Movement Prediction

Amirabbas Jalali
amj301@mail.usask.ca
University of Saskatchewan
Saskatoon, SK, Canada

Seyedeh Mina Mousavifar
sem311@mail.usask.ca
University of Saskatchewan
Saskatoon, SK, Canada

ACM Reference Format:
Amirabbas Jalali and Seyedeh Mina Mousavifar. 2020. Movement Prediction. In *CMPT826: Data and Process Modeling and Analytics, 2020, University of Saskatchewan, SK.* , 5 pages.

1 INTRODUCTION

Nowadays, people use various tracking devices to help them to accomplish daily tasks, such as using GPS to find the fastest route to work. People's location history can be analyzed to predict their future location. The question that we want to answer is how predictable a user's location is when we have a certain amount of location traces. In other words, we want to impute users' locations based on their previous activities.

In this project, we investigate the predictability of users' location traces, using predictive models to impute the users' data gaps. In geospatial studies, a considerable amount of users' records are filtered due to an insufficient amount of records, which waste study resources. However, these missing records can be imputed using predictive models leading to preserve more users in the study.

To predict user location, first, we process our data to build user location matrix in each day for each user individually. This matrix is created for each user, and each coordinate, arranged in 30 rows as the length of study, 288 columns as 5 minutes duty cycles of each day. Each cell in this matrix indicates the user location grid in the respective coordinate. We dub this matrix "User Placement Matrix". After this data processing step, we use imputation techniques to fill the missing data for each user. We use Matrix Factorization [5], and Iterative Imputing [14] to predict and fill the users' placement matrix. We mask a specific portion of our data and fit our models based on the masked data. Then the predictions error on the masked data is computed. The proposed models outperform our baseline, which is replacing all missing data with the average of each column, which demonstrates the effectiveness of our approach.

2 RELATED WORK

The developments in location-acquisition and mobile communication technologies enable individuals to utilize various location services. As the primary task of spatiotemporal data mining, location prediction predicts the next position of an object at a given time. Extensive research has been done on predicting the next location based on individuals and trajectories.

For prediction based on trajectories, Zheng et al. [19] provide the outline of trajectory data mining, including trajectory data preprocessing, pattern mining and classification. Further, it explores the correlations and differences between the existing techniques and provides a public trajectory dataset. Mathew et al. [8] introduces a hybrid method for predicting human mobility using Hidden Markov Models (HMM). This method clusters location histories based on their characteristics and then train an HMM for each cluster to compute the probability of possible sequences and return the next place from the chain with the highest probability. The proposed method considers both location characteristics and the effects of each individual's previous actions. Du et al. [2] extend the HMM models to a Continuous Time Series Markov Model (CTS-MM) to predict locations in real-time. This method uses the Gaussian Mixed Model (GMM) to simulate the posterior probability of a location in the continuous-time series. Then, the probability calculation method and the state transition model of the Hidden Markov Model (HMM) are improved to get the precise location prediction. Lian et al. [6] propose the CEPR(CCollaborative Exploration and Periodically Returning Model) algorithm, which adopts a collaborative filtering technique, and the user past behaviour for location prediction and recommendation. Fan et al. [4] provide a deep learning approach for next location prediction, using both Convolutional Neural Network(CNN) and bidirectional Long-term Short-term memory(LSTM) networks based on trajectories contextual features.

For prediction based on individuals, most of the models use Markov Models. Xu et al. [18] extract taxi traces to create a Probabilistic Suffix Tree and predict short-term routes with Variable-Order Markov models on vehicle passing data. Simons et al. [12] built an HMM for every driver, which predicts the future destination and route of each target. Lin et al. [7] provide a hierarchical Markov model that infers a user's daily movements through an urban community. This method focuses on predicting the destinations of specific individuals based on their historical trajectory patterns. Wu et al. [17] apply Markov Random Field to predict the annotation of location records and the user's destiny. In this approach, performance is enhanced by increasing user's records. Nghia et al. [3] predict user's geolocation at a given tweet's posting time using a geo matrix factorization model in real-time. In this approach, they map tweets into a latent space and then linearly combine these features to predict latitude and longitude. However, their dataset comprises tweets containing lots of semantic information, which makes results prone to be influenced by people's subjective emotions and expressions.

Our approach is similar to Nghia et al. approach but resolves the influence of users' semantic information by only considering users' latitude and longitude. Furthermore, we intend to benchmark matrix imputation methods on location prediction, which hasn't been done before. These imputation methods can also be used to fill missing geological data to obtain more accurate location records.

Data and Process Modeling and Analytics Course'20, Winter, 2020, University of Saskatchewan, SK

Table (1) The Coordinate of Greater Saskatoon

Coordinate	Min	Max
Latitude	52.058367	52.214608
Longitude	-106.7649138128	-106.52225318

Table (2) Summary of Grid

	Min	Max	Mean	STD
Horizontal Grid	0	165	90.7	23.9
Vertical Grid	0	173	75.5	14.3

3 METHODS

3.1 Data Cleaning

In this section, we elaborate on our approach to finding the best possible data to use them as input for the predictive models. To find the user's who have the most participation in the available records, we use the Battery Record table from the SHED10 database.

First of all, from the "battery" table, we remove all participants who have returned less than 50% of the total possible battery records. Because, in our evaluation, we mask portions of the data, and we need to have the original data to compare to the predicted values. Moreover, we are interested in each user's behaviour separately, and having a smaller group of users does not affect our results. In the next step, we use GPS records on the "gps" table to retrieve users' locations. For being more accurate, we only use the records that have GPS accuracy better than 100 meters, because having a small accuracy lead to not capturing enough records, and a large accuracy affect the performance of the models.

Secondly, we remove all GPS traces outside the city limits of Greater Saskatoon. The coordinates can be viewed in table 1.

3.2 Stratification and Aggregation

After filtering, we aggregate the data by taking the average location every 5 minutes, because "Ethica Data" uses a one minute on, four minutes off duty cycle, hour aligned for data collection. Then, we transform data into Universal Transverse Mercator (UTM) coordinates system [13]. To do this, we convert all the measured location values in latitude and longitude into UTM coordinates. We use the pyproj library available in the Python programming language to convert our data to UTM.

Then we assign every point to 100 by 100 meters grid cells. To do this, we first find the minimum X and Y of the UTM coordinates from the data and subtract all the records by these numbers. By the formula below (1), we find the grid number of each record.

$$Grid\ Number_{x,y} = \lceil \frac{UTM_{x,y} - Min(UTM_{X,Y})}{Grid\ Size} \rceil \qquad (1)$$

Where x and y are the UTM coordinates of the record and X and Y are the set of all UTM records in the data. In this study, we set the grid size to 100 meters. Because we only use records that have an accuracy of 100 meters or better; as a result, every point has a higher chance to be in their actual grid cell than using a smaller grid size, but if we use a larger grid size, we get greater probability, but we lose our precision. Table 2 presents summary of grids.

In the following steps, we elaborate on how the users' placement matrix will be created. Users' placement matrix expresses the user location (grid cell) every 5 minutes. There are 288 five minutes time spans in a day. Subsequently, we have 288 columns in our user's placement matrix. Each row of this matrix represents a day in the experiment duration. The experiment duration is one month, and as a result, we have approximately 30 rows in each matrix. The difference between the experiment duration for each user is not indispensable because we model each user individually. We choose this format for the matrix because most of the people have daily routines. Consequently, every row has the potential to be wholly related to others, and it helps to find repetitive patterns in the data easily. Finding the patterns between rows and between columns in this kind of matrices is the essence of the imputation methods.

To create this matrix, we separate each user's records and store them on a sparse matrix. We use the day of the experiment, the number of which 5 minutes it is in a day and the grid label for X and Y coordinates. We have to separate matrix for X and Y coordinates because if we combine these with, for example, a hash function, it will vanish the repetitive characteristic between the days.

To have a way to measure the error of our model, we mask 20% of each user data intentionally and kept the actual data to compare the predicted results with them to find the error of our imputation.

3.3 Modeling

In this section, we elaborate on our predictive models. We aim to bench these approaches and compare them in imputing the missing data. The main reason to choose imputers is the essence of our data. Imputers are working accurately when the matrix of the data has specific inner relations. The columns can be related to others, and rows have this feature likewise. We believe that users' trajectory has this characteristic. Each row, or in other words, each day can be similar to other days. For example, every Monday, a user goes to work for a specific period and then returns home. In each time frame or each column, the users can be in a particular place. For example, a user is always home after 10 pm. As a result, assessing the applicability of imputers to solve this problem is promising.

3.3.1 Simple Filling. Our first approach for imputing the missing matrix cells is replacing each cell with the mean of each column based on the existing data. There are other strategies, such as filling with random numbers, zeros, median, and minimum of the existing data in columns. We use the SimpleFill function of fancyimpute [11] library to implement this method. The reason for choosing the mean method instead of other methods is logical limitations. For instance, filling each missing cell with the lowest latitude and longitude, or with the center of Saskatoon, might create many teleports for the user, which isn't possible for human behaviour.

3.3.2 Matrix Factorization. In this method, we use direct factorization of the incomplete matrix into low-rank U and V, with an L1 sparsity penalty on the elements of U and an L2 penalty on the elements of V to complete the matrix [5]. This method represents days and duty cycles into a shared latent space, using a vector of latent features for a day or a 5-minute span. So, each cell in the user placement matrix is modelled as the inner product of the latent vectors of day and a specific time span. We solve the factorization

by gradient descent [5]. We use the MatrixFactorization function of fancyimpute [11] library to implement this method. We set the learning rate to 0.001, which is a constant indicating the rate of approaching to the minimum. A small value is recommended for learning rate[1], so that small steps are taken towards the minimum without jumping over it. Furthermore, the number of latent factors is set to default 10, which is recommended by the literature[5]. Finally, we set the regularization parameter to zero because our models are user-specific, and we aim to learn each user's behaviour individually without generalization.

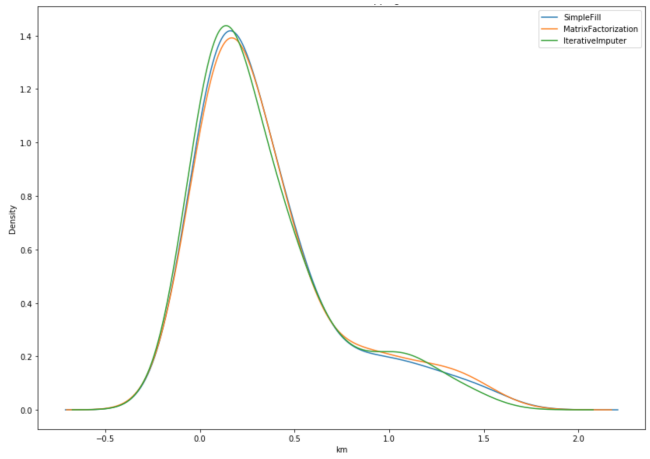
3.3.3 Iterative Imputer. Our final method is iterative imputer [14], which fills missing values by modelling each 5-minute span with missing values as a function of other elements in a round-robin fashion. At each step, a column is considered as the output, and the other columns are treated as inputs. Then a regressor is fit on these inputs and outputs for existing values of output. Finally, the regressor is used to predict the missing values of y. We use the IterativeImputer [14] function of scikit-learn [10] library to implement this method. The order of imputation is set from features with fewest missing values to the most, to minimize the effect of imputed values on the prediction. Moreover, the missing values are initialized by mean column value the same as the *Simple Fill* algorithm considering the limitations of other strategies. Finally, the number of nearest features, which is the number of other features to use to estimate the missing values of each feature column, is set to compare all features to utilize model capability fully.

4 RESULTS

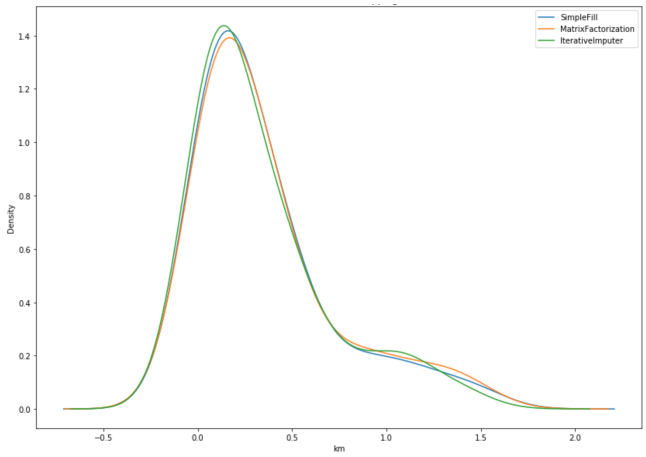
To evaluate our models' performance, we mask points in two different ways, firstly removing random points from the user-placement matrix and secondly, removing one entire day. We masked 20% of the data randomly, which comprise 48% of the data along with missing points. For the second approach, removing weekdays or weekends don't differ much because the imputation methods try to find the similarities between rows, and weekdays are compared to weekdays and weekends are compared to weekends implicitly. To examine the significance of our prediction, we implement a Random top 10 model, which randomly fills masked points from the user's top 10 most visited locations. Finally, we measure *Mean Absolute Error(MAE)* of the predicted grid of masked points in Users' placement matrix from the original grid value. We choose MAE based on Willmott et al. [16] analysis, which indicates that MAE is the most natural measure of average error magnitude for assessing model performance. We compare our models MAE the baselines Random Top10 and Simple Fill in table 3.

Iterative imputer has the best performance and lowest error compared to other methods in both evaluations. Compared to the Random Top10 method, iterative imputer has lower MAE considerably. However, iterative imputer performs significantly better in masking randomly, when there is existing information in a row compared to an empty row, masking one day, because it has some clues to find the similarities. Besides, the horizontal axis has a higher error than the vertical axis because of its higher variance.

Furthermore, we investigate the performance of the model on dwells compared to the trips. Our models are better in predicting the dwells because while dwelling, the user has a low variance,



(a) Horizontal axis



(b) Vertical axis

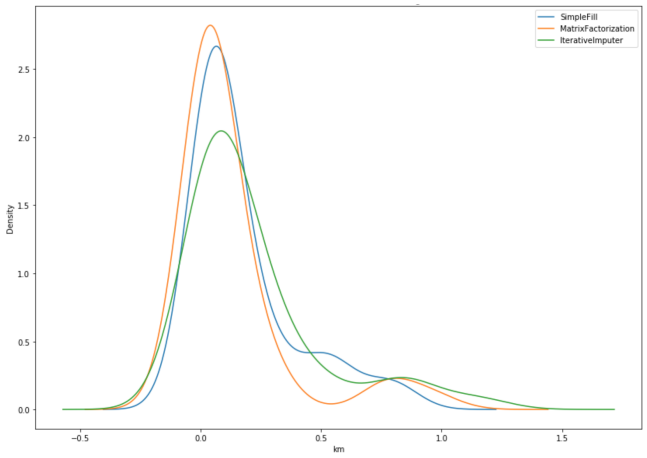
Figure (1) KDE-Plot of models MAE on trips

which makes similarity detection easier for the models. For predicting dwells, matrix factorization has the best performance, and for predicting trips, iterative imputer has the best performance. So, iterative imputer has outperformed our baseline, simple fill algorithm for both trips and on average. Figure 1 2 exhibits Kernel Density Estimate(KDE)[9] of our models MAE on trips and dwells for random masking method respectively.

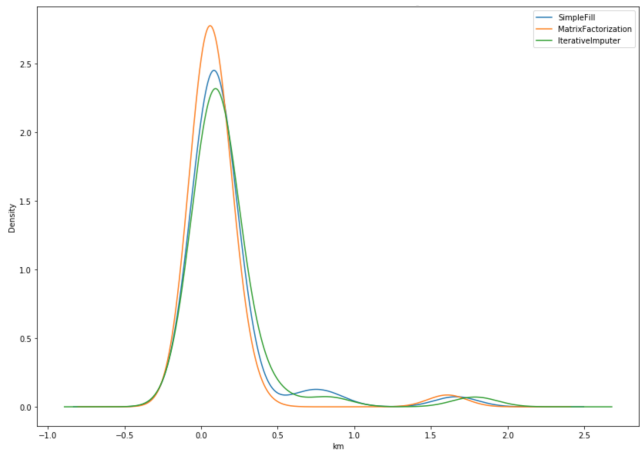
To assess the significance of our results statistically, we performed Welch's two-sample t-test [15] to test whether the error of our models is smaller than the Random Top 10 method. As it is presented in the figures, the errors of our models follow Gaussian distribution and do not have the same variance; therefore, this test is appropriate. Subsequently, we form our hypothesis as the probability of Random Top 10 method error on a user has less error than the other imputation method. We obtain a p-value<0.005 for all imputation methods which refute our hypothesis and indicate that the Random Top10 model doesn't have less MAE than our models.

Table (3) Models Average MAE in m - (horizontal axis, vertical axis)

Masking method	Category	Random Top10	SimpleFill	Matrix Factorization	Iterative Imputer
Random	All	402, 84	309, 48	233, 40	150, 33
	Trip	486, 104	393, 65	301, 54	219, 47
	Dwell	98, 7	77, 5	22, 3	9, 4
One day	All	426, 81	271, 208	296, 220	270, 205
	Trip	519, 90	346, 258	358, 271	341, 251
	Dwell	103, 10	173, 163	127, 109	215, 173



(a) Horizontal axis



(b) Vertical axis

Figure (2) KDE-Plot of models MAE on dwells

5 DISCUSSION AND SUMMARY

Our results indicate that imputation methods succeed in predicting user behaviour within 300 meters and perform significantly better than randomly picking most visited locations. Iterative imputer method has the lowest error for trip prediction, and matrix factorization approach obtains the lowest error for dwell prediction. For validating results visually, we plot the animation of predicted locations, along with existing records. There are scenarios that our models succeed in predicting daily user behaviour, inferring user home and workplace in a day without any record. However, for users with high variability, our models approach the destinations slowly and lack in identifying the final location timely.

Our dataset had a relatively small number of participants and a low variability due to the recruiting process in which most of the participants are students and have similar daily routines. In our results, we observe that our models have a lower error in the axis with the lower variance. Consequently, the performance of the models depends on the diversity of movement behaviour and our models' performance will change in a more diverse dataset. However, iterative imputer might still have the best performance among other methods, due to its ability to perform better than other methods in trip detection, which has higher variability.

In the project, We didn't have enough resources to tweak the hyperparameters of the model, but it has the potential to reduce the prediction error. We can also add some prior knowledge or create a hybrid model to imputing methods to redesign them for our specific task. Several interesting questions can follow after the prediction, such as investigating the impact of grid size on the prediction and the relation between entropy and the performance of the models. During this project, we learned how to formulate our problem and justify our choices, along with discovering how to work with geospatial data. In the case of doing the project again, we would spend more time to formulate the quality of our results more thoroughly.

In conclusion, we investigated the application of imputation methods for movement prediction, which resulted in predicting missing records with an average of 225 meters error and predicting daily trajectories with an average of 345 meters error. Furthermore, the error of imputation methods was significantly lower than randomly choosing the top 10 most visited locations.

REFERENCES

[1] Nikhil Buduma and Nicholas Locascio. 2017. *Fundamentals of Deep Learning: Designing Next-Generation Machine Intelligence Algorithms* (1st ed.). O'Reilly Media, Inc.

[2] Yongping Du, Chencheng Wang, Yanlei Qiao, Dongyue Zhao, and Wenyang Guo. 2018. A geographical location prediction method based on continuous time series Markov model. *Plos One* 13, 11 (2018). https://doi.org/10.1371/journal.pone.0207063

[3] Nghia Duong-Trung, Nicolas Schilling, and Lars Schmidt-Thieme. 2016. Near Real-time Geolocation Prediction in Twitter Streams via Matrix Factorization Based Regression. *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management - CIKM 16* (2016). https://doi.org/10.1145/2983323.2983887

[4] Xiaoliang Fan, Lei Guo, Ning Han, Yujie Wang, Jia Shi, and Yongna Yuan. 2018. A Deep Learning Approach for Next Location Prediction. *2018 IEEE 22nd International Conference on Computer Supported Cooperative Work in Design ((CSCWD))* (2018). https://doi.org/10.1109/cscwd.2018.8465289

[5] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (Aug. 2009), 30–37. https://doi.org/10.1109/MC.2009.263

[6] Defu Lian, Xing Xie, Vincent W. Zheng, Nicholas Jing Yuan, Fuzheng Zhang, and Enhong Chen. 2015. CEPR: A Collaborative Exploration and Periodically Returning Model for Location Prediction. *ACM Transactions on Intelligent Systems and Technology* 6, 1 (2015), 1–27. https://doi.org/10.1145/2629557

[7] Lin Liao, Donald J. Patterson, Dieter Fox, and Henry Kautz. 2007. Learning and inferring transportation routines. *Artificial Intelligence* 171, 5-6 (2007), 311–331. https://doi.org/10.1016/j.artint.2007.01.006

[8] Wesley Mathew, Ruben Raposo, and Bruno Martins. 2012. Predicting future locations with hidden Markov models. *Proceedings of the 2012 ACM Conference on Ubiquitous Computing - UbiComp 12* (2012). https://doi.org/10.1145/2370216.2370421

[9] Emanuel Parzen. 1962. On Estimation of a Probability Density Function and Mode. *Ann. Math. Statist.* 33, 3 (09 1962), 1065–1076. https://doi.org/10.1214/aoms/1177704472

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[11] Alex Rubinsteyn and Sergey Feldman. 2019. Fancy Impute. https://pypi.org/project/fancyimpute/#description

[12] R. Simmons, B. Browning, Yilu Zhang, and V. Sadekar. 2006. Learning to Predict Driver Route and Destination Intent. *2006 IEEE Intelligent Transportation Systems Conference* (2006). https://doi.org/10.1109/itsc.2006.1706730

[13] John P. Snyder. 1987. Map projections: A working manual. *Professional Paper* (1987). https://doi.org/10.3133/pp1395

[14] Stef van Buuren and Karin Groothuis-Oudshoorn. 2011. mice: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software, Articles* 45, 3 (2011), 1–67. https://doi.org/10.18637/jss.v045.i03

[15] B. L. WELCH. 1947. THE GENERALIZATION OF 'STUDENT'S' PROBLEM WHEN SEVERAL DIFFERENT POPULATION VARLANCES ARE INVOLVED. *Biometrika* 34, 1-2 (01 1947), 28–35. https://doi.org/10.1093/biomet/34.1-2.28 arXiv:https://academic.oup.com/biomet/article-pdf/34/1-2/28/553093/34-1-2-28.pdf

[16] Cort J. Willmott and Kenji Matsuura. 2005. Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. https://www.int-res.com/abstracts/cr/v30/n1/p79-82/

[17] Fei Wu and Zhenhui Li. 2016. Where Did You Go: Personalized Annotation of Mobility Records. *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management - CIKM 16* (2016). https://doi.org/10.1145/2983323.2983845

[18] Guangtao Xue, Zhongwei Li, Hongzi Zhu, and Yunhuai Liu. 2009. Traffic-Known Urban Vehicular Route Prediction Based on Partial Mobility Patterns. *2009 15th International Conference on Parallel and Distributed Systems* (2009). https://doi.org/10.1109/icpads.2009.129

[19] Yu Zheng. 2015. Trajectory Data Mining. *ACM Transactions on Intelligent Systems and Technology* 6, 3 (Dec 2015), 1–41. https://doi.org/10.1145/2743025