Chapter 8

Outlier Detection in Categorical, Text, and Mixed Attribute Data

> "We become not a melting pot, but a mosaic. Different people, different beliefs, different yearnings, different hopes, different dreams."–Jimmy Carter

8.1 Introduction

The discussion in the previous chapters has primarily focused on numerical data. However, the setting of numerical data represents a gross oversimplification because categorical attributes are ubiquitous in real-world data. For example, although demographic data may contain quantitative attributes such as the age, most other attributes such as gender, race, and ZIP code are categorical. Data collected from surveys may often contain responses to multiple-choice questions that are categorical. Similarly, many types of data such as the names of people and entities, IP-addresses, and URLs are inherently categorical. In many cases, categorical and numeric attributes are found in the same data set. Such *mixed-attribute data* are often challenging to machine-learning applications because of the difficulties in treating the various types of attributes in a homogeneous and consistent way.

Categorical attributes are also referred to as *discrete* attributes. An important characteristic of such attributes is that the underlying values are inherently unordered. Unlike numerical data, it is often hard to define similarity between different values of the same attribute. For example, if the attribute *Color* takes on the values "Red," "Blue," and "Orange," the semantic distance between "Red" and "Orange" might be less than that between "Red" and "Blue," because the former pair of colors look similar. This situation is common in real-world settings, but it is often not always straightforward to infer such relationships in a data-driven way. These nuances of categorical data create numerous methodological and technical challenges for developing outlier detection algorithms, which are as follows:

- Extreme-value analysis and statistical algorithms are dependent on the use of statistical quantifications such as the mean and standard deviation of numerical values. In

C.C. Aggarwal, *Outlier Analysis*, DOI 10.1007/978-3-319-47578-3_8

the case of categorical data, such statistical quantifications are no longer meaningful.

- Linear models such as principal component methods (PCA) are designed for numerical data. Although the data can be preprocessed into sparse binary representations, the number of entries in the data matrix might be too large and most of the entries might be 0s. This creates computational challenges.
- All proximity-based algorithms are dependent on some notion of distance. However, many proximity quantifications such as the Euclidean distance function are not meaningful or effective for categorical data. Therefore, proximity functions need to be redefined for categorical data in order to enable the use of proximity-based algorithms.
- Many density-based models such as the Parzen-Rosenblatt method and volume-based methods (e.g., LOCI) cannot be easily adapted to categorical data because these methods are implicitly dependent on the notion of distances.

However, all classes of algorithms can be adapted for categorical data with suitable modifications. Because of the importance of similarity computations in outlier detection, this chapter will also review similarity measures for categorical data.

An important observation about categorical data is that it can be transformed into binary data by treating each value of the categorical attribute as a binary attribute. The binary attribute corresponding to the relevant categorical attribute value takes on the value of 1, whereas other binary attributes take on the value of 0. This allows the use of many algorithms that are designed for other sparse (but well-studied) domains such as text data. Furthermore, since binary data is a (rudimentary) special case of numerical data, the existing algorithms for numerical data can also be applied to this case. However, such an approach does not scale well with an increasing number of possible *values* of the categorical attribute. This is because a separate binary field needs to be dedicated to each *distinct* value of the categorical attribute. For an attribute such as the ZIP code, the number of possible values may be on the order of thousands, and therefore thousands of binary attributes may need to be created from a single categorical attribute. Furthermore, since the values of these binary attributes are heavily dependent on one another, it is inefficient to use the binary representation.

For ease in further discussion, the notations for categorical data are introduced here. It is assumed that the d-dimensional data set $\mathcal{D}$ contains N records. When the data set is purely categorical, is assumed that the ith attribute contains n_i possible distinct values. On the other hand, for a mixed-attribute data set, the first d_c attributes are assumed to be categorical, and the remaining are assumed to numerical.

This chapter is organized as follows. Section 8.2 discusses the generalization of probabilistic models to categorical data. Section 8.3 discusses the extension of linear models to categorical data. The extension of proximity-based models to categorical data is studied in section 8.4. The special case of categorical data, corresponding to binary and transaction data, is discussed in section 8.5. The case of text data is studied in section 8.6. Section 8.7 presents the conclusions and summary.

8.2 Extending Probabilistic Models to Categorical Data

Like numeric data, probabilistic outlier detection models for categorical data [578] use generative models. The main difference is that the generating distribution is tailored to cate-

gorical records rather than numerical records. Generative models have the advantage that the complexity of the specific data domain is captured with the choice of the underlying distribution and a specific generative process. Once this generative process has been defined, one can learn the parameters of the corresponding data distribution in a data-driven manner. A specific example of this generative process for numerical data is discussed in section 2.4 of Chapter 2. A *mixture model* is introduced in section 2.4 in which each data point is generated from one of the components of a mixture of Gaussian clusters. The mean and covariance matrix of each Gaussian cluster are then estimated in a data-driven manner.

The case of categorical data is not very different from that of numerical data. Just as a mixture model is defined for numerical data in Chapter 2, we can define a corresponding mixture model for categorical data. The *only* difference is that the probability distribution of each component of this mixture model is now defined for categorical data rather than the Gaussians, which are suited to numerical data. This difference primarily impacts the methodology and specific details of parameter estimation. Nevertheless, the overall framework is almost identical in both cases. Furthermore, such an approach can be easily extended to mixed-attribute data by defining separate probability distributions for the categorical and numerical attributes and combining them in product-wise fashion to create a single multivariate distribution.

Next, we will describe the generative process for categorical data. It is assumed that the mixture contains k components denoted by $\mathcal{G}_1 \ldots \mathcal{G}_k$. Each point in the observed data set is generated from one of k components using the following process:

- Select the mth mixture component with probability α_m for $m \in \{1 \ldots k\}$.
- Generate a data point from $\mathcal{G}_m$.

The observed data points that have low probabilities of being generated by this process (or low *fit* probabilities) are deemed to be outliers.

The values of α_m for $m \in \{1 \ldots k\}$ denote the prior probabilities, and may either be pre-specified to equal values of $1/k$ as a simplification or may be learned in a data-driven manner. Furthermore, the parameters of the distributions of the mixture components need to be learned so as the maximize the log-likelihood of the observed data being generated by the model. Note that the generative process is similar to that discussed in section 2.4 of Chapter 2. The main difference is in the mathematical form of the generative model for the mth cluster (or mixture component) $\mathcal{G}_m$ because these distributions now need to generate categorical points rather than numerical points. A common example of such a distribution is the Bernoulli distribution.

In the *generalized* Bernoulli[1] model, it is assumed that the jth possible categorical value of the ith attribute is generated by cluster m with probability p_{ijm}. The set of all model parameters is (collectively) denoted by the notation Θ.

The expectation-maximization algorithm alternately estimates the generative (assignment) probabilities of points from mixture components (clusters), and also estimates the distribution parameters while fixing assignment probabilities. Consider a d-dimensional data point $\overline{X}$ in which the rth attribute takes on the j_rth possible categorical value of this attribute. Then, the value of the generative probability $g^{m,\Theta}(\overline{X})$ of $\overline{X}$ from the mth mixture

[1]Strictly speaking, Bernoulli models are designed for the case of binary data containing only two possible values for each attribute. The generalized version of the Bernoulli distribution allows more than two outcomes for the random variable. This is particularly well-suited to categorical data with many distinct values.

component is given by the following expression:

$$g^{m,\Theta}(\overline{X}) = \prod_{r=1}^{d} p_{rj_r m} \tag{8.1}$$

Note that the value of $g^{m,\Theta}(\cdot)$ is a discrete probability, unlike the continuous density function $f^{m,\Theta}(\cdot)$ discussed in Chapter 2. Nevertheless, this is the analogous probabilistic quantification for the cluster-specific fit probability. Correspondingly, the posterior probability $P(\mathcal{G}_m|\overline{X},\Theta)$, that the data point $\overline{X}$ is generated by the mth mixture component (cluster) may be estimated as follows:

$$P(\mathcal{G}_m|\overline{X},\Theta) = \frac{\alpha_m \cdot g^{m,\Theta}(\overline{X})}{\sum_{r=1}^{k} \alpha_r \cdot g^{r,\Theta}(\overline{X})} \tag{8.2}$$

This defines the E-step for the categorical scenario. Note that this step provides a soft assignment probability of the data points to the different clusters.

Once the soft-assignment probability is estimated, the probability p_{ijm} is estimated to maximize the log-likelihood fit while treating the assignment probability as fixed. In this case, the maximization of the log-likelihood fit takes on a particularly simple form. While estimating the parameters for cluster m, the *weight* of a record is assumed to be equal to its generative (assignment) probability $P(\mathcal{G}_m|\overline{X},\Theta)$ from mixture component (to cluster) m. The value α_m is estimated in exactly the same way as discussed in Chapter 2. Specifically, α_m is the weighted fraction of records assigned to cluster m on the basis of the soft probabilities computed in the E-step. In the event that the number of data points is small, we add a *Laplacian smoothing* parameter γ to the numerator and $\gamma \cdot k$ to the denominator of this fraction for some small value of $\gamma > 0$. Let $\mathcal{D}_{ij}$ be the subset of points in database $\mathcal{D}$ for which the ith attribute takes on the jth value. For each cluster m, the aggregate *weight* w_{ijm} of the records for which attribute i takes on the jth value in cluster m is estimated as follows:

$$w_{ijm} = \sum_{\overline{X} \in \mathcal{D}_{ij}} P(\mathcal{G}_m|\overline{X},\Theta) \tag{8.3}$$

Then, the value of the probability p_{ijm} may be estimated as ratio of this weight in $\mathcal{D}_{ij}$ to that in the entire database:

$$p_{ijm} = \frac{w_{ijm}}{\sum_{\overline{X} \in \mathcal{D}} P(\mathcal{G}_m|\overline{X},\Theta)} = \frac{\sum_{\overline{X} \in \mathcal{D}_{ij}} P(\mathcal{G}_m|\overline{X},\Theta)}{\sum_{\overline{X} \in \mathcal{D}} P(\mathcal{G}_m|\overline{X},\Theta)} \tag{8.4}$$

In practice, sufficient data may often not be available to estimate the parameters robustly, especially when some of the attribute values may not appear in a cluster (or $w_{ijm} \approx 0$). This can lead to poor parameter estimations. Laplacian smoothing is commonly used in order to address such ill-conditioned probabilities. Let v_i be the number of distinct attribute values for the ith attribute. One can perform Laplacian smoothing by adding a small positive value β to the numerator and a value $v_i \cdot \beta$ to denominator of the aforementioned estimation. The effect of Laplacian smoothing is to bias the estimated probability p_{ijm} towards its prior value of $1/v_i$. This smoothing is especially noticeable in cases where sufficient data is not available to properly estimate p_{ijm}. The smoothed estimation is as follows:

$$p_{ijm} = \frac{\beta + w_{ijm}}{v_i \cdot \beta + \sum_{\overline{X} \in \mathcal{D}} P(\overline{X} \in \mathcal{G}_m|\Theta)} \tag{8.5}$$

This completes the description of the M-step. As in the case of numerical data, the E-step and M-steps are iterated to convergence. Outliers may be declared as data points which either have low assignment probability to their best matching cluster, or have low absolute fit to the generative model. The use of absolute fit as the outlier score is more common, and it may be computed as follows:

$$\text{Score}(\overline{X}) = \sum_{m=1}^{k} \alpha_m \cdot g^{m,\Theta}(\overline{X}) \tag{8.6}$$

Lower values of the score are more indicative of outlierness. Note that this equation can be easily computed, because all the required parameters on the right-hand side have already been estimated by the expectation-maximization method. If desired, extreme value analysis can be used on the outlier scores in order to identify the relevant outliers. However, for reasons discussed in section 6.2.1 of Chapter 6, extreme-value analysis should be applied on the logarithms of the scores rather than the scores themselves. Large negative values would represent outliers.

8.2.1 Modeling Mixed Data

Mixed data is particularly easy to address with probabilistic models in a seamless way because all cluster assignments are evaluated in terms of probabilities. One common problem with the use of mixed data in other models is that of effective *normalization*; it is often difficult to ensure that all attributes are provided equal importance by an unsupervised algorithm in the absence of prior knowledge about the relative importance of attributes. In probabilistic models, all normalization issues that might arise in the context of mixed data sets are automatically addressed because of the use of probabilities to describe the distributions of various types of attributes. This provides a more homogeneous treatment of mixed data sets.

The EM-algorithm is modified by defining the generative probability of each component as a composite function of the different kinds of attributes. For example, the continuous attributes are modeled with a *density function* $f^{m,\Theta}(\overline{X})$, whereas the categorical attributes are modeled with a discrete probability function $g^{m,\Theta}(\overline{X})$. In a scenario containing both categorical and mixed attributes, it is possible to define the following joint density function $h^{m,\Theta}(\overline{X})$, which is a product of these values:

$$h^{m,\Theta}(\overline{X}) = f^{m,\Theta}(\overline{X}) \cdot g^{m,\Theta}(\overline{X}) \tag{8.7}$$

The E-step can then be performed with the use of this joint density function $h^{m,\Theta}(\overline{X})$. The M-step is performed separately on the categorical and numerical attributes in exactly the same way as discussed in Chapter 2 (for numerical attributes) and the discussion above (for categorical attributes).

A specific method for outlier analysis with mixed attribute data sets was proposed by [578]. This technique is an online approach in which temporal discounting is used in order to provide more importance to recent records. It has some differences with the more generic description provided above. For example, the technique in [578] uses the Hellinger scores in order to compute the outlier scores rather than the absolute fit probabilities. Here, a simplified and more general description of probabilistic models is provided. The interested reader may refer to [578] for a description optimized to the online scenario.

8.3 Extending Linear Models to Categorical and Mixed Data

Linear models can be extended to categorical data sets by using the conversion from the categorical data set to the binary scenario. For each categorical attribute, a set of *one-hot* binary dimensions are created, in which only one value is allowed to one corresponding to the relevant attribute value. Since the ith attribute contains n_i possible values, this leads to a data set with dimensionality $\sum_{i=1}^{d} n_i$. This process may lead to a significant expansion of the dimensionality of the data, especially if the numbers of distinct values of many attributes are large.

One problem is that the different attributes may be assigned different importance by such a transformation in an implicit way. This is because the values of n_i vary widely across different attributes as a result of which different attributes are assigned different numbers of dimensions. Furthermore, the frequencies of various attribute values also play a role in their relative presence. For example, a column in which 50% of the binary attributes take on the value of 1, is very different from a column in which only 1% of the attributes take on the value of 1. In such cases, the column that has 50% relative presence is implicitly given greater weight because of the greater variance in its (binary) value. The data set can be normalized by dividing each column by its standard deviation. Consider the jth value of the transformed binary data for which a fraction f_{ij} of the entries take in the value of 1. In such a case, the standard deviation is given by $\sqrt{f_{ij} \cdot (1 - f_{ij})}$. Therefore, one possibility would be to divide that column of the binary data by $\sqrt{f_{ij} \cdot (1 - f_{ij})}$. As in the case of numerical data, this corresponds to a normalization in which the variance of each *derived* binary dimension is set to the same value of 1. Unfortunately, such a normalization will favor attributes with large values of n_i, since their cumulative variance in the principal component matrix will be n_i. In order to account for this, all columns for the ith attribute are divided by $\sqrt{n_i \cdot f_{ij} \cdot (1 - f_{ij})}$. This ensures that the sum of the variances for all the columns corresponding to a particular attribute is equal to 1. The Principal Component Analysis (PCA) method of Chapter 3 can be directly applied to this representation in order to determine the outlying points in the data.

Such an approach can also be used for mixed attribute data sets by adding normalized numerical attributes to this representation. The normalization process for the numerical columns is relatively straightforward. The columns are normalized, so that the variance of each column is one. This ensures that such methods can be used over a diverse data set with mixed attributes without giving too much importance to only one of the attributes. Principal component analysis methods are particularly effective for sparse binary data sets because they can represent the data in a very small number of components.

After the principal components have been estimated, a similar methodology to that discussed in Chapter 3 can be used for outlier modeling. The deviations along the principal components are computed, and the sum of the squares of these values is modeled as a χ^2 distribution with d degrees of freedom. The extreme-values among the different point-specific deviations are reported as the outliers.

8.3.1 Leveraging Supervised Regression Models

It is also possible to use dependent variable modeling (section 3.2.1 of Chapter 3) in which one attribute is fixed as the predicted attribute and other attributes are used to model it. The root-mean squared (RMSE) error of the modeling provides an attribute-specific outlier

score. This approach is repeated with each predicted attribute in turn, and the scores of each data point are averaged over different columnwise predictors. Such an approach is discussed in detail in section 7.7 of Chapter 7, and its extension to mixed-attribute data is discussed in section 7.7.3.

8.4 Extending Proximity Models to Categorical Data

The design of effective similarity measures for categorical data is critical for the effective implementation of proximity-based algorithms such as k-nearest neighbor methods. Although the design of similarity measures for categorical data is a vast research area in its own right, this chapter will introduce the more common similarity measures that are used for outlier detection. For the case of categorical data, it is generally more natural to study similarities rather than distances because many of the measures are naturally based on matching discrete values [93].

Consider two data points $\overline{X} = (x_1 \ldots x_d)$ and $\overline{Y} = (y_1 \ldots y_d)$. Then, the similarity between the points $\overline{X}$ and $\overline{Y}$ is the sum of the similarities on the individual attribute values. In other words, if $S(x_i, y_i)$ denotes the similarity between the attributes values x_i and y_i, then the overall similarity is defined as follows:

$$Sim(\overline{X}, \overline{Y}) = \sum_{i=1}^{d} S(x_i, y_i) \tag{8.8}$$

This similarity measure is therefore dependent on the specific instantiation of $S(x_i, y_i)$ that is used in the aforementioned definition.

The simplest possible option is to fix $S(x_i, y_i)$ to 1 when $x_i = y_i$ and 0, otherwise. This instantiation of $S(x_i, y_i)$ is referred to as the *overlap* measure. This measure is statistically naive but popular because of its simplicity. Its main problem is that it does not account for the relative frequencies of different attributes. For example, a match between two rare attribute values (e.g., two patients with cancer) is statistically more significant than a match between two frequent attributes (e.g., two normal patients). This type of effect is particularly important in the context of applications like outlier analysis where uneven frequency distribution of attribute values might be indicative of abnormal characteristics.

There are two primary classes of methods for measuring similarity in categorical data:

- The aggregate statistical properties of the data (i.e., statistical frequencies of the attributes) can be used to enable better similarity computations [93]. For example, matches between rare attribute values are considered more significant, whereas mismatches between rare attributes are considered less significant.

- The statistical neighborhoods of data points are used to compute similarity. This approach implicitly models the inter-attribute correlations in the neighborhood of a point for similarity computations. For example, if the colors "Red" and "Orange" co-occur more frequently in the neighborhood of a point than the colors "Red" and "Blue," it means that "Red" and "Orange" are semantically more similar than "Red" and "Blue" in that neighborhood. Of course, since the definition of a neighborhood is itself based on similarity, there is an inherent circularity in this type of definition. A typical approach is to initialize the computation with a simple method such as the *overlap* measure, which is subsequently followed by iterative refinement. An example of such a measure is provided in [154]. This type of approach has the advantage that it can capture semantic relationships between related attribute values.

The second approach for similarity computation is clearly much richer because it captures latent relationships between attribute values. However, it is also computationally more intensive. This section will discuss both types of methods.

8.4.1 Aggregate Statistical Similarity

In the context of categorical data, the *aggregate statistical properties* of the data set should be used in computing similarity. For example, consider a case in which an attribute takes on the value of "Normal" 99% of the time, and the value of "Abnormal" 1% of the time. In such a case, the similarity between frequent attribute values should be given less weight than infrequent attribute values. This principle forms the basis of many well-known normalization techniques such as the *Inverse Document Frequency (IDF)* in the information retrieval domain. Mismatches between attribute values should also be weighted differently based on the frequencies of the underlying attribute values.

Dissimilarity between x_i and y_i is more likely when the number n_i of the distinct values of the ith attribute is large or when x_i and y_i are rare. Correspondingly, the *Eskin* measure modifies the overlap measure by using the same value of $S(x_i, y_i) = 1$ when $x_i = y_i$, but also using a non-zero similarity value of $n_i^2/(n_i^2 + 2)$ when $x_i \neq y_i$. Instead of using the number of distinct values of the ith attribute, one could directly use the raw frequency of x_i and y_i in penalizing mismatches between rare values to a smaller degree. Such an alternative is the *Inverse Occurrence Frequency (IOF)* measure, which uses the same value of 1 for matching values and a value of $1/(1 + \log[f(x_i)] \cdot \log[f(y_i)])$, otherwise. Here, $f(x_i)$ and $f(y_i)$ are the raw numbers of records taking on the values of x_i and y_i, respectively, for the ith attribute.

Matches between rare attribute values can be given greater weight. The inverse occurrence frequency discussed above uses the rarity only in mismatches but not in matches. It is also possible to use the inverse occurrence frequency in matches, which makes the computation similar to text data. Let $p_i(x)$ be the fraction of records in which the ith attribute takes on the value of x in the data set. Thus, when $x_i = y_i$, the similarity is equal to $[\log(1/p_i(x_i))]^2$ and 0, otherwise. This type of measure is commonly used in the information retrieval domain to compute similarities in text data [472].

Another similarity computation method that uses the principle of assigning greater weight to (matching) infrequent values is the *Goodall* measure. The original method proposed in [222] is slow. Therefore, the work in [93] uses a number of heuristic approximations that capture the spirit of this measure in more efficient ways. One variant of the *Goodall* measure uses $1 - p_i(x_i)^2$ as the similarity on the ith attribute, when $x_i = y_i$, and 0, otherwise. This particular (efficient and simplified) variant of the *Goodall* measure is referred to as *Goodall3* [93]. In some variants of the measure, multivariate dependencies between different attribute values are utilized in the final computation.

An important observation is that many of the above measures are intuitively similar in the broad principle of weighting different attribute values differently. The major differences lie in how this broad principle is applied by selecting a specific function for weighting attribute frequencies. A number of such measures are proposed in [54, 109, 209, 358, 498] and summarized very well in [93]. The work in [93] performed an extensive evaluation of a variety of categorical similarity measures, many of which are based on aggregate statistics. This work is especially relevant because the evaluations were performed in the context of the outlier detection problem. The broader conclusion of this work is that the use of statistical (aggregation-based) measures definitely improves the quality of the similarity. However, the experiments in the same work showed that no single measure was dominant over all the other measures.

Like the numerical similarity measures discussed in section 4.3.3 of Chapter 4, these similarity measures are heavily *data-dependent*. In the case of categorical data, such similarity measures assume a special importance because of the difficulty in computing numerical distances between points. An even more strongly data-dependent measure is contextual similarity, which is discussed in the next section.

8.4.2 Contextual Similarity

Contextual similarity measures provide a fundamentally different point of view, in which the relationships between data points are used to define relationships between attribute values and vice versa. For example, consider a setting in which we have a *Color* attribute with values like "Red," "Blue," and "Orange." If the data points containing "Red" and "Orange" are very similar to one another on the other attributes, as compared to the data points containing "Red" and "Blue," it is an indication of the greater semantic similarity between the former pair of attribute values. Therefore, a mismatch between "Red" and "Blue' should not be treated similarly to a mismatch between "Red" and "Orange." Note that this observation also applies to the matches between values *across* different attributes. For example, one could binarize the categorical attributes into a binary data set, and try to develop a more general model in which we account not for the matches between individual attributes but also to the matches across different attributes. The simplest approach for measuring contextual similarity is to use the random-forest/hierarchical-clustering method discussed in section 4.3.3. This approach can be easily adapted to categorical data, because many off-the-shelf implementations of random forests and hierarchical clustering are available. The following will introduce an alternative method [154] for binary data, which can also be generalized to categorical data by binarizing the data set.

Consider a binary transaction database from a supermarket in which the attributes are binary, and represent the buying behavior of items such as *Coke*, *Pepsi*, *Mustard*, and so on. In such cases, it is evident that two customers who buy *Coke* and *Pepsi*, respectively, are more likely to show similar buying patterns on the other attributes, than a customer who buys *Mustard*. The converse is also true, where similar pairs of customers (on other attributes) are more likely to buy *Coke* and *Pepsi* rather than *Coke* and *Mustard*. Therefore, it is evident that similarities between attributes can often be inferred from similarities between data points, and vice versa. It is noteworthy that there is an inherent circularity in this relationship. Such a circularity is usually resolved in machine learning algorithms with iterative methods.

The set of rows in which the attribute *Coke* takes on the value of 1 represents a *sub-relation* of the data. When two attributes have similar sub-relations in terms of the underlying patterns, the corresponding attributes are also similar. Therefore, the work in [154] uses an *Iterative Contextual Distance* algorithm, in which the distances are determined by repeating the following steps in circular fashion, after an initialization, which is based on simpler aggregate measures of similarity:

- **Computing distances between points from distances between attributes:** The distances between attributes are used in order to construct a real valued representation of the rows in the data. This real valued representation encodes all useful information about the inter-attribute correlations. Therefore distances between the rows in this real-valued representation automatically encode inter-attribute similarity. Although a number of simpler ways could be used to achieve this, the work in [154] uses a kernel-mapping approach in order to define the real-valued representations of

the rows. The details of this approach may be found in [154]. In order to encode the distances between different attribute values, this approach defines real values for each entry that correspond to the similarity of a particular attribute to the other attributes. The reader is referred to [154] for the specific details of this similarity computation.

- **Computing distances between attributes from distances between points:** The distances between attribute values is defined on the basis of distances between their sub-relations. First, the sub-relations are isolated for each attribute value (e.g., *Coke* customers, and *Pepsi* customers). The real-valued centroid of each of the kernel mapping of the rows in the sub-relations is determined. The L_1 distance between the centroids is used in order to measure the distance between the corresponding attributes. Other more complex measures such as the probabilistic differences between the distributions of the row values could also be used in order to achieve this goal in a more sophisticated way.

These steps are repeated to convergence. Although this algorithm was originally presented in the context of binary (market-basket) attributes, it can easily be generalized to any type of categorical data by using a preprocessing step of binarization.

8.4.2.1 Connections to Linear Models

This approach has a natural connection with the linear models discussed earlier in this chapter. In fact, linear models (in general) and matrix factorization methods (in particular) provide a natural way to compute similarities between rows and columns simultaneously. Let D be an $N \times d$ matrix of binary transaction data. This matrix can be factorized into two low-rank matrices of dimensions $N \times k$ and $d \times k$, where k is the rank of the factorization. We replicate Equation 3.28 of Chapter 3 here:

$$D \approx UV^T \tag{8.9}$$

This form of matrix factorization provides a different way of inferring similarities between rows and columns of D simultaneously with the use of latent representations. The similarities among the N rows of U provide the latent similarities among the rows of D (taking into account inter-attribute correlations), whereas the similarities among the d rows of V provide the latent similarities among the columns of D (taking into account inter-point similarities). This suggests that we can achieve a roughly similar goal by factorizing the matrix D into two low-rank factors and using these factors to compute similarities between points and attributes.

It is possible to use PCA for matrix factorization. However, transaction matrices are very large and sparse, which make them unsuitable for off-the-shelf PCA. This is because the dimensionality of such data sets may be of the order of millions, whereas only a small proportion (e.g., 0.01%) of the entries might be nonzero. One can use any form of constrained or unconstrained matrix factorization using only the non-zero entries in D and a sample of the zero entries. In other words, entry-wise sampling is used on the zero entries in order to make these methods scalable. Such an approach is more efficient, and it amounts to the application of matrix factorization on incomplete data (cf. section 3.5.1 of Chapter 3). Nonzero entries with large differences from their predicted values can be viewed as outlier entries in the matrix. As discussed in Chapter 3, one can consolidate the entry-wise scores into row-wise scores. Furthermore, ensemble methods with the use of multiple samples can be used to make these methods more efficient. If efficiency is not a concern, one can even use kernel PCA to infer nonlinear relationships among the rows and columns.

8.4.3 Issues with Mixed Data

It is fairly straightforward to generalize any proximity-based approach to the case of mixed data, as long as appropriate weights can be assigned to the categorical and numerical components. For example, let us consider two records $\overline{X} = (\overline{X_n}, \overline{X_c})$ and $\overline{Y} = (\overline{Y_n}, \overline{Y_c})$, where $\overline{X_n}$, $\overline{Y_n}$ are the subsets of numerical attributes and $\overline{X_c}$, $\overline{Y_c}$ are the subsets of categorical attributes. Then, the overall similarity between $\overline{X}$ and $\overline{Y}$ is defined as follows:

$$Sim(\overline{X}, \overline{Y}) = \lambda \cdot NumSim(\overline{X_n}, \overline{Y_n}) + (1 - \lambda) \cdot CatSim(\overline{X_c}, \overline{Y_c}) \tag{8.10}$$

The parameter λ regulates the relative importance of the categorical and numerical attributes. The choice of λ can sometimes be a challenging task, especially since the similarities on the two domains may not be of the same scale. In the absence of prior or domain knowledge about the relative importance of attributes, a natural choice would be to use a value of λ that is equal to the fraction of numerical attributes in the data. Furthermore, the proximity in numerical data is often computed with the use of distance functions rather than similarity functions. However, distance values can be converted to similarity values using a method suggested in [93]. For a distance value of $dist$, the corresponding similarity value is denoted by $1/(1+dist)$. Alternatively, one might choose to use the *heat-kernel* value of e^{-dist^2/t^2}, where t is a user-defined parameter.

Further normalization is required to meaningfully compare the similarity value components on the numerical and categorical attributes, which may be in completely different scales. One way of achieving this goal would be to determine the standard deviations in the similarity values over the two domains with the use of sample pairs of records. Each component of the similarity value (numerical or categorical) is divided by its standard deviation. Therefore, if σ_c and σ_n be the standard deviations of the similarity values in the categorical and numerical components, then Equation 8.10 needs to be modified as follows:

$$Sim(\overline{X}, \overline{Y}) = \lambda \cdot NumSim(\overline{X_n}, \overline{Y_n})/\sigma_n + (1 - \lambda) \cdot CatSim(\overline{X_c}, \overline{Y_c})/\sigma_c$$

By performing this normalization, the value of λ becomes more meaningful, as a true *relative weight* between the two components. By default, this weight can be set to be proportional to the number of attributes in each component, unless specific domain knowledge is available about the relative importance of attributes.

8.4.4 Density-Based Methods

Density-based methods can be naturally extended to discrete data, because numerical attributes are often discretized to create frequency profiles. In categorical data, these frequency profiles can be constructed directly on different combinations of values of the discrete attribute. The corresponding methods for numerical data have been discussed in section 4.4.3 of Chapter 4. In the case of categorical data, such methods are very natural to use, since the additional step of discretization does not need to be performed in order to convert the numerical values to discrete values. In the case of mixed attribute data, the numerical attributes may be discretized, whereas the categorical attributes may be retained in their original form. All other steps are identical to the methodology discussed in section 4.4.3 of Chapter 4.

8.4.5 Clustering Methods

As in the case of numerical data, outliers are defined as data points that are not members of clusters, or are not in sufficient proximity of clusters. While numerous clustering methods

exist for categorical data, such methods are often not applied to the categorical domain because of the greater difficulty in defining similarity between individual records and cluster representatives (centroids). A clustering method that is commonly used in the categorical domain for outlier analysis is the EM-method discussed earlier in this section. In that case, the negative logarithms of the fit probabilities can be used to quantify outlier scores. Although it is possible to define outliers on the basis of non-membership to clusters, such methods are not optimized to finding true anomalies in the data and may often discover noise. Stronger ways of scoring data points based on the distance to the nearest cluster and the size of the nearest cluster are discussed in sections 4.2 and 5.2.4. The only difference is that the distances need to be defined for categorical data, as discussed earlier in this chapter. Since clustering methods tend to be sensitive to different parameter choices such as the number of clusters, it is advisable to score each point in multiple ways with different clusterings. The scores from these different clusterings are averaged in order to create the final outlier score.

A discussion of common clustering methods for categorical data is also provided in the bibliographic section of this chapter, although most of these methods are optimized towards finding clusters rather than outliers. Several recent books [23, 33] discuss clustering methods and similarity measures for categorical data. A clustering method for categorical data is described in [29], and an outlier detection method is also integrated into this technique.

8.5 Outlier Detection in Binary and Transaction Data

A significant amount of categorical data is binary in nature. For example, transaction data, which contains customer buying patterns is almost always binary. Furthermore, all forms of categorical data and numerical data can always be transformed to binary data by various forms of discretization. Since binary data is a special case of all forms of categorical and numerical data, most of the methods discussed in this and other chapters can be applied to such data. Nevertheless, transaction data, which exists in a binary form in its natural state, is different from the binary data obtained by artificial conversion in terms of a number of practical aspects. The former type of data is typically very high dimensional and sparse with highly correlated attributes. Therefore, a number of methods have also been designed that are specifically optimized to these types of data. These methods can also be applied to categorical data sets after applying binarization, provided that the resulting binary data sets retain their sparsity.

8.5.1 Subspace Methods

Since transaction data is inherently high-dimensional, it is natural to utilize subspace methods [4] in order to identify the relevant outliers. The challenge in subspace methods is that it is often computationally impractical to define subspaces (or sets of items), which are sparse for outlier detection. For example, in a sparse transaction database containing hundreds of thousands of items, sparse itemsets are the norm rather than the rule. Therefore, a subspace exploration for sparse itemsets is likely to report the vast majority of patterns. The work in [254] addresses this challenge by working in terms of the relationship of transactions to dense subspaces rather than sparse subspaces. In other words, this is a reverse approach of determining transactions that are *not included* in most of the relevant dense subspace clusters of the data. In the context of transaction data, subspace clusters are essentially frequent patterns.

The idea in such methods is that frequent patterns are less likely to occur in outlier transactions, as compared to normal transactions. Therefore, a measure has been proposed in the FP-Outlier work [254], which sums up the support of all frequent patterns occurring in a given transaction in order to provide the outlier score of that transaction. The total sum is normalized by dividing with the number of frequent patterns. However, this term can be omitted from the final score because it is the same across all transactions and does not affect the relative ordering of outlier scores.

Let $\mathcal{D}$ be a transaction database containing the transactions denoted by $T_1 \ldots T_N$. Let $s(T_i, \mathcal{D})$ represent the support of transaction T_i in $\mathcal{D}$. Therefore, if $FPS(\mathcal{D}, s_m)$ represents the set of frequent patterns in the database $\mathcal{D}$ at minimum support level s_m, the frequent pattern outlier factor $FPOF(T_i)$ of a transaction $T_i \in \mathcal{D}$ at minimum support s_m is defined as follows:

$$FPOF(T_i) = \frac{\sum_{X \in FPS(\mathcal{D}, s_m), X \subseteq T_i} s(T_i, \mathcal{D})}{|FPS(\mathcal{D}, s_m)|}$$

Intuitively, a transaction containing a large number of frequent patterns with high support will have high value of $FPOF(T_i)$. Such a transaction is unlikely to be an outlier, because it reflects the major patterns in the data.

As in other subspace methods, such an approach can also be used in order to explain why a data point may not be considered an outlier. Intuitively, the frequent patterns with the largest support, which are also not included in the transaction T_i are considered *contradictory patterns* to T_i. Let S be a frequent pattern not contained in T_i. Therefore, $S - T_i$ is non-empty, and the *contradictiveness* of frequent pattern S to the transaction T_i is defined by $s(S, \mathcal{D}) * |S - T_i|$. Therefore, a transaction which does not have many items in common with a very frequent itemset is likely to be one of the explanatory patterns for the T_i being an outlier. The patterns with the top-k values of contradictiveness are reported as the corresponding explanatory patterns.

At an intuitive level, such an approach is analogous to non-membership of data points in clusters in order to define outliers, rather than directly trying to determine the deviation or sparsity level of the transactions. As was discussed in the chapter on clustering-based methods, such an approach may sometimes not be able to distinguish between noise and anomalies in the data. However, the approach in [254] indirectly uses the weight and number of clusters in the outlier score. Furthermore, it uses *multiple* patterns in order to provide an ensemble score. This is at least partially able to alleviate the noise effects. In the context of very sparse transactional data, in which direct exploration of rare subspaces is infeasible, such an approach would seem to be a reasonable adaptation of subspace methods. One can view the FP-Outlier method as the unsupervised one-class analog of rule-based method in classification. It is noteworthy that frequent pattern mining methods are often used for rule-based methods in classification [364]. In fact, one can view many subspace methods as the one-class analogs of rule-based methods in classification; in the case of the FP-Outlier method, this similarity is particularly obvious. Similar methods have also been used for subspace outlier detection in high-dimensional and numeric data (cf. section 5.2.4 of Chapter 5).

Frequent pattern mining methods are closely related to information-theoretic measures for anomaly detection. This is because frequent patterns can be viewed as a code-book in terms of which to represent the data in a compressed form. It has been shown in [494], how frequent patterns can be used in order to create a compressed representation of the data set. Therefore, a natural extension is to use the description length [497] of the compressed data in order to compute the outlier scores. This approach was further improved in [42].

8.5.2 Novelties in Temporal Transactions

Transaction data are often temporal in nature, in which the individual transactions are associated with a time-stamp. A new transaction that is not consistent with the "normal" model of previous transactions can be viewed as a novelty. A method for novelty detection in fast binary data streams was proposed in [29]. In fact, this approach falls in a general class of proximity-based methods, which will be discussed for text data later in this chapter. Thus, this method is fairly general and can be applied to both text and categorical data.

The broad idea of the approach is to continuously maintain the clusters in the underlying temporal data stream. Data points that do not naturally fit into any of these clusters are regarded as novelties. Thus, for each incoming data point, its largest similarity to the centroids of the current set of clusters is evaluated. If this similarity is statistically too low, then the data point is flagged as a novelty, and it is placed in a cluster of its own. The determination of the statistical threshold on similarity is achieved on the basis of the average similarity of other points to the centroid of the cluster. Subsequently, it is possible that this data point may represent the beginning of a new trend or cluster in the data. Such situations are quite common in many novelty-detection applications in which a detected outlier eventually becomes a normal part of the data. However, in some cases, such data points may continue to remain outliers over time. This issue will be discussed in more detail in Chapter 9 on outlier detection in temporal data.

8.6 Outlier Detection in Text Data

Text data shares a number of conceptual similarities with high-dimensional and sparse transaction data. However, word frequencies are often associated with text attributes; therefore, the data set is typically not binary. Nevertheless, simplified binary representations are sometimes used for text in which only presence or absence of words are considered. Text data can be viewed as a sparse, high-dimensional, and non-negative form of multidimensional data. Therefore, most of the probabilistic, linear, and proximity-based methods for binary and multidimensional data can be generalized to text. However, there are some subtle differences in how these models are implemented because of the sparse, high-dimensional, and non-negative nature of text. There are also some interesting connections between probabilistic and linear models, because some linear (matrix factorization) models for text are also probabilistic in nature. In the following, we will use the same notation as used for multidimensional data throughout the book. Therefore, the frequency-vector of words for the ith document is denoted by $\overline{X_i}$, and it is possible for this vector to be binary.

8.6.1 Probabilistic Models

Probabilistic models are commonly used for *soft* clustering of text data in which documents are assigned probabilities of membership to various clusters. By "soft" clustering, we refer to the fact that documents are not clearly assigned to specific clusters, but have a probability of assignment to each cluster. This approach is essentially an application of the expectation-maximization (EM) algorithm (see Chapter 2) to the text domain.

The main idea is to represent a corpus as a mixture of various distributions, the parameters of which are estimated using a particular document collection. The number of documents is denoted by N, the number of mixture components by k, and the lexicon size (terms) by d. The primary assumptions of the mixture-modeling approach are as follows:

- Each document is assumed to have a probability of belonging to one of the k mixture components denoted by $\mathcal{G}_1 \ldots \mathcal{G}_k$. The probability that the document $\overline{X}_i$ belongs to the cluster $\mathcal{G}_j$ is denoted by $P(\mathcal{G}_j|\overline{X}_i)$. Since documents belong to clusters in a probabilistic sense, this approach creates a soft partitioning. The value of $P(\mathcal{G}_j|\overline{X}_i)$ is estimated using the expectation-maximization approach on a generative model, and is one of the primary outputs of the algorithm.
- Each cluster is associated with a probability vector that quantifies the probability of the different terms in the lexicon for that cluster. Let $t_1 \ldots t_d$ be the d terms in the lexicon. Then, if a document belongs to cluster $\mathcal{G}_j$, the probability that the term t_l occurs in it is given by $P(t_l|\mathcal{G}_j)$. The probability $P(t_l|\mathcal{G}_j)$ is another parameter estimated by the expectation-maximization algorithm.

These assumptions are similar to those of the mixture model discussed in Chapter 2. In fact, the generative model is identical, and the same set of steps is used:

1. Select the rth mixture component with probability $\alpha_r = P(\mathcal{G}_r)$.
2. Generate the term frequencies of a document based on the predefined distribution of $\mathcal{G}_r$. A simple Bernoulli distribution is used for the binary representation of text, although the multinomial model is used to handle explicit word frequencies. The typical parameter used to describe this distribution represents a term generation frequency from the mixture component. For example, the parameter $p_l^{(r)} = P(t_l|\mathcal{G}_r)$ is often used, which is the probability of term t_l being generated from mixture component $\mathcal{G}_r$.

The observed corpus is assumed to be an output of the generative process, and therefore the parameters of each of the mixture components are learned using the generative process. The prior probabilities α_r and the parameters of the generative distribution of mixture component $\mathcal{G}_r$ need to be estimated by this approach. It is noteworthy that the use of a Bernoulli model implicitly assumes that the frequencies of the documents are ignored, and the documents are generated in binary format. The following discussion will assume the use of a Bernoulli distribution, although the basic methodology remains unchanged for the multinomial distribution; the only difference is in the distribution-specific estimation of the underlying parameters.

The probability $P(\mathcal{G}_j|\overline{X}_i)$ of the ith document being generated by the jth cluster can be computed using the Bayes rule as follows:

$$P(\mathcal{G}_j|\overline{X}_i) = \frac{P(\mathcal{G}_j)P(\overline{X}_i|\mathcal{G}_j)}{\sum_{r=1}^{k} P(\mathcal{G}_r)P(\overline{X}_i|\mathcal{G}_r)} = \frac{\alpha_j P(\overline{X}_i|\mathcal{G}_j)}{\sum_{r=1}^{k} \alpha_r P(\overline{X}_i|\mathcal{G}_r)} \tag{8.11}$$

The right-hand side of this equation is easy to estimate with the use of the probability distribution modeled for $\mathcal{G}_j$. For example, if a Bernoulli distribution is assumed for $\mathcal{G}_j$, the right-hand side may be expressed[2] as follows:

$$P(\mathcal{G}_j|\overline{X}_i) = \frac{\alpha_j \cdot \prod_{t_l \in \overline{X}_i} p_l^{(j)} \cdot \prod_{t_l \notin \overline{X}_i} (1 - p_l^{(j)})}{\sum_{r=1}^{k} \alpha_r \cdot \prod_{t_l \in \overline{X}_i} p_l^{(r)} \cdot \prod_{t_l \notin \overline{X}_i} (1 - p_l^{(r)})} \tag{8.12}$$

The main problem here is that we have no idea of what the values of the parameters $p_l^{(j)}$ and α_j ought to be in order to compute the aforementioned equation. After all, the values

[2]We have used notations somewhat loosely in Equation 8.12. Even though $\overline{X}_i$ is not a set (but a binary vector), we are treating it as a bag of words when using expressions like $t_l \in \overline{X}_i$ on the right-hand side.

of $p_l^{(j)}$ can be estimated (using maximum-likelihood methods) only if we knew the probabilistic assignment probability $P(\mathcal{G}_j|\overline{X}_i)$ of the documents to the various clusters. A similar observation holds for the parameter α_j. In other words, the right-hand of Equation 8.12 can be estimated only if we already knew the value on the left-hand side. Therefore, there is an inherent circularity to this relationship, which is resolved with iterative methods.

We start with a random assignment of documents to mixture components, which gives us an initial value of each $P(\mathcal{G}_j|\overline{X}_i) \in \{0, 1\}$. Subsequently, we estimate each α_j as the fraction of documents belonging to $\mathcal{G}_j$ in the initial random assignment. The value of $p_l^{(j)}$ is estimated as the fraction of documents in $\mathcal{G}_j$ that contain the term t_l. Subsequently, the following iterative expectation-maximization steps are applied:

- **(E-Step):** The currently initialized values of the parameters are used to estimate the generative probability $P(\mathcal{G}_j|\overline{X}_i)$ of each document from one of the clusters according to Equation 8.12.
- **(M-Step):** The value of $P(\mathcal{G}_j|\overline{X}_i)$ is treated as the weight of document i in mixture-component j and is used to estimate various parameters as follows. The value of each $p_l^{(j)}$ is estimated as the (weighted) fraction of documents in cluster j that contain term t_l. The value of α_j is computed as the (weighted) fraction of the points belonging to the cluster j. In both cases, Laplacian smoothing may be used to avoid overfitting.

These steps are iterated to convergence. One can compute the fit probability of document $\overline{X}_i$ being generated by the model as the sum of the generative probabilities over various mixture components:

$$P(\overline{X}_i) = \sum_{r=1}^{k} P(\mathcal{G}_r) \cdot P(\overline{X}_i|\mathcal{G}_r) \tag{8.13}$$

$$= \sum_{r=1}^{k} \alpha_r \cdot \prod_{t_l \in \overline{X}_i} p_l^{(r)} \cdot \prod_{t_l \notin \overline{X}_i} (1 - p_l^{(r)}) \tag{8.14}$$

Documents that have low fit probability can be declared outliers. One can also treat $P(\overline{X}_i)$ as an outlier score in which smaller values are more indicative of outliers. Therefore, one can use extreme-value analysis on the score. However, for reasons discussed in section 6.2.1 of Chapter 6, extreme-value analysis should be applied on the logarithms of $P(\overline{X}_i)$ because very low probability values lose numerical stability, and discrimination between the different values is not well-represented by probability values in the context of hypothesis testing.

The EM-algorithm can also be adapted to online scenarios. In such scenarios, the parameters of the model are iteratively updated in the context of an online data stream. In such cases, only a window of the current set of documents is used for the update equations, so that the model is updated over time in order to reflect the current trends in the data. Whenever a document is encountered which does not fit well into the current list of topics, a new topic can be initiated containing this document in a dominant way. Thus, this scenario corresponds to the case of growing clusters.

8.6.2 Linear Models: Latent Semantic Analysis

The goal of Latent Semantic Analysis (LSA) is largely to *improve the underlying data representation*, so as to reduce the impact of noise and outliers. One of the problematic

aspects of the text representation is that of *synonymy* and *polysemy*. Synonymy refers to the fact that multiple words might describe the same concept. For example, a "*car*" might also be referred to as an "*automobile*." Therefore, two documents containing the words "*car*" and "*automobile*," respectively, might not be considered sufficiently similar. In polysemy, the same word might correspond to different concepts. For example, the word "*Jaguar*" might either refer to a car or a cat. Clearly, the presence of such words can be viewed as noise, which can greatly reduce the quality of many applications such as similarity search. Latent semantic analysis is a method to improve the underlying data representation, so as to reduce the impact of such noise. This broader principle has also been discussed in the section on noise correction with PCA in Chapter 3.

LSA is the text-centric avatar of SVD. Specifically, let D be the $N \times d$ term-document matrix in which the (i, j)th entry is the normalized frequency for term j in document i. Then, $D^T D$ is a $d \times d$ matrix, which can be viewed as a similarity matrix between pairs of dimensions. For mean-centered data, it would also be equivalent to a scaled version of the covariance matrix, although one rarely performs mean-centering[3] in sparse domains like text. The largest eigenvectors of $D^T D$ are computed in order to represent the text. In typical text collections, only about 300 to 400 eigenvectors are required for the representation. One excellent characteristic of LSA is that the truncation of the dimensions removes the noise effects of synonymy and polysemy, and the similarity computations are better regulated by the semantic concepts in the data.

It should be further noted that documents that are incoherent, spam, or otherwise incongruent with the remaining data are more easily revealed in the new representation because they will have larger components along the small eigenvalues. Such documents can be more easily distinguished from the data. If desired, LSA can also be used in combination with proximity-based techniques for more effective outlier detection. This is because the quality of the similarity computations is improved in the latent space, and one can therefore identify outliers more easily.

8.6.2.1 Probabilistic Latent Semantic Analysis (PLSA)

PLSA can be considered both a probabilistic method and a linear method like LSA. However, this book has classified it among linear methods because of its natural connections to LSA. LSA is a straightforward application of SVD to the $N \times d$ document-term matrix. As discussed in Chapter 3, one can write a low-rank matrix-factorization of SVD with data matrix D as follows:

$$D \approx (Q\Lambda)P^T = UV^T$$

In the case of LSA, the matrix D is the document-term matrix, the matrices Q and P are $N \times k$ and $d \times k$ matrices, respectively, and Λ is a $k \times k$ diagonal matrix, where k is the rank of the decomposition (i.e., the number of eigenvectors selected by LSA). Therefore, U and V are $N \times k$ and $d \times k$ matrices, respectively. The rows of the matrix $Q\Lambda$ provide the k-dimensional latent representations of the documents and the columns of the matrix P provide the k orthonormal basis vectors of representation. These basis vectors are also the d-dimensional eigenvectors of $D^T D$. One can write the corresponding optimization model

[3]This is a common difference between PCA and SVD. The two are equivalent when the data is mean-centered. However, SVD can be performed on any matrix, whereas PCA is always performed only on mean-centered matrices.

as follows:

$$\begin{aligned}\text{Minimize } & J = ||D - UV^T||^2 \\ & \text{subject to:} \\ & \text{Columns of } U \text{ are mutually orthogonal} \\ & \text{Columns of } V \text{ are mutually orthonormal}\end{aligned}$$

SVD (and LSA) can be shown to be solutions to this optimization model. PLSA is a variation of this optimization model. For example, the objective function of minimizing the Frobenius norm of the residual matrix $(D - UV^T)$ is a way of ensuring that D matches UV^T as closely as possible. There are other ways of changing the objective function to provide a more probabilistic interpretation. The specific optimization model solved by PLSA, which is a probabilistic avatar of nonnegative matrix factorization, is as follows:

$$\begin{aligned}\text{Maximize } & J = \text{Likelihood matching between } D \text{ and } UV^T \\ & \text{subject to:} \\ & U \text{ is nonnegative} \\ & V \text{ is nonnegative}\end{aligned}$$

Whenever one talks of a "maximum likelihood matching" one is talking about a generative process. What is this generative process, and what are the parameters of this process?

The matrix D can be viewed as an outcome of an generative process. In fact, a set of matrices Q, Λ and P can be extracted from U and V, which form the parameters for this generative process. The columns of U are scaled to sum to 1 to create Q. The columns of V are scaled to sum to 1 to create P. The (r, r)th diagonal matrix Λ is *proportional* to the product of the scaling factors of U and V, respectively, for the rth columns. We use the word "proportional" because the diagonal entries are scaled to sum to 1. Therefore, we have $UV^T \propto Q\Lambda P^T$. These parameters are used to define the following generative process for the data matrix D:

1. Select a mixture component $\mathcal{G}_r$ with probability Λ_{rr}.

2. Select the row i of D with probability proportional to the (i, r)th entry of Q and the column j of D with probability proportional to the (j, r)th entry of P. Increment the (i, j)th entry of D by 1.

An implicit assumption is that the row i and column j are selected in a *conditionally independent* way in the second step after selecting the mixture component $\mathcal{G}_r$. One way of interpreting the parameters in Q, P and Λ in the context of the generative process above is as follows. Let $\overline{X}_i$ be the ith document and t_j be the jth term. The (r, r)th entry in the diagonal matrix Λ contains $P(\mathcal{G}_r)$, which is used in the first step of selecting the mixture component. The (i, r)th entry in Q contains $P(\overline{X}_i|\mathcal{G}_r)$, and the (j, r)th entry in P contains $P(t_j|\mathcal{G}_r)$. These probabilities are used in the aforementioned generative process, and are also *exactly* the parameters used by the generative process of section 8.6.1; however, the generative processes used in the two cases are different in spite of similarity in parametrization.

PLSA learns a matrix factorization (and underlying probabilistic parameters) that maximizes the likelihood of the observed document-term frequencies to be generated by this process. The EM steps for PLSA may be defined as follows:

- **(E-step):** Estimate $P(\mathcal{G}_r|\overline{X}_i, t_j)$ for each document-term pair using the currently estimated parameters in conjunction with the Bayes rule:

$$P(\mathcal{G}_r|\overline{X}_i, t_j) = \frac{P(\mathcal{G}_r) \cdot P(\overline{X}_i|\mathcal{G}_r) \cdot P(t_j|\mathcal{G}_r)}{\sum_{s=1}^{k} P(\mathcal{G}_s) \cdot P(\overline{X}_i|\mathcal{G}_s) \cdot P(t_j|\mathcal{G}_s)} \quad \forall r \in \{1 \ldots k\} \tag{8.15}$$

- **(M-step):** Estimate the current parameters in Q, P and Λ by using the conditional probabilities in the first step as weights for entries belonging to each generative component. This is achieved as follows:

$$Q_{ir} = P(\overline{X}_i|\mathcal{G}_r) = \frac{\sum_j P(\overline{X}_i, t_j) \cdot P(\mathcal{G}_r|\overline{X}_i, t_j)}{P(\mathcal{G}_r)} \propto \sum_j D_{ij} P(\mathcal{G}_r|\overline{X}_i, t_j)$$

$$P_{jr} = P(t_j|\mathcal{G}_r) = \frac{\sum_i P(\overline{X}_i, t_j) \cdot P(\mathcal{G}_r|\overline{X}_i, t_j)}{P(\mathcal{G}_r)} \propto \sum_i D_{ij} P(\mathcal{G}_r|\overline{X}_i, t_j)$$

$$\Lambda_{rr} = P(\mathcal{G}_r) = \sum_{i,j} P(\overline{X}_i, t_j) \cdot P(\mathcal{G}_r|\overline{X}_i, t_j) \propto \sum_{i,j} D_{ij} P(\mathcal{G}_r|\overline{X}_i, t_j)$$

  The constants of proportionality are set by ensuring that the probabilities in the columns of P, Q and the diagonal of Λ each sum to 1.

The generative process of section 8.6.1 is somewhat different from the one discussed in this section. The technique in section 8.6.1 generates each document with a single iteration of the generative process, whereas the technique in this section fills up the *entries* of the document-term matrix with the generative process. Note that one can interpret the entries in the (scaled) matrix D as observed values of $P(\overline{X}_i, t_j)$. The (i, j)th entry of D can be interpreted in terms of the generative process to derive a matrix factorization as follows:

$$D_{ij} \propto P(\overline{X}_i, t_j) = \sum_{r=1}^{k} \underbrace{P(\mathcal{G}_r)}_{\text{Select } r} \cdot \underbrace{P(\overline{X}_i, t_j|\mathcal{G}_r)}_{\text{Select } \overline{X}_i,\ t_j} \text{ [Generative probability of incrementing } (i, j)\text{]}$$

$$= \sum_{r=1}^{k} P(\mathcal{G}_r) \cdot P(\overline{X}_i|\mathcal{G}_r) \cdot P(t_j|\mathcal{G}_r) \text{ [Conditional independence]}$$

$$= \sum_{r=1}^{k} P(\overline{X}_i|\mathcal{G}_r) \cdot P(\mathcal{G}_r) \cdot P(t_j|\mathcal{G}_r) \text{ [Rearranging product]}$$

$$= \sum_{r=1}^{k} Q_{ir} \cdot \Lambda_{rr} \cdot P_{jr} = (Q\Lambda P^T)_{ij} \text{ [Note the similarity to LSA]}$$

One can generate outlier scores not only for each document but also for each entry. For each entry, one might use the (absolute value of the) entry-wise residuals of the matrix factorization as an approximation to the outlier score. The document-wise scores are the sum of the squares of the residuals in each row. Similarly, the term-wise scores are the sum of the squares of the residuals in each column. Larger values of the scores are indicative of a greater tendency of a point to be an outlier. Therefore, one can identify document-wise, term-wise, and entry-wise outliers. This is because the PLSA technique is well-suited to explaining the generative process of the entire matrix in an entry-wise fashion, rather than as a generative process in document-wise fashion.

It is noteworthy that PLSA is a special case of nonnegative matrix factorization, in which the objective function uses maximum-likelihood matching. Traditional forms of nonnegative matrix factorization use the Frobenius norm of the residual matrix (like PCA and SVD). As a practical matter, the difference between these two forms of factorization is very limited; however, exactly the same solution is not obtained in the two cases because of differences in the underlying objective function.

PLSA was the first technique in the category of *topic-modeling* techniques. Many other related probabilistic methods such as Latent Dirichlet Allocation (LDA) are commonly used for topic modeling [90, 91]. All classes of topic-modeling methods can be adapted to outlier detection. LDA is particularly useful if the outlier scores of new documents (i.e., those not included in the training data) are to be computed; this situation arises commonly in a temporal or streaming setting. PLSA can compute the scores of only in-sample documents.

8.6.3 Proximity-Based Models

As in all data types, the design of the similarity function is important in text data. The word frequencies need to be normalized in terms of their relative frequency of presence in the document and over the entire collection. For example, a common word such as "*the*" is not as discriminative (or informative) for outlier detection, as compared to a word such as "*car*." A common approach for text processing is the *vector-space based tf-idf* representation [472]. In the tf-idf representation, the term frequency for each word is normalized by the *inverse document frequency*, or *idf*. The inverse document frequency normalization reduces the weight of terms which occur more frequently in the collection. This reduces the importance of common terms in the collection, ensuring that the matching of documents is more influenced by that of more discriminative words which have relatively low frequencies in the collection. Therefore, if n_i is the number of documents in which the ith word occurs, and N is the total number of documents, the term frequency of the ith word is multiplied with $\log(N/n_i)$ in order to emphasize the importance of infrequent words.

After this normalization has been achieved, the cosine similarity function is applied to the vector-space representations in order to measure similarity. The cosine measure is equal to the dot product of two vectors, once they have been normalized to unit length. The idea of normalization is to compute similarities in a fair way between documents of varying lengths. Thus, for two normalized document frequency vectors $\overline{X}$ and $\overline{Y}$, the cosine function $Cosine(\overline{X}, \overline{Y})$ can be defined as follows:

$$Cosine(\overline{X}, \overline{Y}) = \frac{\overline{X} \cdot \overline{Y}}{||\overline{X}|| \cdot ||\overline{Y}||} \tag{8.16}$$

Here, $|| \cdot ||$ represents the L_2-norm. The earliest work on proximity-based models was done in the context of the topic detection and tracking (TDT) project [49, 622]. Most of the methods [47, 48, 49, 622] determine key novelties in the stream by using the following broad two step framework for each incoming document, with respect to a current summary which is maintained for the corpus:

- Compute the similarity of the incoming document to the current summary of the corpus. The summary of the corpus could correspond to either a fine-grained clustering, or a sample of the documents which have been received so far. Report any and all documents that have low similarity with the current summary as anomalies. Alternatively, the similarity to the closest cluster centroid or to the nearest document in the

sample could be reported as the outlier score. Smaller values indicate greater degree of outlierness.

- Update the summary of the incoming documents in the data stream by incorporating the incoming document into an appropriate component of the summary. For example, a clustering approach might adjust the centroid of the closest cluster to reflect the incorporation of the document in the cluster.

Recently, a method has also been proposed for novelty detection in fast data streams [29]. This approach uses an exponential decaying model in order to determine clusters and novelties simultaneously from the data stream. Eventually such novelties are converted into clusters, if more data points arrive which match the content of the document. Such methods are also able to identify novelties that are eventually converted into broad trends in the data stream.

The method of [29] makes the assumption of fixed-memory availability, because of which the number of clusters in the data remain fixed. As a result, a current cluster needs to be ejected from the buffer when a new cluster is inserted. Typically, such ejected clusters are chosen as those (faded) trends that have not been updated for a while. However, it is possible that at a future time, such a trend will re-appear. As a result, a similar cluster may need to be created at a future point in time. Such outliers are referred to as *infrequently recurring outliers*, which are distinct from novelties that were never seen before. Under a *fixed space* constraint, it is often not possible for summarization methods to distinguish between novelties and infrequently recurring outliers. This is because recurring outliers are often dropped from the stream summary to conserve space. However, if the number of clusters is allowed to grow over time, it may be possible to distinguish between such events. Nevertheless, such an approach is likely to slow down over time and may no longer be relevant to the streaming scenario.

8.6.3.1 First Story Detection

The detection of novelties in a stream of text documents is closely related to the problem of *first story detection*. Much of the work on outlier detection in the text domain has been studied in the context of the problem of first-story detection in a streaming context such as a news wire service. Although some related methods for temporal data are studied in Chapter 9, the case of text is presented here, because it is more closely related to the other material in this chapter. Furthermore, there are very few methods for text outlier detection in non-temporal scenarios, and most of the methods for *first-story detection* can trivially be generalized to the non-temporal scenario. In the context of temporal data, such data points are also referred to as *novelties*. Two broad classes of methods exist for this problem, corresponding to proximity-based models and probabilistic models. Although both of these methods have been proposed in the context of temporal data, they can trivially be generalized to the non-temporal scenario, since the temporal component is utilized only from the perspective of data subset selection in the modeling process. In the non-temporal scenario, these methods simply need to be applied to the entire data instead of the previous history. Many of the non-temporal methods in this chapter can also be generalized to the temporal scenario.

8.7 Conclusions and Summary

This chapter studies methods for outlier detection in categorical, text, transaction, and mixed data sets. Such data sets share a number of broad similarities with one another. Therefore, it is natural to study the outlier detection methods for these different problems in a single chapter. Probabilistic models can be adapted to various data domains by defining appropriate generative models. Most of the existing proximity-based outlier detection methods can also be easily adapted to these different domains by defining appropriate similarity functions. Mixed-attribute data sets can be addressed by proximity-methods, as long as appropriate methods are defined for weighting the similarity of each component. Both text and binary data tend to be high dimensional and sparse. Therefore, many of the techniques for one domain can be used for the other, and vice versa. Linear models and matrix factorization methods are often used in such settings.

8.8 Bibliographic Survey

The earliest work on categorical outlier detection was performed in the context of clustering algorithms [210, 220, 226, 283, 307]. In many clustering algorithms, the outliers are identified as a side-product of clustering algorithms. Some of the earliest work on probabilistic clustering for outlier detection in categorical data is proposed in [578]. This approach uses a joint probability model that can address both categorical and mixed attributes. The approach can be effectively used for mixed data, since probabilistic models are able to normalize in an intuitively uniform way over numerical and categorical attributes. Furthermore, the approach uses a time-decaying model for the clustering component. This ensures that it can be used effectively in an evolving data set.

Although linear models have not been studied extensively in the context of outlier detection of categorical data, they are particularly useful because they can naturally account for inter-attribute correlations. A number of such correspondence-analysis methods [265] are specifically optimized for categorical data. However, the potential of these methods has not been fully explored in the context of outlier detection. A particularly important challenge with categorical data surfaces when a large number of potential values exists for a categorical attribute. In such cases, the increasing sparsity of the representation can make correspondence-analysis methods ineffective.

Proximity-based algorithms can be naturally generalized to categorical data as long as a similarity or neighborhood measure is available in order to perform the distance (similarity) computations. However, in most of these cases, the similarity computation is performed with the use of relatively straightforward overlap measures. Categorical similarity measures are crucial to all proximity-based applications such as clustering. Similarity measures in categorical data are closely related to corresponding measures in text data, because the conversion of categorical data to binary data often resembles sparse text data. In both cases, the representation is normalized, so that aggregate statistics in order to provide lower values to attributes which are rare in occurrence. The most common example in the case of text is the use of the inverse document frequency in order to normalize the similarity measures [472]. The corresponding measure in the case of categorical data is referred to as the inverse occurrence frequency.

Many of the similarity measures defined in the literature [39, 87, 154, 346, 423], are *contextual* in the sense that they use the neighborhood of a point in order to compute similarity. Of course, the relationship between similarity derivation and neighborhood computation is

circular, because neighborhood computation itself requires the computation of similarity. Therefore, many of these techniques use simpler measures such as the *Overlap* in order to define the initial neighborhood for similarity computation. Subsequently, iterative methods [154] are used in order to simultaneously refine the neighborhood and the similarity computation. Therefore, it is sometimes also helpful to develop similarity measures, which are based on the aggregate statistics, rather than on specific neighborhoods of data points.

Such aggregate methods either measure similarity only between same values of an attribute (using different kinds of normalizations), or they compute similarity between different values of the same attribute [93, 358]. Some of the common techniques that do not use similarity between values of different attributes are discussed in [209, 222, 472, 109]. Other measures that use functions of both matches and mismatches on different attribute values are proposed in [54, 358, 498]. Many of these similarity measures have been tested in the context of the outlier detection problem [93]. Contextual measures [154] have also been proposed, but have not been widely explored in the context of the outlier detection problem. Similarity measures have also been provided that can be used in the context of categorical and mixed data [596]. The effectiveness of these measures in the context of the outlier detection problem was studied in the same work.

Link-based methods [563] model common attribute values between data instances. The work in [563] models the categorical values as a graph, and distances between data instances are modeled by using the connectivity of this graph. This type of modeling can be used to identify outliers. Distributed link-based methods for finding outliers in categorical and mixed data are proposed in [421]. Two pattern-based methods for anomaly detection are discussed in [42, 155], the first of which is information-theoretic in nature.

The problem of binary and transaction data is especially challenging because of its high dimensionality. As shown in [254], direct adaptations of various methods such as replicator neural networks [567] and clustering [255] do not work effectively. In such data, a given transaction may contain only a very tiny fraction of the available items. The work in [254] shows how to adapt subspace methods to transaction data by finding transactions which do *not* lie in dense subspaces (or do not contain frequent patterns), rather than determining transactions that contain sparse subspaces. Another approach that is based on frequent patterns is proposed in [410]. Methods that use frequent patterns for information-theoretic anomaly detection methods are discussed in [42, 323, 497].

In the context of text data, latent semantic analysis is commonly used to identify and reduce noise. Latent semantic analysis [162] (LSA) is closely related to principal component analysis [296], which is studied in detail in Chapter 3. LSA was originally proposed as a method for retrieval of text [162], and its effects on improving the quality of similarity in text were observed in [425].

The problem of topic detection and tracking has been studied both in the unsupervised scenario [29, 47, 48, 49, 95, 306, 585, 608] and in the supervised scenario [584, 586]. Some of these methods have also been generalized to social streams, which contain a combination of text, linkage, and other side information, such as spatiotemporal data [30, 194, 435]. Most of the methods for novelty detection are based on proximity models, though a few methods [608] are also based on probabilistic models. A variety of probabilistic models such as PLSA and LDA [90, 91, 271] can be used for outlier analysis in text data. In the supervised scenario, it is assumed that training data are available in order to enhance the novelty detection process. This can be achieved either by providing examples of the rare class, the normal class or both. Such methods are often quite useful for detecting novel events in conventional news streams or social media such as *Twitter* [435]. A probabilistic model for determining novelties in text streams is proposed in [608]. Another aggregate

method for detection of correlated bursty topic patterns from coordinated text streams is proposed in [562].

8.9 Exercises

1. Download the *Adult data* from the UCI Machine Learning Repository [203]. Determine outliers with the use of k-nearest neighbor algorithm on the categorical attributes only by using:

   - Match-based similarity measures.
   - Inverse occurrence frequency similarity measure.

   How do the outliers compare to one another?

2. Repeat Exercise 1 by including the numerical attributes in the analysis. Use a Euclidean distance measure on the numerical attributes. Do you obtain the same outliers?

3. Compute the principal components of the mixed representation of the *Adult data* with and without attribute-specific normalization. Determine the outliers by using the χ^2 statistic on the sum of the squares of the deviations along the different principal components. Do you obtain the same set of outliers in the two cases?

4. Repeat the Exercises 1, 2, and 3 with the use of EM-based modeling. How do the outliers compare with the ones obtained in the previous exercises?

5. Repeat the Exercises 1, 2 and 3 with the use of the *Breast Cancer* data set. Use the rare classes in the data set as ground truth in order to construct an ROC curve in these cases. In which case do you obtain the best results?

6. Download the 20-newsgroups data set [624]. Use the following methods to determine the outliers:

   - Use a k-nearest neighbor approach with cosine similarity.
   - Use the probabilistic modeling approach, and report the data points with the least fit as the outliers.

   Do you obtain the same outliers in the two cases?

7. Repeat Exercise 6 with the Reuters-215788 data set [623].