Universal Algorithms for Probability Forecasting *

Fedor Zhdanov

Computer Learning Research Centre,
Department of Computer Science,
Royal Holloway University of London,
Egham, Surrey, TW20 0EX, UK.

fedor@cs.rhul.ac.uk

Yuri Kalnishkan

Computer Learning Research Centre,
Department of Computer Science,
Royal Holloway University of London,
Egham, Surrey, TW20 0EX, UK.

yura@cs.rhul.ac.uk

Multi-class classification is one of the most important tasks in machine learning. In this paper we consider two online multi-class classification problems: classification by a linear model and by a kernelized model. The quality of predictions is measured by the Brier loss function. We obtain two computationally efficient algorithms for these problems by applying the Aggregating Algorithms to certain pools of experts and prove theoretical guarantees on the losses of these algorithms. We kernelize one of the algorithms and prove theoretical guarantees on its loss. We perform experiments and compare our algorithms with logistic regression.

Keywords: Probability prediction, online regression, Brier loss, multi-class classification

1. Introduction

Online prediction is a large area of machine learning (see [3] for an overview) with many practical applications to data mining problems. It focuses on algorithms that learn "on-the-fly". In online regression framework we receive some input and try to predict the corresponding outcome on each time step. The quality of the predictions is assessed by means of a loss function measuring their deviation from true outcomes.

In this paper we consider the multi-dimensional Brier game where outcomes and predictions come from a simplex and can be thought of as probability distributions on the vertexes of the simplex. The problem of prediction in this environment can be thought of as soft multi-class classification of a given input, where the classes correspond to the vertexes of the simplex. The loss function we consider is the

*Earlier versions of this paper appeared in Proceedings of Artificial Intelligence Applications and Innovations, AIAI 2010, vol. 339 of IFIP Advances in Information and Communication Technology, Springer, 2010 and as technical report arXiv:1001.0879 at *CoRR*.

square loss (see [2] for a discussion), i.e., the squared norm of the difference of a prediction and the true outcome.

The idea of universal algorithms originated in the Bayesian learning framework. A universal algorithm captures the power of a (parametric) family of algorithms and performs little worse than the best algorithm in the family in terms of the cumulative loss. Contrary to Bayesian or any other traditional statistical settings, we make no assumptions about the mechanism generating the data (e.g., we assume no underlying distribution). The relative bounds for the performance of a universal algorithm hold in the worst case.

We use Vovk's Aggregating Algorithm (a generalization of the Bayesian mixture) to mix families of predictors. In [15] the Aggregating Algorithm was applied to the case of Brier loss when possible outcomes are in a segment of the real line; this resulted in the Aggregating Algorithm Regression (AAR) also known as the Vovk-Azoury-Warmuth predictor (see [3]).

One algorithm we construct applies a variant of the AAR to predict each coordinate of the outcome separately and then combines these predictions to get a probability prediction. The other algorithm is designed to give probability predictions directly by merging underlying families of asymmetric linear (or kernel) predictors.

An asymmetric predictor works as follows. We single out one component of the outcome assuming it to have the meaning of the "remainder". All components except for the remainder are generated by a linear (or kernel) function, while the remainder is chosen so that the components sum up to one. This approach is motivated by a range of practical scenarios. For example, in a football match either one team wins or the other, and the remainder is a draw (see [16] for online prediction experiments in football). When we analyze a precious metal alloy we may be interested in obtaining a description of the following kind: this alloy has 40% of gold, 35% of silver, and some unspecified addition (e.g., consisting of copper and palladium), which can be thought of as the remainder. Financial applications often try to predict the direction of the price move, i.e., whether the price is going up or down; the remainder in this situation is the price staying close to the current value.

These algorithms are the first computationally efficient online regression algorithms designed to solve linear and non-linear multi-class classification problems. We derive theoretical bounds on the cumulative losses of both the algorithms. We arrive at a surprising conclusion that the component-wise algorithm is better than the second one asymptotically, but worse in the beginning of the prediction process. Their performance on benchmark data sets appears to be very similar.

An overview of the framework can be found in Section 2; a description of the algorithms in the linear case is in Section 3; the theoretical bounds are derived in Section 4.

We extend the class of experts using the kernel trick. We kernelize the second algorithm and obtain a theoretical bound on its loss. The cumulative loss of the kernelized algorithm is compared with the cumulative loss of any finite set of functions

from the Reproducing Kernel Hilbert Space (RKHS) corresponding to the kernel (see Section 5.1 for a definition). The kernelization is described in Section 5. Our experimental results are shown in Section 6. Section 7 makes the conclusions and shows some possibilities for further work.

Competing with linear experts in the case where possible outcomes come from a simplex of dimension higher than 2 was not widely considered by other researchers, so the comparison of theoretical bounds can not be performed. Kivinen and Warmuth's paper [10] includes the case when the possible outcomes lie in a high-dimensional simplex and their algorithm competes with all logistic regression functions. They use the relative entropy loss function $\mathcal{L}$ and get a regret term of the order $O(\sqrt{\mathcal{L}_T(\alpha)})$. Their prediction algorithm is not computationally efficient and it is not clear how to extend their results for the case when predictors lie in an RKHS.

2. Framework

A game of prediction is defined by three components, a space of outcomes Ω, a decision space Γ, and a loss function $\lambda : \Omega \times \Gamma \to \mathbb{R}$. We are interested in the following generalization of the Brier game (see [2]).

Take a finite set Σ with d elements; Σ can be identified with $\{1, 2, \dots, d\}$. Let the space of outcomes $\Omega = \mathcal{P}(\Sigma)$ be the set of all probability measures on Σ. It can be identified with the simplex $\{y = (y^{(1)}, y^{(2)}, \dots, y^{(d)}) \in [0,1]^d \mid \sum_{i=1}^d y^{(i)} = 1\}$ of vectors in $\mathbb{R}^d$. The vertexes of the simplex are the vectors $e_1, e_2, \dots, e_d$ forming the standard basis in $\mathbb{R}^d$; a vector e_i has one at the i-th position and zeros elsewhere. They can be identified with probability measures concentrated on single elements of Σ.

For the decision space we take the hyperplane $\Gamma := \{(\gamma^{(1)}, \dots, \gamma^{(d)}) \in \mathbb{R}^d \mid \sum_{i=1}^d \gamma^{(i)} = 1, \}$ in d-dimensional space containing all the outcomes. The loss function on $y \in \Omega$ and $\gamma \in \Gamma$ is defined by

$$\lambda(y, \gamma) = \sum_{\sigma \in \Sigma} \left(\gamma^{(\sigma)} - y^{(\sigma)}\right)^2 = \|y - \gamma\|^2.$$

For example, if $\Sigma = \{1, 2, 3\}$, $\omega = (1, 0, 0)$, $\gamma^{(1)} = 1/2$, $\gamma^{(2)} = 1/4$, and $\gamma^{(3)} = 1/4$, then $\lambda(\omega, \gamma) = (1/2-1)^2 + (1/4-0)^2 + (1/4-0)^2 = 3/8$. The Brier loss is one of the most important loss functions used to assess the quality of classification algorithms.

The game of prediction is being played repeatedly by a learner receiving some input vectors $x_t \in \mathbf{X} \subseteq \mathbb{R}^n$. This game follows Protocol 1.

We will construct an algorithm capable of competing with all asymmetric linear predictors. On step t an asymmetric linear predictor outputs $\xi_t(\alpha) = (\xi_t^{(1)}(\alpha), \dots, \xi_t^{(d)}(\alpha))'$ (in this paper M' denotes the transpose of a matrix M) de-

Protocol 1 Protocol of forecasting game

$L_0 := 0$.
for $t = 1, 2, \ldots$ **do**
  Reality announces a signal $x_t \in \mathbf{X} \subseteq \mathbb{R}^n$.
  Learner announces $\gamma_t \in \Gamma \subseteq \mathbb{R}^d$.
  Reality announces $y_t \in \Omega \subseteq \mathbb{R}^d$.
  $L_t := L_{t-1} + \lambda(y_t, \gamma_t)$.
end for

fined by

$$\begin{aligned} \xi_t^{(1)}(\alpha) &= 1/d + \alpha_1' x_t \\ &\cdots \\ \xi_t^{(d-1)}(\alpha) &= 1/d + \alpha_{d-1}' x_t \\ \xi_t^{(d)}(\alpha) &= 1 - \xi_t^{(1)} - \cdots - \xi_t^{(d-1)} = 1/d - \left(\sum_{i=1}^{d-1} \alpha_i \right)' x_t, \end{aligned} \tag{1}$$

where $\alpha_i \in \mathbb{R}^n, i = 1, \ldots, d-1$ are vectors of n parameters and $\alpha = (\alpha_1', \ldots, \alpha_{d-1}')' \in \Theta = \mathbb{R}^{n(d-1)}$. In the model defined by (1) the prediction for the last component of an outcome is calculated from the predictions for other components. Let $L_T(\alpha) = \sum_{t=1}^{T} \lambda(y_t, \xi_t(\alpha))$ be the cumulative loss of an asymmetric predictor α over T trials.

3. Derivation of the algorithms

In this section we apply the Aggregating Algorithm (AA) proposed by Vovk in [13] to mix asymmetric linear predictors. The Aggregating Algorithm is a technique for merging predictions of arbitrary experts, i.e., predictors following Protocol 1 and making their predictions available to us. An expert can be a predictor of any kind, e.g., a human expert, an algorithm, or even some uncomputable strategy. In this paper we will mostly consider parametric families of prediction algorithms.

Suppose that we have a pool of experts $\xi_t(\alpha)$ parametrized by $\alpha \in \Theta$ (e.g., the pool of asymmetric linear predictors introduced above is parametrized by $\alpha \in \mathbb{R}^{n(d-1)}$). The Aggregating Algorithm keeps weights $p_{t-1}(\alpha)$ for the experts at each time step t and updates them according to the exponential weighting scheme after the actual outcomes is announced:

$$p_t(\alpha) = \beta^{\lambda(y_t, \xi_t(\alpha))} p_{t-1}(\alpha), \quad \beta \in (0, 1). \tag{2}$$

Here $\beta = e^{-\eta}$, where $\eta \in (0, +\infty)$ is a parameter called the learning rate. This weights update rule ensures that the experts predicting badly receive lower weights. The weights are then normalized to sum up or integrate to one: $p_t^*(\alpha) = p_t(\alpha) / \int_{\alpha \in \Theta} p_t(\alpha) d\alpha$.

The prediction of the algorithm is a combination of the experts' predictions. One possible way to merge experts' predictions is simply to take the weighted average

of the experts' predictions with weights $p_t(\alpha)$ as in [9]. The Aggregating Algorithm uses more sophisticated prediction scheme and sometimes achieves better theoretical performance. On step t it first defines a *generalized prediction* as a function $g_t : \Omega \to \mathbb{R}$ such that

$$g_t(y) = \log_\beta \int_\Theta \beta^{\lambda(y,\xi_t(\alpha))} p^*_{t-1}(\alpha) d\alpha \tag{3}$$

for all $y \in \Omega$. It is a weighted average (in a general sense) of the experts' losses for each possible outcome. It then outputs any γ_t such that

$$\lambda(y, \gamma_t) \le g_t(y) \tag{4}$$

for all possible $y \in \Omega$. If such prediction can be found for any weights distribution on experts the game is called *perfectly mixable*. Perfectly mixable games and other types of games are analyzed in [14]. It is also shown there that for finite pools of experts the AA achieves the best possible theoretical guarantees.

3.1. *Proof of mixability*

In this section we show that our version of the Brier game is perfectly mixable and obtain a function mapping generalized predictions g_t to predictions γ_t satisfying (4). It is shown in Theorem 1 from [16] that the Brier game with a finite number of outcomes is perfectly mixable iff $\eta \in (0, 1]$. We will extend this result to our settings.

The outcome space considered in [16] is a subspace of the outcome space from this paper; it consists of d probability measures fully concentrated on the vertexes of the simplex, which can be identified with points from Σ. We denote the set of vertexes by $\mathcal{R}(\Sigma) = \{e_1, e_2, \dots, e_d\}$. The decision space in [16] is the set of probability measures $\mathcal{P}(\Sigma)$, i.e., points from the simplex. In this paper we had to expand this to the whole hyperplane in order to be able to consider experts defined by (1). We need to prove that inequality (4) still holds for our outcome and decision spaces. We start by showing that any vector from $\mathbb{R}^d$ can be projected into the simplex without increasing the Brier loss.

Lemma 3.1. *For any $\xi \in \mathbb{R}^d$ there exists $\theta \in \mathcal{P}(\Sigma)$ such that for any $y \in \Omega$ the inequality $\lambda(y, \theta) \le \lambda(y, \xi)$ holds.*

Proof. The Brier loss of a prediction γ is the squared Euclidean distance between γ and the actual outcome y in the d-dimensional space. The proof follows from the fact that $\mathcal{P}(\Sigma)$ is a convex and closed set in $\mathbb{R}^d$. □

Lemma 3.2. *Let $p(\alpha)$ be any probability density on Θ and ξ be any pool of experts parametrized by $\alpha \in \Theta$. Then for any $\eta \in (0, 1]$ there exists $\gamma \in \Gamma$ such that for any $y \in \mathcal{R}(\Sigma)$ we have*

$$\lambda(y, \gamma) \le \log_\beta \int_\Theta \beta^{\lambda(y,\xi(\alpha))} p(\alpha) d\alpha.$$

Proof. By Lemma 3.1 for any $\xi(\alpha)$ we can find $\theta(\alpha) \in \mathcal{P}(\Sigma)$ such that the loss of experts does not increase: $\lambda(y, \theta(\alpha)) \leq \lambda(y, \xi(\alpha))$ for any $y \in \mathcal{R}(\Sigma)$. Thus we have

$$\log_\beta \int_\Theta \beta^{\lambda(y,\theta(\alpha))} p(\alpha) d\alpha \leq \log_\beta \int_\Theta \beta^{\lambda(y,\xi(\alpha))} p(\alpha) d\alpha$$

for any $y \in \mathcal{R}(\Sigma)$. By Theorem 1 from [16] the inequality

$$\lambda(y, \gamma) \leq \log_\beta \int_\Theta \beta^{\lambda(y,\theta(\alpha))} p(\alpha) d\alpha.$$

can be satisfied for any $\eta \in (0, 1]$ ($\beta \in [e^{-1}, 1)$). □

A function converting generalized predictions into predictions satisfying (4) is called a substitution function. We shall prove that we can use the same substitution function and the same learning rate parameter η as in [16]. The following lemma extends Lemma 4.1 from [6].

Lemma 3.3. *Let $p(\alpha)$ be a probability density on Θ and ξ be any pool of experts parametrized by $\alpha \in \Theta$. Put*

$$f(y) = \log_\beta \int_\Theta \beta^{\lambda(y,\xi(\alpha))} p(\alpha) d\alpha$$

for every $y \in \mathcal{P}(\Sigma)$. If $\gamma \in \mathcal{P}(\Sigma)$ satisfies $\lambda(z, \gamma) \leq f(z)$ for all $z \in \mathcal{R}(\Sigma)$ then $\lambda(y, \gamma) \leq f(y)$ for all $y \in \mathcal{P}(\Sigma)$.

Proof. For brevity we shall write ξ instead of $\xi(\alpha)$. It is easy to check that $\lambda(y, \gamma) - \lambda(y, \xi) = \sum_{\sigma \in \Sigma} y^{(\sigma)} [\lambda(e_\sigma, \gamma) - \lambda(e_\sigma, \xi)]$, where e_σ is the vector from the standard basis corresponding to $\sigma \in \Sigma$. Note that $\lambda(y, \gamma) - f(y) \leq 0$ is equivalent to $\int_\Theta \beta^{\lambda(y,\xi) - \lambda(y,\gamma)} p(\alpha) d\alpha \leq 1$. Thus due to the convexity of the exponent $\int_\Gamma \beta^{\sum_{\sigma \in \Sigma} y^{(\sigma)} [\lambda(e_\sigma, \xi) - \lambda(e_\sigma, \gamma)]} p(\alpha) d\alpha \leq \sum_{\sigma \in \Sigma} y^{(\sigma)} = 1$. □

We shall use the substitution function defined by the following proposition.

Proposition 3.1. *Let g be a superprediction. For every $\sigma \in \Sigma$ let e_σ be the corresponding vertex of the simplex, i.e., $e_\sigma^{(\rho)} = 0$ if $\sigma \neq \rho$ and $e_\sigma^{(\sigma)} = 1$ for all $\sigma \in \Sigma$. Let $r_\sigma = g(e_\sigma)$. Then there exists $s \in \mathbb{R}$ satisfying the condition*

$$\sum_{i=1}^{d} (s - r_i)^+ = 2,$$

where $x^+ = \max(x, 0)$. If $\gamma = (\gamma^{(1)}, \ldots, \gamma^{(d)})$ is such that

$$\gamma^{(i)} = \frac{(s - r_i)^+}{2}, \quad i = 1, \ldots, d$$

then (4) holds.

The proposition follows from Proposition 1 from [16] and Lemma 3.3 above.

Note that this substitution function allows us to avoid weights normalization in calculating the generalized prediction at each step (i.e., to drop * in the weights distribution) and speed up the computation. Suppose that instead of g_t we can get only $r = g_t(y) + C$, where C is the same for all y. Then predictions γ_t defined by the substitution function from Proposition 3.1 will be the same.

3.2. *Algorithm for multidimensional outcomes*

For the prior weights distribution p_0 over the set $\Theta = \mathbb{R}^{n(d-1)}$ of experts α we take the Gaussian density with a parameter $a > 0$:

$$(a\eta/\pi)^{n(d-1)/2} e^{-a\eta\|\alpha\|^2}.$$

Instead of evaluating the integral in (3) we shall get a shifted generalized prediction r by calculating $r_i = g_T(e_i) - g_T(e_d)$ (we drop the index T in r for brevity). Each component of $r = (r_1, \ldots, r_d)$ corresponds to one of the possible outcomes and $r_d = 0$. Other components, with $i = 1, \ldots, d-1$, are

$$r_i = \log_\beta \frac{\beta^{g_T(e_i) + \sum_{t=1}^{T-1} g_t(y_t)}}{\beta^{g_T(e_d) + \sum_{t=1}^{T-1} g_t(y_t)}} = \log_\beta \frac{\int_\Theta e^{-\eta Q(\alpha, e_i)} d\alpha}{\int_\Theta e^{-\eta Q(\alpha, e_d)} d\alpha}$$

where $Q(\alpha, y)$ denotes the quadratic form:

$$Q(\alpha, y) = \sum_{t=1}^{T} \sum_{i=1}^{d} ((y_t^{(i)} - \xi^{(i)}(x_t))^2.$$

Here $y_t = (y_t^{(1)}, \ldots, y_t^{(d)})$ are the outcomes on the steps before T and $y_T = (y_T^{(1)}, \ldots, y_T^{(d)})$ is a *possible* outcome on the step T.

Let $C = \sum_{t=1}^{T} x_t x_t'$; it is a matrix of size $n \times n$. The quadratic form Q can be partitioned into the quadratic part, the linear part, and the remainder: $Q = Q_1 + Q_2 + Q_3$. Here

$$Q_1(\alpha, y) = \alpha' A \alpha$$

is the quadratic part of $Q(\alpha, y)$, where A is a square matrix with $n(d-1)$ rows (see the exact expression for A in the algorithm below). The linear part is equal to

$$Q_2(\alpha, y) = h'\alpha - 2 \sum_{i=1}^{d-1} (y_T^{(i)} - y_T^{(d)}) \alpha_i' x_T,$$

where $h_i = -2 \sum_{t=1}^{T-1} (y_t^{(i)} - y_t^{(d)}) x_t, i = 1, \ldots, d-1$ make up a big vector $h = (h_1', \ldots, h_{d-1}')'$. The remainder is equal to

$$Q_3(\alpha, y) = \sum_{t=1}^{T-1} \sum_{i=1}^{d} (y_t^{(i)} - 1/d)^2 + \sum_{i=1}^{d} (y_T^{(i)} - 1/d)^2.$$

The ratio for r_i can be calculated using the following lemmas. The integral evaluates as follows:

Lemma 3.4. *Let $Q(\alpha) = \alpha' A \alpha + b' \alpha + c$, where $\alpha, b \in \mathbb{R}^n$, c is a scalar and A is a symmetric positive definite $n \times n$ matrix. Then*

$$\int_{\mathbb{R}^n} e^{-Q(\alpha)} d\alpha = e^{-Q_0} \frac{\pi^{n/2}}{\sqrt{\det A}},$$

where $Q_0 = \min_{\alpha \in \mathbb{R}^n} Q(\alpha)$.

This lemma is proven as Theorem 15.12.1 in [5]. Following this lemma, we can rewrite r_i as $r_i = F(A, b_i, z_i)$, $i = 1, \ldots, d-1$, where

$$F(A, b_i, z_i) = \min_{\alpha \in \Theta} Q(\alpha, e_i) - \min_{\alpha \in \Theta} Q(\alpha, e_d).$$

The variables b_i, z_i and the precise formula for F are given by the following lemma.

Lemma 3.5. *Let*

$$F(A, b, z) = \min_{\alpha \in \mathbb{R}^n} (\alpha' A \alpha + b' \alpha + z' \alpha) - \min_{\alpha \in \mathbb{R}^n} (\alpha' A \alpha + b' \alpha - z' \alpha),$$

where $b, z \in \mathbb{R}^n$ and A is a symmetric positive definite $n \times n$ matrix. Then $F(A, b, z) = -b' A^{-1} z$.

Proof. This lemma is proven by taking the derivative of the quadratic forms w.r.t. α and calculating the minimum: $\min_{\alpha \in \mathbb{R}^n} (\alpha' A \alpha + c' \alpha) = -\frac{(A^{-1}c)'}{4} c$ for any $c \in \mathbb{R}^n$ (see Theorem 19.1.1 in [5]). □

We can see that $b_i = h + (x_T', \ldots, x_T', \mathbf{0}, x_T', \ldots, x_T')' \in \mathbb{R}^{n(d-1)}$, where $\mathbf{0}$ is a zero-vector from $\mathbb{R}^n$. We also have $z_i = (-x_T', \ldots, -x_T', -2x_T', -x_T', \ldots, -x_T')'$. Thus we can calculate $d-1$ differences r_i, assign $r_d = 0$, and then apply the substitution function from Proposition 3.1 to get predictions. The resulting algorithm is Algorithm 1. We will refer to it as mAAR (multi-dimensional Aggregating Algorithm for Regression).

3.3. *Component-wise algorithm*

In this section we derive the component-wise algorithm. It gives predictions for each component of the outcome separately and then combines them in a special way.

First we explain why we cannot directly use the algorithm from [15] in the form of $d-1$ component-wise copies. In [15] one-dimensional experts are "centered" around the center $1/2$ of the prediction interval $[0, 1]$, i.e., $1/2$ is the mode and the mean of the prediction given the prior density on experts. The mean d-dimensional prediction is then $(1/2, \ldots, 1/2, 1 - (d-1)/2))$. If the outcome $y = (1, 0, \ldots, 0)$ occurs, then the mean square loss is $(d-1)/2^2 + (d-1)^2/2^2$.

Note that the mean prediction of experts (1) is $1/d$ in each component. If the outcome $y = (1, 0, \ldots, 0)$ occurs, then the mean square loss is $(d-1)/d^2 + (1-1/d)^2$, which is an improvement for $d > 2$.

Algorithm 1 mAAR for the Brier game

Fix n, $a > 0$. $C = 0, h = 0$.
for $t = 1, 2, \ldots$ **do**
Read new $x_t \in \mathbf{X}$.
$C = C + x_t x_t'$, $A = aI + \begin{pmatrix} 2C & \cdots & C \\ \vdots & \ddots & \vdots \\ C & \cdots & 2C \end{pmatrix}$
Set $b_i = h + (x_t', \ldots, x_t', 0, x_t', \ldots, x_t')'$, where the zero-vector from $\mathbb{R}^n$ is placed at the i-th position, $i = 1, \ldots, d-1$.
Set $z_i = (-x_t', \ldots, -x_t', -2x_t', -x_t', \ldots, -x_t')'$, where $-2x_t'$ is placed at i-th position, $i = 1, \ldots, d-1$.
Calculate $r_i := -b_i' A^{-1} z_i, r_d := 0$, $i = 1, \ldots, d-1$.
Solve $\sum_{i=1}^{d} (s - r_i)^+ = 2$ w.r.t. $s \in \mathbb{R}$.
Set $\gamma_t^{(i)} := (s - r_i)^+ / 2$, $\omega \in \Omega$, $i = 1, \ldots, d$.
Output prediction $\gamma_t \in \mathcal{P}(\Omega)$.
Read outcome y_t.
$h_i = h_i - 2(y_t^{(i)} - y_t^{(d)}) x_t, h = (h_1', \ldots, h_{d-1}')'$.
end for

The experts from [15] are more suitable for the case when each input vector x can belong to many classes simultaneously; they do not allow us to take advantage of the fact that the classification is actually unique (albeit soft).

Now let us consider component-wise experts given by

$$\xi_t^{(i)} = 1/d + \alpha_i' x_t, \quad i = 1, \ldots, d. \tag{5}$$

The derivation of the component-wise algorithm (we call it component-wise Aggregating Algorithm Regression and abbreviate to cAAR) is similar to the derivation of Algorithm 1 for two outcomes. The initial distribution on each component of experts (5) is given by

$$(a\tilde{\eta}/\pi)^{n/2} e^{-a\tilde{\eta}\|\alpha_i\|^2}.$$

Note that the value for $\tilde{\eta}$ here will be different from 1 since the loss function in each component is half of the Brier loss $\lambda(y, \gamma) = (y - \gamma)^2 + (1 - y - (1 - \gamma))^2$. We will further see that $\tilde{\eta} = 2$. The loss of expert $\xi(\alpha_i)$ over the first T trials is

$$\sum_{t=1}^{T} (y_t^{(i)} - 1/d - \alpha_i' x_t)^2 = \alpha_i' \left(\sum_{t=1}^{T} x_t x_t' \right) \alpha_i - 2\alpha_i' \left(\sum_{t=1}^{T} (y_t^{(i)} - 1/d) x_t \right) + \sum_{t=1}^{T} (y_t^{(i)} - 1/d)^2.$$

Instead of the substitution function from Proposition 3.1 we use the substitution function suggested in [15] for the one-dimensional game:

$$\gamma_T^{(i)} = \frac{1}{2} + \frac{g_T(0) - g_T(1)}{2}$$

Therefore, the substitution function can be represented as

$$\begin{aligned}\gamma_T^{(i)} &= \frac{1}{2} + \frac{1}{2}\log_{\tilde{\beta}} \frac{\tilde{\beta}^{g_T(0)}}{\tilde{\beta}^{g_T(1)}} \\ &= \frac{1}{2} + \frac{1}{2}\log_{\tilde{\beta}} \frac{\int_{\mathbb{R}^n} e^{-\tilde{\eta}\alpha_i' B\alpha_i + 2\tilde{\eta}\alpha_i'(E+(0-1/d)x_T) - \tilde{\eta}\left(W+1/d^2\right)} d\alpha_i}{\int_{\mathbb{R}^n} e^{-\tilde{\eta}\alpha_i' B\alpha_i + 2\tilde{\eta}\alpha_i'(E+(1-1/d)x_T) - \tilde{\eta}(W+(1-1/d)^2)} d\alpha_i} \\ &= \frac{1}{d} + \frac{1}{2} F\left(B, -2E - \frac{d-2}{d}x_T, x_T\right) \\ &= \frac{1}{d} + \left(\sum_{t=1}^{T-1}(y_t^{(i)} - 1/d)x_t' + \frac{d-2}{2d}x_T'\right)\left(aI + \sum_{t=1}^{T} x_t x_t'\right)^{-1} x_T \end{aligned} \tag{6}$$

for $i = 1, \ldots, d$. Here $B = aI + \sum_{t=1}^{T} x_t x_t'$, $E = \sum_{t=1}^{T-1}(y_t^{(i)} - 1/d)x_t$, $W = \sum_{t=1}^{T-1}(y_t^{(i)} - 1/d)^2$, $\tilde{\beta} = e^{-\tilde{\eta}}$. The transitions are justified using Lemma 3.4 and Lemma 3.5.

Then this method projects its prediction onto the prediction simplex so that the loss does not increase. We use Algorithm 2 suggested in [11].

Algorithm 2 Projection of a point from $\mathbb{R}^n$ onto probability simplex.

Initialize $I = \emptyset$, $x = \mathbf{1} \in \mathbb{R}^d$.
Let γ_T be the prediction vector and $|I|$ is the dimension of the set I.
while 1 **do**
  $\gamma_T = \gamma_T - \frac{\sum_{i=1}^{d}\gamma_T^{(i)} - 1}{d - |I|}$;
  $\gamma_T^{(i)} = 0, \forall i \in I$;
  If $\gamma_T^{(i)} \geq 0$ for all $i = 1, \ldots, d$ then break;
  $I = I \bigcup \{i : \gamma_T^{(i)} < 0\}$;
  If $\gamma_T^{(i)} < 0$ for some i then $\gamma_T^{(i)} = 0$;
end while

4. Theoretical bound

We derive theoretical bounds for the losses of Algorithm 1 and of the naive component-wise algorithm predicting in the same framework.

4.1. *Component-wise algorithm*

We prove here a theoretical bound for the loss of cAAR. The following lemma is our main tool. It is easy to prove the following statement (Lemma 1 from [15]):

Lemma 4.1. *If the learner follows the Aggregating Algorithm in a perfectly mixable game, then for every positive integer T, every sequence of outcomes of length T, and*

any initial weights density $p_0(\alpha)$ on experts it suffers loss satisfying

$$L_T(\mathrm{AA}(\eta, P_0)) \le \log_\beta \int_\Theta \beta^{L_T(\alpha)} p_0(\alpha) d\alpha. \tag{7}$$

Proof. We proceed by induction in T: for $T = 0$ the inequality is obvious, and for $T > 0$ we have:

$$\begin{aligned} L_T(\mathrm{AA}(\eta, P_0)) &\le L_{T-1}(\mathrm{AA}(\eta, P_0)) + g_T(\omega_T) \\ &\le \log_\beta \int_\Theta \beta^{L^\theta_{T-1}} p_0(\theta) d\theta + \log_\beta \int_\Theta \beta^{\lambda(\omega_T, \xi^\theta_t)} \frac{\beta^{L^\theta_{T-1}}}{\int_\Theta \beta^{L^\theta_{T-1}} p_0(\theta) d\theta} p_0(\theta) d\theta \\ &= \log_\beta \int_\Theta \beta^{L^\theta_T} p_0(\theta) d\theta \ . \end{aligned}$$

We used the inductive assumption, the definition of g_T given by (3), and (2). □

The following theorem provides a bound on loss of one component of the component-wise algorithm.

Theorem 4.1. Let the outcome space in the prediction game be the interval $[A, B], A, B \in \mathbb{R}$. Assume experts' predictions at each step are $\xi_t = C + \alpha' x_t$, where $\alpha \in \mathbb{R}^n$, $C \in \mathbb{R}$ is the same for all the experts α, and $\|x_t\|_\infty \le X$ for all t. There exists a prediction algorithm producing $\gamma_i \in \mathbb{R}, i = 1, \dots, d$ such that for any $a > 0$, every positive integer T, every sequence of input vectors and outcomes of the length T and any $\alpha \in \mathbb{R}^n$ we have

$$\sum_{t=1}^T (\gamma_t - y_t)^2 \le \sum_{t=1}^T (\xi_t - y_t)^2 + a\|\alpha\|_2^2 + \frac{n(B-A)^2}{4} \ln\left(\frac{TX^2}{a} + 1\right). \tag{8}$$

Proof. We need to prove that the game is perfectly mixable (see (4)) and find the optimal parameter η for the algorithm. Considerations similar to the ones in the proof of Lemma 2 from [15] lead to the inequality $\eta \le \frac{2}{(B-A)^2}$. Clearly, Lemma 4.1 holds for our case, so we only need to calculate the difference between the right-hand side of (7)

$$\begin{aligned} \log_\beta \int_{\mathbb{R}^n} d\alpha (a\eta/\pi)^{n/2} \exp\Bigg[&-\eta\alpha' \left(aI + \sum_{t=1}^T x_t x_t'\right) \alpha \\ &+ \eta\, 2\alpha' \left(\sum_{t=1}^T (y_t - C) x_t\right) - \eta \sum_{t=1}^T (y_t - C)^2 \Bigg]. \end{aligned}$$

and the loss of the best expert $\alpha_0' \left(aI + \sum_{t=1}^T x_t x_t'\right) \alpha_0 - 2\alpha_0' \left(\sum_{t=1}^T (y_t - C) x_t\right) + \sum_{t=1}^T (y_t - C)^2$. Here α_0 is the point where the minimum of the quadratic form is attained. Due to Lemma 3.4 this difference equals

$$\frac{1}{2\eta} \ln\det\left(I + \frac{1}{a}\sum_{t=1}^T x_t x_t'\right) \le \frac{n(B-A)^2}{4} \ln\left(\frac{TX^2}{a} + 1\right).$$

We bound the determinant of a symmetric positive definite matrix by the product of its diagonal elements (see [1], Chapter 2, Theorem 7) and use $\eta = \frac{2}{(B-A)^2}$. □

Interestingly, the theoretical bound for the regression algorithm depends only on the size of the prediction interval but not on the location of it. It also does not depend on the concentration point of experts. We use the component-wise algorithm to predict each component separately.

Theorem 4.2. If $\|x_t\|_\infty \le X$ for all t, then for any $a > 0$, every positive integer T, every sequence of outcomes of the length T, and any $\alpha \in \mathbb{R}^{n(d-1)}$ the loss L_T of the component-wise algorithm satisfies

$$L_T \le L_T(\alpha) + da\|\alpha\|_2^2 + \frac{nd}{4} \ln\left(\frac{TX^2}{a} + 1\right). \tag{9}$$

Proof. We extend the class of experts in (1) in (5). The algorithm predicts each component of the outcome separately. Summing theoretical bounds (8) for d components of the outcome, taking $\alpha_d = -\sum_{i=1}^{d-1} \alpha'_i$, and using the Cauchy inequality $\|\sum_{i=1}^{d-1} \alpha_i\|_2^2 \le (d-1)\sum_{i=1}^{d-1} \|\alpha_i\|_2^2$ we get the bound. To give probability forecasts we can project prediction points on the prediction simplex using Algorithm 2. The bound will then hold by Lemma 3.1. □

4.2. *Linear forecasting*

The theoretical bound on the loss of Algorithm 1 is given by the following theorem.

Theorem 4.3. If $\|x_t\|_\infty \le X$ for all t, then for any $a > 0$, every positive integer T, every sequence of outcomes of the length T, and any $\alpha \in \mathbb{R}^{n(d-1)}$ the algorithm mAAR($2a$) satisfies

$$L_T(\text{mAAR}(2a)) \le L_T(\alpha) + 2a\|\alpha\|_2^2 + \frac{n(d-1)}{2} \ln\left(\frac{TX^2}{a} + 1\right). \tag{10}$$

Proof. We apply mAAR with the parameter $b = 2a$. Recall that $C = \sum_{t=1}^{T} x_t x'_t$. Following the argument from the proof of Theorem 4.1 with $\eta = 1$ we obtain the bound. □

We can derive a slightly better theoretical bound: in the determinant of A one should subtract the second block row from the first one and then add the first block column to the second one, then repeat this $d-2$ times.

Proposition 4.1. *Under the conditions of Theorem 4.3 mAAR(a) satisfies*

$$L_T(\text{mAAR}(a)) \le L_T(\alpha) + a\|\alpha\|_2^2 + \frac{n(d-2)}{2} \ln\left(\frac{TX^2}{a} + 1\right) + \frac{n}{2} \ln\left(\frac{TX^2 d}{a} + 1\right). \tag{11}$$

Theoretical bound (10) is worse asymptotically by d than bound (9) of the component-wise algorithm, but it is better in the beginning, especially when the norm of the best expert $\|\alpha\|$ is large. This can happen when the dimension of the input vector is larger than the size of the prediction set: $n \gg T$.

5. Kernelization

In some cases the linear model is not rich enough to describe data well and a more complicated model is needed. We use the so called kernel trick. It is a popular technique in machine learning and it was applied to AAR in [4]. We derive an algorithm competing with all sets of functions with $d-1$ elements from an RKHS.

5.1. *Derivation of the algorithm*

Definition 5.1. A *kernel* on a domain $\mathbf{X}$ is a function $K : \mathbf{X}\times\mathbf{X} \to \mathbb{R}$ satisfying the following two properties. It is symmetric, i.e., for all $x_1, x_2 \in X$ we have $K(x_1, x_2) = K(x_2, x_1)$ and positive semi-definite, i.e, for every positive integer n, all $x_1, \dots, x_n \in \mathbf{X}$ and all $\xi_1, \dots, \xi_n \in \mathbb{R}$ the inequality $\sum_{i,j=1}^{n} K(x_i, x_j)\xi_i\xi_j \geq 0$ holds.

A Hilbert space $\mathcal{F}$ of functions from $\mathbf{X}$ to $\mathbb{R}$ is a Reproducing Kernel Hilbert Space (RKHS) corresponding to a kernel K if for all $x \in \mathbf{X}$ the function $K(x, \cdot)$ (i.e., K considered as the function of the second argument) belongs to $\mathcal{F}$ and for all functions $f \in \mathcal{F}$ the reproducing property $f(x) = \langle f, K(x, \cdot)\rangle_{\mathcal{F}}$ holds.

Classical examples of kernels are Gaussian (RBF) with $K(x_i, x_j) = e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}}$ and Vapnik's polynomial $K(x_i, x_j) = (x_i \cdot x_j + 1)^d$. It is possible to show that for every kernel there is a unique RKHS. Linear combinations $\sum_{i=1}^{n} a_i K(x_i, \cdot)$, where $x_i \in \mathbf{X}$, all belong to the RKHS and are dense in it.

Our algorithm will compete with the following experts:

$$\begin{aligned} \xi_t^{(1)} &= 1/d + f_1(x_t) \\ &\dots \\ \xi_t^{(d-1)} &= 1/d + f_{d-1}(x_t) \\ \xi_t^{(d)} &= 1 - \xi^{(1)} - \dots - \xi^{(d-1)}, \end{aligned} \tag{12}$$

where $f_1, \dots, f_{d-1} \in \mathcal{F}$ are any functions from some RKHS $\mathcal{F}$. We will denote an expert defined by $f_1, \dots, f_{d-1} \in \mathcal{F}$ as in (12) by f and use notation $L_T(f)$ for its loss.

We start by rewriting mAAR in the dual form. Let

$$\begin{aligned} \widetilde{Y}_i &= -2(y_1^{(i)} - y_1^{(d)}, \dots, y_{T-1}^{(i)} - y_{T-1}^{(d)}, -1/2), \\ \overline{Y}_i &= -2(y_1^{(i)} - y_1^{(d)}, \dots, y_{T-1}^{(i)} - y_{T-1}^{(d)}, 0), \\ \widetilde{k}(x_T) &= (x_1' x_T, \dots, x_T' x_T)', \\ \widetilde{K} &= (x_s' x_t)_{s,t=1}^{T} \end{aligned}$$

for $i = 1, \ldots, d-1$ (note that $x'_s x_t$ is the scalar product of x_s and x_t). We show that the predictions of mAAR can be represented in terms of variables defined above. We will need the following matrix property.

Proposition 5.1. *Let B, C be matrices such that the number of rows in B equals to the number of columns in C, and I be identity matrices. If $aI + CB$ and $aI + BC$ are nonsingular then*

$$B(aI + CB)^{-1} = (aI + BC)^{-1} B. \tag{13}$$

Proof. This is equivalent to $(aI + BC)B = B(aI + CB)$, which is true because of distributivity of matrix multiplication. □

Consider a $T(d-1) \times T(d-1)$ matrix $A = \left(aI + \begin{pmatrix} 2\widetilde{K} & \cdots & \widetilde{K} \\ \vdots & \ddots & \vdots \\ \widetilde{K} & \cdots & 2\widetilde{K} \end{pmatrix} \right)$.

Lemma 5.1. *On trial T the values r_i for $i = 1, \ldots, d-1$ in mAAR can be represented as*

$$r_i = \left(\widetilde{Y}_1 \cdots \overline{Y}_i \cdots \widetilde{Y}_{d-1} \right) A^{-1} \left(\widetilde{k}(x_T)' \cdots 2\widetilde{k}(x_T)' \cdots \widetilde{k}(x_T)' \right)'. \tag{14}$$

Proof. By $M = (x_1, \ldots, x_T)$ denote a matrix $n \times T$ of column input vectors. Consider the following matrices of $(d-1)^2$ blocks:

$$B = \begin{pmatrix} 2M & \cdots & M \\ \vdots & \ddots & \vdots \\ M & \cdots & 2M \end{pmatrix} \text{ and } C = \begin{pmatrix} M' & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & M' \end{pmatrix}.$$

Then h_i from the algorithm mAAR equals $h_i = M\overline{Y}'_i \in \mathbb{R}^n$. Decompose $b'_i = \left(\widetilde{Y}_1 \cdots \overline{Y}_i \cdots \widetilde{Y}_{d-1} \right) C$, where only the i-th block column uses $\overline{Y}_i$. The matrix A is equal to $A = aI + BC$. Using Proposition 5.1 we get

$$r_i = -b'_i A^{-1} z_i = -\left(\widetilde{Y}_1 \cdots \overline{Y}_i \cdots \widetilde{Y}_{d-1} \right) \cdot (aI + CB)^{-1} C \left(-x'_T \cdots -2x'_T \cdots - x'_T \right)'.$$

Note that $\widetilde{K} = M'M$ and $\widetilde{k}(x_T) = M'x_T$; thus (14) holds. □

If we replace dot products $x'_i x_j$ in $\widetilde{K}$ and $\widetilde{k}(x_T)$ by an arbitrary kernel $K(x_i, x_j)$ we get a kernelized algorithm (to get predictions one can use the same substitution function from Proposition 3.1 as before). We call this algorithm mKAAR (K for Kernelized).

5.2. *Theoretical bound for the kernelized algorithm*

To derive a theoretical bound for the loss of mKAAR we will use the following matrix determinant identity lemma. This statement is often called the Sylvester identity.

Lemma 5.2. *Let B, C be as in Proposition 5.1, and a be a real number. Then* $\det(aI + BC) = \det(aI + CB)$.

Proof. The proof is by considering a block matrix identity:

$$\begin{pmatrix} I & B \\ 0 & I \end{pmatrix} \begin{pmatrix} I + BC & 0 \\ -C & I \end{pmatrix} = \begin{pmatrix} I & B \\ -C & I \end{pmatrix} = \begin{pmatrix} I & 0 \\ -C & I + CB \end{pmatrix} \begin{pmatrix} I & B \\ 0 & I \end{pmatrix} \quad .$$

We just need to take the determinants of the right- and left-hand sides. □

The main theorem follows from the property of RKHS called Representer theorem (see Theorem 4 in [12]).

Theorem 5.1. Let $g : [0, \infty) \to \mathbb{R}$ be a strictly increasing function. Assume $\mathbf{X}$ is an arbitrary set, and $\mathcal{F}$ is a Reproducing Kernel Hilbert Space of functions on $\mathbf{X}$ corresponding to a kernel $K : \mathbf{X}^2 \to \mathbb{R}$. Assume that we also have a positive integer T and an arbitrary loss function $c : (\mathbf{X} \times \mathbb{R}^2)^T \to \mathbb{R} \bigcup \{\infty\}$. Then each $f \in \mathcal{F}$ minimizing

$$c\left((x_1, y_1, f(x_1)), \ldots, (x_T, y_T, f(x_T))\right) + g(\|f\|_{\mathcal{F}})$$

admits a representation of the form $f(x) = \sum_{i=1}^{T} \alpha_i K(x_i, x)$ for some $\alpha_i \in \mathbb{R}, i = 1, \ldots, T$.

The theoretical bound for the loss of mKAAR is proven in the following theorem.

Theorem 5.2. Let $\mathbf{X}$ be an arbitrary set of inputs and $\mathcal{F}$ be a Reproducing Kernel Hilbert Space of functions on $\mathbf{X}$ with the kernel $K : \mathbf{X}^2 \to \mathbb{R}$. Then for any $a > 0$, any $f_1, \ldots, f_{d-1} \in \mathcal{F}$, any positive integer T, and any sequence of inputs and outputs $(x_1, y_1), \ldots, (x_T, y_T)$

$$L_T(\text{mKAAR}(a)) \le L_T(f) + a \sum_{i=1}^{d-1} \|f_i\|_{\mathcal{F}}^2 + \frac{1}{2} \ln \det \left(I + \frac{1}{a} \begin{pmatrix} 2\widetilde{K} & \cdots & \widetilde{K} \\ \vdots & \ddots & \vdots \\ \widetilde{K} & \cdots & 2\widetilde{K} \end{pmatrix} \right) \quad (15)$$

Here the matrix $\widetilde{K}$ is the matrix of kernel values $K(x_i, x_j)$, $i, j = 1, \ldots, T$.

Proof. The bound follows from the upper bounds for mAAR and the Representer theorem. Let us first consider the case with the scalar product kernel $K(x_1, x_2) = x_1' x_2$. Denote $C = \sum_{t=1}^{T} x_t x_t'$. By Lemma 5.2 and calculations similar to those in the proof of Lemma 5.1 we have the equality of determinants.

Let us show we can use any other kernel instead of the scalar product and get the term with the determinant. The Representer theorem assures that the minimum of the expression $L_T(f) + a\sum_{i=1}^{d-1} \|f_i\|^2_{\mathcal{F}}$ over f-s is reached on a linear combinations of the form $f_i(x) = \sum_{j=1}^{T} \alpha_{i,j} K(x_j, x)$; the functions of this type form a finite-dimensional space so we can use the above argument for linear kernels. □

Corollary 5.1. *Under the assumptions of Theorem 5.2 the following bound holds:*

$$L_T(\mathrm{mKAAR}(a)) \le L_T(f) + a\sum_{i=1}^{d-1} \|f_i\|^2_{\mathcal{F}} + \frac{d-1}{2} \ln\det\left(I + \frac{2}{a}\widetilde{K}\right). \tag{16}$$

Proof. The bound follows from Theorem 7 in Chapter 2 of [1]. □

If we know an upper bound on the kernel values and the time horizon in advance, we can further optimize bound (15).

Corollary 5.2. *Under the assumptions of Theorem 5.2 if $c_{\mathcal{F}} > 0$ is such that $K(x,x) \le c_{\mathcal{F}}$ and if we know the number of steps T in advance and are given $F > 0$, we can choose $a > 0$ such that the bound*

$$L_T(\mathrm{mKAAR}(a)) \le L_T(f) + 2c_{\mathcal{F}} F\sqrt{(d-1)T}, \tag{17}$$

holds for any $f_1, \ldots, f_{d-1} \in \mathcal{F} : \sum_{i=1}^{d-1} \|f_i\|^2_{\mathcal{F}} \le F^2$.

Proof. Bounding the logarithm of the determinant as in Theorem 7 in Chapter 2 of [1] and using the inequality $\ln(1+x) \le x$ we get

$$\begin{aligned}\frac{d-1}{2} \ln\det\left(I + \frac{2}{a}\widetilde{K}\right) &\le \frac{(d-1)T}{2} \ln\left(1 + \frac{2c^2_{\mathcal{F}}}{a}\right) \\ &\le \frac{(d-1)Tc^2_{\mathcal{F}}}{a}.\end{aligned}$$

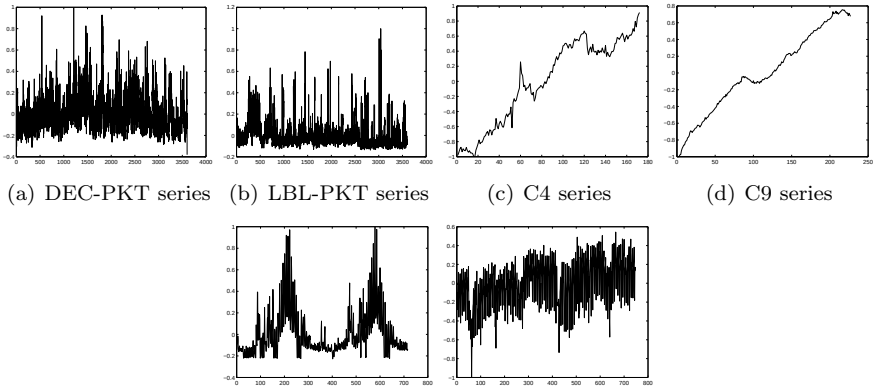
We can choose the optimal value for a minimizing the overall upper bound: $a = \frac{c_{\mathcal{F}}\sqrt{(d-1)T}}{F}$. □

6. Experiments

We run our algorithms on six real world time-series data sets. In the time series we consider there are no signals attached to the outcomes. However we can take vectors consisting of previous observations (we shall take ten of those) and use them as signals. The dataset DEC-PKT[a] contains an hour's worth of all wide-area traffic between the Digital Equipment Corporation and the rest of the world. The dataset LBL-PKT-4[a] consists of observations of an hour of traffic between the Lawrence Berkeley Laboratory and the rest of the world. We transformed both the data sets

[a]Data sets can be found at `http://ita.ee.lbl.gov/html/traces.html`

in such a way that each observation is the number of packets in the corresponding network during a fixed time interval of one second. The other four datasets[b] (C4,C9,E5,E8) relate to transportation data. Two of them, C9 and C11, contain low-frequency monthly traffic measures. The other two, E5 and E8, contain high-frequency day traffic measures. On each of these data sets the following operations were performed: subtraction of the mean value and division by the maximum absolute value. The resulting time series are shown in Figure 1.



(a) DEC-PKT series (b) LBL-PKT series (c) C4 series (d) C9 series

(e) E5 series (f) E8 series

Fig. 1. Time series from 6 data sets.

We used ten previous observations as an input vector for tested algorithms at each prediction step. We are solving the following 3-class classification problem: we predict whether the next value in a time series will exceed the previous value by ϵ or more, fall below the previous value by ϵ or more, or stay within the ϵ-vicinity of the previous value. The value of ϵ is chosen to be the median of all the changes in the data set.

In order to assess the quality of predictions, we calculate the cumulative square loss at the last two thirds of each time series (test set) and divide it by the number of examples (MSE). Since we are considering the online settings, we could calculate the cumulative loss from the beginning of each time series. However calculating the loss from one third of the dataset on allows us to eliminate starting effects, to choose the ridge parameter a fairly on the training set, and to compare the performance of our algorithms with batch algorithms usually used to solve this kind of problem.

The total square loss on the test set takes into account the quality of an algorithm only at the very end of the prediction process, and does not consider the quality during the process. We introduce another quality measure: at each step in the test set we calculate MSE of an algorithm until this step. After all the steps we average

[b]Data sets can be found http://www.neural-forecasting-competition.com/index.htm.

these MSEs (AMSE). Clearly, if one algorithm is better than another on the whole test set (its total MSE is smaller) but was often worse on many parts of the test set (total MSEs of many parts of the set is larger), this will reflect in AMSE. We will demonstrate the purpose of AMSE later.

We compare the performance of our algorithms with the multinomial logistic regression (mLog), which is a standard classification algorithm giving probability predictions

$$\gamma^i_{\text{mLog}} = \frac{e^{\theta_i x}}{\sum_{i=1}^d e^{\theta_i x}}$$

for all the components of the outcome $i = 1, \dots, d$. In our case $d = 3$. Here parameters $\theta_1, \dots, \theta_d$ are estimated from the training set. We apply this algorithm in two regimes: batch regime, where the algorithm learns only on the training set and is tested on the test set (and thus θ is not updated on the test set); and in the online regime, where at each step new parameters θ are found, and only one next outcome is predicted. The second regime is more suitable under online settings, but the first one is more standard and fast. In both the regimes logistic regression does not have theoretical guarantees for the square loss.

We also compare our algorithms with the simple predictor outputting the average of the previous classes (and thus always giving probability predictions).

We are not aware of other efficient algorithms for online probability prediction, and thus use logistic regression and simple predictor as the only baselines. Component-wise algorithms which could be used for online prediction (e.g., Gradient Descent from [8] or Ridge Regression from [7]), have to use normalization from Algorithm 2. Thus they have to be applied in some modified form and can not be directly compared with our algorithms.

The ridge for our algorithms is chosen to achieve the best MSE on the training set, which is the first third of each series. It is important to note that both cAAR and mAAR give initial probability of $1/d$ for each class (see motivation of cAAR in the beginning of Section 3.3: the mean for α is zero, so $\xi_1 = \ldots = \xi_{d-1} = \frac{1}{d}$, and $\xi_d = 1 - (d-1)\frac{1}{d} = \frac{1}{d}$. The class of points which lie within the ϵ-vicinity around the previous points is considered to be the asymmetric class (the d-th component) for mAAR. Indeed, it has the meaning of the remainder compared with the other two classes.

We also run kernelized algorithms with the Gaussian kernel $k(x_1, x_2) = \exp(-s\|x_1 - x_2\|_2^2)$, with s chosen among the numbers $0.001, 0.01, 0.5, 1, 2, 10$. Coordinate-wise algorithm is called cKAAR by analogy with the linear one. Parameter s for the kernel was chosen to be the one which provides the best MSE on the training set. In the online settings the effect of overfitting is reduced: predictions are evaluated on a set different from that where the algorithm is trained. Indeed, first the algorithms are trained on indexes $1 \dots \tilde{T}$, then the predictions are made for $\tilde{T} + 1$ and larger values of T. Kernelized algorithms can be made faster if approximation techniques or symmetric block matrix inversion formulas are used.

The results are shown in Table 1. We highlight the most precise algorithms for different data sets. We also show time needed to make predictions on the whole data set. The algorithms were implemented in Matlab R2010b and run on a laptop with 8Gb RAM and processor Intel Core i5, T7200, 2.40GHz.

As we can see from the table, the online methods perform better than the batch method. Online logistic regression performs well, but is very slow (in comparison with linear algorithms). Our linear algorithms perform similar to each other and comparable to the online logistic regression, but they are much faster. Performance of the kernelized algorithms is generally better than for the linear algorithms, and is often better than all of the competitors. Of course, one can achieve even better performance with kernelized algorithms by choosing a kernel function more suitable for this application, including, e.g., periodic terms in it (and dependency on time) especially for E5 and possibly for other datasets, or some non-zero increasing mean for C4, C9, E8. Our purpose is to demonstrate that the algorithms can be applicable in the real setting, and compare their strengths and weaknesses.

It is clear from the table that mKAAR is generally slower than cKAAR, and has very similar performance. Both kernelized algorithms slow down with time, because at each step t inversion of the matrix $t\times t$ (or $(d-1)t\times(d-1)t$ for mKAAR) is needed, but they also have much more potential for capturing important dependencies in the data. On smaller datasets the kernelized algorithms are much more efficient. They are much faster than logistic regression when applied online on C4 and C9.

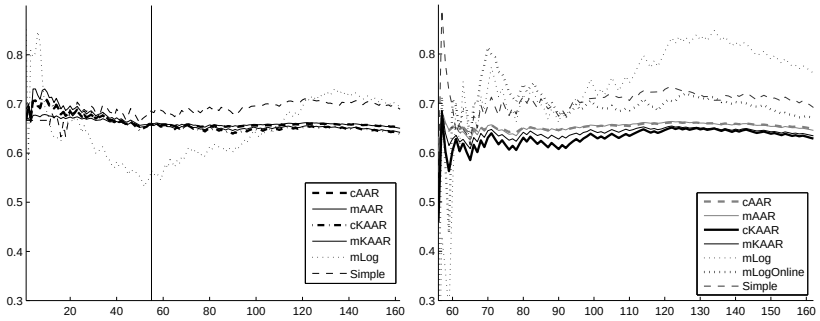
We tried to change the balance between the numbers of instances in each class by increasing and decreasing the width of the tube. This allowed us to check whether the number of instances in the third class (remainder) influenced the difference in performance between the algorithms. We did not observe any significant effect on performance though.

In order to demonstrate online performance of the algorithms, we can look how MSE evolves with time for different algorithms. Figure 2, left, shows the evolution of the MSE over time for the whole C4 set, and the right subfigure shows it for the test set. For the rest of the data sets the pictures look similar. For each point t on the horizontal axis, the vertical axis shows the mean square loss accumulated so far. For the test set graph, the mean error calculation starts from zero (this is the reason why the graphs look different after the test set starts). The performance of the mLogOnline is not shown for the whole set because its starting error is by far out of bounds of the graph, and the scale would not allow us to see the difference between the algorithms.

First, it is important to note that mLog clearly overfits: it performs well on the training set (the set left from the vertical line on the left graph), and then its MSE degrades on the test set. None of the online algorithms has this property, even though some of them are a little unstable at the start of the process. Another interesting feature is that the relative rankings of the performances change with time. Imagine, for example, that the test set ended at 105. Then MSE would show that mLogOnline was better than mLog, whereas is is clear from the graph that

Table 1. The square losses and prediction time (sec) of different algorithms applied for time series prediction. cAAR,mAAR, cKAAR, and mKAAR stand for the new algorithms, mLog states for the logistic regression, mLogOnline states for online logistic regression, and Simple stands for the simple average predictor. The numbers in brackets show the dataset sizes.

Algorithm	**MSE**	**AMSE**	**Time**	**MSE**	**AMSE**	**Time**
DEC-PKT (3602)				LBL-PKT (3600)		
cAAR	0.45906	0.45822	0.25	0.48147	0.4790	0.314
mAAR	0.45906	0.45822	0.624	0.48147	0.4790	0.692
cKAAR	**0.45444**	0.45389	1855.269	**0.46892**	0.4696	1792.648
mKAAR	0.45463	0.45444	20587.56	0.46966	0.47032	20505.947
mLog	0.46107	0.46265	0.234	0.47749	0.47482	0.332
mLog Online	0.45751	0.45762	1162.431	0.47598	0.47398	1351.749
Simple	0.58089	0.57883	0	0.57087	0.5657	0.007
C4 (172)				C9 (227)		
cAAR	0.64834	0.65447	0.008	**0.63238**	0.64082	0.012
mAAR	0.64538	0.65312	0.03	0.63338	0.64055	0.039
cKAAR	**0.62947**	0.63682	0.064	0.63448	0.66975	0.140
mKAAR	0.63453	0.6423	0.467	0.63353	0.66645	1.242
mLog	0.76849	0.77797	0.141	0.97718	0.91654	0.141
mLog Online	0.68164	0.7351	2.391	0.71178	0.75558	4.663
Simple	0.69037	0.69813	0.002	0.65090	0.65348	0.002
E5 (716)				E8 (747)		
cAAR	0.34452	0.34252	0.041	0.29395	0.29276	0.043
mAAR	0.34453	0.34252	0.127	0.29374	0.29223	0.13
cKAAR	0.33897	0.33697	7.023	0.27268	0.26490	7.881
mKAAR	0.3389	0.33694	62.952	**0.2715**	0.26114	72.206
mLog	0.31038	0.30737	0.544	0.31316	0.30382	0.04
mLog Online	**0.30646**	0.30575	216.429	0.27982	0.27068	40.509
Simple	0.58212	0.58225	0.001	0.69691	0.70527	0.001



(a) C4 all (b) C4 test only

Fig. 2. MSE after each step of the C4 set for different algorithms. The graph on the left is for the whole set, and the graph on the right is for the test set only.

the online algorithm became better after around 95 only. For this reason, AMSE is introduced: in this case, AMSE (average level of the MSE line) for mLogOnline would be higher than AMSE for mLog. Another important feature is that the per-

formance stays relatively stable for all our algorithms, and does not converge to zero. This characterizes the power of the experts chosen for prediction (how well the model describes the process). Indeed, if there was an ideal expert (with perfect performance), theoretical bounds (at least in the linear case) would guarantee the convergence of the MSE lines to zero. If there is no ideal expert, and the best expert makes small error at each step, the lines would converge to the value of that error. A different choice of the kernel will change the RKHS where the best expert is located, which may lead to lower convergence line. It is interesting though to see from the left graph that online algorithms improve with time as to be expected.

It is not surprising that cAAR and mAAR provide very similar performance (and the same can be said about the pair cKAAR and mKAAR). Indeed, they compete with the same experts, and the main component in the error is the component of the best expert's loss if the model for the data is not perfect (if best expert's loss is far from zero). Both algorithms converge to the same best expert, so asymptotically we can expect the same performance. In the beginning of the prediction process their performances can be quite different though, and that can be seen on the left graph of Figure 2. Lines for cAAR and mAAR (and for cKAAR and mKAAR) are far from each other at first, but then converge to similar values.

We should also note that logistic regression can be considered in the online regression framework too, and theoretical guarantees on the square loss can be proven for an algorithm competing with logistic regression experts as in [18]. This algorithm can as well be kernelized and guarantees for the kernelized algorithm can be proven (see [17], Section 4.5).

Summarizing the experimental results, we can say that although mLogOnline gives good results on many data sets, it is often not as good as kernelized algorithms. Kernelized algorithms also have the flexibility of changing the kernel and thus having better model for the data. They are fast to run on small data sets, but are very slow on large data sets, often significantly slower than mLogOnline. Linear algorithms often provide similar results, but are much faster to run. They can be better than all the rest of the algorithms for some of the problems, for example if the dimension of the input vectors is very high. Batch logistic regression is very fast to run but often has much worse precision.

7. Discussion

We considered an important generalization of the online classification problem. We presented new algorithms which give probability predictions in the Brier game. Both algorithms do not involve any numerical integration, and can be easily computed. Both algorithms have theoretical guarantees on their cumulative losses. One of the algorithms is kernelized and a theoretical bound is proven for the kernelized algorithm. We performed experiments with linear and kernelized algorithms and showed their strengths and weaknesses by evaluating performance, time to train and predict, flexibility, and stability of the performance. We have provided results for

different scenarios: when time series are growing (C4 and C9), when the noise is more shifted to one of the sides from the mean (DEC and LBL), when there is a periodic component of the series (E5), and in a more stable setting (E8). We compared them with the logistic regression, the benchmark algorithm giving probability predictions.

An important open question is to obtain lower bounds for the loss of our algorithms. Judging from the lower bounds in [15] one can conjecture that the regret term of order $O(\ln T)$ is optimal for the case of linear model (1). It is possible that the multiplicative constant can be improved though.

Acknowledgments

The authors are grateful to Alexey Chernov, Vladimir Vovk, and Alex Gammerman for useful discussions and to anonymous reviewers for helpful comments. This work has been supported by EPSRC grant EP/F002998/1 and ASPIDA grant from the Cyprus Research Promotion Foundation.

References

1. Edwin F. Beckenbach and Richard Bellman. *Inequalities*. Springer, Berlin, 1961.
2. Glenn W. Brier. Verification of forecasts expressed in terms of probability. *Monthly Weather Review*, 78:1–3, 1950.
3. Nicolò Cesa-Bianchi and Gábor Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, Cambridge, UK, 2006.
4. Alexander Gammerman, Yuri Kalnishkan, and Vladimir Vovk. On-line prediction with kernels and the complexity approximation principle. In *Proceedings of the 20th Conference in Uncertainty in Artificial Intelligence*, pages 170–176, 2004.
5. David A. Harville. *Matrix Algebra From a Statistician's Perspective*. Springer, New York, 1997.
6. David Haussler, Jyrki Kivinen, and Manfred K. Warmuth. Sequential prediction of individual sequences under general loss functions. *IEEE Transactions on Information Theory*, 44:1906–1925, 1998.
7. Arthur E. Hoerl and Robert W. Kennard. Ridge Regression: biased estimation for nonorthogonal problems. *Technometrics*, 42:80–86, 2000.
8. Jyrki Kivinen and Manfred K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132:1–63, 1997.
9. Jyrki Kivinen and Manfred K. Warmuth. Averaging expert predictions. In *Proceedings of the 4th European Conference on Computational Learning Theory*, pages 153–167. Springer, Berlin, 1999.
10. Jyrki Kivinen and Manfred K. Warmuth. Relative loss bounds for multidimensional regression problems. *Machine Learning*, 45:301–329, 2001.
11. Christian Michelot. A finite algorithm for finding the projection of a point onto the canonical simplex of $\mathbb{R}^n$. *Journal of Optimization Theory and Applications*, 50:195–200, 1986.
12. Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2002.
13. Vladimir Vovk. Aggregating strategies. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 371–383, San Mateo, CA, 1990. Morgan Kaufmann.

14. Vladimir Vovk. A game of prediction with expert advice. *Journal of Computer and System Sciences*, 56:153–173, 1998.
15. Vladimir Vovk. Competitive on-line statistics. *International Statistical Review*, 69:213–248, 2001.
16. Vladimir Vovk and Fedor Zhdanov. Prediction with expert advice for the Brier game. In *Proceedings of the 25th International Conference on Machine Learning*, pages 1104–1111, 2008.
17. Fedor Zhdanov. *Theory and Applications of Competitive Prediction*. PhD thesis, Department of Computer Science, Royal Holloway University of London, UK, 2011.
18. Fedor Zhdanov and Vladimir Vovk. Competitive online generalized linear regression under square loss. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases 2010*, pages 531–546, 2010.