Survival Kernets: Scalable and Interpretable Deep Kernel Survival Analysis with an Accuracy Guarantee

Carnegie Mellon University Heinz College
INFORMATION SYSTEMS • PUBLIC POLICY • MANAGEMENT

George H. Chen (georgechen@cmu.edu)

Code: https://github.com/georgehc/survival-kernets

Introduction

We propose a new class of neural survival models ("survival kernets"):

- Based on learning a *similarity score* between any two data points (called a *kernel function*)
- Key features:
  - Achieves prediction accuracy competitive with existing state-of-the-art
  - Has a finite-sample accuracy guarantee (for a special case)
  - Represents each point as a combination of "exemplar" training points ⇒ Helpful for model interpretation
  - Scales to large datasets (# of exemplars can be tuned)

No other survival analysis approach has all of these features

Existing interpretable survival models: (can work poorly if assumptions don't hold)

- some models assume linearity and/or survival curve shape constraints [Cox 1972, Prentice & Kalbfleisch 1979, Aalen 1980, Simon et al 2011, …]
- decision trees with few leaves [Ishwaran et al 2008, Bertsimas et al 2022, …]
- survival-supervised clustering [Chapfuwa et al 2020, Nagpal et al 2021]
- survival-supervised topic models [Dawson & Kendziorski 2012, Chen et al 2024]

(no accuracy guarantees; some scale poorly with dataset size)

Background

Survival Data

Example tabular dataset

Feature vector					Observed time
Age (years)	Sex	Diabetic	Temp. (C)	# comorbidities	Time until death (days)
92.7	female	yes	36.0	1	719
78.0	male	no	38.7	1	969
35.5	female	no	39.5	2	≥ 796

(≥ 796: when we stop collecting training data, not everyone has died)

Training data (i.i.d.): $\{(X_i, Y_i, D_i)\}_{i=1}^n$

- X_i: raw input (e.g., feature vector)
- Y_i: observed time ≥ 0
- D_i: event indicator — $1 \Rightarrow Y_i$ is a survival time; $0 \Rightarrow Y_i$ is a censoring time

Prediction for Test Point x with the Conditional Kaplan-Meier Estimator [Beran 1981]

1. Find all unique observed times in which someone died in training data

$$0 < t_1 < t_2 < \cdots < t_m$$

(m = # unique times of death)

2. Build table below with the help of a kernel function (e.g., $\mathbb{K}(x, X_i) = e^{-\|x - X_i\|^2}$)

Time	t_1	t_2	t_3	$\cdots$	t_m
# deaths among training points who look like x	$d_1(x)$	$d_2(x)$	$d_3(x)$	$\cdots$	$d_m(x)$
# training points still alive among those who look like x	$r_1(x)$	$r_2(x)$	$r_3(x)$	$\cdots$	$r_m(x)$

$$d_j(x) = \sum_{i=1}^{n} \mathbb{K}(x, X_i) D_i \mathbb{1}\{Y_i = t_j\}, \qquad r_j(x) = \sum_{i=1}^{n} \mathbb{K}(x, X_i) \mathbb{1}\{Y_i \geq t_j\}$$

3. Predict survival curve for test point x (across time $t \geq 0$):

$$\mathbb{P}(\text{survive beyond time } t | x) \approx \prod_{j=1,\dots,m \text{ s.t. } t_j \leq t} \left(1 - \frac{d_j(x)}{r_j(x)}\right)$$

(this is a function of time t (monotonically decreases starting from 1 at time $t = 0$))

Survival Kernels

Key high-level ideas

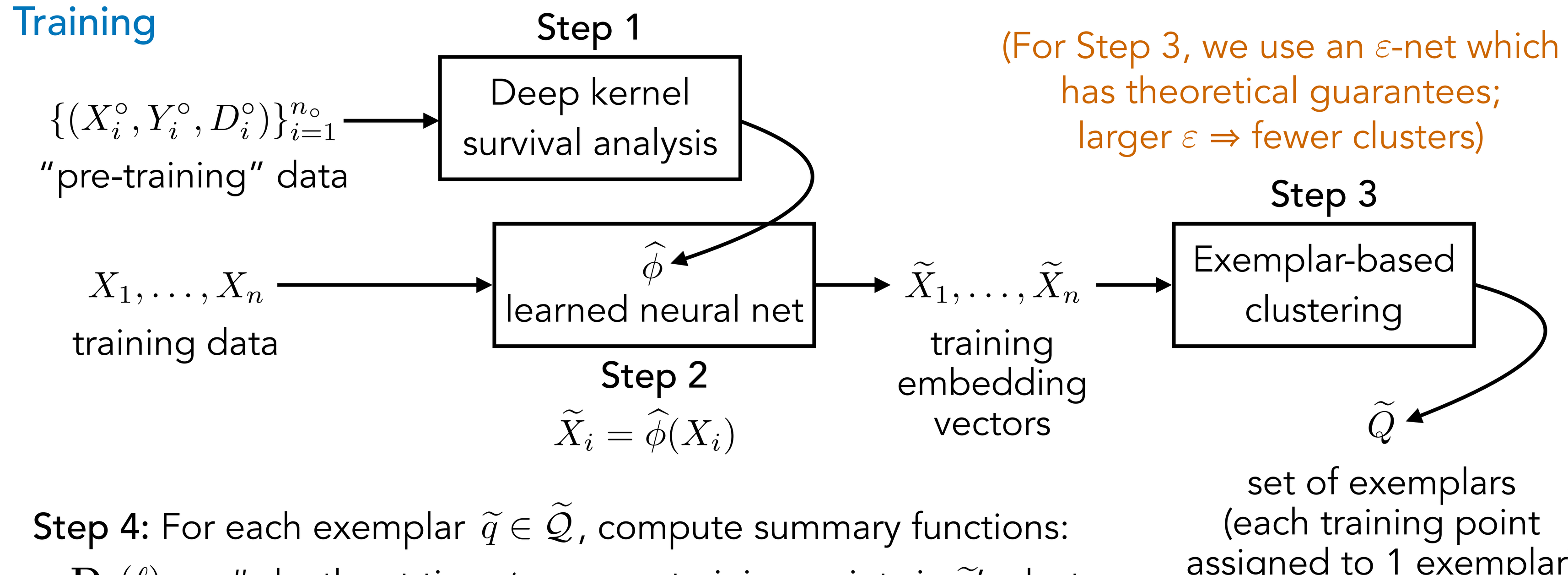
- Use the conditional Kaplan-Meier estimator, where we automatically learn the kernel function using *deep kernel survival analysis* [Chen 2020]

$$\mathbb{K}(x, X_i) = \exp(-\|\phi(x) - \phi(X_i)\|^2)$$

(ϕ = user-specified base neural net)

- At test time, to avoid computing the similarity between a test point and every training point, "compress" the training data using *kernel netting* [Kpotufe & Verma 2017]. This helps with model interpretation!
- To get theoretical analysis to work out, use sample splitting:
  - $\{(X_i^\circ, Y_i^\circ, D_i^\circ)\}_{i=1}^{n_\circ}$ "pre-training" data — for base neural net training
  - $\{(X_i, Y_i, D_i)\}_{i=1}^{n}$ "training" data — for constructing test-time predictor

Training

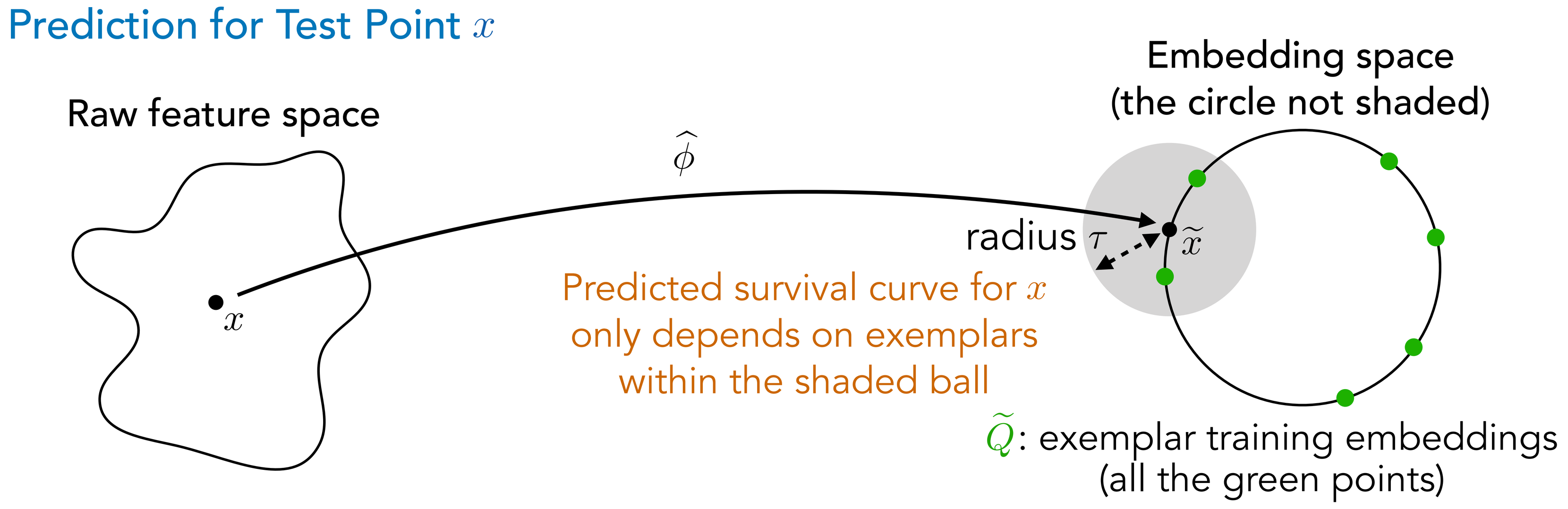


Step 4: For each exemplar $\widetilde{q} \in \widetilde{\mathcal{Q}}$, compute summary functions:

$\mathbf{D}_{\widetilde{q}}(\ell) :=$ # deaths at time t_ℓ among training points in $\widetilde{q}$'s cluster

$\mathbf{R}_{\widetilde{q}}(\ell) :=$ # still alive at time t_ℓ among training points in $\widetilde{q}$'s cluster

Prediction for Test Point x



Prediction uses the same equation as the conditional Kaplan-Meier estimator except:

$$d_j(x) = \sum_{\widetilde{q} \in \widetilde{\mathcal{Q}} \text{ s.t. } \|\widetilde{q} - \widehat{\phi}(x)\| \leq \tau} \mathbb{K}(x, \text{raw representation of } \widetilde{q}) \mathbf{D}_{\widetilde{q}}(j)$$

$$r_j(x) = \sum_{\widetilde{q} \in \widetilde{\mathcal{Q}} \text{ s.t. } \|\widetilde{q} - \widehat{\phi}(x)\| \leq \tau} \mathbb{K}(x, \text{raw representation of } \widetilde{q}) \mathbf{R}_{\widetilde{q}}(j)$$

Implementation Remarks (See Paper for Details)

- We show how XGBoost [Chen & Guestrin 2016] can be used to initialize survival kernet training, outperforming standard neural net random parameter initialization
- Two modifications improve prediction accuracy but we lack theory to explain these: (i) set pre-training data = training data, (ii) fine-tune exemplar summary functions

Theory

Assumptions on neural net's output space (aim to predict well up to time horizon t_{horizon}):

- Distribution of embedding vectors has compact support and low "intrinsic dimension" d' ⇒ contrastive learning can help achieve this [Wang & Isola 2020, Liu et al 2021]
- $\mathbb{P}(Y_i > t_{\text{horizon}} | \widehat{\phi}(X_i)) \geq$ positive constant ⇒ so we see enough data up to the time horizon
- Pdfs of survival time given embedding vector & censoring time given embedding vector are Lipschitz continuous, & censoring cannot almost surely happen for any embedding vector ⇒ close by embedding vectors have similar survival times (also similar censoring times)

Set $\varepsilon = \widetilde{O}(\tau n^{-1/(2+d')})$

Then: $\mathbb{E}\left[\frac{1}{t_{\text{horizon}}} \int_0^{t_{\text{horizon}}} \left(\widehat{S}(t|X) - S(t|X)\right)^2 dt\right] \leq \widetilde{O}(n^{-2/(2+d')})$ — optimal up to log factor [Chagny & Roche 2014]

Experiments

Prediction Accuracy Benchmark

- We use standard datasets that are sufficiently large
- For every method, hold out 20% of training data to treat as validation set for hyperparameter tuning
- Evaluation metric: time-dependent concordance index [Antolini et al 2015] (higher is better)

(Rotterdam/GBSG: train on Rotterdam, test on GBSG; SUPPORT, UNOS, KKBOX: 70%/30% train/test split)

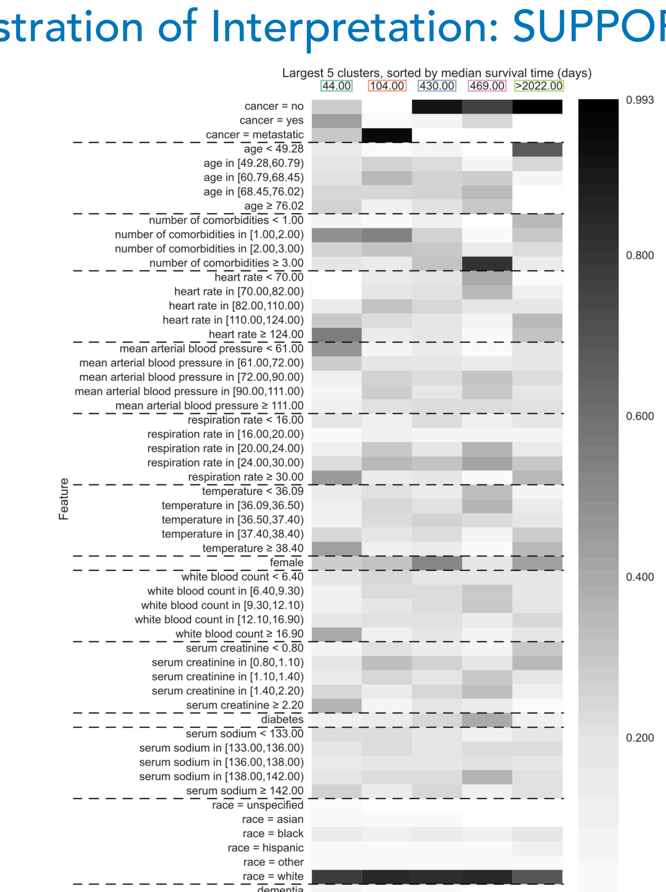
	Rotterdam/GBSG (n=2232, d=7)	SUPPORT (n=8873, d=14)	UNOS (n=62644, d=49)	KKBOX (n=2814735, d=15)
Elastic-net Cox [Simon et al 2011]	0.6660 ± 0.0045	0.6046 ± 0.0013	0.5931 ± 0.0011	0.8438 ± 0.0001
XGBoost [Chen & Guestrin 2016]	0.6703 ± 0.0128	0.6281 ± 0.0031	0.6028 ± 0.0009	0.8714 ± 0.0000
DeepSurv [Katzman et al 2018]	**0.6850** ± 0.0160	0.6155 ± 0.0032	0.5941 ± 0.0021	0.8692 ± 0.0003
DeepHit [Lee et al 2018]	0.6792 ± 0.0121	0.6354 ± 0.0047	0.6170 ± 0.0016	**0.9148** ± 0.0001
Deep Cox Mixtures [Nagpal et al 2021]	0.6763 ± 0.0104	0.6289 ± 0.0047	0.6101 ± 0.0023	0.8830*
Survival kernet (version explained by theory)	0.6510 ± 0.0212	0.6220 ± 0.0026	0.6028 ± 0.0032	0.8952 ± 0.0002
Survival kernet (with the 2 modifications)	0.6719 ± 0.0135	**0.6426** ± 0.0045	**0.6211** ± 0.0025	0.9057 ± 0.0003

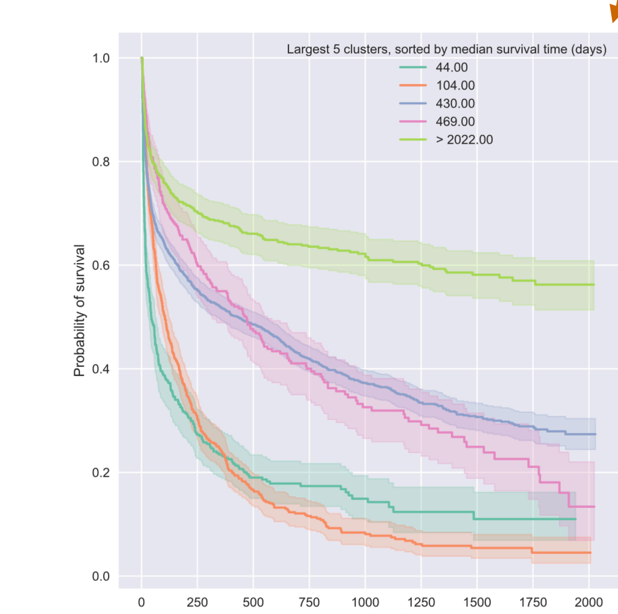
Mean ± std dev over 5 experimental repeats
(* only ran once due to excessive computation time)

More detailed results including on computation time are in the paper

Illustration of Interpretation: SUPPORT Dataset

- Choose which exemplars/clusters to focus on (e.g., largest ones, ones that contribute to the prediction for a specific test point)
- Plot feature heat map & survival curves





- Columns: different exemplars/clusters
- Rows: raw features
- Intensity values: fraction of people in an exemplar's cluster with a particular raw feature

Survival curves are interpreted in a standard manner (this is precisely a Kaplan-Meier plot)

The paper includes visualizations for all datasets along with interpretations

This work was supported in part by NSF CAREER award #2047981, and by Health Resources and Services Administration contract 234-2005-370011C