Deep Recurrent Survival Analysis

Supplemental Material

Kan Ren, Jiarui Qin, Lei Zheng, Zhengyu Yang,
Weinan Zhang, Lin Qiu, Yong Yu
APEX Data & Knowledge Management Lab
Shanghai Jiao Tong University
kren, qinjr, zhenglei, zyyang, wnzhang, lqiu, yyu@apex.sjtu.edu.cn

Appendix

Model Implementations

In this section, we describe in detail the implemented architecture of the proposed model (DRSA).

Recall that, each sample is a triple $(\boldsymbol{x}, z, t_l)$ where $\boldsymbol{x} \in \mathbb{R}^K$ is the K-dimensional vector representing the feature of the sample, $z \in \mathbb{N}^+$ is an integer of the true event time and $t_l \in \mathbb{N}^+$ is an integer of the observing time.

As is illustrated in Figure 1, the input to our DRSA model is the triple $(\boldsymbol{x}, z, t_l)$. For each recurrent unit, we feed the input as $(\boldsymbol{x}, t_j)$ where $1 \leq j \leq l$ is an integer representing the time interval of the current unit.

Note that $\boldsymbol{x}$ is a multi-hot encoded feature vector including a series of one-hot encoded features, and we first put it through an embedding layer as

$$\boldsymbol{e} = embed(\boldsymbol{x}).$$

After getting the embedding vector e, We concatenate the input vector $\boldsymbol{e}$ and the time stamp t_j as

$$\boldsymbol{v}_j = concate(\boldsymbol{e}, t_j).$$

Specifically, we implement Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber 1997) as the recurrent unit $f_{\boldsymbol{\theta}}(\boldsymbol{x}, t_j \mid \boldsymbol{r}_{j-1})$ as

$$\begin{aligned}
\mathbf{f}_j &= \sigma(\mathbf{W}_f \cdot \boldsymbol{v}_j + \mathbf{b}_f) \\
\mathbf{i}_j &= \sigma(\mathbf{W}_i \cdot \boldsymbol{v}_j + \mathbf{b}_i) \\
\mathbf{o}_j &= \sigma(\mathbf{W}_o \cdot \boldsymbol{v}_j + \mathbf{b}_o) \\
\boldsymbol{r}_j &= \mathbf{f}_j \odot \boldsymbol{r}_{j-1} + \mathbf{i}_j \odot tanh(\mathbf{W}_s \cdot \boldsymbol{v}_j + \mathbf{b}_s) \\
\mathbf{l}_j &= \mathbf{o}_j \odot tanh(\boldsymbol{r}_j).
\end{aligned}$$

Here $\boldsymbol{r}_j$ is the hidden state vector of the jth recurrent unit. After we get the output of each unite $\mathbf{l}_j$, a fully connected layer with sigmoid activation function predicts the hazard rate as is described in Eqs. (4) and (8) of our main paper as

$$h_j = \sigma(\mathbf{W}_h \cdot \mathbf{l}_j + \mathbf{b}_h).$$

Then we may calculate the event probability over time $p(z)$ w.r.t. the true event time z and the survival rate $S(t)$ at the given time t as Eqs. (10) and (9) of our main paper, respectively. More details can be referred to our published code.

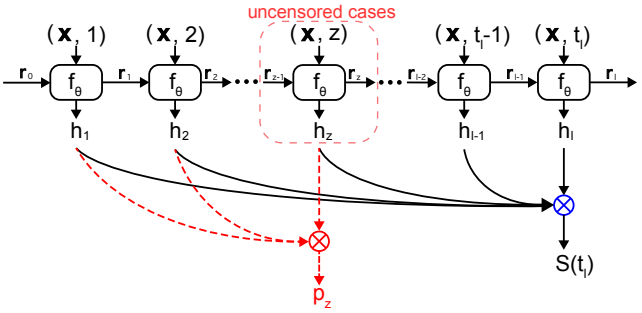


Figure 1: Detailed illustration of Deep Recurrent Survival Analysis (DRSA) model. Note that only the uncensored logs have the true event time and can calculate p_z for the loss L_z. The calculation of p_z and $S(t)$ have been derived in Eqs. (11) and (9) in our main paper, respectively.

Model Efficiency

Here we analyze the computational complexity of our DRSA model. As is shown in Eq. (8), each recurrent unit $f_{\boldsymbol{\theta}}$ takes $(\boldsymbol{x}, t_l, \boldsymbol{r}_{l-1})$ as input and outputs probability scalar h_l and hidden vector $\boldsymbol{r}_l$ to the next unit. Recall that the maximal time interval is L, so the calculation of the recurrent units will runs for maximal L times. We assume the average case time performance of recurrent units $f_{\boldsymbol{\theta}}$ is $O(C)$, which is related to the implementation of the unit (Zhang et al. 2016), e.g., recurrent depth, recurrent skip coefficients, yet can be parallelized through GPU processor. The subsequent calculation is to obtain the multiplication results of h_l or $(1 - h_l)$ to get the results of $p(z)$ and $S(t)$, as that in Figure 1 whose complexity is $O(L)$. Thus the overall time complexity of DRSA model is $O(CL)+O(L) = O(CL)$, which is the same as the original recurrent neural network model. We may also optimize the implementation of the RNN unit through other techniques, such as Quasi-RNN (Bradbury et al. 2017). In many literatures, recurrent neural networks have been deployed in online recommender system (Wu et al. 2016a), online advertising (Zhang et al. 2014) and machine translation system of Google (Wu et al. 2016b), which shows promising time efficiency in large scale online systems and, to some extent, guarantees online inference efficiency for our DRSA model.

Data Statistics of BIDDING

Note that, the dataset of BIDDING used in our experiment contains several subsets, which has been discussed in (Zhang, Yuan, and Wang 2014; Liao et al. 2014). The detailed statistics are provided in Table 1.

Table 1: The statistics of the BIDDING dataset. #: number; AET: averaged event time.

Dataset	Total #	Censored Data #	Censored Rate	AET ($\mathbb{D}_{\text{full}}$)	AET ($\mathbb{D}_{\text{uncensored}}$)	AET ($\mathbb{D}_{\text{censored}}$)	Feature #
1458	3697694	1116644	0.302	69.6696	27.4265	87.9452	12
2259	1252753	396283	0.3163	96.7888	27.1986	128.9877	12
2261	1031479	321931	0.3121	87.6479	18.9	118.8396	12
2821	1984525	228833	0.1153	93.8962	13.2118	104.4125	12
2997	468500	70747	0.151	60.4188	7.2762	69.8711	12
3358	2043032	315010	0.1542	95.4967	21.254	109.0308	12
3386	3393223	819447	0.2415	78.0327	23.8983	95.2682	12
3427	3130560	654989	0.2092	81.965	25.2118	96.9808	12
3476	2494208	723847	0.2902	80.0719	31.2218	100.0453	12
Overall	19,495,974	14,848,243	0.7616	82.0744	25.0484	99.9244	12

Detailed Performance Results of BIDDING

Recall that there are nine independent subsets in BIDDING dataset so that we provide the overall statistics in this table and present the details in Tables 2 and 3.

Table 2: C-index performance on BIDDING dataset, (* indicates p-value $< 10^{-6}$ in significance test).

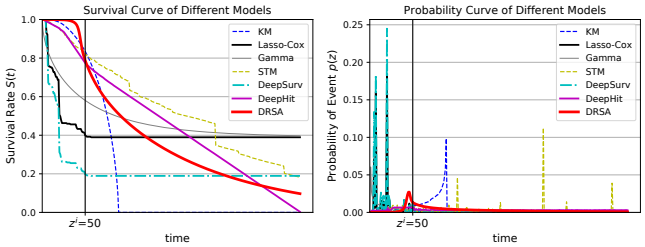
	C-index							
Subset	KM	Lasso-Cox	DeepSurv	Gamma	MTLSA	STM	DeepHit	DRSA
1458	0.698	0.820	0.835	0.698	0.505	0.764	0.861	**0.904***
2259	0.685	0.775	0.791	0.685	0.505	0.768	0.785	**0.876***
2261	0.666	0.847	0.890	0.666	0.508	0.812	0.838	**0.929***
2821	0.677	0.741	0.714	0.678	0.507	0.790	0.810	**0.881***
2997	0.734	0.910	0.852	0.734	0.517	0.835	0.907	**0.919***
3358	0.704	0.866	0.896	0.706	0.542	0.811	0.888	**0.944***
3386	0.716	0.845	0.854	0.719	0.512	0.849	0.881	**0.923***
3427	0.724	0.830	0.845	0.742	0.508	0.798	0.873	**0.901***
3476	0.692	0.865	0.877	0.692	0.505	0.830	0.879	**0.922***
Overall	0.700	0.834	0.840	0.703	0.513	0.807	0.858	**0.911***

Table 3: ANLP performance on BIDDING dataset, (* indicates p-value $< 10^{-6}$ in significance test).

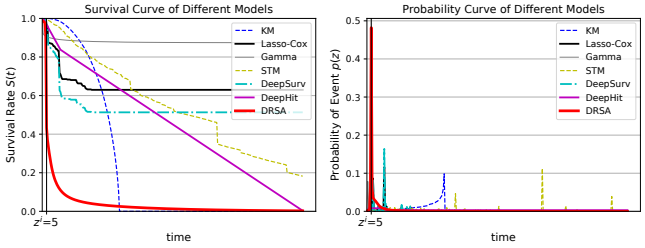
	ANLP							
Subset	KM	Lasso-Cox	DeepSurv	Gamma	MTLSA	STM	DeepHit	DRSA
1458	10.532	38.608	38.652	5.956	9.791	4.761	5.510	**4.088***
2259	14.671	28.234	29.658	6.069	10.248	5.471	5.586	**5.244***
2261	14.665	39.129	39.390	5.986	10.261	4.818	5.442	**4.632***
2821	19.582	43.099	43.072	7.838	9.895	5.572	5.614	**5.428***
2997	16.203	32.849	33.052	5.999	9.167	5.083	5.470	**4.504***
3358	19.253	44.769	44.885	6.736	9.484	5.539	5.616	**5.281***
3386	15.973	39.781	41.943	6.488	8.834	5.228	5.549	**4.940***
3427	16.902	41.558	41.698	6.002	9.090	5.321	5.552	**4.836***
3476	10.507	39.551	39.518	5.710	10.240	4.537	5.554	**4.012***
Overall	15.366	38.620	39.096	6.310	9.668	5.148	5.544	**4.774***

Comprehensive Analysis

Visualization of Forecasted Results We visualize more results of model forecasting for different samples. In Figure 2, we illustrate two forecasting results from each model. The outcome agrees the analysis in our main paper that, our DRSA model accurately placed the highest probability on the true event time z^i, which explains the result of C-index and ANLP metric.



(a) The first sample.



(b) The second sample.

Figure 2: More examples of the forecasting results from the compared models.

Model Convergence on BIDDING Dataset

Recall that our model optimizes over two loss functions, i.e., the ANLP loss L_z and the cross entropy loss L_c. We apply a method (Cao et al. 2018) whose idea is to feed the batch of training data under each one of the two losses alternatively. From the learning curves we can find that (i) DRSA converges quickly and the values of both loss function drop to stable convergence at about the first complete iteration over the whole training dataset. (ii) The two losses are alternatively optimizing and facilitate each other during the training, which proves the learning stability of our model.

Influence of Bucketization

In this additional experiment, we retrain our DRSA model under different bucketization size settings. Specifically, when the bucketized interval size increases, the output survival rate $S(t)$ for $t \in ((l-1) * s_{\text{intv}}, l * s_{\text{intv}})$ will be linearly interpolated to get the approximating value where l is the interval index number. Recall that we implement in our experimental setup as $s_{\text{intv}} = 1$ which conforms to the nature of the integer valued data. From Figure 4 we may find that the bucketization will not hurt the performance unless s_{intv} increases too large. The performance gets even better for C-index when $s_{\text{intv}} = 2$ in CLINIC and $s_{\text{intv}} = 6$ in MUSIC.

Influence of Bucketization on BIDDING Dataset

From Figure 5, the result reflects that the bucketization based on discrete time model is reasonable without hurting the performance. The C-index drops a little yet beats all the baselines when carefully enlarge the bucketization interval size, while ANLP gets lower (better) when the interval size is larger and achieves best when $s_{\text{intv}} = 6$.

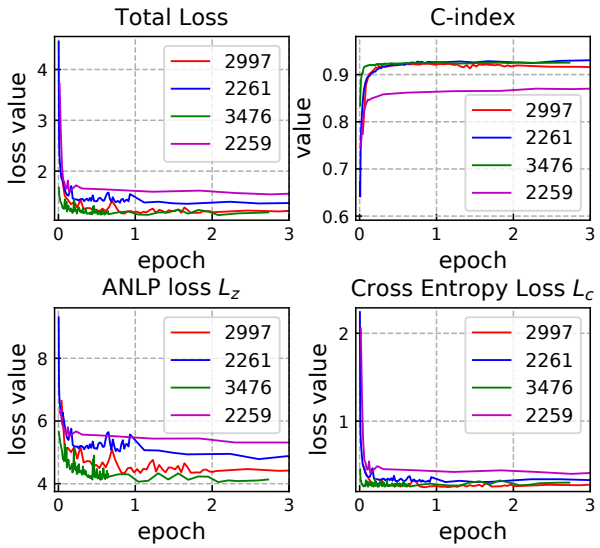


Figure 3: Learning curves of BIDDING subdatasets. Here "epoch" means one iteration over the whole training data and $\alpha = 0.25$ in Eq. (16).

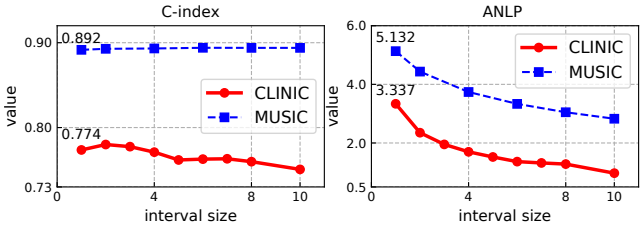


Figure 4: The performance comparison over CLINIC and MUSIC w.r.t. varying bucketized interval sizes. We put the results over BIDDING in the supplemental material.

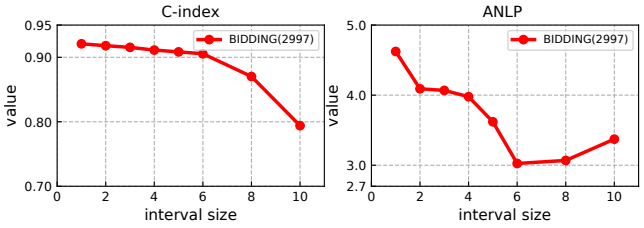


Figure 5: The performance comparison over the subset of BIDDING w.r.t. varying bucketized interval sizes.

References

Bradbury, J.; Merity, S.; Xiong, C.; and Socher, R. 2017. Quasi-recurrent neural networks. *ICLR*.

Cao, X.; Chen, H.; Wang, X.; Zhang, W.; and Yu, Y. 2018. Neural link prediction over aligned networks. In *AAAI*.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation*.

Liao, H.; Peng, L.; Liu, Z.; and Shen, X. 2014. ipinyou global rtb bidding algorithm competition dataset. In *ADKDD*.

Wu, S.; Ren, W.; Yu, C.; Chen, G.; Zhang, D.; and Zhu, J. 2016a. Personal recommendation using deep recurrent neural networks in netease. In *ICDE*.

Wu, Y.; Schuster, M.; Chen, Z.; Le, Q. V.; Norouzi, M.; Macherey, W.; Krikun, M.; Cao, Y.; Gao, Q.; Macherey, K.; et al. 2016b. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Zhang, Y.; Dai, H.; Xu, C.; Feng, J.; Wang, T.; Bian, J.; Wang, B.; and Liu, T.-Y. 2014. Sequential click prediction for sponsored search with recurrent neural networks. *arXiv preprint arXiv:1404.5772*.

Zhang, S.; Wu, Y.; Che, T.; Lin, Z.; Memisevic, R.; Salakhutdinov, R. R.; and Bengio, Y. 2016. Architectural complexity measures of recurrent neural networks. In *NIPS*.

Zhang, W.; Yuan, S.; and Wang, J. 2014. Real-time bidding benchmarking with ipinyou dataset. *arXiv:1407.7073*.