Chapter 5 - High-Dimensional Outlier Detection: The Subspace Method

1 Introduction

With high dimensional data, most features are noisy, which makes distance computations almost useless. Outliers are best detected in a lower dimensional subspace. Identifying the relevant subspaces is hard. There may also be multiple relevant subspaces, so ensembling techniques helps. We have three methods: rarity-based (tries to find subspaces with rare distributions), unbiased (randomly pick subspaces and find outliers - used with bagging), aggregation-based (use cluster/variance/nonuniformity statistics to pick subspaces).

2 Axis-Parallel Subspaces

Here, we select a subset of the features to represent a subspace. We can consider one point at a time and identifying the subspace where they are unique. Another option is to make a subspace model and score points with it (ensembling models here helps).

The first approach is a genetic algorithm. We take each feature and divide it into bins such that each bin has the same fraction $f = 1/\phi$ of the points. If we have k dimensions, we can create a grid where each cell should have about f^k fraction of the points. If it has much fewer than this, the points are probably outliers. If there are N points, the number of points in a cell has expected value Nf^k and standard deviation $\sqrt{Nf^k(1-f^k)}$. If the number of points in cell $\mathcal{D}$ is $n(D)$, we define the sparsity coefficient $S(\mathcal{D}) = \frac{n(\mathcal{D})-Nf^k}{\sqrt{Nf^k(1-f^k)}}$. This is a nice score (you can combine it with extreme value analysis), but there are a ton of grid cells, so we will to use a genetic algorithm to pick the best cells. We generate some candidate solutions, do recombination and mutation, and then make the next generation of solutions. Doing this iteratively helps us find more fit solutions. We represent each grid cell (a solution) as a string where we indicate the cell or put a "don't care' marker". We then do selection, where we rank solutions by fitness (score) and sample them with greater weight given to fitter solutions. We then do crossover where we greedily pick the k best subspaces from $2k$ subspaces chosen from the two parents. We then do mutation where we randomly change values in the candidate. We repeat this until convergence (95% of solutions are the same).

The *HOSMiner* approach defines the following. For point X, we find subspaces where sum of its k-nearest neighbor distances are at least δ. There are some tricks to make this fast.

Feature Bagging is a powerful technique here. We randomly pick an integer r between $\lfloor d/2 \rfloor$ to $d-1$. We then randomly pick r features without replacement. We run our outlier algorithm on this subset. We do this many times and ensemble the results. You can use many different outlier algorithms too, but you need to normalize the scores (and account for the subspace dimensionality) to combine them (sum them or sum ranks). Use LOF or average k nearest neighbors as your outlier algorithm.

You can also do random clustering in different subspaces and ensemble them.

Subspace histograms are another technique. First, randomly pick a subspace. Then, sample some points and build a histogram. Now, normalize the histogram. Next, score each point using the histogram. Do

this whole process many times and average the results for each point. The bins of the histogram can be representing by hashing their coordinates and using a hashtable to look them up. There are some hyperparameters, but there are hueristics to pick them (see the actual book).

Isolation forests are similar to random forests, which are successful in practice. An isolation tree randomly picks features and randomly picks a split point for each feature and splits the data recursively until each point is in its own leaf. Outliers will be the points closest to the root of the tree, if you average over many trees (i.e. the forest). This definition of isolation forest has no parameters, but it tends to be inefficient if we want to make every point have its own leaf. To speed it up, we can pick a sample of the data (e.g. size 256), train a tree, and score all the points with the tree. We can also limit the depth of the tree. To help pick promising features, we can standardize each feature to zero mean and unit variance and then compute the Kurtosis $K(z_1, ..., z_N) = \frac{1}{N}\sum_{i=1}^{N} z_i^4$. Non-uniform features have high Kurtosis. If you limit the height of the tree (e.g. limit to 10), you need to account for this by assigning a credit score. If the leaf node has r points, the credit is $c(r) = \ln(r-1) - \frac{2(r-1)}{r} + 0.5772$, and should be added to the path length.

We've talked about picking random subspaces, but can we pick high contrast subspaces (HiCS) somehow? Once we find these subspaces, we can run an algorithm like LOF on each one and ensemble the results. Indexing our dimensions as $\{1...p\}$, we observe that $P(x_1|x_2, ..., x_p) = P(x_1)$ if the dimension is uncorrelated with the others. To compute the probabilities, we select a random rectangular region, take the points from there, and compute the relative frequencies (we might repeat this M times and average). We then check for independence using hypothesis testing (e.g. with Student's t distribution). We pick these subspaces using the Apriori algorithm and pruning subspaces. This can be computationally expensive though, so a multidimensional Kurtosis measure might be preferable.

Now let's assume we have a single point and want to find subspaces where it looks like an outlier. The $OUTRES$ method searches for subspaces where the data are nonuniform in the locality of the test point. The local density of the point $\bar{X}$ in subspace S is defined as $den(S, \bar{X}) = |\mathcal{N}(\bar{X}, S)| = |\{\bar{Y} : dist_S(\bar{X}, \bar{Y}) \leq \epsilon\}|$. Then, we need to see if $N(\bar{X}, S)$ is uniformly distributed (null hypothesis H_0) or not (H_1). The Kolmogorov-Smirnoff goodness-of-fit test can do this hypothesis test. Once we find many subspaces that pass the test, we can combine outlier scores from those subspaces $OS(\bar{X}) = \prod_i O(S_i, \bar{X})$ (here, low scores mean the point is more likely to be an outlier). To get $O(S_i, \bar{X})$, we first define $dev(S_i, \bar{X}) = \frac{\mu - den(S_i, \bar{X})}{2\sigma}$ where μ and σ are the mean and standard deviations of points in the neighborhood of $\bar{X}$. Then we set $O(S_i, \bar{X}) = \frac{den(S_i, \bar{X})}{dev(S_i, \bar{X})}$ if $dev(S_i, \bar{X}) > 1$ and we set $O(S_i, \bar{X}) = 1$ otherwise. The book has pseudocode for the $OUTRES$ algorithm.

There's a distance-based way to look for interesting subspaces for a given point. First, we let $S(\bar{X})$ be the k nearest neighbors of $\bar{X}$. The subspace $Q(\bar{X})$ is the set of dimensions of $S(\bar{X})$ where variance is small (according to a threshold). $G(\bar{X}$ is the Euclidean distance of $\bar{X}$ to the centroid of $S(\bar{X})$. We normalize this by the number of dimensions in $Q(\bar{X})$ to get $SOD(\bar{X}) = G(\bar{X})/Q(\bar{X})$. This is not a great approach though because it considers dimensions independently.

3 Generalized Subspaces

Sometimes axis-aligned subspaces don't work. In these cases, the data lies on local nonlinear manifolds. We'll combine PCA-like methods with axis-parallel subspace methods to do this.

One approach is to run a clustering algorithm and then take each point's Mahalanobis distance to nearest cluster centroid as outlier score. Ensemble this by using many clusterings (and perhaps many clustering algorithms).

If you are willing to pay $O(N)$ time for each point (so $O(N^2)$ time total), you can do this. First, find the k nearest neighbors of $\bar{X}$. Then do PCA on $\bar{X}$ and its neighbors to figure out its locality sensitive outlier score.

The Rotated Subspace Sampling improves on Feature Bagging. We recommend using LOF for the outlier analysis. First, we randomly rotate the axes. Then, we randomly pick $r = 2 + \lceil\sqrt{d}/2\rceil$ directions (see

book for how this is done) from the axes and project the points onto this subspace. Then, we run the outlier detector on this subspace. We repeat this many times and average (or take the maximum) scores. Make sure to standardize the scores.

In the case of nonlinear subspaces, you need to use spectral methods. First, find k nearest neighbors (with a kernel for similarity) and find the top eigenvectors with PCA. The similarity matrix is noisy because the data is high dimensional. Using a spectral embedding can clean up the similarity matrix. When we clean up the matrix, we can find the k nearest neighbors again and use PCA again. We repeat this a few more times.

You can also solve d regression problems and look for outliers with regression.

4 Discussion of Subspace Analysis

High dimensionality means we have some uncorrelated features, which means we have noise when measuring distance/similarity. The data lie on a low dimensional (and likely nonlinear) manifold, which we don't know about. This is why subspace analysis is hard. You really need to use ensembles to get good results with subspace analysis.

5 Conclusions and Summary

To deal with high dimensional data, we need to find subspaces where the outliers present themselves.