Online Cost-Sensitive Learning for Efficient Interactive Classification

Mohit Kumar MOHIT.X.KUMAR@ACCENTURE.COM
Rayid Ghani RAYID.GHANI@ACCENTURE.COM
Accenture Technology Labs, 161 N Clark St, Chicago, IL 60601 USA

Abstract

A lot of practical machine learning applications deal with interactive classification problems where trained classifiers are used to help humans find *positive* examples that are of interest to them. Typically, these classifiers label a large number of test examples and present the humans with a ranked list to review. The humans involved in this process are often expensive domain experts with limited time. We present an online cost-sensitive learning approach (more-like-this) that focuses on reducing the time it takes for the experts to review and label examples in interactive machine learning systems. We target the scenario where a batch classifier has been trained for a given classification task and optimize the interaction between the classifier and the domain experts who are consuming the results of this classifier. The goal is to make these experts more efficient and effective in performing their task as well as eventually improving the classifier over time. We validate our approach by applying it to the problem of detecting errors in health insurance claims and show significant reduction in labeling time while increasing the overall performance of the system.

1. Introduction

A lot of practical machine learning applications deal with interactive classification scenarios where trained classifiers are used to help humans find *positive* examples that are of interest to them. Typically, these classifiers label a large number of test examples and present the humans with a ranked list to review, verify, and correct (if the classification is incorrect). These humans are often expensive domain experts with limited time and attention. Fraud Detection, Intrusion Detection, Medical Diagnosis, Information Filtering, and Video Surveillance are some examples where these systems are currently being used to aid humans in finding examples of interest. All of these applications involve a limited number of labeled examples with a high cost of labeling, and a large number (millions) of unlabeled examples, with majority of them being negative (skewed class distribution). The goal in these applications is to maximize the efficiency of these domain experts in finding positive examples by providing them with a high proportion of positive examples and helping them verify the classification efficiently.

The motivation for this paper came from the problem of reducing payment errors when processing health insurance claims. The goal is to minimize processing errors by detecting claims that are likely to have errors, and presenting them to human auditors so that they can be corrected before being finalized. The typical process for insured healthcare in the US is that a patient goes to a service provider (medical facility) for the necessary care and the provider files a claim with the patient's health insurance company for the services provided. The insurance company then pays the service provider based on multiple complex factors including eligibility of the patient at time of service, coverage of the procedures in the benefits, contract status with the provider etc.

Payment errors made by insurance companies while processing claims often result in re-processing of the claim. This extra administrative work to re-process claims is known as *rework* and accounts for a significant portion of the administrative costs and service issues of health plans. These errors have a direct monetary impact in terms of the insurance company paying more or less than what it should have. (Anand & Khots, 2008) estimates from a large insurance plan covering 6 million members had $400 million in identified overpayments. In our discussions with major insurance companies, we have found that these errors

Budgeted Learning Workshop at the 27th International Conference on Machine Learning, Haifa, Israel, 2010.

result in loss of revenue of up to $1 billion each year. In addition to the direct monetary impact, there is also an indirect monetary impact since employees need to be hired to *rework* the claim and answer service calls regarding them. According to estimates by an Accenture study, 33% of the administrative workforce is directly or indirectly related to rework processing. As the health insurance costs across the world have increased alarmingly in recent years, insurance companies are being forced to reduce their administrative costs and decrease the insurance premiums. These statistics make the problem of rework prevention extremely important and valuable to the healthcare industry and motivated the work described in this paper.

In the machine learning formulation, we have an (interactive) classification problem with some labeled data available, a large number of unlabeled examples, and human auditors available (at a cost) to look at the classifications performed by the classifier and verify or relabel them. The goal of these auditors is to find as many positive examples as they can in the limited time they have. The machine learning system faces the dilemma of either providing them examples that are very likely to be positive (and thus helping them with their task) or asking them to label examples that will improve future performance (accuracy) of the overall system. Since the user's time is limited, the system needs to explicitly manage this exploration-exploitation tradeoff. Active learning algorithms aim to improve future classifier performance by minimizing the number of labels that need to be acquired from experts, but often at the expense of immediate rewards for the task the humans, used as labelers, are trying to accomplish. The number of examples that need to be labeled is just one aspect when considering the cost of running a machine learning system. Another dimension that is not often discussed is reducing the time it takes to label an example, which is equally important when dealing with interactive business applications where the experts are expensive and have limited time. In this paper, we describe an online cost-sensitive learning framework that focuses on reducing the overall time experts spend on labeling (verifying or correcting) examples provided to them by a classifier. We show that our online cost-sensitive learning strategies result in both improving the accuracy of the baseline system and reducing the time it takes for the auditors to review and label the examples needed to reach that performance.

We do not describe the details of the overall system in this paper which can be found in (Kumar et al., 2010). The Rework Prevention Tool that is the basis of our work in this paper has been developed in conjunction with industry experts from Accenture's Claims Administration group who currently work with most of the large insurance companies in the US. We have applied this system to two large US health insurance companies and results so far show potential savings of over $15-25 million each year for a typical insurer which would have a large effect on the healthcare costs as well as help make the healthcare process smoother.

2. Related Work

There has been work in the areas of active learning, reinforcement learning (exploration exploitation tradeoffs), and contextual bandits that is related to cost-sensitive interactive classification. In addition, there have been exciting advances in the research area of cost-sensitive active learning (Settles et al., 2008), (Haertel et al., 2008), (Margineantu, 2005) where the aim is to minimize the overall cost of training an accurate model, recognizing that the annotation costs are not constant across instances, and can vary considerably. A related area is active feature acquisition where the goal is to select the most informative features to obtain during training (Saar-Tsechansky et al., 2009). Budgeted learning (Guha & Munagala, 2007), (Kapoor & Greiner, 2005) where the goal is to learn the most accurate 'active classifier' based on a fixed learning budget for acquiring training data which is also related to the idea of Exploitation/Exploration from Reinforcement Learning. (Settles, 2009) provides an excellent survey of the latest advancements in the field of Active learning.

3. Background

We formulate the problem of rework prediction as an interactive classification problem and produce a ranked list of claims that need to be manually reviewed. Figure 1 gives a generalized view of the claim processing pipeline. Claims are created by service providers and submitted to the insurance company. They go through automatic validation checks and then the appropriate pricing is applied to the claims using benefit rules and contracts. This takes place automatically in some cases and in other cases, manual intervention is required. Once the pricing is applied, claims get finalized and payment is sent to the service provider. The system we describe in this paper is the module placed after the pricing is applied to detect potential issues with the claim before it is finalized so it can be corrected before payment is sent.

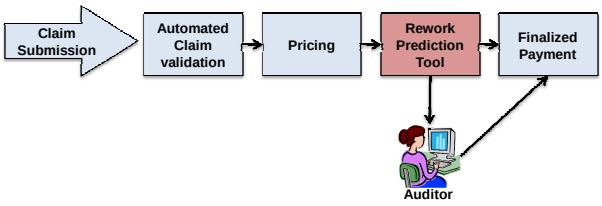

Figure 1. Claim Processing Pipeline

3.1. System Overview

We worked with domain experts to come up with requirements that would make a rework prediction solution practical and useful for insurance companies. Our system consists of the following components: Data Collection, Feature Construction, Model Learning and Selection, Item Scoring, Explanation Generation, User Feedback, and User Interface. We only give a brief description of our baseline system here since details are given in (Kumar et al., 2010).

We obtain data from health insurance companies along with the labels: *rework* or *correct*. The claims assigned the label *rework* are those that were manually examined in the past and contained errors. Those assigned the label *correct* are ones that were manually examined in the past and found to be correct. Once the data is collected and labeled, feature construction is the next step that takes place. There are four classes of information in each claim: Member information, Provider information, Claim Header, and Claim Line Details. Our features are extracted from these classes of information. Overall, we end up with ∼15,000 categorical and numerical features to build our models.

We experimented with SVMs as the main learning algorithms. We used SVMperf (Joachims, 2006) for SVMs because of efficiency reasons. Since we have more than 238,000 features after creating binary features from the categorical data, we experimented with frequency-based feature selection techniques which made our system more efficient both in terms of execution time and storage requirements.

Accurately identifying rework and presenting it to auditors is one aspect of our system. Another equally important aspect is to give the auditors the ability to quickly determine if the claim being presented to them is rework or not. Currently, auditors in most companies are given a claim without any hints about why a claim could be rework. One of our goals is to *explain* the predictions of the classifier to the auditor to reduce the time it takes them to determine the label of the claim. In our case, we use the feature weights learned by the SVM to generate the explanations. (Mladenic & Brank, 2004) have also used a similar approach to perform feature selection by retaining features with high weights.

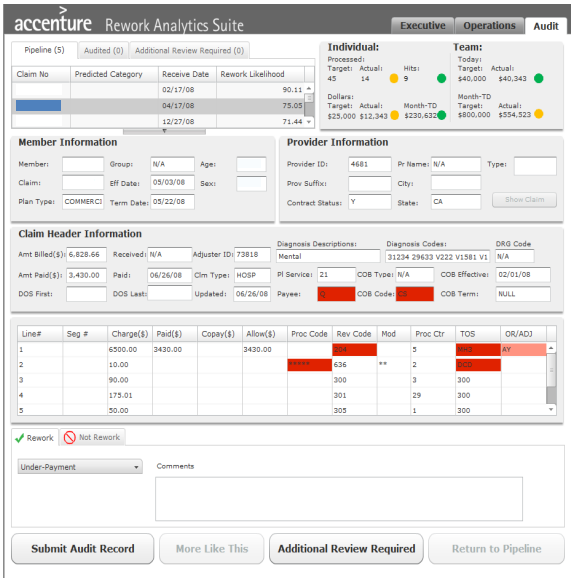

Figure 2. User Interface with the Auditor view (sensitive PHI fields are blanked out).

We also get feedback from auditors to improve future predictions. We obtain claim level feedback (labels of examples) as well as feature-level feedback using the user interface we have developed. We worked with domain experts and designed an interface that was similar to what auditors typically use (a claim form) when auditing the claim but enhanced it with some new features. Figure 2 shows a screenshot of the user interface for the auditors. This interface provides the auditors with a list of claims in their work queue with a rework likelihood score assigned to each claim that is generated by the classifier. In addition, we use the influence scores described earlier to highlight the data fields. The intensity of the highlight color is proportional to the influence score assigned by the classifier. This feature has been designed to focus the attention of the auditor to the influential data fields and help them audit the claim faster. We also allow the auditors to click on any field to *unhighlight* it (if it's highlighted) or to highlight it and give us feedback that field is influential (or not).

4. Online Cost-Sensitive Learning

When we deployed the system described in the previous section and tested it with auditors from a large US health insurer, we got encouraging results but also noticed several problems that affected the efficiency of the auditors when interacting with our rework prediction system.

Context Switching Costs: We used the confidence of the SVM classifier to rank the test examples. The auditors were then provided with the ranked list of claims and started from the top of that list and worked their way down. One problem they experienced was high switching costs when moving from example to example. This switching cost arose from the fact that the auditors spent considerable time (up to 20 minutes) reviewing a claim to figure out whether it was a positive or a negative example. Once they had investigated (and labeled) that claim, they were given the next claim (from the ranked list using the SVM confidence scores) that was often completely different (in terms of the *reason* for flagging) from the previous one. At a later stage, they would be given a claim that was lower down in the ranked list but was scored highly by the classifier for *similar* reasons as an earlier claim. Unfortunately, by that time, they have already gone through enough different claims that they need to redo the investigation and the time it takes them to label it is the same as the earlier ones. We hypothesize that if we can group similar claims together, it would reduce the *context switching* and help make the reviewing times shorter.

Low Precision when Reviewing Top 2% Scored Claims: Based on the class distribution, claim volumes, auditor workload, and audit times, we determined that it's optimal to manually review top 2% of the claims that are scored by our classifier, making precision at 2% the metric we optimize[1]. The scores assigned by the SVM to the top few percent of the claims are fairly similar which results in poor ranking at the top of this list. Also, based on the feedback we got from the auditors, we found that as they went down the list reviewing the claims, they encountered several series of consecutive claims that were all negative examples (and often for similar reasons). We hypothesize that this was because the ranking was purely based on SVM scores and did not have any notion of diversity. Again, grouping similar claims together could help group a lot of these negative examples together and improve the performance of the human auditors.

Intuitive Explanations: The interface we designed for the auditors highlight data fields in the claim form that had high scores using the hyperplane that was learned by the SVM. Even though the weights learned by the SVM result in a classifier that provides high classification accuracy, they were not necessarily intuitive as an explanation for the auditors.

Batch Retraining/Learning: Because we are dealing with a system that is interactive with costly human auditors in the loop, we do not have the time to retrain the system (in real-time) after each claim is reviewed by the auditors. The long wait time that is required for retraining makes the process too expensive since the auditors have to be idle during that time. This situation forces us to deal with the tradeoff between potentially better results (by retraining after every new label) and maximizing the use of the auditor time (by retraining after a certain number of new labels have been acquired).

[1]More details on that are in (Kumar et al., 2010)

4.1. Our Approach for Online Cost-Sensitive Learning

The four issues we described above form the basis of our strategy that uses online cost-sensitive learning in order to maximize the efficiency of the auditors while helping them achieve their goal of finding positive examples. We deal with the scenario where a batch classifier has been trained for a given classification task and focus on optimizing the interaction between the classifier and the domain experts (auditors in our case) who are consuming the results of this classifier. The goal is to make these experts more efficient and effective in performing their task as well as eventually improving the classifier over time.

4.1.1. More-Like-This Strategy

Our *More-Like-This* (MLT) strategy (Algorithm 1) is motivated by the context switching costs, low precision, and explanation issues described above. The goal of this strategy is to group the scored claims into clusters that contain claims that have been scored highly for *similar* underlying reasons. This allows us to provide the auditors with a cluster of claims that are both highly likely to be positive and can also be audited very quickly with small incremental audit costs. The cost gets reduced because once the auditors have invested the effort in reviewing the first couple of claims in a cluster, the rest of the claims will have similar labels due to similar reasons. At the same time, this technique also allows us to disregard a large number of claims that are similar to each other and are in fact negative examples but have been incorrectly ranked highly by the classifier. This is done by our online strategy where we rank the clusters using a variety of metrics (described later) and give the auditors a few claims from a cluster. If the claims are verified by the auditors as being positive, we continue giving them claims from that cluster. Otherwise, we move on to the next cluster thus avoiding a large number of highly scored claims that were in fact negative. This strategy also implicitly provides the auditors with an *explanation* for the classification by providing them with

other claims that are similar to what they have just labeled. MLT works as a post-processing step on the test (unlabeled) examples that have already been classified (scored) by the trained classifier.

Algorithm 1: Online More-Like-This Algorithm

Require: a labeled set L and an unlabeled set U

1. Train classifier C on L
2. Label U using classifier C
3. Select the top m% scored unlabeled examples U_T (we use m=10 in our case)
4. Cluster the examples $U_T \bigcup L$ into k clusters
5. Rank the k clusters using a cluster-based ranking function (described in the next section)
6. For each cluster in the ranked cluster list
   (a) Rank the claims within the cluster based on their informativeness of the quality of the cluster
   (b) For each claim in the ranked list within the cluster
      i. Query human expert for the label of the claim
      ii. Move to the next cluster if the evaluation metric (we use precision in our experiments) calculated over claims labeled so far in the cluster falls below threshold P

The intuition behind our MLT approach is to use a supervised classifier like SVM to initially provide a rough partitioning of the test examples and then use a clustering algorithm to do a more finer-grained partition. The function of the clustering is to take the examples that are scored highly by the classifier and group similar ones together, hopefully creating clusters that correspond to claims that have similar underlying *reasons* for being errors. Our two-step approach of post-ranking (or classification) clustering is also potentially useful in cases where there are different underlying subgroups (or reasons) in the positive class that may be disjoint which could create issues for a binary classifier (unless multiclass labels are available).

In Step 3, we use m=10% for our experiments which selects the top 10% of the examples. Setting m=100% would be equivalent to clustering all the examples without taking into account the classifier scores. Intuitively, we would choose m based on the class priors and the budget available for reviewing the examples.

In Step 4, we vary k (number of clusters) from 200 to 2800. The higher the number of clusters, the more coherent/similar each group will be which can decrease the audit time and provide better explanations but increases the number of clusters to examine. Another trade-off is that the larger the number of clusters the test examples may be split in different clusters thus not resulting in reduction of context-switch cost.

In Step 5, we experiment with a variety of metrics to rank the clusters described in the next section. These ranking metrics are a function of the number of positive examples (from L in the cluster), the mean score assigned by the classifier to the unlabeled examples U in this cluster, the ratio of the positive to negative labeled examples (from L), the inter-cluster similarity score of the cluster, and the intra-cluster similarity of the cluster.

To rank the claims for review within a cluster (in Step 6a), we use the score assigned by the classifier as the metric in our experiments. This is just one example of a metric that can be used. Other metrics we plan to experiment with include distance from the centroid of the cluster (to select prototypical examples from a cluster first), and sample selection metrics from active learning literature (based on uncertainty sampling for example). In Step 6b(ii), we use precision as the metric and set the threshold to 30%. This is chosen empirically based on the baseline precision scores and the class distribution.

4.1.2. Pool-based Active Learning

The *More-Like-This* (MLT) strategy is designed to reduce the time it takes to review (or label) an example and also to provide the auditors with immediate rewards. We also use active learning strategies to improve the future performance of the system. We use the standard pool-based active learning setting and use different sample selection strategies such as density-based and uncertainty sampling. This approach is potentially not immediately rewarding to the human auditors and focuses on future improvement of the classifier.

4.1.3. Hybrid Active Learning with More-Like-This

The tradeoff between the two previous strategies motivates a strategy that combines the benefits of both of them. We propose three different approaches to implement this hybrid strategy.

MLT-then-Active: This strategy performs MLT clustering as described in Section 4.1.1 but then uses active learning (with different sample selection metrics) to select examples to provide to the user for review and labeling.

Hybrid Sample Scoring: Hybrid Sample Scoring strategy assigns a score to every unlabeled example based on both strategies. Each example gets a score using active learning (which indicates it's utility in improving future classifier performance) as well as with the More-Like-This strategy (which indicates it's immediate utility). We can then use a weighted combination of these scores to create a final ranking.

Online Hybrid Learning: This hybrid strategy treats the two strategies independently and alternates among them based on the recent past, availability of expert resources, and business needs.

We have discussed these strategies with domain experts in claims processing and have found them to be quite receptive to the last strategy as a tradeoff. In the experiments presented in this paper, we don't use any hybrid strategies but we believe that these can be highly effective and are currently running experiments to evaluate their suitability in practical situations.

5. Experimental Results

5.1. Data and Experimental Setup

In this paper, we present experimental results using real claims data from a major insurance company in US as part of our project with them. We got approximately 23 million claims spanning 23 months. Of these 23 million claims, 379,000 had been manually labeled. In this labeled data set, around 65% claims were Rework and the rest Correct. It is important to remember that this distribution does not hold in the operational world since the labeled data has a skewed class distribution (described in section 4.1). Since SVMs are not able to handle categorical data we create binary features from the categorical features resulting in around 238,500 features.

We ran two sets of experiments for evaluating the ideas described in this paper. The first experiment was a live study with auditors from the insurance company where we gave the auditors a set of claims to audit based on system's recommendations and different strategies. In this paper, we will not describe the entire user study but just a part of the larger evaluation that motivated and verified the More-Like-This strategy. We present results with a second set of offline experiments that were designed to analyze our strategies in more detail.

5.2. Live System Deployment

We did a user study as part of a larger evaluation of the system to evaluate the usefulness of More-Like-This strategy in a real life deployment. The baseline system precision was 29% where the auditors audited 200 claims and found 58 claims to be Rework. With More-Like-This strategy, we obtained a hit rate of 55% where the auditors audited 307 claims and found 105 claims to be Rework. Thus we got a 90% relative improvement over the baseline system. The average audit time per claim was reduced from 3 min 55 sec for the baseline system to 2 min 52 sec for More-Like-This strategy, a relative reduction of 27% in audit time over baseline system. The reduction in time for auditing as well as the improvement in precision is statistically significant with $p << 0.001$ for Student's T-Test.

We also ran further experiments over a period of 3 weeks with two auditors to validate different aspects of our system but the details/discussion of the study are beyond the scope of this paper.

5.3. Offline experiments

Due to the high cost and limited availability of the auditors, we limited the live experiments using only the strategies that we thought would work the best. To conduct further in-depth analysis of different strategies described in this paper, we ran a large number of offline experiments. We performed 5-fold cross-validation experiments on claims data from last 5 months with 70,179 claims having 52% positive claims.[2]. Since the distribution of the positive class is skewed in historical data we create the test cases in each fold with 5% positive ratio as follows. We select all the negative examples belonging to the fold and then exclusively sample from the remaining examples so that the test data has 5% positive examples. We keep creating test cases until we exhaust all the positive examples in the fold. Thus each test fold has same negative examples but unique positive examples. In our case, we got 4 folds with 20 test cases and 1 fold with 19 test cases with the desirable 5% positive class ratio.

5.3.1. Metrics

For the large insurance company we worked with, the estimated capacity for auditing the claims is about 760 claims per day (379K claims in 23 months) whereas the volume of claims that need to be processed daily is approximately 46,000 claims per day (23 million claims

[2]Since we wanted to run multiple detailed experiments, we had to select a smaller dataset for practical runtime considerations

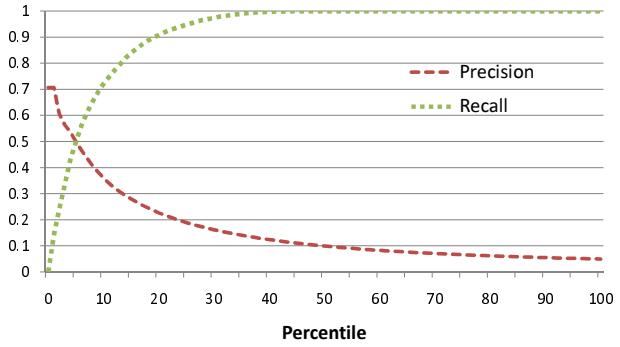

Figure 3. Precision Recall averaged over five folds with approx 20 test sets in each fold

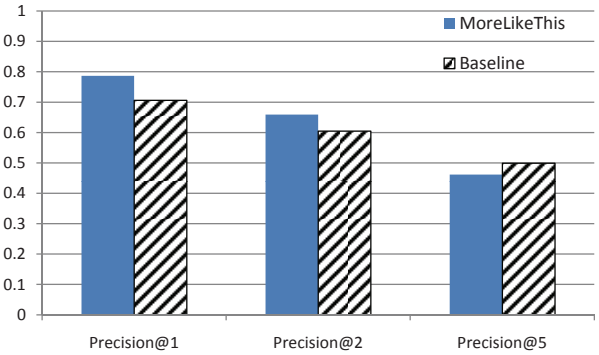

Figure 4. Comparison of performance of More-Like-This strategy with the baseline system

in 23 months). The audit capacity is approximately 1.5-2% of the total daily volume of the claims which is the case, in general, for most of the insurance companies.

Standard metrics such as accuracy, precision, or recall are useful in comparing different models and approaches, but not enough to minimize the number of claims errors, which is the eventual metric. Based on the audit capacity and discussions with domain experts, we decided that we need a metric that evaluates performance for the top 2-5% scored claims to maximize benefits in real scenarios. Hence, we use Precision (or hit rate) at 2nd Percentile as our metric. This is similar to the evaluation metric, Precision at top 10 documents, popularly used in the Information Retrieval community. In our experimental framework, we do have the flexibility to modify the metric easily and do model selection based on different metrics. In the results reported below, we show precision recall graphs where appropriate, as well as our precision at top 2% metric when comparing different models.

5.3.2. Is our system accurate?

We ran five fold validation experiments where each fold has approx 20 test sets with 5% Rework ratio to show the effectiveness of our baseline system. Figure 3 shows the Precision-Recall graph for this experiment averaged over five folds with approx 20 test sets each. The average precision at 2nd percentile is 60.5%. Compared to the 2-5% hit rate that the insurance companies get in their auditing practice currently, this is a significant boost in performance.

5.3.3. How well does More-Like-This work?

We use the More-Like-This strategy on top of the baseline system. We experimented with varying the number of clusters (k ranging from 200 to 2800), the clustering algorithms (Repeated Bisections, Repeated Bisections with global optimization and direct k-way clustering), and different cluster ranking metrics as described in Section 4.1.1. We used CLUTO (Karypis, 2002) for the clustering.

The best performance was achieved with a cluster size of 500 with repeated bisection clustering algorithm. The ranking criterion for the clusters was ratio of training Rework to Correct claims in a cluster followed by mean prediction score for test claims. Figure 4 shows the performance improvement over the baseline system. We obtain 9% relative improvement for Precision at 2nd percentile, which is the metric that we want to optimize. The improvement is statistically significant with $p << 0.001$ for paired, two-tailed Student's T-test.

5.3.4. How well do typical Active sample selection strategies work?

We experimented with the typical pool-based Active sample selection strategies: uncertainty and density based sampling along with the baseline random sampling in a batch learning setting. At every iteration, we select a batch of 1000 examples to query. Uncertainty is measured by the distance to the linear decision boundary of the SVM classifier where the closest instances to the boundary are most uncertain. We do density sampling by clustering the pool of candidate instances into the number of clusters equal to the batch size (1000 in our case) and then selecting the centroid of each cluster as the example to be queried, similar to (Nguyen & Smeulders, 2004). The results are shown in Figure 5. We notice that the performance is quite similar for the different strategies. This result is not too surprising as has previously been pointed out by (Guo & Schuurmans, 2008) that standard strategies applied myopically in batch learning setting are often worse than random querying.

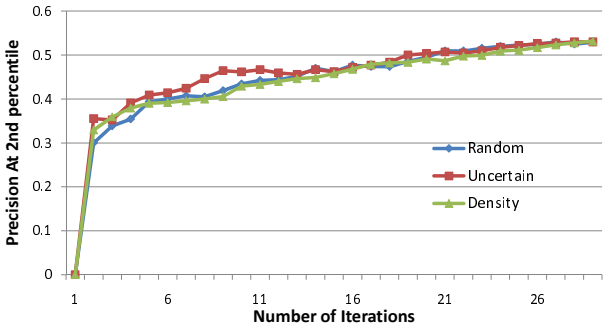

Figure 5. Performance of various Active strategies

6. Summary and Future Work

In this paper, we described an online cost-sensitive learning approach designed for applications where a batch classifier has been trained for a given classification task and focus on optimizing the interaction between the classifier and the domain experts (auditors in our case) who are consuming the results of this classifier. The goal is to make these experts more efficient and effective in performing their task as well as eventually improving the classifier over time. We describe a More-Like-This strategy that uses a supervised classifier to initially provide a rough partitioning of the unlabeled examples and then uses a clustering algorithm to do more finer-grained partitions. We validate our approach on the problem of detecting errors in health insurance claims and show significant reduction in labeling time while increasing the overall performance of the system.

We believe that focusing on reducing the time it takes to label an example is a promising area of research, especially when combined with traditional active learning strategies. This setting occurs in many practical deployments of machine learning systems (including the insurance claim processing system we have built). We are currently exploring ways to combine the two techniques and integrate it into system so the performance can be evaluated. Another area of current research for us is to evaluate user interfaces that can help reduce the labeling cost and also collect feedback to improve classification performance. We have developed one such interface (described earlier) but have not done detailed experiments to evaluate it's effectiveness.

References

Anand, Abhinav and Khots, Dmitriy. A data mining framework for identifying claim overpayments for the health insurance industry. In *INFORMS Workshop on Data Mining and Health Informatics*, 2008.

Guha, Sudipto and Munagala, Kamesh. Approximation algorithms for budgeted learning problems. In *ACM symposium on Theory of computing (STOC)*, 2007.

Guo, Yuhong and Schuurmans, Dale. Discriminative batch mode active learning. In *NIPS*, 2008.

Haertel, Robbie A., Seppi, Kevin D., Ringger, Eric K., and Carroll, James L. Return on investment for active learning. In *NIPS Workshop on Cost-Sensitive Learning*, 2008.

Joachims, Thorsten. Training linear svms in linear time. In *KDD*, 2006.

Kapoor, Aloak and Greiner, Russell. Learning and classifying under hard budgets. In *ECML*, 2005.

Karypis, George. Cluto - a clustering toolkit. Computer sciences technical report, University of Minnesota, 2002.

Kumar, Mohit, Ghani, Rayid, and Mei, Zhu-Song. Data mining to predict and prevent errors in health insurance claims processing. In *KDD*, 2010.

Margineantu, Dragos D. Active cost-sensitive learning. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2005.

Mladenic, Dunja and Brank, Janez. Feature selection using linear classifier weights: interaction with classification models. In *SIGIR*, 2004.

Nguyen, Hieu T. and Smeulders, Arnold. Active learning using pre-clustering. In *ICML*, 2004.

Saar-Tsechansky, Maytal, Melville, Prem, and Provost, Foster. Active feature-value acquisition. In *Management Science*, 2009.

Settles, Burr. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

Settles, Burr, Craven, Mark, and Friedland, Lewis. Active learning with real annotation costs. In *NIPS Workshop on Cost-Sensitive Learning*, 2008.