Trajectory Regression on Road Networks

Tsuyoshi Idé[*] and **Masashi Sugiyama**[†]

[*]IBM Research - Tokyo, goodidea@jp.ibm.com
[†] Department of Computer Science, Tokyo Institute of Technology, sugi@cs.titech.ac.jp

Abstract

This paper addresses the task of trajectory cost prediction, a new learning task for trajectories. The goal of this task is to predict the cost for an arbitrary (possibly unknown) trajectory, based on a set of previous trajectory-cost pairs. A typical example of this task is travel-time prediction on road networks. The main technical challenge here is to infer the costs of trajectories including links with no or little passage history. To tackle this, we introduce a weight propagation mechanism over the links, and show that the problem can be reduced to a simple form of kernel ridge regression. We also show that this new formulation leads us to a unifying view, where a natural choice of the kernel is suggested to an existing kernel-based alternative.

Introduction

Traffic systems have provided the artificial intelligence (AI) community with a number of interesting research topics to date. Traffic volume modeling is a traditional area of AI research (Sekine 1994). Autonomous driving (Urmson et al. 2009) and a stochastic formulation of route planning (Nikolova and Karger 2008) are other recent topics.

This paper addresses a new learning task in traffic systems. We wish to predict the cost of an arbitrary (possibly unknown) trajectory, or a path, on a network. Figure 1 illustrates our problem. We are given a set of N trajectory-cost pairs as the training data,

$$\mathcal{D} \equiv \{(x^{(n)}, y^{(n)}) \mid n = 1, 2, ..., N\}, \qquad (1)$$

where each $x^{(n)}$ represents the n-th trajectory (a sequence of links), and $y^{(n)}$ represents an observed value of the cost of $x^{(n)}$. Our task is to predict the cost of an input trajectory x, which should not be assumed to be exactly included in the data. Clearly, this problem can be viewed as a special type of regression for moving patterns of individual vehicles, and we call this task *trajectory regression*.

Travel-time prediction is a typical application of this task. Note that the problem setting of trajectory cost prediction is of crucial importance in practice. While recent Global Positioning Systems (GPS) allow us to measure spatio-temporal locations, it is extremely hard to obtain complete trajectories in reality. This is mainly due to finite space and time resolutions of GPS, and frequent and unavoidable electromagnetic interference, especially in urban areas (Greaves and Figliozzi 2008). In our problem setting, what we assume in the training set is only the *total* cost of each individual trajectory, which can be easily and inexpensively measured and recorded.

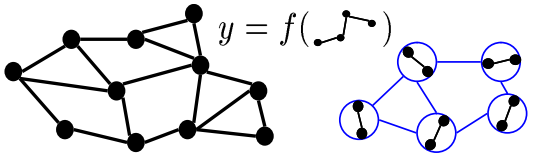

Figure 1: (a) Problem setting. From a set of trajectory-cost pairs, we predict the cost y of an arbitrary input trajectory x. A trajectory refers to a sequence of connected links. (b) Link network defined by the link similarity matrix S.

In a specific context of travel-time prediction, the use of Gaussian process regression (GPR) has been proposed for trajectory regression by Idé and Kato (2009). However, a number of important issues remain unsolved. First, in that work, *all* the trajectories in the training data are assumed to share the *same* origin and destination, which limits practical applicability to a great extent. Second, in the GPR formulation, it is not clear what kind of kernels should be chosen and why.

In this paper, we clearly answer these questions by introducing a new formulation of trajectory regression. Starting from the same noise model as used in GPR, we introduce a mechanism for weight propagation over links as a natural formalization of the intuition that the links neighboring a congested link will also be congested. We show that this task of trajectory cost prediction can be rewritten as a simple form of regularized least squares regression. Finally, we show that our formulation leads to a natural choice of the kernel, and clearly outperforms the GPR with the ad hoc chosen string kernel. Through this analysis, we present the readers with a unifying view for the task of trajectory cost prediction. To the best of our knowledge, this is the first work that clarifies the relationship between the kernel approach and weight propagation over a network in trajectory

regression.

Problem Setting

As described in Introduction, we assume that we are given a data set of N trajectory-cost pairs as Eq. (1). Our goal is to predict the *cost* y spent along a given trajectory x including possibly unseen links on the network.

To make the problem tractable, we restrict our attention to trajectories on networks, where a trajectory is represented as a sequence of link identifications (IDs). We denote the number of links in the entire network by M. Even in this case, however, the dataset $\mathcal{D}$ cannot span all of the possible trajectories since the number of possible trajectories in a network is of a factorial order. Thus, simple strategies such as k-nearest neighbor regression will not work. In addition, in realistic situations, the distribution of the amount of traffic is not at all uniform. In other words, some links have an extremely high number of passages, while others have almost no passages, as shown later by our experiments.

To summarize, main technical challenges are (1) how to handle trajectories including unseen links, and (2) how to handle heterogeneity over the link traffic.

Weight Propagation Model

This section explains how to formalize our task of trajectory cost prediction. First, we look at how to parameterize the link cost. Then we consider the important issue of how to handle little- or no-passage links, and introduce a weight propagation mechanism.

Parameterizing link cost

We start with the assumption about the observation noise for the cost y of a trajectory x as used in the GPR formulation:

$$p(y|x) = \mathcal{N}(y|m(x), \sigma^2), \tag{2}$$

where $\mathcal{N}(\cdot|m, \sigma^2)$ denotes a Gaussian distribution with the mean m and the variance σ^2. To consider the functional form of $m(x)$, let us first look at the simplest situation. If there is no interaction between the moving objects, the mean trajectory cost will be simply given as

$$m_0(x) \equiv \sum_{e \in x} l_e \phi_e^0, \tag{3}$$

where l_e is the length or weight of a link e, and ϕ_e^0 (which we assume known) is a constant representing the cost spent per unit length at e under the condition of no interaction. In travel-time prediction, the total time spent is the sum over the contributions of the individual links.

If an interaction exists between the objects, $m_0(x)$ will not be a good approximation. To handle the *deviation* from the baseline cost, we introduce state variables $\{f_1, ..., f_M\}$ for the individual links. Then we can generally express the cost $m(x)$ as

$$m(x|\boldsymbol{f}) = \sum_{e \in x} l_e(\phi_e^0 + f_e), \tag{4}$$

where we explicitly denoted in m the dependency on $\boldsymbol{f} \equiv [f_1, ..., f_M]^\top$. This is our basic model for $m(x)$. One might include nonlinear terms such as $\sum_{e,e' \in x} l_{e,e'} f_e f_{e'}$. However, we believe that the main contribution is captured by the linear term as in Eq. (4) since such a nonlinear term is not consistent with the intuition that $m(x)$ is roughly proportional to $|x|$, the number of links in x. Instead of using a complex nonlinear model in $m(x)$, we take account of the effect of inter-link interactions in the framework of weight propagation as discussed later.

Loss function

Now our goal is to determine the state variables $\{f_e\}$ from the data $\mathcal{D}$. Based on the Gaussian observation model, one natural approach for learning $\{f_e\}$ is the maximum likelihood. Equivalently, one can minimize the negative log-likelihood given by

$$L(\mathcal{D}, \boldsymbol{f}) \equiv \sum_{n=1}^{N} \left(y^{(n)} - m(x^{(n)}|\boldsymbol{f}) \right)^2, \tag{5}$$

where we omitted constant terms and the unimportant prefactor. This is a well-known quadratic loss function.

By using Eq. (4), it appears that minimizing this loss function is sufficient to find the solution. However, this is not true in general. As mentioned before, the trajectories in $\mathcal{D}$ are not expected to include all of the links. Some of the links do not appear in the loss function and are left undetermined. It is clear that using the baseline costs for such links is not a good solution, since such static information does not reflect any changes in the actual traffic flow.

Predicting the cost of a trajectory including any unseen links appears to be difficult. However, introducing another assumption that we call weight propagation makes it possible. We look at this approach in the next subsection.

Propagation penalty

One of the most important features of networks is the fact that an event at one location can propagate to the neighboring locations through the connecting links. Thus, if a link e has a significant deviation from its baseline state, then e's neighboring links are expected to be influenced by that large deviation. This effect is easily understood by thinking about traffic jams. For example, if 5th Avenue in Manhattan, New York, becomes jammed by traffic, then we naturally expect traffic jams on neighboring Park Avenue and 53rd Street. This suggests that the state variables should satisfy a smoothness condition while minimizing the loss function.

Encouraged by this analysis, we introduce a penalty term into the loss minimization problem,

$$R(\boldsymbol{f}) \equiv \frac{1}{2} \sum_{e=1}^{M} \sum_{e'=1}^{M} S_{e,e'} \left| f_e - f_{e'} \right|^2, \tag{6}$$

where M is the number of links, and the $M \times M$ matrix S is an affinity matrix between the *links* (see Fig. 1 (b)). One reasonable choice of S is a finite-range exponential function such as

$$S_{e,e'} \equiv \begin{cases} 1 & e = e', \\ \omega^{d(e,e')} & d(e,e') \le d_0, \\ 0 & \text{otherwise}, \end{cases} \tag{7}$$

where $S_{e,e'}$ is the (e, e') element of S, and ω is a positive constant less than 1. The exponent $d(e, e')$ is the distance between e and e'. The simplest choice for this is to use one plus the minimum number of hops from e to e'. For example, a link e' sharing the same intersection with e has 0 hops to e, so $d(e, e') = 1$ and $S_{e,e'} = \omega$. While the threshold d_0 has several possibilities, our observation shows that a small value such as 2 or 3 works in many cases if λ is optimized appropriately. Note that a finite value of d_0 corresponds to the common sense notion that a traffic blockage at 5th Avenue will not affect streets in Los Angeles, California. Also, our formulation allows us to handle directional networks by treating individual directions as different links. Thus we assume $S_{e,e'} = S_{e',e}$ hereafter.

Clearly, $R(\boldsymbol{f})$ controls the smoothness over the links. If one link has a large difference in f_e compared to a neighboring link, then $R(\boldsymbol{f})$ imposes a large penalty. In contrast, if the values of the state variables are the same in neighboring links, then this term makes no contribution.

Combining Eqs. (5) and (7), our final objective function is written as

$$\Psi(\boldsymbol{f}|\lambda) = \sum_{n=1}^{N} \left(y^{(n)} - m(x^{(n)}|\boldsymbol{f}) \right)^2 + \frac{\lambda}{2} \sum_{e=1}^{M} \sum_{e'=1}^{M} S_{e,e'} \left| f_e - f_{e'} \right|^2, \quad (8)$$

where λ is a constant that controls the tradeoff between the penalty and the loss, and is determined based on cross validation as explained later.

Matrix representation of the objective

For deeper insight, we derive a matrix representation of Eq. (8). First, for the penalty term, simply by expanding the square, we see that

$$R(\boldsymbol{f}) = \boldsymbol{f}^\top \mathsf{L} \boldsymbol{f}, \quad (9)$$

where L is defined as $\mathsf{L}_{i,j} \equiv \delta_{i,j} \sum_{k=1}^{M} S_{i,k} - S_{i,j}$, and $\delta_{i,j}$ is Kronecker's delta function. This is the definition of the graph Laplacian induced by the affinity matrix S.

Second, for the loss term, let $\tilde{y}^{(n)}$ be

$$\tilde{y}^{(n)} \equiv y^{(n)} - \sum_{e \in x^{(n)}} l_e \phi_e^0, \quad (10)$$

and define $\boldsymbol{y}_N \in \mathbb{R}^N$ as $\boldsymbol{y}_N \equiv [\tilde{y}^{(1)}, \tilde{y}^{(2)}, ..., \tilde{y}^{(N)}]^\top$. In addition, define a link-indicator vector for each trajectory as

$$q_e^{(n)} = \begin{cases} l_e, \text{ for } e \in x^{(n)}, \\ 0, \text{ otherwise.} \end{cases} \quad (11)$$

If we write the matrix whose column vectors are $\{q_e^{(n)}\}$ as $\mathsf{Q} \equiv [\boldsymbol{q}^{(1)}, ..., \boldsymbol{q}^{(N)}] \in \mathbb{R}^{M \times N}$, then we can easily see that the loss function is represented as

$$L(\mathcal{D}, \boldsymbol{f}) = \left\| \boldsymbol{y}_N - \mathsf{Q}^\top \boldsymbol{f} \right\|^2. \quad (12)$$

Putting these equations together, the objective function Eq. (8) can now be written as

$$\Psi(\boldsymbol{f}|\lambda) = \left\| \boldsymbol{y}_N - \mathsf{Q}^\top \boldsymbol{f} \right\|^2 + \lambda \boldsymbol{f}^\top \mathsf{L} \boldsymbol{f}. \quad (13)$$

This is the objective function of regularized least squares regression, and the global optimum solution $\boldsymbol{f}^*$ is given as the solution of the simultaneous equation

$$\left[\mathsf{Q}\mathsf{Q}^\top + \lambda \mathsf{L} \right] \boldsymbol{f} = \mathsf{Q} \boldsymbol{y}_N. \quad (14)$$

Fortunately, $\mathsf{Q}\mathsf{Q}^\top + \lambda \mathsf{L}$ is quite sparse, real, and symmetric, and this equation can be efficiently solved with the use of iterative algorithms such as conjugate gradient (CG) (Golub and Loan 1996).

Optimizing λ

So far, we have treated the regularization parameter λ as a fixed constant. This parameter, however, can be learned from the data by minimizing the leave-one-out cross-validation (LOO CV) error,

$$L_{\mathrm{LOOCV}}(\lambda) \equiv \frac{1}{N} \sum_{n=1}^{N} \left[\tilde{y}^{(n)} - \boldsymbol{q}^{(n)\top} \boldsymbol{f}^{(-n)} \right]^2 \quad (15)$$

with respect to λ. Here, $\boldsymbol{f}^{(-n)}$ is the solution of Eq. (14) for the n-th leave-one-out data defined as $\{(x^{(n)}, y^{(n)}) \mid n = 1, 2, ..., n-1, n+1, ..., N\}$. In the case of regularized least squares regression, it can be shown that one can remove the summation over n to give

$$L_{\mathrm{LOOCV}}(\lambda) = \frac{1}{N} \left\| [\mathrm{diag}(\mathsf{I}_M - \mathsf{H})]^{-1} (\mathsf{I}_M - \mathsf{H}) \boldsymbol{y}^{(n)} \right\|^2, \quad (16)$$

where H is defined by $\mathsf{H} \equiv \mathsf{Q}^\top [\mathsf{Q}\mathsf{Q}^\top + \lambda \mathsf{L}]^{-1} \mathsf{Q}$. Also, I_M is the M-dimensional identity matrix, and diag is the operator to set all of the off-diagonal elements to zero. To find the optimal λ, we need to repeatedly evaluate the value of this LOO CV score for a number of different values of λ.

Algorithm summary

In the *training phase*, we first determine the best λ based on the LOO CV cost function. Then we solve a regularized least squares regression problem to get the state variable $\boldsymbol{f}$.

1. Input: $\mathsf{Q}, \boldsymbol{y}_N$, and the Laplacian matrix L.
2. Algorithm:
   - Find λ^* that minimizes L_{LOOCV} (Eq. (16)).
   - Solve Eq. (14) using the λ^*.
3. Output: $\boldsymbol{f}$.

In the training phase, the most expensive step is the LOO CV to optimize λ. In this step, for each value of λ, we need to solve $[\mathsf{Q}\mathsf{Q}^\top + \lambda]\mathsf{X} = \mathsf{Q}$ w.r.t. X so that $\mathsf{H} = \mathsf{Q}^\top \mathsf{X}$. While this is an $O(M^3)$ operation in general, thanks to the sparsity of $\mathsf{Q}\mathsf{Q}^\top + \lambda$, the CG algorithm allows to perform this step in $O(M^2)$ in practice, provided that the number of nonzero entries is of $O(M)$. The memory requirement is also $O(M^2)$.

In the *test phase*, for an input trajectory x, we compute

$$y = \sum_{e \in x} l_e (f_e + \phi_e^0).$$

Given the $\boldsymbol{f}$ vector, this is of only $O(|x|)$ operations.

We call the present algorithm **RETRACE** (Regularized least squares rEgression for TRAjectory Cost prEdiction) hereafter. Note that RETRACE can be used for both directed and undirected networks if S is properly defined.

Discussion

This section includes a detailed discussion going beyond the previous section with a particular focus on the relationship between the present approach and a Gaussian process regression (GPR) approach (Idé and Kato 2009).

Connection to Gaussian process regression

We have shown that the trajectory cost prediction is reduced to a form of regularized least squares regression. Now it is interesting to explore the relationship with the GPR formulation. First of all, by explicitly writing down the likelihood $p(\boldsymbol{y}_N|\boldsymbol{f})$, we easily see that the following Proposition holds.

Proposition 1 *The optimization problem to minimize Eq.* (13) *is formally equivalent to a MAP (maximum a posteriori) estimation by an improper prior*

$$p(\boldsymbol{f}) \equiv \mathcal{N}(\boldsymbol{f}|\boldsymbol{0}, \mathsf{L}^\dagger), \tag{17}$$

where $\mathsf{L}^\dagger$ *the pseudo-inverse of* L.

Although the prior Eq. (17) is improper due to a zero eigenvalue of L, it has been used to represent spatial smoothness in semi-supervised learning tasks (see, e.g., (Kapoor et al. 2006)).

Encouraged by this equivalence, let us find the Bayesian predictive distribution based on the prior Eq. (17). Since the observation model is Gaussian, the predictive distribution is analytically calculated. The result is

$$\begin{aligned} p(y|x, \boldsymbol{y}_N) &= \int \mathrm{d}\boldsymbol{f} \;\; p(\boldsymbol{f}|\boldsymbol{y}_N)\mathcal{N}(\tilde{y}|m(x|\boldsymbol{f}), \sigma^2) \\ &= \mathcal{N}\left(\tilde{y}\,|\boldsymbol{k}_q^\top \mathsf{C}^{-1}\boldsymbol{y}_N, \sigma^2 + k_q - \boldsymbol{k}_q^\top \mathsf{C}^{-1}\boldsymbol{k}_q\right), \end{aligned} \tag{18}$$

where we defined $\tilde{y}$ in the same way as Eq. (10), and

$$\begin{aligned} \boldsymbol{k}_q \equiv \mathsf{Q}^\top \mathsf{L}^\dagger \boldsymbol{q}, \quad k_q \equiv \boldsymbol{q}^\top \mathsf{L}^\dagger \boldsymbol{q} \\ \mathsf{C} \equiv \sigma^2 \mathsf{I}_N + \mathsf{K}_Q, \quad \mathsf{K}_Q \equiv \mathsf{Q}^\top \mathsf{L}^\dagger \mathsf{Q}. \end{aligned}$$

Here $\boldsymbol{q}$ is the link-indicator vector corresponding to x. By comparing this result with the standard formula of GPR (Rasmussen and Williams 2006), the following Proposition is proved:

Proposition 2 *RETRACE is equivalent to the predictive mean of GPR with a kernel matrix whose* (n, n') *element is given by*

$$\mathsf{K}_{n,n'} = \boldsymbol{q}^{(n)\top} \mathsf{L}^\dagger \boldsymbol{q}^{(n')}. \tag{19}$$

Proposition 2 is important in that it bridges the gap between a kernel matrix and the link similarity matrix. In GPR, one can choose any kernel functions, and a string kernel was chosen in an ad hoc manner in the previous work. A kernel function would give a good performance if it is consistent with the link similarities in the sense of Eq. (19). However, there is no guarantee at all that an arbitrarily chosen kernel gives spatial smoothing consistent with the network topology. In contrast, the present formulation naturally leads to a particular kernel as long as the correct link similarities are given. Since choosing the similarity matrix is much easier than choosing the kernel matrix, we conclude that the present formulation is more practical than the previous work.

Table 1: Summary of data.

	`Grid25`	`Kyoto`
# nodes	625	1 518
# links	2 400	3 478
# generated paths	1 200	1 739

Properties of pseudo-inverse of Laplacian

In Eq. (19), the pseudo-inverse of the Laplacian plays a central role. It is known that its inverse allows an interesting interpretation as the *commute time* (Fouss et al. 2007). In particular, if we define a dissimilarity matrix D from $\mathsf{K}_{n,n'}$ in a standard way as $\mathsf{D}_{n,n'} \equiv \mathsf{K}_{n,n} + \mathsf{K}_{n',n'} - 2\mathsf{K}_{n,n'}$, we see that

$$\mathsf{D}_{n,n'} = (\boldsymbol{q}^{(n)} - \boldsymbol{q}^{(n')})^\top \mathsf{L}^\dagger (\boldsymbol{q}^{(n)} - \boldsymbol{q}^{(n')}). \tag{20}$$

This relationship is essentially the same as the definition of the commute time (Fouss et al. 2007) between n and n'

$$(\boldsymbol{e}^{(n)} - \boldsymbol{e}^{(n')})^\top \mathsf{L}^\dagger (\boldsymbol{e}^{(n)} - \boldsymbol{e}^{(n')}),$$

where $\boldsymbol{e}^{(n)}$ is the indicator vector of the n-th "node". Since the $\boldsymbol{q}^{(n)}$s are the link-indicators of the sample trajectories in our case, $\mathsf{D}_{n,n'}$ can be interpreted as the "commute time" between the n- and n'-th trajectories.

Experiments

In this section, we show some experimental results for travel-time prediction. [1]

Road network and trajectory data

To apply our algorithm, we need road network data and trajectory cost data. For the road network data, we used two networks, as summarized in Table 1. The `Grid25` network is a synthetic square grid with a 25×25 two-dimensional lattice structure. Each of the interior nodes is connected with the neighboring four nodes, and the number of edges is $2\,400$ (note that the links are directed). To be realistic, we set the length and the legal speed limit of each edge to 100 m and 37.5 km/h, respectively.

As for the `Kyoto` network, we used a real digital map of Kyoto City, the same as the one used in (Idé and Kato 2009). Figure 2 shows the network, which contains $1\,518$ nodes and $3\,478$ directed edges. Each edge is associated with its actual length and legal speed limit, as given by the digital map data.

To test the capability of our algorithm, we generated trajectory-cost data using IBM Megaffic Simulator (IBM 2008), which is carefully designed to be capable of reproducing all of the major features of real traffic including interactions between vehicles as well as nondeterministic fluctuations of vehicle behaviors. For `Grid25`, the destination node of each trajectory was set to the central node, whereas the origin node is chosen randomly during the simulation. Thanks to this choice, links around the center of the map are heavily congested. For `Kyoto`, origin and destination

[1] The data used and the detail of the simulation setup are available at `http://ide-research.net/datasets.html`.

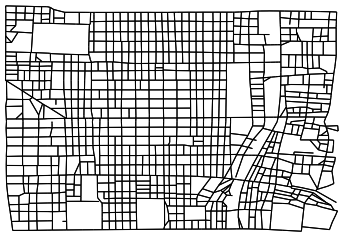

Figure 2: Downtown Kyoto City map for `Kyoto` data.

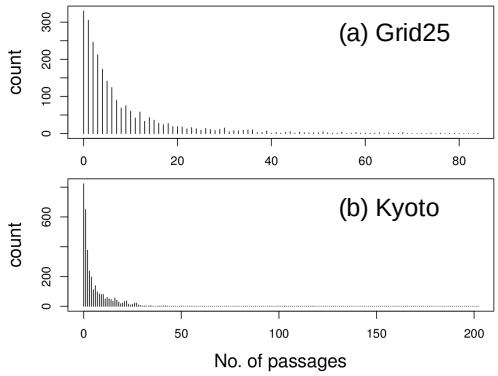

Figure 3: The distribution of the number of passages per link: (a) `Grid25` with $N = 1\,200$, and (b) `Kyoto` with $N = 1\,739$. We see that most of the links have only a few passages.

are chosen randomly. The number of generated trajectories are summarized in Table 1. The reader can refer to the Web page for the detail of the simulation setup. [1]

Methods compared

We compared three approaches for travel time prediction: `RETRACE`, `Legal`, and `GPR`. In `RETRACE`, we fixed $d_0 = 2$ and $\omega = 0.5$. It is confirmed that the result is not sensitive to these parameters if λ is optimized appropriately.

The `Legal` approach is a simplistic method that computes the travel-time of a trajectory using only static information (Eq.(3)). This is equivalent to $f_e = 0$ for all links. For the baseline time, we simply used twice the values computed from the legal speed limits. This is a natural assumption in urban traffic since a vehicle takes twice the time spent with a constant velocity of the legal speed limit, if it accelerates from zero at one end to the legal speed limit and then decelerates to zero towards the other end of the link.

The `GPR` method is Gaussian process regression. We used the p-spectrum kernel with $p = 2$, which gave the best performance in the previous work. However, we confirmed that the general tendency is essentially the same for other choices of p. Hyper-parameters in the model were determined based on the framework of evidence approximation.

Results

We first look at some statistics for the data. Figure 3 shows the distribution of the number of passages per link. We see that the most of the counts are concentrated at the smallest numbers. In fact, 823 links out of the $3\,478$ links do not have any traffic in `Kyoto`. From this distribution, we see that

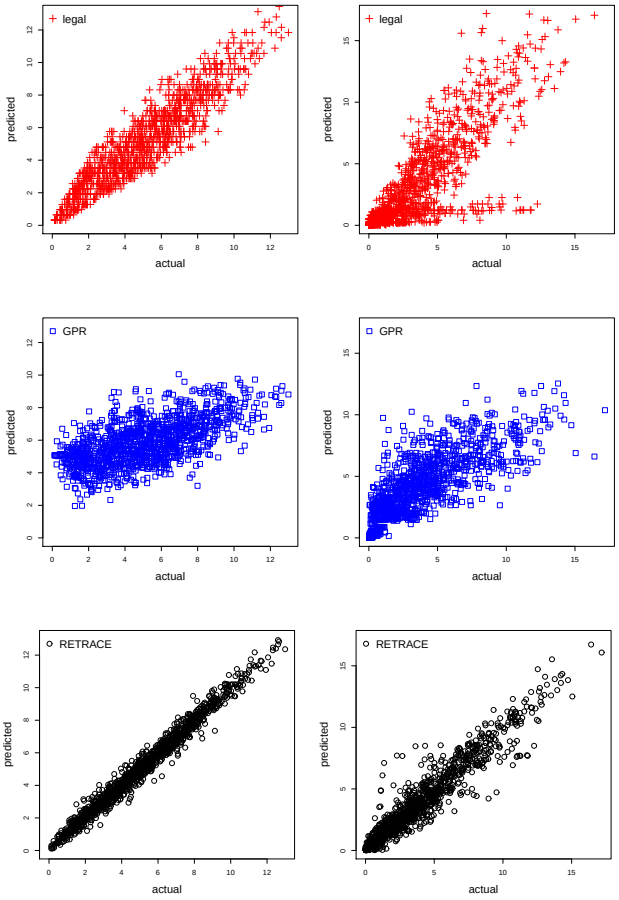

Figure 4: Comparison of travel-times in minute between `Legal`, `GPR`, and `RETRACE` (left column: `Grid25`, right column: `Kyoto`). The horizontal axis is the actual travel time, while the vertical axis is the predicted time.

a weight propagation mechanism is essential for trajectory cost prediction, since, if no such mechanism is included in the model, the weights for the zero-passage links will be left undetermined.

Figure 4 shows comparison between predicted and observed travel times. In the figures, the 45° line corresponds to perfect agreement between the actual and predicted values. For `Grid25`, the LOO CV scheme identified $\lambda^* = 600$ for `RETRACE`, while for `Kyoto` $\lambda^* = 4900$. To evaluate the accuracy, we employed 5-fold CV.

From the figure, we see that `RETRACE` clearly gives the best performance. It is almost surprising that the prediction accuracy of `GPR` is comparable to or even worse than the naive method. This result might seem to contradict the good performance reported in (Idé and Kato 2009). This is mainly due to the fact that the previous work assumes all of the trajectories *share the same* O-D (origin-destination) pair. In contrast, we are interested in more general situations allowing arbitrary O-D pairs. In such a case, it is clear that the string kernel cannot be a good way of evaluating the similarity between trajectories.

In our formulation, the model implicitly includes additional parameters of the baseline costs. In the experimental result of Grid25, the baseline cost seems to be a good zeroth approximation. However, in Kyoto, we observe an interesting *V*-shaped distribution in the figure of Legal. This is an indication of coexistence between congested and free-flow states. In spite of this, it is encouraging to see that RETRACE gave a good predictive performance. This fact indicates the robustness of our approach against the variation of the baseline cost.

In Kyoto, we see that there are several outliers in prediction by RETRACE around the actual cost of 1-2 min. Close inspections show that these are due to dynamic changes of the traffic state. In this data set, some of the links make a transition from free-flow to congested states in the course of simulation. Since we employ the CV scheme for prediction, some of the samples before the transition may be predicted by a model based on the congested state. Although handling nonstationarities of this kind is hard in general, the overall results demonstrate the utility of our approach even in the dynamic situation.

Finally, Table 2 summarizes the relative squared loss per link (factors for the RETRACE), which was averaged over 5 fold CV. As shown, RETRACE clearly outperforms the others, while the GPR gives systematically worse estimates.

Table 2: Relative squared loss per link (5-fold CV).

	RETRACE	Legal	GPR
Grid25	1	5.0	261
Kyoto	1	5.5	3.7

Related work

Our problem of trajectory cost prediction has relationships with (1) trajectory mining, and (2) travel-time prediction in transportation analysis.

Knowledge discovery from trajectories is one of the hottest recent topics in the data mining community. Recent studies include trajectory clustering (Pelekis et al. 2009) and trajectory classification (Lee et al. 2008). In the AI community, trajectory planning is a popular research theme. However, little attention has been paid to *trajectory regression*, which is the focus of the present paper.

In travel-time prediction, there is a long history of research for modeling travel-time (e.g. (Robinson and Polak 2005)). However, almost all of these traditional approaches aim at modeling the travel-time for *a particular link* rather than modeling an entire trajectory.

Conclusion

We have proposed a new framework for predicting the costs of the trajectories in a network. Our algorithm is quite practical in that it gives solutions by solving simple sparse linear equations. We then discussed the relationship to the existing GPR formulation, and showed that our formulation leads to a natural choice of kernel. We demonstrated a high predictive ability of our method based on a real road network. For future work, it would be interesting to generalize the present framework to include time-dependence of the traffic.

Acknowledgements

The authors thank Hiroki Yanagisawa for his help in simulation. TI was supported by the PREDICT program of the Ministry of Internal Affairs and Communications, Japan. MS was supported by the FIRST program.

References

Fouss, F.; Pirotte, A.; Renders, J.-M.; and Saerens, M. 2007. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Trans. on Knowledge and Data Eng.* 19(3):355–369.

Golub, G. H., and Loan, C. F. V. 1996. *Matrix computations (3rd ed.)*. Johns Hopkins University Press, Baltimore, MD.

Greaves, S., and Figliozzi, M. 2008. Collecting commercial vehicle tour data with passive global positioning system technology: Issues and potential applications. *Transportation Research Record* 2049:158–166.

IBM. 2008. Social simulation project, IBM Research – Tokyo, http://www.trl.ibm.com/projects/socsim/.

Idé, T., and Kato, S. 2009. Travel-time prediction using Gaussian process regression: A trajectory-based approach. In *Proc. SIAM Intl. Conf. Data Mining*, 1183–1194.

Kapoor, A.; Qi, Y. A.; Ahn, H.; and Picard, R. 2006. Hyperparameter and kernel learning for graph based semi-supervised classification. In *Advances in Neural Information Processing Systems 18*. MIT Press. 627–634.

Lee, J.-G.; Han, J.; Li, X.; and Gonzalez, H. 2008. Traclass: trajectory classification using hierarchical region-based and trajectory-based clustering. *Proc. VLDB Endow.* 1(1):1081–1094.

Nikolova, E., and Karger, D. R. 2008. Route planning under uncertainty: the canadian traveller problem. In *Proc. of the 23rd Conf. on Artificial intelligence*, 969–974. AAAI Press.

Pelekis, N.; Kopanakis, L.; Kotsifakos, E.; Frentzos, E.; and Theodoridis, Y. 2009. Clustering trajectories of moving objects in an uncertain world. In *Proc. IEEE Intl. Conf. on Data Mining (ICDM 09)*, 417–427.

Rasmussen, C. E., and Williams, C. 2006. *Gaussian Processes for Machine Learning*. MIT Press.

Robinson, S., and Polak, J. W. 2005. Modeling urban link travel time with inductive loop detector data by using the k-NN method. *Transportation research record* 1935:47–56.

Sekine, Y. 1994. *AI in Intelligent Vehicle Highway Systems: Papers from the 1993 Workshop*. AAAI Press.

Urmson, C.; Baker, C.; Dolan, J.; Rybski, P.; Salesky, B.; Whittaker, W.; Ferguson, D.; and Darms, M. 2009. Autonomous driving in traffic: Boss and the Urban Challenge. *AI Magazine* 30(2):17–28.