Two New Kernel Least Squares Based Methods for Regression

Steven Busuttil Yuri Kalnishkan
Alex Gammerman
Computer Learning Research Centre
{steven,yura,alex}@cs.rhul.ac.uk

Technical Report

CLRC — TR — 06 — 01

March 2006

Royal Holloway
University of London

Department of Computer Science
Egham, Surrey TW20 0EX, England

Abstract

Kernel Ridge Regression (KRR) and the Kernel Aggregating Algorithm for Regression (KAAR) are existing regression methods based on Least Squares. KRR is a well established regression technique, while KAAR is the result of relatively recent work. KAAR is similar to KRR but with some extra regularisation that makes it predict better when the data is heavily corrupted by noise. In the general case, however, this extra regularisation is excessive and therefore KRR performs better. In this paper, two new methods for regression, Iterative KAAR (IKAAR) and Controlled KAAR (CKAAR) are introduced. IKAAR and CKAAR make it possible to control the amount of extra regularisation or to remove it completely, which makes them generalisations of both KRR and KAAR. Some properties of these new methods are proved and their predictive performance on both synthetic and real-world datasets (including the well known Boston Housing dataset) is compared to that of KRR and that of KAAR. Empirical results that have been checked for statistical significance suggest that in general both IKAAR and CKAAR make predictions that are equivalent or better than those of KRR and KAAR.

Contents

1 Introduction

In regression we are interested in finding a mathematical relationship between a signal which can be one or more independent variables and its outcome[1]. As an example, consider the case where we are given a training set comprised of the height and weight of adult men as our signals and outcomes respectively. Our task is to find the relationship between these two variables. In the simplest of models the relationship is taken to be a linear one but nonlinear relationships are common in nature. Once this relationship is established, it is possible to predict the outcomes of unseen signals.

The first solution to this problem was that of Least Squares by Legendre and Gauss (independently) in the beginning of the 19th century. Least Squares finds the line (or hyperplane) that fits the data with minimum squared differences, known as square losses. This method however, has some drawbacks including the fact that it fits the training set too well (known as overfitting) and may not generalise well to unseen data. This is especially true if the data is corrupted with noise. Ridge Regression (RR) [Hoerl, 1962] is an improvement on least squares in that it attempts to balance the goodness of fit of the hyperplane with its complexity. This is known as regularisation and results in a solution that is not necessarily the optimal one on the training data but usually generalises better. Ridge Regression works very well on real-world data and is still very popular among statisticians today. A new method, the Aggregating Algorithm for Regression (AAR) was introduced in Vovk [1998] and was shown to be only a little worse than any linear predictor in the online mode of learning. This method can be naturally applied to the batch (offline) case, which is our main focus in this paper[2]. It happens that AAR is similar to RR but with some extra regularisation added.

RR and AAR can be formulated in dual variables (see Saunders et al. [1998] and Gammerman et al. [2004] respectively), where all the data appears in dot products which are then replaced by kernels. By definition, kernels are dot products in some feature space. This means that a hyperplane is found in feature space that corresponds to a nonlinear relationship in input space. We denote the kernel versions of these methods by Kernel Ridge Regression (KRR) and the Kernel Aggregating Algorithm for Regression (KAAR).

From our analyses, we found that in most cases the regularisation of KAAR is too strong and results in performance that is not very good. On the other hand, sometimes KRR's regularisation is not strong enough and this results in KRR's predictions fluctuating a lot. In Section 3 we introduce two new methods, Iterative KAAR (IKAAR) and Controlled KAAR (CKAAR) which are both generalisations of KRR and KAAR. Our methods combine KRR and KAAR in such a way as to be able to dictate the amount of extra regularisation, the choice of which should depend on the data at hand. In Section 3 we proceed to prove some of their theoretical properties, and in Section 4 we report their empirical performance on synthetic and real-world datasets. From these results we conclude that in general both our new methods give a statistically significant advantage over KRR and KAAR.

2 Background

Regression can be defined by the following problem. Given a set of ℓ signal-outcome pairs $(\mathbf{x}_i, y_i) \in \mathbb{R}^m \times \mathbb{R}$, and a new signal $\mathbf{x}_{\ell+1}$, we are required to output a prediction $\gamma_{\ell+1} \in \mathbb{R}$ that approximates the true outcome $y_{\ell+1}$ of $\mathbf{x}_{\ell+1}$. Note that as per convention, all the vectors in this paper are column vectors. The most commonly used measure of goodness of a prediction is the square loss $(y-\gamma)^2$, where a smaller value means a better prediction. Let us model the data by the linear (in the

[1] In other literature, outcomes are also called labels or targets, while signals are also called examples or instances. In this paper we will be using the terms outcomes and signals to be consistent with Gammerman et al. [2004], which inherited the terminology from prediction with expert advice.

[2] However, all the methods described in this paper can be used for both online and batch modes of learning.

parameters) equation

$$y_i = \langle \mathbf{w}, \mathbf{x}_i \rangle + \varepsilon, \tag{1}$$

where $\mathbf{w} \in \mathbb{R}^m$ and $\varepsilon \in \mathbb{R}$ is some noise. Our aim is to find a solution to (1) (i.e. a $\mathbf{w}_{\text{L}}$) that minimises the overall sum of square losses of the predictions on the given data

$$\mathcal{L}_{\text{L}} = \sum_{i=1}^{\ell} (y_i - \langle \mathbf{w}_{\text{L}}, \mathbf{x}_i \rangle)^2. \tag{2}$$

A method to find $\mathbf{w}_{\text{L}}$, known as the method of Least Squares, was derived independently by Legendre and Gauss in 1805 and 1809 respectively. It translates to solving the system of linear equations

$$\mathbf{w}_{\text{L}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}, \qquad \text{where } \mathbf{X} = \begin{bmatrix} \mathbf{x}'_1 \\ \vdots \\ \mathbf{x}'_\ell \end{bmatrix} \text{ and } \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_\ell \end{bmatrix}. \tag{3}$$

2.1 Ridge Regression

Least Squares runs into problems when some features (columns) in $\mathbf{X}$ are highly correlated (either naturally or coincidentally) because the matrix $\mathbf{X}'\mathbf{X}$ becomes close to singular, resulting in unstable solutions. Ridge Regression (RR), first introduced to statistics by Hoerl [1962], differs from least squares in that its objective is to minimise

$$\mathcal{L}_{\text{R}} = \alpha \|\mathbf{w}_{\text{R}}\|^2 + \sum_{i=1}^{\ell} (y_i - \langle \mathbf{w}_{\text{R}}, \mathbf{x}_i \rangle)^2, \tag{4}$$

where α is a fixed positive real number. Finding the solution now involves calculating

$$\mathbf{w}_{\text{R}} = (\alpha \mathbf{I} + \mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}, \tag{5}$$

where $\mathbf{I}$ is the identity matrix. Apart from stabilising the solution (the matrix $(\alpha \mathbf{I} + \mathbf{X}'\mathbf{X})$ is positive definite and therefore nonsingular, since $\alpha > 0$), this technique also includes regularisation in that it restricts the size of the parameters in $\mathbf{w}_{\text{R}}$. This reduces the complexity of the solution, decreasing the risk of overfitting the training data, and consequently generalises better.

2.2 The Aggregating Algorithm for Regression

The Aggregating Algorithm (AA) [Vovk, 1990] is a technique that predicts using expert advice. This means that AA observes the next signal in a sequence and also the predictions of a (possibly infinite) pool of experts. It then merges the experts' predictions and outputs its own prediction which is in a sense optimal. In Vovk [1998] AA was applied to the problem of linear regression resulting in the Aggregating Algorithm for Regression (AAR) which merges all the linear predictors that map signals to outcomes. In this case AAR is optimal in the sense that the total loss it suffers is only a little worse than that of any one particular linear function. It turns out that AAR is similar to RR but with the signal-outcome pair $(\mathbf{x}_{\ell+1}, 0)$ added to the training set, where $\mathbf{x}_{\ell+1}$ is the new signal for which a prediction has to be made. This makes predictions shrink towards 0, with the aim of making them even more resistant to overfitting[3]. AAR aims to find a solution $\mathbf{w}_{\text{A}}$ that minimises

$$\mathcal{L}_{\text{A}} = \alpha \|\mathbf{w}_{\text{A}}\|^2 + \langle \mathbf{w}_{\text{A}}, \mathbf{x}_{\ell+1} \rangle^2 + \sum_{i=1}^{\ell} (y_i - \langle \mathbf{w}_{\text{A}}, \mathbf{x}_i \rangle)^2. \tag{6}$$

[3] It is assumed that the outcomes have a mean of 0.

The AAR solution to the regression problem is therefore

$$\mathbf{w}_{\text{A}} = (\alpha \mathbf{I} + \widetilde{\mathbf{X}}'\widetilde{\mathbf{X}})^{-1}\widetilde{\mathbf{X}}'\widetilde{\mathbf{y}}, \tag{7}$$

where $\widetilde{\mathbf{X}} = (\mathbf{X}', \mathbf{x}_{\ell+1})'$ and $\widetilde{\mathbf{y}} = (\mathbf{y}', 0)'$.

2.3 Kernel Methods

The use of RR and AAR in the real world is limited since they can only model simple linear dependencies. One solution could be to change the underlying assumption that the data is modelled by the linear function (1) and consider the possibility that it is modelled by some nonlinear function. Finding the optimal solution in this case would be much more difficult and one could be prone to getting caught in local minima. Another option could be to map the data to some high dimensional feature space and finding a simple solution as in (1) there. This however, can lead to what is known as the curse of dimensionality where both the computational and generalisation performance degrades as the number of features grow [Cristianini and Shawe-Taylor, 2000]. The kernel trick (first used in this context in Aizerman et al. [1964]) is now a widely used technique which can make a linear algorithm operate in a (possibly high dimensional) feature space without the inherent complexities.

A kernel function k takes two vectors and returns their dot product in some feature space,

$$k(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle, \tag{8}$$

where ϕ is a (nonlinear) transformation to feature space. Usually the mapping ϕ is not performed explicitly, in fact it is not even required to be known. For a function to be a kernel it has to be symmetric, and for all ℓ and all $\mathbf{x}_1, \ldots, \mathbf{x}_\ell \in \mathbb{R}^m$, the kernel matrix $\mathbf{K} = (k(\mathbf{x}_i, \mathbf{x}_j))_{i,j}$, $i, j = 1, \ldots, \ell$ must be positive semi-definite (that is, have nonnegative eigenvalues). There are several standard kernel functions however one could design one's own according to the need. As an example of a standard kernel consider the polynomial kernel, $k_p(\mathbf{x}_i, \mathbf{x}_j) = (\langle \mathbf{x}_i, \mathbf{x}_j \rangle + 1)^d$, which is equivalent to mapping the vectors to a feature space which is spanned by the products of their features (known as monomials) up to the dth degree and taking their dot product there.

Through kernel functions it is therefore possible to perform linear regression using an algorithm like RR or AAR in feature space which would be equivalent to performing a nonlinear regression in input space. To make the use of kernels possible, Ridge Regression and the Aggregating Algorithm for Regression have been reduced into a formulation known as dual variables (see Saunders et al. [1998] and Gammerman et al. [2004] respectively). In this formulation all the signals appear only in dot products. This makes transforming the linear models into nonlinear ones simply a matter of replacing the dot products with a kernel function. The new methods, which we shall call Kernel Ridge Regression (KRR) and Kernel Aggregating Algorithm for Regression (KAAR), respectively calculate the prediction γ for a new example $\mathbf{x}_{\ell+1}$ as follows:

$$\gamma_{\text{KRR}} = \mathbf{y}'(\alpha \mathbf{I} + \mathbf{K})^{-1}\mathbf{k}, \tag{9}$$

where $\mathbf{K} = (k(\mathbf{x}_i, \mathbf{x}_j))_{i,j}$, $i, j = 1, \ldots, \ell$, and $\mathbf{k} = (k(\mathbf{x}_i, \mathbf{x}_{\ell+1}))$, $i = 1, \ldots, \ell$, and,

$$\gamma_{\text{KAAR}} = \widetilde{\mathbf{y}}'(\alpha \mathbf{I} + \widetilde{\mathbf{K}})^{-1}\widetilde{\mathbf{k}}, \tag{10}$$

where $\widetilde{\mathbf{y}} = (\mathbf{y}', 0)'$, $\widetilde{\mathbf{K}} = (k(\mathbf{x}_i, \mathbf{x}_j))_{i,j}$, $i, j = 1, \ldots, \ell + 1$, and $\widetilde{\mathbf{k}} = (k(\mathbf{x}_i, \mathbf{x}_{\ell+1}))$, $i = 1, \ldots, \ell + 1$.

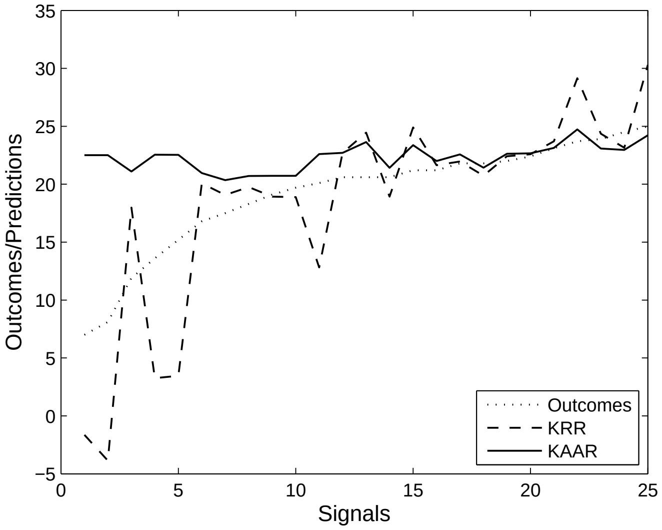

Figure 1 KRR and KAAR approximating a signal-outcome behaviour.

3 Methods

3.1 Motivation and Introduction

Figure 1 shows the predictions of KRR and KAAR on a test set containing 25 signals from a particular permutation[4] of the Boston Housing dataset (described in Section 4.3.4). Note that the signals in the test set have been sorted by their target outcome and that the x-axis represents the number of a signal and is not the signal itself (which is a vector with 13 features). This was done exclusively to make the figure clearer. In this example the mean square loss of KRR is approximately 26.64 while that of KAAR is approximately 29.33. This means that KRR's performance is better. However, analysis of the individual predictions reveals that 44% of KAAR's predictions are more accurate. In addition, as can be seen in the figure, sometimes a better prediction would be somewhere in between those of KRR and KAAR. Is it possible therefore to 'combine' these two methods to give a new method that in general is more accurate than both?

Below we present two new methods that attempt to achieve this. The first method, which we call Iterative KAAR (IKAAR), modifies the KAAR algorithm so that it outputs a sequence of predictions for a particular signal. We show that this sequence starts from the KAAR prediction and converges towards the prediction of KRR giving us a smooth transition from the former to the latter. The second method that we propose uses the fact that in Figure 1 KRR seems to fluctuate a lot with a variance of 78.74, while KAAR is overly rigid having a variance of 1.30 (the variance of the real outcomes is 22.83). We therefore modify KAAR's objective to give us a new method where we can control the rigidness of the predictions. The new method, Controlled KAAR (CKAAR), can be made to behave like KAAR, KRR or something in between them.

[4] For a different permutation of the dataset the figure will be different but the general idea of what we are trying to show holds.

3.2 Iterative KAAR

As we saw in Section 2.2, KAAR is equivalent to KRR with the signal-outcome pair $(\mathbf{x}, 0)$ added to its training set, where $\mathbf{x} = \mathbf{x}_{\ell+1}$ is the new signal. This outputs a prediction γ_{KAAR}. Having 0 as the signal's outcome added to the training set pushes the prediction towards 0 and is what makes KAAR's predictions so rigid. In order to alleviate this we propose a new method, the Iterative Kernel Aggregating Algorithm for Regression (IKAAR). In its first iteration, IKAAR is equivalent to KAAR, in that it adds the pair $(\mathbf{x}, 0)$ to its training set. This produces the prediction γ_{KAAR}. However, in its second iteration, IKAAR replaces the extra pair in its training set with a new pair $(\mathbf{x}, \gamma_{\text{KAAR}})$. This produces another prediction that in turn is used to replace γ_{KAAR} and be added to the training set to make a new prediction. This procedure can be repeated an arbitrary number of times, resulting in several IKAAR predictions for the same signal. We will denote these predictions by $\gamma^{(n)}_{\text{IKAAR}}$ where the index (n) denotes the iteration number. For clarity of notation let $\gamma^{(n)} = \gamma^{(n)}_{\text{IKAAR}}$. We define IKAAR more formally as follows:

$$\gamma^{(n)} = \widetilde{\mathbf{y}}^{(n)\,\prime}(\alpha\mathbf{I} + \widetilde{\mathbf{K}})^{-1}\widetilde{\mathbf{k}}, \tag{11}$$

where $\gamma^{(0)} = 0$, $n \geq 1$, and $\widetilde{\mathbf{y}}^{(n)} = \left(\mathbf{y}', \gamma^{(n-1)}\right)'$. We will later give a formula that computes any $\gamma^{(n)}$ directly, without the need to calculate all the previous predictions $\left(\gamma^{(n-1)}, \gamma^{(n-2)}, \ldots, \gamma^{(1)}\right)$.

Theorem 3.1. *For any signal, IKAAR's predictions start from the KAAR prediction and converge towards that of KRR as the number of IKAAR iterations approaches infinity.*

Proof. It follows from IKAAR's definition that the first prediction $\gamma^{(1)}$ is equivalent to KAAR's prediction. We will now show that IKAAR's predictions for any particular signal converge towards that of KRR as n approaches infinity. We can open up (11) in the following way:

$$\gamma^{(n)} = \begin{bmatrix} \mathbf{y}' & \gamma^{(n-1)} \end{bmatrix} \begin{bmatrix} \mathbf{K} + \alpha\mathbf{I} & \mathbf{k} \\ \mathbf{k}' & k(\mathbf{x}, \mathbf{x}) + \alpha \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{k} \\ k(\mathbf{x}, \mathbf{x}) \end{bmatrix}. \tag{12}$$

From this equation it is clear that we are taking $\gamma^{(n-1)}$ and modifying it to get $\gamma^{(n)}$. We shall show that this transformation of $\gamma^{(n-1)}$ can be characterised by the linear equation

$$\gamma^{(n)} = s\gamma^{(n-1)} + c, \tag{13}$$

where $s, c \in \mathbb{R}$, corresponding to a line. If we manage to show that $0 \leq |s| < 1$ then that would be enough to prove that IKAAR's predictions converge to a fixed point r, such that $r = sr + c$. This follows from the Banach fixed-point theorem, and is also evident in Figure 2, where the dotted lines show the behaviour of $\gamma^{(n)}$ per iteration[5]. For example, in the first iteration $\gamma^{(n-1)} = \gamma^{(0)} = 0$ therefore $\gamma^{(n)} = \gamma^{(1)} = c$. In the second iteration, $\gamma^{(n-1)} = \gamma^{(1)} = c$ therefore $\gamma^{(n)} = \gamma^{(2)}$ is equal to the value on the line where the x-axis is equal to c. This process is repeated and as $n \to \infty$, then $\gamma^{(n-1)} \to \gamma^{(n)}$ and $\gamma^{(n)} \to r$.

In our proof we will be using the following two Lemmas.

Lemma 3.2 (See Press et al. [1994, Section 2.7]). *Suppose we are given a matrix* $\mathbf{A}$ *of size* $n \times n$ *partitioned in the following way*

$$\mathbf{A} = \begin{bmatrix} \mathbf{P} & \mathbf{Q} \\ \mathbf{R} & \mathbf{S} \end{bmatrix},$$

where $\mathbf{P}$ *and* $\mathbf{S}$ *are square matrices of size* $p \times p$ *and* $s \times s$ *respectively* $(p + s = n)$*, and* $\mathbf{Q}$ *and* $\mathbf{R}$ *of size* $p \times s$ *and* $s \times p$ *respectively (not necessarily square). If its inverse is partitioned is a similar manner,*

$$\mathbf{A}^{-1} = \begin{bmatrix} \widetilde{\mathbf{P}} & \widetilde{\mathbf{Q}} \\ \widetilde{\mathbf{R}} & \widetilde{\mathbf{S}} \end{bmatrix},$$

[5] In the figure only the cases where $0 \leq s < 1$ are shown, however the other cases are similar.

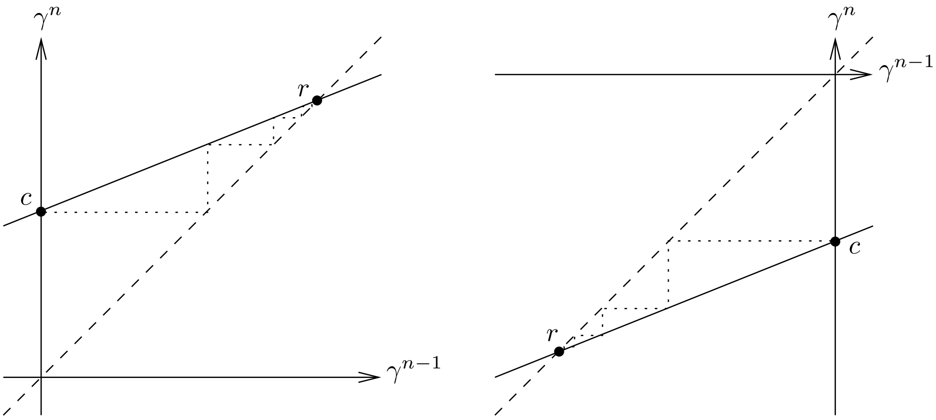

Figure 2 This figure depicts the behaviour of IKAAR's prediction $\gamma^{(n)}$ in relation to the previous prediction $\gamma^{(n-1)}$. The solid line is $\gamma^{(n)} = s\gamma^{(n-1)} + c$ and it is shown for $0 \le s < 1$ and $c > 0$ (left) and $c < 0$. The dashed line is the bisector. The dotted lines show the behaviour of $\gamma^{(n)}$ per iteration, starting from c and converging to r.

then $\widetilde{\mathbf{P}}$*,* $\widetilde{\mathbf{Q}}$*,* $\widetilde{\mathbf{R}}$ *and* $\widetilde{\mathbf{S}}$ *which have the same sizes as* $\mathbf{P}$*,* $\mathbf{Q}$*,* $\mathbf{R}$ *and* $\mathbf{S}$ *respectively, can be calculated by the following formulae (provided all the inverses exist):*

$$\begin{aligned}
\widetilde{\mathbf{P}} &= \mathbf{P}^{-1} + \mathbf{P}^{-1}\mathbf{Q}(\mathbf{S} - \mathbf{R}\mathbf{P}^{-1}\mathbf{Q})^{-1}\mathbf{R}\mathbf{P}^{-1}, \\
\widetilde{\mathbf{Q}} &= -\mathbf{P}^{-1}\mathbf{Q}(\mathbf{S} - \mathbf{R}\mathbf{P}^{-1}\mathbf{Q})^{-1}, \\
\widetilde{\mathbf{R}} &= -(\mathbf{S} - \mathbf{R}\mathbf{P}^{-1}\mathbf{Q})^{-1}\mathbf{R}\mathbf{P}^{-1}, \\
\widetilde{\mathbf{S}} &= (\mathbf{S} - \mathbf{R}\mathbf{P}^{-1}\mathbf{Q})^{-1}.
\end{aligned}$$

Lemma 3.3. *Given a matrix* $\mathbf{A}$*, a scalar* α *and* $\mathbf{I}$ *identity matrices of the appropriate size,*

$$(\mathbf{A}\mathbf{A}' + \alpha\mathbf{I})^{-1}\mathbf{A} = \mathbf{A}(\mathbf{A}'\mathbf{A} + \alpha\mathbf{I})^{-1}.$$

Proof.

$$\begin{aligned}
(\mathbf{A}\mathbf{A}' + \alpha\mathbf{I})^{-1}\mathbf{A} &= (\mathbf{A}\mathbf{A}' + \alpha\mathbf{I})^{-1}\mathbf{A}\mathbf{I} \\
&= (\mathbf{A}\mathbf{A}' + \alpha\mathbf{I})^{-1}\mathbf{A}(\mathbf{A}'\mathbf{A} + \alpha\mathbf{I})(\mathbf{A}'\mathbf{A} + \alpha\mathbf{I})^{-1} \\
&= (\mathbf{A}\mathbf{A}' + \alpha\mathbf{I})^{-1}(\mathbf{A}\mathbf{A}'\mathbf{A} + \alpha\mathbf{A})(\mathbf{A}'\mathbf{A} + \alpha\mathbf{I})^{-1} \\
&= (\mathbf{A}\mathbf{A}' + \alpha\mathbf{I})^{-1}(\mathbf{A}\mathbf{A}' + \alpha\mathbf{I})\mathbf{A}(\mathbf{A}'\mathbf{A} + \alpha\mathbf{I})^{-1} \\
&= \mathbf{I}\mathbf{A}(\mathbf{A}'\mathbf{A} + \alpha\mathbf{I})^{-1} \\
&= \mathbf{A}(\mathbf{A}'\mathbf{A} + \alpha\mathbf{I})^{-1}
\end{aligned}$$

□

Using Lemma 3.2 we can rewrite (12) as follows

$$\begin{aligned}
\gamma^{(n)} &= \begin{bmatrix} \mathbf{y}' & \gamma^{(n-1)} \end{bmatrix} \begin{bmatrix} \mathbf{K} + \alpha\mathbf{I} & \mathbf{k} \\ \mathbf{k}' & k(\mathbf{x},\mathbf{x}) + \alpha \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{k} \\ k(\mathbf{x},\mathbf{x}) \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{y}' & \gamma^{(n-1)} \end{bmatrix} \begin{bmatrix} \mathbf{P} & \mathbf{Q} \\ \mathbf{R} & \mathbf{S} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{Q} \\ k(\mathbf{x},\mathbf{x}) \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{y}' & \gamma^{(n-1)} \end{bmatrix} \begin{bmatrix} \widetilde{\mathbf{P}} & \widetilde{\mathbf{Q}} \\ \widetilde{\mathbf{R}} & \widetilde{\mathbf{S}} \end{bmatrix} \begin{bmatrix} \mathbf{Q} \\ k(\mathbf{x},\mathbf{x}) \end{bmatrix}, \qquad (14)
\end{aligned}$$

where $\mathbf{P} = \mathbf{K} + \alpha\mathbf{I}$, $\mathbf{Q} = \mathbf{R}' = \mathbf{k}$, and $\mathbf{S} = k(\mathbf{x}, \mathbf{x}) + \alpha$ (it will become clear that in this case all the necessary inverses exist). If we now open (14) we get

$$\begin{aligned}
\gamma^{(n)} &= \begin{bmatrix} \mathbf{y}'\widetilde{\mathbf{P}} + \gamma^{(n-1)}\widetilde{\mathbf{R}} & \mathbf{y}'\widetilde{\mathbf{Q}} + \gamma^{(n-1)}\widetilde{\mathbf{S}} \end{bmatrix} \begin{bmatrix} \mathbf{Q} \\ k(\mathbf{x}, \mathbf{x}) \end{bmatrix} \\
&= \mathbf{y}'\widetilde{\mathbf{P}}\mathbf{Q} + \gamma^{(n-1)}\widetilde{\mathbf{R}}\mathbf{Q} + \mathbf{y}'\widetilde{\mathbf{Q}}k(\mathbf{x}, \mathbf{x}) + \gamma^{(n-1)}\widetilde{\mathbf{S}}k(\mathbf{x}, \mathbf{x}) \\
&= \left(\widetilde{\mathbf{R}}\mathbf{Q} + \widetilde{\mathbf{S}}k(\mathbf{x}, \mathbf{x})\right)\gamma^{(n-1)} + \left(\mathbf{y}'\widetilde{\mathbf{P}}\mathbf{Q} + \mathbf{y}'\widetilde{\mathbf{Q}}k(\mathbf{x}, \mathbf{x})\right).
\end{aligned}$$

Therefore in (13),

$$\begin{aligned}
s &= \widetilde{\mathbf{R}}\mathbf{Q} + \widetilde{\mathbf{S}}k(\mathbf{x}, \mathbf{x}), && (15) \\
c &= \mathbf{y}'\widetilde{\mathbf{P}}\mathbf{Q} + \mathbf{y}'\widetilde{\mathbf{Q}}k(\mathbf{x}, \mathbf{x}). && (16)
\end{aligned}$$

We have just found what s and c are. We will now proceed to show that s is always in the interval $[0, 1)$. First we will simplify slightly the definitions of $\widetilde{\mathbf{R}}$ and $\widetilde{\mathbf{S}}$ through the fact that in our case $\mathbf{R} = \mathbf{Q}'$.

$$\begin{aligned}
\widetilde{\mathbf{R}} &= -(\mathbf{S} - \mathbf{R}\mathbf{P}^{-1}\mathbf{Q})^{-1}\mathbf{R}\mathbf{P}^{-1} \\
&= -(\mathbf{S} - \mathbf{Q}'\mathbf{P}^{-1}\mathbf{Q})^{-1}\mathbf{Q}'\mathbf{P}^{-1}, \\
\widetilde{\mathbf{S}} &= (\mathbf{S} - \mathbf{R}\mathbf{P}^{-1}\mathbf{Q})^{-1} \\
&= (\mathbf{S} - \mathbf{Q}'\mathbf{P}^{-1}\mathbf{Q})^{-1}.
\end{aligned}$$

Now, we substitute $\widetilde{\mathbf{R}}$ and $\widetilde{\mathbf{S}}$ in (15) with these equations:

$$\begin{aligned}
s &= \widetilde{\mathbf{R}}\mathbf{Q} + \widetilde{\mathbf{S}}k(\mathbf{x}, \mathbf{x}) \\
&= -(\mathbf{S} - \mathbf{Q}'\mathbf{P}^{-1}\mathbf{Q})^{-1}\mathbf{Q}'\mathbf{P}^{-1}\mathbf{Q} + (\mathbf{S} - \mathbf{Q}'\mathbf{P}^{-1}\mathbf{Q})^{-1}k(\mathbf{x}, \mathbf{x}) \\
&= (\mathbf{S} - \mathbf{Q}'\mathbf{P}^{-1}\mathbf{Q})^{-1}(k(\mathbf{x}, \mathbf{x}) - \mathbf{Q}'\mathbf{P}^{-1}\mathbf{Q}) \\
&= \frac{k(\mathbf{x}, \mathbf{x}) - \mathbf{k}'(\mathbf{K} + \alpha\mathbf{I})^{-1}\mathbf{k}}{k(\mathbf{x}, \mathbf{x}) - \mathbf{k}'(\mathbf{K} + \alpha\mathbf{I})^{-1}\mathbf{k} + \alpha}. && (17)
\end{aligned}$$

Since by definition $\alpha > 0$, we only need to show that $k(\mathbf{x}, \mathbf{x}) - \mathbf{k}'(\mathbf{K} + \alpha\mathbf{I})^{-1}\mathbf{k}$ is nonnegative to reach our goal. Equivalently, we need to show that $k(\mathbf{x}, \mathbf{x}) \geq \mathbf{k}'(\mathbf{K} + \alpha\mathbf{I})^{-1}\mathbf{k}$. We will first show this for the linear kernel (that is, the normal dot product) and subsequently we will generalise the result for the nonlinear kernel case. For the linear kernel we have

$$\begin{aligned}
\mathbf{k}'(\mathbf{K} + \alpha\mathbf{I})^{-1}\mathbf{k} &= (\mathbf{X}\mathbf{x})'(\mathbf{X}\mathbf{X}' + \alpha\mathbf{I})^{-1}\mathbf{X}\mathbf{x} \\
&= \mathbf{x}'\mathbf{X}'(\mathbf{X}\mathbf{X}' + \alpha\mathbf{I})^{-1}\mathbf{X}\mathbf{x} \\
&= \mathbf{x}'\mathbf{X}'\mathbf{X}(\mathbf{X}'\mathbf{X} + \alpha\mathbf{I})^{-1}\mathbf{x}. && (18)
\end{aligned}$$

Note that the last line follows by Lemma 3.3. We now have to show that for every $\mathbf{x}$ the following holds:

$$\mathbf{x}'\mathbf{x} \geq \mathbf{x}'\mathbf{X}'\mathbf{X}(\mathbf{X}'\mathbf{X} + \alpha\mathbf{I})^{-1}\mathbf{x}. \tag{19}$$

In order to do this, we will first reduce (19) to a simpler form. Since $\mathbf{X}'\mathbf{X}$ is symmetric it can be diagonalised so that $\mathbf{X}'\mathbf{X} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}'$, where the columns of the unitary matrix $\mathbf{V}$ are the eigenvectors of $\mathbf{X}'\mathbf{X}$ and $\mathbf{\Lambda}$ is the diagonal matrix made up of the corresponding eigenvalues λ_i. Recall that since $\mathbf{V}$ is a unitary matrix, $\mathbf{V}^{-1} = \mathbf{V}'$, so $\mathbf{V}'\mathbf{V} = \mathbf{V}\mathbf{V}' = \mathbf{I}$.

Performing the substitution $\mathbf{x} = \mathbf{V}\mathbf{z}$ in (19) is the same as considering it in the orthogonal basis formed by the eigenvectors of $\mathbf{X}'\mathbf{X}$. Therefore, showing that (19) holds is equivalent to proving that

$$(\mathbf{V}\mathbf{z})'\mathbf{V}\mathbf{z} \geq (\mathbf{V}\mathbf{z})'\mathbf{X}'\mathbf{X}(\mathbf{X}'\mathbf{X} + \alpha\mathbf{I})^{-1}\mathbf{V}\mathbf{z}. \tag{20}$$

Clearly, the left hand side of (20) is equal to $\mathbf{z}'\mathbf{z}$ since

$$\begin{aligned}\mathbf{x}'\mathbf{x} &= (\mathbf{V}\mathbf{z})'\mathbf{V}\mathbf{z}\\ &= \mathbf{z}'\mathbf{V}'\mathbf{V}\mathbf{z}\\ &= \mathbf{z}'\mathbf{z}.\end{aligned}$$

For the right hand side of (20) we have

$$\begin{aligned}\mathbf{x}'\mathbf{X}'\mathbf{X}(\mathbf{X}'\mathbf{X}+\alpha\mathbf{I})^{-1}\mathbf{x} &= (\mathbf{V}\mathbf{z})'\mathbf{X}'\mathbf{X}(\mathbf{X}'\mathbf{X}+\alpha\mathbf{I})^{-1}\mathbf{V}\mathbf{z}\\ &= \mathbf{z}'\mathbf{V}'\mathbf{V}\mathbf{\Lambda}\mathbf{V}'(\mathbf{V}\mathbf{\Lambda}\mathbf{V}'+\alpha\mathbf{I})^{-1}\mathbf{V}\mathbf{z}\\ &= \mathbf{z}'\mathbf{\Lambda}\mathbf{V}'(\mathbf{V}\mathbf{\Lambda}\mathbf{V}'+\alpha\mathbf{I})^{-1}\mathbf{V}\mathbf{z}\\ &= \mathbf{z}'\mathbf{\Lambda}\mathbf{V}'(\mathbf{V}\mathbf{\Lambda}\mathbf{V}'+\alpha\mathbf{V}\mathbf{V}')^{-1}\mathbf{V}\mathbf{z}\\ &= \mathbf{z}'\mathbf{\Lambda}\mathbf{V}'\left(\mathbf{V}(\mathbf{\Lambda}+\alpha\mathbf{I})\mathbf{V}'\right)^{-1}\mathbf{V}\mathbf{z}\\ &= \mathbf{z}'\mathbf{\Lambda}\mathbf{V}'(\mathbf{V}')^{-1}(\mathbf{\Lambda}+\alpha\mathbf{I})^{-1}(\mathbf{V})^{-1}\mathbf{V}\mathbf{z}\\ &= \mathbf{z}'\mathbf{\Lambda}\mathbf{V}'\mathbf{V}(\mathbf{\Lambda}+\alpha\mathbf{I})^{-1}\mathbf{V}'\mathbf{V}\mathbf{z}\\ &= \mathbf{z}'\mathbf{\Lambda}(\mathbf{\Lambda}+\alpha\mathbf{I})^{-1}\mathbf{z}.\end{aligned}$$

So now, showing that (19) holds is equivalent to proving that

$$\mathbf{z}'\mathbf{z} \geq \mathbf{z}'\mathbf{\Lambda}(\mathbf{\Lambda}+\alpha\mathbf{I})^{-1}\mathbf{z}.$$

Since $\mathbf{X}'\mathbf{X}$ is positive semi-definite all its eigenvalues are nonnegative. Therefore all the elements in the diagonal matrix $\mathbf{\Lambda}(\mathbf{\Lambda}+\alpha\mathbf{I})^{-1}$ are $0 \leq \frac{\lambda_i}{\lambda_i+\alpha} < 1$. It follows that

$$\mathbf{z}'\mathbf{z} > \mathbf{z}'\mathbf{\Lambda}(\mathbf{\Lambda}+\alpha\mathbf{I})^{-1}\mathbf{z},$$

which means that

$$\mathbf{x}'\mathbf{x} > (\mathbf{X}\mathbf{x})'(\mathbf{X}\mathbf{X}'+\alpha\mathbf{I})^{-1}\mathbf{X}\mathbf{x}.$$

We have just proved the linear case. The nonlinear kernel case follows from the linear case in the limit (because of a finite-dimensional approximation similar to Gammerman et al. [2004])

$$k(\mathbf{x},\mathbf{x}) \geq \mathbf{k}'(\mathbf{K}+\alpha\mathbf{I})^{-1}\mathbf{k},$$

which ends this first part of our proof.

We have just shown that $0 \leq s < 1$, therefore the line $\gamma^{(n)} = s\gamma^{(n-1)} + c$ intercepts the bisector at some point r. This means that $\gamma^{(n)}$ converges to this same point. We will now analyse the last term of this equation (that is c) and consequently show that the point r coincides with the prediction made by KRR for the same signal. We will first simplify the definitions of $\widetilde{\mathbf{P}}$ and $\widetilde{\mathbf{Q}}$.

$$\begin{aligned}\widetilde{\mathbf{P}} &= \mathbf{P}^{-1}+\mathbf{P}^{-1}\mathbf{Q}(\mathbf{S}-\mathbf{R}\mathbf{P}^{-1}\mathbf{Q})^{-1}\mathbf{R}\mathbf{P}^{-1}\\ &= \mathbf{P}^{-1}+\mathbf{P}^{-1}\mathbf{Q}(\mathbf{S}-\mathbf{Q}'\mathbf{P}^{-1}\mathbf{Q})^{-1}\mathbf{Q}'\mathbf{P}^{-1},\\ \widetilde{\mathbf{Q}} &= -\mathbf{P}^{-1}\mathbf{Q}(\mathbf{S}-\mathbf{R}\mathbf{P}^{-1}\mathbf{Q})^{-1}\\ &= -\mathbf{P}^{-1}\mathbf{Q}(\mathbf{S}-\mathbf{Q}'\mathbf{P}^{-1}\mathbf{Q})^{-1}.\end{aligned}$$

We now substitute these equations for $\widetilde{\mathbf{P}}$ and $\widetilde{\mathbf{Q}}$ in (16).

$$\begin{aligned}c &= \mathbf{y}'\widetilde{\mathbf{P}}\mathbf{Q}+\mathbf{y}'\widetilde{\mathbf{Q}}k(\mathbf{x},\mathbf{x})\\ &= \mathbf{y}'\mathbf{P}^{-1}\mathbf{Q}+\mathbf{y}'\mathbf{P}^{-1}\mathbf{Q}(\mathbf{S}-\mathbf{Q}'\mathbf{P}^{-1}\mathbf{Q})^{-1}\mathbf{Q}'\mathbf{P}^{-1}\mathbf{Q}-\mathbf{y}'\mathbf{P}^{-1}\mathbf{Q}(\mathbf{S}-\mathbf{Q}'\mathbf{P}^{-1}\mathbf{Q})^{-1}k(\mathbf{x},\mathbf{x})\\ &= \mathbf{y}'\mathbf{P}^{-1}\mathbf{Q}\left(1+(\mathbf{S}-\mathbf{Q}'\mathbf{P}^{-1}\mathbf{Q})^{-1}\mathbf{Q}'\mathbf{P}^{-1}\mathbf{Q}-(\mathbf{S}-\mathbf{Q}'\mathbf{P}^{-1}\mathbf{Q})^{-1}k(\mathbf{x},\mathbf{x})\right)\\ &= \mathbf{y}'\mathbf{P}^{-1}\mathbf{Q}\left(1+(\mathbf{S}-\mathbf{Q}'\mathbf{P}^{-1}\mathbf{Q})^{-1}(\mathbf{Q}'\mathbf{P}^{-1}\mathbf{Q}-k(\mathbf{x},\mathbf{x}))\right)\\ &= \mathbf{y}'\mathbf{P}^{-1}\mathbf{Q}\,(1+(-s))\\ &= \mathbf{y}'(\mathbf{K}+\alpha\mathbf{I})^{-1}\mathbf{k}(1-s)\end{aligned}$$

But $\mathbf{y}'(\mathbf{K}+\alpha\mathbf{I})^{-1}\mathbf{k} = \gamma_{\mathrm{KRR}}$, that is KRR's prediction for the same signal, therefore

$$c = (1-s)\gamma_{\mathrm{KRR}}. \tag{21}$$

This means that (13) can be rewritten as

$$\gamma^{(n)} = s\gamma^{(n-1)} + (1-s)\gamma_{\mathrm{KRR}}. \tag{22}$$

At fixed point r,

$$\begin{aligned} r &= sr + (1-s)\gamma_{\mathrm{KRR}} \\ r - sr &= (1-s)\gamma_{\mathrm{KRR}} \\ (1-s)r &= (1-s)\gamma_{\mathrm{KRR}} \\ r &= \frac{(1-s)\gamma_{\mathrm{KRR}}}{(1-s)} \\ &= \gamma_{\mathrm{KRR}}. \end{aligned}$$

So the fixed point r is in fact the KRR prediction for the signal $\mathbf{x}$.

In summary, we have shown that for any signal, IKAAR's predictions start from KAAR's prediction and always converge towards KRR's prediction. □

Remark 3.4. As it currently stands, to compute the IKAAR prediction for an iteration n it is necessary to compute all the previous ones. We will now show how any prediction can be computed directly. Given (22) and the fact that $\gamma^{(0)} = 0$, we will prove by induction that for all n such that $n \geq 1$,

$$\begin{aligned} \gamma^{(n)} &= \gamma_{\mathrm{KRR}} - s^n\gamma_{\mathrm{KRR}} \qquad (23) \\ &= (1-s^n)\gamma_{\mathrm{KRR}}. \end{aligned}$$

Recall that s is given by (17). Clearly, (23) holds for $n = 1$, since

$$\begin{aligned} \gamma^{(1)} &= s\gamma^{(1-1)} + (1-s)\gamma_{\mathrm{KRR}} \\ &= s\gamma^{(0)} + \gamma_{\mathrm{KRR}} - s\gamma_{\mathrm{KRR}} \\ &= \gamma_{\mathrm{KRR}} - s\gamma_{\mathrm{KRR}}. \end{aligned}$$

Let us assume that (23) holds for any $n \geq 1$. We will now show that it also holds for $n+1$.

$$\begin{aligned} \gamma^{(n+1)} &= s\gamma^{(n)} + (1-s)\gamma_{\mathrm{KRR}} \\ &= s\left(\gamma_{\mathrm{KRR}} - s^n\gamma_{\mathrm{KRR}}\right) + (1-s)\gamma_{\mathrm{KRR}} \\ &= s\gamma_{\mathrm{KRR}} - s^{n+1}\gamma_{\mathrm{KRR}} + \gamma_{\mathrm{KRR}} - s\gamma_{\mathrm{KRR}} \\ &= \gamma_{\mathrm{KRR}} - s^{n+1}\gamma_{\mathrm{KRR}}. \end{aligned}$$

Since (23) holds for $n = 1$ and for $n+1$, then by the inductive principle it follows that it holds for any $n \geq 1$.

Remark 3.5. The convergence of IKAAR's predictions to those of KRR's is exponential. This is made clear in (23).

Remark 3.6. In Gammerman et al. [2004] a formula for KAAR's predictions in terms of KRR's predictions is given. We will now derive the same formula by starting from our formula for IKAAR (23) and taking $n = 1$ (that is, when IKAAR is equivalent to KAAR). Note that this is the same as the definition of c in (21) since from (13) it is clear that $\gamma^{(1)} = c$.

$$\begin{aligned} \gamma^{(1)} &= \gamma_{\mathrm{KRR}} - s\gamma_{\mathrm{KRR}} \\ &= \gamma_{\mathrm{KRR}} - \left(\frac{k(\mathbf{x},\mathbf{x}) - \mathbf{k}'(\mathbf{K}+\alpha\mathbf{I})^{-1}\mathbf{k}}{k(\mathbf{x},\mathbf{x}) - \mathbf{k}'(\mathbf{K}+\alpha\mathbf{I})^{-1}\mathbf{k} + \alpha}\right)\gamma_{\mathrm{KRR}} \\ &= \frac{\left(k(\mathbf{x},\mathbf{x}) - \mathbf{k}'(\mathbf{K}+\alpha\mathbf{I})^{-1}\mathbf{k} + \alpha\right)\gamma_{\mathrm{KRR}} - \left(k(\mathbf{x},\mathbf{x}) - \mathbf{k}'(\mathbf{K}+\alpha\mathbf{I})^{-1}\mathbf{k}\right)\gamma_{\mathrm{KRR}}}{k(\mathbf{x},\mathbf{x}) - \mathbf{k}'(\mathbf{K}+\alpha\mathbf{I})^{-1}\mathbf{k} + \alpha} \\ &= \frac{\alpha\gamma_{\mathrm{KRR}}}{k(\mathbf{x},\mathbf{x}) - \mathbf{k}'(\mathbf{K}+\alpha\mathbf{I})^{-1}\mathbf{k} + \alpha} \end{aligned}$$

3.3 Controlled KAAR

In Figure 1 it can be seen that KRR's predictions follow the general behaviour of the true outcomes, however they fluctuate a lot around them. On the other hand KAAR's predictions do not fluctuate but are too rigid and end up not following the true outcome's behaviour sufficiently well. While the latter's performance might be desirable in cases where the data is corrupted with a high level of noise we conclude that in this case it is too drastic. The reason why KAAR is so rigid is that it tries to minimise the value of the prediction itself (see the second term in (6)). In our new method, the Controlled Kernel Aggregating Algorithm for Regression (CKAAR), we try to control this behaviour by adding a coefficient to this second term such that our objective is to minimise

$$\mathcal{L}_{\text{C}} = \alpha\|\mathbf{w}_{\text{C}}\|^2 + \beta\langle\mathbf{w}_{\text{C}}, \mathbf{x}_{\ell+1}\rangle^2 + \sum_{i=1}^{\ell}(y_i - \langle\mathbf{w}_{\text{C}}, \mathbf{x}_i\rangle)^2, \tag{24}$$

where $0 \leq \beta \leq 1$. It is immediately clear that when $\beta = 0$ CKAAR should behave exactly like KRR and conversely like KAAR when $\beta = 1$. When β is somewhere in between it is hoped that CKAAR will output predictions that are not as rigid as those of KAAR and do not fluctuate as much as those of KRR. In a way, it can be said that we are using some previous knowledge about the 'noisiness' of the data to (hopefully) predict better.

Letting $\mathbf{w} = \mathbf{w}_{\text{C}}$, we can express (24) in matrix notation to give us

$$\begin{aligned}\mathcal{L}_{\text{C}} &= \alpha(\mathbf{w}'\mathbf{w}) + \beta(\mathbf{w}'\mathbf{x}_{\ell+1})^2 + (\mathbf{y} - \mathbf{X}\mathbf{w})^2 \\ &= \alpha(\mathbf{w}'\mathbf{w}) + (\widetilde{\mathbf{y}} - \widehat{\mathbf{X}}\mathbf{w})^2 \\ &= \alpha(\mathbf{w}'\mathbf{w}) + \widetilde{\mathbf{y}}'\widetilde{\mathbf{y}} - 2\mathbf{w}'\widehat{\mathbf{X}}'\widetilde{\mathbf{y}} + \mathbf{w}'\widehat{\mathbf{X}}'\widehat{\mathbf{X}}\mathbf{w},\end{aligned} \tag{25}$$

where $\widehat{\mathbf{X}} = (\mathbf{X}', \sqrt{\beta}\,\mathbf{x}_{\ell+1})'$ and $\widetilde{\mathbf{y}} = (\mathbf{y}', 0)'$. If we differentiate (25) with respect to $\mathbf{w}$, divide throughout by 2 and set it equal to 0 we get

$$\frac{1}{2}\frac{\partial\mathcal{L}_{\text{C}}}{\partial\mathbf{w}} = \alpha\mathbf{w} - \widehat{\mathbf{X}}'\widetilde{\mathbf{y}} + \widehat{\mathbf{X}}'\widehat{\mathbf{X}}\mathbf{w} = 0, \tag{26}$$

which means that the CKAAR solution ($\mathbf{w}_{\text{C}}$) to the regression problem for a new example $\mathbf{x}_{\ell+1}$ is

$$\mathbf{w}_{\text{C}} = (\alpha\mathbf{I} + \widehat{\mathbf{X}}'\widehat{\mathbf{X}})^{-1}\widehat{\mathbf{X}}'\widetilde{\mathbf{y}}. \tag{27}$$

The solution we have just derived is for the linear case only. To handle the nonlinear case we can apply a transformation ϕ to all the signals such that they are mapped to a feature space where we proceed to find a linear solution. Performing these transformations and finding the solution in feature space could be very computationally expensive and sometimes not possible. Therefore, we follow Gammerman et al. [2004] to formulate our solution in dual variables so that it can be used with kernels. Let the new signal be $\mathbf{x} = \mathbf{x}_{\ell+1}$, $\mathbf{M} = (\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_\ell), \sqrt{\beta}\,\phi(\mathbf{x}))'$, and $\boldsymbol{\omega}_{\text{C}}$ be the CKAAR solution in feature space. From (27) we know that

$$\begin{aligned}\gamma_{\text{CKAAR}} &= \boldsymbol{\omega}_{\text{C}}'\phi(\mathbf{x}) \\ &= ((\alpha\mathbf{I} + \mathbf{M}'\mathbf{M})^{-1}\mathbf{M}'\widetilde{\mathbf{y}})'\phi(\mathbf{x}) \\ &= \phi(\mathbf{x})'(\alpha\mathbf{I} + \mathbf{M}'\mathbf{M})^{-1}\mathbf{M}'\widetilde{\mathbf{y}}.\end{aligned}$$

By Lemma 3.3 we know that $(\alpha\mathbf{I} + \mathbf{M}'\mathbf{M})^{-1}\mathbf{M}' = \mathbf{M}'(\alpha\mathbf{I} + \mathbf{M}\mathbf{M}')^{-1}$. Therefore,

$$\begin{aligned}\gamma_{\text{CKAAR}} &= \phi(\mathbf{x})'\mathbf{M}'(\alpha\mathbf{I} + \mathbf{M}\mathbf{M}')^{-1}\widetilde{\mathbf{y}} \\ &= ((\alpha\mathbf{I} + \mathbf{M}\mathbf{M}')^{-1}\widetilde{\mathbf{y}})'\mathbf{M}\phi(\mathbf{x}) \\ &= \widetilde{\mathbf{y}}'(\alpha\mathbf{I} + \mathbf{M}\mathbf{M}')^{-1}\mathbf{M}\phi(\mathbf{x}).\end{aligned}$$

As can be seen we have ended up with a formulation where all the signals appear in dot products, that is, the dual formulation. We can now replace these dot products with kernel functions which

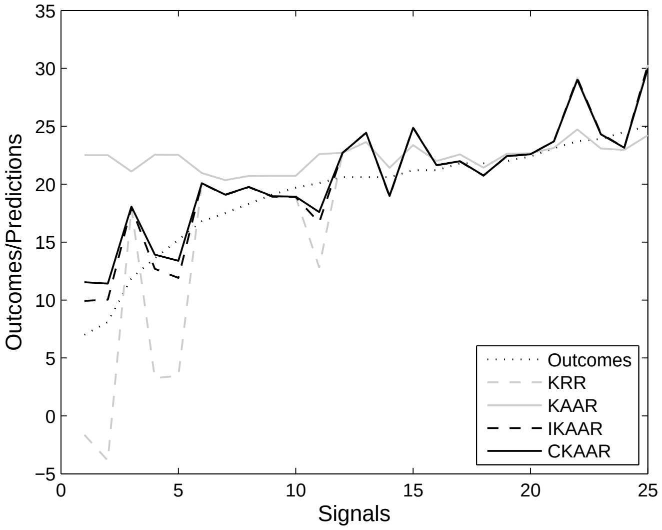

Figure 3 KRR, KAAR, IKAAR and CKAAR approximating a signal-outcome behaviour.

by definition are dot products in some feature space (ϕ is the transformation performed by the kernel). In this way we will effectively be doing linear regression in the feature space induced by the chosen kernel. A prediction for a new signal $\mathbf{x} = \mathbf{x}_{\ell+1}$ using the kernel version of CKAAR is

$$\gamma_{\text{CKAAR}} = \widetilde{\mathbf{y}}'(\alpha\mathbf{I} + \widehat{\mathbf{K}})^{-1}\widehat{\mathbf{k}}, \tag{28}$$

where

$$\widehat{\mathbf{K}} = \begin{bmatrix} k(\mathbf{x}_1,\mathbf{x}_1) & \dots & k(\mathbf{x}_1,\mathbf{x}_\ell) & \sqrt{\beta}\,k(\mathbf{x}_1,\mathbf{x}) \\ \vdots & \ddots & \vdots & \vdots \\ k(\mathbf{x}_\ell,\mathbf{x}_1) & \dots & k(\mathbf{x}_\ell,\mathbf{x}_\ell) & \sqrt{\beta}\,k(\mathbf{x}_\ell,\mathbf{x}) \\ \sqrt{\beta}\,k(\mathbf{x},\mathbf{x}_1) & \dots & \sqrt{\beta}\,k(\mathbf{x},\mathbf{x}_\ell) & \beta\,k(\mathbf{x},\mathbf{x}) \end{bmatrix}, \text{ and } \widehat{\mathbf{k}} = \begin{bmatrix} k(\mathbf{x}_1,\mathbf{x}) \\ \vdots \\ k(\mathbf{x}_\ell,\mathbf{x}) \\ \sqrt{\beta}\,k(\mathbf{x},\mathbf{x}) \end{bmatrix}.$$

Clearly, $(\alpha\mathbf{I} + \widehat{\mathbf{K}})$ is still positive definite since $\widehat{\mathbf{K}}$ is a Gram matrix of vectors in Hilbert space and one of them happens to be multiplied by $\sqrt{\beta}$.

4 Experimental Results

In this section we will be presenting the performance of our new methods in relation to KRR and KAAR on several datasets. However, before doing that we will revisit the motivation of our research and show a new version of Figure 1 with IKAAR and CKAAR predictions included. For these results, which are shown in Figure 3, the CKAAR control parameter $\beta = 0.01$ and IKAAR's chosen iteration $n = 88$. As can be seen, both IKAAR and CKAAR approximate the true outcomes better than either KRR or KAAR, having a square loss of 7.44 and 7.51 respectively. In addition, the variance of IKAAR's predictions is 26.68, whereas that of CKAAR is 21.92. This means that our new methods do not fluctuate as much as KRR and are not as rigid as KAAR. This behaviour was indeed the objective of our research.

4.1 Method

Our experimentation method follows that of Drucker et al. [1997]:

1. Split a random permutation of the dataset into three parts: a training set, a validation set, and a testing set.

2. Train the regression technique on the training set using several combinations of parameters (for example the kernel parameters). This gives us several regressors.

3. Test the performance of all the different regressors on the validation set.

4. Choose the regressor (more specifically, the set of parameters) that performs best on the validation set. Let us call this the best regressor.

5. Train the best regressor on the training set (ignore the validation set) and measure the mean loss it suffers on the testing set.

6. Repeat steps 1 to 5 a specified number of times.

7. Output the average of the mean losses of the regression technique over all the runs.

It is worth noting that in our experiments the actual value of α is calculated by multiplying the specified parameter by the mean of the diagonal (i.e. the trace divided by the dimension) of the kernel matrix. This is done so that the values added to the diagonal do not overshadow the rest of the values in the kernel matrix or conversely, to be large enough to make a difference.

It is always a good idea to somehow normalise or standardise the data prior to applying an algorithm to it. Features that are too big can cause computational problems and a feature that is consistently much larger than another one may be given undue extra importance. Since the spline kernels require that all the features in the signals be nonnegative we chose to normalise the features to the interval [0,1]. Let $\mathbf{X}$ be the matrix containing all the training signals (ℓ in total) in the dataset, one per row. It follows that every column of $\mathbf{X}$ corresponds to all the values of a particular feature in the training set. Let x_{ij} be the element at the ith row and the jth column of $\mathbf{X}$. Given a signal $\mathbf{z}$ of length n, the normalised version of its jth element z_j which we shall denote with $\overline{z}_j$, is calculated by

$$\overline{z}_j = \frac{z_j - m_j}{r_j},$$

where $m_j = \min_{i=1}^{\ell}(x_{ij})$ and $r_j = \left(\max_{i=1}^{\ell}(x_{ij}) - \min_{i=1}^{\ell}(x_{ij})\right)$. Therefore, the normalised version of $\mathbf{z}$ is $\overline{\mathbf{z}} = (\overline{z}_1, \overline{z}_2, \ldots, \overline{z}_n)'$.

The translation of the outcomes vector $\mathbf{y}$ such that it has a mean of 0 is done by taking the difference between each element and the mean of $\mathbf{y}$. Let y_i be the ith element of $\mathbf{y}$ and $\overline{y}_i$ be its corresponding translated value. We carry on the translation like so:

$$\overline{y}_i = y_i - \frac{\sum_{j=1}^{\ell}(y_j)}{\ell}.$$

4.2 Statistical Significance

It is not always obvious to decide whether a difference in some results is really an improvement or not, especially if this difference is small. For instance, could this improvement be due to chance alone? There exist several statistical tests that output the probability that the difference happens by chance (known as a p-value) and therefore is not significant. These statistical significance tests can be broadly split up into two: parametric tests and nonparametric tests. The first assume that the differences follow a particular distribution (typically the normal distribution) while the latter do not make such assumptions. Since we do not know what the distribution of the differences between two methods is, we use nonparametric tests in this paper[6]. Specifically, we use the Fisher

[6] Preliminary tests using the Kolmogorov-Smirnov test for normality (see Hollander and Wolfe [1973]) show that the distribution of the differences between the results of two methods in our experiments is in fact not normal.

Sign Test (FST) and the Wilcoxon Signed Rank Test (WSRT) (see, for example, Hollander and Wolfe [1973]). The FST answers the question of how often a method is better than another and whether it is significant, while WSRT answer the question of how much it is better and once again whether it is significant. More details on these two significance tests are given below. By convention, a p-value of less than 0.05 (or 5%) is taken to mean that the difference is significant.

4.2.1 The Fisher Sign Test

The Fisher Sign Test (FST) is one of the simplest tests available. The only assumption it makes is that the probability of observing a positive difference is equal to that of observing a negative one and that the differences have median value 0. This assumption is the FST's null hypothesis, and therefore the probability that it holds for the given data is calculated.

The procedure for this test is as follows. Count the number of positive (c^+) and negative differences (c^-), ignoring all zero values. The null hypothesis states that approximately 50% of the differences should be negative and the rest should be positive. So what is the probability that $m = \min(c^+, c^-)$ out of $t = c^+ + c^-$ trials turn out to be positive/negative just by chance? Since c^+ and c^- are Binomially distributed, this is found with the following formula:

$$\text{p-value} \leq 2 \times \sum_{i=0}^{m} \binom{t}{i} \times \frac{1}{2^t}.$$

4.2.2 The Wilcoxon Signed Rank Test

The Wilcoxon Signed Rank Test (WSRT) is a significance test that is much more sensitive than the Fisher Sign Test since in addition to the sign of the differences, it also considers their size. The WSRT assumes that it is possible to sort the differences and that they are mutually independent and come from a continuous population that is symmetric about zero. In our case both these assumptions are true.

To carry out this test, the absolute values of the differences are ranked from smallest to greatest, ignoring all zero differences. Ties are given the same rank by averaging the corresponding ranks. To the ranks the original sign of the differences is then affixed (by multiplying them with -1 or $+1$) and the sum of all positive ranks (r^+) is taken. The total number of differences is counted (t) keeping in mind that zero differences are ignored. With these two statistics, r^+ and t it is possible to get the corresponding p-value from an appropriate table.

4.3 Results

We conducted experiments on one artificial dataset called the Mexican Hat dataset and on several real-world datasets. We used four kernels for most of our experiments: a polynomial kernel, a spline kernel, an ANOVA spline kernel, and a Gaussian RBF kernel (for more information on these kernels see, for example, Schölkopf and Smola [2002]). The results and more details on these datasets and their respective experiments (including parameters used) are given in the sections below. Note that for every dataset the average of the mean square losses per run (MSE) and the corresponding variance (Var) of every regression method is reported. Moreover, p-values are given for the result of every method as compared to KRR and KAAR, representing the probability that the differences between them happen by chance (recall that by convention a p-value of less than 0.05 is taken to mean that the difference is significant).

Abalone										
Results by Kernel			WSRT p-values				FST p-values			
Anova	MSE	Var	KRR		KAAR		KRR		KAAR	
KRR	5.078	1.107			+	4.66×10^{-4}			+	2.10×10^{-2}
KAAR	4.806	0.117	+	4.66×10^{-4}			+	2.10×10^{-2}		
IKAAR	**4.793**	0.121	+	3.59×10^{-6}	−	8.74×10^{-1}	+	4.61×10^{-5}	−	5.69×10^{-2}
CKAAR	4.990	0.960	−	4.54×10^{-1}	+	3.62×10^{-2}	−	9.20×10^{-1}	−	8.15×10^{-1}
Spline	MSE	Var	KRR		KAAR		KRR		KAAR	
KRR	5.333	3.242			+	3.86×10^{-6}			+	3.52×10^{-3}
KAAR	**4.868**	0.101	+	3.86×10^{-6}			+	3.52×10^{-3}		
IKAAR	4.868	0.099	+	9.24×10^{-9}	−	2.12×10^{-1}	+	1.12×10^{-9}	+	4.09×10^{-4}
CKAAR	4.989	0.278	+	3.05×10^{-2}	+	3.15×10^{-2}	−	8.86×10^{-2}	−	1.00×10^{0}
Poly	MSE	Var	KRR		KAAR		KRR		KAAR	
KRR	16928.731	2.387×10^{10}			+	8.12×10^{-20}			+	3.06×10^{-17}
KAAR	**4.822**	0.107	+	8.12×10^{-20}			+	3.06×10^{-17}		
IKAAR	4.879	0.201	+	2.89×10^{-21}	+	8.73×10^{-8}	+	5.07×10^{-20}	+	4.83×10^{-13}
CKAAR	173.134	1.341×10^{6}	+	7.94×10^{-7}	+	1.02×10^{-4}	+	8.74×10^{-4}	+	2.05×10^{-2}
RBF	MSE	Var	KRR		KAAR		KRR		KAAR	
KRR	5.734	14.326			+	2.32×10^{-9}			+	5.51×10^{-8}
KAAR	**4.720**	0.110	+	2.32×10^{-9}			+	5.51×10^{-8}		
IKAAR	4.855	0.312	+	7.47×10^{-6}	+	3.53×10^{-7}	+	1.83×10^{-5}	+	3.31×10^{-5}
CKAAR	5.538	14.143	+	6.62×10^{-3}	+	9.06×10^{-6}	−	8.86×10^{-2}	+	8.78×10^{-4}

Table 1 Results for the Abalone dataset. The statistical significance columns show whether the difference in results between a method and KRR or KAAR is significant (+) or not (−). The corresponding p-values are also given.

4.3.1 The Abalone Dataset

The age in years of an abalone is determined by counting the number of rings in a cross-section of its shell through a microscope and adding 1.5. The goal of the Abalone dataset [Newman et al., 1998] is to predict the ages of abalones from 8 features corresponding to physical measurements. These measurements which include the length and weight are relatively easy to obtain. See the results in Table 1.

Dataset size: 4177
Training set size: 1000
Validation set size: 100
Testing set size: 3077
Runs: 100
α: $\{2^{-16}, 2^{-14}, \ldots, 2^{-4}\}$
CKAAR's β: $\{0, 0.3, 0.5, 0.7, 0.8, 0.9, 0.95, 0.99, 1\}$
IKAAR's iteration: $\{1, 21, \ldots, 201\}$
Polynomial kernel degree: $\{2, 4, \ldots, 8\}$
ANOVA spline kernel order: $\{2, 4, \ldots, 8\}$

4.3.2 The Auto-MPG Dataset

The Auto-MPG dataset [Newman et al., 1998] contains details of cars and their performance in terms of their fuel consumption in miles per gallon (mpg). In our experiment the mpg of a car given its attributes was predicted. 7 attributes were used, including features like the number of cylinders in the car's engine and its weight. Signals that have missing values were not included in the experiment. The results are in Table 2.

Dataset size: 392

Auto-MPG										
Results by Kernel			WSRT p-values				FST p-values			
Anova	MSE	Var	KRR		KAAR		KRR		KAAR	
KRR	**8.033**	3.012			+	1.01×10^{-27}			+	6.45×10^{-24}
KAAR	12.719	92.291	+	1.01×10^{-27}			+	6.45×10^{-24}		
IKAAR	8.053	2.627	−	5.53×10^{-1}	+	3.02×10^{-26}	−	1.42×10^{-1}	+	3.21×10^{-19}
CKAAR	8.047	2.521	−	1.30×10^{-1}	+	3.38×10^{-27}	−	1.33×10^{-1}	+	2.38×10^{-23}
Spline	MSE	Var	KRR		KAAR		KRR		KAAR	
KRR	**8.139**	3.428			+	1.10×10^{-28}			+	7.97×10^{-27}
KAAR	13.515	74.498	+	1.10×10^{-28}			+	7.97×10^{-27}		
IKAAR	8.274	4.062	−	9.25×10^{-1}	+	8.46×10^{-28}	+	3.96×10^{-2}	+	1.25×10^{-22}
CKAAR	8.170	2.861	−	6.75×10^{-2}	+	1.36×10^{-28}	−	1.33×10^{-1}	+	3.16×10^{-28}
Poly	MSE	Var	KRR		KAAR		KRR		KAAR	
KRR	8.870	9.804			+	5.83×10^{-24}			+	1.25×10^{-22}
KAAR	13.565	33.258	+	5.83×10^{-24}			+	1.25×10^{-22}		
IKAAR	**8.826**	6.971	−	8.60×10^{-1}	+	2.79×10^{-25}	−	1.75×10^{-1}	+	2.73×10^{-20}
CKAAR	8.838	8.647	−	9.65×10^{-2}	+	1.81×10^{-25}	−	5.69×10^{-2}	+	1.24×10^{-23}
RBF	MSE	Var	KRR		KAAR		KRR		KAAR	
KRR	8.379	5.803			+	7.17×10^{-25}			+	2.73×10^{-20}
KAAR	13.593	71.111	+	7.17×10^{-25}			+	2.73×10^{-20}		
IKAAR	**8.361**	4.717	−	1.73×10^{-1}	+	1.03×10^{-24}	−	2.41×10^{-1}	+	1.75×10^{-19}
CKAAR	8.370	5.466	+	2.03×10^{-2}	+	3.85×10^{-25}	+	2.10×10^{-2}	+	2.01×10^{-21}

Table 2 Results for the Auto-MPG dataset. The statistical significance columns show whether the difference in results between a method and KRR or KAAR is significant (+) or not (−). The corresponding p-values are also given.

Training set size: 200
Validation set size: 50
Testing set size: 142
Runs: 100
α: $\{2^{-10}, 2^{-9}, \ldots, 2^{-5}\}$
CKAAR's β: $\{0, 0.01, 0.05, 0.1, 0.5, 0.9, 0.95, 0.99, 1\}$
IKAAR's iteration: $\{1, 11, \ldots, 81\}$
Polynomial kernel degree: $\{2, 3, \ldots, 6\}$
ANOVA spline kernel order: $\{2, 3, \ldots, 7\}$
RBF kernel σ: $\{2^{-10}, 2^{-8}, \ldots, 2^{2}\}$

4.3.3 The Auto-Price Dataset

The aim for the Auto-Price dataset [Newman et al., 1998] is to predict the price of a car from 15 features which include characteristics like length, weight, number of doors, engine type and insurance risk rating. Those signals in the original dataset that had missing features and 10 nominal features were removed. The results are in Table 3.

Dataset size: 159
Training set size: 100
Validation set size: 30
Testing set size: 29
Runs: 100
α: $\{2^{-10}, 2^{-9}, \ldots, 2^{-5}\}$
CKAAR's β: $\{0, 0.01, 0.05, 0.1, 0.5, 0.9, 0.95, 0.99, 1\}$
IKAAR's iteration: $\{1, 11, \ldots, 101\}$
Polynomial kernel degree: $\{2, 3, \ldots, 6\}$

Auto-Price										
Results by Kernel			WSRT p-values				FST p-values			
Anova	MSE×10^6	Var×10^{13}	KRR		KAAR		KRR		KAAR	
KRR	8.116	18.482			+	8.97×10^{-16}			+	2.54×10^{-16}
KAAR	10.922	4.771	+	8.88×10^{-16}			+	2.54×10^{-16}		
IKAAR	**7.264**	3.722	+	1.21×10^{-3}	+	3.25×10^{-14}	+	1.32×10^{-4}	+	4.22×10^{-13}
CKAAR	8.810	19.522	+	3.45×10^{-2}	+	5.30×10^{-14}	−	1.93×10^{-1}	+	4.83×10^{-13}
Spline	MSE×10^6	Var×10^{13}	KRR		KAAR		KRR		KAAR	
KRR	11.184	34.103			+	1.13×10^{-17}			+	2.01×10^{-21}
KAAR	21.888	16.523	+	1.13×10^{-17}			+	2.01×10^{-21}		
IKAAR	**10.031**	6.888	−	1.29×10^{-1}	+	3.00×10^{-29}	+	2.80×10^{-2}	+	1.59×10^{-28}
CKAAR	11.657	32.250	+	2.38×10^{-2}	+	1.93×10^{-20}	−	6.17×10^{-1}	+	2.63×10^{-25}
Poly	MSE×10^6	Var×10^{13}	KRR		KAAR		KRR		KAAR	
KRR	**7.478**	7.821			+	1.81×10^{-19}			+	3.06×10^{-17}
KAAR	13.515	6.984	+	1.81×10^{-19}			+	3.06×10^{-17}		
IKAAR	7.873	3.931	+	5.56×10^{-3}	+	1.11×10^{-17}	+	5.09×10^{-4}	+	2.46×10^{-13}
CKAAR	8.052	8.525	+	1.84×10^{-4}	+	1.71×10^{-19}	+	8.74×10^{-4}	+	2.54×10^{-16}
RBF	MSE×10^6	Var×10^{13}	KRR		KAAR		KRR		KAAR	
KRR	**7.126**	1.773			+	6.36×10^{-11}			+	5.64×10^{-7}
KAAR	9.550	3.095	+	6.36×10^{-11}			+	5.64×10^{-7}		
IKAAR	7.751	3.166	+	1.91×10^{-3}	+	1.74×10^{-8}	+	4.04×10^{-3}	+	3.48×10^{-7}
CKAAR	7.630	2.558	−	5.77×10^{-2}	+	6.34×10^{-8}	−	4.84×10^{-1}	+	1.92×10^{-6}

Table 3 Results for the Auto-Price dataset. The statistical significance columns show whether the difference in results between a method and KRR or KAAR is significant (+) or not (−). The corresponding p-values are also given.

ANOVA spline kernel order: $\{2, 4, 6, 8, 10, 12, 15\}$
RBF kernel σ: $\{2^{-10}, 2^{-8}, \ldots, 2^{2}\}$

4.3.4 The Boston Housing Dataset

The Boston Housing dataset [Newman et al., 1998] concerns the prices of houses in the suburbs of Boston. A signal corresponds to a particular suburb and contains 13 attributes, including features like the amount of air pollution and the average number of rooms. An outcome is simply the median price of the houses in thousands of dollars. We used the same partitioning of the dataset as in Saunders et al. [1998] so our results are directly comparable to those reported there. The results are in Table 4.

Dataset size: 506
Training set size: 401
Validation set size: 80
Testing set size: 25
Runs: 100
α: $\{2^{-10}, 2^{-9}, \ldots, 2^{-5}\}$
CKAAR's β: $\{0, 0.01, 0.05, 0.1, 0.5, 0.9, 0.95, 0.99, 1\}$
IKAAR's iteration: $\{1, 11, \ldots, 151\}$
Polynomial kernel degree: $\{4, 5\}$
ANOVA spline kernel order: $\{2, 4, 6, 8, 10, 13\}$
RBF kernel σ: $\{2^{-10}, 2^{-8}, \ldots, 2^{2}\}$

4.3.5 The Gaze Dataset

The outcomes in the Gaze dataset [Quiñonero-Candela et al., to appear in 2006] are the horizontal positions of targets displayed on a computer monitor measured in pixels. The corresponding

Boston Housing										
Results by Kernel			WSRT p-values				FST p-values			
Anova	MSE	Var	KRR		KAAR		KRR		KAAR	
KRR	7.307	8.523			+	1.43×10^{-27}			+	1.25×10^{-22}
KAAR	21.818	209.934	+	1.43×10^{-27}			+	1.25×10^{-22}		
IKAAR	**7.207**	11.404	+	7.13×10^{-3}	+	1.01×10^{-27}	+	1.79×10^{-3}	+	1.25×10^{-22}
CKAAR	7.230	12.965	−	6.70×10^{-2}	+	5.85×10^{-28}	−	5.69×10^{-2}	+	6.45×10^{-24}
Spline	MSE	Var	KRR		KAAR		KRR		KAAR	
KRR	7.331	8.786			+	1.74×10^{-28}			+	6.45×10^{-24}
KAAR	23.967	229.143	+	1.74×10^{-28}			+	6.45×10^{-24}		
IKAAR	7.123	10.854	+	1.34×10^{-2}	+	2.67×10^{-28}	+	1.20×10^{-2}	+	6.45×10^{-24}
CKAAR	**7.102**	12.793	+	5.05×10^{-4}	+	6.78×10^{-29}	+	3.52×10^{-3}	+	2.63×10^{-25}
Poly	MSE	Var	KRR		KAAR		KRR		KAAR	
KRR	7.924	9.916			+	1.37×10^{-26}			+	2.01×10^{-21}
KAAR	20.870	190.051	+	1.37×10^{-26}			+	2.01×10^{-21}		
IKAAR	8.260	22.133	−	8.20×10^{-1}	+	3.02×10^{-26}	−	2.71×10^{-1}	+	2.73×10^{-20}
CKAAR	**7.862**	11.995	−	9.59×10^{-2}	+	4.36×10^{-27}	−	1.33×10^{-1}	+	1.25×10^{-22}
RBF	MSE	Var	KRR		KAAR		KRR		KAAR	
KRR	8.375	18.111			+	1.40×10^{-16}			+	2.61×10^{-12}
KAAR	12.507	32.751	+	1.40×10^{-16}			+	2.61×10^{-12}		
IKAAR	8.298	18.402	+	2.95×10^{-2}	+	3.23×10^{-17}	+	1.60×10^{-2}	+	8.28×10^{-14}
CKAAR	**8.297**	18.818	−	8.95×10^{-1}	+	6.18×10^{-17}	−	7.64×10^{-1}	+	8.28×10^{-14}

Table 4 Results for the Boston Housing dataset. The statistical significance columns show whether the difference in results between a method and KRR or KAAR is significant (+) or not (−). The corresponding p-values are also given.

12 features are measurements from head mounted cameras that focus on markers on the monitor and estimate the positions of the eyes of the subject looking at the monitor. Since the cameras occasionally lose their calibration, the dataset contains several severe outliers. Note that only the training and validation sets were used from the original dataset, since the outcomes of the testing set were not accessible. We did not remove any of the signals for our experiments (not even the outliers). See Table 5 for the results.

Dataset size: 450
Training set size: 350
Validation set size: 70
Testing set size: 30
Runs: 100
α: $\{2^{-10}, 2^{-9}, \ldots, 2^{-5}\}$
CKAAR's β: $\{2^{-15}, 2^{-14}, 2^{0}\}$
IKAAR's iteration: $\{1, 11, \ldots, 101\}$
Polynomial kernel degree: $\{2, 4, \ldots, 8\}$
ANOVA spline kernel order: $\{2, 4, \ldots, 12\}$
RBF kernel σ: $\{2^{-10}, 2^{-8}, \ldots, 2^{2}\}$

4.3.6 The Mexican Hat Dataset

The artificial Mexican Hat dataset is generated by the following function

$$y = \frac{\sin(|x|)}{|x|} + \varepsilon, \tag{29}$$

taking x from the interval $[-10, 10]$ and ε being some noise. Plotting this x against y gives a graph that somewhat resembles the cross section of a tradition Mexican hat, hence the name. In our

Gaze										
Results by Kernel			WSRT p-values				FST p-values			
Anova	MSE×10^3	Var×10^6	KRR		KAAR		KRR		KAAR	
KRR	2.259	7.429			+	3.16×10^{-30}			+	3.16×10^{-30}
KAAR	9.000	23.376	+	3.16×10^{-30}			+	3.16×10^{-30}		
IKAAR	**2.030**	0.752	−	1.14×10^{-1}	+	1.58×10^{-30}	−	3.20×10^{-1}	+	1.58×10^{-30}
CKAAR	2.227	7.432	+	6.20×10^{-4}	+	3.16×10^{-30}	+	2.55×10^{-4}	+	3.16×10^{-30}
Spline	MSE×10^3	Var×10^6	KRR		KAAR		KRR		KAAR	
KRR	**2.447**	10.051			+	3.16×10^{-30}			+	3.16×10^{-30}
KAAR	12.676	25.380	+	3.16×10^{-30}			+	3.16×10^{-30}		
IKAAR	33.019	95646.442	−	1.01×10^{-1}	+	8.32×10^{-24}	−	1.46×10^{-1}	+	1.59×10^{-28}
CKAAR	2.989	81.617	+	7.29×10^{-3}	+	8.32×10^{-24}	+	3.52×10^{-3}	+	1.59×10^{-28}
Poly	MSE×10^3	Var×10^6	KRR		KAAR		KRR		KAAR	
KRR	12.786	10968.248			+	5.08×10^{-20}			+	1.25×10^{-22}
KAAR	5.748	6.569	+	5.08×10^{-20}			+	1.25×10^{-22}		
IKAAR	**2.057**	0.595	+	1.34×10^{-2}	+	8.68×10^{-29}	+	2.95×10^{-2}	+	7.97×10^{-27}
CKAAR	11.905	9718.385	+	9.39×10^{-4}	+	1.28×10^{-23}	+	3.22×10^{-5}	+	7.97×10^{-27}
RBF	MSE×10^3	Var×10^6	KRR		KAAR		KRR		KAAR	
KRR	2.744	17.087			+	8.12×10^{-20}			+	2.01×10^{-21}
KAAR	9.694	22.091	+	8.12×10^{-20}			+	2.01×10^{-21}		
IKAAR	**2.386**	7.301	−	1.10×10^{-1}	+	8.92×10^{-22}	−	3.20×10^{-1}	+	6.45×10^{-24}
CKAAR	2.548	12.845	+	3.57×10^{-2}	+	1.04×10^{-20}	+	3.52×10^{-2}	+	6.45×10^{-24}

Table 5 Results for the Gaze dataset. The statistical significance columns show whether the difference in results between a method and KRR or KAAR is significant (+) or not (−). The corresponding p-values are also given.

experiments we took 100 signals from the interval (and their corresponding outcomes) starting from -10 and going up to 10 with a step of 0.2, skipping 0. Two separate experiments were performed using noise (ε) taken from normal distributions with standard deviations 0.2 and 0.5. The results are in Table 6. Note that the testing outcomes are not corrupted by noise, therefore the losses reported are due to the model error only.

Dataset size: 100
Training set size: 50
Validation set size: 30
Testing set size: 20
Runs: 1000
α: 0.1
CKAAR's β: $\{0, 0.01, 0.02, \ldots, 1\}$
IKAAR's iteration: $\{1, 2, \ldots, 5\}$
Polynomial kernel degree: 6

4.3.7 The Relative CPU Performance Dataset

The Relative CPU Performance dataset [Newman et al., 1998] concerns itself with the problem of predicting the relative performance of a CPU given 6 features which include the size of its cache memory and its cycles per second. Two nominal features were removed. See Table 7 for the results.

Dataset size: 209
Training set size: 150
Validation set size: 34
Testing set size: 25
Runs: 100
α: $\{2^{-10}, 2^{-9}, \ldots, 2^{-5}\}$
CKAAR's β: $\{0, 0.01, 0.05, 0.1, 0.5, 0.9, 0.95, 0.99, 1\}$

Mexican Hat with noise from $\mathcal{N}(0, 0.2)$										
Results by Kernel			WSRT p-values				FST p-values			
Spline	MSE×10^{-2}	Var×10^{-4}	KRR		KAAR		KRR		KAAR	
KRR	7.555	4.359			+	6.96×10^{-71}			+	1.42×10^{-67}
KAAR	7.651	4.727	+	6.96×10^{-71}			+	1.42×10^{-67}		
IKAAR	7.552	4.453	+	4.89×10^{-4}	+	1.05×10^{-101}	+	3.02×10^{-9}	+	4.59×10^{-49}
CKAAR	**7.526**	4.413	+	1.70×10^{-53}	+	4.20×10^{-134}	+	7.67×10^{-105}	+	1.56×10^{-119}
Poly	MSE×10^{-2}	Var×10^{-4}	KRR		KAAR		KRR		KAAR	
KRR	9.472	5.827			+	2.06×10^{-26}			+	5.63×10^{-13}
KAAR	9.376	6.099	+	2.06×10^{-26}			+	5.63×10^{-13}		
IKAAR	9.376	5.931	+	1.37×10^{-58}	−	8.53×10^{-1}	+	1.48×10^{-59}	+	1.77×10^{-2}
CKAAR	**9.374**	5.912	+	7.98×10^{-78}	−	1.16×10^{-1}	+	1.27×10^{-79}	−	1.45×10^{-1}
Mexican Hat with noise from $\mathcal{N}(0, 0.5)$										
Results by Kernel			WSRT p-values				FST p-values			
Spline	MSE×10^{-2}	Var×10^{-4}	KRR		KAAR		KRR		KAAR	
KRR	9.182	8.548			+	7.50×10^{-3}			+	9.52×10^{-9}
KAAR	9.163	8.831	+	7.50×10^{-3}			+	9.52×10^{-9}		
IKAAR	9.127	8.738	+	3.50×10^{-4}	+	2.15×10^{-32}	−	6.35×10^{-1}	+	6.89×10^{-15}
CKAAR	**9.116**	8.727	+	4.87×10^{-25}	+	1.67×10^{-47}	+	4.98×10^{-53}	+	9.14×10^{-42}
Poly	MSE×10^{-2}	Var×10^{-4}	KRR		KAAR		KRR		KAAR	
KRR	11.362	11.785			+	9.08×10^{-109}			+	5.42×10^{-64}
KAAR	**11.037**	11.232	+	9.08×10^{-109}			+	5.42×10^{-64}		
IKAAR	11.039	11.164	+	2.88×10^{-115}	+	2.26×10^{-11}	+	1.27×10^{-79}	+	3.51×10^{-13}
CKAAR	11.047	11.169	+	4.36×10^{-125}	+	3.38×10^{-3}	+	8.71×10^{-115}	−	2.09×10^{-1}

Table 6 Results for the Mexican Hat dataset. The statistical significance columns show whether the difference in results between a method and KRR or KAAR is significant (+) or not (−). The corresponding p-values are also given.

IKAAR's iteration: $\{1, 11, \ldots, 101\}$
Polynomial kernel degree: $\{2, 3, \ldots, 6\}$
ANOVA spline kernel order: $\{1, 2, \ldots, 6\}$
RBF kernel σ: $\{2^{-10}, 2^{-8}, \ldots, 2^{2}\}$

4.3.8 The Servo Dataset

For the Servo dataset [Newman et al., 1998] the problem is to predict the rise time of a servomechanism in terms of 4 features: two continuous gain settings and two discrete choices of mechanical linkages. See the results in Table 8.

Dataset size: 167
Training set size: 100
Validation set size: 40
Testing set size: 27
Runs: 100
α: $\{2^{-10}, 2^{-9}, \ldots, 2^{-5}\}$
CKAAR's β: $\{0, 0.01, 0.05, 0.1, 0.5, 0.9, 0.95, 0.99, 1\}$
IKAAR's iteration: $\{1, 11, \ldots, 101\}$
Polynomial kernel degree: $\{2, 3, \ldots, 6\}$
ANOVA spline kernel order: $\{1, 2, \ldots, 4\}$
RBF kernel σ: $\{2^{-10}, 2^{-8}, \ldots, 2^{2}\}$

4.3.9 The Wisconsin Prognostic Breast Cancer Dataset

In the Wisconsin Prognostic Breast Cancer dataset [Newman et al., 1998] the problem is to predict the time for a patient to recur (or her disease free time). 32 features are given, including charac-

Relative CPU Performance										
Results by Kernel			WSRT p-values				FST p-values			
Anova	MSE×10^3	Var×10^8	KRR		KAAR		KRR		KAAR	
KRR	**3.768**	0.250			+	4.60×10^{-10}			+	1.67×10^{-6}
KAAR	11.080	2.514	+	4.60×10^{-10}			+	1.67×10^{-6}		
IKAAR	6.871	1.573	−	5.77×10^{-1}	+	7.67×10^{-8}	−	5.58×10^{-2}	+	2.53×10^{-5}
CKAAR	5.472	1.011	−	8.74×10^{-1}	+	1.37×10^{-10}	−	1.00×10^{0}	+	5.51×10^{-8}
Spline	MSE×10^3	Var×10^8	KRR		KAAR		KRR		KAAR	
KRR	**4.337**	0.366			+	1.34×10^{-12}			+	1.12×10^{-9}
KAAR	15.050	4.031	+	1.34×10^{-12}			+	1.12×10^{-9}		
IKAAR	7.699	1.960	−	2.87×10^{-1}	+	5.68×10^{-14}	−	7.41×10^{-1}	+	1.86×10^{-9}
CKAAR	6.004	1.379	−	3.57×10^{-1}	+	5.50×10^{-14}	−	1.33×10^{-1}	+	1.12×10^{-9}
Poly	MSE×10^3	Var×10^8	KRR		KAAR		KRR		KAAR	
KRR	14.986	22.142			+	5.93×10^{-5}			+	3.22×10^{-5}
KAAR	13.596	3.452	+	5.93×10^{-5}			+	3.22×10^{-5}		
IKAAR	**9.101**	2.554	−	3.44×10^{-1}	+	1.93×10^{-7}	−	3.48×10^{-1}	+	1.92×10^{-6}
CKAAR	15.005	21.856	−	4.75×10^{-1}	+	3.09×10^{-6}	−	2.71×10^{-1}	+	4.69×10^{-6}
RBF	MSE×10^3	Var×10^8	KRR		KAAR		KRR		KAAR	
KRR	**5.835**	0.884			+	1.26×10^{-10}			+	4.34×10^{-9}
KAAR	9.986	1.957	+	1.26×10^{-10}			+	4.34×10^{-9}		
IKAAR	7.124	1.558	−	7.13×10^{-1}	+	9.28×10^{-7}	−	7.49×10^{-1}	+	6.34×10^{-6}
CKAAR	6.571	1.243	−	6.80×10^{-1}	+	3.33×10^{-11}	−	4.84×10^{-1}	+	1.79×10^{-9}

Table 7 Results for the Relative CPU Performance dataset. The statistical significance columns show whether the difference in results between a method and KRR or KAAR is significant (+) or not (−). The corresponding p-values are also given.

Servo										
Results by Kernel			WSRT p-values				FST p-values			
Anova	MSE	Var	KRR		KAAR		KRR		KAAR	
KRR	**0.443**	0.122			+	5.12×10^{-18}			+	3.06×10^{-17}
KAAR	0.625	0.190	+	5.12×10^{-18}			+	3.06×10^{-17}		
IKAAR	0.452	0.122	−	2.37×10^{-1}	+	1.52×10^{-16}	−	1.51×10^{-1}	+	3.30×10^{-15}
CKAAR	0.445	0.124	−	6.62×10^{-1}	+	3.17×10^{-19}	−	1.33×10^{-1}	+	3.21×10^{-19}
Spline	MSE	Var	KRR		KAAR		KRR		KAAR	
KRR	0.447	0.117			+	1.16×10^{-18}			+	1.91×10^{-15}
KAAR	0.651	0.206	+	1.16×10^{-18}			+	1.91×10^{-15}		
IKAAR	0.452	0.118	+	3.19×10^{-2}	+	1.45×10^{-17}	−	6.42×10^{-2}	+	9.83×10^{-14}
CKAAR	**0.445**	0.121	−	6.20×10^{-2}	+	1.33×10^{-20}	+	3.52×10^{-2}	+	3.21×10^{-19}
Poly	MSE	Var	KRR		KAAR		KRR		KAAR	
KRR	**0.530**	0.140			+	8.30×10^{-15}			+	2.61×10^{-12}
KAAR	0.693	0.211	+	8.44×10^{-15}			+	2.61×10^{-12}		
IKAAR	0.545	0.148	−	6.32×10^{-1}	+	2.35×10^{-13}	−	5.94×10^{-1}	+	1.29×10^{-12}
CKAAR	0.538	0.160	−	4.30×10^{-1}	+	4.17×10^{-19}	+	3.52×10^{-2}	+	3.06×10^{-17}
RBF	MSE	Var	KRR		KAAR		KRR		KAAR	
KRR	0.507	0.168			+	6.60×10^{-14}			+	4.83×10^{-13}
KAAR	0.673	0.226	+	6.59×10^{-14}			+	4.83×10^{-13}		
IKAAR	0.514	0.168	−	9.07×10^{-1}	+	7.27×10^{-13}	−	2.84×10^{-1}	+	5.78×10^{-11}
CKAAR	**0.505**	0.153	−	1.00×10^{0}	+	2.05×10^{-15}	−	9.20×10^{-1}	+	1.91×10^{-15}

Table 8 Results for the Servo dataset. The statistical significance columns show whether the difference in results between a method and KRR or KAAR is significant (+) or not (−). The corresponding p-values are also given.

Wisconsin Prognostic Breast Cancer										
Results by Kernel			WSRT p-values				FST p-values			
Anova	MSE×10^3	Var×10^4	KRR		KAAR		KRR		KAAR	
KRR	214708.449	4.610×10^{14}			+	4.14×10^{-13}			+	4.34×10^{-9}
KAAR	**1.153**	3.424	+	4.14×10^{-13}			+	4.34×10^{-9}		
IKAAR	238277.130	5.674×10^{14}	+	6.25×10^{-7}	+	1.59×10^{-4}	+	1.26×10^{-5}	+	2.88×10^{-2}
CKAAR	1.377	379.018	+	4.03×10^{-9}	+	4.92×10^{-5}	+	3.22×10^{-5}	+	3.66×10^{-4}
Spline	MSE×10^3	Var×10^4	KRR		KAAR		KRR		KAAR	
KRR	13.623	200501.264			+	2.94×10^{-13}			+	1.81×10^{-7}
KAAR	**1.140**	3.441	+	2.94×10^{-13}			+	1.81×10^{-7}		
IKAAR	1.169	4.645	+	4.73×10^{-12}	+	1.38×10^{-2}	+	5.64×10^{-7}	−	5.41×10^{-2}
CKAAR	1.172	5.148	+	1.37×10^{-10}	+	1.50×10^{-3}	+	3.22×10^{-5}	+	4.09×10^{-4}
Poly	MSE×10^3	Var×10^4	KRR		KAAR		KRR		KAAR	
KRR	1.529	82.730			+	5.18×10^{-19}			+	1.31×10^{-14}
KAAR	**1.147**	3.251	+	5.18×10^{-19}			+	1.31×10^{-14}		
IKAAR	1.188	5.266	+	7.43×10^{-15}	+	1.64×10^{-4}	+	4.39×10^{-12}	+	1.92×10^{-2}
CKAAR	1.196	7.602	+	1.09×10^{-14}	+	3.44×10^{-4}	+	2.70×10^{-10}	−	8.86×10^{-2}
RBF	MSE×10^3	Var×10^4	KRR		KAAR		KRR		KAAR	
KRR	1.159	3.678			+	2.26×10^{-8}			+	1.67×10^{-6}
KAAR	**1.112**	3.446	+	2.26×10^{-8}			+	1.67×10^{-6}		
IKAAR	1.140	4.148	+	4.39×10^{-3}	+	1.56×10^{-6}	+	2.31×10^{-3}	+	3.88×10^{-4}
CKAAR	1.139	3.775	+	2.42×10^{-2}	+	1.84×10^{-7}	−	9.20×10^{-1}	+	3.37×10^{-4}

Table 9 Results for the Wisconsin Prognostic Breast Cancer dataset. The statistical significance columns show whether the difference in results between a method and KRR or KAAR is significant (+) or not (−). The corresponding p-values are also given.

teristics of the cell nuclei and the tumour size. Four signals that had missing values and 2 features were removed. The results are in Table 9.

Dataset size: 194
Training set size: 100
Validation set size: 50
Testing set size: 44
Runs: 100
α: $\{2^{-10}, 2^{-9}, \ldots, 2^{-5}\}$
CKAAR's β: $\{0, 0.01, 0.05, 0.1, 0.5, 0.9, 0.95, 0.99, 1\}$
IKAAR's iteration: $\{1, 11, \ldots, 101\}$
Polynomial kernel degree: $\{2, 3, \ldots, 6\}$
ANOVA spline kernel order: $\{8, 16, \ldots, 32\}$
RBF kernel σ: $\{2^{-10}, 2^{-8}, \ldots, 2^{2}\}$

5 Conclusion

Below is a quick overview of the general performance of the four different regression methods we have analysed per dataset. To determine the relative performance of methods we took in consideration their mean square losses and whether the differences are statistically significant or not.

Abalone: KAAR and IKAAR are clearly the best methods for this dataset, being slightly better than CKAAR and much better than KRR.

Auto-MPG: KRR, IKAAR and CKAAR are the best methods for this dataset as there is no significant difference between them, while KAAR's strong regularisation seems to be overkill in this instance.

Auto-Price: For this dataset IKAAR and KRR perform equally well and are slightly better than CKAAR and KAAR.

Boston Housing: CKAAR and IKAAR are clearly the best methods for this dataset, with KRR and KAAR coming second and third respectively.

Gaze: IKAAR and CKAAR perform best for this dataset, with KRR close behind and KAAR last.

Mexican Hat: In both experiments (corrupted with different levels of noise), CKAAR is the best method, while IKAAR is the second best. KAAR comes in third, while KRR is the worst.

Relative CPU Performance: KRR, IKAAR and CKAAR are the best methods for this dataset as there is no significant difference between them. KAAR is the worst.

Servo: KRR, IKAAR and CKAAR perform equally well for this dataset and KAAR is worse.

Wisconsin Prognostic Breast Cancer: KAAR is clearly the best method for this dataset, with CKAAR and IKAAR coming in second and KRR last.

In the 10 experiments we carried out[7], CKAAR and IKAAR were the best in 7 of them. KRR and KAAR were the best in 4 and 2 cases respectively. This suggests that our new methods IKAAR and CKAAR are, in general, equivalent or better than KRR and KAAR. This follows from the fact that both our methods are generalisations of KRR and KAAR and can behave as any one of them or something in between that may give better performance. It is worth noting that in most of the experiments, the difference between our two methods was not statistically significant. Therefore, it can be said that CKAAR and IKAAR are more or less equivalent. This is understandable because they were both motivated by a common idea and they try to achieve the same thing, albeit in different ways.

One disadvantage of our methods is that an extra parameter has to be chosen for each of them: the iteration number for IKAAR and the control parameter β for CKAAR. So we are getting better performance at the expense of having to find good values for an extra parameter. In our experiments we chose these values by using validation, as we did for all the other parameters. Future work may concentrate on finding good iterations or β's beforehand by using some heuristics on the data.

References

M. Aizerman, E. Braverman, and L. Rozonoer. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.

N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines (and Other Kernel-Based Learning Methods).* Cambridge University Press, UK, 2000.

H. Drucker, C. Burges, L. Kaufman, A. Smola, and V. Vapnik. Support vector regression machines. In *Proceedings of the 1996 Conference on Advances in Neural Information Processing Systems*, volume 9, pages 155–161. The MIT Press, 1997.

A. Gammerman, Y. Kalnishkan, and V. Vovk. On-line prediction with kernels and the complexity approximation principle. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, pages 170–176. AUAI Press, 2004.

A. E. Hoerl. Application of ridge analysis to regression problems. *Chemical Engineering Progress*, 58:54–59, 1962.

[7] Here we are treating the Mexican Hat experiments corrupted with different noise as separate.

M. Hollander and D. A. Wolfe. *Nonparametric Statistical Methods.* John Wiley & Sons, USA, 1973.

D. J. Newman, S. Hettich, C. L. Blake, and C.J. Merz. UCI repository of machine learning databases, 1998. URL `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C.* Cambridge University Press, UK, second edition, 1994.

J. Quiñonero-Candela, C. E. Rasmussen, F. Sinz, O. Bousquet, and B. Schölkopf. Evaluating predictive uncertainty challenge. In *Evaluating Predictive Uncertainty, Visual Object Classification and Textual Entailment: Selected Proceedings of the 1st PASCAL Machine Learning Workshop.* Springer LNCS, to appear in 2006.

C. Saunders, A. Gammerman, and V. Vovk. Ridge regression learning algorithm in dual variables. In *Proceedings of the 15th International Conference on Machine Learning*, pages 515–521. Morgan Kaufmann, 1998.

B. Schölkopf and A. J. Smola. *Learning with Kernels – Support Vector Machines, Regularization, Optimization and Beyond.* The MIT Press, USA, 2002.

V. Vovk. Aggregating strategies. In M. Fulk and J. Case, editors, *Proceedings of the 3rd Annual Workshop on Computational Learning Theory*, pages 371–383. Morgan Kaufmann, 1990.

V. Vovk. Competitive on-line linear regression. In *Proceedings of the 1997 Conference on Advances in Neural Information Processing Systems*, volume 10, pages 364–370, Cambridge, MA, USA, 1998. MIT Press.