ANOMALY DETECTION IN TIME SERIES

Ryan Lapcevic and Steve Robbins | AC209A | Harvard University

Project Overview

Create an algorithm to detect anomalies on-line for time series data

In many problem domains it is desirable to analyze data dynamically as they are collected over time. This introduces a number of problems for traditional machine learning algorithms, many of which assume the hypothesis of exchangeability applies to the data set. The principle of exchangeability essentially means that past data are representative of future data and can therefore be used to make inferences. In these contexts it may be necessary to test this hypothesis in real time to identify whether the causal factors underlying the data have changed. In this project we have implemented an algorithm to test this hypothesis for streaming data using the randomized power martingale, a statistical tool for determining whether exchangeability is maintained.

Key Concepts

Time Series:

- A time series is a sequence of data points indexed by ordered time. Many datasets fit this description, such as stock market indices, meteorological measurements, and network latency.

Exchangeability:

- If a sequence of examples $z_1, z_2, \cdots, z_n$ is invariant under any permutation of the indices, the distribution is *exchangeable*.
- In other words, the probability of the sequences does not change under reordering, i.e. all sequences with the identical occurrence of examples have same probability (i.e. 1/n!)

Example:

Suppose we have a jar with one black ball (*b),* one white ball *(w),* and one green ball *(g);* and we take the balls out of the jar without replacement. Let X_1, X_2, and X_3 be the sequence by which the balls are drawn.

- **Independent and Identical Distribution (i.i.d.):** Consider the scenario where X_1, X_2, X_3 is *b, w, g,* respectively. Clearly this scenario *violates independence* as once the first ball is drawn *(X_1,* with P(*b*) = 1/3*),* the probability of drawing a *w* or *g* next *(X_2,* P(*w*) = P(*g*) = 1/2*)* are *dependent* on which ball is drawn first (here, X_1 = *b*).
- **Identical Distribution:** The sequence *is identically distributed* as P(X_1, X_2, or X_3) = *b* is identical to P(X_1, X_2, or X_3) = *w* and P(X_1, X_2, or X_3) = *g*.
- **Exchangeability:** The X_1, X_2, and X_3 sequence *is exchangeable* as all of the six (3!) permutations of X_1, X_2, X_3 occur with equal probability (1/3! = 1/6). In other words, P(X_1, X_2, X_3) = {*b, w, g*} = 1/6. The P(X_1, X_2, X_3) = {*g, w, b*} = 1/6. Since every sequence occurs with equal probability, the sequence is *exchangeable.*

All sequences that are i.i.d. are exchangeable. All sequences that are exchangeable are not i.i.d., but all exchangeable sequences are identically distributed.

Data sequences containing subset-sequences that violate exchangeability indicate the presence of anomalies in the data.

Algorithm Flow Chart

Kernel Function

The kernel function generalizes the familiar notion of 'distance' between data points. In our algorithm, we use the Gaussian kernel, which is applicable to a diverse array of different problem domains. More refined kernels can be used to inject domain-specific knowledge into the algorithm's notion of distance.

Adiabatic Iterative SVM / 'Strangeness' Function

Traditional support vector machines (SVM's) work on complete training datasets to identify sets of hyperplanes that separate data points into classes. This approach is problematic for on-line problems, where storing and computing with the complete dataset is often computationally infeasible. Our algorithm implements adiabatic iterative SVM to compute support vectors for new data in an interactive fashion using the support vectors computed using prior data. This approach uses the kernel function to separate data points and compute a strangeness value whose magnitude reflects how different a given data point is from the rest.

Randomized Probability Function

The randomized power martingale function uses the randomized probability of observing a given data point given the prior observed data.

$$p_n = \frac{\#\{i : \alpha_i > \alpha_n\} + \theta_n \#\{i : \alpha_i = \alpha_n\}}{n}$$

Martingale Function

To determine whether a concept change has occurred, we compare the absolute difference of the current and prior martingale function values to a positive constant, t. If t is exceeded, we conclude that a concept change has occurred.

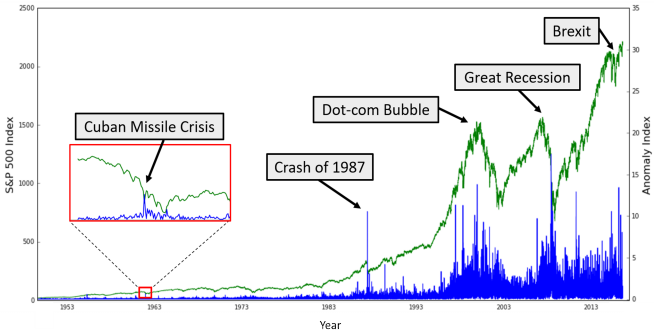
$$0 < |M_n^{(\epsilon)} - M_{n-1}^{(\epsilon)}| < t$$

Sample Code

```
def randomized_power_martingale(stream, strangeness_fn):

    pval = 1        # Likelihood of obtaining data observed so far
    Mns = []        # List of martingale values for each data point
    Mn = 1          # Value of martingale up to nth data point
    alphas = []     # Store list of observed strangeness values
    n = 0           # Number of observed data points
    epsilon = 0.92  # Define Martingale family to use

    # Iterate through data stream
    data_point = stream.next()
    while data_point is not None:
        n += 1      # Increment number of observed data points

        # 1: Compute and store scalar `alpha` measuring 'strangeness' of the nth data point
        an = strangeness_fn(data_point)
        alphas.append(an)

        # 2: Update likelihood of observing the n data points
        theta = random() # Random real between 0 and 1
        num_a_greater = len([a for a in alphas if a > an]) # Number of points stranger than current
        num_a_equal = len([a for a in alphas if a == an]) # Number of points as strange as current
        pval = (num_a_greater + theta*num_a_equal)/n # 'Randomized' pval at point `n`

        # 3: Compute and store value of power martingale at point n
        Mn *= epsilon*pval**(epsilon-1)
        Mns.append(Mn)

        # 4: Move to next data point
        data_point = stream.next()

    # Return all martingale values computed for time series
    return Mns
```

Stock Market Data / Martingale

S&P Index and Anomaly Index over Time



Results & Observations

The empirical results suggest that our martingale-based anomaly detection algorithm successfully identifies concept changes in streaming data. The above plot shows that the value of the martingale changes rapidly during periods of abrupt change in financial markets as measured by the S&P 500 Index. This indicates that the martingale function serves as an effective metric for the significance of change observed in streaming data.

While our algorithm successfully identifies the occurrence of anomalies, it does not provide any information on the exact nature of the anomaly. For instance, the above plot highlights periods of abrupt change in financial markets, but it does not distinguish between market crashes and rallies. In more complicated problem domains, such as the detection of computer network anomalies, several classes of anomaly may be possible. Potential extensions of our algorithm might attempt not only to identify the occurrence anomalies, but also to classify them depending on additional information gleaned from the time series. This would entail a more sophisticated analysis of the time series data stream, perhaps employing an iterative singular value decomposition to extract features as the data are received.

Works Cited

- Gert Cauwenberghs and Tomaso Poggio, "Incremental and Decremental Support Vector Machine Learning," *Advances in Neural Information Processing Systems*, vol. 13, 2001.
- Vladimir Vovk, Ilia Nouretdinov, and Alex J. Gammerman, "Testing Exchangeability Online", ICML 2003.
- Shen-Shyang Ho and Harry Wechsler, "A Martingale Framework for Detecting Changes in Data Streams by Testing Exchangeability," *Pattern Analysis and Machine Intelligence, IEEE Transactions*, vol.32, no.12, 2113-27, 2010.
- Valentina Fedorova, Alex J. Gammerman, Ilia Nouretdinov, Vladimir Vovk, "Plug-in martingales for testing exchangeability on-line", ICML 2012.