INDIAN INSTITUTE OF MANAGEMENT
AHMEDABAD • INDIA

Travel Time Prediction for GPS Taxi Data Streams

Arnab Kumar Laha

Sayan Putatunda

W.P. No. 2017-03-03

March 2017

INDIAN INSTITUTE OF MANAGEMENT
AHMEDABAD-380 015
INDIA

Travel Time Prediction for GPS Taxi Data Streams

Arnab Kumar Laha

Sayan Putatunda

Production & Quantitative Methods Area
Indian Institute of Management Ahmedabad
arnab@iima.ac.in
sayanp@iima.ac.in

Abstract

The analysis of data streams offers a great opportunity for development of new method- ologies and applications in the area of Intelligent Transportation Systems. In this paper, we propose a new incremental learning approach for the travel time prediction problem for taxi GPS data streams in different scenarios and compare the same with four other existing methods. An extensive performance evaluation using four real life datasets indicate that when the drop-off location is known and the training data sizes are small to moderate the Support Vector Regression method is the best choice considering both prediction accuracy and total computation time. However when the training data size becomes large the Randomized K-Nearest Neighbor Regression with Spherical Distance becomes the method of choice. Even when the drop-off location is unknown then the Support Vector Regression method is the best choice when the training data size is small to moderate while for large training data size the Linear Regression method is a good choice. Finally, when continuous prediction of remaining travel time and continuous updating of total travel time along the trajectory of a trip are considered we find that the Support Vector Regression method has the best predictive accuracy. We also propose a new hybrid method which improves the prediction accuracy of the SVR method in the later part of a trip.

Keywords: Data Stream Mining, Incremental Learning, Real-time, Spherical Data Analysis, Stochastic Dominance, Streaming Data

1 Introduction

Nowadays, one can find and avail GPS based cab services such as Uber, Lyft, Didi, Ola etc. almost anywhere in the world. A GPS enabled taxi continuously collects and records the geo-spatial location data for each trip it travels. These recorded geo-spatial location data are often referred to as GPS traces which is a very rich data source for understanding the mobility patterns of passengers and also the demand.

The GPS traces are a rich source for *streaming data* which can be defined as continuous flow of data from a source that arrives at a very fast pace Ellis (2014). Streaming data (a.k.a Data Streams) allows for gathering of real time or near real time insights. Analysis of streaming data comes with a great deal of challenges which we will discuss in details in Section 2.1.

In this paper, we are interested in the travel time prediction problem. For a transport dispatch system, it is useful to know that for how long a cab will be occupied. This can help the transport dispatch system in vehicle allocation and can also help in improving their service efficiency. We propose a new method and adapt four other methods already existing in the literature to the streaming data context and carry out an extensive performance comparison study on four real world datasets. A similar comparison of performance study was conducted using a trajectory level dataset details of which is discussed in Section 9.

The rest of the paper is structured as follows. Section 2 gives a background of various concepts that we will be using in this paper. This is followed by a brief review of the literature in Section 3. In Section 4 we describe the dynamic travel time prediction problem. Section 5 discusses the methodology and Section 6 describes the datasets used in this paper. Section 7 discusses the results of the various experiments conducted. In Section 8 we discuss the static data experiment and the computation time analysis conducted for the different methods. Section 9 presents the analysis of the continuous travel time prediction and continuous updating of total travel time problem with a trajectory level dataset. Finally, Section 10 concludes the paper.

2 Background

2.1 Streaming Data

Streaming data (a.k.a. Data Streams) can be defined as a sequential and continuous flow of data from a source that arrives at a very high speed (Aggarwal, 2006, Muthukrishnan, 2003). The major sources of streaming data presently are customer clickstreams, social networks, GPS data streams, operational monitoring using sensors, mobile data traffic, online advertising and Internet of Things (IoT). Analysis of streaming data can benefit decision making in multiple domains viz. network monitoring, high frequency finance, web mining, etc (Ellis, 2014). Analysis of streaming data comes with its share of challenges which were identified in Aggarwal (2006), Gama (2010) and Babcock et al. (2002) as (a) single pass processing of data since multiple passes are not feasible (b) the presence of *Concept Drift* i.e. the characteristics of the incoming streaming data may change over time and (c) fast near real time analysis of data due to high speed of incoming streaming data.Thus, we see that streaming data is very different from static data.The conventional methods for static data analysis assume that the entire data is always available and multiple passes over the data set is possible. Since both of these assumptions are not true in a streaming data set up, analysis of streaming data needs newer methods.

A major challenge for streaming data mining algorithms is the ability to tackle the *Concept Drift*. Often such algorithms need to discard older data points and update the model parameters frequently to ensure that the model performs well. These algorithms continuously learn from the incoming streaming data which is very much different from batch learning methods where we don't need to update the parameters repeatedly.

The streaming data mining algorithms can be broadly classified into *online learning* or *incremental learning* algorithms based on the update frequency of model parameters.The online learning methods update the model parameters as new observations comes in whereas an incremental learning algorithm updates the parameters when a batch of new training examples comes in (Büttcher et al., 2010). We will discuss this further in the Methodology section of this paper (see Section 5).

2.2 Spherical Data

Sometimes observations come in the form of directions. It can be either in two, three or even higher dimensions. This kind of data is known as Directional data and the area of statistics that deals with such data is known as Directional data analysis. Directional data in two dimensions can be viewed as points on a unit circle and directional data in three dimensions can be viewed as points on a unit sphere. That's why these are called *Circular data* and *Spherical data* respectively (Jammalamadaka and Sengupta, 2001, Mardia and Jupp, 2000). Spherical data arise in various contexts such as in understanding structure of proteins in bioinformatics (Oldfield and Hubbard, 1994), geological studies of paleomagnetism in rocks (Mardia and Jupp, 2000), etc.

The analysis of spherical data is very different from that of linear data. For example let us take the case of a basic summary statistic like mean. The spherical mean of n spherical data points $y_1, \ldots, y_n$ is $\bar{y}_0 = \bar{y}/||\bar{y}||$ and not simply $\bar{y}$ which is the case with trivariate linear data(Mardia and Jupp, 2000). Both Fisher et al. (1993) and Mardia and Jupp (2000) gives a detailed account of different methods for statistical analysis of Spherical data. But most of these methods are in the batch learning setting and application of these methods in the streaming data context has not been previously reported in the literature to the best of our knowledge.

In the context of this paper, we focus on the spherical data analysis for geo-spatial location coordinate data since these location coordinate points can be seen as points on earth (which is approximately spherical). We will incorporate the idea of spherical distance in k-NN regression (see Section 5.2) and will also compare it's performance with conventional k-NN regression i.e. using euclidean distance assuming location coordinates are linear data (please refer to Table 3).

2.3 K Nearest Neighbour (k-NN) Regression

Suppose there are n training pairs $(x_1, \theta_1), \ldots, (x_n, \theta_n)$ where x_j is of dimension p and θ_j indicates the class category of x_j. Now suppose that $\mathcal{T}$ is a test pattern whose class category is not known. Now, if

$$d(\mathcal{T}, x_u) = min\{d(\mathcal{T}, x_j)\} \tag{1}$$

where $j = 1, \ldots, n$ and d is a distance metric, then $\mathcal{T}$ is assigned class θ_u

(Cover and Hart, 1967, Murty and Devi, 2011).

The above rule is referred to as the nearest neighbor rule (Cover and Hart, 1967). A generalization of this rule is the k-NN algorithm where a new observation $\mathcal{T}$ is assigned the the class category that occurs most frequently in $S_{\mathcal{T}}$ where $S_{\mathcal{T}}$ consists of k observations ($k > 1$) from the training set which are nearest to $\mathcal{T}$ as per the distance metric d. In case of a tie, it may be broken by choosing one of the tied classes at random. In a regression set-up the above method can be used for predicting the value of the response for a given test instance by averaging the values of the response for the observations in $S_{\mathcal{T}}$. This method is referred to as k-NN Regression (Navot et al., 2006).

The choice of the distance metric can greatly affect the performance of the k-NN model and the choice of the same depends on the application in hand. The most commonly used distance metric is the Euclidean distance. Alternative distance metrics that have been considered in the literature include Manhattan distance, Mahalanobis distance and many more (Weinberger and Saul, 2009). In Section 5.2 of this paper we have used the *spherical distance* as the distance metric.

2.4 Artificial Neural Networks

The origins of Neural networks can be traced back to studies trying to understand and mathematically represent the information processing in biological systems (Rosenblatt, 1962, Rumelhart et al., 1986, Widrow and Hoff, 1988). In this paper, we focus on a specific class of neural networks called "Feed forward neural networks" (a.k.a Multilayer Perceptrons) (Bishop, 2006). A neural network basically consists of an input layer, an outpur layer and with one or more hidden layers sandwitched between the input and output layers. Each layer consists of one or more nodes (a.k.a. neurons). A basic neural network structure with one hidden layer is given below. Suppose the input layer consists of P nodes coreesponding to the input variables are $x_1, \ldots, x_P$. Then let

$$c_j^{(1)} = \sum_{i=1}^{P} w_{ji}^{(1)} x_i \; + \; w_{j0}^{(1)} \tag{2}$$

where $j = 1, \ldots, N$ and the superscript (1) denotes that all these parameters belong to the first (or hidden) layer of the network. Equation 2 represents N

linear combinations of the input variables $x_1, \ldots, x_P$ where N is the number of nodes in the hidden layer. $w_{j0}^{(1)}$ is known as the *bias*, $w_{ji}^{(1)}$ are the *weights* and c_j^1 are called the activations. These activations are transformed by a non-linear differentiable activation function $g(\cdot)$, as represented below in equation 3.

$$d_j = g(c_j^{(1)}) \tag{3}$$

The d_j's represent the output of the hidden units (i.e. hidden nodes). The d_j's are again combined as shown in equation 4 below to obtain c_k^2's which are the *output unit activations*.

$$c_k^{(2)} = \sum_{j=1}^{N} w_{kj}^{(2)} d_j \; + \; w_{k0}^{(2)} \tag{4}$$

Here $k = 1, \ldots, K$ and K denotes the total number of outputs. Again the superscript (2) denotes that these weights are related to the second layer of the network (in this case the output layer). Then again an activation function $g^2(\cdot)$ is used to transform the $c_k^{(2)}$'s to finally give the network outputs $y_k = g^2(c_k^{(2)})$ (Bishop, 2006). The activation functions are often taken as the sigmoidal function. A neural network can be used for both classification and regression problems.

A neural network can be represented using a network diagram and in case of a feed forward neural networks, there is no closed directed cycles (Bishop, 2006). Ripley (1996), Kreinovich (1991), Hornik et al. (1989) and Hornik (1991) explores the approximation properties of feed forward neural networks and Hornik et al. (1989) describes multilayer feed forward neural networks as "universal approximators". In this paper, we have used a three layer feed forward neural network with one hidden layer and a sigmoid activation function. The number of hidden nodes is taken in accordance with the Geometric pyramid rule (Masters, 1993) which states that for a three layer feed neural network with p inputs and q outputs the number of hidden nodes H in the hidden layer is

$$H = \sqrt{p * q} \tag{5}$$

2.5 Support Vector Regression

The Support Vector Regression (SVR) technique (Vapnik, 1995) is an extension of the Support Vector Machine (SVM) technique for classification (Boser

et al., 1992). Using the *kernel trick* the support vector machines map the input vectors x into a high dimensional feature space $\mathcal{Z}$ (Cortes and Vapnik, 1995). This makes the data linearly separable in the higher dimensional space (Hastie et al., 2009).

Suppose the training data is represented as $\{(x^{(1)}, z_1), \ldots, (x^{(n)}, z_n)\} \subset \mathcal{I} \times \mathbb{R}$ where $\mathcal{I}$ denotes the feature space of the input patterns. A simple linear regression function is represented as given below in equation 6.

$$y(x) = w \cdot x + b \; where \; w \in \mathcal{I}, \; b \in \mathbb{R} \tag{6}$$

Here w and b are the parameters and $(w \cdot x = \sum_{i=1}^{k} w_i x_i)$ represents the dot product of parameter $w = (w_1, \ldots, w_k)$ and input data $x = (x_1, \ldots, x_n)$. In case of support vector regression, the quadratic error function used in ordinary least squares (OLS) linear regression is replaced by an ϵ-sensitive error function (Vapnik, 1995). For any $\epsilon > 0$, the ϵ-sensitive error function F_ϵ has the property that $F_\epsilon(y(x) - z) = 0$ if $|y(x) - z| < \epsilon$ (Bishop, 2006). Here, y(x) and z denotes the prediction and the target respectively. So the goal is to fit a linear function y(x) such that $y(x^{(i)})$ are within the ϵ deviation from the targets z_i for as many of the training data points as possible (Smola and Schölkopf, 2004). Since an ϵ deviation between $y(x^{(i)})$ and z_i is "permissible", so this technique is also known as ϵ-SVR. More precisely, in ϵ-SVR we minimize the regulated error function as formulated by Vapnik (1995) which is given below

$$Min \;\; C \sum_{n=1}^{N} F_\epsilon(y(x^{(n)}) - z_n) + \frac{1}{2}||w||^2 \tag{7}$$

where C is the regularization parameter, F_ϵ and $y(\cdot)$ are as given above.

Vapnik (1995) also gave an alternative formulation for this optimization problem given in equation 7 by using slack variables ζ_n, ζ_n^* as given below

$$\begin{aligned} \text{Minimize} \quad & \frac{1}{2}||w||^2 + C \sum_{n=1}^{N} (\zeta_n + \zeta_n^*) \\ \text{subject to} \quad & z_n \leq y(x_n) + \epsilon + \zeta_n, \\ & z_n \geq y(x_n) - \epsilon - \zeta_n^*, \\ & \zeta_n, \zeta_n^* \geq 0, C > 0. \end{aligned} \tag{8}$$

The above optimization problem can be solved by introducing Lagrange multipliers and by optimizing the Lagrangian (see Bishop (2006) for more details).

In this paper, as is typical for non linear regression problems, we first use the "kernel trick" with the Radial basis function (RBF) kernel discussed in James et al. (2014). We then apply the SVR algorithm as discussed above. The RBF kernel is defined as

$$K(x, x^*) = exp(-\gamma||x - x^*||^2) \tag{9}$$

where $\gamma > 0$ is a parameter. The training time and space complexity for SVR (or SVM) with kernels are $\mathcal{O}(n^3)$ and $\mathcal{O}(n^2)$ respectively where n is the size of the training dataset (Tsang et al., 2005). Thus, we can see SVR requires a high computation time especially for large datasets.

2.6 Ensemble Methods

Ensemble learning is an effective method for increasing the prediction accuracy for out-of-sample data and it is widely used in machine learning and statistics disciplines (Hastie et al., 2009). It basically combines the predictions from various alternative models. Over the years various technique have been developed for creating ensembles e.g- Bagging(Breiman, 1996), Boosting (Freund and Schapire, 1997, Schapire, 1990), Random Forests (Breiman, 2001) and Stacking (Wolpert, 1992). A summary of these techniques for ensemble learning can be found in Hastie et al. (2009).

In this paper, we focus on multi-model ensembles. The simplest way to create such an ensemble is to use a variety of regression models on the training data and then combine their predictions using an averaging scheme. Among the various combination methods, the simplest to implement is the averaging rule since it needs no prior training (Hjort and Claesken, 2003, Kotsiantis and Pintelas, 2005). The other combination techniques available in the literature include weighted average and Bayesian Model Averaging (BMA) (Hjort and Claesken, 2003). We have chosen the simple averagng rule for use in this paper (see Section 7).

2.7 Stochastic Dominance

Suppose there are two distributions P and Q with cumulative distribution functions (CDFs) F_P and F_Q respectively. The distribution Q is said to be stochastically dominant over distribution P at first order, if for every x, $F_P(x) \geq F_Q(x)$ (Davidson, 2008). The distribution Q is said to be second order stochastically dominant over P if for every x, $\int_{-\infty}^{x} F_Q(y)dy \leq$

$\int_{-\infty}^{x} F_P(y)dy$ (Levin, 2006). Stochastic dominance of higher orders can be similarly defined. Also, it is well established fact that stochastic dominance at an order say $\mathcal{S}$ implies stochastic dominance at all orders higher than $\mathcal{S}$ (see Davidson (2008) for more details).

Geometrically the stochastic dominance of the first order can be visualized by examining the plots of the CDFs for the two distributions P and Q together. If P is stochastically dominant over Q of the first order then the CDF of P is to the right of that of Q and they do not cross each other. Since in many situations, as in this paper, the CDFs P and Q are not known they need to be estimated from the data. Since by the Dvoretzky-Kiefer-Wolfowitz inequality (Wasserman, 2010, p. 99) we know that the empirical cumulative distribution function (ECDF) approximates the CDF very well when sample size is large, we use the ECDF for geometrically checking the stochastic dominance of first order in this paper in Sections 7.2.1, 7.3.1 and 9.

2.8 Geodesic Distance

The geodesic distance between predicted and actual drop off points is calculated using the Vincenty inverse formula (Vincenty, 1975) and is called the Geodesic Distance Error GDE. Geodesic distance or Great circle distance can be defined as the shortest path between two points on the surface of the earth (Economou et al., 2004). If there are two points say x and y on the surface of the earth (i.e. a sphere with radius R) then the geosedic distance between these two points is represented by G_{xy}. The radius of earth is assumed to be 6378137 meters and (ϕ_x, μ_x) and (ϕ_y, μ_y) denotes the latitude and longitude of the points x and y. Then the geodesic distance between these two points can be obtained by using the following equation (i.e. equation 10) as described in Gade (2010).

$$G_{xy} = R \cdot \arccos(\sin\phi_x \cdot \sin\phi_y + \cos\phi_x \cdot \cos\phi_y \cdot \cos(\mu_x - \mu_y)) \qquad (10)$$

In this paper, we use the Vincenty inverse formula for calculating the geodesic distances between points on earth's surface. The Vincenty inverse formula assumes earth to be an ellipsoid (WGS84 coordinates) to calculate the geodesic distance. This is more accurate than the great circle distance

methods like Haversine distance (Vincenty, 1975). Chang et al. (2010) discusses the Vincenty formula for the computation of the geodesic distance D_{xy} and it is represented in equation 11 below.

$$G_{xy} = R \cdot \arctan\left(\frac{\sqrt{(\cos\phi_y \sin(\mu_x - \mu_y))^2 + (\cos\phi_x \cos\phi_y - \sin\phi_x \cos\phi_y \cos(\mu_x - \mu_y))^2}}{\sin\phi_x \cdot \sin\phi_y + \cos\phi_x \cdot \cos\phi_y \cdot \cos(\mu_x - \mu_y)}\right) \quad (11)$$

3 Related Work

In this section, we present a brief literature review of the work done on the travel time prediction problem with GPS enabled vehicles. GPS traces are instrumental in finding interesting insights that have applications such as passenger finding (Veloso et al., 2011), hotspot identification (Chang et al., 2010), vacant taxi finding (Phithakkitnukoon et al., 2010), trajectory mapping (Liu et al., 2012), traffic monitoring (Herring et al., 2010), etc. A summary of various applications of GPS traces in the transportation industry is given in Chen (2014). The travel time prediction of vehicles from GPS traces is a challenging problem having lots of applications in the transportation domain such as taxi dispatching (Xie et al., 2013), ridesharing (Ma et al., 2013), etc.

Most of the work in the literature for solving the travel time prediction problem have used batch learning methods. Mendes-Moreira et al. (2012) worked on the long term travel time prediction problem. They did a comparison of three different methods and found that SVM gave the best results out of them. Some of the commonly used methods reported in the literature for solving the travel time prediction problem are linear regression models (Patnaik et al., 2004), artificial neural networks (Jeong and Rilett, 2004), SVR (Wu et al., 2004), etc.

In recent years, there has been some work reported in the literature which has given emphasis to the streaming nature of the GPS data focusing on real time or near real time prediction. Some of the simpler methods used in this context are based on OD matrix and how it evolves over time (see for more details in Moreira-Matias et al. (2016) and Barceló et al. (2010)). Lee et al. (2009) worked on a real-time knowledge based travel time prediction model using OD-pairs and meta-rules.

Tiesyte and Jensen (2008) worked on real time position tracking and travel time prediction of vehicles using the *nearest neighbor technique (NNT)*

technique. Hofleitner et al. (2012) worked on arterial travel time forecasting with streaming data using a hybrid model approach. Recently, some work has been reported in the literature that uses Artificial neural network (ANN) for real-time travel time prediction from GPS data (Bai et al., 2015, Gurmu and Fan, 2014).

Lam et al. (2015) worked on real time prediction of destination and travel time estimation from given partial trajectories. They used an ensemble learning model for trip time prediction. Wang et al. (2014) worked on a real time model for predicting the travel time of a vehicle in a city using the GPS trajectory data of vehicles. Luo et al. (2013) worked on the travel time prediction problem based on the most frequently used path extracted from large trajectory data. Wibisono et al. (2016) discusses prediction and visualization of traffic in a particular region using the FIMT-DD method with streaming data.

4 Dynamic Travel Time Prediction

In this paper, we consider the dynamic travel time prediction (DTTP) problem in three different situations. In the first case, we address the problem of predicting the travel time of a vehicle when the pickup location and the drop-off coordinates are both known. In the second case, we consider the more difficult situation of predicting the travel time when only the pickup location coordinates is known. In the third and final case, we address the prediction of travel time at different points on the trajectory of the vehicle when the drop-off coordinates are known. We explore two different types of problems here. The first one is the continuous prediction of remaining travel time at each point in the trajectory for a trip and the second one is dynamic updating of the total travel time at each point in the trajectory for a particular trip.

5 Methodology

In this section, we propose two new methodologies namely, KNN Regression with Spherical Distance- KNNRSD and Randomized KNN Regression with Spherical Distance - RKNNRSD for predicting the travel time when (a) both the pickup and drop-off location coordinates are known and (b) only the

pickup coordinates are known. The performance of these methods are then compared with feed forward neural networks- ANN, linear regression- LR and support vector regression - SVR suitably adapted for the streaming data setting. (see Section 5.4). We also build some ensembles of these methods and compare their performance (see Section 7).

In this paper, our focus is on building an incremental learning algorithm. Büttcher et al. (2010) mentions that an incremental learning algorithm can be approximated by using a batch learner along with a sliding window . We implement this idea by using a batch learner $\mathcal{L}$ which is fed a sequence of data points p_1, p_2, ... , p_m using a windowing technique (see Section 5.1) where the value of m may vary from window to window.

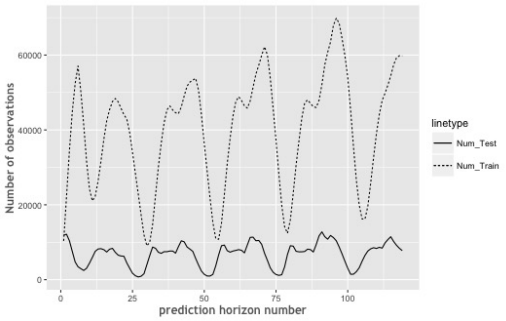
5.1 Damped Window Model

As discussed earlier, the phenomenon of "concept drift" presents a formidable challenge when we are dealing with streaming data. A windowing technique is a powerful technique to handle concept drift. Cao et al. (2006) defines a damped window model as a windowing technique that uses an exponential fading strategy to discard the old data. For this purpose, a fading function is used where the weight assigned to each data point decreases with respect to time t. The fading function is represented below

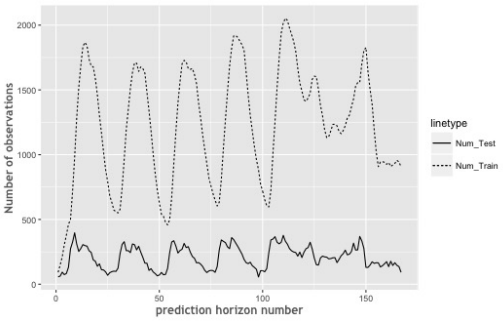
$$g(t) = 2^{-\lambda t},\ where\ \lambda > 0 \tag{12}$$

The decision regarding the discarding of the older data can be influenced by changing the value of the decay factor (λ). If the value of λ is high then less importance is given to the older data compared to the more recent data. In this paper, we have used a variation of the damped window model technique where instead of assigning weights to each individual data points based on the fading function, we assign weights to a batch of data points in an input data stream window and discard the older ones based on an user defined cutoff. Once the length of the window is determined using the damping function mentioned in Equation 12, we treat this as a mini batch and proceed.Therefore, this may also be called a "Mini-batch Window Model" as it has been done in Putatunda (2017).

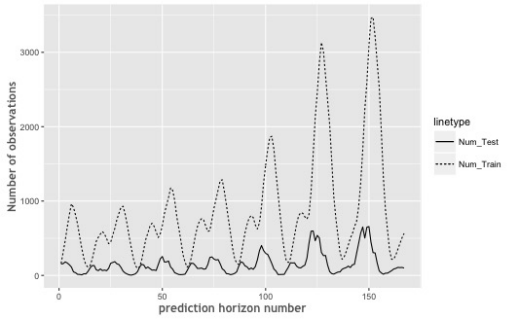
Figure 1 shows the distribution of number of observations in training window and testing window (i.e. the prediction horizons) for the 1 hour Damped window model for the NYC1, PORTO and SFBLACK1 datasets.



(a) NYC1- 1 hour Window



(b) PORTO- 1 hour Window



(c) SFBLACK1- 1 hour Window

Figure 1: Number of observations vs. prediction horizon number for 1 hour window model for different datasets

5.2 K-NN Regression with Spherical Distance

In this paper we use the KNNRSD method for solving the travel time prediction problem. We also propose a variant of the KNNRSD which is discussed in Section 5.3. The KNNRSD has its origins in the literature on Spherical data analysis (see Section 2.2) and K-NN Regression method (see Section 2.3). The motivation behind using this method is that the predictor variables i.e. the pickup and drop-off location coordinates (or just the pickup location coordinates) are points on the surface of earth which can be taken approximately as a sphere. To the best of our knowledge, there has been no work reported in the literature that takes into account the spherical nature of the data while solving the travel time prediction problem for GPS enabled taxis in streaming data context.

In this paper, we treat the predictor variables as spherical data but the response variable (in both cases) is linear. Then we perform the KNN regression with the distance metric as *Spherical distance* which is defined as the shortest route along the surface between two points U and V lying on the surface of a sphere. The Spherical distance for the points U and V on an unit sphere is defined as given below.

$$S_d = \arccos(U \cdot V) \tag{13}$$

where $U \cdot V$ represents the dot product (Ratcliffe, 2006).

For comparison purpose we also use KNN-Regression method where the distance metric is the *Euclidean distance* (see Section 7). We will refer to this method as KNNRED.

The KNNRSD algorithm is described below in Algorithm 1. Since the value of k is chosen from an array given to the user in some cases especially for very sparse datasets we may have training windows, the size of which is less than our chosen value of k. In such cases the we need to modify the choice of k for that window to ensure that the algorithm doesn't terminate. For this window, we take the value of k to be an integer lower than the training window size n. For example, suppose we want to run the k-NN regrssion with spherical distance with $k = 50$ but the number of observations in the training window is say 49, then the algorithm chooses the next value of k from the array which should be lower than 49 i.e. $k = 25$.

Algorithm 1 KNNRSD Pseudocode

Require: $n > 0$, $t > 0$, $k \in [5, 10, 15, 20, 25, 50, 100]$ $where, n \rightarrow Training\ Window\ Size$, $t \rightarrow Test\ Window\ Size$, $k \rightarrow no.\ of\ nearest\ neighbors\ (chosen\ from\ the\ array)$, $k_{[i]} \rightarrow the\ value\ of\ k\ at\ the\ i^{th}\ position\ in\ the\ array$, $i \rightarrow 1\ to\ length(array\ k)$
Choose k^* from array k {Let $k^* = k_{[i]}$}
if $k_{[i]} < n$ **then**
Set $k = k_{[i]}$
Run k-NN Regression {with Spherical distance metric}
else
Initialize $j = 1$
While $k_{[i-j]} > n$
$j \rightarrow j + 1$
End While
Set $k = k_{[i-j]}$
Run k-NN Regression {with Spherical distance metric}
end if

5.3 Randomized K-NN Regression with Spherical Distance (RKNNRSD)

For large training datasets, the K-NN method is computationally expensive. It is a memory based technique and has no training cost since whole training dataset is kept in memory and used for finding similarity with test instances. The time complexity of the K-NN algorithm is $\mathcal{O}(np)$ for a training dataset with n training patterns and p dimensions (Kusner et al., 2014). So the KNN method has speed or memory related issues especially when the dataset size is large. One way to deal with this problem is to reduce the size of the training dataset without adversely affecting the accuracy of the algorithm. Also, by deploying efficient algorithms one can increase the computation speed of the KNN method (Murty and Devi, 2011). Some of the well- known approaches for increasing the computation speed of KNN reported in the literature are cluster based trees (Zhang and Srihari, 2004), the branch and bound technique (Fukunaga and Narendra, 1975, Miclet and Dabouz, 1983), ordered partitions (Kim and Park, 1986), the projection algorithm (Friedman et al., 1975), hashing methods (Papadopoulos and Manolopoulos, 2005) and also use of some specialized data structures such as k-d trees (Shakhnarovich

et al., 2005).

In this paper we propose a new method RKNNRSD where we first perform a simple random sampling of the training dataset. Since this sample is representative of the whole training dataset we replace the training dataset with this randomly sampled subset and proceed to use the KNNRSD algorithm as discussed in Section 5.2. The proportion of the original dataset that needs to be sampled (*sampling rate*) is an important decision variable and later in this paper we do extensive sensitivity analysis to suggest a thumb rule for the same. Later in this paper we demonstrate that this method greatly increases the computation speed without sacrificing the accuracy of the procedure.

It may be noted that sometimes especially for sparse training datasets, the sample size obtained with a sampling rate of $r\%$ may be less than the value of the k nearest neighbors where k is the user defined input. In such cases we run the KNNRSD on the whole training data.

Algorithm 2 RKNNRSD Pseudocode

Require: $n > 0,\ t > 0,\ 0 < r < 1,\ k \in [5, 10, 15, 20, 25, 50, 100],\ s = r \times n,\ where,\ n \rightarrow Training\ Window\ Size,\ t \rightarrow Test\ Window\ Size,\ r \rightarrow Sampling\ Rate,\ s \rightarrow sample\ size\ after\ performing\ SRS\ with\ sampling\ rate\ r,\ k \rightarrow no.\ of\ nearest\ neighbors\ (chosen\ from\ the\ array)$

if $s < k$ **then**

Run KNNRSD (with *training window size* $= n$)

else

Run KNNRSD (with *training window size* $= s$)

end if

5.4 Other Methods

In this paper, we will compare the proposed method with three different methods using the evaluation metrics that are discussed in Section 7. The methods are ANN (Bishop, 1995), SVR (Vapnik, 1995) and linear regression (Fox, 2008). All of these methods are executed along with a Damped window model as is the case with the KNNRSD and RKNNRSD methods discussed above.

LR The usage of the simple linear regression method has been reported in the literature for solving travel time prediction problem mostly in the batch data setting as mentioned in Section 3. Since we are working with streaming data we adapt the linear regression technique to devise an incremental method using a Damped window model in this paper. We call this method LR.

ANN The motivation behind selecting the artificial neural networks for comparison purpose is its popularity for solving the travel time prediction problem as discussed in Section 3. We apply a three layer feed forward neural network with one hidden layer and a sigmoid activation function. The number of hidden nodes is taken to be 2. This is in accordance with the Geometric pyramid rule as discussed in equation 5. We adapt the standard ANN algorithm to the streaming data setting by using a Damped window model. We call this method ANN in this paper.

SVR The SVR method, which is a popular non-linear and non-parametric technique in the batch data setting but it is not that popular in the streaming data set-up. Two major issues are (a) the computation of support vectors need matrix based operations and (b) the support vectors need to be stored in the memory during the training stage which might be infeasible for some applications (Laskov et al., 2006, Moreira-Matias et al., 2016). Apart from this, the time complexity of SVR is $\mathcal{O}(n^3)$ (see Section 2.5) and so for large training datasets, the computation time is very high. But in batch setting, the usage of SVR for solving the travel time prediction problem is reported in the literature and also in some studies it has been seen that SVR performs relatively better than other methods. Taking these into account and also the fact that not much work has been done with SVR for solving the travel time prediction problem in streaming data context, we decide to adapt the SVR algorithm using the Damped window model and use it for comparison purpose. We have implemented an ϵ-SVR method as discussed in Section 2.5 with a RBF kernel. Henceforth we will call this method SVR.

6 Data

We use the New York City (NYC) Taxi and Limousine Commission (TLC) yellow taxi data from 1st January, 2013 to 5th January, 2013 as our primary

dataset. This is a publicly available dataset which can be found at the NYC Taxi and Limousine Commission website (NYC-TLC, 2014). The dataset description and other details related to its pre-processing are given in Section 6.1. Moreover, we have also tested the performance of our methods on other real world datasets which are discussed in Section 6.2. We have taken a subset of the data (period- for few days or 1 week) rather than the entire data available at these public data sources because our focus is mainly on examining the performance of the different methodologies and comparing them. Since we are working with streaming data and are using a windowing approach with a fixed prediction horizon, the total size of the data is not going to affect the performance of these incremental learning algorithms as it would have in the batch data setting.

6.1 NYC Yellow Taxi GPS Data

The NYC Yellow Taxi GPS dataset consists of various attributes related to the taxi pickup and drop-off coordinates, their corresponding timestamps, information related to trip travel time and distance travelled and payment related information. The dataset has been cleaned of missing values and some other anomalies like erroneous GPS coordinate values. The total number of observations in the cleaned dataset for the period 1st to 5th January, 2013 is 824,799. This dataset serves as the primary dataset on which the different models are tested. It is used for tasks such as determining the value of K to be used in KNNRSD, determining the "best" sampling rate for the RKNNRSD, comparing the performance of the various methods etc. We will refer to this dataset as NYC1.

Another slice of this dataset for the time period 13th - 16th January, 2013 containing 680,865 observations (after data cleaning), is used for comparison of performances of the different methods. We will refer to this dataset as NYC2.

For both NYC1 and NYC2, we would be using the attributes as given in Table 1.

6.2 Other datasets

This section gives a brief description of the two other datasets that have been used in this paper for the purpose of testing the performance of the different methods.

Table 1: NYC1 and NYC2 - Attribute Description

Attributes	Description
medallion	an unique id of the taxi - vehicle bound
pickup_datetime	time when the passenger(s) were picked up
dropoff_datetime	time when the passenger(s) were dropped off
pickup_longitude	pickup location's longitude coordinate
pickup_latitude	pickup location's latitude coordinate
dropoff_longitude	drop-off location's longitude coordinate
dropoff_latitude	drop-off location's latitude coordinate
trip_time_in_secs	time taken for the trip
trip_distance	distance travelled in miles

6.2.1 Porto GPS Taxi Data

The Porto GPS taxi dataset is publicly available and was first used in Moreira-Matias et al. (2013). This dataset contains for each trip information regarding taxi stand, call origin details, unique taxi id, unique trip id and the Polyline. The Polyline is a string of GPS coordinates with each coordinate being obtained after every 15 seconds of the trip beginning with the pickup and ending with the dropoff. For our analysis we have taken the trip details for 1st- 7th July, 2013 which consists of 34,768 observations. We obtained the travel time, pickup coordinates (start_latitude, start_longitude) and drop-off coordinates (dest_latitude, dest_longitude) from this dataset. We will refer to this dataset as PORTO.

6.2.2 San Francisco black cars GPS traces

This publicly available dataset consists of anonymized GPS traces of Uber black cars in San Francisco for one week (1st-7th January, 2007) (Henry, 2011). For each trip location coordinates are recorded every 4 seconds starting with pickup and ending with drop-off. The first record for each trip gives the pickup time, pickup latitude and longitude and the last record gives the drop-off time, dropoff latitude and longitude. Also, similar to the PORTO dataset, we obtained the travel time, pickup coordinates and drop-off coordinates for each trip from this dataset. This derived dataset has 24,552 observations after data cleaning. We will refer to this dataset as SFBLACK1.

For the continuous updating and continuous prediction problem as given in Section 9, we take the trajectory level data for the 1st day of the week (1st January, 2007) of the black cars GPS trace data (Henry, 2011). This

dataset has 92,694 observations after data cleaning and comprises location details and the corresponding timestamps for 1791 unique trip IDs. We will refer to this dataset as SFBLACK2.

7 Experimental Results

In this section, we report the findings of the different experiments done on the NYC1 dataset with the four methods discussed in Section 5 to assess their performance. The performance evaluation metrics are discussed in Section 7.1. The entire data stream flow, model building and data analysis is implemented using the software R version 3.3.1(R Core Team, 2016). The R packages Directional (Tsagris and Athineou, 2016), lubridate (Grolemund and Wickham, 2011), nnet (Venables and Ripley, 2002), e1071 (Meyer et al., 2015) and ggplot2 (Wickham, 2009) has been used for doing our experiments. A sensitivity analysis on the choice of λ, window size and sampling rate (for RKNNRSD method) has been carried out and the results are reported in Section 7.2. The experiments were carried out on a system with 24 GB RAM and Intel Xeon processor 2.67 GHZ with a 64 bit Windows Server 2012 OS.

7.1 Evaluation Metrics

To compare the performance of different methods, we will use the Aggregated Mean Absolute Error (AMAE) and the Aggregated Median Absolute Error (AMedAE) as the evaluation metrics.

If we record the Mean Absolute Errors (MAE) for each prediction horizons then the AMAE can be defined as follows.

$$AMAE = \frac{\sum_{i=1}^{n} MAE_i}{n} \tag{14}$$

where n is the number of prediction horizons, MAE_i is the MAE in the i^{th} prediction horizon.

Similarly, if we record the Median Absolute Errors (MedAE) for each prediction horizons then the AMedAE can be defined as follows.

$$AMedAE = \frac{\sum_{i=1}^{n} MedAE_i}{n} \tag{15}$$

where n is the number of prediction horizons, $AMedAE_i$ is the MedAE in the i^{th} prediction horizon.

The *AMAE* and *AMedAE* are the two evaluation metrics used in this paper for comparing the performance of the different methods. The method with the least value of the chosen evaluation metric is considered the "best" method as per that metric. The units of both of these evaluation metrics are in "seconds".

7.2 Results- Travel Time Prediction when the drop-off coordinates are available

In this section, we discuss the results of the various experiments that we have conducted for the travel time estimation problem when both pickup and dropoff coordinates are known as described in Section 4. We first apply the KNNRSD method on the NYC1 dataset and perform a sensitivity analysis to select the most appropriate parameters by comparing the evaluation metrics (AMAE and AMedAE). We do the same for RSKNNRSD as well. Then we compare the results of the RSKNNR (with the chosen parameters) with the other methods viz. LR, ANN and SVR as discussed in Section 5.4.

In table 2 we show the performance of the KNNRSD method for different values of K, window sizes and λ. We vary K= 5, 10, 15, 20, 25, 50 and 100, window sizes = 15 minutes, 30 minutes and 1 hour and $\lambda = 0.25$, 0.5 and 0.75. Please note that we could have also taken much lower window sizes say 2 mins or 5 mins especially for the NYC1 dataset. But for sparse datasets such as SFBLACK1 and PORTO, there are none or very few observations in some of the windows. For example, if we take the case of 5 minutes window size for the 1st hour in the SFBLACK1 dataset, around 25% of the 5 minutes windows contains no observations. This can cause problem for our comparison exercise since in some cases the model will not be updated for time duration greater than 5 minutes. Hence, we focus our study on window sizes of 15 minutes, 30 minutes and 1 hour.

We find that for $K = 25$, the prediction accuracy is highest consistently for different window size and values of λ. Hence, we choose $K = 25$. For the window size, we can see that there isn't much difference in the AMAE/ AMedAE values for three window sizes considered in this experiment. We therefore proceed with a window size of 1 hour. From the discussion in Section 5.1, we see that the value of λ is dependent on the amount of concept drift. For this experimental study we choose a middle value i.e. $\lambda = 0.5$.

In Table 3, we demonstrate that there isn't much difference in AMAE/

Table 2: Performance of KNNRSD method for different values of K, λ and window sizes ($\mathcal{W}$)

K	$\mathcal{W}$	λ					
		0.25		0.5		0.75	
		AMAE	AMedAE	AMAE	AMedAE	AMAE	AMedAE
5		173.16	117.20	168.56	117.68	168.10	121.26
10		166.81	114.71	163.17	114.46	163.30	117.52
15		165.19	114.44	162.22	113.90	163.00	116.73
20	15 min	164.80	114.89	162.46	114.04	163.74	116.24
25		164.83	115.68	163.10	114.51	164.80	116.23
50		166.70	120.32	167.87	118.10	171.03	117.84
100		171.55	127.78	176.20	124.25	181.50	121.53
5		184.84	131.26	173.80	121.35	170.67	118.97
10		178.04	127.64	167.41	117.75	164.90	115.52
15		175.95	126.50	165.69	116.68	163.55	114.77
20	30 min	175.05	125.99	165.25	116.36	163.43	114.62
25		174.67	125.88	165.25	116.31	163.75	114.89
50		174.74	125.97	167.01	117.90	166.87	117.26
100		176.53	127.41	171.71	121.23	173.27	122.02
5		207.63	151.76	185.78	132.32	179.58	125.45
10		200.10	148.12	179.15	128.24	173.14	122.62
15		198.11	147.30	177.07	127.14	171.29	121.71
20	1 hr	196.95	146.88	176.14	126.56	170.66	121.08
25		196.35	146.22	175.78	126.45	170.45	120.95
50		195.57	145.92	175.85	126.49	171.36	121.82
100		196.06	145.88	177.69	127.90	174.48	123.94

AMedAE values if we work with Euclidean distance instead of the spherical distance. So, we proceed with the spherical distance metric in the KNNRSD since it hasn't been explored much in the intelligent transportation domain.

Table 3: Comparison of KNNRSD (Spherical distance) and KNNRED (Euclidean distance) for different values of K using a 1 hour Damped window model with $\lambda = 0.5$ on the NYC1 dataset

K	KNNRSD		KNNRED	
	AMAE	AMedAE	AMAE	AMedAE
5	185.78	132.32	185.96	131.92
10	179.15	128.24	179.39	128.38
15	177.07	127.14	177.34	127.32
20	176.14	126.56	176.39	126.69
25	175.78	126.45	176.01	126.74
50	175.85	126.49	176.06	126.78
100	177.69	127.90	177.83	128.06

In table 4 the performance of RKNNRSD is given for different sampling rates viz. 5%, 10% and 20%, K and window sizes with $\lambda = 0.5$. As before we vary K= 5, 10, 15, 20, 25, 50 and 100 and window sizes = 15 minutes, 30 minutes and 1 hour. For similar reasons as discussed in the case of KNNRSD we choose $K = 25$ and window size = 1 hour. We note that at sampling rate of 20 % the AMAE/ AMedAE values are quite close to that of KNNRSD . Thus we use R25NNRSD20P (i.e. RKNNRSD with $K = 25$ and sampling rate = 20%) for further study.

As discussed in Section 2.6, Ensemble methods sometimes work better than the individual methods. In view of this we create a multi-model ensemble comprising SVR and R25NNRSD20P.

7.2.1 Prediction Accuracy Analysis - Travel Time Prediction when drop-off coordinates are available

In this section, we have studied the five methods viz. (i) R25NNRSD20P, (ii) ANN, (iii) LR, (iv) SVR and (v) the Ensemble of R25NNRSD20P and SVR methods for predicting the travel time using both the pickup and drop-off latitude and longitude as predictor variables. A 1 hour Damped window model with $\lambda = 0.5$ is fixed. This study is carried out on four different datasets namely, NYC1, NYC2, PORTO and SFBLACK1 and the results obtained are given in Table 5. Among the nonensemble methods the prediction accuracy of the SVR method is found to be the best followed by the

Table 4: Performance of RKNNRSD method for different values of K, Sampling Rate and window sizes ($\mathcal{W}$)

K	$\mathcal{W}$	Sampling Rate					
		5%		10%		20%	
		AMAE	AMedAE	AMAE	AMedAE	AMAE	AMedAE
5		195.74	138.84	185.66	130.82	178.00	124.64
10		198.23	140.22	184.86	130.64	175.03	123.28
15		204.46	144.42	188.04	133.03	176.51	124.26
20	15 min	211.55	148.26	191.95	135.80	178.72	126.07
25		218.81	152.79	195.83	138.14	181.15	127.88
50		249.22	172.65	215.69	150.14	192.84	135.70
100		256.74	186.89	247.49	171.02	214.05	148.87
5		193.07	136.75	185.57	130.60	180.30	126.57
10		191.88	135.51	182.51	129.23	175.69	123.73
15		194.80	137.42	183.67	129.73	175.61	124.06
20	30 min	198.40	140.04	185.78	131.28	176.51	124.87
25		202.16	142.44	188.06	132.73	177.82	125.81
50		220.40	153.49	199.14	139.97	185.16	130.64
100		249.19	171.40	218.66	151.82	197.47	138.71
5		197.79	140.56	193.21	136.49	189.90	134.97
10		192.56	138.16	187.74	134.11	184.03	131.63
15		192.54	138.19	186.74	134.08	182.69	130.94
20	1 hr	193.71	138.97	186.91	134.39	182.36	131.04
25		195.44	139.99	187.53	134.57	182.48	131.22
50		205.70	146.08	192.83	138.16	185.07	132.96
100		223.71	156.74	204.18	145.10	191.48	137.34

R25NNRSD20P method. The prediction accuracy of the ensemble comprising SVR and R25NNRSD20P is seen to be close to be comparable with that of the SVR method.

Table 5: Performance of the five methods on NYC1, NYC2, PORTO and SFBLACK1 datasets using a 1 hour Damped window model with $\lambda = 0.5$

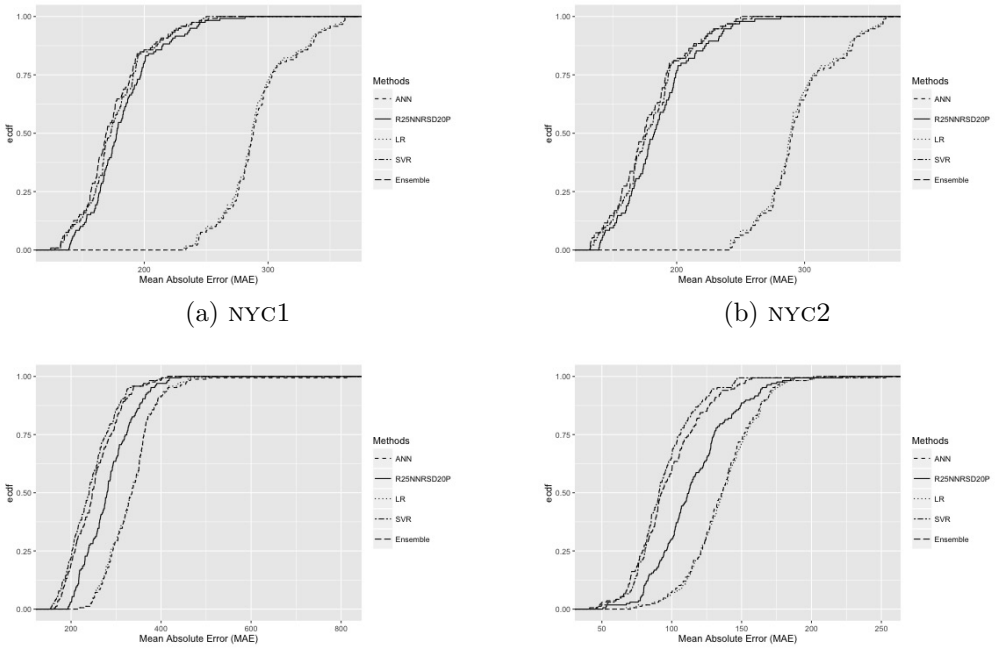
Method Name	Dataset Name							
	NYC1		NYC2		PORTO		SFBLACK1	
	AMAE	AMedAE	AMAE	AMedAE	AMAE	AMedAE	AMAE	AMedAE
R25NNRSD20P	182.48	131.22	185.89	133.72	283.75	184.35	115.37	80.72
LR	290.40	233.17	294.04	236.04	331.56	252.22	137.04	114.97
ANN	292.06	235.80	295.81	238.84	334.98	251.63	136.14	114.29
SVR	176.54	125.52	180.03	128.19	243.92	142.89	106.70	74.80
Ensemble	174.60	125.69	178.07	128.37	251.74	156.18	97.14	68.44

An alternative way to compare the performances of these five methods is by using the concept of Stochastic Dominance as discussed in Section 2.7. We compute the MAE for each prediction horizon and use the same to compute the ECDF of the MAE. The ECDF of the MAE is defined as

$$F_n(x) = \frac{\#\{MAE_i \leq x\}}{n} \tag{16}$$

where n is the number of prediction horizons, MAE_i is the MAE in the i^{th} prediction horizon and $\#\{MAE_i \leq x\}$ is the number of MAE_i that are less than or equal to x.

Figure 2 shows the ECDF plots for the five methods on the four datasets discussed above. For NYC1, we can see from 2(a) that the curves for LR and ANN are to the right of other three. The curve for the Ensemble method is leftmost, followed by SVR and R25NNRSD20P. So, in terms of prediction accuracy, the Ensemble method is the best performer, followed by SVR and R25NNRSD20P. Similar conclusion can be drawn for NYC2 as well (see figure 2(b)). In case of PORTO (see figure 2(c)), we find that the curve for SVR is to the left of the Ensemble method. So SVR is most accurate here followed by the Ensemble Method and R25NNRSD20P. For the SFBLACK1 dataset (see figure 2(d)) we see that the curves for SVR and Ensemble method cross each other. Hence neither of them stochastically dominate the other. This lends further support to our insights obtained from Table 5.



(a) NYC1 (b) NYC2

(c) PORTO (d) SFBLACK1

Figure 2: ECDF plots for the Mean Absolute Errors (MAE) by prediction horizons for each of the five methods on the four dataset

7.3 Results- Travel Time Prediction when drop-off coordinates are not known

In this section, we discuss the travel time estimation problem when only pickup coordinates are known as described in Section 4. As in Section 7.2 we first apply the KNNRSD method on the NYC1 dataset and perform a sensitivity analysis to choose the appropriate values of window size, K, and λ using the evaluation metrics. We do the same for RKNNRSD for the values of window size, K and sampling rate. We then compare the results of the RKNNRSD method (with chosen parameters) with the LR, ANN and SVR as discussed in Section 5.4.

In table 6 we show the performance of the KNNRSD method for different values of K, window sizes and λ. We find that $K = 50$ is a good choice of K keeping in mind both the prediction accuracy and computation time. Moreover, for relatively sparse datasets a higher value of K may lead to overfitting. As before we set window size = 1 hour and $\lambda = 0.5$.

In table 7 the performance of RKNNRSD is given for different sampling rates viz. 5%, 10% and 20%, K and window sizes with λ held fixed at 0.5. For similar reasons as discussed above we choose $K = 50$ and the window size is taken to 1 hour. It can be seen by comparing Table 6 and Table 7 that for sampling rate = 20% the AMAE/ AMedAE values of RKNNSRD are closest to the corresponding values of KNNRSD in terms Absolute Percentage Error. We call RKNNRSD with $K = 50$ and sampling rate = 20% as R50NNRSD20P and use the same for further study.

As discussed in Section 2.6, Ensemble methods sometimes work better than the individual methods. In view of this we create five different ensembles with various combinations of the methods viz. R50NNRSD, SVR, LR and ANN. The first ensemble is a combination of R50NNRSD and SVR and we will refer to this as Ensemble1. The next one i.e. Ensemble2 is a combination of SVR and LR. Ensemble3 is a combination of SVR, LR and R50NNRSD. Ensemble4 is a combination of R50NNRSD and LR. And finally, Ensemble5 is a combination of ANN and LR. Table 8 gives the performance of these five ensembles for a 1 hour Damped window model with $\lambda = 0.5$. We find that the Ensemble2 is the best performer among the five ensembles.

Table 6: Performance of KNNRSD method for different values of K, λ and window sizes ($\mathcal{W}$)

K	$\mathcal{W}$	λ					
		0.25		0.5		0.75	
		AMAE	AMedAE	AMAE	AMedAE	AMAE	AMedAE
5		352.42	270.90	350.00	269.33	349.22	268.50
10		338.05	265.66	335.95	263.51	335.36	263.26
15		333.08	264.25	330.99	262.17	330.27	260.97
20	15 min	330.66	263.24	328.43	261.06	327.85	260.49
25		329.18	262.87	327.05	260.77	326.67	260.18
50		326.40	261.90	324.59	260.61	324.29	259.99
100		325.39	262.23	324.20	261.19	324.16	261.14
5		357.28	273.36	353.55	271.60	352.65	270.66
10		342.60	268.74	339.32	266.6	338.26	265.44
15		337.41	267.52	334.33	264.5	333.24	263.50
20	30 min	334.75	266.93	331.86	263.97	330.76	262.78
25		333.17	266.25	330.32	263.61	329.19	262.52
50		329.88	265.12	327.47	262.76	326.39	262.06
100		328.55	265.05	326.43	263.27	325.70	262.36
5		360.80	278.55	357.43	273.38	355.76	273.68
10		345.63	273.49	342.69	268.86	340.96	267.38
15		340.53	272.23	337.55	267.2	336.01	265.83
20	1 hr	337.97	271.64	334.88	266.79	333.41	265.30
25		336.33	271.10	333.25	266.23	331.87	264.82
50		332.98	270.31	330.11	265.24	328.95	264.33
100		331.42	273.59	328.86	265.23	327.83	264.10

Table 7: Performance of RKNNRSD method for different values of K, Sampling Rate and window sizes ($\mathcal{W}$)

K	$\mathcal{W}$	Sampling Rate					
		5%		10%		20%	
		AMAE	AMedAE	AMAE	AMedAE	AMAE	AMedAE
5		353.5	271.42	352.35	271.03	351.83	270.08
10		340.44	267.38	338.55	265.91	337.56	264.89
15		336.46	267.03	334.5	265.19	333.18	263.72
20	15 min	334.74	267.4	332.5	264.78	330.96	263.28
25		334.17	267.94	331.55	265.15	329.89	263.41
50		334.09	271.55	330.46	266.82	328.36	264.45
100		311.28	270.69	332.11	270.58	328.89	266.59
5		355.12	272.98	354.4	272.46	354.5	272.89
10		341.73	268.38	340.51	267.87	340.04	266.83
15		337.43	267.29	336	266.21	335.24	265.47
20	30 min	335.35	266.99	333.96	265.72	332.9	264.91
25		334.44	267.35	332.86	265.98	331.66	265.09
50		333.15	269.04	331.1	266.53	329.77	265.4
100		334.07	272.08	331.48	268.61	329.73	266.62
5		358.61	275.31	358.06	275.04	357.24	275.15
10		344.37	270.85	343.43	269.98	342.87	269.21
15		339.96	270.07	338.65	268.68	337.87	267.82
20	1 hr	337.86	269.78	336.46	268.44	335.43	267.49
25		336.61	269.23	335.17	268.4	333.98	267.11
50		334.49	269.84	333.09	268.45	331.58	267.13
100		334.27	271.13	332.69	269.53	331.03	267.34

Table 8: Performance of Ensemble1 (combination of R50NNRSD and SVR), Ensemble2 (combination of SVR and LR), Ensemble3 (combination of SVR, LR and R50NNRSD), Ensemble4 (combination of R50NNRSD and LR) and Ensemble5 (combination of ANN and LR) using a 1 hour Damped window model with $\lambda = 0.5$

Method	AMAE	AMedAE
Ensemble1	304.12	239.24
Ensemble2	299.00	234.12
Ensemble3	303.75	243.71
Ensemble4	315.26	260.56
Ensemble5	311.52	256.88

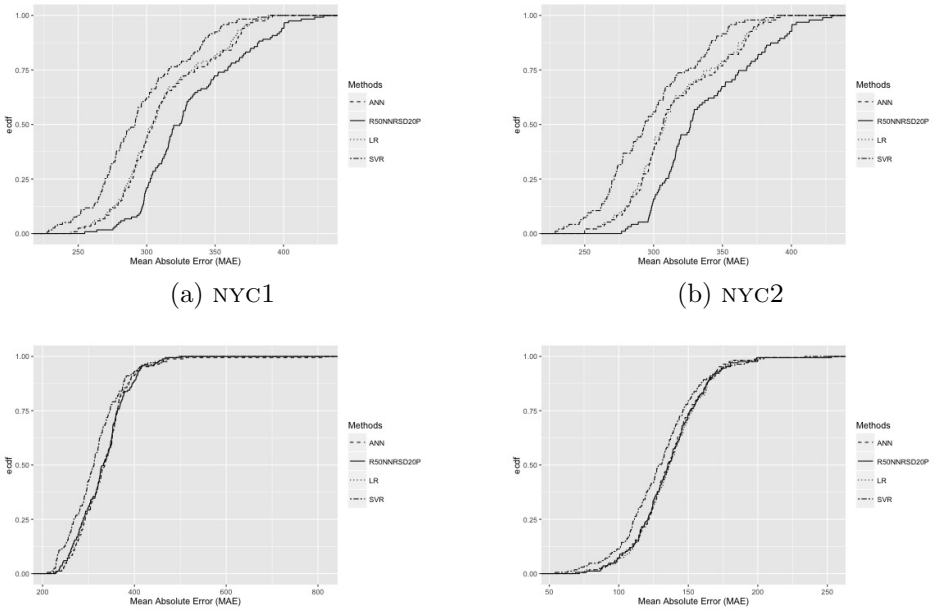
7.3.1 Prediction Accuracy Analysis- Travel Time Prediction when drop-off coordinates are not known

In this section, we have examined the prediction accuracy of the different methods viz. R50NNRSD20P, ANN, LR, SVR and the five different ensembles described in Section 7.3. We consider a 1 hour Damped window model with $\lambda = 0.5$. The results obtained for applying these methods on each of the following datasets namely, NYC1, NYC2, PORTO and SFBLACK1 are given in Table 9. We find that SVR method has the highest predictive accuracy followed by Ensemble2.

Table 9: Performance of the different methods on NYC1,NYC2, PORTO and SFBLACK1 datasets using a 1 hour Damped window model with $\lambda = 0.5$

Method Name	Dataset Name							
	NYC1		NYC2		PORTO		SFBLACK1	
	AMAE	AMedAE	AMAE	AMedAE	AMAE	AMedAE	AMAE	AMedAE
R50NNRSD20P	331.58	267.13	336.73	270.96	331.64	248.19	136.10	111.42
LR	310.97	255.88	315.06	258.94	331.56	252.22	137.05	115.02
ANN	312.22	257.87	316.25	260.94	334.98	251.63	136.15	114.30
SVR	294.83	217.91	298.07	220.16	313.35	212.99	129.96	93.17
Ensemble1	304.12	239.24	307.85	242.01	317.37	226.73	130.82	102.08
Ensemble2	299.00	234.12	302.60	236.66	317.44	227.41	131.38	102.93
Ensemble3	303.75	243.71	307.47	246.49	320.12	233.20	132.21	105.99
Ensemble4	315.26	260.56	319.44	264.01	329.65	248.29	135.98	113.27
Ensemble5	311.52	256.88	315.58	259.95	332.74	251.38	136.49	114.76

Figure 3 shows the ECDF plots of the Mean Absolute Errors (MAE) when the four non-ensemble methods are applied on the four datasets. For NYC1, we can clearly see in figure 3(a) that the curves for R50NNRSD20P, LR and ANN are to the right of the SVR. So the SVR is the best performer here. Among LR, ANN and R50NNRSD20P, we can clearly see that the curve for the R50NNRSD20P is to the right of the other two whereas both the LR and ANN curves cross each other. So LR and ANN are comparable and both of them perform better than the R50NNRSD20P. Similar conclusion can be drawn for NYC2 as well (see figure 3(b)). For both the PORTO and SFBLACK1 datasets, it can be concluded that the SVR is a better performer when compared to the other three methods. Also we cannot say anything about the stochastic dominance of one method over the other among LR, ANN and R50NNRSD20P since their curves cross each other (see figure 3(c)

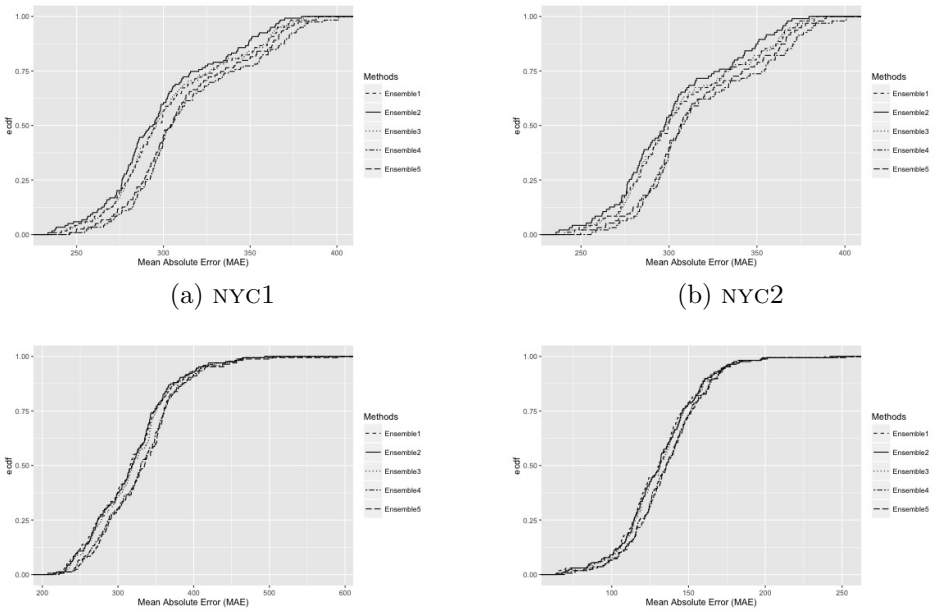


(a) NYC1 (b) NYC2

(c) PORTO (d) SFBLACK1

Figure 3: ECDF plots for the Mean Absolute Errors (MAE) by prediction horizons for SVR, R50NNRSD20P, LR and ANN on each of the four datasets

and 3(d)).



(a) NYC1 (b) NYC2

(c) PORTO (d) SFBLACK1

Figure 4: ECDF plots for the Mean Absolute Errors (MAE) by prediction horizons for different ensembles on each of the four datasets

We can see in table 9 that none of the five ensembles perform better than the SVR but most of them perform better than the other three individual methods i.e. LR, ANN and R50NNRSD20P. In figure 4, we show the ECDF plots for the five different ensembles on each of the four datasets. For NYC1, we can clearly see in figure 4(a) that ECDF curve of the Ensemble2 is to the left of all the other ensembles and hence has the best predictive accuracy. The ECDF curves of Ensemble1 and Ensemble3 cross each other but both of them perform better than Ensemble4 and Ensemble5. Similar conclusion can be drawn for NYC2 as well (see figure 4(b)). For both PORTO and SFBLACK1, we find that ECDF curves of Ensemble1, Ensemble2 and Ensemble3 crosses each other and they are to the left of Ensemble4 and Ensemble5. Hence, Ensemble4 and Ensemble5 have lesser predictive accuracy than the other three ensembles.

8 Discussion

The choice of the suitable method for use needs to take into account the time-accuracy trade-off. Since, in a streaming data context, the paradigm is to provide good results fast rather than giving the best possible solution we need to take into account the the total time taken for the method to run (i.e. sum of training and prediction time) along with the prediction accuracy of the method. The SVR method performs consistently well in terms of prediction accuracy but it takes relatively higher time especially for large datasets. In section 8.1 we investigate the variation of the total time taken by the different methods for different data sizes.

8.1 Static Data Experiment

We first perform an experiment to demonstrate how the time and accuracy varies with respect to the training data size for each of the following three methods viz. SVR, R25NNRSD20P, and LR for travel time prediction when the drop-off location coordinates are known. We take random samples of different sizes for training and test data from the NYC1 dataset. The data size for different training data windows varies from 262 to 70,000 in the three datasets viz. NYC1, porto and SFBLACK1. So for this experiment we consider training datasizes of $250, 500, 750, 1000, 2000, \ldots, 10000, 12000, 14000, \ldots,$ $20000, 25000, 30000, \ldots 70000$. We take $1/5^{th}$ of each of the training data size as test data size. We then apply each of the three methods and plot the MAE and the total time taken with respect to the training data size for these four methods as shown in figure 5.

In figure 5, we can see that even though LR is the fastest of the three methods but it has the worst predictive accuracy among them. The SVR can be recommended for training data size less than 10,000 since it has higher predictive accuracy and the time taken is not very high compared to that of the other methods. But for training data size more than 10000, we find that R25NNRSD20P has the best predictive accuracy and also the total time taken is much lower than that of the SVR method.

We carry out a similar experiment for travel time prediction when the drop-off coordinates is unknown. We again apply the three methods viz. SVR, R50NNRSD20P and LR and plot the MAE and the total time taken with respect to the training data size as shown in figure 6. We find here that in terms of predictive accuracy, SVR is the clear winner. But again for

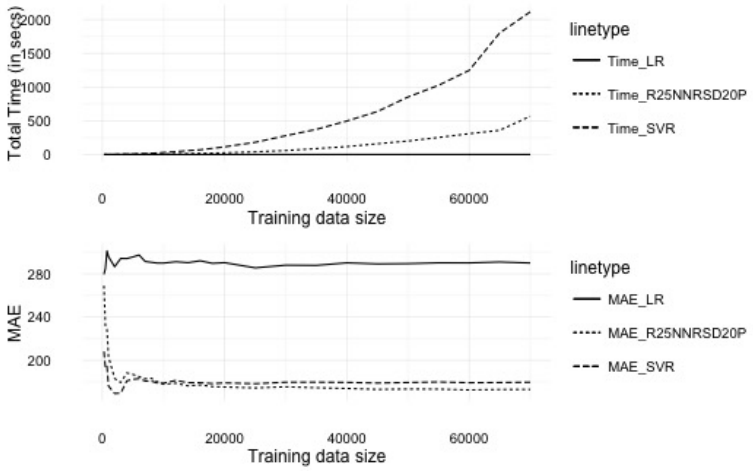


Figure 5: Time and MAE vs. different training data size for static experiment with travel time prediction when the drop-off location coordinates are known

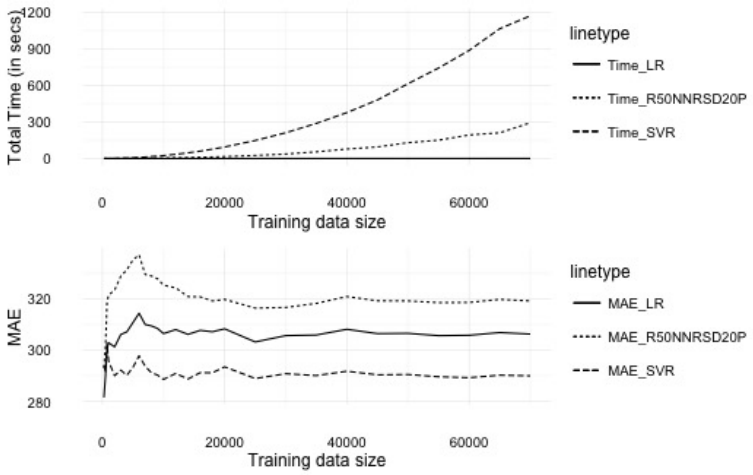


Figure 6: Time and MAE vs. different training data size for static experiment with travel time prediction when the drop-off location coordinates are not known

large datasets the computation time for SVR is high and hence it can be recommended for training data size less than 10000. For training data size greater than 10000, LR is the method of choice.

8.2 Computation Time Analysis- Travel Time Prediction when drop-off coordinates are available

In section 7.2.1, we have seen that the SVR method performs better in terms of predictive accuracy than the R25KNNRSD20P for most of the datasets. However, the SVR method requires comparatively higher computation time. In Table 10 below, we show the average CPU time taken for training and prediction of 25NNRSD (i.e. KNNRSD with $K = 25$), R25NNRSD20P and SVR methods on the NYC1 dataset and a relatively sparse PORTO dataset.

For the NYC1 dataset we observe that even with a window size of 15 minutes the average number of observations in the training data is quite large (8629). In this case the 25NNRSD method is quite competitive in terms of AMAE (being within 1.3% of the best i.e. SVR method) with its average total time being 78% less than that of the SVR method. Hence in such situations 25NNRSD can be a good choice. With larger window size say, 30 minutes or 1 hour we find that R25NNRSD20P becomes a competing a method of choice. Specifically when an 1 hour window is considered which has average training data size of 39879 the R25NNRSD20P has accuracy within 4% of the best i.e. 25NNRSD method while its computing time is 84.6% less than 25NNRSD method. The SVR method in this case has accuracy within 0.5% of the 25NNRSD method but the time taken by it is 576% more than that of 25NNRSD. But in the PORTO dataset where the training data size is small we find that SVR method has best predictive accuracy and the total time taken is also quite small. This is in line with our findings in Section 8.1.

8.3 Computation Time Analysis- Travel Time Prediction when drop-off coordinates are not available

In section 7.3.1, we have seen that between the R50NNRSD20P and the SVR methods, the SVR method performs better in terms of predictive accuracy than the R50NNRSD20P for most of the datasets. But we have discussed earlier that the SVR method is computationally expensive for larger datasets. In Table 11 below, we show the average CPU time taken for training and

Table 10: Time- Accuracy analysis for 25NNRSD, R25NNRSD20P, SVR, LR and ANN on NYC1 and PORTO datasets for different window sizes ($\mathcal{W}$) with $\lambda = 0.5$ for the travel time prediction problem when the drop-off coordinates are known

Dataset	$\mathcal{W}$	Average training data size*	Average Pred horizon size*	Method	Average Training Time (in Secs)	Average Testing Time (in Secs)	Average Total Time (in Secs)	AMAE
NYC1	15 min	8629	1450	25NNRSD	0	5.3038	5.3038	163.1
				R25NNRSD20P	0	1.0117	1.0117	181.15
				SVR	23.1674	0.9368	24.1043	161.03
				LR	0.0164	0.0074	0.0238	280.52
				ANN	0.0759	0.0011	0.0769	287.38
	30 min	20295	3421	25NNRSD	0	22.8227	22.8227	165.25
				R25NNRSD20P	0	3.7570	3.7570	177.82
				SVR	132.7734	4.9974	137.7708	165.77
				LR	0.0557	0.0043	0.0600	284.69
				ANN	0.1226	0.0012	0.1238	288.71
	1 hour	39879	6825	25NNRSD	0	70.3121	70.3121	175.78
				R25NNRSD20P	0	10.8071	10.8071	182.48
				SVR	459.0018	16.0120	475.0138	176.54
				LR	0.0505	0.0047	0.0552	290.4
				ANN	0.1406	0.0014	0.1420	292.06
PORTO	15 min	262	44	25NNRSD	0	0.0107	0.0107	272.98
				R25NNRSD20P	0	0.0058	0.0058	303.82
				SVR	0.0246	0.0027	0.0273	246.10
				LR	0.0049	0.0047	0.0096	321.67
				ANN	0.0141	0.0003	0.0144	321.75
	30 min	619	104	25NNRSD	0	0.0291	0.0291	268.01
				R25NNRSD20P	0	0.0211	0.0211	294.10
				SVR	0.0722	0.0032	0.0754	242.79
				LR	0.0079	0.0024	0.0103	327.01
				ANN	0.0166	0.0008	0.0174	311.73
	1 hour	1233	208	25NNRSD	0	0.0895	0.0895	265.38
				R25NNRSD20P	0	0.0437	0.0437	283.75
				SVR	0.1940	0.0122	0.2062	243.92
				LR	0.0068	0.0030	0.0098	331.56
				ANN	0.0256	0.0003	0.0259	334.98

* rounded up to the nearest integer

prediction of 50NNRSD (i.e. KNNRSD with $K = 50$), R50NNRSD20P, SVR, LR and ANN models along with their AMAE values for the NYC1 and PORTO datasets for different window sizes i.e. 15 min, 30 min and 1 hour.

Table 11: Time- Accuracy analysis for 50NNRSD, R50NNRSD20P, SVR, LR and ANN on NYC1 and PORTO datasets for different window sizes ($\mathcal{W}$) with $\lambda = 0.5$ for the travel time prediction problem when the drop-off coordinates are not known

Dataset	$\mathcal{W}$	Average training data size*	Average Pred horizon size*	Method	Average Training Time (in Secs)	Average Testing Time (in Secs)	Average Total Time (in Secs)	AMAE
NYC1	15 min	8629	1450	50NNRSD	0	5.2667	5.2667	324.59
				R50NNRSD20P	0	0.9849	0.9849	328.36
				SVR	24.1053	1.0549	25.1602	288.49
				LR	0.0150	0.0031	0.0181	304.8
				ANN	0.0767	0.0007	0.0774	308.9
	30 min	20295	3421	50NNRSD	0	22.8144	22.8144	327.47
				R50NNRSD20P	0	3.7393	3.7393	329.77
				SVR	128.8913	5.6280	134.5193	291.78
				LR	0.0419	0.0032	0.0451	307.92
				ANN	0.0859	0.0030	0.0889	310.76
	1 hour	39879	6825	50NNRSD	0	69.3269	69.3269	330.11
				R50NNRSD20P	0	10.7011	10.7011	331.58
				SVR	408.901	18.2975	427.1985	294.83
				LR	0.0462	0.0047	0.0509	310.97
				ANN	0.1592	0.0020	0.1612	312.22
PORTO	15 min	262	44	50NNRSD	0	0.0092	0.0092	327.23
				R50NNRSD20P	0	0.0097	0.0097	327.43
				SVR	0.0225	0.0034	0.0259	314.25
				LR	0.0033	0.0014	0.0047	330.52
				ANN	0.0029	0.0007	0.0036	330.35
	30 min	619	104	50NNRSD	0	0.0378	0.0378	328.67
				R50NNRSD20P	0	0.0286	0.0286	329.75
				SVR	0.0622	0.0042	0.0664	313.59
				LR	0.0083	0.0032	0.0115	329.73
				ANN	0.0095	0.0003	0.0098	329.91
	1 hour	1233	208	50NNRSD	0	0.0878	0.0878	331.80
				R50NNRSD20P	0	0.0310	0.0310	331.64
				SVR	0.1634	0.0084	0.1715	313.35
				LR	0.0071	0.0012	0.0083	331.56
				ANN	0.0114	0.0003	0.0117	334.98

* rounded up to the nearest integer

For the NYC1 dataset we observe that across different window sizes viz. 15 minutes, 30 minutes and 1 hour, the SVR method is consistently the best

performer in terms of prediction accuracy but the total time taken by it also increases with the increase training data size. With larger window size say, 30 minutes or 1 hour we find that LR method becomes a competing a method of choice. Specifically when an 1 hour window is considered which has average training data size of 39879 the LR has accuracy within 5.5% of the best i.e. the SVR method while its computing time is 99.9% less than that of the SVR method. But in the PORTO dataset where the training data size is small we find that SVR method has best predictive accuracy and the total time taken is also quite small. This is in accordance with our findings in Section 8.1.

9 Continuous Prediction and Continuous Updating of travel time for trajectory data

In this section, we discuss the performance of our proposed method RKNNRSD along with other methods viz. LR, ANN, SVR and an Ensemble method on a trajectory level dataset i.e. SFBLACK2 details of which is given in Section 6.2.2. The dataset consists of location coordinates i.e. latitude and longitude recorded every 4 seconds for each trip.

We intend to give continuous updates of the travel time from the time of pickup till drop-off. We assume that the drop-off coordinates are known since it is expected that the passenger would inform the driver about the drop-off location at the time of pickup. Two kinds of updates may be useful to the transport dispatch system and also to the passenger - the remaining time to the reach the destination and the total travel time. In the first case, we calculate the remaining travel time by considering the current latitude and longitude readings as the pickup point and using the algorithms mentioned in Section 7. In the second case, we compute the remaining time taken and then add the time elapsed since pick-up to it to get the total time taken.

We apply and compare the performance of R25NNRSD20P, LR, ANN, SVR and an Ensemble of R25NNRSD20P and SVR for both continuous prediction of remaining travel time and also the continuous updating of total travel time. As before we use a 1 hour Damped window model with $\lambda = 0.5$. For the remaining travel time problem we obtain root mean squared error

(RMSE) for each trip ID, L, as follows:

$$RMSE(L) = \sqrt{\frac{1}{n_L - 1} \sum_{i=1}^{n_L - 1} (t_{Pred,i} - t_{Actual,i})^2} \tag{17}$$

where n_L is the number of recordings for trip-id L, $t_{Pred,i}$ is the predicted remaining travel time at instant i and $t_{Actual,i}$ is the actual remaining travel time, $1 \leq i \leq n_L - 1$. The ECDF of the RMSE is defined as

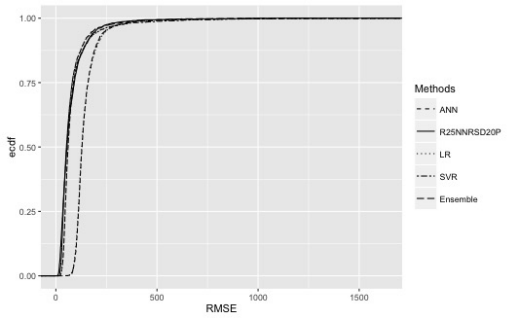
$$G_n(x) = \frac{\#\{RMSE_i \leq x\}}{n} \tag{18}$$

where n is the number of trip IDs, $RMSE_i$ is the RMSE of the i^{th} trip ID and $\#\{RMSE_i \leq x\}$ is the number of $RMSE_i$ that are less than or equal to x.

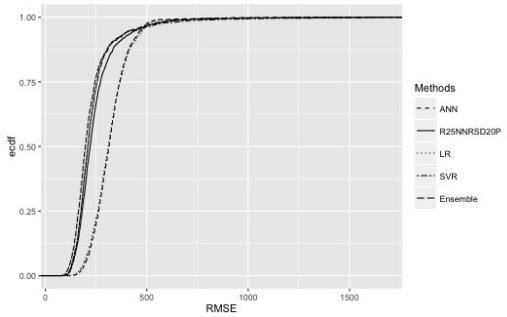
Similarly, for the continuous updating of total travel time problem we obtain RMSE for each trip ID, L, using equation 17 where $t_{Pred,i}$ denotes the predicted total travel time at instant i and $t_{Actual,i} = t_{Actual}$ is the actual total travel time, $1 \leq i \leq n_L - 1$.

The performance comparison is based on the RMSE obtained by using the different methods. We use the ECDF of the RMSEs for each method and check for stochastic dominance as discussed in Section 2.7.

Figure 7 (a) shows the ECDF plot for the RMSE of the five methods mentioned above for the continuous prediction of remaining travel time. We can see that the ECDF of the RMSE for LR and ANN methods are to the right of the other three. So, LR and ANN are stochastically dominant over the other three i.e. their performance is poorer than that of the other three. The ECDFs of R25NNRSD20P and SVR methods are closely placed together. In Figure 7 (b) which shows the ECDF plots for the RMSE for continuous updating of the total travel time, we find that the ECDFs of R25NNRSD20P and SVR methods are closely placed together with the ECDF of SVR method being slightly to the left of that of R25NNRSD20P. Thus the SVR method performs the best in terms of prediction accuracy for this problem. The ECDF of the Ensemble method crosses that of the SVR method in both of these cases indicating that it is a competing method when enough resources to run the ensemble are available.



(a) Continuous prediction of remaining travel time



(b) Continuous Updating of total travel time

Figure 7: Empirical CDF plots for the RMSE of the five different methods for continuous prediction of remaining travel time and continuous updating of total travel time on trajectory data

9.1 The Naive Method

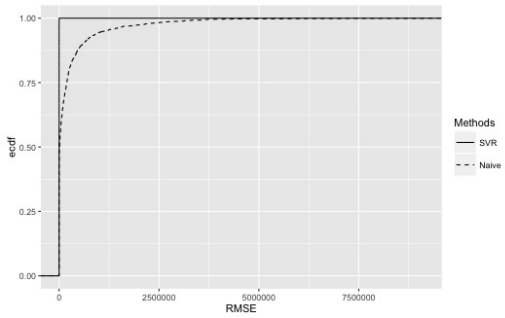
We have applied five different methods for continuous prediction of remaining travel time and for continuous updating of total travel time in the earlier section. We find that the SVR method is consistently a better performer in terms of prediction accuracy. In this section, we explore the dynamic updation of remaining travel time and total travel time at each points in a trajectory of a trip using the distance remaining to be covered and speed information. We will refer to this method as the *Naive method*. Some of the transport dispatch systems may possibly be using this kind of an approach and hence it is interesting to see how it performs vis-a-vis the SVR method.

We implement the Naive method on the SFBLACK2 dataset for both the continuous prediction of remaining travel time and the continuous updating of total travel time problems. We first compute the Geodesic distance (see Equation 11 in Section 2.8) from each point in the trajectory to the drop-off location coordinates to obtain the remaining distance at each point in the trajectory. The speed of the vehicle at a point is calculated by first computing the geodesic distance from the previous point to the current point and the time taken to cover the distance between these two points.

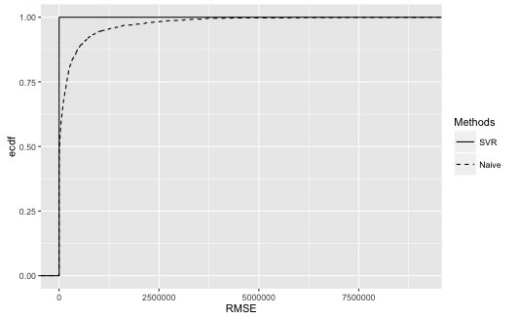
Suppose the previous location coordinates of the cab is at (ϕ_x, μ_x) and the same for the current position is (ϕ_y, μ_y) and suppose the geodesic distance between these two points is l. If it has taken time t to travel between these two points then we calculate the speed v of the cab at the current point as $v = l/t$. For the continuous prediction of remaining travel time problem, we divide the remaining distance by the speed to get the remaining travel time (in secs) at each points in a trajectory for a trip ID. Whereas, for the continuous updating of total travel time problem, we compute the remaining time taken and then add the time elapsed since pick-up to it to get the total time taken. Figure 8 (a) and (b) depicts the ECDF plots for the RMSEs of the Naive and the SVR methods for both of these cases and we see that the Naive method curve is to the right of that of the SVR method. Hence, the SVR method is a better performer in terms of prediction accuracy.

9.2 The Hybrid Method

In this section, we propose another new method for both the continuous prediction of remaining travel time and the continuous updating of total travel time problems by combining the SVR and the Naive methods. In



(a) Continuous prediction of remaining travel time

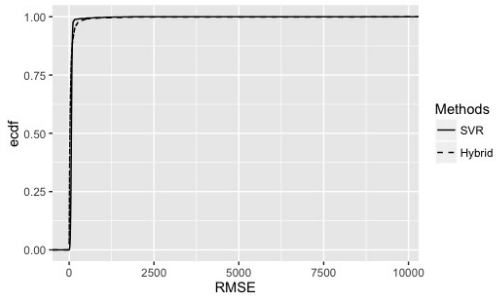


(b) Continuous Updating of total travel time

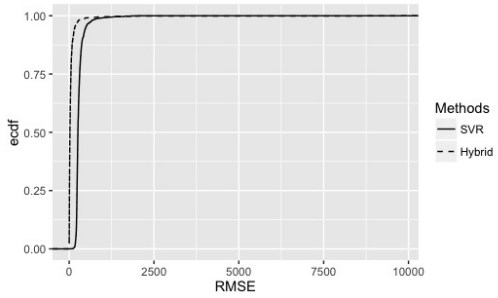
Figure 8: Empirical CDF plots for the RMSE of the Naive and the SVR methods for continuous prediction of remaining travel time and continuous updating of total travel time on trajectory data

section 9.1, we find that the SVR method has a better prediction accuracy than the Naive method. But if we investigate the predicted values for the remaining travel time of the SVR method for each trip IDs, we find that the absolute error (i.e. the absolute difference between the predicted and the actual remaining travel time) of the SVR method is more than that of the Naive method when the trip is nearing its end. We find this pattern consistently in almost all the trip IDs. So we propose a new method that combines both the SVR and the Naive methods and further improves the prediction accuracy of the SVR method. We will refer to this method as the *Hybrid method*. In the Hybrid method, we begin with the SVR method for predicting the remaining travel time (or updating the total travel time) from the start of a trip and then after a pre-specified part of the journey, the algorithm switches to the Naive method for predicting the remaining travel time (or updating the total travel time). To obtain the appropriate point of switching, we conduct a sensitivity analysis for the Hybrid method with different switching points (viz. 65%, 70%, 75%, 80%, 85% and 90%) of the total distance traveled for both the problems using the SFBLACK2 dataset. We find that the Hybrid method with the switching point at 85% of the total distance traveled for a trip is the best performer in terms of prediction accuracy and we refer to this as the Hybrid method for the remaining part of this paper.

We further compare the predictive accuracy of the Hybrid method with the SVR method in the last 15% of the journey for each trip. In figure 9 (a) and (b), we plot the ECDFs for the RMSEs of the Hybrid and the SVR methods for both the problems. We find that for continuous updating of the total travel time problem, the RMSE of the SVR method is stochastically dominant over that of the Hybrid method and hence, we conclude that the performance of the Hybrid method is better than that of the SVR method. However, for the continuous prediction of the remaining travel time, none of the two methods dominate over each other.



(a) Continuous prediction of remaining travel time



(b) Continuous Updating of total travel time

Figure 9: Empirical CDF plots for the RMSE of the Hybrid (with 85% switching point) and the SVR methods for continuous prediction of remaining travel time and continuous Updating of total travel time on the SFBLACK2 dataset for the last 15% of the total distance for each trip IDs

10 Conclusion

In this paper, we have examined the travel time prediction problem and have suggested the best algorithm to use in different situations: (a) When the drop-off location is known and the training data sizes are small to moderate the SVR method is the best choice considering both prediction accuracy and total computation time. However when the training data size becomes large the RKNNRSD becomes the best choice. (b) When the drop-off location is unknown then we find that the SVR method is the best choice when the training data size is small to moderate while for large training data size the LR method is a good choice. (c) When continuous prediction of remaining travel time and continuous updating of total travel time along the trajectory of a trip are considered we find that the SVR method has the best predictive accuracy. We also propose a new hybrid method which improves the prediction accuracy of the SVR method in the later part of a trip.

References

Aggarwal, C. C. (2006). *Data Streams: Models and Algorithms (Advances in Database Systems)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Babcock, B., Babu, S., Datar, M., Motwani, R., and Widom, J. (2002). Models and issues in data stream systems. In *Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '02, pages 1–16, New York, NY, USA. ACM.

Bai, C., Peng, Z.-R., Lu, Q.-C., and Sun, J. (2015). Dynamic bus travel time prediction models on road with multiple bus routes. *Intell. Neuroscience*, pages 63:63–63:63.

Barceló, J., Montero, L., Marqués, L., and Carmona, C. (2010). Travel time forecasting and dynamic origin-destination estimation for freeways based on bluetooth traffic monitoring. *Transportation Research Record: Journal of the Transportation Research Board*, 2175:19–27.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., New York, NY, USA.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pages 144–152, New York, NY, USA. ACM.

Breiman, L. (1996). Bagging predictors. *Mach. Learn.*, 24(421):123–140. Kluwer Academic Publishers, Hingham, MA, USA.

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32. Kluwer Academic Publishers, Hingham, MA, USA.

Büttcher, S., Clarke, C. L. A., and Cormack, G. V. (2010). *Information Retrieval: Implementing and Evaluating Search Engines*. The MIT Press. isbn:0262026511, 9780262026512.

Cao, F., Ester, M., Qian, W., and Zhou, A. (2006). Density-based clustering over an evolving data stream with noise. In *2006 SIAM Conference on Data Mining*, pages 328–339.

Chang, H., Tai, Y., and Hsu, J. (2010). Context-aware taxi demand hotspots prediction. *International Journal of Business Intelligence and Data Mining*, 5(1):3–18.

Chen, C. (2014). *Understanding social and community dynamics from taxi GPS data*. PhD thesis, Institut National des Telecommunications, Evry, France. [Online, Accessed on 10-Jan-2016].

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Mach. Learn.*, 20(3):273–297.

Cover, T. and Hart, P. (1967). Nearest neighbour pattern classification. *IEEE Transactions on Information Theory*, IT-13:21–27.

Davidson, R. (2008). *Stochastic dominance*. Palgrave Macmillan, UK, 2nd edition.

Economou, G., Pothos, V., and Ifantis, A. (2004). Geodesic distance and mst based image segmentation. In *In XII European Signal Processing Conference (EUSIPCO 2004)*, pages 941–944.

Ellis, B. (2014). *Real - Time Analytics: Techniques to analyze and visualize streaming data*. Wiley Publishing, 1st edition. isbn: 1118837916, 9781118837917.

Fisher, N. I., Lewis, T., and Embleton, B. J. J. (1993). *Statistical Analysis of Spherical Data*. Cambridge university press.

Fox, J. (2008). *Applied Regression Analysis and Generalized Linear Models*. SAGE Publications.

Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *journal of computer and system sciences*, 55(1):119–139.

Friedman, J. H., Baskett, F., and Shustek, L. J. (1975). An algorithm for finding nearest neighbours. *IEEE Trans on Computers*, C-24(10):1000–1006.

Fukunaga, K. and Narendra, P. M. (1975). A branch and bound algorithm for computing k nearest neighbours. *IEEE Trans. on Computers*, pages 750–753.

Gade, K. (2010). A non-singular horizontal position representation. *The Journal of Navigation*, 63:395–417.

Gama, J. (2010). *Knowledge Discovery from Data Streams*. Chapman & Hall/CRC, 1st edition.

Grolemund, G. and Wickham, H. (2011). Dates and times made easy with lubridate. *Journal of Statistical Software*, 40(3):1–25.

Gurmu, Z. and Fan, W. (2014). Artificial neural network travel time prediction model for buses using only gps data. *Journal of Public Transportation JPT*, 17(2):45–65.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning*. Springer series in statistics. Springer, 2nd edition. New York.

Henry (2011). Uber gps traces. Github, Retrieved from: `https://github.com/dima42/uber-gps-analysis/tree/master/gpsdata`. [Online, Accessed on 1-Sep-2015].

Herring, R., Hofleitner, A., Abbeel, P., and Bayen, A. (2010). Estimating arterial traffic conditions using sparse probe data. In *Proceedings of the International IEEE Conference on Intelligent Transportation Systems*, pages 929–936.

Hjort, N. L. and Claesken, G. (2003). Frequentist model average estimators. *Journal of the American Statistical Association*, 98(464):879–899.

Hofleitner, A., Herring, R., and Bayen, A. (2012). Arterial travel time forecast with streaming data: a hybrid approach of flow modeling and machine learning. *Transportation Research Part B*, 46(9):1097–1122.

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366.

James, G., Witten, D., Hastie, T., and Tibshirani, R. (2014). *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated.

Jammalamadaka, S. R. and Sengupta, A. (2001). *Topics in circular statistics*. World Scientific Publishing Co. Pte. Ltd. isbn: 9810237782.

Jeong, R. and Rilett, L. R. (2004). Bus arrival time prediction using artificial neural network model. In *Proceedings of the 7th International IEEE Conference on Intelligent Transportation Systems*, pages 988–993. IEEE.

Kim, B. S. and Park, S. B. (1986). A fast nearest neighbour finding algorithm based on the ordered partition. *IEEE Trans on PAMI*, PAMI-8(6):761–766.

Kotsiantis, S. B. and Pintelas, P. (2005). Selective averaging of regression models. *Annals of Mathematics, Computing & Teleinformatics*, 1(3):65–74.

Kreinovich, V. Y. (1991). Arbitrary nonlinearity is sufficient to represent all functions by neural networks: a theorem. *Neural Networks*, 4(3):381–383.

Kusner, M., Tyree, S., Weinberger, K. Q., and Agrawal, K. (2014). Stochastic neighbor compression. In Jebara, T. and Xing, E. P., editors, *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 622–630. JMLR Workshop and Conference Proceedings.

Lam, H. T., Diaz-Aviles, E., Pascale, A., Gkoufas, Y., and Chen, B. (2015). (blue) taxi destination and trip time prediction from partial trajectories. *CoRR*, abs/1509.05257.

Laskov, P., Gehl, C., Krüger, S., and Müller, K.-R. (2006). Incremental support vector learning: Analysis, implementation and applications. *Journal of Machine Learning Research*, 7:1909–1936.

Lee, W.-H., Tseng, S.-S., and Tsai, S.-H. (2009). A knowledge based real-time travel time prediction system for urban network. *Expert systems with Applications*, 36(3):4239–4247.

Levin, J. (2006). *Choice under Uncertainty*. Department of Economics, Stanford University.

Liu, X., Zhu, Y., Wang, Y., Forman, G., Ni, L. M., Fang, Y., and Li, M. (2012). Road recognition using coarse-grained vehicular traces. Technical report, HP Lab.

Luo, W., Tan, H., Chen, L., and Ni, L. M. (2013). Finding time period-based most frequent path in big trajectory data. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 713–724, New York, NY, USA. ACM.

Ma, S., Zheng, Y., and Wolfson, O. (2013). T-share: A large-scale dynamic taxi ridesharing service. *2013 29th IEEE International Conference on Data Engineering (ICDE 2013)*, pages 410–421.

Mardia, K. V. and Jupp, P. E. (2000). *Directional Statistics*. Wiley series in probability and statistics. Wiley, Chichester. Previous ed. published as: Statistics of directional data. London : Academic Press, 1972.

Masters, T. (1993). *Practical Neural Network Recipes in C++*. Academic Press Professional, Inc., San Diego, CA, USA.

Mendes-Moreira, J. a., Jorge, A. M., de Sousa, J. F., and Soares, C. (2012). Comparing state-of-the-art regression methods for long term travel time prediction. *Intell. Data Anal.*, 16(3):427–449.

Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., and Leisch, F. (2015). *e1071: Misc Functions of the Department of Statistics Probability Theory Group (Formerly: E1071) TU Wien*.

Miclet, L. and Dabouz, M. (1983). Approximative fast nearest neighbour recognition. *Pattern Recognition Letters*, 1:277–285.

Moreira-Matias, L., Gama, J., Ferreira, M., Mendes-Moreira, J., and Damas, L. (2013). Predicting taxi-passenger demand using streaming data. *IEEE Transactions on Intelligent Transportation Systems*, 14(3):1393–1402.

Moreira-Matias, L., Gama, J., Mendes-Moreira, J., Ferreira, M., and Damas, L. (2016). Time-evolving O-D matrix estimation using high-speed gps data streams. *Expert systems with Applications*, 44(C):275–288.

Murty, M. N. and Devi, V. S. (2011). *Pattern Recognition: An Algorithmic Approach*. Undergraduate topics in computer science. Springer, New York.

Muthukrishnan, S. (2003). Data streams: Algorithms and applications. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '03, pages 413–413, Philadelphia, PA, USA. Society for Industrial and Applied Mathematics.

Navot, A., Shpigelman, L., Tishby, N., and Vaadia, E. (2006). Nearest neighbor based feature selection for regression and its application to neural activity. In Weiss, Y., Schölkopf, B., and Platt, J. C., editors, *Advances in Neural Information Processing Systems 18 [Neural Information Processing Systems, NIPS 2005, December 5-8, 2005, Vancouver, British Columbia, Canada*, pages 996–1002. MIT Press.

NYC-TLC (2014). TLC trip record data. Available: `http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml`. [Online, Accessed on 17-Nov-2015].

Oldfield, T. and Hubbard, R. (1994). Analysis of c_α geometry in protein structures. *Proteins*, 18:324–337.

Papadopoulos, A. and Manolopoulos, Y. (2005). *Nearest Neighbor Search: A Database Perspective*. Series in Computer Science. Springer.

Patnaik, J., Chien, S., and Bladikas, A. (2004). Estimation of bus arrival times using apc data. *Journal of Public Transportation*, 7(1):1–20.

Phithakkitnukoon, S., Veloso, M., Bento, C., Biderman, A., and Ratti, C. (2010). Taxi-aware map: identifying and predicting vacant taxis in the city. *Ambient Intelligence*, 6439:86–95.

Putatunda, S. (2017). *Streaming Data: New Models and Methods with Applications in the Transportation Industry*. PhD thesis, Indian Institute of Management Ahmedabad.

R Core Team (2016). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Ratcliffe, J. G. (2006). *Foundations of hyperbolic manifolds*, volume 149 of *Graduate texts in mathematics*. Springer, New York, 2nd edition.

Ripley, B. D. (1996). *Pattern Recognition and Neural Networks*. Cambridge University Press.

Rosenblatt, F. (1962). *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan Books, Washington.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation. In Rumelhart, D. E., McClelland, J. L., and PDP Research Group, C., editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1*, pages 318–362. MIT Press, Cambridge, MA, USA.

Schapire, R. E. (1990). The strength of weak learnability. *Machine Learning*, 5(2):197–227.

Shakhnarovich, G., Darrell, T., and Indyk, P. (2005). *Nearest-Neighbor Methods in Learning and Vision*. Neural Information Processing Series. The MIT Press, Cambridge, Massachusetts, USA.

Smola, A. J. and Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222.

Tiesyte, D. and Jensen, C. S. (2008). Similarity-based prediction of travel times for vehicles traveling on known routes. In *GIS*, pages 14:1–14:10.

Tsagris, M. and Athineou, G. (2016). *Directional: Directional Statistics*. R package version 1.9.

Tsang, I. W., Kwok, J. T., and Cheung, P.-M. (2005). Core vector machines: Fast svm training on very large data sets. *Journal of Machine Learning Research*, 6:363–392.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., New York, NY, USA.

Veloso, M., Phithakkitnukoon, S., and Bento, C. (2011). Urban mobility study using taxi traces. In *Proceedings of the International Workshop on Trajectory Data Mining and Analysis*, TDMA '11, pages 23–30, New York, NY, USA. ACM.

Venables, W. N. and Ripley, B. D. (2002). *Modern Applied Statistics with S*. Springer.

Vincenty, T. (1975). Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations. *Survey Review. XXIII*, 176:88–93.

Wang, Y., Zheng, Y., and Xue, Y. (2014). Travel time estimation of a path using sparse trajectories. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 25–34, New York, NY, USA. ACM.

Wasserman, L. (2010). *All of Statistics: A Concise Course in Statistical Inference*. Springer Publishing Company, Incorporated.

Weinberger, K. Q. and Saul, L. K. (2009). Distance metric learning for large margin nearest neighbor classication. *Journal of Machine Learning Research*, 10:207–244.

Wibisono, A., Jatmiko, W., Wisesa, H. A., Hardjono, B., and Mursanto, P. (2016). Traffic big data prediction and visualization using fast incremental model trees-drift detection (fimt-dd). *Knowledge-Based Systems*, 93(C):33–46.

Wickham, H. (2009). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York.

Widrow, B. and Hoff, M. E. (1988). Adaptive switching circuits. In Anderson, J. A. and Rosenfeld, E., editors, *Neurocomputing: Foundations of Research*, pages 123–134. MIT Press, Cambridge, MA, USA.

Wolpert, D. (1992). Stacked generalization. *Neural Networks*, 5:241–259.

Wu, C.-H., Ho, J.-M., and Lee, D. T. (2004). Travel time prediction with support vector regression. *IEEE Transactions on Intelligent Transportation Systems*, 5(4):276–281.

Xie, X., Zheng, Y., Zhang, L., and Yuan, N. J. (2013). T-finder: A recommender system for finding passengers and vacant taxis. *IEEE Transactions on Knowledge & Data Engineering*, 25:2390–2403.

Zhang, B. and Srihari, S. N. (2004). Fast k-nearest neighbour classification using cluster-based trees. *IEEE Trans. PAMI*, 26(4):525–528.