Deep Learning-Survival Analysis

*Training a convolutional neural network to predict time to a (generated) event from MNIST images, using a loss function specific to survival analysis.

Preksha Shah
UMass Lowell.
Machine Learning

I. Introduction

This paper presents the Training a convolutional neural network to predict time to a (generated) event from MNIST images, using a loss function specific to survival analysis. Often in clinical research we aim to estimate the time to and event, which leads to specific techniques that are distinct from regressions and SVM and its termed as survival analysis. This paper will describe how to train a convolutional neural network to predict time to an event from MNIST images, using loss function specific to survival analysis. I will describe the data that I have used and my approach towards solving it and will mentions the requirements for this project as well as the results for better understanding of the work.

II. BASIC TERMS AND QUANTITIES

The basic requirements are the packages to be installed like numpy, matplotlib, pandas, tensorflow, scikit-survival. The objective of survival analysis is to connect covariates and the time of an event. It is similar to regression problem but with little difference, the parts of the training data can only be partially observed. It is modelled to give a continuous non-negative value T. the 2 major functions from which the basic quantities are derived they are the survival function and hazard function. The survival function is a non-increasing function with S(0)=1 and S(∞)=0. The hazard function denotes the approximate probability.

$$h(t) = \lim_{\Delta t \to 0} \frac{P(t \le T < t + \Delta t \mid T \ge t)}{\Delta t} \ge 0$$

III. GENERATE DATA FROM MNIST

We will generate synthetic survival data from MNIST images. I am going to use MNIST dataset and will synthetically generate survival time based on the digit each image represents. Firstly, I assigned each class label to one of the survival time randomly such that few of them correspond to the better one and few to the worse survival. Then I generated risk score which describes how big the risk of experiencing an event is relative to each other. The output I received is: as its clear class labels 2 and 8 belong to risk group 0 and so on.

Out[4]:

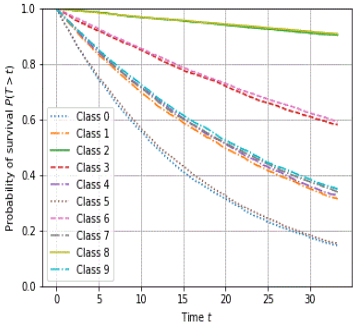
class_label	risk_score	risk_group
0	3.071	3
1	2.555	2
2	0.058	0
3	1.790	1
4	2.515	2
5	3.031	3
6	1.750	1
7	2.475	2
8	0.018	0
9	2.435	2

I followed Bender et al protocol to generate survival times from risk scores and its density function is as follows:

$$f(t|\lambda) = \lambda \exp(-\lambda t), \text{ where } \lambda > 0$$

λ is chosen such that mean survival time is 365 days. And the data is censored randomly, and generated survival data comprised a Boolean event indicator and observed time for each MNIST image. The reason for doing this is to see what risk score mean in terms of survival. The training data is arranged into class labels and estimate the survival function using non-parametric product limit estimator (Kaplan-Meier estimator). The result obtained for 9 classes is as follows:

Out[6]: <matplotlib.legend.Legend at 0x18e6a96a188>



IV. EVALUATING PREDICTIONS

Subjecting training data and test data to censoring is one of the important aspects of survival analysis. The widely used performance measure in censoring the data is Harrell's concordance index. With given

set of observed time and risk score it will check if the ordering of observed time is in concordant with the ordering of risk score. The reason why we cannot generate the perfect result of 1.0 is because the generated survival time are randomly distributed based on risk score and not deterministic function of the risk score.
I took the risk score from which survival time was generated to check how well the model performs if the actual risk score is known.

```
In [7]: cindex = concordance_index_censored(event_test, time_
        test, risk_scores[y_train.shape[0]:])

        print(f"Concordance index on test data with actual ri
        sk scores: {cindex[0]:.3f}")

        Concordance index on test data with actual risk score
        s: 0.705
```

For non-linear survival analysis with neural networks, Cox's proportional hazard model is one of the best, It models the hazard function h(ti) of the i-th subject, conditional on the feature vector xi∈Rp, as the product of an unspecified baseline hazard function h0 (more on that later) and an exponential function of the linear model xTiβ:

$$h(t|x_{i1},\ldots,x_{ip}) = h_0(t)\exp\left(\sum_{j=1}^{p} x_{ij}\beta_j\right) \Leftrightarrow \log\frac{h(t|\mathbf{x}_i)}{h_0(t)} = \mathbf{x}_i^\top\beta,$$

IV. COMPUTING LOSS FUNCTION

I want to sort the data once in descending order by survival time and then in ascendingly update the inner summation of the Cox's PH model which would lead to the linear complexity to compute the loss. I feel like the risk set for the subject with the smallest uncensored survival time would be over the whole dataset which I feel like might be a problem. I think it is kind of impossible to store the whole dataset in the GPU memory, in case if we divide the dataset into parts but it might not be possible to compute the exact loss as all the risk sets may not be accessed and I feel the solution for this could be if the divided parts can be sorted by observed time, instead of the whole dataset.

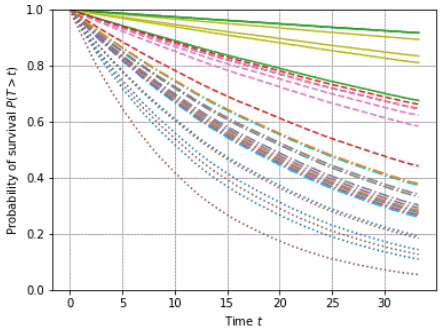
V. CREATING CNN FOR SURVIVAL ANALYSIS

The next step is to create convolutional neural network for survival analysis to learn a high-level representation for MNIST digits such that we can estimate each image's survival function. The CNN follows the LeNet architecture where the last linear has one output unit that corresponds to the predicted risk score. The predicted risk score, together with the binary event indicator and risk set, are the input to the Cox PH loss. In order to observe training, we start with TensorBoard.
Couple of predictions that I could make were:

1. The final concordance index on the validation data is close to the optimal value we computed above using the actual underlying risk scores.
2. The loss during training is quite volatile, which stems from the small batch size (64) and the varying number of uncensored samples that contribute to the loss in each batch. Increasing the batch size should yield smoother loss curves.

VI. PREDICTING SURVIVAL FUNCTIONS

I feel it would be much simpler if we just pass the batch of images and record the predicted risk score. To estimate individual survival function, I need to estimate baseline hazard function h0, which Is done by analogous to the linear Cox PH model and I did it by using breslow's estimator.



Solid lines correspond to images that belong to risk group 0 (with lowest risk), which the model was able to learn. Samples from the group with the highest risk are shown as dotted lines. Their predicted survival functions have the steepest descent, confirming that the model correctly identified different risk groups from images.

VII. CONCLUSION

I tried to build, train, and evaluated a convolutional neural network for survival analysis on MNIST. While MNIST is obviously not a clinical dataset, the exact same approach can be used for clinical data. For instance, Mobadersany et al. used the same approach to predict overall survival of patients diagnosed with brain tumors from microscopic images, and Zhu et al. applied CNNs to predict survival of lung cancer patients from pathological images.

CITATIONS:

Sebastian Polsterl, *Survival Analysis for Deep Learning*
k-d-w.org: 2019 blog

Bo Tang; Ao Li; Bin Li; Minghui Wang, *capsule network for survival analysis*
IEEE: 2019 paper

Sundar V, *Deep learning for Survival Analysis*
Towardsdatascience.com: 2019 blog

Hongming Li; Yinf Xiao; Haoyu Zhong and Yong Fan. *Deep convolutional neural network for imaging data based survival analysis of rectal cancer.*
NCBI: 2019 paper

Yucheng Zhang; Steven Gallinger. *MNIST dataset construction*
Researchgate.com: 2019 blog.