Machine Learning Based GPS Outliers Detection Methods

***Abstract*—Global positioning system (GPS) is essential to our daily life and has been used in different fields, especially in vehicle navigation. However, the error of GPS is always unnegligible, which can lead to the misjudgement of the navigation system and develop into a vehicle accident. To reduce the error of GPS and obtain a more accurate vehicle movement path, we tried machine learning methods to classify the GPS data in this paper. GPS data with large errors were distinguished from relatively accurate ones. Classification methods, including linear regression, logistic regression, support vector machine (SVM), and neural network, have been applied. Their corresponding models we set up were tested on the data from a GPS receiver to evaluate their effectiveness. Simulation time was also recorded to indicate their corresponding running time. The result showed that the data after classification could record a movement path closer to the actual one.**

I. Introduction

GPS is a space-based positioning system owned by United states government. Its principle is based on triangulation [1]. The coordinate of receiver is determined by calculating its distance to four different satellites using the propagation time. However, Many devices can only guarantee reliable (95% Confidence Interval) 7.8 meters positioning accuracy according to the United States government report. The error can even reach 10m level under some conditions. There are many reasons leading to the error. According to [2],the changeable atmospheric conditions, affecting the speed of the signals transmission, can add small errors in the calculated coordinates. In urban area, due to the block and reflections of tall buildings, receiver may get unclear signals which influence the distance calculation. As a result, accurate navigation in urban area is more difficult. The error of GPS will cause large deviation from GPS curved vehicle movement path to the actual one and make the navigation system send the wrong instruction, bring inconvenience to users and even lead to the traffic accidents. How to get precise GPS and actual movement path is a realistic and difficult topic where so many trials have been implemented on.

To solve this problem, Kalman filter [3] has been tried to be added to the GPS receiver to help process the GPS data. However, Kalman filter requirements for linear model and white Gaussian noise [4] are hard to meet. Only using Kalman filter cannot reduce the error to the level we expected. Nowadays, with the unprecedented development of machine learning, high degree of information integration and automation are steadily becoming reality. More smart algorithms have been developed than ever before, enabling fast and massive data processing. Patterns and trends can be more accurately predicted using machine learning techniques. With the help of machine learning technique, processing the large amount GPS data and removing the outliers from the candidates in a smart way has became possible. In this paper, we intended to use machine learning technique to distinguish bad GPS data from relative accurate one and record a more accurate movement path for vehicles. The key idea is to implement classification on every GPS data we received and label it. When trying to determine the coordinate fo one location, we only consider the GPS with good label and ignore the data with bad label.

Two locations far apart may have different atmosphere, which causes different disturbance to the GPS and lead to changes in the classification model. Accordingly, training data should be collected every time the vehicle stops and the parameters for classification model should be updated quite often. Given the need to collect real time data in moving vehicles and complicated in-vehicle communication system, the progress of training and testing must avoid using time-consuming methods in order to reduce the load for controller. Logistic regression is a very mature technology and easy to implement. It has been extensively used in statistics [5]. Logistic models can be updated with new data using gradient descent method. Linear regression algorithm can either work as a predictor or a classifier [6]. The parameters for linear regression model can be solved using normal equation, which save significantly computational time. In addition, Linear regression has a potential to be combined with Kalman filter and particle filter [7]. SVM vector machine can perform both linear and non-linear classification combined with different kernels. It works well in many cases and suited for classification of complex but small- or medium-sized datasets. Artificial neural networks (ANNs) are at the core of Deep Learning [8]. BP (Back Propagation) neural network is one kind of neural network among versatile ANNS. It is powerful and scalable, making them ideal to tackle large and highly complex Machine Learning tasks. Based on the analysis above, linear regression, logistic regression, SVM and BP neural network were selected in this paper and applied to do classification on received GPS data.

The rest of this paper is organized as follows: Section II introduces the details of our regression methods and the way to evaluate them. In Section III, the performances of the regression techniques on GPS classification are demonstrated by experiments and compared to the results without classification. GPS with Kalman Filter is also added for comparison. Simulation time for these classification methods is recorded to evaluate their time consumption. Finally, the conclusions of our study and future work are discussed in Section V.

II. Methodology

A. Current Averaging Method used on GPS receiver

GPS receivers may receive some GPS data for one location at a time. The final coordinate for this location is determined by averaging all collected data. This method is quite useful when data is collected over a period of several hours and then averaged. However, it still has big errors in dynamic

conditions. In this paper, averaging method is implemented after the classification.

B. Features Determinism

A GPS data collected at location t contains a latitude and longitude(x_1^{ti}, x_2^{ti}), where i indicates the index number of a sample. Thus, Latitude and longitude form two direct features for each GPS sample. However, if we use these direct features, the frequency to update model parameters would be very high and bring the load to controller as the GPS data is related to position. Based on that, we proposed a way to normalize the data as($x_1^{ti}-\overline{x_1^t}$, $x_2^{ti}-\overline{x_2^t}$), where $\overline{x_1^t}$, $\overline{x_2^t}$ is the averaged value of all collected GPS data at location t. The experiments showed the result of using only two features was not good enough. The accuracy on testing data is only 15.6% for linear regression. As a result, we want to increase the amount of features and link them to the data from previous locations at the same time. Then we came out a way to establish six features for a GPS sample as:

($x_1^{ti} - \overline{x_1^t}$, $x_2^{ti} - \overline{x_2^t}$, $x_1^{ti} - \overline{x_1^{t-1}}$, $x_2^{ti} - \overline{x_2^{t-1}}$, $x_1^{ti} - \overline{x_1^{t-2}}$, $x_2^{ti} - \overline{x_2^{t-2}}$).

Here we used the average GPS value from location t-1 and t-2 to create the other four features. However, we met a low rank problem after we got the total feature matrix. The rank for feature matrix is 4 and the number of parameters to solve is 6. The parameters cannot be solved. As a result, the model we obtained loses its function of classification. It will give the same labels to all the data sample. As shown in Figure 1, the testing accuracy and training accuracy keep unchanged when we increasing the iteration in logistic regression. We finally

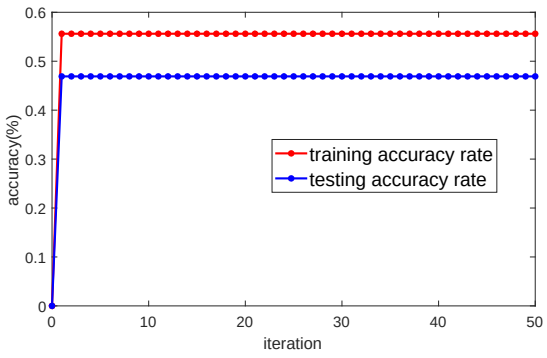


Fig. 1. The low rank problem causes the model losing its ability to do classification

decided to use the square from of the above six features. which increased the rank of feature matrix to 7. The final features for sample GPS data sample are:

($(x_1^{ti} - \overline{x_1^t})^2$, $(x_2^{ti} - \overline{x_2^t})^2$, $(x_1^{ti} - \overline{x_1^{t-1}})^2$, $(x_2^{ti} - \overline{x_2^{t-1}})^2$, $(x_1^{ti} - \overline{x_1^{t-2}})^2$, $(x_2^{ti} - \overline{x_2^{t-2}})^2$).

This feature construct can help us train the correct model and shows the increase in both training and testing accuracy at later experiments.

C. Linear Regression and Logistic Regression

Linear regression and logistic regression in this paper are used to evaluate the relationship between labels("good" or "bad") and the GPS data collected (latitude and longitude).

Linear regression [8] [9] is a typical supervised learning technique. We assume the outputs can be fitted in a linear model. Depending on the number of variables, a linear regression can be categorized as "univariate" or "multivariate". A math expression for the model is called hypothesis function. An multivariate hypothesis function is:

$$\widehat{y} = \theta^T X \tag{1}$$

where X is the input, y is the output. $\widehat{y}$ is the estimation of y. θ are parameters that we need to find. To quantify the quality of our guess, there is a cost function J(θ):

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(\widehat{y_i} - y_i)^2 \tag{2}$$

where m is the number of training data, and i indicates the index. The objective of the training is to minimize the cost function. In linear regression, we can quickly get the result though normal equation.

$$\theta = (X^T X)^{-1} X^T y \tag{3}$$

Although linear regression is commonly used as a predictor, we can define a classification based on its predicted value. It will be labeled as 1 ("good") when the linear regression result is larger than 0 and labeled -1("bad") when the result is less than 0. Then it can work as a classifier in this paper.

Logistic regression [10] is more often used as a classifier. The relationship between output and input is:

$$P(C = 1|x) = f(x) = \frac{1}{1 + e^{-\theta x}} \tag{4}$$

As a convention,we give a positive judgment (l) of a sample when the output is bigger than 0.5. Otherwise, it will be given a negative judgement (0). The system is learning by minimize the cost function:

$$J(\theta) = \sum_{i=1}^{m} log(1 + e^{-c^{'i}\theta^T x^i}) \tag{5}$$

Where $c^{'} \in \{-1, 1\}$ is the logistic loss. The most convenient method to solve the cost function of logistic regression is gradient descent [11], which is the method to update logistic regression model in the experiments shown in this paper.

D. SVM

SVM [8] is a effective and versatile machine learning model. It can be used on linear and non-linear classification. Given labeled training data, SVM aimed to output a hyperplane (decision boundary) which will separate the plane into different parts. The testing data is classified according to which part it belongs to. SVM inherits many traits from logistic regression. We can obtain its cost function based on the cost function of logistic regression. Firstly, we need to define two functions, which is named as $cost_0$ and $cost_1$. The definition is :

$$\begin{aligned} cost_0(z) &= max\{0, k_0(1+z)\} \\ cost_1(z) &= max\{1, k_1(1-z)\} \end{aligned} \tag{6}$$

where k_0 is the slope of the line which is tangent to $-log(1 - \frac{1}{1+e^{-z}})$ and passes the point (-1,0). $k_0 = -k_1$. we replace two logistic regression y=1 and y=0 terms with $cost_0$ and $cost_1$ and get the cost function for SVM:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}[y^i cost_1(\theta^T x^i) + (1-y^i) cost_0(\theta^T x^i)] + \frac{\lambda}{2m}\sum_{i=1}^{m}\theta_j^2 \tag{7}$$

SVM has many kernels to choose from, such as linear kernel, Gaussian RBF kernel, polynomial kernel and string kernel. Considering the requirement of low computation time and a

small scale of training data collected in dynamic progress, linear kernel and Gaussian RBF kernel are chosen in this paper to implement classification on collected GPS.

In linear kernel, the decision boundary is a straight line (the training data is linearly separable). Linear kernel is efficient and works well in many cases. Besides linear kernel, we also want to find a non-linear boundary. Given the training data set is not too large, we tried Gaussian RBF kernel, which is widely used and suitable for many cases. Gaussian RBF has two important hyperparameters gamma (γ) and C. In my design, I used sigma (σ) instead of gamma (γ). Sigma (σ) is the standard deviation of the Gaussian kernel which can also show the steepness of the rise around the landmark. The relationship between sigma (σ) and gamma (γ) is: a larger gamma (γ) corresponds to a smaller sigma (σ). If we increase the value of sigma (σ), we increase the influence range of each instance and obtain a more regular boundary. The change of C value will have an opposite effect on the model. The simulation to study the relationship between accuracy and the value of sigma (σ) and C will be shown in later experiment segment.

E. BP neural network

BP (Back Propagation) neural network is a back-propagation multi-layer feed forward neural network [12]. Neural network is arranged in layered manner and each layer consists of a number of nodes (neurons). The nodes in the same layer are not connected, while the nodes in the adjacent layer connect to each other. Each neuron is a computational unit that takes in inputs from other neurons in previous layer and outputs the result to next layer. This network was originally known as the multi-layer Perceptron network. A typical neural network is shown in the Figure 2 below: where w_{ij}^k is weight

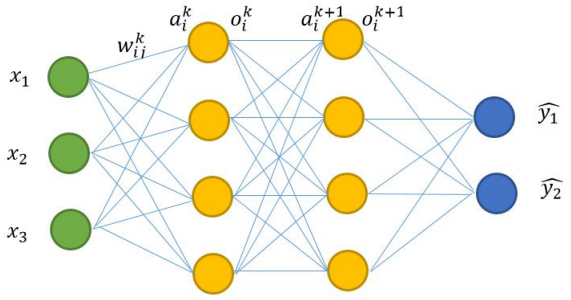


Fig. 2. Typical structure of a neural network

connects neuron i in layer k-1 and neuron j in layer k. a_i^k is the product sum (input) for node i in layer k. o_i^k is the output for node i in layer k. x and $\widehat{y}$ is the input and output for whole network. A BP neural network consists of input layer (the layer with green neurons), hidden layer (the layers with yellow neurons) and output layer (the layer with blue neurons). The number of neurons in input and output layer is decided by the number of features and the output we needed. The size of hidden layers and the number of neurons in each hidden layer is changeable and can influence the functionality of neural network.

The basic idea of training our BP neural network is Gradient Descent using reverse-mode autodiff [8]. At the beginning, we need to give the weights in network an initial value 1 and choose the sigmoid logistic activation function $\frac{1}{1+e^{-a_i^k}}$. Then we computes network output based on the parameters we set and obtains the error between actual output and expected output. Then we need to measure the error contribution from each neuron in the previous hidden layer and so on until the algorithm reaches the input layer. Propagating the error gradient backward can efficiently obtain the error gradient for every connection weights. The last step of this back-propagation algorithm is to slightly adjust the connection weights to reduce the error. The computation progress is repeated until the error is reduced to the level we expected. The learning rate was set as 0.001 in order to avoid overfitting.

F. Processing Flow

The flow of the data processing progress is shown in Figure 3. In current navigation system, after we receiving GPS

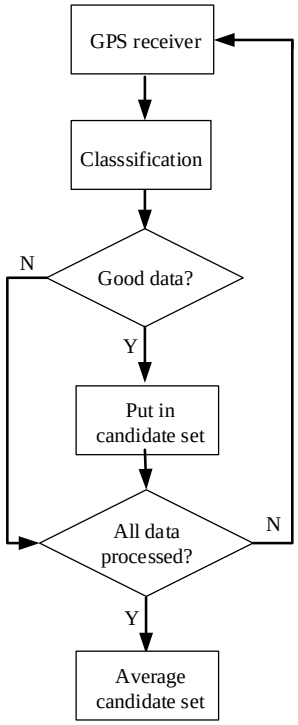


Fig. 3. Flow diagram to show the processing steps after we receive GPS data for a location

data for one location. The averaged value for these data is treated as the final coordinate for this location. As mentioned in introduction, processing GPS in this way will lead to 7.8 meters error and not suitable for moving vehicle. In this paper, we implement classification before we average the data. Only the data with a "good" label will be put into a candidate set. The "bad" labeled data is considered having a big error from actual location. These data will be ignored when we want to determine the coordinate and thus reduce the error from averaging all original data. After we classifying all the received data, we only average the data in candidate set.

G. SST

In order to evaluate the effectiveness of our methods, we introduce total sum of squares (SST). SST is defined as:

$$SST = \sum (p - \widehat{p})^2 \tag{8}$$

where $\widehat{p}$ is the coordinate GPS gives to us and p is the actual coordinate. Each coordinate includes a latitude value and longitude value. $p - \widehat{p}$ computes the distance between coordinate from GPS ($lati1$, $longi1$) and actual coordinate ($\widehat{lati}$, $\widehat{longi}$). It can be computed with Haversine formula [13].

Firstly, we need to compute the difference values for two latitudes and two longitudes individually.

$$\begin{aligned} \Delta lati &= lati - \widehat{lati} \\ \Delta longi &= longi - \widehat{longi} \end{aligned} \tag{9}$$

The Haversine item h is given by:

$$h = sin^2(\frac{\Delta lati}{2}) + cos(lati)cos(\widehat{lati})sin^2(\frac{\Delta longi}{2}) \tag{10}$$

The distance is computed with:

$$d = 2r * arcsin(\sqrt{h}) \tag{11}$$

Where r is the radius of earth ($6371km$). Haversine formula is to compute the distance between two points using their coordinates. SST computes the sum of such distance for all corresponding points from actual path and GPS curved path. It indicates the cumulative error from GPS curved path to actual path. The less SST is, the more similarity between GPS curved movement path and actual path, which is what we expect. We compute SST values for our different classification methods and the way to average all original data without classification in later experiment segment.

III. Experiment Results

The testing system is shown in Figure 4. We only built a straight line testing system as most of the vehicle movements are in line. The actions of direction change can be detected with the help of sensors. The GPS receiver was set on a model

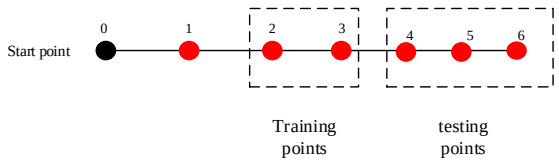


Fig. 4. Total 7 test points, 21000 GPS data were collected to train and test the classification models

car. The GPS receiver started to record the GPS data when the car passed the red testing point illustrated in the Figure 4. There are totally 7 testing points. Each point collected 3000 GPS data and had an independent candidate set. As the features for each GPS sample contains average value of two previous testing point, we should start calculating features from 3rd testing point. Point 2 and 3 contain 6000 samples and were used as training data. Point 4 , 5 and 6 consist of 9000 samples and were treated as testing data. A GPS sample, which gives the location within 4 meters around the actual one is labeled as "good" and given the value 1. The sample, which distance to actual location is more than 4 meters, is labeled as "bad" and given the value 0.

A. Linear Regression accuracy

Normal equation was used to calculate the six parameters for linear regression. After calculation, we tested our linear model both on the training data and testing data. The accuracy on training data is 77.75%(4653/6000). 77.75% training data is given the right label. The testing accuracy is 51.56% (4640/9000). Which improves a lot from using only two features.

B. Logistic Regression accuracy

Gradient descent was used to learn the suitable parameters for our logistic regression model. The training and testing accuracy related to the times of iterations is shown in Figure 5. We can see the testing accuracy starts to increase after 9th

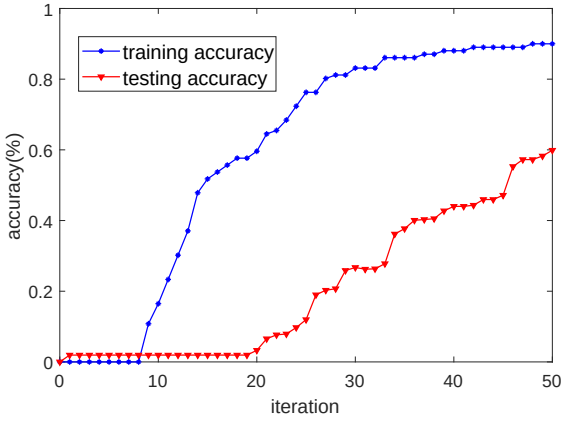


Fig. 5. the trend of testing and training accuracy when the iteration times increases

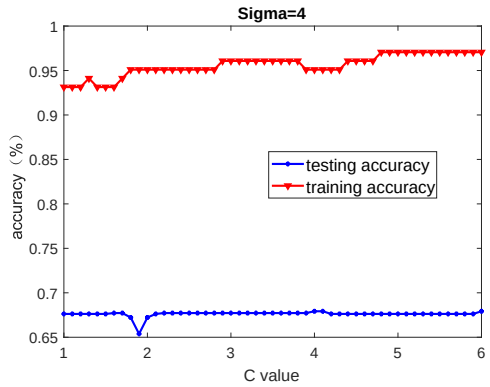
iteration while the testing accuracy starts to increase after 19th iteration. Training accuracy increases quickly at the beginning and slows after 30th iteration. The increase of testing accuracy is quite stable. After 50 iteration, the training accuracy is 89.26% and the testing accuracy is 59.82%.

C. SVM

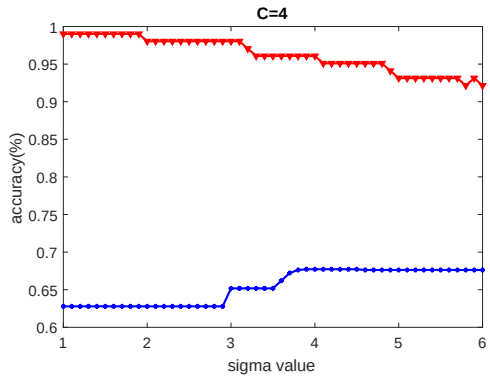
We built the models for SVM with linear kernel and Gaussian RBF. For linear kernel, the training accuracy is 94.12% and the testing accuracy is 60.19%. When deal with RBF kernel, we tried to change the value of sigma(σ) and C and then record the corresponding testing and training accuracy. The result is shown in the Figure 6. Both the sigma (σ)and C changed from 1 to 6. According to the result shown in Figure 6, We can obtain the best testing accuracy when sigma (σ) and C equals to 4. At that time, the training accuracy is 97.06% and the testing accuracy is 63.65%. We met a problem when we want to restore the movement path using the GPS data classified by the SVM RBF. We found that the model built from SVM RBF classified all the GPS data at 6th testing point as bad data, which means we did not get any valid data at 6th testing point. As a result, we used the coordinate from averaging all original data as the final coordinate for 6th testing point. This influences the SST value calculation for SVM RBF.

D. BP neural network

Neural networks classification is influenced by the number of neurons in the hidden layer. In our experiment, we build



(a) sigma is fixed. The change of accuracy related to the value of C



(b) C is fixed. The change of accuracy related to the value of sigma

Fig. 6. accuracy related to the changed of two parameters

a neural network with one hidden layer, change the size of neurons and then record the training accuracy , testing accuracy and simulation time in Matlab. Given the training accuracy is always beyond 99%, we just plotted the result of testing accuracy and simulation time in Figure 7. According to the

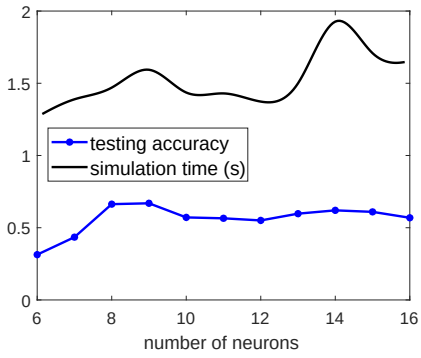


Fig. 7. testing accuracy and simulation time related to the number of neurons

figure, the simulation time for one hidden layer neural network fluctuates around 1.5s. The testing accuracy is around 60% when we change the size of neurons. We can get the best testing accuracy when we increase the number of neurons to 9. The testing accuracy is 66.911% while the training accuracy is 100%. Fixing the number of neurons for each hidden layer as 9, we tried to increase the number of hidden layers. When we used 2 hidden layers, the testing accuracy was 66.411% while the time increased from 1.691s to 1.884s. We obtained a lower accuracy with more time cost. When we increased the number of hidden layers to 3, we got the testing accuracy as 75.944% and the simulation time was 2.269s. We got a 9% more accuracy with the cost of additional 34.2% time, which is not corresponds to our time efficient purpose.

E. SST value

After classification using above five classification methods, We put the data labeled 1 into candidate set and ignored the data labeled 0. Then we implemented averaging on the data in candidate set. The process should be repeated for each testing point. We can record the movement path by connecting the coordinates for 7 points obtained from candidate set. The result was compared to the path curved by original GPS data (without any classification). The paths curved by classified GPS and GPS without classification are shown in Figure 8. The actual path is also shown in Figure 8 as green dot line. The red line is the path curved by averaging original data. The blue line is the GPS curved path with different classification methods.

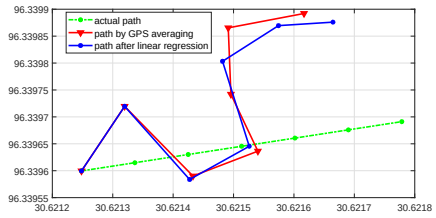
For comparison, we also implemented Kalman Filter [14] and computed the SST values for the paths obtained after using Kalman Filter. In our design, the prediction model for Kalman Filter, which utilizes the information from sensors to predict the position for next testing point, was established based on the velocity of the model car. The movement of the car was obtained using the velocity multiplied by time. Coordinate for next point was predicted based on the movement. The mean value of the velocity was set as actual speed as 1 m/s. The direction of the model car was also set as actual direction. The noise of the prediction model is Gaussian noise, with zero mean value and the standard deviation σ_a. σ_a was changed from $0.01m/s$ (1% error of actual speed) to $0.5m/s$ (50% error of actual speed) and then SST values were computed accordingly. The result is shown in Figure 9. The measurement model for the Kalman filtering is based on the GPS data. The result of measurement model is the average value of all GPS data.

The SST results for different classification methods were calculated in matlab and shown in Table I. We also show the SST values for Kalman filter under different noise of the prediction model. According to SST results, Path with SVM

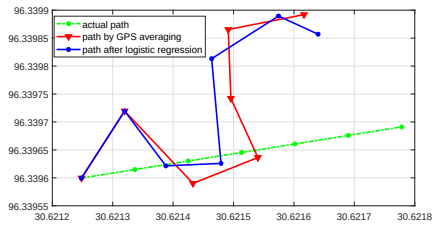
TABLE I. SST

	SST value	improvement from the path without classification
Path without any classification	3129.116	0%
Path with linear regression	2952.399	5.64%
Path with logistic regression	2925.190	6.51%
Path with SVM linear	2868.966	8.34%
Path with SVM RBF	3202.809	−2.3%
Path with BP nerual network	2887.315	7.73%
Path with Kalman Filter(σ_a=0.082)	2992.628	4.35%
Path with Kalman Filter(σ_a=0.086)	3233.508	−3.32%
Path with Kalman Filter(σ_a=0.1)	4044.701	−29.24%
Path with Kalman Filter(σ_a=0.2)	7249.907	−131.671%

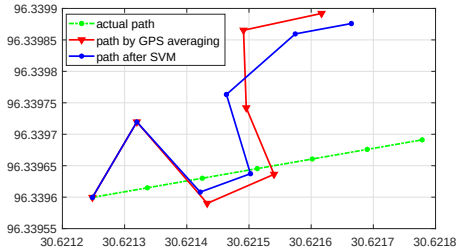
RBF shows a negative influence as this model classified all the GPS data at 6th testing point as “bad” data. We did not get any valid data after classification and can only use the average value of all original data. This may be the reason why SST for SVM RBF is so high. Except SVM RBF, all the classification methods show the positive influence, where linear regression shows the lowest 5% improvement. SVM with linear kernel obtains the lowest SST value and shows a 8% improvement



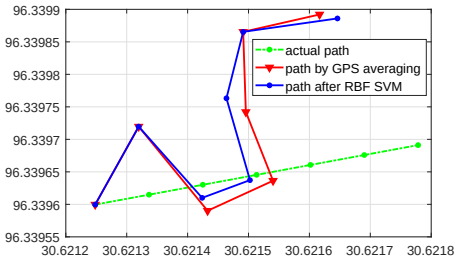
(a) linear regression



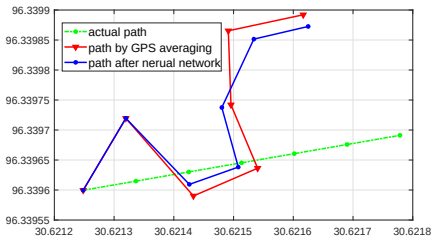
(b) logistic regression



(c) SVM linear



(d) SVM RBF



(e) BP nerual network

Fig. 8. Path curved by GPS with classification.Green line–actual path. blue line– path curved after classification. Red line–path by averaging the original data

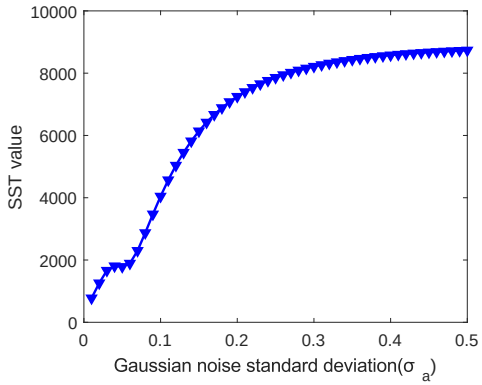


Fig. 9. SST values using Kalman Filter under different σ_a

from averaging all the GPS. The result of SST values prove that classification methods do detect some outliers, then remove them from the candidate set and record a more accurate movement path than just averaging all data. Although our classification methods show at most 7% improvement, they are much better than Kalman Filter when the noise of the velocity model reaches 8.2%. Kalman Filter even shows a negative influence on SST when the noise is bigger than 8.6%. Kalman Filter doubles the SST value when the noise reaches 20%. In addition, many hardware devices need to be fixed in order to collect the velocity information. However, our classification methods only consume some computation time of the controller.

F. Simulation Time

We also recorded the simulation time for each classification methods to train the model with 6000 GPS data and tested with 9000 GPS data in Matlab. The result is shown in Table II. The iteration for logistic regression is 50 and the number of neurons for BP network is 9 in one hidden layer. The

TABLE II. SIMULATION TIME

	time (s)
linear regression	0.8352
logistic regression	0.53576
SVM linear	0.9826
SVM RBF	0.9891
BP nerual network	1.691

simulation time indicates the extra time we need to spend if we want to do classification before averaging the data. According to table II, the time for neural network is obvious higher than others. The time for linear kernel and SVM RBF is close. logistic regression saves almost half time compared to SVM. linear regression saves 17.6% time compared to SVM and costs 35.8% additional time than logistic regression.

G. Analysis

A summary of classification accuracy for five methods is shown in Table III. With an extra time cost less than 2 seconds , we can classify at least 50% of the GPS data correctly. Given the SVM RBF did not output any good label at 6th testing point, it shows a negative influence on SST value. The other four methods can help us record at least 5% more accurate movement path. In order to evaluate the effectiveness of our classification methods, we also compare them to Kalman Filter. For SST values, our methods are more than 2% better than

TABLE III. TRAINING AND TESTING ACCURACY

	training accuracy	testing accuracy
linear regression	77.75%	51.56%
logistic regression	89.02%	59.82%
SVM linear	94.12%	60.19%
SVM RBF	97.06%	63.65%
BP nerual network	100%	66.91%

Kalman Filter when the noise of Kalman Filter reaches 8.2%. Kalman Filter will even show a negative influence on SST when the noise bigger than 8.6%. Our methods are better than Kalman Filter both on the SST values and hardware device cost. We then compare the five classification methods. BP neural network is more complicated to train. Although it shows 6% increase on testing accuracy compared to linear SVM, it costs 72% more time and shows a little higher SST. As a result, BP neural network, when compared to linear SVM, is not very suitable to be embedded into complicated vehicle communication system which requires time-efficient data processing mechanism. The logistic regression shows advantages on SST (0.9%), simulation time (35.8%) and testing accuracy (8.3%) compared to linear regression. Linear regression may be given up when we want to choose between logistic regression and linear regression. SVM RBF is better on testing accuracy and its simulation time is close to linear SVM. However, more test should be implemented to further prove its influence on SST. Logistic regression and linear SVM have their own advantages. Logistic regression is very time-efficient. It can save a lot of time and bring the least load to current communication system. The SST of linear SVM decreases a lot from logistic regression. If the time pressure is allowed, linear SVM shows a better performance on detecting the bad GPS data.

IV. CONCLUSIONS

This paper proposed totally five different machine learning classification methods to classify every GPS data we received. The model parameters should be updated with the collection of new training data at different places. According to our experiments, except SVM RBF, other four methods can help reduce the GPS curved vehicle movement path error caused by averaging all the data. A testing system in a line is established and thousands of GPS data are used to train and test our classification model. SST values are also computed for comparison. According to SST value, the GPS curved path after linear SVM classification obtains the lowest SST and can be improved 8.34% compared to the case without any classification. Our classification methods can record more accurate paths compared to Kalman Filter when its noise bigger than 8.2%. The logistic regression is the most time-efficient method, which almost spends half time than other four methods and thus put least pressure on the controller. It is suitable to be used in the period when in-vehicle communication system is quite busy. Although BP neural network has the highest testing accuracy, linear SVM is better based on its SST and simulation time (72%improvement from BP neural network). Linear SVM is also a good choice when the system has enough spare time. In the future, linear SVM and logistic regression may be combined and applied at different period, which can achieve a balance between time consumption and classification effectiveness.

In the future, more tests should be implemented to test our classification methods. The valid data loss occurred in SVM RBF should be studied and avoided. More technologies, like reinforcement learning and Particle filter should be considered to combine with the classification methods to further improve the performance and obtain a more satisfactory accuracy. The circuit size of our classification methods should be evaluated in future work.

REFERENCES

[1] E. Kaplan and C. Hegarty, *Understanding GPS: principles and applications*. Artech house, 2005.

[2] V. Di Lecce, A. Amato, and V. Piuri, "Neural technologies for increasing the gps position accuracy," in *Computational Intelligence for Measurement Systems and Applications, 2008. CIMSA 2008. 2008 IEEE International Conference on*, pp. 4–8, IEEE, 2008.

[3] M. Bahrami and M. Ziebart, "A kalman filter-based doppler-smoothing of code pseudoranges in gnss-challenged environments," in *Proceedings of the 24th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS 2011)*, pp. 2362–2372, INST NAVIGATION, 2011.

[4] C.-H. Wu, W.-H. Su, and Y.-W. Ho, "A study on gps gdop approximation using support-vector machines," *IEEE Transactions on Instrumentation and Measurement*, vol. 60, no. 1, pp. 137–145, 2011.

[5] X.-M. Chen, "Recursive local polynomial regression estimation and its applications," in *Control Conference (CCC), 2012 31st Chinese*, pp. 2043–2048, IEEE, 2012.

[6] M. Rallis and M. Vazirgiannis, "Rank prediction in graphs with locally weighted polynomial regression and em of polynomial mixture models," in *Advances in Social Networks Analysis and Mining (ASONAM), 2011 International Conference on*, pp. 515–519, IEEE, 2011.

[7] E. Wang, W. Zhao, and M. Cai, "Research on improving accuracy of gps positioning based on particle filter," in *Industrial Electronics and Applications (ICIEA), 2013 8th IEEE Conference on*, pp. 1167–1171, IEEE, 2013.

[8] A. Géron, *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems*. " O'Reilly Media, Inc.", 2017.

[9] A. V. Omelchenko and O. V. Fedorov, "Polynomial regression coefficients estimation in finite differences space," in *Radioelektronika (RADIOELEKTRONIKA), 2015 25th International Conference*, pp. 257–260, IEEE, 2015.

[10] H. Hirose, Y. Soejima, and K. Hirose, "Nnrmlr: A combined method of nearest neighbor regression and multiple linear regression," in *Advanced Applied Informatics (IIAIAAI), 2012 IIAI International Conference on*, pp. 351–356, IEEE, 2012.

[11] S. Bhama and H. Singh, "Single layer neural networks for linear system identification using gradient descent technique," *IEEE Transactions on Neural Networks*, vol. 4, no. 5, pp. 884–888, 1993.

[12] W. Lei and X. Qi, "The application of bp neural network in gps elevation fitting," in *Intelligent Computation Technology and Automation (ICICTA), 2010 International Conference on*, vol. 3, pp. 698–701, IEEE, 2010.

[13] C. N. Alam, K. Manaf, A. R. Atmadja, and D. K. Aurum, "Implementation of haversine formula for counting event visitor in the radius based on android application," in *2016 4th International Conference on Cyber and IT Service Management*, pp. 1–6, IEEE, 2016.

[14] G. Bishop, G. Welch, *et al.*, "An introduction to the kalman filter," *Proc of SIGGRAPH, Course*, vol. 8, no. 27599-3175, p. 59, 2001.