Semi-Supervised Learning in Initially Labeled Non-Stationary Environments with Gradual Drift

Karl B. Dyer and Robi Polikar
Signal Processing & Pattern Recognition Laboratory
Rowan University
Glassboro, NJ, USA
karl.b.dyer@gmail.com, polikar@rowan.edu

***Abstract*—Semi-supervised learning (SSL) in non-stationary environments has received relatively little attention in machine learning, despite a growing number of applications that can benefit from a properly configured SSL algorithm. Previous works in learning non-stationary data have analyzed such cases where both labeled and unlabeled instances are received at every time step and/or in regular intervals; however, to the best of our knowledge, no work has investigated the case where labeled instances are received only at the initial time step, followed by unlabeled instances provided in subsequent time steps. In this proof-of-concept work, we propose a new framework for learning in a non-stationary environment that provides only unlabeled data after the initial time step, to which we refer to as *initially labeled environment*. The proposed framework generates labels for previously unlabeled data at each time step to be combined with incoming unlabeled data – possibly from a drifting distribution - using a compacted polytope sample extraction algorithm. We have conducted two experiments to demonstrate the feasibility and reliability of the approach. This proof-of-concept is presented in two dimensions; however, the algorithm can be extended to higher dimensions with appropriate modifications.**

Keywords-alpha shape; concept drift; non-stationary environment; shape offsets; semi-supervised learning

I. INTRODUCTION

Real world applications that can benefit from machine learning algorithms capable of tracking a non-stationary environment (NSE) have increased substantially in the present digital age. Autonomous or streaming data, such as web usage, financial or climate data, energy usage and pricing, all generate massive amounts of data from constantly evolving sources.

Developing an algorithm that can learn from nonstationary environments is particularly challenging, as such an algorithm must adapt to changes in the underlying distributions over time. The changes in the underlying distributions – often referred to as *concept drift* – can be gradual or abrupt. They are also often assumed to be limited in nature, meaning the changes follow some structure, as completely random fluctuations cannot be learned [1]. However, lack of a standardized metric leaves the definition of limited drift up to each researcher.

Most algorithms designed for NSE have utilized supervised learning algorithms, which are heavily reliant on labeled data provided at each time step. Dependency on labeled data, which typically require human annotators for labeling, makes supervised learning costly, time consuming and often impractical to implement particularly in NSEs due to the streaming nature of the data. To reduce costs associated with gathering labeled data, semi-supervised learning (SSL) techniques are being explored. SSL algorithms utilize small sets of labeled data combined with (possibly) large sets of unlabeled data. Through autonomous data collection, unlabeled data are more easily obtained at relatively lower costs. SSL techniques combine the power of unsupervised learning's ability to group data into "natural" clusters – where *natural* is defined by the similarity measure used by the clustering algorithm – with supervised learning's ability to assign class information to unlabeled data.

There are three general approaches to SSL: generative methods, low-density separation methods, and graph-based methods [2]. Generative methods assume that the data come from a fixed yet unknown distribution, $p(\boldsymbol{x}, y)$, where $\boldsymbol{x}$ and y are the instance features and class label, respectively; and that the decision boundaries can be represented based on class posterior probabilities [3],[4]. Low-density separation methods use information from unlabeled instances to modify a decision boundary created using only labeled data [5],[6]. Graph-based methods construct a graph $G = (V, E)$ with vertices, V, representing instances and edges, E, representing relationships between vertices [7],[8]. Class information is transferred from labeled instances to neighboring unlabeled instances based on the relationship defined by the edge connecting them.

While SSL for NSE is a much underexplored area, there are a growing number of *supervised* NSE algorithms that are heavily reliant on labeled data. Such algorithms generally fall into one of two categories: single-classifier update or ensemble learning. In either case, a test-then-train methodology is commonly used. At each time step new data are received and the current model provides class predictions for all instances. The correct labels are readily available and are used to calculate the model performance. The (single-classifier or ensemble based) model is then modified to incorporate information presented in the current data, which enables the model to adapt to a drift that may be present in the NSE. Single-classifier update approaches, such as [9],[10] modify the model by updating the previous classifier or creating a new classifier to take its place. Windowing methods are often used to decide if enough drift has occurred to warrant modification of the previous model [11],[12]. Ensemble learning uses a different approach of maintaining a bank of classifiers created over time. Each classifier analyzes

K. Dyer and R. Polikar are with the Signal Processing and Pattern Recognition Laboratory in the Electrical & Computer Engineering Department at Rowan University, Glassboro, NJ. This material is based upon work supported by the National Science Foundation under Grant No. ECCS-0926159.

Input: SSL algorithm **BaseClassifier**
α-shape probing radius - α; Offset distance - d
Labeled data
$\mathcal{L}^0 = \{\boldsymbol{x}_l^t \in X\}, \mathcal{Y}^0 = \{y_l^t \in Y = \{1, \ldots, C\}, l = 1, \ldots, M\}$
Do for $t = 0,1, \ldots$
1. Receive unlabeled data, $\mathcal{U}^t = \{x_u^t \in X, u = 1, \ldots, N\}$
2. Call **BaseClassifer** with $\mathcal{L}^t$, $\mathcal{Y}^t$, and $\mathcal{U}^t$
   Obtain $h^t: X \to Y$,
   Let $\mathfrak{D}^t = \{(\boldsymbol{x}_l^t, y_l^t): x \in \mathcal{L}^t \; \forall l\} \cup \{(\boldsymbol{x}_u^t, h_u^t): x \in \mathcal{U}^t \; \forall u\}$
3. Set $\mathcal{L}^{t+1} = \emptyset$, $\mathcal{Y}^{t+1} = \emptyset$
Do for each class $c = 1, \ldots, C$
4. Construct α-shape, $\mathcal{A}_c = f(\alpha, \mathfrak{D}_c^t)$
5. Compact α-shape using offset, $\mathcal{A}_c' = g(\mathcal{A}_c, d)$
6. Extract samples from compacted region
   Let $\mathcal{R}_c = \mathcal{A}_c' \cap \mathfrak{D}_c^t$
   $\mathcal{L}^{t+1} = \mathcal{L}^{t+1} \cup \mathcal{R}_c$
   $\mathcal{Y}^{t+1} = \mathcal{Y}^{t+1} \cup \{y_u: u \in [|\mathcal{R}_c|], y = c\}$
   If $\mathcal{R}_c = \emptyset$
      Generate synthetic data; append to $\mathcal{L}^{t+1}$ and $\mathcal{Y}^{t+1}$
   End if
End
End

Figure 1. COMPOSE Algorithm

the new data independently and assigns class information [13-15]. A final class label is decided by the ensemble of classifiers, combining information from each independent classifier.

When constraining the problem to an environment where labeled data are scarce and unlabeled data are plentiful, use of SSL algorithms usually improve classification. Several approaches [16-20], including one of our previous works [21], utilize SSL techniques that make use of both labeled and unlabeled data present in a data stream. These algorithms still make the assumption that labeled data are available at each time step, although the number of samples is limited. In real world applications, having labeled data available at every time step is rarely true. More likely, labeled data are received sporadically or only once at initialization, while unlabeled data may be abundantly available. To address random, sporadic receipt of labeled instances, Zhang et al. have then created an ensemble of clusters and classifiers. A classifier is created at those time steps when labeled data are present; at all other time steps clusters are formed [22]. A label propagation method then infers a class label for each cluster by weighting information provided in an ensemble of classifiers and clusters created at recent time steps. To the best of our knowledge there has been no effort addressing a NSE where few labeled instances are provided at the initialization and only unlabeled data are received at subsequent time steps. In this paper we refer to such an environment as the *initially labeled environment* (ILE). This paper introduces a new framework, called *COMpacted POlytope Sample Extraction* (COMPOSE), capable of learning in such an ILE. In this paper, we describe this framework and its basic properties, present the preliminary results of our experiments demonstrating that COMPOSE can learn in such non-stationary ILEs, using very limited amount of labeled data, and also that it is model (classifier) independent.

II. COMPOSE

A. An overview of the algorithm

To work in an ILE, COMPOSE extracts labeled data from the current time step to be combined with the next batch of unlabeled data received. At time $t = 0$, the labeled data are provided to COMPOSE as the initial input. At all other time steps, COMPOSE receives only unlabeled data, and subsequently labels a subset of previously provided unlabeled data, as described below. The distribution providing the unlabeled data $p^t(\boldsymbol{x})$ at time t may have drifted from the distribution $p^{t-1}(\boldsymbol{x})$ at time $t-1$. Consistent with other NSE algorithms, we assume limited drift such that the extracted labeled data overlap the newly received unlabeled data. Therefore, at this preliminary stage, limited drift is defined as follows: the distribution $p^t(\boldsymbol{x})$ must have considerable overlap with the distribution $p^{t-1}(\boldsymbol{x})$. For now, we deliberately leave the modifier *considerable* as vague, as we will be analyzing and refining this requirement in terms of distance metrics, such as Kullback-Leibler or Hellinger distance in future efforts. The remainder of this section explains in detail how COMPOSE i) creates α-shapes from the data; ii) compacts (shrinks) the α-shapes; and iii) extracts instances from the compacted α-shapes to serve as labeled data for future time steps.

The algorithm has three inputs: i) BaseClassifier, a SSL algorithm used to classify data at each time step, t; ii) the α-shape probing radius, α; and iii) the alpha shape offset distance, d. The algorithm is initialized with a set of labeled data, $(\mathcal{L}^0, \mathcal{Y}^0)$, at $t = 0$. At each subsequent time step, unlabeled data, $\mathcal{U}^t$, are received, and along with the available labeled data, passed to BaseClassifier (steps 1 and 2 within Fig. 1). BaseClassifier forms a hypothesis h^t by generating a decision boundary mapping $X \to Y$. A combined dataset, $\mathfrak{D}^t$, is constructed by merging $\mathcal{L}^t$ and $\mathcal{U}^t$, where class labels for $\mathcal{U}^t$ are generated by h^t (step 2). Since we are working under the assumption that only unlabeled data are received in future time steps, COMPOSE selects instances from the current time step to be labeled and used as training data at the next time step. These instances are selected from the center-core subspace of the current distribution to maximize the likelihood of overlap with the unlabeled data received from the drifted distribution at the next time step. The future training dataset, $(\mathcal{L}^{t+1}, \mathcal{Y}^{t+1})$, is initialized to be an empty set (step 3). Each class, $c = 1, \ldots, C$, identified by h^t, is analyzed independently, and instances from that class are appended to the labeled data $(\mathcal{L}^{t+1}, \mathcal{Y}^{t+1})$ (steps 4 – 6). These instances are then used as training data in the next time step when a new batch of unlabeled data is received. For each c^{th} class, an α-shape, $\mathcal{A}_c$, is constructed as explained in *Part B* of this section (denoted as function $f(\blacksquare)$ in step 4). $\mathcal{A}_c$ is *compacted* (i.e., shrunk) by the offset distance, d, to produce $\mathcal{A}_c'$ as described in *Part C* (denoted as the polytope compaction function $g(\blacksquare)$ in step 5). A polytope is a geometric object with flat sides that resides in the same dimensionality as X; α-shapes are composed of one or more polytopes. All instances with the current label of c, denoted $\mathfrak{D}_c^t$, that reside in the space spanned by $\mathcal{A}_c'$ are appended to the labeled data set. If no instances reside in the space spanned by $\mathcal{A}_c'$, synthetic data are created by random sampling from $\mathcal{A}_c'$ (step 6).

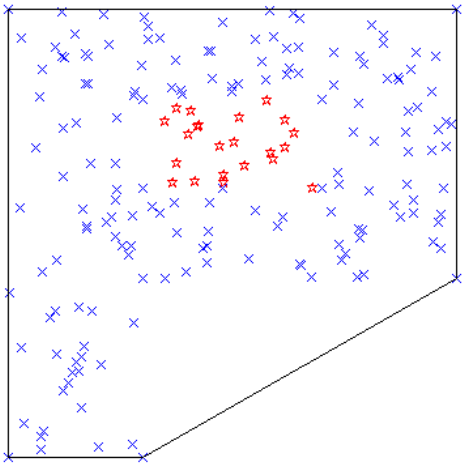

Figure 2. Using a convex hull to describe the "shape" of a set of x's incorporates empty regions of the feature space which may or may not belonging to another class (represented by stars in this figure).

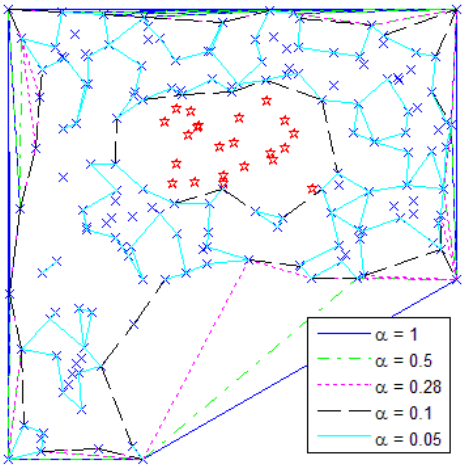

Figure 3. Using alpha shapes to describe the "shape" of a set of x's, empty regions can be realized. Different levels of detail are demonstrated in this figure by varying α. The same dataset is used as in Fig. 2

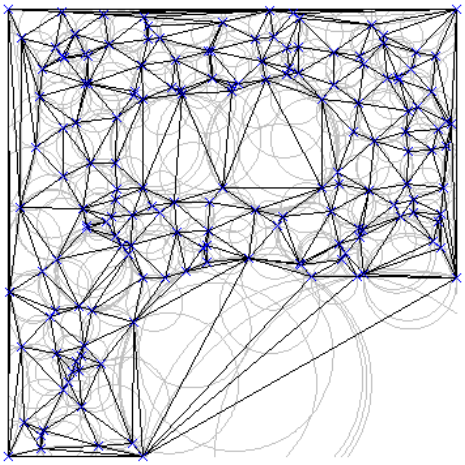

Figure 4. A Delaunay triangulation on the set of x's and the triangulation's circumcircles.

B. *α-Shape Function*

Convex hulls play an important role in machine learning to describe the region in which a set of points reside. By identifying the extreme points of the dataset, the general shape of the data may be inferred [23]. However, while convex hulls define the general shape, they fail to adequately describe low density or completely empty regions in the feature space. Including either of these regions in the convex hull may inaccurately represent the feature space, and in some cases may enclose a region defining another class, as illustrated in Fig. 2. α-shapes are a generalization of the convex hull [24], and address the aforementioned issues by creating polytopes that capture the shape of the set at a level of detail specified by α. For sufficiently large α, the α-shape is the convex hull of the set. As α decreases, the α-shape gradually develops voids and may result in disconnected regions, as shown in Fig. 3. In this preliminary version of the COMPOSE algorithm, the α value is selected heuristically; however, Teichmann et al. proposes a method of determining α dynamically based on the point density of the set [25], which is currently being investigated as an improvement to COMPOSE.

To understand the basic construction of an α-shape in two dimensions, consider a wooden board with several nails driven in halfway. The board represents the feature space and nails are the instances in the feature space. Take a solid flat disc of a fixed diameter, and place the edge in contact with a nail at an extreme point of the set. Roll the disc, circumscribing the nails in a clockwise (CW) or counter-clockwise (CCW) direction until it returns to the starting point. Using a string, connect the nails in the order the disc came in contact with them. Note that unless there is at least one pair of neighboring nails that are further apart than the diameter of the disc, the disc will not be able to slip between the nails (penetrate inside the dataset), resulting in some interior nails not being touched. Disconnected regions often occur in multimodal data or when the disc diameter is chosen suitably small. Repeat this process for nails not inside an existing polygon, creating external polygons (note that a polygon is a two dimensional polytope). Next, set the disc on top of the nails and slide it around looking for a void large enough for the disc to drop into. Again role the disc around in a CW or CCW direction connecting each nail with a string in the order it was touched. Repeat this process until all voids in the nails have been explored, creating internal polygons. The union of external polygons, minus the union of internal polygons constitutes the α-shape of this set of nails at an α value equal to the radius of the disc used to probe the set. Interior points of a large radius disc may become edge points of the α-shape for a smaller radius disc.

The mathematical construction of an α-shape requires triangulation of the data. Each edge of the triangulation is classified by comparing the circumcircles of the triangulation to a specified α value. Details for construction of an α-shape [24], utilizing Delaunay triangulation are outlined below.

1) Delaunay Triangulation: For a set of points **S**, the Delaunay triangulation forms a triangular mesh that connects every point in **S** such that no point in **S** lies inside the circumcircle of any triangle, as shown in Fig. 4. A circumcircle is the smallest circle that encloses all three points of a triangle. De-

launay triangulations may be extended into a D-dimensional space by modifying the criteria: no point in $\mathbf{S}$ lies inside the circum-hypershere of any D-simplex, where a D-simplex is the convex hull of its $D+1$ vertices (e.g., a 3-simplex is a tetrahedron). The computational complexity of this step is $\mathcal{O}(p^2)$, where p is the number of points in $\mathbf{S}$.

2) Classify Simplexes: After the triangulation has been constructed, each $(D-1)$-simplex is classified into one of three categories: interior, regular, or singular. The computational complexity of this step is $\mathcal{O}(s \log s)$, where s is the number of $(D-1)$-simplexes in the triangulation. In two dimensions, the $(D-1)$-simplex is an edge, the D-simplex is a triangle, and α defines the radius of a circle used in classification. All cases analyze the edge as a shared entity of two triangles using notation μ_1 and μ_2 to represent the radii of the smaller and larger circumcircle, respectively.

a) Interior Case: The interior case, $\alpha > \mu_2$, is the most common case and is shown in Fig. 5(a). Interior case simplexes do not contribute to defining the final α-shape. This case connects instances that reside inside the α-shape. As α decreases, the number of interior cases decreases as well.

b) Regular Case: The regular case, $\mu_1 < \alpha < \mu_2$, defines the edges of the α-shape, which occurs when μ_2 is a determinate value or when $\mu_2 = \infty$ (Fig. 5(b) and (c), respectively). Each simplex classified as *regular* contributes to define the final α-shape. When added to the final α-shape, these simplexes form one or more polytopes. As α decreases, the number of regular cases increases.

c) Singular Case: The singular case, $\alpha < \mu_1$, has two subcategories: unattached and attached. In the unattached sub-case, the length of the edge is greater than 2α, as shown in Fig. 5(d). The singular unattached case breaks connections between polytopes formed by the regular simplexes. This sub-case is required to form multiple disconnected polytopes in the α-shape. As α decreases, the number of unattached singular cases increases. In the attached sub-case, the length of the edge is less than 2α, this sub-case is shown in Fig. 5(e). The singular attached simplexes are added to the α-shape defined by the regular simplexes. A singular attached simplex may appear as a spoke that sticks out from a polytope or as a spoke connection between two polytopes. There is no relationship between α and the number of singular attached cases.

This classification technique easily extends into multi-dimensions [24]. For example, working in $3D$, the $(D-1)$ simplex is a triangle, the D-simplex is a tetrahedron, and α defines the radius of a sphere.

3) Finalize α-Shape: In [24] the final α-shape is the union of all simplexes classified as regular and singular attached. In the COMPOSE algorithm, the singular attached simplexes are ignored since they disappear in the polytope compaction function of the algorithm. The final α-shape, $\mathcal{A}$, consists of one or more polytopes defining the set "shape" with level of detail α.

C. Polytope Compaction Function

Shape offsets have been used extensively in image processing and computer aided drawing software to scale en-

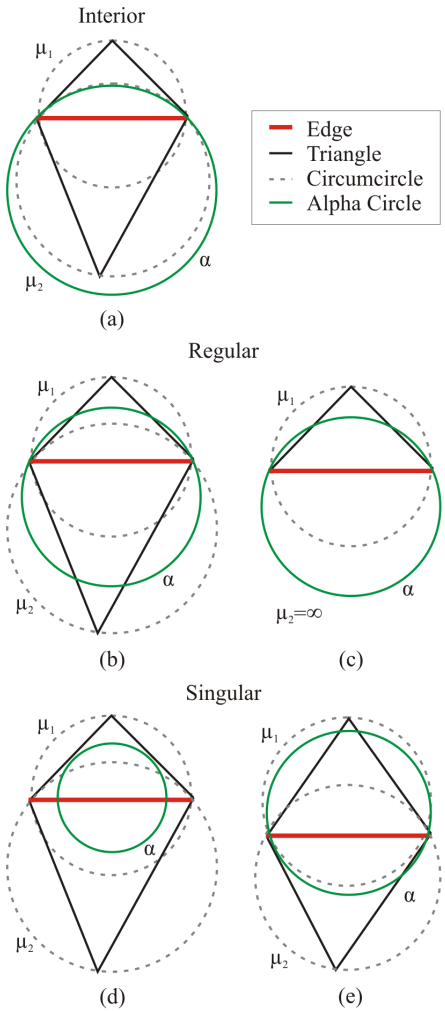

Figure 5. Examples of each type of simplex classification: (a) interior case (b) regular case - definite μ_2 (c) regular case – $\mu_2 = \infty$ (d) singular case - unattached (e) singular case - attached

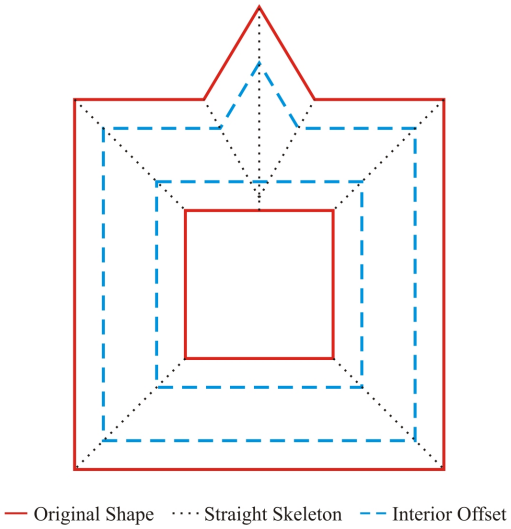

Figure 6. Original shape and its straight skeleton with sample of interior and exterior offsets. Note for a sufficiently large interior offsets, shape information may be lost.

closed regions. Offsets are accomplished in two dimensions by translating the vertices of a polygon along its straight skeleton as described in [26]. The straight skeleton of a polygon is the combination of all arcs that bisect any two edges. An example of a shape and its straight skeleton are shown in Fig 6. This method of constructing a straight skeleton in 2D has a computational complexity of $\mathcal{O}(v^2 \log v)$, where v is the number of vertices. Compacting polytopes in higher dimensions is possible by direct extension of the straight skeleton method, although computational complexity must be revaluated.

Shape offsets are used in COMPOSE to compact (i.e., shrink) each polytope in an α-shape. Currently the offset distance is determined heuristically; however, using methods analyzing point density are being considered for future improvements to COMPOSE. Proper selection of the interior offset distance is important. If the offset is too small, the sampling region will be large and samples selected at the current time step have a higher probability of overlapping the incorrect distributions at the next time step. If the offset is too large, the sampling region may disappear or become too small to encapsulate adequate number of instances from the current time step. Even when using an ideal offset distance, it is possible that no instances will lie within the compacted polytope. COMPOSE handles such cases by generating synthetic data by randomly sampling the compacted polytope.

III. Simulation Experiments and Results

A. Experimental Setup

We designed two experiments using non-stationary binary Gaussian based environments to demonstrate the general properties of the proposed framework. Each experiment was repeated using three different BaseClassifers, one from each of the SSL learning methods. A binary classification experiment was selected so that a low-density separation method, specifically a S^3VM, can be tested (an explanation of S^3VM's limitations is discussed in the next section). Since, to our knowledge, no other algorithms exist that operate in the ILE, Gaussian data were selected so performance of each BaseClassifier can be compared to the optimal Bayes classifier. In each experiment, we assumed Gaussian distributions started in an initial state at

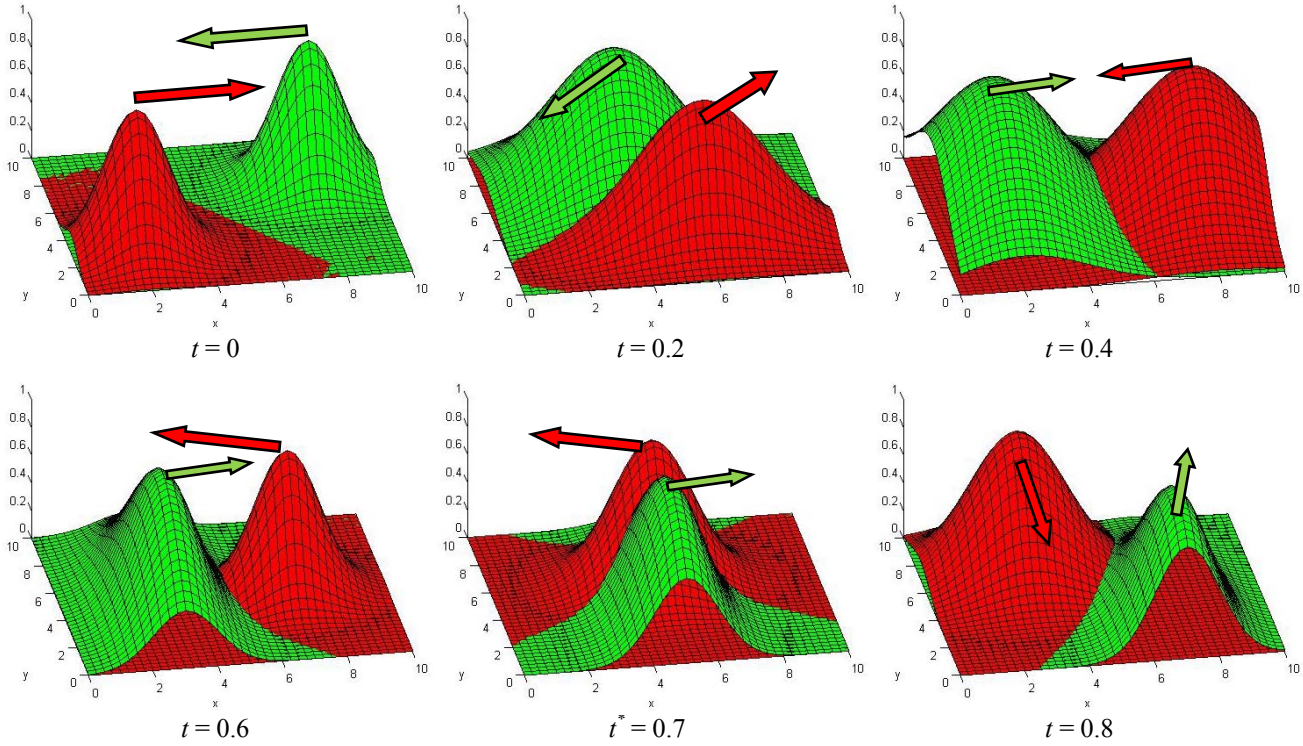

Figure 7. Snapshots of Gaussian environment at intervals where parametric equations governing drift change.
*Parametric equations do not change at $t = 0.7$ but it is included due to the difficulty of classification at this time step.

TABLE I. Parametric Equations Governing Drift of Unimodal Gaussian Experiment

Class	$0 \le t < 0.2$				$0.2 \le t < 0.4$				$0.4 \le t < 0.6$			
	μ_x	μ_y	σ_x	σ_y	μ_x	μ_y	σ_x	σ_y	μ_x	μ_y	σ_x	σ_y
C_1	2 + 20t	3	1 + 5t	1	6 + 10t	3 + 10t	2	1 + 5t	8 - 5t	5	2 - 5t	2 - 5t
C_2	8 - 20t	7	1 + 5t	1 + 5t	4 - 10t	7 - 10t	2	2 + 5t	2 + 5t	5 - 5t	2 - 5t	3

Class	$0.6 \le t < 0.8$				$0.8 \le t \le 1$			
	μ_x	μ_y	σ_x	σ_y	μ_x	μ_y	σ_x	σ_y
C_1	7 - 20t	5 + 10t	1 + 2.5t	1 + 2.5t	3	7 - 20t	1.5	1.5
C_2	3 + 20t	4 - 10t	1	3	7	2 + 25t	1 + 2.5t	3 - 7.5t

$t = 0$, as shown in Fig. 7 and 8. At each subsequent step t, the distributions drift according to parametric equations given in Table I and II, with 200 new *unlabeled* instances presented. The experiments end after 100 steps, at some arbitrary stop time, $t = 1$. Five percent of data presented at $t = 0$ are labeled and the prior probabilities of each class are set to 0.5. The class priors remain constant throughout the experiments.

B. SSL Algorithms

A brief overview of each SSL algorithm, along with the selection of its free parameters used in our experiments, is provided in this section.

a) Generative Method: Generative models assume $p(x, y) = p(x|y)p(y)$, where $p(x|y)$ is an identifiable mixture distribution [27]. In this experiment, we used a cluster-and-label classifier. Clusters were formed using k-means on labeled and unlabeled instances. The number of centers was determined from provided class priors. Clusters were labeled by a simple majority vote of labeled instances in each cluster.

b) Low-Density Separation Method: Low-density separation methods construct a decision boundary by combining the location of the labeled instances with the point density of the unlabeled instances. Using the cluster assumption, points in the same cluster are likely to be of the same class, it can be reasoned that a decision boundary lying in a low-density region would most likely separate dissimilar classes. The most common approach to low-density separation uses a modified support vector machine (SVM) to maximize the margin for both the labeled and unlabeled data [2]. This modified SVM was originally described in [6] as a transductive SVM, but is now more commonly referred to as a semi-supervised SVM (S^3VM). While S^3VMs work well on binary datasets, well studied multiclass methods for inductive SVMs, one-vs-all or one-vs-one, do not directly extend to S^3VMs [28],[29]. One of the more popular implementations of the S^3VM was developed in [5] and was used in our experiments. The S^3VM^{light} algorithm is noted for its robustness, speed, and ability to process large datasets. S^3VM^{light} solves the low-density separation

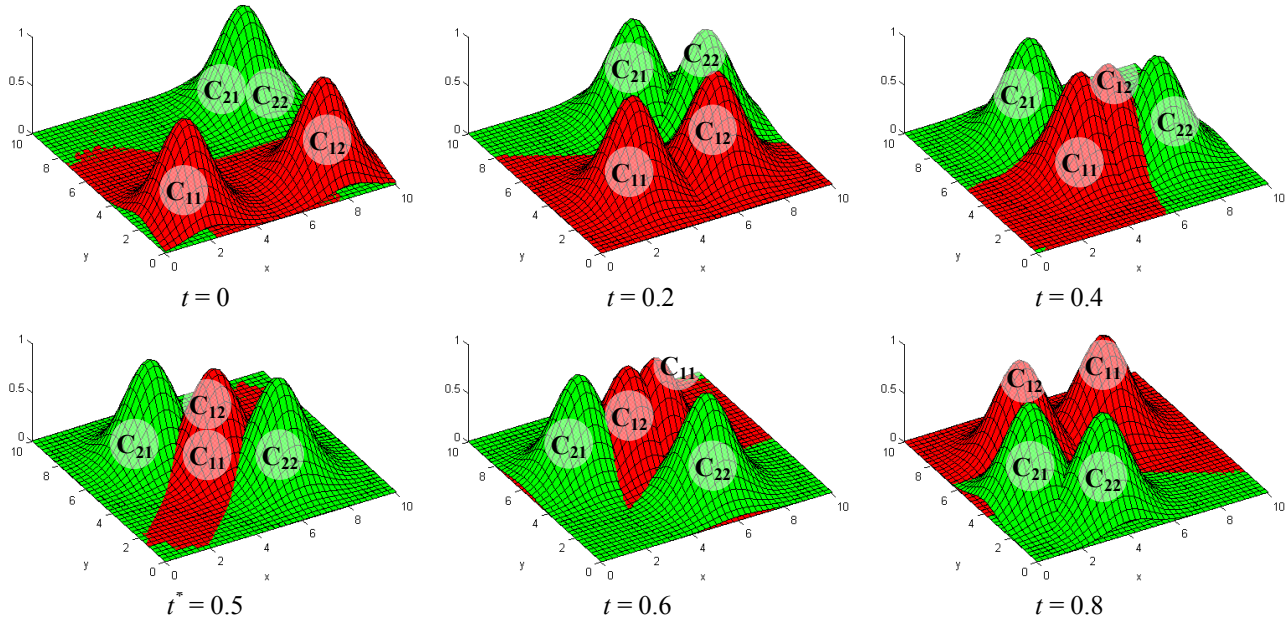

Figure 8. Snapshots of multimodal Gaussian environment at intervals where parametric equations governing drift change.
*Parametric equations do not change at $t = 0.5$ but it is included due to the difficulty of classification at this time step.

TABLE II. PARAMETRIC EQUATIONS GOVERNING DRIFT OF MULTIMODAL GAUSSIAN EXPERIMENT

Class	$0 \le t < 0.2$				$0.2 \le t < 0.4$				$0.4 \le t < 0.6$			
	μ_x	μ_y	σ_x	σ_y	μ_x	μ_y	σ_x	σ_y	μ_x	μ_y	σ_x	σ_y
C_{11}	2 +6t	2 + 6t	1	1	3.2 + 6t	3.2 + 6t	1	1	4.4 + 6t	4.4 + 6t	1	1
C_{12}	8 - 6t	2 + 6t	1	1	6.8 - 6t	3.2 + 6t	1	1	5.6 - 6t	4.4 + 6t	1	1
C_{21}	8 - 10t	8	1	1	6 - 10t	8 - 2.5t	1	1	4 - 7.5t	7.5-7.5t	1	1
C_{22}	8	8 - 10t	1	1	8 - 2.5t	6 - 10t	1	1	7.5-7.5t	4 - 7.5t	1	1

Class	$0.6 \le t < 0.8$				$0.8 \le t \le 1$			
	μ_x	μ_y	σ_x	σ_y	μ_x	μ_y	σ_x	σ_y
C_{11}	5.6 + 6t	5.6 + 6t	1	1	6.8 + 6t	6.8 + 6t	1	1
C_{12}	4.4 - 6t	5.6 + 6t	1	1	3.2 - 6t	6.8 + 6t	1	1
C_{21}	2.5-2.5t	6 - 10t	1	1	2	4 - 10t	1	1
C_{22}	6 - 10t	2.5-2.5t	1	1	4 - 10t	2	1	1

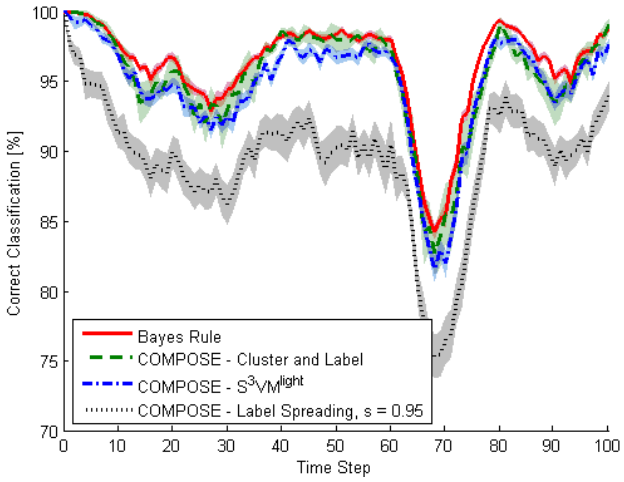

Figure 9. Performance of various SSL algorithms used in COMPOSE on the unimodal Gaussian dataset. Shading represents a 95% confidence interval.

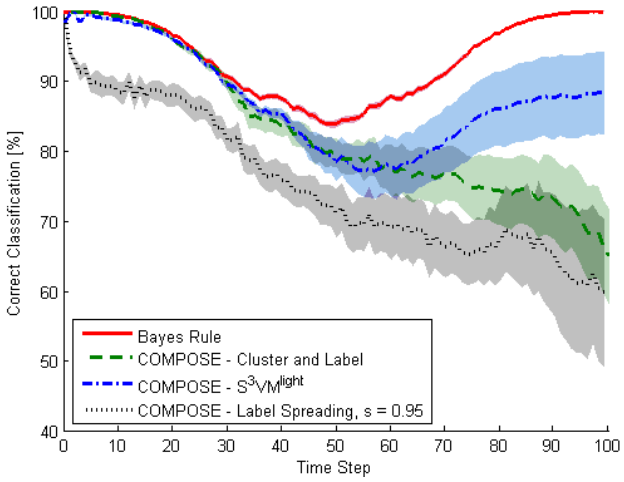

Figure 10. Performance of various SSL algorithms used in COMPOSE on the multimodal Gaussian dataset. Shading represents a 95% confidence interval.

problem by creating an initial decision boundary using an inductive SVM on the labeled data. All unlabeled instances are assigned a tentative class using the inductive SVM decision boundary. A common problem in SSL algorithms is for all unlabeled instances to be assigned to one class, resulting in a degenerate solution. To avoid degenerate solutions, S^3VM^{light} assigns unlabeled instances using a bidding process based on class prior probabilities, which can be estimated – if unknown – from the ratio of labeled instances belonging to each class. After every unlabeled instance has received a tentative label, a pair-wise search is conducted; each pair has an instance from the positive and negative class. If swapping the labels improves the cost function, it is done. In this experiment radial basis kernel was selected with $\sigma = 1$. Class balancing was done using the known class priors.

c) Graph-Based Method: Graph-based methods construct a graph of vertices and edges, where the vertices are the labeled and unlabeled instances and the edges – which may be weighted – define the similarity between instances [27]. A label spreading procedure then transfers class information from labeled vertices to neighboring points [7]. Spread of information is governed by the free parameter, $s \in (0,1)$. As $s \to 1$, a heavier weighting is placed on class information received from neighboring instances, and the importance of the point's initial class information decreases. Conversely, as $s \to 0$, a point's initial class information is regarded very highly and is influenced very little by information from its neighbors. For this experiment the spreading parameter, s, was set to 0.95 to encourage rapid spreading from the tightly clustered mass of labeled points to the extents of the unlabeled points. To avoid degenerate solutions, the label bidding process presented in [8] was implemented. In label bidding, the class priors are multiplied with the total number of unlabeled instances, which determines the total number of labels available for each class. Instances classified by the label spreading algorithm then bid on the available labels. Instances with highest confidence are the first ones to receive their labels.

C. Simulation Results & Discussion

Our primary observation is that COMPOSE very closely follows the performance trend of Bayes rule, regardless of the BaseClassifer, as shown in Fig. 9 and 10. All performance figures are an average of 50 independent trials; the shaded regions represent a 95% confidence interval of each performance trace.

SSL algorithms generally use large quantities of unlabeled data to improve classification. In our experiments, however, only 200 unlabeled instances were provided at each time step, a relatively small number of instances. Pairing the very limited labeled data with the unlabeled data that COMPOSE received and labeled, it was able to track the drifting distribution. We should add that the number of points labeled by COMPOSE and maintained between time steps grows during the first few time steps; however, does not continue to grow throughout the entire experiment. The number of points selected and labeled by COMPOSE depends on the point density of the distribution when the α value and offset parameter are held constant. As the distribution variance (spread) increases and the density drops, the number of points selected generally decrease.

1) Unimodal Gaussian Experiment

The unimodal Gaussian experiment demonstrates that COMPOSE can track both well separated as well as overlapping distributions as they drift in a NSE. Note that at time $t = 0.7$ (see Fig. 7) the two distributions pass through one another, causing all algorithms – including the Bayes classifier – drop their performance, yet COMPOSE is able to track the Bayes classifier quite closely despite the significant overlap.

Of the three BaseClassifiers used with COMPOSE, label spreading performed the poorest, which may be attributed to the placement of labeled instances. When labeled instances from a particular class span a larger area in feature space (albeit, possibly with less density), it is easier for that class to spread its label since spreading can proceed in more directions and overtake a larger area of unlabeled instances faster. In a NSE that provides labeled instances at every time step directly from the underlying distribution, the labeled data are more likely to be scattered throughout the unlabeled data. Using

COMPOSE, however, labeled data are located in a tighter cluster due to sampling from a compacted α-shape. This tight cluster of labeled data decreases the effectiveness of classification through label spreading. SSL algorithms that do not use label spreading, however, do not suffer from such a restriction.

2) Multimodal Gaussian Experiment

The results of the multimodal Gaussian experiment from time step 0 to 50 demonstrates COMPOSE's ability to track multiple modes within a distribution, even as a distribution's modes split or merge. The second half of the experiment exposes another weakness of label spreading (and to some extent cluster-and-label) as a BaseClassifier, however, the widening confidence interval and decrease in average performance after step 50 is due to high overlap in the distributions. The confidence interval widens after the distributions begin to disperse again. The current implementation of COMPOSE uses fixed α and offset parameters, leading to less than optimal results in time steps with high point densities. We note that once a mode is absorbed into the wrong class, it is difficult for it to be corrected at a later time step since no class information is obtained from the true distributions after the initial time step. We expect dynamic selection of α and offset parameters based on point density to improve classification in situations exemplified by the second half of the multimodal Gaussian experiment.

IV. CONCLUSIONS

We introduce a framework, whose preliminary analysis show promising results in utilizing SSL algorithms to classify non-stationary data in an environment, where very limited labeled data are available, and only during the initial time step – arguably one of the most challenging learning environments for a machine learning algorithm. COMPOSE consists of the following components: α-shape construction, α-shape compaction, and sampling of a compacted α-shape; each of which is scalable to multi-dimensions, although the computational complexity needs to be analyzed with respect to dimensionality.

In both experiments, we observed that COMPOSE, particularly when paired with S^3VM, performed remarkably well, and was able to track the changing environment. We also observe some weaknesses, described above, that will be addressed in future work. Improvements to the algorithm mentioned in this paper, such as dynamic selection of the α-shape function's α value and the polytope compaction function's offset value, are currently being explored. These preliminary experiments demonstrate the fundamental principles of COMPOSE, and demonstrate the algorithm is worth further exploration.

REFERENCES

[1] A.Kuh, T.Petsche, and R.L.Rivest, "Learning time-varying concepts," Proc.Conf.Advances Neural Inform.Process.Syst., pp. 183-189, 1990.

[2] O.Chapelle, B.Scholkopf, and A.Zien, Semi-Supervised Learning. Cambridge, MA: MIT Press, 2006.

[3] A.P.Dempster, N.M.Laird, and D.B.Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," J. Roy. Stat. Soc. . Series B (Methodological), vol. 39, no. 1, pp. 1-38, 1977.

[4] X.Zhu and A.Goldberg, Introduction to Semi-Supervised Learning Morgan & Claypool, 2009, pp. 31-32.

[5] T.Joachims, "Transductive Inference for Text Classification using Support Vector Machines," Proc. 16th Int. Conf. Machine Learning, pp. 200-209, 1999.

[6] V.Vapnik and A.Sterin, "On structural risk minimization for overall risk in a problem of pattern recognition," Automation and Remote Control, vol. 10, pp. 1495-1503, 1977.

[7] D.Zhou, O.Bousquet, T.Lal, J.Weston, and B.Scholkopf, "Learning with Local and Global Consistency," in Advances in Neural Information Processing Systems Cambridge, MA: MIT Press, 2004, 16, pp. 321-328.

[8] X.Zhu and Z.Ghahramani, "Learning from Labeled and Unlabeled Data with Label Propagation," Carnegie Mellon University, Pittsburgh, PA,CMU-CALD-02-107, 2002.

[9] D.Obradovic, "On-line training of recurrent neural networks with continuous topology adaptation," IEEE Trans.Neural Netw., vol. 7, pp. 222-228, 1996.

[10] G.Hulten, L.Spencer, and P.Domingos, "Mining Time-Changing Data Streams," Proc. 7th ACM SIGKDD Int. Conf. Knowledge Discovery Data Mining, pp. 97-106, 2001.

[11] G.Widmer and M.Kubat, "Learning in the presence of concept drift and hidden contexts," Mach. Learning, vol. 23, no. 1, pp. 69-101, 1996.

[12] A.Bifet and R.Gavalda, "Learning from Time-Changing Data with Adaptive Windowing," SIAM Int.Conf.Data Mining, pp. 443-448, 2007.

[13] R.Elwell and R.Polikar, "Incremental Learning of Concept Drift in Nonstationary Environments," IEEE Trans. Neural Netw., vol. 22, pp. 1517-1531, 2011.

[14] J.Kolter and M.Maloof, "Dynamic Weighted Majority: An Ensemble Method for Drifting Concepts," J. Mach. Learning Research, vol. 8, pp. 2755-2790, 2007.

[15] W.Street and Y.Kim, "A streaming ensemble algorithm (SEA) for large-scale classification," Proc. 7th ACM SIGKDD Int. Conf. Knowledge Discovery Data Mining, pp. 377-382, 2001.

[16] A.Goldberg, M.Li, and X.Zhu, "Online Manifold Regularization: A New Learning Setting and Empirical Study," Computer, vol. 5211, no. 1, pp. 393-407, 2008.

[17] A.Goldberg, X.Zhu, A.Furger, and J.Xu, "OASIS: Online Active SemI-Supervised Learning," 25th Conf. Artificial Intell., 2011.

[18] M.Masud, J.Gao, L.Khan, and J.Han, "A Practical Approach to Classify Evolving Data Streams: Training with Limited Amount of Labeled Data," IEEE 8th Int.Conf.Data Mining, pp. 929-934, 2008.

[19] P.Li, X.Wu, and X.Hu, "Mining Recurring Concept Drifts with Limited Labeled Streaming Data," J.Mach.Learning Research Proc.2nd Asian Conf.Mach.Learning, pp. 241-252, 2010.

[20] P.Zhang, X.Zhu, and L.Guo, "Mining Data Streams with Labeled and Unlabeled Training Examples," IEEE 9th Int. Conf. Data Mining, pp. 627-636, 2009.

[21] G.Ditzler and R.Polikar, "Semi-supervised Learning in Nonstationary Environments," Proc. Int. Joint Conf. Neural Networks, pp. 2741-2748, 2011.

[22] P.Zhang, X.Zhu, J.Tan, and L.Guo, "Classifier and Cluster Ensembles for Mining Concept Drifting Data Streams," IEEE 10th Int. Conf. Data Mining, pp. 1175-1180, 2010.

[23] S.G.Akl and G.Toussaint, "Efficient convex hull algorithms for pattern recognition applications," Proc. 4th Int. Joint Conf. Pattern Recognition, pp. 483-487, 1978.

[24] H.Edelsbrunner and E.Mucke, "Three-Dimensional Alpha Shapes," Assoc.Computing Machinery Trans.Graph., vol. 13, pp. 43-72, 1994.

[25] M.Teichmann and M.Capps, "Surface Reconstruction with Anisotropic Density-Scaled Alpha Shapes," IEEE Visualization, pp. 67-72, 1998.

[26] O.Aichholzer, D.Alberts, F.Aurenhammer, and B.Gaertner, "Straight skeletons of simple polygons," Proc. 4th Int. Symp. LIESMARS, pp. 114-124, 1995.

[27] X.Zhu, "Semi-Supervised Learning Literature Survey," University of Wisconsin, Madison, WI,Computer Sciences TR 1530, 2008.

[28] L.Xu and D.Schuurmans, "Unsupervised and Semi-supervised Multi-class Support Vector Machines," Proc. 20th Nat. Conf. Artificial Intell., vol. 2, pp. 904-910, 2005.

[29] A.Gonopolskiy, B.Nash, B.Avery, and J.Thomas, "Experimenting with Multi-Class Semi-Supervised Support Vector Machines and High-Dimensional Datasets," unpublished.