Modeling Freeway Traffic with Coupled HMMs

Jaimyoung Kwon
Department of Statistics
University of California
Berkeley, CA 94720
kwon@stat.berkeley.edu

Kevin Murphy
Department of Computer Science
University of California
Berkeley, CA 94720
murphyk@cs.berkeley.edu

Abstract

We consider the problem of modeling loop detector data collected from freeways. The data, which is a vector time-series, contains the speed of vehicles, averaged over a 30 second sampling window, at a number of sites along the freeway. We assume the measured speed at each location is generated from a hidden discrete variable, which represents the underlying state (e.g., congested or free-flowing) of the traffic at that point in space and time. We further assume that the hidden variables only depend on their spatial-temporal neighbors. Such a model is called a coupled hidden Markov model (CHMM). We can fit the parameters of this model using EM. However, since exact inference is intractable, we consider two different approximation schemes: one based on a sequential Monte Carlo technique (particle filtering), and the other based on the Boyen-Koller (BK) algorithm. We show that both algorithms perform well, compared to exact inference, and that the resulting learned model captures many important features of the data. Such a macroscopic model could prove useful for fault diagnosis, and in predicting future traffic patterns, particularly in response to causal interventions.

1 Introduction

Many urban freeways are equipped with induction loop detectors. These detectors can report three kinds of information in real-time [5]: the number of vehicles passing the location during a given time interval (flow rate), the fraction of time that vehicles are over the detector (occupancy), and the vehicle velocities averaged over the time interval. Here, we will restrict ourselves to aggregated velocity data. See Figures 1 and 2 for a sample of the data we are using, which was collected from I-880 in Oakland, California. For details about this dataset, see [8].

In Figure 1, we can clearly see onset, propagation and dissipation of traffic congestion; this appears as the well-known inverted triangular shape [9]. Essentially, traffic gets slower downstream, this effect propagates upstream, only to eventually disappear. There are various theories which try to explain this kind of behavior, based on models of fluid flow,

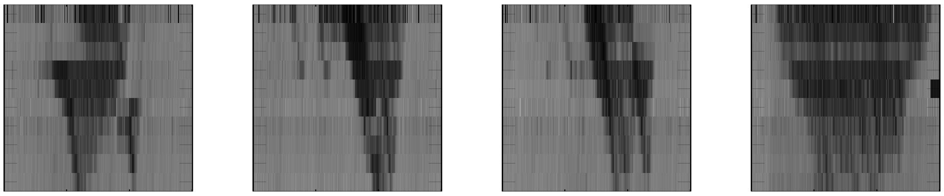


Figure 1: The full data set contains the velocity field for 20 weekdays, 2-7pm, between February 22 and March 19, 1993. The measurements come from 10 loop detectors (0.6 miles apart) in the middle lane of I-880, with 1 measurement per 30 seconds. Here we plot the first four days, temporally subsampled by 2. The x-axis corresponds to time, and the y-axis to space. Vehicles travel upward in this diagram. The darkest gray-scale corresponds to the average velocity of 20 miles per hour (mph) and the lightest to 70 mph. The isolated dark blob on the right edge of the fourth day is probably due to a sensor failure.

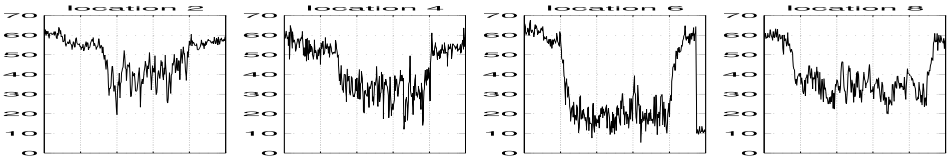


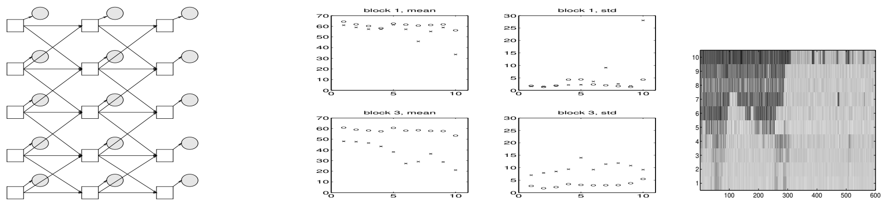
Figure 2: The velocity measurements for day 4 at locations 2, 4, 6 and 8. The x-axis is minutes (0 to 300), the y-axis is mph. The sudden drop from 64 to 10 mph at location 6 (which corresponds to the dark blob in Figure 1) is probably due to a sensor failure.

cellular automata, and microscopic computer simulations. But all of these approaches have limitations, particularly the need to tune many parameters specific to each individual freeway.

In this paper, we try to learn a model of traffic velocities from data. We assume that there is a hidden variable at each of the L detector locations, which takes on Q possible values, typically two (free flow and congestion), and that the observed velocity is a noisy representation of the current state. (We use a discrete hidden variable, since the time series of the average velocity vector is highly nonlinear.)

The most naive model is to assume each hidden state variable evolves independently, resulting in L independent HMMs. However, such a model cannot capture spatial correlation, and hence is incapable of capturing the global dynamics, such as the inverted triangle shape. The other extreme is to assume each variable at time t depends on all the other (hidden) variables at time $t-1$; this results in a single HMM with a state space of size Q^L, which requires $Q^L(Q^L-1)$ parameters just to specify its transition matrix. We therefore adopt a middle ground, and assume each variable only depends on its local neighbors in space and time; such a model has been called a coupled HMM [11], and is shown as a dynamic Bayesian network (DBN) in Figure 3(a). In Section 2, we describe the model in more detail.

We will estimate the parameters using EM. The M step is straightforward, but the E step is in general computationally intractable. We therefore need to use approximate inference. In Section 3, we describe and compare the two approximate inference algorithms that we use, particle filtering [4, 3] and the Boyen-Koller (BK) algorithm [2, 1]. In Section 4, we give the results of learning using exact and approximate inference in the E step. In Section 5, we discuss the adequacy of the model in light of our results.



(a) (b) (c)

Figure 3: (a) A coupled hidden Markov model represented as a dynamic Bayesian network. Square nodes represent discrete random variables (rv's) with multinomial distributions, round nodes represent continuous rv's with Gaussian distributions. Clear nodes are hidden, shaded nodes are observed. Here we show $L = 5$ chains and $T = 3$ timeslices. (b) Maximum likelihood estimates of the mean and standard deviations of the models for block 1 (2-3 pm) and block 3 (4-5 pm). Crosses refer to the congested state, circles to free-flow. (c) Data sampled from the learned model for 4-5pm.

2 The model

Assume there are L loop detector stations indexed by $l = 1, ..., L$, from upstream to downstream. The observed (aggregated) velocity $y_{l,t}$ (mph) at location l and time t has a distribution that depends only on the underlying state variable $x_{l,t} \in \mathcal{S} = \{s_1, \cdots, s_Q\}$. The simplest model is a binary chain with $Q = 2$, where the two states s_C and s_F correspond to 'congestion' and 'free flow'. We initialise the mean for the congested state to be less than the mean for the free-flow state, although we do not enforce this constraint during learning.

We assume that the hidden process of $x_t = (x_{1,t}, \cdots, x_{L,t}) \in \mathcal{S}^{\otimes L}$ is Markovian and its transition probability can be decomposed as $P(x_{t+1}|x_t) = \prod_{l=1}^{L} P(x_{l,t+1}|x_t) = \prod_{l=1}^{L} P(x_{l,t+1}|x_{l-1,t}, x_{l,t}, x_{l+1,t})$, i.e., the traffic state at a location is affected only by the previous state of the neighboring locations. The initial distribution on the state is assumed to decompose as $P(X_1) = \prod_{l=1}^{L} P(X_{l,1})$. Finally, the observed velocity is a Gaussian whose mean and variance depends on the underlying state at the location: $P(y_{l,t}|x_{l,t} = s_k) \sim N(\mu_{l,k}, \sigma^2_{l,k})$. We need $LQ^3(Q-1) + 2Q^2(Q-1)$ parameters to specify the transition model, $2LQ$ to specify the observation model. and LQ to specify the initial distributions. If $L = 10$ and $Q = 2$, this is a total of $88 + 40 + 20 = 148$ parameters.

Because of the non-stationary nature of the data, we split the 5 hour period of each day into five 1-hour blocks (60 time slices). Then we fit a binary state ($Q = 2$) CHMM for each block separately, considering 20 days as independent replicates of the process. We use days 1,3,...,19 as the training set and 2,4,...,20 as the test set. Hence we have 10×60 observations to fit 148 parameters per model. To minimize the chance of overfitting, we could use cross validation or parameter tieing, although we do not do that here.

3 Inference

The simplest approach to exact inference in DBNs is to convert the model to an HMM and use the forwards-backwards algorithm; this takes $O(TQ^{2L})$ time and space. A more sophisticated approach, which exploits the conditional independence structure by using the junction tree algorithm [7], takes $O(T(L-1)Q^{L+F-1})$ time and space, where $F = 3$ is the maximal fan-in of any node. For $L = 10$, $Q = 2$ and $T = 60$, this is about 2 million operations, which is certainly tractable. However, in the future we hope to scale up to modeling complex freeway networks with $L \sim 100$ detectors and $Q \sim 5$ states, so we will need to use approximate inference. We discuss two different approximate inference algorithms below.

3.1 Particle filtering

Particle filtering (PF) [4, 3] is a well-known sequential Monte Carlo technique for approximate filtering, where the posterior is approximated with a set of weighted samples, called "particles". We use $N_s = 100$ particles sampled from the optimal proposal distribution, which takes into account both the previous state and the current evidence. We have found that increasing the number of particles to 1000 makes little difference, presumably because the observations are very informative, and the transitions very deterministic (see Section 4).

Note that our samples are complete paths from $t = 1, \ldots, T$, so we can easily estimate the smoothed joint posterior $P(X_{\pi,t-1}, X_{l,t}|y_{1:T})$, where $\pi = (l-1, l, l+1)$ are the parents of l; we need this quantity to compute the expected sufficient statistics for EM. An alternative would be to run two particle filters, forwards and backwards, to estimate $P(X_t|y_{1:t})$ and $P(X_t|y_{t+1:T})$, and then to combine them [6]. Unfortunately, this takes $O(TN_s^2)$ time.

3.2 Boyen-Koller algorithm

The Boyen-Koller algorithm [2] represents the belief state, $\alpha_t = P(X_t|y_{1:t})$, as a product of marginals over C "clusters", $P(X_t|y_{1:t}) \approx \prod_{c=1}^{C} P(X_{c,t}|y_{1:t})$, where $X_{c,t}$ is a subset of the variables $X_{l,t}, l = 1, \ldots, L$. (The clusters do not need to be disjoint.) Given a factored prior, $\tilde{\alpha}_{t-1}$, we do one step of exact Bayesian updating to compute the posterior, $\hat{\alpha}_t$. In general, this will not be factored as above, so we need to project to the space of factored distributions by computing the marginal on each cluster. The product of marginals then gives the approximate posterior, $\tilde{\alpha}_t$. We can use a similar method for computing the backward messages, $\beta_t = P(y_{t+1:T}|X_t)$, in an efficient manner; these can then be combined to produce an approximate smoothed posterior, $\gamma_t = P(X_t|y_{1:T})$ [1].

The accuracy of the algorithm depends on the size of the clusters that we use to approximate the belief state. Exact inference corresponds to using a single cluster, containing all the hidden variables in a time-slice. The most aggressive approximation corresponds to using L clusters, one per variable; we call this the "fully factorized" approximation, and is the version we use in this paper.

The exact update step uses the junction tree algorithm, and relies on the assumption that updating a factored prior is efficient; in other words, it assumes that the sizes of the cliques in the triangulated graph of the two-slice DBN are small. For the CHMM model, this is indeed the case, since the cliques just correspond to the families (nodes and their parents). Hence the algorithm takes $O(TLQ^{F+1})$ time, where $F = 3$ is the maximal fan-in of any node. (When even one step of exact updating is intractable, the algorithm in [10] can be used.)

3.3 Comparing the inference algorithms

To compare the accuracy of PF and BK relative to exact inference, we computed the smoothed posterior marginal estimates $P(X_{l,t}|y_{1:T})$ using each of the methods on each of the test sequences, and using the estimated parameters. BK yields posterior estimates that are indistinguishable from exact inference to at least three decimal places. PF yields a noisier estimate, but it is still very accurate: define $\Delta_{l,t}$ to be the L_1 difference of the estimates computed using exact and PF; then the empirical mean of this quantity is 0.0139 ± 0.0093 for 100 particles, and 0.0135 ± 0.0028 for 1000 particles. We see that using more particles slightly increases the accuracy and reduces the variance, but it seems that 100 particles is sufficient. The reason for this is the near-deterministic nature of the transitions (see below) and the informativeness of the observations.

Since the inference algorithms perform similarly, we expect the estimated parameters to be similar, too. This is indeed the case for the μ and σ parameters of the observation model, where the differences are not statistically significant (even using only 100 particles). PF does a poorer job at estimating the transition parameters, however, due to the fact that it only counts 100 sample paths per sequence. The total normalized L1 error is 4.9 for BK and 8.0 for PF. Using more particles would obviously help. See Section 4 for a more detailed discussion of the estimated parameters.

In addition to accuracy, speed is a major concern. A single E step takes about 1 second/slice using exact inference, about 1.3 s/slice using BK, and about 0.1 s/slice using PF with 100 particles.[1] The reason that BK is slower than exact (in this case) is because of the high constant factors, due to the complexity of the algorithm, and especially the need to perform the projection (marginalisation) step. Of course, the asymptotic complexity of BK is linear in L, while exact inference is exponential in L, so it is clear that for larger models, BK will rapidly become faster than exact.

4 Learning results

To aid interpretability of the parameters, we initialised the means for state 0 (congestion) to be 40 mph, and for state 1 (free flow) to be 60 mph. All other parameters were initialised randomly. We ran EM until the change in log-likelihood was less than 10^{-4}, which usually took 10–20 iterations. Some of the learned μ and σ values (using exact inference) are shown in Figure 3(b). The parameters for the models for 3-4pm, 4-5pm and 5-6pm are all very similar; we call this the "rush-hour" model. The parameters for the models for 2-3pm and 6-7pm are also very similar; we call this the "offpeak" model.

It is clear that when the traffic is slow, the variance is high, but when the traffic is fast, the variance is low. It is also clear that congestion gets worse as one approaches location 10, which corresponds to the part of I-880 that is near the San Mateo Bridge, a notorious bottleneck. Thus the learned μ and σ values seem sensible. The estimated parameter values are very insensitive to the initialisation and the inference algorithm used in the E step.

Interpretation of the transition parameters is harder. We might hope to estimate the speed with which congestion propagates and clears, as illustrated below.

$$\begin{array}{ccc|ccc} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 \end{array} \tag{1}$$

Here we show a hypothetical example of the hidden state of the traffic at three neighboring locations (traffic flows upwards) evolving over time. 0 means congestion, 1 means free flow. On the left, we see congestion backing up, and on the right, we see it clearing. Thus, let $P_c = P(X_{l,t} = 0 | X_{l-1,t-1} = 1, X_{l,t-1} = 1, X_{l+1,t-1} = 0)$ be the probability with

[1]The first author implemented PF in S-Plus on a 400 MHz PC. The second author implemented exact inference and BK in Matlab on a Sun Ultra. The latter code is part of the Bayes Net Toolbox, and can be downloaded from `www.cs.berkeley.edu/ ~murphyk/Bayes/bnt.html`. These times were measured using sequences of length $T = 60$.

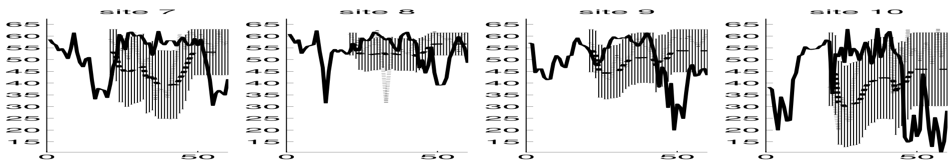


Figure 4: 20 minute-ahead predictions. Solid is the truth, dotted is the naive prediction, dashed (in the middle of the error bars) uses the model.

which site l becomes congested (enters state 0) given that it was previously uncongested (state 1), but that the downstream site $l+1$ was previously congested. Similar, let $P_f = P(X_{l,t} = 1|X_{l-1,t-1} = 1, X_{l,t-1} = 0, X_{l+1,t-1} = 0)$ denote the probability with which site l becomes uncongested. Unfortunately, the estimated values of these quantities do not seem to reveal any meaningful pattern.

One of the advantages of a generative model is that we can simulate future traffic patterns. A typical sample drawn from this model is shown in Figure 3. We see that this resembles the training data; in particular, the model is capable of generating the triangular shape.

Finally, we can use the model to do online prediction. For exact inference, we just compute $\alpha_{t+\Delta|t} = P(X_{t+\Delta}|y_{1:t}) = M^{\Delta}\alpha_{t|t}$, where M is the transition matrix, and then compute $P(Y_{t+\Delta}|y_{1:t})$, which is a mixture of Gaussians with $\alpha_{t+\Delta|t}$ as the mixing weights. See Figure 4 for an example. We compared these predictions to the naive approach of predicting that the future is the same as the present, $\hat{y}_{t+\Delta} = y_t$, for leads up to 20 minutes ahead. For the sequence in Figure 4, the rms error is 10.83 for the naive method and 9.76 for the model-based method. (We are ignoring the predicted σ, i.e., the confidence in the prediction, which is only available for the model-based approach.) Other sequences give similar results. It is clear from these numbers, and from the figure, that our predictions are not very accurate. We discuss ways to improve the model in the next section.

5 Discussion

Perhaps the most serious problem with our approach is that we have learned a separate model for each 1 hour period between 2-7pm, making it tricky to predict across boundaries. One approach would be to use the posterior from the previous model as the prior for the next. Alternatively, we could fit a single (mixture) model, by adding an extra hidden variable to each time slice to represent the current regime; all the other variables could then be conditioned on this. (In this case, the fully factorized version of BK takes $O(TLQ^5)$, since the maximum clique size is 5.) Such a switching model could capture periodic non-stationarities. The number of regimes could be chosen using cross validation, although our results suggest that two might be sufficient, corresponding to rush-hour and off-peak.

We could also choose the number of hidden states for each location based on cross-validation. However, it is clear that $Q = 2$ is inadequate, since it is incapable of distinguishing whether congestion is increasing or decreasing. To see this, consider the $(0, 0, 1)^T$ column in Equation 1. This has two possible successor configurations, depending on the previous configuration, and thus the model is not Markov, i.e., the future is not independent of the past given that the present state is $(0, 0, 1)$. It would be straightforward to use $Q > 2$, or to make the model second-order Markov by adding longer distance dependencies, or to add extra variables to represent the (sign of the) derivative of the speed.

A third weakness is our assumption of Gaussian noise on the observations. Sensor failures, such as those shown in Figure 1, clearly invalidate this assumption. We can use a mixture of Gaussians as the noise model to handle this.

In the future, we plan to try using $Q > 2$ with the switching model. We will also include a deterministic node that encodes the current time; thus predictions will be based both on historical patterns (using the time node) and the current state of the system (using the other hidden nodes). Ultimately, our goal is to predict travel time, rather than just velocity. We plan to build a model that can take as input the belief state about the current conditions, and combine it with historical (supervised) data, to provide a real-time travel forecast engine.

Acknowledgments

We would like to thank Peter Bickel, Michael Jordan, John Rice and Stuart Russell for their helpful comments.

References

[1] X. Boyen and D. Koller. Approximate learning of dynamic models. In *NIPS-11*, 1998.

[2] X. Boyen and D. Koller. Tractable inference for complex stochastic processes. In *UAI*, 1998.

[3] A. Doucet, N. de Freitas, and N. J. Gordon. *Sequential Monte Carlo Methods in Practice*. Springer Verlag, 2000. Forthcoming.

[4] N. Gordon. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proceedings (F)*, 140(2):107–113, 1993.

[5] F. L. Hall. Traffic stream characteristics. In N. Gartner, C. Messer, and A. Rathi, editors, *Traffic Flow Theory*. US Federal Highway Administration, 1996.

[6] M. Isard and A. Blake. A smoothing filter for condensation. In *Proc. European Conf. on Computer Vision*, volume 1, pages 767–781, 1998.

[7] U. Kjaerulff. A computational scheme for reasoning in dynamic probabilistic networks. In *UAI-8*, 1992.

[8] J. Kwon, B. Coifman, and P. Bickel. Day-to-day travel time trends and travel time prediction from loop detector data. *Transportation Research Record*, (1554), 2000. To appear.

[9] A. D. May. *Traffic Flow Fundamentals*. Prentice Hall, 1990.

[10] K. Murphy and Y. Weiss. The factored frontier algorithm for approximate inference in DBNs. Submitted to NIPS-12.

[11] L. Saul and M. Jordan. Boltzmann chains and hidden Markov models. In *NIPS-7*, 1995.