Multivariate convex regression with adaptive partitioning

Lauren Hannah, David Dunson

May 11, 2011

Abstract

We propose a new, nonparametric method for multivariate regression subject to convexity or concavity constraints on the response function. Convexity constraints are common in economics, statistics, operations research and financial engineering, but there is currently no multivariate method that is computationally feasible for more than a few hundred observations. We introduce Convex Adaptive Partitioning (CAP), which creates a globally convex regression model from locally linear estimates fit on adaptively selected covariate partitions. Adaptive partitioning makes computation efficient even on large problems. Convexity itself acts as a regularizer, making CAP resistant to overfitting. We give consistency results for the univariate case. CAP is applied to value function approximation for pricing American basket options with a large number of underlying assets.

Key words: Nonparametric regression, shape constraint, convex regression, treed linear model

1 Introduction

Consider the regression model for \( x \in \mathcal{X} \subset \mathbb{R}^p \) and \( y \in \mathbb{R} \) where

\[
y = f_0(x) + \epsilon,
\]

where \( f_0 : \mathbb{R}^p \rightarrow \mathbb{R} \) is a mean regression function and \( \epsilon \) is a mean 0 random variable. In this paper, we study the situation where \( f_0 \) is subject to a convexity constraint. That is,

\[
f_0(x_1) \geq f_0(x_2) + \nabla f_0(x_1)^T (x_1 - x_2),
\]

(1)

for every \( x_1, x_2 \in \mathcal{X} \), where \( \nabla f_0(x) \) is the gradient of \( f_0 \) at \( x \). Given the observations \( (x_1, y_1), \ldots, (x_n, y_n) \), we would like to estimate \( f_0 \) subject to the convexity constraint; this is called the convex regression problem. Note that convex regression is easily extended to concave regression, because a concave function is the negative of a convex function.

Convex regression problems are frequently found in economics, operations research, financial engineering and statistics. In economics, operations research and financial engineering, it is common for theory to place shape restrictions on functions, such as production curves, preference functions and value-to-go functions. In statistics, shape restrictions like log-concavity are useful in density estimation, including the multivariate setting [Cule et al. 2010, Cule & Samworth 2010, Schüttmacher & Dümbgen 2010]. Although convex regression has been well-explored in the univariate setting, it remains largely uninvestigated in the multivariate setting.

In the univariate setting, there are many computationally efficient algorithms. These methods rely on the ordering implicit to the real line. Ordering reduces the convexity constraint of Equation (1) into a simple ordering of the derivative function,

\[
f'_0(x_1) \leq f'_0(x_2)
\]

(2)
for every $x_1 \leq x_2$ where the derivative of $f_0$ exists. Univariate estimators fall into two groups, piecewise linear and continuous. The least squares estimator (LSE) is piecewise linear; it is found by solving a quadratic program with $n - 2$ linear constraints (Hildreth 1954). Consistency, rate of convergence, and asymptotic distribution of the LSE have been shown by Hanson & Pledger (1976), Mammen (1991) and Groeneboom et al. (2001), respectively. Neelon & Dunson (2004) gives a Bayesian version of a piecewise linear model. Smooth models include shape restricted kernel regression (Birke & Dette 2007) and splines in a Bayesian setting (Meyer 2008, Shively et al. 2011).

Unlike the univariate case, convex functions in multiple dimensions cannot be represented by a simple set of first order conditions. In multiple dimensions, projection to the set of convex functions becomes computationally intensive. Consider the least squares estimator, $\hat{f}_{n}^{LSE}$, found by solving the quadratic program,

$$\min \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

subject to $\hat{y}_j \geq \hat{y}_i + g_i^T(x_j - x_i)$, $i, j = 1, \ldots, n$. The function $\hat{f}_{n}^{LSE}$ has the general form,

$$\hat{f}_{n}^{LSE}(x) = \max_{i \in \{1, \ldots, n\}} \hat{y}_i + g_i^T(x - x_i).$$

The corresponding least squares functional estimate is the maximum of a set of hyperplanes; often, multiple observations are assigned to the same hyperplane. The hyperplane $g_i$ approximately is the gradient of $f_0$ at $x_i$. The LSE becomes impractical due to its size: it has $n^2$ constraints. If $n = 1,000$, a relatively small problem, the LSE becomes almost impossible to find with 1,000,000 constraints. While the LSE for convex regression has been known in the convex optimization community for a long time (Boyd & Vandenberghe 2004), the characterization (Kuosmanen 2008) and consistency (Seijo & Sen 2011) of the least squares problem have only recently been studied. In different approaches, kernel smoothing with a restricted Hessian (Henderson & Parmeter 2009) and transforming the shape constraints into a combinatorial optimization problem (Koushanfar et al. 2010) have been proposed. However, both maintaining a positive semi-definite Hessian for each observation and combinatorial optimization are computationally burdensome.

Nevertheless, the LSE does illustrate two concepts that motivate the algorithm proposed in this paper. First, the constraints in Equation (3) show that it is quite difficult to project an unconstrained estimator into the space of convex functions. Second, Equation (4) shows that it is very easy to construct a convex function: simply take the max over a set of hyperplanes. Therefore, we will work directly in the space of convex functions, specifically piecewise linear convex functions, rather than projecting an unconstrained estimator into that space.

In this paper, we present a new nonparametric method, Convex Adaptive Partitioning, or CAP for short, that approximates the least squares minimizer of Equation (3). Like the exact least squares estimator, the CAP estimator is formed by taking the maximum over a set of hyperplanes. Unlike the least squares estimator, CAP does not project a function into the constrained space. Instead, we adaptively search over the set of hyperplanes, adding one at a time to the model. This lets CAP search directly in the space of convex functions. Starting with an initial partition of $\mathcal{X}$ into two subsets, we fit an unconstrained least squares linear regression to each subset. A rough convex estimate is formed by taking the maximum over the two hyperplanes. By iteratively refining the partition via binary splitting of subsets and adding hyperplanes that improve the fit, we converge to the CAP estimator.
CAP is a computationally efficient estimator. It splits subsets only at a fixed set of knots in cardinal directions in a manner similar to basis function addition in Multivariate Adaptive Regression Splines (MARS) \cite{Friedman1991}. This allows CAP to generate estimates quickly, even for large datasets. Additionally, CAP is suitable for use in higher dimensions (10 to 100) because it uses convexity as a regularizer. CAP has appealing theoretical properties, such as a regression estimate that is convex over all of $\mathbb{R}^p$, guaranteed convergence, and consistency in at least the univariate setting. In this paper, we also explore the performance of CAP as $p$ increases, along with the role of regularization. CAP is empirically tested on multivariate convex regression problems and applied to value function estimation when pricing American basket options with a large number of underlying assets.

This paper is organized as follows. In Section 2, we present the CAP algorithm. In Section 3, we discuss the computational efficiency of CAP and study its behavioral properties under regularization as the number of dimensions grows. In Section 4, we discuss the theoretical properties of CAP, including consistency in the univariate case. In Section 5, we apply CAP and competing algorithms to convex regression problems, including value function estimation for pricing American basket options. In Section 6, we discuss our results and give directions for future work.

2 The CAP Algorithm

Unlike previous approaches to convex regression, we use an adaptive procedure to approximate the least squares estimator for convex regression. The procedure is similar to stepwise fitting of linear models, except hyperplanes, not covariates, are added in a forward stepwise fashion. The question is how one should add hyperplanes.

A natural way of defining hyperplanes is by partitioning the covariate space into a set of $K$ disjoint subsets, $A_1, \ldots, A_K$, and then fitting a hyperplane, $\alpha_k + \beta_k^T x$, within each subset. The resulting function estimate,

$$\hat{f}_n(x) = \sum_{k=1}^K 1_{x \in A_k} (\alpha_k + \beta_k^T x),$$

is not necessarily convex. Placing restrictions on the hyperplanes to maintain convexity is computationally difficult, especially as $p$ gets larger. However, convexity is guaranteed if an estimator is defined as the maximum of a set of hyperplanes,

$$\hat{f}_n(x) = \max_{k \in \{1, \ldots, K\}} (\alpha_k + \beta_k^T x). \quad (5)$$

We propose fitting a regression function like that in Equation (5), defined as a maximum over a set of hyperplanes. Equation (5) is clearly sensitive to the choice of hyperplanes, and our focus is to define a simple and fast iterative algorithm for selecting them. This is done by using the partition of the covariate space defined by Equation (5), and then adaptively selecting subsets to split and refit with new hyperplanes.

CAP relies heavily on partitioning, both the covariate space and the observation space. Suppose there are $n$ observations. At iteration $m$, let $C^m = \{C^m_1, \ldots, C^m_K\}$ be a partition of the observation indices, $\{1, \ldots, n\}$. Let $B^m = \{\alpha^m_k + \beta^m_k^T x\}_{k=1}^K$ be a set of hyperplanes defined at iteration $m$, and let $A^m = \{A^m_1, \ldots, A^m_K\}$ be a partition of the covariate space induced by $B^m$. That is,

$$A^m_k = \left\{ x \in \mathcal{X} : \alpha^m_k + \beta^m_k^T x \geq \alpha_{k'}^m + \beta_{k'}^m^T x \text{ for all } k' \neq k \right\}.$$ 

The relationship between $A^m$, $B^m$ and $C^m$ is shown in Figure 1.
Figure 1: The relationship between $A^m$, $B^m$ and $C^m$. Here, the observation partition $C^m = \{C^m_1, C^m_2, C^m_3\}$ is used to generate the hyperplanes $B^m = \{\alpha_k^m + \beta_k^m x\}_{k=1}^3$. The subset $C_k^m$ generates $(\alpha_k^m, \beta_k^m)$ for $k = 1, 2, 3$. Then, the hyperplanes $B^m$ induce a partition of the covariate space, $A^m = \{A^m_1, A^m_2, A^m_3\}$. Note that the observation subsets, $C_k^m$, may not align with the covariate subsets, $A_k^m$.

CAP uses a partition of the observation space to fit hyperplanes; these hyperplanes are transformed via Equation (7) into a partition of the covariate space. The covariate partition is adaptively split to induce a new observation partitioning, $C^{m+1}$, which in turn is used to fit a new set of hyperplanes.

2.1 The Algorithm
The iterative procedure is as follows.

**Step 0. (Initialize)** Set $m = 1$. Start with all data points in same subset, $C^m_1$. Fit hyperplane $\alpha^m_1 + \beta^m_1 T x$ to the observations $(x_i, y_i)$ where $i \in C^m_1$. Set $K = 1$. Covariate partition $A^m$ is produced; it encompasses the entire covariate space. See Figure 2 for a graphical depiction.

**Step 1. (Partition)** Use hyperplanes $B^m = \{\alpha_k^m + \beta_k^m T x\}_{k=1}^K$ to define a new partition of the observations, $C^{m+1}$, as follows. If $K > 1$, for each hyperplane $k = 1, \ldots, K$, set

$$C_{k}^{m+1} = \left\{ i : \alpha_k^m + \beta_k^m T x_i \geq \alpha_j^m + \beta_j^m T x_i \text{ for every } j \neq k \right\}.$$  

If $K = 1$, $C^{m+1} = C^{m+1}_1 = \{1, \ldots, n\}$.

**Step 2. (Split)** For every observation subset $C_k^{m+1}$, fix a dimension $j \in \{1, \ldots, p\}$. In dimension $j$, fix $D$ knots $a_1^{j,k} < \cdots < a_D^{j,k}$. This is done adaptively by finding the maximum and minimum
values for the observations associated with subset $A_k^m$ in dimension $j$, and then fixing an equally spaced grid with $D$ entries in that region. That is,

$$a_{j,k}^{\ell} = a_{j,k}^{\ell}_{\text{min}} = \min \{x_{ij} : i \in C_{k}^{m+1} \},$$

$$a_{j,k}^{\ell}_{\text{max}} = \max \{x_{ij} : i \in C_{k}^{m+1} \},$$

$$a_{j,k}^{\ell} = a_{j,k}^{\ell}_{\text{min}} + \frac{\ell}{D+1}(a_{j,k}^{\ell}_{\text{max}} - a_{j,k}^{\ell}_{\text{min}}), \quad \ell = 1, \ldots, D. $$

Fix knot $\ell \in \{1, \ldots, D\}$. Create a new partition by dividing $C_{k}^{m+1}$ into two regions,

$$C_{k,j,\ell}^{+} = \{ i : i \in C_{k}^{m+1} \text{ and } x_{i,j} > a_{j,k}^{\ell} \},$$

$$C_{k,j,\ell}^{-} = \{ i : i \in C_{k}^{m+1} \text{ and } x_{i,j} \leq a_{j,k}^{\ell} \}. $$

Define $C_{k,j,\ell}^{m+1}$ as the resulting partition of the observations.

**Step 3. (Fit)** For the fixed subset $k$, dimension $j$ and knot $\ell$, generate the collection of hyperplanes $B_{k,j,\ell}^{m+1}$ by 1) refitting hyperplanes in all subsets without knots, that is, for subsets $C_{k'}^{m+1}$ for $k' \neq k$, and 2) fitting hyperplanes to subsets $C_{k,j,\ell}^{m+1}$ and $C_{k,j,\ell}^{-}$,

$$B_{k,j,\ell}^{m+1} = \{ (\alpha_{1}, \beta_{1}), \ldots, (\alpha_{k-1}, \beta_{k-1}), (\alpha_{k+1}, \beta_{k+1}), \ldots, (\alpha_{K}, \beta_{K}),$$

$$(\alpha_{k,j,\ell}^{+}, \beta_{k,j,\ell}^{+}), (\alpha_{k,j,\ell}^{-}, \beta_{k,j,\ell}^{-}) \},$$(\alpha_{k',\ell}^{+}, \beta_{k',\ell}^{+}) = \arg\min_{\alpha, \beta} \sum_{i \in C_{k'}^{m+1}} (y_{i} - \alpha - \beta^{T}x_{i})^2, \quad k' = 1, \ldots, k-1, k+1, \ldots, K,$$

$$(\alpha_{k,j,\ell}^{-}, \beta_{k,j,\ell}^{-}) = \arg\min_{\alpha, \beta} \sum_{i \in C_{k,j,\ell}^{-}} (y_{i} - \alpha - \beta^{T}x_{i})^2. $$

This not only fits least squares hyperplanes in the split partition, but in all other non-split partitions as well. Step 3 usually increases $K$ by 1, but refits may eliminate some subsets when
Step 5. (Update) If at least one of the following stopping criteria is met,

- $m > M$ for some pre-specified maximal number of iterations $M$, or

- the difference in square error in the estimators $\hat{f}_n^m$ and $\hat{f}_n^{m+1}$ is less than a pre-specified tolerance $\lambda \geq 0$ over the training dataset,

$$\sum_{i=1}^{n} (y_i - \hat{f}_n^{m+1}(x_i))^2 - \sum_{i=1}^{n} (y_i - \hat{f}_n^m(x_i))^2 \leq \lambda,$$

new hyperplanes dominate others. A hyperplane $(\alpha_k, \beta_k)$ is dominated when there exists a subset $S$ of the hyperplane indices of $B_{k,j,\ell}^{m+1}$ where $k \notin S$ and for every $x \in \mathcal{X}$, there exists a $k' \in S$ such that

$$\alpha_k + \beta_k^T x \leq \alpha_{k'} + \beta_{k'}^T x.$$

Remove all dominated hyperplanes from $B_{k,j,\ell}^{m+1}$.

Repeat Step 3 for all subsets $k$, dimensions $j$ and knots $\ell$. Steps 2 and 3 are demonstrated in Figure 3.

Step 4. (Select) Choose the best fitting set of hyperplanes from $\{B_{k,j,\ell}^{m+1}\}$, $k = 1, \ldots, K$, $j = 1, \ldots, p$, $\ell = 1, \ldots, D$. That is, set

$$B^{m+1} = \arg \min_{B_{k,j,\ell}^{m+1}} \sum_{i=1}^{n} \left( y_i - \max_{(\alpha_{k'}, \beta_{k'}) \in B_{k,j,\ell}^{m+1}} (\alpha_{k'} + \beta_{k'}^T x_i) \right)^2.$$

Set $K$ as the number of non-dominated hyperplanes in $B^{m+1}$. Let $\{\alpha_k^{m+1}, \beta_k^{m+1}\}_{k=1}^{K}$ be the non-dominated hyperplanes contained in $B^{m+1}$ and set

$$\hat{f}_n^{m+1}(x) = \max_k \alpha_k^{m+1} + \beta_k^{m+1}^T x.$$

Define $A^{m+1}$ as the covariate partition imparted by the hyperplanes in $B^{m+1}$,

$$A_k^{m+1} = \left\{ x \in \mathcal{X} : \alpha_k^{m+1} + \beta_k^{m+1}^T x \geq \alpha_{k'}^{m+1} + \beta_{k'}^{m+1}^T x \text{ for all } k' \neq k \right\},$$

for $k = 1, \ldots, K$. Set $A^{m+1} = \{A_1^{m+1}, \ldots, A_K^{m+1}\}$.

Step 5. (Update) If at least one of the following stopping criteria is met,

- $m > M$ for some pre-specified maximal number of iterations $M$, or

- the difference in square error in the estimators $\hat{f}_n^m$ and $\hat{f}_n^{m+1}$ is less than a pre-specified tolerance $\lambda \geq 0$ over the training dataset,
stop and set $B_{\text{final}} = B^m$. Set estimator $\hat{f}_n(x)$ as

$$\hat{f}_n(x) = \max_{k=1,...,K} \alpha_k^{\text{final}} + \beta_k^{\text{final}} T_x.$$ 

Otherwise, set $m = m + 1$ and go to Step 1.

### 2.2 Remarks on the CAP Algorithm

The CAP algorithm borrows many of its features from several previous nonparametric regression algorithms. It iteratively fits hyperplanes in Step 3 like the hinging hyperplanes algorithm of Breiman (1993). The knot selection in Step 2 is similar to the basis function selection of Multivariate Adaptive Regression Splines (MARS) (Friedman 1991) and variable selection of Classification and Regression Trees (CART) (Breiman et al. 1984).

Computational issues include the selection of $D$, the number of knots per partition, and $\lambda$, a tolerance parameter. These are discussed in Section 3. Additionally, CAP may encounter problems when the number of observations in a set is less than the number required to fit a hyperplane well. This can be ameliorated in one of two ways. First, we can place a minimum on the number of observations for a hyperplane to be fit or refit. Second, we can use regularization. This is also discussed in Section 3.

### 3 Empirical Analysis of CAP

In this section, we discuss the effects of tunable parameters and how CAP behaves when the number of dimensions $p$ grows.

#### 3.1 Computational Time and Tunable Parameters

CAP has two main tunable parameters, the number of knots $D$, and the error tolerance $\lambda$. Both of these affect the computational time. The computational time is best described in terms of time per iteration, that is, the time to go from $K$ hyperplanes to $K + 1$ hyperplanes. The computational complexity is determined by the number of linear regression models that need to be run. A general refit can be run first for all existing subsets. Then, in each dimension, $D$ linear models are run. However, the pseudo-inverse for each of the daughter subsets can be updated in an online fashion, greatly reducing the computational burden.

The error tolerance $\lambda$ tells CAP not to terminate until the gain between $\hat{f}_n^m(x)$ and $\hat{f}_n^{m+1}(x)$ is less than $\lambda$ on the training data set. Figure 4 shows the marginal computational time for including another hyperplane (top), the marginal squared error gain on the training data (middle), and the marginal squared error gain on the testing data (bottom). Note that the gains level off quite quickly while the computational time increases linearly in the number of hyperplanes. Figure 5 shows a blow-up of the squared error gains for the middle and bottom panels of Figure 4. While gains are always positive on the training data until CAP terminates, the gains can be negative on the testing data. Using cross-validation to determine $\lambda$ in a step-wise fashion not only greatly reduces the computational time of CAP, but it also reduces possible overfitting.

The number of knots, $D$, affects the computational time sublinearly when cleverly implemented, at the cost of storing pseudo-inverses. A higher $D$ value allows one to possibly select fewer hyperplanes before CAP terminates. Larger values of $D$ provide the most benefit in settings where $f_0$ is non-smooth.
Figure 4: The marginal computational time for including another hyperplane (top), the marginal squared error gain on the training data (middle), and the marginal squared error gain on the testing data (bottom) for data in Table 1 with \( n = 10,000 \) and \( p = 20 \).

Figure 5: A closeup of the marginal squared error gain on the training data (top), and the marginal squared error gain on the testing data (bottom) for data in Table 1 with \( n = 10,000 \) and \( p = 20 \). Note that \( M = 9 \) provides a better fit on the testing data than any higher value of \( M \).
Figure 6: Predicted values (blue dots) and true values (red line) for $y = x^2 + \epsilon$, where $\epsilon \sim N(0, 0.1^2)$. CAP was tried across varying data sizes ($n = 100, 1,000, 10,000$) and dimensions ($p = 1, 10, 100$). Linear fits were computed with a least squares estimate. Note that the convexity restriction acts as a regularizer even when a regularizer is not specifically included in the linear estimate.

3.2 Regularization, Overfitting and Dimensionality

In this subsection, we study what happens when CAP is applied to moderate to high-dimensional problems (where $p$ is between 10 and 100). Higher dimensions present two distinct challenges. First, higher dimensions mean more computations. Second, more covariates means more of a chance for spurious correlations between the covariates and the response. Regularization is usually used to counter the second problem, either through L1 or L2 objective function penalties, or by dimension reduction and variable selection.

Again, CAP approaches both of these problems in a different manner than other nonparametric estimators. By only selecting a few hyperplanes and only searching on cardinal directions, the computational costs are minimal. Then CAP uses convexity itself as a regularizer. Convexity is a highly restrictive constraint, and hence reduces overfitting. See Figure 6 for a comparison of CAP across different dimensions and sample sizes.

CAP can have difficulty selecting relevant components when $n$ is within an order of magnitude or two of $p$. In these cases, it can be helpful to include another (explicit) regularizer within the linear regressions, such as ridge regression (Hoerl & Kennard 1970), LASSO (Tibshirani 1996) or LARS (Efron et al. 2004). These regularizers force most coefficients to 0, at the cost of adding some bias to the local linear estimate.

To use a regularized estimator instead of the least squares estimator, change Step 3 of the CAP
Table 1: Algorithms were run on data generated by $y_i = (x_{i,1} + x_{i,2})^2 + \epsilon_i$, where $\epsilon_i \sim N(0, 1^2)$. The covariates, $x_{i,j}$, were generated from $x_{i,j} \sim Unif[-1, 1]$. Only two covariates, $x_{i,1}$ and $x_{i,2}$, have explanatory power; 18 nuisance covariates were added in the $p = 20$ case and 98 nuisance covariates were added in the $p = 100$ case. The GP inference algorithm did not converge for any values when $p = 100$.

algorithm for each set of hyperplanes, $(\alpha_{k,j,\ell}^{m+1}, \beta_{k,j,\ell}^{m+1})$. Regularization for linear regression relies on a shrinkage parameter. Since CAP fits linear models over data subsets with vastly different numbers of observations, the shrinkage parameter cannot be set to a single fixed value. To alleviate this problem, the shrinkage parameter is chosen by cross-validation for every set of proposed hyperplanes, $(\alpha_{k,j,\ell}^{m+1}, \beta_{k,j,\ell}^{m+1})$; this greatly adds to the computational complexity.

Regularization in CAP can guard against overfitting, but it introduces bias and greatly increases computational time. First, the least squares estimator is an unbiased estimator, whereas regularized estimators add bias to reduce variance. This can be a good tradeoff when $n$ is relatively small compared to $p$, but can actually produce worse estimators when $n$ gets large. Second, regularized estimators involve much more computation than the least squares estimate, especially when the linear regularization parameters are chosen by cross-validation.

The effects of regularization are compared empirically in Table 1. Here, CAP with a least squares estimator (CAP, LSE) is compared against CAP with a LARS estimator (CAP, LARS), and other non-parametric regression methods, including MARS, CART and Gaussian processes,
on a high-dimensional regression problem. Algorithms were run on data generated by

\[ y_i = (x_{i,1} + x_{i,2})^2 + \epsilon_i, \]

where \( \epsilon \sim N(0, 0.1^2) \). The covariates, \( x_{i,j} \), were generated from \( x_{i,j} \sim Unif[-1, 1] \). Only two covariates, \( x_{i,1} \) and \( x_{i,2} \), have explanatory power; 18 nuisance covariates were added in the \( p = 20 \) case and 98 nuisance covariates were added in the \( p = 100 \) case. The number of observations was varied across \( n = 100, 500, 1,000, 2,000 \) and \( 10,000 \).

Both CAP methods generally were competitive with or outperformed the other methods, often substantially. The regularized version of CAP outperformed the unregularized version of CAP only when the number of observations was less than two orders of magnitude greater than the number of covariates. Computational times for the regularized method were one to two orders of magnitude greater than times for the unregularized method. Regularization does seem to add value in certain cases, such as those where \( p \) is relatively close in size to \( n \) and computational time is not a significant consideration.

4 Theoretical Properties of CAP

In this section, we discuss some theoretical properties of CAP. We show that CAP converges and we give consistency results in the univariate case under uniform sampling.

4.1 Convergence

Convergence is a fairly straightforward property created because CAP only accepts new partitions that reduce the sum of square errors.

**Proposition 4.1.** Set \( M = \infty \). For any \( n \geq 1 \), the CAP algorithm converges.

**Proof.** CAP iteratively searches over the partition space. As there are a finite number of partitions and CAP only adopts a new partition when the SSE improves, CAP stops after a finite number of steps.

4.2 Consistency

Consistency is a difficult property to study with CAP. Methods that fit locally linear models to defined regions of the covariate space are consistent under fairly weak conditions, but the adaptive nature of the CAP partitions hinders analysis. We show consistency under fairly stringent conditions. We make the following assumptions:

i. \( f_0 \) is a univariate function that is strongly convex with parameter \( \gamma \) and Lipschitz continuous with parameter \( \zeta \),

ii. \( x_i \) are drawn uniformly from \( \mathcal{X} \),

iii. \( \mathcal{X} \subset \mathbb{R} \) is compact with length \( L \), and

iv. \( y_i = f_0(x_i) + \epsilon_i \), where \( \epsilon_i \) are i.i.d. with distribution \( N(0, \sigma^2) \).

To show consistency, we need two lemmas. First, we show that in the limit, a subset split when \( f_0 \) is strongly convex leads to a squared error reduction that is lower bounded by a function of the interval length. Second, we show that in the limit, taking the max over a set of linear least squares estimates only reduces the squared error.
**Lemma 4.2.** Let \([a, b]\) be an interval on \(\mathbb{R}\). Assume i. and ii. hold. Let \((\alpha_1, \beta_1)\) be the linear least squares estimate of \(f_0\) on \([a, (a + b)/2]\) and \((\alpha_2, \beta_2)\) the estimate on \(((a + b)/2, b]\) after \(n\) samples of \(x_i\). Then, for any \(\epsilon > 0\) and \((\alpha_0, \beta_0)\), there exists an \(N > 0\) such that for every \(n > N\),

\[
\int_a^b (f_0(x) - \alpha_0 - \beta_0 x)^2 - \left[ \int_a^{a+b/2} (f_0(x) - \alpha_1 - \beta_1 x)^2 + \int_{a+b/2}^b (f_0(x) - \alpha_2 - \beta_2 x)^2 \right] 
\geq \gamma^2 \frac{1}{768} (b - a)^4 - \epsilon. \tag{6}
\]

**Proof.** The smallest error gain is provided when \((\alpha_0, \beta_0)\) is the LSE of \(f_0\) on \([a, b]\) and when the curvature of \(f_0\) is as flat as possible, that is, \(f_0''(x) = \gamma\). In that case, by integration, \(f_0(x) = g(x) = \gamma x^2/2 + \eta x + \xi\). As \((\alpha_k, \beta_k)\) are all linear estimates, the constant \(\xi\) can be removed w.l.o.g. Likewise, \(g\) can be rotated to \(\gamma x^2/2\) on the interval \([a, b]\), where \(b' - a = (b - a)\sqrt{1 + \eta^2} \geq b - a\). Therefore, w.l.o.g., set \(g(x) = \gamma x^2/2\) and let \((a + b)/2 = 0\). Let \((\hat{\alpha}_0, \hat{\beta}_0), (\hat{\alpha}_1, \hat{\beta}_1)\) and \((\hat{\alpha}_2, \hat{\beta}_2)\) be the LSE of \(g\) on \([a, b], [a, 0]\) and \((0, b]\), respectively; let \((\hat{\alpha}_0, \hat{\beta}_0^m), (\hat{\alpha}_1, \hat{\beta}_1^m)\) and \((\hat{\alpha}_2, \hat{\beta}_2^m)\) be the LSE of \(g\) given \(x_{1n}\). A well known asymptotic result states that if the number of samples on \([a, 0]\) is greater than \((\epsilon/3)^2\), \(\|\hat{\beta}_k - \hat{\beta}_k^m\| < \epsilon/3\). Then for all \(n > N\) for a sufficiently large \(N\),

\[
\int_a^b (f_0(x) - \alpha_0 - \beta_0 x)^2 dx - \left[ \int_a^{a+b/2} (f_0(x) - \alpha_1 - \beta_1 x)^2 dx + \int_{a+b/2}^b (f_0(x) - \alpha_2 - \beta_2 x)^2 dx \right]
\geq \int_{-b/2}^{b/2} (g(x) - \hat{\alpha}_0 - \hat{\beta}_0 x)^2 dx - \left[ \int_{-b/2}^{b/2} (g(x) - \hat{\alpha}_1^m - \hat{\beta}_1^m x)^2 dx + \int_{b/2}^{b+b/2} (g(x) - \hat{\alpha}_2^m - \hat{\beta}_2^m x)^2 dx \right]
\geq \int_{-b/2}^{b/2} \left( \frac{\gamma}{2} x^2 - \hat{\alpha}_0 - \hat{\beta}_0 x \right)^2 dx - \left[ \int_{-b/2}^{b/2} \left( \frac{\gamma}{2} x^2 - \hat{\alpha}_1 - \hat{\beta}_1 x \right)^2 dx + \int_{b/2}^{b+b/2} \left( \frac{\gamma}{2} x^2 - \hat{\alpha}_2 - \hat{\beta}_2 x \right)^2 dx \right]
\geq \int_{-b/2}^{b/2} \left( \frac{\gamma}{2} x^2 - \frac{\gamma}{24} (b - a)^2 \right)^2 dx - \int_{b/2}^{b+b/2} \left( \frac{\gamma}{2} x^2 + \frac{\gamma}{48} (b - a)^2 - \frac{\gamma}{4} (b - a) x \right)^2 dx - \epsilon
\geq \frac{\gamma^2}{720} (b - a)^4 - \frac{\gamma^2}{11,520} (b - a)^4 - \epsilon
\geq \gamma^2 \frac{1}{768} (b - a)^4 - \epsilon.
\]

\[\blacksquare\]

**Lemma 4.3.** Let \([a, c]\) be an interval on \(\mathbb{R}\), and fix \(b\) such that \(a < b < c\). Assume i. and ii. hold. Let \((\alpha_1, \beta_1)\) be the linear least squares estimate of \(f_0\) on \([a, b]\), and let \((\alpha_2, \beta_2)\) be the estimate on \((b, c]\). Then, there exists an \(N > 0\) such that for all \(n > N\),

\[
\int_a^c (f_0(x) - \max_{k \in \{1, 2\}} \alpha_k + \beta_k x)^2 dx \tag{7}
\leq \int_a^b (f_0(x) - \alpha_1 - \beta_1 x)^2 dx + \int_b^c (f_0(x) - \alpha_2 - \beta_2 x)^2 dx.
\]
Proof. Let \([d_1, d_2]\) be the region of \([a, b]\) where \(\alpha_2 + \beta_2 x > \alpha_1 + \beta_1 x\) or the region of \((b, c]\) where \(\alpha_2 + \beta_2 x < \alpha_1 + \beta_1 x\). (Only one of these events may happen.) Since \(f_0\) is strongly convex and \((\alpha_k, \beta_k), k = 1, 2\) are the least squares linear estimates, the lines \(\alpha_1 + \beta_1 x\) and \(\alpha_2 + \beta_2 x\) intersect \(f_0\) exactly twice over \((a, b)\) and \((b, c]\), respectively (for large enough sample sizes). This implies that over the interval \([d_1, d_2]\), both \(\alpha_1 + \beta_1 x < f_0(x)\) and \(\alpha_2 + \beta_2 x < f_0(x)\). Hence Equation (7) holds.

Given these results, we are now ready to state the consistency theorem.

**Theorem 4.4.** Assume that i.–iv. hold. Assume that the number of knots \(D\) is odd and the upper iteration limit \(M = \infty\). Then, for every \(\delta > 0\), a \(\lambda > 0\) can be chosen such that

\[
\lim_{n \to \infty} \frac{1}{n} \sum_{i=1}^{n} \left( f_0(X_i) - \hat{f}_n(X_i) \right)^2 < \delta
\]

almost surely and the number of steps to convergence is less than or equal to \(1536\lambda^2 \zeta^6 / (\gamma^2 \delta^2)\).

**Proof.** Fix \(\delta > 0\), set \(\lambda = (n-1)\delta\). For a fixed \(n\), let \(m_n\) be the number of steps at which the CAP algorithm converged. Let \(A^{m_k}\) be the final set of covariate partitions. Let \(H_n\) be the longest of those partitions, that is, \(H_n = \arg \max_k |A^{m_k}|\). Let \(C_{H_n}^{m_k}\) be the associated observation subset. For convergence, we aim to show that for a fixed \(\rho > 0\), \(|H_n| > \rho\) at most finitely often for \(n = 1, 2, \ldots\)

Suppose that \(|H_n| > \rho\) infinitely often. Then there is a subsequence \((n_k)_{k=1}^\infty\) such that for every \(k\), \(|H_{n_k}| > \rho\). We now consider what happens if we try to split \(H_{n_k}\) into two subsets, and fit estimators within those. Since \(D\) is odd, the error gain is at least good as the gain when \(H_{n_k}\) is split at the midpoint into \(H_{n_k}^-\) and \(H_{n_k}^+\). Let \(C_{H_n}^{m_k}\) and \(C_{H_n}^{m_k}\) be the covariate subsets associated with \(H_{n_k}^-\) and \(H_{n_k}^+\), respectively. By iii., the law of large numbers and Lemma 4.2, taking \(\epsilon = \rho^4 \gamma^2 / 1536\),

\[
\lim_{k \to \infty} \frac{1}{|C_H^{m_k}|} \sum_{i \in C_H^{m_k}} (f_0(x_i) - \alpha_H - \beta_H x_i) - \left[ \frac{1}{|C_H^-^{m_k}|} \sum_{i \in C_H^-^{m_k}} (f_0(x_i) - \alpha_{H^-} - \beta_{H^-} x_i) + \frac{1}{|C_H^+^{m_k}|} \sum_{i \in C_H^+^{m_k}} (f_0(x_i) - \alpha_{H^+} - \beta_{H^+} x_i) \right] \geq \gamma^2 \frac{1}{1536} \rho^4.
\]

By Lemma 4.3, when the max is taken over the resulting hyperplanes to produce the CAP estimator, \(\rho^4 \gamma^2 / 1536\) still serves as a lower bound. Since the tolerance parameter \(\lambda = 0\) and \(M = \infty\), \(|H_n| > \rho\) at most finitely often.

Now we compute an upper bound on \(\hat{f}_n\). Using the law of large numbers,

\[
\lim_{n \to \infty} \frac{1}{n} \sum_{i=1}^{n} \left( f_0(x_i) - \hat{f}_n(x) \right)^2 \leq L \rho^2 \zeta^2. \tag{9}
\]

If \(\rho\) is chosen such that

\[
\rho < \sqrt{\frac{\delta}{L \zeta^2}},
\]

then Equation (8) holds.
Now we place an upper bound on the number of steps until convergence. The upper bound on mean squared error is given in Equation (9). If CAP has not terminated, at least one subset has length of at least $\sqrt{\delta/(L\zeta^2)}$. Therefore, a mean squared error gain of at least $\rho^4\gamma^2/1536$ is generated each step, so at most $1536L^3\zeta^6/\gamma^2\delta^2$ steps are needed until CAP terminates.

Empirical results suggest that CAP is consistent under a much broader set of conditions, such as functions in multiple dimensions and sampling under a non-uniform, non-atomic measure.

5 Applications

In this section, we apply CAP to synthetic convex regression problems. We then use it for value function approximation when pricing American basket options.

5.1 Synthetic Regression Problems

Here we apply CAP to two synthetic regression problems. The first has a non-additive structure and high levels of covariate interaction, while the second has a simple structure that is not well-described by cardinal directions.

**Problem 1**  Here $x \in \mathbb{R}^5$. Set

$$ y = (x_1 + 0.5x_2 + x_3)^2 - x_4 + 0.25x_5^2 + \epsilon, $$

where $\epsilon \sim N(0,1)$. The covariates are drawn from a 5 dimensional standard Gaussian distribution, $N_5(0, I)$.

**Problem 2**  Here $x \in \mathbb{R}^{10}$. Set

$$ y = \exp(x^T p) + \epsilon, $$

where $p$ was randomly drawn from a Dirichlet$(1, \ldots, 1)$ distribution,

$$ p = (0.0680, 0.0160, 0.1707, 0.1513, 0.1790, 0.2097, 0.0548, 0.0337, 0.0377, 0.0791)^T. $$

We set $\epsilon \sim N(0, 0.1^2)$. The covariates are drawn from a 10 dimensional standard Gaussian distribution, $N_{10}(0, I)$.

Results for CAP, CART, MARS, Gaussian processes and the least squares convex estimator are given in Table 2. CAP had $M = 100$, $\lambda = n \times 0.01$ for Problem 1, $\lambda = n \times 0.001$ for Problem 2 and $D = 10$ and was run in Matlab. CART was run through the Matlab function `classregtree`. MARS was run through the Matlab package ARESLab. Gaussian processes were run with the Matlab package `gpml`. The least squares estimate was run with `cvx`; with more than 200 observations, the optimization algorithm did not converge.

Problem 2 demonstrates the exactly how well convexity acts as a regularizer. When training on 10,000 observations, CAP produced 5 hyperplanes. Figure 7 shows the component values for each of these hyperplanes, along with the linear transformation $p$. Notice that each of the hyperplanes is close to $p$ multiplied by a scalar value. Even though no other constraints besides convexity were included, CAP finds the underlying transformation almost exactly.

On Problems 1 and 2, CAP provides a substantial performance boost over its competitors, both in terms of accuracy and computational time. Stagewise variable selection algorithms, like MARS and CART, performed relatively poorly on both problems. Gaussian processes did poorly
<table>
<thead>
<tr>
<th>Method</th>
<th>Unit 1</th>
<th>Unit 2</th>
<th>Unit 3</th>
<th>Unit 4</th>
<th>Unit 5</th>
<th>Unit 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAP</td>
<td>2.0068</td>
<td>1.8935</td>
<td>0.3018</td>
<td>0.1396</td>
<td>0.1044</td>
<td>0.0480</td>
</tr>
<tr>
<td>LSE</td>
<td>4.0174</td>
<td>4.4370</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>GP</td>
<td>7.6612</td>
<td>6.2974</td>
<td>4.4793</td>
<td>3.5518</td>
<td>2.3539</td>
<td>0.8549</td>
</tr>
<tr>
<td>MARS</td>
<td>6.9053</td>
<td>7.2590</td>
<td>6.7261</td>
<td>6.2774</td>
<td>6.0555</td>
<td>5.9023</td>
</tr>
<tr>
<td>CART</td>
<td>10.8879</td>
<td>8.6618</td>
<td>5.7306</td>
<td>4.3998</td>
<td>3.7448</td>
<td>2.4234</td>
</tr>
</tbody>
</table>

Table 2: CAP, Least Squares Estimator (LSE), Gaussian Processes (GP), MARS and CART were run on Problems 1 and 2. Errors are in distance to the true mean function. The lowest error values are bolded.
Figure 7: The x-axis shows the dimension number and the y-axis shows the value for each component, $\beta_{k,i}$ for the 5 hyperplanes produced by CAP for Problem 2 using 10,000 training samples. Each line represents values for a hyperplane $\beta_k$. The solid line with diamonds is $p$; notice that each $\beta_k$ is almost a multiple of $p$.

on Problem 1, due to its relatively high noise and complicated structure. They did much better on Problem 2, as the covariance function can easily capture the axis rotation. However, GPs have an $O(n^3)$ computational requirement, which does not scale well to larger datasets; the GP did not converge when $n = 10,000$ on Problem 2. Although one could use a low rank approximation, such as predictive processes (Tokdar 2007, Banerjee et al. 2008), such approaches are still computationally intensive and we do not consider them further as they are unconstrained. CAP outperforms the LSE in three of the four cases where the LSE converged. It is unclear whether the LSE is overfitting or whether the solver used simply provides poor solutions for problems of this size.

5.2 Pricing Stock Options

Approximate dynamic programming is a modeling method for sequential decision problems, where a value-to-go function is approximated for each state rather than explicitly determined, as it is in traditional dynamic programming. Often value functions are known to be convex or concave in the state variable; this is common in options pricing, portfolio optimization and logistics problems. In some situations, such as when a linear program is solved each time period to determine an action, a convex (and piecewise linear) value function is required for computational tractability. CAP holds great promise for value function approximation in these problems.

To give a simple example for value function approximation, we consider pricing American basket options on the average of $N$ underlying assets. Options give the holder the right—but not the obligation—to buy the underlying, in this case the average of $N$ assets, for a predetermined strike price $K$. In an American option, this can be done at any time between the issue date and the maturity date, $T$. However, American options are notoriously difficult to price, particularly when the underlying asset base is large.
A popular method for pricing American options involves using approximate dynamic programming where continuation values are approximated via regression (Carriere 1996, Tsitsiklis & Van Roy 1999, 2001, Longstaff & Schwartz 2001). We summarize these methods as follows; see Glasserman (2004) for a more thorough treatment. The underlying assets are assumed to have the sample path \(\{X_1, \ldots, X_T\}\), where \(X_t = \{S_1(t), \ldots, S_N(t)\}\) is the set of securities at time \(t\). At each time \(t\), a continuation value function, \(\bar{V}_t(X_t)\), is estimated by regressing a value function for the next time period, \(\bar{V}_{t+1}(X_{t+1})\), on the current state, \(X_t\). The continuation value is the value of holding the option rather than exercising given the current state of the assets. The value function is defined to be the max of the current exercise value and the continuation value. Options are exercised when the current value of the option is greater than or equal to the continuation value.

The procedure to estimate the continuation values is as follows (as summarized in Glasserman (2004)):

0. Define basket payoff function,

\[
h(X_t) = \max \left\{ \frac{1}{N} \sum_{k=1}^{N} S_k(T) - K, 0 \right\}.
\]

1. Sample \(M\) independent paths, \(\{X_1, \ldots, X_T\}\), \(j = 1, \ldots, M\).

2. At time \(T\), set \(\bar{V}_T(X_{Tj}) = h(X_{Tj})\).

3. Apply backwards induction: for \(t = T - 1, \ldots, 1\),
   - given \(\{\bar{V}_{t+1}(X_{t+1})\}_{j=1}^{M}\), regress on \(\{X_{tj}\}_{j=1}^{M}\) to get continuation value estimates \(\{\bar{C}_t(X_{tj})\}_{j=1}^{M}\).
   - set value function,
     \[
     \bar{V}_t(X_{tj}) = \max \left\{ h(X_{tj}), \bar{C}_t(X_{tj}) \right\}.
     \]

We use the value function defined by Tsitsiklis & Van Roy (1999).

To generate a lower bound estimate for the option value, the regression for the continuation values is used with new testing sample paths. Note that the estimated continuation values determine an exercise policy for the option. As this is by definition a suboptimal exercise policy, it provides a lower bound on the value of the optimal policy, and hence the value of the option. To price the option under the approximate policy, we proceed forward in time. At each time step, the current exercise value is compared to the continuation value until either the option is exercised or the time horizon is reached. The values for each sample path are then recorded and averaged to generate a lower bound on the true option value.

In previous literature, \(\{C_t(X_{tj})\}_{j=1}^{M}\) has been estimated by regression splines for a single underlying asset (Carriere 1996), or least squares linear regression on a set of basis functions (Tsitsiklis & Van Roy 1999, Longstaff & Schwartz 2001, Glasserman 2004). Regression on a set of basis functions becomes problematic when \(X_{tj}\) is defined over moderate to high dimensional spaces. Well-defined sets of bases such as radial basis functions and polynomials require an exponential number of functions to span the space, while manually selecting basis functions can be quite difficult. Instead, we propose using CAP; the expected continuation values for a basket option are convex in the asset space. The use of convex structure produces a better estimate over high-dimensional spaces than a naive regression method, while avoiding the pitfalls of selecting basis functions in moderate to high dimensions.

We compared value function regression methods as follows. We simulated 10,000 training samples for a 3-month American basket option with a varying number of underlying assets, \(N\). All asset
<table>
<thead>
<tr>
<th>Assets</th>
<th>Method</th>
<th>Policy Value</th>
<th>Upper Bound</th>
<th>Duality Gap</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CAP</td>
<td><strong>28.0814</strong></td>
<td>30.2054</td>
<td>7.03%</td>
<td>6.2 sec</td>
</tr>
<tr>
<td></td>
<td>CART</td>
<td>11.2093</td>
<td>62.89%</td>
<td>26.9 sec</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Least Squares</td>
<td>28.0402</td>
<td>7.17%</td>
<td>1 sec</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ridge</td>
<td>27.8628</td>
<td>7.76%</td>
<td>9.9 sec</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CAP</td>
<td><strong>24.0101</strong></td>
<td>25.6928</td>
<td>6.55%</td>
<td>14.7 sec</td>
</tr>
<tr>
<td></td>
<td>CART</td>
<td>8.8761</td>
<td>65.45%</td>
<td>34.2 sec</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Least Squares</td>
<td>23.6958</td>
<td>7.77%</td>
<td>0.2 sec</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ridge</td>
<td>23.6033</td>
<td>8.13%</td>
<td>16.4 sec</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>CAP</td>
<td><strong>20.7311</strong></td>
<td>21.6261</td>
<td>4.14%</td>
<td>61.3 sec</td>
</tr>
<tr>
<td></td>
<td>CART</td>
<td>6.4032</td>
<td>70.39%</td>
<td>54.3 sec</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Least Squares</td>
<td>20.4506</td>
<td>5.44%</td>
<td>1.0 sec</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ridge</td>
<td>20.4081</td>
<td>5.22%</td>
<td>48.1 sec</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>CAP</td>
<td><strong>19.0850</strong></td>
<td>21.0776</td>
<td>9.45%</td>
<td>297.3 sec</td>
</tr>
<tr>
<td></td>
<td>CART</td>
<td>5.3364</td>
<td>74.68%</td>
<td>83.4 sec</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Least Squares</td>
<td>17.9135</td>
<td>15.01%</td>
<td>8.3 sec</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ridge</td>
<td>18.7792</td>
<td>10.90%</td>
<td>164.4 sec</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>CAP</td>
<td><strong>18.8339</strong></td>
<td>21.1540</td>
<td>10.97%</td>
<td>680.0 sec</td>
</tr>
<tr>
<td></td>
<td>CART</td>
<td>4.8001</td>
<td>74.68%</td>
<td>83.4 sec</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Least Squares</td>
<td>14.7389</td>
<td>15.01%</td>
<td>30.8 sec</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ridge</td>
<td>18.4423</td>
<td>10.90%</td>
<td>564.7 sec</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>CAP</td>
<td><strong>18.2985</strong></td>
<td>19.2050</td>
<td>4.72%</td>
<td>1790.7 sec</td>
</tr>
<tr>
<td></td>
<td>CART</td>
<td>4.5142</td>
<td>76.49%</td>
<td>141.3 sec</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Least Squares</td>
<td>11.4908</td>
<td>40.12%</td>
<td>55.9 sec</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ridge</td>
<td>17.7775</td>
<td>7.43%</td>
<td>1129.8 sec</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>CAP</td>
<td><strong>17.5320</strong></td>
<td>19.7415</td>
<td>11.91%</td>
<td>3689.1 sec</td>
</tr>
<tr>
<td></td>
<td>CART</td>
<td>4.3842</td>
<td>77.79%</td>
<td>193.8 sec</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Least Squares</td>
<td>7.2953</td>
<td>63.05%</td>
<td>167.2 sec</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ridge</td>
<td>16.4591</td>
<td>16.63%</td>
<td>4308.0 sec</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: CAP, CART, least squares and ridge regularized least squares were compared for pricing American basket options. Lower bounds were generated for each method by implementing the policy given by the value function. Computational times for each method are given in seconds. Upper bounds were generated using Haugh & Kogan (2004). Duality gaps were calculated as a percentage of the upper bound. The best lower bounds for each basket size are bolded.

Sample paths were generated by a geometric Brownian motion with a drift of 0.05 and a volatility of 0.10. All assets had correlation 0.5 and starting value 100. The option had strike price 110. Lower bounds were generated on 50,000 testing sample paths. An upper bound was generated using the dual martingale methods of Haugh & Kogan (2004) from value functions generated using basis functions $(1, Y, Y^2, Y^3, h(Y))$, where $Y_t = 1/N \sum_{i=1}^{N} X_i(t)$ with 5,000 samples. Duality gaps were generated using these values and the lower bounds for each method. All values are in discounted dollars. All computations were run in Matlab on a 2.66 GHz Intel i7 processor.

We compared the following methods: CAP, with $D = 10$, $M = 15$ and $\lambda = 0$; CART using the Matlab function `classregtree`; least squares using the polynomial basis functions

$$ (1, S_i(t), S_i^2(t), S_i^3(t), S_i(t)S_j(t), h(X_t)), \quad i = 1, \ldots, N, \; j \neq i; $$

ridge regression on the same basis functions with ridge parameter chosen by cross-validation each time period from values between $10^{-3}$ and $10^5$.

The basket options were simple—all assets had the same drift, volatility, covariance and starting value—which should be one of the best scenarios for the basis functions used. However, CAP outperformed the other methods in all settings. As the number of assets grew, the vanilla least squares
implementation began to overfit the data. Regularization removed most overfitting issues, but produced computational times similar to those of CAP. Additionally, it still required specification of basis functions.

6 Discussion

Multidimensional convex regression has substantial applications in many fields, yet there are currently no methods to solve moderate to large scale problems. This paper proposes a new, non-parametric algorithm, Convex Adaptive Partitioning (CAP), for use in multidimensional convex regression. Empirical results suggest that CAP is both fast and accurate, even on large scale problems. CAP combines the speed and robustness of stagewise variable selection models, like CART and MARS, with the regularization implicitly provided by convexity constraints. Theoretical results show consistency for CAP in the univariate case, with empirical results suggesting consistency in multiple dimensions.

By providing a computationally efficient estimator for multidimensional convex regression, we hope that CAP opens up many more areas for study, including log-concave density estimation for large datasets, value function approximation under shape restrictions and many others.

References


