Fault recovery mechanism for smart city environments

Lucian Sasu
Siemens Corporate Technology Romania
and Transilvania University of Braşov
Email: lucian.sasu@siemens.com

Dan Puiu
Siemens Corporate Technology Romania
Email: dan.puiu@siemens.com

Septimiu Nechifor
Siemens Corporate Technology Romania
Email: septimiu.nechifor@siemens.com

ABSTRACT

In this paper we present the work done to develop a fault recovery component for the situation when the quality of the data streams from a smart city environment drops. The fault recovery component is able to generate estimated values when the stream generates invalid or missing observations. The component is easy to configure and deploy for a large number of data streams from a smart city environment. As a result of that, the fault recovery component contains an incremental learning model, which is able to train itself during component execution. The paper describes the architecture of the fault recovery component and the result of the evaluation done for three candidate incremental learning models.

I. Introduction

Worldwide and with fast growing interest at European level, an increasing number of cities have introduced Information and Communication Technology enabled services with the scope of improving the operation of various infrastructure assets (e.g. traffic infrastructure, water distribution infrastructure, electrical energy grids) [1]. These data are generated by a wide and heterogeneous range of devices, and they are delivered under various forms and formats. There are a lot of custom made applications or frameworks used by the public companies and by city administration for monitoring the assets.

On the same time, there are an increasing number of initiatives coming from the European cities that support the idea of open data that are made available to citizens and public or private companies. This fuels a new range of applications which can be developed considering the citizens' or companies' needs. Based on this trend, the newly developed applications have to be flexible and to allow migration from one city to another. Examples of such cities are Aarhus (Denmark) or Santander (Spain). ODAA [2] is the Aarhus open data platform.

On the other hand, the reliability of smart city applications is an eminent property to achieve growing acceptance by citizens. This results in a high demand to monitor the quality of the data sources and also a demand for developing compensatory mechanisms in case of failures. As a result of that, the Quality of Information (QoI) is an important issue to take into account in smart city data analysis.

In parallel to this trend, the research community in the Internet of Things (IoT) domain is working to develop various generic frameworks and platforms which support data collection, data processing and application development. Examples of such frameworks are CityPulse [3] or OpenIOT [4]. In order to be reliable and accepted by the citizens, the frameworks must integrate mechanisms for monitoring the quality of data and for providing fault recovery solution if the data quality drops.

CityPulse is a framework for large-scale data analytics which provide information in (near-) real-time, transforms raw data into actionable information, and to enable creating "up-to-date" smart city applications [3]. The CityPulse components can be divided in two main categories as follows: Large scale data stream processing modules: representing the tools which allow the application developer to interact with the heterogeneous and unreliable data sources from the cities; and Adaptive decision support modules: containing the tools which can be used for making various recommendations based on the user context and the current status of the city.

The fault recovery (FR) is the property that enables the system to recover in the event of failure and to continue the operation properly. There are no standard recovery methodologies that can be applied, and the algorithms are implemented based on the possible failure directions. In case of CityPulse framework, the FR will allow the application to continue the operation even if the QoI of the streams drops. This is archived by generating estimated events, using incremental learning models.

The CityPulse framework is developed to operate in highly dynamic environments with large number of heterogeneous data sources such as smart cities. In addition to this the framework can be extended by adding new data sources with small intervention from the domain expert or data scientist. This implies that the FR component should be straightforward to configure and deploy.

In order to generate the estimated events the FR component has to integrate a machine learning (ML) based model. Usually the development of a prediction model implies the collection of a historical data set for the selected data source, and then searching for the best model (hyper-) parameter configuration. Because the city environments have a large number of data streams, the development of a prediction model for each data source is rather tedious. Consequently, the prediction models embedded into the FR component must expose incremental behaviour which allows them to learn in real time, when the quality of the data for the considered source is high.

The only configuration which can be done by the third party application developer is to select the model type and the configuration hyper-parameters. Depending on the domain (e.g. traffic, pollution or weather) different prediction models might be considered. In order to discriminate which model (with what hiper-parameter configuration) should be used for a certain domain, the third party application developer will have to collect a few dataset from the targeted data sources. Having the historical data sets, she can perform a form of model evaluation in order to determine the best model type and fitting hyper-parameters. Once the model type and its hyper-parameters are selected, it can be trained on the historical data sets, then deployed for all the data sources from that domain. Finally, the model will be used to learn and make predictions for FR.

In this paper we assess the capability of three different incremental algorithms to operate within the FR component, based on the conditions explained above. The considered algorithms are: Stochastic Gradient Descent for Regression, Online Passive–Aggressive Regressor, and a custom made incremental learning algorithm using k-nearest neighbours [3].

The paper is structured as follows. Section II briefly presents the main incremental learning solutions. After that section III introduces the extended description of the FR requirements and the evaluation scenarios. In section IV the architecture of the FR component and the results of the model assessment activity are presented. The conclusions of this work are given in section V.

II. Incremental learning systems

Incremental learning allows continuous adjustment of the parameters, based on data coming from a stream. Moreover, an incremental learning model is able to make predictions (classes or continuous values) at any point. The main requirements for incremental learning (also called on–line learning) are: *a)* it improves knowledge based on new data; *b)* the model does not request access to original data, used to train the system so far; *c)* it preserves previously acquired knowledge, and *d)* the model can afford new categories which can be introduced by new training data [5], [6].

There is a large overlap between incremental learning and domain of data stream mining, which is concerned with “extracting knowledge structures represented in models and patterns in potentially infinite streams of data” [7]. In both cases, the strong requirement is that a memory- and time-limited system should be able to process the data streams.

Plenty of models were developed for incremental learning: Support Vector Machines were extended by Cauwenberghs and Poggio in [8], by developing a recursive algorithm; additionally, the resulted model is also able to “unlearn” previously seen training patterns, speeding up leave–one–out based performance assessment step.

Virtually any major class of learning models are considered as candidate for incremental learning: Decision trees were further adapted e.g. in form of Hoeffding tree ([9], [10]), a decision tree induction algorithm which learns from large data streams with stationary generating distribution. Shallow neural networks were employed by Pérez-Sánchez *et al.* in [11] and by Polikar *et al.* in [5]; neuro–fuzzy approaches are discussed in [12], [13], [14]; some of Bayesian models (e.g. Naive Bayes) naturally exhibit on–line learning capabilities, and other methods are proposed: in [15] Lee and Eskenazi propose a sparse Bayesian learning method to allow continuous dialog strategy learning from the interactions with real users; Dimkovski and An [16] develop a novel Bayesian inference method which can run incrementally on a data stream, and significantly outperforms particle filters in a Bayesian neural network application. Incrementally generation of ensemble models is considered, for example, in [17], [18], [19].

Deep learning models are recently studied in context of incremental learning: convolutional neural networks [20], deep belief networks [21], denoising autoencoders for incremental feature learning [22].

Besides the speed of learning, another strength of incremental learning models is concept drifting: changes in data distribution should – and can – be quickly acknowledged by the prediction model; some models specifically target this issue, *e.g.* the approaches proposed in [23] and [24] based on fuzzy sets and fuzzy rules, or deep belief networks [21], [25] based on graph learning, and [9], [26] studying decision trees.

Any incremental learning system should take into account the so–called stability–plasticity dilemma: a learning model should expose plasticity, *i.e.* it should be able to gain new knowledge from new data; it should also be stable to preserve the previously acquired knowledge – stability. Too much stability makes the system conservative regarding new experiences, while too much plasticity could lead to forgetting, *e.g.* catastrophic interference in case of artificial neural networks. There is at least a solution to this dilemma, e.g. through adaptive–resonance theory developed by Carpenter and Grossberg [27].

Beyond theoretical studies and quantitative assessments, incremental learning was put in charge in various domains: for IoT a recent study is [12], where Wang *et al.* propose an incremental model based on probabilistic neural networks and adjustable fuzzy clustering for human activity recognition; also for IoT, in [28], Wenjuan *et al.* propose an incremental learned face detection method based on auto-captured samples. In [29] incremental learning with selective memory helps in localizing the prostate in image–guided radiotherapy, with fast retrieval in real–world clinical requirements; Tsumoto and Hirano [30] propose a new framework for incremental learning based on incremental sampling scheme and evaluate on datasets regarding headaches. Akcakaya *et al.* [31] develop an incremental model for radar–based target detection in nonstationary environments, including a drift detection step.

III. Scenario description and evaluation strategy

The main objective of the FR component is to increase the stability of the system by providing estimated measurements when the quality of a stream is modified (in most cases when the quality drops).

The FR component has two main requirements, as follows:

- has to be generic in order to be able to deploy it for any numeric data source, regardless of the domain;
- has to be easy to deploy (the third-party application developer who is going to use the component should deploy it without having to configure the embedded incremental learning algorithm).

In addition to this the FR component should not be processor intensive because for each data stream from the system an instance of the component has to be deployed. This due to the fact that, in smart cities environments it is highly possible that the applications have to handle more than 1000 data sources. Not in the last place, the precision of the FR instances it is very important.

As a result of that, the main key performance indicators used to evaluate the component are:

- prediction precision, measured through mean absolute error (MAE)
- the processing times needed to learn a new experience and to produce a prediction;

We have used a wide set of heterogeneous data sources, having different distributions, sampling rates, and data variabilities, which include:

- Aarhus traffic data: provides real-time observations about the number of cars (AC) and average speed (AS) from 490 locations of the city (sampling frequency: 5 minutes);
- Aarhus parking data (AP): provides real-time observations about the number of cars from 13 parking garages (sampling frequency: 1 minute);
- Romanian weather data: provides real-time observations about the temperature (RT), the humidity (RH), the wind speed (RW) and pressure (RP) from 160 locations (sampling frequency: 1 hour).

The data sets used can be downloaded from the CityPulse web page: http://www.ict-citypulse.eu/page/content/tools-and-data-sets .

The incremental learning ML model types considered for this work are:

- Stochastic Gradient Descent for Regression (SGDR), a linear model minimizing various regularized empirical loss functions;
- Online Passive–Aggressive Regressor (OPAR) [32];
- A custom made incremental learning algorithm using k-nearest neighbours, which was custom made developed for the purpose of this work (IKNN) [3]).

The three models are based on the Scikit-learn implementations [33]. To make the paper self–contained, a description of IKNN is provided in the next section.

IV. Fault recovery component implementation and evaluation

Figure 1 depict the architecture of the FR component.

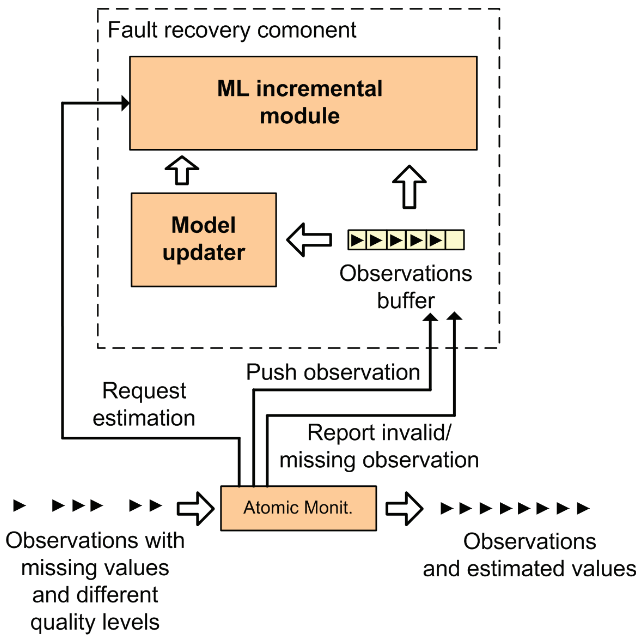
Fault recovery comonent
ML incremental module
Model updater
Observations buffer
Push observation
Request estimation
Report invalid/ missing observation
Atomic Monit.
Observations with missing values and different quality levels
Observations and estimated values

Figure 1. Fault recovery architecture.

The atomic monitoring component is responsible to process the stream of events coming from the data source and to asses the data quality. Based on the assessment and on an internal rating algorithm the QoI for the data stream is calculated. This component is integrated in the CityPulse framework and is fully explained in [3].

The atomic monitoring component interacts with FR one via the following commands:

- *Push observation*: all the stream observations are sent to the FR component when the quality of the stream is high (using this data the incremental learning model is trained);
- *Request estimation*: the estimation is requested when the quality of the stream is low;
- *Report invalid/missing observation*: the component is notified when the QoI os the stream is low (in this way the incremental training process is suspended).

The observations buffer is implemented as a vector, which holds the latest observations generated by the data stream, using the command “Push observation”. A null value is included in the buffer when an invalid or missing observation (using the command “Report invalid/missing observation”) is reported. Every time when the buffer is full (without having null values) it means that there is a complete sequence of valid consecutive observations.

Next the Model Updater module (see Figure 1) monitors the observation buffer. When the module detects that the buffer is full it triggers the partial fit process of the ML incremental model (using the observations from the buffer). In this way the incremental model acquires more experiences.

The Atomic Monitoring component requests estimations from the ML incremental module when there is a missing observation or when QoI of the stream is low. The model generates the estimation based on the sequence of observations from the buffer. The sequence of observations represents more precisely the evolution of the monitored parameter (e.g. traffic) in the last time.

A. *KNN based incremental learning model*

As mentioned above we have implemented an incremental learning model using KNN. In figure 2 the main components of this model are depicted.

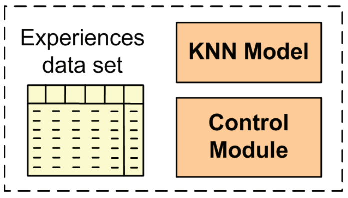

Figure 2. KNN based incremental learning model.

The KNN model block from figure 2 implements the traditional KNN algorithm as it is presented in [34]. In order to make an estimation the KNN model uses the experiences data set.

The control module is responsible on one hand to modify the experiences dataset in order to store the last W valid sequences of observations, stored in rolling window. In our case W represents the rolling windows size and was set to 100. On the other hand, the module is responsible to train in real-time the KNN models (as a result of experiences data set modification) and to trigger it when predictions are needed.

B. *Training and validation procedure*

Each data set is scaled before being used, as SGDR and OPAR tipically require scaled values for training. For each specific data two boundary values were used, either based on the physical limitations of the measured quantities (e.g. humidity always stays between 0% and 100%, or the range of a temperature sensor is known in advance), while for other cases we considered some common–sense ranges (e.g. the average speed of a car in the city is normally at most 200 km/h). We then considered for each of the resulted scaled time series a sequence of n gapless measurements ($2 \leq n \leq 20$), where the first $n-1$ values were used as predictive inputs, and the last value in the sequence was the one to be predicted; the variable n is a shared hyper-parameter for each of the three models considered and is referred as the buffer length in the beginning of section IV and Figure 1.

For the testing stage, the last 30% at the end of each series is reserved and not used during model choice step; these values are used for final performance evaluation. For both model choice and model assessment stages we employ the rolling–origin–recalibration strategy [35], which allows one to include the true value in the training set, after a prediction is made for it and the current error value is updated. This approach is cheap for our incremental models, as we do not require access to the whole previoulsy used training data to recalibrate the models. Note that although standard cross validation is empirically found as being consistent for time series forecasting domain despite breaking the intrinsic ordering of the values [36], we still find more natural to use just the time series' final parts for model choice and assessment.

For each of the three models we considered the following specific hyper-parameters, along with the shared n previosuly mentioned:

- for SGDR, the penalty used for regularization ('penalty') was tried as L_1, L_2 and elastic net; the initial learning rate (η_0) swept the set $\{0.001, 0.01, 0.1, 0.2, 0.3, 0.5\}$; the 'learning_rate' hyper-parameter was either considered constant during training, or updated according to the Scikit-learn policies named 'constant' and 'optimal'[33];
- for OPAR, the maximum setp size C swept the set $\{0.5, 1, 2, 5, 10, 100\}$; the 'loss' function was tried successively as Scikit–learn variants 'epsilon_insensitive' and 'squared_epsilon_insensitive';
- for IKNN, the number of neighbors ('n_neighbors') is varied between 3 and 9, the metric was Minkowski distance with $p \in \{1, 2, 3\}$, and the experiences data is a rolling window of size W, at most 100.

For both SGDR and OPAR a single training epoch was used, *i.e.* n_iter=1. Other unmentioned hyper-parameters were kept with the defaults valued from Scikit–learn, for all the three models.

For each of the three candidate models in turn we used every possible combination of the hyper-parameter values, using the training set consisting of the first 70% of the datasets. A minimum amount of data was used to perform an initial train of models (20 rows$\times n$ values), then the remaining values were used according to the rolling–origin–recalibration strategy. For each combination of hyper-parameter values we computed the MAE and retained those yielding the lowest score. Finally, the whole training set is used to train the models with the corresponding best values of the hyper-parameters and finally used to report the performance on the test set, also following rolling–origin–recalibration strategy.

C. *Experimental results*

Table I contains for each of the 7 datasets the model and its hyper-parameters producing the lowest MAE for the training subset, the MAE computed over the test subset and the time elapsed on preparing the best prediction model on the whole training set.

Some remarks on the quantitative results in Table I follows:

- Only SGDR and IKNN are winning models; OPAR produces MAE values which a larger than the best score by a factor of at least two (results not included);

Data set	Train MAE	Test MAE	Best model and hyper-parameters	Training time (s)
AC	0.039	0.039	SGDR, $n = 3$, $\eta_0 = 0.2$, learning_rate = constant, penalty=L_2	0.001
AS	0.034	0.034	IKNN, $n = 17$, n_neighbors=3, $p = 1$	0.614
AP	0.002	0.002	IKNN, $n = 13$, n_neighbors=3, $p = 1$	1.164
RT	0.001	0.002	SGDR, $n = 3$, $\eta_0 = 0.01$, learning_rate = constant, penalty=elastic net	0.002
RH	0.056	0.058	IKNN, $n = 13$, n_neighbors=3, $p = 1$	0.060
RW	0.034	0.038	IKNN, $n = 20$, n_neighbors=9, $p = 1$	0.038
RP	0.002	0.003	SGDR, $n = 3$, $\eta_0 = 0.5$, learning_rate = constant, penalty=elastic net	0.001

Table I
RESULTS FOR THE 7 DATA SETS. THE TRAINING TIME IS COMPUTED AFTER THE BEST MODEL AND ITS CORRESPONDING HYPER-PARAMETERS ARE OBTAINED. THE ROLLING–ORIGIN–RECALIBRATION WAS USED BOTH FOR TRAINING AND TESTING PHASES.

- IKNN delivers the best performance in 4 cases out of 7, with the price of an increased training time; for the same cases, SGDR requires a few milliseconds training time (results not included);
- IKNN works better with large values of n (13, 17, 20), while SGDR produces its best results small n — 3 in all reported cases;
- IKNN produces its best values always for L_1 distance, which is computationally less demanding than L_2 and L_3; SGDR works best with constant learning rate, which also decreases the computational burden;
- In all 7 cases, the MAE values on the training set are close to the ones computed for the testing set, *i.e.* the train MAE is empirically found as a good approximation for the production–time performance;
- Although in our experiments we used scaled data for IKNN in order to compare the three models over the same data, it worths to emphasize that scaling is not actually necessary for IKNN.

Figure 3 shows the MAE variation for different buffer lengths using AP dataset (it contains roughly 27000 observations). The training time using the same data set is presented in Figure 4.

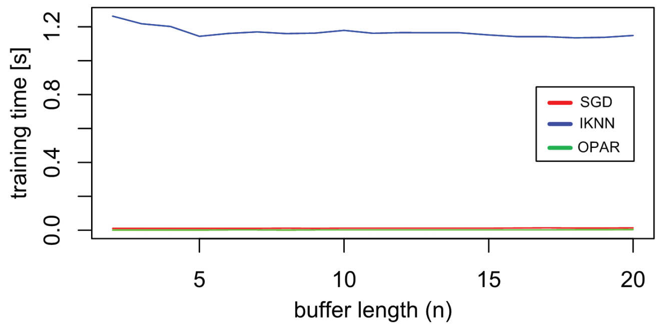

Figure 3. MAE variation for different buffer lengths using AP dataset.

V. CONCLUSIONS

Smart cities are composed of frameworks that support a variety of applications, making use of heterogeneous data sources. These data sources are coming mostly from IoT devices such as sensors, smart phones, cameras and also citizen sensing data collected from social media. Evaluating the quality of these sources is a mandatory process if reliable smart city applications should be provided. As a result of that, the frameworks can expose to third party applications developers fault recovery solution if the data quality from a stream drops.

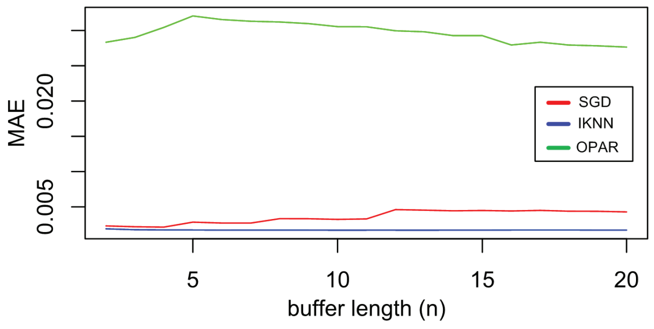

Figure 4. Training time variation for different buffer lengths using AP dataset.

The fault recovery component presented in this paper is integrated into the CityPulse framework [3]. The component is able to train it self when the quality of the stream is high. After that, when the stream fails to deliver a valid observation, the component is triggered to generate an estimated value. The process of monitoring the quality index of the streams is ensured by the atomic monitoring component of the CityPulse framework, as shown in [3].

Three candidate incremental machine learning algorithm have been considered to be integrated in the fault recovery component. The algorithms have been evaluated using seven historical data sets which cover traffic, parking and weather domain. Local learning models (KNN-based) and linear models trained with stochastic gradient descent are found as performing similarly when compared on prediction accuracy score, but quite different when discussing the training speed.

The implementation of the fault recovery component using IKNN model can be found here: https://github.com/CityPulse/FaultRecovery.

ACKNOWLEDGMENT

This work is supported by the EU FP7 CityPulse project under grant No. 603095, http://www.ict-citypulse.eu

REFERENCES

[1] O. C. International, “Smart cities in europe enabling innovation,” Osborne Clarke, Tech. Rep., 2015. [Online].

Available: http://smartcities.osborneclarke.com/general/smart-cities-in-europe-enabling-innovation-report/

[2] ***, "Odaa - open data aarhus." [Online]. Available: http://www.odaa.dk

[3] D. Puiu, P. Barnaghi, R. Tönjes, D. Kümper, M. I. Ali, A. Mileo, J. X. Parreira, M. Fischer, S. Kolozali, N. Farajidavar, F. Gao, T. Iggena, T. L. Pham, C. S. Nechifor, D. Puschmann, and J. Fernandes, "Citypulse: Large scale data analytics framework for smart cities," *IEEE Access*, vol. 4, pp. 1086–1108, 2016.

[4] J. Soldatos, N. Kefalakis, M. Hauswirth, M. Serrano, J.-P. Calbimonte, M. Riahi, K. Aberer, P. P. Jayaraman, A. Zaslavsky, I. P. Zarko, L. Skorin-Kapov, and R. Herzog, *Interoperability and Open-Source Solutions for the Internet of Things: International Workshop, FP7 OpenIoT Project, Held in Conjunction with SoftCOM 2014, Split, Croatia, September 18, 2014, Invited Papers*. Springer International Publishing, 2015, ch. OpenIoT: Open Source Internet-of-Things in the Cloud, pp. 13–25.

[5] R. Polikar, L. Udpa, S. Udpa, S. Member, S. Member, and V. Honavar, "Learn++: An incremental learning algorithm for supervised neural networks," *IEEE Transactions on System, Man and Cybernetics (C), Special Issue on Knowledge Management*, vol. 31, pp. 497–508, 2001.

[6] R. Polikar, L. Udpa, S. Udpa, and V. Honavar, "An incremental learning algorithm with confidence estimation for automated identification of nde signals," *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 51, no. 8, pp. 990–1001, Aug 2004.

[7] M. M. Gaber, A. Zaslavsky, and S. Krishnaswamy, "Mining data streams: A review," *SIGMOD Rec.*, vol. 34, no. 2, pp. 18–26, Jun. 2005.

[8] G. Cauwenberghs and T. Poggio, "Incremental and decremental support vector machine learning," in *Advances in Neural Information Processing Systems (NIPS*2000)*, vol. 13, 2001.

[9] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams," in *ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*. ACM Press, 2001, pp. 97–106.

[10] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '00. New York, NY, USA: ACM, 2000, pp. 71–80.

[11] B. Pérez-Sánchez, O. Fontenla-Romero, and B. Guijarro-Berdiñas, "An incremental learning method for neural networks in adaptive environments," in *Neural Networks (IJCNN), The 2010 International Joint Conference on*, July 2010, pp. 1–8.

[12] Z. Wang, M. Jiang, Y. Hu, and H. Li, "An incremental learning method based on probabilistic neural networks and adjustable fuzzy clustering for human activity recognition by using wearable sensors," *IEEE Transactions on Information Technology in Biomedicine*, vol. 16, no. 4, pp. 691–699, July 2012.

[13] S. Ahmed, N. Shakev, A. Topalov, K. Shiev, and O. Kaynak, "Sliding mode incremental learning algorithm for interval type-2 takagi–sugeno–kang fuzzy neural networks," *Evolving Systems*, vol. 3, no. 3, pp. 179–188, 2012.

[14] C. Zhang and Y. Ma, *Ensemble Machine Learning: Methods and Applications*. Springer, 2012.

[15] S. Lee and M. Eskenazi, "Incremental sparse bayesian method for online dialog strategy learning," *IEEE Journal of Selected Topics in Signal Processing*, vol. 6, no. 8, pp. 903–916, Dec 2012.

[16] M. Dimkovski and A. An, "A bayesian model for canonical circuits in the neocortex for parallelized and incremental learning of symbol representations," *Neurocomputing*, vol. 149, Part C, pp. 1270 – 1279, 2015.

[17] M. D. Muhlbaier, A. Topalis, and R. Polikar, "Learn++.nc: Combining ensemble of classifiers with dynamically weighted consult-and-vote for efficient incremental learning of new classes," *Trans. Neur. Netw.*, vol. 20, no. 1, pp. 152–168, Jan. 2009.

[18] T. Kidera, S. Ozawa, and S. Abe, "An incremental learning algorithm of ensemble classifier systems," in *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, 2006, pp. 3421–3427.

[19] D. Parikh and R. Polikar, "An ensemble-based incremental learning approach to data fusion," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 37, no. 2, pp. 437–450, April 2007.

[20] T. Xiao, J. Zhang, K. Yang, Y. Peng, and Z. Zhang, "Error-driven incremental learning in deep convolutional neural network for large-scale image classification," in *Proceedings of the 22Nd ACM International Conference on Multimedia*, ser. MM '14. New York, NY, USA: ACM, 2014, pp. 177–186.

[21] R. Calandra, T. Raiko, M. P. Deisenroth, and F. M. Pouzols, *Artificial Neural Networks and Machine Learning – ICANN 2012: 22nd International Conference on Artificial Neural Networks, Lausanne, Switzerland, September 11-14, 2012, Proceedings, Part II*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, ch. Learning Deep Belief Networks from Non-stationary Streams, pp. 379–386.

[22] G. Zhou, K. Sohn, and H. Lee, "Online incremental feature learning with denoising autoencoders," in *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics (AISTATS-12)*, N. D. Lawrence and M. A. Girolami, Eds., vol. 22, 2012, pp. 1453–1461.

[23] M. Pratama, S. G. Anavatti, P. P. Angelov, and E. Lughofer, "Panfis: A novel incremental learning machine," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 1, pp. 55–68, Jan 2014.

[24] D. Leite, R. Ballini, P. Costa, and F. Gomide, "Evolving fuzzy granular modeling from nonstationary fuzzy data streams," *Evolving Systems*, vol. 3, no. 2, pp. 65–79, 2012.

[25] J. R. B. Jr, L. Zhao, and A. A. Lopes, "An incremental learning algorithm based on the k-associated graph for non-stationary data classification," *Information Sciences*, vol. 246, pp. 52 – 68, 2013.

[26] J. a. Gama, R. Fernandes, and R. Rocha, "Decision trees for mining data streams," *Intell. Data Anal.*, vol. 10, no. 1, pp. 23–45, Jan. 2006.

[27] S. Grossberg, "Adaptive resonance theory," *Scholarpedia*, vol. 8, no. 5, p. 1569, 2013, revision #145360.

[28] W. Liao, D. Zeng, L. Zhou, S. Wang, and H. Zhong, *MultiMedia Modeling: 21st International Conference, MMM 2015, Sydney,NSW, Australia, January 5-7, 2015, Proceedings, Part I*. Cham: Springer International Publishing, 2015, ch. Wireless Video Surveillance System Based on Incremental Learning Face Detection, pp. 118–127.

[29] Y. Gao, Y. Zhan, and D. Shen, "Incremental learning with selective memory (ilsm): Towards fast prostate localization for image guided radiotherapy," *IEEE Transactions on Medical Imaging*, vol. 33, no. 2, pp. 518–534, Feb 2014.

[30] S. Tsumoto and S. Hirano, *Brain and Health Informatics: International Conference, BHI 2013, Maebashi, Japan, October 29-31, 2013. Proceedings*. Cham: Springer International Publishing, 2013, ch. Incremental Induction of Medical Diagnostic Rules, pp. 409–417.

[31] M. Akcakaya, S. Sen, and A. Nehorai, "A novel data-driven method for nonstationarity detection in radar target detection," *IEEE Signal Processing Letters*, vol. PP, no. 99, pp. 1–1, 2016.

[32] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer, "Online passive-aggressive algorithms," *J. Mach. Learn. Res.*, vol. 7, pp. 551–585, Dec. 2006.

[33] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[34] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. New York: Wiley, 2001.

[35] L. J. Tashman, "Out-of-sample tests of forecasting accuracy: an analysis and review," *International Journal of Forecasting*, vol. 16, no. 4, pp. 437 – 450, 2000, the M3- Competition.

[36] C. Bergmeir and J. M. Benítez, "On the use of cross-validation for time series predictor evaluation," *Information Sciences*, vol. 191, pp. 192 – 213, 2012, data Mining for Software Trustworthiness.