Linear Probability Forecasting

Fedor Zhdanov and Yuri Kalnishkan

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

Abstract. In this paper we consider two online multi-class classification problems: classification with linear models and with kernelized models. The predictions can be thought of as probability distributions. The quality of predictions is measured by the Brier loss function. We suggest two computationally efficient algorithms to work with these problems, the second algorithm is derived by considering a new class of linear prediction models. We prove theoretical guarantees on the cumulative losses of the algorithms. We kernelize one of the algorithms and prove theoretical guarantees on the loss of the kernelized version. We perform experiments and compare our algorithms with logistic regression.

Key words: Online prediction, classification, linear regression, Aggregating Algorithm

1 Introduction

Online prediction provides efficient algorithms which adapt to a predicted sequence "on the fly" [1]. In the online regression framework one assumes the existence of some input at each step; the goal is to predict the outcome on this input. This process is repeated step by step. We consider the multidimensional case where outcomes and predictions come from the probability simplex and can be thought of as probability distributions on the vertices of the simplex. If the outcomes are identified with the vertices of the simplex, this problem can be thought of as the multi-class classification problem of the given input. We prove upper bounds on the cumulative square loss of the learner in comparison with the loss of the best linear function (we say the learner *competes* with these functions).

We develop two algorithms to solve the problem of multidimensional prediction. The first algorithm applies a variant of the AAR (see [2]) to predict each component 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. In order to develop this algorithm we consider an original class of experts which are asymmetrical by the components of their predictions. We are not aware of any other computationally efficient online regression algorithms designed to solve linear and non-linear multi-class classification problems. We come to an unexpected conclusion that the upper bound

on the loss of the component-wise algorithm is better than the upper bound on the loss of the second algorithm asymptotically, but worse in the beginning of the prediction process.

One component of the prediction of the second algorithm has the meaning of a remainder. In practice this situation is quite common. For example, it is popular in financial applications to predict the direction of the price: up, down, or close to the current value. We perform classification experiments with linear algorithms and compare them with logistic regression. The full version of our paper with proofs and detailed explanations can be found in [3].

2 Framework

We are interested in the generalisation of the Brier game from [4]. The set of possible outcomes $\Omega = \mathcal{P}(\Sigma)$ is the set of all probability measures on a finite set Σ with d elements, the prediction set $\Gamma := \{(\gamma_1, \dots, \gamma_d) : \sum_{i=1}^{d} \gamma_i = 1, \gamma_i \in \mathbb{R}\}$ is a hyperplane in d-dimensional space containing all the outcomes. For any $y \in \Omega$ we define the Brier loss

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

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 Protocol 1.

Protocol 1 Protocol of forecasting game

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 $y_t \in \Omega \subseteq \mathbb{R}^d$.
end for

Learner competes with all linear functions (experts) $\xi_t = (\xi_t^1, \dots, \xi_t^d)'$ of x:

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

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 the 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 represented 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 the expert α over T trials and L_T denote the cumulative loss of the learner. The goal of the learner is to ensure that $L_T \leq L_T(\alpha) + R_T$ for all T and α, and some small R_T.

3 Derivation of the algorithms

The component-wise algorithm gives predictions for each component of the outcome separately and then combines them in a special way. To derive this algorithm, we use component-wise experts

$$\xi_t^i = 1/d + \alpha_i' x_t, \quad i = 1, \ldots, d \tag{2}$$

and follow [2]. The algorithm cAAR (component-wise Aggregating Algorithm for Regression) gives its preliminary prediction at the step T

$$\gamma_T^i = \frac{1}{d} + \left(\sum_{t=1}^{T-1} (y_t^i - 1/d) x_t' + \frac{d-2}{2d} x_T' \right) \left(aI + \sum_{t=1}^{T} x_t x_t' \right)^{-1} x_T \tag{3}$$

for $i = 1, \ldots, d$. Then it projects the preliminary prediction onto the probability prediction simplex such that the loss does not increase. This is done by the means of the projection algorithm (Algorithm 1) suggested in [5].

Algorithm 1 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

We apply the Aggregating Algorithm (AA) proposed in [6] to mix experts (1) and make predictions. The resulting algorithm is Algorithm 2. We will further call it the mAAR (the multidimensional Aggregating Algorithm for Regression).

4 Theoretical guarantees

We derive upper bounds for the losses of Algorithm 2 and of the component-wise algorithm predicting in the same framework.

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

Algorithm 2 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 0 is a zero-vector from $\mathbb{R}^n$ is placed at 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 $\sum_{i=1}^{d} (s - r_i)^+ = 2$ in $s \in \mathbb{R}$.
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

Theorem 1. *Let the outcome set in a prediction game be $[A, B], A, B \in \mathbb{R}$. Assume experts' predictions at each step are $\xi_t = C + \alpha' x_t$, where $\alpha \in \mathbb{R}^n$ and $C \in \mathbb{R}$ is the same for all experts. Let $\|x_t\|_\infty \le X$ for all t and take $a > 0$. There exists a prediction algorithm producing predictions γ_t such that for every positive integer T, every sequence of inputs and outcomes of the length T, and any $\alpha \in \mathbb{R}^n$, it achieves*

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

In the multidimensional case the loss of the component-wise algorithm is bounded as in the following theorem.

Theorem 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*

$$L_T \le L_T(\alpha) + da\|\alpha\|_2^2 + \frac{nd}{4} \ln\left(\frac{TX^2}{a} + 1\right). \quad (5)$$

The upper bound for the loss of Algorithm 2 is proven in the following theorem. To prove it, we extend the result of [7] for our game and ensure that the Aggregating Algorithm is applicable here.

Theorem 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* mAAR$(2a)$ *satisfies*

$$L_T(\text{mAAR}(2a)) \le L_T(\alpha) + 2a\|\alpha\|_2^2 + \frac{n(d-1)}{2} \ln\left(\frac{TX^2}{a} + 1\right). \quad (6)$$

The upper bound (6) has worse constant $\frac{n(d-1)}{2}$ in front of the logarithm than $\frac{nd}{4}$ of the bound (5) if $d \geq 3$. In this case (6) is worse asymptotically in T than (5), 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 vectors is larger than the size of the prediction set: $n >> T$.

5 Kernelization

We use the popular in computer learning kernel trick (see, e.g., [8]). Our algorithm competes with the following experts:

$$\begin{aligned} \xi_t^1 &= 1/d + f_1(x_t) \\ &\cdots \\ \xi_t^{d-1} &= 1/d + f_{d-1}(x_t) \\ \xi_t^d &= 1 - \xi^1 - \cdots - \xi^{d-1}. \end{aligned} \tag{7}$$

Here $f_1, \ldots, f_{d-1} \in \mathcal{F}$ are any functions from some RKHS $\mathcal{F}$ (Reproducing Kernel Hilbert Space). We start by rewriting the mAAR in the dual form. Denote

$\widetilde{Y}_i = -2(y_1^i - y_1^d, \ldots, y_{T-1}^i - y_{T-1}^d, -1/2); \overline{Y}_i = -2(y_1^i - y_1^d, \ldots, y_{T-1}^i - y_{T-1}^d, 0);$
$\widetilde{k}(x_T) = (x_1' x_T, \ldots, x_T' x_T)'; \widetilde{K} = (x_s', x_t)_{s,t}$ is the matrix of scalar products

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

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

$$r_i = \left(\widetilde{Y}_1 \cdots \overline{Y}_i \cdots \widetilde{Y}_{d-1}\right) \cdot \left(aI + \widetilde{K}\right)^{-1} \left(\widetilde{k}(x_T)' \cdots 2\widetilde{k}(x_T)' \cdots \widetilde{k}(x_T)'\right)'. \tag{8}$$

To get predictions one can use the same calculations with r_i as in the mAAR. We call this algorithm the mKAAR (K for Kernelized).

5.1 Theoretical guarantee for the kernelized algorithm

The upper bound on the loss of the mKAAR is proven in the following theorem.

Theorem 4. *Assume $\mathbf{X}$ is an arbitrary set of inputs and $\mathcal{F}$ is an RKHS of functions on $\mathbf{X}$ with the given 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 input vectors and outcomes $(x_1, y_1), \ldots, (x_T, y_T)$, mKAAR achieves*

$$L_T(\text{mKAAR}) \leq L_T(f) + a \sum_{i=1}^{d-1} \|f_i\|_{\mathcal{F}}^2 + \frac{1}{2} \ln \det(aI + \widetilde{K}) \tag{9}$$

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

We can represent the bound (9) in another form which is more familiar from the online prediction literature:

Corollary 1. *Under the assumptions of Theorem 4 and if the number of steps T is known in advance, the mKAAR reaches the performance*

$$L_T(\text{mKAAR}) \le L_T(f) + \left(c_{\mathcal{F}}^2 + \sum_{i=1}^{d-1} \|f_i\|_{\mathcal{F}}^2 \right) \sqrt{(d-1)T}, \tag{10}$$

for any $f_1, \dots, f_{d-1} \in \mathcal{F}$.

6 Experiments

We run our algorithms on six real world time-series data sets. DEC-PKT,LBL-PKT-4[1] contain network traffic; C4,C9,E5,E8[2] relate to transportation data.

We use ten previous observations as the input vectors for the tested algorithms at each prediction step. 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 will stay in the 2ϵ tube around the previous value. The measure of the quality of predictions is the mean square loss (MSE) over the last two thirds of each time series (test set). We also introduce another quality measure, which is the average of the MSEs up to each step in the test set (AMSE).

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

$$\gamma_{\text{mLog}}^i = \frac{e^{\theta^i x}}{\sum_{i=1}^3 e^{\theta^i x}}$$

for all the components of the outcome $i = 1, 2, 3$. Here the parameters $\theta^1, \theta^2, \theta^d$ are estimated from the training set (the first third for each series). We apply this algorithm in two regimes. The batch regime, where the algorithm is trained only on the training set and is tested on the test set. The online regime (mLog Online), where at each step new parameters θ are found and only one next outcome is predicted. In both regimes logistic regression does not have theoretical guarantees on the square loss. We also compare our algorithms with the simple predictor (Simple) predicting the average of the ten previous outcomes (it thus always gives probability predictions).

In order to make a prediction, the cAAR requires $O(dn^2)$ operations per step (the most time-consuming operation is the inverse of the matrix, and can be done using Sherman-Morrison formula). The mAAR requires $O((d-1)^2n^2 + d^3)$ operations per step (using Woodbury Formula). Batch logistic regression just gives predictions, which requires $O(n)$ operations per step (we ignore the training time

[1] Data sets can be found http://ita.ee.lbl.gov/html/traces.html.
[2] Data sets can be found http://www.neural-forecasting-competition.com/index.htm.

here). Online logistic regression has to be retrained at each step, and the convergence is not guaranteed. Training algorithms usually use Iteratively Reweighted Least Squares, which in general requires at least $O(dn^3)$ operations per each iteration.

We are not aware of other efficient algorithms for online probability prediction, and thus logistic regression and simple predictor are the only baselines. Component-wise algorithms which could be used for comparison (e.g., Gradient Descent, [9], Ridge Regression, [10]), have to use Algorithm 1 to project their predictions. 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 results are shown in Table 1. As we can see from the table, online methods perform better than the batch method. The online logistic regression performs well, but it is very slow. Our algorithms perform similar to each other and comparable to the online logistic regression, but they are much faster.

Table 1. The square losses and prediction time (sec) of different algorithms applied for time series prediction.

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

7 Conclusions

We present new algorithms which give probability predictions in the online multidimensional 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 perform experiments with the linear algorithms and show that they perform relatively well. We compare them with the logistic regression: the benchmark algorithm giving probability predictions.

Kivinen and Warmuth's work [11] 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 the ones described in [2], 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 Ilia Nouretdinov. This work has been supported in part by EPSRC grant EP/F002998/1 and ASPIDA grant from the CRPF.

References

1. Cesa-Bianchi, N., Lugosi, G.: Prediction, Learning, and Games. Cambridge University Press, Cambridge, UK (2006)
2. Vovk, V.: Competitive on-line statistics. International Statistical Review **69** (2001) 213–248
3. Zhdanov, F., Kalnishkan, Y.: Linear probability forecasting. Technical report, arXiv:1001.0879 [cs.LG], arXiv.org e-Print archive (2009)
4. Brier, G.W.: Verification of forecasts expressed in terms of probability. Monthly Weather Review **78** (1950) 1–3
5. Michelot, C.: 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** (1986) 195–200
6. Vovk, V.: Aggregating strategies. In: Proceedings of COLT'90 (1990) 371–383
7. Vovk, V., Zhdanov, F.: Prediction with expert advice for the Brier game. In: Proceedings of ICML'08 (2008) 1104–1111
8. Gammerman, A., Kalnishkan, Y., Vovk, V.: On-line prediction with kernels and the complexity approximation principle. In: Proceedings of UAI'04 (2004) 170–176
9. Kivinen, J., Warmuth, M.K.: Exponentiated gradient versus gradient descent for linear predictors. Information and Computation **132** (1997) 1–63
10. Hoerl, A.E., Kennard, R.W.: Ridge Regression: biased estimation for nonorthogonal problems. Technometrics **42** (2000) 80–86
11. Kivinen, J., Warmuth, M.K.: Relative loss bounds for multidimensional regression problems. Machine Learning **45** (2001) 301–329