Regularised Additive Least Squares Regression and its Applications

Kirthevasan Kandasamy
kkandasa@andrew.cmu.edu

Calvin McCarter
cmccarte@andrew.cmu.edu

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 requiring exponentially many samples 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 other popular 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 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. $f(x) = w^\top x$ 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 [6]. With rare exceptions, nonparametric methods typically work well only under at most $4 - 6$ dimensions. In addition they 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. To the best of our knowledge, no analogous methods exist in the nonparametric world. In this project we intend to make progress in this problem by treating the *estimate* of the function as an additive function– $\hat{f}(\cdot) = f^{(1)}(\cdot) + f^{(2)}(\cdot) + \cdots + f^{(M)}(\cdot)$.

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 [7, 8, 13]. 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 understoon 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 regression. In fact, Duvenaud et al. [3] show that using additive models in the Gaussian Processes (GP) improves prediction performance. In this work, we propose the *ESP Kernels* which constrain the estimated function to be an addition of simpler functions and provide favourable bias-variance tradeoffs in high dimensions.

The second is on identifying structure in the true function f. In some genomics applications, the function of interest depends on the states of possibly several proteins. However, the true dependence may be just an addition of sparse pairwise dependencies. For instance a function of 100 variables may take the form $f(x_1^{100}) = f^{(1)}(x_1, x_2) + f^{(2)}(x_1, x_9) + f^{(3)}(x_8, x_9)$. Identifying such structure from a set of candidate sets of variables and learning the relevant functions is an important problem in Genomics. We use the additive regression framework by optimising for the individual functions $\hat{f}^{(j)}$ over a space of functions on a subset of variables. In this context, our work extends Sparse Additive Models (SpAM) [13] to multidimensional nonparametric basis functions. Our proposed method also extends recent work on Generalized Additive Models plus Interactions [9]. However, in this work the interaction model was assumed to follow a specific functional form, leading to an optimization method tailored to their interaction model. Our research is also related to existing work on using linear combinations of kernels for kernel learning, called multiple kernel learning [5].

Optimization for our proposed method is complicated by the non-smooth $\ell_{1,2}$-norm regularization penalty. Algorithms for group lasso have addressed this problem through a variety of approaches. Proximal gradient [1] has cheap iterations and relatively fast convergence if combined with acceleration. An block coordinate descent method has also been developed [11]. Further, the general Coordinate Gradient Descent method [15] can also be specialized to $\ell_{1,2}$-penalized problems [4, 10]. Recent work [17] on the group fused lasso has sidestepped the $\ell_{1,2}$-norm penalty, transforming it to a smooth objective with non-negativity constraint. Finally, safe screening rules for the group lasso have been developed [16] which quickly eliminate many of the all-0 groups. For Sparse Additive Models, parameters are typically optimized via the backfitting algorithm [13], a special case of (block) coordinate descent with group sizes of 1. In our work, we experiment with several optimisation methods for non-smooth objectives. In our experiments, Block Coordinate Gradient Descent provided the best performance.

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 and compare several methods to optimise our objective. In Section 4 we present experiments on synthetic and real datasets in both settings described above.

2 Additive Kernel Regression

2.1 Problem Statement & Notation

Let $f : \mathcal{X} \to \mathbb{R}$ be the function of interest. Here $\mathcal{X} \ni x = [x_1, \ldots, x_D] \in \mathbb{R}^D$ and $\mathcal{X} \subset \mathbb{R}^D$. We have data $(X_i, Y_i)_1^n$ 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,

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

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

The work in Hastie and Tibshirani [7] treats $\hat{f}$ as a sum of one dimensional components. In Equation (1) this corresponds to setting $M = D$ and have each $\hat{f}^{(j)}$ 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.

$\hat{f}(x_1, x_2, x_3) = \hat{f}^{(1)}(x_1) + \hat{f}^{(2)}(x_1, x_2) + \hat{f}^{(3)}(x_2, x_3)$. Ravikumar et al. [13] treat $\hat{f}$ as a sparse combination of one dimensional functions. While this is seemingly restrictive than [7], 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. $\hat{f}^{(j)} = \mathbf{0}$ 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 $J(f) = \sum_{i=1}^{n} \ell(f(X_i), Y_i) + \lambda P(f)$ over a nonparametric class of functions $\mathcal{F}$. Here ℓ is a loss function and P is a term that penalises the complexity of the function 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 $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ [14] and takes $\mathcal{F}$ is taken to be the reproducing kernel Hilbert space (RKHS) $\mathcal{H}_k$ corresponding to k. P is taken to be the squared RKHS norm of f and ℓ the squared error loss. Accordingly, KRR is characterised via the optimisation objective,

$$\hat{f} = \operatorname*{argmin}_{f \in \mathcal{H}_k} \sum_{i=1}^{n} (Y_i - f(X_i))^2 + \lambda \|f\|^2_{\mathcal{H}_k}$$

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 $\hat{f}^{(j)} \in \mathcal{H}_{k^{(j)}}$. Accordingly we consider an optimisation problem of the following form where we jointly optimise over $\hat{f}^{(1)}, \ldots, \hat{f}^{(M)}$,

$$\{\hat{f}^{(j)}\}_{j=1}^{M} = \operatorname*{argmin}_{f^{(j)} \in \mathcal{H}_{k^{(j)}}, j=1,\ldots,M} F\left(\{f^{(j)}\}_{j=1}^{M}\right) \quad \text{where,} \tag{2}$$

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

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

Via a representer theorem like argument it is straightforward to show that $f^{(j)}$ will be in the linear span of the reprodcing kernel maps of the training points X_1^n – i.e. $f^{(j)}(\cdot) = \sum_j \alpha^{(j)}{}_i k^{(j)}(\cdot, X_i)$. Then, the j^{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 ${K^{(j)}}_{rc} = k^{(j)}(X_r, X_c)$. After further simplification, the objective can be written as, $\boldsymbol{\alpha} = \operatorname{argmin}_{\boldsymbol{\alpha} \in \mathbb{R}^{nM}} F(\boldsymbol{\alpha})$ where,

$$F_1(\boldsymbol{\alpha}) = \frac{1}{2} \left\| Y - \sum_{j=1}^{m} K^{(j)} \alpha^{(j)} \right\|_2^2 + \lambda \sum_{j=1}^{M} \left({\alpha^{(j)}}^\top K^{(j)} \alpha^{(j)} \right)^{q/2}. \tag{3}$$

Here $\alpha^{(j)} \in \mathbb{R}^n$ $\forall j$, $\boldsymbol{\alpha} = [{\alpha^{(1)}}^\top, \ldots, {\alpha^{(M)}}^\top]^\top \in \mathbb{R}^{nM}$ and $Y = [Y_1, \ldots, Y_n]^\top \in \mathbb{R}^n$. Given the solution to the above, our estimate is obtained via $\hat{f}(\cdot) = \sum_{j=1}^{M} \sum_{i=1}^{n} \alpha^{(j)}{}_i k^{(j)}(\cdot, X_i^{(j)})$. Equation (3) will the (convex) optimisation problem in our algorithm. We call this algorithm Additive Kernel Regression (Add-KR). A natural choice for q in the objective (2) is $q = 2$. However in this work we use $q = 1$ since it encourages a sparse subset of functions as the solution which provides interpretability of the learned models.

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 we 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,

$$k^{(1)}(x,x') = \sum_{1\le i\le D} k_i(x_i,x'_i) \tag{4}$$
$$k^{(2)}(x,x') = \sum_{1\le i_1<i_2\le D} k_{i_1}(x_{i_1},x'_{i_1})k_{i_2}(x_{i_2},x'_{i_2})$$
$$k^{(M)}(x,x') = \sum_{1\le i_1<i_2<\cdots<i_M\le D} \prod_{d=1}^{M} k_{i_d}(x_{i_d},x'_{i_d})$$

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^{th} kernel is just the j^{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,

$$k^{(j)}(x,x') = \frac{1}{j}\sum_{d=1}^{j}(-1)^{d-1}\ \kappa_j\ k^{(j-d)}(x,x')$$

Computing the M kernels this way only requires $O(DM)$ computation. We call these the *ESP Kernels*. A similar kernel using a similar trick for computing it was used in Duvenaud et al. [3].

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 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^{th} candidate group.

3 Implementation

We now describe the implementation and the optimisation procedures used therein. 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 $\boldsymbol{\beta} = [{\beta^{(1)}}^\top,\ldots,{\beta^{(M)}}^\top]$ as,

$$F_2(\boldsymbol{\beta}) = \frac{1}{2}\Big\|Y - \sum_{j=1}^{m} L^{(j)}\alpha^{(j)}\Big\|_2^2 + \lambda\sum_{j=1}^{M}\|\beta^{(j)}\|_2 \tag{5}$$

The objective, in the above form is well studied in optimisation literature as the group LASSO. We consider the following methods to optimise the objectives in (3) and (5).

Subgradient Method on $\boldsymbol{\beta}$ (5)

Because our objective function is non-smooth, the simplest method is to solve it via the subgradient method. We implemented subgradient method with decreasing step sizes for our experiments. While subgradients can be computed cheaply in $O(n^2M)$, we observed poor convergence on our problem.

Subgradient Method on $\boldsymbol{\alpha}$ (3)

Here, we directly performed subgradient method on the objective in (3). The advantage to this method is that it doesn't require the potentially expensive Cholesky decompositions at the start of the algorithm. Despite this, we saw that convergence was slow.

Proximal Gradient Method

Note that we can write the objective (3) as $F(\boldsymbol{\beta}) = G(\boldsymbol{\beta}) + \Psi(\boldsymbol{\beta})$ where G is smooth and Ψ is not. Ψ is the group lasso penalty. Via the Moreau decomposition, and using the fact that the argument in the prox operator is separable, the prox operator can be shown to be,

$$[\text{prox}_{\Psi,t}(\boldsymbol{\beta})]^{(j)} = \text{prox}_{\Psi,t}(\beta^{(j)}) = \begin{cases} \alpha^{(j)} - t\frac{\beta}{\|\beta^{(j)}\|_2} & \text{if } \|\beta^{(j)}\| < t \\ \mathbf{0} & \text{otherwise} \end{cases}$$

Obtaining the prox operator takes $O(n^2M)$ time and has cost comparable to computing the gradient of $G(\boldsymbol{\beta})$. We use the above to implement proximal gradient method as described in the class notes. We use backtracking to determine the step size and experimented both with and without acceleration.

Exact Block Coordinate Descent via Newton trust region

We can optimize the objective function over group $\beta^{(j)}$ with all other groups $\beta^{(i)}, i \neq j$ fixed. The objective is then

$$\arg\min_{\beta^{(j)}} \frac{1}{2}{\beta^{(j)}}^T A_j \beta^{(j)} + b_j^T \beta^{(j)} + \lambda\|\beta^{(j)}\|_2$$

where $A_j = {L^{(j)}}^T L^{(j)}$ and $b_j = -{L^{(j)}}^T(Y - \sum_{i\neq j} L^{(i)}\beta^{(i)})$. This problem can be efficiently solved [11] by converting it to a one-dimensional trust-region problem. When $\|b_j\|_2 \leq \lambda$, we have $\beta^{(j)} = 0$. Otherwise, there exists t_j such that $\beta^{(j)}$ is the solution of the following problem

$$\arg\min_{\beta^{(j)}} \frac{1}{2}{\beta^{(j)}}^T A_j \beta^{(j)} + b_j^T \beta^{(j)}, \text{ such that } \|\beta^{(j)}\|_2 \leq t_j.$$

If t_j is known we have $\beta^{(j)} = t_j * \left(-(t_j A_j + \lambda I)^{-1} b_j\right)$ Because in this case $\|\beta^{(j)}\|_2 = t_j$, we have $\|(t_j A_j + \lambda I)^{-1} b_j\|_2 = 1$. As described in Qin et al. [11], we can use Newton's method to solve this equation efficiently using the eigendecomposition of A_j, which is constant and needs to be computed only once. However, this method does not scale well with n, since we still need to solve an $n \times n$ linear system after t_j is computed. Thus, updating all blocks costs $O(n^3M)$.

Block Coordinate Gradient Descent

Because $\Psi(\boldsymbol{\beta})$, our $\ell_{1,2}$ group lasso penalty, is non-smooth yet block-separable, we can apply the Coordinate Gradient Descent method [15]. For each block of $\beta^{(j)}$, we solve

$$\arg\min_{d_j} \frac{1}{2} d_j^T H_j d_j + \nabla_j G(\boldsymbol{\beta})^T d_j + \lambda\|\beta^{(j)} + d_j\|_2$$

where $H_j \approx \nabla^2_{jj} G(\boldsymbol{\beta}) = {L^{(j)}}^T L^{(j)}$ and $\nabla_j G(\boldsymbol{\beta}) = -{L^{(j)}}^T(Y - \sum_{i\neq j} L^{(i)}\beta^{(i)})$. As suggested Tseng and Yun [15], we use a diagonal matrix to approximate the true block Hessian. We set $H_j = \max(\text{diag}(\nabla^2_{jj} G(\boldsymbol{\alpha})))I_n := h_j I_n$, so that a closed-form solution exists for d_j:

$$d_j = \frac{1}{h_j}\left(\lambda \frac{\nabla_j G(\boldsymbol{\beta}) - h_j\beta^{(j)}}{\|\nabla_j G(\boldsymbol{\beta}) - h_j\beta^{(j)}\|_2}\right).$$

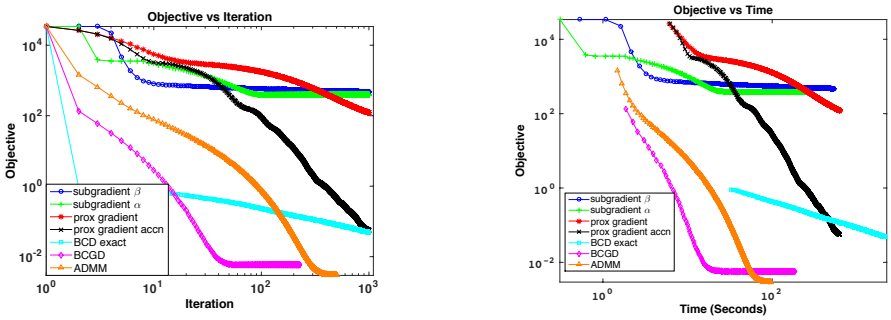
We use backtracking to determine the step size, then update $\beta^{(j)} \leftarrow \beta^{(j)} + td_j$.

The Hessian is constant, so we only need compute its diagonal once. Furthermore, we store and maintain residuals $Y - \sum_i L^{(i)}\beta^{(i)}$, so that updating each block takes $O(n^2)$. Thus, the cost of each iteration is $O(n^2M)$.

Alternating direction method of multipliers (ADMM)

We implemented ADMM for our group lasso problem (5). Using an initial LU factorization, we are able to reduce the iteration cost to $O(n^2M^2)$.

When the number of parameters for each group are small in group LASSO problems, block coordinate descent (BCD) is believed to be the state of the art solver. However, in our case the number of



(a) (b)

Figure 1: (a): Comparison of the different methods to optimise our objective. In (a) The x-axis is the iteration and in (b) the x-axis is time. In both figures the y-axis is the objective. Both figures are in log-log scale.

Dataset (D, n)	Add-KR	KRR	kNN	NW	LL	LQ	GP	SVR
Speech (21, 520)	**0.02269**	0.02777	0.09348	0.11207	0.03373	0.02407	0.02531	0.22431
Music (90, 1000)	**0.91627**	0.91922	1.00001	1.05745	1.25805	1.06482	0.94329	1.07009
Tele-motor (19, 300)	**0.06059**	0.06488	0.13957	0.20119	0.09455	0.08774	0.06678	0.38038
Housing (12, 256)	**0.31285**	0.35947	0.43619	0.42087	**0.31219**	0.35061	0.67566	1.15272
Blog (91, 700)	**1.43288**	1.53227	1.73545	1.49305	1.69234	1.71321	1.64429	1.66705
Forest Fires (10, 210)	0.30675	0.32618	0.40565	0.37199	0.35462	0.33881	**0.29038**	0.70154
Propulsion (15, 400)	0.04167	0.01396	0.15760	0.11237	0.182345	0.19212	**0.00355**	0.74511

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.

parameters is large – growing linearly with 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 ADMM significantly outperformed BCD in our experiments. In fact, we tried several other methods including subgradient method, proximal gradient method and ADMM and found that BCGD/ ADMM performed best. The results are shown in Figures 1(a) and 1(b).

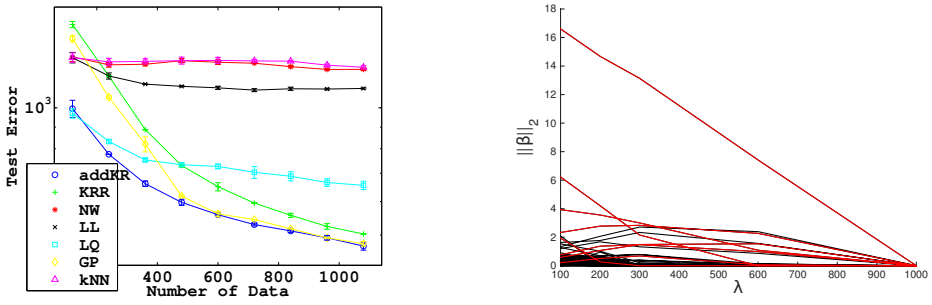
The penalty coefficient λ was chosen using 5-fold cross validation. Our implementation first solves for the largest λ. For successive λ values, we initialise BCGD at the solution of the previous λ. This warm starts procedure significantly speeds up the running time of the entire training procedure.

4 Experiments

4.1 Setting 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 [6, 13] 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 kidge 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 Chang and Lin [2], Rasmussen and Nickisch [12] respectively. For the other methods, we chose hyper parameters using 5-fold cross validation. The Additive Gaussian process model of Duvenaud et al. [3] is also a candidate but we found that inference was extremely



(a) (b)

Figure 2: (a): Comparison of Add-KR using ESP Kernels against other nonparametric methods on a 20 dimensional toy problem. The x-axis denotes the number of training points and the y-axis is the error on a test set. (b): Solution path with $n = 600$ samples for the synthetic problem. The x-axis shows the regularisation parameter while the y-axis plots $\|f^{(j)}\|_{\mathcal{H}_{k^{(j)}}} = \|\beta^{(j)}\|$. The true nonzero functions are depicted in red. As the figure indicates several of the false functions are driven to 0 fast whereas the true functions persist for longer.

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, 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 unit standard deviation. We split the datasets into roughly two halves for training and testing. The results are given in Table 1. Our proposed method outperforms alternatives in most cases.

4.2 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:

$$y_i = f_1(x_{i1}) + f_2(x_{i2}) + f_3(x_{i3}) + f_4(x_{i4}) + f_1(x_{i5}x_{i6}) + f_2(x_{i7}x_{i8}) + f_3(x_{i9}x_{i10}) + f_4(x_{i11}x_{i12}) + \epsilon_i$$

where,

$$f_1(x) = -2\sin(2x), \quad f_2(x) = x^2 - \frac{1}{3}, \quad f_3(x) = x - \frac{1}{2}, f_4(x) = e^{-x} + e^{-1} - 1$$

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 charactersing our RKHSs are of the form $k(x_i, x_i')$ for $i = 1, \ldots, 50$ and $k(x_i, x_i)k(x_j, x_j)$ 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 $\|\hat{f}^{(j)}\|_{\mathcal{H}_{k^{(j)}}} = \|\beta^{(j)}\|_2$ for all kernels against the value of the regularization parameter λ. The results are shown in Figure 2(b). 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%

5 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 optimising 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

[1] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[2] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2011. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[3] David K. Duvenaud, Hannes Nickisch, and Carl Edward Rasmussen. Additive gaussian processes. In *Advances in Neural Information Processing Systems*, 2011.

[4] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. A note on the group lasso and a sparse group lasso. *arXiv preprint arXiv:1001.0736*, 2010.

[5] Mehmet Gönen and Ethem Alpaydın. Multiple kernel learning algorithms. *The Journal of Machine Learning Research*, 12:2211–2268, 2011.

[6] László Györfi, Micael Kohler, Adam Krzyzak, and Harro Walk. *A Distribution Free Theory of Nonparametric Regression*. Springer Series in Statistics, 2002.

[7] T. J. Hastie and R. J. Tibshirani. *Generalized Additive Models*. London: Chapman & Hall, 1990.

[8] John D. Lafferty and Larry A. Wasserman. Rodeo: Sparse Nonparametric Regression in High Dimensions. In *NIPS*, 2005.

[9] Yin Lou, Rich Caruana, Johannes Gehrke, and Giles Hooker. Accurate intelligible models with pairwise interactions. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 623–631. ACM, 2013.

[10] Lukas Meier, Sara Van De Geer, and Peter Bühlmann. The group lasso for logistic regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(1):53–71, 2008.

[11] Zhiwei Qin, Katya Scheinberg, and Donald Goldfarb. Efficient block-coordinate descent algorithms for the group lasso. *Mathematical Programming Computation*, 5(2):143–169, 2013.

[12] Carl Edward Rasmussen and Hannes Nickisch. Gaussian Processes for Machine Learning (GPML) Toolbox. *J. Mach. Learn. Res.*, 2010.

[13] Pradeep Ravikumar, John Lafferty, Han Liu, and Larry Wasserman. Sparse Additive Models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 2009.

[14] Bernhard Scholkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001.

[15] Paul Tseng and Sangwoon Yun. A coordinate gradient descent method for nonsmooth separable minimization. *Mathematical Programming*, 117(1-2):387–423, 2009.

[16] Jie Wang, Jiayu Zhou, Peter Wonka, and Jieping Ye. Lasso screening rules via dual polytope projection. In *Advances in Neural Information Processing Systems*, pages 1070–1078, 2013.

[17] Matt Wytock, Suvrit Sra, and J Zico Kolter. Fast newton methods for the group fused lasso. In *Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence*, 2014.