Penalised Additive Least Squares Models for High Dimensional Nonparametric Regression and Function Selection

Kirthevasan Kandasamy
Calvin McCarter
Carnegie Mellon University, Pittsburgh, PA, USA

Abstract
We describe additive kernel regression (Add-KR), a generalisation of kernel least squares methods for nonparametric regression. Nonparametric methods typically allow us to consider a richer class of functions over parametric methods. However, unlike their parametric counterparts they suffer from high sample complexity in high dimensions and cannot be used to identify structure in the function. A common assumption in high dimensional regression models is to assume that the function is additive. In this work, we leverage this assumption, but considerably generalise existing additive models. We propose a convex optimisation objective for our problem and optimise it using Block Coordinate Gradient Descent. We demonstrate that Add-KR significantly outperforms existing algorithms for nonparametric regression on moderate to high dimensional problems and can be used to identify and exploit structure in the function.

1. Introduction

Given data \((X_i, Y_i)_{i=1}^n\) where \(X_i \in \mathbb{R}^D\), \(Y \in \mathbb{R}\) and \((X_i, Y_i) \sim P\), the goal of least squares regression methods is to estimate the regression function \(f(x) = \mathbb{E}_P[Y|X = x]\). A popular method for regression is linear regression which models \(f\) as a linear combination of the variables \(x\), i.e. \(f(x) = w^T x\) for some \(w \in \mathbb{R}^D\). Such methods are computationally simple and have desirable statistical properties when the problem meets the assumption. However, they are generally too restrictive for many real problems. Nonparametric regression refers to a suite of regression methods that only assume smoothness of \(f\). In particular, they do not assume any parametric form for \(f\). As such, they present a more powerful and compelling framework for regression.

While nonparametric methods consider a richer class of functions, they suffer from severe drawbacks. Nonparametric regression in high dimensions is an inherently difficult problem with known lower bounds depending exponentially in dimension (Györfi et al., 2002). With rare exceptions, nonparametric methods typically work only under at most 4 - 6 dimensions. In addition they typically cannot be used to identify structure in the problem. For instance, in the parametric setting, algorithms such as the LASSO and group LASSO can be used to identify a sparse subset of variables/groups to describe the function. In this project we intend to make progress in both these fronts by treating the estimate of the function as an additive function -- \(f(\cdot) = f^{(1)}(\cdot) + f^{(2)}(\cdot) + \cdots + f^{(M)}(\cdot)\).

Our methods are based on Kernel Ridge Regression (KRR). We minimize the squared-error loss with an RKHS norm penalty to enforce smoothness and identify structure. This leads to a convex objective function where the number of parameters is the product of the number of samples and the number of basis functions.

We present two concrete applications for our framework. The first is on nonparametric regression in high dimensions. Using additive models is fairly standard in high dimensional regression literature (Hastie & Tibshirani, 1990; Ravikumar et al., 2009; Lafferty & Wasserman, 2005). When the true underlying function \(f\) exhibits additive structure, using an additive model for estimation is understandably reasonable. However, even when \(f\) is not additive, using an additive model has its advantages. It is a well understood notion in Statistics that when we only have a few samples, using a simpler model to fit our data may give us a better tradeoff for estimation error against approximation error. This is because additive functions are statistically simpler than more general (non-additive) functions. Typically, in most nonparametric regression methods using kernels such as the Nadaraya-Watson estimator and Kernel Ridge Regression, the bias-variance tradeoff is managed via the bandwidth of the kernel. Using an additive model provides another “knob” to control this tradeoff and provides significant gains in high dimensional regres-
Additive Least Squares Regression

2. Additive Kernel Regression

2.1. Problem Statement & Notation

Let \( f : \mathcal{X} \to \mathbb{R} \) be the regression function \( f(\cdot) = \mathbb{E}[Y|X = \cdot] \). Here \( \mathcal{X} \ni x = [x_1, \ldots, x_D] \in \mathbb{R}^D \) and \( \mathcal{X} \subset \mathbb{R}^D \). We have data \( \{X_i, Y_i\}_{i=1}^n \) and wish to obtain an estimate \( \hat{f} \) of \( f \). In this work, we seek an additive approximation to the function. That is, \( \hat{f} \) can be expressed as,

\[
\hat{f}(x) = f^{(1)}(x) + f^{(2)}(x) + \cdots + f^{(M)}(x) \tag{1}
\]

where each \( f^{(j)} : \mathcal{X} \to \mathbb{R} \).

The work in Hastie & Tibshirani (1990) treats \( \hat{f} \) as a sum of one dimensional components. In Equation (1) this corresponds to setting \( M = D \) and have each \( f^{(j)} \) act on only the \( j \)-th coordinate. In this work, we would like to be more expressive than this model. We will consider additive models on more than just one dimension and more importantly allows for overlap between the groups. For e.g. \( \hat{f}(x_1, x_2, x_3) = f^{(1)}(x_1) + f^{(2)}(x_1, x_2) + f^{(3)}(x_2, x_3) \).

Ravikumar et al. (2009) treat \( \hat{f} \) as a sparse combination of one dimensional functions. While this is seemingly restrictive than (Hastie & Tibshirani, 1990), the sparse approximation may provide favourable bias-variance tradeoffs in high dimensions. Drawing inspiration from this, we will also consider models where \( M \) is very large and seek a sparse collection of groups to approximate the function - i.e. \( f^{(j)} = 0 \) for several \( j \).

2.2. Additive Least Squares Regression via Kernels

One of several ways to formulate a nonparametric regression problem is to minimise an objective of the form

\[
J(f) = \sum_{i=1}^n \ell(f(X_i), Y_i) + \lambda \xi(f) \quad \text{over a nonparametric class of functions } \mathcal{F}.
\]

Here \( \ell \) is a loss function and \( \xi \) is a term that penalises the complexity of \( f \). Several nonparametric regression problems such as Gaussian processes, smoothing splines and natural splines can be formulated this way. Or particular interest to us is Kernel Ridge Regression (KRR) which uses a positive semidefinite kernel \( k : \mathcal{X} \times \mathcal{X} \to \mathbb{R} \) (Scholkopf & Smola, 2001) and takes \( \mathcal{F} \) is taken to be the reproducing kernel Hilbert space (RKHS) \( \mathcal{H}_k \) corresponding to \( k \). \( \xi \) is taken to be the squared RKHS norm of \( f \) and \( \ell \) the squared error loss. Accordingly, KRR is characterised via the optimisation objective,

\[
\hat{f} = \arg\min_{f \in \mathcal{H}_k} \sum_{i=1}^n (Y_i - f(X_i))^2 + \lambda \|f\|_{\mathcal{H}_k}^2.
\]

However, like most nonparametric regression models, KRR suffers from the curse of dimensionality. To obtain an additive approximation we consider \( M \) kernels \( k^{(j)} \) and their associated RKHSs \( \mathcal{H}_{k^{(j)}} \). In equation (1), we will aim for \( f^{(j)} \in \mathcal{H}_{k^{(j)}} \). Accordingly we consider an optimisation

The remainder of this paper is organised as follows. In Section 2 we present the Add-KR procedure and the associated optimisation objective. In Section 3 we present experiments on synthetic and real datasets in both settings described above.
problem of the following form where we jointly optimise over \( \hat{f}^{(1)}, \ldots, \hat{f}^{(M)} \),

\[
\{ \hat{f}^{(j)} \}_{j=1}^{M} = \arg\min_{f^{(j)} \in \mathcal{H}_{k^{(j)}}, j=1,\ldots,M} F \left( \{ f^{(j)} \}_{j=1}^{M} \right)
\]

where,

\[
F \left( \{ f^{(j)} \}_{j=1}^{M} \right) = \frac{1}{2} \sum_{i=1}^{n} \left( Y_i - \sum_{j=1}^{M} f^{(j)}(x^{(j)}) \right)^2 + \lambda \sum_{j=1}^{M} \| f^{(j)} \|_{\mathcal{H}_{k^{(j)}}}.
\]

Our estimate for \( f \) is then \( \hat{f} = \sum_{j} \hat{f}^{(j)}(\cdot) \).

Via a representer theorem like argument it is straightforward to show that \( f^{(j)} \) will be in the linear span of the reproducing kernel maps of the training points \( X_i \) — i.e. \( f^{(j)}(\cdot) = \sum_j \alpha^{(j)} k^{(j)}(\cdot, X_i) \). Then, the \( j^\text{th} \) term in the second summation can be written as \( \alpha^{(j)} \top K^{(j)} \alpha^{(j)} \), where \( K^{(j)} \in \mathbb{R}^{n \times n} \) \( \forall j \) such that \( K^{(j)}_{i\cdot} = k^{(j)}(X_i, X_j) \).

Further, if we take the \( \| \cdot \| \) to be the RKHS norms, the objective can be written as,

\[
F_1(\alpha) = \frac{1}{2} \| Y - \sum_{j=1}^{M} K^{(j)} \alpha^{(j)} \|_2^2 + \lambda \sum_{j=1}^{M} \sqrt{\alpha^{(j)} \top K^{(j)} \alpha^{(j)}}.
\]

Here \( \alpha^{(j)} \in \mathbb{R}^n \) \( \forall j \), \( \alpha = [\alpha^{(1) \top}, \ldots, \alpha^{(M) \top}] \top \in \mathbb{R}^{nM} \) and \( Y = [Y_1, \ldots, Y_n] \top \in \mathbb{R}^n \). Given the solution to the above, our estimate is obtained via \( \hat{f}(\cdot) = \sum_{j=1}^{M} \sum_{i=1}^{n} \alpha^{(j) \top} k^{(j)}(\cdot, X_i) \).

Equation (3) will be the convex optimisation problem in our algorithm. We call this algorithm Additive Kernel Regression (Add-KR).

2.3. Applications

We propose two concrete applications for the additive regression framework proposed above. Our choices of kernels \( k^{(j)}, j = 1 \ldots M \) are different in both settings.

**Application 1 (High Dimensional Regression):** The first is when we wish to reduce the statistical complexity of the function to be learned in large \( D \). A kernel directly defined on \( D \) dimensions is complex since it allows for interactions of all \( D \) variables. We may reduce the complexity of the kernel by constraining how these variables interact. Here we consider kernels of the form,

\[
k^{(1)}(x, x') = \sum_{1 \leq i < D} k_i(x_i, x'_i)
\]

\[
k^{(2)}(x, x') = \sum_{1 \leq i < j \leq D} k_i(x_i, x'_i) k_j(x_j, x'_j)
\]

\[
k^{(M)}(x, x') = \sum_{1 \leq i_1 < \ldots < i_M \leq D} \prod_{d=1}^{M} k_{i_d}(x_{i_d}, x'_{i_d})
\]

Here \( k_i : \mathbb{R} \times \mathbb{R} \to \mathbb{R} \) is a base kernel acting on one dimension. \( k^{(j)} \) has \( \binom{D}{j} \) terms and exhaustively computing all of them is computationally intractable. Fortunately, by observing that the \( j^\text{th} \) kernel is just the \( j^\text{th} \) elementary symmetric polynomial (ESP) on the base kernel values we may use the Newton Girard formula to efficiently compute them recursively. Precisely, by denoting \( \kappa_s = \sum_{i=1}^{D} (k_i(x_i, x'_i))^s \) we have,

\[
k^{(j)}(x, x') = \frac{1}{j} \sum_{d=1}^{j} (-1)^{d-1} \kappa_{d-j} k^{(d)}(x, x')
\]

Computing the \( M \) kernels this way only requires \( O(\binom{DM^2}{M}) \) computation. We call these the ESP Kernels. A similar kernel using a similar trick for computing it was used by Duvenaud et al. (2011).

**Application 2 (Function Selection):** The second setting is when we are explicitly searching for a sparse subset of functions to explain the data. For instance, in neurological and genomics models, while the function of interest has several variables, the interactions are sparse and of lower order. For example, a function of 4 variables may take the form

\[
f(x) = f^{(1)}(x_1) + f^{(2)}(x_2, x_3) + f^{(3)}(x_1, x_4)
\]

That is, the function decomposes as a sum of functions acting on small groups of variables. Given a large set of candidate groups, the task at hand is to recover the groups and the individual functions acting on those groups. In this setting, \( M \) and our RKHSs are determined by the problem — \( \mathcal{H}_{k^{(j)}} \) contains functions on the variables belonging to the \( j^\text{th} \) candidate group. This idea was first explored by (Bach, 2008) using a slightly different objective.

2.4. Implementation

We now describe the implementation details of the above algorithm. Let the Cholesky decomposition of \( K^{(j)} \) be \( K^{(j)} = L^{(j)} L^{(j) \top} \). Denote \( \beta^{(j)} = L^{(j) \top} \alpha^{(j)} \).

Then, our objective can be written in terms of \( \beta = [\beta^{(1) \top}, \ldots, \beta^{(M) \top}] \) as,

\[
F_2(\beta) = \frac{1}{2} \| Y - \sum_{j=1}^{M} L^{(j)} \alpha^{(j)} \|_2^2 + \lambda \sum_{j=1}^{M} \| \beta^{(j)} \|_2
\]

The objective, in the above form is well studied in optimisation literature as the group LASSO. When the number of parameters for each group are small, which is typically the case in group LASSO problems, block coordinate descent...
Additive Least Squares Regression

(BCD) is believed to be the state of the art solver. However, in our case the number of parameters per group is large – equal the number of samples \( n \). In this regime BCD is slow since it requires a matrix inversion at each step. In particular, we found that Block Coordinate Gradient Descent (BCGD) and Alternating Direction Method of Multipliers (ADMM) significantly outperformed BCD in our experiments. In fact, we experimented with several optimisation methods to minimise the objective which included Subgradient method, Proximal Gradient method (with and without acceleration), BCD, BCGD and ADMM. Figure 1 depicts the empirical convergence of these methods on a synthetic problem. In all our experiments, we use BCGD.

The penalty term \( \lambda \) was chosen using 5-fold cross validation. Our implementation first solves for the largest \( \lambda \) value. For successive \( \lambda \) values, we initialise BCGD at the solution of the previous \( \lambda \) value. This warm starts procedure significantly speeds up the running time of the entire training procedure.

3. Experiments

3.1. Application 1: ESP Kernels for High Dimensional Regression

In our implementations of the ESP kernels, for the one dimensional base kernel we use the RBF kernel \( k_i(x, x') = \exp((-x - x')^2/h^2) \) with bandwidth \( h \). Since cross validating on all the kernel bandwidths is expensive, we set it to \( h = c\sigma n^{-0.2} \). This follows other literature (Györfi et al., 2002; Tsybakov, 2008; Ravikumar et al., 2009) using similar choices for kernel bandwidths. The constant \( c \) was hand tuned – we found that the performance of our methods was robust to choices of \( c \) between 5 and 40. The value of \( M \) was also hand tuned and set to \( M = \min(D/4, 10) \).

We compare Add-KR against kernel ridge regression (KRR), Nadaraya Watson regression (NW), locally linear regression (LL), locally quadratic regression (LQ), Gaussian process regression (GP), \( k \) nearest neighbors regression (kNN) and support vector regression (SVR). For GP and SVR we use the implementations in Rasmussen & Nickisch (2010); Chang & Lin (2011) respectively. For the other methods, we chose hyper parameters using 5-fold cross validation. The Additive Gaussian process model of Duvenaud et al. (2011) is also a candidate but we found that inference was extremely slow beyond a few hundred training points (For e.g. it took > 50 minutes with 600 points whereas Add-KR ran in under 4 minutes).

First, we construct a smooth synthetic 20 dimensional function. We train all methods on \( n \) training points where \( n \) varies from 100 to 1100 and test on 1000 points sampled independently. The results are shown in Figure 2(a). Add-KR outperforms all other methods. We suspect that NW,
Additive Least Squares Regression

<table>
<thead>
<tr>
<th>Dataset ((D, n))</th>
<th>Add-KR</th>
<th>KRR</th>
<th>kNN</th>
<th>NW</th>
<th>LL</th>
<th>LQ</th>
<th>GP</th>
<th>SVR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speech (21, 520)</td>
<td>0.02269</td>
<td>0.02777</td>
<td>0.09348</td>
<td>0.11207</td>
<td>0.03373</td>
<td>0.02407</td>
<td>0.02531</td>
<td>0.22431</td>
</tr>
<tr>
<td>Music (90, 1000)</td>
<td>0.91627</td>
<td>0.91922</td>
<td>1.00001</td>
<td>1.05745</td>
<td>1.25805</td>
<td>1.06482</td>
<td>0.94329</td>
<td>1.07009</td>
</tr>
<tr>
<td>Tele-motor (19, 300)</td>
<td>0.00659</td>
<td>0.006488</td>
<td>0.13957</td>
<td>0.20119</td>
<td>0.09455</td>
<td>0.08774</td>
<td>0.06678</td>
<td>0.38038</td>
</tr>
<tr>
<td>Housing (12, 256)</td>
<td>0.31285</td>
<td>0.35947</td>
<td>0.43619</td>
<td>0.42087</td>
<td>0.31219</td>
<td>0.35061</td>
<td>0.67566</td>
<td>1.15272</td>
</tr>
<tr>
<td>Blog (91, 700)</td>
<td>1.43288</td>
<td>1.53227</td>
<td>1.73545</td>
<td>1.49305</td>
<td>1.69234</td>
<td>1.71321</td>
<td>1.64429</td>
<td>1.66705</td>
</tr>
<tr>
<td>Forest Fires (10, 210)</td>
<td>0.30675</td>
<td>0.32618</td>
<td>0.40565</td>
<td>0.37199</td>
<td>0.35462</td>
<td>0.33881</td>
<td><strong>0.29038</strong></td>
<td>0.70154</td>
</tr>
<tr>
<td>Propulsion (15, 400)</td>
<td>0.04167</td>
<td>0.01396</td>
<td>0.15760</td>
<td>0.11237</td>
<td>0.182345</td>
<td>0.19212</td>
<td><strong>0.00355</strong></td>
<td>0.74511</td>
</tr>
</tbody>
</table>

Table 1. The test set errors of all methods on 7 datasets from the UCI repository. The dimensionality and number of training points is indicated next to the dataset. The best method(s) for each dataset are in bold. Add-KR performs best in most of the datasets and is within the top 3 in all of the datasets. In the Forest Fires dataset it is only slightly worse than GP. In the Propulsion dataset, GP significantly outperforms all other methods.

LL and kNN perform very poorly since they make very weak smoothness assumptions about the function.

Next, we compare all methods on 7 moderate to high dimensional datasets from the UCI repository. All inputs and labels were preprocessed to have zero mean and standard deviation 2. We split the datasets into roughly two halves for training and testing. The results are given in Table 1. Add-KR outperforms all alternatives in most cases.

### 3.2. Setting 2: Function Selection

In this section, we study the ability of our method to recover the true function. We use RBF kernels on each group by setting kernel bandwidths for each dimension as same as explained above.

First, we conduct the following synthetic experiment. We generate 600 observations from the following 50-dimensional additive model:

\[
y_i = f_1(x_{i1}) + f_2(x_{i2}) + f_3(x_{i3}) + f_4(x_{i4}) + f_5(x_{i5}x_{i6}) + f_6(x_{i7}x_{i8}) + f_7(x_{i9}x_{110}) + f_8(x_{i11}x_{i12}) + \epsilon_i
\]

where,

\[
f_1(x) = -2\sin(2x), \quad f_2(x) = x^2 - \frac{1}{3},
\]

\[
f_3(x) = x - \frac{1}{2}, \quad f_4(x) = e^{-x} + e^{-1} - 1
\]

with noise \(\epsilon_i \sim \mathcal{N}(0, 1)\). Thus, 46 out of 50 individual features are irrelevant, and 1221 out of 1225 pairwise features are irrelevant. As candidates, we use all functions of first and second order interactions – i.e the kernels characterizing our RKHSs are of the form \(k(x_i, x_i')\) for \(i = 1, \ldots, 50\) and \(k(x_i, x_j)k(x_j, x_j)\) for \(1 \leq i < j \leq 50\). Therefore, in this experiment \(M = 1275\).

We plot the solution path for two independent datasets. The plots give the RKHS norm of the function on each kernel \(\|f(i)\|_{H_i} = \|\beta(i)\|_2\) for all kernels against the value of the regularization parameter \(\lambda\). The results are shown in Figure 2(b). As the figure indicates, several of the false functions are driven to 0 fast whereas the true functions persist for longer. At \(\lambda = 200\) we recover all true nonzero functions for a true positive rate of 100% and have 47 false negatives for a false positive rate of 3.7%.

### 4. Conclusion

We proposed a framework for additive least squares regression. We design our estimate to be a sum of functions where the functions are obtained by jointly optimizing over several RKHSs. The proposed framework is useful for high dimensional nonparametric regression since it provides favourable bias variance tradeoffs in high dimensions. Further, it can also be used for the recovery of sparse functions when the underlying function is additive. Our initial experimental results indicate that our methods are superior or competitive with existing methods in both fronts.

Going forward, we wish to study the theoretical properties of such penalized additive models especially focusing on rate of convergence and sparsistency.

### References


Friedman, Jerome, Hastie, Trevor, and Tibshirani, Robert.


