IEEE Congress on Evolutionary Computation, May 25-28, 2015, at Sendai International Center, Sendai, Japan.

Evolutionary data sampling for user movement classification

Iraklis Varlamis
Department of Informatics and Telematics
Harokopio University of Athens
Athens, Greece
Email: varlamis@hua.gr

***Abstract*—Smartphones are nowadays used for recognizing people's daily activities and habits, by collecting and analysing user activity information in real-time. In order to demonstrate this methodology, we have developed GPSTracker[1] a prototype application for Android phones, which collects position, speed, altitude and time information and performs real-time classification of user's movement. The GPSTracker application also uses geo-location information abouts Points Of Interest (POIs) such as bus or metro routes, parks and stadiums in order to improve the set of features used for the classification of a type of movement. In this work, we use evolutionary algorithms, in order to reduce the number of samples required for training our classifier, without loosing in classification accuracy. The resulting model, a) is able to provide personalized solutions, tuned to each individual users movement abilities, b) better adapts to unbalanced training data, due to the generation of training samples from the existing ones, c) performs an initial sampling of the training data, which reduces requirements for computational resources and improves the classification performance.**

I. Introduction

Most of the approaches that use the terms movement classification or motion classification usually refer to the problem of detecting body (e.g. standing, walking, lying etc) or face (e.g. smile, sleep, sad etc) movements. However, there is another set of approaches that associate movement classification to the problem of detecting the type of a users movement, or the means of transportation that he/she uses. Usually, such applications, use GPS signal data and are driven by the interest to predict the future position of the user e.g. in a cellular network. The aim of our research is more generic, focuses on the automatic detection of a users movement type in real time, and is the first step of a larger project, which aims to detect user habits in a larger time-scale.

The detection of user movement patterns is usually treated as a classification problem ([1], [2]), where a classifier is trained using a set of preclassified samples and a predefined set of features (such as body parts movement, changes in speed, altitude etc.). When a generic classifier is trained to classify a users movement based on a set of predefined features, the result is a generic model, which can hardly adapt to each individual user ability (e.g. to the speed of walking, driving attitude etc) or to temporal changes in the users movement habits. Our aim is to develop personalised solutions for each user, so we must train a different model for each user.

[1]A prototype of $GPSTracker$ for Android (versions 2.2 and newer) can be downloaded from http://galaxy.hua.gr/~it20934/.

The advanced capabilities of mobile phones, allow us to develop personalized solutions. However, there are still performance limitations. In a previous work on movement classification using mobile phones [3], we developed a prototype application for collecting training data for different types of user movement and consequently detecting future user movement types. In that work, the pilot data were collected using mobile phones, but a single movement classification model was build off-line. The resulting classification model was then loaded to the mobile device and used to detect the movement type of any user in the future. In this work, we extend this approach by adding the ability to train the classifier on the mobile phone, using personalised user data.

In the prototype application, during training, the user selects a type of movement and starts collecting data (gps position, speed, altitude etc), thus resulting to a new (training) annotated instance every 5 seconds. Changing to a different movement type, the user informs the application and continues to provide more training data.

The first problem of this approach is that the resulting training dataset can be unbalanced, since the duration of every user movement is not equal (e.g. he/she may drive for 20 minutes, then park and walk to the metro station for another 5 minutes and then move by metro for another 20 minutes. The second issue is the large number of training instances that is collected in a few hours (moving around during the day, say for 10 hours, results in 7200 instances). Due to these issues, in our initial implementation we carefully collected equi-balanced samples for all types of movement and we consequently created the models off-line, in a desktop computer. We subsequently trained a model by applying Weka Random Forest algorithm and the model file was loaded to the mobile device, together with the client application.

In this work, we use evolutionary algorithms, in order to reduce the number of samples required for training our classifier, without loosing in classification accuracy. The resulting model, a) is able to provide personalized solutions, tuned to each individual users movement abilities, b) better adapts to unbalanced training data, due to the generation of training samples from the existing ones, c) performs an initial sampling of the training data, which reduces requirements for computational resources and improves the classification performance. Both the sample selection process and the classification of future movement is performed on the mobile device.

According to our knowledge, this is the first time that evolutionary algorithms for undersampling have been applied to movement classification problems in order to reduce bias from large dataset size and class imbalance. The use of such algorithms in different classification problems [4] has shown promising results, so we decided to evaluate their use in the problem of user movement classification. The next section provides an brief overview of research works in movement classification and then focuses on the evolutionary algorithms that can be applied to the undersampling problem. The details of the algorithms that we evaluate are presented in the following section and the results of our experiments are presented in section V. The initial results of our research on the use of Clonal Selection algorithms to the classification of user movement are very encouraging, since they are better than any other algorithm that we tested before, using the complete set of training samples. Finally, section VI summarizes our findings and provides insights for future improvements.

II. RELATED WORK

Most works that tackle the user movement classification problem in indoor or outdoor environments, collect their training samples using motion detectors and wearable sensors that collect data from a limited set of actors. For example in [5], authors use the EM algorithm for hidden Markov models (HMM) in order to learn a movement model for a single person and predict the persons outdoor location into the future. Hidden Markov models and Bayesian inferences are applied in [6] and [7] to predict indoor and outdoor movement respectively.

When the problem upscales to large groups of users, that provide position information in real time then it becomes harder and probably inneficeint to develop solutions that take advantage of the full set of user-provided data. In our previous study on user movement classification [3], we collected user position information every few seconds and this resulted in a few thousands of training instances in a few hours period. The need for a dynamic model for user movement classification that will be able to adapt to each individual user move abilities (e.g. walking, running or biking speed, driving behaviour, public transport in different traffic conditions etc) and in the same time will make use of the most informative training samples. Having this in mind, we decided to examine the effect of data sampling in the performance of our classification algorithms.

Sampling techniques are frequently used to solve problems concerning the distribution of a dataset, e.g. data skewing or training set imbalance [8] and they involve artificially re-sampling of the data set. Sampling can be achieved either by Under-sampling the majority class, or by Oversampling the minority class, or by combining both techniques. Undersampling consists of reducing the data by eliminating examples belonging to the majority class with the objective of equalizing the number of examples of each class; and oversampling aims to replicate or generate new positive examples in order to gain importance. Apart from the traditional methods for under-sampling (e.g. random sampling [9], stratified sampling [10] etc) or over-sampling [11], which solve the imbalance problem in the preprocessing phase, there exist hybrid methods that combine both approaches and evaluate the effect of sampling in the classification model performance. Evolutionary Undersampling [12] is a method that based on a fitness function, selects the samples that optimise the classifier's performance.

Regarding the selection of a proper sample, there are two goals of interest in Evolutionary Undersampling according to Garcia & Herrera [12].

- Models that aim for an optimal balancing of data without loss of effectiveness in classification accuracy. Such models adopt the so called Evolutionary Balancing Under-Sampling approach.
- Models that seek for an optimal power of classification without taking into account the balancing of data, considering the latter as a sub-objective that may be an implicit process. Such models perform Evolutionary Under-Sampling guided by Classification Measures.

With respect to the types of instance selection that can be carried out in EUS, systems are distinguished:

- If the selection scheme proceeds over any kind of instance, then it is called Global Selection (GS). That is, the chromosome contains the state of all instances belonging to the training data set and removals of minority class instances (those belonging to positive class) are allowed.
- If the selection scheme only proceeds over majority class instances then it is called Majority Selection (MS). In this case, the chromosome saves the state of instances that belong to the negative class and a removal of a positive or minority class instance is not allowed.

According to evolutionary algorithms, the undersampling process is considered as a process of training samples selection from an imbalanced or very large training dataset. In order to solve this problem, every possible subset of instances is represented as a gene of size N, where N is the number of samples in the initial dataset and each position of the gene contains 1 if the respective training sample must be included in the final training set or 0 if not. This representation [13], allows the application of typical genetic algorithms' operation, such as mutation and crossover, without affecting the search space, and needs a fitness function in order to reach an optimal solution. The requirement from such a function is to select a solution (i.e. a subset S of the total training instances N) that increases the classification accuracy ($accuracy$) whilst increasing the percentage reduction ($reduc$) of the training sample size. So a general definition of the fitness of a chromosome which codes the subset S of training samples that will be used to build the model can be:

$$fitness(S) = \alpha \cdot accuracy + (1 - \alpha) \cdot reduc \tag{1}$$

where α is a parameter that defines the importance of accuracy achieved versus the percentage reduction of the size of the training set and $reduc$ is the percentage reduction defined as

$$reduc = 100 \cdot \frac{|TR| - |S|}{|TR|} \tag{2}$$

III. CLONAL SELECTION ALGORITHMS

Another interesting addition in the field of genetics inspired algorithms for the classification via undersampling is the family of Clonal Selection Algorithms which are inspired by the immune systems [14] of live organisms and their response to diseases of the body. The Clonal Selection Theory has been introduced by Burnet [15] and Jerne, in order to describe the functioning of acquired immunity, the ability of an organism to defend itself to a disease via the diversity of antibodies and the survival of those antibodies that better fit an antigen. In biology the immune system has distributed control over the body, operates in many parallel cases, and is adaptive to the needs of the organism. All these features are desirable for solving complex problems in the field of artificial intelligence so such systems become an inspiration for AI community. The clonal selection principle is used to explain the basic features of an adaptive immune response to an antigenic stimulus. According to this principle, only those cells that recognize the antigens are selected to proliferate. The selected cells are subject to an affinity maturation process, which improves their affinity to the selective antigens.

According to the Clonal Selection Theory [16], in a pre-existing group of lymphocytes (specifically B cells), a specific antigen only activates (i.e. selection) its counter-specific cell so that particular cell is induced to multiply (producing its clones) for antibody production. Learning in the immune system involves raising the relative population size and affinity of those lymphocytes that have proven themselves to be valuable by having recognized a given Ag. In analogy to classification with undersampling, we seek to solve the problem using a minimal amount of resources, so the samples that we select must be the ones that when asked to give a prediction (classification) will prove high affinity (i.e. prediction accuracy).

One of the most popular algorithms that implements the clonal selection principle is $CLONALG$ [17], [18]. The steps of the $CLONALG$ algorithm with respect to the aforementioned Clonal Selection Theory can be summarized in the following:

- maintenance of a specific memory set (i.e. training subset)
- selection and cloning of the most stimulated samples
- removal of the nonstimulated training samples
- affinity maturation (by mutation), and
- reselection of the clones proportionally to their antigenic affinity, generation, and maintenance of diversity.

The intuition of the algorithm is that instead of "starting from scratch" every time, we will be able to remove training samples of low affinity, and multiply those with high affinity in each iteration. Such a strategy ensures that both the speed and accuracy of the immune response becomes successively higher after each infection. This is a reinforcement learning strategy [19], where the interaction with the environment gives rise to the continuous improvement of the system capability to perform a given task.

Clonal selection algorithms have been proposed for pattern recognition but not been applied to classification problems [17], [20]. Recently, there are few classification algorithms derived from the concept of Artificial Immune Systems like $CLONAX$ [4], $AIRS$ [21] and $Immunos$ [22], which are providing promising results for many engineering problems. In this work, we compare two algorithms that implement the Clonal Selection principle, namely $CLONCLAS$ and $CSCA$. More specifically, we use the implementation of the algorithms for Weka data mining suite[2], provided by Jason Brownlee [3] and described in details in his technical report [23].

A. *CLONALG and CLONCLAS*

The main aspect of $CLONALG$ algorithm is to develop a memory pool of antibodies that represents a solution to an engineering problem. In this case, an antibody represents an element of a solution or a single solution to the problem, and an antigen represents an element or evaluation of the problem space. The algorithm provides two mechanisms for searching for the desired final pool of memory antibodies. The first is a local search provided via affinity maturation (hypermutation) of cloned antibodies. More clones are produced for antibodies with higher affinity, though the scope of the local search is inversely proportional to the selected antibodies rank. This allows antibodies with low specificity with the antigen, a wider area in the domain which to maturate. The second search mechanism provides a global scope and involves the insertion of randomly generated antibodies into the population to further increase the diversity and provide a means for potentially escaping local optima.

Overall, the implementation of $CLONALG$ for classification, namely the $CLONECLAS$ algorithm process the training sample as follows:

- In the initialization step the training dataset of size N (antibodies) is randomly partitioned into two subsets: the memory samples subset m and the remaining samples subset r. Each antibody in the memory subset is allocated a class, thus allowing it to perform classification by assigning its class to an antigen. When an unseen antigen is exposed to the population, it is allocated the class of the antibody with the highest affinity to the antigen.
    1) In a sequence of steps that are repeated for a predefined number of loops or until a convergence criterion is met, each antibody (i.e. training sample in subset m) is evaluated against a randomly selected antigen (i.e. a training sample, which is treated as unknown).
    2) The antibodies (training samples) are evaluated for their affinity (their ability to properly classify the antigen) and ranked based on their affinity score.
    3) The best training samples are selected for cloning (i.e. selected to be copied to the training sample after mutation) in proportion to their affinity, whereas the worst ones are removed from the training sample.
    4) During the Affinity Maturation step (i.e. mutation) the clones are mutated inversely proportional to their affinity. Clones of high affinity score are mutated less than clones of lower affinity score.

[2]http://www.cs.waikato.ac.nz/ml/weka/

[3]http://wekaclassalgos.sourceforge.net/

- After the completion of all generations, the memory m component contains the most promising training samples and their variations. This will be the final training set for the algorithm. Classification of unseen examples will consequently be performed using an affinity-competitive environment, and could even be implemented using a majority vote k-Nearest Neighbour approach.

B. Clonal Selection Classification algorithm (CSCA)

One of the limitations of $CLONCLAS$ variation of $CLONALG$ was that it was designed for binary character pattern recognition problems domain. $CLONCLAS$ described the need for generalisation through treating each antibody as an exemplar and maintained one antibody per-class, which is not practical for most classification problems. CLONCLAS used an affinity threshold which is an additional parameter to be specified. However, the threshold can be easily achieved naturally using a competitive population of antibodies.

The aim of $ClonalSelectionClassificationAlgorithm$ was to tackle the inefficiencies of $CLONCLAS$ and optimize the classification performance. For this reason, it used the principles of Clonal Selection in order to tackle the classification problem, and redefined the generic steps of the Clonal Selection Theory.

First of all, since supervised classification is a learning process where training samples are use to build the classification model, the CSCA algorithm starts with partitioning the training dataset into segments and uses all the samples in a single segment as evaluation samples E (antigens) and the remaining samples as training T (antibodies). This is a random selection that take place in the initialization step. Classification of an evaluation sample (antigen) is done using the majority class of its k-Nearest Neighbours, defined by Euclidian or Hamming distance.

Both antibodies and antigens belong to classes so the affinity of an antibody, which belongs to a specific class, is defined as its ability to correctly classify antigens of the same class (True Positives) and avoid misclassification of antigens of other classes to its class (False Positives) or antigens of its class to other classes (False Negatives). As a consequence a definition of the affinity score of an antibody (or a set of antibodies), used as training samples in an iteration of the algorithm will be either the accuracy (or average accuracy) or a function similar to the following:

$$Fitness(ab) = \frac{TP + TN}{FP + FN} \tag{3}$$

Based on the calculated fitness score, training samples are filtered and either removed from the set or selected for clonining and maturation according to the following simple rules :

1) The training samples with zero True Positives for their selected class and more than zero False Negatives for another class are switched to the class with most FN hits, or the first class indeed in the case of a tie and their fitness is recalculated.
2) Samples with a zero misclassification score are also removed from the population.
3) Training samples of low fitness (lower than a threshold value ϵ) are removed from the population and do not participate in cloning and maturation.

In the cloning step the number of clones for an antibody is analogous to its fitness (it is essentially its fitness ratio in the population) and defined by the following equation:

$$numClones(ab) = \left(\frac{Fitness(ab)}{\sum_{i \in S} Fitness(i)} \right) \cdot (n \cdot \alpha) \tag{4}$$

where n is the number of antigens and α is an optional scale factor.

In the mutation step, new attribute values are calculated for each selected antibody. The new value for numeric attributes is a random number in a range around the existing attribute value, whereas the range is inversely proportional to the antibody fitness ratio. The new value for nominal attributes are randomly selected.

The initial population size S, the number of generations G, the attribute value bounds and a random number seed r are the required paremeters of $CSCA$, whereas the minimum fitness threshold ϵ, the clonal scale factor α, the number of partitions p for large sets and the number of neighbours k are optional parameters that all default to 1.

IV. User Movement Classification

In a previous work [3] we presented a prototype application, which demonstrated the use of mobile phones for collecting user activity information. The aim of $GPSTracker$ prototype is to record and detect the daily routine of the user. Its architecture (shown in Figure 1) is based on four components, which implement data recording, classification of movement, visualisation of user tracks and data storage in the cloud respectively.

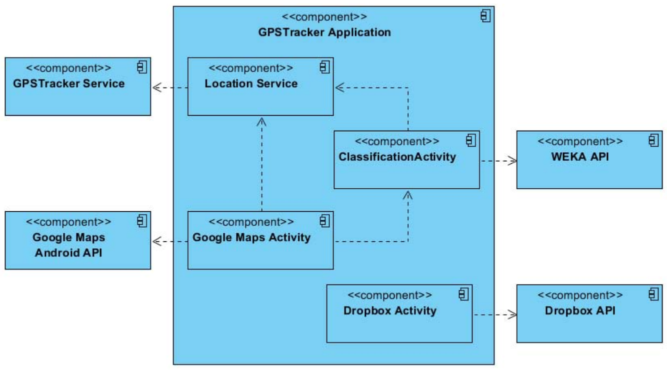

Fig. 1. The architecture of *GPSTracker*.

While extending the set of features employed for movement classification, we put effort in improving the classification results and in the same time in increasing the generalization performance of our solution. So, we first extended the set of features that we employ, by adding geospatial information for public transportation stops and routes as well as parks and stadiums. The complete set of features that we currently work on are presented in Table II. The attributes with blue color, are the new attributes for this work.

TABLE I. FEATURES

Basic	*Derived*
Longitude Latitude	Average speed Smoothed average speed Near Metro Station Inside a polygon (park, stadium etc.) On bus line On metro line
Altitude	Altitude change
Timestamp	Time zone Day of the week Hour of day Is working day
GPS Signal status	-

TABLE II. INITIAL SET OF FEATURES RANKED BY INFORMATION GAIN

$InformationGain$	$Feature$
2.3864	Timestamp
1.5827	Longitude
1.3724	Latitude
0.809	Average Speed
0.3756	GPS Signal Status
0.3584	Smoothed average speed
0.2995	Altitude
0.2777	Day of week
0.0252	Is working day
0	Time zone

A. Feature Selection

The first step in our analysis is to measure the contribution of each feature to the accuracy of the classification model. Using $InformationGain$ as a metric for evaluating attributes, we rank the attributes as follows:

Although Timestamp, Longitude and Latitude features (first zone features) show an increased value of Information Gain, they are in essence limiting the generalization power of our model, since they rely on very specific information, which can easily be biased for the training sample and lead to over fitting. So we decide to remove these attributes and build a classification model without them. We also build a second classification model that employs the newly added geolocation based attributes.

B. Unbalanced training set and under-sampling

A second observation from our initial work, was that the number of training instances that we collected was large, but imbalanced as shown in Table III.

In order to tackle this issue, we examine in this work the use of Clonal Selection algorithms in our multi-class classification problem. In the following section, we present our experimental evaluation methodology, which compares the performance of our currently best algorithm, with that of Clonal Selection algorithms with and without the additional features. The results justify the initial thought, that Evolutionary methods can be applied to the problem of user movement classification with under-sampling. The resulting accuracy is improved and the number of training samples employed is significantly reduced.

TABLE III. CLASS DISTRIBUTION OF THE TRAINING SAMPLES

Attribute	Number of training samples	Ratio of training samples
Walking	770	17%
Running	177	4%
Biking	343	8%
Driving	650	14%
Metro	1256	28%
Bus	534	12%
Motionless	788	17%

V. EVALUATION

The dataset of reference to our work, is a sample of 4518 instances distributed across the different movement types (see Table III above). The classification of every recorded instance is performed using algorithms available by the Weka API and the best results were recorded by the implementation of the Random Forests algorithm [24]. Random forests are an ensemble learning method for classification that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes output by individual trees. Random Forests algorithm outperformed all other tree-based classifiers that we tested such as C4.5 [25] and REP Tree, and the Logistic Model Tree (LMT) [26], [27].

A. Without location and time restricting features

In a first experiment, we removed the three location and time-specific attributes and run the same tree based classification algorithms, some more algorithms such as SVMs, Neural Networks and k-NN and two clonal selection algorithms $CLONALG$ and $CSCA$. All algorithms were tested using the default parameter settings. Concerning the parameters of $CSCA$, the default number of neighbours was used (1-NN) and the initial population size, selected to be 50. This resulted to a few more than 300 antibodies, which corresponds to a reduction rate of 90%. We also tested $CSCA$ with an initial population size of 500 and 1000, which result in a 87% training dataset reduction rate. Finally, we experimented with the minimum fitness threshold ϵ parameter and the clonal scale factor α. The minimum fitness threshold affects the data reduction percentage, since a lower fitness threshold results in the selection of more antibodies. The clonal scale factor affects the mutation of antibodies since a higher clonal scale factor results in bigger changes in the attribute values. The results with different ϵ and α values were almost similar to that of using the default values. The accuracy results when using 10-fold cross-validation on the entire data set are reported in Table IV.

TABLE IV. ACCURACY IN THE TRAINING DATASET USING 10-FOLD CROSS VALIDATION (7 FEATURES)

Algorithm	Accuracy
Random Forests	62.79±1.85
C4.5	**66.82**±1.8
REP Tree	65.52±1.82
LMT	66.40±1.81
1-NN	62.44±1.86
SVM	58.61±1.89
SMO	47.67±1.91
Multilayer Perceptron	55.33±1.91
CLONALG	23.82±1.63
$CLONALG_{1000}$	58.59±1.89
$CSCA_{\|S\|=50}$	63.15 ±1.85
$CSCA_{\|S\|=500}$	65.45 ±1.82
$CSCA_{\|S\|=1000}$	**65.78**±1.82

The results presented in Table IV differ from the results that we reported in our initial work, where all features of Table II

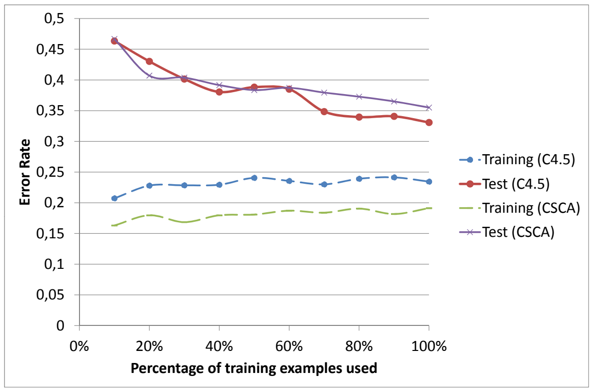

Fig. 2. Learning curves for C4.5 and CSCA, using an increasing number of training samples and 7 features.

were used and the Random Forest algorithm outpeformed all other algorithms with an accuracy higher than 90%. Accuracy scores are lower, which validates our first thought that the initial model will probably not generalize well and justifies the need for pruning the three attributes. A further examination of the results shows that $CSCA$'s performance is comparable to the best achieved performance by C4.5 with a confidence interval of 99%.

In order to evaluate the generalization performance of our model and make sure that we avoid over-fitting, we draw the learning curves of our best models using a holdout set, which we created by performing stratified sampling to the full dataset. The holdout (testing) dataset contains 20% of the full dataset instances and the remaining 80% was used for training. The error rate for the testing and training sets using an increasing number of training samples is depicted in Figure 2.

From the learning curves it is obvious that the classification model does not generalizes well in the case of C4.5. Although overall there is a decrease in the error rate on the held-out set when the training sample size increases, there are cases where the error rate increases or remains stable. On the contrary, $CSCA$ seems to generalize better, since the error rate decreases at all times, when the training set size increases, and this also signifies that we can use additional training samples in order to perform a better under-sampling before classification. Another explanation of the results, is that the error rate, both in training and held-out data is still large, so we must use additional features in order to improve classification. In this direction is the second experiment that follows.

B. *Using derived geo-location information features*

In this second experiment we repeat all the evaluations we performed in the previous experiment, this time using the additional attributes that have been derived by processing the user location information and transportation and park related information. The motivation behind this experiment was that in the initial evaluation dataset we noticed an imbalance on the accuracy provided for different classes. In order to have a better understanding of what mislead our classifier, we performed an error analysis on the holdout samples. The confusion matrix depicted in Table V shows that the confusion was mainly among Walking and Bus and Driving classes. This can be easily explained since the buses frequently stop, and similarly people occasionally stop during their walks (e.g. in traffic lights).

TABLE V. CONFUSION MATRIX OF THE TEST DATA SET.

a	b	c	d	e	f	g	←classified as
95	10	6	13	6	15	9	a = Walking
6	22	3	1	0	1	1	b = Running
7	0	52	1	1	4	6	c = Biking
18	7	0	67	15	16	7	d = Driving
11	2	1	12	200	4	20	e = Metro
7	5	8	12	4	67	4	f = Bus
14	2	4	4	45	5	84	g = Motionless

First we repeat 10-fold cross-validation on the entire data set and report the results in Table VI. Once again, C4.5 demonstrates the best performance among the evaluated algorithms of the first two zones (tree based and others).

TABLE VI. ACCURACY IN THE TRAINING DATASET USING 10-FOLD CROSS VALIDATION (10 FEATURES)

Algorithm	Accuracy
Random Forests	65.80±1.82
C4.5	**69.79**±1.76
REP Tree	67.13±1.8
LMT	69.10±1.77
1-NN	65.01±1.83
SVM	62.33±1.86
SMO	50.75±1.92
Multilayer Perceptron	58.88±1.89
CLONALG	21.74±1.58
$CLONALG_{1000}$	62.73±1.85
$CSCA_{\|S\|=50}$	63.97±1.84
$CSCA_{\|S\|=500}$	66.89±1.8
$CSCA_{\|S\|=1000}$	**69.97**±1.79

As far as it concerns the Clonal Selection Algorithms, the results are better than using fewer attributes, and slightly better to those of C4.5 (see Figure 3).

Another thing that validates the belief that the new attributes improve the classification performance is their ranking using Information Gain as depicted in Table VII. The feature that checks whether the user moves on a bus line is highly informative. The other two features (i.e. "On metro line" and "Inside a polygon") are not very helpful, at least in the current validation dataset.

The last thing to check is whether we have overfitting in our best two algorithms. As before, we plot the results of C4.5 and CSCA in the training and evaluation datasets for an increasing

TABLE VII. EXTENDED SET OF FEATURES RANKED BY INFORMATION GAIN

$InformationGain$	$Feature$
0.5495	Hour of day
0.1570	Average Speed
0.0992	On bus line
0.0960	Day of week
0.0929	Altitude
0.0778	Smoothed average speed
0.0090	Is working day
0.0056	GPS Signal Status
0.0046	On metro line
0.0044	Inside a polygon

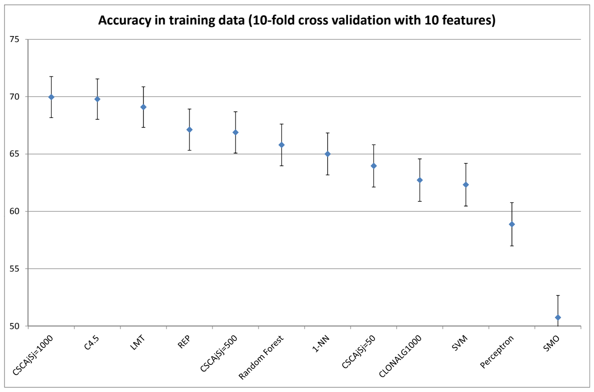

Fig. 3. Comparison of different algorithms using accuracy in the training dataset and 10-fold cross validation with 10 features.

number of training samples. The results are depicted in Figure 4.

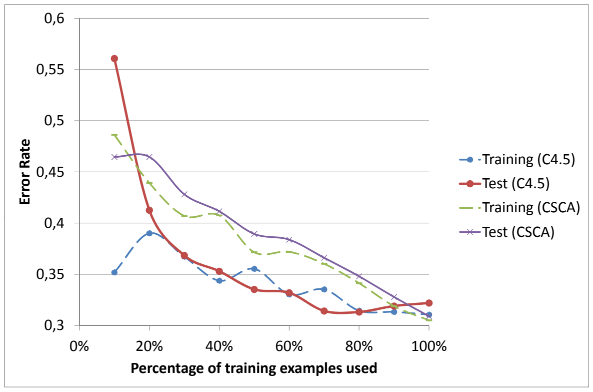

Fig. 4. Learning curves for C4.5 and CSCA, using an increasing number of training samples and 10 features.

Overall, the results in the evaluation dataset are comparable to those in the training dataset. However, the results of C4.5 do not demonstrate over fitting, except for the last two points of the held out data plot (i.e. for training samples larger than 80% of the available training dataset), for which the error rate in unknown samples increases. In the case of CSCA there is no over fitting and there is probably space for more improvement, if more features are employed.

VI. Conclusion

In this work, we presented an application of Clonal Selection algorithms to the problem of user movement type detection. The problem is a multi-class classification problem that requires personalised models to be build for each individual user, in order to be adaptable to the user's moving abilities and habits. The models must be trained using an equal number of training instances for each class and in case of a large training data size, a proper sampling must be performed. The application of CSCA and CLONALG to our problem shows that Clonal Selection algorithms can achieve comparable results to other supervised learning techniques, which are applied to the whole training dataset.

After the successful testing of CSCA in the sample experimental data, it is on our intention to adapt the existing implementation for Weka, to the environment of a mobile device and test it in real conditions, thus creating personalised classification models for the users of our $GPSTracker$ application. We currently work on adding more features concerning user activity, by taking advantage of more sensors provided by the mobile device and by adding more geo-location information (e.g. information about malls, pedestrian areas, etc.), so we expect to further improve the classification accuracy.

Acknowledgment

The author would like to thank Spyridoula Tragopoulou and Ivan Tarbakou for their contribution to the implementation of various aspects of the $GPSTracker$ application, as well as for their assistance in the collection of training and evaluation data.

References

[1] F. Sparacino, "The museum wearable: Real-time sensor-driven understanding of visitors' interests for personalized visually-augmented museum experiences." 2002.

[2] N. Bu, M. Okamoto, and T. Tsuji, "A hybrid motion classification approach for emg-based human–robot interfaces using bayesian and neural networks," *Robotics, IEEE Transactions on*, vol. 25, no. 3, pp. 502–511, 2009.

[3] S. Tragopoulou, I. Varlamis, and M. Eirinaki, "Classification of movement data concerning users activity recognition via mobile phones," in *Proceedings of the 4th International Conference on Web Intelligence, Mining and Semantics (WIMS14)*. ACM, 2014, p. 42.

[4] A. Sharma and D. Sharma, "Clonal selection algorithm for classification," in *Artificial Immune Systems*. Springer, 2011, pp. 361–370.

[5] D. J. Patterson, L. Liao, D. Fox, and H. Kautz, "Inferring high-level behavior from low-level sensors," in *UbiComp 2003: Ubiquitous Computing*. Springer, 2003, pp. 73–89.

[6] J. Petzold, A. Pietzowski, F. Bagci, W. Trumler, and T. Ungerer, "Prediction of indoor movements using bayesian networks," in *Location-and Context-Awareness*. Springer, 2005, pp. 211–222.

[7] D. Ashbrook and T. Starner, "Using gps to learn significant locations and predict movement across multiple users," *Personal and Ubiquitous Computing*, vol. 7, no. 5, pp. 275–286, 2003.

[8] N. V. Chawla, "Data mining for imbalanced datasets: An overview," in *Data mining and knowledge discovery handbook*. Springer, 2005, pp. 853–867.

[9] X. Guo, Y. Yin, C. Dong, G. Yang, and G. Zhou, "On the class imbalance problem," in *Natural Computation, 2008. ICNC'08. Fourth International Conference on*, vol. 4. IEEE, 2008, pp. 192–201.

[10] J. Neyman, "On the two different aspects of the representative method: the method of stratified sampling and the method of purposive selection," *Journal of the Royal Statistical Society*, pp. 558–625, 1934.

[11] A. Liu, J. Ghosh, and C. E. Martin, "Generative oversampling for mining imbalanced datasets." in *DMIN*, 2007, pp. 66–72.

[12] S. García and F. Herrera, "Evolutionary undersampling for classification with imbalanced datasets: Proposals and taxonomy," *Evolutionary Computation*, vol. 17, no. 3, pp. 275–306, 2009.

[13] L. I. Kuncheva and J. C. Bezdek, "Nearest prototype classification: clustering, genetic algorithms, or random search?" *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 28, no. 1, pp. 160–164, 1998.

[14] L. N. De Castro and J. Timmis, *Artificial immune systems: a new computational intelligence approach*. Springer, 2002.

[15] F. M. Burnet, “A modification of jerne’s theory of antibody production using the concept of clonal selection,” *CA: A Cancer Journal for Clinicians*, vol. 26, no. 2, pp. 119–121, 1976. [Online]. Available: http://dx.doi.org/10.3322/canjclin.26.2.119

[16] P. D. Hodgkin, W. R. Heath, and A. G. Baxter, “The clonal selection theory: 50 years since the revolution,” *Nature immunology*, vol. 8, no. 10, pp. 1019–1026, 2007.

[17] L. N. De Castro and F. J. Von Zuben, “Learning and optimization using the clonal selection principle,” *Evolutionary Computation, IEEE Transactions on*, vol. 6, no. 3, pp. 239–251, 2002.

[18] ——, “The clonal selection algorithm with engineering applications,” *Proceedings of GECCO*, vol. 2000, pp. 36–39, 2000.

[19] A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 1998.

[20] J. A. White and S. M. Garrett, “Improved pattern recognition with artificial clonal selection?” in *Artificial Immune Systems*. Springer, 2003, pp. 181–193.

[21] A. Watkins, J. Timmis, and L. Boggess, “Artificial immune recognition system (airs): An immune-inspired supervised learning algorithm,” *Genetic Programming and Evolvable Machines*, vol. 5, no. 3, pp. 291–317, 2004.

[22] J. H. Carter, “The immune system as a model for pattern recognition and classification,” *Journal of the American Medical Informatics Association*, vol. 7, no. 1, p. 29, 2000.

[23] J. Brownlee, “Clonal selection theory & clonalg–the clonal selection classification algorithm (csca),” *Swinburne University of Technology*, 2005.

[24] L. Breiman, “Random forests,” *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct. 2001. [Online]. Available: http://dx.doi.org/10.1023/A:1010933404324

[25] R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann Publishers, 1993.

[26] N. Landwehr, M. Hall, and E. Frank, “Logistic model trees,” *Machine Learning*, vol. 95, no. 1-2, pp. 161–205, 2005.

[27] M. Sumner, E. Frank, and M. Hall, “Speeding up logistic model tree induction,” in *9th European Conference on Principles and Practice of Knowledge Discovery in Databases*. Springer, 2005, pp. 675–683.