Chapter 6
Drift Correction of Chemical Sensors

Artificial olfaction systems that try to mimic human olfaction by using arrays of gas chemical sensors combined with pattern recognition methods represent a potentially economic tool in many areas of industry such as: perfumery, food and drinks production, clinical diagnosis, health and safety, environmental monitoring and process control. However, successful applications of these systems are still largely limited to specialized laboratories. Among others, sensor drift, the lack of stability over time still limit real industrial setups. This chapter presents and discusses an evolutionary based adaptive drift-correction method designed to work with state-of-the-art classification algorithms. The proposed system exploits a leading-edge evolutionary strategy to iteratively tweak the coefficients of a linear transformation able to transparently transform raw sensors measures in order to mitigate negative effects of the drift. The optimal correction strategy is learned without a-priori models or other hypothesis on the behavior of physical-chemical sensors. Preliminary results have been published in [49].

6.1 Introduction

The human sense of smell is a valuable tool in many areas of industry such as: perfumery, food and drinks production, clinical diagnosis, health and safety, environmental monitoring and process control [65] [156]. Artificial olfaction tries to mimic human olfaction by using arrays of gas chemical sensors combined with pattern recognition (PaRC) methods [121]. When a volatile compound contacts the surface of the sensor array, a set of physical changes modify the electrical properties of each sensor material. Such an electronic disturb can be measured and digitalized to be used as a feature for the specific compound. A preliminary calibration phase is used to train the PaRCalgorithm in order to map each gas concentration or class to the responses from the sensor array. The trained model is then used for identification during later measurements. The classification rate of the PaRCsystem determines the final performance of the electronic olfaction system.

Gas sensor arrays represent a potentially economic and fast alternative to conventional analytical instruments such as gas chromatographs. Considerable research into new technologies is underway, including efforts to use nano-engineering to enhance the performance of traditional resistive Metal Oxide (MOX) sensors. However, successful applications of gas sensor arrays are still largely limited to specialized laboratories [118]. Among others, lack of stability over time and high cost of recalibration still limit the widespread adoption of artificial olfaction systems in real industrial setups [115].

The *gas sensor drift* consists of small and non deterministic temporal variations of the sensor response when it is exposed to the same analytes under identical conditions [115]. This problem is generally attributed to sensors aging [140], but it could be also influenced by a variety of sources including environmental factors or thermo-mechanical degradation and poisoning [79]. As a result, sensors' selectivity and sensitivity decrease, changing the way samples distribute in the data space and thus limiting the ability of operating over long periods. PaRCmodels built in the calibration phase become useless after a period of time, in some cases weeks or a few months. After that time the artificial olfaction system must be completely re-calibrated to ensure valid predictions [5]. Up to now, it is impossible to fabricate chemical sensors without drift. In fact, drift phenomena afflict almost all kinds of sensors [123][31][113]. Sensor drift should be therefore taken into account and compensated in order to achieve reliable measurement data from the sensor array.

Algorithms to mitigate negative effects of the gas sensor drift are not new in the field, with the first attempt to tackle this problem dated back to the early 90s [121, chap. 13]. Notwithstanding, the study of the sensor drift is still a challenging task for the chemical sensor community [121][115]. Solutions proposed in the literature can be grouped in three main categories: (i) periodic calibration, (ii) attuning methods, and (iii) adaptive models.

Retraining the PaRCmodel by using a single calibrant or a set of calibrants is perhaps the only robust method to mitigate drift effects even in presence of sensor drift over an extremely long period of time [142]. However, calibration is the most time-intensive method for drift correction since it requires system retraining and additional costs. Hence, it should be used sparingly. Moreover, while this approach is rather simple to implement for physical sensors where the quantity to be measured is exactly known, chemical sensors pose a series of challenging problems. Indeed, in chemical sensing, the choice of the calibrant strongly depends on the specific application especially when the sensing device is composed of a considerable number of cross-correlated sensors [74][73]. This leads to loss of generalization and lack of standardization which, on the contrary, would be highly required by industrial systems.

Attuning methods try to separate and reject drift components from real responses. They can provide significant improvements in the classification rate over a fixed time period, and may also allow to obtain real responses to be used in gas quantitative analysis. Attempts to attune the PaRCmodel by performing component correction based on Principal Component Analysis (PCA) [7][153], Independent Components Analysis (ICA) [110], Canonical Correlation Analysis (CCA), or

Partial Least Squares (PLS) [68] have received considerable attention in the sensor community. Orthogonal Signal Correction (OSC) was recently demonstrated to be one of the best methods to attune PaRCmodels and to compensate drift effects [115]. However, such techniques do not completely solve the problem. One of the main drawbacks is the need of a set of calibration data containing a significant amount of drift allowing to precisely identify the set of components to be corrected or rejected. This might be not the case in industrial setups where calibration data are collected over a short period of time. Moreover, adding new analytes to the recognition library represents a major problem since rejected components might be necessary to robustly identify these new classes. Finally, these methods contain no provisions for updating the model and thus may ultimately be invalidated by time evolving drift effects.

Adaptive models try to adapt the PaRCmodel by taking into account pattern changes due to drift effects. Neural networks such as self-organizing maps (SOMs) [103][166] or adaptive resonance theory (ART) networks [157][99] have been frequently used in the past. They have the advantage of simplicity because no recalibration is required. Yet, two main weaknesses can be identified. First, a discontinuity in response between two consecutive exposures (regardless of the time interval between the exposures) would immediately invalidate the PaRCmodel and would prevent adaptation. Second, a key to obtain reliable results is to set appropriate thresholds for choosing the winning neuron, and this typically requires a high number of training samples owing the complexity of the network topology. Moreover, they are limited to gas classification applications. Whenever both classification and gas quantitative analysis is required, current adaptive methods can be hardly applied to obtain reliable gas concentration measurements [78].

In this chapter we present and discuss an evolutionary based adaptive unsupervised drift-correction methodology designed to work with state-of-the-art classification algorithms. The term unsupervised refers to the fact that drift correction is obtained without considering any specific drift model. Drift effects are directly learned from the set of unlabeled raw measures obtained from the sensor array. This work improves our previous attempt to apply evolutionary methods in the drift correction process [49]. A linear transformation is applied to raw sensor's features to compensate drift effects. Such linear transformation is continuously and slowly evolved to follow drift effects. Evolution is achieved through a covariance matrix adaptation evolutionary strategy (CMA-ES), perfectly suited for solving difficult optimization problems in continuous domain. Compared to existing adaptive solutions, the proposed approach is able to transparently adapt to changes in the sensors' responses even when the number of available samples is not high and new classes of elements are introduced in the classification process at different time frames. Experimental results demonstrate that the suitability of the proposed methodology does not depend on the exploited classifier.

6.2 Method and Theory

he basic steps and concepts of the proposed drift correction process are summarized in Figure 6.1.

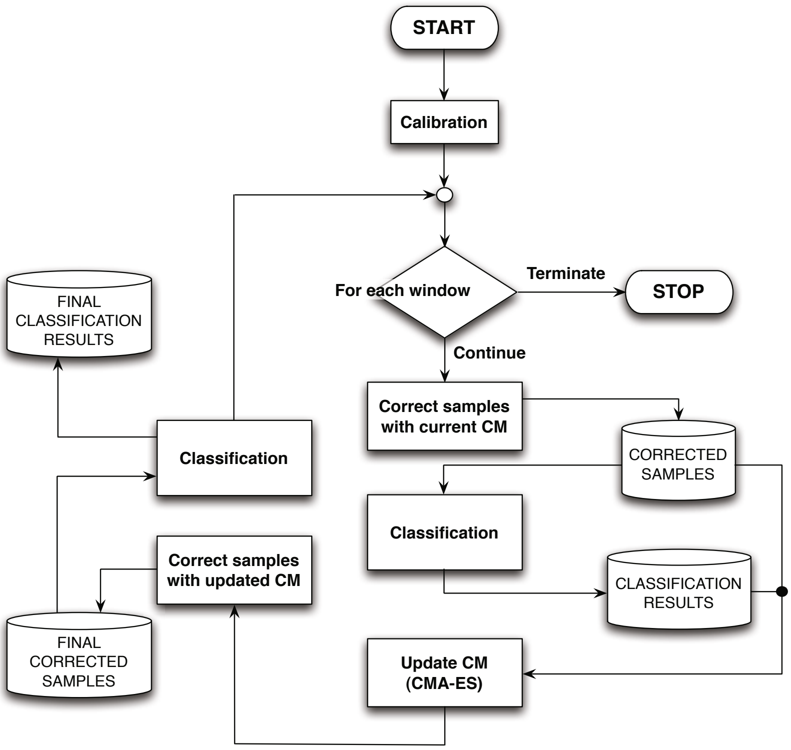

Fig. 6.1 Conceptual steps of the drift correction process

As common in artificial olfaction systems a preliminary calibration phase is used to collect a set of training samples for m different classes y_i ($i \in [1,m]$), each one identifying a specific gas compound. Training samples are used to train a classifier able to map a generic sample $\mathbf{x} \in \mathbb{R}^n$ (where n is the number of sensors in the array) into one of the m available classes:

$$\mathscr{C} : \mathbf{x} \to \{y_1, y_2, \ldots, y_m\} \tag{6.1}$$

Any type of classification algorithm can be theoretically plugged into this system. The idea behind the proposed drift correction method is to reduce variations in the sensors response caused by the sensor drift, thus augmenting the validity window of the classification model.

Once the calibration phase is concluded the system is ready to accept samples to be analyzed and classified. The analysis is performed considering windows of samples. A *window* (W) is a collection of k consecutive measurements obtained by the same sensor array, where the drift may be assumed linear. Windows are not necessarily associated to measurement sessions: a single measurement session may be split into multiple windows; and multiple sessions may be grouped into a single window depending on the specific application and measurement setup. For example, in a laboratory where the same expensive equipment is shared between different research groups, consecutive sessions of measurements could be grouped in the same window, while sessions for the same project that take place after the equipment has been used for another research could be put in a separate window. We denote with $\mathbf{x}_{i,j} \in \mathbb{R}^n$ the j^{th} sample of the i^{th} window W_i. Within a window samples are ordered with ascending sampling time and the same happens for different windows.

According to the definition of Section 6.1 we assume that the sensor drift causes changes in the sensors' response slowly over the time and that both its direction and intensity for each considered sample are not randomly distributed.

For each window W_i the drift correction process performs five computational steps:

1. Each sample $\mathbf{x}_{i,j} \in W_i$ is corrected by applying a *correction factor* (cf) able to mitigate the drift effect (see Section 6.2.1). The result is a set of *corrected samples* denoted as: $\mathbf{xc}_{i,j} \in \mathbb{R}^n$);
2. Each corrected sample $\mathbf{xc}_{i,j}$ is classified using the classifier $\mathscr{C}$ of equation 6.1 trained during the calibration phase (see Section 6.2.2);
3. Corrected samples and classification results are used in an evolutionary process to adapt the current correction factor to the changes of the sensor drift observed in the current window (see Section 6.2.3);
4. Each sample $\mathbf{x}_{i,j} \in W_i$ is corrected again by applying the updated correction factor computed during step 4;
5. Corrected samples are classified again, and the final classification results are provided as outcome of the system.

The following subsections provide details on how the different steps are implemented.

6.2.1 Correction Factor

By considering a sample $\mathbf{x}_{i,j} \in \mathbb{R}^n$ as a point in the n-dimensional space of the sensor array features, the drift effect represents a translation of the point along a preferred direction. Under the hypotheses that in the very short term the variation imposed by the drift is small we can approximate it with a linear translation [7] and we can therefore envision to correct it by applying a linear transformation.

Given a sample $\mathbf{x}_{i,j} \in W_i$ the corrected sample $\mathbf{xc}_{i,j}$ is therefore computed as :

$$\mathbf{xc}_{i,j} = \mathbf{x}_{i,j} + \underbrace{\mathbf{x}_{i,j} \times \mathbf{M}_i}_{\text{correction factor}} \tag{6.2}$$

where $\mathbf{M}_i \in \mathbb{R}^{n \times n}$ is the *correction matrix* for the window W_i generating a correction factor for each feature of the sample obtained as a linear combination of the values of all features in the sample. Considering all features when computing the correction factor allows us to take into account correlations among sensors in the drift phenomena.

The correction factor of feature i of a sample $\mathbf{x}$ can be therefore computed as:

$$cf_i = x[1] \cdot M[1][i] + \ldots + x[n] \cdot M[n][i] \tag{6.3}$$

The correction matrix for the first window ($\mathbf{M}_1$) is initially set to the null matrix, i.e., no correction is applied immediately after calibration.

6.2.2 Classification

Once the drift has been compensated, corrected samples can be classified. State-of-the-art classifiers (e.g., k-NN , Random Forests , etc. [51]) can be applied in this phase without need of modifications to the standard implementations . The possibility of working with any type of external classifier represents one of the strengths of the proposed method, allowing to choose the best PaRCmodel based on the specific application.

6.2.3 Correction Factor Optimization

The correction matrix $\mathbf{M}_i$, used to correct samples of a window W_i, is continuously adapted when passing from a window to the next one. The overall goal of this optimization process is to update $\mathbf{M}_i$ on the basis of the information provided by the samples of W_i in order to follow the evolution of the drift and therefore be prepared for the analysis of the next window W_{i+1}.

The adaptation is obtained using the CMA-ES, a stochastic population-based search method in continuous search spaces, aiming at minimizing an objective function $f : \mathscr{S} \subseteq \mathbb{R}^p \to \mathbb{R}$ in a black-box scenario (see 6.4 for specific details).

In our specific application the solution computed by the CMA-ES during the elaboration of the window W_i identifies the candidate correction matrix for the window W_{i+1} ($\mathbf{M}_{i+1}$). We denote with $\mathbf{M}^{\mathbf{s}}$ the correction matrix obtained from the solution $\mathbf{s} \in \mathscr{S} \subseteq \mathbb{R}^{p=n \cdot n}$ by computing each element as follows:

$$M^{\mathbf{s}}[i][j] = s[(i-1) \cdot n + j], (i \in [1,n], j \in [1,n]) \tag{6.4}$$

The objective function applied in the optimization process, computed considering the set of samples belonging to window W_i, is expressed as:

$$f_i(\mathbf{s}) = \sum_{j=0}^{|W_i|-1} D\left(\mathbf{x}_{i,j} + \mathbf{M}^{\mathbf{s}} \times \mathbf{x}_{i,j}, \mu_{\mathscr{C}(\mathbf{x_{i,j}})}\right) \tag{6.5}$$

It computes the sum of the distances (D) of each corrected sample in W_i ($\mathbf{x}_{i,j} + \mathbf{M}^{\mathbf{s}} \times \mathbf{x}_{i,j}$) from the centroid of the related class in the training set ($\mu_{\mathscr{C}(\mathbf{x_{i,j}})}$). The centroid of a class y is computed as follows:

$$\mu_y = \frac{\sum_{i=1}^{|y|} \mathbf{t}_i^y}{|y|} \tag{6.6}$$

where $|y|$ is the number of training samples for the class y and $\mathbf{t}_i^y$ is the i^{th} training sample for the class. The function $f_i(\mathbf{s})$ tries to measures how corrected samples tend to deviate from the class distributions learnt by the classifier during the calibration phase.

The evolutionary process stops the optimization based on the following stop conditions:

1. The optimum value of the objective function has been reached. Depending on the type of distance function D considered in equation 6.5 (see Section 6.2.4), the optimal value of the objective function can be set to zero or to a lower bound indicating that all corrected samples have been collapsed into a region closed to the the centroid of the class they belong to. Due to the complexity of the optimization process this condition cannot be always reached;
2. During the optimization all candidate solutions in the current population P_c have a value of the objective function differing from that of the other candidates less than a predefined threshold ω_{min}:

$$f_i(\mathbf{s_x}) - f_i(\mathbf{s_y}) < \omega_{min} \qquad \forall x, y \in P_c \tag{6.7}$$

3. The step size σ_{cur} of the CMA-ES (see 6.4) increases more than a predefined threshold $\bar{\sigma}_{max}$ with respect to its initial value σ_{ini}, i.e., the optimization process is trying to explore an area in the search space that is too large; or σ_{cur} decreases more than a predefined threshold $\bar{\sigma}_{min}$, i.e., the optimization process is trying to explore a local minima:

$$\begin{cases} |\sigma_{ini} - \sigma_{cur}| > \bar{\sigma}_{max} \\ |\sigma_{ini} - \sigma_{cur}| < \bar{\sigma}_{min} \end{cases} \tag{6.8}$$

   The initial step size σ_{ini} is used to sample the search space around an initial search point (i.e., a randomly chosen value or a previous solution).

Together with the three defined stop conditions, the optimization is also interrupted if a maximum number of generations has been reached.

6.2.4 Distance Functions

Four types of distances have been used in this work to compute the objective function of equation 6.5:

- *Mahalanobis distance*: the Mahalanobis distance computes the distance between two samples by taking into account how samples distribute in the space. It allows to overcome problems deriving by non spherical distributions of samples:

$$D_m(\mathbf{x}, \mu_c) = \sqrt{(\mathbf{x} - \mu_c) \cdot \mathbf{Cov}^{-1} \cdot (\mathbf{x} - \mu_c)^T} \tag{6.9}$$

  where $\mathbf{Cov}^{-1}$ is the inverse of the covariance matrix for the samples of the training set of class c.
- *Exponential distance*: the Mahalanobis distance of the sample $\mathbf{x}$ is exponentially scaled, as follows:

$$D_x(\mathbf{x}, \mu_c) = e^{D_m(\mathbf{x}, \mu_c)} \tag{6.10}$$

  It exponentially penalizes samples that are moved far from the related centroid.
- *Linear step distance*: the distance of the sample $\mathbf{x}$ from the centroid of its class is computed as a step function as follows:

$$D_{ls}(\mathbf{x}, \mu_c) = \begin{cases} 0 & 0 \leq D_m(\mathbf{x}, \mu_c) \leq D^c_{m_{max}} \\ \frac{D_m(\mathbf{x}, \mu_c)}{D^c_{m_{max}})} - 1 & D^c_{m_{max}} < D_m(\mathbf{x}, \mu_c) \leq 2D^c_{m_{max}} \\ 10^3 & 2D^c_{m_{max}} < D_m(\mathbf{x}, \mu_c) \end{cases} \tag{6.11}$$

  where $D^c_{m_{max}}$ is the maximum Mahalanobis distance of samples of the training set of the class c from the related centroid. This step function gives maximum importance to samples close to the centroid of the related class ($D_{ls}(\mathbf{x}, \mu_c) = 0$) while strongly penalizes samples that are moved far from the centroid of the related class ($D_{ls}(\mathbf{x}, \mu_c) = 10^3$). In the region between the two cases the distance is increased linearly.
- *Exponential step distance*: similarly to D_{ls} the distance is computed as a step function as follows:

$$D_{xs}(\mathbf{x}, \mu_c) = \begin{cases} 0 & 0 \leq D_m(\mathbf{x}, \mu_c) \leq D^c_{m_{max}} \\ e^{Dm(\mathbf{x}, \mu_c)} & D^c_{m_{max}} < D_m(\mathbf{x}, \mu_c) \leq 2D^c_{m_{max}} \\ e^{2Dm^c_{max}} & 2D^c_{m_{max}} < D_m(\mathbf{x}, \mu_c \end{cases} \tag{6.12}$$

  the main difference w.r.t. D_{ls} is the way samples far from the centroid are penalized.

The choice of the best distance function to use depends on the considered dataset. This represents a degree of freedom that allows to tune the drift correction system for the specific application.

6.3 Case Studies and Experimental Results

The proposed methodology has been validated on a set of experiments performed on two datasets: the first composed of simulated data artificially generated, while the second composed of samples obtained from a real application.

The full correction system has been implemented as a combination of Perl and C code. A pool of four classifiers have been considered: k-Nearest Neighbors (*k*NN), Partial Least Square Discriminant Analysis (PLS), Neural Networks (NNET) and Random Forest (RF). All classifiers have been implemented using the Classification And REgression Training (CARET) package of *R*, a free and multi-platform programming language and software environment widely used for statistical software development and data analysis. Details on the specific implementation of the classifiers are available in [92]. The performance of the prediction model of each classifier has been tuned and optimized by performing leave-group-out-cross-validation (LGOCV). For each classifier 50 folds of the training set have been generated with 95% of samples used to train the model while the remaining ones used as test data. The size of the grid used to search the tuning parameters space for each classifier (e.g., k for *K*NN) has been set to 5. This represents a good compromise in terms of computational time of the training phase and optimization results.

Table 6.1 Parameters resulting from the tuning of each classifier

Optimal classifiers parameters for artificial data set			
Classifier	**Parameter**	**Description**	**Value**
*k*NN	*k*	Number of nearest neighbors	37
PLS	ncomp	Number of components one wishes to fit	4
NNET	size	Number of units in the hidden layer	3
	decay	Parameter of weight decay	0.1
RF	mtry	Number of variables randomly sampled as candidates at each split	2
Optimal classifiers parameters for real data set			
Classifier	**Parameter**	**Description**	**Value**
*k*NN	*k*	Number of nearest neighbors	21
PLS	ncomp	Number of components one wishes to fit	6
NNET	size	Number of units in the hidden layer	5
	decay	Parameter of weight decay	0.03
RF	mtry	Number of variables randomly sampled as candidates at each split	4

The optimal parameters obtained from the classifiers tuning phase are reported in table 6.1.

6.3.1 Artificial Dataset

For a preliminary evaluation study we tested the proposed drift correction methodology on simulated data composed of a given number of independent, uncorrelated and randomly distributed Gaussian clusters. The Gaussian model is often regarded as a benchmark in literature for gas chemical sensors data analysis [54]. It therefore provides an effective platform for testing the validity of the proposed approach. Simulated data allow to control the parameters influencing the drift correction capability such as, feature space dimensionality n, number of classes m, separation among clusters α (given in standard deviation units) and drift direction/intensity.

6.3.1.1 Experimental Setup

We considered a data set of 1000 samples belonging to 5 different classes ($m = 5$). Each sample includes 6 features ($n = 6$) simulating a sensor array composed of 6 sensors. The centroid of each class c is randomly drawn according to a multivariate normal distribution in n dimensions $\mu_c = \mathscr{N}\left(\mathbf{0}, \frac{\alpha^2}{2n}\mathbf{I}\right)$ ($\alpha = 12$ in our specific case). Using the term $\frac{\alpha^2}{2n}$ as scaling factor of the variance, the expectation value of the square distance between any two centroids is equal to α^2 independently of n. This allows to have enough separation among classes to build efficient classifiers. In order to control the minimum clusters separation we discarded simulations where, due to the randomness of the process, any two centers are closer than $\alpha/2$. For each class, we generated 250 Gaussian distributed samples with unit variance affected by a drift linear in time according to the following equation:

$$\mathbf{x}(c,t) = \mathscr{N}(\mu_c, \mathbf{I}) + \underbrace{\left(\frac{t}{h} \cdot \mathbf{u_d}\right)}_{\text{drift effect}} \tag{6.13}$$

where h represents a scaling factor for the discrete time t (h has been set to 40 in our specific case to guarantee a significant amount of drift). The term $\mathbf{u_d}$ represents a randomly generated unitary vector in the n-dimensional space describing the direction of the drift applied to each sample of the dataset. In our simulated data all classes are linearly drifted in the same direction, and samples of the different classes are uniformly distributed in time to present similar drift conditions. The effect of the drift is evident by looking at the projection over the first two principal components of the PCA reported in Figure 6.2.

The experimental session included 100 runs of the drift-correction process for each of the four objective functions based on the distances introduced in Section 6.2.4. The first 100 samples of the data set have been used as training data for the PaRCmodel, while the remaining 900 samples have been used as test set to be analyzed. The test set has been processed splitting the data in windows of 50 samples.

6.3.1.2 Results and Discussion

Table 6.2 shows the performance of the proposed system for the five considered classifiers and the four considered objective functions. Results are provided in terms of classification rate on each of the 18 windows and total classification rate (T. Cr.). To better highlight the benefits of the correction process, Table 6.2 reports both the classification rate of each classifier when no correction is applied and the one

Table 6.2 Performance of the drift correction system in terms of classification rate on the artificial data set

Classifier		**Classification rate over windows W_i**																		**T.Cr**
		W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9	W_{10}	W_{11}	W_{12}	W_{13}	W_{14}	W_{15}	W_{16}	W_{17}	W_{18}	
		Classifiers without drift correction																		
*k*NN		1.00	0.96	0.98	0.88	0.88	0.80	0.90	0.78	0.78	0.68	0.66	0.62	0.68	0.60	0.60	0.62	0.60	0.58	0.75
NNET		1.00	1.00	1.00	0.96	1.00	0.98	1.00	0.98	0.98	0.94	0.94	0.82	0.92	0.90	0.86	0.86	0.86	0.82	0.93
PLS		1.00	0.96	0.98	0.88	0.92	0.82	0.86	0.80	0.64	0.64	0.56	0.52	0.38	0.32	0.26	0.30	0.26	0.24	0.63
RF		0.86	0.88	0.86	0.80	0.80	0.80	0.78	0.76	0.76	0.74	0.64	0.62	0.62	0.56	0.48	0.48	0.44	0.46	0.68
		Drift correction using the Mahalanobis distance D_m																		**ΔT.Cr**
*k*NN	Avg	1.00	1.00	0.96	0.99	0.96	0.97	0.95	0.95	0.97	0.95	0.94	0.91	0.89	0.86	0.84	0.81	0.79	0.78	+0.17
	C.I.	.006	.015	.012	.015	.016	.020	.019	.021	.024	.026	.026	.027	.027	.028	.030	.029	.029	.030	
NNET	Avg	0.98	1.00	0.99	0.96	0.97	0.99	0.98	0.99	0.99	0.95	0.97	0.92	0.94	0.91	0.91	0.89	0.84	0.85	+0.02
	C.I.	.002	.000	.003	.003	.003	.002	.004	.002	.002	.013	.012	.015	.015	.015	.017	.020	.026	.029	
PLS	**Avg**	**1.00**	**1.00**	**1.00**	**0.96**	**0.99**	**0.97**	**0.98**	**0.98**	**0.99**	**0.98**	**0.99**	**0.96**	**0.94**	**0.94**	**0.90**	**0.84**	**0.80**	**0.80**	**+0.31**
	C.I.	**.001**	**.000**	**.001**	**.005**	**.004**	**.005**	**.006**	**.005**	**.002**	**.004**	**.006**	**.013**	**.015**	**.017**	**.020**	**.025**	**.030**	**.031**	
RF	Avg	0.99	0.97	1.00	0.99	0.91	0.88	0.84	0.93	0.87	0.85	0.84	0.81	0.83	0.82	0.81	0.82	0.80	0.79	+0.19
	C.I.	.002	.002	.001	.002	.007	.006	.005	.011	.008	.007	.006	.005	.006	.006	.009	.006	.010	.011	
		Drift correction using the linear step distance D_{ls}																		**ΔT.Cr**
*k*NN	Avg	0.98	0.92	0.94	0.88	0.92	0.89	0.88	0.86	0.88	0.85	0.83	0.81	0.79	0.78	0.76	0.76	0.73	0.73	+0.09
	C.I.	.006	.015	.012	.015	.016	.020	.019	.021	.024	.026	.026	.027	.027	.028	.030	.029	.029	.030	
NNET	Avg	0.97	0.92	0.94	0.88	0.89	0.87	0.87	0.85	0.86	0.82	0.82	0.78	0.79	0.77	0.76	0.76	0.73	0.71	-0.10
	C.I.	.005	.013	.011	.015	.019	.022	.021	.024	.025	.025	.029	.028	.031	.029	.031	.032	.034	.036	
PLS	Avg	0.98	0.91	0.93	0.88	0.91	0.87	0.84	0.84	0.82	0.78	0.76	0.74	0.70	0.67	0.66	0.66	0.62	0.61	+0.16
	C.I.	.006	.016	.013	.016	.020	.023	.025	.026	.032	.034	.037	.039	.040	.041	.043	.043	.044	.045	
RF	Avg	0.98	0.92	0.94	0.92	0.87	0.85	0.81	0.81	0.79	0.78	0.77	0.75	0.74	0.73	0.72	0.72	0.69	0.68	+0.12
	C.I.	.006	.008	.009	.012	.015	.016	.014	.018	.016	.017	.017	.017	.020	.019	.021	.024	.026	.026	
		Drift correction using the exponential distance D_x																		**ΔT.Cr**
*k*NN	Avg	1.00	0.99	0.98	0.92	0.95	0.92	0.90	0.88	0.88	0.84	0.83	0.80	0.77	0.77	0.76	0.75	0.70	0.69	+0.10
	C.I.	.000	.002	.003	.015	.015	.016	.018	.022	.024	.028	.028	.031	.032	.032	.033	.033	.035	.035	
NNET	Avg	1.00	1.00	1.00	0.94	0.96	0.92	0.80	0.75	0.77	0.72	0.70	0.66	0.66	0.64	0.64	0.62	0.60	0.59	-0.16
	C.I.	.000	.002	.002	.007	.014	.020	.034	.040	.038	.039	.039	.037	.038	.035	.037	.035	.035	.037	
PLS	Avg	1.00	1.00	0.99	0.93	0.95	0.89	0.83	0.79	0.77	0.74	0.72	0.67	0.65	0.66	0.63	0.60	0.57	0.55	+0.14
	C.I.	.000	.000	.003	.013	.014	.024	.031	.034	.037	.038	.037	.040	.038	.037	.040	.041	.040	.038	
RF	Avg	0.92	0.90	0.91	0.90	0.84	0.82	0.80	0.83	0.78	0.76	0.74	0.72	0.70	0.69	0.66	0.67	0.64	0.62	+0.09
	C.I.	.006	.008	.011	.016	.015	.016	.013	.021	.020	.024	.022	.028	.028	.029	.029	.030	.031	.032	
		Drift correction using the exponential step distance D_{xs}																		**ΔT.Cr**
*k*NN	Avg	1.00	0.99	0.98	0.92	0.92	0.90	0.91	0.88	0.89	0.87	0.86	0.81	0.80	0.79	0.78	0.76	0.73	0.74	+0.11
	C.I.	.000	.003	.006	.011	.016	.016	.016	.019	.021	.023	.024	.027	.025	.028	.028	.026	.027	.028	
NNET	Avg	0.98	0.98	0.98	0.93	0.94	0.94	0.94	0.93	0.92	0.89	0.88	0.82	0.82	0.81	0.80	0.80	0.77	0.76	-0.05
	C.I.	.001	.004	.005	.007	.014	.016	.015	.016	.016	.019	.025	.026	.028	.030	.030	.032	.033	.037	
PLS	Avg	0.99	0.99	0.99	0.94	0.93	0.91	0.91	0.90	0.88	0.87	0.85	0.82	0.79	0.75	0.73	0.72	0.69	0.67	+0.22
	C.I.	.002	.003	.003	.013	.017	.020	.019	.023	.026	.026	.028	.032	.032	.034	.037	.036	.038	.041	
RF	Avg	0.99	0.96	0.99	1.00	0.95	0.95	0.92	0.95	0.91	0.89	0.87	0.83	0.83	0.83	0.80	0.81	0.79	0.76	+0.21
	C.I.	.002	.002	.003	.002	.008	.011	.013	.012	.017	.017	.015	.016	.015	.019	.019	.019	.017	.019	

considering the correction system. Results for the correction system are produced in terms of average classification rate over the 100 considered runs (Avg). In order to evaluate the stability of the results over the different runs, for each average value is reported the related confidence interval (C.I.), computed considering 95% level of confidence. The total classification rate is expressed in this case as the variation w.r.t. the one of the classifier without correction.

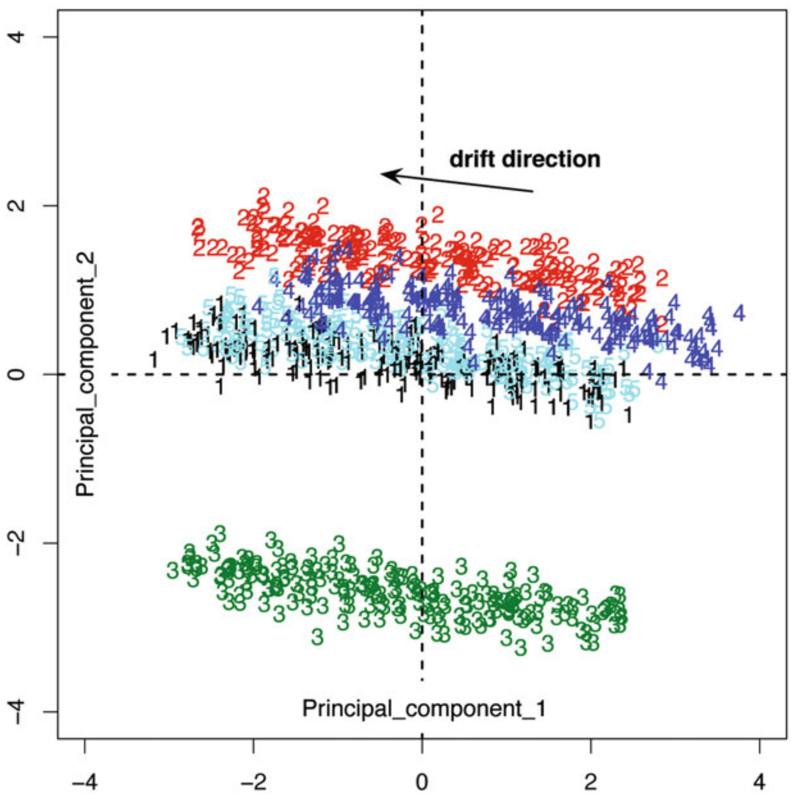

Fig. 6.2 Projection of the first two principal components of the PCA computed for the artificially generated dataset.

Results provided in Table 6.2 confirm that in general, for all considered classifiers, the drift correction process allows to improve the classification rate with results that are quite stable over the different runs. In particular, the two objective functions based on the Mahalanobis distance (D_m) and the exponential step distance (D_{xs}) seem to provide better results. Among the different classifiers, NNET gained lower improvement due to the fact that the classification rate was already quite high even without applying any correction. On the contrary, we observed the most significant improvement w.r.t. the classifier, when applied on raw measures, with PLS system corrected with the objective function based on the Mahalanobis distance.

Figure 6.3 graphically compares the performance of the proposed drift correction method with the Orthogonal Signal Correction (OSC) that, as introduced in Section 6.1, represents a state-of-the-art attuning method to perform drift correction. OSC has been implemented using the osccalc.m function of the PLS toolbox package (ver. 5.5) for MATLAB environment (64 bit, ver. 7.9). For the experiments we chose to remove one orthogonal component. Results are evaluated considering the PLS classifier corrected with the objective function based on the Mahalanobis distance. Since the size of the training set strongly influences the effectiveness of this approach, we provided results considering different values for the training set size (100 samples for osc-100 and 200 samples for osc-200) [115]. The proposed results clearly show how the proposed method outperforms the OSC requiring a reduced set of training data.

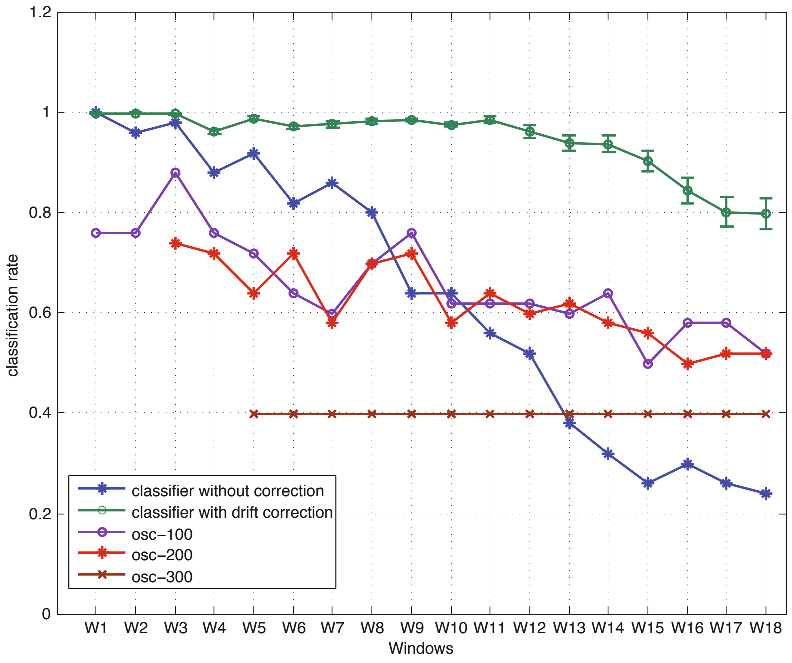

Fig. 6.3 Comparison of the proposed drift correction systems with the OSC for the PLS classifier with the objective function using the Mahalanobis distance D_m.

Finally, to show the ability of the correction process to actually remove the drift component from the considered samples, Figure 6.4 graphically shows the projection over the first two principal components for the corrected dataset for one of the runs performed with the PLS classifier using the Mahalanobis distance(Figure 6.4-a) and for the original data without drift (Figure 6.4-b). The last set of data was stored during the generation of the artificial dataset before inserting the drift component (see equation 6.13). Both plots have been generated using the same projection to allow comparison. The figure confirms how the drift observed in Figure 6.2 has been strongly mitigated allowing a distribution of samples that approximate the one without drift. This is an important results allowing to perform quantitative gas analysis and further examinations on the corrected data overcoming one of the main problems of previous adaptive correction methods (see Section 6.1).

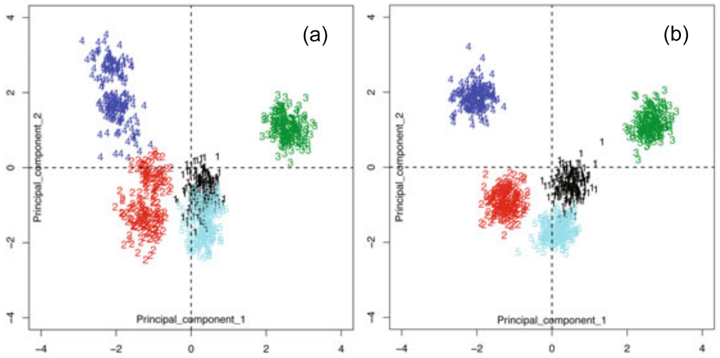

Fig. 6.4 Comparison of the corrected data set (a) with the original data without drift for the artificial data set (b), using PLS classifier

6.3.2 *Real Dataset*

To additionally validate the proposed approach we also performed a set of experiments on a real data set collected at the *SENSOR Lab*, an Italian research laboratory specialized in the development of chemical sensor arrays[1]. All data have been collected using an EOS835 *electronic nose* composed of 6 chemical MOX sensors: further information on sensors and equipment used can be found in [118] and its references. The goal of the experiment is to determine whether the EOS835 can identify five pure organic vapors: ethanol (class 1), water (class 2), acetaldehyde (class 3), acetone (class 4), ethyl acetate (class 5). All these are typical chemical compounds to be detected in real-world applicative scenarios.

6.3.2.1 Experimental Setup

A total of five different sessions of measurements were performed over one month to collect a dataset of 545 samples, a high value compared to other real datasets reported in the literature. While the period of time was not very long, it was enough to obtain data affected by a certain amount of drift. Not all classes of compounds have been introduced since the first session mimicking a common practice in real-world experiments: samples of classes 1 and 2 have been introduced since the beginning;

[1] `http://sensor.ing.unibs.it/`

class 3 is first introduced during the second session, one week later; first occurrences of classes 4 and 5 appear only during the third session, 10 days after the beginning of the experiment. Classes are not perfectly balanced in terms of number of samples, with a clear predominance of classes 1, 2 and 3 over classes 4 and 5. All these peculiarities make this dataset complex to analyze allowing us to stress the capability of the proposed correction system. The effect of the drift is evident by looking at the projection over the first two principal components of the PCA reported in Figure 6.5.

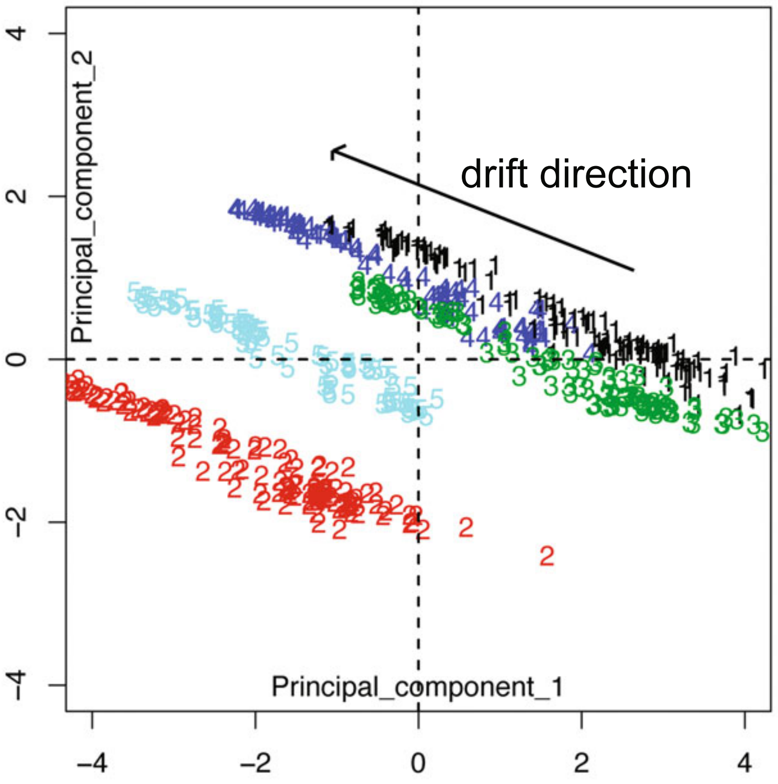

Fig. 6.5 Projection of the first two principal components of the PCA computed for the real dataset.

As for the artificial dataset the experimental session included 100 runs of the drift-correction process for each of the four considered objective functions. The first 20 samples of each class have been used as training data for the PaRCmodel, while the remaining 445 samples have been used as test set. The drift correction process has been applied to windows of 100 samples, with the last one of 45 samples. The bigger size of the windows compared to the artificial dataset is required to tackle the additional complexity of the real data.

6.3.2.2 Results

Table 6.3 summarizes the performance of the drift correction system on the real data set.

Results immediately highlight how the correction process for this particular experiment is harder than that for the artificial data. Main difficulties are connected to the fact that samples from different classes are introduced non homogeneously over the time and the initial interclass distance among the centroids is not enough to avoid partial overlapping of the classes. Moreover, the increased size of the windows increases the effort required by the CMA-ES to compute the appropriate correction matrices. However, the exponential step distance steal produces interesting improvements in the classification rate. Looking also at the results of the artificial dataset this distance seems the best compromise to work with generic data.

PLS corrected with the objective function based on the exponential step distance is the classifier that gained better improvements. Figure 6.6 compares again the results for this case with the correction obtained applying the OSC. This time due to the limited amount of samples, a single case with 100 samples of training has been considered. Again the proposed drift correction approach performs better than the OSC.

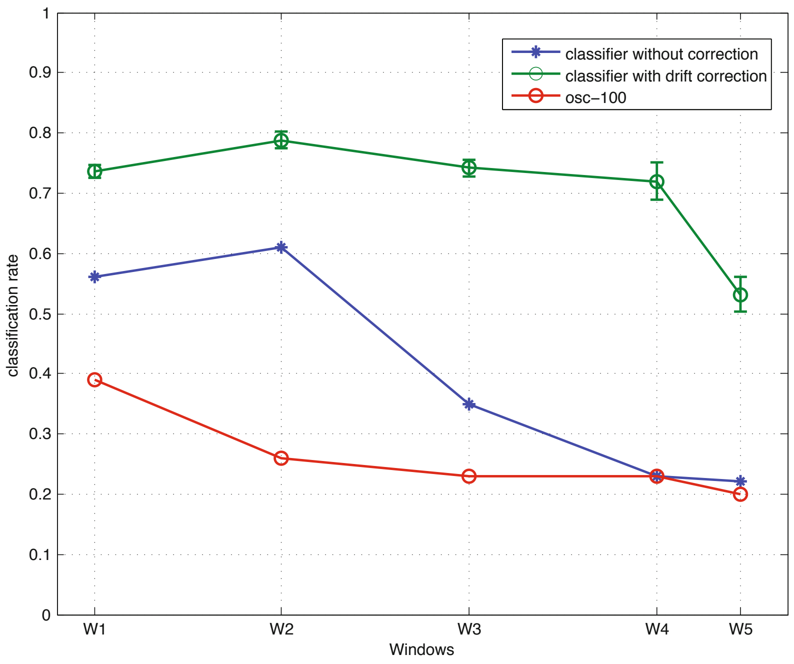

Fig. 6.6 Comparison of the proposed drift correction systems with the OSC for the PLS classifier with objective function using the exponential step distance D_{xs}

[!h]

Table 6.3 Performance of the drift correction system in terms of classification rate on the artificial real set

Classifier		**Classification rate over windows W_i**					**T.Cr**	**Classifier**		**Classification rate over windows W_i**					**T.Cr**
		W_1	**W_2**	**W_3**	**W_4**	**W_5**				**W_1**	**W_2**	**W_3**	**W_4**	**W_5**	
		Classifiers without drift correction Dm													
*k*NN		0.63	0.54	0.35	0.32	0.31	0.45								
NNET		0.56	0.65	0.63	0.47	0.36	0.55								
PLS		0.56	0.61	0.35	0.23	0.22	0.42								
RF		0.86	0.86	0.82	0.70	0.69	0.80								
		Drift correction using the Mahalanobis distance D_m					**ΔT.Cr**			**Drift correction using the exponential distance D_x**					**ΔT.Cr**
*k*NN	Avg	0.66	0.62	0.53	0.55	0.52	+0.13	*k*NN	Avg	0.20	0.24	0.28	0.22	0.27	-0.21
	C.I.	.004	.025	.015	.027	0.32			C.I.	.000	.009	.021	.017	.020	
NNET	Avg	0.28	0.54	0.52	0.50	0.49	-0.09	NNET	Avg	0.02	0.41	0.26	0.31	0.28	-0.30
	C.I.	.009	.009	.016	.022	.026			C.I.	.010	.022	.018	.029	.022	
PLS	Avg	0.40	0.41	0.28	0.30	0.37	-0.07	PLS	Avg	0.20	0.26	0.29	0.26	0.30	-0.16
	C.I.	.016	.015	.018	.021	.024			C.I.	.000	.015	.021	.020	.022	
RF	Avg	0.90	0.78	0.80	0.80	0.80	+0.02	RF	Avg	0.60	0.64	0.22	0.28	0.40	-0.36
	C.I.	.003	.003	.002	.000	.000			C.I.	.004	.023	.020	.029	.037	
		Drift correction using the linear step distance D_{ls}					**ΔT.Cr**			**Drift correction using the exponential step distance D_{xs}**					**ΔT.Cr**
*k*NN	Avg	0.89	0.72	0.55	0.56	0.49	+0.21	*k*NN	Avg	0.71	0.74	0.53	0.53	0.51	+0.16
	C.I.	.006	.022	.023	.033	.033			C.I.	.013	.013	.012	.018	.021	
NNET	Avg	0.81	0.70	0.71	0.68	0.54	+0.16	NNET	Avg	0.70	0.78	0.72	0.82	0.64	+0.19
	C.I.	.013	.023	.030	.046	.038			C.I.	.002	.006	.007	.022	.034	
PLS	Avg	0.86	0.67	0.53	0.55	0.41	+0.21	**PLS**	**Avg**	**0.74**	**0.79**	**0.74**	**0.72**	**0.53**	**+0.31**
	C.I.	.007	.017	.025	.031	.029			C.I.	.011	.013	.014	.031	.029	
RF	Avg	0.95	0.91	0.86	0.85	0.83	+0.08	RF	Avg	0.88	0.77	0.81	0.81	0.90	+0.02
	C.I.	.002	.012	.010	.023	.031			C.I.	.004	.002	.003	.005	.019	

6.4 CMA-ES

The covariance matrix adaptation evolution strategy (CMA-ES) is an optimization method first proposed by Hansen, Ostermeier, and Gawelczyk [72] in mid 90s, and further developed in subsequent years [71], [70].

Similar to quasi-Newton methods, the CMA-ES is a second-order approach estimating a positive definite matrix within an iterative procedure. More precisely, it exploits a *covariance matrix*, closely related to the inverse Hessian on convex-quadratic functions. The approach is best suited for difficult non-linear, non-convex, and non-separable problems, of at least moderate dimensionality (i.e., $n \in [10, 100]$). In contrast to quasi-Newton methods, the CMA-ES does not use, nor approximate gradients, and does not even presume their existence. Thus, it can be used where derivative-based methods, e.g., *Broyden-Fletcher-Goldfarb-Shanno* or *conjugate gradient*, fail due to discontinuities, sharp bends, noise, local optima, etc.

In CMA-ES, iteration steps are called *generations* due to its biological foundations. The value of a generic algorithm parameter y during generation g is denoted with $y^{(g)}$. The mean vector $\mathbf{m}^{(g)} \in \mathbb{R}^n$ represents the favorite, most-promising solution so far. The *step size* $\sigma^{(g)} \in \mathbb{R}_+$ controls the step length, and the *covariance matrix* $\mathbf{C}^{(g)} \in \mathbb{R}^{n \times n}$ determines the shape of the distribution ellipsoid in the search space. Its goal is, loosely speaking, to fit the search distribution to the contour lines of the objective function f to be minimized. $\mathbf{C}^{(0)} = \mathbf{I}$

In each generation g, λ new solutions $\mathbf{x}_i^{(g+1)} \in \mathbb{R}^n$ are generated by sampling a multi-variate normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{C})$ with mean $\mathbf{0}$ (see equation 6.14).

$$\mathbf{x}_k^{(g+1)} \sim \mathcal{N}\left(\mathbf{m}^{(g)}, \left(\sigma^{(g)}\right)^2 \mathbf{C}^{(g)}\right), k = 1, \ldots, \lambda \tag{6.14}$$

Where the symbol $\cdot \sim \cdot$ denotes the same distribution on the left and right side.

After the sampling phase, new solutions are evaluated and ranked. $\mathbf{x}_{i:\lambda}$ denotes the i^{th} ranked solution point, such that $f(\mathbf{x}_{1:\lambda}) \leq \ldots \leq f(\mathbf{x}_{\lambda:\lambda})$. The μ best among the λ are selected and used for directing the next generation $g+1$. First, the distribution mean is updated (see equation 6.15).

$$\mathbf{m}^{(g+1)} = \sum_{i=1}^{\mu} w_i \mathbf{x}_i^{(g)}, w_1 \geq \ldots \geq w_\mu > 0, \sum_{i=1}^{\mu} w_i = 1 \tag{6.15}$$

In order to optimize its internal parameters, the CMA-ES tracks the so-called *evolution paths*, sequences of successive normalized steps over a number of generations. $\mathbf{p}_\sigma^{(g)} \in \mathbb{R}^n$ is the conjugate evolution path. $\mathbf{p}_\sigma^{(0)} = \mathbf{0}$. $\sqrt{2}\frac{\Gamma\left(\frac{n+1}{2}\right)}{\Gamma\left(\frac{n}{2}\right)} \approx \sqrt{n} + \mathcal{O}\left(\frac{1}{n}\right)$ is the expectation of the Euclidean norm of a $\mathcal{N}(\mathbf{0}, \mathbf{I})$ distributed random vector, used to normalize paths. $\mu_{\text{eff}} = \left(\sum_{1=1}^{\mu} w_i^2\right)^{-1}$ is usually denoted as *variance effective selection mass*. Let $c_\sigma < 1$ be the learning rate for cumulation for the rank-one update of the covariance matrix; $d_\sigma \approx 1$ be the damping parameter for step size update. Paths are updated according to equations 6.16 and 6.17.

$$\mathbf{p}_\sigma^{(g+1)} = (1-c_\sigma)\mathbf{p}_\sigma^{(g)} + \sqrt{c_\sigma(2-c_\sigma)\mu_{\text{eff}}}\mathbf{C}^{(g)-\frac{1}{2}}\frac{\mathbf{m}^{(g+1)}-\mathbf{m}^{(g)}}{\sigma^{(g)}} \tag{6.16}$$

$$\sigma^{(g+1)} = \sigma^{(g)} \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\left\|\mathbf{p}_\sigma^{(g+1)}\right\|}{\sqrt{2}\frac{\Gamma\left(\frac{n+1}{2}\right)}{\Gamma\left(\frac{n}{2}\right)}} - 1\right)\right) \tag{6.17}$$

$\mathbf{p}_c^{(g)} \in \mathbb{R}^n$ is the evolution path, $\mathbf{p}_c^{(0)} = \mathbf{0}$. Let $c_c < 1$ be the learning rate for cumulation for the rank-one update of the covariance matrix. Let μ_{cov} be parameter for weighting between rank-one and rank-μ update, and $c_{\text{cov}} \leq 1$ be learning rate for the covariance matrix update. The covariance matrix $\mathbf{C}$ is updated (equations 6.18 and 6.19).

$$\mathbf{p}_c^{(g+1)} = (1-c_c)\mathbf{p}_c^{(g)} + \sqrt{c_c(2-c_c)\mu_{\text{eff}}}\frac{\mathbf{m}^{(g+1)}-\mathbf{m}^{(g)}}{\sigma^{(g)}} \tag{6.18}$$

$$\begin{aligned}\mathbf{C}^{(g+1)} &= (1-c_{\text{cov}})\mathbf{C}^{(g)} + \frac{c_{\text{cov}}}{\mu_{\text{cov}}} \\ &\times \left(\mathbf{p}_c^{(g+1)}\mathbf{p}_c^{(g+1)^T} + \delta\left(h_\sigma^{(g+1)}\right)\mathbf{C}^{(g)}\right) \\ &+ c_{\text{cov}}\left(1-\frac{1}{\mu_{\text{cov}}}\right)\sum_{i=1}^{\mu} w_i \operatorname{OP}\left(\frac{\mathbf{x}_{i:\lambda}^{(g+1)}-\mathbf{m}^{(g)}}{\sigma^{(g)}}\right)\end{aligned} \tag{6.19}$$

where $\operatorname{OP}(\mathbf{X}) = \mathbf{X}\mathbf{X}^{\mathbf{T}} = \operatorname{OP}(-\mathbf{X})$.

Most noticeably, the CMA-ES requires almost no parameter tuning for its application. The choice of strategy internal parameters is not left to the user, and even λ and μ defaults to acceptable values. Notably, the default population size λ is comparatively small to allow for fast convergence. Restarts with increasing population size has been demonstrated [8] useful for improving the global search performance, and it is nowadays included an an option in the standard algorithm.

6.5 Conclusions

In this chapter, we propose an evolutionary based approach able to deal with the drift problem affecting gas sensor arrays. The presented methodology is based on a 5-step flow that corrects and classifies the samples affected by sensor drift by applying a correction factor that mitigates the undesired effects on gas sensors. The correction factor is continuously adapted exploiting an evolutionary process, thus following the changes underwent by the sensor array due to the drift problem.

The proposed approach is flexible enough to work with different state-of-the-art classification algorithms, as experimentally demonstrated, and there is no need of relying upon complex drift models in order to exploit the proposed technique. Moreover, gathered results on artificial and real data sets experimentally corroborate that the proposed methodology performs better than state of the art methods, such as OSC.