Machine Learning for Traffic Prediction

Jarosław Rzeszótko and Sinh Hoa Nguyen

Polish-Japanese Institute of Information Technology,
Koszykowa 86, 02-008 Warszawa, Polska
jrzeszotko@gmail.com, hoa@pjwstk.edu.pl

Abstract. Using machine learning for predicting traffic is described in the context of a competition organized using the TunedIT platform. A heuristic is proposed for reconstructing the route of a car in a street graph from a temporal stream of its coordinates. A resilient propagation neural network for approximating the average velocity on a given street from irregular time series of instantaneous velocities of cars passing through that street is discussed.

Key words: machine learning, data mining, spatio-temporal data mining, GPS, R-Tree, GIS, neural networks

1 Introduction and overview

The increase of population density and of the relative amount of car owners, makes traffic jams an important problem of modern societies. Traffic jams are a major source of discomfort of drivers, but also the cause of an increased number of traffic accidents, especially in large cities. In appreciation of this problem, a competition promoting research on predicting traffic was organized in June 2010 by Tom-Tom, a company producing automotive navigation systems [8]. While the competition consisted of three separate problems, the one analyzed in this paper is the third one, concerning predicting average car velocities on selected streets from instantaneous velocities of a selected group of cars driving through the city. The problem is stated in detail in section 2.

The spatio-temporal character of the data available in the competition raises a range of interesting sub-problems, from which the most immediate one is the problem of finding a cars route through a street graph from a temporal sequence of geographic coordinates retrieved from its GPS. An effective solution is discussed in section 3, consisting of combined usage of the well-known R-Tree data structure (subsection 3.1) and a heuristic developed by the authors for establishing the lane through which the car is driving (subsection 3.2).

Another sub-problem, due to the discrepancy between the kind of data available and the quantity that has to be predicted, is that of reproducing the average velocity on a given street from instantaneous velocities of the cars passing through it in a given time window. This was modelled as a regression problem and solved using a neural network trained with the resilient back propagation algorithm, as is discussed in section 4.

CONCURRENCY, SPECIFICATION AND PROGRAMMING
M. Szczuka et al. (eds.): Proceedings of the international workshop CS&P 2011
September 28-30, Pułtusk, Poland, pp. 442-453

Furthermore, experiments were carried out to assess the error of different neural network architectures when performing this task. The results of those experiments are presented in section 5, together with a comparison between the accuracy of prediction obtained in this work and other results from the contest.

2 Problem statement

The overall basis for the competition is a traffic simulator based on cellular automata written by Paweł Góra from the University of Warsaw [2]. In this particular problem, in a simulation of traffic in Warsaw, one percent of the drivers possesses cars equipped with a navigation system that is connected to the Internet and every ten seconds sends out a **notification** to some hypothetical central server. This notification is build from three parts: from the unique identifier, geographic coordinates and instantaneous velocity of the car. The goal of this challenge is to use a stream of such notifications from the first 30 minutes of a one hour long time window to predict velocities on one hundred selected streets of Warsaw, in two 6 minute time periods, one starting now and another 24 minutes from now.

To make this task possible, a training set was made available. The training set was generated by doing 50 ten-hour long simulations and had information about average velocities on each of the hundred selected streets in each consecutive six minute time window and the complete stream of all the notifications emitted by the selected 1% of the cars. Data is highly voluminous: 500 hundred hours of simulation with 10 time windows per hour and 100 streets selected makes up for 500 000 average velocity values and there is more then 135 million notifications.

The test set, that was used by the competing teams to generate the final solution consisted of streams of notifications from first 30 minutes of each hour of another 50 ten-hour long simulations - altogether around 65 million notifications. Two files describing the structure of streets in Warsaw as a directed graph were also made available, with one file describing the nodes of the graph as points specified by geographical coordinates and the second file describing connections between nodes. The second file also contained information about the length, number of lanes and the allowed maximal velocity on a given street. The structure of the training and test sets is best summarized using a picture:

Fig. 1. Information available in the problem

The solution had to be a vector of predicted values of average velocity of cars passing through each of the hundred selected streets, for two prediction periods, for each hour of the simulation from the test set, so with $2 \cdot 100 \cdot 10 \cdot 50 = 100\,000$ values altogether. The submissions were ranked using the root mean square error function of time required to travel through 1km of a given street, obtained through transforming the predicted and actual velocity values:

$$\sqrt{\frac{1}{n} \cdot \sum_{i=0}^{n} (\frac{60}{x_i} - \frac{60}{\bar{x_i}})^2} \tag{1}$$

n - number of predictions, x - predicted velocities vector, $\bar{x}$ - actual velocities vector

As one can see, the problem is in supervised learning, however, because of the unusual semantics of the data, it is impossible to directly use classic supervised learning techniques. The reasons for this, as well as proposed solutions, will be discussed in further sections.

3 Interpreting data from the navigation system

3.1 Searching for the street

The first problem comes from the fact that the prognosis of interest is to be generated for particular streets, in fact even for particular lanes, while the arriving notifications that form the basis for the prognosis come from particular geographic points specified by a latitude and longitude. Because of that, a way to map the notifications to the streets and lanes they come from is needed. The problem can be formulated more strictly as follows:

Given a sequence of consecutive car positions in the form of its geographic coordinates and a directed planar graph representing the structure of the streets in the city, find the edges of the graph through which the car has driven.

The problem is illustrated by **Figure 2**, with $X(k)$ being the function describing the position of the car in the kth second of the simulation. Since the following conditions hold:

- the structure of the street graph is given
- streets are divided into straight-line segments
- geographic coordinates for small distances can be treated as points in a Cartesian space

It follows that it is possible to check if a given geographic point lies on a given street by calculating the geometric distance between this point and the line segment equivalent to the street. However, with the street graph having over 20 000 nodes and over 35 000 edges, and with the operation of locating the street to be repeated for each of the several million notifications, it is not viable to check all the edges for each notification.

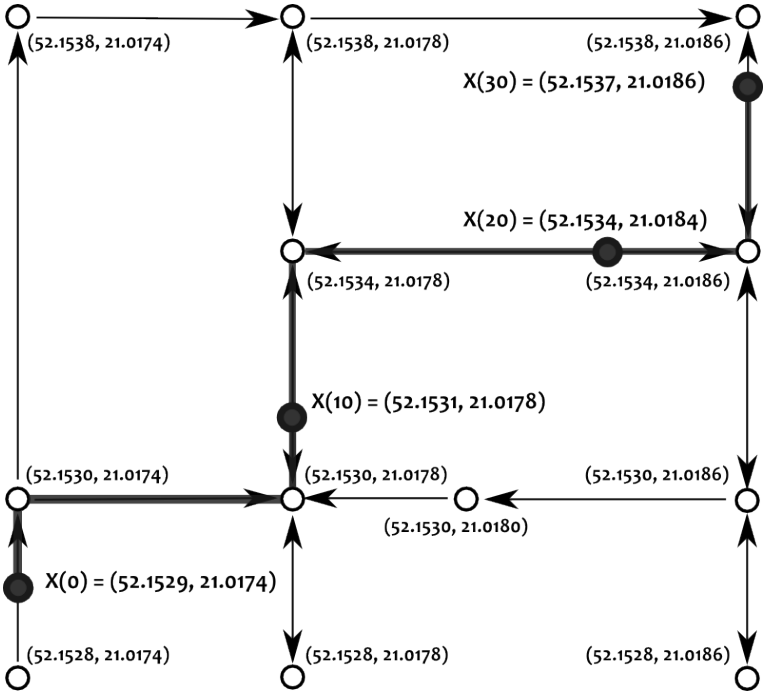

Fig. 2. The problem of interpreting data from the navigation system

There is a wide range of so-called space-partitioning data structures that address problems of a similar kind. In particular, R-trees [3], standing for rectangular trees, turned out to be very well suited for solving this problem. R-trees are a variant of B-trees in that every node in both an R-tree and B-tree can have up to some fixed number k of child nodes, where k is chosen upfront and is the same for the whole tree. Structuring the tree in this manner has important advantages in database applications, which traditionally have to a large extent stored data on hard drives - k can be chosen in a way that makes it possible to store all the objects in a given tree node in the same disk block, which can reduce the amount of hard drive seeks needed to load a group of somehow related objects being the result of a query executed using the tree.

However, while B-trees store objects ordered by some arbitrary comparison criteria, R-trees order the objects in such a way that objects geometrically close to each other have a common parent node with the least possible number of edges between any of the nodes and the common parent. When an R-tree is constructed from a collection of objects represented by rectangles on a plane, those objects are stored in the leaves of tree and an arbitrary node A in the tree is the parent of another node B, if the hyper-rectangle represented by A contains in itself the whole of the hyper-rectangle represented by B.

This structure makes it possible to effectively execute the following kinds of queries:

- for a given rectangle R, finding all the objects lying in R
- for a given rectangle R, finding all object which area partially or fully overlaps the area of R
- for a given rectangle R and a number $k \in N$, finding the k objects nearest to R

Those assumptions are common to all R-trees, however there are various different ways of doing the detailed construction of an R-tree, giving raise to various R-tree variants like Hilbert R-trees, R+-trees or R*-trees . Each of them offers different performance both in terms of asymptotic complexity and in real-world measurements. In this work, Priority R-Trees [1] were used, which give the strongest performance guarantees in terms of asymptotic complexity and have very good real-world benchmark results.

In the solution to this problem, each of the streets in the street graph of Warsaw is circumscribed by a rectangle with its sides parallel to the respective axes of the coordinate system. For each notification, a query is posed on the R-tree for all the rectangles containing the point from which the notification was sent. In the worst case, when this point lies in the rectangle of more then one street, this query results in a couple of possible streets from which the notification could came from and the precise result has to be found by checking the distance between the notification and each of the streets. If the street under consideration is a two-lane one, the decision from which of the lanes did the notification come from is still left to be done and has to be made in a different manner, described in the next subsection.

3.2 Searching for the lane

If only one position of a car is considered at a time, there is no way to find out on which of the lanes of a two-lane street the car is, because those two lanes are represented in the street graph as oppositely aligned edges between the same nodes, so that they both lie on the same straight line - every point lies in equal distance to both of them and there is no other information that could be used to find the lane.

To solve this problem, it is necessary to consider fragments of car routes instead. This was accomplished by loading the notifications into a hash table, where the key was the unique id of one of the cars and the value was the list of notifications sent by the car, ordered from the oldest to the most recent one. Next, two notifications have to be considered at a time: the one for which the lane is to be established and the next notification that was sent from different geographic coordinates - if the car is stopped and sends consecutive notifications from the same position it is impossible to distinguish the lane. Is important to note that this next notification does not have to come from a street connected to the street from which the previous one came from. Such case, serving as an example here, can be seen in **Figure 3**.

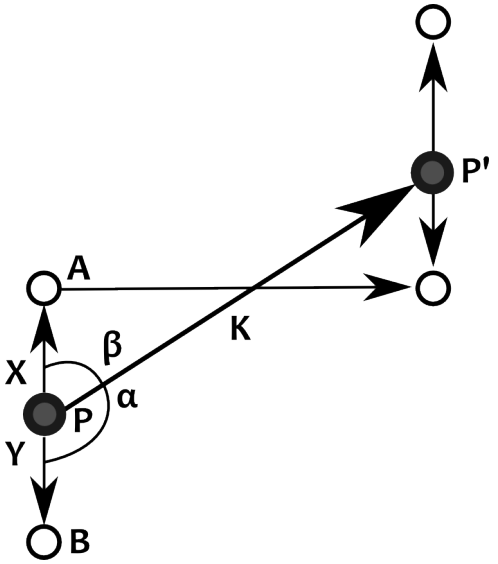

Fig. 3. The problem of finding the lane from which a notification came from

The car is known to drive through one of the edges of the graph connecting some points (equivalent to nodes in the graph) A and B, in some time instant t being at some point P, geometrically positioned on the line segment connecting points A and B. In the next time instant t', the car is at point P', not laying on this segment. In this situation, the answer to the question if the car is moving from point P towards point A or if it is moving towards point B is sought. To answer this question, three vectors have to be considered: the D vector, connecting points P and P', the X vector, connecting points P and A and finally the Y vector, connecting points P and B. The D vector represents the ideal direction in which the car should move to quickest reach its target (the point P') if the structure of the street graph is ignored, and the X and Y vectors the possible routes which the car could really take. Considering the high frequency of the notifications, one can with high probability assume, that the driver always chooses the route along the vector that is the least rotated with respect to the D vector, because this is the fastest way to go from point P to P'. This heuristic is equivalent to the intuitive reasoning which a person interpreting a picture like 3.1 typically does subconsciously. The α and β angles can be computed using the **cosine theorem**.

This way of finding the lane the car was driving through is of course only a heuristic and it is not hard to come up with situations in which it can fail. There is no 100% sure way of reconstructing the route of the car through a street graph on the basic of a discrete stream of notifications where the time periods between notifications are of significant length. The higher the frequency of the notifications, the higher the accuracy of this heuristic through.

This part of the solution to the problem of interpreting data from the GPS was independently developed by the authors. Although R-trees are a very well known data structure and a widely used one in the domain of Geographic Information Systems (GIS), its usage for this problem also turns out not to be obvious. From the publications that were published after the competition about the winners solutions, the work [4] used an *ad hoc* grid-based method for finding the road from which a notification came and it contains no mention of considering different lanes, while [7] acknowledged the existence of the lane problem, but did not propose any solution, claiming no significant impact on the accuracy of the final prediction. In this research, identification of the lane using the heuristic proposed improved the final accuracy of prediction about 10%. The approach chosen also resulted in low preprocessing times, with allocating the notifications to the streets and lanes taking only about 20 minutes, while [4] describes a less precise process (without taking lanes into consideration) taking more then 20 hours. While such absolute times cannot be compared directly due to the differences in hardware and implementing environment used, a difference of almost two orders of magnitude can seldom be attributed only to those differences, but rather to the usage of more efficient algorithms.

4 Approximating the current average velocity

4.1 Problem context

By using the methods described in the previous chapter, the sequence of notifications send out from each of the streets during the 30 minutes that are the basis for the prediction is established. Unfortunately, this data is hard to use directly because of its irregularity and sparseness. To better understand those difficulties, it is valuable to analyze a sample situation from the simulation:

Streets are described by consecutive letters of the alphabet, the bold street G is the one for which prediction is to be made, while $X_k(t)$ is the position of the car with id k, with $k \in \{1, 2, 3\}$. Cars number 1 and 2 begin their journey at points lying near the G street. Car number 1 starts on the L street, sends out the first notification in the zeroth second of the simulation, in under 10 seconds passes through the I street to arrive on the G street in the tenth second of the simulation and send out a notification. The car number 2 arrives on the N street in the tenth second of simulation, drives through the K street, sending out a notification from it, drives through the J street without sending a notification and in the thirtieth second of the simulation sends out a notification from the G street. In the twentieth second of the simulation there was no car on the G street and thus there does not exist a notification from this time. The car number 3 arrives in the area pictured around the sixty fifth second of the simulation on the M street, then sending out a notification in the eightieth second from the G street. As it can be seen, even through each car sends out a notification every ten seconds, for the selected streets there are notifications available only for the tenth, thirtieth and eightieth seconds of the simulation. Information is irregular

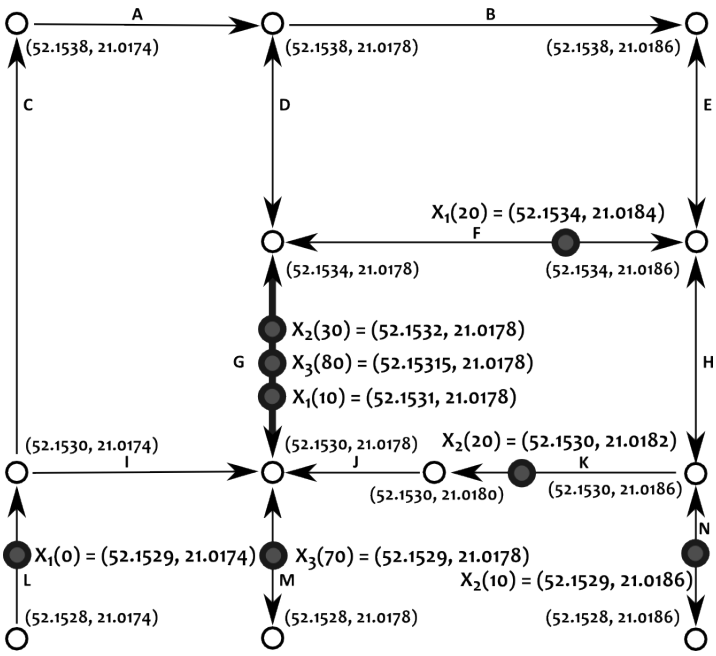

Fig. 4. Irregularity of the data available for each of the streets

and for this reason it can not for example be interpreted as classical time series for the purpose of prediction.

It is also important to keep in mind that only one percent of the drivers in the simulation has cars equipped with a navigation system. Due to the constant movement of cars, the set of streets from which notifications are available changes in each time instant and is a small subset of the set of all streets. Additionally, since the notifications are only sent every ten seconds and the edges of the graph often represent short, few meters long segments, frequently two consecutive notifications do not come from connected edges of the graph (like the car number 1 driving through the I street).

Due to those imperfections of the data available, it is necessary to first reconstruct the approximate time series of the current velocities in the five consecutive six minute time windows, before it will be possible to do any prediction at all. To this end, a neural network based model was constructed, approximating the average velocities in a given six minute time window on a given street. based on the stream of notifications from that street and from this period. Because frequently there is no notification at all on some streets for a particular time period, a simple algorithm for filling out the time series was developed as well. In this manner a regular time series of approximate velocities in the consecutive six minute time windows is reconstructed for each of the streets selected for prediction. This makes it potentially possible to apply traditional methods of time series prediction to this problem.

4.2 Design of a neural network for the problem

Every classical, feed-forward neural network, in particular based on perceptrons represents a particular function, being a combination of the non-linear activation functions of the individual neurons constituting the network. The architecture of the network, that is the number of its inputs and outputs, the number of the hidden layers and the kind of activation functions used, determines the class of functions that the neural networks is capable of approximating. As one can see, both the choice of the architecture of the network, as well as of the method of training, is critically important in the process of designing a model of a particular phenomena. Equally important is the choice of features that will serve as inputs of the network, as the correlation between them and the desired neural network output finally determines the possibility and accuracy of solution of a given problem.

The natural start of the neural network design process is thus the choice of the features that will be used as the inputs of the network. For each six minute time window, for which the approximate current velocity is sought, there is a variable number of notifications available that forms the basis for this approximation. However, the neural network has a fixed number of inputs, it is thus necessary to summarize the data in some way by a fixed number of parameters, correlated as much as possible with the current average velocity on the given street. An obvious candidate set of features is the set of summary statistics: the average, standard deviation, median, minimum or maximum of the sequence of notifications.

Less obvious is the fact, that the results of the network are much improved by using an input which has the higher value the lower is the average value of the notifications from the time window in concern. This is important in the frequent case, when the number of notifications is very low (1-3) and their value is low as well, while the average velocity on the street stays quite high, which is often the case for rarely frequented streets which do not experience traffic jams at all, yet for which a few low notifications still can happen, because the car with the navigation system can park or just start driving. Since the neural network output depends on the overall activation of the inputs, using only features like the average or standard deviation of the stream of notifications it is hard to optimize the weights of the network to yield a high output in this case, while still keeping good accuracy in the more usual cases. A particularly good feature to address this problem turned out to be the ratio between the amount of notifications with very low velocity values (below 5 km/h) to the amount of all notifications.

The next design decision concerned the number of hidden layers and neurons in each of the layers. According to theoretical results, a single hidden layer with $2d+1$ units for d inputs is enough to approximate a continuous function and its derivative with arbitrary precision [5]. A similar architecture turned out to be most successful in this problem; attempts to increase or decrease the number of hidden units increased the approximation error.

Finally, because of failed attempts to train the network using the classical method of back propagation, the network was trained using the less known resilient propagation technique, which augments the back propagation method

with a heuristic that makes the training process much less likely to get stuck in a local minima [6]. Of course, the input data also had to be normalized, and additionally some outlier filtering was applied.

5 Experiments and results

To choose the final best network architecture, an experiment was carried out comparing neural networks with different sets of features as inputs. Because only the training set contained both the notifications and the corresponding values of average velocity for the consecutive six minute time windows, it was divided for the purpose of the experiment into two parts, one used to train the model, second one used to measure the approximation error. The results of the experiment are shown in **Table 1**.

All the neural networks were subjected to 15 000 training epochs. The **A** network turned out to be the best in this experiment, but when using the full training set the **B** network performed slightly better and was chosen as the optimal one.

The final result of the approach outlined in this paper, generated by training the chosen network on the whole training set and treating the approximated current velocity as the prognosis for both prediction periods, in comparison to other competitors, is shown in the table below. As one can see, the correct approximation of the current velocity was enough to score a good place in the competition:

Place in the competition	**Name/nickname**	**RMS error**
1	hamner	6.7719
2	trafficlab	7.4556
3	Andrzej Janusz	7.5779
4	dleshem	8.6653
-	**Result from this paper**	**9.1563**
5	amrkabardy	9.6284
6	Ed Ramsden	12.9057
7	dmlab	13.0690
8	arson	13.6606
9	tinygray	15.1965
10	Tri Kurniawan Wijaya	16.6041
11	Baseline	18.0649
12	xiaohui_li	18.0649
13	LouisDuclosGosselin	18.0649
14	xiao	18.0649
15	Javafish	18.0649

Table 1. Comparison of accuracy of different neural network models for approximation of the current velocity

Approximation method	**RMS error**[1]
Neural network **A** with inputs: 1. Average of velocities from notifications 2. Number of notifications	**9.9240**
Neural network **B** with inputs: 1. Average of velocities from notifications 2. Ratio of low velocity notifications to the number of all notifications	10.0171
Neural network **C** with inputs: 1. Average of velocities from notifications 2. Standard deviation	10.3385
Neural network **D** with inputs: 1. Median of velocities from notifications 2. Ratio of low velocity notifications to the number of all notifications	10.0778
Neural network **E** with inputs: 1. Median of velocities from notifications 2. Standard deviation	10.1794
Neural network **F** with inputs: 1. Average of velocities from notifications 2. Number of notifications with velocities below 5 km/h 3. Number of all notifications	10.5004
Neural network **G** with inputs: 1. Number of all notifications 2. Average of velocities from notifications 3. Median of velocities from notifications 4. Standard deviation 5. Lowest velocity 6. Highest velocity	10.8365

[1] According to equation 1.

6 Conclusions

In summary, the following results were established:

- An efficient, precise way of reconstructing a cars route from its consecutive coordinates was developed, basing on existing algorithms and data structures with proven correctness and known computational complexity. For comparison, solutions [4, 7] use *ad hoc* methods that do not guarantee reasonable performance and ignore the lane problem altogether.
- A resilient propagation neural network model for approximating the average car velocity on a street in the given time window from a sequence of instantaneous velocities of the cars driving through that street was constructed. More specifically:
  - Effectiveness of neural networks in this application was shown
  - The set of features, number of hidden layers and neurons and method of training were established in a way guaranteeing good approximation precision

References

1. Lars Arge, Mark de Berg, Herman J. Haverkort, Ke Yi, *The Priority R-tree: a practically efficient and worst-case optimal R-tree* SIGMOD '04 Proceedings of the 2004 ACM SIGMOD international conference on Management of data, ACM New York, Nowy Jork, USA, 2004, ISBN: 1-58113-859-8
2. Paweł Góra, *Traffic Simulation Framework - a Cellular Automaton-Based Tool for Simulating and Investigating Real City Traffic*, Recent Advances in Intelligent Information Systems, EXIT, Warsaw, Poland, 2009, ISBN 978-83-60434-59-8, p. 641-653
3. Antonin Guttman, *R-Trees - A Dynamic Index Structure for Spatial Searching*, SIGMOD '84 Proceedings of the 1984 ACM SIGMOD international conference on Management of data, ACM New York, Nowy Jork, USA, 1984, ISBN: 0-89791-128-8, p. 47-57
4. Benjamin Hamner, *Predicting Travel Times with Context-Dependent Random Forests by Modeling Local and Aggregate Traffic Flow*, IEEE International Conference on Data Mining Workshops, Sydney, Australia, 2010, ISBN 978-0-7695-4257-7, p. 1357-1359
5. Kurt Hornik, *Approximation capabilities of multilayer feedforward networks*, Neural Networks, Volume 4, Issue 2, 1991, California, USA, ISSN 0893-6080, p. 251-257
6. Martin Riedmiller, Heinrich Braun, *A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm*, IEEE International Conference On Neural Networks, San Francisco, USA, 1993, ISBN 978-0-7803-0999-9, p. 586-591
7. Wei Shen, Yiannis Kamarianakis, Laura Wynter, Jingrui He, Qing He, Rick Lawrence, Grzegorz Swirszcz, *Traffic Velocity Prediction Using GPS Data: IEEE ICDM Contest Task 3 Report*, IEEE International Conference on Data Mining Workshops, Sydney, Australia, 2010, ISBN 978-0-7695-4257-7, p. 1369-1371
8. Marcin Wojnarski, Pawel Góra, Marcin Szczuka, Hung Son Nguyen, Joanna Świetlicka, Demetris Zeinalipour, *IEEE ICDM 2010 Contest: TomTom Traffic Prediction for Intelligent GPS Navigation*, IEEE International Conference on Data Mining Workshops, Sydney, Australia, 2010, ISBN 978-0-7695-4257-7, p. 1372-1376