Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 220 (2023) 1033–1038

Procedia Computer Science

www.elsevier.com/locate/procedia

The 4th International Workshop on Statistical Methods and Artificial Intelligence (IWSMAI)
March 15-17, 2023, Leuven, Belgium

Travel-time prediction using neural-network-based mixture models

Abhishek Sharma*[a], Jing Zhang[b], Daniel Nikovski[b], Finale Doshi-Velez[a]

[a]*SEAS, Harvard University, Allston, MA, USA*
[b]*Mitsubishi Electric Research Labs, Cambridge, MA, USA*

Abstract

Accurate estimation of travel times is an important step in smart transportation and smart building systems. Poor estimation of travel times results in both frustrated users and wasted resources. Current methods that estimate travel times usually only return point estimates, losing important distributional information necessary for accurate decision-making. We propose using neural network-based mixture distributions to predict a user's travel times given their origin and destination coordinates. We show that our method correctly estimates the travel time distribution, maximizes utility in a downstream elevator scheduling task, and is easy to retrain—making it a versatile and an inexpensive-to-maintain module when deployed in smart crowd management systems.

Keywords:
travel-time predictions; mixture modeling; neural networks

1. Introduction

Estimation of travel times is necessary for accurate planning by both users and smart urban-planning systems. For example, a driver plans her route on Google Maps using her current location and the destination, and a smart-building system can use users' walking times to efficiently schedule a group of elevators as soon as they enter the building. We focus on the setting where we must make a prediction using only the known origin and destination, as opposed to using trajectory information from a trip that has already started. This setting is relevant when we need to plan before the user begins her trip, or when collecting location information is expensive or intrusive. The setting is also modular, as a model developed for this problem can be used with a model that first predicts the destination given a user's partial trajectory [12]. This destination can then be used by our model to predict the time to complete the trip.

* Corresponding author. Tel.: +1-413-404-2941. Work done as an intern at Mitsubishi Electric Research Labs.
E-mail address: abhisheksharma@g.harvard.edu

10.1016/j.procs.2023.03.144

Given only the origin and the destination, there can be multiple paths to reach the destination. Most data-driven methods ignore this by assuming a known, fixed noise model for travel times [1], which in most instances is a unimodal Gaussian distribution for travel times. These methods provide a point estimate for the predicted time to destination [10, 9, 13]. This can be misleading if two or more routes are very different, meaning the true distribution of travel times is in fact multimodal. Instead of providing point estimates, our method provides a full distribution which can account for multiple modes of travel. This provides the downstream user (either a human or an automated model) with richer information for accurate decision making.

This paper proposes Neural Travel-time Mixture Model (NTMM), a generic and flexible neural-network model to address the time-to-destination problem which is agnostic to the domain of application and only uses origin and destination coordinates as inputs. Additional contextual inputs such as the time of day can also be incorporated in a straightforward manner. Our method accounts for multiple routes of travel, given the same origin-destination pair by building an accurate travel-time distribution. This distribution can be used to get typical modes of travel, is easy to sample from, and can be used for downstream applications such as group elevator scheduling and outlier detection. We evaluate NTMM on a trajectory dataset from a realistic building traffic simulator, demonstrating that our model accurately estimates the travel-time distribution, is easy to retrain on new trajectory distributions, and enables superior decisions in downstream decision-making tasks.

2. Related Work

Travel-time prediction methods can broadly be viewed as either model-based or data-driven [1]. Model-based methods either use queuing theory or agent-based modeling to build detailed simulators for user movement. These models make strong assumptions about the movement flow, speed, and density [11, 2, 7]. Because they are inherently generative models, sampling user trajectories from them is relatively straightforward. Whereas these trajectories can be used to construct a travel-time distribution, sampling trajectories is slow and expensive. Furthermore, performance degrades for longer-duration simulations [1]. Moreover, precise calibration of physical models could be slow and laborious. In contrast, our method can construct accurate estimates for the travel-time distribution without the extra compute by directly learning the time distribution from observed data. Furthermore, the accuracy and time complexity of prediction do not depend on the duration of the trajectory.

Data-driven methods directly fit a function between the inputs and travel times using historical data. Traditionally, regression methods have been used to predict travel times [13, 6, 9, 10]. These methods provide a point estimate for the travel time by minimizing a loss function (typically squared error). In contrast, we directly predict time-to-destination by providing distributional estimates, which can be used to compute not just mean estimates but also other quantities like different quantiles or error estimates.

3. Problem Formulation

We are provided a dataset $\mathcal{D} = \{(X_n, T_n)_{n=1}^N\}$ consisting of the origin and destination coordinates (X_n) and duration (T_n) of trajectory n. These trajectories could correspond to different users or even the same user at different times—we do not make a distinction between these cases in this work. Using this dataset, our goal is to learn the conditional probability density, $p(T|X)$. This conditional probability density can be parameterized by a model with parameters θ, which we can optimize using maximum likelihood estimation. Since we want to enable a variety of applications using this distribution, we would like to evaluate the probability density function, $p(\cdot|X)$ and the cumulative distribution function, $F(\cdot|X)$. We also want an easy way to sample from this probability distribution.

4. Methods

Background. Mixture density networks use a neural network to model the density of a target variable, conditional on the input features [4]. They do so by using a mixture model to express the probability density of the target. This is an appealing approach for our application, because this model has the flexibility to model completely general probability distributions. The probability density of a target variable Y is represented as a weighted combination of individual densities $p(Y|X) = \sum_{k=1}^{K} w_k(X)\phi_k(Y|X)$, where X is the input variable we condition on. Both w_k and ϕ_k are outputs of a neural network. For our application, we use individual Gaussian probability density functions. The output of the network is therefore K sets of weights, w_k, means, μ_k, and standard deviations, σ_k.

Method: Mixture density network for travel times. Given current position and destination, the user can take multiple routes. Predicting a point estimate for the time to destination (TTD) is necessarily going to be inaccurate, especially for multi-modal, complex TTD distributions. In contrast, Neural Travel-time Mixture Model (NTMM) provides a distributional prediction for the TTD by using a mixture density network model to account for the multi-modality in the travel times. The neural network model is able to learn complex non-linear relationships between the origin and the destination.

Since the travel times are positive, we use the mixture density network to model the travel times in log space before using an exponential transformation to get the probability distribution for travel times. By change-of-variables over the expression for probability distributions, we can find the probability density for the time variable T:

$$p(T|X) = p(\log T|X)/T = \sum_{k=1}^{K} w_k(X;\theta) \cdot \phi_k(\log T|X;\theta)/T \tag{1}$$

where θ denotes the parameters of the neural network. As mentioned above, ϕ_k denotes the Gaussian probability density function with mean μ_k and standard deviation σ_k.

This mixture density representation by NTMM is useful because the resulting distribution can be used (i) to evaluate the conditional cumulative distribution function (CDF), $F(T|X)$, ii) to evaluate the probability density as seen earlier, iii) and to generate samples, $T_i \sim p(T|X)$. This enables a wide variety of downstream tasks, where these features are needed. We will explore one such application in the Experiments section. The CDF can be evaluated as $F(T|X) = \sum_{k=1}^{K} w_k\Phi_k(\log T|X;\mu_k,\sigma_k)$, where Φ_k is the CDF for the Gaussian distribution. A sample T_i from the mixture density $p(T|X)$ can be generated using the implied generative distribution [5]: $Z \sim \mathbf{Cat}(w_1,\ldots,w_K), Y \sim \mathcal{N}(\mu_Z,\sigma_Z), T \leftarrow \exp(Y)$, where $\mathbf{Cat}(\cdot)$ and $\mathcal{N}(\cdot)$ are Categorical and Gaussian distributions respectively.

Learning. Given a dataset of inputs $\mathbf{X} = \{X_n\}_{n=1}^{N}$ and times $\mathbf{T} = \{T_n\}_{n=1}^{N}$, the log-likelihood function of NTMM is $\log p(\mathbf{T}|\mathbf{X};\theta) = \sum_{n=1}^{N} \log p(T_n|X_n;\theta)$. We learn the parameters by maximizing this log-likelihood, and using a regularization term that consists of priors for $\{\pi_k\}$ and $\{\sigma_k\}$. The resulting objective is:

$$\max_{\theta} \log\left(\sum_{k=1}^{K} w_k(X_n;\theta)\phi_k(Y_n|X_n;\theta)\right) + \lambda \sum_{k=1}^{K} \left(\log p(\pi_k(X_n;\theta)) + \log p(\sigma_k(X_n;\theta))\right) \tag{2}$$

The hyperparameters in the objective are (a) the number of components (K), (b) the regularization penalty (λ), (c) the number of nodes in the two dense layers, and (d) the learning rate of the optimizer.

Implementation details. We use the PyTorch library to define our models and use automatic differentiation to compute gradients of our objective. We use the Adam optimizer and treat its learning rate as a hyperparameter. We tune the hyperparameters by using the Bayesian Optimization module in the Ray Tune software package [8].

5. Data and Environment

We use the SimTread[1] simulation software package to simulate realistic trajectories of people moving in an indoor setting. We set the parameters to generate two sets of trajectories; each parameter setting imitates realistic movement scenarios on a building floor. In the first setting (SimTread v1), employees leave one of the three conference rooms (located in different parts of the floor) to go to the elevator. On their way, an employee can stop by their desk, make a

[1] https://www.vectorworks.cn/en/community/partner-community/partner-products/product/simtread/

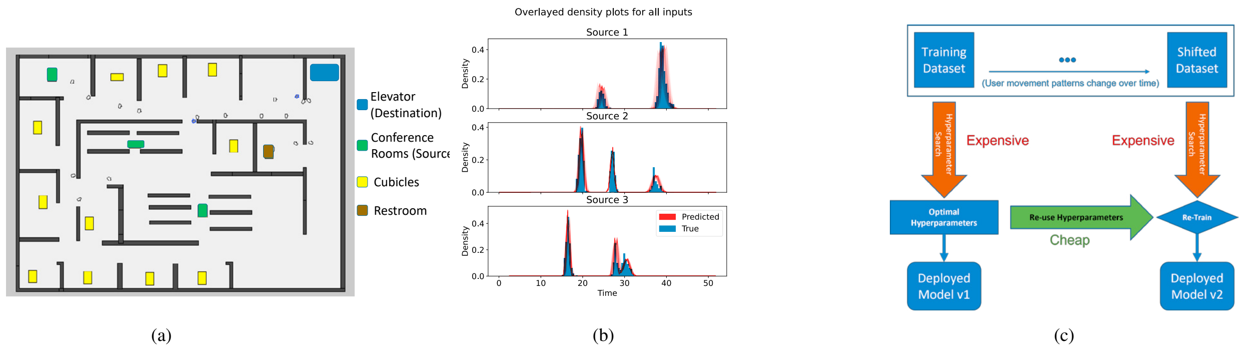


Fig. 1: (a) Floor plan on the SimTread v1 setting. The elevator (destination) is in the top corner, and the conference rooms (origins) are at different locations of the floor. (b) Density fits of the neural-net mixture to the trajectories with three different conference rooms as origins and one destination (elevator)—each pair with multiple possible intermediate points. Each red curve is the estimated density function conditioned on the origin-destination coordinates from the test set. The blue plot is the empirical density function of the test set. The NTMM correctly learns that different origins result in different travel-time distributions, including how many modes there are in the distribution. (c) Workflow for re-deploying model after retraining on a dataset with modified movement patterns. We can either re-do the expensive step of hyperparameter search, or re-use the hyperparameters from a previous deployment. We show that we can do the latter, because the hyperparameters can transfer without a significant drop in performance (Figure 2).

stop by the rest-room, or make both these stops (Figure 1a). The employee can also choose to go the elevator directly. Clearly, the travel time distribution is quite complex (Figure 1b, blue), and predicting a single number for the travel time is not reasonable, especially if the time itself will be used to plan resources / make other decisions. SimTread v2 consists of trajectories with more than one destination, and much more complex trajectory routes. For our prediction, we only use the start and end coordinates as inputs and the trajectory duration as the target.

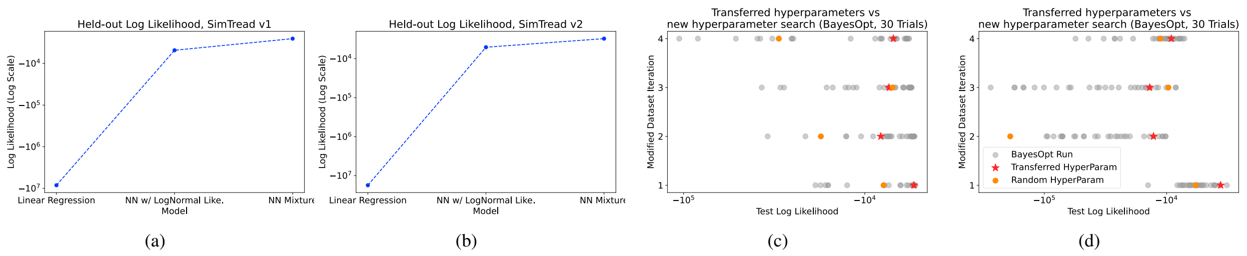


Fig. 2: (a,c): Held-out log likelihood when trained on the data from SimTread v1 environment (top) and performance of the optimal hyperparameters on this data in environments created by perturbing source and intermediate points (bottom). (b,d): Held-out log likelihood and hyperparameter transfer results for SimTread v2, which is a more complex version of SimTread v1. The bottom plots display the log-likelihoods of trained models with hyperparameters obtained in a new hyperparameter search in grey. The log-likelihood of models trained with hyperparameters of original environments is plotted in red, and a random hyperparameter from a reasonable space is plotted in orange. Takeaway: NTMM outperforms baselines in held-out likelihood and the transferred hyperparameters have good performance on modified environment settings.

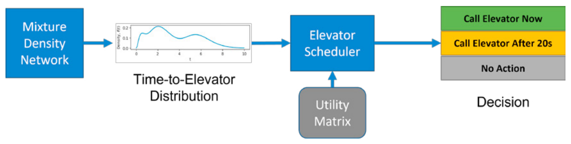
6. Experiments

Baselines. To validate that our model correctly learns the travel-time distributions, we compare our model's held-out log likelihood to two baselines. The first is a neural network model with a Log-Normal likelihood, which is equivalent to log-transforming the times and then minimizing the squared error. This is the standard architecture for doing regression using neural networks. In addition to predicting the mean, we also predict the standard deviation—which lets us compute the probability density function. The second baseline is a linear regression model, which is a simple model that is also often used in practice. To get the probability density, we obtain the maximum likelihood estimates for both mean and standard deviation.

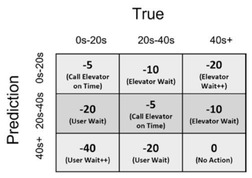
Experiment: Elevator scheduling. Next, we investigate if having correct distribution estimates is even necessary, or if a Gaussian likelihood is sufficient. Using our SimTread v1 environment, we use the time-to-elevator distribution to decide whether to call the elevator, to call after 20 seconds, or not to take an action. For this decision, the elevator needs to know which of the following time intervals the user will arrive in : $(0s-20s)$, $(20s-40s)$, $(40s, \infty s)$. We evaluate the benefit of NTMM for a downstream task in decision-theoretic setting. In this setting, the goal is to maximize the utility of our decision [3]. The utility of taking various decisions, $u(c, c')$, depends on the decision we take (c), and also the decision that should have been taken (c'). We summarize the utility function in Figure 3b. Using the utility function, we can estimate the *conditional gain* of choosing an action c for a test input X: $G^*(h = c|X) = \int_t u(c, t)p^*(t|X)dt$, where $p^*(t|X)$ is the true conditional density of travel times. If we have a model $p_m(t|X)$ instead, the conditional gain of the model is $G_m(h = c|X) = \int_t u(c, t)p_m(t|X)dt$, which we can use to compute the optimal decision $h^*(X)$ and optimal prediction $h_m(X)$: $h^*(X) = \arg\max_c G^*(c|X), h_m(X) = \arg\max_c G_m(c|X)$. We can compare the quality of our predicted decision $h_m(X)$ against the optimal decision $h^*(X)$ by computing the gap: $\Delta_m = G^*(h^*(X)|X) - G^*(h_m(X)|X)$. This gap is non-negative, and zero if and only if we take the optimal decision. We can expect to do well on this task only if our uncertainty estimates for the different classes are correct. We build an empirical distribution for $p^*(t|X)$ on our test set and use it to evaluate Δ_m for all the models.

Experiment: Evaluate Fast retraining. When deployed, any model requires retraining on new data—perhaps on movement patterns for which the initial hyperparameters are not optimal anymore. However, we do not want to perform an expensive hyperparameter search every time we train, and our model has higher utility if it can reuse previous hyperparameters without a large drop in performance. We test whether the hyperparameters we searched for our original dataset can still give good (i.e., better than random) performance when our training dataset changes. For both SimTread v1 and v2, we sample four more datasets, varying both the origin and intermediate points. We retrain NTMM on the hyperparameters we found after a search on the original datasets, and see how the held-out log likelihood compares to the case when we re-do the hyperparameter search from scratch.

7. Results

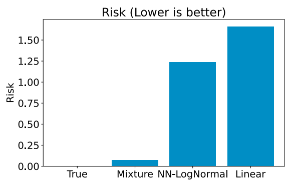


(a)



Prediction \ True	0s-20s	20s-40s	40s+
0s-20s	-5 (Call Elevator on Time)	-10 (Elevator Wait)	-20 (Elevator Wait++)
20s-40s	-20 (User Wait)	-5 (Call Elevator on Time)	-10 (Elevator Wait)
40s+	-40 (User Wait++)	-20 (User Wait)	0 (No Action)

(b)



(c)

Fig. 3: (a) Workflow for the elevator scheduling task. The time-to-elevator distribution (output of our model) is an input to the elevator scheduling module. This module uses the time-to-elevator distribution and the utility matrix to decide an action. (b) Utility matrix with utilities of classifying user times into different classes for the elevator scheduler (justifications are provided in parentheses). The utilities depend on both the true class as well as the predicted class. Correct classification leads to highest utility, and the utilities are asymmetric—misclassification that leads to user waiting is worse than misclassification that leads to elevator waiting. (c) Risk of prediction decisions on the elevator scheduling task. The task uses the estimate for a user's travel time distribution and the utility of predicting specific actions to decide whether to call the elevator for the user. NTMM achieves the lowest risk when compared to optimal decisions since it is able to correctly quantify its uncertainty about each of the classes.

NTMM accurately models the travel-time distribution. We see from Figure 1b that the density predicted by NTMM captures all the distinct modes in the time-to-destination distribution. We also see in the held-out log-likelihood—NTMM outperforms naive linear regression as well as a neural network with a Gaussian likelihood. This strengthens the claim that we ought to consider the full distribution of travel times.

NTMM enables accurate downstream decisions by correctly quantifying uncertainty. We see from Figure 3c that NTMM achieves risk closest to optimal decisions on held-out data. This demonstrates the usefulness of having distributional estimates for travel times—such estimates enable downstream decision-making in a way that point estimates just cannot. Our elevator scheduling task is just an example, and other downstream tasks include detecting anomalous trajectories and comparison of different user behaviors on a distributional level. Given that the distributional estimates themselves are quite accurate, it is not surprising that our uncertainties about user's travel times are correctly quantified as compared to a Gaussian or a Log-Normal likelihood.

NTMM hyperparameters can be re-used even when the data distribution changes. Figure 2 shows the performance of NTMM retrained on modified versions using transferred hyperparameters as well as optimal hyperparameters. The optimal hyperparameters here are obtained after rerunning Bayesian Optimization. We see that transferring hyperparameters does not cost us too much in terms of performance. This reduces the compute requirement of the deployed system, because the cost of retraining is marginal compared to a full hyperparameter search.

8. Conclusion and Future Work

We propose Neural Travel-time Mixture Model, a neural network model that leverages mixture distributions to provide distributional estimates for travel time distributions given only the origin and destination information. We highlight the importance of providing accurate distributional estimates for travel times, as opposed to prevalent approach of only providing point estimates. We argue that point estimates lose important information if the travel-time distributions are multi-modal or have long tails. NTMM is able to capture these complexities in the target distribution, and is cheap to retrain when in deployment. Furthermore, we present a case-study of how our model's output can be used by an elevator scheduler. This example shows why it is important to not just have distributional estimates, but also to correctly quantify uncertainties in the target space.

These features make it a useful module in a trajectory modeling software that predicts the destination of users given their partial trajectories. In the future, we plan to also incorporate hierarchy in the model so that similar user patterns can be pooled to give more accurate predictions at a per-user level. Another interesting future direction involves adding contextual information (e.g., time of day) to the inputs, as it can improve estimates of the travel time distributions.

References

[1] Bai, M., Lin, Y., Ma, M., Wang, P., 2018. Travel-time prediction methods: A review, in: Qiu, M. (Ed.), Smart Computing and Communication, Springer International Publishing, Cham. p. 67–77. doi:10.1007/978-3-030-05755-8_7.

[2] Ben-Akiva, M., Bierlaire, M., Burton, D., Koutsopoulos, H.N., Mishalani, R., 2001. Network state estimation and prediction for real-time traffic management. Networks and Spatial Economics 1, 293–318. doi:10.1023/A:1012883811652.

[3] Berger, J.O., 1985. Statistical Decision Theory and Bayesian Analysis. Springer Science & Business Media.

[4] Bishop, C.M., 1994. Mixture density networks .

[5] Bishop, C.M., 2006. Pattern recognition and machine learning. volume 4. Springer.

[6] Duan, Y., L.V., Y., Wang, F.Y., 2016. Travel time prediction with lstm neural network, in: 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), p. 1053–1058. doi:10.1109/ITSC.2016.7795686.

[7] Kimura, T., Sano, T., Hayashida, K., Takeichi, N., Minegishi, Y., Yoshida, Y., Watanabe, H., 2019. Representing crowds using a multi-agent model – development of the simtread pedestrian simulation system. JAPAN ARCHITECTURAL REVIEW 2, 101–110. doi:10.1002/2475-8876.12073.

[8] Liaw, R., Liang, E., Nishihara, R., Moritz, P., Gonzalez, J.E., Stoica, I., 2018. Tune: A research platform for distributed model selection and training URL: http://arxiv.org/abs/1807.05118.

[9] Siripanpornchana, C., Panichpapiboon, S., Chaovalit, P., 2016. Travel-time prediction with deep learning, in: 2016 IEEE Region 10 Conference (TENCON), IEEE, Singapore. p. 1859–1862. URL: http://ieeexplore.ieee.org/document/7848343/, doi:10.1109/TENCON.2016.7848343.

[10] Tak, S., Kim, S., Yeo, H., 2014. Travel time prediction for origin-destination pairs without route specification in urban network, in: 17th International IEEE Conference on Intelligent Transportation Systems (ITSC), p. 1713–1718. doi:10.1109/ITSC.2014.6957940.

[11] Takaba, S., Morita, T., Hada, T., Usami, T., Yamaguchi, M., 1991. Estimation and measurement of travel time by vehicle detectors and license plate readers, in: Vehicle Navigation and Information Systems Conference, 1991, p. 257–267. doi:10.1109/VNIS.1991.205771.

[12] Tsiligkaridis, A., Zhang, J., Taguchi, H., Nikovski, D., 2020. Personalized destination prediction using transformers in a contextless data setting, in: 2020 International Joint Conference on Neural Networks (IJCNN), IEEE. p. 1–7.

[13] Wu, C.H., Ho, J.M., Lee, D., 2004. Travel-time prediction with support vector regression. IEEE Transactions on Intelligent Transportation Systems 5, 276–281. doi:10.1109/TITS.2004.837813.