Fraud Detection by Generating Positive Samples for Classification from Unlabeled Data

Levente Kocsis KOCSIS@SZTAKI.HU
András György GYA@SZIT.BME.HU
Machine Learning Research Group
Computer and Automation Research Institute of
the Hungarian Academy of Sciences

Abstract

In many real world (binary) classification problems it is easy to obtain unlabeled data, but labeled data are very expensive or simply unavailable. In certain cases, however, such as in the problem of detecting frauds in (computer) games, or insider trading in stock markets, one can assume that the unlabeled data contains very few samples from one class (fraudulent plays or insider trades), but it is possible to generate synthetic data from this class. Training a naive classifier on the above data is particularly suited for detecting frauds in Markov decision problems if the feature vectors of the classifier are composed of the frequency a player abates from the optimal policy in each state and the associated excess reward. Based on a synthetic example in blackjack, we demonstrate that the above classification method can perform quite well even in the case the generated positive samples come from a distribution different to the real one. The method is also applied to identify possibly fraudulent trades in the stock market.

1. Introduction

Consider a situation when a casino would like to detect automatically cheating poker players (or at least highlight the suspicion of cheating). Since in electronic casinos all aspects of the game are monitored constantly it is rather easy to collect training data for an automated cheat detector (similar data collection is possible in real casinos, by, e.g., cameras). Thus, when such a detector is ready, it is easy to feed it with input data. However, the main problem is that the collected data is not labeled, and so devising a general cheat detector is not straightforward.

A similar problem occurs in stock markets, where the use of certain extra information for trading is illegal and hence should be detected by market regulators. The detection of such behavior, called insider trading is a hard problem, as the social network spreading out the insider information is not known in general. Therefore, detecting such fraudulent trades must rely on recognizing unusual trading patterns of individual investors or brokers. In case the trades made by individual investors are available separately, one can easily check big returns for the investors, leading to a good starting point for the detection of suspicious trades. However, if trades are aggregated by brokers, or by a few (sub-) accounts of brokers, the problem becomes really complicated, as position closing trades, and hence realizing big returns, cannot be recognized for sure. Furthermore, brokers may also gain on information of near future orders of their customers. While usually a vast amount of trading data is available, detected frauds are scarce, hence practically only unlabeled data is available.

Both of the above examples represent a classification problem where it is easy to collect unlabeled data but usually no labeled data is available. On the other hand, one can suspect that the vast majority of the unlabeled data falls in one class (non-fraudulent data), and it is possible to generate synthetic data from the other class (fraudulent data). In particular, one can simulate cheating players in a casino (who are in the possession of some extra information, e.g., the value of some face-down card, or players who form a coalition), or, in case of the stock market, one can easily generate fraudulent trades in retrospect based on the future price changes. This synthetic data can be used to transform the unsupervised learning problem into one with positive and unlabeled data. Solutions to such problems have been proposed in several works. Under the general semi-supervised framework it is often assumed that the unlabeled data has some nice (e.g., identifiable) mixture distribution,

Appearing in *Proceedings of the ICML Workshop on Machine Learning and Games*. Copyright 2010 by the author(s)/owner(s).

and after the components of these distributions are identified, a few labeled data is satisfactory to obtain the corresponding labels (Castelli & Cover, 1995; Castelli & Cover, 1996; Zhu, 2007). The unlabeled and positive data case has also been studied specifically (Ward et al., 2008; Elkan & Noto, 2008); in these works it is usually assumed that either the unlabeled and positive data come from the true underlying distribution of the data and the positive data are labeled with a certain probability, or that the unlabeled data come from the true distribution, and the positive data follow the true distribution, as well. In contrast, we cannot make such assumptions as it would be too optimistic to assume that real world data is drawn from identifiable distributions or that we can generate synthetic samples from the true underlying positive distributions (in the case of detecting insider trades the latter would require covering all possible cheating scenarios). In this paper we demonstrate that despite of these facts, the above transformation of the problem may yield reasonable results.

In Section 2 we give a formal description of the above classification problem and propose a naive solution. In Section 3 we show how this solution can be applied for detecting unusual player behavior in Markov decision problems (MDPs) leading to extra rewards. The resulting algorithm is applied to detect frauds in a synthetic example of blackjack in Section 4, while in Section 5 we illustrate the application of the method to detect fraudulent trades in stock markets.

2. Learning from positive and unlabeled data

Let (X, Y) be a pair of random variables taking values in $\mathcal{C} \times \{0, 1\}$, where $\mathcal{C}$ is a compact subset of the d-dimensional Euclidean space $\mathbb{R}^d$, and let $(X_1, Y_1), \ldots, (X_n, Y_n)$ be drawn independently from the distribution of (X, Y). From this sample we can observe only $U = (X_1, \ldots, X_n)$, called the *unlabeled data*, and we need to construct a function $f(\cdot) = f(\cdot, U)$ to estimate the class corresponding to X, that is, to minimize the error probability $P(Y \neq f(X))$.

To facilitate this process, assume that we can generate another sample $P = (\hat{X}_1, \ldots, \hat{X}_m)$ drawn independently according to the distribution of a random variable $\hat{X} \in \mathbb{R}^d$ that is close to the conditional distribution of X given $Y = 1$. P is called the *positive data*. Now our goal is to construct another estimate $g(\cdot) = g(\cdot, U, P)$ to estimate the class Y for a given X, that is, to minimize $P(Y \neq g(X))$.

Assuming $\gamma = P(Y = 1) \ll 1$, it is a natural approach to label all samples from U negative and all samples from P positive, and approximate g by a consistent solution $\hat{g}$ of the resulting classification problem.

Similar setups have been considered in the literature, such as the case-control scenario of Ward et al. (2008) where it is assumed that P is drawn according to the true conditional distribution of X given $Y = 1$, while in the model of Elkan and Noto (2008) (P, U) follows the true underlying distribution with P selected completely random from the positive examples. These papers concentrate on improving $\hat{g}$, and the proposed methods basically reweight the samples based on the fact that U is a mixture of positive and negative data. This reweighting has no real effect in our situation as the proportion of positive samples γ is assumed to be very small in the unlabeled data set U. Moreover, in Sections 4 and 5 we will use the RProp algorithm (Riedmiller & Braun, 1993) to determine $\hat{g}$, and, as usual in similar situations, we reweight the samples during the training to ensure balanced estimation. Ward et al. (2008) also propose, beside reweighting, a theoretically well-founded tuning of the threshold parameter using cross validation from another sample, but this leads to similar threshold selection as in our naive approach.

Under various conditions it is easy to show that any consistent classifier $\hat{g}$ that is trained on (P, U) with the proposed artificial labeling is also consistent on the original problem. The next result is a simple example for such a case.

Proposition 1 *If the distribution of $\hat{X}$ equals the conditional distribution of X given $Y = 1$, then $\hat{g}$ converges to the optimal classifier for (X, Y) as $m, n \to \infty$ if* $\liminf_{n\to\infty} m/n > \gamma$.

3. Anomalous behavior in MDPs

MDPs are particularly suitable to describe many games and other real world situations. In this framework an agent moves in a state space and can perform different actions in each state. When an action is performed, the state is changed in some probabilistic way, and an occasional reward is paid to the agent. A cheat would have some insight in future state transitions or rewards that would not be possible by playing within the rules of the game, and will choose a sequence of actions that can exploit such an additional information.

Formally, in an MDP framework an agent moves in a state space $\mathcal{S}$: starting from a state s_0, at each time instant $t = 0, 1, 2, \ldots$ the agent chooses an action $a_t \in \mathcal{A}$ (where the set $\mathcal{A}$ is called the action space), and arrives to a new state $s_{t+1} \in \mathcal{S}$ according to the (given) conditional distributions $p(s_{t+1}|s_t, a_t)$, and receives some random rewards r_t whose distribution may depend on s_t, s_{t+1}, and a_t. In MDPs describing games usually there is a terminal state $\hat{s}$ indicating the end of the game, that is, no matter what action the agent chooses, it stays in the same state and receives no reward. The goal of the agent is to maximize its cumulative reward $R(s_0) = \sum_{t=0}^{\infty} r_t$ (note that under general conditions on p this sum contains almost surely only finitely many elements).

The agent's behavior is described by a possibly random policy $\pi : \mathcal{S} \to \mathcal{A}$ meaning that if the agent is at state s then it will perform action $\pi(s)$. An optimal policy π^* is one maximizing the expected cumulative reward $\mathbb{E}R(s_0)$. It can be shown (see, e.g., Sutton & Barto, 1998) that under general conditions on the MDP there exists an optimal policy π^* that is deterministic and independent of the initial state s_0.

In this setup we assume that we have a several game long record of each player, and hence it is possible to collect statistics about the behavior of each player at certain states of the MDP. Thus, a data instance consists of a sequence of (state, action, reward) triplets resulting from the sequence of actions performed by an agent. When one wants to decide whether a player cheats or not, it is intuitive to use as features the frequency of abating from the optimal policy in particular states, and the reward received following the action biased by the expected reward of an optimal policy associated to the particular state (or state-action). More formally, each player is described as follows. Assume that n games are recorded for a given player. For each state-action pair (s, a), let $\bar{\pi}(s, a)$ denote the frequency the player chose action a at state s. Furthermore, for each state s consider all the games when the player was at state s and chose a suboptimal action (i.e., different to $\pi^*(s)$), and let $\bar{R}(s)$ denote the average reward, computed over these games, collected after the first time the agent visited state s and chose a suboptimal action.

Then the features describing the player may be composed of the frequency of making suboptimal decisions in each state and the resulting extra reward:

$$\left\{ \left(1 - \bar{\pi}(s, \pi^*(s)), \bar{R}(s) - \mathbb{E}R^*(s)\right), s \in \mathcal{S} \right\}$$

where $\mathbb{E}R^*$ is the expected cumulative reward of the optimal policy π^*.

Obviously, in many MDPs it is very hard to find the optimal policy π^*, but often it is possible to construct a reasonable estimate. In these situations the approximate optimal policy and the resulting achievable result may play the role of the optimal quantities above. When there are more then one optimal actions in a state (or the expected reward of a suboptimal action is inferior by just a small margin than that of the optimal action), the features describing the player would include the frequencies of deviating from the set of (almost) optimal actions, and the corresponding extra reward. In states where more than one optimal actions are available, additional information not available to a fair player may differentiate these actions. To detect such situations, it might be interesting to augment the features for suboptimal actions with the frequencies of choosing different optimal actions and the resulting extra rewards.

The above model concerns to games when the agent acts solitary in its environment. However, in multi-player games the MDP framework may also be a sufficiently good model if the opponents usually play obliviously with respect to specific players' actions (this may be a sufficiently good approximation in many practical situations and very restrictive in others). In this case the above approach is applicable to detect colluding players (various types of collusions are described in, e.g., Smed et al., 2006). If two (or more) players play fairly then their optimal strategy is just the product of the optimal strategies derived from the MDP model. On the other hand, if they collude, usually higher rewards can be achieved for both of them, since in a coalition when a player deviates from an optimal policy, often another member of the coalition will benefit, or information that enables a player a higher than expected reward can be facilitated by the other player. Thus, following the single-player model, the two players are described by the frequencies they abate from the optimal product policy and the resulting extra rewards. Naturally, if several players are in the game, all possible player groups have to be tried out (in practice, however, only those who can be suspected to collide), but this is not a serious impediment.

4. Blackjack

Blackjack is one of the most popular casino card games. In essence, it is a game played against a dealer with the objective of getting a higher card total than the dealer, without going over 21 (*busting*). At the beginning of each round, the player and the dealer receive an initial hand of two cards each. One of the dealer's card is face up and the other is face down (the *hole* card). Subsequently, the player can choose to *hit* (i.e. request a further card) until he deems too risky to ask for more cards (avoiding the possibility of going over 21), at which point he *stands*. In general there may be other actions available to the player, but these are the most essential ones, and we deemed them sufficient for the purpose of our experiments. If the player has not bust, it is the dealer's turn to request further cards, and she is doing so according to a predetermined strategy that depends solely on the value of her cards. For the dealer we adopted the *hit on soft* strategy, that is, she asks for more cards if the total value of her cards is less than 17 or 17 including an ace. Further choices made regarding the possible variants of the game include the 3:2 payoff for a player blackjack (an ace plus a figure or a 10), and dealing from a 6 deck shoe.

A 'fair' player bases his policy on the upcard of the dealer and his current sum. In our setup we include players who play optimally given this information, and also players that deviate randomly from the optimal policy depending on how much worse the suboptimal action is (essentially, a soft-max policy). We consider cheats that have knowledge about either the next card to be dealt or the hole card. In either case the cheats may attempt to disguise their strategy by mixing it with the optimal fair strategy. While in the tests we examine both types of cheating we add to the training set only cheats using the next card (and cheating in all occasions), considering the second type of cheating as ‘unexpected’.

Our data consists of 100,000 fair players (N), two sets of 10,000 cheats relying on the next card (TN) or the hole card (TH), with various degrees of disguise, and a set of 1,000 cheats that are playing optimal knowing the next card to be dealt (C). The description for each player is collected from 1,000 games played by the player, and includes the rate of non-optimal actions in each state (defined by the current sum and the upcard of the dealer) and the payoff obtained for each state (there are approximately 300 states). In the training the fair players and a sample of 1,000 players from TN are labeled as non-cheats, while the players from C are labeled as cheats. We trained a neural network using the RProp algorithm, and the performance of the classification of the trained network on the two test sets TN and TH are shown in Figure 1, top. We may observe that only significantly disguised players avoid detection for the cheats relying on the next card, while a large portion of the cheats relying on the hole card are detected despite not having any labeled sample for this type of cheating. The average payoff obtained by the players of the test sets in their respective 1,000 games are shown in Figure 1, bottom. We note that there is a large overlap in the result of cheats and non-cheats. This overlap excludes the possibility of detecting cheats based mainly on their gain. One can assume that cheats can access/utilize extra information in certain states of the game, and therefore our approach seems more suitable for detecting cheats than naively comparing observed gains with the optimal (expected) gain, without considering player behavior at different states separately.

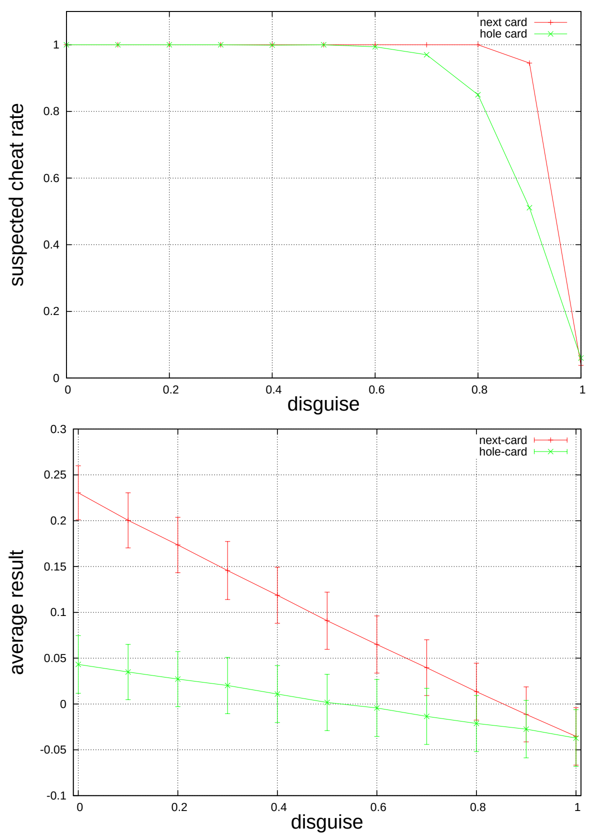

Figure 1. The frequency of rating a player suspect for cheats using the next card or the hole card (top), and the average payoff obtained by the cheats (bottom). The cheats are shown with varying level of disguise. Note that, in fact, players with $disguse = 1$ are not cheats. For the payoff the standard deviation over the players is also shown.

5. Fraudulent trades

We demonstrate our method on the problem of detecting fraudulent trades on the Budapest Stock Exchange.[1] In this exchange market regulators see only aggregated data for each broker desk; thus the initiator of the trades are unknown, and so corresponding opening and closing trades cannot be determined, further complicating the problem. Also, there are very few known cases when proven insider trading has occurred. Thus, to detect fraudulent trades, only a large collection of unlabeled data is available (i.e., with no indication of being fraudulent or not), and in the current practice unsupervised anomaly detection methods (e.g., unusual values of some simple statistics) are used to detect frauds. For illustration purposes we selected ECONET, a small stock that is often the target of speculations. Therefore, from time to time, it is very volatile with large variations in the stock price, while it is quite illiquid in other periods with very few trades only. The source data contains the sequence of orders for this stock for the period 2005–2006, and, for each order only the standard information, such as type (market/limit), price, volume and time, augmented with the identification number of the originating broker desk is recorded.

Based on the price sequence we have generated fraudulent orders that utilize big price changes; this models a perfect fraud situation where the insider traders are aware of the future price changes. We have generated two types of fraudulent orders. One type aims to achieve an at least 1% return in a short period of at most 2 minutes, while the other type

[1]By detecting fraudulent trades we mean identifying trades with strong suspicion of being fraudulent. Usually such trades are then further investigated by the regulators of the stock exchange.

aims to win 7% within 2 days. The orders are inserted randomly in the time period of length 300 and 500 seconds, respectively, preceding the preselected cheating time. To make the generated orders as similar to the real ones as possible, other characteristics of the orders are based on those of the original ones in the preceding and following 3 days, with volumes that can be 1.5 times bigger than observed.

In the resulting classification problem, each order is represented by a set of features that include the maximum potential gains for 1 minute, 3 hours and 1000 minutes (that is, how much money could be earned by closing the position at the optimal price in the given time period, calculated in continuous trading time), the maximum potential loss in 1 hour, and a simple representation of the order book containing the total volume and the average price of the orders within 2, 5, and 8 percent of the mid-price of the book from both the seller and buyer side. For the original orders the original features were kept that were obtained by simulating the real trading, while for the artificial fraudulent orders, the features were generated by simulating the sequence of the real and the artificially generated orders (note that the effect of each inserted extra order vanishes very fast).

Based on the training set obtained this way, the RProp algorithm was used to rank the orders according to how suspicious they seemed. Figure 2 shows the order book after the arrival of the order ranked most suspicious by our method, together with the graphs showing the evolution of the stock price and traded volume slightly before and after the order is placed (in the price and volume graphs the order arrives at time 0). Although we cannot be certain of any fraudulent activity, it can be noted that this order is a relatively aggressive buy order, arriving to the strong side of the buyer book (though it could have been more aggressive by immediately executing with an existing sell order), when there is a strong bias in the book favoring sell orders, and soon after the order is filled the price increases a lot.

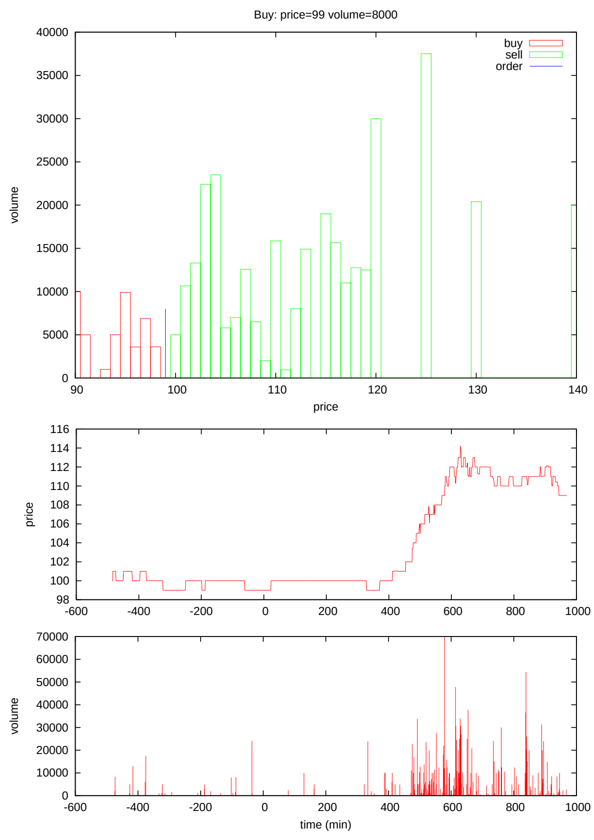

Figure 2. Order book, price, and traded volume for the most suspicious order for ECONET.

6. Conclusion

Fraud detection is a typical example of real world semi-supervised learning problems: it is very easy to obtain a vast amount of unlabeled data, but data corresponding to declared frauds is very hard to collect. On the other hand, unlike in many semi-supervised learning problems, it is often possible to automatically generate synthetic data corresponding to fraudulent behavior. While in the standard classification problem from unlabeled and positive data it is usually assumed that the positive data is generated according to the real distribution, we cannot make such an assumption here. We have demonstrated that if the unlabeled data contains relatively few positive samples, the naive approach of solving a classification problem where all the unlabeled data are deemed to be negative, may yield reasonable performance in certain cases. We described briefly how the method can be applied to detect fraudulent behavior in single- and multi-player games that can be modeled by Markov decision processes. The resulting algorithm was tested with encouraging results in a slightly simplified version of blackjack with synthetic (fair and fraudulent) players, and for detecting fraudulent trades in real stock market data.

Acknowledgment

The authors wish to thank the anonymous referees for constructive comments. This work was supported in part by the PASCAL2 Network of Excellence under EC grant no. 216886 and by the Hungarian National Office for Research and Technology under the project ASTOR Jedlik 2/004/05.

References

Castelli, V., & Cover, T. M. (1995). On the exponential value of labeled samples. *Pattern Recogn. Lett.*, *16*, 105–

111.

Castelli, V., & Cover, T. M. (1996). The relative value of labeled and unlabeled samples in patternrecognition with an unknown mixing parameter. *IEEE Transactions on Information Theory*, *42*, 2102–2117.

Elkan, C., & Noto, K. (2008). Learning classifiers from only positive and unlabeled data. *14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 213–220). Las Vegas, Nevada.

Riedmiller, M., & Braun, H. (1993). A direct adaptive method for faster backpropagation learning: The RPROP algorithm. *Proceedings of the IEEE International Conference on Neural Networks* (pp. 586–591). IEEE Press.

Smed, J., Knuutila, T., & Hakonen, H. (2006). Can we prevent collusion in multiplayer online games? *Proceedings of the Ninth Scandinavian Conference on Artificial Intelligence (SCAI 2006)* (pp. 168–175).

Sutton, R., & Barto, A. (1998). *Reinforcement learning: An introduction*. Bradford Book. MIT Press.

Ward, G., Hastie, T., Barry, S., Elith, J., & Leathwick, J. R. (2008). Presence-only data and the EM algorithm. *Biometrics*, *65*, 554–563.

Zhu, X. (2007). Semi-supervised learning literature survey. Computer Sciences TR 1530, University of Wisconsin Madison, available at `http://pages.cs.wisc.edu/~jerryzhu/pub/ssl_survey.pdf`.