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

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

In this section, the convergence of RNN with approximate gradient is proved. 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

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:

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

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

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

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

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

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[**?**], 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.

2 Appendix C. Segregated-BPTT, experimental results on its convergence and training speed

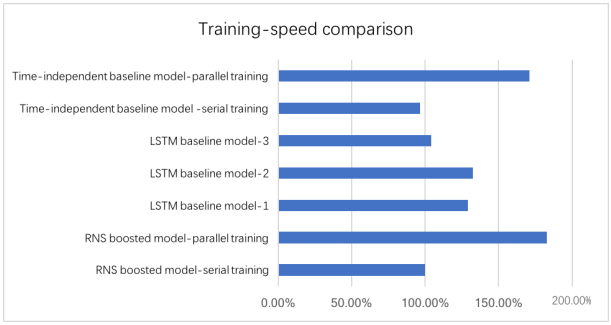

Figure 1: Training speed comparison

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, 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 after parallelization. The relative time consumption before and after applying parallelization is listed in figure 6. Figure 6 also listed the relative training speed of other models involved in this paper. Note that the implementation of segregated BPTT algorithm on LSTM baseline models involves a large amount of coding, hence are not included in the comparison. For serial training, a single GTX-1080Ti graphic card is used and for parallel training, 4 GTX-1080Ti graphic cards are used.

Because using segregated BPTT facilitated parallelization of the model and enables applying batch-dependent loss and batch-normalization after parallelization, we tested the training time per training step before and after applying parallelization. The relative time consumption (with serial model is set to be the baseline) for doing a single gradient descent step before and after parallelization on a 4 card machines is listed below:

As can be seen from figure 6, parallelization indeed accelerated training of the model, however, the speed boost is slightly far away from the theoretical speed boost of 400%. After inspection of the outputted tensorflow logs, we believe the main reason is due to both the dataset size and network

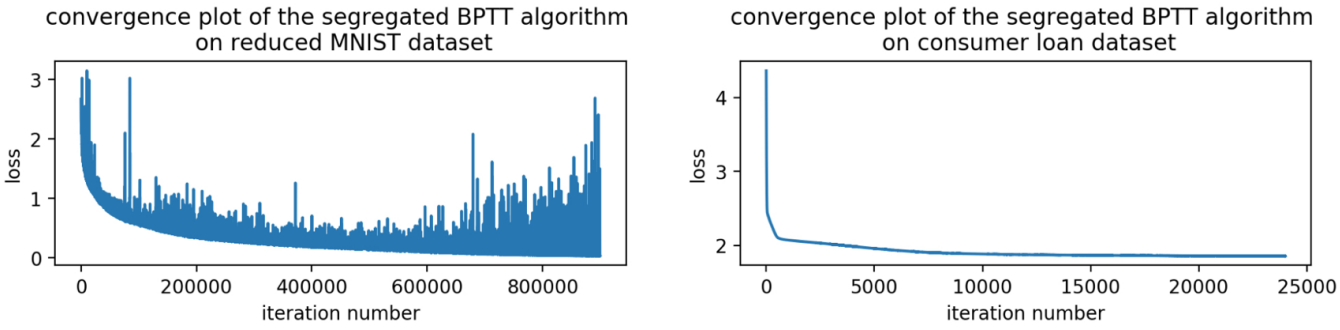

Figure 2: Convergence plot of Segregated BPTT on customer lending dataset and MNIST dataset

size is relatively small and overhead of parallelization overweighted the benefit of parallelization.

The convergence of the segregated BPTT algorithm is tested in training of our RNS enhanced model on consumer loan dataset and 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 7 below. As indicated by Figure 7, the convergence of the segregated BPTT algorithm is very smooth in the consumer loan dataset, however, in the reduced MNIST dataset, though the model eventually converged as is theoretically proven, the convergence rate is low and the loss curve contains a huge amount of sharp peaks, indicating the training loss fluctuate wildly during the training.

The reason for its low performance on training of RNN on reduced MNIST dataset 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). When the time steps form a long sequence, the errors in approximate gradient will accumulate during this back-propagation process, which result in large fluctuation of the loss curve during the training process. The problem of not being able to process long time sequence and high temporal dependency can be alleviated by recomputing the exact gradient every 100 or 200 training steps. This method is not used in producing the convergence curve, but will be used during productive training process.