Influence of hyperparameters on online aggregation with countable experts

S. M. Kunin-Bogoiavlenskii, A. V. Zukhba, R. D. Zukhba
kunin-bogoiavlenskii.sm@phystech.edu; a__l@mail.ru; zukhba.rd@phystech.edu
Moscow Institute of Physics and Technology

Aggregating forecasts from multiple experts is a valuable method to improve prediction accuracy. This paper examines the influence of hyperparameters on the performance of the aggregation algorithm for a countable number of experts. We implement a time series generator with specified properties and an aggregating forecasting model. We conduct a series of experiments with various hyperparameters of the algorithm (including initialization weights, mixing update coefficients, etc.), and propose a new mixing scheme, used in the algorithm. The experiments confirm that these hyperparameters have a significant influence on the algorithm's performance.

Keywords: *online learning; aggregating algorithm; prediction with experts' advice; Fixed Share, Mixing Past Posteriors (MPP)*

1 Introduction

This work is inspired by the algorithm, developed in the article [1], considering online game of prediction with experts' advice. The data is presented as a time series, consisting of outcome pairs — «signal» and «response». In contrast to the classical statistical theory of sequential prediction, we make no assumptions about the nature of the data (which could be deterministic, stochastic, etc.). We use machine learning methods to build forecasters within a game-theoretic approach. The online learning master model considers a series of reference forecasters, referred to as experts, to build its opinion by combining their predictions.

The general prediction algorithm with expert advice follows this structure: Learning progresses in trials at discrete points in time. During each step , expert models, based on past observational data subsamples, provide their predictions. The master model then makes a decision using the chosen aggregating algorithm. At the end of the trial, the generator presents the true outcome, and both the master and expert models are scored using a loss function. The difference between the master's cumulative losses and the expert's cumulative losses is defined as regret. The traditional goal of the aggregating algorithm is to keep it as small as possible.

We use special assumptions about the data generation structure when building forecasting strategies. It is assumed that there are multiple generators, whose structure is unknown to the predictors. The time series is obtained by merging segments, each produced by one of the generators. These segments are called areas of stationarity, and can be studied using machine learning methods. Each corresponding local predictive model will be constructed based on data from the area of stationarity and can be then successfully applied in other areas of stationarity generated by the same generator.

In this formulation of the forecasting problem, the series of prediction steps is divided into segments that frame arbitrary sequences of expert strategies. The sequence of segments and its associated sequence of experts is called a partition. The modified goal of the aggregating algorithm is to perform well relative to the best partition. Accordingly, the new concept of algorithm regret is the difference between the algorithm's losses and the cumulative losses of the sequence of experts. This change allows for a more accurate modeling of real-life conditions, where the nature of responses may change over time and different experts may predict with varying degrees of success depending on the current trend.

The corresponding algorithm is called Fixed Share [2]. A further proposed generalization of it is the Mixing Past Posteriors (MPP) method [3]. A characteristic feature of the problem considered in [1] is the absence of a predefined set of competing expert strategies, as was the case in the works cited above. Instead, new expert strategies are constructed at each step of the online learning process. The master must aggregate the forecasts of all expert strategies constructed up to that time in real-time at each step. Algorithm GMPP, proposed in [1], is the foundation of our experiments.

In our work, we systematically explore the impact of various hyperparameters on the algorithm, including different weight functions for initializing experts, various mixing schemes for combining expert predictions, and different update coefficient functions. We also analyze algorithm performance in situations of lack of information, experimenting with different sizes of the expert training window and noise level in the data. We propose a new mixing scheme called «Increasing Past Share» that emphasizes older expert predictions, doing so more smoothly than the default scheme in the GMPP algorithm, which helps to incorporate more historical information.

2 Problem statement

Let $x_1, x_2, \ldots$ be the sequence of signals belonging to the sample space $\mathcal{X}$. The masters goal is to predict the sequence of the corresponding responses $y_1, y_2, \ldots$ belonging to the outcome space $\mathcal{Y}$. We assume that the master knows $a, b \in \mathbb{R}$ — the bounds of the responses sequence, s.t. $\forall i \quad a \leqslant y_i \leqslant b$. There is a countable number of experts $i \in \mathcal{N}$, where $\mathcal{N}$ is the natural numbers set. Let $\lambda : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ be the nonnegative loss function.

At each step t every expert $i \in \mathcal{N}$ provides his prediction $f_t^i = f_t^i(x_t) \in \mathcal{Y}$. After obtaining them the master gives his prediction $p_t = p_t(x_t) \in \mathcal{Y}$. Next, the generator reveals the true outcome y_t, and the losses are computed. Let $h_t = \lambda(p_t, y_t)$ be the master's loss , and $l_t^i = \lambda(f_t^i, y_t)$ be the loss of expert $i \leqslant t$. For $i > t$ (i.e. expert i is not yet initialized), $l_t^i = h_t$.

With designations $L_T^i = \sum_{t=1}^T l_t^i$, representing the cumulative loss of expert i during the first T steps, and $H_T = \sum_{t=1}^T h_t$, representing the master's cumulative loss during the first T steps, we can define the master's regret relative to the expert i as $R_T^i = H_T - L_T^i$.

The best partition is formed based on a hindsight analysis of the experiment results, assuming we know the segment boundaries. For each segment, the best partition predictions are set equal to the predictions of the best local expert (the one that has the lowest cumulative sum on that segment). Master's regret relative to the best partition is defined as $R_T = H_T - L_T$, where L_T is the cumulative loss of the best partition. We use R_T as a metric in our experiments.

Building upon the concepts of Fixed Share and Mixing Past Posteriors, the GMPP algorithm, described below, extends these ideas to handle a countable number of experts.

Algorithm GMPP

Initialize the weights w_1^i so that $\sum\limits_{i \in \mathcal{N}} w_1^i = 1$

for $t = 1, 2, \ldots$ **do**

- Expert f^t initialization
- Signal x_t is received
- Experts' predictions $f_t^i = f_t^i(x_t), 1 \leqslant i \leqslant t$

- Computation of normalized weights of experts $1 \leqslant i \leqslant t$: $\widehat{w}_t^i = \dfrac{w_t^i}{\sum_{j=1}^t w_t^j}$
- Master's prediction evaluation $\gamma_t = Subst(\mathbf{f_t}, \widehat{\mathbf{w}}_\mathbf{t}) = \dfrac{a+b}{2} + \dfrac{1}{2\eta(b-a)} \cdot \ln \dfrac{\sum\limits_{i\in\mathcal{N}} \hat{w}_t^i e^{-\eta(b-f_i)^2}}{\sum\limits_{i\in\mathcal{N}} \hat{w}_t^i e^{-\eta(a-f_i)^2}}$,

where $\widehat{\mathbf{w}}_\mathbf{t} = (\widehat{w}_t^1, \widehat{w}_t^2, \ldots \widehat{w}_t^t)$, $\mathbf{f_t} = (f_t^1, f_t^2, \ldots f_t^t)$
- True outcome y_t is revealed
- Computation of master's loss $h_t = \lambda(p_t, y_t)$ and experts' losses: $l_t^i = \begin{cases} \lambda(f_t^i, y_t), & \text{if } i \leqslant t \\ h_t, & \text{if } i > t \end{cases}$
- **Loss Update** weights modification

$$\widetilde{w}_t^i = \frac{w_t^i e^{\eta l_t^i}}{\sum\limits_{j=1}^t w_t^j e^{-\eta l_t^i} + e^{-\eta h_t}(1 - \sum\limits_{j=1}^t w_t^j)}$$

- **Mixing Update** weights modification

$$\widetilde{w}_{t+1}^i = \alpha_t \widetilde{w}_1^i + (1-\alpha_t)\widetilde{w}_t^i$$

end for

3 Experiment

We implement the synthetic time series generator and the GMPP algorithm itself. We then conduct experiments to analyze the effects of varying hyperparameters. As can be seen from the algorithm, the time computational complexity of the algothim is quadratic. Therefore , we fix certain length of time series for all experiments.

3.1 Data generation

We define generators $G_1, G_2, \ldots, G_k$ by their weight vectors $\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_k \in \mathbb{R}^d$. Here, d represents the dimensionality of the signals.

We generate signals as a series of vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T \sim \mathcal{N}(0, I_d)$, indicating they are drawn from a zero-mean Gaussian distribution with an identity covariance matrix .

We randomly divide this sequence of vectors into a series of segments $\{S_1, S_2, \ldots, S_m\}$ $= \{\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{|S_1|}\}, \{\boldsymbol{x}_{|S_1|+1}, \boldsymbol{x}_{|S_1|+2}, \ldots, \boldsymbol{x}_{|S_1|+|S_2|}\}, \ldots, \{\boldsymbol{x}_{T-|S_m|+1}, \boldsymbol{x}_{T-|S_m|+2}, \ldots, \boldsymbol{x}_T\}\}$. Importantly, each segment S_i corresponds to a specific generator $G_{g(i)}$ according to a random function g. This function determines which generator is responsible for creating responses for that segment. In our experiment, $g(i)$ is uniformly distributed in $\{1, 2, \ldots, k\} \backslash \{g(i-1)\}$, so adjacent segments correspond to different generators.

Ultimately, we obtain the series of responses $y_1, y_2, \ldots, y_T$ by taking the dot product of each signal vector with the weight vector of the corresponding generator and adding the noise: $y_i = \langle \boldsymbol{w}_{g(i)}, \boldsymbol{x}_i \rangle + \varepsilon_i$, where $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$. If $y_i \notin [a, b]$, we regenerate x_i until this condition is met. For our properties, bootstrap shows this happens less than in 4% of the cases, so can be considered insignificant.

In our experiment, we fix the following data characteristics:

- Time series length $T = 2000$,

– Signals dimension $d = 10$,
– Generator weight vectors are drawn from a uniform distribution $U_{[-10,10]^d}$,
– Responses bounds $[a, b] = [-40, 40]$,
– $|S_1|, |S_2|, |S_{m-1}|$ are drawn from a uniform distribution $U_{[50,300]}$, and $|S_m| = T - \sum_{i=1}^{m-1} |S_i|$
– Number of generators $k = 5$,
– White noise variance $\sigma^2 = 1$

Also, to ensure robust performance from the outset, we insert a short priming sequence at the beginning of the time series. This sequence includes one segment from each generator, sizes of segments are also drawn from $U_{[50,300]}$. Importantly, the data from this priming sequence is not included in the regret calculation. Without such a sequence, losses in the beginning of the main series would be much higher than in general, which would deteriorate analysis of hyperparameters influence.

3.2 Experts

Each expert f_t is initialized as a linear regression model. This model is trained on a fixed window of l past observations. The training process yields a weight vector, which is calculated as $\theta_t = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$, where $\boldsymbol{X}^T = (\boldsymbol{x}_{t-1}, \boldsymbol{x}_{t-2}, \dots \boldsymbol{x}_{t-l}),\ \boldsymbol{y} = (y_{t-1}, y_{t-2}, \dots y_{t-l})$.

Given a new signal vector $\boldsymbol{x}$, the expert's prediction is obtained by calculating the dot product between the weight vector and the signal vector: $f_t(\boldsymbol{x}) = \langle \boldsymbol{\theta}_t, \boldsymbol{x} \rangle$

3.3 Hyperparameters

We investigate the impact of various hyperparameters on the GMPP algorithm's performance, focusing on initialization weights, mixing scheme, mixing update coefficients, and expert window size.

1. Initialization Weights (w_1^t):
   – Default: $w_1^t = \frac{1}{(t+1)\ln^2(t+1)}$
   – Experimental:
     – $w_1^t = \frac{1}{(t+4)\ln(t+4)\ln^2\ln(t+4)}$, representing a slower decaying function.
     – $w_1^t = \frac{1}{t^\alpha}$, with $\alpha \in (1, 2]$, to analyze the impact of faster decay rates.
     – $w_1^t = \frac{1}{t^\alpha}$, with $\alpha \in (0, 1]$, to analyze the behavior with diverging weights.
2. Mixing Update Coefficients (α_t):
   – Default: $\alpha_t = \frac{1}{t+1}$
   – Experimental:
     – $\alpha_t = \frac{1}{(t+1)^\beta}$, with $\beta \in (0, 2]$, to analyze different decay speeds.
     – $\alpha_t = \frac{1}{t+c}$, with varying c, to analyze the influence of a constant shift.
     – $\alpha_t = \frac{1}{c}$, with $100 \leqslant c \leqslant 10000$
     – $\alpha_t = \frac{1}{e^{t/3}}$, exponential decay.
3. Mixing Update Scheme:
   While GMPP in [6] utilizes a specific scheme, we explore various Mixing Fixed-Share Update schemes as presented in [3]:

$$\widetilde{w}_{t+1}^i = \sum_{q=1}^{t} \beta_t(q) \widetilde{w}_q^i$$

- Start Vector Share (default in GMPP) — emphasizes the initial and the most recent weight vectors:

$$\beta_t(q) = \begin{cases} \alpha_t, & q = 1 \\ 0, & 1 < q < t \\ 1 - \alpha_t, & q = t \end{cases}$$

- Uniform Past Share — assigns equal weight to all past weight vectors :

$$\beta_t(q) = \begin{cases} \alpha_t \frac{1}{t}, & 1 \leqslant q < t \\ 1 - \alpha_t, & q = t \end{cases}$$

- Decaying Past Share — assigns decreasing weights to past weight vectors:

$$\beta_t(q) = \begin{cases} \alpha_t \frac{1}{(t-q)^\gamma} \frac{1}{Z_t}, & 1 \leqslant q < t \\ 1 - \alpha_t, & q = t \end{cases}$$

,with $Z_t = \sum_{q=1}^{t-1} \frac{1}{(t-q)^\gamma}, \gamma > 0.$

We also propose a new mixing scheme:

- Increasing Past Share — assigns increasing weights to past weight vectors:

$$\beta_t(q) = \begin{cases} \alpha_t (t-q)^\gamma \frac{1}{Z_t}, & 1 \leqslant q < t \\ 1 - \alpha_t, & q = t \end{cases}$$

,with $Z_t = \sum_{q=1}^{t-1} (t-q)^\gamma, \gamma > 0.$

4. Window Size (l): We vary train window size $l \in \{5, 10, 20, 50, 100\}$ to check algorithm performance in situations when experts lack information or when they often train on pieces made by different generators.
5. Noise Variance (σ^2): This parameter controls the level of Gaussian noise added to the generated responses, allowing us to assess the algorithm's robustness to noise. We experiment with various values of σ^2 to observe its impact on regret.

4 Results

We analyze the impact of various hyperparameters on the GMPP algorithm's performance, focusing on initialization weights, mixing schemes, mixing update coefficients, and expert window size. Our primary performance metric is the mean regret relative to the best partition, calculated from four independent runs for each hyperparameter configuration.

4.1 Impact of Initialization Weights

Table 1 explores the influence of different ways of initializing the weights of experts (the importance assigned to each expert at the start of the algorithm) on the regret. We highlight the lowest regret values in bold.

The results show that the best weight function can depend on the mixing scheme chosen. For example, in default scheme «Start Vector Share» weight functions based on diverging series yielded lower regret than those based on converging series. In this case, the $1/x^{0.5}$ weight function achieved the best performance (see Figure 1). When employing converging weight series, schemes «Increasing Past Share» and «Uniform Past Share» appear to be better than others.

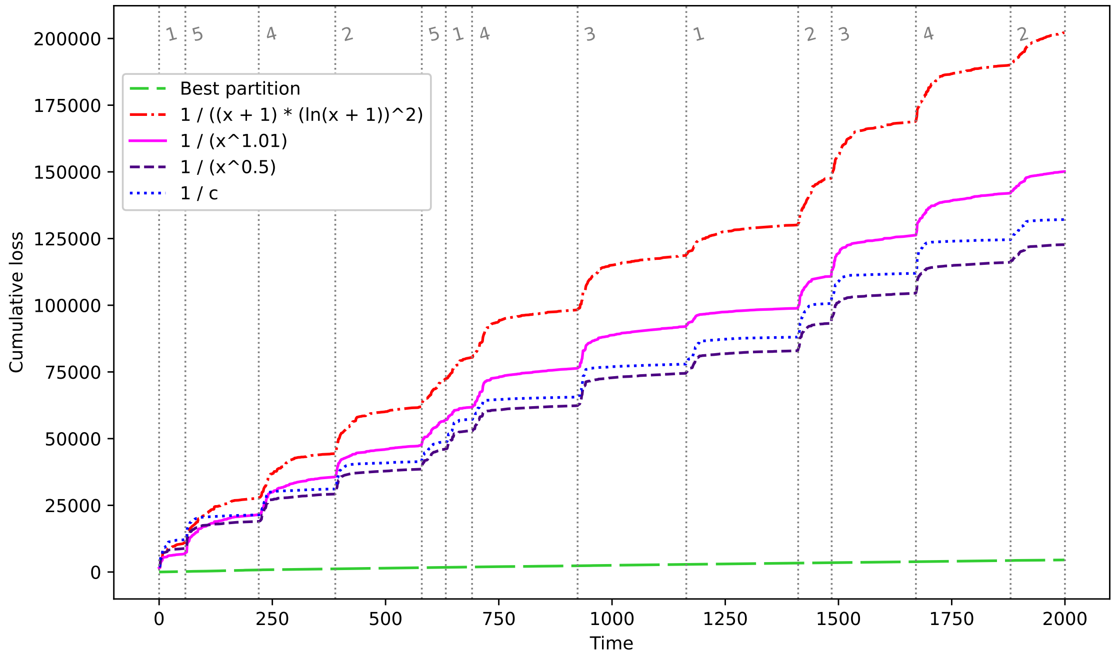


Figure 1 Total loss for different weight functions (alpha function is defaut, $\sigma^2 = 1$, window size = 10)

	Mixing Type			
Weight Function	Decaying Past	Increasing Past	Start	Uniform Past
$1/((x+1)\ln^2(x+1))$	122513.26	122383.91	175594.64	117175.93
$1/c$	128114.78	117241.85	113998.26	119703.68
$1/(x^{0.1})$	127435.12	115101.52	111276.08	117830.57
$1/(x^{0.3})$	126769.51	112900.36	**108946.40**	115819.07
$1/(x^{0.5})$	126191.99	111221.38	**108630.68**	114153.63
$1/(x^{0.7})$	125352.96	**109775.49**	112027.20	112378.87
$1/(x^{0.9})$	123969.39	**109408.73**	122832.21	**110783.66**
$1/x$	122965.58	**110376.37**	131907.11	**110486.71**
$1/(x^{1.01})$	123066.72	**110438.09**	132268.30	**110569.83**
$1/(x^{1.1})$	123963.12	**111050.44**	135253.09	111366.88
$1/(x^{1.2})$	124846.17	111873.03	137976.99	112298.14
$1/(x^2)$	133051.62	124473.63	148080.01	123449.14
$1/((x+4)\ln(x+4)\ln^2(\ln(x+4)))$	121229.77	119413.95	169065.24	114823.76

Table 1 Regret with different weight functions and mixing types (alpha function is defaut, $\sigma^2 = 1$, window size = 10)

4.2 Influence of Training Window Size

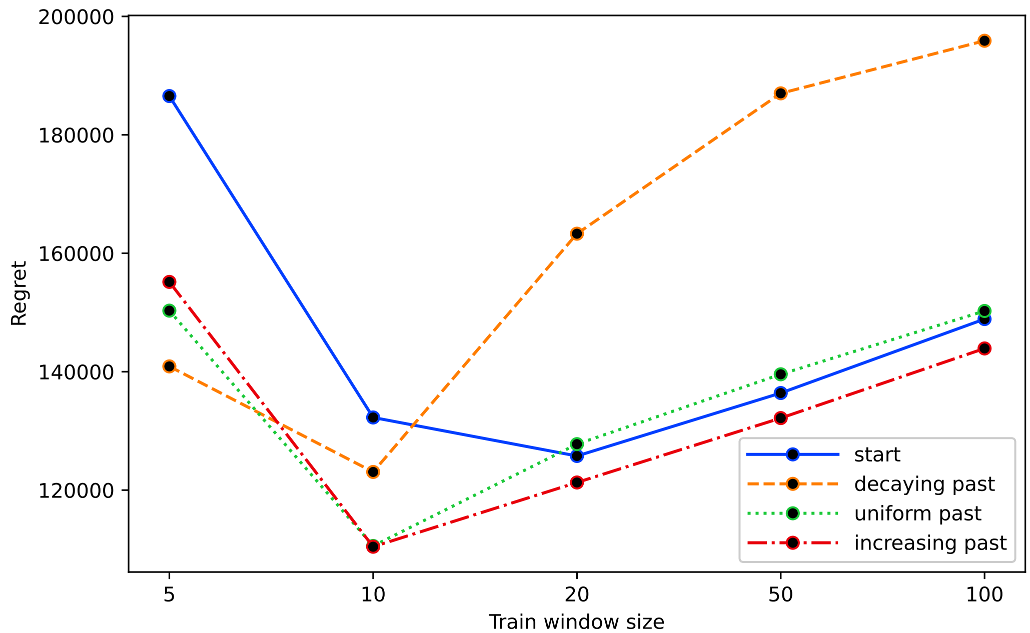


Figure 2 Regret with different mixing schemes and window sizes (alpha function is defaut, weight function is $1/x^{1.01}$, $\sigma^2 = 1$)

Figure 2 shows that the performance of the GMPP algorithm is generally sensitive to the training window size. The best performance is achieved with a window size of 10, suggesting that a balance between having enough past data to train a good model and not being too influenced by older, potentially irrelevant data is important.

The different mixing schemes show varying performance, with «Increasing Past Share» and «Uniform Past Share» schemes demonstrating comparable and generally superior performance. While the «Decaying Past Share» scheme achieves the lowest regret with smaller windows, it suffers from dramatically higher regret as window size increases. Conversely, the default GMPP scheme «Start Vector Share» initially exhibits higher regret but eventually converges towards the performance of the top-performing schemes.

4.3 Impact of Mixing Update Coefficients

Table 2 examines the effect of different coefficients used in the Start Vector Share Mixing Update on the regret. The update coefficient controls how much the past weights distribution among experts influences the new distribution versus the current distribution. The table once again confirms that the weights initialization function $1/x^{0.5}$ is the best among others, and shows that with this function, the default mixing update coefficient $\alpha_t = 1/(t+1)$ stands out as a strong performer, with no serious competitors among other update coefficient functions.

	Weight function			
Alpha function	$\frac{1}{(x+1)\ln^2(x+1)}$	$\frac{1}{x^{1.01}}$	$\frac{1}{x^{0.5}}$	$\frac{1}{c}$
$1/(t+1)$	175594.64	132268.30	**108630.68**	113998.26
$1/(t+1)^{0.5}$	245494.10	207099.32	164079.31	141554.07
$1/(t+1)^{1.5}$	130339.21	125699.71	130185.66	131329.66
$1/(t+1)^2$	136029.09	134929.54	133876.06	132709.43
$1/e^{t/3}$	136413.87	135409.06	134012.15	132760.88
$1/(t+10)$	175411.17	132208.09	**108638.55**	114051.47
$1/(t+100)$	173760.98	131623.56	**108732.56**	114568.90
$1/(t+1000)$	162129.20	127165.47	**110389.16**	118512.52
$1/100$	220008.81	163117.40	129035.87	117521.60
$1/500$	198094.83	141385.79	111706.58	**108969.41**
$1/1000$	184324.73	135612.25	**109449.15**	111709.95
$1/5000$	147561.46	121905.50	115147.43	123741.66
$1/10000$	136414.42	119357.81	120634.74	127510.93
$1/50000$	130783.65	125141.06	129963.25	131542.49

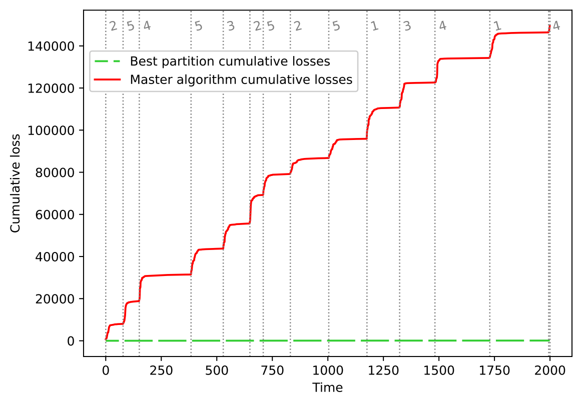
Table 2 Regret with different Alpha and Weight Functions in Start Vector Share Update scheme ($\sigma^2 = 1$, window size = 10)

4.4 Influence of Noise

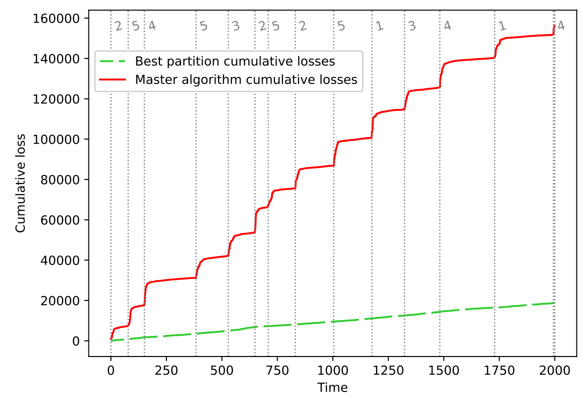
Table 3 investigates how the noise in the generated responses affects the regret of the algorithm. Surprisingly, as the noise level increases, the regret decreases, and at some point, it even becomes negative (see Figure 3). The reason of this is that with size of the window equal only to 10, high noise can dramatically deteriorate training of the linear regression, leading to huge mistakes of the experts. As a result, the master algorithm outperforms the best partition.

	Mixing scheme			
Noise variance	decaying past	increasing past	start	uniform past
0.10	131348.25	114564.74	131578.01	115927.25
1	123066.72	110438.09	132268.30	110569.83
2	109753.81	105398.29	136043.26	103136.06
5	76132.46	92343.75	144554.62	83630.45
6	17155.14	89032.63	146382.86	25178.60
7	-18273.00	29417.82	144827.02	-9208.74
8	-83796.21	-13307.43	147510.57	-55905.83
10	-649917.41	-120166.34	90089.56	-377184.32
12	-828973.51	-1123130.94	-420588.94	-1354731.42

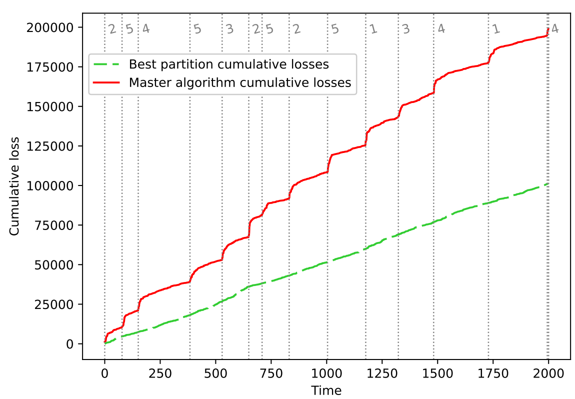
Table 3 Regret with different Mixing Update schemes and Noise Variance (mixing scheme is default, alpha function is defaut, weight function is $1/x^{1.01}$, window size = 10)



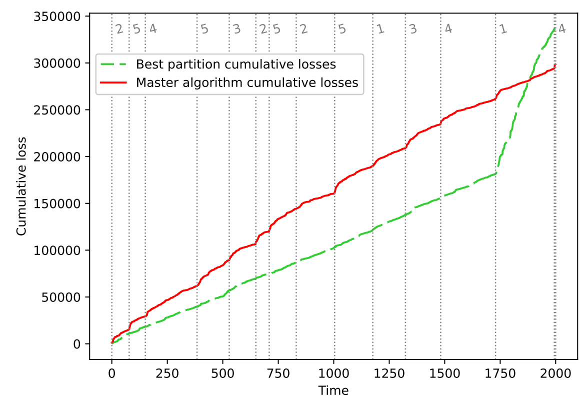
$\sigma^2 = 0.1$



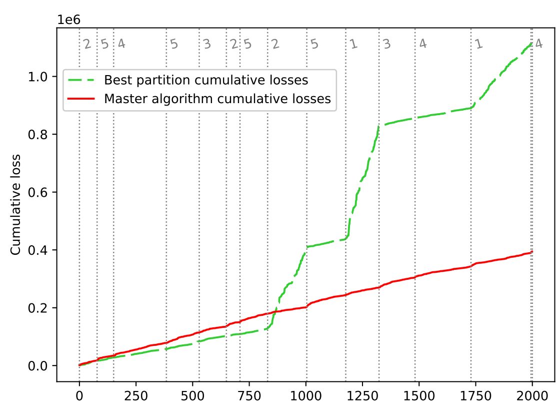
$\sigma^2 = 2$



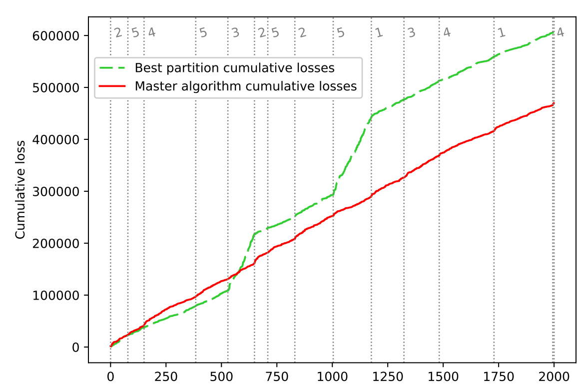
$\sigma^2 = 5$



$\sigma^2 = 8$



$\sigma^2 = 10$



$\sigma^2 = 12$

Figure 3 Total losses with different noise variance σ^2 (mixing scheme is «decaying past», alpha function is defaut, weight function is $1/x^{1.01}$, window size $= 10$, random seed $= 103$)

4.5 Gamma parameter in Increasing Past Share Mixing Update

Table 4 specifically examines the impact of the γ parameter in the «Increasing Past Share» mixing scheme. γ controls how quickly the weight given to past expert predictions increases. The table shows that the impact of the gamma parameter varies depending on the weight function. Interestingly, overall the difference across γ is relatively small.

	Weight function			
γ	$\frac{1}{(x+1)\ln^2(x+1)}$	$\frac{1}{x^{1.01}}$	$\frac{1}{x^{0.5}}$	$\frac{1}{c}$
0.5	119783.90	112279.88	118155.78	**110151.26**
1	122383.91	111221.38	117241.85	**110438.09**
2	126865.67	**110089.54**	116216.44	111563.64
4	133447.69	**109160.56**	115299.68	113922.30

Table 4 Regret with different γ in Increasing Past Share Mixing Update scheme (alpha function is defaut, $\sigma^2 = 1$, window size = 10)

4.6 Impact of Mixing Scheme

Mixing scheme	start			increasing past		
α_t \ w_1^x	$\frac{1}{x^{1.1}}$	$\frac{1}{x^{0.5}}$	$\frac{1}{c}$	$\frac{1}{x^{1.1}}$	$\frac{1}{x^{0.5}}$	$\frac{1}{c}$
$1/(t+1)$	132268	**108631**	113998	**110438**	111221	117242
$1/(t+1)^{0.5}$	207099	164079	141554	183749	160168	145260
$1/(t+1)^{1.5}$	125700	130186	131330	131002	131850	131691
$1/(t+1)^2$	134930	133876	132709	135243	133936	132722
$1/\ln(t+1)$	418163	340332	309672	400131	337777	312109
$1/e^{t/3}$	135409	134012	132761	135409	134012	132761
$1/(t+10)$	132208	**108639**	114051	**110259**	111266	117299
$1/(t+100)$	131624	**108733**	114569	**109744**	111719	117835
$1/(t+1000)$	127165	**110389**	118513	**110754**	115362	121522
$1/100$	163117	129036	117522	129874	125704	120981
$1/500$	141386	111707	**108969**	112134	**110618**	112210
$1/1000$	135612	**109449**	111710	**109844**	**110794**	115281
$1/5000$	121906	115147	123742	115334	121027	125928
$1/10000$	119358	120635	127511	121016	125808	128881
$1/50000$	125141	129963	131542	130965	131895	131885

Table 5 Mean values with different Mixing Update schemes and Weight and Alpha functions ($\sigma^2 = 1$, window size = 10)

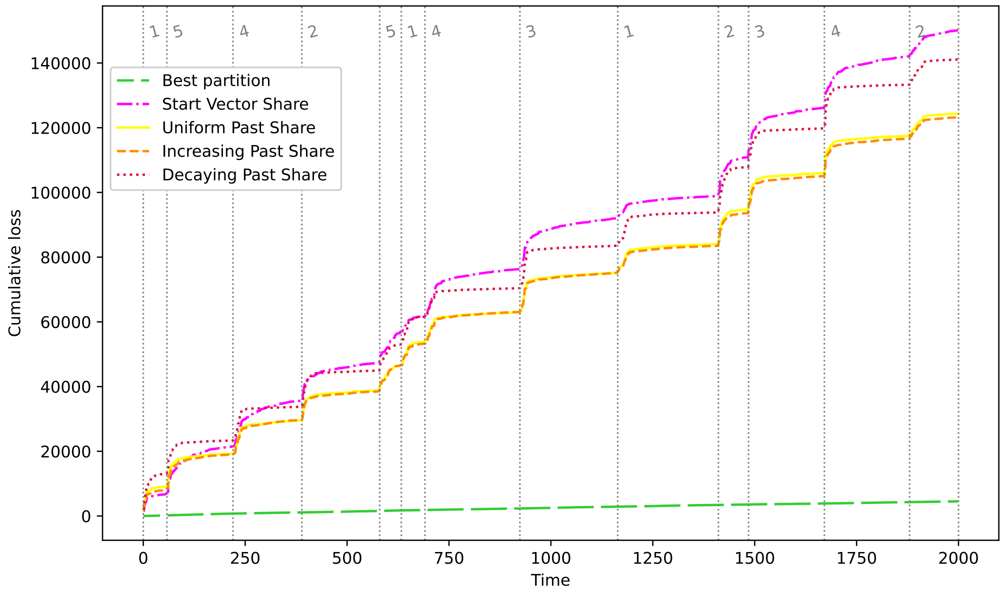


Figure 4 Total loss with different mixing schemes
(alpha function is defaut, weight function is $1/x^{1.01}$, $\sigma^2 = 1$, window size = 10)

Table 5 focuses on the performance of the default GMPP «Start Vector Share» and the newly proposed «Increasing Past Share» mixing schemes with different weight functions and update coefficients. The «Start Vector» scheme generally performs better with the $1/x^{0.5}$ weight function, while «Increasing Past» works better with the $1/x^{1.1}$ weight function.

Figure 4 compares cumulative losses of master algorithm with different mixing shemes. Amusingly, «Uniform Past Share» and «Increasing Past Share» schemes exhibit almost identical total loss curves, both significantly lower than the other schemes. This consistent behavior is observed across multiple random runs of the algorithm and aligns with the findings presented in Figure 2, where these two schemes demonstrated equal regret at a window size of 10.

5 Conclusion

This study investigated the impact of key hyperparameters on the performance of the GMPP algorithm for online prediction of locally stationary time series with a countable number of experts, which in our work were linear regression models. Our experiments systematically explored the influence of initialization weights, mixing update coefficients, mixing schemes, and expert window size on the algorithm's regret relative to the best partition.

Our results demonstrate the importance of carefully tuning these hyperparameters, as their optimal values can vary depending on the chosen mixing scheme and data characteristics. Notably, the default GMPP «Start Vector Share» scheme performed best with diverging initialization weights, like $1/x^{0.5}$, and default mixing coefficients. Furthermore, we proposed a new mixing scheme, «Increasing Past Share», which gradually assigns increasing importance to older expert predictions. In our specific environment, data, and expert configuration, this scheme, combined with certain weight initialization functions, exhibited promising performance. Our experiments also revealed that increasing noise levels in the data could, in some cases, lead to lower, even negative, regret. This suggests the algorithm's potential to outperform the best partition in highly noisy environments.

Future work could focus on further exploring the theoretical underpinnings of these empirical observations, as well as investigating the robustness and generalizability of the findings across a broader range of datasets and experimental settings. Exploring alternative expert models beyond linear regression could further enhance the algorithm's ability to adapt to more complex data.

References

[1] V. V'yugin, V. Trunov. 2023. Prognozirovanie lokal'no statsionarnykh dannykh s ispol'zovaniem predskazanii ekspertnykh strategiy [Prediction of Locally Stationary Data Using Prediction with Expert Advice]. Available at: `http://www.jip.ru/2023/470-487-2023.pdf`

[2] M. Herbster, M. Warmuth. 1998. *Tracking the best expert*. Available at: `https://link.springer.com/content/pdf/10.1023/A:1007424614876.pdf`

[3] O. Bousquet, M. Warmuth. 2002. *Tracking a small set of experts by mixing past posteriors*. Available at: `https://www.jmlr.org/papers/volume3/bousquet02b/bousquet02b.pdf`

[4] N. Cesa-Bianchi, G. Lugosi. 2006. *Prediction, Learning, and Games*. Available at: `https://ii.uni.wroc.pl/~lukstafi/pmwiki/uploads/AGT/Prediction_Learning_and_Games.pdf`

[5] Hyndman, R. J. & Athanasopoulos, G., 2nd edition. 2018. *Forecasting: Principles and Practice. OTexts: Melbourne, Australia* . Available at: `https://otexts.com/fpp2/`

[6] V. V'yugin. 2022. Matematicheskie osnovy mashinnogo obucheniya i prognozirovaniya [Mathematical Foundations of Machine Learning and Forecasting]. Available at: `http://iitp.ru/upload/publications/6256/vyugin1.pdf`