Abstract.
Stochastic search involves finding a set of controllable parameters that minimizes an unknown objective function using a set of noisy observations. We consider the case when the unknown function is convex and a metamodel is used as a surrogate objective function. Often the data are non-i.i.d. and include a observable state variable, such as applicant information in a loan rate decision problem. State information is difficult to incorporate into convex models. We propose a new semi-convex regression method that is used to produce a convex metamodel in the presence of a state variable. We show consistency for this method. We demonstrate its effectiveness for metamodeling on a set of synthetic inventory management problems and a large, real-life auto loan dataset.

Key words. asymptotic properties; machine learning; multivariate convex functions; metamodeling; nonparametric regression; simulation optimization

AMS subject classifications. Find AMS codes

1. Introduction. Stochastic search seeks to find a set of controllable parameters \( x \in \mathcal{X} \) that minimize \( J(x) \), where the function \( J(x) \) is unknown and only available through simulation or observational data. Metamodels use a set of observed data to approximate \( J \) over the entire space \( \mathcal{X} \), usually through regression.

When metamodeling is used to approximate objective functions, it faces particular challenges. First, it may be hard to find the minimum of a metamodel estimator, \( \bar{J} \). Search is feasible for large decision spaces if \( J \) is convex and the metamodel preserves that property. Second, the data used to approximate \( J \) may not be i.i.d. Often, if the data are observational or generated by a sequential decision process, they may contain an observable state variable. For example, a loan provider may want to determine the interest rate it should offer to a specific applicant based on a large database of previous loans and information about the current applicant, like credit history, loan size, loan type, etc. In this case, the observable state variable would be the information about the current applicant. Some of the previous data come from applicants similar to the current one, some does not.

We examine search problems that are convex in a decision parameter \( x \) and include information on an observable state variable \( s \),

\[
\min_{x \in \mathcal{X}} J(x | S = s) = \min_{x \in \mathcal{X}} \mathbb{E}[F(x, s, Z)].
\]

Here \( x \in \mathbb{R}^p \) is the controllable parameter, \( \mathcal{X} \) is a decision set, \( S \in \mathcal{S} \) is a state variable, \( Z : \Omega \to \Psi \) is a random outcome that maps from a sample space \( \Omega \) to an outcome space \( \Psi \), and \( F : \mathbb{R}^p \times \mathcal{S} \times \Psi \to \mathbb{R} \) is an objective function that is convex in \( x \) for almost every \( z \). In this paper, we introduce semi-convex regression to approximate
$E[F(x, s, Z)]$ from a set of $n$ observations, $\{(x_i, s_i, y_i)\}_{i=1}^n$ where $y_i = F(x_i, s_i, z_i)$. By semi-convex regression, we mean regression subject to the constraint that an estimator $\bar{F}(x, s)$ is convex only in $x$ and unconstrained in $s$. Along with producing a function that is easy to search, shape constraints like convexity and semi-convexity act as a regularizer on the estimator: they reduce the size of the model space and often result in a higher quality output than unconstrained models.

Regression subject to convexity constraints, called “convex regression,” has been studied since the 1950’s [39]. There are a variety of computationally efficient, theoretically sound univariate methods [22, 11, 65]. Multivariate methods have proven more difficult. Traditional methods rely on fitting a least squares objective function subject to first or second order convexity constraints [40, 42, 2, 3, 62, 45]; these methods only scale to a few thousand observations due to the number of constraints. Recently, more computationally efficient methods have been developed that are based on fitting a function that is the maximum over a set of hyperplanes. The hyperplanes can be found through the convex hull of a smoothed estimate [1] or adaptive partitioning [46, 33]. None of these methods, however, can be easily extended to semi-convex settings.

A natural way to include a state variable with convex regression is through discretizing the state. For example, we might divide all of the borrowers into groups with “good credit history” and “bad credit history.” Then we would fit a convex regression within each group. However, credit history is usually described through a FICO score, which ranges from 300 to 850. Most datasets do not have enough data to make an accurate prediction if divided into individual FICO scores. Moreover, applicants are usually described by more than a FICO score; each additional attribute makes the amount of data required for prediction grow exponentially.

To address this problem, we combine kernel density estimation with the Convex Adaptive Partitioning [33] method for convex regression to produce a new semi-convex regression estimator, called Kernelized Convex Adaptive Partitioning (K-CAP). Like convex regression methods, the estimator is the maximum over a set of hyperplanes in the convex dimensions. Unlike convex regression, the hyperplane parameters vary with the state variable. This requires significant modifications to the underlying adaptive partitioning methods. We show consistency for K-CAP. We demonstrate its effectiveness on a set of synthetic convex optimization problems with observable state variables and application to a loan dataset where offered rates are determined by applicant and exogenous state variables.

The contributions of this paper are 1) a novel nonparametric regression method for semi-convex functions and consistency conditions, 2) application of semi-convex regression to metamodeling for stochastic search problems with an observable state variable, and 3) an empirical demonstration of the efficacy of these methods. The rest of the paper is organized as follows. Section 2 is a literature review; Section 3 presents and analyzes a new method for nonparametric regression in semi-convex settings, K-CAP; Section 4 demonstrates an application of those methods to storage and adaptive pricing problems; Section 5 gives conclusions and future research directions.

2. Related Literature. In this section we review the literature on metamodeling and convex regression.

2.1. Metamodeling. Metamodeling is found in a variety of fields. Response surfaces are metamodels fit in a traditionally online manner using regression methods such as second order parametric models [12], radial basis functions [64], neural networks [26] or stochastic kriging [5]; see [6] for a review. In general, these meth-
ods are not shape restricted and cannot be easily searched. However, quadratic local approximations are easily searched, but they only provide local estimates for optima.

In computer science, metamodels are often used for contextual bandits and reinforcement learning. Contextual bandits are a variant of the traditional bandit problem, where an observable “covariate” or state variable has been added to each bandit. These problems are generally solved through estimating reward functions (or metamodels) for each bandit-covariate pair. Estimation methods include linear combinations of basis functions [44], discretization the state space [56], random histograms, nearest neighbors [76] or adaptive partitioning [66]. Reward functions are fit without shape restrictions because decisions are selected from a finite set of actions.

Reinforcement learning is the study of sequential decision problems. We might be in state $S$ and take an action $a$, which influences the next state that we visit, $S'$. The choice of the action $a$, needs to consider the expected value of being in state $S'$. A host of algorithms have been proposed to solve this problem class (see [55, 8, 67, 70, 52, 7, 68, 17, 54], or [7]), where the major complication is that we do not know and have to approximate the value of being in state $S'$. Our methods are closest to $Q$-learning, which estimates the value of a $Q$-function, $Q(S,a)$, or $E[F(x,s,Z)]$ in our notation. In reinforcement learning, most action spaces are finite and small, so little attention has been given to maintaining convexity of $Q(S,a)$ in $a$ when the action space is continuous, vector-valued and convex. However, vector-valued action spaces can be handled by retaining convexity along with the use of the post-decision state variable, as illustrated in [53].

The stochastic programming community solves sequential decision problems in somewhat different manner than reinforcement learning. Instead of generating solutions from a set of state-action-reward triplets, a scenario tree is generated to represent possible exogenous outcomes and observable states. Geometrically, our proposed methods are similar to cutting plane methods, like L-shaped decomposition [72, 74, 38] and stochastic subgradient methods [58], which fit a convex expected value function through the maximum of a set of hyperplanes. See [59] for an overview of decomposition methods for stochastic programming. Unlike other metamodel methods, cutting plane methods are explicitly convex in the decision variable. However, it is difficult to incorporate an observable state variable into cutting plane methods, which is usually done by producing a set of value functions that are indexed by the nodes of the scenario tree, effectively producing a discretization of the state space [9, 10].

2.2. Convex Regression. Convex regression has been well studied in a univariate setting, where convexity is equivalent to an increasing derivative function. Methods include the least squares estimator [39, 22, 34, 47, 29, 24, 27], order restricted splines [71, 48, 49, 65], constrained kernel regression [11], and Bernstein polynomials [18]. Most of these methods are computationally efficient for large problem sizes.

Multivariate convex regression is more difficult: the increasing derivative condition in a univariate setting turns into a set of circular conditions in multiple dimensions. However, it has received renewed attention in recent years. The first proposed method minimized a least squares objective function while constraining the gradients of the estimator under a first order condition [39, 40]; only recently have characterization and consistency of this estimator been studied [42, 62, 45]. This method is computationally difficult for more than a few thousand observations because the estimator is found by solving a quadratic program with $n(n - 1)$ constraints; an entropy minimization method of [4] has similar computational complexity. Another
approach has been to fit a function with a positive semidefinite Hessian at a set of fixed points through semidefinite programming [57, 2, 3, 37]. Although this method is consistent, semidefinite programming also does not scale to more than a few thousand observations. [1] proposed smoothing the data and then using the convex hull of the smoothed data as an estimator. This method preserves the asymptotic properties of the smoother and scales to larger $n$ but is computationally infeasible for more than 3 or 4 dimensions.

Recently, sparse piecewise linear methods have been introduced [46, 33]. These models are defined by a set of $K << n$ hyperplanes, $(\alpha_k, \beta_k)_{k=1}^K$, where the estimator is produced by taking the maximum over the set of hyperplanes,

$$\bar{F}(x) = \max_{k=1, \ldots, K} \alpha_k + \beta_k^T x.$$  

Usually, the hyperplanes are generated by partitioning the data into $K$ subsets and then fitting a hyperplane within each subset. These models are computationally efficient—the most demanding steps involve fitting linear models—and still produce a low error estimator. The estimator in [46] is created by using a Gauss-Newton method to iteratively fit linear models to a fixed number of subsets. While this algorithm generally produces a low error estimator, it is sensitive to the number of subsets, the initial partition, and may not even converge. The Convex Adaptive Partitioning (CAP) estimator of [33] finds a partition through tree-based methods, producing a consistent algorithm.

3. Semi-Convex Regression. To fit the metamodel, we use $n$ observations, $(x_i, s_i, y_i)_{i=1}^n$, where $y_i = F(x_i, s_i, z_i)$, to approximate the objective function, $\bar{F}_n(x|s) \approx \mathbb{E}[F(x, s, Z)]$ in a manner that maintains convexity in $x$ for each $s \in S$,

$$\bar{F}_n(\lambda x_1 + (1 - \lambda)x_2 | s) \leq \lambda \bar{F}_n(x_1 | s) + (1 - \lambda) \bar{F}_n(x_2 | s), \quad (3.1)$$

$\forall \lambda \in (0, 1), \forall x_1, x_2 \in X$. That is, the function is convex in the decision dimension $x$ for a given non-convex dimension (state) $s$. We define a function to be semi-convex if it satisfies the requirements in Equation (3.1). In the loan example, we might expect the log probability of loan acceptance to be concave based on the interest rate offered, making this the convex decision dimension. However, that acceptance probability also depends upon applicant FICO score, loan term, and current prime interest rate. The decision maker has no control over these variables and there is no shape restriction expected in these dimensions, so we call them the set of “non-convex dimensions.”

Semi-convex models need to fit a convex model across a subset of the space and an unconstrained model across the rest. Sparse, piecewise linear models are the only convex regression methods that scale to settings with more than a few thousand observations. In their existing form, they cannot be forced to include non-convex dimensions. Non-convex dimensions can be introduced in a number of ways, including partitioning or through state-dependent coefficients. In a partitioned model, the data are divided according to the state variable and a convex model is fit within each subset. This is potentially computationally expensive and reduces the amount of data available to fit each function. Alternatively, we can fit a model where the coefficients depend on the state,

$$\bar{F}(x | s) = \max_{k=1, \ldots, K(s)} \alpha_k(s) + \beta_k(s)^T x. \quad (3.2)$$

Convex models fit across a partitioned state variable are similar to stochastic programming solution methods, where a subset would represent an entire scenario, or
history. However, instead of generating estimates scenario by scenario, we interpolate across scenarios, or state, as well.

We propose creating a state-dependent model by weighting the observations according to the state, and then fitting an adaptive partitioning-based convex regression model. If we have \( K \) subsets, we would like to (approximately) solve

\[
\alpha(s), \beta(s) \in \arg \min_{\alpha,\beta} \sum_{i=1}^{n} w_n(s, s_i) \left(y_i - \max_{k=1,\ldots,K} \alpha_k + \beta_k^T x_i\right)^2.
\]

The state-dependent weights, \( \{w_n(s, s_i)\}_{i=1}^{n} \), are generated through kernel functions. In a setting where \( K \) is fixed and Equation (3.3) can be solved optimally, this produces a class of models where \( \alpha(s) \) and \( \beta(s) \) are smooth varying functions of \( s \), like varying coefficient models [35, 41]. However, solving Equation (3.3) to optimality is infeasible for \( K > 1 \). A good solution can be generated by splitting and modifying existing subsets, but the result is usually not smoothly varying.

To find a good approximate solution to Equation (3.3), we modify the Convex Adaptive Partitioning algorithm of [33] to accommodate weighted observations, resulting in Kernelized Convex Adaptive Partitioning (K-CAP). The subset splitting methods remain much the same between the two methods, but heavy modification is required in acceptable subset size, selection of partition size and the proof of consistency. We first discuss kernel weights and then K-CAP.

### 3.1. Kernel Weights

Kernel weights approximate a conditional density using kernel functions, \( K(s) \). Common choices are the Gaussian kernel,

\[
K_h(s) = \prod_{j=1}^{d} \left( \frac{1}{\sqrt{2\pi}} \right) \exp\left\{ -\frac{s_j^2}{2h_j^2} \right\},
\]

or a fixed window kernel,

\[
K_h(s) = \begin{cases} 1 & \text{if } s \in \left[ -\frac{h_1}{2}, \frac{h_1}{2} \right] \times \cdots \times \left[ -\frac{h_d}{2}, \frac{h_d}{2} \right], \\ 0 & \text{otherwise} \end{cases}
\]

where the vector \( h = (h_1, \ldots, h_d) \) is called the bandwidth. Kernel weights have the advantage of being simple and easy to implement, but are problematic in moderate to high dimensions, \( (d \geq 5) \), due to the curse of dimensionality. In higher dimensions, bandwidths need to cover most of the space in every dimension to incorporate a reasonable proportion of the observations. For example, if observations are sampled uniformly in \([0, 1]^{10}\), the subspace \([0.25, 0.75]^{10}\) would contain only about 0.1% of the samples despite having a bandwidth that is half of every dimension. However, there are a variety of methods to adaptively select bandwidths in higher dimensions [61, 43], with varying degrees of computational complexity.

The least complicated weighting scheme is based on the Nadaraya-Watson estimator [50, 73]. If \( K(s) \) is the kernel and \( h_n \) is the bandwidth after \( n \) observations, define

\[
w_n(s, s_i) = \frac{K_{h_n}(s - s_i)}{\sum_{j=1}^{n} K_{h_n}(s - s_j)}.
\]

Unlike higher order locally polynomial methods [25], the Nadaraya-Watson kernel estimator is guaranteed to produce positive weights.
Under the standard sampling and bandwidth assumptions, \( h_n \) satisfies
\[
n^{1/2} \prod_{j=1}^{d} h_{n,j} \to \infty
\]
and \( h_{n,1}, \ldots, h_{n,d} \to 0 \) as \( n \to \infty \), with \( b_1 \leq \min_{i,j} \frac{h_{n,i}}{n_{n,j}} \leq \max_{i,j} \frac{h_{n,i}}{n_{n,j}} \leq b_2 \) for some \( 0 < b_1 \leq b_2 < \infty \) for all \( n \), it is trivial to extend the consistency results for sample average approximation summarized in [63] to the case with an observable state variable using kernel arguments.

### 3.2. Kernelized Convex Adaptive Partitioning.

The K-CAP algorithm works in ways similar to typical tree regression models like Classification and Regression Trees (CART) [15], piecewise polynomial trees [19] and other variants [51, 31]. The inputs are a query state \( s \) and data \((x_i, s_i, y_i)_{i=1}^n\). Given these, a model is fit in the following manner.

First, weights for each of the observations are generated with kernels, \((w_n(s, s_i))_{i=1}^n\) as \( w_n(s, s_i) = K_h(s - s_i) \). Then the data are adaptively partitioned, starting with all of the data in a single subset. Let \( C_{K(s)} = \{C_1(s), \ldots, C_{K(s)}(s)\} \) be a partition of the observations, with \( C_k(s) \subset \{1, \ldots, n\} \) called a subset. Let \( A_{K(s)} = \{A_1(s), \ldots, A_{K(s)}(s)\} \) be the corresponding partition of the covariate space: \( i \in C_k(s) \Rightarrow x_i \in A_k(s) \). Given \( C_{K(s)} \), a weighted linear model is fit to the data within each subset,

\[
(\alpha_k(s), \beta_k(s)) \in \arg \min_{\alpha, \beta} \sum_{i \in C_k(s)} w_n(s, s_i) (y_i - \alpha - \beta^T x_i)^2.
\]

A convex function is generated by taking the maximum over the linear models. The new convex function also induces a partition of the decision space, \( A'_{K(s)} = \{A'_1(s), \ldots, A'_{K(s)}(s)\} \), where each subset \( A'_k(s) \) is the region dominated by the hyperplane \((\alpha_k(s), \beta_k(s))\),

\[
A'_k(s) = \{x : \alpha_k(s) + \beta_k(s)^T x \geq \alpha_j(s) + \beta_j(s)^T x \text{ for all } j \neq k\}.
\]

Let \( C'_k(s) \) denote the partition of the indices associated with \( A'_k(s) \). The observation partition–decision space partition–hyperplane triplet defines a model for \( K(s) \) subsets, \( M_{K(s)}(s) \).

To refine the partition, we seek to add one subset in a way that minimizes the weighted sum of square errors,

\[
\alpha'(s), \beta'(s) \in \arg \min_{\alpha, \beta} \sum_{i=1}^n w_n(s, s_i) \left( y_i - \max_{k=1, \ldots, K(s)+1} \alpha_k + \beta_k^T x_i \right)^2.
\]

Fully solving Equation (3.4) is infeasible, but a good solution can be generated by splitting and modifying existing subsets.

We refine the partition in a two step process: selecting a subset split from a collection of candidate splits, and then a partition refit that aligns \( C_{K(s)}(s) \) with \( A'_{K(s)}(s) \). Candidate splits are generated by splitting the data within a subset and fitting new linear models to each part of the split; this is done for each subset. Since the function in Equation (3.4) is max over a set of hyperplanes, the best split for a subset cannot be analytically determined (unlike [15] and other models). Instead, we make a set
of candidate splits for each subset, first fixing a dimension, then ordering the data according to that dimension, and finally splitting the data based on quantiles in that dimension (defined by a set of \( L \) knots). For subset \( k \), dimension \( j \) and knot \( a_\ell \) determined by quantile \( \ell \), we create two new subsets,

\[
\hat{C}_{k,j,\ell,-}(s) = \{ i : i \in C_k(s), x_{i,j} \leq a_\ell \}, \quad \hat{C}_{k,j,\ell,+}(s) = \{ i : i \in C_k(s), x_{i,j} > a_\ell \}.
\]

Weighted linear models are fit to the observations within these new subsets. This, along with the new decision space partition, define a candidate model. The candidate model with the lowest weighted squared error is selected.

Similarly refining the partition in this manner produces poor models because the observations are not always represented by the hyperplane they generated. That is, there exist many \( i \) where \( i \in C_k(s) \), but \( i \notin C_k'(s) \). These misrepresentations can be reduced by allowing \( C_K(s) \) to redefine \( C_K(s) \) through a partition refit. Set \( C_k(s) = C_k'(s) \) for \( k = 1, \ldots, K \), and use \( C_K(s) \) to define a new model. This new model is accepted if certain criteria are met, such as a minimal number of observations within each subset, \( n_{\text{min}} \) and a non-expansion of the maximal subset diameter. This technique is used in [46], except that K-CAP does it only once to reduce computational costs and avoid cycling.

The partition is refined until no subset can be split without having fewer than \( n_{\text{min}} \) observations. Once the terminal point is reached with \( K(s) \) subsets, all of the accepted models, \( M_1(s), \ldots, M_{K(s)}(s) \), are compared. Models with larger numbers of subsets may generalize poorly due to overfitting.

We set the minimal number of observations as \( n_{\text{min}} = \max_k D K_k \frac{n}{\log(n)} \), where \( K_k = \frac{1}{n_k} \sum_{i \in C_k(s)} \left( \frac{1}{n_k} K_h(s - s_i) \right)^2 \); here \( D \) is left as a tunable parameter. This value produces \( O(\log n) \) subsets. It is larger than that needed to maintain consistency of the estimator, but smaller subsets sizes produce poorer quality estimators and increase computational time.

K-CAP has two tunable parameters: \( D \) and \( L \). While larger values of each result in a larger model selection space and lower model error, they add computational complexity. We have found \( D = 3 \) and \( L = 10 \) to be a good tradeoff between accuracy and computational complexity. See [33] for a more thorough discussion of sensitivity to these parameters. We now briefly summarize the K-CAP algorithm below.

**Kernelized Convex Adaptive Partitioning (K-CAP)**

1. **Weight data.** Set weights \( \left( w_n(s, s_i) \right)_{i=1}^n \) as \( w_n(s, s_i) = K_h(s - s_i) \).
2. **Initialize partition.** Set \( K(s) = 1 \); place all observations into a single observation subset, \( C_1(s) = \{1, \ldots, n\} \); \( A_1(s) = \mathcal{X} \); this defines model \( M_1(s) \).
3. **Split.** Refine partition by splitting a subset.
   a. **Generate candidate splits.** Generate candidate model \( \hat{M}_{k,j}l(s) \) by 1) fixing a subset \( k \), 2) fixing a decision dimension \( j \), 3) dividing the data in subset \( k \) and dimensions \( j \) according to knot \( a_\ell \). Find the hyperplane parameters \( (\alpha_{k,j}^l(s), \beta_{k,j}^l(s)) \) \( l=1 \) by

\[
(\alpha_{k,j}^l(s), \beta_{k,j}^l(s)) \in \arg\min_{\alpha, \beta} \sum_{i \in C_{k,j}(s)} w_n(s, s_i) \left( y_i - \alpha - \beta^T x_i \right)^2.
\]

Models are created for \( L \) knots, all \( p \) dimensions and \( K(s) \) subsets.
   b. **Select split.** Choose the model \( M_{K(s)+1}(s) \) from the candidates that minimizes global mean squared error on the training set and satisfies \( \min_k |C_k(s)| \geq n_{\text{min}} \). Set \( K(s) = K(s) + 1 \).
4. **Refit.** Use the partition induced by the hyperplanes to generate model $M_k(s)$. Set $M_{K(s)}(s) = M_{K(s)}(s)$ if for every subset $C_k(s)$ in $M_k(s)$, $|C_k(s)| \geq n_{\text{min}}$.

5. **Stopping conditions.** If for every subset $C_k(s)$ in $M_k(s)$, $|C_k(s)| < 2n_{\text{min}}$, stop fitting and proceed to step 6. Otherwise, go to step 3.

6. **Select model size.** Each model $M_k(s)$ creates an estimator,

$$F_{nk}(x | s) = \max_{j=1, \ldots, k} \alpha_j(s) + \beta_j(s)^T x.$$  

Use generalized cross-validation to select final model $M^*(s)$ from $\{M_k(s)\}_{k=1}^K$.

### 3.3. Asymptotic Properties of K-CAP.**

We give conditions for consistency of the K-CAP estimator in both $L_\infty$ distance from the mean function and distance between the approximated optimal value and the true optimal value. Since K-CAP is an adaptive partitioning method, consistency can be shown by combining proof methods from other adaptive partitioning models [15, 19, 51, 31, 33] with methods from varying-coefficient linear models [41]. To show $L_\infty$ consistency, we take a two-step approach by first showing consistency for the mean function and first derivatives of the model created when linear models are fit to the data within each subset, and then using those results to show that the estimator created by taking the maximum across those hyperplanes is also consistent. Let $M^*_n(s)$ be the un-pruned K-CAP model after $n$ observations. Define the piecewise-linear, discontinuous estimator based on $M^*_n(s)$ and the K-CAP estimator as, respectively,

$$F^*_n(x | s) = \sum_{k=1}^{K_n(s)} (\alpha_k(s) + \beta_k(s)^T x) 1_{\{x \in A_k(s)\}},$$  

$$\bar{F}_n(x | s) = \max_{k=1, \ldots, K_n(s)} \alpha_k(s) + \beta_k(s)^T x.$$  

Here $K_n(s)$ is the partition size, $A_1(s), \ldots, A_{K_n(s)}(s)$ are the covariate partitions and $(\alpha_k(s), \beta_k(s))_{k=1}^{K_n(s)}$ are the hyperplanes associated with $M^*_n(s)$. Each subset $A_k(s)$ has an associated diameter, $d_{nk}(s) = \max_{x_1, x_2 \in A_k(s)} ||x_1 - x_2||_2$. Define the empirical covariate mean for subset $k$ as $\bar{x}_k(s) = \frac{1}{|C_k(s)|} \sum_{i \in C_k(s)} x_i$. For $x_i \in A_k(s)$, define

$$\Gamma_i = \begin{bmatrix} [1, \ldots, 1] \\ d_{nk}(s)^{-1} (x_i - \bar{x}_k(s)) \end{bmatrix}, \quad G_k(s) = \sum_{i \in C_k(s)} \frac{1}{h^2} K_h(s - s_i) \Gamma_i \Gamma_i^T.$$  

Note that $(\alpha_k(s), \beta_k(s)) = G_k(s)^{-1} \sum_{i \in C_k} K_h(s - s_i) \Gamma_i y_i$ whenever $G_k(s)$ is nonsingular.

Let $x_1, \ldots, x_n$ be i.i.d. random variables and let $F(x | s)$ be the true mean function. We make the following assumptions:

- **A1.** $X \subset \mathbb{R}^p$ is convex and compact, $S \subset \mathbb{R}^d$ is compact and $F(x | s)$ is Lipschitz continuous on $X \times S$ and continuously twice differentiable on $X$ with Lipschitz parameter $\zeta$.

- **A2.** There is an $a > 0$ such that $\psi(a | x, s) = \mathbb{E} [e^{a|Y - F(x | s)|} | X = x, S = s]$ is bounded on $X$.

- **A3.** Let $\lambda_k$ be the smallest eigenvalue of $|C_k(s)|^{-1} \sum_{i \in C_k(s)} \Gamma_i \Gamma_i^T$ and $\lambda_n = \min_k \lambda_k$. Then $\lambda_n$ remains bounded away from 0 in probability as $n \to \infty$.

- **A4.** The diameter of the partition $\max_k d_{nk}(s) \to 0$ in probability as $n \to \infty$.  

A5. There exists an $m_n \to \infty$ such that the number of observations in each subset satisfies $n_k \geq m_n \min_{i=1, \ldots, K_n(s)} d_{nk}^{-1}(s) (K^*)^{1/2} n^{1/2} \log(n)$ in probability as $n \to \infty$, where $K^* = \frac{1}{n} \sum_{i \in C_k(s)} \left( \frac{1}{n} K_h(s - s_i) \right)^2$.

A6. The bandwidth $h$ satisfies $n^{1/2} h^d \to \infty$ and $h \to 0$ as $n \to \infty$.

A7. The observable state $s$ and the observed decisions $x$ are randomly sampled i.i.d. from distributions with densities $g_n(s)$ and $g_n(x)$, which are absolutely continuous over $S$ and $\mathcal{X}$, respectively.

Assumption A1. provides regularity conditions on the decision set and expected reward function. Assumption A2. requires subexponential tails on the noise distribution. Assumption A3. is a regularity condition on the covariate distribution to ensure the uniqueness of the linear estimates; it can be satisfied through a combination of sampling from a distribution that is absolutely continuous with respect to the Lebesgue measure over $\mathcal{X} \times S$ and algorithm partition choice. Assumption A4. is a condition that can be included in the algorithm and checked along with the minimal number of observations. If $\mathcal{X}$ is given, it can be computed directly, otherwise it can be approximated using $\{x_i : i \in C_k(s)\}$. Assumption A5. ensures that there are enough observations in the terminal nodes to fit the linear models. Assumption A6. places shrinkage conditions on the bandwidths. Assumption A7. places regularity conditions on the state and decision sampling distributions. Assumptions A1.-A2. are regularity conditions on the underlying problem that are satisfied in many settings where the noise is not heavy-tailed and its distribution changes smoothly with the state and decision. Assumptions A3.-A6. are restrictions on the algorithm. Assumptions A3. and A7. are standard restrictions on the sampling distribution.

**Theorem 3.1.** If conditions A1. through A7. hold, the kernel is Gaussian or fixed window, and $F(x \mid s)$ is convex in $x$ for all $s \in S$, then,

$$
\max_{k=1, \ldots, K_n(s)} \sup_{x \in A_k(s)} \left| \alpha_k(s) + \beta_k^T(s)x - F(x \mid s) \right| \to 0,
$$

$$
\max_{k=1, \ldots, K_n(s)} \sup_{x \in A_k(s)} \left\| \beta_k - \nabla_x F(x \mid s) \right\|_\infty \to 0,
$$

in probability as $n \to \infty$.

The K-CAP algorithm is similar to the SUPPORT algorithm of [19], except the refitting step of K-CAP allows partition subsets to be polyhedra with at most $D \log(n)$ faces, and the observations are weighted by kernels. Theorem 3.1 is similar to Theorem 1 of [19]; to prove our theorem, we modify parts of the proof in [19] that rely on a fixed number of polyhedral faces and account for the error introduced through the varying-coefficient model. Theorem 3.2 extends to consistency to $\bar{F}_n(x \mid s)$ under the $\ell_\infty$ metric and Theorem 3.3 gives consistency conditions for the optimal sets. All proofs are given in the Appendix.

**Theorem 3.2.** Suppose that assumptions A1. through A7. hold, the kernel is Gaussian or fixed window, and $F(x \mid s)$ is convex in $x$ for all $s \in S$. Then, for every $s \in S$,

$$
\sup_{x \in \mathcal{X}} \left| \bar{F}_n(x \mid s) - F(x \mid s) \right| \to 0
$$

in probability as $n \to \infty$.

Let $\theta^*(s)$ be the optimal solution value and $\theta_n(s)$ be the approximate value for state $s$,

$$
\theta^*(s) = \min_{x \in \mathcal{X}} F(x \mid s), \quad \theta_n(s) = \min_{x \in \mathcal{X}} \bar{F}_n(x \mid s).
$$
Similarly, let $T^*(s)$ and $T_n(s)$ be the set of optimal solutions for the true and approximated problems, respectively,

$$T^*(s) = \{ x \in \mathcal{X} : F(x | s) = \theta^*(s) \}, \quad T_n(s) = \{ x \in \mathcal{X} : \hat{F}_n(x | s) = \theta_n(s) \}.$$  

**Theorem 3.3.** Suppose that assumptions A1 through A7 hold, the kernel is Gaussian or fixed window, and $F(x | s)$ is convex in $x$ for all $s \in S$. Then for every $s \in S$, $|\theta_n(s) - \theta^*(s)| \to 0$ in probability and

$$d(T_n(s), T^*(s)) = \max \left\{ \sup_{x_1 \in T_n(s)} \inf_{x_2 \in T^*(s)} |x_1 - x_2|, \sup_{x_1 \in T^*(s)} \inf_{x_2 \in T_n(s)} |x_1 - x_2| \right\} \to 0$$

in probability as $n \to \infty$.

### 3.4. Selecting a Partition Size for K-CAP

The terminal model of K-CAP often overfits the data. Since K-CAP is highly sensitive to the choice of partition size, $K(s)$, we use model selection to choose a value for $K$. Cross-validation is one such method, but it becomes computationally demanding for larger values of $n$. When $n$ is large, we recommend using generalized cross-validation (GCV) [28] to approximate leave-one-out cross-validation.

When run to termination, K-CAP produces models $M_1(s), \ldots, M_K(s)$, each of which contains a collection of linear models. Let $\hat{f}_n(x)$ be the linear estimator given $n$ observations. In a weighted linear regression setting, leave-one-out cross-validation can be approximated by

$$\sum_{i=1}^{n} w_n(s_i) (y_i - \hat{f}_{n, i-1}(x_i))^2 = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{f}_{n, i-1}(x_i))^2 = \frac{1}{n} \sum_{i=1}^{n} \left( \frac{\hat{y}_i - \hat{f}_n(x_i)}{1 - H_{ii}} \right)^2 \tag{3.5}$$

$$\approx \frac{1}{n} \sum_{i=1}^{n} \left( \frac{\hat{y}_i - \hat{f}_n(x_i)}{1 - \text{Tr}(H)/n} \right)^2, \tag{3.6}$$

where $\hat{Y} = \hat{W}^t Y$, $\hat{X} = \hat{W}^t X$, $\hat{W}$ is the diagonal matrix composed of $w_n(s - s_1), \ldots, w_n(s - s_n)$, $H_{ii}$ is the $i$th diagonal element of the hat matrix, $\hat{H} = \hat{X}^T \hat{X}^{-1} \hat{X}^T$, and $\hat{f}_{n, i-1}$ is the estimator conditioned on all of the data minus element $i$. Equation (3.6) is the expectation of the right hand side of Equation (3.5), making the approximation accurate for larger sample sizes.

K-CAP is a collection of weighted linear models. We would like to approximate the weighted leave-one-out cross-validation error. We note that K-CAP is comprised of $K(s)$ linear models, so we apply GCV subset by subset to get the approximation for a model with $K$ subsets,

$$CV(K) = \sum_{i=1}^{n} w_n(s_i) (y_i - \hat{F}_{n,K,-i}(x_i | s))^2$$

$$= \sum_{i=1}^{n} w_n(s_i) \left( y_i - \alpha_{k(i), -i}(s) - \beta_{k(i), -i}^T(s)x_i \right)^2$$

$$\approx \sum_{i=1}^{n} w_n(s_i) \left( y_i - \alpha_{k(i)}(s) - \beta_{k(i)}^T(s)x_i \right)^2 \frac{1 - 1_i \in C_{k(i)} \text{Tr}(H_{k(i)}/n_{k(i)})}{1 - 1_i \in C_{k(i)} \text{Tr}(H_{k(i)})/n_{k(i)})},$$

where $k(i) \in \arg \max_{k=1, \ldots, K} \alpha_k(s) + \beta_k^T(s)x_i$. The approximation is both over the diagonal element of the hat matrix and the subset of observation $i$. 


Fig. 3.1. K-CAP and bagged K-CAP with $M = 100$. Here $X_i \sim \text{Unif}[-3, 3]$, $S_i \sim \text{Unif}[0, 1]$ and $y_i = (x_i + s_i)^2 + \epsilon_i$, $\epsilon_i \sim N(0, 1)$. The functions were created for $s = 0.75$ with $n = 5,000$ and a Gaussian kernel on $3$ i.i.d. training sets. Both estimators produce low error predictions, but bagged estimator (dashed line) is smoother than the original K-CAP estimator (solid line). There is less variation in the location of the minima for the smoother bagged estimator than the K-CAP estimator.

3.5. Ensemble Methods and Stochastic Search. While K-CAP is a fast, low error prediction method, it can produce poor results when used directly to approximate an objective function. This happens because K-CAP produces a sparse, piecewise linear estimator, as shown in Figure 3.1. When a piecewise linear function is minimized, the optimal solution lies on a vertex or a boundary. Since data from large regions are averaged to form the hyperplanes, the locations of these vertices are unstable. Recently, [32] proposed using ensemble methods, like bagging [13], smearing [14] and random searches, with piecewise linear convex estimators for objective function approximation. The ensemble methods that work well for these problems add randomness to a set of estimators, which are then averaged. Noise can be introduced through random weights on the training data (bagging), adding noise to the training data (smearing) or refining partitions along random directions. These types of ensemble methods reduce estimator variance by introducing uncertainty into the “hard” decisions, namely the boundary between hyperplanes [16]. Averaging over randomized estimators produces a smoother estimator, which is much more stable when used for objective function approximation. The results are still piecewise linear and preserve the asymptotic properties of the underlying estimator [16, 32]. An overview of bagging and smearing are given below.

Bagging: Create a new weighted training dataset $(x_i^{(m)}, y_i^{(m)}, w_i^{(m)})_{i=1}^n$ by sampling with replacement from the original dataset, $(x_i, y_i, w_n(s, s_i))_{i=1}^n$ $n$ times for $m = 1, ..., M$. Make a piecewise linear estimator for this dataset using K-CAP,
\((a_k^{(m)}(s), \beta_k^{(m)}(s))_{k=1}^{K^{(m)}(s)}\). Average these estimators to get the bagged estimator,

\[
\bar{F}(x | s) = \frac{1}{M} \sum_{m=1}^{M} \max_{k=1,...,K^{(m)}(s)} a_k^{(m)}(s) + \beta_k^{(m)^T}(s)x. \tag{3.7}
\]

See Figure 3.1 for a comparison K-CAP and bagged K-CAP.

**Smearing:** Create a new weighted training dataset \((\tilde{x}_i^{(m)}, \tilde{y}_i^{(m)}, \tilde{w}_i^{(m)})_{i=1}^{n}\) by setting \(\tilde{x}_i^{(m)} = x_i, \tilde{w}_i^{(m)} = w_n(s, s_i), \) and \(\tilde{y}_i^{(m)} = y_i + \eta^{m,i}, \) where \(\eta^{m,i} \sim N(0, \sigma^2)\) for \(m = 1, ..., M.\) As with bagging, make a piecewise linear estimator for this dataset using K-CAP, \((\tilde{a}_k^{(m)}(s), \tilde{\beta}_k^{(m)}(s))_{k=1}^{K^{(m)}(s)}\). These are averaged to produce the smeared estimator; results are similar to those for bagging.

4. Applications. We apply semi-convex regression metamodeling to two synthetic inventory management problems and a real-life online auto loan dataset.

4.1. Multi-product Inventory Management with Constraints. We consider a two product inventory management problem with budget and space constraints where the product perishes after one time period. Here, a store is selling products \(A\) and \(B.\) The manager 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. Inventory not sold by the end of the time period is lost. Let \((x_A, x_B)\) be the stocking decisions for \(A\) and \(B\) respectively; it is subject to a budget constraint, \(c_A x_A + c_B x_B \leq b,\) 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,

\[
\max_{x_A, x_B} -c_A x_A - c_B x_B + E[p_A \min(x_A, D_A) + p_B \min(x_B, D_B) | S = s] \quad (4.1)
\]

subject to: 
\[c_A x_A + c_B x_B \leq b,\]
\[r_A x_A + r_B x_B \leq r.\]

We generated data for Problem (4.1) in the following way. Demand and two state variables were generated according to an underlying state variable. There were 3 states with a Markov transition function; the transition parameters were \([7, 1.952, 1.048; 1.353, 7, 1.647; 1.440, 1.560, 1.7].\) In each state, the demand was generated from a two-dimensional Poisson distribution with parameters \((5, 10), (20, 30),\) and \((40, 15).\) The cost parameters were \((c_A, c_B) = (1, 1.3)\) with budget \(b = 50;\) the storage constraints were \((1.2, 1.1)\) with maximum storage \(r = 50,\) and the sale values were \((1.5, 2.0).\) Previously observed demand levels were used as the observable state variable. Solution methods were compared with previously observed demands of \((5, 10), (20, 30), (40, 15),\) and \((65/3, 55/3).\) The last value is not part of the training set, but may be of interest for prediction. We generated 10 training sets with \(n = 10,000\) and \(n = 100,000;\) for each previously observed demand testing level, one testing set of 100,000 observations was produced. Upper bounds were generated by solving the optimization problem with the testing scenarios as the outcomes. The training set decisions were generated in two ways: 1) uniformly over the decision space, and 2) in a Gaussian distribution with the mean equal to the decision generated by the upper bound and a variance of 2. Results for each training set were averaged; these are shown in Figure 4.1.

The following methods were compared:
1. **K-CAP.** Bandwidth selection was the same as in the function-based case; one observed decision per demand observation was used.

2. **Bagged K-CAP.** Similar to K-CAP except the estimator is generated through bagging with \( M = 100 \).

3. **Sample Average Approximation.** Sample average approximation using values from observed states; no observed samples for \((65/3, 55/3)\).

4. **Upper Bound.** The solutions were produced by finding the optimal decision based on the testing sets. This provides an upper bound to the true optimal decision.

Sample average approximation outperformed its competitors when it could be implemented. However, there were no samples from the state \((65/3, 55/3)\), so sample average approximation could not be used for that setting. K-CAP does not require previous samples from a query state so it could be used in all settings. Bagged K-CAP generally outperformed K-CAP. Both methods with sampling targeted around optimal decisions outperformed those with uniform sampling. This is likely because most of the samples came from “undesirable” locations. As the training set grew, K-CAP spent comparatively more effort to fit a function in these areas, rather than locations near optima. Since the functions are shape constrained, changes of fit in one location can greatly affect fit in other locations. Targeted sampling effectively eliminated this problem.
4.2. Multi-location Inventory Management with Constraints. The multi-location problem is an extension of the previous problem to a set of warehouses with spatially distributed demand and distance-based shipping costs. A company has a set of warehouses that house a perishable inventory, each with storage constraints and an overall budget constraint. Each time period, a set of orders are generated from a state-dependent spatial demand distribution. The orders are filled with inventory from the warehouses, but subject to a spatially dependent shipping cost. Additionally, unmet demand faces a penalty and all unsold items are lost. The manager must order inventory to maximize the expected profit using a demand observation from the previous time period.

Inventory management problems with perishable inventory can be viewed as two-stage stochastic optimization problems. Partial information about the demand at time $t + 1$ can be inferred from the demand at time $t$. Using this information, we would like to find the optimal ordering policy. Let $R(\pi(x_t), D_t | D_{t+1}, L_{t+1})$ be the reward generated under decision policy $\pi$ with order decision $x_t$, previously observed total demand $D_t$, demand realization $D_{t+1}$, and locations of that demand realization, $L_{t+1}$. The demand distribution changes along with some exogenous state of the world, $S_t$. The optimal policy maximizes

$$v^*(D_t) = \max_{x_t} \mathbb{E}[R(x_t, D_t | D_{t+1}, L_{t+1})]. \quad (4.2)$$

We use metamodeling with K-CAP to fit Equation (4.2).

To define the dataset, we select $n$ samples of order decision–observed demand pairs, that is, $(x_t(\omega_i), D_t(\omega_i))_{i=1}^n$. For each sample $\omega_i$, $m$ i.i.d. second stage samples of demand distribution–subsequent demand pairs, $(D_{t+1}(\omega_{i,j}), L(\omega_{i,j}))$, were generated conditionally given state $D_t(\omega_i)$. To reduce variance, we averaged the $m$ second stage samples to generate a random reward,

$$\hat{R}(\omega_i) = \frac{1}{m} \sum_{j=1}^m R(x_t(\omega_i), D_t(\omega_i) | D_{t+1}(\omega_{i,j}), L(\omega_{i,j})). \quad (4.3)$$

K-CAP was run with $x_t$ as the decision variable, $D_t$ as the observable state and $\hat{R}$ as the response.

The problem dynamics are as follows. A list of parameters is given in Table 4.1. Demand dynamics are driven by a hidden Markov model with 3 demand states, denoted by $S_t$. Given a demand state, the number of orders, $D_{t+1}$, is drawn from a Poisson distribution, $D_{t+1} | S_t = s \sim \text{Pois}(\lambda_s)$. Locations for each of the $D_{t+1}$ orders are drawn i.i.d. from a spatial demand distribution $p(L_{t+1,i} | s)$. These distributions are shown in Figure 4.2.

The reward function for the demand that arrives between time $t$ and $t + 1$ for a given demand $D_{t+1}$ and demand locations $L_{t+1} = (L_{t+1,i})_{i=1}^{D_{t+1}}$ is the sale price times the number of units sold minus the shipping costs, unmet demand costs and ordering costs. Since the shipping costs are location dependent, the reward is found by solving the following for the best shipping arrangement, $y$, for a given inventory level and
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>( x_t )</td>
<td>order at time ( t )</td>
</tr>
<tr>
<td>( D_{t+1} )</td>
<td>amount of demand from time ( t ) to ( t+1 )</td>
</tr>
<tr>
<td>( L_{t+1,i} )</td>
<td>location of order ( i ) for ( i = 1, \ldots, D_{t+1} )</td>
</tr>
<tr>
<td>( S_t )</td>
<td>demand distribution state at time ( t )</td>
</tr>
<tr>
<td>Decision Variables</td>
<td></td>
</tr>
<tr>
<td>( \text{order} )</td>
<td>order cost per unit</td>
</tr>
<tr>
<td>( p_{\text{sale}} )</td>
<td>sale price per unit</td>
</tr>
<tr>
<td>( c_{\text{ship}} )</td>
<td>shipping cost per unit distance</td>
</tr>
<tr>
<td>( c_{\text{unmet}} )</td>
<td>cost for unmet demand per unit</td>
</tr>
<tr>
<td>( n_{\text{dist}} )</td>
<td>number of distribution centers</td>
</tr>
<tr>
<td>( L_{\text{dist}}^i )</td>
<td>location of distribution center ( i ) for ( i = 1, \ldots, n_{\text{dist}} )</td>
</tr>
<tr>
<td>( B_{\text{max}} )</td>
<td>maximum storage level</td>
</tr>
<tr>
<td>( p_{\text{state}} )</td>
<td>number of demand states</td>
</tr>
<tr>
<td>( \gamma )</td>
<td>discount rate</td>
</tr>
</tbody>
</table>

Table 4.1

Variables and parameters for the inventory management problem.

![Fig. 4.2](image)

Fig. 4.2. State dependent demand densities. In all figures, ‘+’ denotes a warehouse location. Spatial demand for state 1, with \( \lambda_1 = 20 \) (A), state 2, with \( \lambda_2 = 40 \) (B), and state 3, with \( \lambda_3 = 80 \) (C).

demand,

\[
R(x_t, D_t | D_{t+1}, L_{t+1}) = \max_y \sum_{i=1}^{D_{t+1}} \sum_{j=1}^{n_{\text{dist}}} y_{i,j} \left( p_{\text{sale}}c_{\text{ship}}d(L_{t+1,i}, L_{\text{dist}}^j) - c_{\text{order}} x_{t,j} - c_{\text{unmet}} \sum_{i=1}^{D_{t+1}} y_{i,\text{unmet}} \right)
\]

subject to: \( 0 \leq y_{i,j} \leq 1, \quad 0 \leq y_{j,\text{unmet}} \leq 1, \quad y_{j,\text{unmet}} + \sum_{j=1}^{n_{\text{dist}}} y_{i,j} = 1. \)

Here, \( d(L_{t+1,i}, L_{\text{dist}}^j) \) is a distance function between demand location \( L_{t+1,i} \) and distribution location \( L_{\text{dist}}^j \). Equation (4.4) is a linear program and can be solved efficiently for large values of \( D_{t+1} \) and \( n_{\text{dist}} \).
The parameters were set as follows. The maximum storage level for each of the three locations was set to 40. There were three demand states with mean values (20, 40, 80), respectively. Spatial distributions were generated with a weight-varying Gaussian mixture model, shown in Figure 4.2. The order price was 5; the sale price was 15. The shipping cost was 1.5 per unit distance; the unmet demand cost was 1. The state transition matrix was \[
\begin{bmatrix}
0.7 & 0.1952 & 0.1048 \\
0.1353 & 0.7 & 0.1647 \\
0.1440 & 0.1560 & 0.7
\end{bmatrix}.
\]

We compared the following methods:

1. **K-CAP.** 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 \(\sigma_j\) is defined as \(\min(\text{sd, interquartile range}/1.349)\) [36]; one observed decision per demand observation was used.

2. **Bagged K-CAP.** Similar to K-CAP except the estimator is generated through bagging with \(M = 100\).

3. **Upper Bound.** The solutions were produced by finding the optimal decision based on the testing sets. This provides an upper bound to the true optimal decision.

Sample average approximation was not be used in this setting because generating scenario-dependent value functions were too computationally intensive. Each value for a state decision pair was generated by solving a non-trivial linear program.

Both K-CAP methods were tested under two training decision sampling regimes: uniform sampling over the decision space and Gaussian sampling with the upper bound decisions as a mean and a standard deviation of 10. Sampled values were truncated at 0 and the maximum storage level.

The previous demand was generated in a two-step process: first, an underlying state was generated from the limiting distribution of the transition matrix, second, a demand was generated given that state. This was done for \(n = 10,000\) and \(n = 50,000\) states; for each observed demand-decision pair, \(N = 5\) demand realizations were generated. The response values for each observed demand-decision pair were set to be the average reward across the \(N\) realizations. For each \(n\), 10 i.i.d. training sets were generated. Policies were tested on observed demand levels of \(s = [20, 30, 40, 60, 80]\) with a single testing set comprised of 2,000 demand realizations for each observed demand level. We used Gaussian kernels with bandwidth selected through a Gaussian distribution approximation implemented in the Matlab function *ksdensity*. The number of bagging subsamples was set at \(M = 100\). Results are given in Figure 4.3.

At all previously observed demand levels and training set sizes, bagged K-CAP outperformed ordinary K-CAP substantially. Targeted sampling offered some benefit over uniform sampling for the bagged estimator, but often produced worse results when the non-ensemble estimator is used.

### 4.3. Online Auto Lending

The Online Auto Lending dataset [69] includes all information about loans approved by an online lender in the United States between July 2002 and November 2004. Loan applicants supplied the lender with the type of loan they wanted (term and amount) along with personal information, such as their FICO scores. Given this information, the lender either did or did not approve the loan; if the loan was approved, the lender quoted a percentage rate (APR). The loan applicants could either decide to exercise the loan offer within 45 days. The dataset has records for 119,059 approved new car loans and about 7.7% of these were exercised.

When approving a loan, the lender needs to specify an APR. If the rate is too high, the applicant will probably not use the loan. If it is too low, the applicant will probably
use the loan, but the lender will not make much in loan interest. Moreover, applicants have different acceptance probabilities based on their own characteristics, such as their FICO score and loan term, and market characteristics, such as the current prime rate. For example, an applicant with a FICO score of 620 (a subprime level) and a term of 72 months would have a much higher likelihood of accepting a loan offer with a 9% APR than another applicant with a FICO of 850 and a term of 36 months. For each loan, acceptance probabilities should be decreasing in the rate and approaching zero with higher rates; these are well-modeled by log-concave functions, which provide a more flexible class of functions than alternatives like generalized linear models. Log-concave functions are a popular choice for density estimation [23, 20, 21, 60], but have potential to fit response surfaces as well because they are strongly unimodal. An expected return for a given APR is made by multiplying the net present value for an accepted loan with the probability that it will be accepted. If the net present value is also log-concave, the expected return is log-concave in the APR.

We used the dataset to build a partially shape constrained model for loan acceptance probability using K-CAP, treating the FICO score, loan term and application date as an observable state variable with the log probability of acceptance as the response. Application dates are used instead of prime interest rates because they also capture temporal patterns. The probability that a loan is accepted given the state was estimated through kernel density regression. The dataset was randomly divided into a training set of 95,000 training observations and 5,000 testing observations across 10
We used the following methods to predict the probability that a loan would be accepted:

1. **Constant.** The dataset average of $p = 0.077$ was used for the predicted value.
2. **Logistic Regression.** We used the `glm` function in R to fit a logistic regression.
3. **Kernel Regression.** 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 $\sigma_j$ is defined as $\min(sd, \text{interquartile range}/1.349)$ [36].
4. **K-CAP.** Response values were obtained from kernel regression; bandwidth selection was 2 times the “rule of thumb.”
5. **Bagged K-CAP.** Similar to K-CAP except the estimator is generated through bagging with $M = 5$.

All estimated probabilities were capped at a maximum of $1 - 10^{-7}$ and a minimum of $10^{-7}$. Predictive performance was measured by observed entropy between the predicted acceptance probability and the observed acceptance value (0 or 1), given by

$$H(x_i, s_i) = -p(y_i \mid x_i, s_i) \log(p(y_i \mid x_i, s_i)).$$

Results are shown in Figure 4.4. Logistic regression offered very little improvement over constant prediction (0.30% average improvement), while kernel regression had larger gains (4.36% on average). Both K-CAP and bagged K-CAP had much greater improvement over constant prediction (9.73% and 9.66%, respectively). Most K-CAP estimates are fit with a single hyperplane; this is likely because most APRs offered were within a narrow band around what maximized expected loan values and hence were well fit by a linear model. Therefore there was little improvement with larger bagging samples.
Shape constraints are useful when the estimated acceptance probability is used to decide an APR for a loan offer. The expected value of a loan is the probability of acceptance times the net present value of an accepted loan. We computed the net present value of a loan using the coupon cash flow discounted by the prime interest rate at the time a loan was offered. Default events or early repayments were not included in the NPV due to lack of data.

Estimated acceptance probabilities and expected values are shown for two randomly selected loans in Figure 4.5 to demonstrate the differences between shape-restricted and non-shape restricted models. The first loan was for a term of 72 months, with a requested amount of $39,500 and an applicant FICO score of 692 at a date where the prime rate was 1.39%; the loan was offered at 8.45% and was not accepted. The second loan was for a term of 60 months, with a requested amount of $18,641 and an applicant FICO score of 712 at a date where the prime rate was 1.84%; the loan was offered at 5.29% and was accepted.

Figure 4.5 shows that the shape-restricted estimators produce results that are more suitable for optimization than kernel estimates, which often have multiple local optima. Although maximizing a function in a univariate setting like this is computationally efficient, shape restrictions make multivariate optimization much more efficient. Using the estimated expected values, the first loan should have been offered at a 9.8% (kernel regression) or 4.65 % (K-CAP) APR and the second loan should have been offered at 7.25% (kernel regression) or 4.55 % (K-CAP) APR. The estimated optimal APR rates are low likely because default costs are not included in the
NPV computations. A default cost of $5,000 would bring the K-CAP APR to 8.45% for the first loan (not an entirely unlikely value due to the size of the first loan, its long term and relatively low applicant FICO score), while a cost of $400 would bring the K-CAP APR to 5.4% on the second loan.

5. Conclusions and Future Directions. In this paper, we proposed methods for stochastic search in the presence of convexity and an observable state variable. The solution methods we proposed created deterministic surrogate objective functions by fitting partially shape-constrained functions: convex in the decision variable, unconstrained in the state. This allowed the surrogate functions to be easily searched by a commercial solver. Since there are currently no semi-convex regression methods, we introduced Kernelized Convex Adaptive Partitioning and found consistency conditions.

K-CAP scales well to large problem sizes (efficient for 100,000+ observations). While there are many unconstrained methods that scale well to large data sets, there are relatively few multivariate convex regression methods that scale well. Most produce an estimator by solving large quadratic programs or semidefinite programs. These are methods are slow in practice, requiring $O(n^{3.5})$ or more flops. K-CAP uses a newer sparse linear estimator approach, which avoids the computational problems associated with older convex regression methods. Observed complexity for sparse linear models is closer to $O(n \log^2 n)$; see [33] for numerical comparisons. When this approach is combined with ensemble methods, K-CAP consistently produces high quality results for regression and optimization.

K-CAP is effective for moderate dimensionality ($p + d \leq 10$), but the algorithm is limited by general nonparametric data requirements. Convexity does reduce the data requirements: lower bounds dictated by the $\epsilon$-covering number of the set of convex functions scale with $p/2$ rather than $p$, like unconstrained functions [30]. However, even this is not particularly helpful when $p$ is large. Developing dimension reduction methods for shape constrained inference would allow these methods to be used effectively on high dimensional problems.

We believe that semi-convex metamodeling can be particularly useful for stochastic search using observational data where an observable state variable is present. Open questions include how best to sample decisions, particularly in a cost-sensitive, online setting.

Appendix A. Proof of Theorem 3.1. Proof of Theorem 3.1. Fix $s \in S$. The claims of Theorem 3.1 follow if we can show that as $n \to \infty$,

$$\max_{k=1,\ldots,K_n(s)} d_{nk}(s)^{-1} \left| \hat{\beta}_k(s) - \Delta(\bar{x}_k, s) \right| \to 0$$

in probability, where $\hat{\beta}_k(s) = [\alpha_k(s), \beta_k(s)]^T$ and $\Delta(\bar{x}_k, s)$ is the $p + 1$ dimensional vector whose components are $d_{nk}^{[u]}(s)D^u F(\bar{x}_k | s)$. Here $u$ is the matrix of partial derivative indicators that are used to generate $\hat{\beta}_k$, $[u]$ is the $p + 1$ dimensional vector that is the sum of the rows and $D^u$ is the derivative operator. Through a Taylor expansion of the mean function, $F(x | s) = \Delta(\bar{x}_k, s)[1, x_1 - \bar{x}_1, \ldots, x_p - \bar{x}_p]^T + r(x, \bar{x}, s)$. We note that

$$\hat{\beta}_k(s) = \left[ \sum_{i \in C_k(s)} \frac{1}{n_d} \Gamma_i \Gamma_i^T K_h(s - s_i) \right]^{-1} \left[ \sum_{i \in C_k(s)} \Gamma_i K_h(s - s_i)y_i \right]$$
when \( \left[ \sum_{i \in C_h(s)} \Gamma_i \Gamma_i^T K_h(s) \right]^{-1} \) exists, although this is not always the case. Instead, we use standard calculations to approximate

\[
(1 + o_P(1)) |\hat{\beta}_k(s) - \Delta(\bar{x}_k, s)| = g_u(s)^{-1} \left[ \sum_{i \in C_h(s)} \Gamma_i \Gamma_i^T \right]^{-1} \hat{R}_k(s),
\]

where

\[
\hat{R}_k(s) = \sum_{i \in C_h(s)} \Gamma_i K_i(s) y_i - \sum_{i \in C_h(s)} \Gamma_i \Gamma_i^T K_h(s - s_i) \Delta(\bar{x}_k, s),
\]

\[
= \sum_{i \in C_h(s)} \Gamma_i K_h(s - s_i) \left( \Gamma_i^T \Delta(\bar{x}_k, s_i) + r(x_i, \bar{x}_k, s_i) + \epsilon_i \right)
- \sum_{i \in C_h(s)} \Gamma_i \Gamma_i^T K_h(s - s_i) \Delta(\bar{x}_k, s),
\]

\[
= \sum_{i \in C_h(s)} \Gamma_i K_h(s - s_i) [\Gamma_i^T (\Delta(\bar{x}_k, s_i) - \Delta(\bar{x}_k, s)) + r(x_i, \bar{x}_k, s_i) + r(x_i, \bar{x}_k, s) + \epsilon_i].
\] (A.1)

The first two terms of Equation (A.1) converge to 0 in probability using standard kernel arguments. For each component of the first term,

\[
\left[ \sum_{i \in C_h(s)} \Gamma_i \Gamma_i^T K_h(s - s_i) (\Delta(\bar{x}_k, s_i) - \Delta(\bar{x}_k, s)) \right] \ell
\]

\[
= \sum_{i \in C_h(s)} \sum_{j=0}^d \left[ \Gamma_i \Gamma_j (\Delta_j(\bar{x}_k, s_i) - \Delta_j(\bar{x}_k, s)) \right] K_h(s - s_i),
\]

\[
= \left( 1 + O_P \left( \left[ \frac{\log(n_k)}{n_k h^d} \right]^{1/2} \right) \right)
\times \sum_{i \in C_h(s)} \sum_{j=0}^d \int \left[ \Gamma_i \Gamma_j (\Delta_j(\bar{x}_k, t) - \Delta_j(\bar{x}_k, s)) \right] K_1 \left( \frac{s - t}{h} \right) g_u(t) dt,
\]

which converges to 0 in probability as \( n_k \to \infty \) due to conditions A3. and A6. Using similar arguments, the second term of Equation (A.1) converges to 0 in probability. From conditions A3. and A4., the third term of Equation (A.1) converges to 0 in probability.

To show that the fourth term of Equation (A.1), the random error term, converges to 0, we modify the arguments of [15], [19] and [33]. Note that \( \epsilon_i \) is a mean 0 random variable; condition A2. implies that there exist \( k_1 > 0 \) and \( r > 0 \) such that

\[
p(w \mid x, s) \leq 2 \exp(k_1 w^2/2) \text{ for all } x \in \mathcal{X}, s \in \mathcal{S} \text{ and } 0 \leq w \leq r \text{ using arguments from Lemma 12.27 of [15]. Now suppose that } A_k(s) \text{ is a fixed polyhedron, all the data points } (x_i, s_i) \text{ are fixed, deterministic points in } \mathcal{X} \times \mathcal{S} \text{ and the corresponding } \epsilon_i \text{'s are independent random variables. We modify Lemma 12.26 of [15] and apply it to each component of the } p + 1 \text{ dimensional vector,}
\]

\[
d_{nk}(s)^{-m} \left[ \sum_{i \in C_h(s)} K_h(s - s_i) \right]^{-1} \sum_{i \in C_h(s)} \Gamma_i K_h(s - s_i) \epsilon_i.
\]
There exist constants $k_2 > 0$, $k_3 > 0$ and $\omega_0 > 0$ such that

$$
\mathbb{P}
\left(
\frac{1}{n_k h^d} \sum_{i \in C_h(s)} \Gamma_i K_h(s - s_i) \epsilon_i \mid > \omega
\right)
\leq k_2 \exp\left\{-k_3 d_{nk}^2(s) h^2 \omega^2 \sum_{i \in C_h(s)} K_h^2(s - s_i)\right\}
$$

whenever $\omega \leq \omega_0$. We then modify Lemma 12.27 of [15] to account for the fact that the subsets are produced by the intersection of at most $K_n(s)$ hyperplanes. The VC dimension bounds the number of polyhedral faces for each subset to be $(K_n(s) \log(n))^{(p+2)}$. Following the proof for 12.27, Equation (A.2) shows that for a sequence $m_n \to \infty$,

$$
\mathbb{P}
\left(
\left|\hat{\beta}_k(s) - \Delta(x_k, s)\right| > \omega \mid (x_i, s_i, y_i)_{i=1}^n
\right)
\leq k_2 c
\leq k_2 c
\leq k_2 c_n^2 m_n^{1/2} d_{nk}(s)
$$

conditioned on the event that $n_k \geq m_n^{1/2} \log(n) (K_n^*)^{1/2}$ for every $k$, where $K_n^* = \frac{1}{n_k} \sum_{i \in C_h(s)} \left(\frac{1}{h^2} K_h(s)\right)^2$. This event also implies that $K_n(s) \leq \frac{n^{1/2} d_{nk}(s)}{(K_n^*)^{1/2} \log(n)}$. Using the VC dimension of the partition,

$$
\mathbb{P}
\left(
\left|\hat{\beta}_k(s) - \Delta(x_k, s)\right| > \omega \text{ for each } A_k(s) \text{ and } n_k \geq m_n^{1/2} \log(n) (K_n^*)^{1/2}
\right)
\leq k_2 \frac{2^{1+K_n(s) \log(n)(p+2)} n_k^{K_n(s) \log(n)(p+2)} - k_3 \omega^2 m_n^{1/2} d_{nk}(s)}{(K_n^*)^{1/2} \log(n)}
$$

Since A5 ensures that $m_n \to \infty$, the result holds. \(\square\)

**Appendix B. Proof of Theorem 3.2.** Proof of Theorem 3.2. The proof is the same as Theorem 2 in [33]. Fix $\epsilon > 0$; let $d_\mathcal{X}$ be the diameter of $\mathcal{X}$. Choose $N$ such that for every $n \geq N$

$$
\mathbb{P}
\left\{
\max_{k=1, \ldots, K_n(s), x \in A_k(s)} \left|\alpha_k(s) + \beta_k^T(s)x - F(x \mid s)\right| > \frac{\epsilon}{\zeta d_\mathcal{X}}
\right\} < \epsilon/2,
$$

$$
\mathbb{P}
\left\{
\max_{k=1, \ldots, K_n(s), x \in A_k(s)} \|\beta_k(s) - \nabla_x F(x \mid s)\|_\infty > \frac{\epsilon}{\zeta d_\mathcal{X}}
\right\} < \epsilon/2.
$$

Fix a $\delta$ net over $\mathcal{X}$ such that at least one point of the net sits in $A_k(s)$ for each $k = 1, \ldots, K_n(s)$. Let $n_\delta$ be the number of points in the net and let $x_\delta^k$ be a point.
Then,
\[
\mathbb{P}\left\{ \sup_{x \in \mathcal{X}} |\bar{F}_n(x | s) - F(x | s)| > \epsilon \right\} = \mathbb{P}\left\{ \left| \sup_{x \in \mathcal{X}} \left[ \max_{k=1,\ldots,K_n} \alpha_k(s) + \beta_k^T(s)x - F(x | s) \right] \right| > \epsilon \right\},
\]
\[
\leq \mathbb{P}\left\{ \max_{i=1,\ldots,n_\delta} \left[ \max_{k=1,\ldots,K_n(s)} \alpha_k(s) + \beta_k^T(s)x_i^\delta - F(x_i^\delta | s) \right] > \epsilon \right\},
\]
\[
\leq \mathbb{P}\left\{ \max_{i=1,\ldots,n_\delta} \left| \sum_{k=1}^{K_n(s)} \left( \alpha_k(s) + \beta_k^T(s)x_i^\delta \right) \mathbb{1}_{\{x_i^\delta \in A_k(s)\}} - F(x_i^\delta | s) \right| > \frac{\epsilon}{\zeta d_X} \right\},
\]
\[
< \epsilon. \quad \Box
\]

### Appendix C. Proof of Theorem 3.3.

**Proof of Theorem 3.3:** Fix \( s \in \mathcal{S} \). The claim that \( |\theta_n(s) - \theta^*(s)| \to 0 \) in probability follows from Theorem 3.2. We show the second claim by contradiction. Suppose that \( d((T_n(s), T^*(s))) \to 0 \) in probability. Then, there exists \( \epsilon > 0 \) and \( \gamma > 0 \) such that \( \mathbb{P}(d(T_n(s), T^*(s)) > \epsilon) > \gamma \) for at least one subsequence. There exists a set \( \Omega_0 \subset \Omega \), a subsequence \( (n_k) \), and a \( \gamma_0 > 0 \) such that \( \mathbb{P}(\Omega_0) > \gamma_0 \) and \( \Omega_0 = \{ \omega \in \Omega : d(T_n(s), T^*(s)) > \epsilon \} \). Restricting the space to \( \Omega_0 \) and subsequence \( (n_k) \), Theorem 3.2 and the continuity of \( F \) implies that there exists a set \( \Omega_1 \subset \Omega_0 \), a subsequence \( (n_{k_\ell}) \subset (n_k) \), \( \gamma_1 > 0 \), and \( \delta > 0 \) such that
\[
\Omega_1 = \left\{ \omega \in \Omega_0 : \sup_{x \in \mathcal{X}} |\bar{F}_{n_{k_{1\ell}}}(x | s) - F(x | s)| < \frac{\delta}{4}, \quad \inf_{x \in T_{n_{k_{1\ell}}}(s)} |F(x | s) - F(x^* | s)| > \frac{\delta}{2} \right\}
\]
and \( \mathbb{P}(\Omega_1) > \gamma_1 \). However, on \( \Omega_1 \),
\[
|\theta_{n_{k_{1\ell}}}(s) - \theta^*(s)| = \sup_{x_1 \in T_{n_{k_{1\ell}}}(s), x_2 \in T^*(s)} |\bar{F}(x_1 | s) - F(x_2 | s)| > \delta/2.
\]
This implies \( \mathbb{P}\left( |\theta_{n_{k_{1\ell}}}(s) - \theta^*(s)| > \delta/2 \right) > \gamma_1 \), contradicting \( |\theta_n(s) - \theta^*(s)| \to 0 \) in probability. \( \Box \)

### Appendix D.

**REFERENCES**


[66] A. Slivkins, Contextual bandits with similarity information, in COLT, 2011.