Chapter 8 - Outlier Detection in Categorical, Text, and Mixed Attribute Data

1 Introduction

Extreme value analysis, PCA, Linear Models, proximity algorithms, and LOCI all assume numerical data, so we need to handle categorical variables. One option is to simply one-hot encode each categorical variable, but this is hard to scale if there are many categories.

2 Extending Probabilistic Models to Categorical Data

A mixture model has component weights α_m and distributions $\mathcal{G}_m$ for $m \in \{1...k\}$. We use a Gaussian distribution for numerical data and a Bernoulli distribution for categorical data. Let p_{ijm} be the probability that the i^{th} attribute takes on its j^{th} category as determined by $\mathcal{G}_m$. Now consider data point $\bar{X}$ and let j_r be the category of the r^{th} attribute. Let Θ be the mixture model parameters. We have $g^{m,\Theta}(\bar{X}) = \prod_{r=1}^{d} p_{rj_rm}$. The E-step is then $P(\mathcal{G}_m|\bar{X},\Theta) = \frac{\alpha_m g^{m,\Theta}(\bar{X})}{\sum_{r=1}^{k} \alpha_r g^{r,\Theta}(\bar{X})}$ - these are soft assignments. The α_m values are set to the weighted fraction of points assigned to that component (you can add small value λ to numerator and $k\lambda$ to denominator for Laplacian smoothing). Letting, $\mathcal{D}_{ij}$ be data points where attribute i takes on category j, we have $p_{ijm} = \frac{\sum_{\bar{X}\in\mathcal{D}_{ij}} P(\mathcal{G}_m|\bar{X},\Theta)}{\sum_{\bar{X}\in\mathcal{D}} P(\mathcal{G}_m|\bar{X},\Theta)}$. To smooth this, let v_i be the number of distinct values taken on by attribute i and add small value β to numerator and $v_i\beta$ to denominator. This is the M-step. The outlier score is just $Score(\bar{X}) = \log \sum_{m=1}^{k} \alpha_m g^{m,\Theta}(\bar{X})$.

If you have some categorical attributes and some numerical attributes, you can use two mixture models: $h^{m,\Theta}(\bar{X}) = f^{m,\Theta}(\bar{X}) g^{m,\Theta}(\bar{X})$.

3 Extending Linear Models to Categorical and Mixed Data

If you one-hot encode categorical attribute i (assume it takes on n_i possible values and that it takes on value j with relative frequency f_{ij}), make sure to divide each element of the vector by $\sqrt{n_i f_{ij}(1-f_{ij})}$. For numerical attributes, just make them zero mean and unit variance. Now you can use this data for PCA and linear regression.

4 Extending Proximity Models to Categorical Data

Define similarity between $\bar{X} = (x_1, ..., x_d)$ and $\bar{Y} = (y_1, ..., y_d)$ be $Sim(\bar{X}, \bar{Y}) = \sum_{i=1}^{d} S(x_i, y_i)$. If we set $S(x_i, y_i) = 1$ if $x_i = y_i$ and 0 otherwise, we call this overlap similarity. This isn't great because it ignores aggregate statistical properties of the data (e.g. it is interesting if two users have the same address, but not if they have the same gender) and it ignores local neighborhoods (e.g. Red is more similar to Orange than to Green).

The Eskin measure takes the overlap measure, but returns $\frac{n_i^2}{n_i^2+2}$ if $x_i \neq y_i$. The Inverse Occurrence Frequency (IOF) measure returns $\frac{1}{1+\log f(x_i)+\log f(y_i)}$ if they do not match (here f represents count). Another option is to return $(\log \frac{1}{p_i(x_i)})^2$ if $x_i = y_i$ and 0 otherwise.

To measure contextual similarity (e.g. Red is more similar to Orange than to Green), we need to compare similarity amongst the other attributes. You can do this with random-forest/hierarchical-clustering from Chapter 4. You can also use the Iterative Contextual Distance algorithm (which assumes binary categorical feature). In the first step, you compute a real vector representation for each row by measuring its distance to other data points (distance is measured as distance between attributes). In the second step, you consider each attribute and compute the centroid of points where it is 1 and the centroid of points where it is 0. The L1 distance between the centroids is the attribute distance. Repeat these two steps until convergence. You can alternatively do matrix factorization $D \approx UV^T$ and use the low-rank matrices as distances (this only works on binary categorical features). You can't use PCA here.

If you have a mix of numerical and categorical variables, define $Sim(\bar{X}, \bar{Y}) = \lambda NumSim(\bar{X}_n, \bar{Y}_n)/\sigma_n + (1-\lambda)CatSim(\bar{X}_c, \bar{Y}_c)/\sigma_c$. Set λ to be the fraction of numerical attributes. To turn a distance into a similarity, use $\frac{1}{1+dist}$ or $\exp \frac{-dist^2}{t}$

Density methods already discretize numerical data, so they can handle categorical data easily.

For clustering methods, just use the mixture model described earlier in this chapter.

5 Outlier Detection in Binary and Transaction Data

Transaction data is usually binary and sparse (it indicates which items in a catalog that a user has bought). A dense subspace indicates a frequent pattern (i.e. many customers buy this group of items) and is therefore not indicative of outliers. Given transation database $\mathcal{D}$ containing $T_1, ..., T_N$, let $s(T_i, \mathcal{D})$ be the support of T_i in $\mathcal{D}$. Let $FPS(\mathcal{D}, s_m)$ be the set of frequent patterns at minimum support level s_m. The Frequent Pattern Outlier Factor is $FPOF(T_i) = \frac{\sum_{X \in FPS(\mathcal{D}, s_m), X \subseteq T_i} s(T_i, \mathcal{D})}{|FPS(\mathcal{D}, s_m)|}$

6 Outlier Detection in Text Data

You can represent a document with a word frequency vector (or a vector with a 1 if word appears and 0 otherwise = bag of words). We can use a mixture model where $P(\mathcal{G}_j|\bar{X}_i)$ is the probability document i belongs to component j and $P(t_l|\mathcal{G}_j)$ is the probability that term l occurs in a document of component j. We use a Bernoulli distribution for bag of words and a multinoulli distribution for word frequency vectors. We have $P(\mathcal{G}_j|\bar{X}_i) = \frac{\alpha_j P(\bar{X}_i|\mathcal{G}_j)}{\sum_{r=1}^k \alpha_r P(\bar{X}_i|\mathcal{G}_r)}$. To compute this, we need the Bernoulli parameters p_l^j which is the probability of term t_l appearing in component j. So, we use the EM algorithm. First, randomly hard-assign documents to mixture components and estimate α_j as the fraction of documents in component j and p_l^j as the fraction of documents in component j that have term t_l. The E-Step computes the soft assignments $P(\mathcal{G}_j|\bar{X}_i)$ using the equation shown before. The M-Step sets p_l^j to the weighted fraction of documents in component j that have term t_l and α_j is the weighed fraction of documents belonging to component j. We then have $P(\bar{X}_i) = \sum_{r=1}^k \alpha_r \prod_{t_l \in \bar{X}_i} p_l^r \prod_{t_l \notin \bar{X}_i} (1 - p_l^r)$.

Synonemy is when two words describe the same concept (e.g. car and automobile) and polysemy is when a single word describes multiple concepts (e.g. lead means either a dense metal or to be a leader). Latent Semantic Analysis (LSA) is SVD for text. Let D be an $N \times d$ matrix where D_{ij} is the normalized frequency of term j in document i. Compute the top 300 or 400 eigenvalues of $D^T D$ (this approximation mitigates synonemy and polysemy). Documents with large values on the small eigenvectors are likely noise or outliers. Probalistic LSA is extension where we create a generative process for generating documents (that can be formulated as SVD) and use the EM algorithm to figure out the parameters (i.e. the elements of the SVD matrices). Read the book for more info. Honestly, this topic modeling stuff seems a bit outdated given that we have word/document embeddings and deep learning.

When representing a word's frequency in a document, use the term-frequency inverse-document-frequency (TF-IDF) instead. Then normalize each document vector to unit length. Then you can compute similarity of documents with cosine distance $Cosine(\bar{X}, \bar{Y}) = \frac{\bar{X} \cdot \bar{Y}}{||\bar{X}||||\bar{Y}||}$. Now that we have a similarity function, we can use proximity based methods.

For a streaming application, you can create document vectors and cluster them. When you get a document online, you measure distance to nearest cluster. If it's too far, it's an outlier and starts a new cluster. Otherwise, you can add it to an existing cluster and move its centroid. This a useful technique for first story detection (where you are looking for the first story about some new topic).

7 Conclusions and Summary

We can extend our models for categorical variables, including text.