Analysis of Particle Filters as an Effective Approach to Handling Concept Drift in Streaming Data

Tegjyot Singh Sethi
Computer Engineering and Computer Science Department
University of Louisville
Louisville, KY
Email: t0seth01@louisville.edu

***Abstract*—The abstract goes here.**

I. Introduction

With the Big Data revolution, there is an ever increasing influx of data which is diverse, dynamic and distributed. The view of data is shifting from one that assumes that data is available in large databases ready for analysis, to one where the data is viewed as a dynamic stream which evolves over time and predictions need to be made in time for them to be of any value. Analyzing streaming data poses unique challenges in terms of making efficient use of time and memory resources and being able to provide good predictive capabilities all at the same time. An additional challenge to mining streaming data is that the models need to adapt to changes to the environment, known as Concept Drift[4][3]. Concept drift makes any model obsolete unless it is retrained on a part of the stream to account for the changes in the input data distribution.

In this paper, a particle filtering based approach to mining data streams, as proposed in [16], is analyzed. Particle filtering is a sequential Monte Carlo approach capable of estimating the posterior distribution of a system's state over time. Particle filters have been used in literature for object tracking applications [1]. With modifications to the basic particle filtering algorithm to use the training accuracy as a measure of particle's performance, instead of likelihood, the PF_LR approach of [16] is made amenable to be used as a stream classification system. The PF_LR algorithm builds a classifier by combining features of all the particles, which themselves represent individual models, making it essentially an ensemble classification approach.

The paper is organized as follows: Section 2 presents a brief overview of popular ensemble techniques used for stream classification and also presents the AUE_NB and the LB_HT methodologies which are used later for comparative analysis. Section 3 describes the PF_LR approach as is used for the classification of streaming data. Section 4 presents description of datasets used and the experimental results obtained. The TJSS dataset was generated specifically for the purpose of analyzing the response of the algorithm in different scenarios.Section 5 presents concluding remarks and avenues for further research in this area.

II. Literature Review

One the most widely used techniques for dealing with streaming data classification in the presence of concept drift is the use of ensemble of classifiers [8]. The effectiveness of ensemble classifiers stems from combining the predictions of several weak classifiers to produce an overall strong prediction. Traditional ensemble classifiers for streams, such as the Simple Voting Ensemble (SVE) [5] and the Weighted Ensemble(WE) [13][6], employ a window based chunk approach. The SVE builds an ensemble using all samples within a chunk and then aggregates the predictions based on a majority vote. The WE also make classifiers on each chunk, but uses a weighted aggregation with higher weight given to models that perform well on the most recent chunk. Dynamic ensembles which can grow and prune models when needed are described in [7], and are more efficient as they generate classifiers only when needed.

The SEA ensemble learning methodology [5] is based on the SVE approach, while the Accuracy Updated Ensemble(AUE) and the Leverage Bagging(LB) are methodologies based on the weighted ensemble approach. In this paper, comparisons is made with the AUE approach with a Naive Bayes base classifier (AUE_NB) and the LB with a Hoeffding tree base classifier(LB_HT). Hoeffding trees are incremental, anytime decision tree induction algorithms that are capable of learning from massive data streams, assuming that the distribution generating examples does not change over time [14]. Hoeffding trees exploit the fact that a small sample can often be enough to choose an optimal splitting attribute. These methodologies used for comparison are already implemented in the MOA framework from Weka [15], and are used as is.

III. Particle Filtering for Classification

Particle filtering is an online Monte Carlo method for performing inference in state-space models where the system evolves dynamically over time [2]. It is used to estimate the posterior density of the state space by directly implementing the Bayesian inference recursion equation. Sequential importance sampling is used to approximate the likelihood function using a tractable chosen distribution. By doing this, particles in subsequent chunks are generate near the ones with high likelihood values. The basic steps in estimating the parameter β of the system is as follows:

1) Generate M particle from the chosen proposal function $q(.|B^{n-1})$ where n is the current chunk.
2) Calculate M incremental weights $w_i^{(n|n-1)}$ for each particle, wehre the weight $w^{(n|n-1|)}$ is given as: $w^{(n|n-1|)} = \frac{p_n(\beta^n|x^n,\beta^{(n-1)})}{q_n(\beta^n|\beta^(n-1))}$ Here $p(\beta, x)$ represents

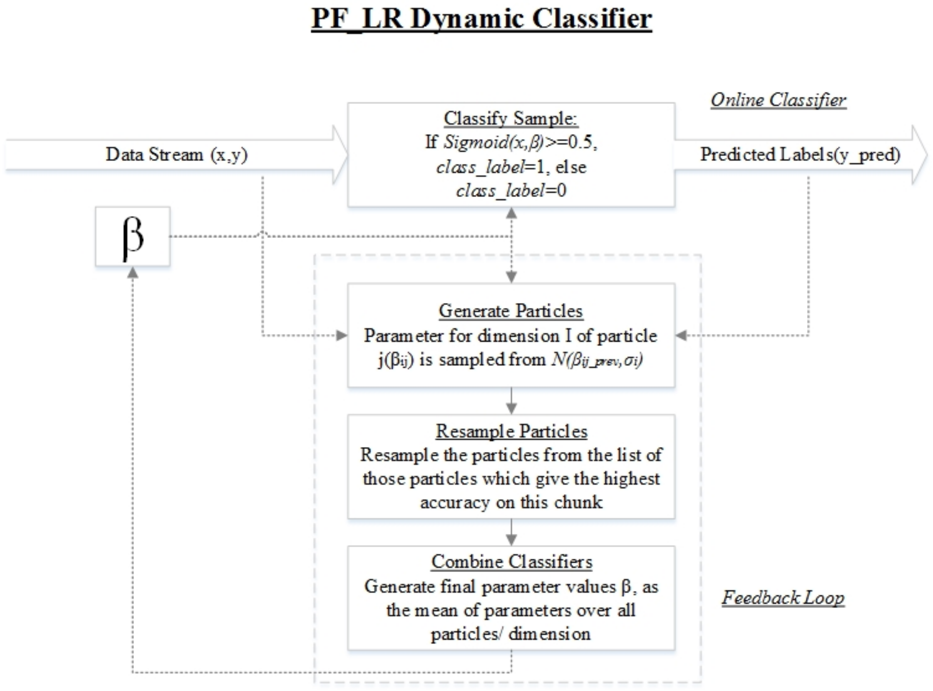

PF_LR Dynamic Classifier
Online Classifier
Data Stream (x,y)
Classify Sample:
If Sigmoid(x,β)>=0.5, class_label=1, else class_label=0
Predicted Labels(y_pred)
β
Generate Particles
Parameter for dimension I of particle j(βij) is sampled from N(βij_prev,σi)
Resample Particles
Resample the particles from the list of those particles which give the highest accuracy on this chunk
Combine Classifiers
Generate final parameter values β, as the mean of parameters over all particles/ dimension
Feedback Loop

Fig. 1: Overview of PF_LR

the posterior distribution of the model parameters and q_n is the chose proposal function.

3) Calculate the weights $w_i^{(1:n)} = w_i^{(1:n-1)}.w_i^{(n|n-1)}$

4) Resample $w_i^n, X_i^{1:n}$ to find M equal weight particles $1/N, \bar{X_i^{1:n}}$

In the proposed methodology of [16](PF_LR), a modification to the basic particle filtering approach was suggested, where instead of using the likelihood function, the training accuracy was used to generate particles in subsequent time intervals. For the task of classification the parameter β should represent the model boundaries as such the Particle filtering algorithm will be an ensemble classification methodology with the final model boundary being a combination of the parameters of all the particles which perform well on the training set. In case of the PF_LR, the basic model is a logistic regression model and hence β represent the coefficient of the regression equation fit on the training data chunk.

The description of the model's functioning is shown in Fig. fig:blkoverall. Each particle in the algorithm represents a parameter set for a logistic regression model. The *Online Classifier* takes the input sample and the parameter values from the last chunk and predicts the class labels for the sample. The predicted label and the actual labels are then passed to the *Feedback Loop*, where the model is retrained to maintain accuracy after a concept drift occurs. The retraining process starts with generation of particles sampled from a Normal distribution centered around the particles position form the previous chunk. The accuracy of each such particle is computed and then a Resampling step is applied to generate M particles from the set of those particles which give the highest training accuracy. This step ensures diversity in the particle set and prevents it from degenerating and getting stuck in a local optima. Since, the batch size is less than M, there are more than one particles which give the maximum accuracy , in most chunks. The final model is generated by taking a mean of the parameters of the resampled particles. In this way the PF_LR algorithm is similar to a bagged ensemble classifier with majority voting. The generated parameter β are used towards the predictions in the next chunk and the cycle is repeated. The complexity of this approach is linear with *O(MN)*, where M is the number of particles and N is the length of the stream.

IV. Experimental Analysis

In this section, the experimental evaluation is presented considering two synthetic streams and one real world streaming dataset. The performance of the analyzed PF_LR approach is presented and compared to that of existing drift handling systems namely: the Accuracy Updated Ensemble and the Leveraged Bagging-Hoeffding Trees approach. The details of the datasets used, experimentation performed and results obtained is presented here.

A. Description of Datasets used

Two synthetic data streams: SEA [5] and TJSS stream and one real world data stream: the EM dataset[**?**][9], are used for performing the experimental evaluation. The datasets are described here. All attributes are normalized to the range of [0,1] using the min-max normalization approach.

1) SEA Dataset: The SEA dataset is a synthetic data stream of 60,000 examples, 3 attributes and 2 classes. Attributes are numeric between 0 and 10, only two are relevant. Drift occurs every 15,000 samples. The drift is generated according to the following rule: the class label is 1 for $relevant\_feature1 +$

$relevant\_feature2 < \theta_{drift}$, where θ_{drift} is a threshold which is varied as-8,9,7,9.5. The data stream has 10% noise.

2) Synthetic TJSS Stream : The TJSS stream is a synthetic two dimensional dataset which exhibits various types of dynamic drifting scenarios that can cause a model to degrade. The stream consists of 25660 samples with two classes.Some salient features of the stream are: Non spherical clusters, clusters with extensions, clusters with total reversal of class distribution, new clusters appearing over time, etc. The Fig. 2 shows the progression of the stream at incremental timestamps. In Fig. 2a) , the initial 10% of the stream is shown, following which the extension of the cluster is shown in b) along with appearance of a new cluster. In c) merging clusters are demonstrated and in d, e), further extension, cluster inceptions and model drifts are exhibited by the stream. The final phase of the stream demonstrates a pure model drift where the class definition of a cluster is totally reversed keeping the model boundary undisturbed. This synthetic stream was generated to evaluate the performance of the GC3 Framework in various drifting scenarios and observe its behavior and response to each of these changes.

3) Electricity Market dataset: The EM dataset were obtained from TransGrid, the electricity supplier in New South Wales, Australia. The data represents fluctuations in the electricity pricing (up/down) from May 1996 to December 1998, with the output recorded relative to a moving average of the last 24 hours. These prices were affected by various external factors such as market demand, weather and time of day; they evolve seasonally and show sensitivity only to short-term events. This data is representative of a real world scenario where the type of drift is not known in advance, is usually erratic and affected by several external factors. It enables us to gain insight into the performance of the framework in a real world scenarios where no prior information on the type of concept drift can be known. The dataset here considers only the last 5 attributes as relevant, contains 45312 samples and two classes.

B. *Parameters and Methodologies used*

The classification performance of the PF_LR methodology is compared against the Accuracy Updated Ensemble and the Leverage Bagging Hoeffding Tree approach, which are available in the MOA tool of Weka [15]. The PF_LR approach was implemented in Python and the performance of all results were compared based on the accuracy of classification. The accuracy is also evaluated against a traditional static baseline Logistic Regression model trained on 10% of the stream, to assess the efficacy of the models as a drift handling system and also to establish the datasets to be suitable for such a kind of evaluation.

The PF_LR methodology was evaluated on 50 for the two synthetic streams and 10 samples/chunk for the EM stream. The σ_i was taken as 0.1 for all i as specified in [16]. The number of particles was taken as 100. The LB_HT approach and the AUE_NB were used with all default parameters in Weka and the accuracy was evaluated with the the same chunk size as used for the PF_LR approach. The accuracy was reported 1000 samples at a time to see the performance over time.

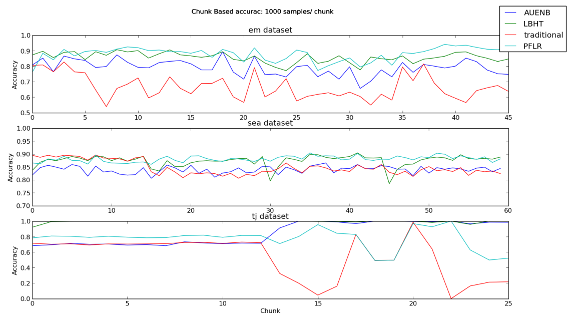


Fig. 4: Accuracy comparison over time 1000 samples/chunk

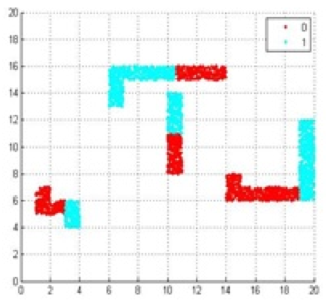
C. *Results and Analysis*

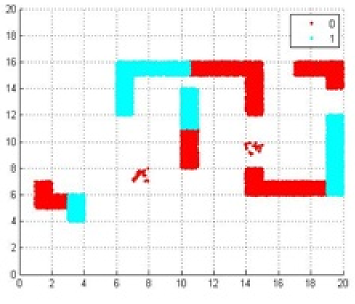
The four models are compared over the streams and the average accuracy over time is presented in Fig. 3. The accuracy over time/chunk is given in Fig. 4, where the average accuracy over 1000 samples is presented for a more fine grained assessment of the drift handling mechanism of the methodologies. The comparison of accuracy is presented in Fig. 5 and Table. I. The PF_LR was run 10 times for each data stream and the mean accuracy values is presented.

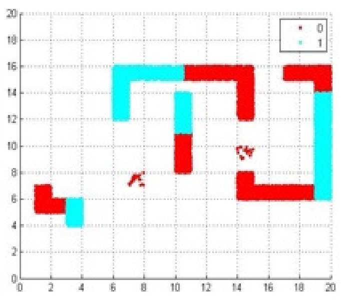
The red line in Fig. 3 shows the baseline which is the performance of a static model trained on 10% of the initial stream. The average accuracy of the LB_HT, AUE_NB and the PF_LR models all have better performance than the traditional model. This indicates that the drifts all have significant dynamics over time and the three drift handling methodologies are capable of detecting and managing changes in the stream. The LB_HT and the PF_LR models perform well over the three datasets.

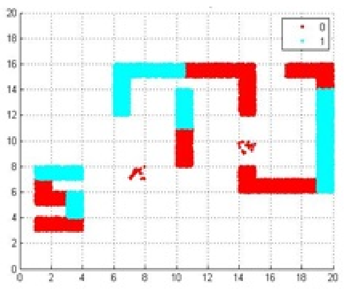
The chunk accuracy plots of Fig. 4 demonstrate the behaviour of the PF_LR model under different drifting scenarios. For the SEA data stream, the drift occurs at chunk 15,30,45,60. This can be seen as a dip at these chunk values and then a subsequent rise indicative of the drift being handled effectively. For the traditional model, each drift leads to a continuous loss in performance from which the system has no way of recovering. In case of the TJSS stream, there is no drift till chunk 13, this is indicated by the accuracies of all models following a similar trajectories. Following the chunk 13, there are different drifting aspect of the stream which are effectively handled by the PF_LR to give better performance than the the traditional static model. However, it can be seen that the AUE_NB and the LB_HT perform much better on this data stream. The EM stream has unknown drifting components and as such is most representative of a real world scenario. The PF_LR performs well on this data demonstrating its ability to be used effectively in scenarios where the drift is unknown and unpredictable. The overall accuracy at the end of the stream is presented in Table. I.

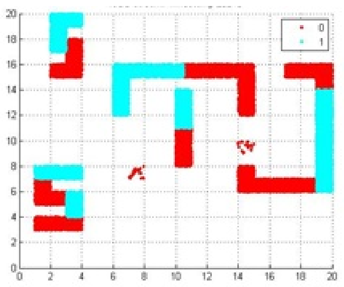
1) Discussion: The PF_LR methodology performs better than the traditional static model and produces comparative results when compared to other drift handling methodologies. However, in case of the TJSS stream, its performance is not as good. The TJSS stream was generated to analyze the performance of a drift handling system in its ability to handle various types of drifts. An analysis of the predicted stream


(a) 10% of TJSS stream (t=2566)


(b) Extension of Cluster, New cluster inception (t=16840)


(c) Merger of Cluster(t=17640)


(d) Extension of cluster with model shift(t=20040)


(e) Inception of new cluster(t=23240)

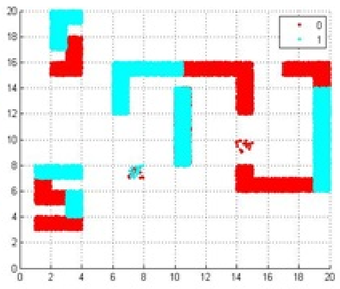

(f) class distribution reversal(t=25660)

Fig. 2: Evolution of TJSS stream over time

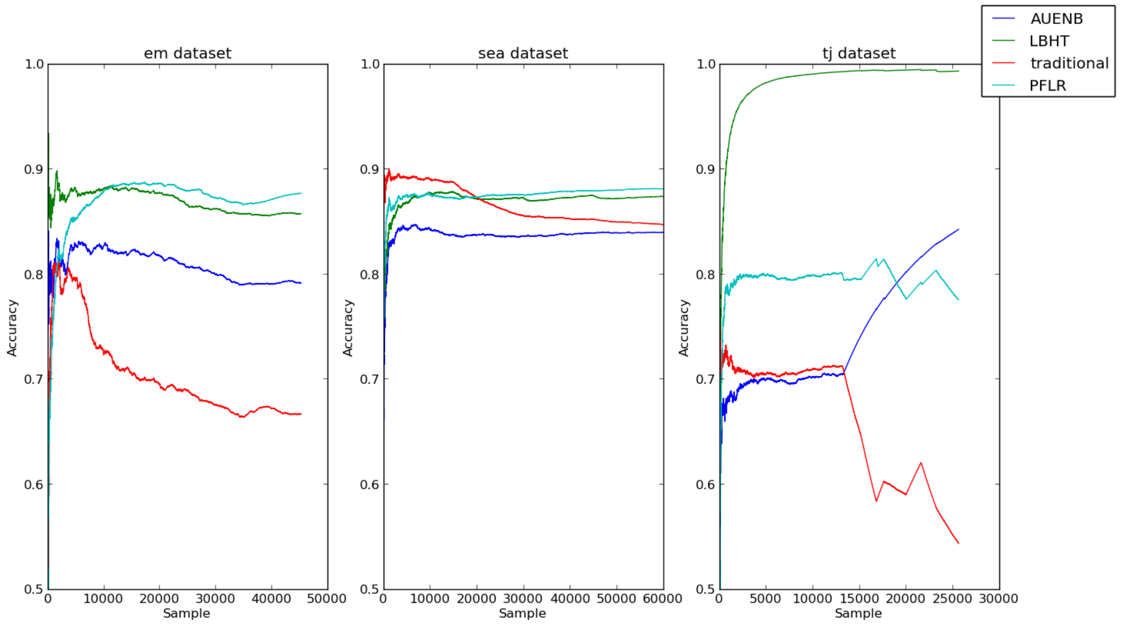


Fig. 3: Accuracy comparison over time

TABLE I: Accuracy comparison of LB_HT,AUE_NB,PF_LR and the traditional model

Model/DataStream	SEA	TJSS	EM
Pf_LR	0.8792±0.004	0.7095±0.003	0.849±0.002
Traditional	0.8467	0.5432	0.6665
LB_HT	0.8735	0.9928	0.8569
AUE_NB	0.8392	0.8422	0.7913

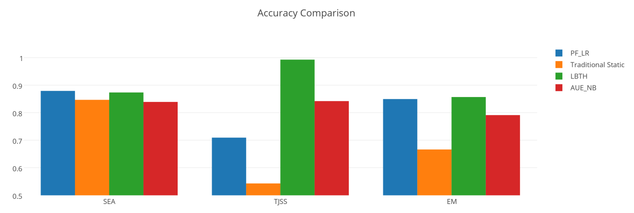


Fig. 5: Overall performance comparison

accuracy per chunk of 100 samples provides for a fine grained understanding of the performance of the PF_LR algorithm, as depicted in Fig. 6. From the figure it can be seen that the model is able to recognize most types of drifts and is able to recover from them fairly soon. This is indicated by a dip in the accuracy followed by a quick ascent.

From the Fig. 6 it can be seen that the PF_LR algorithm follows similar trajectory as the traditional static model till the chunk 130. This is the portion of the stream which does not demonstrate any drift. After this chunk, the performance of the PF_LR sample is increases indicating its ability to adapt well to changes in data distribution (Changes in the T portion of the dataset) that maintain the class boundary. From chunk 180-210, it is seen that the performance of both models is similar. This is the portion of the stream which has extension in a direction opposite to the established class boundary. The PF_LR model has a hard time recognizing these changes. It is however able to account for the newly formed S cluster as seen

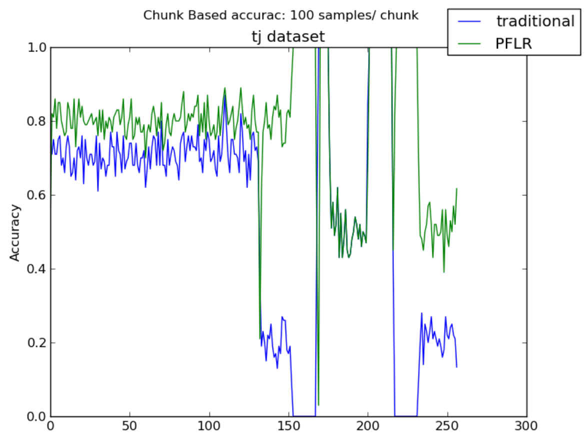


Fig. 6: Accuracy per chunk of 100 samples for the traditional and PF_LR model

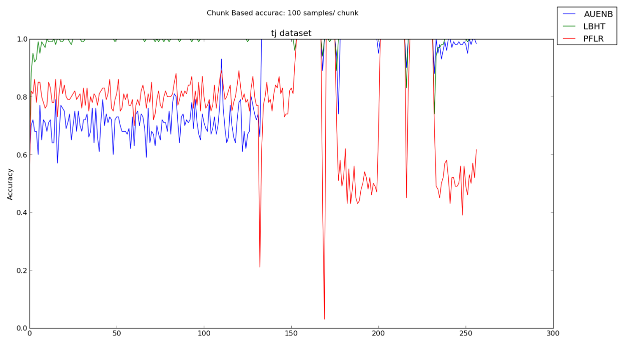


Fig. 7: Accuracy per chunk of 100 samples

in Fig. 2 e). The final change in the stream which represents a reversal of class labels while maintaining the boundary, causes a loss in the PF_LR's performance. But as can be seen from the Fig. 6, this drop in performance is not as bad as the traditional model, implying the ability of the particle filtering model to react effectively although slowly to the drift purely in the class distribution.

The Fig. 7, compares the performance of the drift handling methodologies on the TJSS stream. It can be seen that the LB_HT performs exceptionally well on this dataset. This is indicative of its predictive power over non noisy data streams, irrespective of the nature and frequency of drifts. the AUE_NB approach also performs well on the stream. However, the initial performance before the drift is lower than any of the other methodologies. It should be observed that all the methodologies have similar width of recovery, in case the drift is detected by them. The PF_LR and the AUE_NB have higher drop in accruacy but all of them recover by the same chunk size to give higher prediction accuracy. A comparison of these algorithms indicate that the LB_HT and the AUE_NB are superior than PF_LR in detecting local drifts where a portion of the input data distribution might change while keeping the rest constant. The PF_LR is effective in capturing drifts over noisy real world streams (like the EM stream), especially in scenarios where the drift effects the entire input data distribution globally.

V. Conclusion and Future work

In this paper a particle filtering based online classification approach, the PF_LR algorithm, was analyzed as a viable candidate for handling concept drift in streaming data. The algorithm was analyzed on two synthetic datasets, SEA and TJSS stream, and one real world dataset, the EM stream. The results obtained showed that the PF_LR algorithm is an effective drift handling system as it always performs better than a static model trained on only a subset of the stream. When compared with other state of the art drift handling systems, the AUE_NB and the LB_HT,it's performance was competitive. The PF_LR was able to handle all types of drifts but had a hard time in detecting drifts which affect a subset of the data space, at a time. However, its ability to perform drift handling without explicit drift detection, its linear complexity and its robustness to noise makes it attractive for use as a stream classification system.

This paper assumed that all input samples are labeled and that the label is available immediately after a chunk is processed. This is not true in a streaming environment milieu as labeling can be both time consuming and costly. Thus, the efficacy of using the PF_LR methodology with only partial labeling is an interesting area fro further research.

REFERENCES

[1] Ristic B, Arulampalam S, Gordon N (2004) Beyond the Kalman filter: Particle filters for tracking applications. Artech house.

[2] Andrieu C, Doucet A, Holenstein R (2010). Particle markov chain monte carlo methods. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 72(3), 269-342.

[3] Zliobaite I (2009) Learning under concept drift: an overview. Tech. rep., Technical report, Vilnius University, 2009 techniques, related areas, applications Subjects: Artificial Intelligence

[4] Tsymbal A (2004) The problem of concept drift: definitions and related work. Computer Science Department, Trinity College Dublin

[5] Street WN, Kim Y (2001) A streaming ensemble algorithm (SEA) for large-scale classification. In: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, pp 377–382

[6] Littlestone N, Warmuth MK (1989) The weighted majority algorithm. In: 30th Annual Symposium on Foundations of Computer Science, IEEE, pp 256–261

[7] Kolter JZ, Maloof MA (2007) Dynamic weighted majority: An ensemble method for drifting concepts. The Journal of Machine Learning Research 8:2755–2790

[8] Kong X, Yu P (2011) An ensemble-based approach to fast classification of multi-label data streams. In: 7th International Conference on Collaborative Computing: Networking, Applications and Worksharing, IEEE, pp 95–104

[9] Gong-De G, Nan L, Li-Fei C (2012) Classification for concept-drifting data streams with limited amount of labeled data. In: International Conference on Automatic Control and Artificial Intelligence (ACAI 2012), IET, pp 638–644

[10] Harries M, Wales NS (1999) Splice-2 comparative evaluation: Electricity pricing

[11] Rokach L (2010) Ensemble-based classifiers. Artificial Intelligence Review 33(1-2):1–39

[12] Chen S, He H (2011) Towards incremental learning of nonstationary imbalanced data stream: a multiple selectively recursive approach. Evolving Systems 2(1):35–50

[13] Wang H, Fan W, Yu PS, Han J (2003) Mining concept-drifting data streams using ensemble classifiers. In: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, pp 226–235

[14] Hoeglinger S, Pears R (2007) Use of hoeffding trees in concept based data stream mining. In: Third international conference on information and automation for sustainability ICIAFS 2007, IEEE.

[15] Bifet A, et al.(2010) Moa: Massive online analysis; The Journal of Machine Learning Research 11: 1601-1604

[16] Fok R, An A, Wang X (2013) Mining Evolving Data Streams with Particle Filters.