

Mining Extremely Skewed Trading Anomalies

Wei Fan, Philip S. Yu, and Haixun Wang

IBM T.J.Watson Research, Hawthorne NY 10532, USA,
{weifan,psyu,haixun}@us.ibm.com

Abstract. Trading surveillance systems screen and detect anomalous trades of equity, bonds, mortgage certificates among others. This is to satisfy federal trading regulations as well as to prevent crimes, such as insider trading and money laundry. Most existing trading surveillance systems are based on hand-coded expert-rules. Such systems are known to result in long developing process and extremely high “false positive” rates. We participate in co-developing a data mining based automatic trading surveillance system for one of the biggest banks in the US. The challenge of this task is to handle very skewed positive classes ($< 0.01\%$) as well as very large volume of data (millions of records and hundreds of features). The combination of very skewed distribution and huge data volume poses new challenge for data mining; previous work addresses these issues separately, and existing solutions are rather complicated and not very straightforward to implement. In this paper, we propose a simple systematic approach to mine “very skewed distribution in very large volume of data”.

1 Introduction

Trading surveillance systems screen and detect anomalous trades of equity, bonds, mortgage certificates among others. Suspicious trades are reported to a team of analysts to investigate. Confirmed illegal and irregular trades are blocked. This is to satisfy federal trading regulations as well as to prevent crimes, such as insider trading and money laundry. Most existing trading surveillance systems are based on hand-coded expert-rules. Such systems are known to result in long developing process and extremely high “false positive” rates. Expert rules are usually “yes-no” rules that do not compute a score that correlates with the likelihood that a trade is a true anomaly. We learned from our client most of the predicted anomalies by the system are false positives or normal trades mistakenly predicted as anomalies. Since there are a lot of false positives and there is no score to prioritize their job, many analysts have to spend hours a day to sort through reported anomalies and decide the subset of trades to investigate.

We participate in co-developing a data mining based automatic trading surveillance system for one of the biggest banks in the US. There are several goals to use data mining techniques. i) The developing cycle is automated and will probably be much shorter; ii) The model ideally should output a score, such as, posterior probability, to indicate the likelihood that a trade is truly anomalous; iii) Most importantly, the data mining model should have a much lower

false positive rate and higher recall rate, which will help the analysts to focus on the “really bad” cases. Besides engineering works to understand the task and data, the real technical challenge is to mine very skewed positive classes (e.g. $< 0.01\%$) in very large volume of data (e.g., millions of records and hundreds of features). The unique combination of very skewed distribution and huge data volume poses new challenges for data mining.

Skewed distribution and very large data volume are two important characteristics of today’s data mining task. Skewed distribution refers to the situation where the interesting or positive instances are much less popular than un-interesting or negative instances. For example, the percentage of people in a particular area that donates to one charity is less than 0.1% ; the percentage of security trading anomalies is less than 0.01% in the US. Skewed distribution also has unbalanced loss functions. For example, classifying a real insider trading as a normal transaction (false negatives), means millions of dollars of loss and law suit against a bank; while false positives, i.e., normal trades classified as anomaly, is a waste of time for the bank’s analysts. One big problem of skewed distribution is that many inductive learners completely or partially ignore the positive examples, or in the worst case, predict every instance as negative. One well cited case in data mining community is the KDDCUP’98 Donation Dataset. Even the positives are around 5% in the training data (not very skewed at all compared with trading anomalies), using C4.5 decision tree, a pruned tree has just one node that says “nobody is a donor”; an unpruned tree predicts as small as 4 household as donors while the actual number of donors are 4873. These kind of models are basically useless. Besides skewed distribution, another difficulty of today’s inductive mining is very large data volume. Data volume refers to the number of training records multiplied by the number of features. Most inductive learners have non-linear asymptotic complexity and requires data to be held in main memory. For example, decision tree algorithm has complexity of approximately $O(k \times n \cdot \log(n))$, where k is the number of features and n is the number of data records. However, this estimate is only true if the entire data can be held in main memory. When part of the data is on secondary storage, “trashing” will take place and model construction will take significantly longer period of time.

There has been extensive research in the past decade on both skewed distribution and scalable algorithms. Related work is reviewed in Section 4. However, most of these known solutions are rather complicated and far from being straightforward to implement. On the other hand, skewed distribution and large data volume learning are solved separately; there is no clear way to combine existing approaches easily. In this paper, we propose a simple systematic approach to mine “very skewed distribution in large volume of data”. The basic idea is to train ensemble of classifier (or multiple classifiers) from “biased” samples taken from the large volume of data. Each classifier in the ensemble outputs posterior probability $P(\ell|\mathbf{x})$ that \mathbf{x} is an instance of class ℓ . The probability estimates from multiple classifiers in the ensemble are averaged to compute the final posterior probability. When the probability is higher than a threshold, the trade will be classified as anomalous. Different thresholds will incur different true positive and

false positive rates. The best threshold is dictated by a given loss function in each application. To handle “skewed” positive class distribution, the first step is to generate multiple “biased” samples where the ratio of positive examples are intentionally increased. To find out the optimal ratio for positive examples, we apply a simple “binary” search like procedure. After the sample distribution is determined, multiple biased samples of the same size and distribution are sampled from the very large dataset. An individual classifier is trained from each biased sample. We have applied the proposed approach to a trading surveillance application for one of the biggest banks in the US. The percentage of positives is less than 0.01%, and the data volume on one business line is 5M records with 144 features.

2 The Framework

The training proceeds in three steps. We first need to find out how much data can be held in main memory at a time. We then apply a binary search algorithm to find out the optimal ratio of positives and negatives to sample from the dataset which gives the highest accuracy. Finally, we generate multiple biased samples from the training data set and compute a classifier from each biased sample. Since our algorithm is built upon concepts of probabilistic modeling and loss functions, we first review its important concepts.

Probabilistic Modeling and Loss Function. For a target function $t = F(\mathbf{x})$, given a training set of size n , $\{(\mathbf{x}_1, t_1), \dots, (\mathbf{x}_n, t_n)\}$, an inductive learner produces a model $y = f(\mathbf{x})$ to approximate the true function $F(\mathbf{x})$. Usually, there exists \mathbf{x} such that $y \neq t$. In order to compare performance, we introduce a loss function. Given a loss function $L(t, y)$ where t is the true label and y is the predicted label, an optimal model is one that minimizes the average loss $L(t, y)$ for all examples, weighted by their probability. Typical examples of loss functions in data mining are 0-1 loss and cost-sensitive loss. For 0-1 loss, $L(t, y) = 0$ if $t = y$, otherwise $L(t, y) = 1$. In cost-sensitive loss, $L(t, y) = c(\mathbf{x}, t)$ if $t = y$, otherwise $L(t, y) = w(\mathbf{x}, y, t)$. In general, when correctly predicted, $L(t, y)$ is only related to \mathbf{x} and its true label t . When misclassified, $L(t, y)$ is related to the example as well as its true label and the prediction. For many problems, t is nondeterministic, i.e., if \mathbf{x} is sampled repeatedly, different values of t may be given. The optimal decision y_* for \mathbf{x} is the label that minimizes the expected loss $E_t(L(t, y_*))$ for a given example \mathbf{x} when \mathbf{x} is sampled repeatedly and different t 's may be given. For 0-1 loss function, the optimal prediction is the most likely label or the label that appears the most often when \mathbf{x} is sampled repeatedly. Put in other words, for a two class problem, assume that $P(\ell|\mathbf{x})$ is the probability that \mathbf{x} is an instance of class ℓ . If $P(\ell|\mathbf{x}) \geq 0.5$, the optimal prediction is class ℓ . When mining skewed problems, positives usually carry a much higher “reward” than negatives when classified correctly. Otherwise, if positives and negatives carry the same reward, it is probably better off to predict “every one is negative”. Assume that positives carry a reward of \$100, and negatives carry reward of \$1, we predict

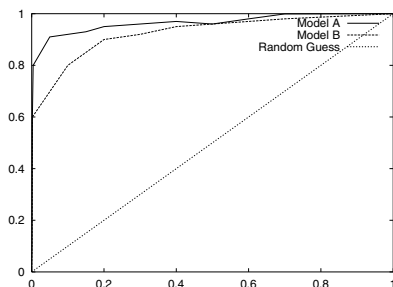


Fig. 1. ROC Example

positives when $P(+|\mathbf{x}) > 0.01$. Comparing with 0.5, the decision threshold of 0.01 is much lower.

For some applications, an exact loss function is hard to define or may change very frequently. When this happens, we employ an ROC-curve (or Receive Operation Characteristics curve) to compare and choose among models. ROC is defined over true positive (TP) and false positive rates (FP). True positive rate is the percentage of actual positives that are correctly classified as positives, and false positive rate is the percentage of negatives mistakenly classified as positives. An example ROC curve is shown in Figure 1. The diagonal line on the ROC is the performance of random guess, which predicts true positives and true negatives to be positive with the same probability. The top left corner (true positive rate is 100% and false positive rate is 0%) is a perfect classifier. Model A is better than model B at a particular false positive rate if its true positive rate is higher than that of B at the false positive rate; visually, the more an ROC curve closer to the top left corner, the better its performance is. For classifiers that output probabilities $P(\ell|\mathbf{x})$, to draw the ROC, we choose a decision threshold t ranging from 0 to 1 at a chosen step (such as 0.1). When $P(\ell|\mathbf{x}) > t$, the model predicts \mathbf{x} to be positive class ℓ . We compute true positive and false positive rates at each chosen threshold value.

The probability estimates by most models are usually not completely continuous. Many inductive models can only output a limited number of different kind of probability estimates. The number of different probability estimates for decision trees is at most the number of leaves of the tree. When this is known, the decision thresholds to try out can only be those probability outputs of the leaves. Any values in between will result the same recall and precision rates as the immediately lower thresholds.

Calculating Probabilities. The calculation of $p(\ell_i|x)$ is straightforward. For decision trees, such as C4.5, suppose that n is the total number of examples and n_i is the number of examples with class ℓ_i in a leaf, then $p(\ell_i|x) = \frac{n_i}{n}$. The probability for decision rules, e.g. RIPPER, can be calculated in a similar way. For naive Bayes classifier, assume that a_j 's are the attributes of x , $p(\ell_i)$ is the prior probability or frequency of class ℓ_i in the training data and $p(a_j|\ell_i)$

is the prior probability to observe feature attribute value a_j given class label ℓ_i , then the score $n(\ell_i|x)$ for class label ℓ_i is: $n(\ell_i|x) = p(\ell_i) \prod p(a_j|\ell_i)$ and the probability is calculated on the basis of $n(\ell_i|x)$ as $p(\ell_i|x) = \frac{n(\ell_i|x)}{\sum n(\ell_{i'}|x)}$

Choosing Sample Size. In order to scale, sample size cannot be more than that of available main memory. Assume that data records are fixed in length. To find out approximately how many records can be held in main memory, we simply divide the amount of available main memory by the size (in byte) of each record. To take into account main memory usage of the data structure of the algorithm, we only use 80% of the estimated size as an initial trial and run the chosen inductive learner. We then use “top” command to check if any swap space is being used. This estimation can just be approximate, since our earlier work has shown that the significantly different sampling size does not really influence the overall accuracy [1].

Choosing Biased Sampling Ratio. Since positive examples are extremely skewed in the surveillance data set, we choose to use all the positive examples while varying the amount of negative examples to find out the best ratio that results in best precision and recall rates. Finding the exact best ratio is a non-trivial task, but an approximate should be good enough in most cases. It is generally true that when the ratio of negatives decreases (or the ratio of positive increases), both the true positive rate and false positive rate of the trained model are expected to increase. Ideally, true positive rate should increase at a faster rate than false positive rate. In the extreme case, when there are no negatives sampled at all, the model will predict any \mathbf{x} to be positive, resulting in perfect 100% true positive rate but false positive rate is also 100%. Using this heuristic, the simple approach to find out the optimal amount of negatives to sample is to use progressive sampling. In other words, we reduce ratio of negatives progressively by “throwing out” portions of the negatives in the previous sample, such as by half. We compare the overall loss (if a loss function is given) or ROC curves. This process continues until the loss starts to rise. If the loss of the current ratio and previous one is significantly different (the exact significance depends on each application’s tolerance), we use a binary search to find the optimal sampling size. We choose the median ratio and computes the loss. In some situations if fewer negatives always result in higher loss, we reduce the ratio of positives while fixing the number of negatives.

Training Ensemble of Classifiers. After an optimal biased sampling distribution is determined, the next step is to generate multiple biased samples and compute multiple models from these samples. In order to make each sample as “uncorrelated” as possible, the negative examples are completely disjoint; in other words, each negative sample is used only once to train one base classifier. Since training multiple models are completely independent procedures, they can be computed either sequentially on the same computer or on multiple machines in parallel. In a previous work [1], we analyze the scalability of averaging ensemble

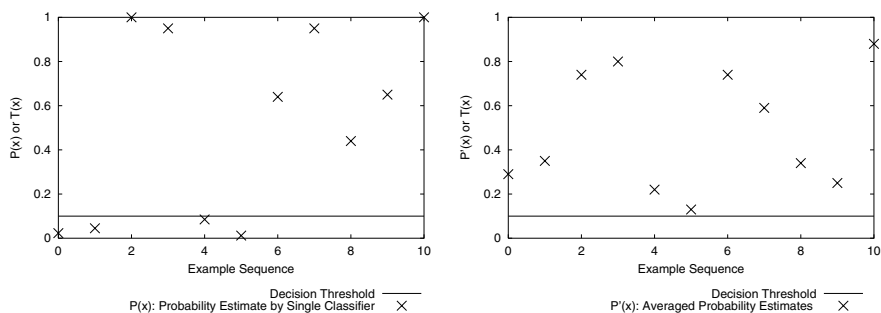


Fig. 2. Decision plots

in general. As a summary, it has both linear *scalability* and *scaled scalability*, the best possible scalability physically achievable.

Predicting via Averaging. When predicting a test example \mathbf{x} , each model in the ensemble outputs a probability estimate $P_i(\ell|\mathbf{x})$ that \mathbf{x} is an instance of positive class ℓ . We use simple averaging to combine probability output from K models $P(\ell|\mathbf{x}) = \frac{\sum_i P_i(\ell|\mathbf{x})}{K}$. We then use the techniques discussed previously to make optimal decision.

Desiderata. The obvious advantage of the above averaging ensemble is its scalability. The accuracy is also potentially higher than a single model trained from the entire huge dataset (if this could happen). The base models trained from disjoint data subsets make uncorrelated noisy errors to estimate posterior probability $P(\ell|\mathbf{x})$. It is known and studied that uncorrelated errors are reduced by averaging. Under a given loss function, different probability estimates on the same example may not make a difference to final prediction. If the decision threshold to predict \mathbf{x} to be an instance of the positive class is 0.01, probability estimates 0.1 and 0.99 will make exactly the same final prediction.

The multiple model is very likely more accurate than the single model because of its stronger bias towards predicting skewed examples correctly and skewed examples carry more reward than negative examples. Inductive learners have tendency to over-estimate probabilities. For example, decision tree learners try to build “pure” nodes of a single class. In other words, the leaf nodes tend to have one class of data. In this case, the posterior probability tends to be very close values to 0’s and 1’s. However, the averaging ensemble has an interesting “smoothing effect” to correct this over-estimation problem. Since each sample is mostly uncorrelated, the chances that all of the trained models predict close values to 0’s and 1’s are rare. In other words, the averaged probability estimates are smoothed out evenly towards the middle range between 0 and 1. Since the decision threshold to predict positive is less than 0.5 (usually much less than 0.5), it is more likely for true positives to be correctly classified. Since true positives carry much higher rewards than negatives, the overall effect is very likely to result in higher accuracy.

In Figure 2, we have one conjectured example to show the idea. In both plots, each “cross” represents a skewed positive example. Its location on the x-axis is just to separate each example away from each other. However the y-axis is the estimated probability $P(+|\mathbf{x})$ that \mathbf{x} is a positive example. The horizontal line is the value of the decision threshold to predict \mathbf{x} to be a member of positive class, i.e., we predict positive iff $P(+|\mathbf{x}) > t$. The left plot is the results of the single classifier trained from entire data as a whole (on a computer with enormous amount of memory) and the lost plot is the results of the averaging ensemble. As shown in the conjectured plot, all the probability estimates of averaging ensemble tend to move to the middle of the y-axis. This results in its correct prediction of the two examples on the bottom left corner, which were incorrectly classified as negative by the single model.

3 Experiment

The weekly data on one business line has approximately 5M records. We have one week data for training and another week for testing. The original number of features are 144 which include a lot of id’s (such as trader id, broker id, entry id, legal entity id among others), time and location information, and so on. Some of these fields cannot be used directly. For example, if a particular combination of id’s is an anomaly at a particular time, some inductive learner will pick that as a good rule which is obviously wrong. To avoid coincidences like this, we have encoded all id’s as either Y if this field is not empty or N otherwise. All dates are removed from the data; however the differences in days among all dates are computed and added into the original feature set. There are 34 different types of trading anomalies. Their total prior probability is less than 0.01%. Some of these anomalies are more frequent than others. A few of these anomalies are extremely rare; we only have 65 positive instances out 5M for one of the anomalies.

There are three learning tasks requested by the bank. The first task is to predict if a trade is an anomaly without identifying its particular type. The second task is to predict if a trade is a particular type of anomaly, and the same trade can be more than one type of anomaly. The last task is to identify some types of extremely rare anomalies. We were only given the labelled training data to generate the ensemble and *unlabelled* testing data to predict. Only after we sent our predictions to the bank, the true labels of the testing data were provided to us. Results were verified by our client.

We have used C4.5 decision tree algorithm to compute the base level models.

Task 1: Predicting Anomalies. The first task is to predict if a trade is an anomaly (any of the known types). The ratio of anomalies in the data is about 0.01%. It took about 20 mins to find the optimal biased sampling rate. The best ratio is 2% positives. The training took less than 2 hours on a PC running Linux.

The ROC curve of the results is shown in Figure 3. There are three curves in the plot. The random guess is the diagonal line. The curve on the top is “upper bound” of accuracy on the dataset, and the curve below that is the result of

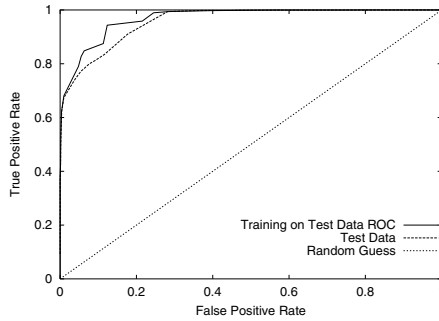


Fig. 3. ROC curve to predict anomalies

Table 1. True Positive Rate (tp) to Predict Anomaly Types

Type	num	tp(%)	Type	num	tp(%)	Type	num	tp(%)	Type	num	tp(%)
183	7027	99.9	101	2543	100.0	125	4028	99.9	129	4973	99.8
157	1645	98.6	108	503	90.7	152	172	100.0	119	272	82.4
110	1095	99.2	124	155	86.5	128	243	64.2	121	79	1.3
114	109	27.5	105	122	96.7	109	62	67.7	113	89	97.8
127	378	87.8	150	139	84.9	137	41	78.0	102	124	20.2
153	347	98.8	118	22	95.5	140	15	0.0	158	10	100.0
161	3	33.3	*112	3	0.0	*116	1	0.0	165	6	0.0
126	2	0.0	139	2	0.0	156	14	0.0	*154	24	0.0
171	1	0.0	166	3	0.0						

our model. Since it is not known how accurate a model can possibly be trained from the feature set, we have plotted the curve of “training error on the same test data”. In other words, we trained an additional ensemble on the test data. Obviously, training and testing on exactly the same dataset is very likely to have higher accuracy than training from a different dataset. We use this curve as the upper bound of accuracy. As clearly shown in the figures, the upper bound curve and our model’s curve are very close.

Task 2: Predicting Anomaly Types. The second task is to predict the types of anomalies. The data set is exactly the same as the task 1, however the class label is either normal or one of the anomaly types. We used the same sampling ratio as found in Task 1. The training took about 4 hrs on the same computer. The criteria to predict x as a type of anomaly is if its estimated probability is the highest than the probability to be of normal or any other types. The results are shown in Table 1. We list the types of the anomaly (denoted by a “code” at the request of the bank for confidentiality concerns), the number of instances in the training data and the true positive rate of the learned ensemble. If a particular type of anomaly does not appear in the training data, we put an * in front of the type code. For anomalies that are frequent (at least a few hundred instances), the model has very high true positive rate (at least 80%).

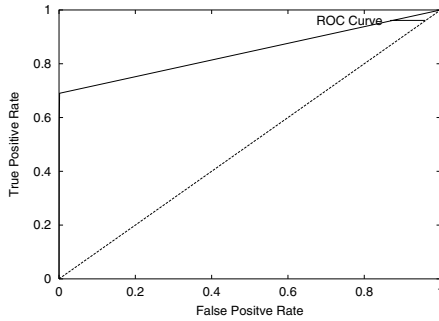


Fig. 4. ROC curve to predict a rare type of anomaly

Task 3: Predicting Extremely Rare Anomaly Types. Since some of the anomaly are extremely rare. For example, there are only 119 cases of anomaly type “114” in the test data. Since there are 34 different types of anomalies in total and some anomalies have thousands of instances in the dataset, those extremely rare anomalies are likely ignored by the learner due to their statistical insignificance. To solve this problem, we train an ensemble that only identifies on particular type of extremely rare anomaly. The bank is interested to predict anomaly type “114” which we only have 65 positive examples in the training data. The ROC curve to predict 114 is shown in Figure 4. The ensemble reaches true positive rate of about 70% with very low false positive rate (less than 0.01%). However, 30% of true anomalies are not able to be detected by the model. Our client conjectures that the feature set we use are not sufficient. We may have to look at multiple records to see a sequence of events.

4 Related Work

To scale up decision tree learning, SPRINT [2] generates multiple sorted attribute files from the original dataset. Decision tree is constructed by scanning and splitting attribute lists, which eliminates the need for large main memory. Since each attribute list is sorted, for a data file with f attributes and N examples, the total cost to produce the sorted lists is $f \cdot O(N \cdot \log(N))$. External sort is used to avoid the need for main memory. Each attribute list has three columns—a unique record number, the attribute value and the class label; the total size of attribute lists is approximately three times the size of the original dataset. When a split takes places at a node, SPRINT reads the attribute lists at this node and breaks them into r sets of attribute lists for r child nodes, for a total of f file read and $r \cdot f$ file writes. More recently, BOAT [3] constructs a “coarse” tree from a sample of the dataset that can fit into main memory. The splitting criterion in each node of the tree is tested against multiple decision trees trained from bootstrap samples of the sampled data. It refines the tree later by scanning the complete dataset, resulting in a total of two complete dataset read. Chan

proposed meta-learning [4] to scale up inductive learning. He has also applied the data reduction technique, however meta-learning build a tree of classifiers and only combine class label instead of probabilities. It has been shown that the quality of the decision tree is sometimes worse than a single decision tree. Most importantly, meta-learning cannot estimate its predictive accuracy until the model is fully constructed. In addition, meta-learning has sublinear speedup.

Previous research on skewed data mostly casts the problem into a cost-sensitive problem where skewed examples receive a higher weight. Some of previous proposed techniques are effective when both the cost-model is clearly defined and don't change often and the volume of the data is small. When one of these conditions fails, existing approaches are not applicable. Our proposed approach is a effective solution when the volume is too big and/or the cost model cannot be clearly defined.

5 Conclusion

In this paper, we have proposed an ensemble based approach to mine extremely skewed data. We have applied this approach on a very large trading surveillance data (5M with 144 features) with very skewed positives (less than 0.01%) from one of the biggest banks in the US. In three different tasks, the training on a desk top PC running Linux took less than 4 hrs. The ROC curve on unlabelled test data is as good as the best can be possibly obtained on this dataset. When the data volume is huge and positive examples are rare, the proposed ensemble approach provides a satisfactory solution.

References

1. Fan, W., Wang, H., Yu, P.S., Stolfo, S.: A framework for scalable cost-sensitive learning based on combining probabilities and benefits. In: Second SIAM International Conference on Data Mining (SDM2002). (2002)
2. Shafer, J., Agrawl, R., Mehta, M.: SPRINT: A scalable parallel classifier for data mining. In: Proceedings of Twenty-second International Conference on Very Large Databases (VLDB-96), San Francisco, California, Morgan Kaufmann (1996) 544–555
3. Gehrke, J., Ganti, V., Ramakrishnan, R., Loh, W.Y.: BOAT-optimistic decision tree construction. In: Proceedings of ACM SIGMOD International Conference on Management of Data (SIGMOD 1999). (1999)
4. Chan, P.: An Extensible Meta-learning Approach for Scalable and Accurate Inductive Learning. PhD thesis, Columbia University (1996)