Protected probabilistic regression

Vladimir Vovk

практические выводы
теории вероятностей
могут быть обоснованы
в качестве следствий
гипотез о *предельной*
при данных ограничениях
сложности изучаемых явлений

On-line Compression Modelling Project (New Series)

Working Paper #34

First posted May 18, 2021. Last revised July 31, 2021.

Project web site:
http://alrw.net

Abstract

This paper proposes a procedure for protecting probabilistic regression algorithms against changes in the distribution of the incoming data. It is inspired by the success of the recent conformal test martingales. The procedure combines the probability integral transformation and betting martingales. Simulation and empirical studies give promising results.

Contents

1 Introduction

This paper is inspired by the power of betting martingales used in conformal prediction. It has been repeatedly observed in collaboration with our industrial partners (such as Astra Zeneca and Stena Line) that the distribution of the data usually changes, often drastically, after a prediction algorithm has been trained and a prediction rule is ready to be deployed, with the change in distribution easily detectable by conformal test martingales. This is likely to lead to a significant deterioration in the performance of the prediction rule.

The idea of this paper is to turn successful betting accomplished by a test martingale into improvement in the quality of a prediction rule. In the case of conformal testing, we bet against the hypothesis of exchangeability. In the case of our proposed procedure, we bet against the prediction rule computed from the training set (the *base prediction rule*, as we will call it). It seems plausible that in the absence of exchangeability the prediction rule will work poorly on the test data, and a suitable test martingale (with initial value 1) will attain high values when betting against the prediction rule. This automatically leads to an improved prediction rule.

Formally, the method proposed in this paper is independent of conformal prediction. In particular, it does not depend on the IID assumption (and can be applied, e.g., to time series). However, conformal prediction not only motivates it but also provides required technical tools.

Conformal testing consists of two steps: first we turn an IID sequence of observations into a sequence of p-values distributed uniformly in $[0,1]^\infty$ (thus turning a composite null hypothesis into a simple one), and then we gamble against the p-values being independent and uniformly distributed. A very similar strategy works for betting against the base prediction rule. Lévy's [6, Section 39] probability integral transformation turns a prediction rule for probabilistic regression into a source of IID random variables uniformly distributed in $[0,1]$. This plays the role of the first step of conformal testing. We can then apply the numerous betting martingales developed in conformal testing. Such betting martingales translate into test martingales that gamble successfully against the original prediction rule. However, since a test martingale is essentially the same thing as the likelihood ratio with the original prediction rule in the denominator, a successful test martingale provides us with a new (*protected*) prediction rule (the one in the numerator) that outperforms the original prediction rule.

In designing betting martingales we can apply all arsenal of the tools developed in the area of tracking the best expert in prediction with expert advice, such as switching between different experts, averaging over parameters, sleeping, etc. See, e.g., [9] for a basic review. We will use, however, the most basic methods.

This paper proposes to use testing procedures (namely, betting martingales, such as the Simple Jumper and Mean Jumper) for developing better and more robust prediction algorithms. In this respect it is reminiscent of the method of defensive forecasting [8, Chapter 12], which starts from a test martingale (more generally, a strategy for Sceptic) and then develops a prediction algorithm that

prevents the test martingale (more generally, Sceptic's capital) from growing. An advantage of our current procedure is that in typical cases it is computationally more efficient (in particular, it never requires finding fixed points or solving equations, as in defensive forecasting).

In principle, it is possible that the protected prediction rule will perform worse than the original rule on the actual data. We will introduce the notion of the price of protection, which is the worst possible drop in the performance of the protected prediction rule as compared to the original one. We are particularly interested in procedures with a finite price of protection.

We will start in Section 2 from discussing Lévy's probability integral transformation, which generates IID random variables distributed uniformly in $[0,1]$ (the analogue of p-values in conformal prediction). The topic of Section 3 is online testing of the probability integral transforms for uniformity. Section 4 combines results of the previous two sections for the purpose of protecting prediction algorithms and introduces the notion of the price of protection. A toy simulation study is described in Section 5, but the protection procedure of Section 4 is general and widely applicable; this will be further discussed in Section 7.

2 Probability integral transformation

The key fact that makes conformal testing possible is that conformal prediction outputs p-values that are independent and distributed uniformly in $[0,1]$. Without assuming that the observations are IID, we have a similar phenomenon for the probability integral transformation: if a probability forecasting system outputs probabilistic forecasts with distribution functions $F_1, F_2, \dots$ and $y_1, y_2, \dots$ are the corresponding observations, the values $F_1(y_1), F_2(y_2), \dots$ are independent and distributed uniformly in $[0,1]$. (To avoid complications, let us assume that all F_n are continuous.)

The uniformity of the probability integral transforms was used by Lévy [6, Section 39] as the foundation of his theory of denumerable probabilities (which allowed him to avoid using the then recent axiomatic foundation suggested by Kolmogorov in his *Grundbegriffe* [5]). Modern papers, including [2], usually refer to Rosenblatt [7], who disentangled Lévy's argument from his concern with the foundations of probability; Rosenblatt, however, refers to Lévy's 1937 book [6] in his paper.

The probability integral transformation can be used for testing the underlying probability forecasting system considered as the data-generating distribution. See, e.g., [2, Sections 3.8 and 4.7]. According to [3, Section 3.1], it forms the cornerstone of checking calibration of probability forecasts.

Remark 1. The probability integral transformation can be regarded as a way of normalizing the labels, so that standard martingales become more likely to work. In principle, all the methods proposed in this paper can be applied directly to the labels rather than to their transforms.

Algorithm 1 Simple Jumper betting martingale ($(u_1, u_2, \dots) \mapsto (S_1, S_2, \dots)$)

$C_{-1} := C_0 := C_1 := 1/3$
$C := 1$
for $n = 1, 2, \dots$:
    for $\epsilon \in \{-1, 0, 1\}$: $C_\epsilon := (1 - J)C_\epsilon + (J/3)C$
    for $\epsilon \in \{-1, 0, 1\}$: $C_\epsilon := C_\epsilon b^{(\epsilon)}(u_n)$
    $S_n := C := C_{-1} + C_0 + C_1$

Algorithm 2 Simple Jumper protection ($(F_1, F_2, \dots) \mapsto (F'_1, F'_2, \dots)$)

$C_{-1} := C_0 := C_1 := 1/3$
$C := 1$
for $n = 1, 2, \dots$:
    for $\epsilon \in \{-1, 0, 1\}$: $C_\epsilon := (1 - J)C_\epsilon + (J/3)C$
    $\bar{\epsilon} := (C_1 - C_{-1})/C$
    $F'_n := B^{(\bar{\epsilon})}(F_n)$
    for $\epsilon \in \{-1, 0, 1\}$: $C_\epsilon := C_\epsilon b^{(\epsilon)}(F_n(y_n))$
    $S_n := C := C_{-1} + C_0 + C_1$

3 Testing probability forecasting systems

To turn the probability integral transforms $u_1, u_2, \dots$ into a test martingale we use, as in [10,11,14], the *Simple Jumper* betting martingale given as Algorithm 1, where

$$b^{(\epsilon)}(u) := 1 + \epsilon(u - 0.5). \tag{1}$$

In the next section we set $J := 0.01$, as in [14]. For the intuition behind Simple Jumper, see [14] (and the more complicated Sleepy Jumper is described in detail in [12, Section 7.1]).

A safer option than the Simple Jumper is the Mean Jumper betting martingale [10], which is defined to be the average of Simple Jumpers over a finite set $\mathcal{J}$ of J including $J = 1$ (such as $J \in \mathcal{J} := \{10^{-3}, 10^{-2}, 10^{-1}, 1\}$). The inclusion of $J = 1$ is convenient since the corresponding Simple Jumper is identical 1, and so the Mean Jumper never drops in value below $1/|\mathcal{J}|$.

4 The protection procedure

Given a prediction algorithm A and a betting martingale S, our protection procedure produces the prediction algorithm A' such that S is the likelihood ratio process $\mathrm{d}A'/\mathrm{d}A$.

If the predictive distribution function output by A is $F_n = F_n(y)$, the corresponding predictive density is $f_n = f_n(y) = F'_n(y)$, and the betting function output by S is b_n, the protected predictive density is $b_n(F_n)f_n$. It integrates to 1 since $b_n(F_n)f_n = (B_n(F_n))'$, where B_n is the indefinite integral $B_n(v) := \int_0^v b_n$

of b_n, so that $B'_n = b_n$. We can see that the distribution function for the protected algorithm is $B_n(F_n)$.

The procedure of protection is given as Algorithm 2, where

$$B^{(\epsilon)}(v) := \int_0^v b^{(\epsilon)}(u)\,\mathrm{d}u = \frac{\epsilon}{2}v^2 + \left(1 - \frac{\epsilon}{2}\right)v$$

(cf. (1)). Algorithm 2 uses the fact that the Simple Jumper outputs betting functions (1) for $\epsilon = \bar{\epsilon}$. Let us check this fact. According to Algorithm 1, the value of the martingale at the last step is

$$\begin{aligned}\sum_\epsilon C_\epsilon b^{(\epsilon)}(u) &= \sum_\epsilon C_\epsilon(1 + \epsilon(u - 0.5)) \\ &= C_E(1 + E(u - 0.5)) + C_0 + C_{-E}(1 - E(u - 0.5)) \\ &= (C_E + C_0 + C_{-E}) + (C_E - C_{-E})E(u - 0.5) \\ &\propto 1 + \frac{C_E - C_{-E}}{C_E + C_0 + C_{-E}}E(u - 0.5),\end{aligned}$$

where we assume that the range of ϵ is $\{-E, 0, E\}$. (At this time $E = 1$, but we will use $E = 2$ in Section 5.) We can see that the slope of the resulting straight line is

$$\epsilon := \frac{C_E - C_{-E}}{C_E + C_0 + C_{-E}}E.$$

In this paper we evaluate the performance of the original and protected algorithms using the log loss function; namely the loss of a predictive density f for a realized outcome y is $-\log f(x)$ (two pictures in the experimental Section 5 will use base 10 logarithms). It is a proper loss function, meaning that the optimal expected loss is attained by the true predictive density [4, Section 4].

When S is a Mean Jumper martingale, the protected algorithm is guaranteed not to lose much as compared with the original algorithm when the quality is measured by the log loss function: namely, the cumulative log loss for A' is at most the cumulative log loss for A plus $\log|\mathcal{J}|$. More generally, the *price of protection* for a protection procedure is the supremum over all finite data sequences of the log loss of the protected prediction rule minus the log loss of the base prediction rule. For the Mean Jumper procedure, the price of protection is $\log|\mathcal{J}|$.

5 A simulation study

We consider a dataset that consists of independent Gaussian observations: the first 1000 are generated from $N(0, 1)$, and another 1000 from $N(1, 1)$. Our base prediction algorithm does not know that there is a changepoint at time 1000 and always predicts $N(0, 1)$. The seed of the pseudo random number generator (in NumPy) is always 2021.

First we run the Simple Jumper martingale (Algorithm 1 with $J = 0.01$) on our dataset. The left panel of Figure 1 shows its trajectory; it loses capital before the changepoint, but quickly regains it afterwards. Its final value is 1.100×10^{91}.

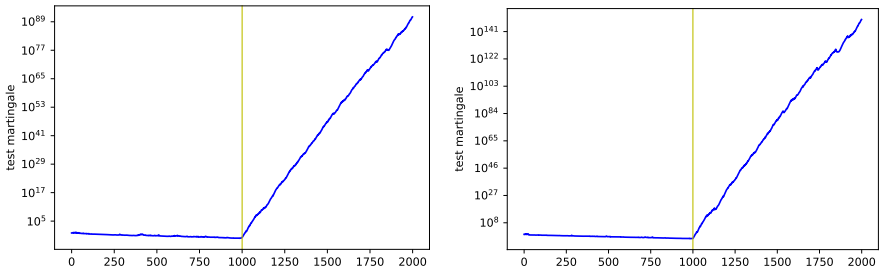

Figure 1: The Simple Jumper test martingale. Left panel: the standard one ($\epsilon \in \{-1, 0, 1\}$). Right panel: $\epsilon \in \{-2, 0, 2\}$.

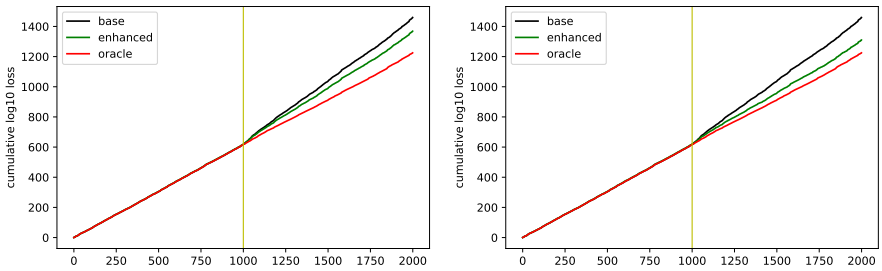

Figure 2: The cumulative log losses of three prediction algorithms (to the left of the changepoint the three lines coincide or are visually indistinguishable). Left panel: $\epsilon \in \{-1, 0, 1\}$. Right panel: $\epsilon \in \{-2, 0, 2\}$.

Remark 2. The possibility of losing so much capital before the changepoint (the value of the Simple Jumper at the changepoint is 0.0114) shows that using the Simple Jumper is risky. If we want to play safe, we can use the Mean Jumper instead of the Simple Jumper. As mentioned above, this will bound our loss to $\log |\mathcal{J}|$ as compared with the original algorithm.

The cumulative log loss of the protected version of the base prediction algorithm is shown as the green line in the left panel of Figure 2. The black line corresponds to the base algorithm, and the red line to the impossible *oracle algorithm*, which knows the truth and predicts with $N(0, 1)$ before the changepoint and $N(1, 1)$ afterwards. According to Figure 1 (left panel), the difference between the final values of the black and green lines is about 91.

To understand better the mechanism of protection in this case, notice that the Simple Jumper outputs betting functions b of the form (1), where $\epsilon \in [-1, 1]$ (usually $\epsilon \notin \{-1, 0, 1\}$). The corresponding predictive distributions $b(F)f$ (where f is the standard normal density and F its distribution function) are shown in the left panel of Figure 3 for five values of ϵ. We can see that our

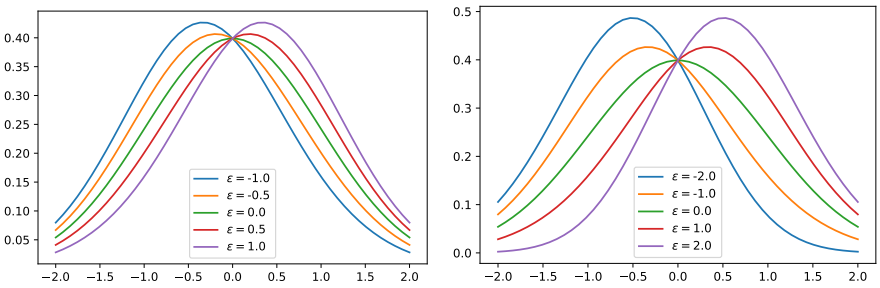

Figure 3: The protected predictive distributions. Left panel: the range of ϵ is $\{-1, -0.5, 0, 0.5, 1\}$. Right panel: $\epsilon \in \{-2, -1, 0, 1, 2\}$.

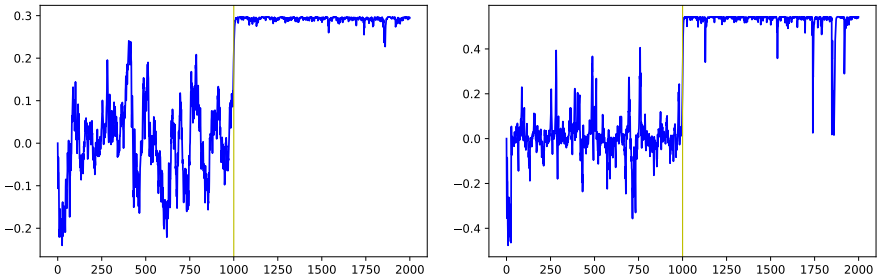

Figure 4: The protected point predictions (medians of the protected predictive distributions). Left panel: $\epsilon \in \{-1, 0, 1\}$. Right panel: $\epsilon \in \{-2, 0, 2\}$.

range of ϵ, $\epsilon \in [-1, 1]$, is not sufficiently ambitious and does not allow us to approximate $N(1, 1)$ well.

Replacing the range $\{-1, 0, 1\}$ for ϵ by $\{-2, 0, 2\}$, we obtain the right panel of Figure 3. The right-most graph in that panel now looks closer to the density of $N(1, 1)$. We cannot extend the range of ϵ further without (1) ceasing to be a calibrator. (Of course, the calibrator does not have to be linear, but let us stick to the simplest choices in this version of the paper.)

Using the range $\{-2, 0, 2\}$ for ϵ leads to the right panels of Figures 1 and 2. We can see that in the right panel of Figure 2 the performance of the protected algorithm is much close to that of the oracle algorithm than in the left panel.

Figure 2 provides useful and precise information, but it is not very intuitive. A cruder approach is to translate the probabilistic forecasts into point predictions. Figure 4 uses the medians of predictive distributions as point predictions. In the case of the base algorithm, the prediction is always 0 (the median of $N(0, 1)$), for the oracle algorithm it is 0 before the changepoint and 1 afterwards, and for the protected algorithm the predictions are shown in the figure. We can see that the right panel of Figure 4 is a better approximation to the

oracle predictions.

Remark 3. To compute the point predictions shown in Figure 4, we can use the representation of the betting function b for the Simple Jumper in the form (1) with

$$\epsilon := \frac{C_E - C_{-E}}{C_E + C_0 + C_{-E}} E,$$

where $\{-E, 0, E\}$ is the range of ϵ (so that $E = 1$ in the left-hand panels and $E = 2$ in the right-hand ones). The indefinite integral of the betting function is

$$B(v) = \int_0^v b(u)\,\mathrm{d}u = \int_0^v (1 + \epsilon(u - 0.5))\,\mathrm{d}u = \left(1 - \frac{\epsilon}{2}\right) v + \frac{\epsilon}{2} v^2 = v - \frac{\epsilon}{2} v(1 - v).$$

Solving the quadratic equation $B(v) = 0.5$ we get

$$v = \frac{\epsilon - 2 + \sqrt{\epsilon^2 + 4}}{2\epsilon}. \tag{2}$$

Since the distribution function of the protected probability forecast is $B(F)$, where $F = N(0, 1)$ is the distribution function of the original probability forecast, we obtain the median of the protected distribution as the v quantile of $N(0, 1)$, with v defined by (2).

Custom-made betting functions

Lemma 1. *Let $F : \mathbb{R} \to \mathbb{R}$ be a smooth bijection, and let g be a pdf on its domain. Then the image of g under the mapping F is the pdf*

$$u \mapsto \frac{g(F^{-1}(u))}{F'(F^{-1}(u))}.$$

In terms of the cdf the statement is much easier: if G is the cdf corresponding to g, so that $G' = g$, the image of G under the mapping F is the cdf

$$u \mapsto G(F^{-1}(u)).$$

We are interested in the case where F is the cdf for $N(0, 1)$ and g is the pdf of $N(1, 1)$. Then, since

$$\frac{g(y)}{f(y)} = \frac{\exp(-(y-1)^2/2)}{\exp(-y^2/2)} = \exp(y - 1/2),$$

the optimal betting function is

$$b(u) = \frac{g(F^{-1}(u))}{f(F^{-1}(u))} = \exp(F^{-1}(u) - 1/2).$$

This optimal betting function is shown in Figure 5 in blue. For comparison, the polynomial calibrators $(d + 1)u^d$ are also shown for $d = 1$ (linear), $d = 2$ (quadratic), and $d = 3$ (cubic).

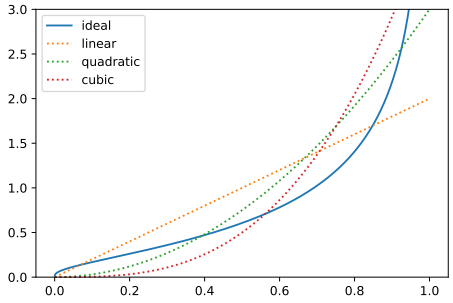

Figure 5: The optimal betting function after the changepoint (in blue).

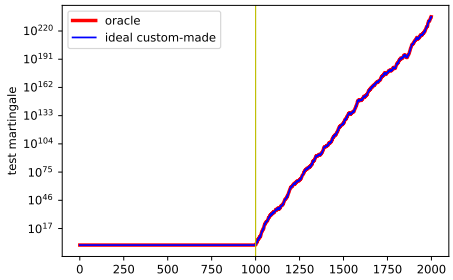

Figure 6: The oracle and ideal custom-made martingales.

Figure 6 shows the oracle and ideal custom-made martingale; their trajectories are visually indistinguishable (unless drawn in different colours and widths, as in the figure) since they are merely different implementations of the same martingale.

The ideal custom-made martingale is utterly unrealistic since it knows the precise data-generating distribution. See Algorithm 3, where R is the parameter ($R := 0.001$ in this paper), S is the total sleeping capital, and A is the total active capital; b is the betting function.

Algorithm 3 Sleeping betting martingale $((u_1, u_2, \dots) \mapsto (S_1, S_2, \dots))$

$S := 1$
$A := 0$
for $n = 1, 2, \dots$:
    $S := (1 - R)S$
    $A := A + RS$
    $A := Ab(u_n)$
    $S_n := S + A$

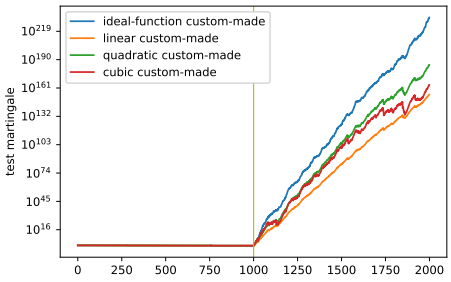

Figure 7: The ideal-function, linear, quadratic, and cubic custom-made martingales.

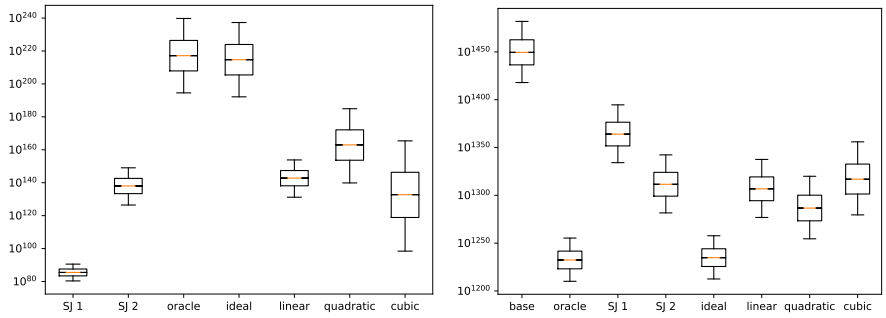

Figure 8: The boxplots corresponding to Figures 1, 2, 6, and 7 over 10^6 trials. Left panel: The final values of the test martingales. Right panel: The cumulative log-loss function.

Figure 7 shows the results of applying Algorithm 3 to four betting functions: the ideal one given in Figure 5 in blue, and the linear, quadratic, and cubic ones (dotted in Figure 5). Of the polynomial functions, the best one is quadratic; linear and cubic are worse (and the higher degrees becomes even worse).

The boxplots corresponding to Figures 1, 2, 6, and 7 are shown in Figure 8. Whereas the previous figures correspond to a fixed seed for the pseudorandom number generator, Figure 1 summarizes the statistics over the first 10^6 seeds.

Figure 9 shows the CRPS losses of the base and protected algorithms. The difference between the two algorithms is more noticeable, since the CRPS loss function has a natural lower bound, 0.

6 Empirical studies

An ideal dataset for empirical studies would have timestamps for all observations, so that we can choose a cut-off point in time for dividing the dataset into

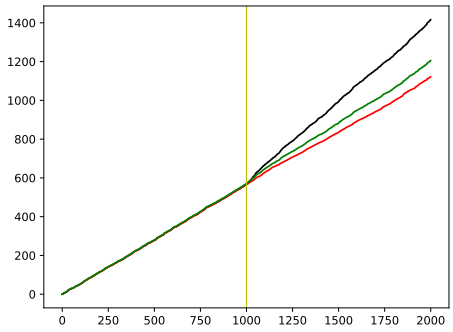

Figure 9: The CRPS losses of the base and protected algorithms for the simulated data.

a training set (before the cut-off) and a test set (after the cut-off). This is what is done in [13, Section 5] for the case of classification.

However, in this version of the paper we only consider the Boston Housing dataset [12, Appendix B.2]. The base prediction algorithm outputs predictions starting from observation number 6, so that it outputs 500 predictions overall. For each new object x_n, $n \geq 7$, we find its 6 nearest neighbours x_i, $i \in \{1, \ldots, n-1\}$, and output as the prediction for the label y_n of x_n the Gaussian distribution $N(\mu, \sigma)$, where μ is the average label y_i for the 6 nearest neighbours and σ is the adjusted empirical standard deviation

$$\sigma := \sqrt{\frac{1}{5} \sum_i (y_i - \mu)^2},$$

the sum being over the 6 nearest neighbours. At each step n we normalize the objects by subtracting from each attribute its mean and then dividing it by its standard deviation (like the standard scaler in scikit-learn and unlike in [12, Appendix B.3]).

The left panel of Figure 10 shows the performance of a Simple Jumper martingale; its final value is 4.504×10^{15}. The right panel shows how it translates into the cumulative log loss. Of course, the protected algorithm performs better (since the martingale gains capital), but the difference does not look impressive. Perhaps this is because of the large amount of noise in the data; as Figure 2 shows, even the huge difference between predicting $N(1, 1)$ correctly and mistaking it for $N(0, 1)$ does not look very significant.

To see the improvement achieved by protection, we can evaluate the quality of the corresponding point predictions. For the base algorithm, the point predictions are given simply by μ, and for the protected algorithm, they are defined as before: see the paragraph containing (2). Table 1 shows that the protected version performs significantly better. Figure 11 visualizes $y - \hat{y}$, where $\hat{y}$ is computed by the base or protected algorithms, but it's a mess. (My general

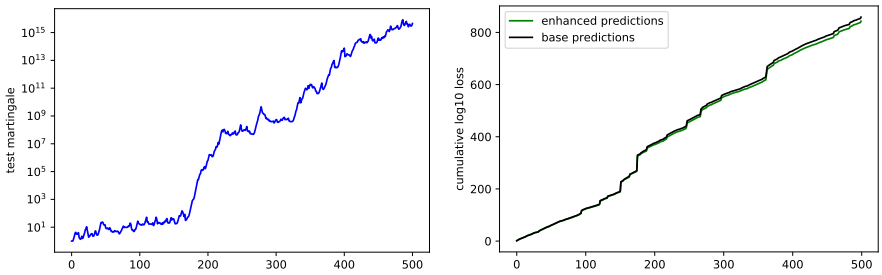

Figure 10: Left panel: The Simple Jumper martingale with parameters $E = 2$ and $J = 0.1$. Right panel: The cumulative log loss of the base and protected algorithms.

Table 1: Errors of the base and protected algorithms

	absolute error	MSE
base algorithm	3.445	5.334
protected algorithm	2.884	4.492

impression is that the protected predictions are better, but it's difficult to be sure.)

The left panel of Figure 12, in which the residuals $y - \hat{y}$ of the base algorithm are sorted, shows that the protection procedure tends to move the residuals towards 0, especially in the case of negative residuals. The right panel shows that this is not just an illusion caused by regression to the mean.

Figure 13 shows the CRPS losses of the base and protected algorithms on the Boston Housing dataset. In numbers, the mean CRPS for the base algorithm is 2.599 and the mean CRPS for the protected algorithm is 2.440.

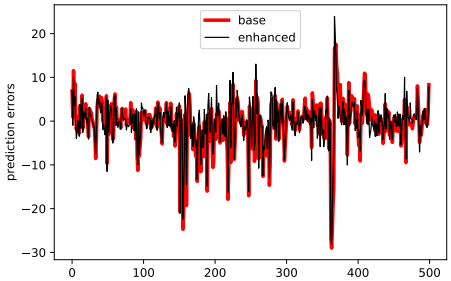

Figure 11: The errors of the base and protected algorithms, as described in text.

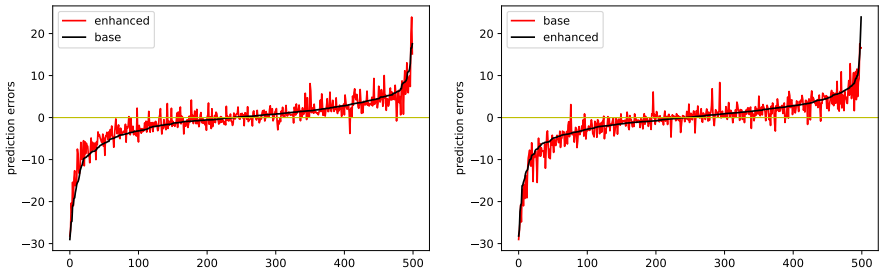

Figure 12: Left panel: The errors of the base and protected algorithms with the former sorted. Right panel: The errors of the base and protected algorithms with the latter sorted.

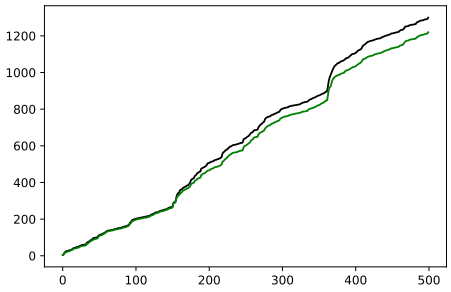

Figure 13: The CRPS losses of the base and protected algorithms.

7 Conclusion

This section briefly discusses possible directions of further research.

To understand better the potential of the new method, further simulation studies and, more importantly, empirical studies are required. In particular, this paper uses only one proper loss function, namely the log loss function. An interesting alternative is CRPS, or continuous ranked probability score [4, Section 4.2].

Another direction is to improve the performance of test martingales and, therefore, protected prediction algorithms in various model situations, similarly to [11]. The framework of Section 5 is an example of such a model situation.

It is important to get rid of the assumption that the predictive distribution is continuous, which we made in Section 2. This is needed, e.g., to cover the case of classification. This could be achieved by adapting the smoothing procedure [12, (2.20)], which is standard in conformal prediction.

This paper assumes that the observations y_n are real numbers, whereas the standard setting of machine learning is where the observations are pairs (x_n, y_n).

Our method is applicable in this case as well if we assume that the x_n are constant. The cleanest approach, however, would be not to assume anything about the x_n and use the game-theoretic foundations of probability [8]. For the game-theoretic treatment of the probability integral transformation, see [2, Theorem 2(b)].

The method used in this paper, Simple Jumper, ignores the x_n, and so it is only about calibration (and not resolution, as described in [1]). We can make our methods depend on x (e.g., use different Simple Jumpers for men and women).

Gneiting et al. [3, Section 3.1] discuss different deviations from perfect calibration. In this paper our base calibrators only gambled against small or large PIT values: see (1). We could also gamble against the PIT values being too categorical (or too timid) by using, instead,

$$b^{(\epsilon)}(u) := \begin{cases} 1 - \epsilon(u - 0.25) & \text{if } u \le 0.5 \\ 1 + \epsilon(u - 0.75) & \text{if } u \ge 0.5. \end{cases}$$

Acknowledgments

Glenn Shafer's and Nell Painter's advice is gratefully appreciated. This work has been supported by Amazon and Stena Line.

References

[1] A. Philip Dawid. Probability forecasting. In Samuel Kotz, Norman L. Johnson, and Campbell B. Read, editors, *Encyclopedia of Statistical Sciences*, volume 7, pages 210–218. Wiley, New York, 1986.

[2] A. Philip Dawid and Vladimir Vovk. Prequential probability: Principles and properties. *Bernoulli*, 5:125–162, 1999.

[3] Tilmann Gneiting, Fadoua Balabdaoui, and Adrian E. Raftery. Probabilistic forecasts, calibration and sharpness. *Journal of the Royal Statistical Society B*, 69:243–268, 2007.

[4] Tilmann Gneiting and Adrian E. Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102:359–378, 2007.

[5] Andrei N. Kolmogorov. *Grundbegriffe der Wahrscheinlichkeitsrechnung*. Springer, Berlin, 1933. English translation: *Foundations of the Theory of Probability*. Chelsea, New York, 1950.

[6] Paul Lévy. *Théorie de l'addition des variables aléatoires*. Gauthier-Villars, Paris, 1937. Second edition: 1954.

[7] Murray Rosenblatt. Remarks on a multivariate transformation. *Annals of Mathematical Statistics*, 23:470–472, 1952.

[8] Glenn Shafer and Vladimir Vovk. *Game-Theoretic Foundations for Probability and Finance*. Wiley, Hoboken, NJ, 2019.

[9] Vladimir Vovk. Derandomizing stochastic prediction strategies. *Machine Learning*, 35:247–282, 1999.

[10] Vladimir Vovk. Testing for concept shift online, On-line Compression Modelling project (New Series), `http://alrw.net`, Working Paper 31, December 2020.

[11] Vladimir Vovk. Conformal testing in a binary model situation, On-line Compression Modelling project (New Series), `http://alrw.net`, Working Paper 33, April 2021.

[12] Vladimir Vovk, Alex Gammerman, and Glenn Shafer. *Algorithmic Learning in a Random World*. Springer, New York, 2005.

[13] Vladimir Vovk, Ivan Petej, and Alex Gammerman. Protected probabilistic classification, On-line Compression Modelling project (New Series), `http://alrw.net`, Working Paper 35, July 2021.

[14] Vladimir Vovk, Ivan Petej, Ilia Nouretdinov, Ernst Ahlberg, Lars Carlsson, and Alex Gammerman. Retrain or not retrain: Conformal test martingales for change-point detection, On-line Compression Modelling project (New Series), `http://alrw.net`, Working Paper 32, February 2021.