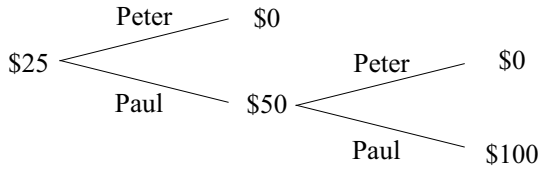
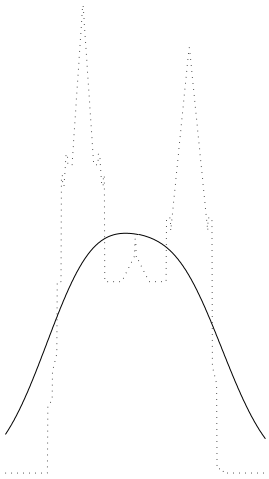
Experiments with the K29 algorithm

Vladimir Vovk





The Game-Theoretic Probability and Finance Project

Working Paper #9 (draft)

October 3, 2004

Project web site:
http://www.probabilityandfinance.com

Abstract

The K29 algorithm for probability forecasting (proposed in [6]) is studied empirically on a popular benchmark data set.

Contents

1 Algorithm

We are interested in the following game of probability forecasting between two players, Forecaster and Reality:

FOR $n = 1, 2, \ldots$:
Reality announces $x_n \in \mathbf{X}$.
Forecaster announces $p_n \in [0, 1]$.
Reality announces $y_n \in \{0, 1\}$.

On each round, Forecaster's goal is to predict Reality's move y_n chosen from the *label space*, always taken to be $\{0, 1\}$ in this paper. His move, the *probability forecast* p_n, is interpreted as the probability he attaches to the event $y_n = 1$. At the beginning of each round, Reality releases some information to Forecaster; this piece of information, the *object* x_n, is chosen from an *object space* $\mathbf{X}$.

The K29 algorithm, proposed in [6], is the following strategy for Forecaster. Let $K : ([0, 1] \times \mathbf{X})^2 \to \mathbb{R}$ be a Mercer kernel. After seeing the object x_n on round n Forecaster outputs an arbitrary root $p = p_n$ of the equation $S_n(p) = 0$, where

$$S_n(p) = \sum_{i=1}^{n-1} K((p, x_n), (p_i, x_i))(y_i - p_i);$$

if this equation has no roots (in which case S_n never changes sign),

$$p_n := (\operatorname{sign}(S_n) + 1)/2. \tag{1}$$

A natural way to build a Mercer kernel on $([0, 1] \times \mathbf{X})^2$ from Mercer kernels on $[0, 1]^2$ and $\mathbf{X}^2$ is to use the operation of tensor product (see, e.g., [5, 4]). The kernel on $[0, 1]^2$ arrived at in [6] was

$$K(p, p_i) = \exp\left(-\frac{(p - p_i)^2}{4\sigma^2}\right); \tag{2}$$

it is known in machine learning as the Gaussian kernel (in more familiar parameterizations, however, $4\sigma^2$ is replaced by $2\sigma^2$ or c). As kernels on $\mathbf{X}^2$ we will take the polynomial kernels

$$K(x_n, x_i) = (x_n \cdot x_i)^d, \tag{3}$$

where d is a positive integer.

2 Experimental results

In this draft we report results of experiments with the data set known as The Insurance Benchmark (TIC), or CoIL 2000, data set; it is available from the UCI KDD repository [3]. This data set consist of 5822 training examples and 4000 test examples; we merged these into one data sequence of 9822 examples,

Degree	1	2	3	4	5
Overall MSE	0.10885	0.10840	0.10850	0.10862	0.10902
Training MSE	0.10968	0.10915	0.10943	0.10976	0.11041
Test MSE	0.10763	0.10731	0.10713	0.10697	0.10699
Degree	6	7	8	9	10
Overall MSE	0.10965	0.11055	0.11173	0.11331	0.11530
Training MSE	0.11131	0.11248	0.11389	0.11561	0.11790
Test MSE	0.10725	0.10773	0.10859	0.10996	0.11153

Table 1: MSE for K29 on the TIC data set

first training and then test examples. Each example consists of 85 attributes and a binary label. Table 1 shows the performance of the K29 algorithm on the (merged) TIC data set for K the tensor product

$$K((p, x_n), (p_i, x_i)) := K(p, p_i)K(x_n, x_i)$$

of (2) and (3). We used the value 0.1 for the parameter σ in (2) (as can be seen from the derivation in [6], σ can be interpreted as the accuracy in the estimation of $p_n \in [0, 1]$ we are aiming for, so $\sigma = 0.1$ is a natural first guess) and we used the values given in the first row of Table 1 for the parameter d (which we call the *degree* of the polynomial kernel) in (3). Since the attributes of the TIC data set are discrete and often nominal (i.e., their numerical values serve only as labels and do not measure anything), each attribute taking, say, m values was replaced by m binary attributes all of which but one were zero; in other words, we used, instead of (3),

$$K(x_n, x_i) = \left(\sum_{j=1}^{85} \mathbb{I}_{x_{n,j}=x_{i,j}} \right)^d ,$$

where $\mathbb{I}_E$ is defined as 1 if E holds and 0 otherwise. For each d the second row of Table 1 gives the value of the *mean squared error* (MSE, also known as the Brier score [1]) $\frac{2}{N} \sum_{n=1}^{N} (y_n - p_n)^2$, where $N = 9822$ is the size of the data set; the third and fourth rows give the analogous doubled averages over the training examples and the test examples, respectively.

To compare K29 to other algorithms, in Table 2 we reproduce some results of [8] (their Table 2). The results of [8] use the same attributes as the winning entry of the CoIL 2000 challenge (which might give some bias, since there were quite a few, 43, submitted entries; see [2]). "SVM" is the linear kernel SVM with $C = 1$, as implemented in the SvmFu package. "Sigmoid" and "PAV" are two calibration methods.

Additional results are given in [7] (Table 4), whose best MSE on the test set is 0.10742 (for the binned naive Bayes with a suitably chosen bin width); other interesting figures in that table are: the MSE over the test set is 0.11900 for the trivial "all zero" algorithm and 0.11192 for the almost as trivial "all base rate" algorithm.

Method	Training MSE	Test MSE
NB	0.12845	0.13551
Sigmoid NB	0.10536	0.10905
PAV naive Bayes	0.10315	0.10818
SVM	0.11942	0.11889
Sigmoid SVM	0.11080	0.11122
PAV SVM	0.10974	0.11200

Table 2: MSE for various off-line algorithms on the TIC data set (the last column is important)

Degree	1	2	3	4	5
Test MSE	0.11033	0.10884	0.10752	0.10688	0.10668
Degree	6	7	8	9	10
Test MSE	0.10680	0.10715	0.10775	0.10862	0.10992

Table 3: MSE over the test set for K29 run in the off-line fashion on the TIC data set

The results of Table 1 (even the fourth row) and Table 2 (and similar results in [7]) are not directly comparable, since the K29 algorithm continues learning even after reaching the test examples. Table 3 gives the MSE over the test set for the "off-line K29" algorithm: the prediction p_n for each test example (x_n, y_n) is computed as the 5823th prediction by the K29 algorithm fed with the 5822 training examples and then with x_n. Now the results in Table 2 (the "Test MSE" column) and Table 3 can be compared; K29 performs better than the other algorithms for a range of d (surprisingly, and perhaps accidentally, the test MSE improves when K29 is run off-line rather than on-line).

3 Is using own past forecasts useful?

A strange feature of the K29 algorithm is that when computing p_n, Forecaster uses his own forecasts p_i, $i < n$. One might suspect that Forecaster needs his own past forecasts to cover up his confusion by making the sequence $p_1, \dots, p_n$ look consistent; otherwise, knowing Reality's moves would have been sufficient.

One explanation is that K29 was derived with no assumptions whatsoever about how Reality chooses her moves; in particular, it is not assumed that she is oblivious (i.e., does not pay attention to Forecaster's moves). If Reality is not oblivious, the dependence of p_n on p_i, $i < n$, is not surprising.

However, in the usual forecasting problems (such as the experiments with the TIC data set described in the previous section) Reality is oblivious, and so one might expect that eliminating dependence on p_i in the K29 algorithm will not impair its performance. In view of the interpretation of kernels as measuring similarity in the feature space ([4], §1.1), the K29 choice of p_n can

Degree	1	2	3	4	5
Overall MSE	0.11204	0.11179	0.11155	0.11131	0.11105
Training MSE	0.11228	0.11203	0.11179	0.11154	0.11128
Test MSE	0.11169	0.11145	0.11121	0.11096	0.11071
Degree	6	7	8	9	10
Overall MSE	0.11078	0.11051	0.11024	0.11002	0.10990
Training MSE	0.11102	0.11076	0.11054	0.11040	0.11043
Test MSE	0.11043	0.11013	0.10981	0.10946	0.10912
Degree	11	12	13	14	15
Overall MSE	0.10995	0.11032	0.11108	0.11232	0.11405
Training MSE	0.11075	0.11150	0.11281	0.11475	0.11727
Test MSE	0.10879	0.10859	0.10856	0.10880	0.10937

Table 4: MSE for the pure object K29 algorithm (run on-line) on the TIC data set

be interpreted as choosing the average of y_i over observed instances close to the current instance. In the K29 algorithm, "instance" is understood as the pair (object,probability) and for the current example we use a postulated probability. If we ignore the probability part of the instances and replace p_i with p, we obtain the following modification of the K29 algorithm, which we will call the *pure object K29* algorithm. Let $K : \mathbf{X}^2 \to \mathbb{R}$ be a Mercer kernel. After seeing the object x_n on round n Forecaster outputs any root $p = p_n$ of the equation $S_n(p) = 0$, where

$$S_n(p) = \sum_{i=1}^{n-1} K(x_n, x_i)(y_i - p); \tag{4}$$

if there is no root, define p_n by (1). Table 4 shows, however, that the pure object K29 algorithm does nor work as well as the original K29 algorithm.

A less radical change to the K29 algorithm is to use $(y_i - p_i)$ instead of $(y_i - p)$ in (4); the results for this variant are shown in Table 5. It appears that using own past forecasts is useful even with an oblivious Reality.

Acknowledgments

I am grateful to David Lindsay, who reproduced some of the experiments described in this paper. This work was partially supported by BBSRC (grant 111/BIO14428), EPSRC (grant GR/R46670/01), MRC (grant S505/65), and European Commission (grant IST-1999-10226).

References

[1] Glenn W. Brier. Verification of forecasts expressed in terms of probability. *Monthly Weather Review*, 78:1–3, 1950.

Degree	1	2	3	4	5
Overall MSE	0.11204	0.11179	0.11154	0.11129	0.11103
Training MSE	0.11227	0.11203	0.11178	0.11153	0.11127
Test MSE	0.11169	0.11145	0.11120	0.11095	0.11069
Degree	6	7	8	9	10
Overall MSE	0.11075	0.11047	0.11019	0.11053	0.11052
Training MSE	0.11099	0.11073	0.11049	0.11130	0.11153
Test MSE	0.11040	0.11010	0.10976	0.10941	0.10905
Degree	11	12	13	14	15
Overall MSE	0.11058	0.11107	0.11217	0.11371	0.11552
Training MSE	0.11182	0.11275	0.11425	0.11610	0.11842
Test MSE	0.10877	0.10863	0.10915	0.11025	0.11131

Table 5: MSE for the variant K29 algorithm (run on-line) on the TIC data set

[2] Charles Elkan. Magical thinking in data mining: Lessons from CoIL challenge 2000. In *Proceedings of the Seventh International Conference on Knowledge Discovery and Data Mining (KDD'2001)*, pages 426–431. ACM Press, 2001.

[3] Seth Hettich and Steven D. Bay. *The UCI KDD Archive*. University of California, Department of Information and Computer Science, Irvine, CA, 1999, `http://kdd.ics.uci.edu`.

[4] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

[5] Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

[6] Vladimir Vovk, Akimichi Takemura, and Glenn Shafer. Defensive forecasting, The Game-Theoretic Probability and Finance project, `http://probabilityandfinance.com`, Working Paper #8, September 2004.

[7] Bianca Zadrozny and Charles Elkan. Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers. In Carla E. Brodley and Andrea P. Danyluk, editors, *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 609–616, San Francisco, CA, 2001. Morgan Kaufmann.

[8] Bianca Zadrozny and Charles Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 694–699. ACM Press, 2002.