World Academy of Science, Engineering and Technology
International Journal of Mathematical and Computational Sciences
Vol:1, No:1, 2007

Traffic Flow Prediction using Adaboost Algorithm with Random Forests as a Weak Learner

Guy Leshem, and Ya'acov Ritov

***Abstract*—**Traffic Management and Information Systems, which rely on a system of sensors, aim to describe in real-time traffic in urban areas using a set of parameters and estimating them. Though the state of the art focuses on data analysis, little is done in the sense of prediction. In this paper, we describe a machine learning system for traffic flow management and control for a prediction of traffic flow problem. This new algorithm is obtained by combining *Random Forests* algorithm into *Adaboost* algorithm as a weak learner. We show that our algorithm performs relatively well on real data, and enables, according to the Traffic Flow Evaluation model, to estimate and predict whether there is congestion or not at a given time on road intersections.

***Keywords*—**Machine Learning, Boosting, Classification, Traffic Congestion, Data Collecting, Magnetic Loop Detectors, Signalized Intersections, Traffic Signal Timing Optimization.

I. Introduction

TRANSPORTATION systems are an integral part of a modern day society, designed to provide efficient and economical movement between the component parts of a country and offer maximum possible mobility to all citizens. Road transportation is a critical link between all the other modes of transportation and their proper functioning. Signalized intersections, as a critical element of an urban road transportation system, regulate the flow of vehicles through urban areas. As a result, traffic flowing through signalized intersections is filtered by the signal system causing vehicular delays, which increases the total travel time through an urban road network, thus resulting in a reduction in the speed, reliability, and cost-effectiveness of the transportation system. Furthermore, longer delays yield to degradation of the environment by increasing air and sound pollution. For all these reasons, it is important to predict and minimize these delays. The model commonly used to evaluate traffic congestion called the "*Traffic Flow Evaluation*" model, see [13]. To obtain training data, most systems rely on a network of sensors to estimate traffic parameters in real-time. Currently, the dominant technology for this purpose is the use of magnetic loop detectors, which are embedded underground at almost every urban intersection and measure traffic parameters of vehicles passing above them. In this model, the criterion of interest for prediction is the *Level Of Service* (LOS) for each intersection, namely at which cycle of the signal light a vehicle will cross the intersection in mean. This quantity depends on the signal cycles (red and green lights) and also on the occupancy value, which is the average time a given car stands over the detector per signal cycle. Note that the occupancy value and the signal setup depend on many other important parameters such as, for example, the hour and day of measurements; we refer to [13] for an exhaustive description of all parameters. According to the Division of Transportation and Highways Engineering determination, traffic flow will be considered as congestion if No. of signal cycles till the vehicle will cross the junction > 4. Several software, such as "PATH" (UC Berkeley) and "NISS Digital Government II" (DGII) have been developed to deal with the computation of the LOS. Nevertheless, they focus more on data analysis, whereas little is done in the sense of prediction. Namely, these methodologies take the time information into little account, thus losing valuable information for traffic control. The objective of this study is to provide an advanced methodology of identification and prediction of urban traffic obstructions. In this contribution we present a methodology in order to predict and minimize incoming traffic flow congestion. The paper is organized as follows: Section 2 presents an overview of *Adaboost* (see [16]) and *Random Forests* introduced in [1]. Section 3 details our model and the proposed method of prediction; this approach uses Random Forests as a first predictor (also called *weak learner* in further sections) and combines it within a committee framework *AdaBoost*, in order to boost its accuracy. The proposed algorithm can be used as well for the optimization of the signal settings over time. An application on real data is presented in Section 4, which shows the adequacy of the proposed method, and shows experiment and results. Section 5 details our conclusions.

Guy Leshem, PhD student is with Department of Statistics, The Hebrew University of Jerusalem, Israel (e-mail: gleshem@cc.huji.ac.il).

Ya'acov Ritov, Professor is with Department of Statistics, The Hebrew University of Jerusalem, Israel (e-mail: yaacov@mscc.huji.ac.il).

II. Algorithmic Part

A. AdaBoost

The main idea of boosting algorithms is combining many simple and moderately accurate classifiers (called weak classifiers) into a single, highly accurate classifier for the task at hand. The weak classifiers are trained sequentially and, conceptually, each of them is trained mostly on the examples,

which were most difficult to classify, by the preceding weak classifiers. The boosting algorithm takes as input a training set of m examples $S=\{(x_1,y_1),\ldots,(x_m,y_m)\}$ where each instance x_i is a vector of attribute values that belongs to a domain or instance space X, and each label y_i is the class label associated with x_i that belongs to a finite label space Y. In this contribution we will only focus on binary classification problems, that is $Y=\{-1,+1\}$. The final classifier, h, is constructed by a weighted vote of the individual classifiers $h_1,h_2,\ldots,h_m$. Each classifier is weighted according to its accuracy for the distribution p_l that it was trained on. We present as a reminder the pseudo-code of the "classic" *Adaboost*, and refer to [9] for further results on this algorithm.

- Input: a set S, of m labeled examples: $S=((x_i,y_i),i=(1,2,\ldots,m))$, with labels in Y.
- Learn (a learning algorithm)
- A constant L.

[1] Initialize for all i: $w_l(i)=\frac{1}{m}$ initialize the weights

[2] for l=1 to L do

[3] for all i: $p_l(i)=\frac{w_l(i)}{\sum_i w_l(i)}$ compute normalized weights

[4] h_l:=Learn(S,p_l) call Learn with normalized weights

[5] $\varepsilon_l=\sum_i p_l(i)[h_l(x_i\neq y_i)]$ call Learn with normalized weights

[6] *if* $\varepsilon_l > 1/2$ *then* calculate the error of h_l

[7] L=l-1

[8] go to 12

[9] $\beta_l=\frac{\varepsilon_l}{1-\varepsilon_l}$

[10] for all i: $w_{l+1}(i)=w_l(i)\beta_l^{1-[h_l(x_i\neq y_i]}$ compute new weights

[11] end for

[12] Output: $h_{final}(x)=\arg\max_{y\in Y}\sum_{l=1}^{L}(\log\frac{1}{\beta_l})[h_l(x)=y]$

As we can see in this algorithm, it is necessary to define at first a learning algorithm. For instance, we can choose Random Forests algorithms, which are detailed in the following section.

B. Random Forests

Breiman [1] has developed an ensemble classification approach that displayed outstanding performance with regard prediction error on a suite of benchmark datasets. This development, known as "Random Forests", is a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest. The generalization error for forests converges a.s. to a limit as the number of trees in the forest becomes large. The error of a forest of tree classifiers depends on the strength of the individual trees in the forest and the correlation between them. Internal estimates monitor error, strength, and correlation and these are used to show the response to increasing the number of features used in the splitting. The common element in all of these procedures is that for the k-th tree, a random vector θ_k is generated, independent of the past random vectors $\theta_1\ldots\theta_{k-1}$, but with the same distribution, and a tree is grown using the training set and θ_k, resulting in a classifier $h(x,\theta_k)$ where x is an input random vector. For instance, in bagging, the random vector θ is a random sample of size N chosen with replacement from the N examples of the training set. After a large number of trees are generated, they "vote" for the most popular class. The pseudo-code of Random Forests is as follows:

[1] Initially select the number K of trees to be generated.

[2] For k=1 to K do

[3] A Vector θ_k is generated

[4] Construct Tree $h=(\underline{x},\underline{\theta_k})$ using any decision tree algorithm.

[5] Each Tree casts 1 vote for the most popular class at X.

[6] The class at X is predicted by selecting the class with max votes.

[7] Return a hypothesis h_l.

[8] End For

One practical interest in Random Forests is that it performs nicely even in the case of missing data in X, since some rules of replacement for the missing data can be plugged-in, like the median of all obtained values as stated in Breiman [1]. This often occurs in our situation; say if some loop detectors are broken. Another advantage of the Random Forest paradigm is that there is no need for cross-validation or a separate test set to get an unbiased estimate of the test set error. Indeed, this error is estimated internally, during the run, as follows: each tree is constructed using a different bootstrap sample from the original data. About one-third of the cases are left out of the bootstrap sample and not used in the construction of the k-th tree. We put each case left out in the construction of the k-th tree down the k-th tree to get a classification. That way, a test set classification is obtained for each case in about one-third of the trees. At the end of the run, take j to be the class that got most of the votes every time case n was oob (out-of-bag). The proportion of times that j is not equal to the true class of n averaged over all cases is the oob error estimate. This method has proven to be unbiased in many tests [1].

III. Methodology: The AdaBoost-Random Forest Procedure

When combining Adaboost and random forest methods, we can distinguish two possible ways. The first one is "boost in forest", where an AdaBoost classifier is built for each random vector θ_k (i.e., a collection of variables), and to get by that sequences of "simple" AdaBoost classifiers, each with small number of variables. Instead, we chose a second approach, which uses Random Forests as a weak learner. The philosophical idea of the weak learner algorithms is to find quickly weak assumptions with moderate error rate, since it is

clear from a numerical point of view that AdaBoost would work faster with simple weak learners than with learners that build forests of trees, which is of interest in our real-time application.

A. *Advantages of the Combination*

- The final classifier of this approach (AdaBoost algorithm with random forest as weak learning) combines the good performances of Random Forests and Adaboost, we can therefore expect it to be very accurate with minor misclassification and strong predication ability.
- It has an effective method for dealing with missing data and maintains accuracy when a large proportion of the data are missing.
- It is an effective method for predication of multi-class classification problems.
- Note also that using Random Forests as a weak learner in AdaBoost enables to choose, in Step 6 of the Random Forest algorithm, a way of prediction depending on the weights, which could give better results.
- Because the new algorithm using random forests as weak learner and boosting unbiased classification, there is no need for cross-validation for "Adaboost-RandomForests" algorithm.

We present results on a known dataset (satimage), in order to validate the proposed method experimentally. Fig. 1 illustrates the differences between a standard weak learner (namely, a threshold) and a Random Forest used a weak learner. We can see that the choice of Random Forest leads to better results on the test error. Note also that Drucker obtained in [3], using C4.5 decision trees as weak learners a test error of 0.1 on the same dataset, for a number of cycles of the same magnitude. The proposed method thus performs relatively well, compared to a more refined approach as decision trees.

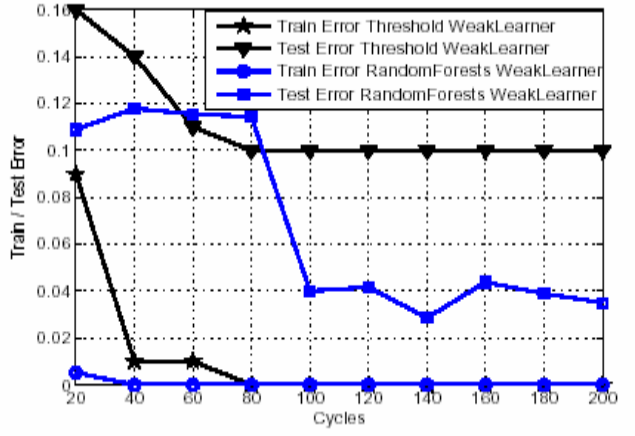

Fig. 1 Error curve for AdaBoost algorithm with classical weak-learner (threshold), and with Random Forest as weak learners

Fig. 2 illustrates the performances of the proposed algorithm in the case of missing data. In this experiment we randomly withdrew about 20% of the train and test data. Following Breiman [1], a possible way to replace missing data is to take the median of all explicit variables of X. Here we used a nearest neighbors approach, that is replace the missing data by the mean of the nearest components of X around this missing value. This makes sense physically, as each line of X also characterizes in our application a line of the road, thus it is reasonable to assume that the traffic state in a highway is the same in every line. We can see that missing data do not affect much the efficiency of the method.

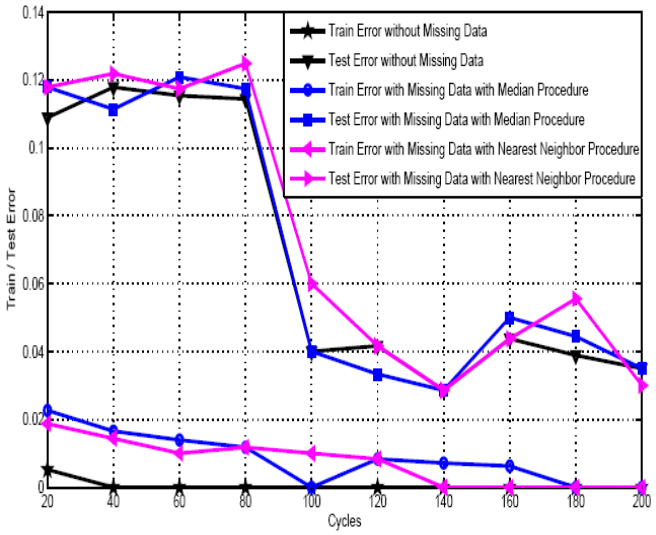

Fig. 2 Error curve for AdaBoost algorithm with Random Forest as weak learners, with (**median** of all explicit variables, and the mean of the **nearest** components of X around this missing value) / without missing data

B. *Use of AdaBoost-Random Forest for Prediction*

We now discuss how the proposed algorithm can be used in order to predict traffic congestion. In order to do that, it is necessary to explain more explicitly how our data depend on time. Clearly traffic information can vary according to both days and hours. In this section, we will denote by $X_{m\times n,d,t}$ a dataset of size **m** with **n** features, collected on day **d** and at hour **t** and by $Y_{m\times 1,d,t}$ the class labels associated to $X_{m\times n,d,t}$. The problem of prediction is therefore, given a dataset $X_{m\times n,d,t}$ known for every t, to estimate $Y_{\tilde{m}\times 1,d+\Delta d,t+\Delta t}$, where Δd and Δt respectively define a variation of day and a variation of hour. The proposed estimation procedure can then be decomposed in four steps:

i. Collect two samples of data, $X_{m\times n,d,t}$ and $X_{m\times n,d,t+\Delta t}$, $\Delta t > 0$,

ii. Compute the class labels $Y_{m\times 1,d,t+\Delta t}$,

iii. Compute a new training set $Z_{m\times n+1,d}$, using $X_{m\times n,d,t}$ and $Y_{m\times 1,d,t+\Delta t}$ by concatenation, that is

$Z_{m\times n+1,d} = [X_{m\times v,d,t}; Y_{m\times 1,d,t+\Delta t}]$,

iv. Collect the test set $X_{\tilde{m}\times n,d+\Delta d,t}$ ($\Delta d > 0$), and compute $Y_{\tilde{m}\times 1,d+\Delta d,t+\Delta t}$ using the AdaBoost-Random Forest algorithm on $Z_{m\times n+1,d}$ and $X_{\tilde{m}\times n,d+\Delta d,t}$.

IV. Application to Traffic Analysis and Prediction

As mentioned, control of signalized intersections is a critical element to optimize urban road transportation systems. We present in this section an application of our prediction method to the minimization of time delays at intersection. Indeed, if predicted early enough, say half an hour before the congestion actually happens; traffic control systems can change the signal timing plans according to optimization process of signalized intersections, thus allowing circumventing traffic congestion before it actually happens. We briefly describe in the next section data collection for traffic control, and present results on the proposed method. The reader interested in a complete description of how to build a dataset in that framework should refer to [13].

A. Data Set Collection: A Quick Overview

Traffic management and information systems (TMIS) currently rely on a system of magnetic loop detectors, which are buried underneath at almost every urban intersections and which measure traffic parameters of vehicles passing over them. Table I is an example of the data set (training and testing set) collected by Jerusalem Traffic Flow Management Control and used by our system.

TABLE I
EXAMPLE TO TRAIN FILE / TEST FILE

DAY	TIME	INT	DET	LINK	POS	GRE	DIS	VOL	OCC	Y
1	7:00	20	20-24	24	2	25	40	24	21	1
1	7:00	30	30-33	33	1	20	35	76	33	-1
1	7:00	40	40-42	42	3	30	40	18	12	1
.	.	.	.	.	.	.	.	.	.	.

We briefly describe the explanatory variables in a train or test file: **Day** is the day number (from 1=Sunday to 7=Saturday), **Time** is the time when a vehicle passes over a given detector, **Gre** is the green light duration, **Vol** is the number is vehicles passing a detector per light cycle, **Occ** is the average that a car spends on a given detector and **Y** is our label class, with value 1 if congestion happens and -1 otherwise. The variables **Det** (Detector ID), **Dis** (distance between the detector and the stop line), **Int** (Intersection ID), **Pos** (detector position from right (1=right, 2= two from right...)) and **Link** (lane ID) characterizes a given intersection, and their description will be omitted here.

B. Traffic Flow Evaluation Model (Determination of Hypothesis Class for Train File)

Data collected from detectors regarding traffic flow in signalized intersections are evaluated according to the following model, which was developed in [13]. Shortly, identification of traffic congestion is based on the occupancy value (which may be deduced from the measurements) and the saturation level value, which depends on the programming of traffic light and capacity and is calculated according to the green light duration allowed to each vehicle. The main disadvantage of using the saturation level as a measure of performance is that it is impossible to measure demand on the detectors, instead we have access only to the volume of traffic that succeeded in passing over them. The capacity constitutes an upper bound approximately to this value. Upon entrance to signalized intersections, where detectors are in use and during the red light period, the routine queue arrives to the detector, the absolute value of occupancy that identifies traffic congestion is nonexistent. The interconnection between the occupancy measured on the detectors and the quality of the traffic flow depend on the green light duration allowed each vehicle upon entrance to the junction. Therefore, the determination that occupancy value confirms traffic congestion is not an absolute determination and is only relative to the expected value, which reflects the duration of a green light for each vehicle upon entrance to the junction. For identification of traffic congestion and determination of "critical occupancy value" defined as expected occupancy in the situation demand (per lane) is identical to capacity. This given values can be calculated. The decision making process for the determination of traffic flow quality will be executed according to the following stages:

1. A calculation of critical occupancy value and saturation level.
2. An estimation of traffic flow quality in the lane based on the comparison between actual occupancy and critical occupancy, and saturation level.
3. An estimation of traffic flow quality in the segment (link) based on traffic flow situation in lane.

Assuming that at the end of the green light period there is no queue on the stop line, it is possible to describe traffic phenomenon in the access to the junction in as follows: at the beginning of the red light period, the vehicles passing over the detector with free speed, slow down near the stop line, and stop at the edge of the queue which begins to form near stop line. In this time period, the volume of vehicles measured by the detector is identical to the demand, and critical occupancy is identical to the capacity. Continuation of this red light period, when the edge of the queue begins to approach the detector (assuming that the distance between the detector and the stop line "35m"), the speed of the vehicles begin to reflect approach to the queue, which means that the speed is slower than the speed of free journey. Therefore progressively the vehicles increase traffic volume. At certain values of distance between the detectors and the stop line, and during the red light period, a queue may approach the detector, and for this

short while Volume =0 and Occupancy =100%. This phenomenon continues until the end of the red light period and on the beginning of the green light period. At this time, two phenomena occur in parallel. In the stop line, the process of queue release begins and vehicles continue to arrive and join the edge of the queue. Two these phenomena can be describe as shock wave. When two shock waves meet, the last car in the queue begins to move. The speed characteristics of the vehicles passing over the detector during the release of the queue are similar to the speed characteristics of queue accumulation, but in inverse direction. The speed of a vehicle increases until it reaches free speed. A simplified calculation of critical occupancy was carried out and three types of situations were defined:

1. Journey at free speed on the detector in the worth volume to the capacity.
2. Journey at queue release speed on the detector and in the volume of saturation flow.
3. Stopping on the detector (Volume =0).

Fig. 3 is a graphical summary pf all possible situations taken into account in our procedure.

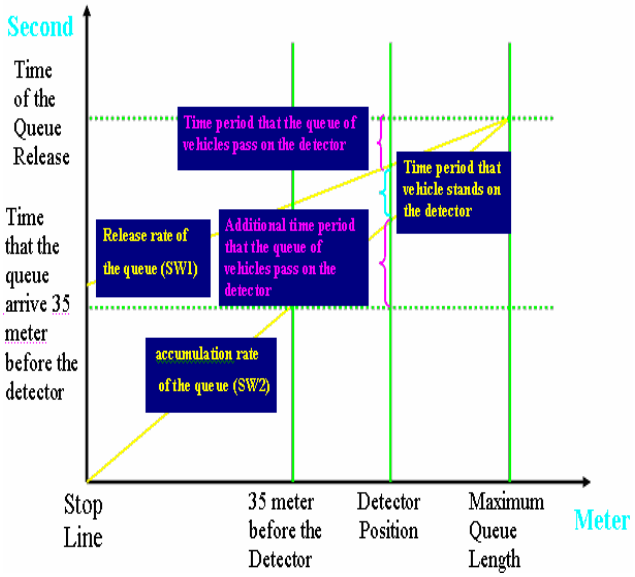

Fig. 3 The graph describes process of accumulation of the queue and his release (as in Mahalel and Gal-Tzur [13])

In this theory the vehicles joining the queue move rapidly and steady until joining the queue, and also depart rapidly and steady immediately upon release from the queue. We can see a point in the time and in the expanse at which the queue arrives at the detector, and a point at which the queue begins to exit, the point of the shocks wave of queue release reaches the detector. The time period between these two points is the range that the vehicles stand on the detectors. This model allows to calculation of Level of Service (LOS) of the intersection, or simply "at which cycle of the signal light the vehicle will cross the intersection". According to the Division of Transportation and Highways Engineering determination, traffic flow will be considered as congestion if the vehicle will cross the intersection at the fourth or more cycle of the signal. Consequently, given traffic parameters previously described, it is possible to compute the Level of Service (LOS), roughly 1 if LOS considers it as under-congestion and -1 otherwise. Those data will be used as labels.

C. Experimental Setup and Results

The idea of the prediction procedure, which is explained in the previous section, can be decomposed in these two steps:

$$X_{m\times n,d,t} \rightarrow Y_{m^*\times 1,d,t+\Delta t}\ (train\ file)$$

$$X_{\tilde{m}\times n,d^*,t} \rightarrow Y_{\tilde{m}^*\times 1,d^*,t+\Delta t}\ (Test\ file)$$

Jerusalem Traffic Management and Information Systems collected the data used in our experiment. The algorithm is trained on this train file (**X** is the samples of traffic data, with matrices size **m** x **n** which were produced from specific date **d** and specific time **t** with class labels **m*** which were produced from same date **d** but from different time **t+Δt**), and the test samples ($\tilde{m}$ which produce from latterly date **d*** and specific time **t**) were classified by the trained classifier. Because the algorithm classifier labels of future times, it is actually predication of traffic flow. Back to our experiment, training sets contains samples of data for one week (d=1.10-7.10) computed at (t=07:00) with class labels obtained using the "evaluation of the traffic flow in the signal intersection model" at time (t+Δt=07:30). Test sets contain samples of data collected the following week (d+Δd=8.10-15.10) at (t=07:00) in order to predict class labels for (d+Δd) at time (t+Δt). The results are displayed in Fig. 4.

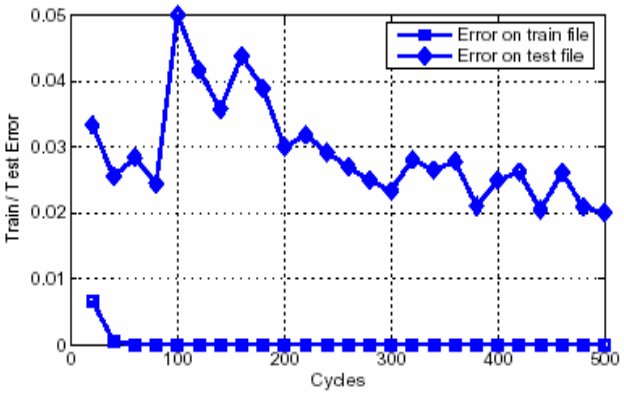

Fig. 4 First prediction results on Jerusalem data set: one week predict one week later

We also used a test set made of data for one day (d+Δd=10.10) at (t=07:00) in order to predict the class labels (d+Δd) at time (t+Δt). Results are displayed in Fig. 5.

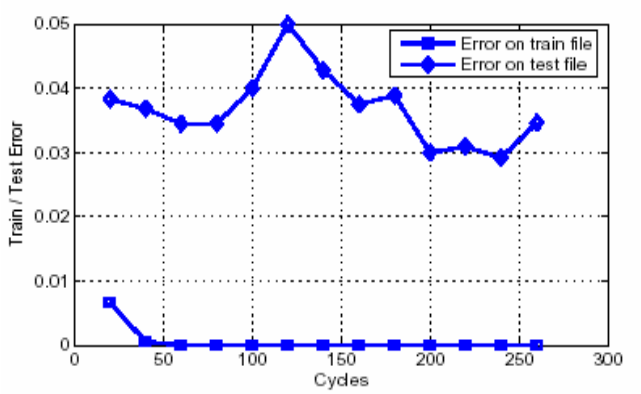

Fig. 5 First prediction results on Jerusalem data set: one week predict one day later

In order to check the quality of the traffic prediction, we also used the microscopic traffic simulator MITSIM [14]. This simulator represents networks at the lane level and simulates movements of individual vehicles using car-following, lane changing, and traffic signal response logic. Network of Jerusalem transportation system with approximately 100 intersections, with real data of signal timing and similar vehicles movements to the testing set were implemented to simulate traffic in Jerusalem. The cross-validation was done as follows: (a) run the simulator with different types of origin-destination data and gets output, (b) create train files for the machine learning from the output, (c) creation of the test files from origin-destination data, (d) predict traffic congestion, (e) run the simulation with test files as origin-destination data, and get output, and (f) compare the output of test files with the predicted values. We found an error rate of approximately 7%. In comparison, the naive predictor (consisting in the estimation of the future class labels by its current value) gives an error rate of 16%.

V. Conclusion

In this paper we addressed the issue of traffic congestion prediction, using a hybrid Adaboost-Random Forest Algorithm. A new method of prediction was proposed, which gives very promising results on both simulations and real data. The optimization of traffic lights according to given predictions, at well as a numerical study of its efficiency, should be investigated in further contributions.

References

[1] Breiman Leo. (2001) {Random Forests} Machine Learning 45 (1), 5-32 (Original Article).
[2] Breiman Leo. (2003) {Manual on setting up, using, and understanding random forestsv3.1}.
[3] Drucker H, Wu D, Vapnik V. (1999) {Support vector machines for spam categorization}. IEEE Transactions on Neural Networks.
[4] Katja Remlinger. (2002) {Paper report base on Breiman, L.(2001) Random Forests}. Machine Learning, 35, 5-32.
[5] Iyer R, Lewis D, Schapire R, Singer Y and Singhal A (2000) {Boosting for document routing}. In Proceedings of the Ninth International Conference on Information and Knowledge Management.
[6] J. R. Quinlan. {Bagging, boosting and C4.5}. Proc. of 13th AAAI, pp. 725730, 1996.
[7] J. Friedman, T.Hastie, R. Tibshirani. (1998) {Logistic Regression: a Statistical View of Boosting}. Tech. report July 23.
[8] Marko Robnik-Sikonja. (2004) {Improving Random Forests}. In J.F. Boulicaut et al.(eds): Machine Learning, ECML 2004.
[9] Schapire E. Robert. (1990) {The Strength of Weak Learnability}. Machine Learning, 5, 197-227.
[10] Theodoro Koulis. (2003) {Random Forests: Presentation Summary}. April 1.
[11] T. G. Dietterich. (1999) {An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization}. Machine Learning, Vol. 32, No.1, pp.122.
[12] Transportation Research Board.{Highway Capacity Manual, National Research Council, Washington, DC}, 2.
[13] Mahalel David, Gal-Tzur Ayelet, (2003) {Evaluation of the traffic flow In the signalized intersections. Final report of the Division of Transportation and Highways Engineering, Technion Institute}.
[14] Yang Qi, (1997) {A Simulation Laboratory for Evaluation of Dynamic Traffic Management Systems. Ph.D. Thesis, at Department of Civil and Environmental Engineering}.
[15] Anil Kamarajugadda, Byungkyu Park, (2003){Stochastic Traffic Signal Timing Optimization. Final report of ITS Center project: Signal timing algorithm}.
[16] Y. Freund, R. E. Schapire (1996), Experiments with a New Boosting Algorithm. In *Machine Learning: Proceedings of the Thirteenth International Conference.*