Adaptive Methods for Classification in Arbitrarily Imbalanced and Drifting Data Streams

Ryan N. Lichtenwalter and Nitesh V. Chawla

The University of Notre Dame, Notre Dame, IN 46556, USA

Abstract. Streaming data is pervasive in a multitude of data mining applications. One fundamental problem in the task of mining streaming data is distributional drift over time. Streams may also exhibit high and varying degrees of class imbalance, which can further complicate the task. In scenarios like these, class imbalance is particularly difficult to overcome and has not been as thoroughly studied. In this paper, we comprehensively consider the issues of changing distributions in conjunction with high degrees of class imbalance in streaming data. We propose new approaches based on distributional divergence and meta-classification that improve several performance metrics often applied in the study of imbalanced classification. We also propose a new distance measure for detecting distributional drift and examine its utility in weighting ensemble base classifiers. We employ a sequential validation framework, which we believe is the most meaningful option in the context of streaming imbalanced data.

Keywords: Sequential learning, stream mining, imbalanced data, skew, concept drift, Hellinger distance.

1 Introduction

Data stream mining is a prolific area of research. In recent years, many papers have been published either directly related to stream mining or addressing challenges in a particular stream mining application. Concept drift is a problem inherent in stream mining that continues to limit performance. Classifier ensembles have been applied in several incarnations to reduce variance, but they cannot combat bias. More recently the problem of class imbalance has been considered in the context of existing stream mining research. The intersection of these difficulties makes for a uniquely interesting and demanding research topic that can contribute to practically every data-driven or data-related field. As data streams become increasingly ubiquitous and prolific, the importance of solving their unique and interrelated challenges grows.

Research in stream mining primarily falls into one of two families: incremental processing [1,2,3,4] or batch processing [5,6]. Figure 1 illustrates. In the former approach, each processing step handles one instance. Labels for an instance are

T. Theeramunkong et al. (Eds.): PAKDD Workshops 2009, LNAI 5669, pp. 53–75, 2010.

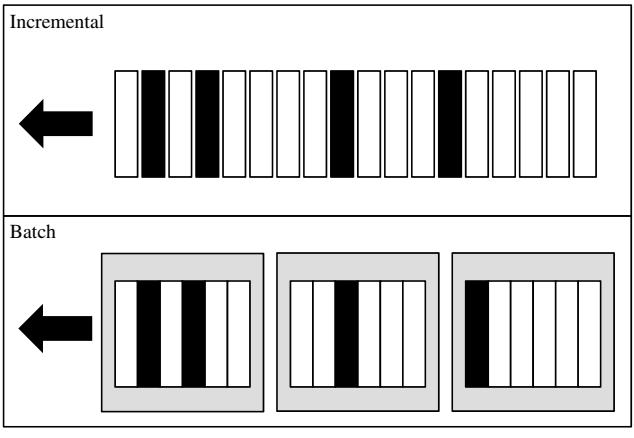

Fig. 1. An illustration of the difference in the way incremental and batch processing methods expect and handle incoming streams of data. With the arrival of each subsequent unit, class labels become available for the previous unit.

assumed to be available immediately after classifying the instance. Instance processing methods have an inherent advantage in terms of their adaptiveness due to this assumption. At time t, instance i_t is received and classified with some model trained on instances i_1 through i_{t-1}. At time $t+1$, a label arrives for instance i_t and unlabeled instance i_{t+1} becomes available for classification. In the latter approach, each batch is a collection of data instances that arrives during a particular period. At time t, batch b_t is received and classified with some model trained on batches b_1 through b_{t-1}. At time $t+1$, labels arrive for batch b_t and unlabeled instances in batch b_{t+1} become available for classification. While the problem spaces for these two approaches do intersect, often the instance processing assumption is unrealistic. In many scenarios, large sets of new data and class labels arrive with low temporal granularity such as weeks, months, or years. The principles presented in this paper operate in the batch processing space.

2 Challenges in Data Streams

Stream mining presents inherent difficulties not present in static distributions. Most significantly, since data streams are generated over time by an underlying hidden function, changes in that function can cause various forms of data drift. Secondarily, the rare class problem becomes more difficult because positive class events, which appear infrequently in a static distribution, may be available in even smaller quantities per unit time in data streams.

2.1 Distributional and Concept Drift

Since data streams are generated over time by an underlying hidden function, changes in that function can cause changes in data distributions and posterior

probabilities, a phenomenon termed *concept drift*. Concept drift occurs when any of the following are present: $\Delta P(\boldsymbol{f})$, a change in the distribution of feature values; $\Delta P(c|\boldsymbol{f})$, a change in the conditional probability of class labels for particular feature values; and both $\Delta P(c|\boldsymbol{f})$ and $\Delta P(\boldsymbol{f})$, when both the feature distribution and the conditional probability of class label appearance change. Examples of these three forms of drift are illustrated in Figure 2.

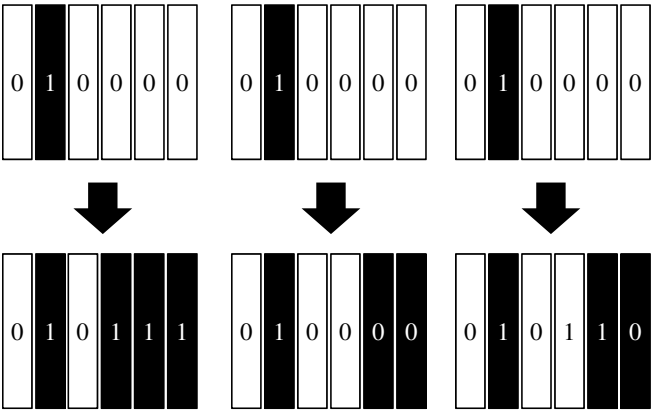

Fig. 2. An illustration of the three different forms of concept drift. The class value is denoted by colors and the feature value is denoted by numbers.

- $\Delta P(\boldsymbol{f})$ (left):
  - $P(0) = \frac{5}{6} \rightarrow P(0) = \frac{1}{3}$
  - $P(1) = \frac{1}{6} \rightarrow P(1) = \frac{2}{3}$
- $\Delta P(c|\boldsymbol{f})$ (middle):
  - $P(white|0) = 1 \rightarrow P(white|0) = \frac{3}{5}$
  - $P(black|0) = 0 \rightarrow P(black|0) = \frac{2}{5}$
- $\Delta P(c|\boldsymbol{f})$ and $\Delta P(\boldsymbol{f})$ (right):
  - $P(0) = \frac{5}{6} \rightarrow P(0) = \frac{1}{2}$
  - $P(1) = \frac{1}{6} \rightarrow P(1) = \frac{1}{2}$
  - $P(white|0) = 1 \rightarrow P(white|0) = \frac{2}{3}$
  - $P(white|1) = 0 \rightarrow P(white|1) = \frac{1}{3}$
  - $P(black|0) = 0 \rightarrow P(black|0) = \frac{1}{3}$
  - $P(black|1) = 1 \rightarrow P(black|1) = \frac{2}{3}$

Henceforth, we designate changes that occur solely in feature distributions, $\Delta P(\boldsymbol{f})$, with the term *distributional drift*.

2.2 Class Imbalance

Imbalanced data is a persistent problem in general data mining contexts. The problem becomes especially difficult in streams. Consider data with 12,000 instances and a class ratio of 100:1. This leaves 120 positive class examples. This

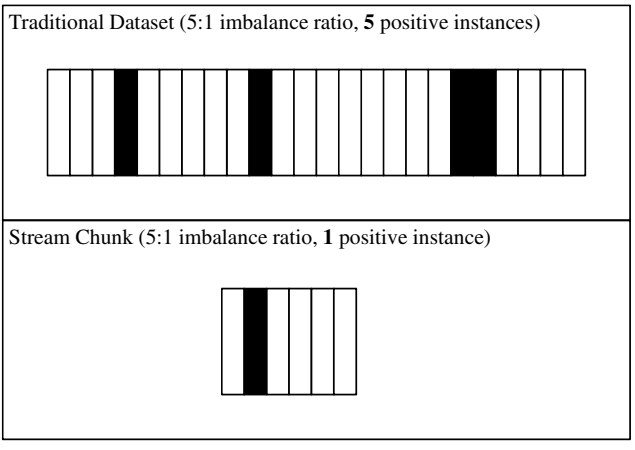

Fig. 3. An illustration of the manifestation of the class imbalance problem in the stream mining domain

may be sufficient to define the class boundaries and leaves plenty of data after application of a good resampling approach. In a streaming context this data may appear over the course of a year. If we assume a uniform temporal distribution of the positive class and want a good model for monthly predictions, we have only 10 positive class examples each month on which to train. This may be insufficient. In reality, some months may exhibit no positive class occurrences at all while others have hundreds. Figure 3 illustrates a simplistic example.

2.3 Problem Intersections

When imbalanced data and concept drift are both present in a data stream, they can cause several confounding effects. First, class frequency inversions for a class c with positive value c_i and negative value c_j can occur such that prior probabilities $P(c_i) << P(c_j)$ may become $P(c_i) >> P(c_j)$ later in the stream. An effective model must be able to adapt to changes of these types. Furthermore, for a given decision threshold θ, posterior probabilities may change for the positive class such that $P(c_i|\boldsymbol{f}) >> \theta$ may become $P(c_i|\boldsymbol{f}) << \theta$. The lack of prevalence of the positive class may result in difficulty assimilating the new concept, thereby rendering models unable to detect the class effectively for an extended period of time.

3 Evaluation Methods

There are two fundamental methods of evaluating classification performance in data streams under the batch paradigm. The first is to evaluate the classifier over a single batch among the many batches of data. The second is to evaluate

the classifier over all of the batches in the data stream and either examine them individually or use some notion of average performance. We highlight the core ideas underlying these two fundamental methods.

Evaluating the classifier over a single batch of data has been used in [5], where the last batch of data is used to render performance scores. Such an evaluation framework might somehow seek to report a concept of average classification accuracy by fulfilling some combination of these three goals:

1. Reporting results on a batch of data representative of the stream as a whole.
2. Reporting results on a batch of data for which classifier performance is representative of some notion of typical or average performance.
3. Reporting results on a batch of data sufficiently late in the stream so that the classifier has captured the concept.

The first and second may seem at first to be the same, but in fact they are not. A simple example proves this. Suppose a heavily drifting binary classification problem C with batches b_1 to b_n and class labels $\in \{0, 1\}$. Further suppose a dumb model M that always adapts to drift by classifying all instances of b_i with the predominant label from b_{i-1}. Now suppose that batches b_1 to b_{n-1} alternate such that instances $\in \{b_i | i = 2m, m \in \mathbb{Z}\}$ have label 0, instances $\in \{b_i | i = 2m + 1, m \in \mathbb{Z}\}$ have label 1, and b_n has a uniform distribution of the two classes. When M is evaluated according to the first goal, b_n is used and M is reported to obtain 50 percent classification accuracy. When M is evaluated according to the second goal, any one of b_1 through b_{n-1} is used and M is reported to obtain 0 percent classification accuracy.

The example is contrived, but nonetheless illustrates that the first two goals are not always aligned in terms of the results they output. Unfortunately, the first goal can output virtually meaningless results and the second goal may be difficult to achieve without evaluating model performance on all batches anyhow. The result is that it may be very difficult to decide on which batch one should evaluate and report performance, especially in the context of the third goal.

The third goal itself may or may not be desirable. Since one of the principle objectives in many approaches to data stream mining is to overcome various forms of drift, waiting until the model has achieved a stable state with respect to its classification performance in the stream may inflate the performance metric. Even in data streams with static distributions and concepts, using a single batch such as the last one does not include information about how quickly the target concept was learned. Two models M_1 and M_2 that reach performance level P somewhere in n batches of data certainly are not equally effective if M_1 reaches P after $\frac{n}{2}$ batches and M_2 reaches P after $\frac{n}{4}$ batches. The second model, in most circumstances should be considered clearly superior, but evaluation on the last batch will consider them the same.

All of these problems may be avoided by employing the second fundamental method. Using all batches does not suffer from the problem of finding representative data. Indeed, in data streams where concepts fluctuate through time, the

existence of any meaningful sense of representative data is dubious since a single batch fundamentally lacks the ability to represent the temporal dimension. Using all batches also guarantees that not only the peak performance is captured in the evaluation, but also the speed at which that performance is reached. For these reasons, we advocate, and unless otherwise stated, employ the evaluation method that averages classification performance metrics over all batches in the data stream.

4 Boundary Definition

We present two findings about how to resample in stream mining to achieve better performance under heavy class imbalance. The first, in line with other findings such as in [7], is that the performance improvement in propagating rare-class instances arises from defining the class boundary better rather than simply providing more positive class instances on which to train. The second arises from the first. In addition to propagating rare-class instances, we propagate instances in the negative class that the current model misclassifies. These help further define the boundary thus increasing precision while minimally affecting recall.

5 Data Set Distance

It is desirable to adapt to drift in a data stream as quickly as possible. Adjusting ensemble weights based on classification performance metrics is a common method, but this is always one batch behind. In a worst case scenario, the data stream fluctuates such that performance on batch i and performance on batch $i+1$ are inversely correlated. Applying weights based on performance metrics in such a scenario will perform worse than not using any weighting scheme at all. We can do better by examining distributional and information-theoretic properties of the data in the past and comparing the results to the testing batch.

5.1 Hellinger Distance

Hellinger distance has recently been used with excellent effect in detecting classifier performance degradation due to distributional changes [8] and even as a decision tree splitting criterion [9]. For this reason, we employ it to construct a consistent measure of distance between two separate data sets or data batches. After discretizing numeric attributes into some number of equal-width bins, we can define Hellinger distance in two batches X and Y for a given feature f as:

$$HD(X,Y,f) = \sqrt{\sum_{v \in f} \left(\sqrt{\frac{|X_{f=v}|}{|X|}} - \sqrt{\frac{|Y_{f=v}|}{|Y|}} \right)^2} \tag{1}$$

Algorithm 1. Misclassified Propagation

Require: unlabeled batch of instances $\boldsymbol{b}$
model m
real class labels l
desired percent positive p
set of positive examples $\boldsymbol{P}$
set of misclassified negative examples $\boldsymbol{N}$
Ensure: resampled batch $\boldsymbol{b}$
updated set of positive examples $\boldsymbol{P}$
updated set of misclassified negative examples $\boldsymbol{N}$

for $instance \in \boldsymbol{b}$ **do**
  $label \leftarrow$PREDICT$(m, instance)$
  if $l_{instance} = +$ **then**
    $\boldsymbol{pos} \leftarrow \boldsymbol{pos} \cup instance$
  else
    $\boldsymbol{neg} \leftarrow \boldsymbol{neg} \cup instance$
    if $l_{instance} \neq label$ **then**
      $\boldsymbol{mneg} \leftarrow \boldsymbol{mneg} \cup instance$
    end if
  end if
end for
while $|\boldsymbol{pos}| < p * (|\boldsymbol{pos}| + |\boldsymbol{neg}| + |\boldsymbol{mneg}|)$ **do**
  if $|\boldsymbol{neg}| > 0$ **then**
    REMOVE-RANDOM($\boldsymbol{neg}$)
  else
    REMOVE-RANDOM($\boldsymbol{mneg}$)
  end if
end while
$\boldsymbol{b} \leftarrow \boldsymbol{pos} \cup \boldsymbol{neg} \cup \boldsymbol{mneg}$
return $\boldsymbol{b}, \boldsymbol{P}, \boldsymbol{N}$

5.2 Information Gain

To extend Hellinger distance to measure the distance between two data sets, we must first devise some method of accounting for the difference in feature relevance. Simple aggregating functions fail to capture this important information. For two features f_i and f_j in training sets X and Y and testing set Z, it is possible that $HD(X, Z, f_i) + HD(X, Z, f_j) << HD(Y, Z, f_i) + HD(Y, Z, f_j)$ and the model trained on X performs more poorly than the model trained on Y. This can happen when $HD(X, Z, f_i) - HD(Y, Z, f_i) < HD(X, Z, f_j) - HD(Y, Z, f_j)$ and f_i is a more relevant feature. In other words, the two distributions of instances for f_i are more proximate than the two distributions of instances for f_j, but f_j is a weaker feature. If we assume that feature relevance remains roughly stable over time we can use the information gain on labeled data to inform the Hellinger distance. Information gain for a batch X is defined as the decrease in entropy H of a class c conditioned upon a particular feature f.

$$IG(X, f) = H(X_c) - H(X_c|X_f) \tag{2}$$

We observe immediately that the product of information gain and Hellinger distance behaves well for most ordinary situations. Because both Hellinger distance and information gain are well-defined, their product is also well-defined with the range $[0, \sqrt{2}]$ and has a sensible meaning. When information gain is high and Hellinger distance is high, the product is high, corresponding to when a highly discriminating feature has significantly drifted. When information gain is high and Hellinger distance is low, the product is low, corresponding to when a highly discriminating feature has remained stable. When information gain is low and Hellinger distance is high, the product is, unfortunately, low.

In this final case, problems may arise. Consider the most extreme case of a batch in which there are no rare class instances. In these cases, there is no reduction in entropy by conditioning on any feature because the initial class distribution has no entropy. The judged difference between this batch and any other for the feature is 0 no matter how divergent the feature distributions are. In reality, our distance function should fall back on Hellinger distance. This way it would not consider two data batches to have 0 distance just because the training set has no rare-class instances and the otherwise useless "predict 1" model is only applied when Hellinger distance indicates that it should be. We can easily achieve this with a simple smoothing, leading us to the following distance function for a single feature.

$$HDIG(X, Y, f) = HD(X, Y, f) * (1 + IG(X, f)) \tag{3}$$

To obtain a normalized distance between two complete data sets, we can apply an arbitrary aggregation function to these relevance-weighted feature distribution distances. For our experiments, we use summation and arrive with the following distance function for two data sets X and Y, which is well defined in the range $[0, 2\sqrt{2}]$ for all data sets:

$$D(X, Y) = \frac{\sum_{f \in X} HDIG(X, Y, f)}{|f \in X|} \tag{4}$$

We apply algorithm 2 to incoming batches to classify their instances with distributions that were most similar in the past.

A distance function measuring distance between data sets X and Y need not be correlated with the performance on Y of a model trained on X for it to be logical, consistent, and useful. For this particular application, however, such a correlation is indicative of a strong potential to provide good weights. We illustrate in Figure 4 a plot of the correlation between the distance and the F_1-measure that results from weighting the ensemble according to the distance. Pearson correlation coefficient provides a rough single number representing the effectiveness of the distance function. For the illustrated example, the coefficient is 0.435. More important is the performance of the function when distances vary greatly. Ignoring the noise in the lower right quadrant of the plot, we observe substantial performance differences correspond to substantial distance differences.

Algorithm 2. Distance-Weighted Ensembles

Require: batches $b_1 \ldots b_t$
models $m_1 \ldots m_t$
unlabeled batch b_{t+1}
Ensure: Probability distributions $\boldsymbol{P}$ for instances $\in b_{t+1}$

for $i = 1$ to t **do**
  $distance_i \leftarrow D(b_i, b_{t+1})$
  $weight_i \leftarrow \frac{1}{distance_i}$
end for
NORMALIZE(w)
for $instance \in b_{t+1}$ **do**
  for $i = 1$ to t **do**
    $\boldsymbol{P}_{instance} \leftarrow \boldsymbol{P}_{instance} + w_i \cdot \text{PREDICT}(m_i, instance)$
  end for
end for
return $\boldsymbol{P}$

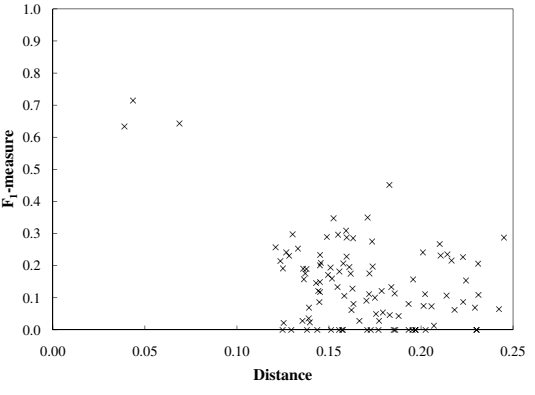

Fig. 4. The correlation between HD-IG distance output and weighted ensemble classification performance. This plot is computed based on the `compustat` data set.

6 Meta-classification

The product of Hellinger distance and information gain ensures that distances appropriately reflect influential features. There are two caveats: (1) it is still possible that the influence of $\Delta P(c|\boldsymbol{f})$ is unrelated to or overpowers $\Delta P(\boldsymbol{f})$ and (2) the distances are constructed respecting the importance of the most powerful features only as measured by the training batch. Feature relevance may change over time, which poses a problem when assigning weights to old batches based on properties such as information gain. Without class labels, it is impossible to determine with complete certainty the relevance of a feature in testing data.

Algorithm 3. Ensembles Weighted Using Meta-Classification

Require: ordered labeled batches $b_1 \ldots b_t$
models m_1 to m_t
unlabeled batch b_{t+1}
regression classifier R
list of performance features $\boldsymbol{F}$
objective performance metric f_o
Ensure: probability distributions $\boldsymbol{P}$ for instances $\in b_{t+1}$
updated regression classifier R

TRAIN(R)
for $i \leftarrow 1$ to n **do**
  $w_i \leftarrow$PREDICT(R, b_i)
end for
NORMALIZE($weights$)
for $instance \in b_{t+1}$ **do**
  for $i = 1$ to t **do**
    $\boldsymbol{P_{instance}} \leftarrow \boldsymbol{P_{instance}} + w_i \cdot$PREDICT($m_i, instance$)
  end for
  for $feature \in b_{t+1}$ **do**
    ADD-FEATURE($meta, HD(b_i, b_{t+1}, feature)$
  end for
  for $f \in F$ **do**
    ADD-FEATURE($meta$,EVALUATE-METRIC($P_{instance}, f$)
  end for
  ADD-CLASS($meta$,EVALUATE-METRIC($P_{instance}, f_o$)
  ADD-INSTANCE($R, \boldsymbol{meta}$)
end for
return $\boldsymbol{P}, R$

We present another method of incorporating distributional divergence measures into ensemble model weighting that can respond to posterior probability drift. We propose a meta-level weighting classifier with feature vector $HD(X, Y, f) \forall f \in \boldsymbol{f}$, and a sliding window of classifier performance metrics. The Hellinger distance metrics provide information about distributional divergence at the level of individual features while the performance metrics allow for weighting based on strong trends in classification success patterns. The class is the performance metric to optimize. After each base classifier makes its prediction and receives performance results, whether the prediction contributes to the ensemble or not, a new instance is generated in the weighting classifier data set. The weighting classifier must be rebuilt and and must then perform regression analysis on the features. In our experiments, we use simple linear regression. Algorithm 3 delineates the process in its most general form. There are many possible modifications such as using only a subset of best base classifiers and modifying frequency with which information is added to the meta-classifier.

Table 1. The format of the data on which regression is performed to predict performance. The example represents a single instance of the metadata containing information about a model trained on b_i and previously used to classify b_j.

Weighting Classifier Features		Class
Distributional	Performance	Objective Metric
$HD(b_i, b_{t+1}, f) \forall f \in b_i$	$MSE(b_j)$	$AUC\text{-}ROC(b_{t+1})$

We leave these for future consideration. The algorithm does not provide the mundane details of training the base classifier, but instead assumes that this is performed separately. In Table 1 we provide an illustration of the weighting classifier data format with AUC-ROC as the optimization objective:

In the same spirit as Section 5.2, we provide Figure 5 to quantify the effectiveness of the meta-classification scheme. The correlation coefficient is 0.577 for F_1-measure and a similar value for other performance measures. This particular example depicts the ability of the meta-classification scheme to handle drift in posterior probabilities alone.

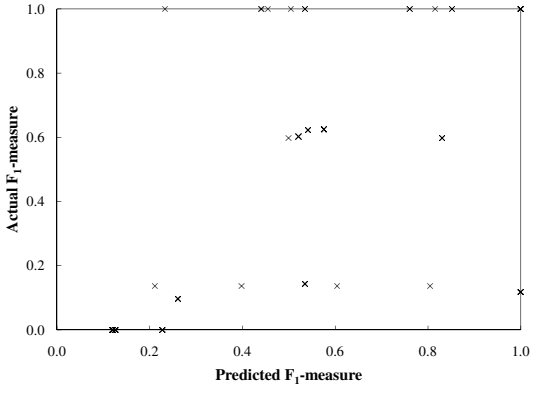

Fig. 5. The correlation between the predicted performance and the actual weighted ensemble classification performance. This plot is computed based on the `STAGGER` data set.

7 Weighting

Even after rendering a distance performance prediction, it is necessary to apply some function to transform outputs to weights. In the case of a distance d, weight should be inversely proportional to closeness, suggesting a function such as $\frac{1}{d}$. In the case of most performance metrics, for which higher values are better, the most simple function given a prediction value p is to assign p directly as the weight.

There are two problems with using such simple functions generally: (1) differences in the values may be too insignificant to create discriminating weights and (2) many bad models can outweigh few good ones. At the same time, we have to be careful not to lose the effectiveness of the ensemble at reducing variance in cases where differences in the base classifiers are insubstantial. We can attack the first problem by using more complex functions such as exponential or power functions. For a distance d our weight would then be x^{-d} or d^{-x} using some base or exponent x. Given a sufficiently large number of batches, any such function, no matter what disparity it creates in the model weights, will allow possibly much better classifiers to be out-ruled. To combat this, we can incorporate the number of ensemble members b to give us functions such as d^{-b} or d^{-bx}.

While these considerations are important, the require a completely separate study involving many data sets and extensive parameter exploration. Inevitably there will still be some small number of classification problems that defy any single general function. For these reasons, we acknowledge this difficult issue, but sidestep it and apply linear weighting functions to achieve all of our results.

8 Data

We operate on different categories of data sets with drastically different properties. Chief among these in number are UCI data sets [10]. They are not inherently sequential and exhibit no concept drift, but we consider these data sets because of their class imbalance. We render them as data streams by randomizing the order of instances and processing them in batches. Most notably, we use `thyroid`, `covtype`, `optdigits`, and `letter-recognition` for direct comparison with [5].

We also present several generated and real world data sets that vary greatly in their degrees of imbalance and distributional drift. The `can` data set captures properties of a crunching can over time and exhibits imbalance at a ratio of 52:1 and extreme distributional drift in the form of blips of positive class instances. `compustat` includes both class imbalance and concept drift, as the training and testing sets represent several years of changing data. The `football` data set comprises features derived from 2003-2008 college football statistics available on ESPN and the class is whether the home team wins. Features drift over time due to underlying changes, the most significant of which are clock rule changes that affect game time. We use a version of the `kddcup2008` data set, which contains instances of breast cancer and exhibits extreme imbalance. `text` is a text-recognition data set with a large number of features. Finally, we include an ordered variation of the Wharton `wrds` data set.

Characteristics of all of these data sets are available in Table 2. Lack of public availability of source data is a problem with many works focusing on streaming data. It is therefore difficult to repeat or verify experiments. For this reason, all data sets for experiments in this paper are publicly available.

Table 2. Data Set Characteristics

	Instances		Features		Properties	
Name	All	Positive	Nominal	Numeric	Imbalance	Ordered
adult	48,842	11,687	8	6	3.2:1	No
boundary	3,505	123	175	0	27.5:1	No
breast-w	569	212	0	30	1.7:1	No
cam	18,916	942	132	0	19.1:1	No
can	443,872	8,360	0	9	52.1:1	Yes
compustat	13,657	520	0	20	25.3:1	Yes
covtype	38,500	2,747	0	10	13.0:1	No
estate	5,322	636	0	12	7.4:1	No
football	4,288	1,597	2	11	1.7:1	Yes
fourclass	862	307	0	2	1.8:1	No
german	1,000	300	0	24	2.3:1	No
ism	11,180	260	0	6	42.0:1	No
kddcup2008	102,294	623	2	123	163.2:1	No
letter	20,000	789	0	16	24.3:1	No
oil	937	41	0	49	21.9:1	No
ozone-1h	2,536	73	0	72	33.7:1	Yes
ozone-8h	2,534	160	0	72	14.8:1	Yes
page	5,473	560	0	10	8.8:1	No
pendigits	10,992	1,142	0	16	8.6:1	No
phoneme	5,400	1,584	0	5	2.4:1	No
phoss	11,411	613	0	480	17.6:1	No
pima	768	268	0	8	1.9:1	No
satimage	6,430	625	0	36	9.3:1	No
segment	2,310	330	0	19	6.0:1	No
splice	1,000	483	0	60	1.1:1	No
STAGGER	12,000	5,303	3	0	1.3:1	Yes
svmguide1	3,089	1,089	0	4	1.8:1	No
text	11,162	709	0	11,465	14.7:1	Yes
thyroid	7,200	166	15	6	42.4:1	No
wrds	99,200	48,832	2	39	1.0:1	Yes

9 Results

We first examine the degree to which the boundary definition (BD) method can improve classification metrics on static data sets. Because this method seeks only to improve performance in the domain of extreme imbalance, we compare it to the other existing work [5], from which it borrows the idea of propagating minority class instances. In Tables 3(a), 3(b), and 3(c) we directly compare the performance for several data sets reported in [5]. We duplicated their methods (SE) for constructing the data sets and evaluating performance as closely as possible. For `letter` we created a separate data for each individual letter with that letter as the positive class. For `optdigits` we followed the same process for each number. We select class 2 as the negative class and class 4 as the positive class in `covtype`. From `thyroid` we create two data sets with class 3 as the

Table 3. Direct comparison of BD and SE

(a) Precision and Recall

	Precision				Recall			
	BL	SE-	SE	BD	BL	SE-	SE	BD
covtype	0.980	0.603	0.770	**0.993**	0.958	**0.999**	0.998	**0.999**
letter	**0.836**	0.398	0.519	0.780	0.719	**0.963**	0.955	0.934
optdigits	0.869	0.765	0.834	**0.915**	0.827	**0.954**	0.952	0.952
thyroid1	**0.914**	0.527	0.728	0.897	0.907	**1.000**	**1.000**	**1.000**
thyroid2	0.940	0.617	0.857	**0.957**	0.982	**1.000**	**1.000**	**1.000**

(b) F_1-measure and PRBEP

	F_1-measure				PRBEP			
	BL	SE-	SE	BD	BL	SE-	SE	BD
covtype	0.969	0.748	0.869	**0.996**	0.968	0.880	0.953	**0.997**
letter	0.771	0.558	0.667	**0.847**	0.739	0.775	0.826	**0.891**
optdigits	0.846	0.847	0.887	**0.932**	0.833	0.896	0.915	**0.936**
thyroid1	0.910	0.686	0.836	**0.945**	0.899	0.876	0.912	**0.937**
thyroid2	0.960	0.762	0.922	**0.978**	0.941	0.860	0.922	**0.975**

(c) AUC-ROC and AUC-PR

	AUC-ROC				AUC-PR			
	BL	SE-	SE	BD	BL	SE-	SE	BD
covtype	0.975	0.999	**1.000**	**1.000**	0.944	0.840	0.951	**1.000**
letter	0.891	0.987	0.989	**0.993**	0.675	0.785	0.847	**0.941**
optdigits	0.909	0.990	0.990	**0.993**	0.768	0.934	0.949	**0.972**
thyroid1	0.951	0.998	**0.999**	**0.999**	0.854	0.897	0.913	**0.934**
thyroid2	0.991	0.995	0.998	**0.999**	0.934	0.862	0.920	**0.975**

negative class: one with 1 as the positive class and the other with 2 as the positive class. After constructing the data sets from the UCI source data, we randomized the order of the instances and repeated all framework experiments ten times. The baseline method (BL) is to train a model on batch t with no resampling and use it to test on batch $t + 1$. For instructive purposes, we also examine the performance of positive class propagation when only the misclassified instances are propagated (SE-). We report several widely accepted imbalanced classification metrics including the area under the receiver operating characteristic curve (AUC-ROC) as implemented in WEKA [11], precision, recall, F_1-measure, and the area under the precision-recall curve (AUC-PR) as implemented in [12]. We observe that AUC-PR is a more discriminating measure than AUC-ROC in the domain of imbalance. Bold font indicates the highest value, but not necessarily statistical significance.

From Table 3(a) we see immediately that propagating misclassified negative class instances improves precision. The method creates a more complete boundary for training. In all cases, it is much closer to baseline precision, while

sacrificing very little in recall. In some cases, it even improves precision beyond the baseline, which uses no resampling. Table 3(b) illustrates the trade-off performance on precision and recall and we see here that BD outperforms SE in terms of F_1-measure by at least 5 percent and as much as 26 percent. Finally, BD always performs at least as well as SE in AUC-ROC while outperforming it for every data set in the more discriminating AUC-PR measure.

For a broader view of the performance of BD with respect to the baseline and with respect to SE, we provide comprehensive coverage of 21 data sets from the UCI repository. To obtain these results, we simply execute each framework ten times on each data set. The data sets have precisely the properties described in 2. Each cell intersecting a method and a data set represents the rank of the method on that data set for the metric in the heading. A rank of 1 means the method performs best, and a rank of three means it performs worst. At the bottom, we provide the mean rank and the overall rank. The overall rank is calculated using the Nemenyi procedure as described in [13] with $\alpha = 0.05$. Statistical significance is indicated between two ranks when a value of one or greater separates them. This comes into play for the F_1-measure column where BD performs significantly better than BL, but SE does not perform statistically significantly better than BL, and BD does not perform statistically significantly better than SE.

Table 4. Performance on Static Data Sets

	Precision			Recall			F_1-measure			AUC-ROC			AUC-PR		
Name	BL	SE	BD	BL	SE	BD	BL	SE	BD	BL	SE	BD	BL	SE	BD
adult	1	3	2	3	1	2	2	3	1	3	2	1	3	2	1
boundary	3	2	1	3	1	2	3	2	1	3	2	1	3	2	1
breast-w	2	3	1	3	2	1	3	2	1	3	2	1	3	2	1
cam	1	2	3	3	1	2	3	1	2	3	1	2	3	1	2
covtype	2	3	1	3	1	2	3	2	1	3	2	1	3	2	1
estate	3	2	1	3	1	2	3	1	2	3	1	2	3	1	2
fourclass	3	2	1	3	1	2	3	2	1	3	2	1	3	2	1
german	1	2	3	3	1	2	3	1	2	3	1	2	3	2	1
ism	1	2	3	3	1	2	1	2	3	3	2	1	2	3	1
letter	1	3	2	3	2	1	1	3	2	3	2	1	3	2	1
oil	1	3	2	3	1	2	1	2	3	3	1	2	3	2	1
page	1	2	3	3	1	2	1	3	2	3	2	1	3	2	1
pendigits	1	3	2	3	1	2	2	3	1	3	2	1	3	2	1
phoneme	2	3	1	3	1	2	3	2	1	3	2	1	3	2	1
phoss	1	2	3	3	1	2	3	1	2	3	1	2	3	1	2
pima	1	3	2	3	1	2	3	1	2	3	2	1	3	2	1
satimage	1	3	2	3	1	2	3	2	1	3	2	1	3	2	1
segment	1	3	2	3	1	2	2	3	1	3	2	1	2	3	1
splice	3	2	1	3	2	1	3	2	1	3	2	1	3	2	1
svmguide1	1	3	2	3	1	2	3	1	2	3	1	2	3	1	2
thyroid	1	3	2	3	1	2	1	3	2	3	2	1	3	2	1
MEAN	1.5	2.6	1.9	3.0	1.1	1.9	2.4	2.0	1.6	3.0	1.7	1.3	2.9	1.9	1.2
OVERALL	1	2.5	1.5	3	1	2	2.5	2	1.5	2.5	2	1.5	2.5	2	1.5

Table 5. Performance on Drifting Data; Weighting Alone

(a) Precision and Recall

	Precision				Recall			
	SA	HD	HDIG	MW	SA	HD	HDIG	MW
adult	0.792	**0.793**	**0.793**	0.792	0.526	0.526	0.526	0.526
can	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
compustat	0.246	0.264	**0.345**	0.275	0.005	0.007	0.011	**0.018**
covtype	0.837	**0.838**	**0.838**	0.837	**0.683**	**0.683**	0.681	0.682
football	0.763	0.763	0.760	**0.764**	0.669	0.670	0.669	**0.671**
kddcup2008	0.038	0.046	0.046	**0.092**	0.005	0.013	0.013	**0.020**
ozone-1h	0.144	0.118	0.118	**0.150**	0.095	0.095	0.095	0.095
ozone-8h	0.154	**0.200**	0.200	0.155	0.061	**0.119**	**0.119**	0.061
STAGGER	0.656	0.656	0.656	**0.741**	0.454	0.454	0.454	**0.588**
text	0.657	**0.659**	**0.659**	0.641	0.693	0.693	0.693	**0.696**
wrds	0.971	0.971	0.971	0.971	**0.875**	**0.875**	0.861	**0.875**

(b) F_1-measure and PRBEP

	F_1-measure				PRBEP			
	SA	HD	HDIG	MW	SA	HD	HDIG	MW
adult	0.631	0.631	0.631	0.631	0.620	0.620	0.620	0.620
can	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
compustat	0.010	0.014	0.021	**0.034**	0.519	0.519	0.519	0.519
covtype	0.751	**0.752**	0.751	**0.752**	0.536	0.536	0.536	0.536
football	0.712	0.713	0.711	**0.714**	0.688	0.688	0.688	0.688
kddcup2008	0.009	0.017	0.017	**0.025**	0.160	0.160	0.160	0.160
ozone-1h	0.104	0.100	0.100	**0.106**	0.516	0.516	0.516	0.516
ozone-8h	0.072	**0.142**	**0.142**	0.072	0.534	0.534	0.534	0.534
STAGGER	0.508	0.508	0.508	**0.630**	0.711	0.711	0.711	0.711
text	0.671	**0.672**	**0.672**	0.664	0.534	0.534	0.534	0.534
wrds	**0.918**	**0.918**	0.904	**0.918**	0.462	0.462	0.462	0.462

(c) AUC-ROC and AUC-PR

	AUC-ROC				AUC-PR			
	SA	HD	HDIG	MW	SA	HD	HDIG	MW
adult	0.892	0.892	0.892	0.892	0.763	0.763	0.763	0.763
can	0.815	0.815	0.815	0.815	0.007	0.007	0.007	0.007
compustat	0.741	0.743	**0.744**	0.743	0.207	0.228	**0.251**	0.226
covtype	0.970	0.970	0.970	0.970	0.834	0.834	0.834	0.834
football	0.855	0.855	0.855	0.855	0.776	**0.777**	**0.777**	0.776
kddcup2008	0.780	**0.790**	0.780	0.767	**0.124**	0.122	0.122	0.121
ozone-1h	0.784	**0.795**	0.794	0.784	0.139	**0.158**	**0.158**	0.139
ozone-8h	**0.748**	0.747	0.746	0.744	0.175	**0.176**	0.175	0.174
STAGGER	0.744	0.739	0.739	**0.795**	0.807	0.807	0.807	**0.849**
text	**0.937**	**0.937**	**0.937**	0.934	0.617	**0.618**	**0.618**	0.610
wrds	0.973	0.973	0.973	0.973	0.698	0.698	0.698	0.698

We can conclude that both BL and BD achieve better precision than SE, which loses precision with respect to BL because it tends to push the border beyond more negative class instances. Although BL usually achieves higher precision than BD, the results are not significant with $\alpha = 0.05$. Although we see that SE outranks BD in terms of recall, a quick look at both Table 3(a) and much of the data underlying the ranks show that the benefit of SE over BD with respect to BL is often negligible given the precision gains. Although individual F_1-measure results suggest that BD often outperforms SE, we cannot say this with $\alpha = 0.05$, nor can we say with confidence that SE outperforms BL. Finally, on both AUC-ROC and AUC-PR, despite frequent wins by BD, the results are not statistically significant at $\alpha = 0.05$. For AUC-PR, at $\alpha = 0.10$ we can say that SE performs worse than BD.

Next, we examine how these methods perform on data that exhibits distributional or concept drift, especially in the context of class imbalance. As control factors, we include `adult` and `kddcup2008`, which exhibit no form of data drift. We also include the benchmark data set, `STAGGER`, which exhibits only concept drift and no distributional drift. `STAGGER` also does not exhibit meaningful imbalance. Table 5 illustrates results for simple average voting (SA), ensembles weighted using Hellinger distance (HD), ensembles weighted using the HDIG distance (HDIG), and ensembles weighted using a meta-classification scheme (MW).

These results indicate that applying weights to ensemble members instead of using the simple average of unweighted probability outputs without any underlying sampling does little to improve classification. The authors of [14] found a similar result in which modifications to a majority voting scheme produced small and inconsistent improvements. `compustat` is an exemplary data set for showing the usefulness of the HD-IG method. Many of the performance metrics show large differences. The exemplary data set for exhibiting the usefulness of the MW method is `STAGGER`. Its posterior drift is detected by the performance features in the performance history data.

Applying ensemble weights seems to produce much more consistent results after resampling. This result is surprising since resampling changes the distribution of the models without changing the distribution of the underlying data used for the distance computation. It is meaningful because HD-IG systematically improves the recall and F_1-measure for almost all of the data sets, although the improvements are minor and apply almost equally to the randomized data in `adult` and `covtype`.

10 Timing and Parallelism

Because stream mining frameworks and algorithms are often employed in time-critical or resource constrained situations, it is more important than in typical non-streaming applications that they perform efficiently. Aside from any specific performance or resource constraints, mining data streams has an inherent requirement that the classification task have a throughput at least equal to the rate at which instances or batches arrive. Research like that in [15] even explores the topic specifically.

Table 6. Performance on Drifting Data; Weighting and Resampling

(a) Precision and Recall

	Precision			Recall		
	BD	BD-HDIG	BD-MW	BD	BD-HDIG	BD-MW
adult	0.614	**0.618**	0.614	0.781	**0.783**	0.776
can	0.027	0.027	0.027	0.011	**0.014**	0.012
compustat	0.142	**0.161**	0.128	0.326	**0.396**	0.374
covtype	0.728	**0.734**	0.728	0.892	**0.899**	0.897
football	0.774	0.774	0.774	0.651	**0.655**	0.650
kddcup2008	**0.071**	0.062	0.070	0.370	**0.379**	**0.379**
ozone-1h	0.063	**0.074**	0.055	0.455	**0.552**	0.431
ozone-8h	0.133	0.134	**0.149**	0.422	**0.428**	0.425
STAGGER	0.569	0.598	**0.686**	0.454	0.454	**0.588**
text	0.510	0.511	**0.558**	**0.831**	0.816	0.821
wrds	0.973	0.973	**0.974**	**0.874**	0.801	0.873

(b) F_1-measure and PRBEP

	F_1-measure			PRBEP		
	BD	BD-HDIG	BD-MW	BD	BD-HDIG	BD-MW
adult	0.686	**0.689**	0.685	0.783	**0.785**	0.784
can	0.012	**0.016**	0.014	0.021	0.021	0.021
compustat	0.185	**0.217**	0.183	0.701	0.701	0.701
covtype	0.796	**0.802**	0.798	0.708	0.709	0.709
football	0.707	**0.709**	0.706	**0.782**	0.781	**0.782**
kddcup2008	**0.105**	0.098	**0.105**	**0.325**	0.323	0.323
ozone-1h	0.110	**0.129**	0.096	0.701	0.701	0.701
ozone-8h	0.199	0.203	**0.218**	0.700	0.700	0.700
STAGGER	0.475	0.488	**0.607**	0.815	**0.818**	0.813
text	**0.629**	0.611	0.625	0.700	0.700	0.700
wrds	**0.918**	0.854	**0.918**	0.918	0.918	**0.919**

(c) AUC-ROC and AUC-PR

	AUC-ROC			AUC-PR		
	BD	BD-HDIG	BD-MW	BD	BD-HDIG	BD-MW
adult	0.905	**0.906**	0.905	0.783	**0.784**	0.783
can	0.819	**0.821**	0.819	0.026	0.026	0.026
compustat	0.770	0.774	**0.780**	0.207	**0.210**	0.184
covtype	**0.982**	0.981	**0.982**	0.866	0.866	0.866
football	**0.862**	0.861	**0.862**	**0.795**	0.794	**0.795**
kddcup2008	0.931	**0.940**	0.936	0.201	**0.225**	0.196
ozone-1h	0.751	**0.785**	0.727	**0.108**	0.102	0.095
ozone-8h	0.724	**0.728**	0.725	0.159	0.165	**0.168**
STAGGER	0.709	0.704	**0.753**	0.788	0.785	**0.812**
text	0.941	0.936	**0.942**	**0.625**	0.617	0.622
wrds	**0.974**	0.973	**0.974**	**0.975**	0.974	**0.975**

The increased time requirement for BD is minimal but evident since it must consider the misclassified negative class instances through time. We demonstrate here that the performance benefit of DW is offset by a modest increase in computational requirements. We begin with a theoretical examination and move to an empirical study. Hellinger distance may be computed on a single feature in $O(e)$ where e is the number of instances in the batch. Computing Hellinger distance for all features is $O(|\boldsymbol{f}| \cdot e)$. As each batch with unclassified instances arrives, all previous batches must be compared with only the new batch. The number of Hellinger distance computations for each batch therefore increases linearly with the number of batches. For a batch i there will be $O(|\boldsymbol{f}| \cdot e) \cdot O(i)$ operations for ensemble construction. Because the information gain component of the distance measure can be saved for each feature in each batch after computation, its growth through time is described by $O(1)$ and its time requirement for each batch is $O(e)$, which is subsumed by the Hellinger distance computation. The overhead requirement for preparing the weights for all batches in a stream up to and including batch n is such:

$$\sum_{i=0}^{n} O(|\boldsymbol{f}| \cdot e + 1) = \frac{n \cdot (n+1)}{2} \cdot O(|\boldsymbol{f}| \cdot e) = O\left(n^2 \cdot |\boldsymbol{f}| \cdot e\right) \tag{5}$$

The task also enjoys simple parallelization. Distance computations for a batch b_i and the incoming test batch b_{t+1} are entirely independent and may easily be done in parallel. Each of the ensemble members can run independently to produce its probability estimations or classification output on new unlabeled instances.

We include BL, SE, BD, and HD for comparison. We omit HD-IG because information gain computation results can be saved resulting in negligible contributions to time requirements. We also omit MW since its complexity is dependent on the choice of classifier and the specific nature of the problem at hand, although we do observe that its growth will be at least linear growth in the number of batches. Theoretically, BL should exhibit no increase in time requirements with incoming batches. SE should exhibit small increases with each batch as SE requires all previously constructed models to provide a probability estimation and training time increases for the increasingly large resampled batches.

For empirical study, we use `kddcup2008` data only because the data volume is sufficient to accommodate 80 batches of 1278 instances each. Times include the entire training and evaluation process for each batch. We made no effort to improve the performance of any of the algorithms over any others. We observe that all methods except BL theoretically exhibit, with varying coefficients, linear growth in the number of batches over time. Finally, Figure 6 shows all methods requiring more time than BL for almost all batches. This is only a consequence of the fact that we divided the data into batches with small numbers of instances so that sufficient batches would exist to illustrate the growth patterns. In reality, the BL method will require longer whenever methods that undersample remove a significant portion of the batch.

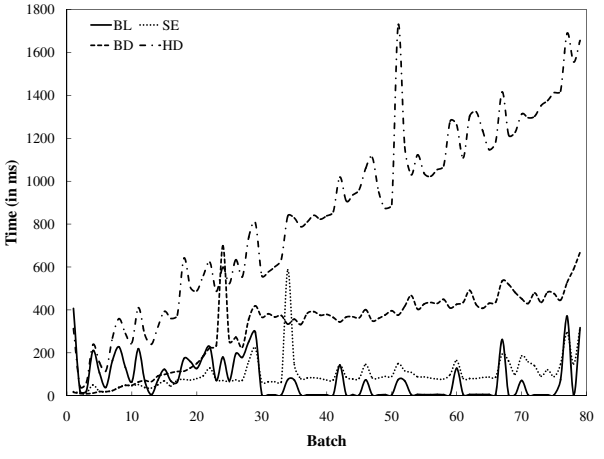

Fig. 6. The time requirement in milliseconds for processing batches from 1 through 80

11 Conclusions

Generating classifier ensembles to detect and address concept drift in a proactive manner is a powerful new solution. It performs dramatically better than existing methods on extremely complicated data streams with complex concept drift and high degrees of imbalance. Simultaneously, it sacrifices no significant performance on simpler data sets with no concept drift and only moderate imbalance. Often classification improvements are measured in a few percentage points of accuracy or small fractional increases in AUC-ROC. Our method achieves gains in several performance metrics that are as great as an order of magnitude with respect to preexisting methods of handling the intersecting problems of concept drift and imbalance.

We acknowledge that the assumption that feature relevance is stable over time does not always hold. Several works in the field observe that it is not [16,17,18,19]. We can say that for all the data sets on which we tested, the information gain adjusted distance function performed better than Hellinger distance alone. For arbitrarily complex drifts in $\Delta P(\boldsymbol{f})$ or drifts in $\Delta P(c|\boldsymbol{f})$ signaled by $\Delta P(\boldsymbol{f})$, distributional divergence performs well. It is theoretically impossible to detect with certainty that a new batch contains a posterior probability shift. In situations like these, it may become necessary to sample a few instances from the upcoming batch as in [20]. Another alternative is to combine performance-based weighting with the weighting proposed here.

We believe that the distance measure proposed herein is an excellent way to measure the distance between two data sets. In its current form, it only uses information gain from one of the two data sets to provide a distance, but it can easily be extended to accept information gain from two sources when class labels are available from both. We hope to compare this measure of data set distance to several others to compare properties and usefulness.

12 Related Work

Researchers have extensively studied class imbalance. The most popular approaches for handling imbalance are various forms of resampling to achieve a more balanced distribution, including random undersampling of the negative class and SMOTE [21]. A more recent approach is an active learning system capable of efficiently querying large data sets to find informative examples [22].

Together, [23] and [24] serve as an excellent overview of the principles of concept drift. An early seminal paper in overcoming concept drift was [25]. Since then, there has been much research in data stream mining to attempt to react to concept drift as quickly as possible. Many batch approaches employ some form of ensemble technique. Some use simple average [5], while others employ weighting [3,6]. With various levels of sophistication, some systems maintain a sliding window of examples or models. [26] and [27] dynamically adjust window size and the former provides strict performance guarantees. We observe that a contiguous window, even an optimally sized one, may fail to contain recurring concepts if they were previously forgotten by the system. This concept of periodicity is briefly discussed in [1]. The FLORA systems have the ability to reactivate old concepts outside the window [4].

We are aware of few publications that directly address the notion of class imbalance in combination with concept drift in data streams. The FLORA systems include distinct rules for positive examples [4], but do not directly target them. Work in [28] focuses on detecting concept drift in potentially adversarial scenarios such as the perpetration of fraud. In [5], which specifically addresses imbalance, the primary contribution was the idea of propagating positive class examples to overcome class imbalance.

Much of our work focuses on distributional divergence. Although the topic has not been explored extensively in general stream mining, it is a familiar concept in the related field of novelty detection, which is the task of realizing the occurrence of previously unobserved or infrequently observed concepts. An excellent review of distributional measures in novelty detection appears in [29]. Although it does not implement or report results for any methods, [24] addresses the potential of using distributional considerations to detect concept drift. The work in [30] proposes modeling the underlying distributions in data and using the models to detect change. In [31], distributional measures are used for ensemble weighting. Finally, [20] uses distributional measures in the forms of expected loss due to feature dissimilarity and decision tree leaf statistics to determine when data distributions are changing outside an acceptable level.

References

1. Becker, H., Arias, M.: Real-time ranking with concept drift using expert advice. In: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 86–94. ACM Press, New York (2007)
2. Hulten, G., Spencer, L., Domingos, P.: Mining time-changing data streams. In: KDD 2001: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 97–106. ACM, New York (2001)

3. Kolter, J.Z., Maloof, M.A.: Dynamic weighted majority: An ensemble method for drifting concepts. Journal of Machine Learning Research 8, 2755–2790 (2007)
4. Widmer, G., Kubat, M.: Learning in the presence of concept drift and hidden contexts. Machine Learning 23, 69–101 (1996)
5. Gao, J., Fan, W., Han, J., Yu, P.S.: A general framework for mining concept-drifting data streams with skewed distributions. In: SDM 2007: Proceedings of the SIAM International Conference on Data Mining (2007)
6. Wang, H., Fan, W., Yu, P.S., Han, J.: Mining concept-drifting data streams using ensemble classifiers. In: KDD 2003: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 226–235. ACM, New York (2003)
7. Han, H., Wang, W.Y., Mao, B.H.: Borderline-smote: A new over-sampling method in imbalanced data sets learning. Advances in Intelligent Computing, 878–887 (2005)
8. Cieslak, D.A., Chawla, N.V.: Detecting fractures in classifier performance. In: ICDM 2007: Seventh IEEE International Conference on Data Mining, pp. 123–132 (2007)
9. Cieslak, D.A., Chawla, N.V.: Learning decision trees for unbalanced data. In: European Conference on Machine Learning. Springer, Heidelberg (2008)
10. Asuncion, A., Newman, D.: Uci machine learning repository (2007)
11. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005)
12. Davis, J., Goadrich, M.: The relationship between precision-recall and roc curves. In: ICML 2006: Proceedings of the 23rd international conference on Machine learning, pp. 233–240. ACM, New York (2006)
13. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. The Journal of Machine Learning Research 7, 1–30 (2006)
14. Street, N.W., Kim, Y.: A streaming ensemble algorithm (sea) for large-scale classification. In: KDD 2001: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 377–382. ACM, New York (2001)
15. Haghighi, P.D., Gaber, M.M., Krishnaswamy, S., Zaslavsky, A., Seng, L.: An architecture for context-aware adaptive data stream mining. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) ECML 2007. LNCS (LNAI), vol. 4701. Springer, Heidelberg (2007)
16. Blum, A.: Empirical support for winnow and weighted-majority algorithms: Results on a calendar scheduling domain. Machine Learning 26, 5–23 (1997)
17. Forman, G.: Tackling concept drift by temporal inductive transfer. In: SIGIR 2006: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, pp. 252–259. ACM, New York (2006)
18. Harries, M., Horn, K.: Detecting concept drift in financial time series prediction using symbolic machine learning. In: Eighth Australian Joint Conference on Artificial Intelligence, pp. 91–98. World Scientific Publishing, Singapore (1995)
19. Widmer, G.: Tracking context changes through meta-learning. Machine Learning 27, 259–286 (1997)
20. Fan, W., Huang, Y.a., Wang, H., Yu, P.S.: Active mining of data streams. In: Proceedings of the Fourth SIAM International Conference on Data Mining, Society for Industrial Mathematics, pp. 457–461 (2004)
21. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, P.W.: Smote: Synthetic minority over-sampling technique. Journal of Artificial Intelligence Research 16, 341–378 (2002)

22. Ertekin, S., Huang, J., Bottou, L., Giles, L.: Learning on the border: Active learning in imbalanced data classification. In: CIKM 2007: Proceedings of the sixteenth ACM Conference on information and knowledge management, pp. 127–136. ACM, New York (2007)
23. Kelly, M.G., Hand, D.J., Adams, N.M.: The impact of changing populations on classifier performance. In: KDD 1999: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 367–371. ACM, New York (1999)
24. Kuncheva, L.I.: Classifier ensembles for detecting concept change in streaming data: Overview and perspectives. In: Proceedings of the 2nd Workshop SUEMA 2008 (ECAI 2008), pp. 5–10 (2008)
25. Schlimmer, J.C., Granger, R.H.: Incremental learning from noisy data. Machine Learning 1, 317–354 (1986)
26. Bifet, A., Gavaldá, R.: Learning from time-changing data with adaptive windowing. In: SIAM International Conference on Data Mining, SDM 2007 (2006)
27. Klinkenberg, R.: Using labeled and unlabeled data to learn drifting concepts. In: Workshop notes of the IJCAI 2001 Workshop on Learning from Temporal and Spatial Data, pp. 16–24 (2001)
28. Phua, C., Miles, K.S., Lee, V., Gayler, R.: Adaptive spike detection for resilient data stream mining. In: Proceedings of the sixth Australasian conference on Data mining and analytics (AusDM 2007), pp. 181–188. Australian Computer Society, Inc., Darlinghurst (2007)
29. Markou, M., Singh, S.: Novelty detection: A review - part 1: Statistical approaches. Signal Processing 83, 2481–2497 (2003)
30. Korn, F., Muthukrishnan, S., Wu, Y.: Modeling skew in data streams. In: SIGMOD 2006: Proceedings of the 2006 ACM SIGMOD international conference on Management of data, pp. 181–192. ACM, New York (2006)
31. Nishida, K., Yamauchi, K., Omori, T.: Ace: Adaptive classifiers-ensemble system for concept-drifting environments. Multiple Classifier Systems, 176–185 (2005)