Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Cybernetics

BACHELOR PROJECT ASSIGNMENT

Student: David M i l e c

Study programme: Open Informatics

Specialisation: Computer and Information Science

Title of Bachelor Project: Identification of Cycling Segments in Raw GPS Trajectories

Guidelines:

1. Survey existing methods and approaches for identifying the mode of transport from GPS trajectory data.
2. Formalize the problem of the identification of cycling segments from the GPS trajectory.
3. Select and implement suitable methods to solve the problem defined in step 2 with the emphasis on practical applicability.
4. Evaluate the implemented methods on a selected real-world GPS trajectory dataset.

Bibliography/Sources:

[1] Zheng, Yu, "Trajectory Data Mining: An Overview", ACM Transaction on Intelligent Systems and Technology, 2015
[2] Zheng, Yu, et al. "Understanding mobility based on GPS data." Proceedings of the 10th international conference on Ubiquitous computing. ACM, 2008.
[3] Zheng, Yu, et al. "Learning transportation mode from raw gps data for geographic applications on the web." Proceedings of the 17th international conference on World Wide Web. ACM, 2008.

Bachelor Project Supervisor: Ing. Pavol Žilecký

Valid until: the end of the summer semester of academic year 2017/2018

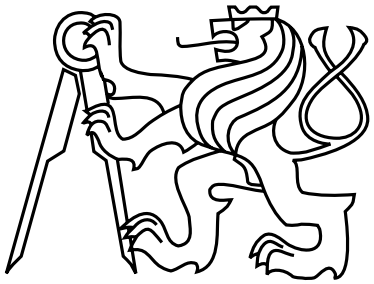
L.S.

prof. Dr. Ing. Jan Kybic
Head of Department

prof. Ing. Pavel Ripka, CSc.
Dean

Prague, January 6, 2017

Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Cybernetics



Bachelor's Project

Identification of Cycling Segments in Raw GPS trajectories

David Milec

Supervisor: Ing. Pavol Žilecký

Study Programme: Open Informatics

Field of Study: Computer and Information Science

May 26, 2017

Acknowledgements

At first, I would like to thank my supervisor Ing. Pavol Žilecký for his professional guidance and support he has provided. Also, I wish to thank Jan Mrkos and David Svoboda for their valuable feedback. Last, but not least, I would like to thank my parents and Anička Vlasáková for their support and encouragement.

Author statement for undergraduate thesis

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, date May 26, 2017 .....

Abstract

Recent advances in technology have generated extensive amount of spatial trajectories which contain a lot of information about movement. In my bachelor thesis I propose a method for classifying user's movement into two modes using GPS trajectory data. I focused on distinguishing between trajectories recorded while riding a bicycle and trajectories recorded while not riding a bicycle. I define this problem as a trajectory bicycle classification. My thesis also maps the present knowledge in data mining field focused on trajectory classification problem. Accordingly to state of the art I used methods as stay point segmentation for partitioning of a trajectory into single modal segments. For classification I used Random forest, SVM and Neural network models. From these methods Random forest shows the best results with accuracy of 90,5% when testing on perfectly segmented data. However, it was outperformed by SVM on data segmented with method I used where SVM had the best result with accuracy of 75,9%. To test the scalability of used method I perform evaluation on unlabeled dataset of bike rides, where SVM model performed the best. I also compared method I used with other methods and they perform better. However, I tested different filters and their combinations. I also evaluated segmentation performance based on different thresholds which is something not mentioned in available literary sources.

Keywords: trajectory classification, GPS, bicycle, SVM, Neural network, Random forest

Abstrakt

Pokrok v technologii umožnil vytváření velkého množství prostorových trajektorií, které obsahují mnoho informací o pohybu. Ve své bakalářské práci navrhuji metodu, která klasifikuje uživatelův pohyb do dvou modů s použitím GPS trajektorií. Soustředil jsem se na rozlišení mezi trajektoriemi nahranými na kole a nahranými jinde než na kole a definoval jsem problém jako classifikaci cyklistických trajektorií. Dále práce mapuje aktuální situaci v oboru data miningu s důrazem na klasifikační problémy. Použil jsem segmentaci na základě stay pointů pro rozdělení trajektorií na jednotlivé způsoby dopravy. Pro klasifikaci jsem použil klasifikátory Náhodný les, SVM a Neuronovou síť. Z těchto klasifikátorů měl nejlepší výsledky Náhodný les pro ručně segmentovaná data dosahující přesnosti 90,5 %, ale jeho výkon byl horší pro data segmentovaná mojí segmentací. Na techto datech měl nejlepší výsledek SVM s přesností 75,9 %. Abych otestoval škálovatelnost metody, otestoval jsem jí na datasetu cyklistických trajektorií bez označení, kde SVM měl nejlepší výsledek. Porovnal jsem metodu s ostatními metodami a zjistil, že některé mají vyšší přesnost než mnou použitá metoda. Já jsem ale otestoval různé typy filtrů a ukázal, jak se liší kvalita segmentace na základě různých prahů, což je něco, co jsem v žádném jiném článku nenašel.

Klíčová slova: klasifikace trajektorií, GPS, kolo, SVM, Neuronová síť, Náhodný les

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The advances in technology have generated extensive amount of spatial trajectories. A spatial trajectory is a trace generated by a moving object in space. This trajectory is usually represented by a series of chronologically ordered points. Each point consists of a coordinate and a time stamp.

Trajectories can be different and can contain a vast amount of useful informations. These informations must be somehow extracted from them. This is a goal of a trajectory data mining. For example from a mouse movement trajectory I could create a model identifying what is user doing on his computer. Another example can be from life of animals where trajectories can be used to determine where to build bridges over highways or how migration routes change over time.

Subfield of a trajectory data mining is a trajectory classification. A goal of the trajectory classification is to classify a trajectory into different modes of transportation. First step would be to classify a trajectory as generated by stationary or moving object. A more complex problem is to classify a trajectory into modes like car, bike, walk or run. With the information about user's modes of transportation advertisement can be targeted better e.g. gas station recommendations for car users or sport shops for runners. Also some helpful information about closures based on his mode of transportation can be send. Those informations can also be used to show where new bus lines are needed and where it would be wise to build new transportation infrastructure.

There are applications that can be used for a bicycle trip planning and also collect bicycle trajectories. One of those applications is Urbancycler[1] where cyclist can upload trajectories and earn badges and rewards. To prevent upload of trajectories recorded in car or while walking, method that can detect trajectories recorded on bicycle is needed. To create such method I use GPS trajectories recorded by people. Next use can be creating a bicycle profile from GPS trajectories in one city and using that information to match bicycle infrastructure of the city to the needs of people.

In this work I cover research done in field of trajectory classification in Chapter 2. In Chapter 3 I formally specify trajectory classification problem as supervised learning problem and define used terms. Next I cover my approach to the problem and all steps needed in

[1] http://urbancyclers.com/app.php

Chapter 4. In Chapter 5 I show how my method is implemented and in Chapter 6 I evaluate my method and show results.

Chapter 2

Related Work

2.1 Machine learning

Trajectory classification uses machine learning which is a computer science field that lets the computer learn without being specifically programed [9]. It evolved from study of pattern recognition and artificial intelligence. Machine learning constructs algorithms that can learn from data and make predictions on data.

There are two types of learning algorithms, supervised learning and unsupervised learning. Unsupervised learning tries to infer a function from unlabeled data to describe its unknown structure. I will use supervised learning, which is type of learning that uses input data labeled with desired output and goal of the algorithm is to find general rule mapping inputs to outputs.

Dataset is needed in a supervised machine learning. It is set of entries which will be used as inputs and every entry have its label, which will be used as desired output. To use each entry I need some significant values from it. Those values are called features and on those features algorithm learns.

2.2 Solutions used

In trajectory classification a wide range of methods have been used [5, 10, 19, 18]. Those methods even use different sensors. First sensor used is WiFi receiver that measures only strength of received signal. Next GSM mobile signal sensor also measures strength of signal but its range is much bigger. Last sensor employed is GPS receiver and it is most common nowadays.

2.3 Solutions based on sensor data

In this section I mention methods that use WiFi and GSM signal.

First article [5] tries to infer location of the user and whether the user is in motion or not. For this goal authors use WiFi receivers. This method uses WiFi signal strengths from different transmitters and applies a Hidden Markov model. This system computes user's

location with median error of 1,53m and can infer whether the user is in motion or not with accuracy of 87%. Disadvantage of this method is requirement for calibration for exact building and WiFi transmitters which prevents it from being easily used.

Next paper uses an approach [12] that tries to infer user's complex goals in indoor environment from WiFi signals using Dynamic Bayesian networks. Inferring people goals when moving could be used for automation. E.g. powering up of electronic devices when incoming person is detected and monitoring of people without cameras could be possible. But this method also has some disadvantages. First, on site calibration is required. Another inconvenience is that firstly the computing of a location from WiFi signal is needed and only after that the goal inferring method can be employed. Thus using trajectories with location would be much easier and that is why GPS inference dominates WiFi methods. However, it can not be used indoor.

The article [10] aims to classify moving or stationary user and then to distinguish moving between driving and walking. Sensor used is a cell phone showing signal strength from different GSM cells and also cell's IDs. From this, seven different features are extracted and used in two stage classification. First to classify moving or stationary and after that to distinguish moving between driving and walking using a boosted logistic classifier. Main advantage of this method is that only a mobile phone is needed. The method also computes approximate number of steps with accuracy comparable to commercial pedometers. Main reason for this method was low GPS coverage. However, nowadays GPS coverage improved a lot and is comparable to GSM coverage. First disadvantage of this method is that it is not as accurate as GPS inference. And second disadvantage lies in taking into account cell densities in metropolitan area where it was tested and created therefore it would not work in different areas.

2.4 Solutions based on GPS data

In this section I mention methods that use GPS trajectories for the classification.

In [19] authors attempt to classify a taxi status from its GPS trajectory as occupied or unoccupied. This method used density based clustering algorithm to detect the parking places and uses parking places as points where status of taxi can change. It also uses minimum bounding ratio to road segment from cluster to distinguish a parking lot and a traffic jam. It also uses a map matching which is algorithm that tries to project each point of trajectory into corresponding road segment. Finally, it uses a Hidden Markov model to make the inference. Disadvantage of this method is that it uses road graph and map matching algorithm so if road graph does not exist for the area it cannot be employed. This method proposes combination of Decision tree and Hidden Markov model. It is better than simple models and advantage of this method is that it can be used in real life taxi monitoring.

Authors in [6] use data to infer the mode of transportation and user's current goal which is target location of the user. It also predicts his places of interest like a home and work places. Using all of this the method can track and predict user's movement even with noise or loss of a GPS signal. Inference mode used a Dynamical Bayesian network. This method not only focuses on inferring mode of transportation but also user's goals which is something not much methods does. Method also suffers from needing road graph. Last described

disadvantage is that goals are detected as GPS signal loss thus goals that are outdoors or generally somewhere with good GPS coverage will not be detected. But advantage is that the method is less prone to noise in GPS signal or even its loss because of movement goal tracking and map matching.

Other methods use GPS data to infer the mode of transportation using variety of techniques. In [8] Bayesian filtering is used to estimate a state from GPS data creating Gaussian mixtures for different mode of transportation and using EM algorithm to find optimal model parameters. This method also predicts future user's movement with accuracy of 50% for five city blocks. An accuracy for more blocks drops to 30% for twelve city blocks. It also requires a road graph.

In [15] authors experimented with different type of segmentation. Segmentation is process which creates shorter trajectories from a trajectory. Three types of segmentation are done, based on time window, distance window and stay points. Conclusion is that a stay points segmentation outperforms both time based segmentation and distance based segmentation. Time based segmentation and distance based segmentation was done with multiple different lengths and time intervals. Stay point segmentation uses idea that for changing a mode of transportation a person must stop so stay points should segment trajectory perfectly. Conditional random field is proposed as a classifier but it falls behind SVM and Decision tree. Thus, in final method, segmentation by stay points is used and extracted features are learned by Decision tree to do the inference afterwards. After inference post processing is employed which takes into account probability of transitions between two modes to further improve accuracy.

Method proposed in [14] is also mode classification method but in this method authors experiment with using even more features than other methods. Apart from features like speed, distance and acceleration they also extract heading change rate, stop rate and velocity change rate. After that a graph based on change points is built with probabilities of each mode on edges of the graph. A graph post processing is employed to further increase accuracy. They also compare extracted features and show that stop rate is most discriminative feature when used alone. Also these enhanced features furthermore improve classification accuracy which implies that there is almost no correlation between basic and enhanced features. This method trades higher accuracy achieved by more features for time and computational complexity.

Method [18] uses GPS data and tries to infer walking, cycling, driving, bus and subway. For this purpose it uses a stay point segmentation and basic features which are average speed, distance, average acceleration, heading change rate. Authors also tried to use multiple different classifiers from which their own custom classifier based on deep learning is better than others and, apart from that, a Random forest is better than a SVM, a Decision tree and a Bayesian network. This method uses road network and as features bus stop closeness and subway stop closeness are added to standard features to improve accuracy. Disadvantage of using road graph and extracting more features is rewarded with the best accuracy amongst all other methods I have found in this field of work.

Testing multiple approaches leads always to efficient results and this could be find in [11] because authors employ also GIS apart from inferring modes of transportation from GPS data. They also try to use extracted features with different methods such as a Naive Bayes, a Bayesian network, a Decision tree and a Multilayer perceptron from which the Decision

Method	Accuracy	Year	Article	Notes
KNN (11) with PCA	86,00%	2015	[7]	
Dynamic Bayes net	84,00%	2003	[8]	
Naive Bayes	91,60%	2011	[11]	with GIS
Bayesian network	92,50%	2011	[11]	with GIS
Decision tree	92,20%	2011	[11]	with GIS
Random forest	93,70%	2011	[11]	with GIS
Multilayer perceptron	83,30%	2011	[11]	with GIS
Bayesian network	91,16%	2016	[18]	
Decision tree	90,65%	2016	[18]	
Random forest	92,45%	2016	[18]	
SVM	91,67%	2016	[18]	
Based on deep learning	93,32%	2016	[18]	
Moving window SVM	88,00%	2012	[1]	
Deep network	67,90%	2016	[3]	

Table 2.1: Methods overview showing accuracy, year of publication and source.

tree outperforms other methods. This article interestingly compares different methods and also it is first article using not only bus and subway stops but takes into account features that are proximity to rail lines and also a real time location of buses to increase accuracy. Disadvantage lies in necessity of having GIS information to determine real time bus locations.

Other method [7] in this field uses a KNN. As almost any other methods it starts by segmentation by change points, extracting features and after that is uses PCA and a KNN-11 to infer transportation modes. This approach focuses on extreme generality so it can be used on GPS data of different disaggregation level with almost the same accuracy. Employing PCA with KNN allows to drop some restrictions on assumptions about trip structures resulting in very general method.

The last method [3] is different from any other previous method because it uses deep networks. Authors want to use the strength of convolution networks, which lays in learning from images. Therefore, for each trajectory, image is created and network extract features from it. I mentioned this method to show that approaches can be surprisingly different and that methods does not have to be similar. This method is also almost only modern method that does not use any information from outside like road graph resulting in pretty poor accuracy of 67,9%. Also there is no need for segmentation because deep network creates image of trajectory classification corresponding which itself segments the trajectory.

In Table 2.1 is overview of some methods used with their accuracy and year they were published. As articles show by using multiple common methods wide range of them can be used on the task with not so big difference in accuracy. But it can also be seen that random forest is at the top in both articles only outperformed by custom classifier based on deep learning in [18]. An article [18] and [3] uses the Geolife dataset and others use data they gathered by themselves.

Chapter 3

Problem Specification

In this section I specify *bicycle trajectory classification* problem as supervised learning problem on set of trajectories. In order to do that I need to specify few terms first.

3.1 Trajectory

GPS trajectory T is represented by ordered set of points p_i:

$$T = (p_1, p_2, ..., p_n), \quad p_i = (x_i, y_i, t_i)$$

where (x_i, y_i) are coordinates in a plane coordinate and t_i is timestamp. Trajectory can have label l that indicates whether trajectory was recorded while riding a bicycle or not. Label can have one of the values $\{bike, not\ bike\}$.

3.2 Feature extraction

Furthermore, I need to extract features from each trajectory. Next I have to choose which features to use. Each feature is a real number f_j derived from trajectory T for classification purposes using extraction function $g_j(T) \to f_j$.

3.3 Classifier training

Next step is training of the classifier. Process of classification is displayed in Figure 3.1. First, classifier C is provided with a set of trajectories $S = (T_0, \ldots, T_n)$ and a set of corresponding labels $L = (l_0, \ldots, l_n)$ where n is number of trajectories in set S. From each trajectory $T_i \in S$ vector of features $F = (f_i^0, \ldots, f_i^k)$, where k is number of features. Classifier C was trained using those feature vectors.

3.4 Segmentation

Segmentation is required for classification because trajectory can contain multiple modes of transportation. It is partitioning $P(T) \rightarrow U$ that takes trajectory T and creates set of subtrajectories $U = (T_0, \ldots, T_l)$ such that $T_i \cap T_j = \emptyset \; for \; i \neq j, i, j \in 0, \ldots, l$.

3.5 Prediction

Last step is prediction of labels. Trained classifier C is provided with a set of trajectories $S = (T_0, \ldots, T_n)$ where n is number of trajectories. Using segmentation new set of segmented trajectories is created. $S^{sgm} = (T_0^{sgm}, \ldots, T_m^{sgm})$ where m is number of segmented trajectories. From each segmented trajectory $T_i \in S^{sgm}$ vector of features $F = (f_i^0, \ldots, f_i^k)$, where k is number of features. Trained classifier then generates a set of predicted class labels $L = (l_0, \ldots, l_m)$.

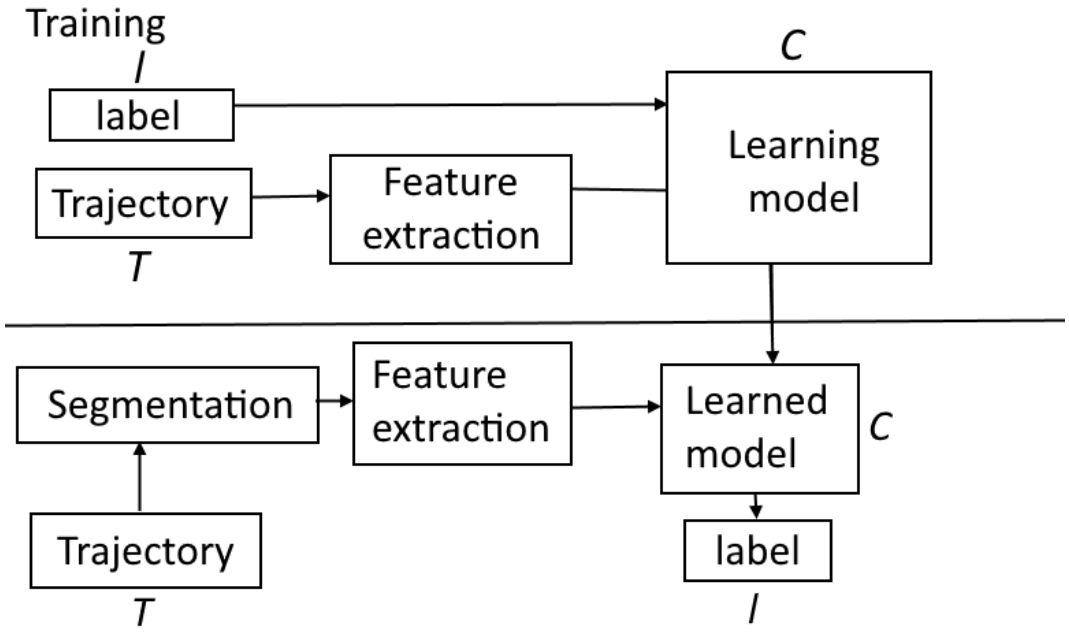


Figure 3.1: Diagram of supervised classification problem

Chapter 4

Solution Approach

This chapter shows how *bicycle trajectory classification* problem is solved and describes all steps required to do so.

Every method of *bicycle trajectory classification* consists of four vital steps shown in Figure 4.1. Skipping any of these steps can lead to significant reduction in resulting accuracy. First step described in Section Preprocessing is filtering of data to minimize noise and detect outliers. Second step is segmentation described in Section Segmentation which tries to partition trajectories into segments where is only bicycle or not bicycle. Third step in Section Feature extraction is extraction of important features for classification. This is the most computationally expensive part because of large amount of data being processed. Last part is choosing and training classifier in Section Classification model.

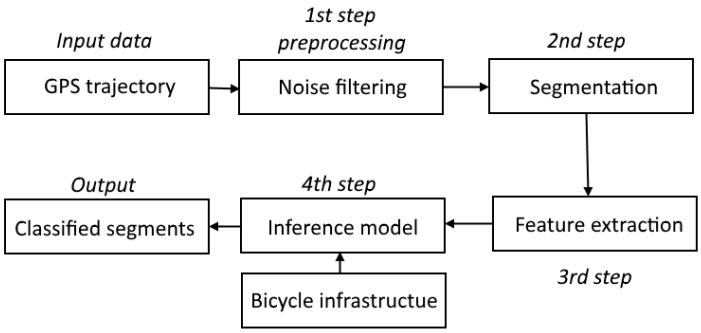


Figure 4.1: Model

4.1 Preprocessing

First step in a preprocessing is modifying data for use that I need so that it can be loaded and used by segmentation and classification.

4.1.1 Noise filtering

4.1.1.1 Introduction

GPS trajectories are never perfectly accurate, due to sensor noise. This is displayed in Figure 4.2. Noise is present when working with almost any sensors. Occasional poor GPS signal mostly in a big cities can create extreme outliers. Sometimes the noise is acceptable and recorded point difference from the point's original location is only few meters. In other scenarios with really poor signal the point noise can be more that hundred meters which can be problem for deriving useful information from it, e.g. speed of moving object. To minimize noise I need to filter noise points from the trajectory. [13] Next I will talk about filters commonly used in trajectory filtering.

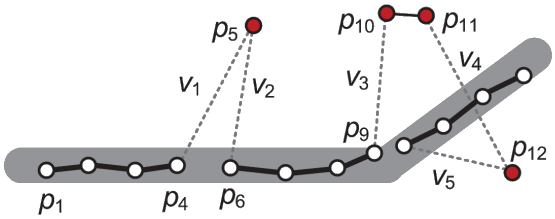


Figure 4.2: Trajectory noise

4.1.1.2 Filters

Mean and Median filter

In Mean and Median filter approximation of a true point p position is calculated as mean or median of point p and points that come before p and after p in time. How much points will be used is determined by size of sliding window that is used. A bigger sliding window is required when dealing with consecutive noise points but results in bigger error between approximation and point's true position. A median filter is more robust and can handle extreme errors better because in a mean filter one extreme error can move right points around him far from true position. When dealing with trajectories where sampling rate is very low, mean and median filters are not a good choice.

Kalman and Particle filter

Kalman and particle filters model both measurement noise and dynamics of the trajectory. Kalman filter assumes linear model and Gaussian noise and the particle filter drops those assumptions for a more general algorithm.

Particle filtering starts by generating P particles from the initial distribution. The second step is "importance sampling," which uses the model to simulate how the particles change

over one timestamp. Third step computes weight for all particles. Larger weights correspond to particles that are better supported by the measurement. Weight are then normalized to sum up to one. Then new set of particles is selected proportional to weights and finally a weighted sum is computed to get the point's approximate position.

The Kalman and Particle filters are harder to implement than other mentioned filters and depend very much on the measurement of the initial location. When first point in trajectory is noisy, the effectiveness of filter can drop significantly. [13]

Heuristic based filter

All previously mentioned filters replace the measurement with some approximation of point's true position. Heuristic based filter is different. It employs outlier detection using speed. If speed to some point is over some set threshold this point is regarded as outlier and therefore removed from the trajectory. This filter can filter very well extreme noise points but setting the threshold is based on heuristics.

4.2 Segmentation

Next step in *bicycle trajectory classification* is segmentation. It consist of cutting the trajectory in smaller parts preferably so that the part would correspond to only one mode of transportation.

First the trajectory is cut into smaller trips by time or distance. Next I cut the trajectory between points where time difference is over one minute and after that I can segment the remaining trajectory that can still contain multiple modes of transportation.

To do this I employed segmentation based on stay point detection. Stay point shown in Figure 4.3 is region bigger than distance threshold D_t where user stayed for time bigger than time threshold T_t. In trajectory it is set of consecutive points $T = (p_i, \ldots, p_j)$ and $\forall t : i < t < j, Distance(p_i, p_k) < D_t, Distance(p_i, p_j) > D_t \; and \; Time(p_i, p_j) > T_t$. Stay point divides trajectory into two non overlapping segments.

Based on [15] a stay point segmentation outperformed other methods as time window segmentation and distance window segmentation.

I made method that detects stay points based on distance and speed between individual points and tested multiple distance and speed thresholds. However, this did not work. Later I added feature that stay point begins only after multiple consecutive points satisfy the threshold but it was even worse. New stay point detection is based on checking whether distance between points that is over some threshold is travelled in time over some threshold.

4.3 Feature extraction

From segmented trajectories I need to extract features that I use to train my model. More features means better accuracy but also higher time complexity. In formulas used in feature extraction n is the length of trajectory and p_0 is starting point of trajectory and p_n is ending point of the trajectory. Features that I extracted are:

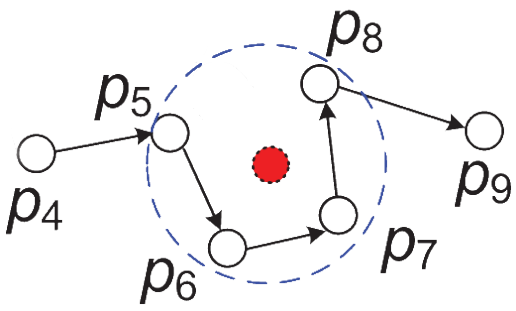


Figure 4.3: Stay point

4.3.1 Distance

Distance of recorded trajectory can help identify bicycle because it is not possible to cover large distances on bicycle in single trajectory. I compute a distance simply as a sum of distances between neighbouring points in a trajectory.

$$Distance = \sum_{i=0}^{n-1} Distance(p_i, p_{i+1})$$

4.3.2 Time

Time of a trip is also a feature that can determine if bicycle is chosen. Of course bicycle trip for multiple hours is possible but when using bicycle as transportation device time of trajectory will be smaller. I compute time as difference between a time at the start and a time at the end of the trip.

$$Time = Time(p_0, p_n)$$

4.3.3 Average speed

An average speed is also a great feature to distinguish bicycle. It is true that a car and a bus can have a small average speed because of traffic jams and bus stops but despite that it will be mostly higher than speed of a bicycle. An average speed is computed from a distance and a time of a trajectory.

$$Average\ speed = \frac{Distance}{Time}$$

4.3.4 Maximal speed

An another feature that can exclude bicycle because its maximal speed is very limited. There is one big disadvantage of this feature though. It can be very prone to noisy data so preprocessing is very much advised when using this feature. Maximal speed is computed as maximum of partial speeds between two consecutive points.

$$p_i.v = \frac{Distance(p_i, p_i + 1)}{Time(p_i, p_i + 1)}, \quad Maximal\ speed = Max(p_i.v), \forall i : 0 < i < n - 1$$

4.3.5 Average acceleration and deceleration

Big objects accelerate much slower than small and light objects thus an acceleration is another feature used to our advantage. A bicycle can accelerate much faster than than a train or bus. An acceleration is computed as sum of partial acceleration divided by their count.

$$p_i.t = Time(p_i, p_{i+1}), \quad p_i.c = \frac{p_{i+1}.v - p_i.v}{p_{i+1}.t - p_i.t}$$

$$p_i.c+ = \begin{cases} p_i.c & \text{if } p_i.c >= 0 \\ 0 & \text{if } p_i.c < 0 \end{cases} \quad p_i.c- = \begin{cases} 0 & \text{if } p_i.c >= 0 \\ p_i.c & \text{if } p_i.c < 0 \end{cases}$$

$$Acceleration = \frac{\sum_{i=0}^{n-1} p_i.c+}{count(p_i.c+)}, \quad Deceleration = \frac{\sum_{i=0}^{n-1} p_i.c-}{count(p_i.c+)}$$

4.3.6 Altitude changes upward and downward

Feature that can be helpful to discriminate walking and running from bicycle because on foot the altitude changes can be much more extreme than on bicycle. On the other hand it can also discriminate bicycle from other type of transports because on bicycle changes of altitude can be bigger than in car, bus or even train. It is computed as change in altitude between two nodes divided by distance between them.

$$p_i.a = Altitude(p_i, p_{i+1}), \quad p_i.a+ = \begin{cases} p_i.a & \text{if } p_i.a >= 0 \\ 0 & \text{if } p_i.a < 0 \end{cases} \quad p_i.a- = \begin{cases} 0 & \text{if } p_i.a >= 0 \\ p_i.a & \text{if } p_i.a < 0 \end{cases}$$

$$p_i.d = Distance(p_i, p_{i+1}), \quad p_i.d+ = \begin{cases} p_i.d & \text{if } p_i.a >= 0 \\ 0 & \text{if } p_i.a < 0 \end{cases} \quad p_i.d- = \begin{cases} 0 & \text{if } p_i.a >= 0 \\ p_i.d & \text{if } p_i.a < 0 \end{cases}$$

$$Altitude\ change\ up = \frac{\sum_{i=0}^{n-1} p_i.a+}{\sum_{i=0}^{n-1} p_i.d+}, \quad Altitude\ change\ down = \frac{\sum_{i=0}^{n-1} p_i.a-}{\sum_{i=0}^{n-1} p_i.d-}$$

4.3.7 Change of heading rate

Change of heading rate represent how fast the traveling object turned during the trajectory. It can be used again for distinguishing between big heavy transportation vehicles and bicycle. It is computed as average angle between three consecutive points.

$$a = Distance(p_{i-1}, p_i), \quad b = Distance(p_i, p_{i+1}), \quad c = Distance(p_{i-1}, p_{i+1})$$

$$p_i.h = \arccos((a^2 + b^2 - c^2)/(2ab)), \quad Change\ of\ heading\ rate = \frac{\sum_{i=1}^{n-1} p_i.h}{n-2}$$

4.3.8 Proximity of bicycle infrastructure

Last feature that I used is Proximity of bicycle infrastructure. In every point of the trajectory a distance to closest bicycle infrastructure is computed and feature is average over all points in trajectory. I believe that this feature should help greatly with classification of bicycle parts of trajectory because bicycle trajectories will surely be much closer on average to bicycle infrastructure.

$$p_i.b = BikeIntrastructureDistance(p_i.b), \quad Proximity = \frac{\sum_{i=0}^{n} p_i.b}{n}$$

4.4 Classification model

For classification model I chose three classifiers. Random forest because for most cases as shown in Chapter 2 it outperformed other learning algorithms. I will also try SVM because it is known to perform well in two-class classification problems. Last classifier is basic Neural network because it does not tend to overfit that much compared to random forest which can make it better when working with data gathered in different places.

4.4.1 Random forest

A random forest is an estimator that uses multiple decision trees. A decision tree is classifier that in phase of learning builds a tree graph. In nodes of a graph it puts rules based on features that distinguish most of the data and proceeds from top to bottom. Trees are trained with sub samples of a dataset and with random features selected. This practice is used to prevent over fitting and increase accuracy with respect to the decision tree. In a random forest the depth can be limited to prevent over fitting. In classification phase decision tree simply takes features given and apply rules from top to bottom and when it reaches leaf it gives prediction based on what is in the leaf. In random forest every tree gives a decision and final decision is average of all decisions made as shown in Figure 4.4.

Random forest can be learned on any data without any kind of metric known. Because my features have no common metric it is a big advantage. It is very fast in decision time and very robust to outliers. Disadvantages of random forest is over fitting but it is much more robust than decision tree. Also there is no well justified learning algorithm backed up by mathematical proofs and theory as are other classifiers that in learning phase target e.g training error. Later *Rfc* will be used for Random forest classifier.

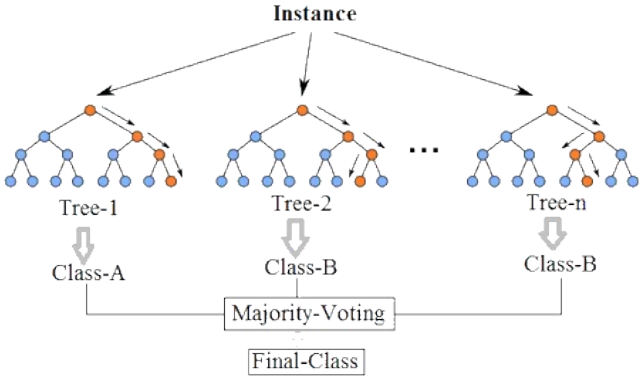


Figure 4.4: Random forest

4.4.2 Support Vector Machines

Support Vector Machines (SVM) is machine learning method that tries to find hyperplane that separates feature space such that training data from different classes lies in different half-spaces. Optimal hyperplane is one from which minimal feature distance is the highest. This can not be used on any data but only on data that can be separated by hyperplane. Those data are called linearly separable. To extend this to work with data that are not linearly separable kernel transformations are used. This transforms feature space to mostly higher dimensional space in order to achieve linear separability as shown in Figure 4.5[1]. Commonly used kernels are linear: $\langle x, x' \rangle$, polynomial: $(\gamma \langle x, x' \rangle) + r)^d$, sigmoid: $\tanh(\gamma \langle x, x' \rangle) + r)$ and rbf: $\exp(-\gamma |x, x'|^2))$ where γ, r and d are constant parameters that can be tuned. Sometimes even kernel trick does not make the data linearly separable and SVM is allowed to misclassify data points but for each misclassified point penalty is added. This version is called soft-margin SVM.

I am interested in SVM because it can handle two class problems very well. Also SVM has regularization parameter which can set how much I want to push down training error in trade for more general solution. Secondly with kernel trick with enough knowledge about the problem it is possible to build custom kernel for the problem. Also SVM optimization problem can not get stuck in local minimum and can be computed by known efficient methods.

[1] https://github.com/nicolaspanel/node-svm

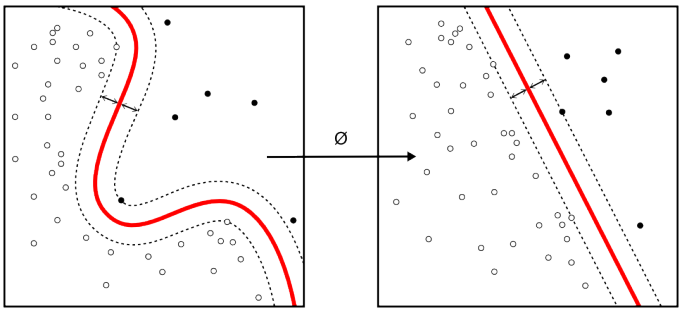


Figure 4.5: SVM Kernel trick

Lastly SVM is built on huge amount of theory. Disadvantages of SVM are that data needs to be normalized and also big part is choice of kernel and it may be hard to tune parameters for the optimal result. In following text *Svc* is used to refer to the SVM classifier.

4.4.3 Neural network

Idea behind Neural network is inspired by human brain which is really good at learning. Neural network can very well handle problems with multiple classes and it can do both classification and regression. It can be tuned for good generalization but it requires huge amount of data. Also, there is no guarantee to reach global minimum while learning.

Basic building blocks of Neural network are neurons as shown in Figure 4.6 [4]. Neurons have inputs $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$, weights $\mathbf{w} = (w_1, \ldots, w_n) \in \mathbb{R}^n$ and bias $b \in \mathbb{R}$. Activation function $f : \mathbb{R} \to \mathbb{R}$ in neuron produces output $y \in \mathbb{R}$ based on input which is computed from given weights $\mathbf{w}$, bias b and the input $\mathbf{x}$.

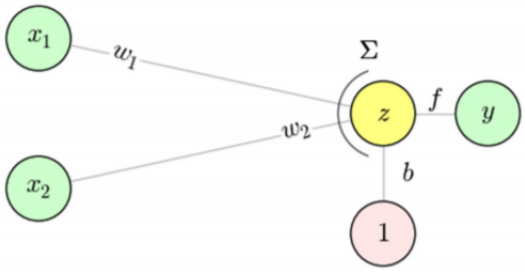


Figure 4.6: Neuron

One layer of neural network consists from unconnected neurons. When more layers connect together it creates neural network. Inputs are connected to all neurons of the first layer and all layers are connected this is architecture that I used and it is called Feed-forward neural network. I used two and three layer fully connected network. As activation function I used Rectified linear unit (ReLu): $ReLu : f(x) = max(0, x)$. In next text *Nnc* is used for Neural network classifier.

Learning of Neural network is done by setting weights $\mathbf{w}_i$ with backpropagation algorithm. Other algorithms for extreme searching can be used but backpropagation is used in almost all cases. This algorithm computes outputs and then computes gradients and using gradient descent it updates weights. To avoid getting stuck in local minimum not all training samples are used in each step of gradient descent. Number of samples used is called batch size.

Chapter 5

Implementation

In this chapter I cover programming language that I used and explain why. I describe datasets I had and show its computed statistics. I also describe data that I gathered without using trajectories and how I gathered them. Finally, I show how I implemented all steps from preprocessing to classification.

5.1 Language

As programming language I chose python because of its simplicity and great versatility. Python is also very popular in scientific circles. Furthermore, many great machine learning libraries exists for python.

5.2 Datasets

For my research, training and testing I had two datasets available.

5.2.1 Geolife

First one is Geolife, dataset collected by Microsoft Research Asia during Geolife project [17, 14, 16]. It was collected by 182 users over five-year period. 73 users also provided labels containing modes of transportation for their trajectories such as car, train, bike or walk. Trajectories are recorded in few major cities around the world but most of them take place in Beijing. In this dataset labels are separate files pointing inside trajectory files. From that I made two datasets one with trajectories corresponding to one mode of transportation by cutting the original dataset based on labels. Second dataset I made are original trajectories but with a tag of transportation mode in every GPS point.

5.2.2 DPNK

DPNK dataset was provided to me by my supervisor and it is dataset that was recorded mostly by cyclists in the Czech Republic.

Feature	Geolife (bike)	Geolife (Not bike)	DPNK
Distance [m]	3237.537	6230.102	8365.907
Time [s]	1488.879	1875.695	2406.706
Max speed [m/s]	15.434	16.026	11.559
Average speed [m/s]	2.665	3.630	4.242
Acceleration [m/s^2]	0.214	0.177	0.115
Deceleration [m/s^2]	-0.223	-0.178	-0.115
Altitude up	0.802	0.858	0.111
Altitude down	-0.790	-0.827	-0.112
Angle [deg]	0.019	0.029	0.029
Proximity [m]	3.700	7.009	25.126

Table 5.1: Datasets average trajectory features

5.2.3 Datasets statistics

Table 5.1 shows statistics of average trajectory features of both datasets. From Geolife only part of trajectories recorded in Beijing is used and also Geolife is separated to bike part and not bike part. In statistics it can already be seen that some feature averages are very different in bike part of Geolife dataset and DPNK dataset.

5.2.4 Altitude data

Geolife dataset contains altitude data from GPS but this altitude data is extremely inaccurate. DPNK does not contain any altitude data at all. To get altitude data I used elevation Api and added required data to the dataset. Elevation Api returns elevation for GPS position. Because of elevation API request limit this process took about four days. When I needed to replace elevation data in Geolife which is much bigger than DPNK dataset I instead used SRTM elevation data. SRTM is short for Shuttle Radar Topography Mission and contains topographical data collected by NASA from satellites[1]. SRTM provided me with data that is less accurate than elevation API but much more accurate than data previously contained in the dataset.

5.2.5 Bicycle data

For Proximity feature I needed way to get closest point of bike infrastructure. First I tried to use Open street maps API[2] but requesting for every point in trajectory was extremely slow and could not be used. I needed to get the data in some file that my program could later use. I decided to extract trajectories from Geolife that were recorded in Beijing. After that I downloaded Beijing cycle map data from OpenStreetMap[3] using Overpass Turbo[4] and modified its data structure to single lines entries so it could be loaded faster.

[1] https://github.com/tkrajina/srtm.py
[2] http://wiki.openstreetmap.org/wiki/API_v0.6
[3] https://www.openstreetmap.org/
[4] https://overpass-turbo.eu/

For DPNK dataset I needed data from whole Czech Republic which was too much for Overpass Turbo to handle so I downloaded map osm file for Czech republic and used OSM-Filter[5] to extract bike infrastructure from the data. I also modified data structure to single line entries to reduce size.

5.2.6 K-D tree

For fast computing of nearest neighbour I used K-D tree. K-D tree, which is short for K-dimensional computation tree, is space partitioning structure used to organize points in k-dimensional space. In my case the space is two dimensional. K-D tree is binary tree where every node is a point. Every non-leaf point can be viewed as splitting hyperplane and all nodes in left subtree are on one side of the hyperplane and all points in the right subtree are on the other side of the hyperplane. Nearest neighbour searching complexity in K-D tree is O(log n).

5.3 Preprocessing

For preprocessing I implemented both *heuristic* and *median* filters myself. For *heuristic filter* I used speed threshold 50 m/s and tested every two consecutive nodes against this threshold. If speed was above the threshold I removed node from trajectory and tested new neighbor node with the old one. For *median filter* I used window of size 21 and implemented it as filter applied to trajectory which is passed as array of lines. Filter computes median of points in the window and then replaces the old point with the newly computed values. All filters are implemented in module `preprocessing.py`.

5.4 Feature extraction

I have class Node in module `node.py` for GPS points. Node has methods to compute basic features like distance and time. I also included function to compute angle of three Nodes and distance to closest bicycle infrastructure using given K-D tree with infrastructure points.

Next I have module `feature_extraction.py` where I have functions for creating feature file from trajectories in specified folder.

5.5 Classification

First I used labeled trajectories and features extracted from them to train my classifiers. When I have trained classifiers all I have to do is extract features and feed them to the classifiers for prediction. For *Rfc* I used scikit-learn [2] implementation of random forest. For *Svc* I used scikit-learn implementations of *LinearSVC*, *NuSVC* and *SVC* modules. Finally for building of *Nnc* I used Keras [6].

[5] http://wiki.openstreetmap.org/wiki/Osmfilter
[6] https://github.com/fchollet/keras

Chapter 6

Evaluation

In this chapter I will evaluate segmentation and cover problems I encountered. Next I will evaluate all classifiers that I used and show how different parameters can influence their performance. Next is comparison of results and testing of preprocessing and features using learned classifiers.

6.1 Segmentation

For segmentation quality I used metric:

$$M = 1 - \frac{not\_split}{all\_splits}$$

where not_split are changes in transportation mode not detected by segmentation and all_splits are all changes of transportation modes in trajectories. When I tested segmentation using method described in Chapter 4 I realized that metrics M I chose is not that good. Problem is that for stay points containing more trajectory points less non detected stay points are in trajectory but also less points are in the resulting segments. As can be seen in graph shown in Figure 6.1 as M rises less and less points are kept in the trajectory. I multiplied M and fraction of points left to get the method that has good quality but leaves enough points in the trajectory. I used multiplication instead of addition to receive well balanced method. As shown in Table 6.1 when I increase T_t with the same D_t M is decreasing and Points left are increasing. When I increase D_t with same T_t M is increasing and Points left are decreasing. Best results are for segmentation with $D_t = 14$ and $T_t = 18$. This segmentation has M 0,767 while still leaving 75,9% points in the trajectory.

6.2 Classification

To compare different methods I need to measure method's performance. This is done by comparing output labels with ground truth labels. By comparing ground truth with labels predicted by my method, I can measure quality of the predictions. All measures are based on number of True and False positives (TP and FP), meaning a number of correctly and

D_t	T_t	**M**	**Points left**	**Multiplied**
14	10	0,834	0,633	0,528
14	12	0,821	0,656	0,539
14	14	0,803	0,705	0,566
14	16	0,781	0,731	0,571
14	**18**	**0,767**	**0,759**	**0,582**
14	20	0,755	0,77	0,581
14	22	0,728	0,791	0,576
14	24	0,721	0,797	0,574
14	26	0,693	0,812	0,562
14	28	0,686	0,817	0,561
10	18	0,722	0,802	0,579
12	18	0,732	0,78	0,571
16	18	0,774	0,737	0,57
18	18	0,774	0,71	0,549
20	18	0,793	0,686	0,544
22	18	0,821	0,661	0,543
24	18	0,832	0,642	0,534

Table 6.1: Segmentation results

	Positive ground truth	Negative ground truth
Positive prediction	TP	FP
Negative prediction	FN	TN

Table 6.2: Outcomes of binary classifier

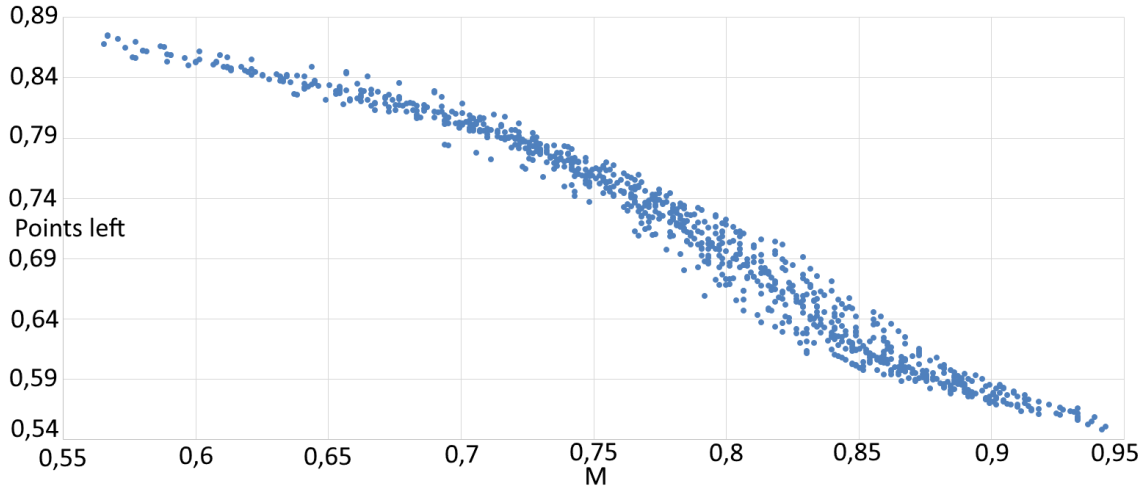


Figure 6.1: Dependency of segmentation quality metric M and points left in dataset after performing segmentation.

incorrectly predicted positives. True and False negatives (TN and FN), meaning a number of correctly and incorrectly predicted negatives. It is illustratively showed in Table 6.2.

With these terms I can define metrics used for classification as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \quad Precision = \frac{TP}{FP + TP}, \quad Recall = \frac{TP}{FN + TP}$$

6.2.1 Random forest

I tested *Rfc* with different number of trees and computed not only accuracy but also time required for learning and classification. Results are shown in Table 6.3. From the results it is obvious that because of rising classification time for real time classification programs getting over 1000 trees can be a problem. Also, the increase in accuracy is not that big to warrant the trouble. In the following text I use *Rfc* with 100 trees.

6.2.2 SVM

I tested SVM because it is known to be good classifier for two-class problems which is exactly what I am dealing with. I tried different models from scikit-learn [2]. I tested different kernels and parameters and created Table 6.4 where I show my best results for each kernel and it's parameters. C is regularization parameter. For *NuSVC* it is not possible to set it because it instead controls number of support vectors. For polynomial kernel number in brackets is degree of polynomial and *Cf* is independent term in kernel function. For next computation I use *SVC* with rbf kernel as described in Chapter 4 and $C = 1000000$.

Trees	Accuracy	Learn time [s]	Class time [s]
10	0.893	0.125	0.107
20	0.902	0.131	0.106
30	0.903	0.138	0.107
40	0.906	0.150	0.107
50	0.903	0.155	0.107
60	0.904	0.162	0.106
70	0.903	0.187	0.107
80	0.904	0.175	0.106
90	0.906	0.286	0.107
100	0.905	0.291	0.108
200	0.906	0.467	0.108
300	0.905	0.645	0.109
400	0.906	0.837	0.108
500	0.905	1.010	0.209
600	0.906	1.204	0.209
700	0.906	1.414	0.210
800	0.908	1.844	0.274
900	0.906	1.797	0.210
1000	0.906	1.978	0.312
5000	0.906	9.841	1.157
10000	0.906	20.083	2.283
50000	0.907	101.676	11.005
100000	0.907	220.128	23.561

Table 6.3: Comparison of different number of trees in *Rfc* using Geolife data segmented by hand with five fold cross validation.

Module	Accuracy	Learn	Class	Kernel	C	Cf
LinearSVC	0.760	0.174	0.001	linear	100	
NuSVC	0.407	0.073	0.008	linear		
NuSVC	0.811	0.163	0.011	polynomial(8)		1
NuSVC	0.631	0.141	0.020	sigmoid		0
NuSVC	0.798	0.179	0.014	rbf		
SVC	0.775	52.229	0.009	linear	1000000	
SVC	0.822	2.707	0.007	polynomial(3)	1000000	3
SVC	0.794	0.164	0.007	sigmoid	100000	0
SVC	0.835	9.025	0.012	rbf	1000000	

Table 6.4: Comparison of different *Svc* configurations using Geolife data segmented by hand with five fold cross validation.

First	Second	Third	Batch	Epochs	Accuracy	Learn time	Class time
32	32	0	128	50	0.810	2.580	0.073
32	32	32	128	250	0.834	10.936	0.074
128	128	0	128	500	0.851	31.008	0.093
128	128	128	128	500	0.827	35.409	0.114
128	128	128	256	1000	0.844	64.207	0.130
128	128	128	64	500	0.821	52.965	0.097
256	256	0	128	1000	0.851	107.996	0.127
256	256	256	512	1500	0.858	144.377	0.120

Table 6.5: Comparison of different *Nnc* configurations using Geolife data segmented by hand with five fold cross validation.

6.2.3 Neural networks

I also experimented with neural networks and tried to construct simple one to test it's performance on my data. I spend some time setting different number of neurons, epochs and reached results that are not that far from random forest results and are in Table 6.5.

6.2.4 Results

From my tests it can be shown that best classifier for my problem is *Rfc* with best accuracy 90,6%. Next was *Nnc* with best accuracy 85,8% and last *Svc* with accuracy 83,5%. This is tested with five fold cross validation and without segmentation.

6.2.5 Classification results with segmentation

After using segmentation that I chose based on it's performance I tested different classifiers with it. I also measured precision and recall in order to be able to compare with prior art. Results with segmentation are shown in Table 6.6. Point-wise results compare predictions for each point and its ground truth. Trajectory-wise scores compares prediction for trajectory with ground truth in each point of trajectory. Only if each point's ground truth corresponds to trajectory prediction the trajectory is classified correctly. When comparing Table 6.6 with previous results it can be seen that using segmentation decreases accuracy. Even though *Rfc* was best at perfectly segmented test data it is worst when dealing with non perfect segmentation and *Svc* and *Nnc* give better results. In points-wise experiments best accuracy of 87,9% was achieved by *Nnc* 87,9% and best precision of 83% by *Svc*. Best recall of 93,6% is achieved by *Rfc* while lacking on precision as it is only 73%. Trajectory-wise best accuracy and precision was achieved by *Svc* with values 75,9% and 79,6%. Highest recall 87,5% again by *Rfc* and this time precision is only 6,2% worse than *Svc* compared to Point wise where the difference was 10,3%.

Compared to results without segmentation where *Rfc* dominated in this case of segmented trajectories *Svc* is the best. It can be because *Rfc* over fitted for perfect segmentation values.

	Point wise			Trajectory wise		
Classifier	**Accuracy**	**Precision**	**Recall**	**Accuracy**	**Precision**	**Recall**
SVM	0.865	0.833	0.871	0.759	0.796	0.794
Neural network	0.879	0.830	0.916	0.745	0.747	0.854
Random forest	0.817	0.730	0.936	0.741	0.734	0.875

Table 6.6: Results on Geolife data segmented by my segmentation method.

Filter	Improvement
Mean	1,95%
Median	4,33%
Heuristic	3,03%
Mean + Median	1,95%
Mean + Heuristic	3,9%
Median + Heuristic	**5,19%**

Table 6.7: Filter improvement

6.2.6 DPNK dataset

I also tested classification on DPNK dataset to evaluate generality and scalability of my classifiers. First attempt of only using *Rfc* ended with 25% of trajectories classified as bicycle. Considering almost all of the trajectories in DPNK should be bicycle this result was very unsatisfactory. I also tried *Nnc* on this because it should be more generalizing and it got to 42,1% which is much better but still not the result I would like to and *Svc* classified 45,3% as a bicycle. I computed difference between averages in datasets and modified dataset without labels to have same averages as training dataset by subtracting the difference and with that I had 73% bike trajectories. This result is still not that good but compared to starting 25% it is much better. After using tree with 300 minimum samples in leaf in order to generalize more I classified 85,6%.

6.3 Preprocessing

When choosing which filter to use I tested Mean, Median and Heuristic based filters to remove noise from the data. I have chosen not to use Kalman and Particle filter because of harder implementation and also because of their disadvantages which can be reduce quality when handling GPS trajectories which often start with noisy point.

I tested different filters to see how much the classification will improve. Results obtained with *Rfc* are shown in 6.7. From these measurements it can be seen that Median + Heuristic filter performed the best in this testing and therefore I used this filtering in my method.

6.4 Feature extraction

I tested different feature combinations to determine which features are best for distinguishing between bicycle and not bicycle and tried classifiers with different amount of different features. Results are shown in a Table 6.8. I put there all single feature classifications from which the best is Average speed and also all nine feature classification. It can be seen that the worse is the one without bicycle infrastructure Proximity. From other number of features I only selected the one combination that had the best accuracy. I tested all three classifiers and results show that *Svc* has problem with only one feature classification and *Rfc* can classify 73,8% right using only average speed *Nnc* even 80,9%. Results show that when using only nine features *Rfc* has worse result without average speed which is also best feature alone for *Rfc*. *Nnc* and *Svc* have worse result without distance and *Svc* accuracy difference from testing with all features is 24,1% which is significant. Overall increasing tendency of accuracy with more features is noticeable.

6.5 Conclusion

As can be seen from the results *Rfc* outperformed other types of classifiers for hand made segmentation even though Neural network was not that far behind. With more tweaking it could probably get even closer. Using my method of segmentation made the results worse, with *Svc* outperforming other classifiers. With *Svc* accuracy of 75,9%, precision of 79,6% and recall of 79,4% my method is worse than best method in the research field. I think that it is because competing algorithms use more map extracted features. I see room for improvement in segmentation method.

Classifier trained on dataset from Beijing had very bad result when used with data from Czech Republic and data had to be modified to increase classification accuracy. In this direction more research can be done to discover how to modify dataset or classifiers in order to classify data from different places.

Dst	**Tme**	**Max**	**Avg**	**Acc**	**Dcc**	**Aup**	**Adw**	**Ang**	**Pro**	**Rfc**	**Svc**	**Nnc**
Yes	No	No	No	No	No	No	No	No	No	0.651	0.529	0.650
No	**Yes**	No	No	No	No	No	No	No	No	0.589	0.529	0.537
No	No	**Yes**	No	No	No	No	No	No	No	0.565	0.529	0.586
No	No	No	**Yes**	No	No	No	No	No	No	**0.738**	0.529	**0.809**
No	No	No	No	**Yes**	No	No	No	No	No	0.556	0.529	0.565
No	No	No	No	No	**Yes**	No	No	No	No	0.552	0.530	0.578
No	No	No	No	No	No	**Yes**	No	No	No	0.641	0.560	0.719
No	No	No	No	No	No	No	**Yes**	No	No	0.615	0.559	0.704
No	No	No	No	No	No	No	No	**Yes**	No	0.543	0.529	0.589
No	No	No	No	No	No	No	No	No	**Yes**	0.595	0.529	0.654
Yes	No	No	**Yes**	No	No	No	No	No	No	0.865	0.563	0.670
Yes	No	No	**Yes**	No	No	**Yes**	No	No	No	0.881	0.619	0.602
Yes	No	No	**Yes**	No	**Yes**	**Yes**	No	No	No	0.889	0.618	0.663
Yes	No	No	**Yes**	No	**Yes**	**Yes**	No	No	**Yes**	0.897	0.600	0.703
Yes	No	**Yes**	**Yes**	No	**Yes**	No	**Yes**	No	**Yes**	0.906	0.627	0.775
No	**Yes**	**Yes**	**Yes**	No	**Yes**	**Yes**	**Yes**	No	**Yes**	0.905	0.600	0.749
Yes	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	No	0.896	0.831	0.817
Yes	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	No	**Yes**	0.905	0.832	0.784
Yes	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	No	**Yes**	**Yes**	0.902	0.831	0.759
Yes	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	No	**Yes**	**Yes**	**Yes**	0.901	0.827	0.798
Yes	**Yes**	**Yes**	**Yes**	**Yes**	No	**Yes**	**Yes**	**Yes**	**Yes**	0.901	0.827	0.812
Yes	**Yes**	**Yes**	**Yes**	No	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	0.902	0.828	0.802
Yes	**Yes**	**Yes**	No	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	0.882	0.826	0.770
Yes	**Yes**	No	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	0.902	0.828	0.751
Yes	No	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	0.903	0.638	0.728
No	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	0.901	0.594	0.695
Yes	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	0.905	0.835	0.858

Table 6.8: Feature accuracy of classification, shortcuts used are Dst for Distance, Tme for Time, Max for Maximal speed, Avg for Average speed, Acc for Acceleration, Acc for Deceleration, Aup for Altitude change upward, Adw for Altitude change downwards, Ang for Change of heading rate, Pro for Proximity of bicycle infrastructure.

Chapter 7

Conclusion

In my work I tried to create a program that could detect whether a trajectory was recorded on bicycle or not. To do this, I used methods found in the literature and combined them with a novel method described in this thesis. Basis of this method is the use of a new feature that to my knowledge has not been used before in trajectory classification. This feature is proximity of bicycle infrastructure and is extracted from map data.

Using my novel feature, I trained classifiers and compared their results. *Rfc* detected correct trajectories with accuracy 90,5%, *Svc* with accuracy 83,5% and *Nnc* with accuracy 85,8%. After first round of testing I employed segmentation that I chose based on segmentation metrics and points left in dataset. Then I tested all three classifiers in second round of experiments this time also with precision and recall while using point-wise and trajectory-wise metrics. In the experiments with point-wise metric best accuracy of 87,9% was achieved by *Nnc*. Trajectory-wise best accuracy of 75,9%, precision of 79,4% and recall of 79,4% were achieved by *Svc*. This shows that when dealing with segmented trajectories *Svc* can be better than both *Rfc* and *Nnc*.

When testing classifiers on DPNK dataset that should only contain bicycle trajectories *Rfc* classified 25% of trajectories as bike. *Nnc* classifier 42,1% of trajectories as bike and *Svc* classified 45,3% of trajectories as bike. After comparing feature averages for both datasets I moved features space of DPNK dataset to match that of Geolife. After that, *Rfc* with with 300 minimum samples in leaf classified 85,6% of trajectories as bicycle trajectories.

Overall results show that my method is working. However, best method in the field [18] classified trajectories with precision 93,58% and recall 93,32% which is much better than my method. I see room for improvement in segmentation. More work can be done in researching how to modify datasets or classifiers so it can be used on dataset from different parts of the world.

Bibliography

[1] Adel Bolbol, Tao Cheng, Ioannis Tsapakis, and James Haworth. Inferring hybrid transportation modes from sparse gps data using a moving window svm classification. *Computers, Environment and Urban Systems*, 36(6):526–537, 2012.

[2] David Cournapeau. scikit-learn, 2007. URL <http://scikit-learn.org/stable/>.

[3] Yuki Endo, Hiroyuki Toda, Kyosuke Nishida, and Jotaro Ikedo. Classifying spatial trajectories using representation learning. *International Journal of Data Science and Analytics*, 2(3-4):107–117, 2016.

[4] O. Drbohlav J. Matas, B. Flach. Neural networks, 2015. URL <https://cw.fel.cvut.cz/wiki/_media/courses/a4b33rpz/pr_09_nn_2015_11_27.pdf>.

[5] John Krumm and Eric Horvitz. Locadio: Inferring motion and location from wi-fi signal strengths. In *mobiquitous*, pages 4–13, 2004.

[6] Lin Liao, Donald J Patterson, Dieter Fox, and Henry Kautz. Learning and inferring transportation routines. *Artificial Intelligence*, 171(5-6):311–331, 2007.

[7] Akram Nour, Jeffrey Casello, and Bruce Hellinga. Developing and optimizing a transportation mode inference model utilizing data from gps embedded smartphones. In *Transportation Research Board 94th Annual Meeting*, number 15-5027, 2015.

[8] Donald J Patterson, Lin Liao, Dieter Fox, and Henry Kautz. Inferring high-level behavior from low-level sensors. In *International Conference on Ubiquitous Computing*, pages 73–89. Springer, 2003.

[9] Arthur L Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 3(3):210–229, 1959.

[10] Timothy Sohn, Alex Varshavsky, Anthony LaMarca, Mike Y Chen, Tanzeem Choudhury, Ian Smith, Sunny Consolvo, Jeffrey Hightower, William G Griswold, and Eyal De Lara. Mobility detection using everyday gsm traces. In *International Conference on Ubiquitous Computing*, pages 212–224. Springer, 2006.

[11] Leon Stenneth, Ouri Wolfson, Philip S Yu, and Bo Xu. Transportation mode detection using mobile phones and gis information. In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 54–63. ACM, 2011.

[12] Jie Yin, Xiaoyong Chai, and Qiang Yang. High-level goal recognition in a wireless lan. In *Proceedings of the national Conference on Artificial Intelligence*, pages 578–584. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2004.

[13] Yu Zheng. Trajectory data mining: an overview. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 6(3):29, 2015.

[14] Yu Zheng, Quannan Li, Yukun Chen, Xing Xie, and Wei-Ying Ma. Understanding mobility based on gps data. In *Proceedings of the 10th international conference on Ubiquitous computing*, pages 312–321. ACM, 2008.

[15] Yu Zheng, Like Liu, Longhao Wang, and Xing Xie. Learning transportation mode from raw gps data for geographic applications on the web. In *Proceedings of the 17th international conference on World Wide Web*, pages 247–256. ACM, 2008.

[16] Yu Zheng, Lizhu Zhang, Xing Xie, and Wei-Ying Ma. Mining interesting locations and travel sequences from gps trajectories. In *Proceedings of the 18th international conference on World wide web*, pages 791–800. ACM, 2009.

[17] Yu Zheng, Xing Xie, and Wei-Ying Ma. Geolife: A collaborative social networking service among user, location and trajectory. *IEEE Data Eng. Bull.*, 33(2):32–39, 2010.

[18] Xiaolu Zhu, Jinglin Li, Zhihan Liu, Shangguang Wang, and Fangchun Yang. Learning transportation annotated mobility profiles from gps data for context-aware mobile services. In *Services Computing (SCC), 2016 IEEE International Conference on*, pages 475–482. IEEE, 2016.

[19] Yin Zhu, Yu Zheng, Liuhang Zhang, Darshan Santani, Xing Xie, and Qiang Yang. Inferring taxi status using gps trajectories. *arXiv preprint arXiv:1205.4378*, 2012.

Appendix A

User Guide

In this work I created program that is on CD with my work and can be started.

A.1 Program usage

A.1.1 Requirements

Program uses python modules that are not present in common python distributions. Those modules are scipy, numpy and scikit-learn.

A.1.2 Data

When using own data it must be in right format. For trajectory format is:

$$longitude, latitude, 0, altitude, time1, time2, time3$$

on every line, where altitude is in feet and time1 is number of days (with fractional part) that have passed since 12/30/1899. time2 is YYYY-MM-DD and time3 is hh:mm:ss.

Example: 39.977781,116.333398,0,817,39728.4313888889, 2008-10-07,10:21:12,subway

A.1.3 Running program

Program can be executed from cmd by "python main.py" but requires arguments to do something.

A.1.4 Arguments

First argument engages different modes. p for preprocessing, c for classification, s for segmentation and f for feature extraction

A.1.4.1 Preprocessing

When using preprocessing next argument defines filter used and it can be med for median, mean for mean and heu for heuristic. Third argument for preprocessing is folder from which trajectories are loaded and fourth argument is folder to which filtered trajectories are saved. For mean and median filter optional argument can be used at the start -wnumber defines sliding window size.

Example: python main.py -w10 p med ../data/unfiltered some_folder

takes trajectory data from unfiltered folder on CD, performs median filtering and saves results in "some_folder".

A.1.4.2 Segmentation

When using segmentation next arguments are distance_threshold, time_threshold, folder_in and folder_out where distance and time thresholds are parameters of segmentation and folder in and out are folder from which trajectories are loaded for segmentation and folder to which are trajectories then saved.

Example: python main.py s 10 20 ../data/tosegment some_folder

takes trajectory data from tosegment folder on CD, performs segmentation with distance threshold 10 and time threshold 20 and saves result into "some_folder".

A.1.4.3 Feature extraction

When using feature extraction second argument is input folder for features extraction and third argument is name of feature file created.

Example: python main.py f ../data/unfiltered path/features

creates feature file "path/features.f" from trajectory data in unfiltered folder on CD.

A.1.4.4 Classification

When using classification next argument is classifier type, nn for neural network, rf for random forest and svm for SVM. Next argument is classification mode selection. tr for training, pr for prediction, ts for testing and tn for testing segmented trajectories.

Training

Training has two more arguments, first is feature file from which will classifier learn and second is name of classifier that will be saved.

Testing

In testing first argument is name of classifier (neural net only name and svm and random forest with extensions) second argument is feature file used for testing. Output is accuracy, precision and recall of classification.

Prediction

In prediction first argument is name of classifier (neural net only name and svm and random forest with extensions) and second argument is feature file used for prediction. Output prediction will be saved to file with same name as feature file but with extension .pr. -1 is bike and 1 is not bike.

Segmented classification

Last mode is classification of segmented. There I cannot create feature file so first argument is again classifier name (neural net only name and svm and random forest with extensions) and second argument is folder in which segmented trajectories are. Output is point vise accuracy, precision and recall and trajectory vise accuracy, precision and recall. This mode can run for very long time based on number of segments.

Examples: python main.py c svm tr ../data/features.f path/svm

creates new svm classifier "path/svm.svm" learned on feature file in data folder.

python main.py c rf ts ../data/rf.rf ../data/features.f

loads learned random forest from ../data/rf.rf and uses it to classify feature file ../data/features.f and then outputs accuracy, precision and recall of classification.

python main.py c rf pr ../data/rf.rf path/features.f

loads learned random forrest from ../data/rf.rf and predicts trajectories saved in path/features.f creates file path/features.pr with prediction if possible else prints predictions to console

python main.py c rf tn ../data/rf.rf ../data/tosegment

loads classifier from ../data/rf.rf and data from ../data/tosegment and outputs classification results for segmented trajectories.

tosegment data is not segmented but can be used for demonstration how it works. Or segmented data can be created with segmentation.

Appendix B

CD Structure

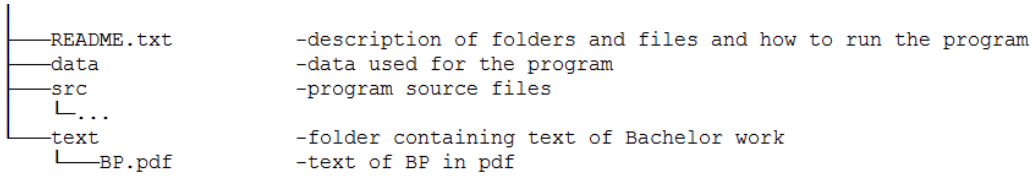


Figure B.1: List of CD structure