Appendix for paper multi-step prediction of financial asset return probability density function using parsimonious autoregressive sequential model

1 Appendix A, Brief precision test on Gauss-Legendre integration rule

Gauss-Legendre integration quadrature rule is used in this study to conduct numerical integration. The integration grid point used in Gauss-Legendre integration quadrature rule is given by the roots of Legendre polynomial. Table 1 (located in page 2) illustrate our trail integration result using Gauss-Legendre integration quadrature rule on calculating $E(r)$, $E(r^2)$,$E(r^3)$ and $E(r^4)$. Of normal distribution and E(r) of t-distribution ($E(r^3)$ and $E(r^4)$ values for t-distribution is not known. Hence, are not tested), theoretical values are also given. We tested the quadrature rule in two different grid fineness level (100 integration points and 1000 integration points). Integration grid used in actual study in this paper is conducted using 10000 integration points, which is even more precise than result given in table 1

2 Appendix B, Segregated-BPTT algorithm, mathematical proof of its convergence

Instead of directly proving the convergence of Algorithm 1 in the main part of the paper, in this section, we demonstrated the convergence of general recurrent neural network (RNN) with our segregated back propagation through time (segregated BPTT) algorithm. The following prove are applicable to both our algorithm 1 as well as a broad spectrum of approximate RNN training methods.

We first provide proof that the error of the approximate hidden state is proportion to the learning rate and then proceed to proof that error of the approximate gradient is proportion to the learning rate times the true gradient gradient. After that, we will prove that gradient descent using our approximate gradient will converge.

In the following proof, $\boldsymbol{h}_t^k$ is the hidden state at time step t and gradient descent optimization loop number k, $\boldsymbol{x}_t$ is the input vector at time step t, ϕ is the nonlinear activation function and we assume its derivative is Lipschitz continuous,$\boldsymbol{W}_h^k$ and $\boldsymbol{W}_x^k$ are two weight matrices at gradient descent optimization loop number k. Because the gradient descent update on $\boldsymbol{W}_h$ and $\boldsymbol{W}_x$ is used, we have

$$\boldsymbol{W}_h^k = \boldsymbol{W}_h^{k-1} + \lambda \frac{\partial loss}{\partial \boldsymbol{W}_h}|_{k-1} \tag{1}$$

$$\boldsymbol{W}_x^k = \boldsymbol{W}_x^{k-1} + \lambda \frac{\partial loss}{\partial \boldsymbol{W}_x}|_{k-1} \tag{2}$$

Lemma 1. *Given that*

$$\tilde{\boldsymbol{h}}_t^k = \phi(\boldsymbol{W}_h^k \tilde{\boldsymbol{h}}_{t-1}^{k-1} + \boldsymbol{W}_x^k \boldsymbol{x}_t) \tag{3}$$

$$\tilde{\boldsymbol{h}}_0^k = \boldsymbol{W}_x^k \boldsymbol{x}_t \quad \forall \quad k \in N \tag{4}$$

$\overrightarrow{0}$ *is the zero vector.*

Then $\tilde{\boldsymbol{h}}_t^k = \tilde{\boldsymbol{h}}_t^{k-1} + \lambda \boldsymbol{c}_0$, where $\boldsymbol{c}_0$ is a vector that contains terms that are at least 0th order with respect to λ

Proof. we proof the lemma inductively:

For $t = 0$, the theorem hold by definition.

For $t > 0$ assume that

$$\tilde{\boldsymbol{h}}_{t-1}^{k-1} = \tilde{\boldsymbol{h}}_{t-1}^{k-2} + \lambda \boldsymbol{d}_0 \tag{5}$$

Using equation VII.1 and VII.2

$$\tilde{\boldsymbol{h}}_t^k = \phi(\boldsymbol{W}_h^k \tilde{\boldsymbol{h}}_{t-1}^{k-1} + \boldsymbol{W}_x^k \boldsymbol{x}_t) \tag{6}$$

$$\tilde{\boldsymbol{h}}_t^k = \phi((\boldsymbol{W}_h^{k-1} + \lambda \boldsymbol{D}_1)(\tilde{\boldsymbol{h}}_{t-1}^{k-2} + \lambda \boldsymbol{d}_0) + (\boldsymbol{W}_x^{k-1} + \lambda \boldsymbol{D}_2)\boldsymbol{x}_t) \tag{7}$$

Where $\boldsymbol{D}_1$ and $\boldsymbol{D}_2$ are matrix independent of λ. Apply Taylor expansion and collect all λ independent terms into vector $\boldsymbol{c}_0$, and we can arrive on following equation:

$$\tilde{\boldsymbol{h}}_t^k = \phi(\boldsymbol{W}_h^{k-1} \tilde{\boldsymbol{h}}_{t-1}^{k-2} + \boldsymbol{W}_x^{k-1} \boldsymbol{x}_t) + \lambda \boldsymbol{c}_0 \tag{8}$$

$$\tilde{\boldsymbol{h}}_t^k = \tilde{\boldsymbol{h}}_t^{k-1} + \lambda \boldsymbol{c}_0 \tag{9}$$

$\boldsymbol{c}_0$ is a vector that contains terms that are at least 0th order with respect to λ. Hence, the Lemma holds. □

Lemma 2. *Given that*

$$\boldsymbol{h}_t^k = \phi(\boldsymbol{W}_h^k \boldsymbol{h}_{t-1}^k + \boldsymbol{W}_x^k \boldsymbol{x}_t) \tag{10}$$

$$\tilde{\boldsymbol{h}}_t^k = \phi(\boldsymbol{W}_h^k \tilde{\boldsymbol{h}}_{t-1}^{k-1} + \boldsymbol{W}_x^k \boldsymbol{x}_t) \tag{11}$$

$$\boldsymbol{h}_0^k = \boldsymbol{W}_x^k \boldsymbol{x}_t \quad \forall \quad k \in N \tag{12}$$

$$\tilde{\boldsymbol{h}}_0^k = \boldsymbol{W}_x^k \boldsymbol{x}_t \quad \forall \quad k \in N \tag{13}$$

Then $\boldsymbol{h}_t^k = \tilde{\boldsymbol{h}}_t^k + \lambda \boldsymbol{c}_1$, where $\boldsymbol{c}_1$ is a vector that contains terms that are at least 0th order with respect to λ

tasks	$\mathcal{N}(\mu=1,\sigma=2)$ $E(r)$	$\mathcal{N}(\mu=1,\sigma=2)$ $E(r^2)$	$\mathcal{N}(\mu=1,\sigma=2)$ $E(r^3)$
Integration grid points 100	0.99999994	4.9999995	12.999999
Integration grid points 1000	1.0	5.0	13.0
Theoretical value	1.0	5.0	13.0
tasks	$\mathcal{N}(\mu=1,\sigma=2)$ $E(r^4)$	$t-dist(df=9,\mu=0,\sigma=1)$ $E(r)$	$t-dist(df=9,\mu=0,\sigma=1)$ $E(r^2)$
Integration grid points 100	73.0	-0.00000001	1.2857101
Integration grid points 1000	73.0	-0.00000001	1.2857149
Theoretical value	73.0	0.0	1.2857142

Table 1: The numerical integration result using Gauss-Legendre quadrature rule in comparison with theoretical values

Proof. we proof the lemma inductively:

For $t=0$, $\boldsymbol{h}_0^k = \tilde{\boldsymbol{h}}_0^k$.

For $t>0$, assume that

$$\boldsymbol{h}_{t-1}^k = \tilde{\boldsymbol{h}}_{t-1}^k + \lambda \boldsymbol{e}_0 \tag{14}$$

where $\boldsymbol{e}_0$ is a vector that contains terms that are at least 0th order with respect to λ.

Insert equation (VII.14) into equation (VII.10), we have

$$\boldsymbol{h}_t^k = \phi(\boldsymbol{W}_h^k(\tilde{\boldsymbol{h}}_{t-1}^k + \lambda \boldsymbol{e}_0) + \boldsymbol{W}_x^k \boldsymbol{x}_t) \tag{15}$$

Using Lemma 1, $\tilde{\boldsymbol{h}}_t^k = \tilde{\boldsymbol{h}}_t^{k-1} + \lambda \boldsymbol{c}_0$, we can rewrite equation above as:

$\boldsymbol{h}_t^k = \phi(\boldsymbol{W}_h^k(\tilde{\boldsymbol{h}}_{t-1}^{k-1} + \lambda \boldsymbol{e}_1) + \boldsymbol{W}_x^k \boldsymbol{x}_t)$, with $\boldsymbol{e}_1 = \boldsymbol{e}_0 + \boldsymbol{c}_0$

As long as derivative of ϕ exist, Taylor expansion gives following equation:

$$\begin{aligned}\boldsymbol{h}_t^k = \phi(\boldsymbol{W}_h^k \tilde{\boldsymbol{h}}_{t-1}^{k-1} + \boldsymbol{W}_x^k \boldsymbol{x}_t) + \\ \lambda \frac{\partial \phi(\boldsymbol{W}_h^k \tilde{\boldsymbol{h}}_{t-1}^{k-1} + \boldsymbol{W}_x^k \boldsymbol{x}_t)}{\partial(\boldsymbol{W}_h^k \tilde{\boldsymbol{h}}_{t-1}^{k-1} + \boldsymbol{W}_x^k \boldsymbol{x}_t)}^T \boldsymbol{e}_1 + o(\lambda^2)\end{aligned} \tag{16}$$

We can rewrite the equation as: $\boldsymbol{h}_t^k = \tilde{\boldsymbol{h}}_t^k + \lambda \boldsymbol{c}_1$, where $\boldsymbol{c}_1$ is a vector that contains terms that are at least 0th order with respect to λ □

Lemma 3. $\tilde{\boldsymbol{g}}_t^k = \tilde{\boldsymbol{g}}_t^{k-1} + \lambda \boldsymbol{c}_2$.

The proof of Lemma 3 is the same as Lemma 1, It is omitted for brevity.

Lemma 4.

$$\boldsymbol{g}_{t-1}^k = \boldsymbol{g}_t^k \frac{\partial \phi(\boldsymbol{z}_t^k)}{\partial \boldsymbol{z}_t^k} \frac{\partial \boldsymbol{z}_t^k}{\partial \boldsymbol{h}_{t-1}^k} \tag{17}$$

with $\boldsymbol{z}_t^k = \boldsymbol{W}_h^k \boldsymbol{h}_{t-1}^k + \boldsymbol{W}_x^k \boldsymbol{x}_t$ *and* $\frac{\partial \boldsymbol{z}_t^k}{\partial \boldsymbol{h}_{t-1}^k} = \boldsymbol{W}_h^k$

$$\tilde{\boldsymbol{g}}_{t-1}^k = \tilde{\boldsymbol{g}}_t^{k-1} \frac{\partial \phi(\tilde{\boldsymbol{z}}_t^k)}{\partial \tilde{\boldsymbol{z}}_t^k} \frac{\partial \tilde{\boldsymbol{z}}_t^k}{\partial \tilde{\boldsymbol{h}}_{t-1}^{k-1}} \tag{18}$$

with $\tilde{\boldsymbol{z}}_t^k = \boldsymbol{W}_h^k \tilde{\boldsymbol{h}}_{t-1}^{k-1} + \boldsymbol{W}_x^k x_t$ *and* $\frac{\partial \tilde{\boldsymbol{z}}_t^k}{\partial \tilde{\boldsymbol{h}}_{t-1}^{k-1}} = \boldsymbol{W}_h^k$

$$\boldsymbol{g}_T^k = \tilde{\boldsymbol{g}}_T^k \tag{19}$$

T is the last time step.
Then $\tilde{\boldsymbol{g}}_{t-1}^k = \boldsymbol{g}_{t-1}^k + \lambda \boldsymbol{c}_3$

Proof. First, if t is the last step of RNN, i.e. $t=T$, then by definition, $\boldsymbol{g}_T^k = \tilde{\boldsymbol{g}}_T^k$.

For t not equal to last step, assume that

$$\tilde{\boldsymbol{g}}_t^k = \boldsymbol{g}_t^k + \lambda \boldsymbol{f}_0 \tag{20}$$

$\boldsymbol{f}_0$ is a vector of contains terms that are at least 0th order with respect to λ. Insert the assumption to equation VII.18 and use Lemma 3:

$$\tilde{\boldsymbol{g}}_{t-1}^k = (\boldsymbol{g}_t^k + \lambda(\boldsymbol{f}_0 - \boldsymbol{c}_2)) \frac{\partial \phi(\tilde{\boldsymbol{z}}_t^k)}{\partial \tilde{\boldsymbol{z}}_t^k} \boldsymbol{W}_h^k \tag{21}$$

$$\tilde{\boldsymbol{g}}_{t-1}^k = (\boldsymbol{g}_t^k + \lambda \boldsymbol{f}_1) \frac{\partial \phi(\tilde{\boldsymbol{z}}_t^k)}{\partial \tilde{\boldsymbol{z}}_t^k} \boldsymbol{W}_h^k \tag{22}$$

where $\boldsymbol{f}_1 = \boldsymbol{f}_0 - \boldsymbol{c}_2$

Using Lemma 2 and expanded $\frac{\partial \phi(\tilde{\boldsymbol{z}}_t^k)}{\partial \tilde{\boldsymbol{z}}_t^k}$ using Taylor expansion, we can rewrite $\frac{\partial \phi(\tilde{\boldsymbol{z}}_t^k)}{\partial \tilde{\boldsymbol{z}}_t^k} = \frac{\partial \phi(\tilde{\boldsymbol{z}}_t^k)}{\partial \tilde{\boldsymbol{z}}_t^k} + \lambda \boldsymbol{F}_2$, $\boldsymbol{F}_2$ is a matrix independent of λ. The math is omitted here due to that Taylor expansion of a vector valued multiple variable function will produce a 3-D tensor in the second order term. Therefore, we have

$$\tilde{\boldsymbol{g}}_{t-1}^k = (\boldsymbol{g}_t^k + \lambda \boldsymbol{f}_1) \frac{\partial \phi(\tilde{\boldsymbol{z}}_t^k)}{\partial \tilde{\boldsymbol{z}}_t^k} \boldsymbol{W}_h^k \tag{23}$$

$$= (\boldsymbol{g}_t^k + \lambda \boldsymbol{f}_1)(\frac{\partial \phi(\tilde{\boldsymbol{z}}_t^k)}{\partial \tilde{\boldsymbol{z}}_t^k} + \lambda \boldsymbol{F}_2) \boldsymbol{W}_h^k \tag{24}$$

$$\begin{aligned}= \boldsymbol{g}_{t-1}^k + \lambda \frac{\partial \phi(\tilde{\boldsymbol{z}}_t^k)}{\partial \tilde{\boldsymbol{z}}_t^k} \boldsymbol{W}_h^k \boldsymbol{f}_1 \\ + \lambda \boldsymbol{W}_h^k \boldsymbol{F}_2 \boldsymbol{g}_t^k + o(\lambda^2)\end{aligned} \tag{25}$$

For λ that is sufficiently small, $o(\lambda^2)$ can be neglected. Hence

$$\tilde{\boldsymbol{g}}_{t-1}^k = \boldsymbol{g}_{t-1}^k + \lambda \boldsymbol{c}_3 \tag{26}$$

where $\lambda \boldsymbol{c}_3$ collects all terms involved, and $\boldsymbol{c}_3$ is a vector that contains terms that are at least 0th order with respect to λ. Hence, the theorem holds. □

In equation 1.6 of Bertsekas and Tsitsiklis' paper[Bertsekas and Tsitsiklis, 2000], they proved that gradient descent with approximate gradient and gradient error that took the form of eqn. VII.26 converges, hence the convergence of our algorithm using approximate gradient is proven.

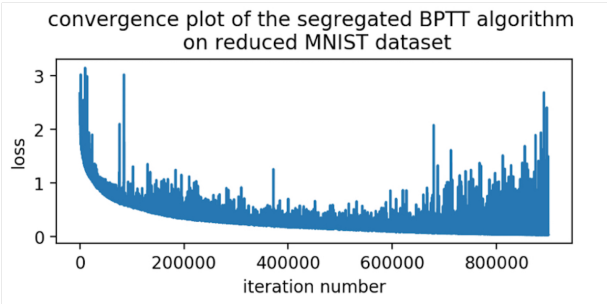

Figure 1: Convergence plot of Segregated BPTT on MNIST dataset

3 Appendex C. Segregated-BPTT, experimental results on its convergence and training speed

Similar to appendix B of this paper, in appendix C, we tested the convergence of the algorithm on a task harder than training our PA-MS-DR model. The essense of our segregated-BPTT algorithm is using approximation to split the temporal dependencies between time-steps. In appendix C, the segregated-BPTT algorithm is migrated to accelerate the training of a Jordan type RNN model [**?**] on MNIST classification task. This task involves 28 time-steps, which is much harder for the segregated BPTT algorithm to train than training the PA-MS-DR model, since segregated BPTT algorithm uses approximation to tackle dependencies between time-steps.

Accelerate the training of RNN to be as fast as convolutional neural networks (CNN) has attracted the attention of many researchers. In lei's work [Lei *et al.*, 2017], the author proposed a highly efficient recurrent structured named simple recurrent unit (SRU), which accelerate the training by minimizing the temporal dependency between time steps. Inside the structure of SRU, all gates that controls information passing can be computed independent of time, the only temporal dependent components are the cell variable.

Contrast with the SRU, our methods accelerate the training of RNN by modifying the training method instead of the recurrent structure. By using hidden states and gradients of hidden states from last gradient descent step, time steps of the RNN are decoupled, and parallelization in temporal dimension is now feasible.

In addition, because segregated BPTT allows parallelization of the model in temporal dimension, we are able to apply batch-dependent loss and batch-normalization during parallellel training of the model. Parallel training of the model on 4 GPUs accelerated the training speed by 190%.

The convergence of the segregated BPTT algorithm is tested in training of a simplified RNN model on the reduced MNIST dataset. The simplified RNN model is a modfied Jordan type RNN, it uses two hidden layers to process input x_t instead of one. The convergence plot is shown in figure 1 below. As is shown in figure 1, the training converged in 80000 training loops. Although the convergence loss curve contains a large amount of sharp peaks, it nevertheless converges as is theoretically proven. The reason for these sharp peaks is because the segregated BPTT algorithm is not designed for tasks where the temporal dependency is highly significant and a very long sequence of the time steps exist (28 time steps). Training of our model in the paper using the segregated BPTT algorithm doesnot possess these problems. In addition, the problem of not being able to process long time sequence and high temporal dependency can be greatly alleviated by re-computing the exact gradient every 100 or 200 training steps.

4 Appendix D, Quantile violation results

The quantile violation ratio result of table 2 is listed here to avoid confusion. It should be noted that, higher quantile violation ratio doesn't necessarily means a worse prediction. The Kupiec's proportion of failures coverage test [Kupiec, 1995] gives lower and upper bound for ideal quantile violation ratio and its testing result is shown in table 2 in main part of this paper.

References

[Bertsekas and Tsitsiklis, 2000] Dimitri P Bertsekas and John N Tsitsiklis. Gradient convergence in gradient methods with errors. *SIAM Journal on Optimization*, 10(3):627–642, 2000.

[Kupiec, 1995] Paul H. Kupiec. Techniques for verifying the accuracy of risk management models. *Social Science Electronic Publishing*, 3(2):73–84, 1995.

[Lei *et al.*, 2017] Tao Lei, Yu Zhang, and Yoav Artzi. Training rnns as fast as cnns. *arXiv preprint arXiv:1709.02755*, 2017.

	S&P 500	NASDAQ 100	Nikkei 225	EUR-USD	JPY-USD
AR-GJR-GARCH-*t*	0.0182	0.0091	0.0211	0.0244	0.0149
AR-GARCH-*t*	0.0182	0.0091	0.0211	0.0252	0.0158
GJR-GARCH-*t*	0.0309	0.0091	0.0236	0.0211	0.0124
GARCH-*t*	0.0291	0.0091	0.0236	0.0236	0.0116
PA-MS-DR-*t*	0.0256	0.0235	0.0181	0.0242	0.003
MS-DR-*t*	0.0182	0.0203	0.0328	0.0505	0.002
AR-GJR-GARCH	0.0364	0.02	0.0374	0.0309	0.0241
AR-GARCH	0.0355	0.0182	0.0366	0.0301	0.0232
GJR-GARCH	0.0345	0.0182	0.0333	0.0309	0.0232
GARCH	0.0355	0.0164	0.0341	0.0301	0.0232
PA-MS-DR	0.0246	0.0245	0.0198	0.0505	0.004
MS-DR	0.0246	0.0256	0.0172	0.04545	0.005

Table 2: The ratio of quantile violations. Kupiec's proportion of failures coverage test gives lower bond and upper bond of 0.0203 and 0.0813 for S & P 500, NASDAQ 100 and Nikkei 225 data, for EUR-USD and JPY-USD, the upper bond and lower bonds are 0.0182 and 0.0818