

Linear Probability Forecasting

Fedor Zhdanov and Yuri Kalnishkan
 Computer Learning Research Centre,
 Department of Computer Science,
 Royal Holloway University of London,
 Egham, Surrey, TW20 0EX, UK
 {fedor,yura}@cs.rhul.ac.uk

Abstract

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 suggest two computationally efficient algorithms to work with these problems and prove theoretical guarantees on their losses. We kernelize one of the algorithms and prove theoretical guarantees on its loss. We perform experiments and compare our algorithms with logistic regression.

1 Introduction

Online prediction is a wide area of machine learning (see Cesa-Bianchi and Lugosi, 2006). Its algorithms can be applied to different data mining problems (see for example Freund and Schapire, 1997). Online prediction provides efficient algorithms which adapt to a predicted process “on fly”. In online regression framework we assume the existence of some input at each step and try to predict an outcome on this input. This process is repeated step by step. We consider multi-dimensional Brier game where outcomes and predictions come from a simplex and can be thought of as probability distributions on the vertices of the simplex. If the outcomes are identified with vertices of the simplex this problem can be thought of as the multi-class classification problem of the given input.

In the simple case the dependence between the input and its outcome is assumed to be linear; linear regression minimising the expected loss is studied in statistics. As opposite to the traditional statistical setting, the learner in online prediction does not make any statistical assumptions about the data generating process. Its goal is to predict as well as the best linear function on input. Instead of looking for the best linear function, our learner considers all linear functions and makes his prediction by mixing them in a certain way at each prediction step. We prove theoretical bounds on the cumulative loss of the learner in comparison with the cumulative loss of the best linear function (we

say the learner competes with these functions). We consider the square loss: mean square error is one of the benchmark measures for classification algorithms (see Brier, 1950).

We use Vovk’s Aggregating Algorithm (a generalization of the Bayesian mixture) to mix functions (as in Aggregating Algorithm Regression, AAR: see Vovk, 2001). This method has previously been applied to the case when possible outcomes lie in a segment of the real line, and so the prediction was one-dimensional. We develop two algorithms to solve the problem of multi-dimensional prediction. The first algorithm applies a variant of AAR to predict each coordinate of the outcome separately, and then combines these predictions in a certain way to get probability prediction. The other algorithm is designed to give probability predictions directly; these are first computationally efficient online regression algorithm designed to solve linear and non-linear multi-class classification problems. We derive theoretical bounds on the losses of both algorithms. We come to an unexpected 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 is very similar.

One component of the prediction of the second algorithm has the meaning of a remainder. In practice this situation is quite common. For example, in a football match either one team wins or the other, and the remainder is a draw (see Vovk and Zhdanov (2008) for online prediction experiments in football). When we analyse a precious metal alloy we may look for a description of the following kind: the alloy has 40% of gold, 35% of silver, and some addition (e.g., copper and palladium). It is common for financial applications to predict the direction of the price: the price can go up, down, or stay close to the current value. We perform classification experiments with linear algorithms and compare them with logistic regression.

A description of the framework can be found in Section 2, description of the algorithms can be found in Section 3, and derivation of the theoretical bounds can be found in Section 4.

We look for a way to extend the class of experts using the kernel trick. We kernelize the second algorithm and prove 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 RKHS given by a kernel parameter it uses. Kernelization process is described in Section 5. Our experiments are shown in Section 6. Section 7 makes the conclusions and shows some possibilities for prospective work.

2 Framework

A game of prediction contains three components: a space Ω of outcomes, a decision space Γ , and a loss function $\lambda : \Omega \times \Gamma \rightarrow \mathbb{R}$. We are interested in the generalisation of the Brier game from Brier (1950) where the space of outcomes $\Omega = \mathcal{P}(\Sigma)$ is the set of all probability measures on a finite set Σ with d elements, $\Gamma := \{(\gamma_1, \dots, \gamma_d) : \sum_{i=1}^d \gamma_i = 1, \gamma_i \in \mathbb{R}\}$ is a hyperplane in d -dimensional space

containing all the outcomes, and for any $y \in \Omega$ we define the loss

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

For example, if $\Omega = \{1, 2, 3\}$, $\omega = 1$, $\gamma\{1\} = 1/2$, $\gamma\{2\} = 1/4$, and $\gamma\{3\} = 1/4$, $\lambda(\omega, \gamma) = (1/2 - 1)^2 + (1/4 - 0)^2 + (1/4 - 0)^2 = 3/8$. 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 $x_t \in \mathbf{X} \subseteq \mathbb{R}^n$, and follows prediction protocol 1.

Protocol 1 Protocol of forecasting game

$L_0 := 0$.
for $t = 1, 2, \dots$ **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

We find an algorithm which is capable of competing with all linear functions (we call them experts) $\xi_t = (\xi_t^1, \dots, \xi_t^d)'$ on x :

$$\begin{aligned} \xi_t^1 &= 1/d + \alpha'_1 x_t \\ &\dots \\ \xi_t^{d-1} &= 1/d + \alpha'_{d-1} x_t \\ \xi_t^d &= 1 - \xi_t^1 - \dots - \xi_t^{d-1} = 1/d - \left(\sum_{i=1}^{d-1} \alpha_i \right)' x_t, \end{aligned} \tag{1}$$

where $\alpha_i = (\alpha_i^1, \dots, \alpha_i^n)'$, $i = 1, \dots, d-1$. In the model (1) the prediction for the last component of an outcome is calculated from the predictions for other components. Denote $\alpha = (\alpha'_1, \dots, \alpha'_{d-1})' \in \Theta = \mathbb{R}^{n(d-1)}$. Then any expert can be presented as $\xi_t = \xi_t(\alpha)$. Let also $L_T(\alpha) = \sum_{t=1}^T \lambda(y_t, \xi_t(\alpha))$ be the cumulative loss of an expert α over T trials.

3 Derivation of the algorithms

In this section we describe how we apply the Aggregating Algorithm (AA) proposed in Vovk (1990) to mix experts and make predictions. The algorithm keeps weights $P_{t-1}(d\alpha)$ for the experts at each prediction step t , and updates them by the exponential weighting scheme after the actual outcomes is announced:

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

Here $\beta = e^{-\eta}$, where $\eta \in (0, \infty)$ is a learning rate parameter. This weight update ensures that the experts which predict badly at the step t receive less weight. The weights are then normalized $P_t^*(d\alpha) = \frac{P_t(d\alpha)}{P_t(\Theta)}$.

The prediction of the algorithm is a combination of the experts' predictions. It is suggested in Kivinen and Warmuth (1999) that the prediction is simply the weighted average of the experts' predictions with weights $P_t(d\alpha)$. The Aggregating Algorithm uses more sophisticated prediction scheme, and sometimes achieves better theoretical performance. It first defines a *generalised prediction* at any step t as a function $g_t : \Omega \rightarrow \mathbb{R}$ such that

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

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

$$\lambda(y, \gamma_t) \leq g_t(y) \quad (4)$$

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 Vovk (1998). It is also shown there that for countable (and thus finite) number of experts the AA achieves the best possible theoretical guarantees.

3.1 Proof of mixability

In this section we prove that our game is perfectly mixable and show a function that can be used to give predictions satisfying (4).

It is shown in Theorem 1 Vovk and Zhdanov (2008) that the Brier game with finite number of outcomes is perfectly mixable iff $\eta \in (0, 1]$. The two authors of that paper consider the outcome space of d probability measures concentrated in points of Σ . We denote this space by $\mathcal{R}(\Sigma)$. They consider experts giving predictions from all probability measures $\mathcal{P}(\Sigma)$. We need to prove that the inequality (4) holds for our experts (1) (who can give predictions outside of the probability simplex) and our outcome space Ω (the whole probability simplex, not only its vertices). Lemma 2 describes the first part, but first we need to state an additional statement. The following lemma shows that any vector from \mathbb{R}^d can be projected into simplex without increasing the Brier loss.

Lemma 1. *For any $\xi = (\xi_1, \dots, \xi_d) \in \mathbb{R}^d$ there exists $\theta = (\theta_1, \dots, \theta_d) \in \mathcal{P}(\Sigma)$ such that for any $y \in \Omega$ we have $\lambda(y, \theta) \leq \lambda(y, \xi)$.*

Proof. The Brier loss of a prediction γ is a square Euclidean distance between γ and the actual outcome y in a d -dimensional space. The proof follows from the fact that Ω is a convex and closed set in \mathbb{R}^d . \square

Lemma 2. *Let $P(d\alpha)$ be any probability distribution on Θ . Then for any $\eta \in (0, 1]$ there exists $\gamma \in \Gamma$ such that for any $y \in \mathcal{R}(\Sigma)$ we have*

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

Proof. By Lemma 1 for any $\xi(\alpha)$ we can find $\theta(\alpha) \in \mathcal{P}(\Sigma)$ such that the loss of experts decreases: $\lambda(y, \theta(\alpha)) \leq \lambda(y, \xi(\alpha))$ for any $y \in \mathcal{R}(\Sigma)$. Thus we have

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

for any $y \in \mathcal{R}(\Sigma)$. We can take the same prediction $\gamma \in \Gamma$ that satisfies the necessary inequality with θ instead of ξ . By Theorem 1 in Vovk and Zhdanov (2008) such prediction exists for any $\eta \in (0, 1]$ ($\beta \in [e^{-1}, 1)$). \square

A way to convert the generalised prediction into the prediction of AA is called a substitution function. We prove that we can use the same substitution function and the same learning rate parameter η as for the case of finite number of possible outcomes. Such a function is proposed in Vovk and Zhdanov (2008). This is an extension of Lemma 4.1 from Haussler et al. (1998).

Lemma 3. *Let $P(d\alpha)$ be a probability distribution on Θ and put*

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

for every $y \in \Omega$. Then if γ is such a prediction that $\lambda(z, \gamma) \leq f(z)$ for any $z \in \mathcal{R}(\Sigma)$ then $\lambda(y, \gamma) \leq f(y)$ for any $y \in \Omega$.

Proof. For the typographical reasons we will write ξ instead of $\xi(\alpha)$. It is easy to ensure that $\lambda(y, \gamma) - \lambda(y, \xi) = \sum_{\sigma \in \Sigma} y\{\sigma\} [\lambda(z_\sigma, \gamma) - \lambda(z_\sigma, \xi)]$ for $z_\sigma\{\rho\} = 0$ if $\sigma \neq \rho$ and $z_\sigma\{\rho\} = 1$ if $\sigma = \rho$. We also have that $\lambda(y, \gamma) - f(y) \leq 0$ is equivalent to $\int_{\Theta} \beta^{\lambda(y, \xi) - \lambda(y, \gamma)} P(d\alpha) \leq 1$. Thus due to the convexity of the exponent function $\int_{\Gamma} \beta^{\sum_{\sigma \in \Sigma} y\{\sigma\} [\lambda(z_\sigma, \xi) - \lambda(z_\sigma, \gamma)]} P(d\alpha) \leq \sum_{\sigma \in \Sigma} y\{\sigma\} = 1$. \square

Let us denote the i -th possible outcome from $\mathcal{R}(\Sigma)$ by $y\{i\}$, $i = 1, \dots, d$. We use the substitution function defined by the following proposition:

Proposition 1. *Let $r_i = g(y\{i\})$, and $x^+ = \max(x, 0)$. Define $s \in \mathbb{R}$ by the requirement*

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

If the prediction of the Aggregating Algorithm is given by

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

then (4) holds.

This function allows us to avoid weights normalization in calculating the generalized prediction at each step (avoid * in the weights distribution), which would be computationally inefficient. Suppose we can get only $r = g_t(y) + C$ instead of $g_t(y)$, where C is the same for all y . Then predictions γ_t defined by the substitution function from Proposition 1 will be the same as if we calculated the generalized prediction with weights normalization.

3.2 Algorithm for multidimensional outcomes

We set the prior weights distribution P_0 over the set $\Theta = \mathbb{R}^{n(d-1)}$ of experts α to have the Gaussian density with a parameter $a > 0$:

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

Instead of taking the integral in (3) we get a shifted generalised prediction r by calculating $r_i = g_T(y\{i\}) - g_T(y\{d\})$ (we omit the index T in r for brevity). Each component of $r = (r_1, \dots, r_d)$ corresponds to one of the possible outcomes, so $r_d = 0$. Other components, $i = 1, \dots, d-1$:

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

where by $Q(\alpha, y)$ we denote the quadratic form:

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

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

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

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

is a quadratic part of $Q(\alpha, y)$. Here A is a square matrix with $n(d-1)$ rows (see the 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 $h_i = -2 \sum_{t=1}^{T-1} (y_t^i - y_t^d) x_t$, $i = 1, \dots, d-1$ make up a big vector $h = (h_1', \dots, h_{d-1}')'$. The remainder is equal to

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

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

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

The proof of this lemma can be found in Harville (1997, Theorem 15.12.1). Following this lemma, we can rewrite r_i as $r_i = F(A, b_i, z_i)$, $i = 1, \dots, d-1$, where

$$F(A, b_i, z_i) = \min_{\alpha \in \Theta} Q(\alpha, y^i) - \min_{\alpha \in \Theta} Q(\alpha, y^d).$$

Variables b_i, z_i and the precise formula for F are defined by the following lemma

Lemma 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 in F by 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 Harville, 1997, Theorem 19.1.1). \square

We can see that $b_i = h + (x'_T, \dots, x'_T, \mathbf{0}, x'_T, \dots, x'_T)' \in \mathbb{R}^{n(d-1)}$, where $\mathbf{0}$ is a zero-vector from \mathbb{R}^n . We also have $z_i = (-x'_T, \dots, -x'_T, -2x'_T, -x'_T, \dots, -x'_T)'$. Thus we can calculate $d-1$ differences r_i , assign $r_d = 0$, and then apply the substitution function from proposition 1 to get predictions. The resulting algorithm is Algorithm 1. We will further call it 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 should not directly use the algorithm and the theoretical bound proposed in Vovk (2001). Vovk's experts do not allow us to take advantage of the fact that only one outcome is possible to happen at each moment. They are more suitable for the case when each input vector x can belong to many classes simultaneously in case of classification. In other words, they are centered around the center $1/2$ of the prediction interval $[0, 1]$: $\xi_i = 1/2 + \alpha_i x$. Assume that the number of outcomes is very large and the distribution on experts is normal $N(0, \sigma^2)$ with small σ . Then the average experts' prediction is $(1/2, \dots, 1/2, 1 - (d-1)/2)$, and the average loss of the experts on trials with the same outcome $y = y\{i\}$ (we can take $y = (1, 0, \dots, 0)$) is $(d-1)/2^2 + (d-1)^2/2^2$.

Algorithm 1 mAAR for the Brier game

Fix $n, a > 0$. $C = 0, h = 0$.

for $t = 1, 2, \dots$ **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 $b_i = h + (x_t', \dots, x_t', 0, x_t', \dots, x_t')'$, where 0 is a zero-vector from \mathbb{R}^n is placed at i -th position, $i = 1, \dots, d-1$.

Set $z_i = (-x_t', \dots, -x_t', -2x_t', -x_t', \dots, -x_t')'$, where $-2x_t'$ is placed at i -th position, $i = 1, \dots, d-1$.

Calculate $r_i := -b_i' A^{-1} z_i, r_d := 0, i = 1, \dots, d-1$.

Solve $\sum_{i=1}^d (s - r_i)^+ = 2$ in $s \in \mathbb{R}$.

Set $\gamma_t^i := (s - r_i)^+ / 2, \omega \in \Omega, i = 1, \dots, d$.

Output prediction $\gamma_t \in \mathcal{P}(\Omega)$.

Read observation y_t .

$h_i = h_i - 2(y_t^i - y_t^d)x_t, h = (h_1', \dots, h_{d-1}')'$.

end for

Components of experts (1) concentrate around the point $1/d$, and so experts have the average loss $(d-1)/d^2 + (1-1/d)^2$. This loss is smaller than the loss of Vovk's experts for large values of d .

Our component-wise experts are expressed by

$$\xi_t^i = 1/d + \alpha_i' x_t, \quad i = 1, \dots, d. \quad (5)$$

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

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

Note that the value for $\tilde{\eta}$ here will be different from 1 since the loss function by 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 1 we use the substitution function suggested in Vovk (2001) 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

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

for $i = 1, \dots, d$. 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 4 and Lemma 5.

Then this method projects its prediction onto the prediction simplex such that the loss does not increase. We use the projection algorithm suggested in Michelot (1986).

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 $i = 1, \dots, d$ then break;

$$I = I \cup \{i : \gamma_T^i < 0\};$$

If $\gamma_T^i < 0$ for some i then $\gamma_T^i = 0$;

end while

4 Theoretical bound

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

4.1 Component-wise algorithm

We prove here the theoretical bound for the loss of cAAR. The following lemma is the main tool helping us to prove our theorems. It is easy to prove the following statement (Lemma 1 from Vovk (2001)):

Lemma 6. *If the learner follows the Aggregating Algorithm in a perfectly mixable game, then for every positive integer T , every sequence of outcomes of the*

length T , and any initial weights distribution on experts $P_0(d\alpha)$ it suffers loss satisfying for any $\alpha \in \Theta$

$$L_T(\text{AA}(\eta, P_0)) \leq \log_\beta \int_\Theta \beta^{L_T(\alpha)} P_0(d\alpha). \quad (7)$$

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

$$\begin{aligned} L_T(\text{AA}(\eta, P_0)) &\leq L_{T-1}(\text{AA}(\eta, P_0)) + g_T(\omega_T) \\ &= \log_\beta \int_\Theta \beta^{L_{T-1}^\theta} P_0(d\theta) + \log_\beta \int_\Theta \beta^{\lambda(\omega_T, \xi_t^\theta)} \frac{\beta^{L_{T-1}^\theta}}{\int_\Theta \beta^{L_{T-1}^\theta} P_0(d\theta)} P_0(d\theta) \\ &= \log_\beta \int_\Theta \beta^{L_T^\theta} P_0(d\theta). \end{aligned}$$

Here the second equality follows from the inductive assumption, the definition (3) of g_T , and (2). \square

The loss of the component-wise algorithm by one component is bounded as in the following theorem.

Theorem 1. *Let the outcome space in the prediction game be $[A, B]$, $A, B \in \mathbb{R}$. 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 $\|x_t\|_\infty \leq X, \forall t$. 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*

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

Proof. We need to prove that the game is perfectly mixable (see (4)) and find the optimal parameter η for the algorithm. Implications similar to the ones in the proof of Lemma 2 from Vovk (2001) lead to the inequality $\eta \leq \frac{2}{(B-A)^2}$. Clearly, Lemma 6 holds for our case, so we need only to calculate the difference between the right-hand side of (7)

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

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. Then due to Lemma 4 this difference will be equal to

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

We bound the determinant of a symmetric positive definite matrix by the product of its diagonal elements (see Beckenbach and Bellman (1961), Chapter 2, Theorem 7) and use $\eta = \frac{2}{(B-A)^2}$. \square

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 2. *If $\|x_t\|_\infty \leq X, \forall t$, 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*

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

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 $\|\sum_{i=1}^{d-1} \alpha_i\|_2^2 \leq (d-1) \sum_{i=1}^{d-1} \|\alpha_i\|_2^2$ 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 1. \square

4.2 Linear forecasting

The theoretical bound for the loss of the Algorithm 1 is

Theorem 3. *If $\|x_t\|_\infty \leq X, \forall t$, then for any $a > 0$, every positive integer T , every sequence of outcomes of the length T , and any $\alpha \in \mathbb{R}^{n(d-1)}$ mAAR(2a) satisfies*

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

Proof. We apply mAAR with the parameter $b = 2a$. Recall that $C = \sum_{t=1}^T x_t x_t'$. Following the line of the proof of Theorem 1 with $\eta = 1$ we get the theoretical bound. \square

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 2. *In the conditions of Theorem 3 $\text{mAAR}(a)$ satisfies*

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

The theoretical bound (10) is worse asymptotically by d than the 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 in the important case 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 can be considered not rich enough to describe data well, and a more complicated model is needed. We use a popular in computer learning kernel trick, firstly applied to the AAR in Gammernan et al. (2004). We derive an algorithm competing with all sets of functions from an RKHS with $d-1$ elements.

5.1 Derivation of the algorithm

Definition 1. Let us take $x_1, \dots, x_n \in \mathbf{X}$. A *kernel function* is a nonnegative function $K : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$ satisfying $\sum_{i,j=1}^n K(x_i, x_j) \xi_i \xi_j \geq 0$ for all positive integers n , all $x_1, \dots, x_n \in \mathbf{X}$, and $\xi_1, \dots, \xi_n \in \mathbb{R}$.

An RKHS contains all linear regressors $\langle \Phi(\cdot), h \rangle_H$ defined by means of a feature map (for all the definitions see Schölkopf and Smola, 2002). It can also be defined in a different equivalent way as a functional Hilbert space with continuous evaluation functional $\varphi : f \in \mathcal{F} \mapsto f(x)$ for each $x \in \mathbf{X}$. We will use the notation $c_{\mathcal{F}}(x)$ for the norm of this functional: $c_{\mathcal{F}}(x) := \sup_{f: \|f\|_{\mathcal{F}} \leq 1} |f(x)|$ and for the embedding constant $c_{\mathcal{F}} := \sup_{x \in \mathbf{X}} c_{\mathcal{F}}(x)$ and assume $c_{\mathcal{F}} < \infty$.

Our algorithm competes with the following experts:

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

Here $f_1, \dots, f_{d-1} \in \mathcal{F}$ are any functions from some RKHS \mathcal{F} . We start by rewriting mAAR in the dual form. Denote

$$\begin{aligned} \tilde{Y}_i &= -2(y_1^i - y_1^d, \dots, y_{T-1}^i - y_{T-1}^d, -1/2), \\ \bar{Y}_i &= -2(y_1^i - y_1^d, \dots, y_{T-1}^i - y_{T-1}^d, 0) \\ \tilde{k}(x_T) &= (x_1' x_T, \dots, x_T' x_T)', \\ \tilde{K} &= (x_s', x_t)_{s,t} \text{ is the matrix of scalar products} \end{aligned}$$

for $i = 1, \dots, d-1$, $s, t = 1, \dots, T$. We show that the predictions of mAAR can be represented in terms of variables defined above. We will need the following matrix property.

Proposition 3. *Let B, C be matrices such that the number of rows in B equals to the number of columns in C , and identity matrices I . If $aI + CB$ and $aI + BC$ are nonsingular then*

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

Proof. This is equivalent to $(aI + BC)B = B(aI + CB)$. That is true because of distributivity of matrix multiplication. \square

$$\text{Let us set } A = \left(aI + \begin{pmatrix} 2\tilde{K} & \cdots & \tilde{K} \\ \vdots & \ddots & \vdots \\ \tilde{K} & \cdots & 2\tilde{K} \end{pmatrix} \right).$$

Lemma 7. *On trial T values r_i for $i = 1, \dots, d-1$ in mAAR can be represented as*

$$r_i = \begin{pmatrix} \tilde{Y}_1 & \cdots & \bar{Y}_i & \cdots & \tilde{Y}_{d-1} \end{pmatrix} \cdot A^{-1} \begin{pmatrix} \tilde{k}(x_T)' & \cdots & 2\tilde{k}(x_T)' & \cdots & \tilde{k}(x_T)' \end{pmatrix}'. \quad (14)$$

Proof. By $M = (x_1, \dots, x_T)$ denote a matrix $n \times T$ of column input vectors. Let us set

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

Then h_i from the algorithm mAAR equals $h_i = M\bar{Y}_i \in \mathbb{R}^n$. Decompose $b'_i = \begin{pmatrix} \tilde{Y}_1 & \cdots & \bar{Y}_i & \cdots & \tilde{Y}_{d-1} \end{pmatrix} C$, where only the i -th block uses \bar{Y}_i . The matrix A is equal $A = aI + BC$. Using proposition 3

$$r_i = -b'_i A^{-1} z_i = - \begin{pmatrix} \tilde{Y}_1 & \cdots & \bar{Y}_i & \cdots & \tilde{Y}_{d-1} \end{pmatrix} \cdot (aI + CB)^{-1} C \begin{pmatrix} -x'_T & \cdots & -2x'_T & \cdots & -x'_T \end{pmatrix}'.$$

Note that $\tilde{K} = M'M$ and $\tilde{k}(x_T) = M'x_T$, thus (14) holds. \square

If instead of dot product in $\tilde{K}, \tilde{k}(x_T)$ we can choose a different kernel (classical examples of kernels are Gaussian (RBF): $K(x_i, x_j) = e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}}$, Vapnik's polynomial $K(x_i, x_j) = (x_i \cdot x_j + 1)^d$, etc.). To get predictions one can use the same substitution function from Proposition 1. 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.

Lemma 8 (Matrix determinant identity). *Let B, C be as in Proposition 3, and a is a real number. Then $\det(aI + BC) = \det(aI + CB)$.*

Proof. The proof is by considering a block matrix identity. \square

The main theorem follows from the property of RKHS called Representer theorem (see Schölkopf and Smola, 2002, Theorem 4.2).

Theorem 4 (Representer theorem). *Denote by $g : [0, \infty) \rightarrow \mathbb{R}$ a strictly monotonic increasing function. Assume \mathbf{X} is an arbitrary set, and \mathcal{F} is a Reproducing Kernel Hilbert Space of functions on \mathbf{X} with the given kernel $K : \mathbf{X}^2 \rightarrow \mathbb{R}$. Assume we also have a positive integer T and an arbitrary loss function $c : (\mathbf{X} \times \mathbb{R}^2)^T \rightarrow \mathbb{R} \cup \{\infty\}$. Then each minimizer $f \in \mathcal{F}$ of*

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

admits a representation of the form $f(x) = \sum_{i=1}^T \alpha_i K(x_i, x)$ for any $x \in \mathbf{X}$ and reals $\alpha_i, i = 1, \dots, T$.

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

Theorem 5. *Assume \mathbf{X} is an arbitrary set of inputs and \mathcal{F} is a Reproducing Kernel Hilbert Space of functions on \mathbf{X} with the given kernel $K : \mathbf{X}^2 \rightarrow \mathbb{R}$. Then for any $a > 0$, any $f_1, \dots, f_{d-1} \in \mathcal{F}$, any positive integer T , and any sequence of inputs and outputs $(x_1, y_1), \dots, (x_T, y_T)$*

$$L_T(\text{mKAAR}) \leq L_T(f) + a \sum_{i=1}^{d-1} \|f_i\|_{\mathcal{F}}^2 + \frac{1}{2} \ln \det A \quad (15)$$

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

Proof. The bound follows from Theorem 3 for mAAR and the Representer theorem. Let us first consider the case with scalar product kernel. Denote $C = \sum_{t=1}^T x_t x_t'$. By Lemma 8 and calculations similar to ones in the proof of Lemma 7 we have the equality of determinants. So we can use any other kernel instead of scalar product to get the term with the determinant. The Representer theorem assures that the minimum of the expression $L_T(f) + a \sum_{i=1}^{d-1} \|f_i\|_{\mathcal{F}}^2$ by f -s is reached on a linear regressor. \square

We can represent the bound (15) in another form which is more familiar from the on-line prediction literature:

Corollary 1. *Under assumptions of Theorem 5 and if we know the number of steps T in advance and are given $F > 0$, the mKAAR reaches the performance*

$$L_T(\text{mKAAR}) \leq L_T(f) + 2c_{\mathcal{F}}F\sqrt{(d-1)T}, \quad (16)$$

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

Proof. Bounding the logarithm of the determinant we have $\ln \det A \leq (d-1)T \ln \left(1 + \frac{2c_{\mathcal{F}}^2}{a}\right)$. We can choose the value for a where the minimum is achieved: $a = \frac{c_{\mathcal{F}}\sqrt{(d-1)T}}{F}$. \square

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. Data set DEC-PKT¹ contains an hours worth of all wide-area traffic between Digital Equipment Corporation and the rest of the world. Data set LBL-PKT-4¹ consists of observations of another hour of traffic between the Lawrence Berkeley Laboratory and the rest of the world. We transformed both the data sets 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 datasets² (C4,C9,E5,E8) relate to transportation data. Two of them (C9,C11) contain low-frequency monthly traffic measures. Two of them (E5,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.

We used ten previous observations as an input vector for tested algorithms at each prediction step. We are solving the 3-class classification problem: we predict whether the next value in a time series will be more than the previous value plus a precision parameter ϵ , less than that value, or lies in the 2ϵ tube around the previous value. The precision ϵ is chosen to be the median of all the changes in a 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 setting, we could calculate the cumulative loss from the beginning of each time series. However our approach is not sensitive to starting effects, it allows us to choose the ridge parameter a fairly on the training set, and it allows us to compare the performance of our algorithms with batch algorithms, which would be normally used to solve this problem.

The 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

¹Data sets can be found <http://ita.ee.lbl.gov/html/traces.html>.

²Data sets can be found <http://www.neural-forecasting-competition.com/index.htm>.

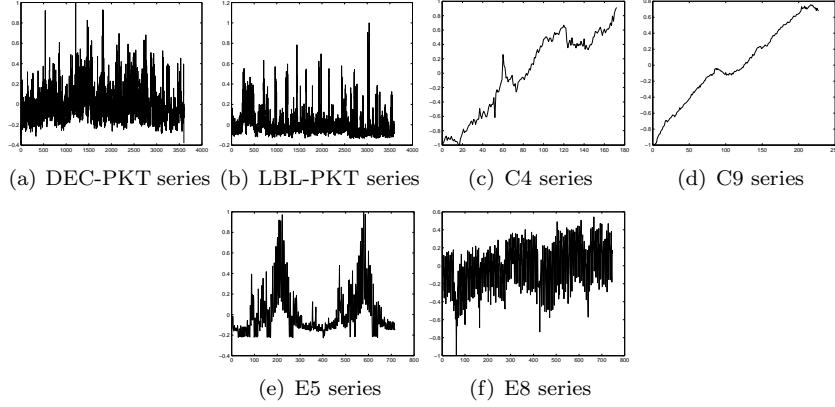


Figure 1: Time series from 6 data sets.

test set we calculate MSE of an algorithm until this step. After all the steps we average 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 measure takes it into account.

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

$$\gamma_{\text{mLog}}^i = \frac{e^{\theta^i x}}{\sum_{i=1}^d e^{\theta^i x}}$$

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 fair to compare with online algorithms, but the first one is standard and faster. In both regimes logistic regression does not have theoretical guarantees on the square loss.

We also compare our algorithms with the simple predictor predicting the average of the ten previous outcomes (and thus it always gives probability predictions).

We are not aware of other efficient algorithms for online probability prediction, and thus logistic regression and simple predictor as the only baselines. Component-wise algorithms which could be used for online prediction (e.g., Gradient Descent, Kivinen and Warmuth 1997, Ridge Regression, Hoerl and Kennard 2000), have to use normalization by Algorithm 2. Thus they have to be applied in a different way than they are described in the corresponding papers, and can not be fairly compared with our algorithms.

The ridge for our algorithms is chosen to achieve the best MSE on the training set: the first third of each series. 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 R2007b and run on the laptop with 2Gb RAM and processor Intel Core 2, T7200, 2.00GHz.

As we can see from the table, online methods perform better than the batch method. Online logistic regression performs well, but is very slow. Our algorithms perform similar to each other and comparable to the online logistic regression, but are much faster.

7 Discussion

We consider 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 algorithms and showed that they perform relatively well. We compared them with the logistic regression: the benchmark algorithm giving probability predictions.

Competing with linear experts in the case where possible outcomes lie in a more than 2-dimensional simplex was not widely considered by other researchers, so the comparison of theoretical bounds can not be performed. Kivinen and Warmuth’s work Kivinen and Warmuth (2001) includes the case when the possible outcomes lie in a more than 2-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 $O(\sqrt{\mathcal{L}_T(\alpha)})$ which is upper unbounded in the worst case. Their prediction algorithm is not computationally efficient and it is not clear how to extend their results for the case when the predictors lie in an RKHS.

We can prove lower bounds for the regret term of the order $O(\frac{d-1}{d} \ln T)$ for the case of the linear model (1) using methods similar to ones described in Vovk (2001), and lower bounds for the regret term of the order $O(\sqrt{T})$ for the case of RKHS. Thus we can say that the order of our bounds by time step is optimal. Multiplicative constants may possibly be improved though.

Acknowledgments

Authors are grateful for useful comments and discussions to Alexey Chernov, Vladimir Vovk, and Alex Gammerman. This work has been supported by EP-SRC grant EP/F002998/1 and ASPIDA grant from the Cyprus Research Promotion Foundation.

References

- Edwin F. Beckenbach and Richard Bellman. *Inequalities*. Springer, Berlin, 1961.
- Glenn W. Brier. Verification of forecasts expressed in terms of probability. *Monthly Weather Review*, 78:1–3, 1950.
- Nicolò Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games*. Cambridge University Press, Cambridge, UK, 2006.
- Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. System Sci.*, 55: 119–139, 1997.
- Alexander Gammerman, Yuri Kalnishkan, and Vladimir Vovk. On-line prediction with kernels and the complexity approximation principle. In *UAI*, pages 170–176, 2004.
- David A. Harville. *Matrix algebra from a statistician’s perspective*. Springer, New York, 1997.
- David Haussler, Jyrki Kivinen, and Manfred K. Warmuth. Sequential prediction of individual sequences under general loss functions. *IEEE Trans. Inform. Theory*, 44:1906–1925, 1998. ISSN 0018-9448.
- Arthur E. Hoerl and Robert W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 42:80–86, 2000.
- Jyrki Kivinen and Manfred K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132:1–63, 1997.
- Jyrki Kivinen and Manfred K. Warmuth. Averaging expert predictions. In *Computational learning theory (Nordkirchen, 1999)*, volume 1572 of *Lecture Notes in Comput. Sci.*, pages 153–167. Springer, Berlin, 1999.
- Jyrki Kivinen and Manfred K. Warmuth. Relative loss bounds for multidimensional regression problems. *Machine Learning*, 45:301–329, 2001. ISSN 0885-6125.
- C Michelot. A finite algorithm for finding the projection of a point onto the canonical simplex of \mathbb{R}^n . *J. Optim. Theory Appl.*, 50:195–200, 1986.
- Bernhard Schölkopf and Alexander J. Smola. *Learning with kernels: Support Vector Machines, regularization, optimization, and beyond*. MIT Press, Cambridge, MA, USA, 2002.
- Vladimir Vovk. Aggregating strategies. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 371–383, San Mateo, CA, 1990. Morgan Kaufmann.

- Vladimir Vovk. A game of prediction with expert advice. *J. Comput. System Sci.*, 56:153–173, 1998.
- Vladimir Vovk. Competitive on-line statistics. *International Statistical Review*, 69:213–248, 2001.
- Vladimir Vovk and Fedor Zhdanov. Prediction with expert advice for the Brier game. In *ICML '08: Proceedings of the 25th International Conference on Machine Learning*, pages 1104–1111, New York, NY, USA, 2008. ACM.

Set/Algorithm	MSE	AMSE	Time
DEC-PKT			
cAAR	0.45906	0.45822	0.578
mAAR	0.45906	0.45822	1.25
mLog	0.46107	0.46265	0.375
mLog Online	0.45751	0.45762	2040.141
Simple	0.58089	0.57883	0
LBL-PKT			
cAAR	0.48147	0.479	0.579
mAAR	0.48147	0.479	1.266
mLog	0.47749	0.47482	0.391
mLog Online	0.47598	0.47398	2403.562
Simple	0.57087	0.5657	0.016
C4			
cAAR	0.64834	0.65447	0.015
mAAR	0.64538	0.65312	0.062
mLog	0.76849	0.77797	0.016
mLog Online	0.68164	0.7351	4.328
Simple	0.69037	0.69813	0.016
C9			
cAAR	0.63238	0.64082	0.015
mAAR	0.63338	0.64055	0.063
mLog	0.97718	0.91654	0.031
mLog Online	0.71178	0.75558	10.625
Simple	0.6509	0.65348	0
E5			
cAAR	0.34452	0.34252	0.078
mAAR	0.34453	0.34252	0.219
mLog	0.31038	0.30737	1.109
mLog Online	0.30646	0.30575	446.578
Simple	0.58212	0.58225	0
E8			
cAAR	0.29395	0.29276	0.078
mAAR	0.29374	0.29223	0.25
mLog	0.31316	0.30382	0.109
mLog Online	0.27982	0.27068	83.125
Simple	0.69691	0.70527	0.016

Table 1: The square losses and prediction time (sec) of different algorithms applied for time series prediction. cAAR and mAAR state for the derived algorithms, mLog states for the logistic regression, mLogOnline states for online logistic regression, and Simple stands for the simple average predictor.