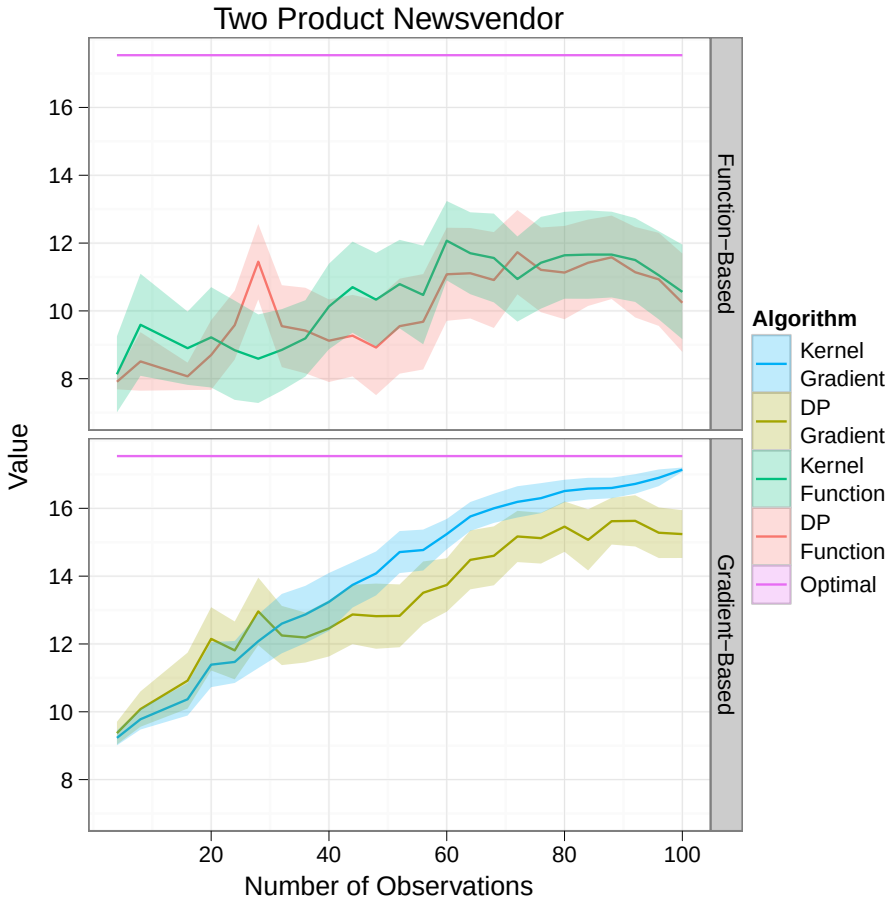

Figure 1: Gradient-based (bottom) and function-based (top) methods as a function of number of data points sampled. Results are averaged over 100 test problems with observed demand.

6.2 Hour ahead wind commitment

The details of the hour ahead wind commitment problem from Section 1 are as follows. A wind farm manager must decide how much energy to promise a utility an hour in advance, incorporating knowledge about the current state of the world. The decision is the amount of wind energy pledged, a scalar variable. If more energy is pledged than is generated, the difference must be bought on the regulating market, which is expensive with a price that is unknown when the decision is made; otherwise, the excess is lost. The goal is to maximize expected revenue. The observable state variable is the time of day, time of year, wind history from the past two hours, contract price and current regulating price,

$$
\begin{aligned}
T_i^D &= \text{time of day}, & T_i^Y &= \text{time of year},\\
P_i^R &= \text{current spot price}, & P_i^C &= \text{contract price},\\
W_{i-1} &= \text{wind speed an hour ago}, & W_i &= \text{current wind speed},\\
S_i &= \text{observable state variable} & &= (T_i^D, T_i^Y, P_i^C, P_i^S, W_i, W_{i-1}),\\
x_i &= \text{amount of energy pledged}, & Y_{i+1}(x) &= P_i^C\, x \;-\; P_{i+1}^S \max\left(x - W_{i+1}, 0\right).
\end{aligned}
$$

The revenue that the wind farm receives, $Y_{i+1}(x)$, depends on the variables P_{i+1}^S and W_{i+1}, which are not known until the next hour. We used wind speed data from the North American Land Data Assimilation System with hourly observations from 2002–2005 in the following locations:

1. *Amarillo, TX.* Latitude: 35.125 N, Longitude: 101.50 W. The data have strong daily and seasonal patterns. The mean wind level is 186.29 $(m/s)^3$ with standard deviation 244.86.

2. *Tehachapi, CA.* Latitude: 35.125 N, Longitude: 118.25 W. The data have strong seasonal patterns. The mean wind level is 89.45 $(m/s)^3$ with standard deviation 123.47.

Clean regulating and contract price data for the time period were unavailable, so contract prices were generated by Gaussian random variables with a mean of 1 and variance of 0.10. Regulating prices were generated by a mean-reverting (Ornstein-Uhlenbeck) process with a mean function that varies by time of day and time of year [51]. The data were analyzed separately for each location; they were divided by year, with one year used for training and the other three used for testing.

6.2.1 Wind Competitors

The following methods were compared on this dataset:

1. *Known wind.* The wind is known, allowing maximum possible commitment, $x_i = W_{i+1}(\omega_{i+1})$. It serves as an upper bound for all of the methods.

2. *Function-based with kernel weights.* Function-based optimization where the weights are generated by a Gaussian kernel. Bandwidth is selected according to the "rule of thumb" method of the *np* package for R,
$$h_j = 1.06\sigma_j n^{-1/(4+d)},$$
where σ_j is defined as min(sd, interquartile range/1.349) [22].

3. *Function-based with Dirichlet process weights.* Function-based optimization with Dirichlet process based weights. We model the state distribution with the following hierarchical model,
$$\begin{aligned} P &\sim DP(\alpha, \mathbb{G}_0), & \theta_i|P &\sim P, \\ T_i^D|\theta_i &\sim \text{von Mises}(\mu_{i,D}, \phi_D), & T_i^Y|\theta_i &\sim \text{von Mises}(\mu_{i,Y}, \phi_Y), \\ P_i^C|\theta_i &\sim N(\mu_{i,C}, \sigma_{i,C}^2), & P_i^R|\theta_i &\sim N(\mu_{i,R}, \sigma_{i,R}^2), \\ W_i|\theta_i &\sim N(\mu_{i,W1}, \sigma_{i,W1}^2), & W_{i-1}|\theta_i &\sim N(\mu_{i,W2}, \sigma_{i,W2}^2), \end{aligned}$$
$$\theta_i = (\mu_{i,D}, \mu_{i,Y}, \mu_{i,C}, \sigma_{i,C}^2, \mu_{i,S}, \sigma_{i,S}^2, \mu_{i,W1}, \sigma_{i,W1}^2, \mu_{i,W2}, \sigma_{i,W2}^2).$$
We modeled the time of day, T_i^D, and year, T_i^Y, with a von Mises distribution, an exponential family distribution over the unit sphere; the dispersion parameters, ϕ_D and ϕ_Y, are hyperparameters. The base measure was Normal-Inverse Gamma for P_i^C, P_i^R, W_i and W_{i-1} and uniform for the means of T_i^D and T_i^Y. 100 posterior samples were drawn using Gibbs sampling with a collapsed sampler for all conjugate dimensions after a 1,000 iteration burn-in and 10 iteration pulse between samples.

4. *Ignore state.* Sample average approximation is used,
$$\bar{F}_n(x|s) = \frac{1}{n}\sum_{i=0}^{n-1} Y_{i+1}(x).$$

Method/Location	2002	2003	2004	2005
Tehachapi, CA				
Known Wind	97.5	94.5	73.7	91.8
Function with Kernel	78.8 *(80.8%)*	77.3 *(81.8%)*	58.9 *(79.9%)*	72.1 *(78.5%)*
Function with DP	**85.1** ***(87.3%)***	**82.6** ***(87.4%)***	**63.9** ***(86.7%)***	**79.6** ***(86.7%)***
Ignore State	30.4 *(31.1%)*	31.1 *(32.9%)*	22.8 *(30.9%)*	29.3 *(31.9%)*
Amarillo, TX				
Known Wind	186.0	175.2	184.9	175.2
Function with Kernel	155.1 *(83.4%)*	149.6 *(85.4%)*	154.7 *(83.7%)*	146.2 *(83.5%)*
Function with DP	**168.2** ***(90.4%)***	**160.6** ***(91.7%)***	**167.1** ***(90.4%)***	**159.4** ***(91.0%)***
Ignore State	70.3 *(37.8%)*	68.7 *(39.2%)*	69.6 *(37.6%)*	66.1 *(37.7%)*

Table 1: Mean values of decisions by method, year and data set. Percentages of the upper bound, Known Wind, are given for the other methods.

6.2.2 Wind Results

Results are presented in Table 1. We display the value of each algorithm, along with percentages of Known Wind for the other three methods. Function-based optimization with both types of weights outperformed the algorithm in which the state variable was ignored by a large margin ($\geq$45% of the best possible value). Dirichlet process weights outperformed kernel weights by a smaller but still significant margin (5.6–8.2% of best possible value). This is because the DP weights put substantial values on many more observations than kernel weights did in areas with high wind; in effect, kernel weights simply used the one or two closest observations in these areas for prediction.

7 Discussion

We presented new model-free methods to solve stochastic search problems with an observable state variable, function-based optimization and gradient-based optimization. We provided conditions for convergence for each. Both algorithms rely on weighting observations; we gave an easily implementable weighting function (kernels) and a more complex weighting function (Dirichlet process based). Empirical analysis shows that Dirichlet process weights add value when the state variable distribution is moderate to high dimensional or has super-Gaussian tails; this was the case in the wind commitment problem. Our results also show that function-based optimization may be unstable in situations with high noise and more complex functional representations; in these cases, gradient-based methods may provide superior results.

More generally, this work shows that statistics and machine learning can provide solutions to currently intractable search and optimization problems. Traditional search and optimization methods are designed to handle problems with large, complex decision spaces. However, there are many problems where complexity is derived from elements aside from the decision, such as an observable state variable. Statistics and machine learning offer an array of tools, such as clustering, density estimation, regression and inference, that may prove useful in solving problems that are now currently avoided.

References

[1] C. E. Antoniak, *Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems*, The Annals of Statistics, 2 (1974), pp. 1152–1174.

[2] A. Barron, M. J. Schervish, and L. Wasserman, *The consistency of posterior distributions in nonparametric problems*, The Annals of Statistics, 27 (1999), pp. 536–561.

[3] A. Bemporad and C. Filippi, *An algorithm for approximate multiparametric convex programming*, Computational Optimization and Applications, 35 (2006), pp. 87–108.

[4] D. P. Bertsekas, *Dynamic programming and optimal control*, vol. 2, Athena Scientific, 2007.

[5] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming (Optimization and Neural Computation Series, 3)*, 1996.

[6] J. R. Birge and F. Louveaux, *Introduction to stochastic programming*, Springer Verlag, 1997.

[7] D. Blackwell and J. B. MacQueen, *Ferguson distributions via Polya urn schemes*, The Annals Statistics, 1 (1973), pp. 353–355.

[8] R. K. Cheung and W. B. Powell, *SHAPE-A stochastic hybrid approximation procedure for two-stage stochastic programs*, Operations Research, 48 (2000), pp. 73–79.

[9] T. M. Cover and E. Ordentlich, *Universal portfolios with side information*, IEEE Transactions on Information Theory, 42 (1996), pp. 348–361.

[10] H. Daumé and D. Marcu, *A Bayesian model for supervised clustering with the Dirichlet process prior*, The Journal of Machine Learning Research, 6 (2005), pp. 1551–1577.

[11] M. D. Escobar and M. West, *Bayesian Density Estimation and Inference Using Mixtures.*, Journal of the American Statistical Association, 90 (1995), pp. 577–588.

[12] J. Fan and I. Gijbels, *Data-driven bandwidth selection in local polynomial fitting: variable bandwidth and spatial adaptation*, Journal of the Royal Statistical Society. Series B (Methodological), 57 (1995), pp. 371–394.

[13] T. S. Ferguson, *A Bayesian analysis of some nonparametric problems*, The Annals of Statistics, 1 (1973), pp. 209–230.

[14] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin, *Bayesian data analysis*, CRC press, 2004.

[15] S. Ghosal, J. K. Ghosh, and R. V. Ramamoorthi, *Posterior consistency of Dirichlet mixtures in density estimation*, The Annals of Statistics, 27 (1999), pp. 143–158.

[16] J. K. Ghosh and R. V. Ramamoorthi, *Bayesian Nonparametrics*, Springer, 2003.

[17] P. Glasserman, *Gradient estimation via perturbation analysis*, Springer, 1990.

[18] G. A. Godfrey and W. B. Powell, *An adaptive, distribution-free algorithm for the newsvendor problem with censored demands, with applications to inventory and distribution*, Management Science, 47 (2001), pp. 1101–1112.

[19] A. Goldenshluger and A. Zeevi, *Woodroofes one-armed bandit problem revisited*, The Annals of Applied Probability, 19 (2009), pp. 1603–1633.

[20] G. Gürkan, A. Yonca Özge, and S. M. Robinson, *Sample-path solution of stochastic variational inequalities*, Mathematical Programming, 84 (1999), pp. 313–333.

[21] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*, Springer, 2009.

[22] T. Hayfield and J. S. Racine, *Nonparametric econometrics: The np package*, Journal of Statistical Software, 27 (2008), pp. 1–32.

[23] E. Hazan and N. Megiddo, *Online learning with prior knowledge*, Lecture Notes in Computer Science, Springer, 2007, pp. 499–513.

[24] K. Healy and L. W. Schruben, *Retrospective simulation response optimization*, in Proceedings of the 1991 Winter Simulation Conference, 1991, 1991, pp. 901–906.

[25] D. P. Helmbold, R. E. Schapire, Y. Singer, and M. K. Warmuth, *On-line portfolio selection using multiplicative updates*, Mathematical Finance, 8 (1998), pp. 325–347.

[26] Y. C. Ho, *On the perturbation analysis of discrete-event dynamic systems*, Journal of Optimization Theory and Applications, 46 (1985), pp. 535–545.

[27] H. Ishwaran and L. F. James, *Generalized weighted Chinese restaurant processes for species sampling mixture models*, Statistica Sinica, 13 (2003), pp. 1211–1236.

[28] M. C. Jones, J. S. Marron, and S. J. Sheather, *A brief survey of bandwidth selection for density estimation.*, Journal of the American Statistical Association, 91 (1996).

[29] H. T. Jongen and G. W. Weber, *On parametric nonlinear programming*, Annals of Operations Research, 27 (1990), pp. 253–283.

[30] J. Kiefer and J. Wolfowitz, *Stochastic estimation of the maximum of a regression function*, The Annals of Mathematical Statistics, 23 (1952), pp. 462–466.

[31] A. J. Kleywegt, A. Shapiro, and T. Homem-de Mello, *The sample average approximation method for stochastic discrete optimization*, SIAM Journal on Optimization, 12 (2002), pp. 479–502.

[32] S. N. MacEachern, *Estimating normal means with a conjugate style Dirichlet process prior*, Communications in Statistics-Simulation and Computation, 23 (1994), pp. 727–741.

[33] S. N. MacEachern and P. Müller, *Estimating mixtures of Dirichlet process models*, Journal of Computational and Graphical Statistics, (1998), pp. 223–238.

[34] M. Medvedovic and S. Sivaganesan, *Bayesian infinite mixture model based clustering of gene expression profiles*, Bioinformatics, 18 (2002), pp. 1194–1206.

[35] E. A. Nadaraya, *On estimating regression*, Theory of Probability and its Applications, 9 (1964), pp. 141–142.

[36] R. M. Neal, *Markov chain sampling methods for Dirichlet process mixture models*, Journal of Computational and Graphical Statistics, 9 (2000), pp. 249–265.

[37] N. C. Petruzzi and M. Dada, *Pricing and the newsvendor problem: A review with extensions*, Operations Research, 47 (1999), pp. 183–194.

[38] J. Pitman, *Some developments of the Blackwell-MacQueen urn scheme*, Lecture Notes-Monograph Series, 30 (1996), pp. 245–267.

[39] E. L. Plambeck, B. R. Fu, S. M. Robinson, and R. Suri, *Sample-path optimization of convex stochastic performance functions*, Mathematical Programming, 75 (1996), pp. 137–176.

[40] W. B. Powell, *Approximate Dynamic Programming: Solving the curses of dimensionality*, Wiley-Blackwell, 2007.

[41] W. B. Powell, A. Ruszczyński, and H. Topaloglu, *Learning algorithms for separable approximations of discrete stochastic optimization problems*, Mathematics of Operations Research, 29 (2004), pp. 814–836.

[42] M. L. Puterman, *Markov decision processes: Discrete stochastic dynamic programming*, John Wiley & Sons, Inc. New York, NY, USA, 1994.

[43] D. Ralph and S. Dempe, *Directional derivatives of the solution of a parametric nonlinear program*, Mathematical Programming, 70 (1995), pp. 159–172.

[44] P. Rigollet and A. Zeevi, *Nonparametric Bandits with Covariates*, Arxiv preprint 1003.1630, (2010).

[45] H. Robbins and S. Monro, *A stochastic approximation method*, The Annals of Mathematical Statistics, 22 (1951), pp. 400–407.

[46] S. M. Robinson, *Analysis of sample-path optimization*, Mathematics of Operations Research, 21 (1996), pp. 513–528.

[47] A. Rodriguez, D. B. Dunson, and A. E. Gelfand, *Bayesian nonparametric functional data analysis through density estimation*, Biometrika, 96 (2009), pp. 149–162.

[48] R. Y. Rubinstein and B. Melamed, *Modern simulation and modeling*, Wiley, 1998.

[49] R. Y. Rubinstein and A. Shapiro, *Discrete event systems: Sensitivity analysis and stochastic optimization by the score function method*, John Wiley & Sons Inc, 1993.

[50] J. Sarkar, *One-armed bandit problems with covariates*, The Annals of Statistics, 19 (1991), pp. 1978–2002.

[51] E. S. Schwartz, *The stochastic behavior of commodity prices: Implications for valuation and hedging*, The Journal of Finance, 52 (1997), pp. 923–973.

[52] B. Shahbaba and R. M. Neal, *Nonlinear Models Using Dirichlet Process Mixtures*, Journal of Machine Learning Research, 10 (2009), pp. 1829–1850.

[53] A. Shapiro, *Asymptotic analysis of stochastic programs*, Annals of Operations Research, 30 (1991), pp. 169–186.

[54] A. Shapiro, T. Homem-de Mello, and J. Kim, *Conditioning of convex piecewise linear stochastic programs*, Mathematical Programming, 94 (2002), pp. 1–19.

[55] A. Shapiro and Y. Wardi, *Convergence analysis of stochastic algorithms*, Mathematics of Operations Research, 21 (1996), pp. 615–628.

[56] P. Sørensen, A. D. Hansen, and P. A. C. Rosas, *Wind models for simulation of power fluctuations from wind farms*, Journal of Wind Engineering and Industrial Aerodynamics, 90 (2002), pp. 1381–1402.

[57] J. C. Spall, *Introduction to stochastic search and optimization: estimation, simulation, and control*, John Wiley and Sons, 2003.

[58] R. S. Sutton and A. G. Barto, *Introduction to reinforcement learning*, MIT Press Cambridge, MA, USA, 1998.

[59] S. Tokdar, *Posterior consistency of Dirichlet location-scale mixture of normals in density estimation and regression*, Sankhya: The Indian Journal of Statistics, 67 (2006), pp. 90–110.

[60] J. N. Tsitsiklis and B. Van Roy, *Regression methods for pricing complex American-style options*, IEEE Transactions on Neural Networks, 12 (2001), pp. 694–703.

[61] S. G. Walker, *New approaches to Bayesian consistency*, The Annals of Statistics, 32 (2004), pp. 2028–2043.

[62] C. C. Wang, S. R. Kulkarni, and H. V. Poor, *Arbitrary side observations in bandit problems*, Advances in Applied Mathematics, 34 (2005), pp. 903–938.

[63] G. S. Watson, *Smooth regression analysis*, Sankhyā: The Indian Journal of Statistics, Series A, 26 (1964), pp. 359–372.

[64] M. Woodroofe, *A one-armed bandit problem with a concomitant variable*, Journal of the American Statistical Association, 74 (1979), pp. 799–806.

[65] Y. Yang and D. Zhu, *Randomized allocation with nonparametric estimation for a multi-armed bandit problem with covariates*, Annals of Statistics, 30 (2002), pp. 100–121.