An Efficient Scheme for Prototyping kNN in the Context of Real-Time Human Activity Recognition

Paulo J. S. Ferreira[1](✉), Ricardo M. C. Magalhães[1](✉), Kemilly Dearo Garcia[2](✉), João M. P. Cardoso[1](✉), and João Mendes-Moreira[1](✉)

[1] INESC TEC, Faculty of Engineering, University of Porto, Porto, Portugal
{up201305617,up201502862,jmpc,jmoreira}@fe.up.pt
[2] University of Twente, Enschede, The Netherlands
k.dearogarcia@utwente.nl

Abstract. The Classifier kNN is largely used in Human Activity Recognition systems. Research efforts have proposed methods to decrease the high computational costs of the original kNN by focusing, e.g., on approximate kNN solutions such as the ones relying on Locality-sensitive Hashing (LSH). However, embedded kNN implementations need to address the target device memory constraints and power/energy consumption savings. One of the important aspects is the constraint regarding the maximum number of instances stored in the kNN learning process (being it offline or online and incremental). This paper presents simple, energy/computationally efficient and real-time feasible schemes to maintain a maximum number of learning instances stored by kNN. Experiments in the context of HAR show the efficiency of our best approaches, and their capability to avoid the kNN storage runs out of training instances for a given activity, a situation not prevented by typical default schemes.

Keywords: k-Nearest Neighbor · Classification · kNN prototyping · LSH · Human Activity Recognition (HAR)

1 Introduction

Human Activity Recognition (HAR) (see, e.g., [1]) aims to recognize the activities performed by humans through the analyses of a series of observations using sensors, e.g., carried by users.

k-Nearest Neighbor (kNN) [2] is one of the most popular classifiers used in HAR systems. kNN is based on lazy learning, which means that it does not have an explicit learning phase. Instead, it memorizes the training objects, keeping them in a buffer memory. The kNN popularity in classification problems comes from its simplicity and straightforward implementation. The main disadvantage

This work has been partially funded by FCT project POCI-01-0145-FEDER-016883.

of kNN is the necessity of high storage requirements in order to retain the set of training instances and the number of distances to be calculated [5]. This problem becomes more evident when incremental/online learning is used in devices with limited memory and computing capacity. In such devices, it is necessary to limit the number of instances kNN can store and provide efficient real-time schemes to substitute/update the training instances.

In this paper, we propose substitution schemes that, when it is necessary to store a new instance during the incremental learning phase, allow to keep the number of instances limited and at the same time maintain the distribution of the number of instances per activity, thus preventing, during the incremental phase, that certain activities run out of instances. In addition, the paper evaluates the substitution schemes and compares their results with the results of using the default scheme which substitutes the oldest instance stored.

This paper is organized as follows. Section 2 introduces some related work. Section 3 describes the implication of kNN and LSH [11] + kNN limitation as well as descriptions of the substitution schemes proposed. Section 4 describes the setup used in the experiments performed. Then, Sect. 5 shows experimental results and, finally, Sect. 6 draws some conclusions and summarizes future work planned.

2 Related Work

This section presents previous work using kNN for classification. The presented approaches use different methods to select and remove instances from the kNN buffer memory.

The method ADWIN (ADaptive sliding WINdowing) [9] monitors statistical changes in a sliding window. The window stores all the instances since the last detected change. The window is partitioned into two sub-windows of various size until the difference of their average error exceeds a threshold, depending on the size of the sub windows and a confidence parameter, a change is detected and the older window is dropped.

The method PAW (Probabilistic Approximate Window) [8], stores in a sliding window only the most recent and relevant instances according to a probabilistic measure. To decide which instance to maintain or discard, the algorithm randomly select the instances that is kept in the window, which makes a mix of recent and older instances.

In [8], ADWIN is coupled with kNN with PAW. In this case, the ADWIN is used to keep only the data related to the most recent concept of the stream and the rest of instances are discarded.

The method SWC (Sliding Window Clustering) [10], instead of storing all instances that fit in a sliding window (for representing both old and current concepts), stores compressed information about concepts and instances close to uncertainty border of each class. The clusters are compressed stable concepts and the instances are possible drifts of these concepts.

The approaches proposed in this paper are focused on simple schemes in order to avoid execution time and energy consumption overhead and bearing in mind the implementation of kNN in memory and computing power constrained wearable devices.

3 *k*NN Substitution Schemes

In our HAR system, features are extracted from the raw data from sensors, are normalized, and then input to a machine learning (ML) classifier, such as kNN. The classifiers were implemented using the MOA library [6], since it allows dealing with data streams and offers a collection of incremental ML algorithms.

The kNN used already includes its own substitution scheme. In this scheme (herein mentioned as "default"), whenever a new training instance arrives and the maximum limit of instances that kNN can store is reached, the new instance substitutes the oldest instance.

When the number of instances that can be stored by the kNN is limited and with the continued substitution of the oldest instance, whenever a new training instance arrives, it may happen that, due to this scheme, activities run out of training instances stored in kNN. This may cause classification errors when kNN tries to classify an instance for which it has none or insufficient training instances. Because of this, it is important to propose efficient substitution schemes able to prevent a given activity from running out of training instances stored by kNN.

The following eight simple, energy/computationally efficient, and real-time feasible substitution schemes are proposed and implemented:

- **Default:** In the kNN used (present in the MOA library [6]) when the instance limit was reached, whenever a new training instance arrives and needs to be stored, it replaces the oldest instance stored in kNN (regardless of the activity);
- **SS1:** randomly selects an instance and replaces it with the new instance of the same class;
- **SS2**: selects the oldest instance of a certain class and replaces it with the new instance of the same class;
- **SS3:** classifies the new instance. If the new instance is incorrectly classified, SS1 is applied. Otherwise the new instance is discarded;
- **SS4:** classifies the new instance. If the new instance is incorrectly classified, SS2 is applied. Otherwise the new instance is discarded;
- **SS5**: classifies the new instance. If the new instance is correctly classified, the scheme randomly selects to apply SS1 or to discard the new instance. If the instance is classified incorrectly it applies SS1;
- **SS6**: classifies the new instance. If the new instance is correctly classified, the scheme randomly selects to apply SS2 or to discard the new instance. If the instance is classified incorrectly it applies SS2;
- **SS7**: classifies the new instance. If the new instance is incorrectly classified, the new instance replaces its nearest neighbor;
- **SS8**: classifies the new instance. If the new instance is correctly classified, the scheme randomly selects to replace the nearest neighbor of the new instance by the new instance, or to discard the new instance. If the new instance is incorrectly classified, the new instance replaces its nearest neighbor.

*k*NN is a lazy learner, which means that it does not have an explicit learning phase, but instead stores the training instances. For each learning instance, a vector of features is stored in conjunction with its label (class). Imposing a maximum limit of N means that the *k*NN can only store N instances. For LSH [11] + *k*NN, a maximum limit implies that a maximum number of instances are stored in the hash tables of the method. In our experiments, we use the LSH parameters empirically selected in our preliminary experiments, i.e., 20 hash tables, 1 random projection per hash value, and 10 as the width of the projection.

4 Experimental Setup

4.1 Dataset, Feature Extraction and Normalization

On our experiments, we used PAMAP2 (Physical Activity Monitoring for Aging People) [7], a dataset where 18 different activities were recorded for 9 different individuals, each wearing 3 IMUs (Inertial Measurement Units) and a heart-rate monitor. Each IMU presents 3D-data from accelerometer, gyroscope and magnetometer.

For each sliding window with sensor data, and according to the number of axis of each sensor, several features are extracted. In the case of our prototypes the features are associated to 3D and 1D sensors. Each 1D and 3D sensor has, respectively, 2 (mean and standard deviation) and 10 features (x mean, y mean, z mean, mean of the sum of the 3 axes, x standard deviation, y standard deviation, z standard deviation, xy Pearson Correlation, xz Pearson Correlation, and yz Pearson Correlation) extracted. Additionally, all features were subsequently normalized by dividing each value in a vector with the vector's magnitude.

4.2 Setup

In the experiments presented in this paper, our HAR system begins with a model built offline using the N−1 users. Then, we emulate the use of the HAR system and the real-time behavior using data from a different user representing the actual user of the system. After preliminary studies, user 5 was selected to represent the user of the system. This selection was because user 5 is the one of the users with more activities collected and also being the user with the highest number of instances. A sliding window of 300 sensor readings and a 10% overlap were used. With these settings and the data of the remaining 8 users, 6186 training instances are considered for creating the offline model.

In these experiments, the maximum number of instances the *k*NN (k = 3) could store was limited to 100, 200, 500, and from 1000 to 8000 with increments of 1000. The limit of 8000 allows storing the entire original training set plus the instances for incremental training. In order to reduce the initial training set with 6186 instances, a random-based filter reduces the original training set for the various limits tested, maintaining the global distribution of instances per activity of the original set.

5 Experimental Results

5.1 Substitution Schemes Comparison

In these experiments, all substitution schemes presented above were evaluated varying the kNN and LSH kNN storage maximums. Figure 1 shows the results obtained.

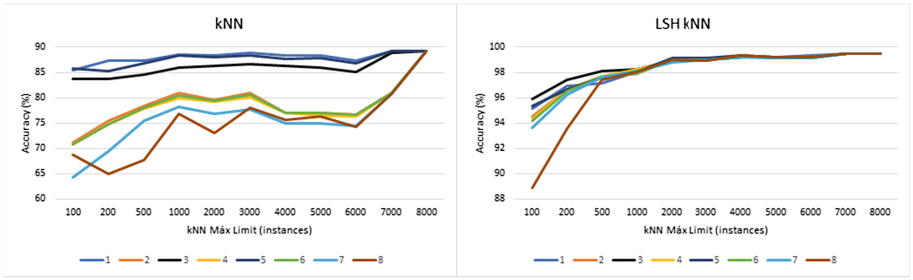

Fig. 1. Impact on accuracy of substitution schemes for the different limits considered: (a) kNN; (b) LSH kNN

According to Fig. 1(a), the best substitution schemes are 1, 3 and 5, with the scheme 1 showing slightly better results than the other two. The substitution scheme 8 is the one that presents the worst results of all. The results show a significant difference between the schemes with better results (1, 3 and 5) and those with worst results (2, 4, 6, 7 and 8). On average, there is a decrease of about 9% in accuracy between the best and worst schemes.

The convergence of all the lines of the chart to the same point is due to the fact that for 8000 all the learning instances are kept stored in the kNN. So, for the 8000 limit, substitution schemes are not applied.

According to Fig. 1(b), all substitution schemes behave similarly and differences in accuracy are minimal. However, scheme 3 achieves the best average accuracy ($\approx$ 98.56%), while scheme 7 presents the worst result ($\approx$ 98.17%). Globally, there is a difference of $\approx$ 0.39% on average.

Since all schemes achieve an already high accuracy, the substitutions done incrementally are not capable of improving or decreasing the accuracy of the classifier in a considerable amount on average.

The best scheme, 1, is compared with the methods kNN (default scheme), kNN with PAW ($k\text{NN}_P$) and kNN with PAW with ADWIN ($k\text{NN}_{PA}$). The results are present in Table 1.

The results presented in Table 1 show that all methods obtained very close accuracies, with $k\text{NN}_{PA}$ being the only one that got lower accuracy than our best scheme. However, none of the other methods guarantee that no class runs out of instances in the kNN feature space.

Table 1. Average accuracy for each method.

	kNN (SS 1)	kNN (default)	$k\text{NN}_P$	$k\text{NN}_{PA}$
Average accuracy	88.54	88.90	88.75	88.51

5.2 Proposed Substitution Schemes vs Default Scheme

In order to study the impact of the substitution schemes, an experiment was carried out where in one case the substitution scheme which obtained better results in the previous experiment was used and in the other case, no substitution scheme was used. In this experiment, the accuracy was measured throughout the incremental training, that is, the accuracy is calculated whenever a new instance is classified. Figure 2 shows the results obtained.

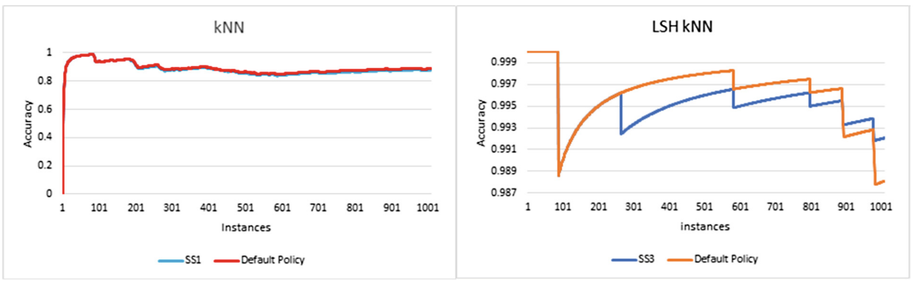

Fig. 2. Accuracy of kNN throughout the incremental training, using a maximum limit of 5000 instances, the best substitution scheme over the default scheme.

Analyzing the results of the left chart of Fig. 2, it can be concluded that the best proposed substitution scheme can match the results of the default scheme of kNN, failing to improve on the default scheme. Focusing on the right chart, we can conclude the proposed best scheme for LSH kNN is not capable of improving the default scheme for the most part, only achieving higher accuracy on the last classifications performed.

The limits used so far, were high enough to keep instances for all activities. In order to test the behavior in one extreme case of the default scheme as compared to the proposed substitute schemes, an experiment was performed where all 1008 instances of User 5 were used for incremental training and then used again for testing the accuracy of kNN. For this experiment, the maximum limit of instances that kNN can store has been set to 100, 200, and 500. PAMAP users data is organized by activity. Using the default scheme and such small limits, it turns out that in incremental training the kNN feature space will only save instances of the latest User 5 activities.

The results presented in Table 2 show that the default substitution scheme achieves very low accuracy. This is due to the fact that the default scheme always

Table 2. Impact on accuracy for the different limits and when considering online learning with longer time periods followed by classification of activities.

Substitution scheme	kNN limits			LSH kNN limits		
	100	200	500	100	200	500
1	68.25	86.81	91.27	86.21	86.41	85.12
2	57.04	64.68	73.91	77.78	73.91	74.11
3	70.83	85.81	89.78	97.42	99.50	99.80
4	57.54	64.38	73.91	91.17	98.41	96.53
5	67.16	85.81	90.67	88.00	87.40	90.18
6	57.44	64.19	74.01	84.42	85.91	87.10
7	58.23	68.25	74.11	94.84	97.92	98.81
8	54.96	63.10	73.71	80.46	83.63	87.00
Default	11.71	20.54	50.40	11.81	20.73	51.88

eliminates the oldest, leading to situations where a reduced number of activities is stored in the model. In this case, the kNN using the default scheme conducts to an incremental update of the model that in the end may represent only the latest activities used in the online training. This results in a misclassification of activities that do not correspond to those latest activities learned. In these situations, the use of the proposed substitution schemes allowed to considerably increase the accuracy of both classifiers, as substitutions occur at the activity level, thus preventing certain activities from running out of training instances stored and keeping the distribution of instances by activity.

As expected, the results show the importance of updating the model with instances of the user using the HAR system. This is the reason why the substitution scheme of the oldest instance in the model provides good results whenever the limit is enough or the sequence of activities do not override completely the model and make it not representative. This is what happens in the scenario forced by the learning of the instances of user 5 and then testing with the same instances.

6 Conclusion

Although kNN is one of the most used classifiers in Human Activity Recognition (HAR) systems, its storage requirements increase with the number of learning instances. This paper proposed schemes to substitute/update the set of learning instances for kNN classifiers in order to maintain a maximum number of training instances. The proposed schemes are evaluated in the context of a Human Activity Recognition (HAR) system. Eight different substitution schemes were proposed and tested to allow incremental online training when there is a maximum limit of instances that the kNN can store. This approach is useful in the case of implementing a HAR system in devices with limited memory (e.g., wearable devices). The experimental results show the efficiency of some of the eight schemes evaluated.

Ongoing work is focused on additional experiments with datasets from other domains. As future work, we plan to continue researching substitution/update schemes and compare with other more computational intensive schemes and verify the overheads regarding response time and energy consumption.

References

1. Zhang, S., Wei, Z., Nie, J., Huang, L., Wang, S., Li, Z.: A review on human activity recognition using vision-based method. J. Healthc. Eng. **2017**, 31 (2017). Article ID 3090343
2. Cover, T.M., Hart, P.E.: Nearest neighbor pattern classification. IEEE Trans. Inf. Theory **13**(1), 21–27 (1967)
3. Su, X., Tong, H., Ji, P.: Activity recognition with smartphone sensors. Tsinghua Sci. Technol. **19**(3), 235–249 (2014)
4. Calvo-Zaragoza, J., Valero-Mas, J.J., Rico-Juan, R.J.: Improving kNN multi-label classification in Prototype Selection scenarios using class proposals. Pattern Recogn. **48**(5), 1608–1622 (2015)
5. Garcia, S., Derrac, J., Cano, J., Herrera, F.: Prototype selection for nearest neighbor classification: taxonomy and empirical study. IEEE Trans. Pattern Anal. Mach. Intell. **34**(3), 417–435 (2012)
6. Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B.: MOA: massive online analysis. J. Mach. Learn. Res. **11**, 1601–1604 (2010)
7. Reiss, A., Stricker, D.: Introducing a new benchmarked dataset for activity monitoring. In: The 16th IEEE International Symposium on Wearable Computers (ISWC) (2012)
8. Bifet, A., Pfahringer, B., Read, J., Holmes, G.: Efficient data stream classification via probabilistic adaptive windows. In: Proceedings of the 28th Annual ACM Symposium on Applied Computing, pp. 801–806. ACM, March 2013
9. Bifet, A., Gavalda, R.: Learning from time-changing data with adaptive windowing. In: Proceedings of the 2007 SIAM International Conference on Data Mining, pp. 443–448. Society for Industrial and Applied Mathematics, April 2007
10. Garcia, K.D., de Carvalho, A.C.P.L.F., Mendes-Moreira, J.: A cluster-based prototype reduction for online classification. In: Yin, H., Camacho, D., Novais, P., Tallón-Ballesteros, A. (eds.) IDEAL 2018. Lecture Notes in Computer Science, vol. 11314, pp. 603–610. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03493-1_63
11. Indyk, P., Motwani, R.: Approximate nearest neighbors: towards removing the curse of dimensionality. In: Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing, pp. 604–613. ACM (1998)