L20: Outliers

In almost all data sets there are outliers. This prevalence of outliers is even more prominent in large datasets since these are often gathered through some automated system. This means that there is likely no one manually scanning for anomalies. And modern day sensing systems typically favor ease of gathering data over accuracy. Verification is often the most expensive parts, and people assume that we can deal with them later.

That is the topic here, dealing with outliers ... and other sorts of noise.

In general, *outliers are the cause of, and solution to, all of data mining's problems*. That is, any problem in data mining people have, they can try to blame it on outliers. And they can claim that if they were to remove the outliers the problem would go away. That is, when data is "nice" (there are no outliers) then the structure we are trying to find should be obvious. It turns out the solution to the outlier problem is at the same time simple and complicated.

Here we will discuss two general approaches for dealing with outliers:

- Find and remove outliers
- Density-based Approaches
- Techniques which are resistant to outliers.

20.1 Removal of Outliers

The basic technique to identify outliers is very straightforward, and in general should not be deviated from:

1. Build a model M of dataset P
2. For each point $p \in P$ find the residual $r_p = \mathtt{d}(M(p), p)$, where $M(p)$ is the best representation of p in the model M.
3. If r_p is too large, then p is an outlier.

Although this seems simple and fundamental, we needed to wait until this part of the class to define this properly.

- We have seen a variety of techniques for building models M (clusters, regression).
- We have seen various distances that could make sense as $\mathtt{d}(M(p), p)$, and their trade-offs.
- We understand the difficulties of choosing a cut-off for what constitutes "too large."

It should also be clear that there is not one "right" way of deciding *any* of these steps. Should be do clustering or linear regression or polynomial regression? Should we look at vertical of projection distance, use L_1 or L_2 distance? Should remove the furthest k points, or the furthest 3% of points?

For consider a set P that is drawn from a standard 1-dimensional Gaussian distribution. We could model this as a single cluster using the mean as a center. The the residual is the distance to the center. Now which points are outliers? Lets compare then against standard deviations?

- If we remove all points at 1 standard deviation: we remove about 1/3 of all points.
- If we remove all points at 2 standard deviations: we remove about 1/20 of all points.
- If we remove all points at 3 standard deviations: we remove about 1/300 of all points.
- If we remove all points at 4 standard deviations: we remove about 1/16000 of all points.

So what is the right answer? ...
Well, there is no "right" answer. If we fit the model with the right center, then why did we care about outliers in the first place; we still mined the right structure M! (Perhaps you say $M(p)$ should not be the mean, but some measure of how well it "fits" the Gaussian, but even in this approach you will reach a similar conclusion.)

Another question is: does process "converge"? ...
If we remove the outliers from P, and then repeat, do we still find more outliers? Say we always remove the 10 points with highest residuals. Then we can repeat this $|P|/10$ rounds until there are no points left. So you must have a way to determine if you should stop.

Down-weighting. One alternative to completely removing outliers is to just down-weight them. That is, when say finding the mean in k-means clustering, give inlier points a weight of 1 (as normal) and outlier points a weight of say $1/2$ or $1/10$. This number may depend on the residual.

A more formal way of doing this can be through the use of a kernel (such as a Gaussian kernel), or some other similarity. Where points are weighted based on their similarity to the model M. Given a model M, points are reweighted, then the model is recomputed, and points are reweighted, and so on ...

20.2 Density-Based Approach

The idea is: (1) regular points have dense neighborhoods, and (2) outlier points have non-dense neighborhoods. Using the distance to the closest point is not robust, so we can use the distance to the kth closest point as measure of density. But what is k? Alternatively, we can measure density by counting points inside of a radius r ball, or more robustly using the value of a kernel density estimate (using kernel with standard deviation r). But this requires a value r?

So this techniques needs a value k or r, and then another threshold to determine what is in, and what is out. Sometimes the value k or r is apparent from the applications, and we can use an "elbow" technique for determining outliers.

But more seriously, this assumes that the density should be uniform throughout a data set. The "edge" will always have less density.

Reverse Nearest Neighbors: Given a point set P for each point $p \in P$ we can calculate it nearest neighbor q (or more robustly its kth nearest neighbor). Then for q we can calculate its nearest neighbors r in P (or its kth nearest neighbor). Now if $\mathbf{d}(p,q) \approx \mathbf{d}(q,r)$ then p is probably not an outlier.

This approach adapts to the density in different regions of the point set. It is also expensive to compute. And density (or relative density) may not tell you how far a point is from a model.

Heavy-Tailed Distributions: These types of distributions, which appear more and more frequently at internet scale, serve as a warning towards certain density filters for outliers.

Consider Zipf's Law: *the frequency of data is proportional to its rank.*

Let X be a multiset so each $x \in X$ has $x = i \in [u]$. For instance all words in a book. Then $f_i = |\{x \in X \mid x = i\}|/|X|$. Now sort f_i so $f_i \geq f_{i+1}$, then according to Zipf's law: $f_i \approx c(1/i)$ for some constant c.

For instance, in the Brown corpus, an enormous text corpus the three most common words are "the" at 7% ($f_{\text{the}} = 0.07$); "of" at 3.5% ($f_{\text{of}} = 0.035$); and "and" at 2.8% ($f_{\text{and}} = 0.028$). So the constant is roughly 0.07.

This also commonly happens when looking at customers at large internet companies. Amazon was built in some sense on the heavy tail.

- The top 10,000 books can be sold at physical book stores.

- The top 1 million books can be sold threw Amazon., since they only need to sell say 1000 of them in the entire country to break even. They still stock these in their central warehouse.
- The top 100 million books can now still be sold by Amazon, but at a higher price; they can print them to order.

This is easiest to see with books (and hence Amazon was built that way), but this happens with all sorts of internet products: movies, search results, advertising, food.

This has various screwy effects on a lot of the algorithms we study. And, to repeat, this rarely is seen at small scale, this only really comes into the picture in large rich data sets. Here are some of the screwy things that can happen:

- When doing PCA, there is not sharp drop in the singular values (which is *very* common with small datasets).
- If 30% of words in a book occur less than 0.0001% of the time (say occur at most twice), then can we just ignore them. Many LDA (text topic modeling softwares) do this, but should they really throw away 30% of data. Many NLP people think this puts an artificial cap on what LDA can do for text.
- We can find different structure at different scales, e.g. hierarchical PCA. Perhaps, each principal component is actually a series of 15 clusters (that happen to line up), and each of the clusters should have a component that points in a different direction. Think of classification of customers, or structure of human body (from atom, to protein, to cell, to organ, to being). Sometimes the components interact in ways that were not apparent at a large scale, so a pure hierarchy is never quite correct.

20.3 Robust Estimators

If you can't beat them, embrace them.

The main problem with the first approach is that in order to find a model M to build residuals $\{r_p = \|p - M(p)\| \mid p \in P\}$ on determine outliers $\{p \in P \mid r_p > \tau\}$, is that it needs a good model M. But if we already have a good model M, then why do we can about outliers?

So here we discuss properties of techniques that build a model M and are resistant, or *robust*, to outliers. Given a model $M(P)$, its *breakdown point* is an upper bound of the fraction of points in P that can be moved to ∞ and for $M(P)$ not to also move infinitely far from where it started. For instance for single point estimators in the 1 dimension, the mean has a break-down point of $1/n$, while the median has a breakdown point of $1/2$. An estimator with a large breakdown point is said to be a *robust estimator*.

In general, most estimators that minimize the sum of square errors (like PCA, least squares, k-means, and the mean) are *not* robust, and thus are susceptible to outliers.

Techniques that minimize the sum of errors (like least absolute differences, Theil-Sen estimators, k-median clustering, and the median) are robust, and are thus not as sensitive to outliers. Notice an analog to PCA is not there. This is an open research question of what the best answer is, as far as I know.

However, many of the L_1-regularization techniques (like Lasso) have the easy-to-solve aspects of least squares, but also simulate some of the robustness properties.