2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

Optimal Bayesian Classification in Nonstationary Streaming Environments

Jehandad Khan, Nidhal Bouaynaya, Robi Polikar

***Abstract*—A novel method of classifying data drawn from a nonstationary distribution with drifting mean and variance is presented. The novelty of the approach is based on splitting the problem of tracking a nonstationary distribution into separate classification and time series state estimation problems. State space models for drift in both the mean and variance are presented, which are then successfully tracked using a Kalman filter and a particle filter for the linear and non-linear parts respectively. Preliminary results, which show the promising potential of the approach, are also presented, along with concluding remarks for potential uses of the proposed approach.**

I. Introduction

Most classification algorithms rely on the underlying assumption that the distribution generating the data is stationary. However, this is a very restricting assumption, since many real-world problems generate data whose underlying distributions change over time. Real world applications that generate such nonstationary data include climate change, remote-sensing applications, metagenomic applications (genomic analysis of environmental samples, where species abundance change dramatically along unknown environmental gradients), analysis of web-user interest, identification of financial fraud from transaction data, prediction of energy demand and pricing, among many others. Also relevant are installations with limited access (e.g., oil pipelines, building foundations, extreme geographic locations, etc.), where subsequent data later collected from embedded sensors can be subject to a variety of nonstationary changes; e.g., cracks from freeze-thaw cycles, shifting tectonic plates, etc. The stationarity assumption is often used to simplify the mathematical setting of the problem, and thus also simplify the derived solutions. However, this simplifying assumption forces the problem into a subspace of the original problem, often resulting in suboptimal solutions. Taking the nonstationary nature of the problem into consideration would allow us to take advantage of the full richness of the data, resulting in more accurate classification and prediction in tracking nonstationary environments.

The nonstationarity, also known as *concept drift*, can be treated using a variety of approaches such as domain adaptation [1] [2], covariate shift [3] or more generally as sample selection bias [4], or with specific ensemble based approaches such as Learn^{++}.NSE [5], DWM [6] and SEA [7]. These techniques acknowledge that the probability distribution which generated the data at any point in time is different from the probability distribution on which the classifier will make its prediction i.e., $p_s(x, y) \neq p_t(x, y)$ where p_s and p_t are the source and target distributions, respectively, for the features x and labels y. These approaches rely on different assumptions about the source and target distributions: for example, in covariate shift it is assumed that the support of $p_s(x, y)$ contains the support of $p_t(x, y)$ [8], thus the source and target distributions may be different but still are related. Moreover, it is also assumed that there is sufficient amount of labeled and unlabeled data available in the source and target domain, respectively.

The aforementioned algorithms also require a large amount of labeled data (at least from the source domain), rendering the availability or the high cost associated with obtaining labeled data a potential obstacle in using these approaches. In medical diagnostics, for example, it is highly desirable that the learning algorithm is trained using a minimum number of subjects, typically due to the scarcity of consenting subjects, the monetary cost associated with running diagnostic tests, or even the rarity of the disease. Semi Supervised Learning (SSL) has been used for such scenarios of limited availability of labeled training data, wherein the class information is propagated from small number of labeled data to more abundant unlabeled data instances [9] using such approaches as density separation, decision boundary detection or by constructing a graph. The primary focus of SSL techniques has been in stationary data environments, but there has been some recent advances that deal with data generated from nonstationary data distributions. These methods still have the canonical SSL implementation at their core with an exterior modification that caters for the drifting probability distributions. However, most such approaches still require that labeled data be available at each time point [10]. Active Learning (AL) is another approach used to tackle the limited data availability by selecting instances from the data that provide maximum information about the class boundaries, and then requesting the corresponding labels. AL algorithms rely on the imminent availability of the labels for any requested instances, an unrealistic expectation in certain applications.

Our work focuses on the nonstationarity of the source and target distributions generating the data, as well as the scarcity of labeled data instances. As stated above, these two problems are typically dealt with separately; but it is not unusual that both scenarios manifest themselves at the same time, hence

J. Khan, N. Bouaynaya and R. Polikar are with the Dept. of Electrical & Computer Engineering at Rowan University. (email: khanj6@students.rowan.edu, {bouaynaya,polikar}@rowan.edu).

This material is based upon work supported by the National Science Foundation grants ECCS-1310496, CRI CNS-0855248, EPS-0701890, EPS-0918970, MRI CNS-0619069, and OISE-0729792. This project is also supported by Award Number R01GM096191 from the National Institute Of General Medical Sciences (NIH/NIGMS).

978-1-4799-1484-5/14/$31.00 ©2014 IEEE

warranting concurrent solution. We have previously addressed this problem within the context of initially labeled streaming environment scenario (ILSE). In ILSE, very few labeled instances (e.g., 5% as suggested in [11]) are available initially, followed by a stream of unlabeled instances —with no future labeled instances —drawn from a nonstationary environment, for which we proposed the COMpacted POlytope Sample Extraction (COMPOSE) algorithm.

COMPOSE deals with the problem of nonstationary data stream using α-shapes. α shapes are geometrical constructs, generalizations of convex hulls, which are then compacted (shrunk) to determine the core (central) region that represent the space from which the current data points are most likely to be drawn at the next time step. These instances are called *core supports* and serve as the labeled instances to be propagated to the next time step, to help the next step's SSL algorithm in labeling the new unlabeled instances. Under the generally mild assumption of gradual drift, the region of probability distribution represented by these instances would be shared by the class distributions at both time points [12].

The COMPOSE algorithm as described in [12] is an intuitive computational geometry-based algorithm. In this contribution, we propose an entirely different statistical point of view for the same problem, namely, classification of streaming nonstationary data drawn from an ILSE. We consider the data as generated from a nonstationary stochastic process. Under the Gaussianity assumption, characterization of the nonstationary distribution is equivalent to determining the time-varying mean and variance, which provide the complete information regarding the time evolution of the distribution. Hence, we propose to formulate the nonstationary classification problem as a state estimation problem, the state vectors being the mean and variance of the data that are subsequently estimated using the Kalman filter or the particle filter, respectively, for each class of data. A Bayes classifier is employed for classification of newly received data, and the newly classified data are then used to update the mean and variance state estimates for all classes. This iterative procedure is repeated at each time point without the requirement of new labels in the future. Like COMPOSE, and its sister algorithm COMPOSE.AL [13] labeled data —if available at subsequent time steps —can easily be incorporated in the algorithm, but certainly is not a requirement and would only serve to improve the state estimates. Unlike COMPOSE, which relies on a geometrical construct (α-shapes) to develop a sense of the data, we here use a stochastic state space formulation to summarize the information contained in the data, giving a complete description of the underlying probability distributions and their dynamics.

The rest of this paper is organized as follows: In Section II we provide an overview of the problem along with the formulation of the state space models. In Section III, we discuss the Bayesian estimation framework and explain how we use the Kalman filter to track the mean of the distribution, and how a Gaussian time series may be transformed into a state space model with observable variance, and then a particle filter may be used to track it. Finally in Section IV we present the results of simulations using the proposed scheme. Finally, we provide our conclusion and some recommendations for future work.

II. Problem Formulation

We assume that each class of data is drawn from a time varying probability distribution $p(x(t), \theta(t))$ where $x(t)$ is the set of features and $\theta(t)$ is the set of time varying parameters at time t, which completely describe the probability distribution function. While, for the purposes of this approach, we assume that the data are generated from a Gaussian distribution, the proposed approach may be applied to any distribution if i) the distribution is completely defined by a set of parameters; ii) the parameters are related to the observed data in a state space formulation. For example, and in particular, any distribution that can be described by a Gaussian Mixture Models (GMM) can easily be represented using the proposed scheme. Without any loss of generality, we assume that the parameters θ are following a random walk model, and can therefore be written in a state space form as (1)

$$\theta(t) = \theta(t-1) + v(t) \quad (1)$$

$$x(t) = f(\theta(t)) + w(t) \quad (2)$$

where $v(t)$ is the source of Gaussian noise for the random walk, $f(\theta(t))$ is a function of the parameters of the distribution, and $w(t)$ is a random noise modeling the uncertainty of the observation $x(t)$. For Gaussian data, mean and variance constitute sufficient parameters since they completely define the distribution, though they cannot be written in a single state space for reasons described later.

At the initial time step ($t = 0$), and only at this initial time step, we receive labeled data instances for each class. Using these labeled instances we estimate the distribution parameters for each class. At the next time step, the distribution of each class is assumed to have changed in accordance with Equation (1), forming a new distribution namely $p(x(t+1), \theta(t+1))$. For example, if the data are Gaussian and the mean shifts by a small random amount, then any new data generated would be from a different distribution governed by the new mean. The random walk model is a very general framework for evolution dynamics, and is capable of modeling any random perturbation that the data distribution may undergo. The system proceeds in this manner generating new data at each time instance from a distribution with updated parameters. A graphic representation of the general procedure is shown in Figure 1.

Ideally, all parameters for a given class distribution should be governed by the same state dynamics and be observable using a single observation model. Such a model would enable the use of only one tracking algorithm. However, in the present scenario, i.e., Gaussian class distributions with the mean and variance drifting in a random walk, only the mean can readily be observed using a linear observation dynamics [14], while the variance cannot be directly observed from the data samples. A transformation of the problem is therefore necessary, as outlined in section III-B. Moreover, the mean observation dynamics is linear allowing us to estimate the distribution in

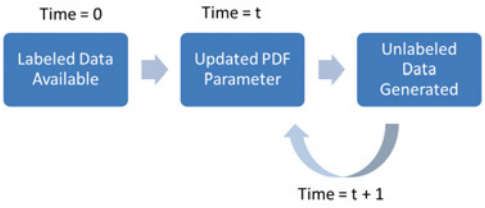


Fig. 1. The general problem description. At the initial time step only a set of labeled data are generated, thereafter the distribution undergoes transition in parameters while generating data at each time step.

an optimal manner using a Kalman Filter, whereas the variance of the class distribution does not afford this luxury, and hence requires an asymptotically optimal particle filter . We address both issues by splitting up the distribution parameter state space for the mean and variance with estimation procedures running separately; more specifically, the Kalman filter estimates the mean of the class distribution, and the particle filter estimates the variance of the class distribution as depicted in Figure 2.

Algorithm Outline

The proposed algorithm comprises the following steps: 1) the initial labeled data are used to estimate the initial values of the distribution parameters, i.e., the mean and variance; 2) each new batch of data received at time t is classified based on the previous estimates of the parameters using a Bayes classifier; 3) the mean of the data for each class is updated based on the newly classified data points; 4) the newly estimated mean is used to centralize the data around the mean; 5) the centralized data are then used to estimate the data variance. Thereafter the algorithm repeats with each subsequent arrival of new batch of data at the next time point. Figure 2 and Algorithm 3 outline these steps in their sequential order.

III. Bayesian Estimation of the Drifting Density

Bayesian estimation pertains to the estimation of system state based on its state space description. Most commonly, the estimation is done in a recursive manner with a prediction step and an update step. Under certain conditions, Bayesian estimation provides guarantees for the accuracy of the estimation and bounds on the errors. These guarantees make Bayesian estimation an attractive approach for time series problems. Kalman filter is a class of Bayesian state estimators for a linear system, perturbed by Gaussian noise, and it is a recursive minimum mean square error estimator, which provides the optimum estimate [14]. We now describe how we use the Kalman filter to track the mean of each class.

A. Kalman Filtering of the Drifting Mean

We assume no particular model for the drift of the distribution and model it as a random walk. The mean of each class distribution is modeled using the following dynamics

$$\mu_k^c = \mu_{k-1}^c + w_k^c \quad (3)$$
$$x_k^c = \mu_k^c + v_k^c \quad (4)$$

with

$$w_k^c \sim \mathcal{N}(0, Q)$$
$$v_k^c \sim \mathcal{N}(0, \sigma_k^c)$$

where k is the time index, $c = 0, 1, \cdots$ is the class index, μ_k^c and μ_{k-1}^c is the estimate of the mean at time k and $k-1$, respectively for the class c. x_k^c is the observed sample at time k for class c. Q is the variance of the random walk perturbation and σ_k^c is the variance of the c^{th} class distribution. A separate state space is maintained for each class of data and is tracked independently. Under the independence assumption of the features, each feature may be tracked independently in a similar manner. Hence, in Eqs. (3) and (4) μ_k^c is the mean of each feature considered independently.

The Kalman filter equations that solve the system described by Equations (3) and (4) are outlined in Algorithm 1, where $\hat{\mu}_{i|j}$ denotes the estimate of the mean μ at time i given observations up to time j, and $P_{i|j}$ is the error covariance matrix at time i given observations up to time j. K_k is the Kalman gain computed at each time point and σ_{k-1}^c is the variance of class c at time $k-1$. The Kalman filter runs at each time step when a new observation becomes available, thus giving us the estimate of the mean for each class at each time point k. Once the mean of the distribution is known, the next step is to estimate the variance of each class.

Algorithm 1 Tracking the mean using the Kalman Filter

1: *Initialization*

2: μ_0 = sample mean and σ_0^c = sample variance from the labeled data and $P_{0|0} = I$

3: *Prediction*

$$\hat{\mu}_{k+1|k} = \hat{\mu}_{k|k}$$
$$P_{k+1|k} = P_{k|k} + Q$$

4: *Measurement update*

$$K_k = P_{k+1|k}(P_{k+1|k} + \sigma_{k-1}^c)^{-1}$$
$$\hat{\mu}_{k+1|k+1} = \hat{\mu}_{k+1|k} + K_k(x_k - \hat{\mu}_{k+1|k})$$
$$P_{k+1|k+1} = (I - K_k)P_{k+1|k}$$

B. Particle Filtering of the Drifting Variance

Unlike the mean of a probability distribution, which can be directly estimated from the data samples we receive, the variance cannot be estimated from the data directly. This is because we cannot write in state space form the data that we have and the variance of the class distribution, as in Eq. (4). More specifically, we wish to have an observation dynamics that would relate each data element to the distribution variance.

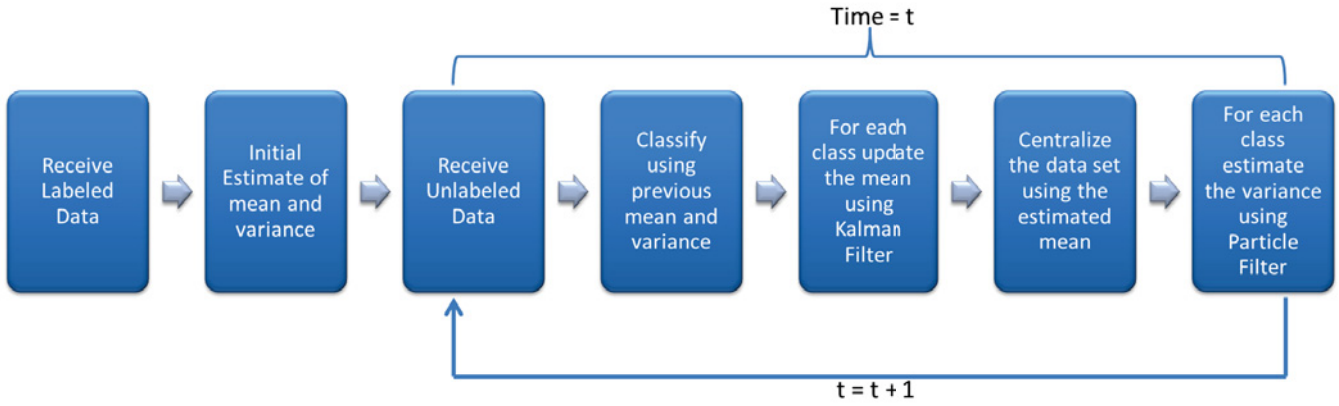


Fig. 2. The outline of the algorithm indicating each step necessary, the initial labeling steps and the steps required at each time point when new data is available to be classified.

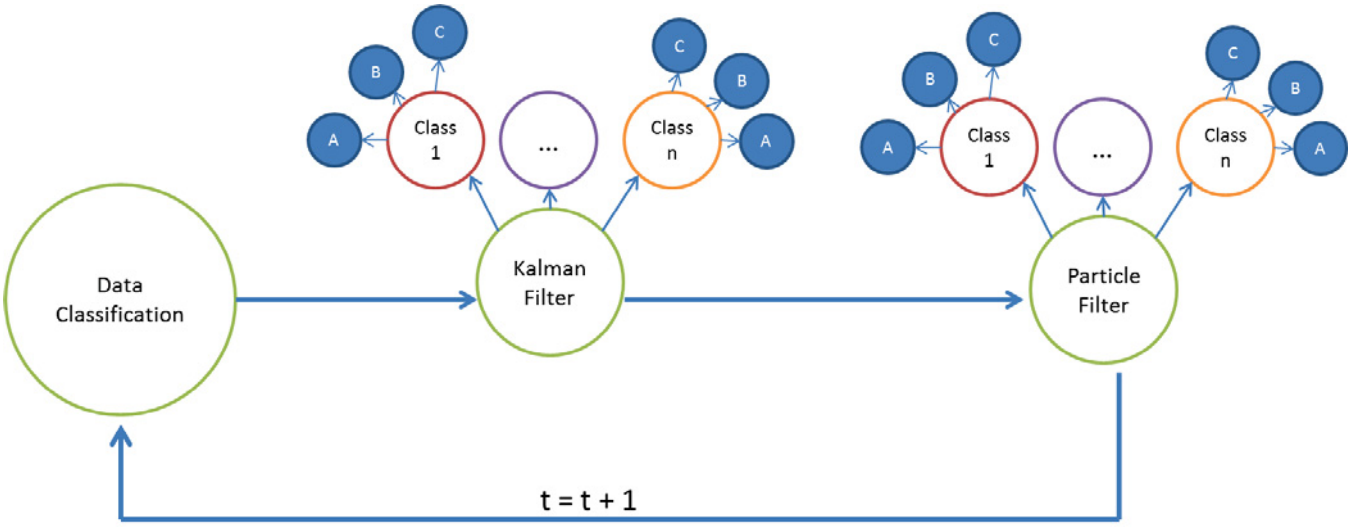


Fig. 3. The parameter estimates for each class may be updated in parallel both at the Kalman filter stage and the Particle filter stage at each time point. If the features are independent, they may also be processed in parallel shown above as the letters A, B and C, indicating the high level of parallelism of our approach.

Thus, we seek a functional mapping f that would relate to the observed data as

$$x(t) = f(\theta(t)) \tag{5}$$

where θ is the distribution parameter, variance in this case. To overcome this problem and make the variance of the data directly observable from the data, we follow the approach adopted in [15, Chapter 13]. The time-series is first transformed so that a state-space model with variance as the state can be formulated. However, in this process, the model is no longer a Gaussian; and hence the Kalman filter is no longer the optimum state estimator. We, therefore, use the particle filter, which is a class of Sequential Monte Carlo(SMC) methods, to estimate the variance in the non-Gaussian model [16].

This transformation requires the following assumptions: i) the features of the data are independent; and ii) the variance of data samples at subsequent time points is the same. The transformation squares the original data elements and adds them together according to the following formula:

$$s_m \quad = \quad \tilde{x}^2_{2m-1} + \tilde{x}^2_{2m}. \tag{6}$$

where m is the time index for the transformed series since two samples from the original time series contribute to only one observation. $\tilde{x}_m$ is the centralized data sample, where the mean estimated in the previous step of the algorithm is subtracted from the received data, to make it zero mean, and s_m becomes the data element for the transformed series. The requirement for zero mean data in this step necessitates that the Kalman

filter estimate of the mean be made available prior to the estimation of the variance, which enables the centralization of the data. Recall our assumption that the variance of two consecutive samples is the same, i.e., $\sigma^2_{2m-1} = \sigma^2_{2m}$. This assumption is somewhat similar to the gradual drift assumption adopted in COMPOSE [12], as well as many other concept drift algorithms.

Since the sum of square of two zero-mean Gaussian random variables follows the Chi-Squared distribution, we have the following distributions for the random variables mentioned in Eq. (6)

$$\begin{aligned} \tilde{x}_{2m-1} &\sim \mathcal{N}(0, \sigma^2_{2m-1}) \\ \tilde{x}_{2m} &\sim \mathcal{N}(0, \sigma^2_{2m}) \\ s_m &\sim \mathcal{X}^2(2) \end{aligned}$$

The probability distribution function for s_m is therefore given by

$$p(s_m) = \frac{1}{2\sigma_m} e^{(-\frac{s_m}{2\sigma_m})}.$$

We define a new random variable as the logarithm of s_m,

$$z_m = \log(\frac{s_m}{2}), \tag{7}$$

which then follows the double exponential distribution, with mean $t_m = \log(\sigma^2_m)$, pdf given by Eq. 8 and denoted as $\mathcal{D}(\mu = 0, \beta = 1)$.

$$g(z_m) = e^{(z_m - \log \sigma^2_m) - e^{(z_m - \log \sigma^2_m)}}. \tag{8}$$

The state space model for the transformed series z_m can now be written as

$$t_m = t_{m-1} + u_m \tag{9}$$

$$z_m = t_m + n_m \tag{10}$$

where

$$\begin{aligned} u_m &\sim \mathcal{N}(0, 1) \\ n_m &\sim \mathcal{D}(\mu = 0, \beta = 1) \end{aligned}$$

$t_m = \log(\sigma^2_m)$ is now the trend of the transformed time series. Thus, tracking t_m leads to the estimate of σ^2_m. The state space enumerated above is nonlinear since the observation dynamics are perturbed by a non-Gaussian noise. Therefore, we employ the particle filter [16] to track the system in (9).

C. Implementation of the Particle Filter

The particle filter (PF) is a Monte Carlo method that sequentially approximates a target density using a set of samples, called particles. Unlike the Kalman filter, which assumes the underlying distributions to be Gaussian and requires the state dynamics to be linear, the PF is a general Bayesian estimation framework that makes no assumptions about the state, observation dynamics or the distributions of the noise, rendering it a very general and powerful technique capable of solving a much wider and more general class of state estimation problems. PF is a numerical technique that estimates the posterior probability of the state in a recursive manner, where the integration necessary for the mean estimates is done using a Monte Carlo approximation.

Algorithm 2 gives the outline of our procedure adopting the particle filter for tracking the drifting variance. The PF samples from a *proposal density* denoted as $\pi(t)$, which expresses our belief in the likely location of the state. Based on the new observation received, we calculate the weight of each particle using (11). It is well known that, with the evolution in time, the particle weights tend to degenerate [16]; hence we determine the number of effective samples using (12), and resample the particles according to their weight distribution. This step is also known as the *survival of the fittest* since particles with higher weight are more likely to be resampled than the ones with lower weights. Finally, the weighted sample mean of the particles gives the posterior state estimate of the system. With each new observation that becomes available, these steps are repeated, and the variance estimate of each class is updated.

Algorithm 2 Tracking the variance using the particle filter

1: *Initialization* Sample i particles from the proposal density

$$\begin{aligned} \forall i &= 1, \cdots, N \\ t_m &\sim \pi(t_k | t^{(i)}_{0:m-1}, z_{0:m}) \end{aligned}$$

2: Update state particles according to state dynamics (Eq 9)

3: *Calculate the weight of each particle*

$$w^{*(i)}_m = w^{*(i)}_{m-1} \frac{p(z_m | t^{(i)}_m) p(t^{(i)}_m | t^{(i)}_{m-1})}{\pi(t^{(i)}_k | t^{(i)}_{0:m-1}, z_{0:m})} \tag{11}$$

4: *Normalize the weights*

$$\tilde{w}^{(i)}_m = \frac{w^{*(i)}_m}{\sum_{j=1}^{N} w^{*(j)}_m} \tag{12}$$

5: *Resample if required*

$$\hat{N}_{eff} = \frac{1}{\sum_{i=1}^{N} (\tilde{w}^{(i)}_m)^2}$$

if($\hat{N}_{eff} < N_{thresh}$) re-sample particles

6: *Estimate State*

$$t_m = \sum_{i=1}^{N} \tilde{w}^{(i)}_m t^{(i)}_m$$

D. Classification of Nonstationary Gaussian Data

Algorithm 3, and the following paragraphs, describe the complete outline of the proposed algorithm. The only input required to the algorithm is the number of classes in the data.

1) Initial Batch of Labeled Data: As soon as the first batch of labeled data becomes available, we compute the sample mean and covariance to determine the initial estimate of the mean and covariance. Hence, we have an initial estimate of the mean and covariance of each class separately; and thus,

Algorithm 3 Classification of nonstationary Gaussian Data

Input: Number of Classes: N_c
  Initial labeled data
  for each class $c = 0, 1, \cdots N_c$
    calculate sample mean μ_0^c and sample variance σ_0^c
  end for
  for $t = 1, 2, \cdots$
    receive unlabeled data
    call Bayes Classifier and assign class labels using mean μ_{t-1}^c and variance σ_{t-1}^c
    for each class $c = 1, \cdots N_c$
      call Kalman Filter to update mean μ_t^c
    end for
    for each class $c = 1, \cdots N_c$
      subtract mean μ_t^c from data
      transform data using Eq. (6) and (7)
      call particle filter to update variance σ_t^c
    end for
  end for

every class can be tracked independently. The parallelism of the problem in the number of classes simplifies both the analysis and renders the problem more amenable to parallel implementation. Figure 3 illustrates how different classes can be processed in parallel. At each tracking step, mean and variance of each class can be processed independently and in parallel. More details on the computational aspects of the algorithm are discussed in section IV-D.

2) Bayesian Classification: After the first batch of data is used to estimate the initial distributions of the classes, the algorithm is ready to accept the next batch of data. At each time step, we receive a new batch of data whose classes are unknown. Bayes Classifier is used to determine the classes of the incoming samples using the previously calculated mean and variance. The classes are assumed to be independent of each other; hence, the likelihood of a sample is determined as the product of the likelihoods corresponding to each class. Each sample is then assigned to the class with the maximum posterior probability. All new data instances are now split into sub-datasets for each class to be processed independently.

3) Kalman Filtering Step: Once the data has been classified into the different classes, the Kalman filter is used for each class to update its mean, as detailed in Section III-A.

4) Particle Filter Step: Once the mean for each class is available from the Kalman filter, the data are centralized for that class. Thereafter, the data are transformed into the new time series z_m using (6). The particle filter then provides the estimate for the class variance, hence, the algorithm iteratively updates the estimates for the mean and variance of each class, which are then used by the Bayes classifier to determine the class of each newly received data instance.

IV. Simulation Results

We designed two types of experiments to determine the effectiveness of the algorithm. Initially we consider only one class of data, to establish the tracking performance of the approach, which would later form the cornerstone of the classification step. Having ascertained tracking performance, we present results for a two-feature, two-class classification problem with stochastic variation in both the mean and variance, where the proposed algorithm successfully tracks the two classes and gives a classification accuracy that follows the optimal Bayes classifier - ran in a completely supervised manner with all parameters known - remarkably closely, despite the unsupervised and nonstationary nature of the setting.

A. Design of the Experiment

Bivariate Gaussian distributions for each class with means and variances that are constantly drifting over time are generated. The mean and variance of the pdf generating data belonging to each class is governed by the following equations.

$$\begin{bmatrix} \mu_x^c(t) \\ \mu_y^c(t) \end{bmatrix} = \begin{bmatrix} r\cos(\phi(t) + c \times (\pi/N_c)) + u_x \\ r\sin(\phi(t) + c \times (\pi/N_c)) + u_y \end{bmatrix} \quad (13)$$

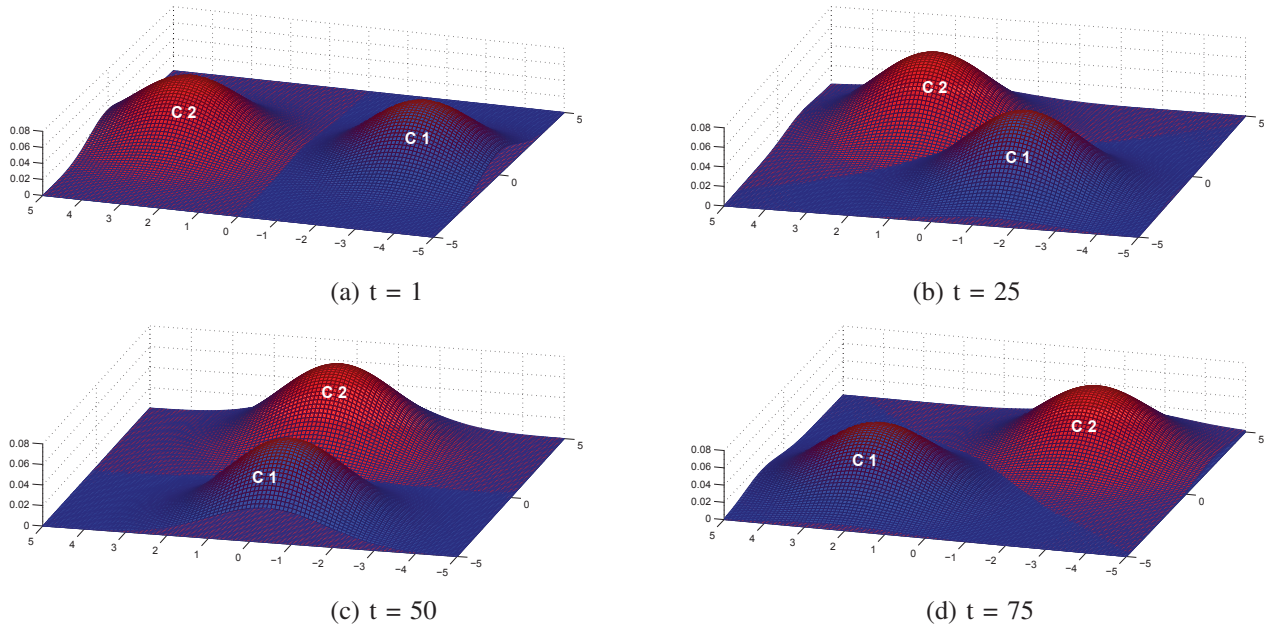
$$\sigma^c(t) = \sigma^c(t-1) + s(t) \quad (14)$$

where $c = 0, 1, \cdots$ is the class index, N_c is the total number of classes in the system, μ_x^c and μ_y^c are the x and y components of the mean of the c^{th}th class distribution, r is the radius of rotation, and u_x and u_y are zero-mean Gaussian perturbations to the deterministic dynamics, adding a measure of uncertainty to the evolution dynamics. ϕ is the time dependent angle of rotation which advances with each time step in a linear manner. $\sigma^c(t)$ is the variance of the c^{th} class distribution, which follows a random walk state update, perturbed by a zero mean Gaussian noise $s(t)$. Observe from Eq. (13) that the mean of this distribution rotates around the origin.

During the evolution of the simulation, the two distributions rotate about the origin, 180° out of phase, and by the end of the simulation the two distributions have completely replaced each other in the position of their means. We observed, as described below, that the proposed algorithm was easily able to track this nonlinear motion path as well as the variance giving a classification accuracy that closely matched that of the optimal Bayes classifier running in a supervised manner. Figure 4 (next page) illustrates the evolution of the probability distributions for the two classes during the simulation.

B. Tracking Performance

Since a large part of the algorithm pertains to tracking the mean and variance of the class distributions, it is important to assess the tracking accuracy of the algorithm. The state dynamics used by the Kalman and the particle filters are given by (4) and (9) respectively. For these experiments, the particle filter employed a zero mean Gaussian distribution with variance 10 as its initial proposal density, 1000 particles and a re-sampling threshold set to 80%. Only one class of data for the system described in section IV-A is (initially) generated for 100 time steps. Each time step has an epoch of 100 data instances. Figure 5 shows the percent tracking error during the evolution of the mean of the distribution in the x-axis



(a) t = 1 (b) t = 25

(c) t = 50 (d) t = 75

Fig. 4. The shifting probability distributions are shown during the different phases of the simulation

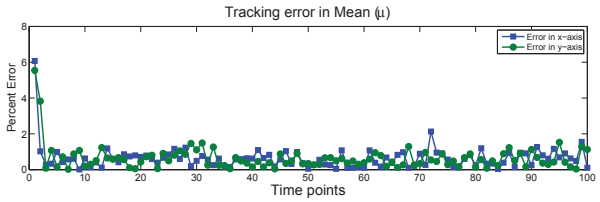


Fig. 5. Tracking performance of the algorithm for the mean of a single distribution (Showing both x and y components of the multivariate Gaussian distribution)

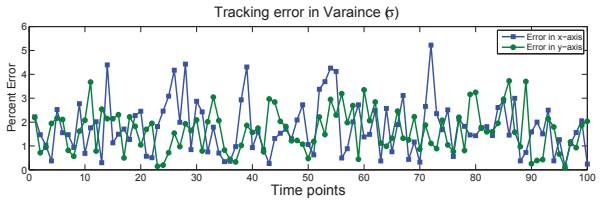


Fig. 6. Tracking performance of the algorithm for the variance of a single class distribution

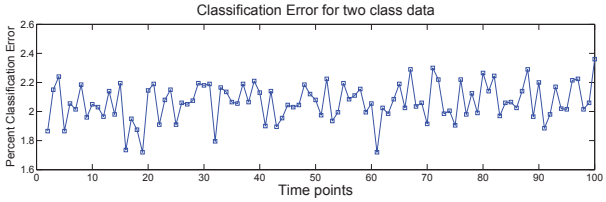


Fig. 7. Classification error over the course of time

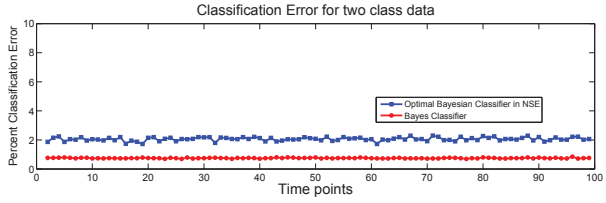


Fig. 8. Error comparison of the proposed algorithm with completely supervised Bayes Classifier

(blue) and y-axis (green), of the drifting two dimensional distribution. The algorithm is able to track the mean to an error of 2%. The error curves are averaged over 100 Monte Carlo runs. Successful tracking of the mean to a high degree of confidence is crucial because estimation of the variance relies on an accurate knowledge of the mean. Thus, errors in the estimation of the mean will propagate to the estimation of the variance. Figure 6 shows the tracking performance for the variance of the same distribution to an accuracy of 5% of the actual value of σ (variance).

C. Classification Performance

To test the classification performance of the data, two probability distributions based on the parameter evolution dynamics described in (13) were generated. At each time step, 100 data samples from each distribution were generated and shuffled to form a monolithic data stream (except for the first time step, t=0, where the labels were provided to initialize the algorithm). Thereafter, with each time step, the distribution parameters (mean and variance) were updated according to Eq. (13) and then sampled from unlabeled data only. The variance of the random walk components u_x and u_y in Equation (13) were set as 0.1 for both classes, while the variance of $s(t)$

was set as 0.05. The radius of rotation for the distributions r was chosen as 3 and the initial variance of both classes was set to 2. Once again 100 Monte Carlo runs were carried out. Figure 7 shows the mean percent classification error for Monte Carlo runs for these experiments. The performance is also compared to a completely supervised Bayes Classifier with complete knowledge of the mean and variance of the data at all times. Figure 8 shows that the algorithm is able to perform remarkably well - following the optimal Bayes classifier very closely - irrespective of such an unfair comparison against the proposed approach that runs in an unsupervised manner.

D. Computational Complexity

One of the major advantages of the proposed approach over COMPOSE [12] and other numerical techniques is its lower computational complexity. There are two computational aspects to the algorithm: the computation required for the Kalman filter, which must precede the particle filter, and the computation required for the particle filter itself. The computational operations required by the Kalman filter are matrix operations which can easily be performed on a general purpose computer using conveniently available software technology such as MATLAB. For a sufficiently high state dimension (of the order 1000) the algorithm can be coded in a High Performance Computer (HPC)-friendly high level language to leverage the available computational power. Moreover, instead of keeping track of a group of data elements, only the mean and variance are tracked, substantially reducing the memory requirements. The particle filter requires a fixed number of particles to track the state variable (variance in this case). The number of particles is closely related to the dimension of the state vector and the non-linearity or non-Gaussianity of the model, but not the amount of data, thus the memory requirements of the particle filter are also independent of the amount of data that need to be processed. This is in contrast to COMPOSE and other related methodologies which have a computational complexity related to both cardinality and the amount of the data [12] (in fact, COMPOSE is exponential in data dimensionality). The price paid, however, is the cost of the knowledge of the probability distributions generating the data. Depending on the application at hand this might be a restrictive assumption or may be easily managed if either the distributions are assumed to be Gaussian or known *a priori*. A Second prominent computational aspect of the proposed approach is its parallelism. As depicted in Figure 3, the approach can leverage the parallelism of modern computing systems by breaking the task at the class level, and if independent, even at the feature level. These assumptions are not uncommon in machine learning paradigms.

V. Conclusion and future work

We proposed a novel proof-of-concept approach for parametric tracking of drifting class distributions. A state space formulation for tracking the mean and variance of each class distribution is shown, for which simulation results have been quite promising in both tracking and classification. Our future work includes relaxing the independence assumption of the features for both the Kalman filter and the particle filter. Moreover, the assumption of two consecutive data samples having the same variance may also be relaxed. Future work also includes extending the proposed approach to Gaussian Mixture Models (GMMs) and general multimodal distributions.

References

[1] Joaquin Quionero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D Lawrence, *Dataset shift in machine learning*, The MIT Press, 2009.

[2] Anna Margolis, *Automatic Annotation of Spoken Language Using Out-of-Domain Resources and Domain Adaptation*, Ph.D. thesis, University of Washington, 2011.

[3] Hidetoshi Shimodaira, "Improving predictive inference under covariate shift by weighting the log-likelihood function," *Journal of statistical planning and inference*, vol. 90, no. 2, pp. 227–244, 2000.

[4] James J Heckman, "Sample selection bias as a specification error," *Econometrica: Journal of the econometric society*, pp. 153–161, 1979.

[5] Ryan Elwell and Robi Polikar, "Incremental learning of concept drift in nonstationary environments," *Neural Networks, IEEE Transactions on*, vol. 22, no. 10, pp. 1517–1531, 2011.

[6] J Zico Kolter and Marcus A Maloof, "Dynamic weighted majority: An ensemble method for drifting concepts," *The Journal of Machine Learning Research*, vol. 8, pp. 2755–2790, 2007.

[7] W Nick Street and YongSeog Kim, "A streaming ensemble algorithm (SEA) for large-scale classification," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, San Francisco, CA, USA*, 2001, pp. 377–382.

[8] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan, "A theory of learning from different domains," *Machine Learning*, vol. 79, no. 1-2, pp. 151–175, 2010.

[9] Olivier Chapelle, Bernhard Schölkopf, Alexander Zien, et al., *Semi-supervised learning*, vol. 2, MIT press Cambridge, 2006.

[10] Gregory Ditzler and Robi Polikar, "Semi-supervised learning in non-stationary environments," in *IEEE International Joint Conference on Neural Networks (IJCNN 2011), San Jose, California, USA*, 2011, pp. 2741–2748.

[11] Karl B Dyer and Robi Polikar, "Semi-supervised learning in initially labeled non-stationary environments with gradual drift," in *IEEE International Joint Conference on Neural Networks (IJCNN 2012), Brisbane, Australia*, 2012, pp. 1–9.

[12] Karl B Dyer, Robert Capo, and Robi Polikar, "Compose: A semisupervised learning framework for initially labeled nonstationary streaming data," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 1, pp. 12–26, 2014.

[13] Robert Capo, Karl B. Dyer, and Robi Polikar, "Active learning in nonstationary environments," in *IEEE International Joint Conference on Neural Networks (IJCNN 2013), Dallas, Texas, USA*, 2013, pp. 1–8.

[14] Dan Simon, *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*, Wiley. com, 2006.

[15] Genshiro Kitagawa, *Introduction to time series modeling*, CRC press, 2010.

[16] Arnaud Doucet and Adam M Johansen, "A tutorial on particle filtering and smoothing: Fifteen years later," *Handbook of Nonlinear Filtering*, vol. 12, pp. 656–704, 2009.