Multivariate Arrival Times with Recurrent Neural Networks for Personalized Demand Forecasting

Tianle Chen [1] Brian Keng [2] Javier Moreno [2]

[1] University of Toronto
tianle@utstat.utoronto.ca

[2] Rubikloud Technologies Inc.
{brian.keng, javier.moreno}@rubikloud.com

Overview

Personalized Demand Forecasting

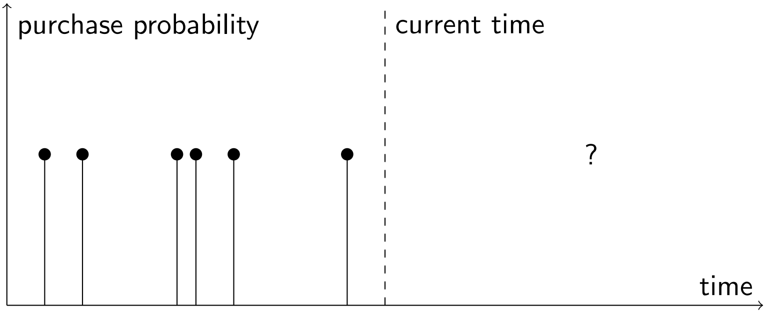


Figure: Forecasting next purchase

Personalized Demand Forecasting

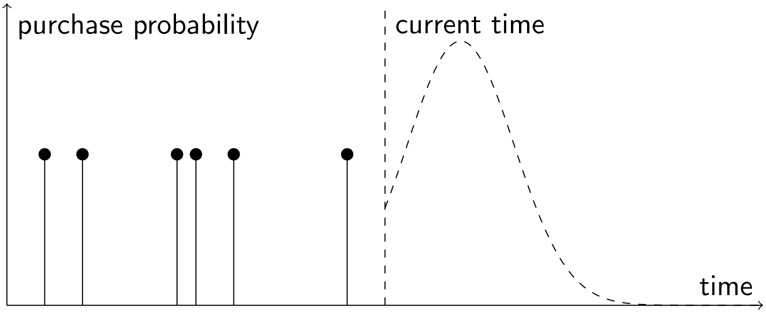


Figure: Forecasting next purchase

Personalized Demand Forecasting

Want:

- For each customer, each product, predict time of next purchase.
- Replenishable products are important driver of revenue.

Some problems:

- Cross-product and sequential dependencies
- Heterogeneous population
  - Different demographics may motivate similar purchase patterns for some products.
- Sparse observations of actual purchases
  - High category-level purchases doesn't relate to high product-level purchases

Cox Processes

Models arrival counts under time-dependent intensity in the case where there are many arrivals, typically with several arrivals per unit time. Suppose in $[t, t+1]$, the arrival intensity is λ_t, then the number of arrivals ($k \in \mathbb{N}_0$) during that time is:

$$N_t \sim \texttt{Poisson}(\lambda_t); \;\; P(N_t = k) \propto \lambda_t^k / k! \tag{1}$$

Typically, $\lambda_t = \beta^\top X$ so that the optimal β can be found through a maximum-likelihood approach.

Survival Models

Models single arrival time (i.e. Time of Death) in the case where most observations only involve a lower limit of arrival time (i.e. Age). Likelihood in the case where person is still alive is

$$P(\text{Time-of-Death} > \text{age}) \tag{2}$$

Modeling survival usually involves a hazard rate, i.e. the instantaneous rate of death, where $h_i(t) = (\beta^\top X_i)h(t)$ for some base hazard rate.

$$h(t) = \lim_{h \to 0} \frac{P(\text{Time-of-Death} \in (t, t+h])}{P(\text{Time-of-Death} > t)} \tag{3}$$

Linearity

Current applications to Cox Processes and Survival Models require analytical gradients and this requires linearity in covariates.

- Cox Process:

$$\lambda_t = \beta^\top X$$

- Survival Models:

$$h_i(t) = (\beta^\top X_i)h(t)$$

Machine Learning Approaches to Arrival Processes

- Binary Prediction with Trees and Ensemble Methods: Split training period into 2 periods, predict if Customer buys.

$$\hat{\theta} = \text{argmin}_{\theta}\text{loss}\ \left(p_{\texttt{[t1:t2]}}, f(X_{\texttt{[t0:t1]}}, \theta)\right)$$

$$p_{\texttt{[t2:t3]}} = f(X_{\texttt{[t0:t2]}}, \hat{\theta})$$

- Sequence-to-Sequence Prediction with RNNs.

RNN Approaches to Arrival Processes

- Distance Minimization with RNNs: Estimate next time of purchases.

$$\hat{\theta} = \text{argmin}_{\theta} \sum_{t} d(T - t, f(X_t, \theta))$$

- Next Period Binary Predictions: Estimate probability of buying during the next time period.

$$\hat{\theta} = \text{argmin}_{\theta} \sum_{t} loss(p_{[t,t+h]}, f(X_t, \theta))$$

- Survival Modeling with RNNs: Replace linear multiplier to hazard rate with output of neural network

$$h_i(t) = f(X, \theta) h(t);\ \hat{\theta} = \theta_{\text{MLE}}$$

MAT-RNN

- RNN:
  - Non-linear joint predictions ✓
  - Non-linear time dependence ✓
- Survival-based approach:
  - Distributional estimates for arrival times ✓
  - Multiple Arrival Times ✓
  - Multivariate Arrivals ✓
- Sequence-to-Sequence predictions:
  - Some attempt at minimizing error at each time step ✓
  - Easy predictions in what-if scenarios ✓

Modeling Arrival Times Directly

If we have a distributional estimate for the next arrival time, we can have both **Binary Predictions** as well as **Point Estimates**.

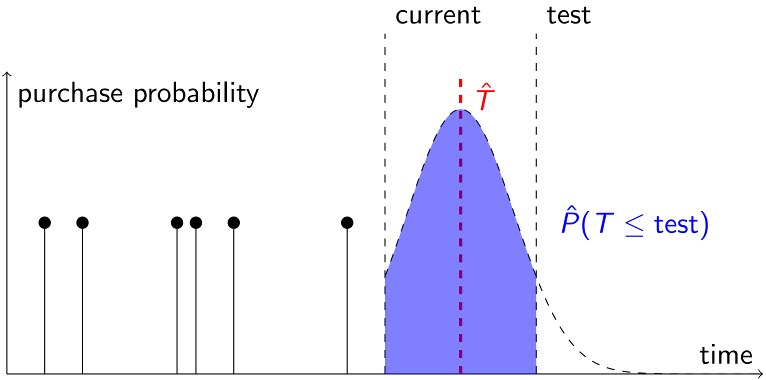


Figure: Forecasting next purchase

Conditional Excess

- At a particular time t, the number of arrivals observed is $N(t)$.
- We wish to predict the subsequent inter-arrival time $Y_{N(t)+1}$.
- We know that Time-Since-Event$(t) = \text{tse}(t)$ has passed since previous arrival.
- Clearly, we know that $Y_{N(t)+1} > \text{tse}(t)$.
- Define the (Conditional Excess) random variable :

$$Z_t = Y_{N(t)+1} - \text{tse}(t) \mid Y_{N(t)+1} > \text{tse}(t)$$

Conditional Excess

Z_t is the:

Remaining Time till Next Arrival,
Conditioned on tse(t) having passed.

- Distribution of Z_t is modeled only through $Y_{N(t)+1}$.
- Recurrent Neural Network outputs parameters for $Y_{N(t)+1}$.
- `WTTE-RNN` assumes an RNN predicts parameters for $[Y_{N(t)+1} - \text{tse}(t)]$ as distinct Remaining Times, which is a special case of this approach (i.e. if Y_i are memoryless).

Log-Likelihood

Depending on whether the next arrival is observed, we can define a log-likelihood loss:

$$l_t(\theta_t) = \begin{cases} \log P(Z_t > \text{tte}(t)) & \text{otherwise} \\ \log P(Z_t = \text{tte}(t)) & \text{if uncensored} \end{cases} \tag{4}$$

Log-Likelihood

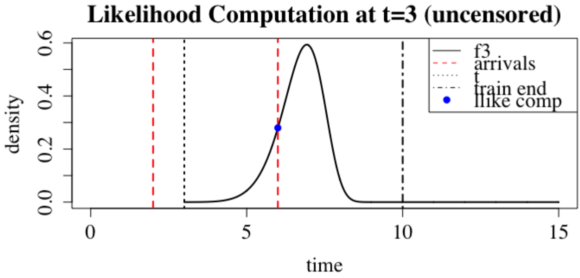


Figure: If next arrival is observed then log-likelihood is density

Log-Likelihood

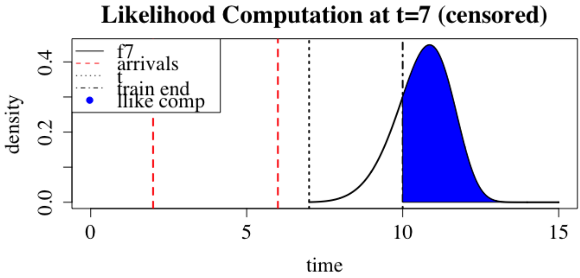


Figure: If next arrival is unobserved then log-likelihood is survival

Log-Likelihood

Table: Likelihood Compared to Binary/Point Approaches using RNNs

Method (Loss Function)	Flexible Prediction	Sparse Observations	Partial Information
Binary classification		✓	
Point estimation	✓		
Likelihood	✓	✓	✓

Conditional Independence

Consider a Recurrent Neural Network parametrized by Θ. There's some function g parametrized only by Θ that iteratively updates a latent state h_t and outputs a prediction for the (induced) distribution of Z_t (parametrized by θ_t):

$$(\theta_t, h_t) = g(h_{t-1}, X_t \mid \Theta)$$

Conditional Independence: RNN

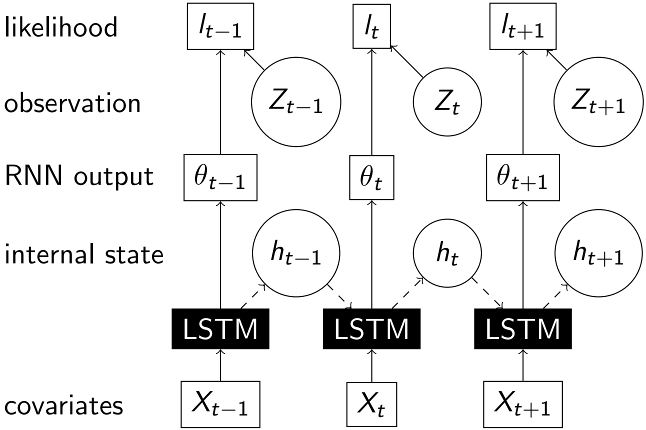


Figure: RNN Computational Flow: Outputs (θ_t) are generated by an LSTM parametrized by Θ. Log-likelihoods at each time are computed as log of densities parametrized by θ_t, evaluated at z_t

Conditional Independence

Assume that $\{Z_t\}$ are independent given $\{h_t\}$. Then for arbitrary events E_t, we can compute the joint probability of $\{Z_t\}$.

$$P(\{Z_t \in E_t\}_{t=1}^{\tau} | \; \{h_t\}_{t=1}^{\tau}) = \prod_{t=1}^{\tau} P(Z_t \in E_t | \; h_t) \tag{5}$$

Hence, we can compute the overall log-likelihood as a sum.

$$l(\{\theta_t\}) = \sum_t l_t(\theta_t) \tag{6}$$

Since $l(\{\theta_t\})$ is a deterministic function of Θ, we can find the optimal Θ by maximizing the likelihood.

Multivariate Arrivals

- Consider p different arrival processes of interest.
- Define associated conditional excess random variable:

$$Z_{i,t} = Y_{i,N_i(t)+1} - \text{tse}(i,t) \mid Y_{i,N_i(t)+1} > \text{tse}(i,t)$$

- Define: $Z_t = [Z_{1,t}, \ldots, Z_{p,t}]$, where $\{Z_{i,t}\}$ are independent given $\{h_t\}$ as before.
- Let RNN output: $\theta_t = [\theta_{1,t}, \ldots, \theta_{p,t}]$
- Define $l_{i,t}(\theta_{i,t})$ likewise.
- By conditional independence,

$$l_t(\theta_t) = \sum_i l_{i,t}(\theta_{i,t})$$

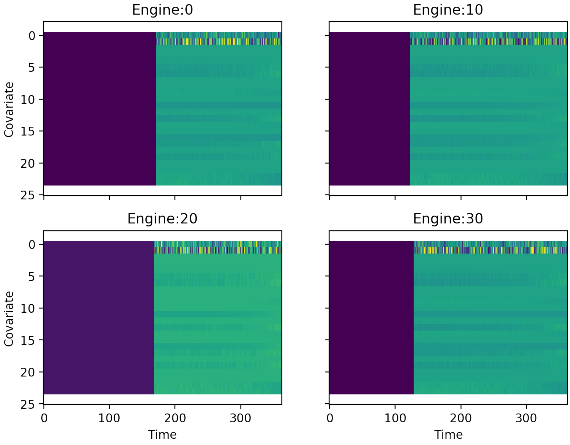
Experimental Results

Experimental results on 2 datasets:

- `CMAPSS` Engine Failure Data:
  Point Estimation problem to estimate Remaining Useful Lifetime.
- Retail Data from a Large Retailer:
  Binary Prediction problem for whether customer will purchase products during testing period.

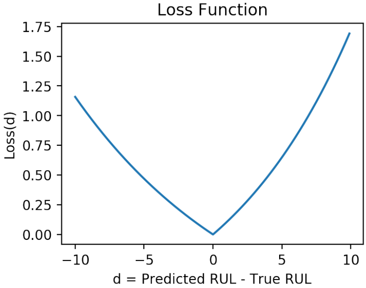
CMAPSS

High dimensional dataset on engine performance with 26 sensor measurements and operational settings.



CMAPSS: Loss function

A custom loss function was defined for the PHM08 conference competition that was based on this dataset, where over-estimation is more heavily penalized.



CMAPSS: Results

Table: Comparison on RNN-based Methods for CMAPSS $W = 64$

lr	iters	loss	MAT-RNN	WTTE-RNN	SQ-LOSS
1e-3	1e2	MCL	**41.79**	275.73	262.39
		rMSE	**32.82**	41.05	42.50
	1e4	MCL	**41.79**	275.73	262.39
		rMSE	**32.82**	41.05	42.50
1e-4	1e2	MCL	**45.84**	355.48	446.88
		rMSE	**33.16**	42.10	47.53
	1e4	MCL	**41.79**	275.73	262.39
		rMSE	**32.82**	41.05	42.50
1e-5	1e2	MCL	1926.13	**386.16**	10041.36
		rMSE	53.16	**42.04**	60.22
	1e4	MCL	**29.34**	36.84	262.39
		rMSE	**28.85**	31.49	42.50

CMAPSS: Results

Predictions of the best performing model, which is achieved by `MAT-RNN` with `iters=1e4, lr=1e-5`.

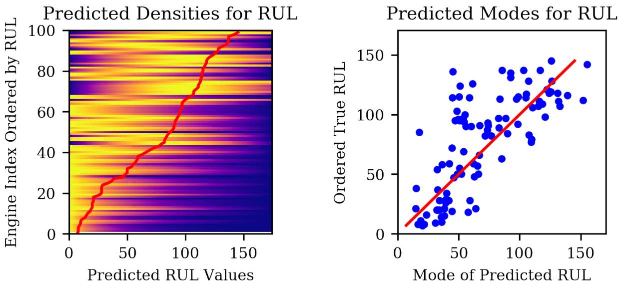


Figure: Predicted RUL Density and Mode on C-MAPSS and True RUL (in red) for `MAT-RNN` with `iters=1e4, lr=1e-5`.

Joint Product Purchases

- Trained on weekly purchases over 1.5 years (78 weeks) from 2014-01-01 to 2015-06-30.
- Predict customer purchases among a basket of products within 4 weeks after the end of training, using ROC-AUC as performance metric.
- Covariates used are Recency, Frequency and Monetary (RFM) metrics computed for each time at different aggregation levels (all, in-basket, single-product).

Joint Product Purchases: Summaries

Table: Data Summary of Product Baskets

basket	**SKUs**	**customers (x1000)**	μ_{overall}	$\mu_{\text{per-sku}}$	p_{others}	p_{trial}
bars	6	44	4.78	0.79	0.71	0.43
deli	12	79	3.58	0.29	0.55	0.62
floss	11	200	2.58	0.23	0.40	0.64
pads	7	317	**2.26**	0.32	**0.28**	**0.66**
soda	8	341	2.97	0.37	0.45	0.63

Joint Product Purchases: Results

We count the number of products for which the benchmarks outperforms RNG-Fand the average ROC-AUC for each product category.

Table: ROC-AUC Performances.

basket	customers (x1000)	# Improved Over RNG-F: SQ-LOSS	MAT-RNN	SKUs
bars	44	0	**2**	6
deli	79	4	**8**	12
floss	200	10	**11**	11
pads	317	4	**7**	7
soda	341	1	**8**	8

Joint Product Purchases: Do Multiple Single Models Work?

- Same network structure, covariates.
- Different network for each product, model trained on single product.
- 8x parameters for collection of single product models.

Joint Product Purchases: Do Multiple Single Models Work?

Table: Comparison of ROC-AUC performance on soda for single and joint `MAT-RNN` models

sku	single	joint	diff
1	0.8868	**0.8897**	+0.0029
2	0.8073	**0.8686**	+0.0614
3	0.8331	**0.8605**	+0.0274
4	0.8501	**0.8761**	+0.0260
5	0.8445	**0.8829**	+0.0384
6	0.8193	**0.8615**	+0.0422
7	0.8640	**0.8909**	+0.0269
8	0.7742	**0.8840**	+0.1098

Future Work

- Scaling to 1000x products?
  Sparse transactions to dense matrices is a problem.
- Adding more per-individual, per-product features to predict
- First-arrival prediction?

Last Frame

Questions?

`tianle91.github.io`