Matrix Factorization for Travel Time Estimation in Large Traffic Networks

Krzysztof Dembczyński[1], Wojciech Kotłowski[1], Przemysław Gaweł[1,2], Adam Szarecki[2], and Andrzej Jaszkiewicz[1]

[1] Institute of Computing Science, Poznań Univesity of Technology, Piotrowo 2, 60-965 Poznań, Poland

[2] NaviExpert sp. z o.o., Dobrzyckiego 4, 61-692 Poznań, Poland

Abstract. Matrix factorization techniques have become extremely popular in the recommender systems. We show that this kind of methods can also be applied in the domain of travel time estimation from historical data. We consider a large matrix of travel times in which the rows correspond to short road segments and the columns to 15 minute time slots of a week. Then, by applying matrix factorization technique we obtain a sparse model of latent features in the form of two matrices which product gives a low-rank approximation of the original matrix. Such a model is characterized by several desired properties. We only need to store the two low-rank matrices instead of the entire matrix. The estimation of the travel time for a given segment and time slot is fast as it only demands multiplication of the corresponding row and column of the low-rank matrices. Moreover, the latent features discovered by the matrix factorization may give an interesting insight to the analyzed problem. In this paper, we introduce that kind of the model and design a fast learning algorithm based on alternating least squares. We test this model empirically on a large real-life data set and show its advantage over several standard models for travel estimation.

1 Introduction

Travel time estimation is a fundamental and important part of many traffic-related systems. For example, the online personal car navigation requires the travel time estimates for all road segments in the traffic network for finding the shortest, or better to say, the fastest path between any two points in the network. Therefore, to get reasonable travel time estimates, we need to use a source of data that is able to cover the entire traffic network, or at least a part of it. The common approaches to travel time estimation are still based on loop detectors [13] or some other stationary sensors [7], so the majority of the current research focuses on single paths [13], freeways [12], or a subset of urban arterial roads [2]. Thus, we have to rely on other sources of data such as GPS-devices installed in the vehicles. The usage of GPS data in travel time prediction [4] is still a novel approach. The GPS data have an advantage of covering a large part of the traffic network, however, they are unfortunately sparse and unevenly distributed, what makes the prediction and estimation much harder.

© Springer-Verlag Berlin Heidelberg 2013

Let us assume that our goal is to perform accurate estimates for each single road segments and 15 minute time slots of a week. Different models can be used for this task. The simplest method may rely on averaging the travel times for a given road segment and a time slot. However, such an approach would require a lot of observations to deliver reliable estimates and to store estimates for each segment and time slot. A remedy for these problems is to use some more general features of road segments and time slots or to compute estimates on less grain level, for example, for a given road segment. There are also many other statistical and learning methods that have already been considered for travel time estimation from GPS data, like Kalman filters [8], ARIMA models [2], linear regression [10], neural networks [13] and copula based estimation [14], to mention just a few of them. In this paper, however, we follow another way.

We show that matrix factorization techniques that have become extremely popular in recommender systems [11,9] can be successfully applied to the task of the travel time estimation from historical data in a large traffic network. Typically, in the recommender system applications, the rows of the matrix correspond to users, while the columns to products, movies, or songs. Such a matrix is sparse, since it is rather unlikely that a given user would buy all possible products or watch all the movies. This matrix is then approximated by a product of two low-rank matrices that represent latent features of users and products. In travel time estimation, the rows of the matrix may correspond to short road segments and the columns to 15 minute time slots of a week. Then, by applying matrix factorization techniques we obtain a sparse model in which the latent features describe road segments and time slots. Such a model is characterized by several desired properties. We only need to store the two low-rank matrices instead of the entire matrix. The estimation of the travel time for a given segment and time slot is fast as it only demands multiplication of the corresponding row and column of the low-rank matrices. Moreover, the latent features discovered by the matrix factorization may give an interesting insight to the analyzed problem. By using a specific regularization over the time slots we can obtain smooth features that represents time characteristics of segments.

In this paper, we introduce that kind of the model and design a fast learning algorithm based on alternating least squares. We test this model empirically on a large real-life data set and show its advantage over several standard models for travel estimation.

2 Problem Statement

The goal is to predict a travel time y_{st} for a given road segment $s \in \{1, \ldots, S\}$ in a given time point t. The task is then to find a function $f(s,t)$ that predicts accurately the value of y_{st}. The accuracy of a single prediction $\hat{y}_{st} = f(s,t)$ is measured by a loss function $L(y_{st}, \hat{y}_{st})$ which determines the penalty for predicting $\hat{y}_{st}$ when the true value is y_{st}. We will use the squared error as the loss function: $L(y_{st}, \hat{y}_{st}) = (y_{st} - \hat{y}_{st})^2$. The set of historical data $\{(y_i, s_i, t_i)\}_{i=1}^{N}$ is used by a learning procedure to construct function $f(s,t)$ in order to minimise loss over future data. We additionally assume that for each road segment s and time point t, a vector of attributes is available: $\mathbf{x}_{st} = (x_{st1}, x_{st2}, \ldots, x_{stn})$. Thus the data can be represented in regular tabular form $\{(y_i, \mathbf{x}_i)\}_{i=1}^N$. Exemplary attributes include: segment id, type of the road, type of surrounding area, geographical information and segment length (which will be denoted as l_s).

Below, we first discuss simple granular models and Bayesian averaging of them as two baseline models. Then, we introduce the variant of matrix factorization suited for travel time estimation.

3 Granular Models

The granular model is based on averaging over specified granulation of data. A granule can be defined by a conjunctive rule G_m:

$$G_m(\mathbf{x}) = \prod_{j=1}^{n} [\![x_j \in S_j]\!]$$

where S_j is a subset of a domain of j-th attribute, and $[\![P]\!]$ is 1 if predicate P is satisfied, 0 otherwise. In other words, $G_m(\mathbf{x})$ indicates whether $\mathbf{x}$ belongs to the granule being the intersection of conditions $x_j \in S_j$. All granules $\{G_m\}_1^M$ are disjoint and cover the entire feature space, i.e., for any $\mathbf{x}$ there exists only and exactly one m for which $G_m(\mathbf{x}) = 1$. The prediction is computed as:

$$f(\mathbf{x}_{st}) = l_s \sum_{m=1}^{M} \alpha_m G_m(\mathbf{x}_{st}), \quad \alpha_m = \frac{\sum_{i=1}^N y_i G_m(\mathbf{x}_i)}{\sum_{i=1}^N l_i G_m(\mathbf{x}_i)} \tag{1}$$

where M is number of granules and α_m is an estimate of a travel time for a length unit computed as an average over training observations belonging to the m-th granule. The main problem is to determine the right granulation, to achieve the desired bias-variance trade-off, with coarse granules yielding biased predictions, while fine granules may yield high variance. In this work we use several simple groupings of attribute values for granules.

One can notice that formula (1) resembles prediction function used in decision tree [3] and rule models [6,5]. The main difference is that functions G_m are not directly induced, but given a priori and based on simple grouping of attribute values. Since in the considered case, there are only few attributes available and some of them are of specific kind like road segment id (a nominal attribute with a huge number of values) such an approach seems to be reasonable.

4 Bayesian Averaging

Bayesian averaging allows combining two models built on different levels of granulation, based on the variance and the number of observations in the granules. Let f_{s1} and f_{s2} be two granular models. The first model f_{s1} is assumed to be computed on coarse granules and can be treated as a prior expectation for a predicted value. In turn, f_{s2} gives a fine-grain prediction. Bayesian averaging is then defined by

$$f_s(\mathbf{x}) = (1-\lambda)f_{s1}(\mathbf{x}) + \lambda f_{s2}(\mathbf{x}),$$

where λ might be tuned empirically. However, we can determine λ using the Bayesian analysis as follows. Let μ_0 be the mean travel time within the coarse granule m_1. We assume that the mean travel time μ for each fine granule m_2 within coarse granule m_1 is drawn from a Gaussian distribution with mean μ_0. Then, the observations y_i in the fine granule m_2 are drawn from a Gaussian distribution with mean value μ. If the parameters of the distributions are estimated from the data, one can show [1], that the Bayesian posterior would lead to the following value of λ:

$$\lambda = \frac{\hat{\sigma}^2_{m_1}}{\hat{\sigma}^2_{m_1} + \hat{\sigma}^2_{m_2}/n_{m_2}}, \tag{2}$$

where $\hat{\sigma}^2_{m_1}$ is the variance of all observations within the coarse granule G_{m_1}, while $\hat{\sigma}^2_{m_2}$ is the variance of observations within the fine granule G_{m_2}, and n_{m_2} is number of observations in granule G_{m_2}. If the term $\hat{\sigma}^2_{m_2}/n_{m_2}$ is small comparing to $\hat{\sigma}^2_{m_1}$ (many observations and/or large variance within the coarse granule), then f_{s2} tends to dominate the final prediction, whereas if the term is high, then the overall predictions is strongly shrunk towards the coarse-grained model f_{s1}.

5 Matrix Factorization

Let us consider a matrix Y, which rows correspond to road segments and columns to 15 minute time slots defined over a week (672 time slots in total). The task is to find a compressed (storage efficient) an approximation of Y that also has all of the missing values from Y supplemented (which may be many due to sparsity of the data).

The approximation of Y can be given by a product of two low-rank matrices: $Y \simeq \hat{Y} = UM^T$ where U is an $S \times K$ and M is a $J \times K$ matrix, where S is the number of road segments, and J the number of time slots. K defines a rank of $\hat{Y}$ and can be seen as a number of latent features describing road segments and time slots.

The elements y_{sj} of matrix Y are mapped from training observations $\{(y_i, s_i, t_i)\}_{i=1}^N$, in such a way that a time point t_i is transformed to a corresponding time slot j. We will use $i \mapsto sj$ to express this mapping. Let us also note that elements of Y may, in fact, contain more than one entry, and many of the elements will contain no entry at all.

Formally, we try to find optimal matrices $U^* = [u_{sk}]$ and $M^* = [m_{jk}]$, which are the solution of:

$$(U^*, M^*) = \arg \min_{(U,M)} \mathcal{L}(Y, UM^T),$$

where

$$\mathcal{L}(Y, UM^T) = \sum_{i \mapsto sj} (y_{sj} - \sum_{k=1}^{K} u_{sk} m_{jk})^2.$$

In the above formulation, the rank K of matrix $\hat{Y}$ plays a role of regulariser that prevents overfitting of the model to training data. An additional regularisation is usually introduced in this type of algorithms. For example, the Frobenius norm of matrices U and M:

$$\sum_{sk} u_{sk}^2, \qquad \sum_{jk} m_{jk}^2,$$

respectively, is usually minimised along with the error term. This prevents the coefficients to have values far from zero.

The Frobenius regularisation for segments, however, requires some calibration of the matrix entries to justify regularization to zero. In the following, we set up the first vector of U to be equal to the original length of the segment: $u_{s1} = l_s$. Values for m_{j1} can be then interpreted as average travel times for a length unit (i.e., average inverse velocity).

In turn, for time slots we control the difference between consecutive time slots, as the time slots close to each other should result in a similar prediction. This can be achieved by smoothing a prediction over time slots using a specific regularisation term of the following form:

$$\sum_{jk} (m_{jk} - m_{(j-1)k})^2 = \sum_{jk} m_{jk}^2 + m_{(j-1)k}^2 - 2 m_{jk} m_{(j-1)k},$$

where we assume that if $j = 1$ then $j - 1 = J$, as also if $j = J$, then $j + 1 = 1$. Thus, the function $\mathcal{L}$ to be minimised is given by:

$$\mathcal{L}(Y, UM^T) = \sum_{i \mapsto sj} (y_{sj} - \sum_{k=1}^{K} u_{sk} m_{jk})^2 + \lambda_1 \sum_{sk} u_{sk}^2 + \lambda_2 \sum_{sk} (m_{jk} - m_{(j-1)k})^2$$

The parameters of the problem are then K, as also λ_1 and λ_2 that controls the strength of regularisation.

The regularised learning problem is unfortunately non-convex, thus the minimisation of our objective $\mathcal{L}(Y, UM^T)$ requires a special method to solve. Our method is iterative. We first set up $u_{s1} = l_s$ for all s. Then, in each iteration ($k = 1, \ldots, K$) we compute u_{sk} (except u_{s1}) and m_{jk}, for all s and j, by minimising:

$$\mathcal{L}^k = \sum_{i \mapsto sj} (\Delta y_{sj} - u_{sk} m_{kj})^2 + \lambda_1 \sum_{s} u_{sk}^2 + \lambda_2 \sum_{j} (m_{jk} - m_{(j-1)k})^2,$$

where

$$\Delta y_{sj} = y_{sj} - \sum_{k'=1}^{k-1} u_{sk'} m_{jk'}.$$

Minimising $\mathcal{L}^k$ is still hard and requires an iterative procedure, which is guaranteed to converge to a local minimum. In each iteration of the procedure, we first find the optimal u_{sk}, for all s, given fixed m_{jk}. Next, we fix new values of u_{sk} and find the optimal m_{jk} for all j. This process is repeated until convergence, and usually only several iterations are sufficient to get a stable solution.

The solution for u_{sk} given fixed m_{jk} can be obtained by taking the negative gradient of $\mathcal{L}^k$ and setting it to 0:

$$-\frac{\partial \mathcal{L}}{\partial u_{sk}} = 2\sum_j (\varDelta y_{sj} - u_{sk} m_{jk}) m_{jk} - 2\lambda_1 u_{sk} = 0 \quad s = 1, \dots, S.$$

It is easy to see that the solution is given by:

$$u_{sk} = \frac{\sum_j \varDelta y_{sj} m_{jk}}{\sum_j m_{jk}^2 + \lambda_1} \quad s = 1, \dots, S. \tag{3}$$

The penalty term for m_{jk} is more complex and couples consecutive time slots. The negative gradient is given by:

$$-\frac{\partial \mathcal{L}}{\partial m_{sj}} = 2\sum_s (\varDelta y_{sj} - u_{sk} m_{jk}) u_{sk} - 2\lambda_2 (2m_{jk} + m_{(j-1)k} + m_{(j+1)k}),$$

where $j = 1, \dots, J$. By setting all negative gradients to 0 we end up with the following system of linear equations:

$$m_{jk} + \frac{\lambda_2 (m_{(j-1)k} + m_{(j+1)k})}{2(\sum_s u_{sk}^2 + \lambda_2)} = \frac{\sum_s \varDelta y_{sj} u_{sk}}{\sum_s u_{sk}^2 + \lambda_2} \quad j = 1, \dots, J. \tag{4}$$

We solve this system of linear equation by using Gauss-Seidel method which proceeds in an iterative way.

6 Experimental Results

6.1 Data and Methodology

Real GPS floating car data used in the experiment were delivered by NaviExpert, a Polish car navigation company. The data consist of map-matched (projected to road segments) travel time observations, associated with a time stamp and additional attributes such as the length of a road segment, road category (highway, freeway, urban road, etc.), type of surrounding area (city, village, out-of-the-city), geographical coefficients, etc.

The observations cover the city of Warsaw, the capital of Poland, with surroundings — a rectangular envelope with a side of about 85km around the centre at 52.2391°N 21.0227°E, which constitutes an area of above 7000km^2. The observations span from the 1st of September 2009 to the 31st of December 2009. In total, the data set contains 6 808 061 observations that are sparse and unevenly distributed in time and space.

Table 1. RMSE for static models with varying size of training set: we used 6.25%, 12.5%, 25%, 50%, 100% of training instances. RMSE is given in percent ([%]) in comparison to the global mean model and in minutes ([min])

model	$6.25\%N$		$12.5\%N$		$25\%N$		$50\%N$		$100\%N$	
	[%]	[min]	[%]	[min]	[%]	[min]	[%]	[min]	[%]	[min]
GM	100.00	0.391	100.00	0.390	100.00	0.391	100.00	0.394	100.00	0.395
LLG	95.94	0.375	96.08	0.374	95.90	0.375	95.55	0.376	95.37	0.377
RSG	89.97	0.352	88.57	0.345	87.01	0.340	85.27	0.336	83.70	0.331
BA	89.42	0.349	88.45	0.345	87.06	0.340	85.43	0.336	84.00	0.332
MF	89.69	0.351	87.67	0.342	84.84	0.332	82.37	0.324	80.42	0.318

The models were trained and tuned on the training set. Then, the resulting models were evaluated on the testing set. The training set covered the first three months of data while the test set covered the whole month of December 2009. As performance measure we used the root mean square error (RMSE).

6.2 Results

In the experiment we compared the following four models: low-level granular (LLG) model, single road segment (SRG) model, Bayesian averaging (BA) of two above models, and the matrix factorization (MF). All the results are related to the simple global mean (GM) model, for which the average of travel times was computed over all training observations.

LLG is a simple granular model in which a granule is built as a combination of values of the following attributes: time periods (possible values: morning, noon, afternoon, nights-and-weekends), type of the road (possible values: highway, main road, normal), and type of surrounding area (possible values: town, village, out-of-city). SRG is also a simple granular model which prediction is computed as average travel time over all observations from a given road segment. In case of a low traffic on a segment, the prediction may have a high variance, which is the main drawback of this model. To overcome the above problem, we used BA to combine LLG and SRG models with each other.

Obviously, MF is the most complex model used in this experiment that requires tuning of additional parameters. In this regards, we isolated from the training set a validation part that contained observations from November. However, computational costs of this algorithm are not very high. The single run of the algorithm takes around 20 seconds on Inter Core2 2.4GHz machine with 2 GB of RAM. We computed up to $K = 20$ factors for matrices U and M. We found the optimal value $K = 10$ on the validation set. Regularisation parameters were also optimized on the validation set.

Table 1 shows the results given in plain RMSE (in minutes) and the percentage RMSE in relation to the GM model's prediction error for different sizes of

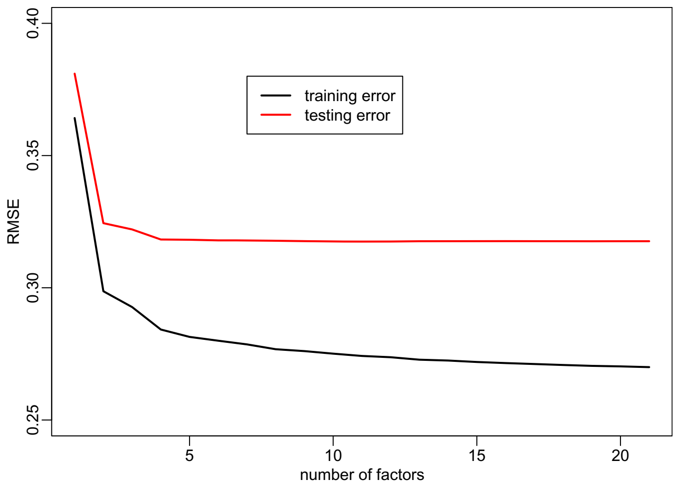

Fig. 1. Training and testing RMSE of MF model are given as a function of K, i.e., number of factors of matrices U and M

training set. We varied the size of training set by taking from 6.25% to 100% of training instances. One can easily observe that the MF model performs the best. In the case of small training set, BA has a small advantage over MF and SRG. If the size of training size increases, then SRG and MF starts to outperform BA, however, in the case of the latter the improvement is more pronounced. These three models are significantly better than GM and LLG models. In Figure 1 we show performance on training and testing set of MF with respect to number of factors K. Significant improvement can be noticed up to 5 factors. For next factors, RMSE computed on testing set does not decrease, however, no overfitting occurs.

Let us also underline that the MF model has an additional advantage that the latent features discovered by matrix factorization can be nicely interpreted. Figure 2 shows the first 5 factors of matrix M and distribution of values on the corresponding 5 factors of matrix U. The first factor (black) can be interpreted as a average travel time for a length unit (inverse average velocity). The next factors can be interpreted as changes in travel times depending on a type of a road segment. The second factor (red) seems to indicate road segments that are sensitive for traffic congestion in morning and afternoon hours of working days. The third (green) and fourth (blue) factor indicate in- and out-of-city segments, while the fifth factor (light-blue) indicates "Friday afternoon and weekend" roads.

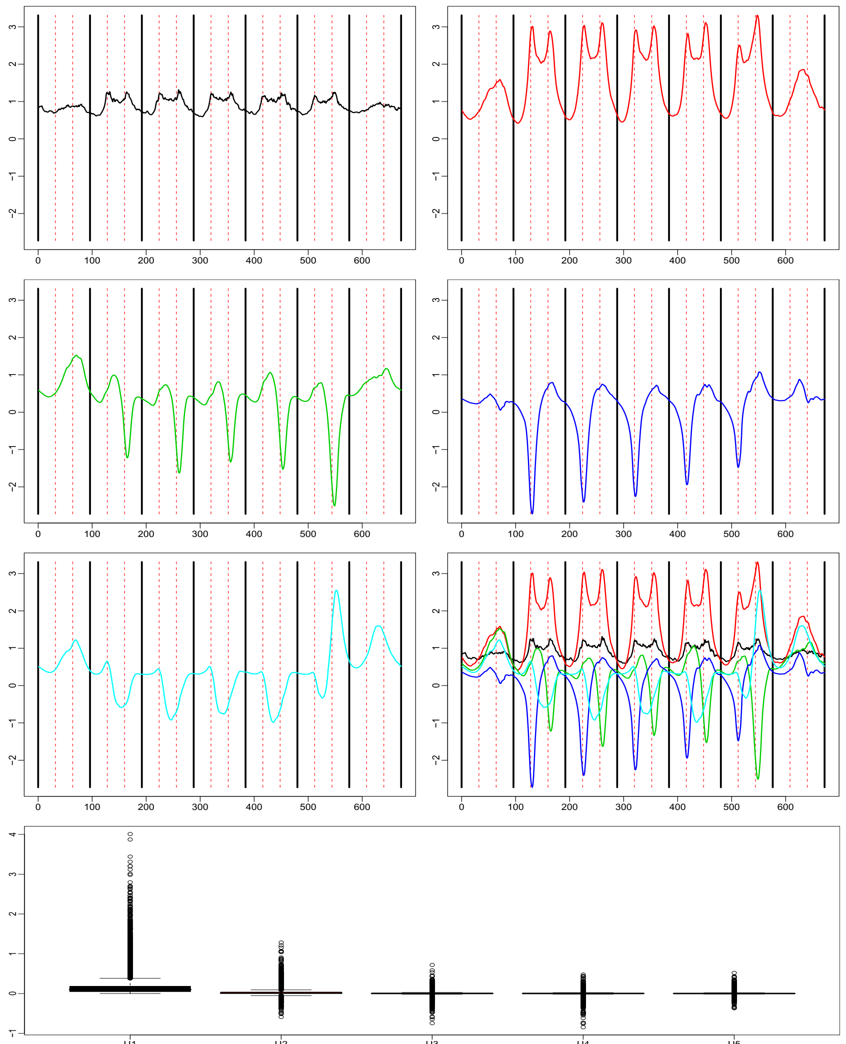

Fig. 2. From top: first five factors of matrix M that represents different characteristics of road segments. A unit of x-axis corresponds to an interval of 15 minutes (thus, a range of the x-axis is from 1 to 672). At bottom: box plot (with outliers) that shows distribution of values on corresponding factors of matrix U.

7 Conclusions

We shown that matrix factorization techniques can be successfully applied in the context of travel time estimation for large traffic networks. In the exhaustive experiment on real-world data the matrix factorization outperformed standard approaches to this problem being still competitive in terms of computational costs. Moreover, the latent features obtained as a by-product of the matrix factorization give us further insight into the problem and can be valuable source of information. There are also many possible directions for the future research. The static model could be improved by treating some road segments separately. Specific segments with dense data could also be modelled differently, e.g. by kernel estimation methods or locally weighted regression.

Acknowledgements. This work has been supported by the Polish National Science Centre, grant no. N N519 441939.

References

1. Berger, J.O.: Statistical Decision Theory and Bayesian Analysis, 2nd edn. Springer (1993)
2. Billings, D., Yang, J.: Application of the ARIMA Models to Urban Roadway Travel Time Prediction-A Case Study. In: IEEE International Conference on Systems, Man and Cybernetics, SMC 2006, vol. 3 (2006)
3. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and Regression Trees. Wadsworth (1984)
4. Brosch, P.: A service oriented approach to traffic dependent navigation systems. In: IEEE Congress on Services - Part I, pp. 269–272 (2008)
5. Dembczyński, K., Kotłowski, W., Słowiński, R.: Ender - a statistical framework for boosting decision rules. Data Mining and Knowledge Discovery 21(1), 52–90 (2010)
6. Friedman, J.H., Popescu, B.E.: Predictive learning via rule ensembles. Annals of Applied Statistics 2(3), 916–954 (2008)
7. Hiramatsu, A., Nose, K., Tenmoku, K., Morita, T.: Prediction of travel time in urban district based on state equation. Electronics and Communications in Japan 92(7), 1–11 (2009)
8. Liu, H., Lint, H.v., Zuylen, H.v., Zhang, K.: Two distinct ways of using kalman filters to predict urban arterial travel time. In: IEEE Intelligent Transportation Systems Conference, ITSC 2006., pp. 845–850. IEEE (2006)
9. Paterek, A.: Improving regularized singular value decomposition for collaborative filtering. In: Proc. KDD Cup Workshop at SIGKDD 2007, 13th ACM Int. Conf. on Knowledge Discovery and Data Mining, pp. 39–42 (2007)
10. Rice, J., Zwet, E.V.: A simple and effective method for predicting travel times on freeways. IEEE Transactions on Intelligent Transportation Systems, 5(3), 200–207 (2004)
11. Srebro, N., Rennie, J., Jaakkola, T.: Maximum Margin Matrix Factorizations. In: Proc. of Advances in Neural Information Processing Systems (2005)

12. Van Lint, J.: Reliable real-time framework for short-term freeway travel time prediction. Journal of Transportation Engineering 132, 921 (2006)
13. Van Lint, J., Hoogendoorn, S., Van Zuylen, H.: Accurate freeway travel time prediction with state-space neural networks under missing data. Transportation Research Part C 13(5-6), 347–369 (2005)
14. Wan, K., Kornhauser, A.: Turn-by-turn routing decision based on copula travel time estimation with observable floating-car data. In: Transportation Research Board 89th Annual Meeting. No. 10-2723 (2010)