Evolving on-line prediction model dealing with industrial data sets

Petr Kadlec and Bogdan Gabrys

Abstract— **In this work we present an instance of an architecture for the development of robust evolving predictive models. The architecture provides a conceptual framework for the development of such models while at the same time it provides mechanisms for the minimisation of effort needed for the development and maintenance of the models. These mechanisms deal with the model and parameter selection, model training, validation and adaptation. Another challenge for the proposed instance is to deal with an industrial data set containing several issues like missing data, outliers, drifting data, etc. This fact calls for high robustness of the deployed models. The success of the models lays in the goal oriented application of several concepts like ensemble building, local learning, parameter cross-validation which are provided by the architecture and exploited by the discussed instance.**

I. Introduction

Applying state-of-the-art predictive modelling techniques from computational intelligence to industrial problems, which are targeted in this work, remains a challenging task. It is often the case that the techniques which show superior performance on *clean* data, fail to deliver the same performance on raw real-life data and are outperformed by simpler methods which show higher robustness towards the imperfect data. Even worse, the behaviour of the predictive techniques often can not be predicted which causes further problems with the model selection and parametrisation. The failure to deliver acceptable performance is often caused by outliers, missing values, measurement noise and drifts which are common in industrial data sets. The most common way how to deal with this problem is by attempting to *clean* the data by applying various pre-processing steps. In this way the data is transformed in favour of the modelling techniques (see [1] for a review of such case studies). However, the drawback of this approach is that because the data can dramatically change from case to case, each new case requires new time consuming manual pre-processing. Furthermore, once the data is pre-processed the correct predictive method has to be selected. This selection is critical for the performance of the whole model since different techniques have different strengths and weaknesses. Very often one can not see a-priori which technique fits best the data and different methods and their parameters have to be tried. Even more critically, in industrial environment the model developers often have their favourite technique and focus only on these without taking any other approaches into account which is definitely not of advantage for the final performance of the model.

The most commonly applied techniques to industrial modelling problems are ranging from statistically based Principle Component Regression [2], Partial Least Squares Regression [3] and Support Vector Machines [4] to techniques from computational intelligence like Multi-Layer Perceptron [5] and Neuro-Fuzzy Systems [6]. Although many applications of these techniques have been published (see e.g. [1], [7] for reviews) most of the authors claim that a certain effort has to be spent on the preparation of the data (i.e. data pre-processing) as well as the techniques (i.e. parameter selection). Another problem is that one also can not separate the two previously discussed tasks, i.e. data pre-processing and predictive technique selection and parametrisation due to their mutual influence on each other. This fact further increases the number of possibilities to be tested in order to identify a well performing model.

In this work we propose another way for dealing with industrial modelling tasks. We approach the task by applying established techniques from statistical machine learning and computational intelligence like parameter cross-validation [8], ensemble techniques [9], meta learning [10] to name just a few of them. This is achieved by applying an instance of a general conceptual architecture proposed in [11] for the development of a robust predictive model. The conceptual architecture does not focus only on the automated method and parameter selection but provides also means for the adaptation and evolution of the decisions in order to adjust to the ever changing environment as it is often present in the case of real-life modelling tasks. The instance proposed in this work exploits this functionality and enables the development of an effective evolving model while at the same time the effort required for its development is kept low.

The rest of this paper is organised in the following way: Section II shows a brief overview of the conceptual architecture and outlines its most critical aspects necessary for the understanding of the proposed instance. This is followed by a methodology for the development of the model and the way in which the data is typically provided in industrial environment in Section III. Section IV is the main contribution of this paper as it presents the actual instance of the architecture and shows the mechanisms applied in order to achieve the high robustness and adaptive capabilities. The model is then evaluated in Section V by applying it to two real-life data sets. Finally, the paper is concluded in Section VI.

II. Architecture overview

This section gives a brief overview of the architecture which is instantiated in this work. The architecture is in more detail discussed in [11]. Due to space limitations the figure showing the general structure of the architecture can not be shown here however Fig. 6 showing an instance of the

Petr Kadlec and Bogdan Gabrys are with the Computational Intelligence Research Group, Bournemouth University, Bournemouth, BH12 5BB, United Kingdom (email: {pkadlec, bgabrys}@bournemouth.ac.uk).

978-1-4244-2754-3/09/$25.00 ©2009 IEEE

architecture can be used to see its structure. The architecture consists of eight main modules which are together with their functions outlined in Table I.

TABLE I

MODULES AND THEIR FUNCTIONS

Module	Function
Data Source	Data maintenance and provision
PPMP, CLMP	Pools of pre-proc. and computational learning methods
Paths	Environment where computational paths are maintained
Path Combinations	Environment where path combinations are maintained
ISM	Partitioning of the data space, receptive fields building
MLL	Management of the high level functions
GPE	Evaluation of the performance of the model
Expert Knowledge	Parameter control, inserting a-priori knowledge, etc.

The aim of the architecture development was to define an environment which unifies the main concepts from statistical machine learning and computational intelligence into a single complex structure with focus on dealing with industrial and application oriented tasks. The key concepts represented within the architecture are: (i) ensemble building; (ii) local learning; and (iii) meta learning. Another particular focus was set on the evolutionary and adaptation capabilities of the architecture.

The information processing within the model is structured in a hierarchical manner. At the lowest level of the architecture, there is a diverse set of data processing units called *computational path* which are maintained in the Paths module (see Fig. 1 for the internal path structure). The paths consist of an arbitrary number of pre-processing methods and one computational learning method. At the next level, the paths are combined to *path combinations* which, apart from the fact that they operate in another data space, do not differ from the paths. At the highest level of the complexity, a management of the underlying levels which evolves the architecture towards the global goal defined by the underlying task (e.g. best predictive performance in the Mean Squared Error sense), takes place.

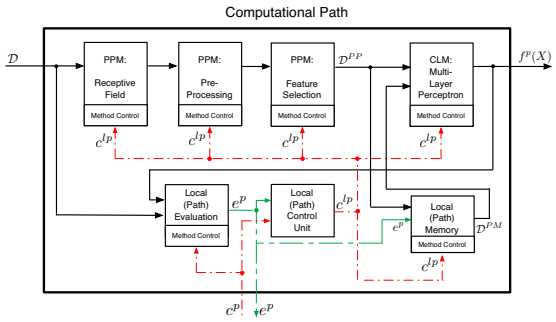

Fig. 1. Computational path internal structure

As mentioned above, the architecture provides several mechanisms for its adaptation. These mechanisms are represented at all three levels of information processing as shown in Fig. 2. The instance of the architecture presented in Section IV shows the implementation of adaptive mechanisms at the path combination level (loop b in Fig. 2) and at the meta level (loop c in Fig. 2)

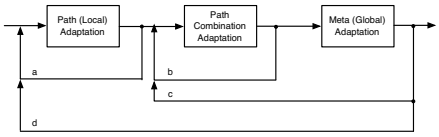

Fig. 2. Adaptation loops provided within the architecture

Due to the space limitation further details of the architecture are skipped but interested readers can refer to [11] for more details.

III. METHODOLOGY

In this work we assume the availability of two different types of data, namely a batch of *historical data* and a stream of *real-time data* which governs the structure of the instance proposed in Section IV.

In many industrial cases there are automated procedures for recording all measurement being done within the industrial processes. These historical recordings describe the behaviour of the process in the past. We assume that a set of such data (i.e. the historical data) is available and can be applied for the initial training of the model.

Once the initial model building phase is finished, the model is applied in an on-line operation and has to deal with the real-time data stream. The real-time data is arriving in an incremental way, i.e. one sample (or a batch of samples) after another. In general, the sampling rate between the input and the target data can differ and additionally there can also be sys between them. The correct target values can be applied to the evaluation of the model performance and its adaptation during the on-line phase.

IV. INSTANCE OF THE ARCHITECTURE

This section presents an instance of the architecture discussed in Section II. According to the methodology discussed in Section III one can distinguish two phases of the development. The first stage, during which an initial version of the model is built, is based on the historical data set. This stage covers Steps 1 to 5 presented later in this section (see Fig. 3). After this stage the model is deployed and provides predictions for the (unlabelled) samples for the on-line data (see Fig. 4). The target values are provided after a certain delay (i.e. after making the prediction). Further on, they are exploited for the adaptation of the model during the on-line phase as discussed in Step 6.

Next the particular steps of the algorithm are presented.

A. Step 1 - Building of receptive fields

The initial step of the algorithm is the partitioning of the available historical data $\mathcal{D}^{hist}$ into receptive fields. Ideally, each receptive field should represent a distinct state of the data. This goal is approached by using concept drift detection techniques, as each receptive field corresponds to a concept

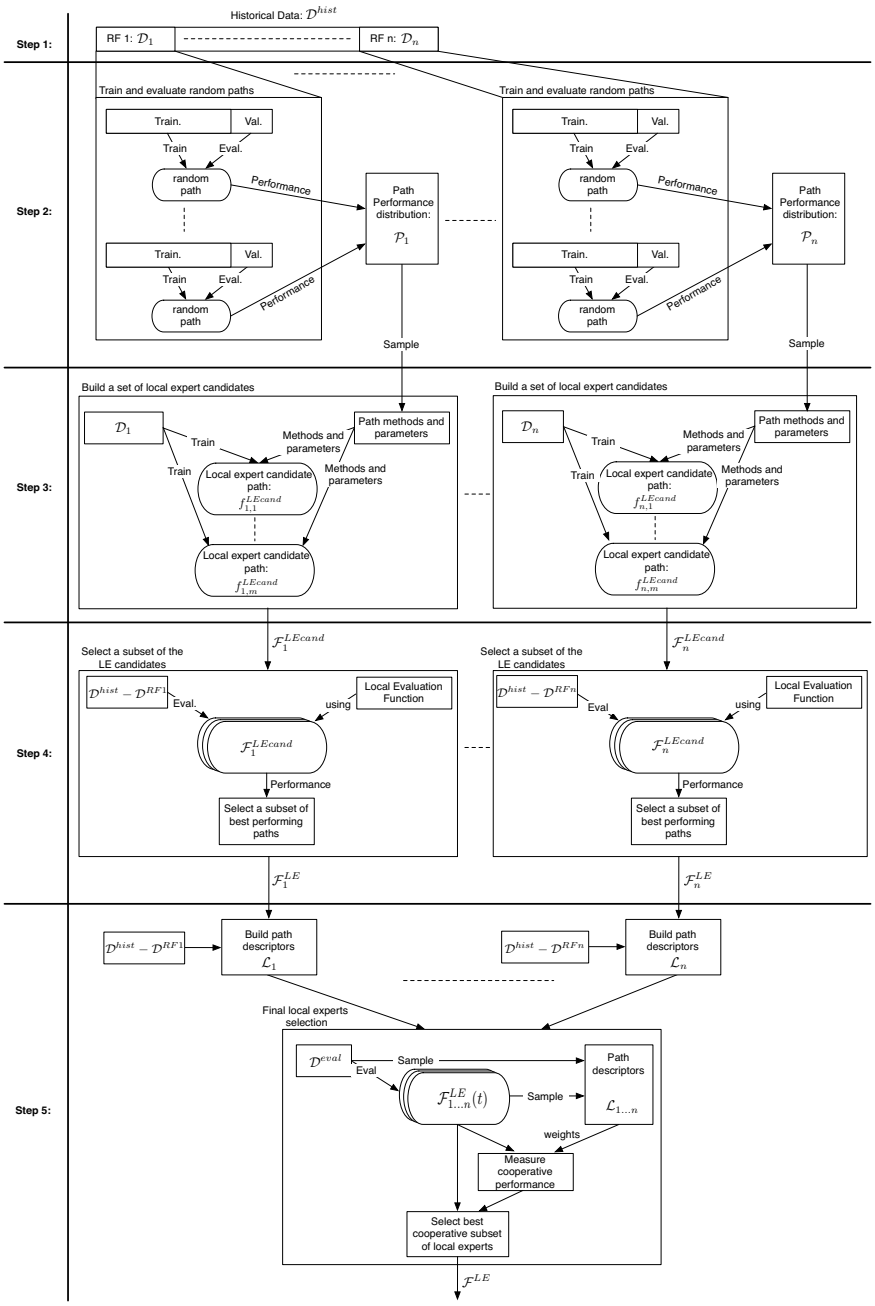

Fig. 3. The proposed algorithm, training phase

of the data (see e.g. [12] for concept drift overview). Further details of the implementation of the receptive field building can be found in [13].

The partitioning of the historical data is performed within the Instance Selection Management (ISM) of the architecture. From the ISM, the built receptive fields are submitted to the Pre-Processing Methods Pool (see Fig. 6).

The outcome of this stage is a set of partitions $\mathcal{D}_{1...n}$ of the historical data.

B. Step 2 - Learning of performance distributions

Provided the receptive fields, the next step is learning the performance distributions $\mathcal{P}_{1...n}$ of all available pre-processing and computational learning techniques and their parameters within each receptive field $i := 1 \ldots n$. For a given number of iterations, there is a computational path built and evaluated for each iteration. The pre-processing and predictive techniques as well as their parameters (e.g. PCA with 10 principle components and MLP with 5 hidden units) are drawn randomly from the set of available methods

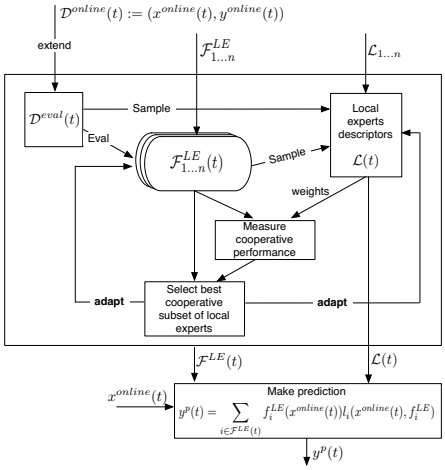

Fig. 4. Step 6: the on-line phase

provided by the two method pools (PPMP, CLMP) shown in Fig. 6. The paths are trained on a random sub-sample of the receptive field data samples and evaluated on independent data (remaining samples from the same receptive field). Provided the performance of the paths, the performance distribution $\mathcal{P}_i$ is updated at the corresponding position (see Fig. 7 for an example of the performance distribution function). After a sufficient number of iterations, the performance distribution shows the relative performance of the methods within the receptive field.

The distributions $\mathcal{P}_i$ are stored within the Meta Level Learning (MLL) module of the architecture, from where the distributions will be sampled in order to obtain the parameters for the deployed paths as shown in the next step.

C. Step 3 - Building of local expert candidates

During this step, a set of computational paths $\mathcal{F}_i^{LEcand}$ is built for each receptive field and deployed in the Paths module. The methods and parameters for the paths (or local expert) candidates $f_{i,j}^{LEcand}$ are chosen by sampling the performance distributions stored in the MLL module. Since the performance distributions $\mathcal{P}_i$ show the relative performance of the full path, i.e. the pre-processing and predictive methods, a single sampling of these provides the parameters for the whole path.

In order to increase the robustness of the paths, each of them is a committee based on cross-validation of the data samples within the receptive field. The diversity of the built computational path is further increased by selecting a random subset (e.g. 80%) of the samples within the receptive field as an input for the cross-validation. After the training the sets of local expert candidates $\mathcal{F}_i^{LEcand}$ are available in the Paths module.

D. Step 4 - Paths selection in a competitive environment

In order to remove paths showing poor generalisation performance from the pools, the paths are evaluated on independent data $\mathcal{D}_i^{indep}$. The independent data are a subset of the historical data which was not used during the training of the particular path (i.e. $(\mathcal{D}_i^{indep} \in \mathcal{D}^{hist}) \cap (\mathcal{D}_i^{indep} \notin \mathcal{D}_i)$). The evaluation is done using the Local Performance Function implemented within the paths (see Fig. 1). In this work we use an evaluation function which is a combination of the Root Mean Squared Error (RMSE) and the correlation coefficient. Having the performance of the paths a subset of best performing paths is selected. This scenario resembles a competitive environment where only the best paths pass this stage and are maintained as part of the model.

This functionality is implemented mainly in the Paths module of the architecture. The performance of the paths is reported to the Path Control from where a subset of the paths with the best performance $\mathcal{F}_i^{LE}$ is selected and kept while the remaining paths are removed from the Paths module.

E. Step 5 - Paths selection in a cooperative environment

Prior to this step, there is a path descriptor $l_{i,j} \in \mathcal{L}_i$ built for each of the paths, see Fig. 8 for an example of such a descriptor. This descriptor is a 2D map showing the performance of the local expert in the space of each input and the target variable. Therefore each path descriptor consists of a set of 2D descriptors where the number of them corresponds to the number of input variables. At a later stage (see Section IV-F), this maps will be read at the positions of the input variables and of the local expert predictions and the sampled values used as combination weights for the calculation of the final prediction.

The aim of this step is to select a subset of the paths which cooperatively optimise their performance. The cooperative performance is assessed by evaluating the distance between a weighted sum (the weights are obtained from the path descriptors) of the selected paths' predictions and the correct target values. The evaluation is based on a random subset of the historical data $\mathcal{D}^{eval} \in \mathcal{D}^{hist}$. The best team of local experts is forwarded to the Path Combinations module where they build the final model.

In contrast to the previous step, there are no paths removed from the Paths module at this stage. It will be shown later that one of the evolving capabilities of the model is a dynamic re-evaluation of the paths ensemble and thus all paths which entered this stage have to be kept.

The path descriptors $\mathcal{L}_i$ are stored within the Meta Level Learning module because they have to be accessible from the Path Combinations module. The selected paths are combined in the Path Combinations module where the final predictions are built.

F. Step 6 - On-line prediction and adaptation

The proposed approach provides high robustness towards issues of industrial data and an automated method and parameter selection, but this is not the only remarkable property of it. Another advantage is the possibility to deal with changing data by applying an evolving approach described in this section and shown in Fig. 4. The changing data is the aspects which very often leads to the necessity for manual tuning and re-training of predictive models in industrial environment and

thus to deal with this problem is one of the main focuses of the architecture discussed in Section II and the proposed model as an instance of the architecture makes use of this functionality.

There are several positions where the model can be updated. Starting from the path level, the particular path can be adapted using e.g. the moving window technique, which is the traditional way for the adaptation of predictive models. However, since there are several problems with this technique, like the estimation of the length of the moving window, we avoid this techniques and focus on the adaptation at higher levels of the model.

The model performs a prediction $y^p(t)$ given the unlabelled instances of the on-line data stream $x^{online}(t)$ which is a weighted combination of the particular paths' predictions. The weights are obtained by sampling the path descriptors at position given on one hand by the input data $x^{online}(t)$ and by the path prediction $f_i^{LE}(t)$ on the other hand.

As shown in Fig. 4, each time a target value $y^{online}(t)$ of the on-line data sample $\mathcal{D}^{online}(t)$ is received, the diversity of the ensemble $\mathcal{F}^{LE}$ is checked and adapted if necessary. This action corresponds to a dynamic re-building of the team of prediction experts which allows to maintain a constant level of performance of the team despite the changing environment.

Another possibility for adaptation is updating the path descriptors. Provided the target value, the descriptors are adapted locally in the neighbourhood of the input and target value. As the main purpose of the descriptors is to provide the combination weights, adapting the descriptors is equal to the adaptation of the path's contribution to the final prediction. The adaptation of the descriptors is performed by applying the Delta rule [14]. Applying this adaptation rule ensures that the local experts improve their cooperative performance in the neighbourhood of the current sample.

Finally, it should be noted that an important property of the two applied adaptation approaches is their algorithmic independence. This means that the mechanisms can be applied independently to the underlying pre-processing and predictive techniques.

V. Experiments

In this section, two soft sensors for the on-line prediction of the target variable are presented as a practical implementation of the architecture instance discussed in Section IV. For the experimental evaluation we follow the methodology from Section III and split the available data into two sets. A set of historical data (30% of the available data sample) and on-line data which are the residual 70% of samples. This split of the available data is justified by the focus on the evolutionary properties of the model.

A. The data sets

1) The drier data set: The target values of this data set are laboratory measurements of the residual humidity of the chemical process product. The data set has 19 input features, most of them being temperatures, pressures and humidities measured within the processing plant. The data set consists of 1219 data samples covering almost seven months of the operation of the process. It consists of raw unprocessed data as it were recorded by the process information and measurement system. For this reason some of the input variables present some common issues of industrial data like measurement noise, missing values, data outliers, etc. Figure 5 shows two examples of variable affected by outliers, missing values and measurement noise.

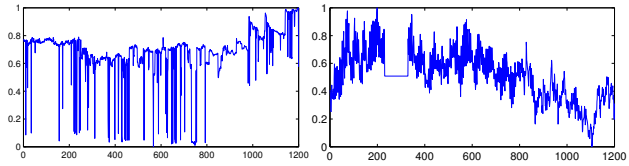

Fig. 5. Drier pressure variable showing measurement noise and data outliers

2) The thermal oxidiser data set: This regression data set deals with the prediction of exhaust gas concentration of an industrial process. The task is to predict the concentrations of NO_x in the exhaust gases. The data set consists of 36 input features (i.e. hard sensor measurements). The input features are physical values like concentrations, flows, pressures and temperatures measured during the operation of the plant. The data set consists of 2053 sample points. For this data set similar statements as for the drier data (see Section V-A.1) are valid, i.e. the data are raw process data exhibiting a lot of issues like data outliers or missing values.

B. The implementation

Figure 6 shows the developed model as an instance of the architecture shown in Section II. Apart from the structure of the instance, which is inherited from the general architecture, the figure shows the mechanisms which are implemented within the different modules. The following paragraphs present the key aspects of the instantiated model.

The Data Source module serves the data to the other parts of the architecture according to the previously defined methodology, i.e. the first 30% of the samples as a batch of historical data $\mathcal{D}^{hist}$ and the rest as a stream of on-line data $\mathcal{D}^{online}$ with delayed target values.

The Pre-Processing Methods Pool (PPMP) provides the following objects:

- Standardisation (STD): mapping the particular variable to the range $(0, 1)$
- Smoothing filter (SF): Smoothing the particular variables using an averaging sliding window
- Robust Principle Component Analysis (PCA): Variable transformation [15]

The following methods are implemented in the Computational Learning Methods Pool (CLMP):

- Multiple Linear Regression (MLR): linear regression object [16]

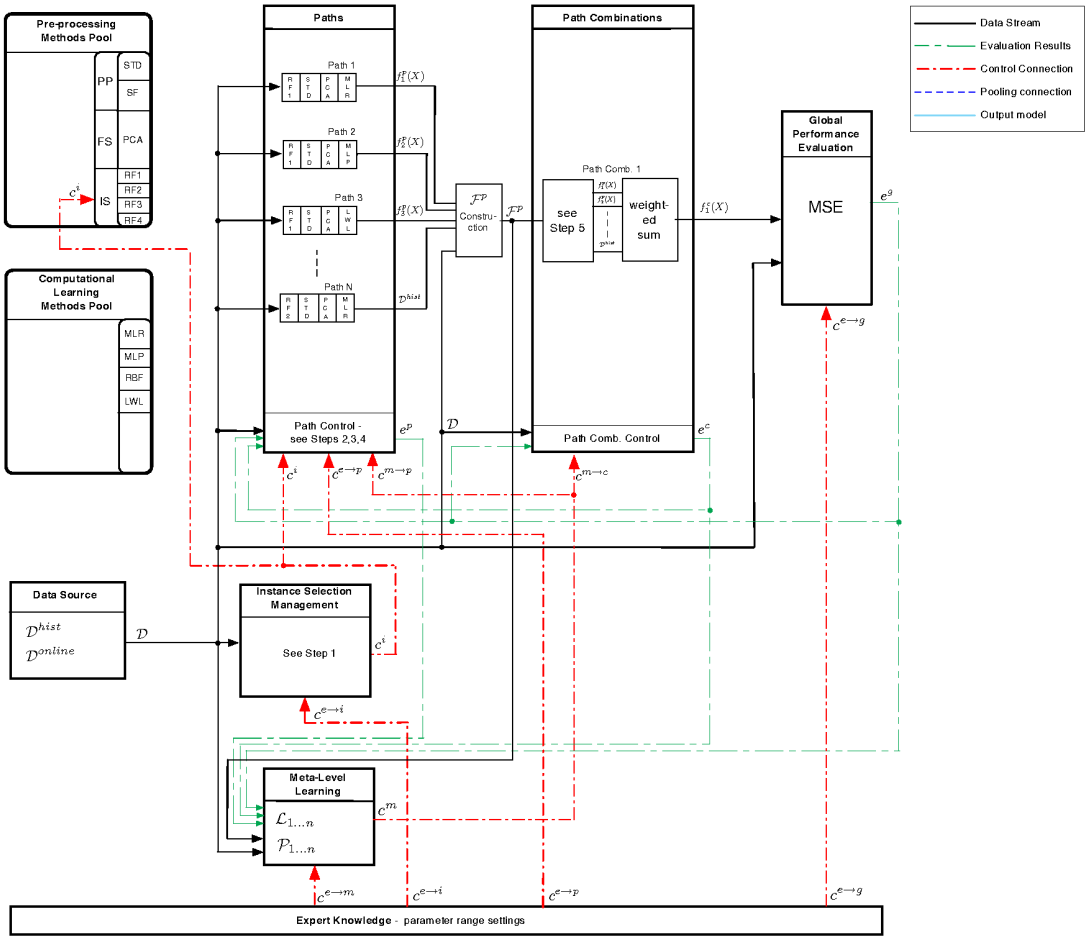

Fig. 6. Instance of the architecture

- Multi-Layer Perceptron (MLP): object providing the MLP form of an Artificial Neural Network [5]
- Radial Basis Function Network (RBF): another variant of an ANN [5]
- Locally Weighted Learning (LWL): lazy-learning type of algorithm [17]

The algorithm applied within the Instance Selection (IS) module is in further detail described in [13]. The historical data is partitioned into a set of receptive fields (e.g. four in the case of the drier data set, i.e. $n = 4$ in Fig. 3) which are submitted from the IS module to the PPMP module.

The Paths module provides an environment for the maintenance of the computational paths. During Step 3 of the algorithm, there is a set (100 for each receptive field in this experiment, i.e. $m = 100$ in Fig. 3) of local expert (path) candidates launched. The launching is handled from the Path Control part of the Paths module. The parameters for the paths are obtained by drawing from the performance distribution managed in the Meta-Level Learning module. Each of the path is evaluated for its generalisation performance according to the Local Performance Function implemented in the path object (see Fig. 1) which in this work is a combination of the Root Mean Squared Error and correlation coefficient. According to Section IV-C, only a subset (10%) of the paths is kept in the Paths module and the rest of the paths is removed.

The Path Combination module implements the combination of the paths. The paths to be combined have to be selected as it was described in Section IV-E and IV-F. The final selection of the local experts is performed on the basis of their cooperative performance of the subset and is performed from the Path Combination Control at the end of the training phase and adapted during the on-line phase. In order to be able to combine the predictions of the particular paths, the combination weights have to be obtained from the path descriptors stored in the MLL module (see bellow).

The Meta-Level Learning (MLL) module performs several functions within this instance. First of all, there are the distributions of the methods' and parameters' performances $\mathcal{P}_i$ stored (discussed in Section IV-B). The parameter ranges within which the methods are evaluated are as follows:

- STD- no parameters
- SF- length of the smoothing filter: $[1, 4, 7, 10]$
- PCA- covered variance of the data space:

$[90\%, 95\%, 99\%]$

- MLR- no parameters
- MLP- number of hidden units: $[1, 3, 5, 7, 9, 11]$
- RBF- number of Gaussian centres: $[6, 9, 12]$
- LWL- number of samples used to build a local model: $[10, 50, 100]$

The performance distributions are built separately for each receptive field. An example of such a performance distribution is shown in Fig. 7. The figure shows dominant performance of the LWL method (with its parameter value equal to 100) in combination with several pre-processing methods (e.g. PCA covering 90% of the data variance and four samples long smoothing filter). Another function

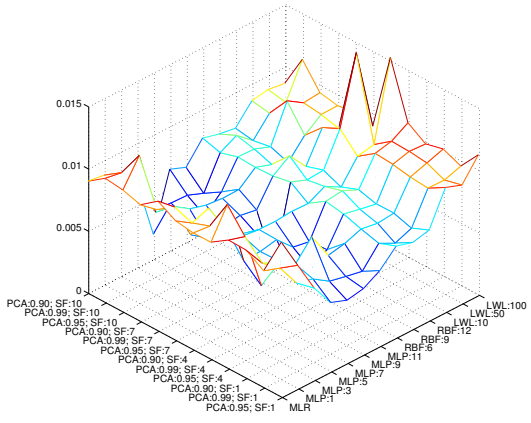

Fig. 7. Performance distribution of various methods and their parameters

provided by the MLL is storing the path descriptors discussed in Section IV-E. Example of which is shown in Fig. 8.

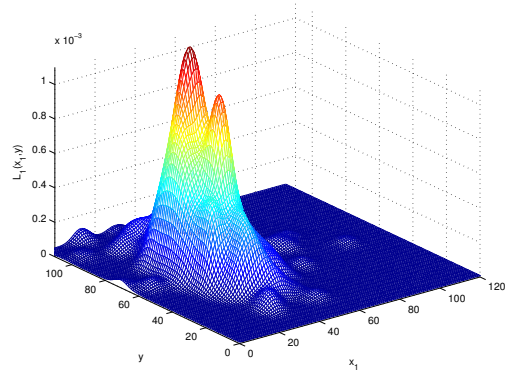

Fig. 8. Example of a path descriptor

The Global Performance Evaluation (GPE) module defines the global function which has to be optimised. Since the focus of this case study is on obtaining best possible prediction of the target variable, there is the Mean Squared Error implemented in this module.

The role of the Expert Knowledge is rather minor for this case. It is limited to setting the parameter ranges of the methods as listed above.

C. Results evaluation

In this section the performance of the developed soft sensor is assessed. The proposed algorithm is compared to another state-of-the-art adaptive algorithm based on local learning, namely the Locally Weighted Projection Regression (LWPR) [18]. LWPR has some attractive features, like local dimensionality reduction, which makes it relevant to industrial data modelling. First, a deeper analysis of the trained model based on the drier data set, which was presented in Section V-A.1, is provided. This is followed by the application of both algorithms with exactly the same parameters to another industrial data set in Section V-A.2.

The parameter set-up of our algorithm was already discussed in Section V-B. After the training phase, there are four receptive fields and a set of eight paths forming the final ensemble. This initial ensemble consists of four MLR models and four MLPs models. This ensemble is changing throughout the on-line phase, while the number of members varies between 8 and 30. As for the LWPR method, its parameters were optimised using an exhaustive search through the space of reasonable values. Best results were achieved using following parameter values: (i) `init_D`=4; (ii) `diag_only`=0; (iii) `w_gen`=0.9; (iv) `w_prune`=0.9; (v) `penalty`=1e-7; (vi) `meta_learning`=0; (vii) `update_d`=0; and (viii) `kernel`=Gaussian. This set-up leads to the building of 124 receptive fields based on the historical data. These are extended to 366 receptive fields during the on-line phase which is a large number compared to the four receptive fields of our model.

As next, both models were applied to the stream of on-line data. Each time after making a prediction on an incoming sample, the models were provided the correct target value, our model using the mechanisms described in Section IV-F (i.e. adapting the teams of experts and the path descriptors) and LWPR using its intrinsic adaptation method (for details see [18]). The predictions of both models and the correct target values are shown in Fig. 9. One can see from the figure, that despite the impurities in the data both models provide good, and almost equal, performance ($MSE = 4.59e-3$ and correlation coefficient$= 0.46$ of our model vs. $MSE = 4.34e-3$, correlation coefficient$= 0.45$ of LWPR). This shows that both models succeed adapting with the data.

However, this situation changes dramatically when applying the models to another data set. Both of the models were applied with the same parameter set-up as in the previous experiment. After using the same procedure as before, the LWPR approach fails to deliver a stable model. Whereas the model based on the technique proposed in this work again delivers a well performing model as shown in Fig. 10. In fact, after a parameter optimisation the LWPR method delivers similar performance to the one of our (unoptimised) model. This demonstrates that the discussed approach provides, apart

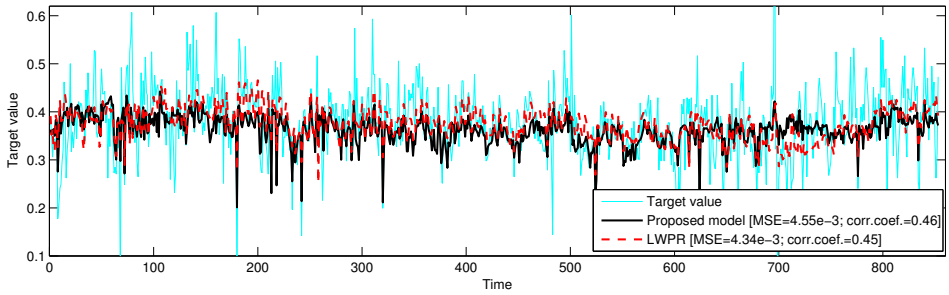

Fig. 9. Predictions on the independent test data of the adaptive models (drier data set)

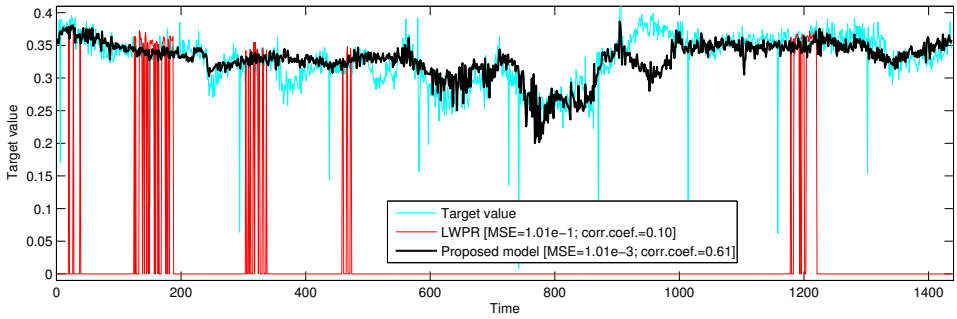

Fig. 10. Predictions on the independent test data of the adaptive models (thermal oxidizer data set)

from the ability to adapt with the incoming stream of data, also capability to adjust its structure in accordance with the provided historical data which is a vital ability for the achievement of the goals discussed in the introduction.

VI. Conclusions

This work demonstrates the applicability of an architecture for the development of evolving data-driven models which was proposed earlier by the authors. An instance of the architecture, which makes use of some of the mechanisms provided for model development and maintenance, is shown to have adaptation ability at different levels. A model developed according to the architecture shows comparable performance to another adaptive model based on the Locally Weighted Projection Regression (LWPR) where the parameters of the LWPR method were adjusted to deliver optimal performance for the given modelling task. It is also presented that without any additional parameter optimisation the LWPR technique fails to deliver a working model on another data set. This is in contrast to the instance of the architecture which succeeds to deliver a working model for the new data set without any parameter changes. These results demonstrate that the developed model is able to evolve on one hand with changing data and on the other hand with is able to adapts its structure with the underlying data set and thus allows the application of the same model across different modelling tasks.

References

[1] Y. Dote and S. J. Ovaska, "Industrial applications of soft computing: A review," in *Proceedings of the IEEE*, vol. 89, 2001, pp. 1243–1265.
[2] I. T. Jolliffe, *Principal Component Analysis*. Springer, 2002.
[3] H. Abdi, "Partial least squares (pls) regression," *Encyclopedia of Social Sciences, Research Methods. Thousand Oaks (CA): Sage (2003)*, 2003.
[4] V. N. Vapnik, *Statistical learning theory*. Wiley New York, 1998.
[5] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford University Press, USA, 1995.
[6] J. S. R. Jang, C. T. Sun, and E. Mizutani, *Neuro-fuzzy and soft computing*. Prentice Hall Upper Saddle River, NJ, 1997.
[7] P. Kadlec, B. Gabrys, and S. Strandt, "Data-driven soft sensor in the process industry," *accepted for publication in Computers and Chemical Engineering*.
[8] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, vol. 2, p. 11371145, 1995.
[9] L. I. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-IEEE, 2004.
[10] R. Vilalta and Y. Drissi, "A perspective view and survey of meta-learning," *Artificial Intelligence Review*, vol. 18, no. 2, pp. 77–95, 2002.
[11] P. Kadlec and B. Gabrys, "Self-adapting architecture for on-line predictive modelling soft sensors," *submitted to Systems, Man and Cybernetics, Part B, IEEE Transactions on*.
[12] G. Widmer and M. Kubat, "Learning in the presence of concept drift and hidden contexts," *Machine Learning*, vol. 23, no. 1, pp. 69–101, 1996.
[13] P. Kadlec and B. Gabrys, "Adaptive local learning soft sensor for inferential control support," in *CIMCA 2008*. Vienna: IEEE, 2008.
[14] R. O. Duda, P. E. Hart, and D. G. Stork, *Patten Classification*. John Wiley and Sons, 2001.
[15] C. Croux and A. Ruiz-Gazen, "High breakdown estimators for principal components: the projection-pursuit approach revisited," *Journal of Multivariate Analysis*, vol. 95, no. 1, pp. 206–226, 2005.
[16] A. J. Dobson, *An Introduction to Generalized Linear Models*. Chapman Hall/CRC, 2002.
[17] E. Frank, M. Hall, and B. Pfahringer, "Locally weighted naive bayes," in *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2003, pp. 249–256.
[18] S. Vijayakumar, A. D'Souza, and S. Schaal, "Incremental online learning in high dimensions," 2005.