Queries over a Data Stream

By Andrew Cash, and Erik Wood

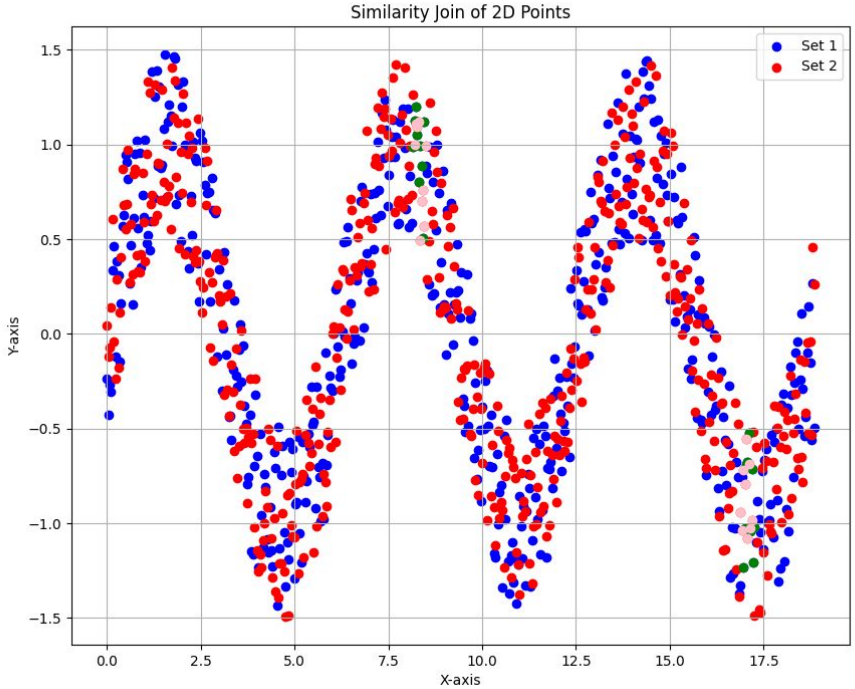
Intro - 1

The Similarity Join Query is useful for finding correlations between different streams of data. Specific use cases include cleaning sensor data, finding patterns in stock prices, and internet traffic analysis.

Intro - 2

The similarity join query (SJ) in the context of multiple time-series data streams has multiple applications in the field of data engineering. Given two time-series databases *R* and *S*, an SJ query will return all possible pairs of subsequences, <r, s>, in which the distance between r and s is less than or equal to a threshold variable, ε (epsilon). To dynamically maintain stream data, a sliding window structure is used, where old data is pushed out as new data is pushed in.

Figure 1



Here, the red and blue dots represent different series of data. The pink and green dots represent subsequences of size w that are similar, since the Euclidean distance between them is less than the threshold ε

Intro - 3

To calculate each similar subsequence within each time series, a variety of approaches can be used.

The simple one is brute force, where every possible subsequent is compared. Naturally, this is lacking in performance.

Intro - 4

A more novel way may use indexing and dimensionality reduction.

Fast subsequence matching in time-series databases (https://dl.acm.org/doi/pdf/10.1145/191843.191925)

suggests using a sliding window over each stream of data, S_i, and capturing its features using Discrete Fourier Transform. This is then used to index an R*-tree, which allows for efficient access of the original data.

Intro - 5

Our method is a little bit different. The goal was to index each non-query series, S_i, using a B+-tree. The B+-tree would allow for range searches over the data for effective pruning, and even supports the nature of time-series data, since the leaf nodes comprise a linked list.

Intro - 6

Consider the first point in a query sequence q. If this point is the first coordinate in a sliding window, then any similar points must be within at least Epsilon distance away from it.

This can be proven by considering the most extreme case of a legal similarity join. To calculate the distance between subsequences, we take the sum of their corresponding coordinator points. If one point is Epsilon distance away from its corresponding point, then all other points must be equal to their corresponding points. Essentially, any points past the radius Epsilon can be pruned.

Project Description - 1

Brief descriptions of your project: Implementing and evaluating variations of techniques for the Similarity Join operation over multiple streams of time-series data. Displaying the data queried in various ways to better understand what the data represents.

Project Description - 2

Challenges and technical contributions (new problems or new solutions?) In our project we are using a new technique to perform similarity joins on a data series to query results that have similarities.

Project Description - 3

The workload distribution for each member in your team: Is 50/50 split because we only have two people in our group so it makes the most sense to split the work evenly based on each other strengths and weaknesses.

Project Background - 1

Related papers:

Efficient Similarity Join Over Multiple Stream Time Series by Xiang Lian, Student Member, IEEE, and Lei Chen, Member, IEEE.

Fast Subsequence Matching in Time-Series Database by Christos Faloutsos, M. Ranganathan, and Yannis Manolopoulos

Software tools: Python, Numpy, Pandas, Scipy, Matplotlib.

Project Background - 2

Required hardware: A computer to run the queries, and a stream of data. Related programming skills: Ability to use python and libraries in python.

Problem Definition - 1

Formal (mathematical) definitions of problems: Whenever T_i gets a new data item T_i[t+1], Similarity join obtains and then outputs all subsequences from stream time series T_i(1 <= j <= m) that epsilon-match with the new subsequence

S_new = T_i[t - n + 2 : t + 1] from T_i, where n = s * w.

Problem Definition - 2

Challenges of tackling problems: Are the ease of implementing mathematical problems and getting results to be accurate.
A brief summary of general solutions in your project: Taking a data stream and running a query over two time-series databases and returning all possible pairs of the subsequent data.

Figure 2

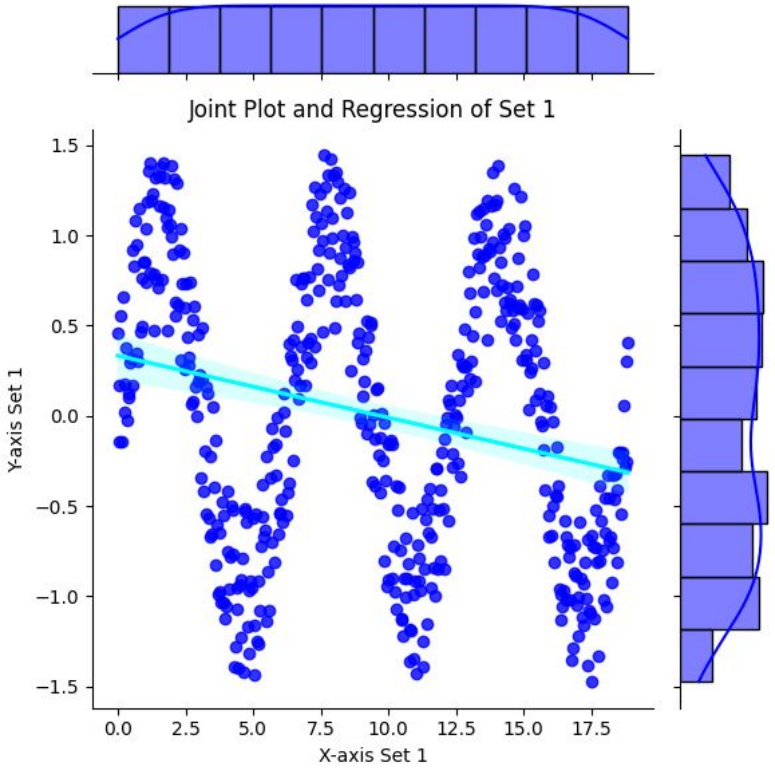

Joint Plot and Regression of Set 1
Y-axis Set 1
X-axis Set 1
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

Figure 3

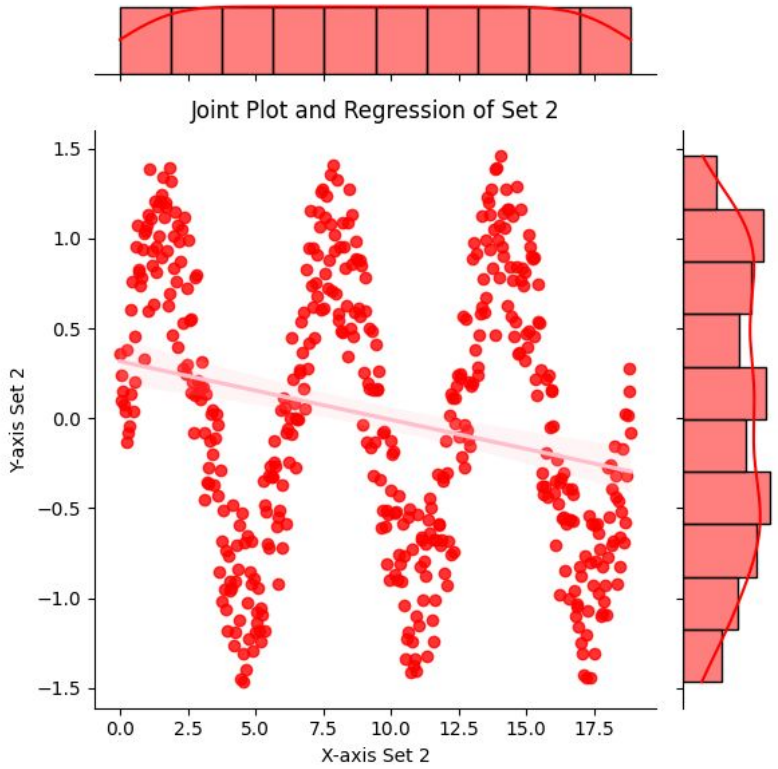

Joint Plot and Regression of Set 2
Y-axis Set 2
X-axis Set 2
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

Figure 4