On-line Unsupervised Outlier Detection Using Finite Mixtures with Discounting Learning Algorithms

Kenji Yamanishi
NEC Corporation
4-1-1,Miyazaki,Miyamae,
Kawasaki,Kanagawa
216-8555,JAPAN
k-yamanishi@cw.jp.nec.com

Jun-ichi Takeuchi
NEC Corporation
4-1-1,Miyazaki,Miyamae,
Kawasaki,Kanagawa
216-8555,JAPAN
tak@ap.jp.nec.com

Graham Williams
CSIRO Mathematical and
Information Sciences
GPO Box 664, Canberra ACT
2601, Australia
Graham.Williams@cmis.csiro.au

ABSTRACT

Outlier detection is a fundamental issue in data mining, specifically in fraud detection, network intrusion detection, network monitoring, etc. SmartSifter, which we abbreviate as SS, is an outlier detection engine adrressing this problem from the viewpoint of statistical learning theory. This paper provides a theoretical basis for SS and empirically demonstrates its effectiveness. SS detects outliers in an on-line process through the on-line unsupervised learning of a probabilistic model (using a finite mixture model) of the information source. Each time a datum is input SS employs an *on-line discounting learning algorithm* to learn the probabilistic model. A score is given to the datum based on the learned model, with a high score indicating a high possibility of being a statistical outlier. The novel features of SS are: 1) it is adaptive to non-stationary sources of data; 2) a score has a clear statistical/information-theoretic meaning; 3) it is computationally inexpensive; and 4) it can handle both categorical and continuous variables. An experimental application to network intrusion detection shows that SS was able to identify data with high scores that corresponded to attacks, with low computational costs. Further experimental application has identified a number of meaningful rare cases in actual health insurance pathology data from Australia's Health Insurance Commission.

1. INTRODUCTION

The problem of outlier detection is one of the most fundamental issues in data mining. It is closely related to fraud detection, network intrusion detection, etc., since criminal or suspicious activities may often induce statistical outliers.

We focus on the issue of *on-line unsupervised outlier detection* in which the following conditions are required:
1) *Outliers are detected based on unsupervised learning of the information source.* In general statistical approaches to outlier detection, one first learns an underlying model of the mechanism for data-generation from examples, and then evaluates how large a given input datum is deviated relative to the model (see e.g., [1]). We require here that the learning process be unsupervised. Although the supervised learning based approach is popular in fraud detection ([2],[9],[11]), the unsupervised learning based one is not only more technically difficult but also more practically important, since in real situations labeled examples might not be available.

2) *The process is on-line.* That is, every time a datum is input, it is required to evaluate how large the datum is deviated relative to a normal pattern. In contrast, most existing work on outlier detection (e.g., [1],[8],[9],[11]) in statistics and data mining is concerned with batch-detection processes, in which outliers can be detected only after seeing the entire dataset. The on-line setting is more realistic than the batch one when one deals with the tremendous amount of data in network monitoring, etc.

Note that there exists only a few work (e.g. [3]) focusing on the on-line unsupervised learning based approach. This paper introduces SmartSifter (abbreviated as SS throughout this paper), which meets the requirements 1) and 2).

The approach of SmartSifter is as follows:
I) SS uses a probabilistic model as a representation of an underlying mechanism of data-generation. The model takes a hierarchical structure, in which a *histogram density* with a number of cells is used to represent a probability density over the domain of categorical variables and for each cell a *finite mixture model* is used to represent the probability density over the domain of continuous variables.
II) Every time a datum is input, SS employs an on-line learning algorithm to update the model. We have developed the *SDLE (Sequentially Discounting Laplace Estimation)* algorithm for learning the histogram density for the categorical domain (Sec.2.1), and the *SDEM (Sequentially Discounting Expectation and Maximizing)* algorithm (Sec.2.2) for learning the finite mixture for the continuous domain. Their most important feature is that they *discount* the effect of past examples in the on-line process.
III) SS gives a score to each datum on the basis of the learned model, measuring how large the model has changed after learning. Thus a high score indicates a high possibility that the datum is an outlier.

The novel features of SS are summarized as follows:

a) *SS is adaptive to non-stationary sources.* In conventional statistical approaches, it is usually assumed that an underlying information source for data-generation is stationary [1]. This is, however, not realistic when one deals with drifting sources or time-series data. The discounting algorithm employed by SS learns the source forgetting out-of-date statistics of data, using a decay factor, hence is expected to be adaptive to non-stationary sources.

b) *A score calculated by SS has a clear statistical/ information-theoretic meaning.* In most previous work, a score is calculated using heuristics such as cost [4],[5] and lacks a statistical justification, while the Mahalanobis distance [1], the quadratic distance [5],[8] were used to score outliers in some work. SS defines a score for a datum in terms of a statistical distance measuring how large the distribution learned from the datum has moved from the one before learning. Thus it is natural to interpret that a datum of high score is, with high probability, an outlier.

c) *SS is computationally inexpensive.* The computational complexity of SS for calculating a score for each datum is linear in the number of parameters in the model and quadratic in the number of variables. For example, it can process about 90,000 data with four attributes in 28 seconds (Pentium III 550MHz).

d) *SS can deal with both categorical and continuous variables.* To our knowledge, SS is the first on-line unsupervised outlier detector that can deal with both categorical and continuous variables.

The design of SS was inspired by the work by Burge and Shaw-Taylor [3]. Our work differs from [3] in the following regards: 1) SS treats both categorical and continuous variables, while [3] deals only with continuous ones. 2) While [3] uses two models in the algorithm: the long term model and the short term one, SS unifies them into one model with the aim of a clearer statistical meaning and a lower computational cost. 3) SS uses either a parametric representation for a probabilistic model or a non-parametric one, while only a non-parametric one is used in [3]. In Sec.3.1, we compare our parametric method with the non-parametric one to show that the former outperforms the latter both in accuracy and computation costs.

We empirically demonstrate the practical effectiveness of SS using the network intrusion dataset (KDD Cup 1999) [7]. Although it was used in the context of a supervised intrusion detector learning problem in KDD Cup 1999, we used it in the scenario of on-line unsupervised intrusion detection. In our experiment, for a network access log dataset of size 458,078 including 1687 intrusions, it detected 55% intrusions in the top 1% data of highest scores, and 82% intrusions in the top 5% data of highest scores. We also used a health insurance pathology dataset supplied by the Australian Health Insurance Commission to demonstrate that SS was able to identify several meaningful rare cases in it.

2. OUTLINE OF SMARTSIFTER

Let $(\mathbf{x},\mathbf{y})$ denote a datum where $\mathbf{x}$ denotes a vector of categorical variables and $\mathbf{y}$ denotes a vector of continuous variables. We write the joint distribution of $(\mathbf{x},\mathbf{y})$ as $p(\mathbf{x},\mathbf{y}) = p(\mathbf{x})p(\mathbf{y}|\mathbf{x})$. Let $p(\mathbf{x})$ be represented by a *histogram density* with a finite number of disjoint cells (Sec.2.1), and for each cell, for all $\mathbf{x}$s that fall into it, let $p(\mathbf{y}|\mathbf{x})$ be represented by an identical form of a *finite mixture model* (Sec.2.2). Hence we prepare as many finite mixture models as cells in the histogram density. Consider the situation where a sequence of data is given: $(\mathbf{x}_1,\mathbf{y}_1), (\mathbf{x}_2,\mathbf{y}_2)\cdots$ in an on-line process. The fundamental steps of SmartSifter are given as follows:

1. Given the tth input datum $(\mathbf{x}_t,\mathbf{y}_t)$, identify the cell that $\mathbf{x}_t$ falls into and update the histogram density using the SDLE algorithm (Sec. 2.1) to obtain $p^{(t)}(\mathbf{x})$. Then, for that cell, update the finite mixture model using the SDEM/SDPU algorithm (Sec.2.2) to obtain $p^{(t)}(\mathbf{y}|\mathbf{x})$. For other cells, set $p^{(t)}(\mathbf{y}|\mathbf{x}) = p^{(t-1)}(\mathbf{y}|\mathbf{x})$.
2. Calculate a score for the datum on the basis of the models before and after updating (Sec.2.3).

2.1 Categorical Variables

Below we describe how to learn the histogram density over the categorical domain. Let the number of categorical variables be n. Let the range of the ith categorical variable be $\mathcal{A}^{(i)} = \{a_1^{(i)},\cdots,a_{u_i}^{(i)}\}$ $(i=1,\cdots,n)$. Classify them to obtain a finite number of disjoint sets: $\{A_1^{(i)},\cdots,A_{v_i}^{(i)}\}$ $(i = 1,\cdots,n)$, where $A_j^{(i)}\cap A_k^{(i)} = \emptyset$ $(j\neq k)$ and $\mathcal{A}^{(i)} = \cup_{j=1}^{v_i}A_j^{(i)}$. We call the cell $A_{j_1}^{(1)}\times\cdots\times A_{j_n}^{(n)}$ the $(j_1,\cdots,j_n)$th *cell*. We have $M = v_1\times\cdots\times v_n$ cells in total. This induces a partitioning of the domain.

Given such a partitioning of the domain, a *histogram density* forms a probability distribution, which takes a constant value on each cell. The histogram density is specified by a parameter $\theta = (q_1,\cdots,q_k)$ where $\sum_{j=1}^k q_j = 1$, $q_j\geq 0$ and q_j denotes the probability value for the jth cell. If there are L_j symbols in the j-th cell, the probability value of each symbol $\mathbf{x}$ in it is given by $p(\mathbf{x}) = q_j/L_j$.

We introduce here the SDLE algorithm. This is a variant of the Laplace law, by which one estimates the probability value over a discrete domain with $\hat{p}(a) = (T_a+\beta)/(T+M\beta)$ for each element a where $0<\beta<1$, T is the size of a data sequence, T_a is the number of occurrences of a in the sequence, and M is the number of elements in the discrete domain. The SDLE algorithm is specified by a discounting parameter $r_h(>0)$ where a smaller r_h indicates that SDLE has a larger influence of past examples (see Fig.1). The Laplace law is reduced to the case of $r_h\longrightarrow 0$ in SDLE.

2.2 Continuous Variables

Next we describe how to learn the model over the continuous domain. We consider two versions: the parametric version and the non-parametric one.

2.2.1 Parametric Version

In the parametric version we employ as a finite mixture model over the continuous domain a *Gaussian mixture model*:

$$p(\mathbf{y}|\theta) = \sum_{i=1}^{k} c_i p(\mathbf{y}|\mu_i,\Lambda_i),$$

where k is a positive integer, $c_i\geq 0$, $\sum_{i=1}^k c_i = 1$ and each $p(\mathbf{y}|\mu_i,\Lambda_i)$ is a d-dimensional Gaussian distribution with density specified by mean μ_i and covariance matrix Λ_i:

$$p(\mathbf{y}|\mu_i,\Lambda_i) = \frac{1}{(2\pi)^{d/2}|\Lambda_i|^{1/2}}\exp\left(-\frac{1}{2}(\mathbf{y}-\mu_i)^T\Lambda_i^{-1}(\mathbf{y}-\mu_i)\right)$$

where $i=1,\cdots,k$ and d is the dimension of each datum. We set $\theta = (c_1,\mu_1,\Lambda_1,\cdots,c_k,\mu_k,\Lambda_k)$.

```
SDLE Algorithm
A partitioning of the domain: $\{A_1^{(i)}, \cdots, A_{v_i}^{(i)}\}$ $(i = 1, \cdots, n)$,
$r_h, \beta$: given.
 Step 1. /* Initialization */
     Let $T(j_1, \cdots, j_n) = 0$ $(1 \le j_i \le v_i,\ i = 1, \cdots, n)$.
     $t := 1$
 Step 2. /* Parameter Updating */
     while $t \le T$   ($T$: sample size)
         Read $\mathbf{x}_t = (x_1, \cdots, x_n)$
         For each $(j_1, \cdots, j_n)$-th cell,
           $T_t(j_1, \cdots, j_n) := (1 - r_h)T_{t-1}(j_1, \cdots, j_n) + \delta_t(j_1, \cdots, j_n)$
           $q^{(t)}(j_1, \cdots, j_n) := \frac{T_t(j_1, \cdots, j_n) + \beta}{(1 - (1 - r_h)^t)/r_h + k\beta}$
         For each $\mathbf{x} \in A_{j_1}^{(1)} \times A_{j_2}^{(2)} \times \cdots A_{j_n}^{(n)}$,
           $p^{(t)}(\mathbf{x}) := \frac{q^{(t)}(j_1, \cdots, j_n)}{|A_{j_1}^{(1)}| \cdot |A_{j_2}^{(2)}| \cdots |A_{j_n}^{(n)}|}$
     where $\delta_t(j_1, \cdots, j_n) = 1$ if the $t$-th datum falls into the
     $(j_1, \cdots, j_n)$-th cell, and $\delta_t(j_1, \cdots, j_n) = 0$ otherwise.
     $t := t + 1$
```

Figure 1: SDLE Algorithm

First, we review the incremental EM algorithm [10] for learning Gaussian mixture models. Letting s be an iteration index, we define sufficient statistics $S_i^{(s)} (i = 1, \cdots, k)$ by

$$S_i^{(s)} = \left(c_i^{(s)}, \bar{\mu}_i^{(s)}, \bar{\Lambda}_i^{(s)}\right)$$

$$\stackrel{\text{def}}{=} \frac{1}{t} \cdot \left(\sum_{u=1}^{t} \gamma_i^{(s)}(u), \sum_{u=1}^{t} \gamma_i^{(s)}(u) \cdot \mathbf{y}_u, \sum_{u=1}^{t} \gamma_i^{(s)}(u) \cdot \mathbf{y}_u \mathbf{y}_u^T\right),$$

where

$$\gamma_i^{(s)}(u) \stackrel{\text{def}}{=} \frac{c_i^{(s-1)} p(\mathbf{y}_u | \mu_i^{(s-1)}, \Lambda_i^{(s-1)})}{\sum_{i=1}^{k} c_i^{(s-1)} p(\mathbf{y}_u | \mu_i^{(s-1)}, \Lambda_i^{(s-1)})}. \qquad (1)$$

We also define $S_i^{(s)}(v)$ $(i = 1, \cdots, k)$ for $\mathbf{y}_v$ by

$$S_i^{(s)}(v) \stackrel{\text{def}}{=} \frac{1}{t} \cdot \left(\gamma_i^{(s)}(v), \gamma_i^{(s)}(v) \cdot \mathbf{y}_v, \gamma_i^{(s)}(v) \cdot \mathbf{y}_v \mathbf{y}_v^T\right).$$

The incremental EM algorithm for Gaussian mixture models consists of the following E-step and M-step [10]:

E-step: Choose a datum $\mathbf{y}_u$ from the sequence $\mathbf{y}^t$. Given $\theta^{(s-1)}$, compute

$$S_i^{(s)}(u) = \frac{1}{t} \cdot \left(\gamma_i^{(s)}(u), \gamma_i^{(s)}(u) \cdot \mathbf{y}_u, \gamma_i^{(s)}(u) \cdot \mathbf{y}_u \mathbf{y}_u^T\right).$$

Then, set $S^{(s)} = S^{(s-1)} - S^{(s-1)}(u) + S^{(s)}(u)$. (2)

M-step: Compute the new estimate $\theta^{(s)}$ by

$$\mu_i^{(s)} = \bar{\mu}_i^{(s)}/c_i^{(s)} \text{ and } \Lambda_i^{(s)} = \bar{\Lambda}_i^{(s)}/c_i^{(s)} - \mu_i^{(s)} \mu_i^{(s)T}.$$

The point is that in the E-step the sufficient statistics $S^{(s-1)}$ is updated relative to for arbitrarily chosen $\mathbf{y}_u$. By repeating the iteration of the E and M steps w.r.t.s, $\theta^{(s)}$ converges.

We introduce here the SDEM algorithm by modifying the incremental EM algorithm as follows:

(A) Choose $\mathbf{y}_u$ in time order, i.e., choose $\mathbf{y}_s$ at the sth round in the E-step. Make only one iteration of the E and M steps for each s. This makes the parameters updated every time a datum is input.

(B) Introduce a discounting parameter r $(0 < r < 1)$ to modify the updating rule (2) into the following:

$$S_i^{(s)} := (1 - r)S_i^{(s-1)} + r \cdot (\gamma_i^{(s)}(s), \gamma_i^{(s)}(s)\mathbf{y}_s, \gamma_i^{(s)}(s)\mathbf{y}_s\mathbf{y}_s^T),$$

This rule makes the statistics exponentially decay with a factor $(1 - r)$ as the stage proceeds.

Note: If the modification is (B) only, without (A), we obtain the algorithm proposed by Nolan (see e.g. [10]). The details of the SDEM algorithm are in Fig. 2. Notice that in Fig. 2, a parameter α is introduced in order to improve the stability of the estimates of c_i, which is set to 1.0 ~ 2.0. Usually $c_i^{(0)} = 1/k$ and $\mu_i^{(0)}$s are set so that they are uniformly distributed over the data space. The computation time for the SDEM algorithm at each round is $O(d^2k)$ where d is the dimension of the data and k is the number of Gaussian distributions.

```
SDEM Algorithm ($r, \alpha, k$: given)
Step 1. /* Initialization */
   Set $\mu_i^{(0)}, c_i^{(0)}, \bar{\mu}_i^{(0)}, \Lambda_i^{(0)}, \bar{\Lambda}_i^{(0)} (i = 1, ..., k)$.
   $t := 1$
Step 2. /* Parameter Updating */
   while $t \le T$  ($T$:sample size)
     Read $\mathbf{y}_t$
     for $i = 1, 2, ..., k$
       $\gamma_i^{(t)} := (1 - \alpha r)\frac{c_i^{(t-1)} p(\mathbf{y}_t | \mu_i^{(t-1)}, \Lambda_i^{(t-1)})}{\sum_{i=1}^{k} c_i^{(t-1)} p(\mathbf{y}_t | \mu_i^{(t-1)}, \Lambda_i^{(t-1)})} + \frac{\alpha r}{k}$
       $c_i^{(t)} := (1 - r)c_i^{(t-1)} + r\gamma_i^{(t)}$
       $\bar{\mu}_i^{(t)} := (1 - r)\bar{\mu}_i^{(t-1)} + r\gamma_i^{(t)} \cdot \mathbf{y}_t$
       $\mu_i^{(t)} := \bar{\mu}_i^{(t)}/c_i^{(t)}$
       $\bar{\Lambda}_i^{(t)} := (1 - r)\bar{\Lambda}_i^{(t-1)} + r\gamma_i^{(t)} \cdot \mathbf{y}_t\mathbf{y}_t^T$
       $\Lambda_i^{(t)} := \bar{\Lambda}_i^{(t)}/c_i^{(t)} - \mu_i^{(t)}\mu_i^{(t)T}$
     $t := t + 1$
```

Figure 2: SDEM Algorithm

2.2.2 Non-Parametric Version

In the non-parametric version we employ as a finite mixture model over the continuous domain a *kernel mixture model*:

$$p(\mathbf{y}|q) = \frac{1}{K} \sum_{i=1}^{K} w\left(\mathbf{y} : q_i\right), \qquad (3)$$

where K is given, $q = \{q_1, \cdots, q_K\}$ is called a set of *prototypes*, $w(\cdot : q_i)$ is the kernel function defined as a Gaussian density with mean q_i and variance matrix $\Sigma = diag(\sigma^2, \cdots, \sigma^2)$ for a positive constant σ, and d is the dimension of a datum.

The difference between the parametric and non-parametric versions is that the coefficient vector and the variance matrix for a finite mixture are variable in the former model, while they are fixed in the latter model.

We introduce here the *SDPU (Sequentially Discounting Prototype Updating) algorithm* for on-line learning prototypes in the kernel mixture. This is based on Grabec's algorithm [6]. For a given data sequence $\mathbf{y}^t = \mathbf{y}_1 \cdots \mathbf{y}_t$, first define:

$$f(\mathbf{y}|\mathbf{y}^t) = \sum_{\tau=1}^{t} A(t, \tau)w(\mathbf{y} : \mathbf{y}_\tau),$$

where $A(t, t) \stackrel{\text{def}}{=} r$, $A(t, \tau) \stackrel{\text{def}}{=} r(1 - r)^{t-\tau-1}$ for $\tau \le t - 1$ and $0 < r < 1$ is a discounting factor. Note that $\sum_{\tau=1}^{t} A(t, \tau) =$

1 holds. Hence $f(\mathbf{y}|\mathbf{y}^t)$ is a weighted sum of $w(\mathbf{y} : \mathbf{y}_\tau)$ $(\tau = 1, \cdots, t)$ where the weight becomes large as τ increases. Next define the square error of $p(\mathbf{y}|q)$ to $f(\mathbf{y}|\mathbf{y}^t)$ by

$$\bar{\epsilon}^2(q : \mathbf{y}^t) = \int \left(p(\mathbf{y}|q) - f(\mathbf{y}|\mathbf{y}^t)\right)^2 d\mathbf{y}.$$

For a new input $\mathbf{y}_{t+1}$, the SDPU algorithm updates a prototype $q^{(t)}$ into $q^{(t)} + \Delta q^{(t)}$ by choosing $\Delta q^{(t)}$ so that $\bar{\epsilon}^2(q^{(t)} + \Delta q^{(t)} : \mathbf{y}^t\mathbf{y}_{t+1})$ is minimal under the condition that $\bar{\epsilon}^2(q : \mathbf{y}^t)$ is minimized by $q = q^{(t)}$. The details are shown in Fig.3.

```
SDPU Algorithm  ($r, \sigma, K$: given)
Step 1.  /* Initialization */
  Set $q_i^{(0)} (i = 1, ..., K)$ so that they are uniformly distributed.
  $t := 1$
Step 2  /* Prototype Updating */
  while $t \leq T$  ($T$:sample size)
    Read $\mathbf{x}_t$
    for all $(j, m, k, l)$
      $B_{kl}^{(t)} := r\Big(K \cdot (x_{t+1,l} - q_{kl}^{(t)}) \exp(-\frac{|x_{t+1} - q_k^{(t)}|^2}{4\sigma^2})$
              $- \sum_{i=1}^K (q_{il}^{(t)} - q_{kl}^{(t)}) \exp(-\frac{|q_i^{(t)} - q_k^{(t)}|^2}{4\sigma^2})\Big)$
      (Here $q_{kl}^{(t)}$ denotes the $l$th component of $q_k^{(t)}$.)
      $C_{jmkl}^{(t)} := (\delta_{ml} - \frac{(q_{kl}^{(t)} - q_{jl}^{(t)})(q_{km}^{(t)} - q_{jm}^{(t)})}{2\sigma^2}) \exp(-\frac{|q_k^{(t)} - q_j^{(t)}|^2}{4\sigma^2})$
    Solve the linear equation: $\sum_{j,m} C_{jmkl}^{(t)} \Delta q_{jm}^{(t)} = B_{kl}^{(t)}$
    $(k = 1, ..., K, l = 1, ..., d)$ for all $(j, m)$
      $q_{jm}^{(t+1)} := q_{jm}^{(t)} + \Delta q_{jm}^{(t)}$
    $t := t + 1$
```

Figure 3: SDPU Algorithm

Note: The computation time for the SDPU algorithm at each round is $O(d^2K^2)$ where d is the dimension of data and K is the number of prototypes. The SDPU algorithm coincides with Grabec's algorithm [6] in the case where we let r be time dependent as $r = 1/t$.

The non-parametric version of SS can be thought of as a variant of Burge and Show-Taylor's program [3] since they are both based on Grabec's algorithm. Major differences between them are 1) methods of discounting past examples and 2) methods of score calculation.

2.3 Score Calculation

Below we give a method of calculating a score for a datum. Let $p^{(t)}(\mathbf{x}, \mathbf{y})$ be the joint distribution obtained at time t, i.e., $p^{(t)}(\mathbf{x}, \mathbf{y}) = p^{(t)}(\mathbf{x})p^{(t)}(\mathbf{y}|\mathbf{x})$. Given an input $(\mathbf{x}_t, \mathbf{y}_t)$ at time t, we define its *Hellinger score* as

$$S_H(\mathbf{x}_t, \mathbf{y}_t) = \frac{1}{r^2} \sum_{\mathbf{x}} \int \left(\sqrt{p^{(t)}(\mathbf{x}, \mathbf{y})} - \sqrt{p^{(t-1)}(\mathbf{x}, \mathbf{y})}\right)^2 d\mathbf{y},$$

where r is a discounting parameter where we set $r_h = r$. Intuitively, this score measures how large the distribution $p^{(t)}$ has moved from $p^{(t-1)}$ after learning from $(\mathbf{x}_t, \mathbf{y}_t)$.

We also define another score called the *logarithmic loss* as

$$S_L(\mathbf{x}_t, \mathbf{y}_t) = -\log p^{(t-1)}(\mathbf{x}_t) - \log p^{(t-1)}(\mathbf{y}_t|\mathbf{x}_t).$$

From the viewpoint of information theory, the logarithmic loss can be thought of as the codelength required to encode $(\mathbf{x}_t, \mathbf{y}_t)$ under the assumption that a datum is generated according to a probability density $p^{(t-1)}$.

3. EXPERIMENTAL RESULTS

3.1 Network Intrusion Detection

We applied SmartSifter to the dataset KDD Cup 1999 [7] prepared for network intrusion detection. The purpose of the experiment was to detect as many intrusions as possible in an on-line setting without making use of the labels. Although in KDD Cup 1999 the data labels were used in training for supervised intrusion detection, we used them only for the evaluation of SS. Hence any supervised approach to the dataset is not fairly comparable with SS.

Each datum in KDD Cup 1999 is specified by 41 attributes (34 continuous and 7 categorical) and a label describing attack type (22 kinds: normal, back, buffer_overflow, etc.) where all labels except "normal" indicate an attack. We used four of the original 41 attributes (service, duration, src_bytes, dst_bytes) because these four were thought of as the most basic attributes. Only 'service' is categorical. The number of service kinds is 41, and we classified them into {http, smtp, ftp, ftp_data, others}. Since the continuous attribute values were concentrated around 0, we transformed each value into a value far from 0, by $y = \log(x + 0.1)$.

The original dataset contains 4,898,431 data, including 3,925,651 attacks (80.1%). This rate of attacks is too large for statistical outlier detection. Therefore, we produced a subdataset SF consisting of 976,157 data, including 3,377 attacks (0.35%) by picking up the data whose attribute *logged_in* is positive. Attacks that successfully *logged_in* are called *intrusions.* We further produced from SF two kinds of datasets SF50 and SF10 by random sampling, where SF50 consists of 50% of SF, and SF10 consists of 10% of SF. For SF10 and SF50, the first 8,000 data and the first 30,000 data were respectively not scored but used only for training because the model would not be well-trained in the early stages.

We generated two SF10s by making different random samplings. For each of them, we ran both parametric and nonparametric versions of SS where the former is denoted by SS while the latter by SS*. The parameters of SS are set to $r = 0.0002, r_h = 0.0003, k = 2, \alpha = 2.0$, while those of SS* are set to $r = 0.0002, r_h = 0.0003, K = 10, \sigma = 0.2$. SS processed 97,621 data in 28.2 seconds, while SS* did in 244.8 seconds. Table 1 shows the number of intrusions SS(*) detected where the logarithmic loss was used as a score. Here *coverage* is the ratio of the number of detected intrusions to that of intrusions in the dataset that SS(*) scored. SS is superior to SS* both in accuracy and computation time. This implies that the SDEM algorithm with a Gaussian mixture works better than Grabec's algorithm using a kernel mixture, which is also a basis of the program in [3].

We also generated four SF50s by making different random samplings or using an additional categorical variable 'protocol-type' of range { tcp, udp , imcp }. Table 2 shows the number of intrusions SS detected in the four SF50s. Remarkably, SS was able to detect 82% intrusions in the top 5% data of highest scores.

3.2 Outliers in Medical Pathology Data

Australia's Health Insurance Commission (HIC) administers the universal health insurance scheme known as Medicare, in operation since 1975. CSIRO Australia is working with the HIC to explore the utilization of pathology services.

top ratio	♯ of intrusions included (coverage)			
	SS		SS*	
1%	129 (40%)	124 (37%)	17 (5%)	21 (6%)
3%	226 (70%)	222 (65%)	43 (13%)	51 (15%)
5%	251 (77%)	257 (76%)	64 (19%)	78 (23%)
10%	319 (98%)	331 (98%)	206 (62%)	132 (39%)

Table 1: Detected Intrusions in SF10

top ratio	♯ of intrusions included (coverage)			
	service		service & protocol_type	
1%	922 (55%)	968 (57%)	351 (21%)	346 (21%)
3%	1282 (76%)	1260 (75%)	1150 (68%)	1165 (69%)
5%	1391 (82%)	1284 (76%)	1310 (78%)	1304 (77%)
10%	1617 (96%)	1576 (94%)	1633 (97%)	1636 (97%)

Table 2: Detected Intrusions in SF50

A dataset covering 2 years with over 32 million pathology transactions is the basis of the study. Each transaction records the type of medical test performed, an encrypted identifier for the rendering doctor, the laboratory performing the test, and the doctor who requested the test, and other information relating to the type of doctor and patient.

SmartSifter has been used on this dataset for two distinct functions. The first used SS to identify errors in the data. An initial application to the original transaction dataset identified transactions that were unusual—outliers. Many of the transactions identified had invalid data items (dollar amounts had extra zeros, ages of patients were greater than 100 years, etc.). The dataset for analysis can thus be cleansed of these records.

The primary purpose of SS is to explore various transformations of the transaction dataset. These include aggregating doctors (20,000 entities), patients (4,000,000 entities), and laboratories (150 entities). SS was able to identify, for example, laboratories that have patterns of behavior that are out of the ordinary. One dataset used consisted of just 7 variables for each laboratory, including percentage distributions over five test categories (e.g., Chemical tests, Microbiology tests, Immunology tests, etc.), the number of different patients dealt with by the laboratory, and the frequency of tests performed (Table 3). SS identified laboratories 109, 126, and 114 with consistently high distance scores. Two of these laboratories specialized in Tissue Pathology (109 and 114) and the other does no Microbiology nor Tissue Pathology (126). The "normal" situation for this dataset indicates that pathology providers generally perform a spread of tests.
Further analyses with the HIC are in progress, employing a wide variety of techniques drawn from data mining and statistics in general. The use of SS provides insights into the data, both through the identification of erroneous data and through the identification of entities that may require further investigation.

Lab	T_1	T_2	T_3	T_4	T_5	F_1	F_2
108	0.64	0.19	0.10	0.04	0.01	0.32	0.03
109	**0**	**0**	**0**	**1**	**0**	**1**	**0.43**
110	0	0.00	0	0.57	0.42	0.89	0.09
111	0.29	0.13	0.02	0.05	0.49	0.32	0.01
114	**0**	**0**	**0**	**1**	**0**	**0.96**	**0.30**
116	0.41	0.05	0.05	0.01	0.46	0.46	0.31
119	0	0	0	0.5	0.5	0.5	0.5
126	**0.33**	**0**	**0.33**	**0**	**0.33**	**1**	**1**
127	0.33	0.18	0.01	0.01	0.45	0.33	0.01

Table 3: Sample laboratory data.

4. CONCLUSION

This paper has proposed SmartSifter as a program for on-line unsupervised outlier detection. We gave a statistical theory for SS and demonstrated its effectiveness in terms of accuracy and computation time through the two experiments: network intrusion detection for KDD Cup 1999 and rare event detection for the health insurance pathology dataset provided by Australian Health Insurance Commission. It is expected that the framework of on-line unsupervised outlier detection using SS would be further applied into a wide variety of data mining issues other than fraud detection, such as trend detection, topic identification.

Acknowledgement. The authors sincerely express their gratitude to Dr.Yoshifumi Yamamoto of NEC Information Systems for writing the codes for SmartSifter.

5. ADDITIONAL AUTHORS

Additional author: Peter.Milne (CSIRO Mathematical and Information Sciences, GPO Box 664, Canberra ACT 2601, Australia, email: Peter.Milne@cmis.csiro.au).

6. REFERENCES

[1] V. Barnett and T. Lewis, *Outliers in Statistical Data*, John Wiley & Sons, 1994.

[2] F. Bonchi, F. Giannotti, G. Mainetto, and D. Pedeschi, A classification-based methodology for planning audit strategies in fraud detection, in *Proc. of KDD-99*, pp:175–184, 1999.

[3] P. Burge and J. Shaw-Taylor, Detecting cellular fraud using adaptive prototypes, in *Proc. of AI Approaches to Fraud Detection and Risk Management*, pp:9–13, 1997.

[4] P. Chan and S. Stolfo, Toward scalable learning with non-uniform class and cost-distributions: A case study in credit card fraud detection, in *Proc. of KDD-98*, AAAI-Press, pp:164–168 (1998).

[5] T. Fawcett and F. Provost, Activity monitoring: noticing interesting changes in behavior, in *Proc. of KDD-99*, pp:53–62, 1999.

[6] I. Grabec, Self-organization of Neurons described by the maximum-entropy principle, *Biological Cybernetics* vol. 63, pp:403–409, 1990.

[7] http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[8] E. M. Knorr and R. T. Ng, Algorithms for mining distance-based outliers in large datasets, in *Proc. of the 24th VLDB Conference*, pp:392–403, 1998.

[9] W. Lee, S. J. Stolfo, and K. W. Mok, Mining in a data-flow environment: experience in network intrusion detection, in *Proc. of KDD-99*, pp:114–124, 1999.

[10] R. M. Neal and G. E. Hinton, A view of the EM algorithm that justifies incremental, sparse, and other variants, ftp://ftp.cs.toronto.edu/pub/radford/www/publications.html 1993.

[11] S. Rosset, U. Murad, E. Neumann, Y. Idan, and G. Pinkas, Discovery of fraud rules for telecommunications-challenges and solutions, in *Proc. of KDD-99*, pp:409–413, 1999.