Location 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. Location 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 are investigating the possible risks for users who share their locations with online routing services. First of all, predicting users' future locations based on their location history can violate their security. On the other side, manipulating user's decisions by providing personalized recommendations based on their predicted location can be considered as a privacy issue. However, we can use predictive models to impute our missing data, and it can help us to have a better understanding of users' behaviour, and it can lead to mutual benefits for both researchers and the community.

To predict users' location, first we need to process our data to build each users' location matrix in each day and every 5 minutes. 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 K-Nearest Neighbours [17], Singular Value Decomposition [9], Matrix Factorization [5], and Iterative Imputing [16] 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 we can evaluate our prediction using the masked data and find our prediction error. If we get a reasonable error, for example, performing better than replacing all missing data with the average of each column, we find the users' movement pattern, and we can predict their location. If we can't beat that, these models can not predict users' movement behaviour.

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. [20] 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). They applied forward algorithms to compute the probability of possible sequences and return the next place from the chain with the highest probability. 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 are based on Markov Models. Xu et al. [19] 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. [13] 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. [18] 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] uses matrix factorization to select features and predicts the user's geographic location in real-time. However, their dataset is composed of 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

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 used the Battery Record table from the SHED10 database.

First of all, from the "battery" table, we removed all participants who have returned less than 75% of the total possible battery records because those users hadn't enough data to analyze and extract a meaningful pattern. We are interested in each user's behaviour separately. Consequently, having a smaller group of users does not affect our results. After filtering those users out, we have the list of our desired users. In the next step, we used GPS records on the "gps" table to retrieve users' locations. For being more accurate, we only used the records that have GPS accuracy better than 100 meters.

Secondly, we removed all GPS traces outside the city limits of Greater Saskatoon. You can see the coordinates that we use in the following table.

3.2 Stratification and Aggregation

After these filterings, we try to aggregate the data by taking the average location every 5 minutes. We pick this time duration, as mentioned in the first assignment, "Ethica Data uses a one minute on, four minutes off duty cycle, hour aligned." Then, we transformed data into Universal Transverse Mercator (UTM) coordinates system [14]. To do this, we converted all the measured location values in latitude and longitude into UTM coordinates. We used 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. We will try different grid sizes in the future and compare the results to find the best setting for our project.

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 about one month, and as a result, we have around 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 peoples 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 used 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 masked 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. For the final project, we want to bench these approaches and compare them in imputing the missing data.

3.3.1 Simple Filling. The first approach to impute the missing matrix cells is replacing each cell with the mean of each column based on the existing data. We can try other simple strategies to fill the missing data. We can replace the missing data with zeros, median, and minimum of the existing data in columns. We can randomly fill the missing data too. We used the SimpleFill function of fancyimpute [12] library to implement this method.

3.3.2 K-Nearest Neighbours. In this method, we deploy K-Nearest Neighbour (KNN) [17] with K equal to 5 to fill the missing cells. In this approach, we find the top-K nearest rows and fill the missing data based on them. To find these rows, we find the mean squared error (MSE) between the rows of the matrix and pick the top-K rows with the least errors. We only considered rows that have data on their corresponding cell. We used the KNN function of fancyimpute [12] library to implement this method.

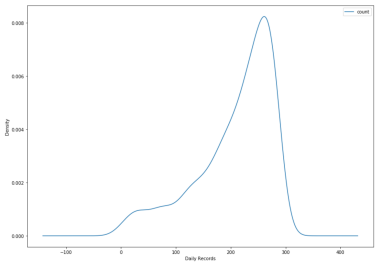
3.3.3 SVD with Soft Thresholding. In this method, we deploy Singular Value Decomposition (SVD) with Soft Thresholding. Inspired by the softImpute package for R, which is based on Spectral Regularization Algorithms for Learning Large Incomplete Matrices by Mazumder et al. [9]. We used the SoftImpute function of fancyimpute [12] library to implement this method.

3.3.4 Iterative SVD. In this method, we complete the matrix by iterative low-rank SVD decomposition. It should be similar to SVDimpute from Missing value estimation methods for DNA microarrays by Troyanskaya et al. [15]. We used the IterativeSVD function of fancyimpute [12] library to implement this method.

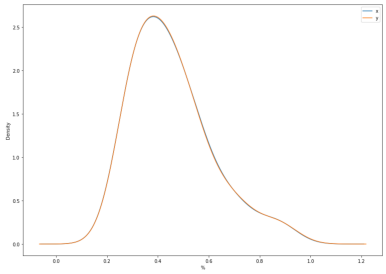
3.3.5 Matrix Factorization. In this method, we used direct factorization of the incomplete matrix into low-rank U and V, with an L1 sparsity penalty on the elements of U and an L2 penalty on the elements of V to complete the matrix [5]. We solved the factorization

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



(a) Density Plot of Daily User Records



(b) Density Plot of Missing Data Percentage

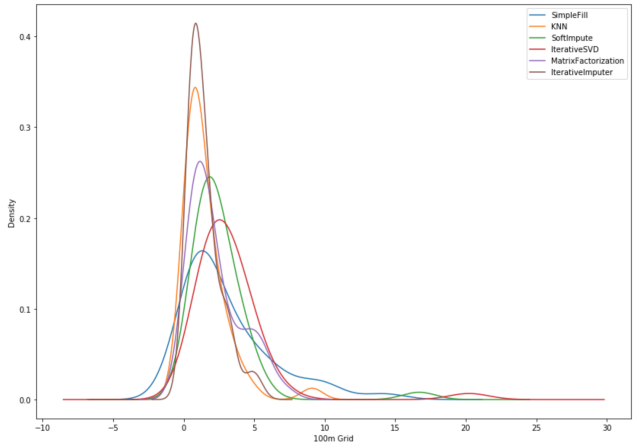
by gradient descent. We used the MatrixFactorization function of fancyimpute [12] library to implement this method.

3.3.6 Iterative Imputer. In this method, we used a strategy for imputing missing values by modelling each feature with missing values as a function of other features in a round-robin fashion. We used the IterativeImputer [16] function of scikit-learn [11] library to implement this method.

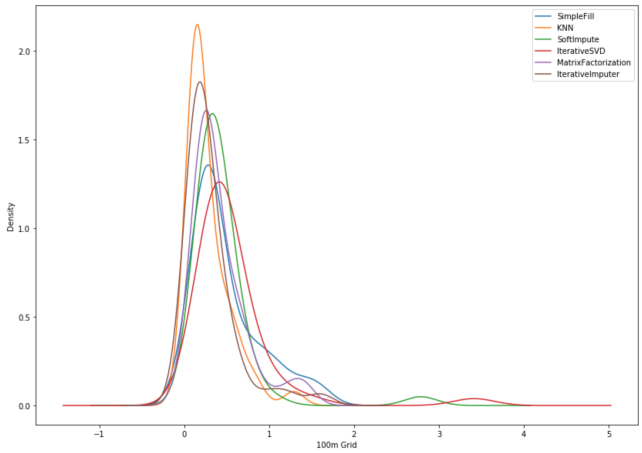
4 DATA ANALYSIS

4.1 Input

- **Users**: After filtering users based on battery records, we have 41 users from 101 participants in the study.
- **Grid Coordinate**: The latitude and longitude of records is converted to grids using UTM coordinate. Table 2 presents summary of grids.
- **Daily Records**: The GPS records are converted to Users' placement matrix on a daily basis. Figure 1a shows the density of users' daily GPS records. Because users automatically provide GPS records, they follow Gaussian distribution, and the linear axis fits it properly.
- **Missing Records**: To evaluate the performance of our models, we masked 20% cells of Users' placement matrix. The density plot of users' missing cells is plotted in figure 1b, which have Gaussian distribution.



(a) Horizontal axis



(b) Vertical axis

Figure (2) KDE-Plot of Models predicted MAE

Table (3) Models Average MAE(in Grid size)

Model	Horizontal MAE	Vertical MAE
SimpleFill	3.062235	0.492045
K-Nearest Neighbor	1.511758	0.280537
Soft Impute	2.611061	0.437107
Iterative SVD	3.316086	0.561560
Matrix Factorization	2.207322	0.411324
Iterative Imputer	1.472025	0.310712

4.2 Output

- **Evaluation**: To evaluate our models performance, we measure *Mean Absolute Error(MAE)* of predicted grid of masked points in Users' placement matrix from original grid value and then plot *Kernel Density Estimate(KDE)*[10] of our models MAE in Figure 2. Models error follow Gaussian distribution, and the linear axis fits it properly.
  Table 3 exhibits a summary of average MAE of our models on each coordinate.

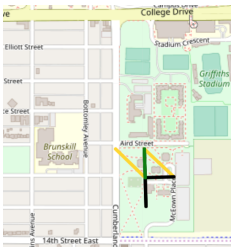
- **User Traces:** For validating results intuitively, we plot animation of predicted locations, along with existing records. Figure 3 shows a specific user on two different days. Green lines show the trace between predicted masked location and previous location, which is the same as the original location. Yellow lines indicate the trail between incorrect predicted masked locations and the former location. Red lines represent the trace between the predicted missing point and previous location, and black lines show the path between the existing future and current locations.
  In figure 3a, we observe user path from university to home. Then, in figure 3b, no data exists on this specific day, but our model predicted the user would go to university and then come back home. To further validate our prediction, in figure 3c, we observe user location at night. Consequently, *Iterative Imputer* model succeeded in inferring home and work location in the related time of day visually.

5 PROPOSAL

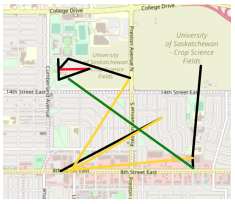
For future work, we plan on testing methods based on users' weekly activity and compare the impact of weekly and daily information on the performance of our models. To achieve this goal, we must change the number of columns to 288*7 (number of 5 minutes in a week) and the number of rows to 4 (number of weeks in a month). We can improve the results of the proposed models based on the essence of people's movements. For example, we can refine the predicted matrices based on our prior knowledge about our users, like the average speed of the user, and the maximum grid cell they can pass in 5 minutes, etc. We can investigate the Modifiable Areal Unit Problem (MAUP) [1] in our project to check the persistency of our results in different grid sizes. After obtaining acceptable performance on our model, we can expand our problem to investigate predictability on a subset of GPS data. We explore the amount of data that is adequate for predicting user behaviour.

(a) User original home trace

(b) User predicted trace on a day with no data



(c) User location at night



(d) User sample trace

Figure (3) User Trace Animation for Iterative Imputer

REFERENCES

[1] Shawna J. Dark and Danielle Bram. 2007. The modifiable areal unit problem (MAUP) in physical geography. *Progress in Physical Geography: Earth and Environment* 31, 5 (2007), 471–479. https://doi.org/10.1177/0309133307083294

[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] Rahul Mazumder, Trevor Hastie, and Robert Tibshirani. 2009. Spectral Regularization Algorithms for Learning Large Incomplete Matrices.

[10] 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

[11] 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.

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

[13] 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

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

[15] O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein, and R. B. Altman. 2001. Missing value estimation methods for DNA microarrays. *Bioinformatics* 17, 6 (Jan 2001), 520–525. https://doi.org/10.1093/bioinformatics/17.6.520

[16] 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

[17] I Wasito and B Mirkin. 2005. Nearest neighbour approach in the least-squares data imputation algorithms. *Information Sciences* 169, 1-2 (Jun 2005), 1–25. https://doi.org/10.1016/j.ins.2004.02.014

[18] 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

[19] 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

[20] 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