Leveraging Neural Networks as Kernels for Survival Analysis

James Brofos†, Rui Shu, Frank Zhang†, Matthew Jin, and Michael Downs†

The MITRE Corporation, Stanford University, and Dartmouth College

Abstract

We introduce a novel deep neural network as a kernel function for transforming right-censored survival data. We use Bayesian optimization to produce a neural network architecture, which performs best on out-of-sample data using benchmark survival analysis data sets. We demonstrate that by employing a neural network kernel, the concordance index may be improved over that obtained with standard Cox proportional hazards analysis.

Introduction

Survival analysis treats the problem of predicting the time until a specified event occurs in an individual. Neural networks have presented an effective means of learning useful representations of raw input data [1]. In this work, we consider learning a neural network representation that is then provided as input to a Cox model for survival analysis *post-hoc*.

Let $(\mathbf{X}, \mathbf{y}, \boldsymbol{\delta})$ denote a survival analysis data set, where $\mathbf{X}_i$ denotes the raw features for the i^{th} record, y_i is that record's time of event, and δ_i is a binary censoring variable. A common procedure in survival analysis is to apply a Cox proportional hazards model to the input features [2], which can be estimated by maximizing the partial log-likelihood function [3]. Our survival models are evaluated according to their concordance index, which we compute by following the method in [2].

Implementation Details

We learn a set of basis functions $\{\phi_1, \ldots, \phi_k\}$ that correspond to each of the k activations in the last hidden layer of a neural network. The network is trained on survival data $(\mathbf{X}, \mathbf{y}, \boldsymbol{\delta})$ by maximizing the partial log-likelihood function of a Cox model. The basis functions are parametrized by the weights of the neural network and are learned by stochastic gradient ascent. We use the software library Theano [4, 5] to compute the gradient of the partial log-likelihood.

Denoting $\boldsymbol{\Phi}_i = \left(\phi_1\left(\mathbf{X}_i\right), \ldots, \phi_k\left(\mathbf{X}_i\right)\right)$ to be the k-dimensional learned kernel representation of the input features for the i^{th} record, we provide $(\boldsymbol{\Phi}, \mathbf{y}, \boldsymbol{\delta})$ as input to a Cox model *post-hoc*, the parameters of which are estimated through Newton-Raphson iterations on the partial log-likelihood. In particular, we maximize,

$$\ell\left(\boldsymbol{\beta}\right) = \sum_{i:\delta_i=1} \left(\boldsymbol{\Phi}_i'\boldsymbol{\beta} - \log \sum_{j:y_j>y_i} \exp\left(\boldsymbol{\Phi}_j'\boldsymbol{\beta}\right)\right),$$

with respect to all the weights in the neural network that permit us to parameterize the basis functions.

Hyperparameter Selection

We use Bayesian optimization [6] to tune the hyperparameters of the neural network using a squared exponential kernel function with the expected improvement acquisition function to maximize the concordance. We optimize across the number of hidden layers, the number of hidden units in each layer, the learning rate, the number of training epochs and the batch size used during training.

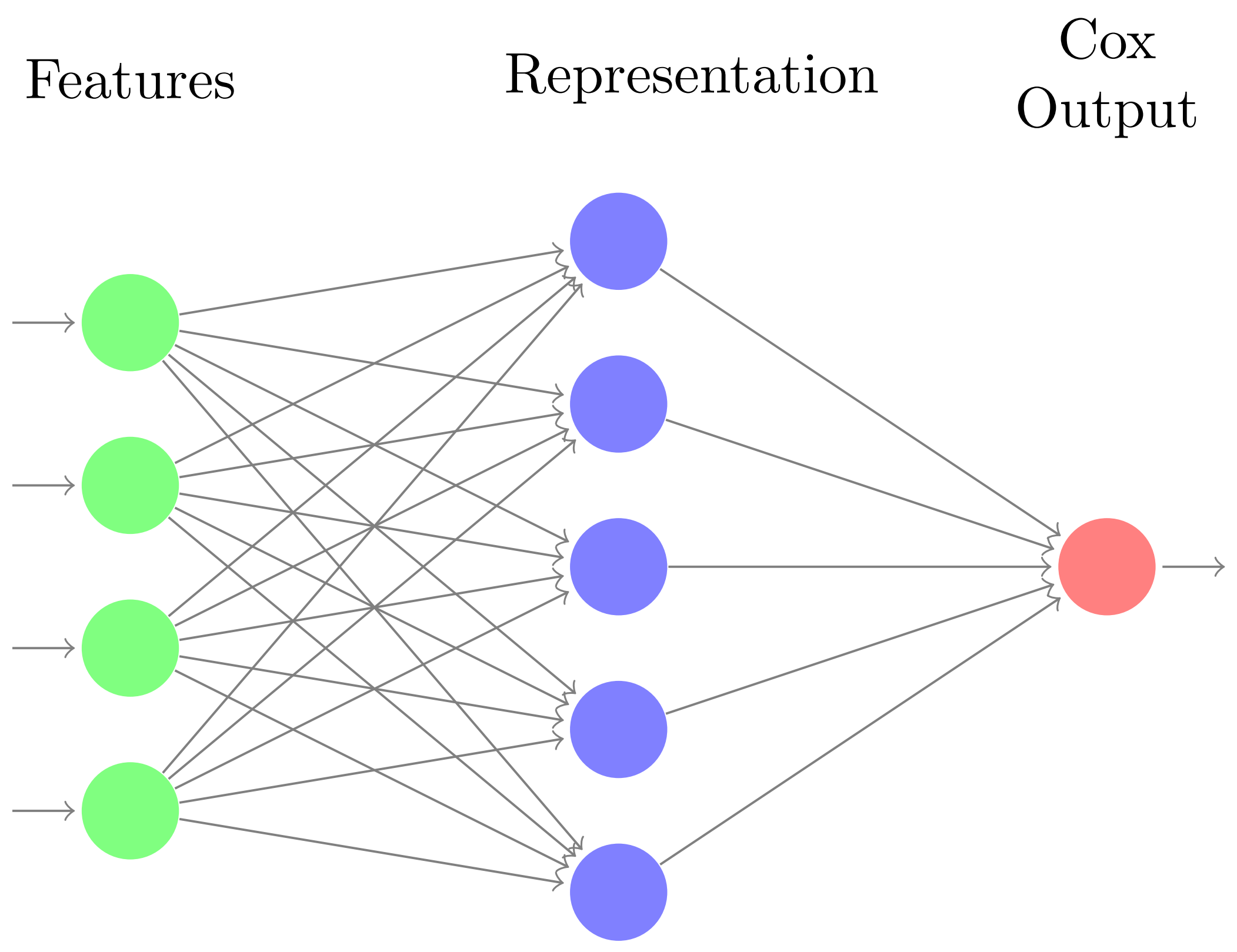

Figure 1: Here we show a smaller-scale version of the architecture used in our experiments. The central idea is to learn a useful representation of the survival data in the final hidden layer of a neural network, which can then be provided as features to a Cox proportional hazards model. In our work, we use hyperbolic tangent activation functions and a momentum-based gradient ascent algorithm.

In total, there are 45,900 possible configurations of hyperparameters. In our Bayesian optimization procedure, we assume a quadratic prior mean that reflects our belief that the optimal configuration lies on the interior of the parameter space [7].

Numerical Experiments

We partition the data set into training and testing components, where ten randomly selected records comprise the test set. A Cox model and a neural network kernel are then estimated and the concordance index is computed on the test data.

This process is repeated ten times and the average concordance is computed and compared with the benchmark. Notice that this experimental setup permits the concordance between the network kernel and the Cox model to be directly compared via a paired t-test. Results of these experiments are shown in Table 1 and Figure 2.

Experiment	Cox	NN Kernel	p-value
Veteran	61.2%	**69.2%**	0.0048
Lung	55.0%	**64.4%**	0.0095
Rats	58.6%	**63.6%**	0.0252

Table 1: We evaluate our neural network kernel algorithm versus the Cox model on benchmark survival analysis data sets using the concordance index as a metric. Each algorithm was reproduced five times with the same hyperparameter configuration but varying training and testing sets. We display the p-value of the paired t-test against the null hypothesis that the concordances of the network kernel and the Cox model are pairwise equal.

Conclusion

This work proposed to use a neural network with a Cox partial log-likelihood objective function as a kernel to construct features for survival analysis. We apply a Bayesian optimization procedure to identify the network architecture with the optimal hyperparameter configuration. We find that by leveraging a deep network's representation, the out-of-sample concordance index can be improved by several percentage points relative to the state-of-the-practice.

[†] These authors' affiliation with The MITRE Corporation is provided for identification purposes only, and is not intended to convey or imply MITRE's concurrence with, or support for, the positions, opinions or viewpoints expressed by the author.

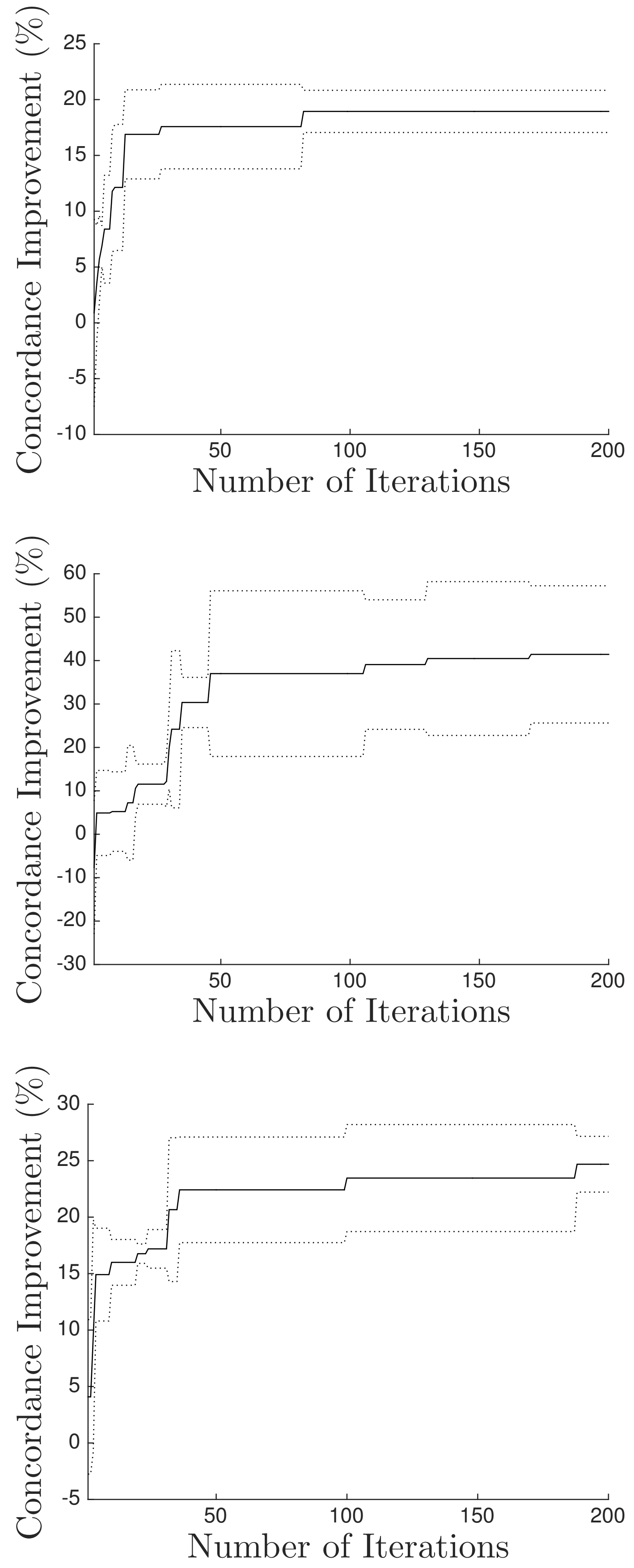

Figure 2: We show the behavior of the Bayesian optimization procedure for tuning the hyperparameters of the neural network kernel. We plot the average maximum percentage increase in the concordance index as well as the 95% confidence interval. We show performance for veteran lung cancer data [8], NCCTG's lung cancer data [9], and tumor incidence in rats [10].

References

[1] Bengio, Y., Mesnil, G., Dauphin, Y. & Rifai, S. (2012). Better Mixing via Deep Representations. *ICML'2013.*

[2] Ishwaran, H., Kogalur, U.B., Blackstone, E.H. (2008) *et al.* Random survival forests. Ann Appl Statist, 2, pp. 841âĂŞ860.

[3] Cox, D. R. & Oakes, D. (1984). Analysis of Survival Data. New York: Chapman & Hall. ISBN 041224490X.

[4] Bastien, F., Lamblin, P., Pascanu, R., Bergstra, J., Goodfellow, I., Bergeron, A., Bouchard, N., Warde-Farley, D. & Bengio, Y. (2012) Theano: new features and speed improvements. *NIPS deep learning workshop.*

[5] Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D. & Bengio, Y. (2010) Theano: A CPU and GPU Math Expression Compiler. *Proceedings of the Python for Scientific Computing Conference (SciPy).*

[6] Snoek, J., Larochelle, H. & Adams, R.P. (2012) Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems (NIPS).*

[7] Snoek, J., Rippel, O., Swersky, K., Kiros, R., Satish, N., Sundaram, N., Patwary, M.M.A., Prabhat, and Adams, R.P. (2013) Scalable Bayesian Optimization Using Deep Neural Networks. *arXiv:1502.05700 [stat.ML]* URL `http://arxiv.org/abs/1502.05700`.

[8] Kalbfleisch, D. & Prentice, R.L. (1980) The Statistical Analysis of Failure Time Data. Wiley, New York.

[9] Loprinzi, C.L., Laurie, J.A., Wieand, H.S., Krook, J.E., Novotny, P.J., Kugler, J.W., Bartel, J., Law, M., Bateman, M., Klatt, N.E. *et al.* (1994) Prospective evaluation of prognostic variables from patient-completed questionnaires. In *Journal of Clinical Oncology.*

[10] Lee, E.W., Wei, L.J., & Amato, D. (1992) Cox-type regression analysis for large number of small groups of correlated failure time observations. In *Survival Analysis, State of the Art.*