Detecting Data Stream Dependencies on High Dimensional Data

Jonathan Boidol[1,2] and Andreas Hapfelmeier[2]
[1]*Institute for Informatics, Ludwig-Maximilians University, Oettingenstr. 67, D-80538, Munich, Germany*
[2]*Corporate Technology, Siemens AG, Otto-Hahn-Ring 6, D-81739, Munich, Germany*

Keywords: Sensor Application, Online Algorithm, Entropy-based Correlation Analysis.

Abstract: Intelligent production in smart factories or wearable devices that measure our activities produce on an ever growing amount of sensor data. In these environments, the validation of measurements to distinguish sensor flukes from significant events is of particular importance. We developed an algorithm that detects dependencies between sensor readings. These can be used for instance to verify or analyze large scale measurements. An entropy based approach allows us to detect dependencies beyond linear correlation and is well suited to deal with high dimensional and high volume data streams. Results show statistically significant improvements in reliability and on-par execution time over other stream monitoring systems.

1 INTRODUCTION

Large-scale wireless sensor networks (WSN) and other forms of remote monitoring, reaching from personal activity to surveillance of industrial plants or whole ecological systems are advancing towards cheap and widespread deployment. This progress has spurred the need for algorithms and applications that work on high dimensional streaming data. Streaming data analysis is concerned with applications where the records are processed in unbounded streams of information. Popular examples include the analysis of streams of text, like in twitter, or the analysis of image streams, like in flickr. However, there is also an increasing interest in industrial applications. The nature and volume of this type of data make traditional batch learning exceedingly difficult, and fit naturally to algorithms that work in one pass over the data, i.e. in an online-fashion. To achieve the transition from batch to online algorithms, window-based and incremental algorithms are popular, often favoring heuristics over exact results.

Instead of relying only on single stream statistics to e.g. detect anomalies or find patterns in the data, this paper is concerned with a setting where we find many sensors monitoring in close proximity or closely related phenomena, for example temperature sensors in close spacial proximity or voltage and rotor speed sensors in large turbines. It appears obvious that we should be able to utilize the – in some sense redundant, or rather shared – information between sensor pairs to validate measurements. The task at hand becomes then to reliably and efficiently compute and report dependencies between pairs or groups of data streams. We can imagine such a scenario in the context of smart homes or smart cities with personal monitoring or automated manufacturing that form the internet of things. A particular application could be the validation of sensor readings in the context of multiple cheap sensors where measurements are possibly impaired by limited technical precision, processing errors or natural fluctuations. Then, unusual readings might either indicate actual changes in the monitored system or be due to these measuring uncertainties. Finding correlations helps differentiate such cases.

The best known indicator for pairwise correlation is Pearson's correlation coefficient ρ, essentially the normalized covariance between two random variables. Direct computation of ρ, however, is prohibitively expensive and, more problematic, it is only a suitable indicator for linear or linear transformed relationships (Granger and Lin, 1994). Non-linearity in time-series has been studied to some extent and may arise for example due to shifts in the variance (Fernandez et al., 2002) or simply if the underlying processes are determined by non-linear functions.

We propose an algorithm that is used to detect dependencies in high volume and high dimensional data streams based on the *mutual information* between time series. The three-fold advantages of our approach are that mutual information captures global dependencies, is algorithmically suitable to be calculated in an incremental fashion and can be computed

Boidol, J. and Hapfelmeier, A.
Detecting Data Stream Dependencies on High Dimensional Data.
DOI: 10.5220/0005953303830390
In *Proceedings of the International Conference on Internet of Things and Big Data (IoTBD 2016)*, pages 383-390
ISBN: 978-989-758-183-0

efficiently to deal with high data volume without the need for approximation short-cuts. This leads to a dependency measure that is significantly faster to calculate and more accurate at the same time.

The remainder of this paper is organized as follows: We will present the background in information theory for mutual information, introduce the terminology to use it in a streaming algorithm and explain our main algorithm called MID in section 2. Section 3 contains the experimental evaluations on one synthetic and four real world datasets. We conclude and suggest possible future work in section 4.

2 MUTUAL INFORMATION DEPENDENCY

This section introduces the necessary background to the concept mutual information and shows our adaptation into *MID*, a convenient, global measure to detect dependencies between data streams.

2.1 Correlation and Independence

(Dionisio et al., 2004) argue that mutual information is a practical measure of dependence between random variables directly comparable to the linear correlation coefficient, but with the additional advantage of capturing global dependencies, aiming at linear and nonlinear relationships without knowledge of underlying theoretical probability distributions or mean-variance models.

StatStream(Zhu and Shasha, 2002) and PeakSimilarity(Seliniotaki et al., 2014) are algorithms to monitor stream correlation. Both employ variants of a discrete fourier transformation (DFT) to detect similarities based on the data compression qualities of DFT. More specifically, they exploit that DFT compresses most of a time series' information content in few coefficients and develop a similarity measure on these coefficients. The similarity measure for Peak Similarity is defined as

$$peak\,similarity(X,Y) = \frac{1}{n} \cdot \sum_{i=1}^{n} \frac{1 - |\hat{X}_i - \hat{Y}_i|}{2 \cdot max(|\hat{X}_i|, |\hat{Y}_i|)}$$

where X and Y are the time series we want to compare and $\hat{X}_i, \hat{Y}_i$ the n coefficients with the highest magnitude of the respective Fourier transformations.

The similarity measure of Stat Stream is similarly defined on the DFT coefficients as

$$stat\,stream(X,Y) = \sqrt{\sum_{i=1}^{n} (\bar{X}_i - \bar{Y}_i)^2}$$

but here $\bar{X}_i, \bar{Y}_i$ are the largest coefficients of the respective Fourier transformations of the *normalized* X and Y.

StatStream also uses hashing to reduce execution time, but the choice of hash functions is highly application specific. PeakSimilarity relies on a similarity measure specially defined to deal with uncertainties in the measurement, but requires in-depth apriori knowledge of a cause-and-effect model to do so.

We develop our own measure based on mutual information and compare its accuracy and execution time to the DFT-based measures and the correlation coefficient.

2.2 Mutual Information

Mutual information is a concept originating from Shannon information theory and can be thought of as the predictability of one variable from another one. We will exploit some of its properties for our algorithm. Since the mathematical aspects are quite well-known and described extensively elsewhere, e.g. (Cover, 1991), we will review just the basic background and notation needed in the rest of the paper. The mutual information between variables X and Y is defined as

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log \left(\frac{p(x,y)}{p(x)p(y)} \right) \quad (1)$$

or equivalently as the difference between the Shannon-entropy $H(X)$ and conditional entropy $H(X|Y)$:

$$\begin{aligned} I(X;Y) &= H(Y) - H(Y|X) && (2)\\ &= H(X) - H(X|Y) && (3)\\ &= H(X) - H(X,Y) + H(Y). && (4) \end{aligned}$$

Shannon-entropy and conditional entropy are defined as

$$H(X) = \sum_{x \in X} p(x) \log \left(\frac{1}{p(x)} \right) \quad (5)$$

$$H(X|Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log \left(\frac{p(y)}{p(x)p(y)} \right). \quad (6)$$

$I(X;Y)$ is bounded between 0 and $max(H(X),H(Y)) = \log(max(|X|,|Y|))$ so we can define a normalized $\hat{I}(X;Y)$ which becomes 0 if X and Y are mutually independent and 1 if X can be predicted from Y and vice versa. This makes it easily comparable to the correlation coefficient and also forms a proper metric.

$$\hat{I}(X;Y) = 1 - \frac{I(X;Y)}{\log(max(|X|,|Y|))}. \quad (7)$$

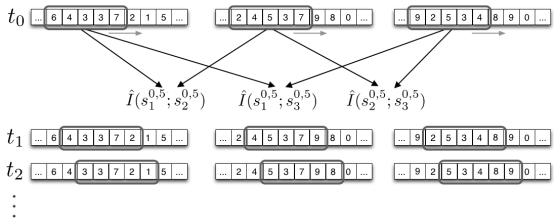


Figure 1: Sliding window and pairwise calculation of $\hat{I}$ for a data stream with window size $w = 5$ and $|S| = 3$.

Next, we want to compute $\hat{I}$ for pairs of streams $s_i \in S$ at times t. The streams represent a measurement series $s_i = (\dots, m_t^i, m_{t+1}^i, m_{t+2}^i, \dots)$ without beginning or end so we add indices $s_i^{t,w}$ to denote measurements from stream s_i from time t to $t+w-1$, i.e. a window of length w. $|S|$ is the *dimension* of the overall data stream S in the sense that every s_i represents a series of measurements of a different type and/or different sensor. We will drop indices where they are clear from the context. Our goal is then to efficiently calculate the stream dependencies D_t for all points t in the observation period $t \in [0; \inf)$

$$D_t^w = \{\hat{I}(s_i^{t,w}, s_j^{t,w}) | s_i, s_j \in S\}. \tag{8}$$

Figure 1 demonstrates the basic window approach for a stream with three dimensions.

2.3 Estimation of PDFs

Two problems remain to determine the probability distribution functions (PDFs) we need to calculate entropy and mutual information. First, data streams often contain both nominal event data and real values. Consequentially our model needs to deal with both continuous and discrete data types. Second, the underlying distribution of both single stream values and of the joint probabilities is usually unknown and must be estimated from the data.

There are three basic approaches to formulate a probability distribution estimate: Parametric methods, kernel-based methods and binning. Parametric methods need specific assumptions on the stochastic process and kernel-based methods have a large number of tunable parameters where sensible choices are difficult and maladjustment will lead to biased or erroneous results.(Dionisio et al., 2004) Binning or histogram-based estimators are therefore the safer and more feasible choice for continuous data which have been well studied (Paninski, 2003; Kraskov et al., 2008), and a natural fit for discrete data. They have been used convincingly in different applications.(Dionisio et al., 2004; Daub et al., 2004; Sorjamaa et al., 2005; Han et al., 2015)

Quantization, the finite number of observations and the finite limits of histograms – depending on the specific application – might lead to biased results. However (Dionisio et al., 2004) argue that both equidistant and equiprobable binning lead to a consistent estimator of mutual information.

Of the two fundamental ways of discretization - equal-width or equal-frequency - equal-width binning is algorithmically slightly easier to execute, since it is only necessary to keep track of the current minimum and maximum. Equal frequency binning requires more effort, but has been shown to be the better estimator for mutual information.(Bernhard et al., 1999; Darbellay, 1999) We confirmed this in a separate set of experiments and consequentially use equal frequency binning for our measure.

The choice of the number of bins b is a critical problem for a reliable method. (Hall and Morton, 1993) point out that histogram estimators may be used to construct consistent entropy estimators for 1-dimensional samples and describe an empiric method for histogram construction depending on the number of data points n in the sample and the expected range of values R. Their rule balances bias and variance components of the estimation error and reduces to $b \geq \frac{R}{n^{-0.32}}$. Typical ranges and sample sizes in our intended applications would result in a choice of $b \in [10, 100]$.

For our algorithm, we discretize on a per-window-basis. A window-wise discretization gives us a local view on the data since it depends only on the properties of the data in the window but is also limited to the data currently available in the window. We call $\hat{I}(X;Y)$ with per-window discretization **MID** – mutual information dependency. For greater clarity, we add pseudocode for MID as Algorithm 1.

The new incoming values possibly change the histogram boundaries in the window and therefore the underlying empirical probability distribution at each step which gives a runtime of $O(w \cdot n)$ after n steps. We evaluate MID on real-valued data in section 3.

Algorithm 1: Window-wise Computation of Dependencies.

```
1:  procedure MID(data streams S)
2:      for s^{t,w} ∈ S do
3:          ŝ ← Discretize(s^{t,w})
4:          P ← getPDF(ŝ)              ▷ generate PDFs
5:          H ← entropy(P)
6:          CH ← condEntropy(P)        ▷ for all pairs
7:          Î ← norm(H, CH)              of streams
8:          yield Î
9:      end for
10: end procedure
```

3 EXPERIMENTAL EVALUATION

We evaluate MID against two other algorithms for stream correlation monitoring first on three synthetic dataset and second on four real life datasets. The Results for the synthetic data is shown in Figure 2 and for the real datasets are shown in Figures 3 to 6, Tables 1 and 2 show an overview to compare methods with each other.

3.1 Synthetic Data

We created a synthetic datasets with four time series of 6400 datapoints each. Each consists of a non-linear function of the elapsed time t in the first 3200 time steps and gaussian noise in the second half. The functions are chosen similar to the first Friedman data set.(Friedman, 2001):

$$f(t,i) = \begin{cases} t \mod 400, & \text{if } i = 0. \\ sin(t) + sin(t/3+20), & \text{if } i = 1. \\ t + t^2, & \text{if } i = 2. \\ \sqrt{1-t^2}, & \text{if } i = 3. \end{cases} \quad (9)$$

The advantage of the synthetic data is a clear knowledge of the dependency (and predictability) in the data which has to be inferred in other data without prior knowledge. We call the resulting data set *NL*.

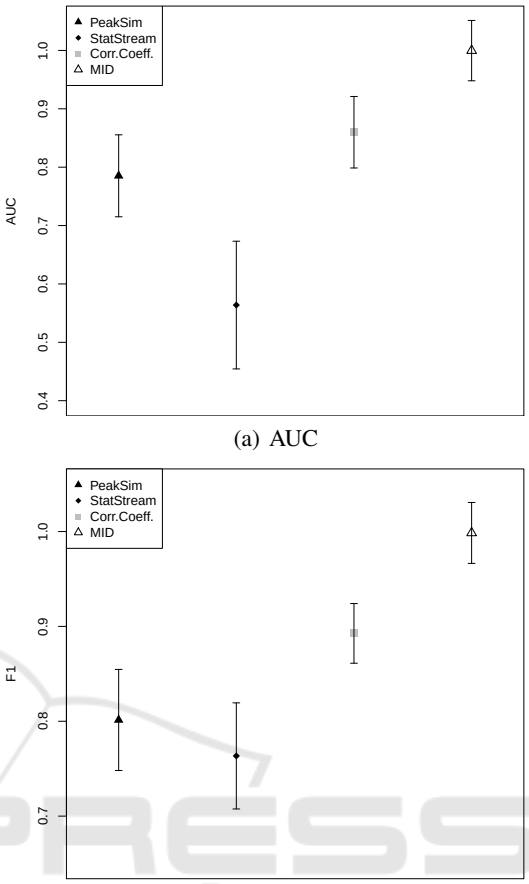
3.2 Stream Datasets

We use four datasets to evaluate our algorithm with different numbers of time steps and dimensions, ranging from 32.000 to 332 million measurements in total. They have been used to emulate the high volume data streams consistently and allow comparison of the methods.

NASDAQ (*NA*) contains daily course information for 100 stock market indices from 2014 and 2015, with 600 indicators (including e.g. open and high course or trading volume) over 320 days in total.(The NASDAQ Stock Market, 2015)

PersonalActivity (*PA*) is a dataset of motion capture where several sensors have been placed on five persons moving around. The sensors record their three-dimensional position. This dataset contains 75 data points each from 5.255 time steps.(Kaluža et al., 2010)

OFFICE (*OL*) is a dataset by the Berkley Research Lab, that collected data about temperature, humidity, light and voltage from sensors placed in a lab office. We use a subset of 32 sensors since there are large gaps in the collection. The subset still contains some gaps that have been filled in with a missing-value indicator. In total this datasets contains 128 measurements over 65.537 time steps.(Bodik et al., 2004)



(a) AUC

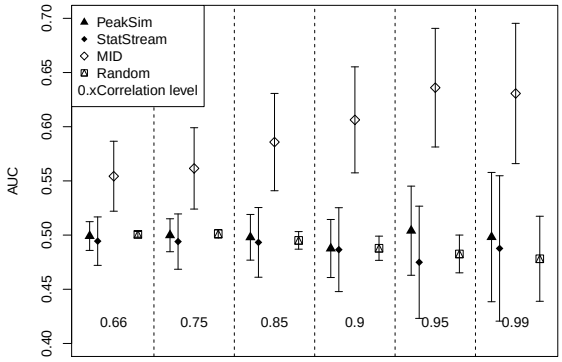
(b) F1-score

Figure 2: (a) Area under ROC curve and (b) F1-value on *NL* dataset.

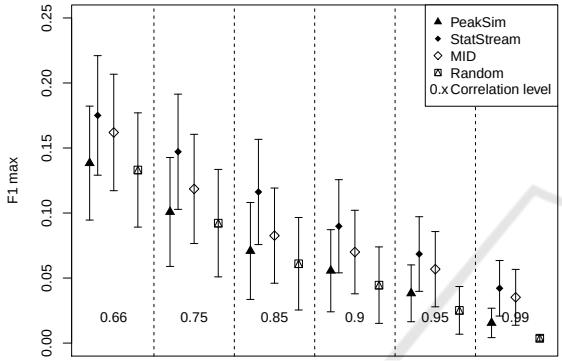
TURBINE (*TU*) contains measurements from 39 sensors in a turbine over half an hour with measurements every quarter of a millisecond. This is the largest of our datasets and contains 39 measurements over about 8.5 million time steps.(Siemens AG, 2015)

3.3 Experimental Settings

Window size w determines the scale of correlation we are interested in and eventually has to be chosen by the user. For the purpose of this evaluation we set it $w = 80$ for the synthetic data and equivalent to 1 second for the turbine dataset, 30 seconds for the other sensor datasets, and to 4 weeks for the stock market dataset. The number of bins b for the discretization needs to be small enough to avoid singletons in the histogram but large enough to map the data distribution – we considered criteria for a sensible choice in section 2.3. As a compromise we chose $b = 20$ for

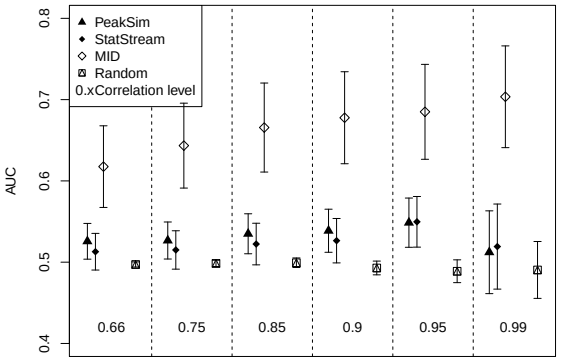


(a) AUC

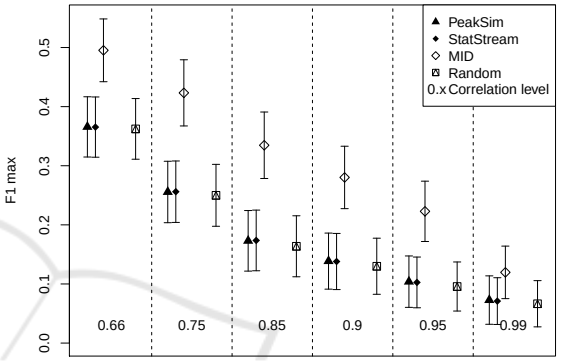


(b) F1-score

Figure 3: (a) Area under ROC curve and (b) maximum F1-value on *TU* dataset. Areas separated by dashed ines show performance at different levels of desired correlation.



(a) AUC



(b) F1-score

Figure 4: (a) Area under ROC curve and (b) maximum F1-value on *OL* dataset. Areas separated by dashed lines show performance at different levels of desired correlation.

the experiments. PeakSim and StatStream use a parameter n that determines the number of DFT-peaks used, and influences runtime and memory in a similar way b influences MID. Consequently we set n equal to b, which is very close to the choice of n in (Zhu and Shasha, 2002) and (Seliniotaki et al., 2014).

We calculate dependency of every dimension with every other, e.g. voltage with temperature. So, for a dataset $n \times d$ i.e. with n steps and d dimensions we calculate $(n-w) \cdot \binom{d}{2}$ dependency scores. Statistical significance is determined with a standard two-sided t-test.

3.4 Evaluation Criteria

For the synthetic dataset we provide the area under ROC curve as classification measure that is independent from the number of true positives in the dataset:

$$AUC = P(X_1 > X_2), \tag{10}$$

where X_1 and $X2$ are the scores for a positive and negative instance respectively.

Also, we report the F1-measure, i.e. the harmonic mean of precision and recall:

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}, \tag{11}$$

$$precision = \frac{TP}{TP+FP}, \tag{12}$$

$$recall = \frac{TP}{TP+FN}. \tag{13}$$

As a positive in the evaluation we label every instance as 1 if it is generated from a function, and as 0 if it is generated by noise (c.f. 3.1). The ground truth to achieve is then simply the mean of ones and zeros in a window.

For the datasets where true dependencies are not known, we chose to evaluate our algorithms at six levels of correlations, from weak to strong correlation, where we deem a windowed pair of streams with correlation coefficient above 0.66, 0.75, 0.85, 0.9, 0.95 and 0.99 respectively as dependent. Accordingly, we classify each window as 0 or 1. For each level, we report for each algorithm AUC and the maximum F1-score, i.e. the highest F1-score along the precision recall curve generated by moving the threshold that separates predicted positives from predicted negatives.

This likely underestimates the number of positives in the data but provides a lower bound for the performance of our algorithm. We see in the synthetic data how Pearsons's correlation coefficient underestimates the dependency of the data streams but performs surprisingly well through linear approximating.

3.5 Results

Figures 2 to 6 show F1-measure ($\pm$ one standard deviation) and AUC ($\pm$ one standard deviation) for the five datasets. For the synthetic dataset *NL* we included the Pearson's correlation coefficient. **Random** has been determined for the non-synthetic datasets by allocating a random value uniformly chosen from $[0,1]$ as dependency measure to each pair of stream windows.

Table 1: Direct overview of all (non-synthetic) datasets: We count significant improvement in AUC (p-value < 0.1 in a two-sided t-test) of row vs. column in 24 experiments. MID scores a total of 48.

AUC improvement	vs.		
	MID	PkSim	SStr
MID	-	24	24
PeakSim	0	-	15
StatStream	0	1	-

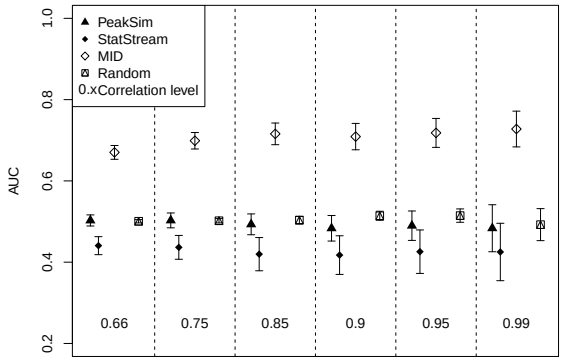
Table 2: Direct overview of all datasets: We count significant improvement in F1 value (p-value < 0.1 in a two-sided t-test) of row vs. column in 24 experiments. MID scores 33 wins.

F1 improvement	vs.		
	MID	PkSim	SStr
MID	-	19	14
PeakSim	0	-	1
StatStream	5	17	-

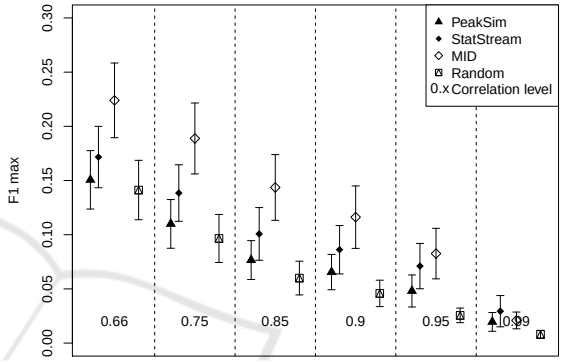
Between PeakSim and StatStream we see little clear difference: PeakSim generally does better than StatStream in AUC but worse if we look at recall and precision. Both perform considerably worse than MID. This holds for both the synthetic and the real datasets.

In our synthetic data, MID shows close to perfect scores, improving significantly ($p < 0.1$ in a two-sided t-test) over the correlation- or DFT-based measures. In the other four datasets it also almost always improves on the other compared methods.

Considering the area under the ROC curve, we see our method in the window-based version clearly outperforming the other correlation measures in all datasets. Altogether MID significantly outperforms



(a) AUC



(b) F1-score

Figure 5: (a) Area under ROC curve and (b) maximum F1-value on *PA* dataset. Areas separated by dashed lines show performance at different levels of desired correlation.

the DFT-based methods in 48 out of 48 direct comparisons.

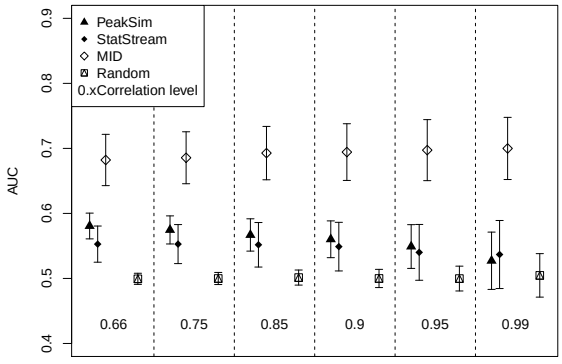
The maximum F1-value shows a similar picture: We see MID outperforming the DFT-based methods in 33 out of 48 cases. Table 2 shows the complete matrix of pairwise comparisons for the F1-value. MID performs well on all data sets, the difference however tends to fall within the margin of error when higher levels of correlation are examined where only few positives are present in the data.

In summary, as proxy for the correlation coefficient, MID works significantly better than DFT-based methods based on AUC and F1-score.

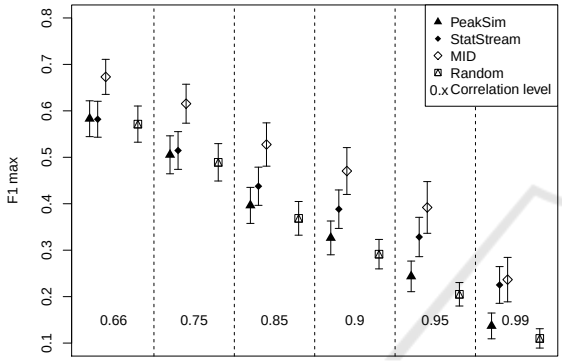
3.6 Execution Time

All experiments have been performed on a PC with an Intel Xeon 1.80GHz CPU and consumer grade hardware, running a Linux with a current 64-bit kernel, and implemented in python 3.4. Figure 7 shows execution times over 5 runs of different correlation measures.

Considering that the number of pairwise depen-



(a) AUC



(b) F1-score

Figure 6: (a) Area under ROC curve and (b) maximum F1-value on *NA* dataset. Areas separated by dashed lines show performance at different levels of desired correlation.

dencies grows quadratic in the number of monitored dimensions, computation speed is an essential factor to deal with high dimensional data. Clearly, the direct calculation of the correlation coefficient is not competitive for large datasets and higher data volume within a window. MID appears about on par with PeakSim and StatStream.

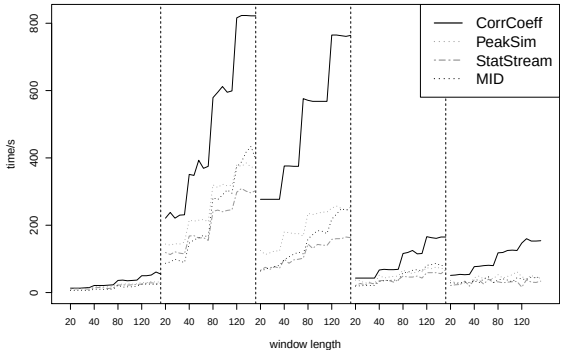


Figure 7: Execution time averaged over 5 runs with increasing window length on the (from left to right) *NL*, *TU*, *OL*, *PA* and *NA* dataset.

4 CONCLUSION

We developed mutual information, a concept from information theory, into a metric that can help to evaluate sensor readings or other streaming data. We describe an incremental algorithm to compute our mutual information based measure with time complexity linear to the length of the data streams. The competitive execution time is achieved with a suitable discretization technique. We evaluated our algorithm on four real life datasets with up to 8.5 million records and against two other algorithms to detect correlations in data streams. It is as more accurate for detecting dependencies in the data than other approximation algorithms.

In future work we want to address the choice of a suitable parameter value for the window length or eliminate the static window altogether. Extending the search for dependencies from pairwise to groups of 3 or more streams increases the computational complexity but brings the potential to extend the analysis to an entropy-based ad-hoc clustering.

Mutual information brings a different perspective to stream analysis that is independent from assumptions on the distribution of or relationship between the data streams.

REFERENCES

Bernhard, H.-P., Darbellay, G., et al. (1999). Performance analysis of the mutual information function for nonlinear and linear signal processing. In *Acoustics, Speech, and Signal Processing, 1999. Proceedings., 1999 IEEE International Conference on*, volume 3, pages 1297–1300. IEEE.

Bodik, P., Hong, W., Guestrin, C., Madden, S., Paskin, M., and Thibaux, R. (2004). Intel lab data. *http://db.csail.mit.edu/labdata/labdata.html*.

Cover, T. M. (1991). Ja thomas elements of information theory.

Darbellay, G. A. (1999). An estimator of the mutual information based on a criterion for conditional independence. *Computational Statistics & Data Analysis*, 32(1):1–17.

Daub, C. O., Steuer, R., Selbig, J., and Kloska, S. (2004). Estimating mutual information using b-spline functions–an improved similarity measure for analysing gene expression data. *BMC bioinformatics*, 5(1):118.

Dionisio, A., Menezes, R., and Mendes, D. A. (2004). Mutual information: a measure of dependency for nonlinear time series. *Physica A: Statistical Mechanics and its Applications*, 344(1):326–329.

Fernandez, D. A., Grau-Carles, P., and Mangas, L. E. (2002). Nonlinearities in the exchange rates returns

and volatility. *Physica A: Statistical Mechanics and its Applications*, 316(1):469–482.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232.

Granger, C. and Lin, J.-L. (1994). Using the mutual information coefficient to identify lags in nonlinear models. *Journal of time series analysis*, 15(4):371–384.

Hall, P. and Morton, S. C. (1993). On the estimation of entropy. *Annals of the Institute of Statistical Mathematics*, 45(1):69–88.

Han, M., Ren, W., and Liu, X. (2015). Joint mutual information-based input variable selection for multivariate time series modeling. *Engineering Applications of Artificial Intelligence*, 37:250–257.

Kaluža, B., Mirchevska, V., Dovgan, E., Luštrek, M., and Gams, M. (2010). An agent-based approach to care in independent living. In *Ambient intelligence*, pages 177–186. Springer.

Kraskov, A., Stögbauer, H., and Grassberger, P. (2008). Estimating mutual information. *Physical review E*, 69(6):066138.

Paninski, L. (2003). Estimation of entropy and mutual information. *Neural computation*, 15(6):1191–1253.

Seliniotaki, A., Tzagkarakis, G., Christofides, V., and Tsakalides, P. (2014). Stream correlation monitoring for uncertainty-aware data processing systems. In *Information, Intelligence, Systems and Applications, IISA 2014, The 5th International Conference on*, pages 342–347. IEEE.

Siemens AG (2015). Gas turbine data. .

Sorjamaa, A., Hao, J., and Lendasse, A. (2005). Mutual information and k-nearest neighbors approximator for time series prediction. *Artificial Neural Networks: Formal Models and Their Applications–ICANN 2005*, pages 752–752.

The NASDAQ Stock Market (2015). Nasdaq daily quotes. *http://www.nasdaq.com/quotes/nasdaq*.

Zhu, Y. and Shasha, D. (2002). Statstream: Statistical monitoring of thousands of data streams in real time. In *Proceedings of the 28th international conference on Very Large Data Bases*, pages 358–369. VLDB Endowment.