Detecting and Tracking Concept Class Drift and Emergence in Non-Stationary Fast Data Streams

Brandon S. Parker and Latifur Khan
University of Texas at Dallas
2601 North Floyd Road
Richardson, TX 75083-0688 U.S.A.
{brandon.parker, lkhan}@utdallas.edu

Abstract

As the proliferation of constant data feeds increases from social media, embedded sensors, and other sources, the capability to provide predictive concept labels to these data streams will become ever more important and lucrative. However, the dynamic, non-stationary nature, and effectively infinite length of data streams pose additional challenges for stream data mining algorithms. The sparse quantity of training data also limits the use of algorithms that are heavily dependent on supervised training. To address all these issues, we propose an incremental semi-supervised method that provides general concept class label predictions, but it also tracks concept clusters within the feature space using an innovative new online clustering algorithm. Each concept cluster contains an embedded stream classifier, creating a diverse ensemble for data instance classification within the generative model used for detecting emerging concepts in the stream. Unlike other recent novel class detection methods, our method goes beyond detecting, and continues to differentiate and track the emerging concepts. We show the effectiveness of our method on several synthetic and real world data sets, and we compare the results against other leading baseline methods.

Introduction

While the data mining disciplines of classification and clustering are well studied for static data, only within the last two decades has data stream mining both been of academic interest and need due to the recent explosion of data stream availability through social media feeds, *cloud* services, and *Big Data* concepts. Not only has the volume of data grown tremendously, but the very nature of data has been revolutionized with ideas of soft ontologies and schemaless data streams and stores. These alterations in our fundamental view of data pose a challenge to the tradition view of data mining and the existing traditional data mining algorithms. One of the well known problems in stream mining is *concept drift*. However, a lesser explored issue is the emergence of entirely new concept classes within the stream, called *novel classes*. Figure 1 depicts some fundamental drift issues present in data streams. The primary contributions to change within a non-stationary stream can be categorized as concept drift, feature evolution, and concept evolution.

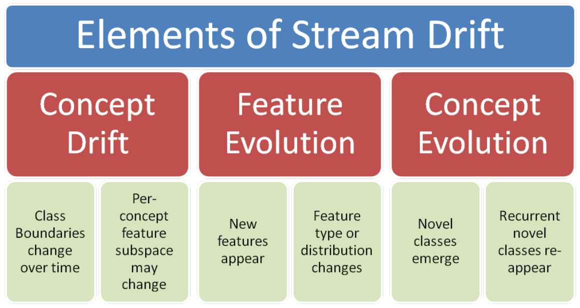
Elements of Stream Drift
Concept Drift
Feature Evolution
Concept Evolution
Class Boundaries change over time
Per-concept feature subspace may change
New features appear
Feature type or distribution changes
Novel classes emerge
Recurrent novel classes re-appear

Figure 1: Three main challenges of dynamic data stream drift with two examples of each

Since the approach developed in this research effort encompasses both stream clustering and ensemble classifier creation, it has several similarities to various sub-areas of stream mining research. None of the other methods, however, adequately address all the stream mining issues as our approach, an algorithm named SluiceBox.

Among the key contributions of this research, the algorithmic method described herein provides:

- a new approach to novel class detection that extends beyond mere detection into also subspace locating and concept tracking,
- a new emergent class detection and tracking method that is easily combined with advancing classification methods, as demonstrated using Leveraging Bagging (Bifet, Holmes, and Pfahringer 2010), and
- a new online clustering methodology that combines dynamic cluster quantity with subspace optimization.

In the next section, we explore some important prior work relevant to our approach and problem domain. The third section then describes our classifier and clustering design in greater detail followed by a section providing experimental results demonstrating the comparative capabilities of three variations of the SluiceBox algorithm compared with two baseline methods. The paper then concludes with a section discussing future work.

Background and Related Work

The closest works to the SluiceBox method are ECSMiner (Masud et al. 2011) and its feature evolution improvement DXMiner (Masud et al. 2010). In both, the authors address all three core drift issues, but they use simple ensemble voting, handle only pre-normalized continuous-valued features, and retain a homogenized conglomeration of all features in all ensemble model evaluations. (Masud et al. 2011) outlines an approach to use an ensemble of hyper-spheres to capture the decision boundary for classes as the stream is processed. These methods only use continuous-valued features, and normalize these features so that they all range from (0.0,1.0) in the data set as a pre-processing task.

ECSMiner takes a chunk-based approach to stream mining, wherein it caches a segment of the data stream (typically between 500 and 5000 data points), and then predicts the labels and novelty of the data points within that data chunk, and subsequently uses the data chunk to update the classification ensemble model. Prediction of a label for a given test data instance is provided via ensemble voting from the constructed multi-chunk ensemble. Note that the algorithm simply identifies novel instances on a per instance basis, without regard for locating or tracking the novel concepts in the feature space. Within each ensemble member classifier, the decision to identify a data point as a novel concept instance uses the $qNSC$ identified in (Masud et al. 2011) and (Masud et al. 2010), which measures a combined value of both the separation of the new data point from existing concept classes and the cohesion or local density of nearby outliers.

Our approach differs significantly from ECSMiner and DXMiner in that it uses all potential attributes (numeric and nominal) without pre-normalization, is fully online instead of chunk-based, offers full concept tracking including subspaces and emerging novel clusters, and can be easily paired with other stream classifiers for semi-supervised learning.

Grimson and Stauffer (Stauffer and Grimson 1999) offers an interesting insight into stream mining and novel class detection as well. The focus of (Stauffer and Grimson 1999) is object extraction from motion imagery data, so the overall domain and solution is significantly different than the domain of instance classification and novel class detection in generic data streams, but initial concept of separating background data from foreground data using a Gaussian process can be loosely applied for finding outliers in a data stream. Furthermore, their approach for creating and growing new background models dynamically as the stream progresses is conceptually aligned with our need to create new multi-dimensional clusters and use them to collect data observations until the new group can be considered a sufficiently large set to declare as a novel concept, or until labeled data is provided. In a similar domain to (Stauffer and Grimson 1999), Zweig et al. in (Zweig, Eshar, and Weinshall 2012) use a tiered discriminative classifier in order to discern novel class objects in imagery.

Design

The SluiceBox algorithm uses all incoming data points in an unsupervised fashion to update a collection of sub-

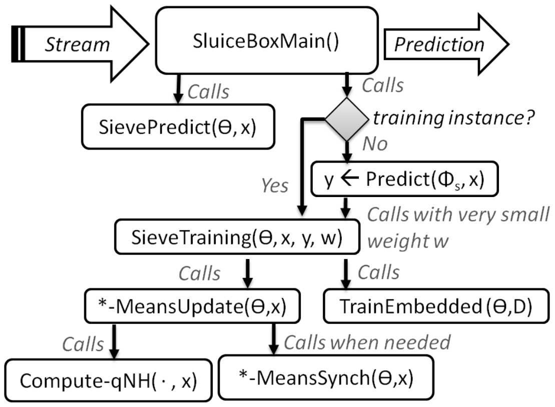

Figure 2: High level flow and call graph for the SluiceBox method with classifier models ϕ, cluster models θ, attribute vector x, label y and weights w

space aware clusters. Unlike traditional K-Means clustering (Hastie, Tibshirani, and Friedman 2009), this clustering method has the ability to vary the number of clusters as necessary, similar to the capability of x-Means (Pelleg and Moore 2000), and can also vary the subspace covered by each cluster, similar to the capabilities of methods described in (Cheng, Fu, and Zhang 1999). For any true supervised training (i.e. labeled) data, the classifier model (notated ϕ) embedded within each cluster model (notated θ) is trained. For unlabeled (i.e. 'test') data, an imputed label is provided at a drastically reduced training weight in order to provide a minimal *best guess* among the classifiers until true training data arrives. The overall data flow and functional call graph is depicted in Figure 2. The two principle functions are described in pseudo code below.

This formulation of clusters with embedded classifiers can be viewed as an ensemble of learners, diversified by logical separation in the feature space. Therefore, label prediction follows a typical weighted majority voting method, using the distance from the query point to each cluster as the weight applied to each cluster-embedded classifier's prediction.

Algorithm 1 lists the main procedure for the SluiceBox method. The set of clusters persist as Θ, and the number of clusters are bounded by input parameters that provide a minimum (K_{min}) and maximum (K_{max}) limit. The algorithm assumes that an initial warm-up phase provides a relatively small set of contiguous training data to boot strap the model. While the length of this warm-up phase is entirely configurable, the value of 5,000 data points was used as it tended to provided all algorithms under test with the best start up window. Within the warm-up phase, all data instance are used as supervised training data, but after the warm-up phase, the occurrence of training data is modeled by the test harness as a uniform distribution with probability of ground truth being $P(\epsilon)$. During the warm-up phase, no metrics were counted against *any* of the algorithms in the experiments reflected later in this paper, since multiple algorithms exhibited the same trait of very low accuracy in the initial segment of the data stream. Lines 2-4 of Algorithm

Algorithm 1 SluiceBox High Level Process

```
Require: Assistant Classifier $\Phi_s$
Require: Sieve Cluster model $\Theta$
Require: Data Stream $D$
Require: Training Fraction $\epsilon$
 1: for all $x \in D$ do
 2:     if in warmpup phase then                          ▷ First 5000 data points
 3:         TRAIN($\Theta$, x, $y(x)$, $\omega$ )
 4:         TRAIN($\Phi_s$, x, $y(x)$, $\omega$)
 5:     else
 6:         if $x \in D_L$ then                           ▷ Labels appear with $P(\epsilon)$
 7:             SIEVETRAIN($\Theta$, x, $y(x)$, $\omega$)              ▷ Algo. 2
 8:             TRAIN($\Phi_s$, x, $y(x)$, $\omega$)
 9:         else
10:             $h_\Phi \leftarrow$ PREDICT($\Phi_s$,x)
11:             $h_\Theta \leftarrow$ SIEVEPREDICT($\Theta$,x)
12:             if $h_\Theta$ = Novel then
13:                 $h(x) \leftarrow Novel$
14:             else if $h_\Theta$ == Outlier then
15:                 PUSH($\Xi$,x)
16:             else
17:                 $h(x) \leftarrow (h_\Phi \cdot w_\Phi) + (h_\Theta \cdot w_\Theta)$
18:             end if
19:             SIEVETRAIN($\Theta$, x, $h_\Phi$, $\omega_{SSL}$)              ▷ Algo. 2
20:         end if
21:     end if
22:     for all $x \in \Xi$ past their deadline do
23:         repeat lines 10-19, except 14-16                  ▷ finalize
24:     end for
25:     TRAIN($\Theta_u$, x, $h_\Phi$, $\omega \leftarrow 1$)
26:     Report $h(x)$
27: end for
```

1 represent this warm-up phase, where the initial models Θ and Φ_s are trained.

Lines 6-8 of Algorithm 1 cover the case when training (i.e., labeled) data does enter the system. Training data is often sparse and latent compared to the unlabeled data instance within the stream. When training data arrives, it is used to train both the cluster model Θ (line 7) and the co-training classifier model Φ_s (line 8). The co-training classifier is an optional capability, and the experimental results shown later depict variations of SluiceBox with and without using Φ_s.

Lines 10-19 of Algorithm 1 are the main thrust of the process, wherein data points enter the system, obtain a predicted label, and are immediately used for unsupervised training. Specifically, we obtain two hypotheses in lines 10-11 from the ensemble of classifiers embedded in the cluster model Θ and the co-training mode Φ_s respectively. If the cluster model Θ predicts (lines 12-13) that the data point is a member of a novel class (that is, lands in a cluster with no trained labels), then the prediction for the query point x is finalized as a novel class and given a serial identifier pertaining to the novel concept cluster to which it most likely belongs. Note that, unlike ECSMiner and related methods, the declaration of novelty is cluster-based, which provides additional feature-space location and tracking of the novel concepts within the model. If the query point lands in a labeled concept cluster within the cluster model Θ, then the combined weighted vote of Θ and Φ_s are used as the final

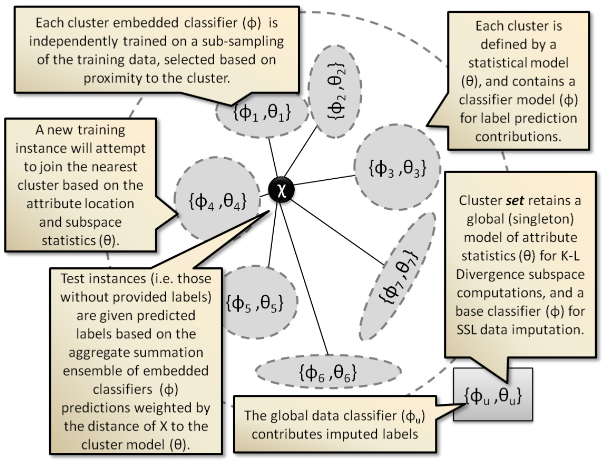
Each cluster embedded classifier (ϕ) is independently trained on a sub-sampling of the training data, selected based on proximity to the cluster.
Each cluster is defined by a statistical model (θ), and contains a classifier model (ϕ) for label prediction contributions.
A new training instance will attempt to join the nearest cluster based on the attribute location and subspace statistics (θ).
{ϕ1 ,θ1}
{ϕ2 ,θ2}
{ϕ3 ,θ3}
{ϕ4 ,θ4}
X
{ϕ5 ,θ5}
{ϕ6 ,θ6}
{ϕ7 ,θ7}
Cluster set retains a global (singleton) model of attribute statistics (θ) for K-L Divergence subspace computations, and a base classifier (ϕ) for SSL data imputation.
Test instances (i.e. those without provided labels) are given predicted labels based on the aggregate summation ensemble of embedded classifiers (ϕ) predictions weighted by the distance of X to the cluster model (θ).
{ϕu ,θu}
The global data classifier (ϕu) contributes imputed labels

Figure 3: SluiceBox Modeling Structure

prediction label for x (line 17). The weights w_Θ and w_Φ are configurable such that a balance can be achieved between them as needed. As demonstrated in the experimental results shown later, SluiceBox is extremely adept at detecting and tracking emerging concept classes, but performs only moderately well in overall prediction accuracy of known labels. The inclusion and usage of the Φ_s model is specifically to mitigate this and enhance the general prediction accuracy of the method. From another viewpoint, this architecture allows SluiceBox to be used in conjunction with any other strong classifier such that the novel class discovery can be relegated to SluiceBox, effectively providing a fast snap-on enhancement to existing classification methods that would be represented by Φ_s.

Even if Φ_s is not used in the final prediction (by setting $w_\Phi \leftarrow 0$), the predicted label for x is used with a very small weight ω_{SSL} (e.g. 0.001) as shown in line 19 in order to provide the cluster-embedded ensemble a reasonable hypothesis basis in the presence of sparse training information. Note that when an actual training instance enters the system, the training weight of 1.0 (line 7) would quickly overpower and correct the semi-supervised training weights provided by Φ_s to the cluster-embedded ensemble members.

The final case of the unsupervised training block is depicted in lines 14-15, where the query data point is found not to be a likely member of any current cluster. In this case, the query instance x is placed on to a very short delay queue of size δ (where typically $5 \leq \delta \leq 50$). More technically, the data point x is placed on the queue with a deadline of δ future instance counts. In other words, the hypothesis for x must be finalized after exactly δ more data points have entered the system, regardless of other conditions or events.

Clustering

Algorithm 2 illustrates how the clustering model is incrementally updated as new data instances arrive. The overall classifier and cluster model is illustrated in Figure 3. Lines 2-5 take care of maintaining the limited data queue/cache (typically set to 2,000 instances). Line 6 then retrieves the

Algorithm 2 SluiceBox AnyMeans Live Update Function

```
Require: Instance Cache $D^\tau$
Require: Max Cluster Limit $K_{max}$
Require: Learning Rate $\lambda$
Require: Minimum Cluster Size $q$
 1: function SIEVETRAIN(ClusterModel $\Theta$, Instance $x$)
 2:     PUSH($D^\tau$,$x$)
 3:     if $size(D^\tau) > \tau$ then
 4:         REMOVELAST($D^\tau$)
 5:     end if
 6:     $c_{min} \leftarrow$ FINDNEARESTCLUSTER($\Theta$, $x$)
 7:     for all $c \in \Theta$ do
 8:         $w_c \leftarrow w_c \times \lambda$
 9:         if $c == c_{min}$ then
10:             $w_c \leftarrow w_c + (1 - \lambda)$
11:         end if
12:     end for
13:     $dist \leftarrow$ DISTANCE($c_{min}$,$x$)
14:     if $dist <$ GETRADIUS($c_{min}$) then
15:         ASSOCIATE($c_{min}$, $x$)
16:         RECOMPUTESTATISTICS($c_{min}$)
17:     else              ▷ Check for novel concept cluster emergence
18:         $qNH_{min} \leftarrow$ COMPUTEQNH($c_{min}$, $x$)
19:         $qNH_{out} \leftarrow$ COMPUTEQNH($\Xi$, $x$)
20:         $qNSC \leftarrow \frac{qNH_{min} - qNH_{out}}{Max(qNH_{min}, qNH_{out})}$
21:         if $qNSC > 0$ then
22:             $c_{new} \leftarrow$ CREATENEWCLUSTER($x$)
23:             if $|\Theta| > K_{max}$ then
24:                 REMOVESMALLESTWEIGHT($\Theta$)
25:             end if
26:             $\Theta \leftarrow \Theta \cup c_{new}$
27:         end if
28:     end if
29:     if resynch timer expired then
30:         ANYMEANSSYNCH($\Theta$, $D^\tau$)
31:         TRAINEMBEDDEDLEARNERS($\Phi$, $D^\tau$)
32:     end if
33: end function
```

cluster nearest to the new data instance x from the cluster model Θ.

Lines 7-13 of Algorithm 2 provides a reinforcement learning based relevance management. Here, every cluster's relevance is reduced by multiplying the current relevance value w_c by the decay (or learning) factor λ in line 8. The one cluster most likely to contain the new data instance, however, is then given a boost to its weight by adding $(1 - \lambda)$ in line 10. Line 12 then stores the new weight. These weights should not be confused with the weights used for prediction discussed earlier, as the prediction weighting is based solely on the distance from the query point to the cluster. These cluster weights are used internally to determine which cluster is the *weakest* and thus will be sacrificed in order to make room for a new emerging cluster under the appropriate conditions, which is explained momentarily.

Line 13 captures the distance between the nearest cluster and the new data instance x. If the data instance is a likely member of the nearest cluster, then it is added to the cluster, and the cluster statistics are recomputed (lines 15-16). Otherwise, the new data point is considered an outlier, and a subprocess determines whether there are sufficient conditions to

	Classes		Features		# of
DataSet	Total	Active	Num	Nom	Records
IRND5M	100	17	10	0	5000000
KDD99	23	8	34	8	4898431
PAMAP2	13	3	53	1	2872533
FC	7	3	54	0	500000

Table 1: Data Sets Characteristics

create a new cluster representing an emerging concept cluster. Note that a *new* cluster is not necessarily a *novel* concept cluster, in that rapid concept drift or other feature evolution changes could cause an existing concept to present another centroid or dense region in the feature space. The new cluster is only considered a novel cluster if it does not end up with any labeled training data assigned to it.

Lines 18-26 handle the potential emergence of a new cluster, and its addition to the cluster model Θ. Lines 18 and 19 find the qNH neighborhoods exactly the same as discussed in (Masud et al. 2011), but used here to determine when to create a new cluster. If, as depicted in line 21, the $qNSC$ value is positive (i.e. greater than zero), then a sufficient number of outlier data points have a sufficient density as compared to the data points in the nearest cluster, and the outlier points are sufficiently separated from the known clusters (as established by the *else* condition in line 17 from line 14). In this case, a new cluster is created containing the new data instance and the outlier points that fell within the q-Neighborhood (qNH_{out}) that was discovered.

If the maximum number of clusters already exists, as checked in line 23, then the weakest cluster currently within the cluster model set Θ is removed in line 24, after which the new cluster can be safely added in line 26. Note that the selection of the cluster with the smallest weight utilizes the weighting of the clusters that is established in lines 8-12, such that the removed cluster is likely an older cluster which has not been updated in some time.

Finally, lines 29-32 handle the case where periodic resynchronization is configured, and the resynchronization frequency timeout has occurred. The *TrainEmbeddedLearners* procedure follows the online training function native to the embedded classifier (e.g., Naive Bayes).

Once at the conclusion of the warm-up phase (first 5,000 training points), and then subsequently on a periodic (if configured) basis, the *-Means ("*any*-Means") clustering algorithm batch-oriented process is invoked. While the incremental processing function described above is sufficient to *maintain* the clustering under moderate streaming data conditions, the resynchronization process is used to initially establish (and periodically re-establish) the clustering model more aggressively and following an iterative expectation-maximization (E-M) methodology (Hastie, Tibshirani, and Friedman 2009), (Mitchell 1997), (Russell and Norvig 2003). Compared to the typical depiction of the E-M algorithm, this process is more complex as it contains three separate maximization sub-steps, separated by incremental expectation estimation steps.

DataSet	ECSMiner	LB	SB	SBLB	SBStrm
FC@0.001	48.09	**60.19**	29.29	26.83	21.92
FC@0.05	38.03	**89.39**	67.33	73.99	49.89
FC@0.1	47.52	**91.23**	71.30	79.93	54.14
FC@0.25	56.52	**92.36**	75.73	83.82	59.14
FC@1.0	65.96	**93.34**	77.35	87.20	63.67
IRND5M@0.001	10.86	47.59	78.62	**83.84**	80.99
IRND5M@0.05	30.18	**84.36**	42.49	66.55	34.91
IRND5M@0.1	41.51	**85.91**	44.15	78.22	38.43
IRND5M@0.25	56.87	**87.50**	44.93	84.25	40.44
IRND5M@1.0	68.44	**89.17**	45.35	87.95	42.08
KDD99@0.001	19.91	**97.41**	51.51	95.41	42.68
KDD99@0.05	67.83	**99.18**	94.25	97.99	78.38
KDD99@0.1	75.86	**99.31**	95.70	98.28	80.37
KDD99@0.25	87.04	**98.55**	96.32	97.61	82.69
KDD99@1.0	94.28	**98.41**	96.66	97.69	85.03
PAMAP2@0.001	27.55	**63.97**	38.95	61.30	30.88
PAMAP2@0.05	7.01	**74.96**	40.19	62.04	47.27
PAMAP2@0.1	3.95	**77.31**	44.35	64.97	39.87
PAMAP2@0.25	10.37	**80.28**	49.32	69.14	30.67
PAMAP2@1.0	52.27	**82.02**	54.09	73.01	19.27
Rank Avg	4.40	**1.15**	3.15	2.05	4.25

Table 2: Experimental Classifier Accuracy Results showing algorithm comparison for different data sets (Higher values are better).

Empirical Results

In order to test the methods described herein, we leveraged the MOA framework (Kranen et al. 2012) and tested against the implemented ECSMiner (ECSM) (Masud et al. 2011) and LeveragingBagging (LB) (Bifet, Holmes, and Pfahringer 2010) algorithms. We then tested three variations of our SluiceBox method. The first, notated as *SB*, is the SluiceBox method with a resynchronization occurring every 20,000 data points and the semi-supervised advisor algorithm contributing with a training weight of only 1/1000. The second variation, noted as *SBLB* uses the Leveraging Bagging (LB) algorithm as the semi-supervised advisor algorithm with a training weight contribution of 1/1000 and a final prediction vote weight equal to the cluster ensemble vote. Finally, the variation notated as *SBStrm* is like the *SB* variation, but *without* any resynchronization (i.e. pure online method). For all variations of our method, a data cache size of 2,000 data points, a retry queue size of 5, and a maximum cluster limit of 75 were used. Due to the variation in the scores down the table, the average score is only moderately useful, so an additional summary metric, *Average Row Rank*, is also provided, indicating the average ranking of each algorithm for each data set row, where a rank of 1 indicated the algorithm is the *best* for the row.

We utilized three baseline real world data sets: PAMAP2 (Anguita et al. 2012), Forest Cover (FC)(Bache and Lichman 2013), and KDD Cup '99, from the UCI repository (Bache and Lichman 2013). We also used two synthetic data sets using a newer *Induced Random Non-Stationary Data* generator plug-in created for MOA. The characteristics of these data sets are depicted in Table 1, with number of class labels that occur in each data set (as Total) and the maximum number of concurrent labels (as Active) in a 5000-point sample window. When the warmup phase contains the maximum number of concurrent active labels, the PAMAP2 has 10 novel classes, but IRND5M has 83 novel classes, providing a better evaluation set for the algorithms. In the reported results, we vary the quantity of training data (i.e. training fraction) for each data set, notated as a suffix to the data set abbreviation indicating the decimal fraction of labeled data used. The SluiceBox and ECSMiner methods are generative methods - a necessary condition for novel class detection. The Leveraging Bagging method, however, is a discriminative model using decision trees. As noted in (Lasserre and Bishop 2007), discriminative models tend to find larger decision margins and perform better with adequate date, whereas generative models will have the potential for higher accuracy when labeled data is not as prevalent. When true training data is sparse (as indicated by the decimal value following the '@' sign on each data set), the number of cohesive unlabeled concept clusters increases proportional to the labeled concept clusters, providing the novel class detection portion of the method a greater opportunity to contribute to overall accuracy. This is also the cause of the accuracy increase for the IRND5M data set with @0.001 in our method —the larger abundance of new labels in the stream and our method can detect all the novel class instances (i.e., low FNR for @0.001 in Table 3). The drop in accuracy with increasing ground truth prevalence is due to the fact that generative models are less effective with existing class instance classification and fewer number of novel class instances will be present and detected in the stream.

We report two categories of results below. First is the overall accuracy of the classifiers with regard to predicting the correct label (or correctly predicting novelty). The results are shown in Table 2. Note that the Leveraging Bagging (LB) (Bifet, Holmes, and Pfahringer 2010) method is quite accurate overall. However, in the presence of more novel class emergence and low training data quantity, as seen in the first IRND5M line, the SluiceBox method boost the accuracy of leverage bagging (see the SBLB column) where alone Leverage Bagging experiences high error. Compared to the capability comparable ECSMiner, all of the SluiceBox variations achieve higher accuracy ratings and a better overall average ranking. Observe that overall the SBLB column demonstrates a positive symbiotic relationship effectively using the strengths of the LB and SB methods as appropriate to the data. The second result set, Table 3 focuses strictly on the False Positive Rate (FPR) and False Negative Rate (FNR) of novel class detection, not label accuracy. Obviously the Leveraging Bagging algorithm will never predict a novel class, so it would have a perfect FPR, and likewise a 100% FNR. The *SBStrm* variation, despite its lower overall accuracy, does the best at the novel class detection (best FNR). Looking at the column summary row at the bottom, the ECSM provides a lower false positive rate, but a significantly worse false negative rate (i.e. it misses true novel concepts) compared to the SBStream variation.

Conclusion

The presented methodology demonstrates an improved approach to handling concept evolution and novel class emergence found in data streams, as demonstrated the theoretical

DataSet	ECSM		LB		SB		SBLB		SBStrm	
	FPR	FNR	FPR	FNR	FPR	FNR	FPR	FNR	FPR	FNR
FC@0.001	**0.04**	89.51	n/a	100.00	72.41	14.42	73.72	28.54	89.76	**3.11**
FC@0.05	**2.06**	83.28	n/a	100.00	20.58	**29.18**	20.83	36.42	36.68	46.86
FC@0.1	**3.92**	48.56	n/a	100.00	15.40	**36.63**	15.12	38.31	33.17	43.44
FC@0.25	**3.93**	46.87	n/a	100.00	11.16	**38.43**	11.48	40.91	29.08	48.21
FC@1.0	**3.75**	45.28	n/a	100.00	8.84	**38.70**	8.75	42.15	23.07	48.99
IRND5M@0.001	**0.19**	97.50	n/a	100.00	79.61	8.42	86.31	**1.83**	87.33	5.26
IRND5M@0.05	**0.94**	96.24	n/a	100.00	3.40	94.97	22.66	**46.50**	14.26	79.19
IRND5M@0.1	**0.50**	97.49	n/a	100.00	0.54	98.96	9.57	**71.01**	8.22	84.81
IRND5M@0.25	0.22	98.53	n/a	100.00	**0.02**	99.90	3.99	**84.18**	6.51	86.46
IRND5M@1.0	**0.07**	99.35	n/a	100.00	1.48	97.48	1.46	93.29	7.02	**86.81**
KDD99@0.001	16.00	96.57	n/a	100.00	19.35	66.74	**2.50**	78.07	39.58	**40.25**
KDD99@0.05	11.09	49.33	n/a	100.00	2.76	64.52	**1.31**	74.89	20.23	**21.83**
KDD99@0.1	8.99	64.51	n/a	100.00	1.82	72.74	**1.13**	82.63	18.73	**15.43**
KDD99@0.25	5.19	51.33	n/a	100.00	1.39	78.12	**1.14**	84.53	16.29	**17.66**
KDD99@1.0	4.15	67.18	n/a	100.00	1.35	78.16	**0.90**	88.77	14.05	**49.64**
PAMAP2@0.001	26.07	**44.35**	n/a	100.00	39.59	49.04	16.28	55.34	**5.50**	85.51
PAMAP2@0.05	83.13	**2.65**	n/a	100.00	37.48	64.10	**19.31**	68.25	34.57	100.00
PAMAP2@0.1	69.27	37.25	n/a	100.00	33.14	62.27	**17.06**	72.86	46.43	**31.49**
PAMAP2@0.25	27.33	85.16	n/a	100.00	29.12	63.09	**13.79**	87.60	60.46	**17.98**
PAMAP2@1.0	**6.14**	87.76	n/a	100.00	28.54	60.27	11.04	81.92	76.61	**9.49**
Rank Avg	**1.95**	2.95	n/a	4.95	2.40	2.40	2.05	2.65	3.60	**2.00**

Table 3: Novel Class Experimental Error Results showing algorithm comparison for different data sets (Lower values are better).

basis and empirical evidence presented herein. The SluiceBox method goes beyond the previous established methods like ESCMiner, in that it not only *detects* the emergence of a novel class, but actually *locates* and *tracks* the concepts as deviate and drift, and SluiceBox can be used as a snap-in enhancement to provide these capabilities to any other classifier, enhancing both. Future research will continue to improve the overall classification accuracy and speed of the SluiceBox method to reduce the dependency on co-trained classifiers for such contexts.

Acknowledgments

Supervisory role from author L. Khan supported by NSF Award No. CNS-1229652 and DMS-1322353.

References

Anguita, D.; Ghio, A.; Oneto, L.; Parra, X.; and Reyes-Ortiz, J. L. 2012. Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine. In *Proceedings of the 4th international conference on Ambient Assisted Living and Home Care*, IWAAL'12, 216–223.

Bache, K., and Lichman, M. 2013. UCI machine learning repository.

Bifet, A.; Holmes, G.; and Pfahringer, B. 2010. Leveraging bagging for evolving data streams. In *Proceedings of the 2010 European Conference on Machine Learning and Knowledge Discovery in Databases: Part I*, ECML PKDD'10, 135–150. Berlin, Heidelberg: Springer-Verlag.

Cheng, C.-H.; Fu, A. W.; and Zhang, Y. 1999. Entropy-based subspace clustering for mining numerical data. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '99, 84–93.

Hastie, T.; Tibshirani, R.; and Friedman, J. 2009. *The Elements of Stasticial Learning*. New York: Springer-Verlag, 2nd edition.

Kranen, P.; Kremer, H.; Jansen, T.; Seidl, T.; Bifet, A.; Holmes, G.; Pfahringer, B.; and Read, J. 2012. Stream data mining using the moa framework. In *Proceedings of the 17th International Conference on Database Systems for Advanced Applications - Volume Part II*, DASFAA'12, 309–313. Berlin, Heidelberg: Springer-Verlag.

Lasserre, J., and Bishop, C. M. 2007. Generative or Discriminative? Getting the Best of Both Worlds. *BAYESIAN STATISTICS* 8:3–24.

Masud, M. M.; Chen, Q.; Gao, J.; Khan, L.; Han, J.; and Thuraisingham, B. 2010. Classification and novel class detection of data streams in a dynamic feature space. In *Proceedings of the 2010 European conference on Machine learning and knowledge discovery in databases: Part II*, ECML PKDD'10, 337–352.

Masud, M.; Gao, J.; Khan, L.; Han, J.; and Thuraisingham, B. 2011. Classification and novel class detection in concept-drifting data streams under time constraints. *Knowledge and Data Engineering, IEEE Transactions on* 23(6):859–874.

Mitchell, T. 1997. *Machine Learning*. Singapore: McGraw-Hill, 2nd edition.

Pelleg, D., and Moore, A. W. 2000. X-means: Extending k-means with efficient estimation of the number of clusters. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, 727–734.

Russell, S., and Norvig, P. 2003. *Artificial Intelligence: A Modern Approach*. New Jersey: Prentice Hall, 2nd edition.

Stauffer, C., and Grimson, W. E. L. 1999. Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 2, –252 Vol. 2.

Zweig, A.; Eshar, D.; and Weinshall, D. 2012. Identification of novel classes in object class recognition. In Weinshall, D.; Anemller, J.; and van Gool, L., eds., *Detection and Identification of Rare Audiovisual Cues*, volume 384 of *Studies in Computational Intelligence*. Springer Berlin Heidelberg. 47–55.