Quantification and Semi-Supervised Classification Methods for Handling Changes in Class Distribution

Jack Chongjie Xue
Department of Computer and Information Science
Fordham University
441 East Fordham Road, Bronx, NY 10458
718-817-3190
xue@cis.fordham.edu

Gary M. Weiss
Department of Computer and Information Science
Fordham University
441 East Fordham Road, Bronx, NY 10458
718-817-0785
gweiss@cis.fordham.edu

ABSTRACT
In realistic settings the prevalence of a class may change after a classifier is induced and this will degrade the performance of the classifier. Further complicating this scenario is the fact that labeled data is often scarce and expensive. In this paper we address the problem where the class distribution changes and only unlabeled examples are available from the new distribution. We design and evaluate a number of methods for coping with this problem and compare the performance of these methods. Our quantification-based methods estimate the class distribution of the unlabeled data from the changed distribution and adjust the original classifier accordingly, while our semi-supervised methods build a new classifier using the examples from the new (unlabeled) distribution which are supplemented with predicted class values. We also introduce a hybrid method that utilizes both quantification and semi-supervised learning. All methods are evaluated using accuracy and F-measure on a set of benchmark data sets. Our results demonstrate that our methods yield substantial improvements in accuracy and F-measure.

Categories and Subject Descriptors
I.2.6 [**Artificial Intelligence**]: Learning-*induction*

General Terms
Algorithms, Measurement, Design

Keywords
Semi-supervised learning, quantification, classification, concept drift, class distribution

1. INTRODUCTION
In real-world data mining settings it is often the case the classification "concept" we are trying to learn may change over time and, in particular, may change after a classifier is induced. This problem is known as *concept drift* [14] and in this paper we focus on a specific type of concept drift where the class distribution changes over time, yielding a *distribution mismatch* [7] problem. This problem occurs frequently. For example, epidemiologists often find that although the cause of a disease is stable, the prevalence of the disease changes over time. The same phenomenon has been found in help-desk support applications, where the occurrence of certain support issues varies over time (e.g., there are more reports of cracked computer screens on July 4, the U.S. Independence day [7]). This problem of a changing class distribution is further complicated by the fact that labeled examples are often scarce or costly to obtain—and it may not even be possible to label newly acquired examples in a timely manner.

This paper focuses on two research questions associated with the data mining scenario just described: 1) How can we maximize classification performance when the class distribution changes but is unknown, and 2) How can we utilize unlabeled data from the changed class distribution to accomplish this goal?

More formally, the class of problems we study has some *original* distribution, D_{orig}, from which we are provided a set of labeled examples, $ORIG_{label}$, with class distribution $ORIG_{CD}$. At some point the distribution of data changes to D_{new} with a new but unknown class distribution, NEW_{CD}, and from this distribution we are provided with a set of unlabeled examples, $NEW_{unlabel}$. For evaluation purposes we are also provided with labeled examples, NEW_{eval}, drawn from D_{new}. Given this terminology we can state our learning problem more precisely.

Problem statement:

Given: $ORIG_{label}$ drawn from D_{orig} (with $ORIG_{CD}$)
$NEW_{unlabel}$ drawn from D_{new} (with unknown NEW_{CD})
NEW_{eval} drawn from D_{new}

Do: Construct the classifier *C*, using $ORIG_{label}$ and/or $NEW_{unlabel}$, which yields the best possible classification performance on NEW_{eval}.

We introduce Figure 1 to illustrate the distribution mismatch problem and to establish some performance goals for our work. Figure 1 shows how two baseline methods, Naïve and Oracle, perform for a two-class data set when the original class distribution is balanced (i.e., $ORIG_{CD}$ = 1:1 with a positive class rate of 50%) but then is altered so that the class distribution for the new distribution (NEW_{CD}) varies between 1% and 99% positive examples, in 1% increments (details of the experiment are provided in Section 3).

The **Naïve** approach ignores the unlabeled data and the fact that the class distribution may change and utilizes the classifier induced from $ORIG_{label}$ to classify the examples in NEW_{eval}. The **Oracle** method provides a potential upper bound for achievable performance by building a classifier using $NEW_{unlabel}$ with the true class labels "uncovered." Evaluation is based on NEW_{eval}.

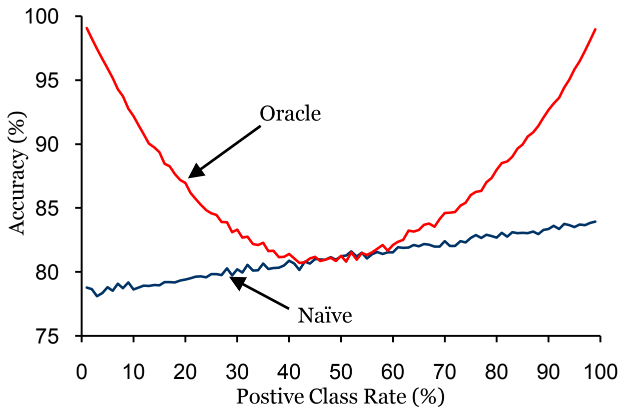

Figure 1. Classifier Performance on Adult Data Set

The results for Naïve clearly demonstrate the distribution mismatch problem since the accuracy of Naïve degrades with respect to the "desired" performance of Oracle for most cases where $ORIG_{CD} \neq NEW_{CD}$ (the shape of these curves is discussed in Section 4). We can state our performance goals in terms of these two baseline methods: we will develop methods that perform strictly better than Naïve and approach the performance of Oracle.

In this paper we utilize two basic techniques for improving classifier performance beyond that of the Naïve approach: class distribution estimation (CDE) and semi-supervised learning (SSL). The CDE technique exploits the fact that if we can estimate the class distribution from which future examples will be drawn, then we can adjust the original classifier to account for the differences in class distribution. In this paper we describe and analyze two CDE-based methods: an iterative method of our own design and a quantification-based method based on a quantification technique [7]. The CDE-based methods use $NEW_{unlabel}$ in the learning process, but only to estimate NEW_{CD}. Our semi-supervised learning methods, on the other hand, use examples from $NEW_{unlabel}$ in the classifier induction process, where these new examples are assigned predicted class labels. We introduce two main SSL-based methods: a simple method that only uses the examples from $NEW_{unlabel}$ to build the classifier and a self-training [20] variant that iteratively merges the examples from $ORIG_{label}$ with $NEW_{unlabel}$. Finally, we introduce a hybrid method that integrates features from the CDE-based and SSL-based methods. We evaluate all methods using accuracy and F-Measure.

The remainder of this paper is structured as follows. Section 2 describes our methods in detail. Section 3 presents our experiment methodology and our results are then presented and analyzed in Section 4. Related work is described in Section 5 and Section 6 summarizes our conclusions and discusses areas for future research.

2. METHODS

In this section we describe the methods used to handle changes in class distribution, with the exception of the **Naïve** and **Oracle** methods, which were introduced earlier. Recall that these methods serve as lower and upper performance bounds, respectively, for our methods. Our class distribution estimation methods require some background before they can be properly understood and this background is provided in Section 2.1. The CDE-based methods are then described in Section 2.2, the SSL-based methods in Section 2.3, and the hybrid CDE/SSL method in Section 2.4.

2.1 Understanding Class Distribution Changes

In this section we discuss the impact that a changing class distribution has on classification and how we can compensate for this change in class distribution. We begin by introducing some basic terminology. Table 1 shows a standard confusion matrix for a two-class domain, where all predictions can be categorized as true positives (TP), false negatives (FN), false positives (FP), and true negatives (TN).

Table 1. Confusion Matrix for Binary Classification

		Predicted Class	
		positive	negative
Actual Class	positive	TP	FN
	negative	FP	TN

The following terms are defined based on the values in the confusion matrix: positive rate (*pr*), negative rate (*nr*), distribution mismatch ratio (*dmr*), true positive rate (*tpr*), and false positive rate (*fpr*). We use the prime symbol (′) to denote the values associated with the new distribution.

$$pr = (TP+FN)/(TP+FN+FP+TN)$$
$$nr = (FP+TN)/(TP+FN+FP+TN)$$
$$dmr = (pr/nr) : (pr'/nr')$$
$$tpr = TP/(TP+FN)$$
$$fpr = FP/(FP+TN)$$

Now that we have introduced the basic terms we can discuss what happens if the class distribution changes and how we can compensate for these changes. This topic is described in detail from a theoretical perspective by Elkan [6] and an applied perspective by Weiss and Provost [16]. In the interest of clarity we discuss the issue from the applied perspective and use an example to motivate the key concepts and explain the relevant equations.

In our example the data set drawn from the original distribution has 900 positive examples and 100 negative examples ($pr = 9/10$, nr=1/10) and the data drawn from the new distribution has 200 positive and 800 negative examples (pr'=2/10, nr'=8/10). The distribution mismatch ratio *dmr* indicates the factor by which the ratio of positive to negative examples changes between the original and new distribution. For this example $dmr = 9{:}¼$ or, equivalently, 36:1. Thus, based on the *ratio* of the positive rate to negative rate, the positives are 36 times more prevalent in the original distribution than in the new distribution. Note that if we used the *fraction* of positive examples rather than the positive to negative ratio, then *dmr* would only be 4.5 (i.e., .9/.2), but if fractions are used then equation 1 becomes much more complex and difficult to understand [16].

We can adjust for changes in class distribution during the classifier induction process by any of these three methods [6]: 1) sampling (or reweighting) the training examples so as to alter the class distribution to match the new distribution, 2) altering the probability thresholds used to determine the class label, or 3) altering the ratio of misclassification costs between false positive and false negative predictions. We employ the third method because the learning package that we use, WEKA, supports cost-sensitive learning and thus no changes were required to the learning algorithm. We use equation 1 to determine the cost ratio (the ratio of a false positive to false negative prediction) that should be used when building the classifier:

$$\mathrm{COST}_{FP} : \mathrm{COST}_{FN} = (pr/nr) : (pr'/nr') \quad (1)$$

Returning to our example, the cost ratio $\mathrm{COST}_{FP} : \mathrm{COST}_{FN}$ would equal 36:1. This adjustment can informally be shown to be correct as follows. Without loss of generality, imagine that we build a decision tree classifier and at an arbitrary leaf node there are P positive and N negative examples. Without cost-sensitive learning, the leaf will be labeled with the majority class. A cost-sensitive learner will classify the leaf to minimize the total cost and in this case the costs will be perfectly balanced if P=36N, since a positive label yields a cost of $36 \times \mathrm{COST}_{FN}$ and a negative prediction yields a cost of $1 \times \mathrm{COST}_{FP}$. If the positive rate is above this it will be labeled positive and if below this it will be labeled negative. This is the desired behavior since the new distribution will cause the ratio of positive to negative examples, as noted earlier, to decrease by a factor of 36 (i.e., a leaf with a positive to negative class ratio of 36:1 using ORIG_{label} corresponds to a class ratio of 1:1 when using NEW_{eval} and the 1:1 ratio is the normal threshold for labeling a classification "rule").

2.2 Class Distribution Estimation Methods

We introduce three class distribution estimation (CDE) methods in this section. Since quantification is the task of estimating the class distribution of new data, these CDE methods can also be considered quantification-based methods. The key difference between the quantification task and our task is that for quantification the ultimate goal is to estimate the prevalence of each class, whereas in our case this is only an intermediate step—the ultimate goal is to improve classification performance on data drawn from a new distribution. For all CDE-based methods the final classifier is induced from ORIG_{label} with the cost ratio computed with equation 1, utilizing the estimate of NEW_{CD} produced by the specific CDE method (note NEW_{CD} determines the pr'/nr' ratio).

Our **CDE-Iterate** method iteratively generates estimates of NEW_{CD}. It first builds a classifier C_1 using ORIG_{label} and then uses C_1 to classify $\mathrm{NEW}_{unlabel}$. The initial estimate of NEW_{CD} is then calculated from these predictions. However, assuming $\mathrm{ORIG}_{CD} \neq \mathrm{NEW}_{CD}$, the original predictions will be biased and will tend to underestimate the change in class distribution (this bias is why we need the "adjustment" in the first place). To compensate for this, the process is repeated but in the second iteration the classifier C_2 is built using the cost ratio calculated using equation 1 with the estimate of NEW_{CD} from the first iteration. The expectation is that additional iterations will reduce the undesired bias and the subsequent classifiers will be better able to classify $\mathrm{NEW}_{unlabel}$, yielding more accurate estimates of NEW_{CD}. The iterations terminate once a pre-specified maximum number of iterations is exceeded (in this paper we only report results for the first 3 iterations). **CDE-Iterate-*n*** refers to the classifier produced by the *n*th iteration of this method.

To make the algorithm more concrete, we specify the **CDE-Iterate** algorithm in Figure 2 using pseudo-code. The algorithm begins by initializing the values for the cost ratio to the default values (line 1), builds an initial classifier C_1 from ORIG_{label} (line 2) and then calculates the *pr/pn* ratio for ORIG_{label} (line3), where the Pos() and Neg() functions return the number of positive and negative examples for the specified data sets (ORIG_{label}). The algorithm then iterates in lines 4-10 until the maximum number of iterations (maxIterations) is reached. In this loop this algorithm first uses the previous classifier C_i to classify $\mathrm{NEW}_{unlabel}$ (line 6). Then in lines 7-8 the distribution mismatch ratio (*dmr*) is computed and the cost ratio information is updated. In line 9 a new classifier C_{i+1} is generated using ORIG_{label} with the updated cost ratio. Finally, once the loop terminates the last classifier is returned (line 11).

```
CDE-Iterate ($ORIG_{label}$, $NEW_{unlabel}$)
1.  $COST_{FP} = COST_{FN} = 1$;
2.  $C_1$ = build_classifier($ORIG_{label}$, $COST_{FP}$, $COST_{FN}$);
3.  Pos2Neg = Pos($ORIG_{label}$) / Neg($ORIG_{label}$);
4.  for (i=1; i<maxIterations; i++)
5.  {
6.      $NEW_{label}$ = Classify($NEW_{unlabel}$, $C_i$);
7.      dmr = Pos2Neg : ( Pos($NEW_{label}$) / Neg($NEW_{label}$) );
8.      $COST_{FP} = COST_{FN} \cdot$ dmr;
9.      $C_{i+1}$ = build_classifier($ORIG_{label}$, $COST_{FP}$, $COST_{FN}$);
10. }
11. return $C_{i+1}$;
```

Figure 2. Pseudo-code for the CDE-Iterate Algorithm

The **CDE-AC** method relies on the Adjusted Count (AC) quantification technique [7] to estimate NEW_{CD}. The method uses equation 2 to produce its *adjusted* estimate of the positive rate of the new distribution, *pr**. Note that it still requires *pr′*, the unadjusted estimate of NEW_{CD}, which is calculated in the same manner as it was calculated in the first iteration of CDE-Iterate—a classifier is built from ORIG_{label}, used to classify $\mathrm{NEW}_{unlabel}$, and *pr′* is calculated from the predicted class labels. Furthermore, *tpr* and *fpr*, which are associated with the original distribution, are estimated by using 10-fold cross validation on ORIG_{label}. Equation 2 essentially compensates for the fact that *pr′* will underestimate changes in class distribution due to the undesired bias discussed earlier (see Forman [7] for further details and a derivation of equation 2). The estimate of NEW_{CD} can easily be calculated from the adjusted positive rate, pr*.

$$pr^* = (pr' - fpr) / (tpr - fpr) \quad (2)$$

Finally, in order to evaluate the effectiveness of our two CDE-based methods, we introduce the **CDE-Oracle** method, which obtains the correct value of NEW_{CD} (via an oracle) and then uses equation 1 to determine the appropriate cost ratio. The classifier is then induced from ORIG_{label} using this cost ratio. One would expect this method to be an upper bound on the performance of all CDE-based methods.

2.3 Semi-Supervised Learning Methods

In this section we discuss semi-supervised learning (SSL) methods [4], which induce a classifier using examples from $\mathrm{NEW}_{unlabel}$. Unlike the methods in the previous section, no explicit adjustment is made for differences in class distribution. Our first SSL-based method, **SSL-Naïve**, is quite straightforward. It builds a classifier *C* from ORIG_{label}, uses *C* to label $\mathrm{NEW}_{unlabel}$, and then uses the labeled version of $\mathrm{NEW}_{unlabel}$ to build a new classifier, *C′*. Note that this method does not *directly* use any of the labeled data from the original distribution when building the final classifier.

Our next SSL-based method is more sophisticated in that it induces a classifier using examples from both the original and new distributions. This method, **SSL-Self-Train**, uses a semi-supervised technique known as self-training [20]. As the case with the SSL-Naïve method, this method starts by building a model based on $\text{ORIG}_{\text{label}}$ and uses it to classify $\text{NEW}_{\text{unlabel}}$. However, this method then moves the examples from $\text{NEW}_{\text{unlabel}}$ that have the most confident predictions (i.e., above the median confidence level) into $\text{ORIG}_{\text{label}}$. In this case confidence is based on how close the class membership probability estimate is to 1.0. The above set of steps is repeated until either all of the examples in $\text{NEW}_{\text{unlabel}}$ have been merged with those in $\text{ORIG}_{\text{label}}$ or a maximum number of iterations have been executed. In our implementation a maximum of 4 iterations are executed.

2.4 Hybrid Method

We combine the class distribution estimation and semi-supervised self-training methods into a **Hybrid** method, where the goal is to exploit the power of the CDE technique but also include data from the new distribution when training the classifier. The hybrid method starts like the CDE-Iterate method: it builds a classifier *C* from $\text{ORIG}_{\text{label}}$, applies *C* to $\text{NEW}_{\text{unlabel}}$ to estimate NEW_{CD}, uses equation 2 to determine the appropriate cost ratio, and then regenerates the classifier from $\text{ORIG}_{\text{label}}$ using this cost information. It then applies the self-training technique—it uses the adjusted classifier to relabel $\text{NEW}_{\text{unlabel}}$ and then effectively "moves" the examples where the confidence of the predicted label is above the median value to $\text{ORIG}_{\text{label}}$. This method then determines the appropriate cost ratio (equation 2) to account for differences in the class distribution of $\text{ORIG}_{\text{label}}$ and NEW_{CD}. This calculation takes into account the fact that the class distribution of $\text{ORIG}_{\text{label}}$ changes as new examples are moved into it. This process is then repeated until all examples have been removed from $\text{NEW}_{\text{unlabel}}$ or a maximum of 4 iterations have been executed. Due to space limitations we do not report the results for each iteration, as was done for CDE-Iterate. In summary, this method is essentially the semi-supervised self-training method, but where we compensate for the difference between the class distribution of the training data and NEW_{CD}.

3. EXPERIMENT METHODOLOGY

This section describes our experimental setup. It describes the data sets that we use, the specific experiments that we run, the classifier induction algorithm we employ, and the metrics that we use to evaluate classifier performance. The methods that we evaluate were described in Section 2 and are not described here.

Table 2. Description of 5 UCI Data Sets

Dataset	Size	% Pos	Partition
Adult	48,842	23.9%	4,700
Covertype	581,012	48.8%	8,000
Letter-Vowel	20,000	19.4%	1,500
Magic Gamma	19,020	64.8%	2,600
Spambase	4,601	39.4%	700

Table 2 describes the five UCI data sets [3] that we use in this study. Any data sets that had more than two classes were converted into two-class data sets. The Covertype data set was converted to two classes by designating "2" as the positive class and all other values as the negative class, while the Letter-Vowel data set was converted by designating the vowels as the positive class and all other letters as the negative class. The positive class is the minority class for all data sets except for the Magic Gamma data set, because the documentation for that data set specifically states that the majority class is the class of interest. Table 2 shows the original data set size, the percentage of the examples that belong to the positive class, and the number of examples in each partition (explained shortly).

As described earlier, the problem we investigate requires three data sets: $\text{ORIG}_{\text{label}}$, $\text{NEW}_{\text{unlabel}}$ and NEW_{eval}. We generate each of these by splitting the original data set into three equal-sized partitions. For our experiments the positive rate of $\text{ORIG}_{\text{label}}$ is fixed at 50% while the positive rate of $\text{NEW}_{\text{unlabel}}$ and NEW_{eval} but will be varied between 1% and 99% in 1% increments. In order to generate the desired class distributions without duplicating any examples, the size of the partitions must be limited. We use the maximum possible partition size for each data set and these partition sizes are specified in the last column of Table 2 (the value displayed for Covertype is not the maximum possible value but was reduced due to the size of the data set and time constraints).

Table 3 shows the results of this partitioning process for the Adult data set. Each of the three partitions contains 4,700 examples, even as the positive rate (Pr) varies. The maximum number of positive or negative examples required is 11,656 (2,350 + 2 • 4,653), which for the positives examples occurs when Pr= 99% and for the negative examples when Pr=1%. Since the original data set contains 11,673 (23.9% of 48,482) positive and 37,168 negative examples, these partitions can be generated without duplicating examples. In this case 17 positive examples are not used because fractional examples cannot be used to generate the appropriate positive rates.

Table 3. Partitions for Adult Data Set

Splits	Pr	# Pos	# Neg	Tot
$\text{ORIG}_{\text{label}}$	50%	2,350	2,350	4,700
$\text{NEW}_{\text{unlabel}}$, NEW_{eval}	1%	47	4,653	4,700
	2%	94	4,606	4,700
	…	…	…	…
	98%	4,606	94	4,700
	99%	4,653	47	4,700

In our methodology the partition $\text{ORIG}_{\text{label}}$ is created first and does not vary as the 99 partitions for $\text{NEW}_{\text{unlabled}}$ and NEW_{eval} are generated. In order to produce more reliable results, the experiments, and thus the partitioning process, were repeated 10 times and the results presented in this paper are averages over those 10 runs (Covertype experiments were repeated only 4 times due to time constraints).

All experiments in this paper utilize the J48 [17] implementation of C4.5 [12] from WEKA release 3.5.8. Our CDE-based methods compensate for changes in class distribution using WEKA's cost-sensitive learning capabilities, which in WEKA are implemented using example reweighting. All of our methods described in Section 2 are implemented as wrapper-based learners and could easily be applied to other base learning methods. The classification performance of our methods is evaluated using both accuracy and F-measure. The F-measure is defined as the harmonic mean between precision and recall and is defined in equation 3.

$$\text{F-measure} = 2 \bullet \text{precision} \bullet \text{recall} / (\text{precision} + \text{recall}) \qquad (3)$$

We track the performance of F-measure because we are interested in what happens when a class distribution becomes highly skewed and accuracy is known to be an inappropriate evaluation measure in these cases [11]. One might expect AUC to be a more natural choice than F-measure given its current popularity in the data mining community—and in fact we did track AUC for all experiments. However, we do not report these results because ROC curves are, by design, not sensitive to changes in class distribution and hence are an inappropriate evaluation measure for this problem (the results were also uninteresting in that most methods performed similarly).

4. EXPERIMENT RESULTS

In this section we present and analyze our experimental results. We begin by analyzing the detailed results for one representative data set, Adult, and then analyze more highly summarized results for all five data sets. The detailed accuracy results for the Adult data set are shown in Figure 3 and the same data is shown at a slightly less granular level (i.e., only data from 13 of the 99 positive rates are shown) in Table 4.

Figure 3 shows that three of the methods, Naïve, SSL-Naïve, and SSL-Self-Train, perform much worse than the remaining methods that are displayed, including all CDE-based methods and the Hybrid method. Of the three methods that perform poorly, Naïve's performance is in the middle, with SSL-Naïve performing the worst for low positive rates and best at high positive rates. With the exception of CDE-Iterate-1, the remaining methods all perform about the same for high positive rates but vary at low positive rates. Here again CDE-Iterate-1 does the worst, while CDE-AC does the best and Hybrid is in the middle. Because of difficulties visually differentiating between the curves, the CDE-Iterate-2 and CDE-Iterate-3 methods are not shown, although their performance is better than CDE-Iterate-1 but worse than CDE-AC (their more highly summarized performance is provided in Table 5). The CDE-Oracle and Oracle methods are not shown because in the figure they were indistinguishable from the CDE-AC method.

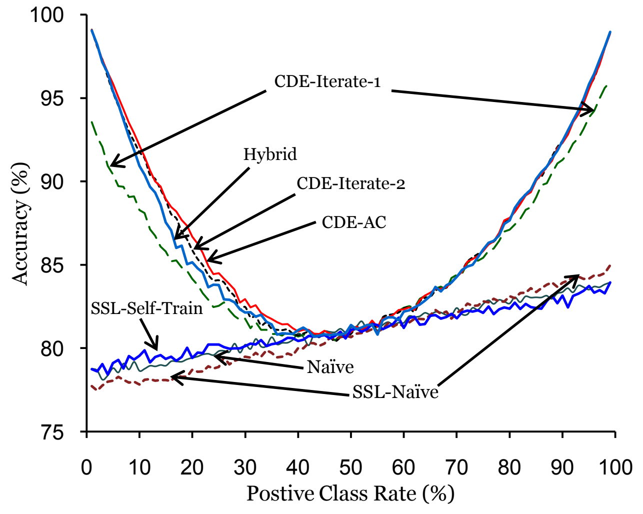

Figure 3. Accuracy Peformance for Adult Data Set

It is worth commenting on the shapes of the curves in Figure 3. Three of the curves are nearly linear and this includes the Naïve method, which is the easiest to analyze. Since the Naïve method ignores the new distribution in the learning phase, we might expect its performance to be linear and parallel to the x-axis (i.e., invariant with respect to positive class rate). However, the observed performance is not inconsistent with this, which simply means that the accuracy of the induced classifier is not the same for both classes even though the classes are equally represented in the training data. The performance curves for the other methods, including the Oracle method (not shown) exhibit a "U" shape with a minimum near a positive class rate of 50%. This is simply due to the fact that it is easiest to achieve high accuracy when a data set is highly skewed—and the strategies that exhibit the "U" shape can adapt to the skewed distribution.

The overall performance characteristics of the methods are shown more effectively in Table 4, which includes the results for all 10 methods. Of particular value are the averages for the methods over the different positive rates, shown in the last row (these averages are computed over all 99 positive class rates, not just the 13 that are displayed). The Oracle and Naïve methods determine the range of expected behavior, while the CDE-Oracle provides what should be an upper bound on the performance of the CDE-based methods. First, note that the CDE-Oracle provides performance very close to that of the overall Oracle method. Based on the averages we see that the CDE-Iterate methods get progressively better with additional iterations and that the CDE-AC method is the best overall performing CDE-based method. As we saw in Figure 3 the SSL-based methods perform poorly and in fact, based on average performance, perform worse than the Naïve method. The Hybrid method performs in the middle range of the CDE-Based methods and thus shows promise if it can be improved. The overall results suggest that CDE-AC is a very good method and looking at each individual row, we see that not only does it have the best average performance, but it performs best or nearly best for each positive rate. Table 5 will present a more summarized view of the data in Table 4, but for all five data sets.

Table 4. Detailed Accuracy Results for Adult

	Baseline		CDE					SSL		Hybrid
Pr	Oracle	Naive	Oracle	Iter-1	Iter-2	Iter-3	AC	Naive	Self-Tr.	
1	99.07	78.78	99.08	93.56	99.10	99.10	99.08	77.73	78.74	99.04
5	95.93	78.81	95.87	90.42	95.34	95.93	95.88	78.04	79.11	95.41
10	92.20	78.64	92.13	88.28	91.77	92.22	92.16	77.82	79.55	90.93
20	86.95	79.42	86.63	84.15	85.88	86.68	86.69	78.69	79.41	85.14
30	83.31	80.21	83.09	81.78	82.47	82.73	82.94	79.49	80.12	82.14
40	81.42	80.88	81.47	80.70	80.77	80.80	81.01	80.00	80.44	81.09
50	81.27	81.24	81.24	81.01	80.86	80.79	80.96	80.65	81.10	81.40
60	82.12	81.52	82.05	82.43	82.39	82.38	82.33	81.29	81.21	82.20
70	84.60	82.41	84.63	84.40	84.29	84.20	84.29	82.12	81.80	84.29
80	87.99	82.68	87.86	87.11	87.81	87.82	87.81	83.12	82.37	87.66
90	92.65	83.36	92.71	91.29	92.45	91.89	92.30	83.92	83.16	92.42
95	95.83	83.51	95.77	93.85	95.64	95.56	95.54	84.17	83.74	95.84
99	98.98	83.93	99.00	95.97	98.96	99.00	98.87	84.97	83.95	98.96
Ave	**86.83**	**81.13**	**86.84**	**85.38**	**86.58**	**86.62**	**86.75**	**80.86**	**81.09**	**86.42**

The F-measure results for the Adult data set are displayed in Figure 4. Because the curves are even harder to distinguish than for accuracy, the figure was simplified slightly—the *x*-values are shown at 5% increments, curves that were essentially indistinguishable were labeled together using one of the curves, some of the 10 methods were omitted, and the positive rate is clipped at 70% because the relative performance of the methods does not vary after that point.

The results for F-measure are interesting in that they vary greatly from those for accuracy—and the Oracle does not perform best. As Table 6 will show us, this behavior is relatively consistent over all data sets, so it is worth further analysis. First, Figure 4 shows that the same three methods that did poorly for accuracy for both high and low positive class rates (Naïve, SSL-Self-Train, and SSL-Naïve) exhibit the same behavior for F-measure. All other methods perform similar to one another for high positive class rates. But at low positive class rates CDE-Iterate-2 consistently does the best. The Oracle and CDE-AC methods perform nearly identically to one another, and the Hybrid method sometimes does better and sometimes worse than Oracle and CDE-AC. We defer the discussion of this interesting behavior until after we introduce the F-measure results for the five data sets in Table 6.

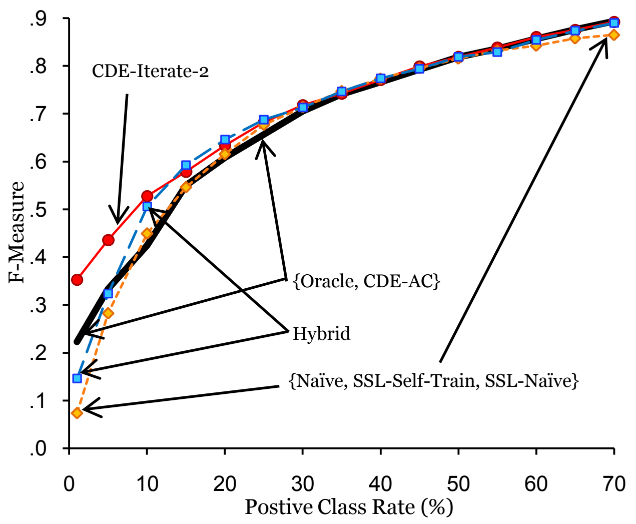

Figure 4. F-Measure Performance for Adult Data Set

We now present the more summarized results for all five data sets. Table 5 provides these results for accuracy, which is similar to Table 4 but averages the results over all 99 positive rates. The methods are sorted in order of decreasing average accuracy, so the best methods are toward the top. When determining which method is the best practical method, the Oracle and CDE-Oracle methods are excluded, since in practice oracles are not available. The best performing practical method for each individual data set is underlined and from this we see that CDE-AC is not just the best when averaged over all five data sets, it performs best on each individual data set. The overall performance of the other methods is roughly consistent with what we saw for Adult. Again, we see that the CDE-Iterate method benefits from additional iterations and the Hybrid method shows some promise. We again see that the SSL-based methods perform poorly—and worse than the Naïve method which essentially ignores the changing distribution completely.

Table 5. Summary Accuracy Results

Method	Adult	Cover Type	Letter Vowel	Magic Gamma	Spam Base	Ave
Oracle	86.83	85.30	89.12	85.75	91.79	88.10
CDE-Oracle	86.84	85.02	88.98	85.94	91.69	88.06
CDE-AC	<u>86.75</u>	<u>85.06</u>	<u>88.68</u>	<u>85.65</u>	<u>91.82</u>	<u>87.94</u>
CDE-iterate-3	86.62	84.91	<u>88.68</u>	85.32	91.74	87.80
Hybrid	86.42	84.68	88.24	85.35	91.58	87.61
CDE-iterate-2	86.58	84.32	88.40	84.80	91.72	87.55
CDE-iterate-1	85.38	82.82	87.61	83.47	91.47	86.60
Naive	81.13	78.83	85.22	80.29	88.76	83.40
SSL-Self-Train	81.09	79.05	85.24	79.99	88.55	83.29
SSL-Naive	80.86	77.67	82.93	79.54	87.96	82.57

Table 6 shows the summary performance of the ten methods over the five data sets for F-measure. The methods are again ordered by decreasing average performance, and since the values in the Adult column are decreasing, as they were in Table 5, the relative efficacy of each method for the Adult data set matches the pattern over all data sets. The key conclusions here are that the SSL-based methods do poorly and that the Oracle and CDE-AC methods are in the middle of the performance range and the CDE-Iterate methods outperform these two methods.

Table 6. Summary F-Measure Results

Method	Adult	Cover Type	Letter Vowel	Magic Gamma	Spam Base	Ave
CDE-iterate-2	<u>.777</u>	<u>.748</u>	<u>.819</u>	<u>.761</u>	<u>.861</u>	<u>.799</u>
CDE-iterate-1	.773	.742	.811	.755	.859	.794
CDE-iterate-3	.752	.738	.810	.749	.858	.787
Hybrid	.769	.725	.803	.747	.843	.785
Oracle	.760	.730	.812	.737	.850	.784
CDE-Oracle	.760	.723	.812	.736	.854	.784
CDE-AC	.755	.716	.807	.742	.845	.781
Naive	.742	.718	.784	.736	.825	.767
SSL-Self-Train	.741	.720	.785	.731	.824	.766
SSL-Naive	.738	.704	.749	.728	.814	.755

While not all of our results could necessarily be predicted a priori, most of our results are not surprising and can be explained. For example, for accuracy it makes some sense that the CDE-Iterate method performs better after one iteration because of the undesirable bias mentioned in Section 2. CDE-based methods outperform SSL-based methods because, as we shall discuss in Section 5, class distribution estimation (i.e., quantification) is fundamentally an easier task than classification and thus the CDE-based methods introduce less uncertainty than the SSL-based methods, which require us to classify examples from the new distribution before a classifier can be retrained.

The F-measure results that have the CDE-Iterate methods outperforming the CDE-Oracle and Oracle methods are harder to justify, but we feel we have a plausible explanation. Our explanation begins with the Oracle method, which uses the correct (hidden) labels from the new distribution to train a classifier. While this seems like the perfect strategy, most classification methods are optimized for accuracy maximization and often sacrifice performance of the minority class for improved performance of the majority class—which will likely degrade the F-measure performance, due to the need to achieve high recall values. Thus we can see that the Oracle method may produce poor F-measure values when trained on data with low positive class rates. Because the CDE-Iterate method will train a classifier using data with a positive rate of 50%, it has the opportunity to achieve better F-measure results. The cost ratio used to adjust the classifier can undermine this by placing less emphasis on the positive class when the new distribution has a lower estimated positive rate, but, as discussed in Section 2, the CDE-Iterate method will tend to underestimate any changes in class distribution, especially in the initial iteration. This explains why CDE-Iterate-1 performs well.

This above explanation may explain the observed results, but is there any useful lesson here? There is a lesson, but it not a new one. If one wants to perform better on one class than another, it is appropriate to bias the learner toward that class and if one wants to do well on a measure that balances the performance of both classes, then one should not bias the learner toward one class (e.g., by training on data mainly from one class). Thus, we should take this into account when "adjusting" a classifier to compensate for changes in class distribution. This suggests an extension to our work—how to adjust for a changing class distribution when either the classes are not equally important or are equally important. Interestingly enough, there has been work on cost quantification [7] and these methods would be appropriate to counteract changes to the class distribution when we have knowledge about the relative importance of the different classes.

5. RELATED WORK

In this section we describe some related work, although much of the most relevant work was already mentioned in Section 2. The problem of improving classifier performance in response to unknown changes in class distribution has been studied previously [1, 9, 13] and the most relevant and successful approach thus far has involved expectation maximization (EM) [9, 13] and this approach has also been adapted to the related problem of classifying non-stationary data sequences [18]. Our CDE-Iterate method is a variant of the basic EM method, but there are some minor differences. Namely, the CDE-Iterate method thresholds the generated probability estimates to assign a class label and from this generates the class distribution estimate, while EM uses the probability estimates directly, without applying any threshold. In addition, CDE-Iterative "adjusts" the original model using cost-sensitive learning while the prior work using EM adjusted the original model's posterior probability outputs. Based on our results and the published results in prior work, we believe that both the EM and CDE-Iterate methods are similar and perform similarly (however, as we discuss in Section 6, we view CDE-AC as superior to both methods).

In some situations one may have no information (e.g., unlabeled examples) about future changes in class distribution but still wants to maximize classifier performance on future data [19]. The approach in this case is not to adapt to changing conditions, but rather to build a "robust" classifier that performs well under a wide variety of situations [2]**.** In fact, this desire for classifiers that exhibit robust behavior over wide ranges of class distributions and misclassification costs is the primary reason that ROC analysis [10] has gained such prominence in the data mining community. While this approach of generating robust classifiers has its advantages and is applicable to our problem, it clearly is not the best strategy when the class distribution of the new data is known or can be reliably estimated—and as we have seen in this paper, we can reliably estimate the new class distribution when unlabeled data from the new distribution is available.

While our work focused on improving classifier performance in response to changes in the class distribution of the data, a related, but different, task is the quantification task. Quantification involves estimating the prevalence (i.e., class distribution) of the classes over time [7]. Quantification is a *simpler* task than classification because one can often come up with good estimates of a class distribution without producing accurate predictions for individual examples. As an example, auto insurance agencies can accurately predict what fraction of their policy holders will have an accident without being able to accurately predict which specific policy holders will have an accident. As we have seen this paper, quantification methods can also help solve our more complex task (i.e., our CDE methods perform quantification then model adjustment). Quantification techniques are discussed in detail by Forman [7, 8] and while we applied some of those methods to our problem (i.e., Adjusted Count) there are other relevant methods that could be analyzed in the future (e.g., Median Sweep, Mixture Model).

Our CDE-based methods compensate for changes in the class distribution via cost-sensitive learning and the EM accomplishes the same thing by changing the probability thresholds. A third way of compensating for differing class distributions is to sample from the original distribution so that its class distribution matches the estimated class distribution of the new data. Many appropriate sampling methods exist [15], such as oversampling and undersampling, as well as more advanced methods which make better use of the data [5].

Since our learning task involves labeled and unlabeled data, semi-supervised learning methods [4, 20] are relevant. The results thus far for the SSL-based methods have not been very promising, probably due to the fact that, as just discussed, it is easier to accurately estimate the class distribution of the unlabeled examples than to classify them. We believe, however, that the CDE-based methods can ultimately benefit from semi-supervised learning and thus we feel that semi-supervised learning warrants further study.

6. CONCLUSION AND FUTURE WORK

In this paper we evaluated several methods for dealing with the situation where the class distribution can change after an initial classifier is built and only unlabeled data from the new distribution is available. Our results clearly indicate that the naïve approach of *not doing anything* leads to very poor results but that there are very effective methods for dealing with this problem. The class distribution estimation based methods generally performed the best and when accuracy must be maximized the CDE-AC method is the best choice based on its performance, computational requirements (i.e., only a single iteration) and the fact that there are no parameters, such as the number of iterations, to set. The CDE-based methods performed relatively close to the ORACLE method, especially in comparison to the Naïve method, which performed poorly. In general we found that the CDE-Iterate-2 method outperforms the CDE-Iterate-1 method, indicating that our iterative process does lead to improved class distribution estimates. Relative to the CDE-based methods, the semi-supervised learning methods did poorly. The hybrid method did much better than the SSL-based methods, but consistently underperformed the CDE-based methods. The results for F-measure were quite different and the reasons for this were discussed in Section 4. However, the results were still quite clear, with the CDE-Iterate-2 method performing best, and all of the CDE-based methods were effective, significantly outperforming the naïve strategy.

The relatively poor performance of the SSL-based methods, for both accuracy and F-measure, is quite notable. This failure was explained earlier by the fact that class distribution estimation is fundamentally an easier task than classification and hence our class distribution estimates are thus going to be more accurate than the estimates of the class values for the new, unlabeled, data. A second insight, however, is related to the task itself. Semi-supervised learning is typically employed when labeled training data is very scarce but unlabeled data abounds. That is not the case in our problem setting or in our experimental setup and was not the motivation for the use of semi-supervised learning. In our setting we assume that there is sufficient labeled data to build a reasonable classifier. The problem is that the class distribution changes and that is the motivation for the use of semi-supervised learning (i.e., we want to learn from the more representative, but unlabeled, examples). Given this understanding and the fact that CDE is easier than classification, the disparity in the results is explained. Nonetheless, in theory the use of the new data for training could improve overall classifier performance and we do believe that SSL methods can be of use in this context. We discuss improvements to the SSL methods in our discussion of future work toward the end of this section.

This paper provides a number of contributions. First, we evaluate methods that have not previously been used to address the changing class distribution problem—and these new methods—especially CDE-AC—are shown to work well. Furthermore, the CDE-AC method that we recommend is simpler and easier to implement than iterative methods. The failure of the SSL-based methods is also notable and should help guide future research. We also provide a more comprehensive empirical study of the problem than past work: we analyze a total of ten methods, including several baseline methods for comparison, evaluate our results with respect to F-measure in addition to accuracy, combine multiple approaches (i.e., Hybrid), and evaluate our methods under a large number (99) and range (1% - 99%) of class distributions. Finally, the problem we address does occur in many realistic settings and given the current state-of-the-art of data mining tools, it is feasible for practitioners to implement our solution (i.e., most tools provide one of the three methods that we discuss for adjusting a classifier). Finally, we also hope that our work will bring deserved attention to the problem of changing class distributions and the fact that good solutions do exist.

Although our results for the CDE-based methods are quite encouraging, there is certainly some room for improvement, since even a 1% difference in accuracy between our method and the ORACLE method is significant. One improvement would be to adapt the CDE-Iterative method to automatically terminate once the class distribution estimate converges. This might improve overall performance over any specific CDE-Iterate-*n* method and would eliminate the problem of identifying the appropriate number of iterations. It is possible that such a CDE-converge method would outperform CDE-AC. We have performed some preliminary research in this area and have found that in many cases convergence does occur, although we need to refine our notion of convergence (e.g., to handle small cyclical fluctuations in the estimates). As discussed, semi-supervised learning should enable us to improve our CDE results further by making additional data available for classifier induction, although the clear superiority of the CDE-based methods over the SSL-based methods indicate that it may not be easy to develop an effective hybrid approach. One possibility we are investigating is to use a classifier that accepts class membership probabilities in the training phase, so that the uncertain predictions generated by semi-supervised learning can be factored into the learning process.

We also would like to apply our current methods, and some new variants of these methods, to more complex problem settings. One such problem setting differs only slightly, where $\text{NEW}_{\text{unlabel}}$ and NEW_{eval} are replaced by a single data set that serves both purposes. In this new scenario the goal is to exploit new data to build a new classifier and then classify that *same* new data, rather than our current setting where additional data drawn from the new distribution is classified. This new setting would be an example of transductive learning. A more ambitious setting that we intend to study is where the concept drift is of a much less restrictive form, such that the actual concept can change over time, rather than just its class distribution. The CDE-based methods may still prove useful in this more challenging setting—since the class distribution will most likely change if the concept changes—but the CDE-based methods clearly will not be sufficient by themselves. In this more complex setting semi-supervised learning will have more to offer and hybrid methods may then perform best.

7. REFERENCES

[1] Alaiz-Rodriguez, R., Guerrero-Curieses, A. and Cid-Sueiro, J. Minimax Regret Classifier for Imprecise Class Distributions. *Journal of Machine Learning Research, 8* (2007), 103-130.

[2] Alaiz-Rodriguez, R. and Japkowicz, N. Assessing the Impact of Changing Environments on Classifier Performance. In the *Proceedings of the 21st Canadian Conference in Artificial Intelligence* (2008), 13-24.

[3] Asuncion, A. and Newman, D. J. UCI Machine Learning Repository. http://www.ics.uci.edu/~mlearn/ MLRepository.html (2007).

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

[5] Chawla, N. V., Bowyer, K. W. and Kegelmeyer, W. P. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research, 16* (2002), 321-357.

[6] Elkan, C. The foundations of cost-sensitive learning. In the *Proceedings of the 17th International Joint Conference on Artificial Intelligence* (2001), 973-978.

[7] Forman, G. Quantifying counts and costs via classification. *Data Mining Knowledge Discovery, 17*, 2 (2008), 164-206.

[8] Forman, G. Counting Positives Accurately Despite Inaccurate Classification. In *ECML* (2005), 564-575.

[9] Latinne, P., Saerens, M. and Decaestecker, C. Adjusting the Outputs of a Classifier to New a Priori Probabilities May Significantly Improve Classification Accuracy: Evidence from a multi-class problem in remote sensing. In the *Proceedings of the 18th International Conference on Machine Learning* (2001), 298-305.

[10] Provost, F. and Fawcett, T. Robust Classification for Imprecise Environments. *Machine Learning, 42*, 3 (2001), 203-231.

[11] Provost, F., Fawcett, T. and Kohavi, R. The Case against Accuracy Estimation for Comparing Induction Algorithms. In the *Proceedings of the 15th International Conference on Machine Learning* (1998), 445-453.

[12] Quinlan, R. J. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[13] Saerens, M., Latinne, P. and Decaestecker, C. Adjusting the outputs of a classifier to new a priori probabilities: A simple procedure. *Neural Computing, 14* (2002), 14-21.

[14] Tsymbal, A. The problem of concept drift: Definitions and related work. Computer Science Department, Trinity College Dublin, 2004.

[15] Weiss, G. M. Mining with rarity: a unifying framework. *SIGKDD Explorations Newsletter, 6*, 1 (2004), 7-19.

[16] Weiss, G. M. and Provost, F. Learning when training data are costly: The effect of class distribution on tree induction. *Journal of Artificial Intelligence Research, 19* (2003), 315-354.

[17] Witten, I. H. and Frank, E. *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd ed. Morgan Kaufmann, June 2005.

[18] Yang, C. and Zhou, J. Non-stationary data sequence classification using online class priors estimation. *Pattern Recognition, 41*, 8 (2008), 2656-2664.

[19] Zadrozny, B. and Elkan, C. Learning and making decisions when costs and probabilities are both unknown. In the *Proceedings of the 7th International Conference on Knowledge Discovery and Data Mining* (2001), 204-213.

[20] Zhu, X. *Semi-Supervised Learning Literature Survey*. Computer Science Department, University of Wisconsin-Madison, 2005.