Outlier Detection in Sparse Data with Factorization Machines

Mengxiao Zhu[1,2], Charu C. Aggarwal[3], Shuai Ma[1,2,*], Hui Zhang[1,2], Jinpeng Huai[1,2]
[1] SKLSDE Lab, Beihang University, China
[2] Beijing Advanced Innovation Center for Big Data and Brain Computing, China
[3] IBM T. J. Watson Research Center, New York, USA
{zhumx, mashuai, zhanghui, huaijp}@buaa.edu.cn charu@us.ibm.com

ABSTRACT

In sparse data, a large fraction of the entries take on zero values. Some examples of sparse data include short text snippets (such as tweets in Twitter) or some feature representations of categorical data sets with a large number of values, in which traditional methods for outlier detection typically fail because of the difficulty of computing distances. To address this, it is important to use the latent relations between such values. Factorization machines represent a natural methodology for this, and are naturally designed for the massive-domain setting because of their emphasis on sparse data sets. In this study, we propose an outlier detection approach for sparse data with factorization machines. Factorization machines are also efficient due to their linear complexity in the number of non-zero values. In fact, because of their efficiency, they can even be extended to traditional settings for numerical data by an appropriate feature engineering effort. We show that our approach is both effective and efficient for sparse categorical, short text and numerical data by an extensive experimental study.

KEYWORDS

outlier detection; sparse data; factorization machine

1 INTRODUCTION

The problem of outlier detection has been studied extensively in the literature because of its numerous applications in a variety of domains such as intrusion detection, fraud detection, and fault detection [2]. An overview of various algorithms for outlier detection may be found in [2, 15]. Many of the well-known methods for outlier detection use distance-based methods [8, 14, 23, 32]. In addition local normalization [14] is used by some of these methods. Another class of methods [36] uses the Mahalanobis distance to the centroid as the outlier score of a given data point.

These methods are mostly effective for conventional numerical data. It is also possible to extend some of these methods to binary and categorical data in a limited way. However, in many numerical and binary domains, the data is extremely sparse, and most attributes take on zero values. Furthermore, many categorical domains have a *massive-domain* property according to which the number of distinct values of an attribute is very large; this is an indirect form of sparsity when the categorical data is converted to binary form. In such domains it is generally more difficult to robustly compute distances between pairs of records. In order to explain this point, we will discuss the specific examples of sparse massive-domain categorical data and short textual documents. We will discuss how these different domains are not very amenable to the use of traditional outlier detection methods.

Massive-domain Categorical Data. In many domains, the categorical attributes might be drawn from massive domains; in other words, each attribute takes on one of a very large number of values. For example, consider a telecommunication application in which two of the attributes are source and destination IP-addresses, and a third attribute corresponds to one of a hundred different intrusion types. The number of possible IP-addresses may be more than 10^6, and the number of possible intrusions is 100 although some of them might be related. As a result, simple distance measures like the hamming distance are not very effective in these domains. Given a pair of records, it is unlikely that even a single pair of attribute values will be the same between them. On the other hand, the hamming distance does not use the implicit correlations between the various attribute values, which makes it fail in the sparse setting.

One can convert a categorical data set to a binary form by using a binarization process in which each categorical set of values is converted to a set of binary attributes. For example, for the source IP-address, we could create as many binary attributes as the number of possible IP-addresses. Exactly one of these attributes will take on the value of 1 (depending on the value of the source IP-address), and all the other binary attributes will take on the value of 0. One could apply a similar process to the destination IP-addresses and the intrusion type. Unfortunately, this process will greatly expand the number of attributes, although the representation will be sparse because most attributes will take on zero values. Straightforward distance measures will be inadequate because they fail to account for the implicit relationships between different IP-addresses and intrusion types.*Many common data domains that use social network user identifiers, names, addresses, email addresses and URLs can be categorized within this setting.*

Short Text Snippets. In recent years, the problem of mining short text snippets has gained increasing importance because of the greater prevalence of short text segments in social media, chat forums, and so on. It is often notoriously difficult to find outliers in such data sets because of the challenges of using a small number of words to infer the anomalous characteristics of a data point. Clearly, the co-occurrence of certain semantic groups of words might be unusual; however, it would not be reflected in straightforward distance measures that are commonly used in outlier detection algorithms.

CIKM'17, November 6–10, 2017, Singapore.
© 2017 ACM. ISBN 978-1-4503-4918-5/17/11…$15.00
DOI: http://dx.doi.org/10.1145/3132847.3132987

1.1 Challenges in Sparse Data

There are several challenges of performing outlier detection in such sparse data sets. One of the most important challenges is that the use of straightforward distances between points does not capture the true semantic distances because of underlying inter-attribute dependencies. Consider an example of basic information for movies in which the attributes correspond to actor, actress, and movie type. It may be the case that actor Bob and actress Alice often star in movies with the same type. As a result, a future movie that Alice and Bob starred may not be surprising enough to qualify as an outlier. A straightforward use of the hamming distance or the cosine distance will typically not be able to capture such similarities. For example, if actor Bob and actress Alice are correlated in terms of their starred movie type, then there is an inherent similarity between the two distinct attribute values "Bob" and "Alice", even though the hamming distance between these two distinct attribute values in the binarized form is always 2.

In such data sets, traditional methods fail because of the difficulty in computing distances between the data records. These types of similarities can be inferred only from the *aggregate statistical characteristics* of the data sets, because they can capture the relationships between various attribute values in an implicit way. Factorization machines provide the ability to construct any non-linear manifold implied by a polynomial kernel by using the aggregate statistical structure of the data; this provides a model of the normal data set. The nonlinearity of the manifold structure is particularly important in the sparse setting. For example, one can learn the similarity between "*Alice*" and "*Bob*" only by examining their co-occurrence with other attribute values; this implies that we need to learn at least second-order (i.e., quadratic) interactions between attributes to model the manifold structure. This also distinguishes the method from linear dimensionality reduction methods such as PLSA [2] or PCA [36], which are known not to work very well in the extremely sparse settings of massive-domain data or short text snippets. Although it is particularly difficult to learn such manifolds (especially nonlinear ones) in very sparse data sets, factorization machines are able to achieve this goal by learning the latent structure of a large parameter space rather than learning the parameters directly. Once the manifold has been learned, we are able to use the distance of a point from the manifold as an outlier *score*. Larger values of the score are more indicative of outlierness.

Figure 1 shows a toy example of such an outlier with only three attributes (in a relatively dense data set), although it is far more difficult to pictorially represent these settings in sparse data sets. This is because the sparse domains discussed in this paper can have millions of attributes, most of the attribute values are 0s once they have been converted into binary forms, and the relations between pairs of attributes are increasingly difficult to infer. For example, it is extremely uncommon for *both* of a pair of attributes to take on the value of 1. As a result, the covariance structure of the data set (which is crucial for methods like PCA) often contains little discriminative information for analysis. Although it is possible to use such methods for *moderately* sparse domains, an increasing level of sparsity eventually leads to the failure of such methods, e.g., it has been shown experimentally that off-the-shelf dimensionality reduction methods do not work very well for short text snippets with increasing level of sparsity unless they are aggregated into larger documents [20].

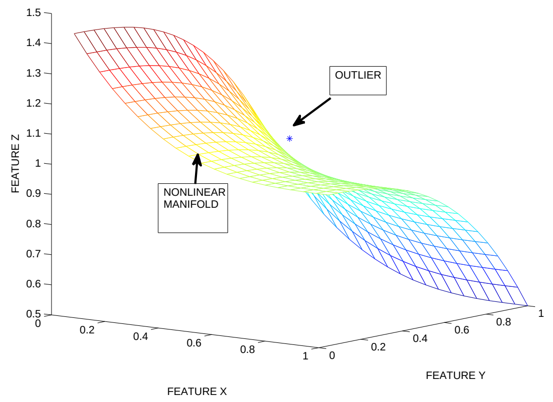


Figure 1: A nonlinear manifold and an associated outlier. Accurate estimation of such manifolds is challenging in high-dimensional and sparse data because of the overfitting caused by an increased number of optimization parameters.

In such settings, the *indirect* relationships between pairs of attributes (in terms of their co-occurrences with other attribute values) is crucial in inferring their similarity. This is the reason that inferring higher-order relationships between attributes is particularly important in these settings. Factorization machines [33, 34] provide a natural approach for inferring such higher-order relationships without causing the overfitting inherent in a large parameter space. Although we restrict our approach to second-order factorization machines in this paper, it is possible, in principle, to use even higher-order factorization machines to capture more complex forms of the latent structure of the underlying data set.

1.2 Merits of Factorization Machines

Although factorization machines have been used earlier in the context of recommender systems, they have been known to have the potential to solve other data mining problems such as classification in sparse domains [3, 33, 34]. This work provides a new application of factorization machines by creating a model of normal data with the approach. Data points that deviate from this normal model are reported as outliers. For sparse domains, with a small number of non-zero entries, this approach provides a robust methodology to perform outlier detection, which is generally difficult with conventional methods. Furthermore, the approach can also be used for conventional data sets by using feature engineering on the original data to create a sparse representation. The main advantage of the approach over distance-based methods is that of *computational complexity*; whereas distance-based methods typically have quadratic complexity, which means that even a few hundred thousand points in the data set can make such methods intractable, factorization machines have a linear complexity in the number of non-zero values in the sparse representation [33, 34]. Therefore, aside from the dual advantages of effectiveness and efficiency in sparse data, they provide significant efficiency advantages in the conventional numerical setting while providing competitive or better effectiveness results to other methods. We will experimentally show the advantages of this technique over competing methods in Section 4.

1.3 Related Work

The problem of outlier detection has been widely studied in the research community because of its numerous applications such as intrusion detection, financial fraud detection, and fault detection [2, 15]. Many existing outlier detection methods [4–10, 12–14, 16, 18, 19, 21–25, 27–32, 35, 36, 38–43] have been proposed, and can be classified in term of the data domains, such as numerical, categorical and text data. (a) Many of the known methods for numerical data use distance-based methods [8, 9, 14, 23, 27, 31, 32, 35, 36] in order to determine outliers, and some are also extended to the high-dimensional setting [4, 24, 25, 28, 30]. (b) Different from numerical data, categorical data is inherently unordered, which makes it hard to assign the similarity between different attribute values. In order to solve this problem, several methods are designed, including combining multiple categorical similarity measures [12], pattern-based methods [6, 38], probability tests [16] and coupled based random walk [29]. (c) In the context of text data, distance-based methods [5, 7, 13] and a variety of probabilistic models [10, 19, 41, 43] are common for the noise removal or for the detection of interesting anomalies [2]. (d) outlier detection has studied for time-series [22] and graph data [18, 21, 39, 42]. (e) In addition, image outliers have also been investigated with matrix factorization [40]. Although these methods are effective for many multi-dimensional data sets, most of them do not work well for sparse data because of the latent relationships between the attribute values, which can be inferred only from the overall statistical structure of the underlying data. Most of these methods are only effective for a specific data domain, and cannot apply to various data domains in a unified way. Further, different from matrix factorization for two categorical variables, factorization machines include a nested parallel factor analysis model for multi-categorical variables [33].

In recent years, factorization machines [33, 34] have been used for effective prediction in recommender systems [3]. Although the primary applicability of factorization machines today is in the context of recommender systems, it has been shown [33, 34] that these methods can also be used for sparse classification and regression. (a) In this paper, we further show that *factorization machines can also be useful in the unsupervised setting.* (b) In particular, we show that *factorization machines can also be used for outlier detection.* (c) The approach can be applied not only to sparse data sets, but reasonable results can even be obtained with *conventional numerical data sets* with a suitable *feature engineering effort.* The results of this paper have the potential to further broaden the applicability of factorization machines to various data domains.

Organization. This paper is organized as follows. Section 2 sets up the problem definition and the factorization model for outlier detection in sparse data. The gradient descent steps for model computation are discussed in Section 3. The experimental results are presented in Section 4, followed by conclusions in Section 5.

2 FACTORIZATION MACHINES FOR OUTLIER DETECTION

In this section, we describe the factorization machine model used for outlier detection. First, we will set up the notations and definitions. We assume that we have a data set $\mathcal{D}$ with n records and d dimensions. The d-dimensional records are denoted by $\overline{Z_1} \dots \overline{Z_n}$, and each record $\overline{Z_i}$ is a vector with d dimensions denoted by $(z_{i1} \dots z_{id})$. Note that this vector is typically created after a feature engineering effort on the original data. The nature of the feature engineering is slightly different, depending on the domain from which the data set is drawn. For example, in the case of a categorical data set, a binary attribute is created for each *value* of a categorical attribute in the original data. Typically, the feature engineering effort converts the original data set to a *sparse and nonnegative* representation. Therefore, the value of d is often very large in real settings, although most of the values of z_{ij} are typically zeros. We will discuss several examples of this feature engineering for various domains in the next section. We will show that several sparse domains, which cannot easily be addressed with traditional models can be handled easily with factorization machines. Furthermore, one can even successfully use the approach for traditional data sets (with numerical attributes) by using sparse transformations, even though the approach is not specifically designed for them.

2.1 Feature Engineering for Various Domains

Factorization machines require various types of feature engineering, depending on the domain at hand. Two particularly useful domains in which factorization machines are particularly useful include massive-domain categorical data, and short text segments. In such cases, there are no known outlier detection techniques that can provide robust results; factorization machines provide a neat way to perform outlier detection in these settings because of their robustness to sparsity. In the following, we provide an overview of the feature engineering effort required for various types of data sets. Note that it is possible for a data set to contain mixed attribute types, in which case the appropriate type of feature engineering is applied to the corresponding attribute. The resulting features are then concatenated to create the overall data set. It is noteworthy that mixed attribute types are notoriously difficult to handle in outlier detection [2]; factorization machines ease this problem by creating a (roughly) homogenous attribute type as a result of the initial feature engineering effort. Therefore, a data set contain massive-domain categorical attributes, short text snippets, as well as numerical attributes will be automatically converted to sparse form as a result of the steps discussed in the following sections.

2.1.1 Massive-Domain Categorical Data. Massive-domain categorical data is defined as a data set in which the number of possible values of a categorical attribute is very large. For example, when an attribute corresponds to a name, an address, or an IP-address, the number of possible values is extremely large. In such cases, it is difficult to meaningfully compute distances between data points with traditional methods. A simple approach to converting these data sets to a sparse representation is the process of *binarization.* For each *value* of each categorical attribute, a new binary attribute is created. The value of binary attribute is either 0 or 1 depending on whether or not that record takes on the categorical value at hand. Therefore, for each categorical attribute in the original data set, a large number of binary attributes will be created, and only a single one of them will take on the value of 1. Therefore, the number of 1s will be exactly equal to the total number values of the categorical attributes in the original data. It is noteworthy that this data type is particularly sparse in massive-domain settings.

2.1.2 Short Text Snippets and Tweets. Short text snippets occur commonly in many real-life settings such as chat forums or tweets. For example, in Twitter, the number of characters in a Tweet is restricted to a a small number of characters, as a result of which the number of keywords is small. In this case, the dimensionality of the data set is equal to the number of distinct keywords, and only the keywords contained in the text snippet have non-zero values. Furthermore, if a particular tweet contains r non-zero keywords, then the corresponding frequency of that attribute is $1/\sqrt{r}$. Therefore, the attribute values are normalized so that the sum of the squares is always equal to one. This is important in this setting because each text snippet might contain a different number of keywords.

2.1.3 Conventional Numerical Data. In conventional numerical data sets, we use the process of *soft discretization.* Soft discretization is a variant of hard discretization in which more information is retained about the values of records. Let σ_i be the standard deviation of the ith attribute and μ_i be the mean. All data records whose values lie in the range $(\mu_i - \sigma_i, \mu_i + \sigma_i)$ lie in one of Φ equi-depth intervals. Each such interval corresponds to a newly created attribute. The value of the attribute is zero, if the data record does not lie in that interval. Otherwise, the value of the attribute is 1. Furthermore, one attribute is created for data points in which the ith attribute is less than $\mu_i - \sigma_i$ and one attribute is created for data points in which the ith attribute is greater than $\mu_i + \sigma_i$. For data records whose ith attribute value x_i is less than $\mu_i - \sigma_i$, the value of the newly created attribute value is set to $|x_i - (\mu_i - \sigma_i)|/\sigma_i$. For data records whose ith attribute value x_i is larger than $\mu_i + \sigma_i$, the newly created attribute value is set to $|x_i - (\mu_i + \sigma_i)|/\sigma_i$. Since at most $(\Phi + 2)$ attributes are created for each numerical attribute in the original data, the total number of attributes increases by a factor of $(\Phi + 2)$, even though most of the values are zero. Therefore, sparsity is retained in this type of data set. The value of Φ is set depending on the size of the data. The value of Φ is set to 100 for any data set containing more than 1000 points; For data sets with less than 1000 points, the value of Φ is set to $\lfloor n/10 \rfloor$, where n is the total number of points in the data set.

2.2 Modeling with Factorization Machines

In this section, we will discuss the modeling approach used for outlier detection with factorization machines. After the feature engineering effort, we have an $n \times d$ matrix $Z = [z_{pq}]$. We assume that all points (i.e. d-dimensional rows of Z) can be modeled to lie on a non-linear manifold of the following form:

$$g + \sum_{i=1}^{d} b_i x_i + \sum_{i=1}^{d} \sum_{j=i+1}^{d} w_{ij} x_i x_j = 0. \tag{1}$$

Here, g is the global bias variable, $(b_1, \ldots, b_d)$ are the dimension-specific biases, and $W = [w_{ij}]$ is a $d \times d$ matrix of coefficients. The variables $(x_1, \ldots, x_d)$ can be instantiated with any of the rows of Z, and it is desired to learn the coefficients, so that Equation (1) is satisfied as closely as possible. Note that Equation (1) is very similar to the condition implied by a second-order polynomial kernel; therefore, the data points are assumed to lie on a nonlinear manifold in the d-dimensional space.

The number of coefficients w_{ij} is d^2. This can be very large in sparse settings because the value of d is typically extremely large after the feature engineering process. As a result, overfitting is extremely likely in sparse settings. In our previous example of actors and actresses, if a particular actress *Alice* and actor *Bob* do not co-occur in the (sparse) data set, it is difficult to meaningfully estimate the coefficient w_{ij} between them. However, factorization machines provide a meaningful technique to estimate such coefficients by imposing a low-rank structure on the coefficient matrix $W = [w_{ij}]$. In particular, each w_{ij} can be expressed as the dot product of two low-rank (i.e., k-dimensional) vectors $\overline{v_i}$ and $\overline{v_j}$ as follows:

$$w_{ij} = \overline{v_i} \cdot \overline{v_j}. \tag{2}$$

Here $\overline{v_i}$ and $\overline{v_j}$ are two k-dimensional vectors of the form $(v_{i1}, \ldots, v_{ik})$ and $(v_{j1}, \ldots, v_{jk})$, respectively. This assumption reduces the number of coefficients from d^2 to $d \cdot k$, which scales linearly with the dimensionality of the data set. As a result, Equation (1) can now be rewritten with the low-rank assumption as follows:

$$g + \sum_{i=1}^{d} b_i x_i + \sum_{i=1}^{d} \sum_{j=i+1}^{d} (\overline{v_i} \cdot \overline{v_j}) x_i x_j = 0. \tag{3}$$

Equation (3) can be viewed as an *unsupervised* avatar of the regression modeling equation used in recommender systems. The main difference is that the right-hand side of Equation (3) is 0 rather than the value of the rating in a recommender system. This is because our goal is to design a model of the normal data points under the (commonly used) assumption that the majority of the points show normal behavior. Points which lie on the non-linear manifold of normal points will (approximately) evaluate to zero, when they are substituted in the left-hand side of Equation (3). Deviations from this model of normal points are tagged as outliers. Therefore, the basic idea is that for any particular d-dimensional instantiation $\overline{Z_p} = (z_{p1} \ldots z_{pd})$ of data points, the outlier score $O(\overline{Z_i})$ is given by substituting the values of $\overline{Z_p}$ in Equation (3) and using the modulus of the scored value:

$$O(\overline{Z_p}) = \mathrm{LQ}(Z_p)|g + \sum_{i=1}^{d} b_i z_{pi} + \sum_{i=1}^{d} \sum_{j=i+1}^{d} (\overline{v_i} \cdot \overline{v_j}) z_{pi} z_{pj}|. \tag{4}$$

Larger values of the score are more indicative of outlier-like behavior. Here $\mathrm{LQ}(Z_p)$ is the quotient of the average number of words in all short text documents divided by the number of words in document Z_p, and $\mathrm{LQ}(Z_p) = 1$ for categorical and numerical data record Z_p. The reason to introduce $\mathrm{LQ}(Z_p)$ for short text data is because the number of words in documents is often different.

The values of the parameters g, b_i and $\overline{v_i}$ $(i \in [1, d])$ are determined by minimizing the mean-squared error of all the data points. Therefore, the objective function J is defined as follows:

$$\text{Minimize } J = \frac{1}{2} \sum_{p=1}^{n} O(\overline{Z_p})^2. \tag{5}$$

The main problem with this model is that it will always lead to the trivial solution in which each g, b_i, and v_{ij} is set to 0. However, this model involves a non-convex optimization problem, and it often obtains a non-trivial local solution [11]. Moreover, even if the trivial solution is obtained under extremely rare cases, we could reset the parameters and re-train the data. Besides that, this model usually has a large number of model parameters, especially when k is large. This makes it prone to over-fitting [33]. To overcome this,

```
Algorithm FMOutlier (Data Set: $\mathcal{D}$);
begin
  Perform feature engineering on data set $\mathcal{D}$ to create sparse
      representation $\overline{Z_1} \dots \overline{Z_n}$;
  Initialize $O(\overline{Z_1}) \dots O(\overline{Z_n})$ to zeros;
  for iterations = 1 to $t$ do
    Randomly partition $\overline{Z_1} \dots \overline{Z_n}$ into $m$ folds $S_1 \dots S_m$;
    for $r = 1$ to $m$ do
      Learn each $\theta \in \{g, b_1 \dots b_d, v_{11} \dots v_{dk}\}$ by solving
          optimization model of Equation (6) using points
          in $\cup_{q \neq r} S_q$ (Procedure LearnParameters in Figure 3);
      For each $\overline{Z_i} \in S_r$, compute $O(\overline{Z_i})$ using Equation (4);
      $O_{tot}(\overline{Z_i}) := O_{tot}(\overline{Z_i}) + O(\overline{Z_i})$;
  return $(O_{tot}(\overline{Z_1}) \dots O_{tot}(\overline{Z_n}))$.
end
```

Figure 2: Unsupervised learning for outlier detection with factorization machines

L2 regularization is applied, the objective function is, therefore, updated as follows:

$$\text{Minimize } J = \frac{1}{2}\sum_{p=1}^{n} O(\overline{Z_p})^2 + \sum_{\theta \in \Theta} \lambda_\theta \theta^2, \text{ such that} \tag{6}$$

$$\sum_{\theta \in \Theta} \lambda_\theta \theta^2 = \lambda_g g^2 + \sum_{i=1}^{d} \lambda_{b_i} b_i^2 + \sum_{i=1}^{d}\sum_{j=1}^{k} \lambda_{v_{ij}} v_{ij}^2. \tag{7}$$

Here (1) Θ is the space consisting of parameters $g, b_1 \dots b_d, v_{11} \dots v_{dk}$, and θ refers to one of them; (2) λ_θ is the regularization value for parameter θ, and we adopt the same value λ for all parameters, which can be different for different parameters.

The parameters of this optimization problem can be learned with gradient descent. This procedure is described in Section 3 in detail together with a pseudo-code. In this section, we focus mainly on the optimization model and overall framework of the solution.

One challenge with the procedure is that when an outlier is scored while including the data point within the data set used for performing the training, overfitting is possible. In other words, we are learning the parameters using $\overline{Z_1} \dots \overline{Z_n}$ and, at the same time, using these learned parameters to score $\overline{Z_1} \dots \overline{Z_n}$. The problem with this approach is that a true outlier might be missed because that point is itself included as a part of the normal data. Although the inclusion of (a small number of) outliers does not significantly affect the scores in general, the effect on the specific outliers included within the training data set can be quite drastic. Therefore, a point should be scored without including it in the data set used to perform the training. This is achieved with the use of cross-validation. The data set is partitioned into m segments. For each segment, the remaining $(m-1)$ segments are used to score the points within the segment. In our implementations, we used $m = 2$, so that one half of the data set is used for computing the outlier scores of the remaining half of the data set. As a result, one can compute an out-of-sample outlier score for each data point. However, the problem with this approach is that the resulting outlier score can be somewhat sensitive to the randomized choice of the partition used. By throwing away half of the points during training, one is not using the full knowledge in the data set for computing the model of the normal data. Therefore, the partitioning of the data into m segments is repeated t times in a randomized way to further improve the robustness of the estimation process. This will result in t different scores of each data point. The scores of each data point from the t different executions are averaged (or summed) in order to create the final outlier score. The overall procedure is illustrated in Figure 2. This type of approach has the additional benefit of variance reduction because of its natural ensemble-centric approach [1].

3 MODEL COMPUTATION

In this section, we will discuss the solution methodology for the factorization machine model. A natural approach to solving this problem is that of gradient-descent. Let θ be a parameter of the parameter optimization problem. Note that θ could be the global bias g, any of the dimension-specific biases $b_1 \dots b_d$, or any of the components v_{ij} of the latent vectors $\overline{v_1} \dots \overline{v_d}$. In order to perform the gradient descent, the partial derivative of the objective function J needs to be computed, in order to make the following gradient-descent steps for each θ.

$$\theta \Leftarrow \theta - \alpha \frac{\partial J}{\partial \theta} \tag{8}$$

Here, $\alpha > 0$ is the step-size, which corresponds to the learning rate. This update is performed for each distinct parameter $\theta \in \{g, b_1 \dots b_d, v_{11} \dots v_{dk}\}$. However, it is often inefficient to compute the gradient direction because each data point contributes to the partial derivative, and therefore one must compute a summation over a large number of data points. When the number of data points is large, this process can be inefficient. A second approach is to use stochastic-gradient descent to speed up the optimization process. In stochastic gradient-descent, the idea is to perform the updates based on the contribution of a *single randomly chosen point in the data.* Therefore, the update step is as follows:

$$\theta \Leftarrow \theta - \alpha \left[\frac{\partial J}{\partial \theta}\right]_{\text{Contributed by } p\text{th data point}}. \tag{9}$$

In this case, one cycles through all the data points in random order, and performs the update based on the aforementioned computation. The term “stochastic gradient descent” derives its name from the fact that the points are processed in stochastic order. The resulting *point-wise* update for the data point $\overline{Z_p}$ can be shown to be as follows:

$$\theta \Leftarrow \theta - \alpha\{\mathsf{LQ}(Z_p)[g + \sum_{i=1}^{d} b_i z_{pi} + \sum_{i=1}^{d}\sum_{j=i+1}^{d} (\overline{v_i} \cdot \overline{v_j}) z_{pi} z_{pj}] \cdot \frac{\partial O(\overline{Z_p})}{\partial \theta} + 2\lambda\theta\}. \tag{10}$$

Here, $\frac{\partial O(\overline{Z_p})}{\partial \theta}$ is the gradient of the right-hand side of Equation (4) with respect to the parameter θ, which is defined as follows:

$$\frac{\partial O(\overline{Z_p})}{\partial \theta} = \mathsf{LQ}(Z_p) \cdot \begin{cases} 1 & \theta \text{ is } g \\ z_{pi} & \theta \text{ is } b_i \\ z_{pi} \sum_{j=1}^{d} v_{js} \cdot z_{pj} - v_{is} \cdot z_{pi}^2 & \theta \text{ is } v_{is}. \end{cases} \tag{11}$$

The overall approach cycles through the training data points one by one in order to learn the parameters. Note that the training data set is constructed using only a subset S of the data set $\overline{Z_1} \dots \overline{Z_n}$,

```
Algorithm LearnParameters (Training Subset: S);
begin
  Randomly initialize $\{g, b_1 \ldots b_d, v_{11} \ldots v_{dk}\}$ to values in $(0, 1)$
  repeat
    Randomly shuffle all data points in $S$;
    for each $\overline{Z_p} \in S$ in shuffled order do
      Update $g$ according to Equation (10)
      for each $i$ such that $z_{pi} \neq 0$ do
        Update $b_i$ according to Equation (10);
        Update each $v_{i1} \ldots v_{ik}$ according to Equation (10);
  until convergence;
end
```

Figure 3: Learning parameters with gradient descent

based on the cross-validation procedure discussed earlier. For any given point in this set S, all the parameters θ do not need to be updated because many of the updates in Equation (10) evaluate to zero, when θ is v_{is}, and the value of z_{pi} is 0. Therefore, one only needs to update those values of v_{is} for which z_{pi} is non-zero. This is important from a computational efficiency perspective. These parameters are used to score the points stored in the test portion of the data set. The process is repeated t times over the various cross-validation folds, and the averaged outlier score is reported.

We make use of AdaGrad [17] in our implementation, a modified stochastic gradient descent algorithm with an adaptive learning-rate that dynamically incorporate knowledge of the geometry of the data observed in earlier iterations to perform more informative gradient-based learning. It has a base learning rate, which is multiplied with a value that is updated after each iteration.

A pseudo-code of the procedure is illustrated in Figure 3. This procedure shows the learning process within a single cross-validation fold of the data where $m - 1$ segments of the data (denoted by S) are used to create the training model. The basic idea is to process the data points in S in random order, as would be required by stochastic gradient-descent. For each processed point, $\overline{Z_p}$, the updates of Equation (10) need to be performed to the various parameters $\theta \in \{g, b_1 \ldots b_d, v_{11} \ldots v_{dk}\}$. However, we need to update only those b_i for which z_{pi} is non-zero, and only those v_{is} for which z_{pi} is non-zero. This is important in the sparse setting, because it leads to improved computational efficiency. The updates are repeated to convergence. Typically, only a small number of rounds of the updates are required in order to achieve high-quality results.

3.1 Analysis of Computational Complexity

Although factorization machines provide the only reasonable method for outlier detection in sparse data sets, they provide the additional benefit of improving the computational complexity over conventional distance-based methods like LOF. This complexity is also an advantage in relatively dense data sets (e.g., conventional numerical data for which factorization machines are not specifically designed). In such cases, one is still able to obtain high-quality results by converting the data sets to the sparse domain, but the scalability for a single cycle of the stochastic gradient descent method requires a time complexity that is linear in the number of non-zero values in the data set, as k is typically a small constant, e.g. 10 and 50 [33, 34].

The update of Equation (10) can be performed in constant time, and the number of times it needs to be performed (in a single cycle of stochastic gradient descent) is exactly equal to the number of non-zero values in the training data. Therefore, the overall complexity of a single cycle of stochastic gradient descent is $O(k \cdot |S| \cdot A)$, where A is the average number of non-zero entries in each data record. For example, the value of A in a short text snippet would be equal to the number of distinct words in the snippet. If T cycles of stochastic gradient descent are required, the complexity is $O(k \cdot |S| \cdot A \cdot T)$ for a training procedure. However, the training procedure needs to be repeated over m cross-validation folds and t iterations. Furthermore, the value of $|S|$ is $n \cdot (m-1)/m$. As a result, the overall computational complexity of FMOutlier is $O(k \cdot n \cdot (m-1) \cdot A \cdot T \cdot t)$.

Note that the value of m is typically 2, and the values of T and t are often less than 5. Therefore, the overall complexity is approximately equal to $k \cdot 25$ times the number of non-zero entries in the entire data set, which provides a *linear* scalability. Techniques like LOF are known to scale quadratically with the number of data points. For data sets of even reasonably large size, such methods may not remain practical, whereas factorization machines provide a computationally viable solution. Therefore, even though the results of this paper are focused on sparse data domains, they can also be used to provide high-quality outliers in conventional data sets, e.g. numerical data, in an efficient way. Even in such cases, we show that factorization machines can often provide competitive results in terms of quality.

4 EXPERIMENTAL STUDY

Using real-life sparse categorical, short text and conventional numerical data, we conducted an extensive experimental study to evaluate the effectiveness and efficiency of our outlier detection method FMOutlier, compared with existing classical and recent competitive methods.

4.1 Experimental Settings

We first introduce the settings of our experimental study.

4.1.1 Data sets. We chose real-life sparse categorical and short text data and conventional numerical data, summarized in Table 1.

(1) Categorical data. (a) Publication is derived from Microsoft Academic Graph data set, a large collection of publications [37]. (b) Food is derived from City of Chicago Data Portal, records inspections of restaurants and other food establishments in Chicago from January 1, 2010 to present (https://data.cityofchicago.org). (c) Imdb is a movie review data set (http://www.imdb.com). For outlier detection, we selected a set of records with categorical attributes and without missing values from these data sets.

As there are no ground-truth outliers in these data sets [16, 26], we adopted the method of Random Shifting as follows. (1) Select shifting attributes. We computed the standard deviation for the occurrence times of values for each attribute, and selected $\lceil 1/2 attributes \rceil$ with higher standard deviations as the shifting attributes; (2) Select alternative values for outliers. For attributes with no more than 100 different values, we manually selected values that occur less obviously than the other values; For attributes with more than 100 different values, we selected values that occur at

Data Sets	Records	Attributes	Distinct Values	Outliers (%)
Publication	44,484	11	203,012	1.12
Food	25,629	11	84,098	1.17
Imdb	29,145	11	178,007	1.37
Tweets	2,510	8.56	5,431	1.51
GNewsT	11,299	6.23	9,213	1.69
GNewsS	11,299	22.19	22,311	1.69
Smtp	95,156	3	207	0.03
Shuttle	45,830	9	328	0.53
Annthyroid	7,200	6	317	7.42

Table 1: Summary of real-life data sets.

most 10 times; (3) Shift values. We randomly selected about 1%-2% records of the original data, and shifted the values of shifting attributes to their alternative values.

(2) Short text data. (a) Tweets [41] consists of 2,472 tweets highly relevant to 89 queries. (b) Google News [41] consists of the titles and snippets of 11,109 news articles about 152 events, which is further divided into TitleSet (GNewsT), SnippetSet (GNewsS) that contain the titles and snippets, respectively. Note that, there is an article without title and an article without snippet, so the size of both GNewsT and GNewsT is 11,108.

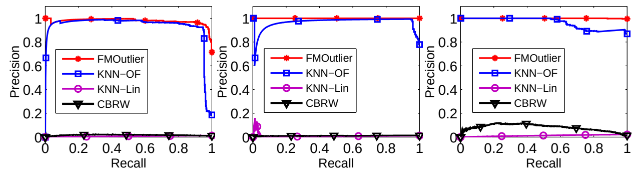
As there are no ground-truth outliers in these data sets, we treated the original data as normal records and manually generated outliers similar to the method in [41]. For Tweets, We generated 38 outliers, each of which has 9 words formed with 2-7 random letters. Similarity, we generated 191 outliers with 6 words for GNewsT and 191 outliers with 22 words for GNewsS, respectively.

(3) Numerical data. (a) Smtp is from Outlier Detection DataSets (ODDS) (http://odds.cs.stonybrook.edu), which has 95,156 3-dimensional records with 30 outliers. (b) Shuttle is from UCI Machine learning repository, and is a 9-dimensional multi-class classification data set (https://archive.ics.uci.edu/ml/datasets.html). Here, the training and test data were combined. Its four smallest classes were combined to form the outlier class. Hence, Shuttle has 45,830 records with 244 outliers. (c) Annthyroid is from ODDS, which has 7,200 6-dimensional records with 534 outliers.

4.1.2 Comparison Algorithms. We have carefully chosen algorithms to compare with our FMOutlier approach.

(1) Categorical data. (a) Several similarity measures for KNN have been proposed to compute the similarity of two categorical data instances [12], among which we chose KNN-Lin and KNN-OF with the best performance. (b) CBRW [29], namely coupled biased random walks, is a very recent outlier detection method for categorical data with a time complexity depending on the number of different values in a data set if the number of different values is larger than the size of the data set.

(2) Short text data. (a) KNN-Text adopts TF-IDF as the representation of text data, and uses the cosine function to measure the similarity [2]. (b) LDA is a popular probability-based method in text clustering [10]. Documents that have a low probability of belonging to their closest cluster are declared as outliers [2]. (c) GSDPMM [41] is a very recent outlier detection method for text data, which is actually the collapsed Gibbs sampling algorithm for the Dirichlet Process Multinomial Mixture model.



(a) Publication (b) Food (c) Imdb

Figure 4: Effectiveness on categorical data: precision-recall curve

Data Sets	KNN-OF	KNN-Lin	CBRW	FMOutlier
Publication	0.9309	0.0072	0.0170	**0.9785**
Food	0.9670	0.0099	0.0114	**1**
Imdb	0.9633	0.0127	0.0812	**0.9991**

Table 2: Effectiveness on categorical data: average precision.

(3) Numerical data. LOF [14] is a widely used outlier detection method for numerical data based on the local density estimation [6].

4.1.3 Implementation. We implemented all algorithms using C++ with no parallelization. Parameters were set as follows. (1) For KNN-OF and KNN-Lin, $K = 10$ for the Kth nearest neighbor by [12]; (2) For CBRW, its damping factor $a = 0.95$ by [29]; (3) For KNN-Text, its $K = 10$; (4) For LDA, its hyper-parameters $a = K/50$ and $\beta = 0.1$ by [41] and , where K is the number of topics; (5) For GSDPMM, its number of topics $K = 1$, hyper-parameters $\beta = 0.02$ and $a = 0.1 * N$ by [41], where N is the number of records; (6) For LOF, its Kth nearest neighbor $K = 5$ by [27]; (7) For FMOutlier, we set $a = 0.01$, $k = 50$, $\lambda = 1$ for categorical data, $a = 0.1$, $k = 50$, $\lambda = 0.2$ for text data, and $a = 0.01$, $k = 10$, $\lambda = 0.2$ for numerical data, respectively, where a is the base learning rate, k is the dimensionality of parameter v, and λ is the regularization parameter. All experiments were conducted on a machine with 2 Intel Xeon E5-2650 2.3GHz CPUs and 64GB of Memory, running 64 bit Windows 7 professional System. All tests were repeated over 3 times and the average is reported here.

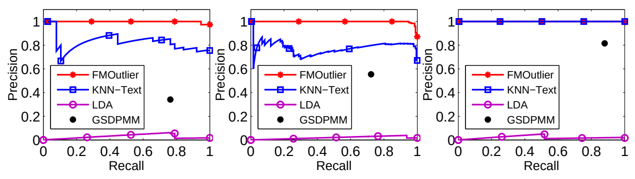
4.2 Experimental Results

We tested the effectiveness and efficiency of FMOutlier on three kinds of datasets, respectively. We next present our findings.

4.2.1 Effectiveness Tests. First, we evaluated the effectiveness of FMOutlier compared with the methods presented above. To measure the effectiveness, we adopted the *average precision*, the average of the precision values across recall levels [6], which is suitable for measuring algorithms that generate a score of the "outlierness" of an record. As FMOutlier is not deterministic, we repeated each test 5 times, and computed the mean average precision. When we plotted the precision-recall curve, the closest results to the average were used; As GSDPMM only checks whether an record is an outlier or not, we just report its precision and recall.

Exp-1.1: Effectiveness on massive-domain categorical data. To evaluate the effectiveness, we varied the recall level of the results, and plotted the precision-recall curves of KNN-OF, KNN-Lin, CBRW and FMOutlier.

The results are reported in Figure 4 and Table 2. When varying the recall level, KNN-Lin and CBRW consistently have lower precision for all data sets, KNN-OF performs unstably as recall level changes, while FMOutlier always has the highest precision for all



(a) Tweets (b) GNewsT (c) GNewsS

Figure 5: Effectiveness on short text data: precision-recall curve

Data Sets	KNN-Text	LDA	GSDPMM	FMOutlier
Tweets	0.8282	0.0287	0.3412	**0.9994**
GNewsT	0.7802	0.0193	0.5542	**0.9980**
GNewsS	**1**	0.0212	0.8155	**1**

Table 3: Effectiveness on short text data: average precision.

recall levels. Indeed, FMOutlier improves the average precision over (KNN-OF, KNN-Lin, CBRW) by (4.76%, 97.13%, 96.15%) on Publication, (3.3%, 99.01%, 98.86%) on Food, (3.58%, 98.64%, 91.79%) on Imdb, on average, respectively.

Exp-1.2: Effectiveness on short text data. To evaluate the effectiveness, we varied the recall level of the results, and plotted the precision-recall curves of KNN-Text, LDA and FMOutlier.

The results are reported in Figure 5 and Table 3. When varying the recall level, LDA consistently has the lowest precision on all data sets, KNN-Text performs unstably on both Tweets and GNewsT, and has a high precision on GNewsS, possibly because GNewsS has more words in one record, and gives more information for KNN-Text to tell outliers. Besides, the performance of GSDPMM is worse than KNN-Text and FMOutlier, while FMOutlier always has the highest precision for all recall levels. Indeed, FMOutlier improves the average precision over (KNN-Text, LDA, GSDPMM) by (17.12%, 97.07%, 65.82%) on Tweets, (21.78%, 97.87%, 44.38%) on GNewsT, (0%, 97.88%, 18.45%) on GNewsS, on average, respectively.

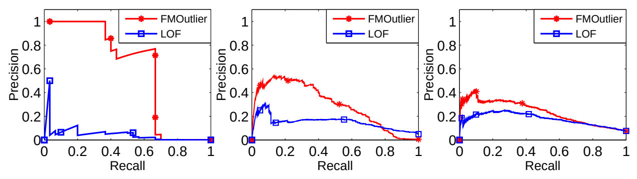
Exp-1.3: Effectiveness on numerical data. To evaluate the effectiveness, we varied the recall level of the results, and plotted the precision-recall curves of LOF and FMOutlier.

The results are reported in Figure 6 and Table 4. When varying the recall level, the precision of LOF increases first and then decreases on three data sets with the increment of recall levels, while FMOutlier keeps the highest precision first, and then decreases on Smtp, and varies in the same way as LOF on Shuttle and Annthyroid, since some normal records show higher outlierness than outliers and some defined outliers may not be true outliers. Moreover, FMOutlier consistently performs better than LOF on Smtp and Annthyroid, and performs better than LOF on Shuttle when recall is less than 0.8. Indeed, FMOutlier improves the average precision over LOF by 54.94% on Smtp, 14.31% on Shuttle, 5.92% on Annthyroid, respectively.

4.2.2 *Efficiency Tests.*

Second, we evaluated the efficiency of our FMOutlier compared with other methods.

Exp-2.1: Impacts of the sizes of data sets. To evaluate the impacts of data sizes for categorical data, following [41], we copied Food five times to construct a big data set called LongFood. We also keep the distribution of different values unchanged by changing values, and the number of different values in LongFood is 5

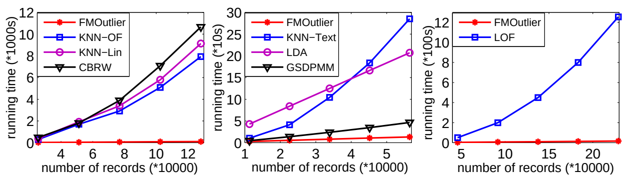


(a) Smtp (b) Shuttle (c) Annthyroid

Figure 6: Effectiveness on numerical data: precision-recall curve

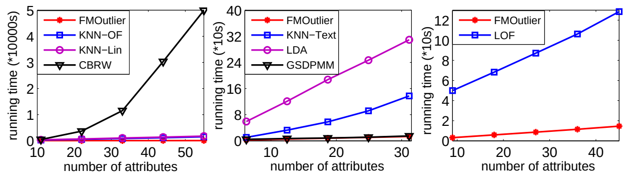
Data Sets	LOF	FMOutlier
Smtp	0.0434	**0.5928**
Shuttle	0.1478	**0.2909**
Annthyroid	0.1756	**0.2348**

Table 4: Effectiveness on numerical data: average precision.



(a) Categorical data (b) Short Text data (c) Numerical data

Figure 7: Efficiency tests: varying the sizes of data sets



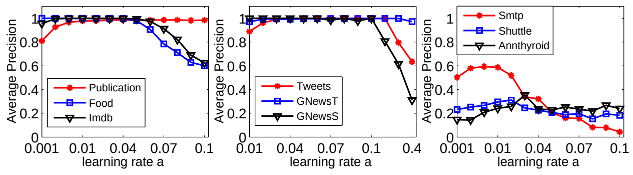
(a) Categorical data (b) Short Text data (c) Numerical data

Figure 8: Efficiency tests: varying the number of attributes

times of Food. When we varied the number of records from 25629 to 25629*5 on LongFood, the running time could be obtained for each approach, and is reported in Figure 7(a). Similarity, we conducted experiments by selecting GNewsT for short text data and Shuttle for numerical data. The corresponding results of GNewsT and Shuttle are reported in Figure 7(b) and Figure 7(c), respectively.

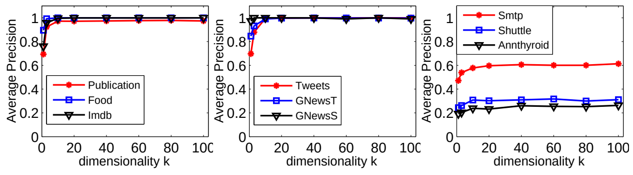
When varying the number of records, the running time of LDA, FMOutlier and GSDPMM increases nearly linearly, and the running time of other algorithms increases nearly quadratically with the increment of the number of records. Moreover, FMOutlier consistently runs faster than all the competitors, especially for larger number of records. Indeed, FMOutlier is on average (48.7, 56.2, 64.3), (12.6, 15.5, 2.8) and (46.4) times faster than (KNN-OF, KNN-Lin, CBRW), (KNN-Text, LDA, GSDPMM) and (LOF), respectively.

Exp-2.2: Impacts of the number of attributes. To evaluate the impacts of the number of attributes, for categorical data, we copied Food horizontally (and changed values) five times to construct a high-dimensional data set, called HDFood. When varied the number of attributes from 11 to 55 on HDFood, the running time could be obtained for each method, and is reported in Figure 8(a). Similarity, we conducted experiments by selecting GNewsT for short text data and Shuttle for numerical data. Their results are reported in Figure 8(b) and Figure 8(c), respectively.



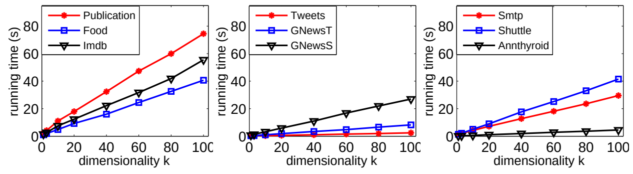
(a) categorical data (b) short text data (c) numerical data

Figure 9: Impacts of a on the effectiveness



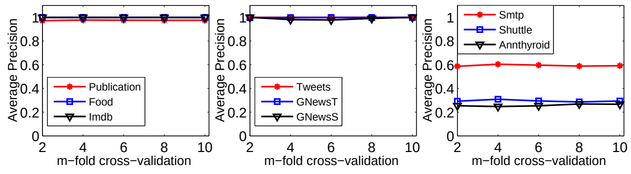
(a) categorical data (b) short text data (c) numerical data

Figure 10: Impacts of k on the effectiveness



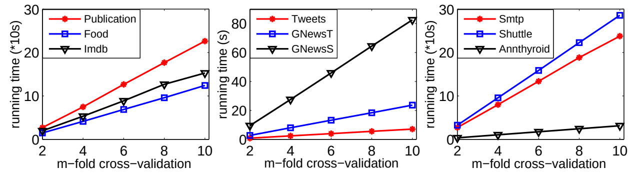
(a) categorical data (b) short text data (c) numerical data

Figure 11: Impacts of k on the efficiency



(a) categorical data (b) short text data (c) numerical data

Figure 12: Impacts of m on the effectiveness



(a) categorical data (b) short text data (c) numerical data

Figure 13: Impacts of m on the efficiency

When varying the number of attributes, the running time of CBRW increases nearly quadratically, while all the other methods increase nearly linearly with the increment of the number of attributes. Moreover, FMOutlier consistently runs faster than all the competitors. Indeed, FMOutlier is on average (17.3, 21.1, 271.1), (7.2, 22.8, 1.2) and (11.1) times faster than (KNN-OF, KNN-Lin, CBRW), (KNN-Text, LDA, GSDPMM) and (LOF), respectively.

4.2.3 Impacts of parameters on effectiveness and efficiency. Finally, we evaluated the impacts of the learning rate a, dimensionality k, number m of cross-validation folds and regularization parameter λ. We fixed these parameters to their default values by default, and tested the average precision and running time.

Exp-3.1: Impacts of learning rate a. To evaluate the impacts of the learning rate a, we varied a from 0.001 to 0.4. The results of the average precision are reported in Figure 9. Note that a has no impacts on efficiency as the number of iterations is fixed.

When varying a, the average precision of FMOutlier first increases, keeps almost stable after that, and, finally, decreases on all data sets. The results tell us that the selection of a does have certain impacts on the convergence of FMOutlier, such that larger a might lead to oscillation and smaller a might result in trapping in bad solutions. However, as shown in the results, FMOutlier is quite robust to the selection of a, as long as a is within a certain range on all data sets. Note that we set a to (0.01, 0.1, 0.01) for (categorical, text, numerical) data by default, respectively, and the value of a can be adjusted reasonably for other data sets.

Exp-3.2: Impacts of dimensionality k. To evaluate the impacts of the dimensionality k, we varied k from 1 to 100. The results of the average precision and running time are reported in Figures 10 and 11, respectively.

When varying k, the average precision of FMOutlier increases rapidly when k is less than 10, increases slowly when k is less than 50, and keeps almost stable after that on all data sets. To guarantee the effectiveness of FMOutlier, it suffices to set k to 50 on all kinds of data sets. Indeed, the change of the average precision when varying k from 50 to 100 is only (-0.10%, -0.29%, 0.23%) on (categorical, text, numerical) data on average, respectively. When varying k, the running time of FMOutlier increases nearly linearly with the increment of k on all data sets. Indeed, FMOutlier is very efficient, and could finish all the tests in 74.50 seconds when k was 100 on all data sets. Note that the value of k can be further reduced to 10 for achieving a better efficiency at the price of a small loss of the average precision.

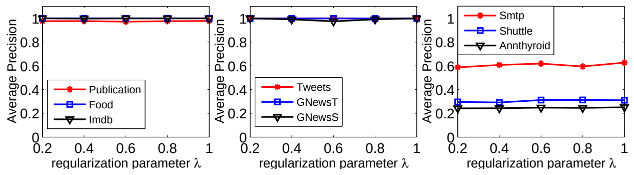
Exp-3.3: Impacts of number m of cross-validation folds. In our approach, we fix the number m of cross-validation folds to 2. To verify the rationale, we further evaluated the impacts of m by varying its value from 2 to 10. The results of the average precision and running time are reported in Figures 12 and 13.

When varying m, the average precision of FMOutlier keeps almost stable on all data sets. Indeed, the change of the average precision when varying m from 2 to 10 is only (0.07%, 0.04%, 0.61%) for (categorical, text, numerical) data on average, respectively. It is obvious that the running time of FMOutlier increases linearly with the increment of m on all data sets. That is, fixing $m = 2$ is a good choice for FMOutlier.

Exp-3.4: Impacts of regularization parameter λ. To evaluate the impacts of the regularization parameter λ, we varied λ from 0.2 to 1. The results of the average precision are reported in Figure 14. Similar to **Exp-3.1**, λ has no impacts on efficiency.

When varying λ, the average precision of FMOutlier keeps almost stable on all data sets. Indeed, the change of the average precision when varying λ from 0.2 to 1 is only (0.09%, 0.03%, 2.11%) for (categorical, text, numerical) data on average, respectively. This result indicates that FMOutlier is quite robust to λ, and it is easy to select a proper value of λ on all kinds of data sets.

Summary. From these experimental results we find the following. (1) FMOutlier is effective for outlier detection in sparse categorical



(a) categorical data (b) short text data (c) numerical data

Figure 14: Impacts of λ on the effectiveness

and short text data, and even conventional numerical data with a suitable feature engineering effort. Indeed, FMOutlier on average improves the average precision over (KNN-OF, KNN-Lin, CBRW) by (3.88%, 98.26%, 95.60%) for categorical data, over (KNN-Text, LDA, GSDPMM) by (12.97%, 97.61%, 42.88%) for short text data, and over (LOF) by (25.06%) for numerical data, respectively.

(2) FMOutlier is efficient and provides a linear scalability. As data sets grow larger, the advantage of FMOutlier becomes more obvious. Indeed, FMOutlier is on average (48.7, 56.2, 64.3), (12.6, 15.5, 2.8) and (46.4) times faster than (KNN-OF, KNN-Lin,CBRW), (KNN-Text, LDA, GSDPMM) and (LOF), respectively, in our tests. The gap becomes larger as the increment of the sizes of data sets.

(3) FMOutlier introduces the learning rate a, dimensionality k, number m of cross-validation folds, and regularization parameter λ for the sake of practicability and flexibility in real-life applications. However, FMOutlier is robust to these parameters in our tests.

5 CONCLUSIONS

We have proposed an unsupervised outlier detection approach for sparse data with factorization machines. We have also experimentally shown that our approach has the dual advantages of effectiveness and efficiency, and can be applied not only to sparse massive domain categorical and short text data, but reasonable results can even be obtained for conventional numerical data with a suitable feature engineering. The results of this study have the potential to further broaden the applicability of factorization machines.

Although we have shown excellent results with the use of a simple discretization methodology in this study, it is possible that other more sophisticated sparse-coding methods can also be combined with our approach to achieve even better results. This is an interesting topic that deserves a further study.

ACKNOWLEDGMENT

This work is supported in part by NSFC U1636210, 973 Program 2014CB340300, NSFC 61421003 & 61322207, Special Funds of Beijing Municipal Science & Technology Commission. For any correspondence, please refer to Shuai Ma.

REFERENCES

[1] Charu C. Aggarwal. 2012. Outlier ensembles: position paper. *SIGKDD Explorations* 14, 2 (2012), 49–58.
[2] Charu C. Aggarwal. 2013. *Outlier Analysis*. Springer, New York.
[3] Charu C. Aggarwal. 2016. *Recommender Systems - The Textbook*. Springer.
[4] Charu C. Aggarwal and Philip S. Yu. 2001. Outlier Detection for High Dimensional Data. In *SIGMOD*. 37–46.
[5] Charu C. Aggarwal and Philip S. Yu. 2010. On clustering massive text and categorical data streams. *Knowl. Inf. Syst.* 24, 2 (2010), 171–196.
[6] Leman Akoglu, Hanghang Tong, Jilles Vreeken, and Christos Faloutsos. 2012. Fast and reliable anomaly detection in categorical data. In *CIKM*.
[7] James Allan, Ron Papka, and Victor Lavrenko. 1998. On-Line New Event Detection and Tracking. In *SIGIR*.
[8] Fabrizio Angiulli and Clara Pizzuti. 2002. Fast Outlier Detection in High Dimensional Spaces. In *PKDD*.
[9] Kanishka Bhaduri, Bryan L. Matthews, and Chris Giannella. 2011. Algorithms for speeding up distance-based outlier detection. In *KDD*.
[10] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *ACM JMLR* 3 (2003), 993–1022.
[11] Mathieu Blondel, Akinori Fujino, and Naonori Ueda. 2015. Convex Factorization Machines. In *ECML PKDD*.
[12] Shyam Boriah, Varun Chandola, and Vipin Kumar. 2008. Similarity Measures for Categorical Data: A Comparative Evaluation. In *SDM*.
[13] Thorsten Brants and Francine Chen. 2003. A System for new event detection. In *SIGIR*.
[14] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jorg Sander. 2000. LOF: Identifying Density-Based Local Outliers. In *SIGMOD*.
[15] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly detection: A survey. *ACM Comput. Surv.* 41, 3 (2009), 15:1–15:58.
[16] Kaustav Das and Jeff G. Schneider. 2007. Detecting anomalous records in categorical datasets. In *KDD*.
[17] John C. Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *ACM JMLR* 12 (2011), 2121–2159.
[18] Manish Gupta, Jing Gao, and Jiawei Han. 2013. Community Distribution Outlier Detection in Heterogeneous Information Networks. In *ECML PKDD*.
[19] Thomas Hofmann. 1999. Probabilistic Latent Semantic Indexing. In *SIGIR*.
[20] L. Hong and B. Davison. 2010. Empirical study of topic modeling in twitter. In *First Workshop on Social Media Analytics*.
[21] Renjun Hu, Charu C. Aggarwal, Shuai Ma, and Jinpeng Huai. 2016. An embedding approach to anomaly detection. In *ICDE*.
[22] Eamonn J. Keogh, Jessica Lin, and Ada Wai-Chee Fu. 2005. HOT SAX: Efficiently Finding the Most Unusual Time Series Subsequence. In *ICDM*.
[23] Edwin M. Knorr and Raymond T. Ng. 1998. Algorithms for Mining Distance-Based Outliers in Large Datasets. In *VLDB*.
[24] Hans-Peter Kriegel, Matthias Schubert, and Arthur Zimek. 2008. Angle-based outlier detection in high-dimensional data. In *KDD*.
[25] Aleksandar Lazarevic and Vipin Kumar. 2005. Feature bagging for outlier detection. In *KDD*.
[26] Yen-Cheng Lu, Feng Chen, Yating Wang, and Chang-Tien Lu. 2016. Discovering Anomalies on Mixed-Type Data Using a Generalized Student- t Based Approach. *IEEE TKDE* 28, 10 (2016), 2582–2595.
[27] Mario Lucic, Olivier Bachem, and Andreas Krause. 2016. Linear-Time Outlier Detection via Sensitivity. In *IJCAI*.
[28] Emmanuel Müller, Matthias Schiffer, and Thomas Seidl. 2011. Statistical selection of relevant subspace projections for outlier ranking. In *ICDE*.
[29] Guansong Pang, Longbing Cao, and Ling Chen. 2016. Outlier Detection in Complex Categorical Data by Modeling the Feature Value Couplings. In *IJCAI*.
[30] Ninh Pham and Rasmus Pagh. 2012. A near-linear time approximation algorithm for angle-based outlier detection in high-dimensional data. In *KDD*.
[31] Milos Radovanovic, Alexandros Nanopoulos, and Mirjana Ivanovic. 2015. Reverse Nearest Neighbors in Unsupervised Distance-Based Outlier Detection. *IEEE TKDE* 27, 5 (2015), 1369–1382.
[32] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. 2000. Efficient Algorithms for Mining Outliers from Large Data Sets. In *SIGMOD*.
[33] Steffen Rendle. 2012. Factorization Machines with libFM. *ACM TIST* 3, 3 (2012), 57.
[34] Steffen Rendle, Zeno Gantner, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2011. Fast context-aware recommendations with factorization machines. In *SIGIR*.
[35] Saket Sathe and Charu C. Aggarwal. 2016. LODES: Local Density Meets Spectral Outlier Detection. In *SDM*.
[36] M. Shyu, S. Chen, K. Sarinnapakorn, and L. Chang. 2003. A novel anomaly detection scheme based on principal component classifier. In *ICDMW*.
[37] Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-June(Paul) Hsu, and Kuansan Wang. 2015. An Overview of Microsoft Academic Service (MAS) and Applications. In *WWW*.
[38] Koen Smets and Jilles Vreeken. 2011. The Odd One Out: Identifying and Characterising Anomalies. In *SDM*.
[39] Hanghang Tong and Ching-Yung Lin. 2011. Non-Negative Residual Matrix Factorization with Application to Graph Anomaly Detection. In *SDM*.
[40] Liang Xiong, Xi Chen, and Jeff G. Schneider. 2011. Direct Robust Matrix Factorizatoin for Anomaly Detection. In *ICDM*.
[41] Jianhua Yin and Jianyong Wang. 2016. A model-based approach for text clustering with outlier detection. In *ICDE*.
[42] Weiren Yu, Charu C. Aggarwal, Shuai Ma, and Haixun Wang. 2013. On Anomalous Hotspot Discovery in Graph Streams. In *ICDM*.
[43] Jian Zhang, Zoubin Ghahramani, and Yiming Yang. 2004. A Probabilistic Model for Online Document Clustering with Application to Novelty Detection. In *NIPS*.