IMPROVING THE AGGREGATING ALGORITHM FOR REGRESSION

Steven Busuttil, Yuri Kalnishkan and Alex Gammerman
Department of Computer Science,
Royal Holloway, University of London,
Egham, Surrey, TW20 0EX, UK.
{steven,yura,alex}@cs.rhul.ac.uk

ABSTRACT
Kernel Ridge Regression (KRR) and the recently developed Kernel Aggregating Algorithm for Regression (KAAR) are regression methods based on Least Squares. KAAR has theoretical advantages over KRR since a bound on its square loss for the worst case is known that does not hold for KRR. This bound does not make any assumptions about the underlying probability distribution of the data. In practice, however, KAAR performs better only when the data is heavily corrupted by noise or has severe outliers. This is due to the fact that KAAR is similar to KRR but with some fairly strong extra regularisation. In this paper we develop KAAR in such a way as to make it practical for use on real world data. This is achieved by controlling the amount of extra regularisation. Empirical results (including results on the well known Boston Housing dataset) suggest that in general our new methods perform as well as or better than KRR, KAAR and Support Vector Machines (SVM) in terms of the square loss they suffer.

KEY WORDS
Machine Learning, Regression, Least Squares, Kernels

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]. In the simplest of models this relationship is taken to be linear 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 which finds the line (or hyperplane) that fits the data with minimum squared differences, known as square losses. This method however, can fit 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 by noise. Ridge Regression (RR) [1] 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 optimal on the training data but usually generalises better. RR works very well on real world data and is still very popular today. The Aggregating Algorithm for Regression (AAR) [2] (see also the Vovk-Azoury-Warmuth algorithm in [3]) is a relatively new method and is 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 [4] and [5] 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).

For KAAR we have a general worst case upper bound on its loss (in the online context) that does not hold for KRR [2, 5]. KRR is optimal only under some probabilistic assumptions [6]. KAAR's bound does not make any assumptions on the underlying probability distribution of the data, which makes it applicable to a much wider group of datasets. In particular, this bound does not require the data to be independently identically distributed (i.i.d.). In many practical applications i.i.d. is unrealistic to assume. Theoretically, this implies that KAAR should suffer smaller loss than KRR in general. However, from our empirical analyses we found that in most cases KAAR's regularisation is too strong and results in big losses being suffered. On the other hand, sometimes KRR's regularisation is not strong enough and this results in its predictions fluctuating a lot.

In Section 3 we introduce new methods, principally Iterative KAAR (IKAAR) and Controlled KAAR (CKAAR), which modify KAAR in such a way as to be able to control the amount of extra regularisation, the choice of which should depend on the data at hand. In Section 4 we report their empirical performance on two real world datasets. From these results we conclude that in general our new methods perform as well as or better than KRR, KAAR and Support Vector Machines (SVM) [7, 8] in terms of the square loss they suffer.

[1] In other literature, outcomes are also called labels/targets, while signals are also called examples/instances.

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

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 equation

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

where $\mathbf{w} \in \mathbb{R}^m$ and $\varepsilon_i \in \mathbb{R}$ 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 $\mathbf{w}_{\text{L}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$, where $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_\ell)'$ and $\mathbf{y} = (y_1, \ldots, y_\ell)'$.

2.1 Ridge Regression

Least Squares runs into problems when some features in $\mathbf{X}$ are highly correlated because the matrix $\mathbf{X}'\mathbf{X}$ becomes close to singular, resulting in unstable solutions. Ridge Regression (RR), first introduced to statistics by Hoerl [1], differs from Least Squares in that its objective is to minimise

$$\mathcal{L}_{\text{R}} = \alpha \|\mathbf{w}_{\text{R}}\|^2 + \sum_{i=1}^{\ell} (y_i - \langle \mathbf{w}_{\text{R}}, \mathbf{x}_i \rangle)^2, \tag{3}$$

where α is a fixed positive real number. Finding the solution now involves calculating $\mathbf{w}_{\text{R}} = (\alpha\mathbf{I} + \mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$, where $\mathbf{I}$ is the identity matrix. Apart from stabilising the solution (since $\alpha > 0$ the matrix $(\alpha\mathbf{I} + \mathbf{X}'\mathbf{X})$ is positive definite and therefore nonsingular), this technique also includes regularisation in that it favours a $\mathbf{w}_{\text{R}}$ with smaller elements. 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) [9] 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. 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 [2]. 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. Moreover, AAR's bound does not make any assumptions on the probability distribution of the data. It turns out that AAR is similar to RR but with the signal-outcome pair $(\mathbf{x}_{\ell+1}, 0)$ added to its training set, where $\mathbf{x}_{\ell+1}$ is the new signal for which a prediction is to be made. This makes predictions shrink towards 0, with the goal of making them even more resistant to overfitting (it is assumed that the mean of the outcomes is 0). AAR aims to find a solution $\mathbf{w}_{\text{A}}$ that minimises

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

The AAR solution to the regression problem is therefore $\mathbf{w}_{\text{A}} = (\alpha\mathbf{I} + \widetilde{\mathbf{X}}'\widetilde{\mathbf{X}})^{-1}\widetilde{\mathbf{X}}'\widetilde{\mathbf{y}}$, 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. The kernel trick (first used in this context in [10]) is now a widely used technique which can make a linear algorithm operate in feature space without the inherent complexities. A kernel function k takes two vectors and returns their dot product in some feature space, $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$, 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 (have non-negative eigenvalues).

Through kernel functions it is therefore possible to perform linear regression in feature space which would be equivalent to performing nonlinear regression in input space. Accordingly, RR and AAR have been reduced into a formulation known as dual variables (see [4] and [5] respectively), where 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 the Kernel Aggregating Algorithm for Regression (KAAR), respectively calculate the prediction γ for a new example $\mathbf{x}_{\ell+1}$ as follows:

$$\gamma_{\text{KRR}} = \mathbf{y}'(\alpha\mathbf{I} + \mathbf{K})^{-1}\mathbf{k}, \tag{5}$$

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,

$$\gamma_{\text{KAAR}} = \widetilde{\mathbf{y}}'(\alpha\mathbf{I} + \widetilde{\mathbf{K}})^{-1}\widetilde{\mathbf{k}}, \tag{6}$$

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 $\widetilde{\mathbf{k}} = \big(\mathbf{k}', k(\mathbf{x}_{\ell+1}, \mathbf{x}_{\ell+1})\big)'$.

3 Methods

In [2, Theorem 1] a worst case upper bound on the loss of the Aggregating Algorithm for Regression (AAR) is derived. This bound essentially compares the cumulative loss of AAR in the online mode against the loss of the best linear predictor. A similar bound is also given for the more general kernel case (KAAR) in [5]. It is shown in [2, Theorem 3] that this bound does not hold for Ridge Regression (RR), therefore AAR has a better theoretical worst case performance bound than RR. On the other hand RR can be shown to be a Bayesian method (see, for example, [6, Section 10.3]). This means that under certain probabilistic assumptions on the data, RR has optimal properties on average. Although we cannot realistically expect these assumptions to hold for real world datasets, RR is known to perform very well in practice. Moreover, through a thorough empirical analysis it became evident that many times KAAR's predictions are overly rigid while KRR's predictions sometimes fluctuate too much. It was also observed that occasionally a better prediction would be somewhere in between those of KRR and KAAR. As we saw in Section 2, KRR and KAAR are rather similar from a computational perspective. Is it possible therefore to combine these two methods to give a new method that in general is more accurate than both? In this section we present three new methods that attempt to achieve this.

3.1 Simple Convex Combination

One of the simplest ways of combining KRR and KAAR is to take a convex combination of their predictions. This new 'method', which we have dubbed KOKO makes its predictions as follows:

$$\gamma_{\text{KOKO}} = (1-\theta)\gamma_{\text{KRR}} + \theta\gamma_{\text{KAAR}},$$

where θ is a scalar from the interval $[0,1]$.

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. 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 a sequence of 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:

$$\gamma^{(n)} = \widetilde{\mathbf{y}}^{(n)\prime}(\alpha\mathbf{I} + \widetilde{\mathbf{K}})^{-1}\widetilde{\mathbf{k}}, \tag{7}$$

where $\gamma^{(0)} = 0$, $n \geq 1$, and $\widetilde{\mathbf{y}}^{(n)} = \left(\mathbf{y}', \gamma^{(n-1)}\right)'$. Note that in Section 3.4 we give an explicit formula that computes $\gamma^{(n)}$ directly for any n.

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 signal converge towards that of KRR as n approaches infinity. We can open up (7) in the following way:

$$\begin{aligned}\gamma^{(n)} &= \begin{bmatrix} \mathbf{y} \\ \gamma^{(n-1)} \end{bmatrix}' \begin{bmatrix} \mathbf{K}+\alpha\mathbf{I} & \mathbf{k} \\ \mathbf{k}' & k(\mathbf{x},\mathbf{x})+\alpha \end{bmatrix}^{-1} \\ &\quad \times \begin{bmatrix} \mathbf{k} \\ k(\mathbf{x},\mathbf{x}) \end{bmatrix}. \end{aligned} \tag{8}$$

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

$$\gamma^{(n)} = s\gamma^{(n-1)} + c, \tag{9}$$

where $s, c \in \mathbb{R}$. If we manage to show that $0 \leq |s| < 1$ then it would follow from the Banach fixed point theorem that IKAAR's predictions converge to a fixed point r, such that $r = sr + c$. Therefore, as $n \to \infty$, then $\gamma^{(n-1)} \to \gamma^{(n)}$ and $\gamma^{(n)} \to r$.

If we do the inversion in (8) by partitioning (see Lemma A.1), we get

$$\begin{aligned}\gamma^{(n)} &= \begin{bmatrix} \mathbf{y} \\ \gamma^{(n-1)} \end{bmatrix}' \begin{bmatrix} \mathbf{P} & \mathbf{Q} \\ \mathbf{R} & \mathbf{S} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{Q} \\ k(\mathbf{x},\mathbf{x}) \end{bmatrix} \\ &= \left(\widetilde{\mathbf{R}}\mathbf{Q} + \widetilde{\mathbf{S}}k(\mathbf{x},\mathbf{x})\right)\gamma^{(n-1)} \\ &\quad + \left(\mathbf{y}'\widetilde{\mathbf{P}}\mathbf{Q} + \mathbf{y}'\widetilde{\mathbf{Q}}k(\mathbf{x},\mathbf{x})\right), \end{aligned} \tag{10}$$

where $\mathbf{P} = \mathbf{K} + \alpha\mathbf{I}$, $\mathbf{Q} = \mathbf{R}' = \mathbf{k}$, and $\mathbf{S} = k(\mathbf{x},\mathbf{x}) + \alpha$ (in this case all the necessary inverses exist). Making substitutions for $\widetilde{\mathbf{P}}$, $\widetilde{\mathbf{Q}}$, $\widetilde{\mathbf{R}}$, and $\widetilde{\mathbf{S}}$ in (10) we get,

$$s = \frac{k(\mathbf{x},\mathbf{x}) - \mathbf{k}'(\mathbf{K}+\alpha\mathbf{I})^{-1}\mathbf{k}}{k(\mathbf{x},\mathbf{x}) - \mathbf{k}'(\mathbf{K}+\alpha\mathbf{I})^{-1}\mathbf{k} + \alpha}, \tag{11}$$

$$c = \mathbf{y}'(\mathbf{K}+\alpha\mathbf{I})^{-1}\mathbf{k}(1-s). \tag{12}$$

We will now proceed to show that s is always in the interval $[0,1)$. Since by definition $\alpha > 0$, we only need to show that $k(\mathbf{x},\mathbf{x}) \geq \mathbf{k}'(\mathbf{K}+\alpha\mathbf{I})^{-1}\mathbf{k}$ to reach our goal. We will first show this for the linear kernel (the dot product)

and subsequently we will generalise the result for the nonlinear kernel case. Therefore, for the linear kernel we have to show that for every $\mathbf{x}$ the following holds:

$$\mathbf{x}'\mathbf{x} \geq (\mathbf{X}\mathbf{x})'(\mathbf{X}\mathbf{X}'+\alpha\mathbf{I})^{-1}\mathbf{X}\mathbf{x} = \mathbf{x}'\mathbf{X}'\mathbf{X}(\mathbf{X}'\mathbf{X}+\alpha\mathbf{I})^{-1}\mathbf{x}. \quad (13)$$

In order to do this, we will first reduce (13) 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. Since $\mathbf{V}$ is a unitary matrix, $\mathbf{V}^{-1} = \mathbf{V}'$ and $\mathbf{V}'\mathbf{V} = \mathbf{V}\mathbf{V}' = \mathbf{I}$.

Performing the substitution $\mathbf{x} = \mathbf{V}\mathbf{z}$ in (13) is the same as considering it in the orthogonal basis formed by the eigenvectors of $\mathbf{X}'\mathbf{X}$. Therefore, showing that (13) holds is equivalent to proving that $(\mathbf{V}\mathbf{z})'\mathbf{V}\mathbf{z} \geq (\mathbf{V}\mathbf{z})'\mathbf{X}'\mathbf{X}(\mathbf{X}'\mathbf{X} + \alpha\mathbf{I})^{-1}\mathbf{V}\mathbf{z}$. This reduces 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 $\mathbf{x}'\mathbf{x} > \mathbf{x}'\mathbf{X}'\mathbf{X}(\mathbf{X}'\mathbf{X} + \alpha\mathbf{I})^{-1}\mathbf{x}$. 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 [5]), therefore $k(\mathbf{x},\mathbf{x}) \geq \mathbf{k}'(\mathbf{K} + \alpha\mathbf{I})^{-1}\mathbf{k}$.

We have just shown that $0 \leq s < 1$, therefore $\gamma^{(n)}$ converges to some point r. In the definition of c (see (12)), the term $\mathbf{y}'(\mathbf{K}+\alpha\mathbf{I})^{-1}\mathbf{k}$ is in fact KRR's prediction, therefore $c = \gamma_{\text{KRR}}(1-s)$. This means that (9) can be rewritten as

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

At fixed point r, we have $r = sr + \gamma_{\text{KRR}}(1-s)$, implying that $r = \gamma_{\text{KRR}}$, which ends our proof. □

3.3 Controlled KAAR

KAAR's predictions are so rigid because it tries to minimise the value of the predictions themselves (see the second term in (4)). 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

$$\mathcal{L}_{\text{C}} = \alpha\|\mathbf{w}_{\text{C}}\|^2+\beta\langle\mathbf{w}_{\text{C}},\mathbf{x}_{\ell+1}\rangle^2+\sum_{i=1}^{\ell}(y_i-\langle\mathbf{w}_{\text{C}},\mathbf{x}_i\rangle)^2, \quad (14)$$

where $\beta \geq 0$. 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 CKAAR will output predictions that are not as rigid as those of KAAR and do not fluctuate as much as those of KRR, whereas when $\beta > 1$, CKAAR will provide even more regularisation than KAAR does.

Letting $\mathbf{w} = \mathbf{w}_{\text{C}}$, we can express (14) in matrix notation to give $\mathcal{L}_{\text{C}} = \alpha(\mathbf{w}'\mathbf{w}) + \widetilde{\mathbf{y}}'\widetilde{\mathbf{y}} - 2\mathbf{w}'\widehat{\mathbf{X}}'\widetilde{\mathbf{y}} + \mathbf{w}'\widehat{\mathbf{X}}'\widehat{\mathbf{X}}\mathbf{w}$, where $\widehat{\mathbf{X}} = (\mathbf{X}', \sqrt{\beta}\,\mathbf{x}_{\ell+1})'$ and $\widetilde{\mathbf{y}} = (\mathbf{y}', 0)'$. If we differentiate this with respect to $\mathbf{w}$, divide throughout by 2 and set it equal to 0 we get $\frac{1}{2}\frac{\partial\mathcal{L}_{\text{C}}}{\partial\mathbf{w}} = \alpha\mathbf{w} - \widehat{\mathbf{X}}'\widetilde{\mathbf{y}} + \widehat{\mathbf{X}}'\widehat{\mathbf{X}}\mathbf{w} = 0$. This means that the CKAAR solution ($\mathbf{w}_{\text{C}}$) to the regression problem for a new example $\mathbf{x}_{\ell+1}$ is

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

The solution we have just derived is for the linear case only. To handle the nonlinear case we follow [5] to formulate our solution in dual variables so that it can be used with kernels. The kernel version of CKAAR makes a prediction for a new signal $\mathbf{x} = \mathbf{x}_{\ell+1}$ in the following way:

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

where $\widehat{\mathbf{k}} = \big(k(\mathbf{x}_1,\mathbf{x}),\ldots,k(\mathbf{x}_\ell,\mathbf{x}),\sqrt{\beta}\,k(\mathbf{x},\mathbf{x})\big)'$ and

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

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 $\sqrt{\beta}$.

3.4 Summary and comparison

Below are formulations of the predictions for $\mathbf{x}$ made by KAAR, KOKO, IKAAR and CKAAR in terms of KRR's prediction γ_{KRR}, where $z = k(\mathbf{x},\mathbf{x}) - \mathbf{k}'(\mathbf{K} + \alpha\mathbf{I})^{-1}\mathbf{k}$.

$$\begin{aligned}
\gamma_{\text{KAAR}} &= \gamma_{\text{KRR}}\left(1 - \frac{z}{z+\alpha}\right) \\
\gamma_{\text{KOKO}} &= \gamma_{\text{KRR}}\left(1 - \theta\left(\frac{z}{z+\alpha}\right)\right), 0 \leq \theta \leq 1 \\
\gamma_{\text{IKAAR}} &= \gamma_{\text{KRR}}\left(1 - \left(\frac{z}{z+\alpha}\right)^n\right), n \geq 1 \\
\gamma_{\text{CKAAR}} &= \gamma_{\text{KRR}}\left(1 - \frac{z}{z+\alpha/\beta}\right), \beta \geq 0 \\
\gamma_{\text{KRRT}} &= \gamma_{\text{KRR}}\,(1-t), 0 \leq t \leq 1
\end{aligned}$$

These formulations (mainly obtained by using Lemma A.1) give us computational advantages and they also allow us to understand our new methods better. It is immediately clear that all these methods 'scale down' (in different ways) KRR's prediction towards 0 (recall that $\alpha > 0$, that we have shown that $z \geq 0$, and that we assume that the mean of the outcomes is 0) in an effort to combat noise and outliers. Unlike KAAR, our new methods KOKO, IKAAR and CKAAR have an extra 'mixture' parameter which controls (or can completely remove) this extra regularisation. This formulation of IKAAR makes it clear that the convergence is exponential and that there is no iterative procedure to be solved, since it computes the prediction for any iteration directly. We have also included another method, dubbed KRRT to compare our methods against. KRRT simply scales down KRR's prediction using a scalar, whereas our methods take in consideration the signal for which the prediction is being made.

4 Experimental Results

Our experimentation method is similar to that of [11] in that for each dataset we run experiments on 100 different random permutations of itself. The best parameters for each method are chosen using validation and then the experiments are rerun with these parameters fixed. In this section we report the average of the mean square losses per run (MSE) of the methods mentioned in this paper and also that of a Support Vector Machine (SVM) (see [7, 8, 12]). We also show the statistical significance (according to the Wilcoxon Signed Rank Test (WSRT); see, for example, [13]) of the difference in losses of all methods and those of KRR and SVM, which is the probability that it happens by chance (known as a p-value). We do not report the p-values as compared to KAAR since these were always very small.

The results achieved on two well known real world datasets, the Boston Housing [14] and Gaze [15] datasets, are in Table 1. For clarity, p-values that are greater than or equal to 0.05 are prefixed with an asterisk ($*$) meaning that the differences are not statistically significant (by convention). We also ran experiments on the artificial Mexican Hat dataset and on other real world datasets including Abalone, Auto-MPG, Auto-Price, Relative CPU Performance, Servo and Wisconsin Prognostic Breast Cancer datasets (all from [14]). These are not included in this paper due to space limitations, however, all the results obtained are similar to the ones reported here and support our conclusions. For our experiments we used four kernels: a polynomial kernel, a spline kernel, an ANOVA spline kernel, and a Gaussian RBF kernel (see, for example, [8]).

5 Discussion and Conclusion

To determine the relative performance of methods we take in consideration their mean square losses and whether the differences are statistically significant or not. In the results reported here it is clear that CKAAR performs best, suffering smaller losses than KRR, KAAR and SVM. The performance of IKAAR is similar but not as good, while the performance of KOKO, the simple convex combination of KRR and KAAR, is slightly worse and only a little better than KRR. It is understandable that CKAAR and IKAAR perform better than KOKO, since they have a better theoretical motivation. KRRT, the 'control' method that scales down KRR's prediction by a fixed scalar does not offer any advantages and is more or less equivalent to KRR. This indicates that the scaling done by our methods, which depends on the signal for which a prediction is being made, is effectively what gives them their advantage in performance. These results (and others not reported here) suggest that our new methods IKAAR and in particular CKAAR, in general suffer a smaller or equal amount of loss than KRR, KAAR and SVM.

One disadvantage of our methods (compared to KRR but not SVM) is that a value for an extra parameter has to be chosen. In our experiments we did this by using validation, as we did for all the other parameters. Future work may concentrate on finding good values for these parameters beforehand by using some heuristics on the data. It is of interest that for both experiments reported here, the average value of CKAAR's β was roughly equal to 0.02, which means that only a little extra regularisation was added.

A Appendix

Lemma A.1 (See [16, Section 2.7])**.** *Suppose that 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 in the same manner,* $\mathbf{A}^{-1} = \begin{bmatrix} \widetilde{\mathbf{P}} & \widetilde{\mathbf{Q}} \\ \widetilde{\mathbf{R}} & \widetilde{\mathbf{S}} \end{bmatrix}$*, 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):* $\widetilde{\mathbf{P}} = \mathbf{P}^{-1} + \mathbf{P}^{-1}\mathbf{Q}(\mathbf{S} - \mathbf{R}\mathbf{P}^{-1}\mathbf{Q})^{-1}\mathbf{R}\mathbf{P}^{-1}$; $\widetilde{\mathbf{Q}} = -\mathbf{P}^{-1}\mathbf{Q}(\mathbf{S} - \mathbf{R}\mathbf{P}^{-1}\mathbf{Q})^{-1}$; $\widetilde{\mathbf{R}} = -(\mathbf{S} - \mathbf{R}\mathbf{P}^{-1}\mathbf{Q})^{-1}\mathbf{R}\mathbf{P}^{-1}$; $\widetilde{\mathbf{S}} = (\mathbf{S} - \mathbf{R}\mathbf{P}^{-1}\mathbf{Q})^{-1}$.

References

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

[2] V. Vovk. Competitive on-line statistics. *International Statistical Review*, 69(2):213–248, 2001.

[3] N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

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

[5] 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.

[6] V. Vovk, A. Gammerman, and G. Shafer. *Algorithmic learning in a random world*. Springer, USA, 2005.

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

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

	Boston Housing			Gaze		
Method	**MSE**	**Statistical Significance**		**MSE**	**Statistical Significance**	
Poly		KRR	SVM	$\times 10^3$	KRR	SVM
KRR	9.19		$* 3 \times 10^{-01}$	2.00		2×10^{-12}
SVM	9.20	$* 3 \times 10^{-01}$		2.66	2×10^{-12}	
KAAR	23.39	5×10^{-22}	1×10^{-22}	5.78	1×10^{-29}	1×10^{-20}
KOKO	8.92	2×10^{-03}	1×10^{-02}	1.98	1×10^{-02}	1×10^{-13}
IKAAR	8.59	1×10^{-03}	3×10^{-02}	2.00	3×10^{-05}	2×10^{-12}
CKAAR	**8.37**	8×10^{-04}	1×10^{-04}	**1.96**	7×10^{-04}	3×10^{-14}
KRRT	9.05	3×10^{-02}	$* 6 \times 10^{-02}$	2.01	$* 5 \times 10^{-01}$	8×10^{-13}
Spline		KRR	SVM	$\times 10^3$	KRR	SVM
KRR	7.62		2×10^{-05}	2.05		3×10^{-19}
SVM	8.63	2×10^{-05}		3.09	3×10^{-19}	
KAAR	25.24	5×10^{-25}	1×10^{-27}	12.92	2×10^{-30}	2×10^{-30}
KOKO	7.51	3×10^{-02}	3×10^{-06}	2.00	1×10^{-02}	3×10^{-21}
IKAAR	**7.10**	1×10^{-02}	5×10^{-08}	2.00	2×10^{-04}	5×10^{-20}
CKAAR	7.19	3×10^{-03}	4×10^{-08}	**1.93**	5×10^{-03}	4×10^{-22}
KRRT	7.57	$* 2 \times 10^{-01}$	5×10^{-06}	2.04	$* 4 \times 10^{-01}$	6×10^{-20}
Anova		KRR	SVM	$\times 10^3$	KRR	SVM
KRR	7.48		$* 9 \times 10^{-01}$	1.92		8×10^{-05}
SVM	7.29	$* 9 \times 10^{-01}$		2.13	8×10^{-05}	
KAAR	22.39	4×10^{-27}	3×10^{-28}	8.25	2×10^{-30}	2×10^{-30}
KOKO	7.42	$* 1 \times 10^{-01}$	$* 8 \times 10^{-01}$	1.90	6×10^{-03}	1×10^{-05}
IKAAR	7.38	9×10^{-03}	$* 9 \times 10^{-01}$	1.92	1×10^{-04}	5×10^{-05}
CKAAR	**7.28**	3×10^{-02}	$* 3 \times 10^{-01}$	**1.88**	1×10^{-03}	5×10^{-06}
KRRT	7.45	$* 4 \times 10^{-01}$	$* 9 \times 10^{-01}$	1.92	$* 3 \times 10^{-01}$	5×10^{-05}
RBF		KRR	SVM	$\times 10^3$	KRR	SVM
KRR	8.15		$* 8 \times 10^{-02}$	3.52		3×10^{-09}
SVM	8.29	$* 8 \times 10^{-02}$		2.99	3×10^{-09}	
KAAR	25.65	9×10^{-29}	4×10^{-29}	11.26	3×10^{-15}	2×10^{-30}
KOKO	8.12	$* 3 \times 10^{-01}$	2×10^{-02}	3.26	$* 6 \times 10^{-01}$	2×10^{-09}
IKAAR	8.10	$* 9 \times 10^{-02}$	4×10^{-02}	2.81	5×10^{-04}	9×10^{-12}
CKAAR	**8.00**	$* 1 \times 10^{-01}$	2×10^{-03}	**2.29**	5×10^{-02}	1×10^{-11}
KRRT	8.13	$* 5 \times 10^{-01}$	3×10^{-02}	3.46	$* 5 \times 10^{-01}$	2×10^{-09}

Table 1. Results for the Boston Housing [14] and Gaze [15] Datasets.

[9] 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.

[10] 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.

[11] 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.

[12] T. Joachims. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods: Support Vector Learning*. MIT Press, Cambridge, MA, 1999.

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

[14] D. J. Newman, S. Hettich, C. L. Blake, and C. J. Merz. UCI repository of machine learning databases, 1998.

[15] J. Quiñonero-Candela, I. Dagan, B. Magnini, and F. D'Alché-Buc, editors. *Evaluating Predictive Uncertainty, Visual Object Categorization and Textual Entailment*, volume 3944 of *Lecture Notes in Computer Science*, Heidelberg, Germany, 2006. Springer.

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