BIRDNEST: Bayesian Inference for Ratings-Fraud Detection

Bryan Hooi* Leman Akoglu‡
Neil Shah* Mohit Kumar§
Alex Beutel* Disha Makhija§
Stephan Günneman† Christos Faloutsos*

Abstract

Review fraud is a pervasive problem in online commerce, in which fraudulent sellers write or purchase fake reviews to manipulate perception of their products and services. Fake reviews are often detected based on several signs, including 1) they occur in short bursts of time; 2) fraudulent user accounts have skewed rating distributions. However, these may both be true in any given dataset. Hence, in this paper, we propose an approach for detecting fraudulent reviews which combines these 2 approaches in a principled manner, allowing successful detection even when one of these signs is not present. To combine these 2 approaches, we formulate our Bayesian Inference for Rating Data (BIRD) model, a flexible Bayesian model of user rating behavior. Based on our model we formulate a likelihood-based suspiciousness metric, Normalized Expected Surprise Total (NEST). We propose a linear-time algorithm for performing Bayesian inference using our model and computing the metric. Experiments on real data show that BIRDNEST successfully spots review fraud in large, real-world graphs: the 50 most suspicious users of the Flipkart platform flagged by our algorithm were investigated and all identified as fraudulent by domain experts at Flipkart.

1 Introduction

Online reviews play an important role in informing customers' purchasing decisions. This has led to the problem of fake reviews, in which businesses write or purchase fake reviews in order to raise the popularity of their products or services. Hence, it is crucial for online commercial platforms to identify and remove these reviews, in order to maintain customers' trust in the accuracy of their reviews.

Various inputs such as rating, review text, timestamp etc. may be available for detection systems; in this work we focus on ratings and timestamps as they are commonly available and informative features. Informally, our problem is:

PROBLEM 1. (INFORMAL) ***Given*** *a set of users and products, and timestamped ratings (e.g.* 1 *to* 5 *stars) by users for products,* ***compute*** *a suspiciousness score for each user.*

Currently, a number of algorithms use a temporal approach to detect ratings fraud [6, 7, 28]. These focus on catching products that receive a large number of positive or negative reviews in a short time, motivated by the 'bursty' nature of fraudulent reviews when a store wishes to rapidly increase their popularity or defame their competitors. An alternative approach based on rating distributions is to focus on finding users who rate products very differently from other users [12, 17]. These focus on detection of suspicious behavior by users or products in terms of their deviation from normal practice.

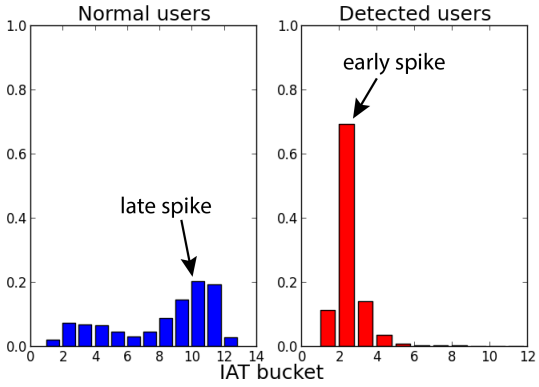
In this paper, we aim to combine both approaches in a principled way by constructing a Bayesian model for rating behavior, then formulating a likelihood-based metric which measures how much a user deviates from the rest of the users.

The Bayesian approach also provides a principled solution to the conceptually difficult problem of finding a good tradeoff between users with extreme rating distributions vs. users with larger number of ratings. Is a user with 50 ratings (average rating 5.0) more suspicious than a user with 500 ratings (average rating 4.95)? Bayesian methods allow us to quantitatively answer this question. Namely, our Bayesian method combines the rating distribution and number of ratings to estimate our *beliefs* about the rating characteristics of a user in a way that captures our uncertainty, which then determines how suspicious the user is.

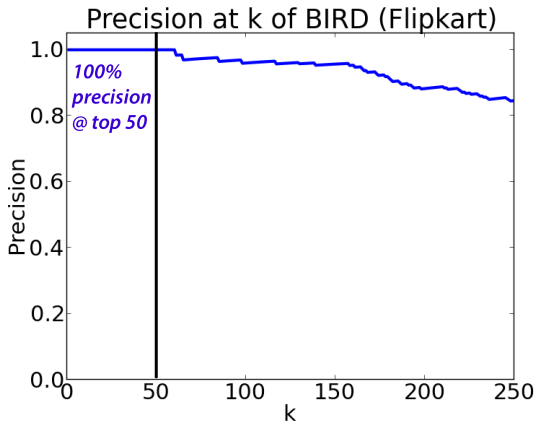
Our contributions are:

- **Theoretically sound user behavior model**: we define a Bayesian model for the data based on a mixture model which captures different types of user behavior. This model then allows us to determine how much an anomalous user deviates from normal behavior.
- **Suspiciousness metric**: we define a likelihood-based metric which measures how much a user deviates from normal behavior.
- **Algorithm**: we propose a scalable and effective algorithm for learning the Bayesian model and evaluating suspiciousness.
- **Effectiveness**: we show that our method successfully spots review fraud in large, real-world graphs, with

*Carnegie Mellon University
†Technische Universität München
‡Stony Brook University
§Flipkart



(a) **Common pattern observed that detected users' ratings are more 'bursty' than normal users.**. Times between a user's ratings were bucketed logarithmically; detected users have shorter times between ratings.



(b) **BIRDNEST is effective in practice**, with 211 users of the top 250 flagged by BIRDNEST involved in fraud.

Figure 1: **BIRDNEST combines temporal and rating information in a principled manner to detect fraud with high precision.** Inspecting the most suspicious 100 users shows their strongly anomalous patterns.

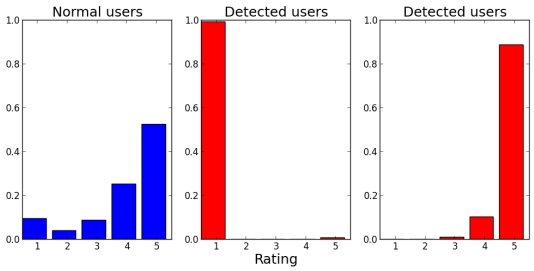


Figure 2: **Common pattern observed that detected users' ratings deviate strongly from normal users**: inspecting the detected users shows that they consists of two groups: highly negative users (middle) and highly positive users (right).

precision of over 84% on the top 250 Flipkart users flagged by our algorithm.

Reproducibility: our code is open-sourced at `www.andrew.cmu.edu/user/bhooi/ratings.tar`.

2 Background and Related Work

Content-based approaches A significant portion of opinion fraud comes from customer reviews online. Customer reviews have been long studied [8], and many methods for review fraud focus on review text, such as [5, 11, 19]. While these methods are illuminating, many sites only have ratings without text, or text is easily manipulated. Therefore, in our setting, we focus on ratings and their temporal characteristics, as review text is not always available.

Graph-based approaches Much of the existing work in fraud or anomaly detection on graphs has focused on detecting fraud in pure graphs; that is, graphs with no node or edge labels. This includes spectral methods which use eigen-decomposition or singular value decomposition (SVD) to group similar nodes in the graph [10, 21, 24]. [27] uses an iterative approach to label as honest and dishonest. Approaches based on Markov Random Fields and belief propagation have also been used to identify dense or suspicious subgraphs [1, 20]. [29] detects spammers through graph-based measures measuring self-similarity and neighborhood diversity. However, these methods do not make use of key temporal and rating data.

Temporal methods for fraud detection There are a number of works on anomaly detection in multivariate time series [4, 16, 22, 26]. [3] focuses on fraudulent temporal patterns in graphs, and [6] found suspicious inter-arrival times between events in social media. A couple of works address temporal patterns of reviews, e.g. [28] detects spam singleton reviews and [7] detects time periods of unusual activity. However, our goal is to compute a general, principled, likelihood-based measurement of how suspicious each user is. In this regard, [9] offers a general suspiciousness metric for count data but is not suitable for ratings data.

Behavior modeling and fraud detection A wide body of research has focused on understanding user behavior and especially rating behavior. In particular ratings have been studied by the recommendation systems community, with both frequentist [13] and Bayesian models [23] demonstrat-

ing great success. Additionally some models have worked to take into account temporal features [14], and others have captured the bimodal patterns in ratings data [2].

Other behavior models have been proposed to detect users who deviate from normal practice in a meaningful way [12, 17]. In [25] a similar problem of finding anomalies in temporal rating data was treated with information theoretic arguments. By taking a Bayesian approach, we develop a significantly different perspective on the problem and our resulting metric of suspiciousness is more flexible, allowing for explicit priors, unique posteriors for each user, and easy extensions to other distributions.

3 Bayesian Model

3.1 Motivating Example We start by illustrating why a Bayesian approach is helpful. Consider users Alice, Bob and Carol whose rating distributions are as given in Figure 3. For example, Alice rated 4 products, all with 5 stars. Bob did the same, 50 times. Carol gave about 300 ratings, and exhibits a 'hockey-stick' distribution, which is close to the average over all users. Which user is the most suspicious (i.e. likely fraudulent)? Our goal is to come up with a principled and intuitive measure of how suspicious each user is.

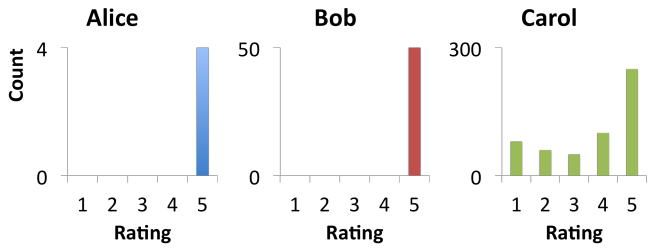


Figure 3: Rating distributions of example users. The histograms show how many times each user gave each star rating.

Why does Alice's low rating count makes her less suspicious than Bob? Our answer is: since we only have 4 products rated by Alice, we have little information about her true (i.e. long-term) rating behavior. She may simply be a normal user who appears unusual as her first few ratings were high, but given more ratings, she would converge to a more typical distribution. Bob, however, is much less likely to be a normal user: we can say with greater certainty that his true rating behavior is anomalous.

Intuitively, deciding how suspicious each user is involves a two-step process: first, we estimate our beliefs for what that user's true rating distribution is. Second, we estimate how suspicious we believe they are, given our beliefs. For Alice, our beliefs are highly uncertain: we cannot be confident that her rating distribution is unusual. For Bob, we are confident that his rating distribution is fairly skewed toward 5s. For Carol, we know her rating distribution with high confidence, but it is not suspicious.

The Bayesian approach applies this intuition in a principled manner. It first sets a prior, estimated from data, representing our 'default' beliefs about users's rating behavior. It then estimates our beliefs (in the form of a posterior distribution) about their rating distribution. Finally, we compute how suspicious we believe them to be, averaging over their posterior distribution. Figure 4 illustrates how posterior distributions capture the information we need to identify a user as suspicious. The posterior distributions in Figure 4 refer to our beliefs about each user's true long-term average rating, expressed as a probability distribution. The point estimates refer to each user's observed average rating, which do not capture how much more certain we are in Bob's case than Alice, and hence how much more suspicious Bob is.

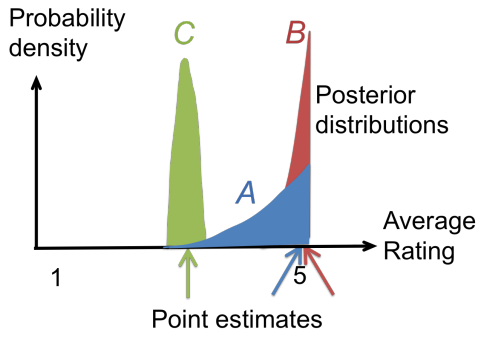


Figure 4: **Posterior distributions, not point estimates, mark a user as suspicious.** Bob is suspicious because our beliefs about his true average rating are both narrow and close to 5, while Alice is less suspicious because our beliefs about her true average rating are more spread out.

Alternatives that don't work (z or t tests) What about instead performing a standard hypothesis test (such as a z or t-test) for each user's average rating (or any other quantity associated with their rating distribution), to see whether their average rating differs significantly from the population? The problem with this approach lies with users like Carol, who differ slightly from the population but have a large number of ratings. Even as normal (non-fraudulent) users, we expect their true average rating to differ slightly from that of the population (say, by 0.1) just due to inter-person variation.

Given enough ratings, however, even such a small difference could produce arbitrarily small p-values under such a hypothesis test, since the test correctly concludes that there is an extremely small probability of drawing Carol's observed average rating if her true average rating were equal to that of the population. However, such small differences are not suspicious. The Bayesian approach would instead estimate Carol's posterior distribution as in Figure 4 and conclude that it is both narrow and entirely non-suspicious, which is a more

sensible result.

3.2 Proposed Model Table 1 summarizes the notation used in this paper.

Table 1: Commonly used notation in this paper. Vectors are in **bold**.

Parameter	Interpretation
m	No. of users
n_i	No. of ratings given by user i
s	No. of star levels (e.g. $s = 5$ for 1 to 5 stars)
x_{ij}	Rating of the jth rating given by user i
b	base of logarithm for temporal bucketing
Δ_{ij}	temporal bucket of the jth rating by user i
Δ_{max}	temporal bucket with highest index
$\mathbf{x_i}, \boldsymbol{\Delta_i}$	Vector $(x_{ij})_{j=1}^{n_i}$ (resp. $\Delta_{ij})_{j=1}^{n_i}$)
X, Δ	Matrix containing all the (x_{ij}) (resp. (Δ_{ij}))
n_{il}^x, n_{il}^Δ	No. of times user i gave rating (resp. time) l
$\mathbf{n}_i^x, \mathbf{n}_i^\Delta$	Vector $((n_{i1}^x), \ldots, (n_{is}^x))$ (resp. $((n_{i1}^\Delta), \ldots)$)
K	No. of clusters
π_k	Probability of a random user being in cluster k
z_i	Cluster (or mixture component) of user i
$\mathbf{p_i}, \mathbf{q_i}$	User i's rating (resp. temporal) distribution
$\boldsymbol{\alpha_k}, \boldsymbol{\beta_k}$	Dirichlet parameters for cluster k
F_x, F_Δ	Global distributions; refer to (4.5)

In our problem setting, users are indexed $i = 1, \ldots, m$. User i has n_i ratings, indexed by $j = 1, \ldots, n_i$. The ratings in stars given by user i are denoted by the variables $x_{ij} \in \{1, 2, \ldots, s\}$ (e.g. for star ratings from 1 to 5 we have $s = 5$). Similarly to [6], we preprocess the rating timestamps by computing its time difference from the previous rating, i.e. the difference between its timestamp and the timestamp of the last rating given by the same user. We then bucket the time differences according to the integer part of the log base b, where b is chosen to result in close to 20 buckets. The temporal bucket of the jth rating of user i is denoted $\Delta_{ij} \in \{1, 2, \ldots, \Delta_{max}\}$ for $j = 1, \ldots, n_i$, analogous to x_{ij}.

Using time differences instead of raw timestamps makes it possible to detect either unusually rapid rating of products by a user (due to having a concentration of small time differences), or unusually regular patterns, such as rating products once every hour. Both of these patterns suggest bot-like or spammy behavior, which we would like to detect. Moreover, the discretized i.e. multinomial approach allows us to flexibly detect a wide range of possible deviations from normal behavior without assuming a more restrictive parametric form, such as a Gaussian distribution.

We will consider the ratings X and time differences Δ to be generated based on a model. From a high level, our generative model for user behavior is a mixture model in which each user belongs to one of K clusters: in general, there is no single type of user behavior, so we use clusters to capture different types of user behavior. Each cluster represents a certain type of rating distribution and temporal distribution for the users in that cluster.

Let $k = 1, \ldots, K$ index into the K clusters. For each user i, we first generate which cluster they belong to, $z_i \in \{1, 2, \ldots, K\}$, from a Multinomial($\boldsymbol{\pi}$) distribution, where π_k, the kth entry of $\boldsymbol{\pi}$, is the probability that a random user is generated in cluster k.

Even within a single cluster, it would not be reasonable to expect all users to behave exactly the same way. Thus, instead of using a single rating/temporal distribution per cluster, we allow small deviations per user. We do this by associating a common Dirichlet prior with each cluster: each user has their individual rating distribution drawn from this prior. We denote user i's rating distribution by $\mathbf{p_i}$, a vector of length s of nonnegative entries which sums to 1, where the jth entry of this vector gives their probability of giving the jth rating. Thus, we draw user i's rating distribution $\mathbf{p_i}$ from a Dirichlet($\boldsymbol{\alpha}_{z_i}$). Similarly, $\mathbf{q_i}$ represents user i's temporal distribution, and we draw $\mathbf{q_i} \sim$ Dirichlet($\boldsymbol{\beta}_{z_i}$).

Finally, to generate user i's ratings, we draw each rating x_{ij} based on user i's rating distribution: $x_{ij} \sim$ Multinomial($\mathbf{p_i}$). Similarly, for the temporal buckets, we draw each Δ_{ij} from a Multinomial($\mathbf{q_i}$) distribution.

The generative model we have described is summarized in (3.1).

$$
\begin{aligned}
z_i &\sim \text{Discrete}(\boldsymbol{\pi}) \\
\mathbf{p_i} | z_i = k &\sim \text{Dirichlet}(\boldsymbol{\alpha}_k) \\
x_{ij} &\sim \text{Multinomial}(n_i, \mathbf{p_i}) \\
\mathbf{q_i} | z_i = k &\sim \text{Dirichlet}(\boldsymbol{\beta}_k) \\
\Delta_{ij} &\sim \text{Multinomial}(n_i, \mathbf{q_i})
\end{aligned} \tag{3.1}
$$

The corresponding graphical model is given in Figure 5.

4 Proposed Algorithms

4.1 Fitting our Bayesian Model (BIRD) Algorithm 1 fits data to the model of Fig.5, by using a greedy hill climbing approach to maximize the overall likelihood function. In this algorithm, we iteratively adjust each parameter and the cluster assignments z until convergence. Each of the $\arg\max$ lines in the algorithm can be solved efficiently, which we next describe how to do.

Cluster parameters

Here we fix z and compute $\arg\max_{\boldsymbol{\alpha}_k} P(X, \Delta | \boldsymbol{\alpha}_k, z)$ in Line 10; adjusting with respect to $\boldsymbol{\beta}$ will be similar. Note that adjusting $\boldsymbol{\alpha}_k$ only affects the likelihood with respect to x_i, for i in cluster k. Thus we are equivalently maximizing $\prod_{i:z_i=k} P(x_i | \boldsymbol{\alpha}_k, z)$.

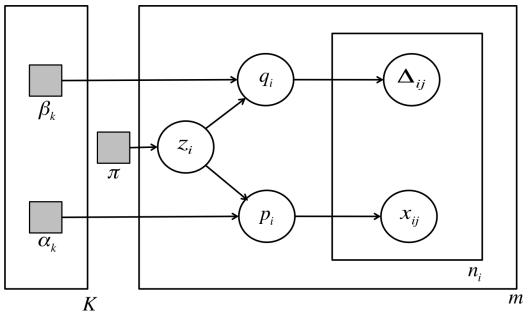


Figure 5: Graphical model describing users, ratings and rating times. User i's mixture component z_i determines how we generate their individual multinomial parameter vectors p_i, q_i, which then generate x_{ij} and Δ_{ij} as samples from these multinomial distributions.

To be clear, here $P(x_i|\boldsymbol{\alpha}_k, z)$ refers to the marginal likelihood, i.e. the probability of generating x_i, after marginalizing out $\mathbf{p_i}$. Thus we need to find the maximum likelihood update for $\boldsymbol{\alpha}_k$ given the x_i for i in cluster k, which were sampled from the two-step process of first generating $\mathbf{p_i} \sim \text{Dirichlet}(\boldsymbol{\alpha}_k)$ and then generating $x_i \sim \text{Multinomial}(\mathbf{p_i})$. This two-step process is also known as the Dirichlet-multinomial distribution; [18] provide fixed-point iteration methods for maximum likelihood estimation of $\boldsymbol{\alpha}_k$ in this setting. Specifically, we repeat until convergence, for each $k = 1, \ldots, K$ and $l = 1, \ldots, s$:

$$\alpha_{kl}^{new} = \alpha_{kl} \frac{\sum_{i=1}^{m} \frac{n_{il}^x}{n_{il}^x - 1 + \alpha_{kl}}}{\sum_{i=1}^{m} \frac{n_i^x}{n_i^x - 1 + \sum_{l'} \alpha_{kl'}}} \tag{4.2}$$

Similarly, the update for β is:

$$\beta_{kl}^{new} = \beta_{kl} \frac{\sum_{i=1}^{m} \frac{n_{il}^\Delta}{n_{il}^\Delta - 1 + \beta_{kl}}}{\sum_{i=1}^{m} \frac{n_i^\Delta}{n_i^\Delta - 1 + \sum_{l'} \beta_{kl'}}} \tag{4.3}$$

Cluster assignments In Line 15, we fix the cluster parameters and fit the maximum likelihood cluster assignment z_i. Note that changing z_i only affects the likelihood with respect to user i. Referring to our graphical model in Figure 5, maximizing $P(X, \Delta|z_i = k)$ is equivalent to finding:

$$z_i = \arg\max_k \pi_k P(\mathbf{x_i}|z_i = k) P(\boldsymbol{\Delta_i}|z_i = k) \tag{4.4}$$

To compute $P(\mathbf{x_i}|z_i = k)$, note that this is the probability of drawing $\mathbf{x_i}$ from a Dirichlet-multinomial distribution with known parameter $\boldsymbol{\alpha}_k$.

Let $n_{il}^x = \sum_{j=1}^{n_i} 1\{x_{ij} = l\}$ be the number of user i's ratings that equal l, and similarly $n_{il}^\Delta = \sum_{j=1}^{n_i} 1\{\Delta_{ij} = l\}$. The marginal distribution of a Dirichlet-multinomial distribution (after marginalizing out $\mathbf{p_i}$) is known to be

$$P(x_i|z_i = k) = \frac{\Gamma(A_k)}{\Gamma(n_i + A_k)} \prod_{l=1}^{s} \frac{\Gamma(n_{il}^x + \alpha_{kl})}{\Gamma(\alpha_{kl})}$$

where Γ is the gamma function, and $A_k = \sum_l \alpha_{kl}$. The term $P(\boldsymbol{\Delta_i}|z_i = k)$ can be computed in the same manner. Since z_i is discrete, we can thus maximize (4.4) by computing $\pi_k P(\mathbf{x_i}|z_i = k) P(\boldsymbol{\Delta_i}|z_i = k)$ for each value of k and choosing the maximizing value of k.

Posterior distributions of p and q Here we explain how to compute the posterior distributions in Line 18 of Algorithm 1. Let $\mathbf{n}_i^x = ((n_{i1}^x), \ldots, (n_{is}^x))$ and $\mathbf{n}_i^\Delta = ((n_{i1}^\Delta), \ldots, (n_{is}^\Delta))$. At this point the entire iterative process of estimating the hyperparameters and cluster assignments is complete, and we have to compute the posterior distributions of $\mathbf{p_i}$ and $\mathbf{q_i}$ given the data X and Δ. $\mathbf{p_i}$ has a Dirichlet($\boldsymbol{\alpha}_{z_i}$) prior, so by the conjugate prior property of Dirichlet distributions, its posterior distribution is Dirichlet($\alpha_{z_i} + \mathbf{n}_i^x$). Similarly, the posterior distribution of β is Dirichlet($\beta_{z_i} + \mathbf{n}_i^\Delta$).

Number of clusters We select the number of clusters K using the Bayesian Information Criterion (BIC).

Convergence As we can see from Algorithm 1, each adjustment to $\boldsymbol{\pi}, \alpha, \beta$ or z is an $\arg\max$ step and increases the overall likelihood $P(X, \Delta|z; \boldsymbol{\pi}, \alpha, \beta)$. Because the overall likelihood is bounded, this must converge.

4.2 NEST: Proposed Metric for Detecting Suspicious Users

Algorithm 1 gives us the posterior distributions $P(\mathbf{p_i}|x_i, \Delta_i)$ and $P(\mathbf{q_i}|x_i, \Delta_i)$ for the user parameters. In this section, we propose a suspiciousness metric, NEST (Normalized Expected Surprise Total). Recalling Figure 4, the overall idea is to compute user i's suspiciousness, averaged over their posterior distribution.

We will compute suspiciousness with respect to rating and temporally, then normalize and combine them to ensure that each has equal influence. This is a practically motivated decision that ensures that even in settings where one of the variables has a much finer resolution than the other (i.e. it is bucketized into more buckets), neither variable will dominate the other in determining suspiciousness.

We now explain how to compute user i's suspiciousness is in terms of their ratings distribution; the same formulas directly apply to the temporal distribution, and we explain how to combine the scores in (4.7).

Global Distribution Recall that our Bayesian model BIRD gives us an estimate for the distribution underlying the rating behavior of all users, in the form of a mixture of Dirichlet($\boldsymbol{\alpha}_k$) distributions with mixture coefficients $\boldsymbol{\pi}_k$ (and similarly, mixture of Dirichlet($\boldsymbol{\beta}_k$) distributions for temporal

Algorithm 1 Fitting parameters for the model in Figure 5. X is a matrix containing all the x_{ij} and Δ is a matrix containing all the Δ_{ij}.

procedure BIRD (X, Δ)
  Output:
  cluster hyperparameters $\boldsymbol{\pi}, (\boldsymbol{\alpha}_k, \boldsymbol{\beta}_k)_{k=1}^K$
  posterior distributions for each user's rating and temporal distribution $P(\mathbf{p_i}), P(\mathbf{q_i})$
  while not converged **do**
    Adjust cluster proportions $\boldsymbol{\pi}$
    $\pi_k = \sum_{i=1}^m 1\{z_i = k\}/m$
    for $k = 1, \ldots, K$ **do**
      Adjust cluster hyperparameters $\boldsymbol{\alpha}_k, \boldsymbol{\beta}_k$
      $\boldsymbol{\alpha}_k^{new} = \arg\max_{\boldsymbol{\alpha}_k} P(X, \Delta | \boldsymbol{\alpha}_k, z)$ (4.2)
      $\boldsymbol{\beta}_k^{new} = \arg\max_{\boldsymbol{\beta}_k} P(X, \Delta | \boldsymbol{\beta}_k, z)$ (4.3)
    end for
    for $i = 1, \ldots, m$ **do**
      Adjust users' assignments to clusters:
      $z_i^{new} = \arg\max_k P(X, \Delta | z_i = k)$ (4.4)
    end for
  end while
  Compute user posterior distributions:
  $P(\mathbf{p_i}|X, \Delta) = \text{Dirichlet}(\boldsymbol{\alpha}_{z_i} + \mathbf{n}_i^x)$
  $P(\mathbf{q_i}|X, \Delta) = \text{Dirichlet}(\boldsymbol{\beta}_{z_i} + \mathbf{n}_i^\Delta)$
end procedure

distributions). Denote this global distribution by F_x (resp. F_Δ):

F_x can be thought of as our estimate for the distribution of $\mathbf{p_i}$ in general over all users ($\mathbf{p_i}$ is the true rating distribution of user i).

$$F_x(\mathbf{p}) = \sum_{k=1}^{K} \boldsymbol{\pi}_k \text{ Dirichlet}(\mathbf{p}; \boldsymbol{\alpha}_k) \tag{4.5}$$

where $\text{Dirichlet}(\mathbf{p}; \boldsymbol{\alpha}_k)$ refers to the probability of generating $\mathbf{p}$ under a $\text{Dirichlet}(\boldsymbol{\alpha}_k)$ distribution.

Surprise Denote $\tilde{\mathbf{p}}_\mathbf{i} := P(\mathbf{p_i}|x_i, \Delta_i)$, the posterior distribution of $\mathbf{p_i}$ given the data. To be clear, observe that $\tilde{\mathbf{p}}_\mathbf{i}$ is a distribution over multinomial vectors. Recall that we estimate $\tilde{\mathbf{p}}_\mathbf{i}$ as part of BIRD: $\tilde{\mathbf{p}}_\mathbf{i}$ is a $\text{Dirichlet}(\boldsymbol{\alpha}_{z_i} + \mathbf{n}_i^x)$ distribution. $\tilde{\mathbf{p}}_\mathbf{i}$ represents our beliefs about user i's rating distribution.

For the sake of intuition, imagine $\tilde{\mathbf{p}}_\mathbf{i}$ was a point mass, i.e. we had perfect knowledge of user i's rating distribution: assume that it consists of a point mass at $\mathbf{p}$. Recall that the posterior distribution $\tilde{\mathbf{p}}_\mathbf{i}$ is a distribution over multinomial vectors, so $\mathbf{p}$ here is a multinomial vector. Then user i's suspiciousness could be calculated as *surprise* or negative log likelihood under the global distribution F_x evaluated at the rating distribution $\mathbf{p}$:

$$\text{surprise}(\mathbf{p}) = -\log F_x(\mathbf{p})$$

The less likely $\mathbf{p}$ is, the more suspicious the user is. This makes sense because F_x is our estimate for the global distribution from which all the usersare drawn from; the lower the log-likelihood of $\mathbf{p}$, the more anomalous user i is when compared to this distribution. Thus, we use *surprise* to estimate the suspiciousness of a rating distribution $\mathbf{p}$.

Expected Surprise Now returning to the general case, when $\tilde{\mathbf{p}}_\mathbf{i}$ is a posterior distribution. In this case, we compute the average over $\tilde{\mathbf{p}}_\mathbf{i}$ of the *surprise* $-\log F_x(\mathbf{p})$: that is, now $\mathbf{p}$ is drawn at random from this posterior distribution $\tilde{\mathbf{p}}_\mathbf{i}$. Averaging the *surprise* gives us the posterior mean (or 'Bayes estimate') of user i's suspiciousness, which can be regarded as our 'best estimate' of user i's suspiciousness given our knowledge of them.[1] Thus, we use *expected surprise* to estimate the suspiciousness of a user based on their posterior distribution $\tilde{\mathbf{p}}_\mathbf{i}$.

DEFINITION 1. (EXPECTED SURPRISE)
The expected surprise for user i measures how surprising user i rating distribution is averaged over its posterior distribution, and is given by:

$$s_x(i) = -\mathbb{E}_{\mathbf{p} \sim \tilde{\mathbf{p}}_\mathbf{i}} \log F_x(\mathbf{p}) \tag{4.6}$$

The expected surprise $s_\Delta(i)$ with respect to the temporal distribution is computed similarly.

As discussed, $-\mathbb{E}_{\mathbf{p} \sim \tilde{\mathbf{p}}_\mathbf{i}} \log F_x(\mathbf{p})$ measures the expected suspiciousness of a sample rating distribution drawn at random from the posterior distribution $\tilde{\mathbf{p}}_\mathbf{i}$, where suspiciousness of a single rating distribution is given by its *surprise* or negative log likelihood under F_x.

Normalized Expected Surprise Total (NEST) In our dataset, we use both ratings and temporal data. Using ratings data, we compute posterior distribution $\tilde{\mathbf{p}}_\mathbf{i}$ and the resulting *expected surprise* $s_x(i)$; using temporal data similarly gives us posterior distribution $\tilde{\mathbf{q}}_\mathbf{i}$ and *expected surprise* $s_\Delta(i)$. To combine these, we could simply add them; however, if one had a larger range of possible values than the other, the one with the largest range could end up dominating the sum. To give both terms comparable influence, we normalize them by their respective standard deviations. Let $\sigma_x = \text{std.dev}(s_x(1), \ldots, s_x(m))$ and σ_Δ be defined analogously. Then NEST is defined as:

DEFINITION 2. (NEST)
NEST *measures how jointly suspicious user i is based on*

[1] The posterior mean of a parameter is also known as the minimum mean square error estimator, as it minimizes expected least squares loss. It has desirable properties such as consistency under fairly general conditions, and is widely used in practice. [15]

his or her ratings and temporally, and is given by:

$$\text{NEST}(i) = \frac{s_x(i)}{\sigma_x} + \frac{s_\Delta(i)}{\sigma_\Delta} \tag{4.7}$$

Note that there is no need to normalize s_x and s_Δ additively (e.g. by subtracting the mean scores) since that would simply shift all the scores by the same amount.

Computing NEST Fitting BIRD gives us the posterior distribution for user i's rating distribution $\tilde{\mathbf{p}}_\mathbf{i} = P(\mathbf{p_i}|X, \Delta)$ (Line 19 in Algorithm 1); we get $\tilde{\mathbf{q}}_\mathbf{i} = P(\mathbf{q_i}|X, \Delta)$ similarly. We also know the full global distribution $F_x(\mathbf{p}) = \sum_{k=1}^{K} \boldsymbol{\pi}_k \text{ Dirichlet}(\mathbf{p}; \boldsymbol{\alpha}_k)$. Hence, we can compute the expected surprise $s_x(i) = -\mathbb{E}_{\mathbf{p}\sim\tilde{\mathbf{p}}_\mathbf{i}} \log F_x(\mathbf{p})$ by taking a fixed number of samples from $\tilde{\mathbf{p}}_\mathbf{i}$ and repeatedly evaluating their log-likelihood under F_x. We can then compute the expected surprise values $s_\Delta(\cdot)$ and combining the two as shown in (4.7).

5 Experiments

We conducted experiments to answer the following questions: **Q1. Effectiveness on real data**: does BIRDNEST catch fraud on real data? **Q2. Scalability**: does it scale to large datasets? **Q3: Interpretability**: can the results of the Bayesian model BIRD and the scores given by NEST be interpreted in a real-life setting?

We implemented BIRDNEST in Python; all experiments were carried out on a 2.4 GHz Intel Core i5 Macbook Pro, 16 GB RAM, running OS X 10.9.5. The code is available for download at `www.andrew.cmu.edu/user/bhooi/ratings.tar`. We test BIRDNEST on a variety of real world datasets: table 2 offers details on the datasets we used.

Table 2: Datasets used.

Dataset	**# of users**	**# of products**	**# of ratings**
`Flipkart`	1.1M	550K	3.3M
`SWM` [1]	0.97M	15K	1.1M

5.1 Q1: Effectiveness

Evaluation on `Flipkart` data `Flipkart` is an online e-commerce platform on which merchants sell products to customers, on which customers review products from 1 to 5 stars. We applied BIRDNEST to detect the 250 most suspicious users and provided them to Flipkart; these accounts were investigated and hand-labelled by Flipkart, finding that 211 users of the top 250 flagged by BIRDNEST were involved in fraud. Figure 1b shows the algorithm's precision at k: for various values of k up to 250: note that precision for the most suspicious users is very high: e.g. precision of 1.0 for the first 50 users. These are substantial findings for Flipkart. One common pattern that the domain-experts found was that most of the users labeled as fraudulent are either spamming 4/5 star ratings to multiple products from a single seller (boosting seller's ratings), or spamming 1/2 star ratings to multiple products from another seller (defaming the competition).

Figure 2 plots the averaged rating distributions of users within each group: that is, for each user we computed their frequency of giving each rating from 1 to 5; Figures 2 and 1a takes the average of the rating distributions for users in the corresponding group. Examining the detected users in Figure 2, a common pattern we find is that they consist of extreme polarized rating distributions as well as temporal distributions. The detected users consist of highly negative users (who give only 1 ratings) and highly positive users (who give mostly 5 ratings, with a relatively small fraction of 4s). Similarly, Figure 1a shows the common pattern that detected users contain much shorter temporal differences than normal users.

Evaluation on `SWM` data The `SWM` datasets consists of software product (app) reviews. The dataset was collected by [1] by crawling all the app reviews in the entertainment category from an anonymous online app store. Each review consists of review text as well as a rating from 1 star to 5 stars.

We find clear evidence of fake reviews in the dataset: for example, the most suspicious user posted a block of 27 reviews for the same app all within the span of less than a week, all five-star ratings with near-identical review title and text, as shown in Table 3. Moreover, the review text shows clear signs of being a fake review: they advertise a code associated with an app: typically, users advertise such codes because they give some benefit to the owner when new users download the app via one of these codes.

Table 3: **BIRDNEST detects clearly fake reviews in the `SWM` data:** Example of a 5-star review by the user flagged as most suspicious by BIRDNEST. 27 such near-identical reviews were present for the same app (only trivial differences between them were present, such as the number of dollars signs). All 10 of the top 10 user accounts flagged by BIRDNEST contain similar advertisements for codes.

AWESOMEApp4FreeMoney!!! $$$$$$
All first time users will need a CODE after downloading this app. So download it now and use my CODE for bonus points. CODE: ...

Aside from this block of reviews, almost all of this

user's reviews also consist of similar blocks of repeated text advertising the code. In fact, all 10 of the top suspicious user accounts flagged by BIRDNEST contain advertisements for codes in similar contexts, often accompanied with promises of free cash, points and gift cards.

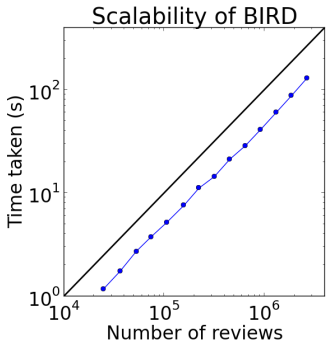


Figure 6: **BIRDNEST is fast and scalable:** running on our 1.1M user, 3.3M ratings dataset in around 2 minutes. BIRDNEST shows linear growth in computation time.

5.2 Q2: Scalability Assume that we run $(\#it)$ iterations of the outer loop of Algorithm 1, let m be the number of users, and K the number of clusters. In each iteration, adjusting $\boldsymbol{\pi}$ takes $O(m)$ time; adjusting each $\boldsymbol{\alpha}_i$, $\boldsymbol{\beta}_i$ and z_i all take $O(K)$ time. Hence, the algorithm takes $O((\#it)mK)$ time, which is linear. Figure 6 shows that the algorithm is fast and its computation time grows linearly in practice.

5.3 Q3: Interpretability In Section 3.1, we motivated the Bayesian approach by giving three users, Alice, Bob and Carol. We explained how the Bayesian approach captures our intuitions about these users through posterior distributions. We now use real data from Flipkart to verify that BIRDNEST indeed conforms to the intuitions that motivated this approach, and that the posterior distributions from BIRDNEST are interpretable and useful in real-life settings.

We selected 3 real Flipkart users: `alice`, `bob` and `carol` (names changed to maintain anonymity). They were chosen to match the rating frequencies of Alice, Bob and Carol in 3.1. We computed each user's posterior distribution of their true rating distribution using BIRDNEST. From this we computed the posterior distribution of their true, long-term average rating, by simulating $10,000$ draws of $\mathbf{p_i}$ from their Dirichlet posterior distribution. We display these with their NEST values in Figure 7.

Agreeing with intuition, both `alice` and `carol` are nonsuspicious, while `bob` is very suspicious, as indicated by the NEST scores: `alice` is ranked around $189,000$th most suspicious, `bob` is ranked around 800th, and `carol` is ranked around $10,000$th, out of the 1.1 million users. The NEST scores given by our algorithm are interpretable as *expected surprise* values, i.e. they are in units of log-likelihood, so a unit difference (after normalization) represents an exponential increase in likelihood. As such, we see from this example that BIRDNEST conforms to intuition in the way it uses posterior distributions to measure uncertainty. Moreover, the posterior distributions of average rating or other quantities can be plotted via simulation and used for further understanding and investigation.

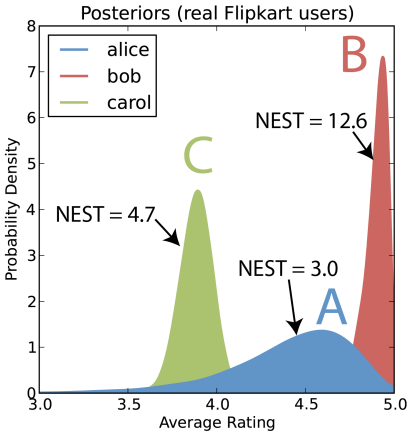


Figure 7: **BIRDNEST is interpretable and agrees with intuition:** the posterior distributions for users capture both the location and certainty about a user's true rating distribution. Note that `bob`'s NEST is highest as his entire posterior distribution is extreme (far from other users).

6 Conclusion

In this paper, we developed BIRD, a Bayesian inference approach for ratings data, and NEST, a principled likelihood-based suspiciousness metric for fraud detection. Our method provides a principled way to combine rating and temporal information to detect rating fraud, and to find a tradeoff between users with extreme rating distributions vs. users with larger number of ratings. Our contributions are:

- **Theoretically sound user behavior model**: we define a Bayesian model for the data based on a mixture model which captures different types of user behavior. This model then allows us to determine how much an anomalous user deviates from normal behavior.
- **Suspiciousness metric**: we define a likelihood-based metric which measures how much a user deviates from normal behavior.
- **Algorithm**: we propose a scalable and effective algorithm for learning the Bayesian model and evaluating suspiciousness.
- **Effectiveness**: we show that our method successfully spots review fraud in large, real-world graphs, with precision of over 84% for the top 250 Flipkart users

flagged by our algorithm.

References

[1] L. Akoglu, R. Chandy, and C. Faloutsos. Opinion fraud detection in online reviews by network effects. *ICWSM*, 13:2–11, 2013.

[2] A. Beutel, K. Murray, C. Faloutsos, and A. J. Smola. Cobafi: collaborative bayesian filtering. In *Proceedings of the 23rd international conference on World wide web*, pages 97–108. ACM, 2014.

[3] A. Beutel, W. Xu, V. Guruswami, C. Palow, and C. Faloutsos. Copycatch: stopping group attacks by spotting lockstep behavior in social networks. In *Proceedings of the 22nd international conference on World Wide Web*, pages 119–130, 2013.

[4] H. Cheng, P.-N. Tan, C. Potter, and S. A. Klooster. Detection and characterization of anomalies in multivariate time series. In *SDM*, pages 413–424. SIAM, 2009.

[5] S. Feng, R. Banerjee, and Y. Choi. Syntactic stylometry for deception detection. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 171–175. Association for Computational Linguistics, 2012.

[6] A. Ferraz Costa, Y. Yamaguchi, A. Juci Machado Traina, C. Traina Jr, and C. Faloutsos. Rsc: Mining and modeling temporal activity in social media. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 269–278. ACM, 2015.

[7] S. Günnemann, N. Günnemann, and C. Faloutsos. Detecting anomalies in dynamic rating data: A robust probabilistic model for rating evolution. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 841–850. ACM, 2014.

[8] M. Hu and B. Liu. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM, 2004.

[9] M. Jiang, A. Beutel, P. Cui, B. Hooi, S. Yang, and C. Faloutsos. A general suspiciousness metric for dense blocks in multimodal data. In *Data Mining (ICDM), 2015 IEEE International Conference on*. IEEE, 2015.

[10] M. Jiang, P. Cui, A. Beutel, C. Faloutsos, and S. Yang. Inferring strange behavior from connectivity pattern in social networks. In *Advances in Knowledge Discovery and Data Mining*, pages 126–138. Springer, 2014.

[11] N. Jindal and B. Liu. Opinion spam and analysis. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, pages 219–230. ACM, 2008.

[12] N. Jindal, B. Liu, and E.-P. Lim. Finding unusual review patterns using unexpected rules. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1549–1552. ACM, 2010.

[13] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434. ACM, 2008.

[14] Y. Koren. Collaborative filtering with temporal dynamics. *Communications of the ACM*, 53(4):89–97, 2010.

[15] E. L. Lehmann and G. Casella. *Theory of point estimation*, volume 31. Springer Science & Business Media, 1998.

[16] X. Li and J. Han. Mining approximate top-k subspace anomalies in multi-dimensional time-series data. In *Proceedings of the 33rd international conference on Very large data bases*, pages 447–458. VLDB Endowment, 2007.

[17] E.-P. Lim, V.-A. Nguyen, N. Jindal, B. Liu, and H. W. Lauw. Detecting product review spammers using rating behaviors. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 939–948. ACM, 2010.

[18] T. Minka. Estimating a dirichlet distribution, 2000.

[19] M. Ott, Y. Choi, C. Cardie, and J. T. Hancock. Finding deceptive opinion spam by any stretch of the imagination. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 309–319. Association for Computational Linguistics, 2011.

[20] S. Pandit, D. H. Chau, S. Wang, and C. Faloutsos. Netprobe: a fast and scalable system for fraud detection in online auction networks. In *Proceedings of the 16th international conference on World Wide Web*, pages 201–210. ACM, 2007.

[21] B. Prakash, M. Seshadri, A. Sridharan, S. Machiraju, and C. Faloutsos. Eigenspokes: Surprising patterns and community structure in large graphs. *PAKDD, 2010a*, 84, 2010.

[22] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. In *ACM SIGMOD Record*, volume 29, pages 427–438. ACM, 2000.

[23] R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *Proceedings of the 25th international conference on Machine learning*, pages 880–887. ACM, 2008.

[24] N. Shah, A. Beutel, B. Gallagher, and C. Faloutsos. Spotting suspicious link behavior with fbox: an adversarial perspective. In *Data Mining (ICDM), 2014 IEEE International Conference on*, pages 959–964. IEEE, 2014.

[25] N. Shah, A. Beutel, B. Hooi, L. Akoglu, S. Gunnemann, Makhija, M. Kumar, and C. Faloutsos. Edgecentric: Anomaly detection in edge-attributed networks. *arXiv preprint*, 2015.

[26] A. Vahdatpour and M. Sarrafzadeh. Unsupervised discovery of abnormal activity occurrences in multi-dimensional time series, with applications in wearable systems. In *SDM*, pages 641–652. SIAM, 2010.

[27] G. Wang, S. Xie, B. Liu, and P. S. Yu. Review graph based online store review spammer detection. In *Data mining (icdm), 2011 ieee 11th international conference on*, pages 1242–1247. IEEE, 2011.

[28] S. Xie, G. Wang, S. Lin, and P. S. Yu. Review spam detection via temporal pattern discovery. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 823–831. ACM, 2012.

[29] J. Ye and L. Akoglu. Discovering opinion spammer groups by network footprints. In *Machine Learning and Knowledge Discovery in Databases*, pages 267–282. Springer, 2015.