Specialist Experts for Prediction with Side Information

Yuri Kalnishkan*†, Dmitry Adamskiy*†, Alexey Chernov‡†, and Tim Scarfe*†
*Department of Computer Science,
Royal Holloway, University of London, Egham, TW20 0EX, United Kingdom
‡School of Computing, Engineering and Mathematics,
University of Brighton, Brighton, BN2 4GJ, United Kingdom
† Computer Learning Research Centre,
Royal Holloway, University of London, Egham, TW20 0EX, United Kingdom
Email: yura@cs.rhul.ac.uk, Dmitry.Adamskiy@rhul.ac.uk, chernov@cs.rhul.ac.uk, tim@developer-x.com

***Abstract*—The paper proposes the vicinities merging algorithm for prediction with side information. The algorithm is based on specialist experts techniques. We use vicinities in the side information domain to identify relevant past examples, apply standard learning techniques to them, and then use prediction with expert advice tools to merge those predictions. Guarantees from the theory of prediction with expert advice ensure that helpful vicinities are selected dynamically. The algorithm automatically converges on the right vicinities from an initial broad selection. We apply the resulting algorithms to two problems, prediction of implied volatility of options and prediction of students' performance at tests. On the problem of predicting implied volatility, the algorithm consistently outperforms naive competitors and a highly-tuned proprietary method used in the industry. When applied to the students' performance, the algorithm never falls behind the baseline and outperforms it when the side information is beneficial.**

I. Introduction

We consider the on-line protocol where at each trial $t = 1, 2, \ldots$ a learner observes a side information vector (*signal*) x_t and attempts to predict an outcome y_t, which is shown to the learner later. The performance of the learner is measured by means of the cumulative loss. For related research on this framework see [1], [2], and Section 11 of the monograph [3].

The on-line learning protocol is different from the more traditional batch learning framework where a whole training set of examples (x_i, y_i) is given to the learner at once. The on-line learning protocol applies in the situations when the information is revealed to us gradually and we need to take actions on the basis of what we have seen so far. As time passes, the structure of the data may evolve. On-line learning requires approaches and algorithms often distinctly different from those in batch learning. However, on-line learning is often neglected by machine learning practitioners who interpret on-line data analysis tasks in a batch paradigm thus overlooking important temporal features of the data.

The on-line learning protocol is somewhat akin to the time series framework. What makes on-line learning different is that it does not necessarily assume a statistical model governing the behaviour of outcomes as the theory of time series does. The relation between on-line learning and time series is similar to the relation between machine learning and statistics in general – while learning may use statistical models, it is not restricted to them.

In the on-line learning framework the learner is faced with the problem of identifying relevant past examples. Suppose that at trial T a signal x_T has been received. What are the relevant past examples (x_t, y_t)? Which examples should the learner use for training? One natural answer is to look at most recent examples. However, the signals x_t in the recent examples may be quite different from the current x_T. Should the learner go back further to fetch examples that are older but have signals x_t closer to the current x_T? Doing so would run the risk of relying on obsolete information.

In this paper we develop a method based on prediction with expert advice with specialist experts and carry out its empirical study. The theory of prediction with expert advice (see monograph [3] for an overview) is concerned with an optimal merging of predictions of strategies called experts. A specialist expert (the concept was introduced in [4]) is a prediction strategy that may refrain from making a prediction (*sleep*) at certain trials. We define specialist experts that monitor vicinities of signals x_t and become activated when signals from their respective vicinities occur. They make predictions based on the history of examples with signals from their vicinities. We then use methods of prediction with expert advice to merge the predictions of the specialist experts. Theoretical guarantees ensure that the cumulative loss of the resulting strategy is not much worse than the loss of every specialist expert at the trials when it is awake. Methods of prediction with expert advice effectively allow us to achieve a trade-off between going back in time and further away in space.

We apply this idea to two particular problems, predicting implied volatility of options and predicting students' performance in tests. The results are then analysed.

Although the problem of prediction with expert advice has been extensively studied for the past twenty years, practical applications of the theory are few and far between. In [5] the methods of prediction with expert advice are applied to aggregate predictions of sports results calculated from the odds quoted by bookmakers. The numbers of bookmakers were very small, four in one empirical study and eight in another. In this paper we deal with hundreds of experts. In [6], which is most similar in approach to our paper, specialist experts are used to predict electricity consumption. However, the methods of aggregation in [6] are based on the exponential weighted average algorithm, which is known to be theoretically suboptimal (see Section 3.3 in [3]) for many loss functions including the square loss used in [6].

In this paper we use methods from [7], [8] based on the aggregating algorithm introduced in [9], [10]. We hope the empirical studies of this paper will provide useful intuitions for the theory of prediction with expert advice.

The structure of this paper is as follows. Section II describes the methods of prediction with expert advice for specialist experts. Section III describes the main algorithm proposed by this paper – *vicinities merging* – in a generalised setting. Section IV describes its application to predicting implied volatility of options and Section V its application to predicting students' results at tests.

II. Preliminaries

A. Prediction Framework

This paper is concerned with prediction in the following framework. Let outcomes $\omega_1, \omega_2, \dots$ from an *outcome space* Ω occur successively in discrete time. A *learner* or *prediction algorithm* tries to predict each outcome and outputs a prediction γ_t from a *prediction space* Γ on the basis of a *signal* x_t from a *domain* X each time before it sees the outcome ω_t. The quality of predictions is assessed by means of a *loss function* $\lambda : \Gamma \times \Omega \to [0, +\infty]$. The framework is summarised in Protocol 1.

Protocol 1.

```
FOR $t = 1, 2, \dots$
    nature announces $x_t \in X$
    learner outputs $\gamma_t \in \Gamma$
    nature announces $\omega_t \in \Omega$
    learner suffers loss $\lambda(\gamma_t, \omega_t)$
ENDFOR
```

Over T trials the learner suffers the cumulative loss $\mathrm{Loss}_T = \sum_{t=1}^{T} \lambda(\gamma_t, \omega_t)$. We denote the loss of the learner by Loss_T or by $\mathrm{Loss}_T(\mathrm{Algorithm})$ with Algorithm being the notation for the learner or the algorithm the learner uses.

In this paper we are interested in two cases. In the first case the prediction and outcome spaces are a real interval $\Omega = \Gamma = [A, B]$ and the square loss function $\lambda_{\mathrm{sq}}(\gamma, \omega) = (\gamma - \omega)^2$ is used. In the second case the outcomes are bits from $\Omega = \{0, 1\}$, the predictions are taken from the interval $\Gamma = [0, 1]$ and the logarithmic loss given by $\lambda_{\log}(\gamma, 0) = -\log(1 - \gamma)$ and $\lambda_{\log}(\gamma, 1) = -\log \gamma$ is used.

B. Prediction with Expert Advice

The problem of prediction with expert advice can be summarised as follows. Suppose that there is a pool Θ of *experts*. Throughout this paper we assume that Θ is finite. The experts try to predict the outcomes from the same sequence, and their predictions $\gamma_t(\theta)$ are made available to the learner before it outputs its own according to Protocol 2. The goal of the learner is to suffer total loss not much greater than the loss of the best expert in the pool.

Protocol 2.

```
FOR $t = 1, 2, \dots$
    experts $\theta \in \Theta$ announce predictions $\gamma_t(\theta) \in \Gamma$
    learner outputs $\gamma_t \in \Gamma$
    nature announces $\omega_t \in \Omega$
    each expert $\theta \in \Theta$ suffers loss $\lambda(\gamma_t(\theta), \omega_t)$
    learner suffers loss $\lambda(\gamma_t, \omega_t)$
ENDFOR
```

Over T trials each expert θ suffers the cumulative loss $\mathrm{Loss}_T(\theta) = \sum_{t=1}^{T} \lambda(\gamma_t(\theta), \omega_t)$ and the learner suffers the cumulative loss $\mathrm{Loss}_T = \sum_{t=1}^{T} \lambda(\gamma_t, \omega_t)$. The aspiration is that an inequality of the form $\mathrm{Loss}_T \lesssim \mathrm{Loss}_T(\theta)$ holds for all $T = 1, 2, \dots$ and $\theta \in \Theta$.

The theory of prediction with expert advice does not impose any restrictions on the law generating outcomes ω_t or on the internal working of experts. Guarantees provided by the theory usually hold for all sequences of outcomes ω_t and all experts' predictions. One may think of 'nature' and 'experts' as mere names for the slots in the protocol rather than computational agents with particular properties.

C. Aggregating Algorithm

The aggregating algorithm (see [1], [10] for a detailed overview) generalises Bayesian mixtures of probabilistic hypotheses (see, e.g., Section 2 of [11]); it is identical to the Bayesian mixture for the logarithmic loss. However, it is more general in that it admits many other loss functions such as the square loss function.

AA takes the following parameters: a learning rate $\eta \in (0, +\infty)$ and an initial distribution over the set of experts Θ; a distribution can be represented by an array of initial weights $p_0(\theta)$, where $\theta \in \Theta$.

The algorithm maintains an array of weights $w_t(\theta), \theta \in \Theta$. Their initial values are $w_0(\theta) = p_0(\theta), \theta \in \Theta$, and they are updated according to the rule

$$w_t(\theta) = w_{t-1}(\theta) e^{-\eta \lambda(\gamma_t(\theta), \omega_t)} = p_0(\theta) e^{-\eta \, \mathrm{Loss}_t(\theta)} \quad .$$

At trial t upon observing the experts' predictions $\gamma_t(\theta)$ the learner outputs a prediction γ_t that for every possible $\omega \in \Omega$ satisfies the condition

$$\lambda(\gamma_t, \omega) \leq c(\eta) g_t(\omega) \quad , \tag{1}$$

where

$$g_t(\omega) = -\frac{1}{\eta}\ln\frac{1}{\sum_{\theta\in\Theta} w_{t-1}(\theta)}\sum_{\theta\in\Theta} w_{t-1}(\theta)e^{-\eta\lambda(\gamma_t(\theta),\omega)} \tag{2}$$

and $c(\eta)$ is a constant determined by the loss function λ. The constant is defined in such a way that γ_t can always be found. One can show (see Lemma 1 from [1]) that

$$\mathrm{Loss}_T(AA) = \sum_{t=1}^{T}\lambda(\gamma_t,\omega_t) \leq c(\eta)\,\mathrm{Loss}_T(\theta) + \frac{c(\eta)}{\eta}\ln 1/p_0(\theta) \ . \tag{3}$$

The aggregating algorithm thus performs nearly as well as the best expert loss-wise. It was proven in [10] that the upper bounds on the loss of the aggregating algorithm are optimal.

If the prediction and outcome spaces are an interval $\Omega = \Gamma = [A, B]$ and the square loss function is $\lambda_{\mathrm{sq}}(\gamma,\omega) = (\gamma-\omega)^2$, then for η satisfying $0 < \eta \leq \frac{2}{(B-A)^2}$ we have $c(\eta) = 1$ (see [1], [12]) and therefore the optimal value is $\eta = \frac{2}{(B-A)^2}$. For these values of η we can use a simple *substitution function*

$$\gamma_t = \frac{A+B}{2} - \frac{g_t(B) - g_t(A)}{2(B-A)}$$

mapping g_t to a γ_t satisfying (1).

If the outcome space is $\{0, 1\}$, prediction space is $\Gamma = [0, 1]$, and the loss function is logarithmic, then $c(\eta) = 1$ for $\eta \in (0, 1]$ and for the optimal $\eta = 1$ we can take

$$\gamma_t = \frac{1}{\sum_{\theta\in\Theta} w_{t-1}(\theta)}\sum_{\theta\in\Theta} w_{t-1}(\theta)\gamma_t(\theta) \ ;$$

this is precisely the Bayesian prediction.

D. Specialist Experts

Suppose that an expert in the prediction with expert advice framework can abstain from making a prediction at trial t. If it does so, we say that it *sleeps* at trial t. One may want to obtain an equivalent of bound (3) ensuring that the learner competes well with every expert θ at the trials where θ is awake.

The concept of a specialist expert was proposed in [4]. In this paper we will be discussing specialist experts as a special case of the theory of expert evaluators after [7], [8]. A simple extension of the AA may be used for specialist experts.

If an expert θ sleeps at trial t, let us assume that it suffers notional loss $\lambda(\gamma_t, \omega_t)/c(\eta)$ (if $c(\eta) = 1$ we can simply say that it 'goes with the crowd' and subscribes to yet unknown γ_t whatever it may be) and apply the AA. The weight of a sleeping expert is updated according to this notional loss. If Θ_{sleep} is the set of experts sleeping at trial t and Θ_{awake} is the set of experts that are awake (not sleeping) at trial t, then (1) becomes

$$e^{-\eta\lambda(\gamma_t,\omega)/c(\eta)}\sum_{\theta\in\Theta} w_{t-1}(\theta) \geq \sum_{\theta\in\Theta_{\mathrm{awake}}} w_{t-1}(\theta)e^{-\eta\lambda(\gamma_t(\theta),\omega)} + \sum_{\theta\in\Theta_{\mathrm{sleep}}} w_{t-1}(\theta)e^{-\eta\lambda(\gamma_t,\omega)/c(\eta)} \ .$$

Clearly, all terms corresponding to sleeping experts cancel out and we get

$$\lambda(\gamma_t,\omega) \leq \frac{c(\eta)}{\eta}\ln\frac{\sum_{\theta\in\Theta_{\mathrm{awake}}} w_{t-1}(\theta)e^{-\eta\lambda(\gamma_t(\theta),\omega)}}{\sum_{\theta\in\Theta_{\mathrm{awake}}} w_{t-1}(\theta)},$$

i.e., the formula is identical to that from the aggregating algorithm except that the sum is taken over the experts that are awake. Note that we still need to update the weights of sleeping experts so that they are assigned the correct weights upon waking. We will call this algorithm *aggregating algorithm with sleeping experts (AAS).*

Arguing in the same way as for the AA, we get a bound similar to (3). By dropping equal terms in the losses on the left- and right-hand sides we obtain

$$\mathrm{Loss}_T^{(\theta)}(AAS) \leq c(\eta)\,\mathrm{Loss}_T^{(\theta)}(\theta) + \frac{c(\eta)}{\eta}\ln 1/p_0(\theta) \ , \tag{4}$$

where the sum in $\mathrm{Loss}^{(\theta)}$ is taken only over trials when expert θ was awake.

III. Algorithms

In this section we describe an algorithm for a predictor working in the environment of Protocol 1.

Consider a finite subset $\{U_1, U_2, \ldots, U_K\} \subseteq 2^X$ such that $\cup_{k=1}^{K} U_k = X$. We will call sets U_k vicinities because it is natural to choose them in such a way that elements of U_i are in some respect akin to each other. Each vicinity U_i generates a specialist expert that predicts as follows. The expert is awake only at trials t where $x_t \in U_i$. It maintains the sequence $(x_{t_1}, y_{t_1}), (x_{t_2}, y_{t_2}), \ldots$ of outcomes for such trials and uses the series to make predictions for trials t where $x_t \in U_i$. For t such that $x_t \notin U_i$ the expert makes no predictions. The experts are then merged using the aggregating algorithm for specialist experts. We will call this the *vicinities merging method.*

We assume that predicting outcomes in the sequence of examples $(x_{t_1}, y_{t_1}), (x_{t_2}, y_{t_2}), \ldots$ is straightforward, at least for some convenient vicinities U_k. In the simplest case, the elements $y_{t_1}, y_{t_2}, \ldots$ form a predictable time series and the signals x_{t_k} can be ignored. This happens in our first application of predicting implied volatility. In more difficult cases signals x_{t_k} are necessary, but the dependency of ys on xs is more straightforward on U_k and can be learned with better precision. The aggregating algorithm with sleeping

Table I
DATASETS SUMMARY

Dataset name	Underlying asset	Maturity	Number of transactions
eeru1206	EERU shares	Dec 2006	13,152
gzp307	Gazprom shares	Mar 2007	10,985
rtsse307	RTSSE index	Mar 2007	8,410

experts effectively identifies "useful" vicinities or, in other terms, it automatically finds sets of relevant past examples.

The general sleeping experts bound (4) provides useful insights into the performance of the merging method. The regret term in (4) is proportional to $\ln(1/p_0(\theta))$ and if we use uniform initial weights it grows logarithmically with the number of vicinities. Thus the user need not be too concerned about the choice of vicinities and can use a generously representative selection. The method will automatically converge on the right ones.

The computational complexity of the merging method depends on the underlying prediction algorithms working on the vicinities and the number of vicinities. The overhead brought about by merging depends on which loss function is employed. However, for most natural loss functions (including the square and logarithmic loss) the time taken by merging is linear in the number of vicinities.

IV. RTSSE IMPLIED VOLATILITY PREDICTION

A. Problem Setup

The datasets provided to us by the Russian Trading System Stock Exchange (RTSSE) contain logs of consecutive option transactions. Each dataset includes transactions with an option maturing at a particular date on a particular underlying asset; see Table I for details. The datasets date back to mid-2000s, when the market at RTSSE was experiencing steady unperturbed growth. The underlying assets we consider are the shares of EERU (a utility company), the shares of Gazprom (a major gas exporter), and the RTSSE index. These datasets were previously studied in [13].

We consider each transaction as a signal x_t and an outcome y_t. The signal x_t consists of the strike price X of the option used in the transaction, the price of the underlying asset S, time left to the maturity of the option T, and a bit showing whether the option is a put or a call; the outcome y_t is the *implied volatility* calculated for the transaction using the Black-Scholes formulas. (Note that the option price is not one of our attributes; otherwise the problem reduces to the trivial task of learning the Black-Scholes formula.)

Definitions and further details on options and implied volatility are available from finance textbooks such as [14], [15]. There is no unique and generally accepted theory explaining the evolution of implied volatility. However, it is a meaningful financial parameter. It is effectively equivalent to the option price and often used instead of the price in quotes.

In this paper we approach the problem of finding the implied volatility from a purely machine learning perspective. The quality of the prediction is measured by the squared deviation of the predicted implied volatility from the true implied volatility.

In this paper we analyse three datasets each containing a log of transactions with options having a particular maturity date on a particular underlying asset. The properties of the datasets are summarised in Table I.

B. Preliminary Analysis

The behaviour of volatility is illustrated in Figures 1 and 2, which show dependencies of volatility on the strike near the beginning and near the end of eeru1206. Initially volatility is relatively flat. Towards the end of the dataset it becomes much more variable and the dependency of volatility on strike takes on a characteristic shape often called the *volatility smile*. The range of the plots was capped at 1, so occasional outliers exceeding 1 are not shown.

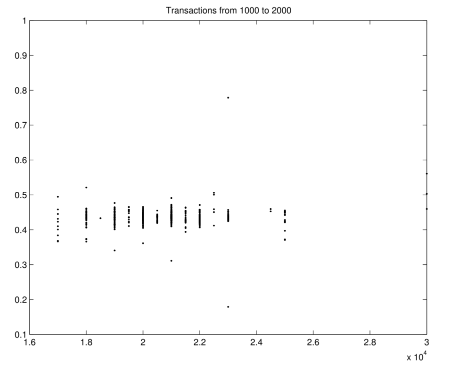

Figure 1. Volatility vs strike, transactions 1000-2000

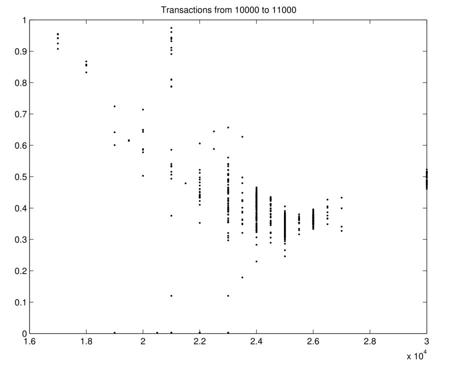

Figure 2. Volatility vs strike, transactions 10000-11000

The number of possible strikes is limited. While theoretically the strike can have any real value, stock exchanges usually restrict strikes to some round numbers in order to improve liquidity. Thus one can consider splitting the time series into separate time series, one for each strike. However, it is apparent from the pictures that some strikes are much more common than others. While for popular strikes the most recent transaction is normally quite near in time, for rare strikes it may be far away.

One thus is naturally drawn to the idea of grouping strikes together and forming combined time series for groups of strikes. However, the pictures do not seem to suggest a natural grouping. It is known that most transactions happen for strikes near the current underlying price and the strikes on the sides are less frequently used. While the pictures seem to be generally congruent with this observation, there is no regular pattern in the distribution of strikes.

Here the ideas of the paper come naturally. Let us create multiple vicinities of strikes, introduce associated specialist experts, and let the AAS converge on the relevant ones.

C. Applying the Algorithm

Let S consisting of $s_1 < s_2 < \ldots < s_L$ be the set of all strikes for a dataset. Let a vicinity of diameter d of strikes be a set of d consecutive strikes; there are $L-d+1$ vicinities of diameter d. A *simple vicinity* of signals corresponding to a vicinity of strikes is the set of all signals with strikes in that vicinity.

A *compound vicinity* of signals is specified by a vicinity of diameter d of strikes and a bit in $\{0, 1\}$. The bit specifies whether the option in the transaction is a put or a call. There are $2(L-d+1)$ compound vicinities of diameter d. Note that some of them may give rise to empty time series if, say, there were no transactions on put options with particular strikes.

A specialist expert monitors a vicinity of signals, be it simple or compound, and applies a time series method to predict the volatility for the next signal falling inside the vicinity.

An elementary method for predicting time series, the exponentially weighted mowing average[1] (EWMA) demonstrated robust performance on the dataset. Given a series $y_1, y_2, \ldots, y_T$, the value $\hat{y}_{T+1} = \mu y_T + (1-\mu)\hat{y}_T$ is predicted, where $\hat{y}_T$ is the prediction output at the previous step and μ is a parameter called the smoothing factor. The values $\mu = 0.2, 0.25, \ldots, 0.95, 1$ were considered (the value $\mu = 1$ corresponds to predicting the last element). At the beginning where no previous element is available, we used the default value of 0.3.

A sleeping expert is thus specified by a vicinity, simple or compound, and a value μ. We evaluated the performance using the cumulative squared loss so we applied the AAS for square loss. In order to apply AAS, predictions of time series methods were capped at 1 and thus for the purpose of merging we assumed that the prediction and outcome space were $[0, 1]$.

Let $\mathcal{L}_{D,\mu}$ be the prediction algorithm obtained by aggregating all specialist experts with simple and compound vicinities of diameter $1, 2, \ldots, D$ and the smoothing factor μ. The number of specialist experts covered by $\mathcal{L}_{D,\mu}$ is $\sum_{d=1}^{D} 3(L-d+1) = 3D(L+1-(D+1)/2)$, where L is the number of strikes. For eeru1206, rtsse307, and gzp307 the numbers of strikes are 24, 15, and 17, respectively, so the numbers of sleeping experts covered by $\mathcal{L}_{L,\mu}$ are 900, 360, and 459, respectively. We use a uniform prior on the experts.

[1]Experiments with more advanced methods such as ARIMA and GARCH returned inferior results so they are not discussed in the paper.

Table II
PERFORMANCE OF $\mathcal{L}$ ON THE WHOLE DATASET

Dataset	Naive		$\mathcal{L}$	Gain	KRR	RTSSE
name	μ	loss	loss		loss	loss
eeru1206	0.6	166.12	158.60	4.5%	151.97	185.05
rtsse307	0.9	155.56	150.53	3.2%	48.80	156.32
gzp307	0.55	40.80	39.3	3.7%	42.58	48.55

Table III
PERFORMANCE OF $\mathcal{L}$ ON THE TRUNCATED DATASET

Dataset	Naive		$\mathcal{L}$	Gain	KRR	RTSSE
name	μ	loss	loss		loss	loss
eeru1206	0.45	129.00	123.74	4.1%	129.48	137.9
rtsse307	0.4	9.78	8.81	9.9%	9.45	8.04
gzp307	0.35	13.96	12.82	8.2%	14.19	14

D. Experimental Results

In this section we compare the results of vicinities merging against various competitors.

The parameters μ and D can be tuned to improve performance. However, we do not do this to ensure fairness of the comparison. Let $\mathcal{L}$ be the prediction algorithm obtained by applying the aggregating algorithm to $\mathcal{L}_{D,\mu}$ with all $D = 1, 2, \ldots, L$ and $\mu = 0.2, 0.25, \ldots, 0.95, 1$ with equal initial weights.

As the main competitor we take the naive algorithm $\mathcal{N}_\mu$ defined as follows. It works on compound vicinities (compound vicinities yield better results) of size 1 and performs no aggregation. For each strike it maintains two time series of values of volatility, for put and for call options, and uses EWMA with the smoothing factor μ to predict the next element. The algorithm $\mathcal{N}_\mu$ thus takes the same advantage as $\mathcal{L}_{D,\mu}$ of exponential smoothing but does not exploit recent transactions on neighbouring strikes.

Table II shows the cumulative losses over the whole dataset. We quote the loss of $\mathcal{L}$ and of the *retrospectively best* naive competitor $\mathcal{N}_\mu$. We see that on all three datasets $\mathcal{L}$ achieves an improvement. As $\mathcal{L}$ and $\mathcal{N}_\mu$ use the same smoothing methods, the improvement should be attributed to the use of neighbouring strikes by $\mathcal{L}_{D,\mu}$.

Figures 3, 4 and 5 display the graphs of regrets $\mathrm{Loss}_t(\mathcal{L}) - \mathrm{Loss}_t(\mathcal{N}_\mu)$ with the retrospectively best μ for the three datasets.

For comparison purposes we also include the cumulative loss of kernel ridge regression (KRR) and the proprietary

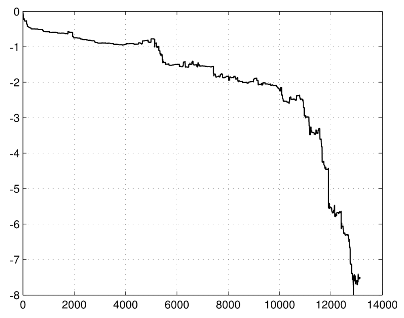

Figure 3. Regret $\mathrm{Loss}_t(\mathcal{L}) - \mathrm{Loss}_t(\mathcal{N}_\mu)$ with the retrospectively best $\mu = 0.6$ for eeru1206

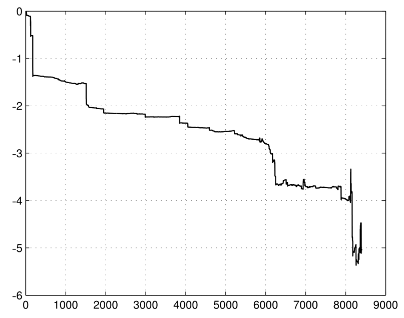

Figure 4. Regret $\mathrm{Loss}_t(\mathcal{L}) - \mathrm{Loss}_t(\mathcal{N}_\mu)$ with the retrospectively best $\mu = 0.9$ for rtsse307

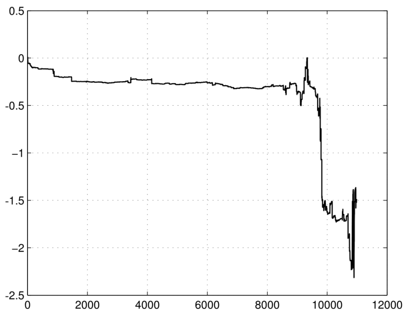

Figure 5. Regret $\mathrm{Loss}_t(\mathcal{L}) - \mathrm{Loss}_t(\mathcal{N}_\mu)$ with the retrospectively best $\mu = 0.55$ for gzp307

algorithm used by RTSSE. We have chosen KRR as a benchmark because KRR with the polynomial kernel can be seen as an extension of polynomial modelling often used to study volatility surfaces (see, e.g., [16]). The KRR results we quote are those using Vapnik's polynomial kernel with the retrospectively selected best degree, ridge parameter, and the size of the sliding window. While the losses of $\mathcal{L}$ and $\mathcal{N}$ are better than those of the proprietary algorithm, the ridge regression performs better and in the case of rtsse307 dramatically better. This calls for an explanation.

The analysis of loss graphs reveals that $\mathcal{L}$ and $\mathcal{N}$ suffer most of their loss over a small tail of the dataset. This is a known effect in finance: right before the maturity of the option, the implied volatility behaves erratically. Some papers (e.g., [17]) even use the parameter $\sigma\sqrt{\tau}$, where τ is time left to maturity and goes to zero, so that fluctuations of implied volatility near maturity become irrelevant.

Table III shows the cumulative losses on the datasets apart from some (relatively small) tail. The size of the tail was chosen individually – this is where loss starts to grow out of proportion. The tail of 2000 examples was dropped from eeru1206 and tails of 1000 from rtsse307 and gzp307. One can see that the losses on truncated datasets are dramatically reduced. The prediction algorithm $\mathcal{L}$ still outperforms the retrospectively best naive competitor and even outperforms ridge regression.

V. Students' Performance at Tests

A. Problem Setup

The "What do you know" dataset consists of 4,851,475 training examples. Each example is an instance of a student answering a question on the Grockit system while preparing for either ACT, GMAT or SAT exams. Each example contains information such as the ID of the student, question number, track and subtrack, tag string for the question, as well as the outcome. The outcome tells us whether the question was answered correctly or not (there are also skipped questions and timed-out questions, but we interpret these outcomes as incorrect answers). We are allowed to give predictions from the interval $[0, 1]$. Intuitively a prediction can be interpreted as the probability of the question being answered correctly. The performance of the algorithm is evaluated by the capped logarithmic loss function. This is equivalent to capping predictions close to 0 or 1 at 0.01 and 0.99 respectively.

1) Batch vs On-line: The original competition was staged as a batch learning problem. The test set was built in the following manner. Of the students who attempted at least six questions, some were selected at random. Then one of the questions they attempted was selected at random (excluding the first six) and inserted into the test set. The selected question and all questions answered by this student later were removed from the dataset. What remained of the dataset after this procedure, formed the training set.

We believe that the on-line setting is more naturally suited to approach this task. Indeed, students answer questions on Grockit sequentially with results being available immediately after each trial. New students appear and students tend to learn over time. However, choosing the on-line mode makes it difficult to compare the results against any off-line benchmark.

2) The benchmark: The competition invited participants to improve on the state-of-the-art algorithm. The benchmark provided was a Rasch-type model predicting the probability of student i, $i = 1, 2, \ldots, N$ answering question j, $j = 1, 2, \ldots, M$ as

$$P(\text{correct answer}|i, j) = \frac{e^{\alpha_i - \beta_j}}{1 + e^{\alpha_i - \beta_j}} \ ,$$

where α_i is a parameter that can be interpreted as the strength of student i and β_j is a parameter that can be interpreted as the difficulty of question j. In the batch mode the parameters are fitted on the training set.

One natural way to estimate these parameters is to treat the problem as logistic regression where each example is represented by a sparse feature vector containing indicators of the student and the question from the example.

We used the implementation of logistic regression from the Vowpal Wabbit [18] package. This implementation is flexible, very fast, and can be run in the on-line mode. By running it on the whole dataset in the on-line mode we obtained loss per element $\mathrm{Loss}_T(\mathrm{Baseline})/T = 0.5572$ (or 0.2420 if logarithm to the base 10 is used to measure loss).

B. *Tag-based Experts*

Each question in the dataset has a tag string. The tags are quite diverse ranging from very informative (such as "maths" or "trigonometric functions" describing the domain of the question) to technical and less helpful for predicting the outcome (such as "multiple choice"). There are 281 tags in total. We would like to use this side information in the on-line mode to get a performance improvement if the information is useful while making sure that the performance does not deteriorate if the information is irrelevant.

A naive approach is to include tags as additional features in logistic regression. The feature vector can be expanded to include indicators of tags.

We also propose an algorithm implementing the vicinity merging idea. Let each tag specify a vicinity. A signal falls in the vicinity if it has the tag in the tag string. For each tag we create a specialist expert E_{tag} which is an on-line logistic regression trained on the subset of examples where the tag is present in the tag string. Each expert is awake at trials where its tag is present in the tag string.

The Learner's prediction is then obtained by aggregating the experts' predictions using the aggregating algorithm for specialist experts (AAS). To guarantee that our performance is never worse than that of the baseline algorithm (plus a small constant), we add the always-awake baseline to the mix (in other words, we include the vicinity containing all signals). We call the resulting prediction algorithm the *tag-based mixture*.

C. *Results and Discussion*

Figure 6 shows the performance of the tag-based mixture on the Grockit dataset. The curves plotted on the graph are the regrets w.r.t. the baseline algorithm, i.e., the differences $\mathrm{Loss}_t - \mathrm{Loss}_t(\mathrm{Baseline})$ for three algorithms: naive algorithm using tags as features, tag-based mixture and tag-based mixture with baseline omitted.

We observe that tag-based mixture achieves some improvement with respect to the baseline algorithm. The no-baseline mixture is shown to highlight the fact that while the tag-based mixture is never worse than the baseline, it improves on it exactly on the sections where tag-based predictors perform well.

A possible explanation why the tag-based mixture can improve on the baseline algorithm is that the mixture learns the specialisation of a student. The coefficients α_i restricted to a particular tag reflect the student's strength on questions falling under this tag. For meaningful tags these coefficients are more useful than those of the baseline algorithm and useless tags are downplayed by the aggregating algorithm automatically.

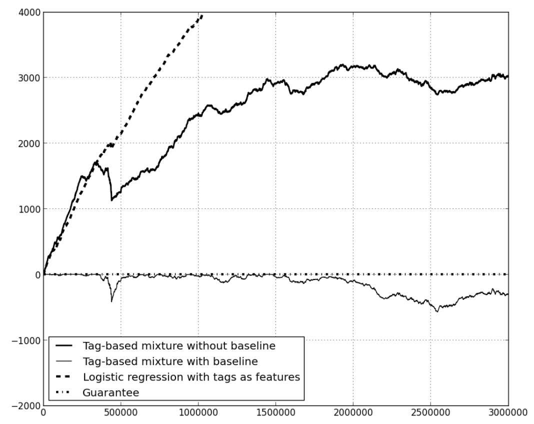

Figure 6. Tag-based AA on Grockit data. Regret w.r.t. baseline algorithm

By contrast, the naive algorithm is not good at this. Adding tag information to feature vectors amounts to adding coefficients γ_{tag} to the Rasch model:

$$P(\text{correct answer}|i, j, \text{tags}) = \frac{e^{\alpha_i - \beta_j + \sum_{\text{tags}} \gamma_{\mathrm{tag}}}}{1 + e^{\alpha_i - \beta_j + \sum_{\text{tags}} \gamma_{\mathrm{tag}}}} .$$

These coefficients are hard to interpret. They are not representing students' strength and the difficulty of the questions is best represented by β_j themselves.

While Figure 6 demonstrates that vicinity merging is capable of making an improvement in performance compared to the baseline, the performance of our algorithm is very inconsistent. The regret often goes up as well as down.

One may speculate that the algorithm needs the student to answer a significant number of questions. To verify this hypothesis we tested the tag-based mixture on the subset of the training set restricted to the students answering more than 1000 questions. Indeed on this dataset the tag-based mixture consistently outperformed the baseline predictor (see Figure 7). The naive algorithm using tags as features still performs badly in this situation.

This observation leads to the following idea on how to improve the performance of the tag-based mixture on the whole dataset. Let all tag experts sleep while the student is answering his or her first N questions and only the baseline predict. The motivation is as follows. As tags are often sparse, tag experts need more time to accumulate sufficiently many examples to train on. While they train, the weights assigned to them by the AAS drop and later they will need time to recover. Even after their predictions have become good, there will be some time before they start contributing to the mixture. Note that we stay within the sleeping experts conceptual framework and simply adjust the time when the experts sleep.

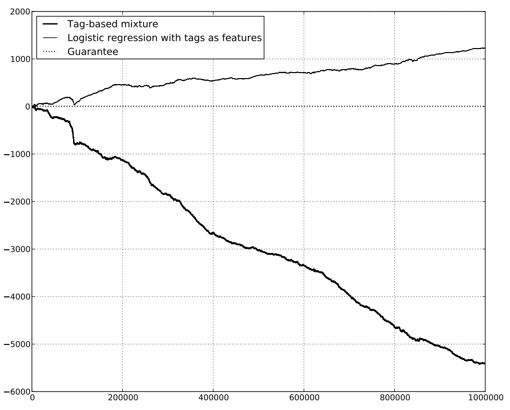

Figure 7. Tag-based AAS on the most active students. Regret w.r.t. baseline algorithm

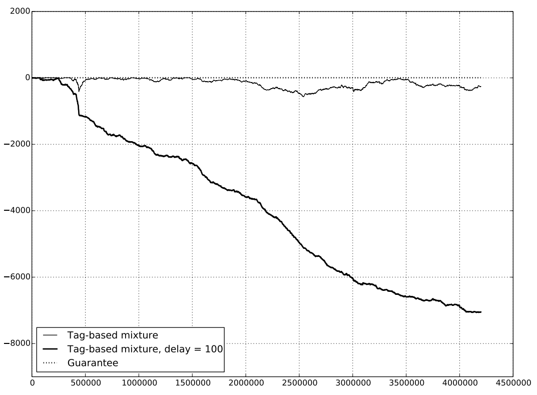

Figure 8. Tag-based AAS with delay $N = 100$

The choice of the *delay* N is an interesting question. From what has been said, it follows that N is linked to the sparsity of tags. We have 281 tags with a few tags per questions. The delay of $N = 100$ appears to be a natural choice. (Delays of 10 and 1000 were shown to give inferior performance.)

Figure 8 shows the regrets w.r.t. the baseline algorithm of the tag-based mixture (same as in Figure 6) and the regret of the mixture of tag experts with the delay of 100. The mixture of experts with the delay exhibits a consistent improvement in performance.

Acknowledgment

Yuri Kalnishkan and Dmitry Adamskiy have been supported by the Leverhulme Trust grant RPG-2013-047 'Online self-tuning learning algorithms for handling historical information'.

The authors would like to thank Russian Trading System Stock Exchange (now defunct through merging) and Dr. Michael Vyugin in particular for providing the data.

References

[1] V. Vovk, "Competitive on-line statistics," *International Statistical Review*, vol. 69, no. 2, pp. 213–248, 2001.

[2] K. S. Azoury and M. K. Warmuth, "Relative loss bounds for on-line density estimation with the exponential family of distributions," *Machine Learning*, vol. 43, pp. 211–246, 2001.

[3] N. Cesa-Bianchi and G. Lugosi, *Prediction, Learning, and Games*. Cambridge University Press, 2006.

[4] Y. Freund, R. E. Schapire, Y. Singer, and M. K. Warmuth, "Using and combining predictors that specialize," in *Proceedings of STOC'97*. ACM, 1997, pp. 334–343.

[5] V. Vovk and F. Zhdanov, "Prediction with expert advice for the Brier game," *Journal of Machine Learning Research*, vol. 10, pp. 2445–2471, 2009.

[6] M. Devaine, P. Gaillard, Y. Goude, and G. Stoltz, "Forecasting electricity consumption by aggregating specialized experts," *Machine Learning*, vol. 90, no. 2, pp. 231–260, 2013.

[7] A. Chernov, Y. Kalnishkan, F. Zhdanov, and V. Vovk, "Supermartingales in prediction with expert advice," *Theoretical Computer Science*, vol. 411, no. 29-30, pp. 2647–2669, 2010.

[8] A. Chernov and V. Vovk, "Prediction with expert evaluators' advice," in *Algorithmic Learning Theory, ALT 2009, Proceedings*, ser. LNCS, vol. 5809. Springer, 2009, pp. 8–22.

[9] V. Vovk, "Aggregating strategies," in *Proceedings of the 3rd Annual Workshop on Computational Learning Theory*. San Mateo, CA: Morgan Kaufmann, 1990, pp. 371–383.

[10] ——, "A game of prediction with expert advice," *Journal of Computer and System Sciences*, vol. 56, pp. 153–173, 1998.

[11] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[12] A. Chernov and F. Zhdanov, "Prediction with expert advice under discounted loss," in *Proccedings of ALT 2010*, vol. LNAI 6331. Springer, 2010, pp. 255–269, see also arXiv:1005.1918 [cs.LG].

[13] S. Busuttil and Y. Kalnishkan, "Weighted kernel regression for predicting changing dependencies," in *Machine Learning: ECML 2007*. Springer, 2007, pp. 535–542.

[14] J. C. Hull, *Options, Futures, and Other Derivatives*, 6th ed. Prentice Hall, 2006.

[15] P. Wilmott, *Paul Wilmott introduces quantitative finance*, 2nd ed. Wiley, 2007.

[16] S. Goncalves and M. Guidolin, "Predictable dynamics in the S&P 500 index options implied volatility surface," *The Journal of Business*, vol. 79, no. 3, pp. 1591–1635, 2006.

[17] K. Demeterfi, E. Derman, M. Kamal, and J. Zou, "A guide to volatility and variance swaps," *The Journal of Derivatives*, vol. 6, no. 4, pp. 9–32, 1999.

[18] J. Langford, L. Li, and A. Strehl, "Vowpal Wabbit online learning project," 2007.