Chapter 9 - Time Series and Multidimensional Streaming Outlier Detection

1 Introduction

Assume we have a time series and want to find outliers within it. One kind of outlier is an abrupt change in the value, which is useful for series with high continuity (e.g. two consecutive sensor readings are usually almost identical). In cases where we have low continuity (e.g. news articles coming down a wire) we have some overall trend, but consecutive points may be a good deal different from each other. Here, we want to detect novelties (points very different from historical points) and changes (a change to the overall trend of the series).

A contextual outlier is a point that is different from its neighbors. A collective outlier is a subsequence of contiguous points that are different from the rest of the neighbors.

Our algorithms might run offline or online. Offline gives us access to history and lets us use more complex algorithms. We might have labels for time series, chunks of time series, or for points. In this case, use supervised learning.

2 Prediction-Based Outlier Detection in Streaming Time Series

We often care about deviations (contextual outliers). We care about correlations accross time and accross different series. An autoregressive model, $AR(p)$ defines the current time series value as a function of the past: $X_t = \sum_{i=1}^{p} a_i X_{t-i} + c + \epsilon_t$ for window size p and error terms (outlier scores) ϵ_t. We can infer parameters with least squares as follows. Let D be a matrix where column i is $X_i, ..., X_{n-p+i-1}$ and the final column is all 1. Let $\bar{y}$ be the column vector $X_{p+1}, ..., X_n$ and let Θ be the column vector $a_p, a_{p-1}, ..., a_1, c$. We minimize $||\bar{y} - Da||^2$ and get $a = (D^T D + \alpha I)^{-1} D^T \bar{y}$ where α is a regularization parameter. You can also do this online with an efficient algorithm for matrix inversion.

The moving average model $MA(q)$ is $X_t = \sum_{i=1}^{q} b_i \epsilon_{t-i} + \mu + \epsilon_t$ - you need nonlinear methods to fit this. The $ARMA(p, q)$ model is $X_t = \sum_{i=1}^{p} a_i X_{t-i} + \sum_{i=1}^{q} b_i \epsilon_{i-1} + c + \epsilon_t$. If p and q are too large, you overfit. Pick them with leave-one-out cross validation. Sometimes, a time series is nonstationary (e.g. random walk). So, we take successive differences and then use $ARMA(p, q)$, this is $ARIMA(p, q)$. We also might take a logarithm of each value before taking differences.

We can extend all the above models to multiple time series. For example: $X_t^j = (\sum_{k=1}^{d} \sum_{i=1}^{p} a_i^{kj} X_{t-i}^k) + c^j + \epsilon_t^j$. This is computationally intensive because the matrix is larger. So, you can just select a subset of streams. Alternatively, you can create hidden variables and do univariate time series analysis for each. The Muscles technique uses the matrix inversion lemma can help do matrix inversion incrementally as you add new points, but is slow when you have too many series, so it uses a subset of them. Basically, given a stream that we want to predict, we greedily pick the most correlated streams until we have enough (this algorithm is called Selective Muscles). The problem with these multiple time series method is they have a lot of coefficients and do not handle outliers well.

The SPIRIT algorithm handles multiple time series with PCA. We compute the $d \times d$ covariance matrix between streams and project each data point onto the top k eigenvectors (the rest are basically just constant values). This yields k uncorrelated streams that can be processed independently. We can then transform the results back (along with the $d - k$ constant streams) to the original space. Then you can compute the error terms as usual. In order to compute a running covariance, you need to keep a running sum for each series and a running pairwise sum for each pair. This lets you compute $Cov(\bar{X}^j, \bar{X}^k) = \frac{\sum_{i=1}^t X_i^j X_i^k}{t} - \frac{\sum_{i=1}^t X_i^j}{t} - \frac{\sum_{i=1}^t X_i^k}{t}$.

You can represent groundtruth by getting a set of outlier timestamps $T_1, ..., T_r$ (these are the primary abnormal events - we will also find other anomalies called secondary abnormal events). First, run one of the previous time series analysis algorithms and compute error terms. Normalize them to zero mean and unit variance: $z_t^1, ..., z_t^d$. The alarm level at time t is then $Z_t = \sum_{i=1}^d \alpha_i z_t^i$. The alarm level during primary events is $Q^P(\alpha_1, ..., \alpha_d) = \frac{\sum_{i=1}^r Z_{T_i}}{r}$ and during normal events is $Q^n(\alpha_1, ..., \alpha_d) = \frac{\sum_{i=1}^n Z_i}{n}$. We aim to minimize $Q^P(\alpha_1, ..., \alpha_d) - Q^n(\alpha_1, ..., \alpha_d)$ subject to the regularization $\sum_{i=1}^d \alpha_i^2 = 1$. Learn this with some off the shelf optimizer.

3 Time Series of Unusual Shapes

How do we find collective outliers? We may want to detect if an entire time series has an anomalous shape. Alternatively, we may want to find subsequences within a time series that are anomalous. We focus on the full series case because you can always solve the subsequence case by breaking a time series into small pieces. Let us assume all series are normalized to zero mean and unit variance and that all series have the same length n.

Numeric multidimensional transformation (NDT) lets us take a time series and turn it into a vector that we can compare to other time series. Discrete sequence transformation (DST) turns our time series into sequences of symbols that we can analyze using methods in chapter 10.

For NDT, we can use the Haar Wavelet. The first element of our vector is the global average. We then split the series in half and compute difference between the averages of each half. Then recursively process each half to get the remaining $n - 2$ coefficients. We cannot drop any of the n coefficients because they may be useful for outlier detection (you can also compare them with Euclidean distance). Now, we have a bunch of n dimensional vectors with no temporal dependency, so we can use other outlier techniques. The discrete Fourier transform is an alternative to the Haar Wavelet. Use Fourier transform when you have seasonality and use wavelets when you have series that usually do not change much over small ranges.

For DST, we use Symbolic Aggregate Approximation. First, we split the time series into windows and compute the average of each window. Now we assume the averages are distributed as a Gaussian, which we break into equal probability mass bins and assign each average its bin ID (we now have each bin ID appearing roughly the same number of times).

To detect unusual shapes, you can treat the time series as a trajectory and use the methods from Chapter 11. TROAD is a good algorithm for this. This is a hard problem.

The Hotsax technique runs a sliding window over the series, converts each into a multidimensional vector, and then computes each window's Euclidean distance to other windows to find outliers. You could also use Dynamic Time Warping instead of Euclidean distance. To make this computationally efficient, we can use pruning. We have a nested for-loop where the inner loop looks for the k nearest neighbors. We keep a running update of the top-r outliers, so we can terminate the inner loop early if it is clearly not going to be in the top-r. There are other techniques for doing this (see book).

If you turn a time series into a sequence of symbols, you can use a Hidden Markov Model.

You can slide a window over the series, treat each window as a point, and use PCA. You can also do this if you have multiple time series. Use kernel PCA if you want non-Euclidean similarity measures like Dynamic Time Warping or edit distance. You can also use one-class SVM in this way.

If you have labeled data, learn a model for the normal case and abnormal class. Then just measure distance to both to see which is a better fit. Alternatively, convert your time series into symbols and use a Hidden Markov Model to classify.

4 Multidimensional Streaming Outlier Detection

Autocorrelation and continuity may be weaker in multiple time series than in univariate time series. Again, we may have deviation (novelty) outliers and collective (change in trend) outliers.

To detect novelties, just keep a window of points and measure distance to each point. The LOF algorithm has been extended to the incremental scenario. You can also keep running clusters and measure distance to them. A novelty is far from existing clusters and creates a new cluster. You can also build a mixture model (they are well suited for online algorithms). The SPOT algorithm is good for high dimensional streaming time series.

To detect trend changes, you can use velocity density estimation (see book for the map). This is like kernel density estimation, but using temporal kernels. You can set the kernel width to determine long term or short term trends. There are also techniques to fit probability distributions to sections of the time series and look for statistically significant differences.

Supervision can indicate rare-class outliers, novel class outliers, and infrequently recurring outliers. You also need some unsupervised learning to detect completely new kinds outliers. For rare classes, you can use a classifier that can handle nonstationary distributions. You need to account for the class imbalance here (see chapter 7). For novel classes, you should have your regular supervised classifier and an unsupervised clustering model for novel classes so that you can tell if a novel class (i.e. not in training set) has been seen before during test time. For infrequently recurring outliers, you could just mark them as novel, but it's better to remember a distribution for this outlier and store it in case you see it again.

5 Conclusions and Summary

Time series can be univariate or multiple series. Outliers can be novel or collective. You can use supervised, unsupervised, and hybrid techniques.