Correlation by Compression

Kailash Budhathoki [◦] Jilles Vreeken [◦]

Abstract

Discovering correlated variables is one of the core problems in data analysis. Many measures for correlation have been proposed, yet it is surprisingly ill-defined in general. That is, most, if not all, measures make very strong assumptions on the data distribution or type of dependency they can detect.

In this work, we provide a general theory on correlation, without making any such assumptions. Simply put, we propose correlation by compression. To this end, we propose two correlation measures based on solid information theoretic foundations, i.e. Kolmogorov complexity. The proposed correlation measures possess interesting properties desirable for any sensible correlation measure. However, Kolmogorov complexity is not computable, and hence we propose practical and computable instantiations based on the Minimum Description Length (MDL) principle.

In practice, we can apply the proposed measures on any type of data by instantiating them with any lossless real-world compressors that reward dependencies. Extensive experiments show that the correlation measures works well in practice, have high statistical power, and find meaningful correlations on binary data, while they are easily extendible to other data types.

1 Introduction

Discovering correlation is a key task in knowledge discovery. Intuitively it is clear what correlation is – things that happen or change together – formally, however, it is surprisingly ill-defined. While there exist many measures of correlation, all make assumptions on the type of data and the dependencies they can consider. Pearson correlation, for example, only works on pairs of continuous univariate random variables, and can only detect linear correlations.

A large branch of correlation analysis methods are built on statistical independence. That is, they rely on the joint probability of x and y to deviate from their probability under the independence assumption, i.e. $p(x, y) \neq p(x)p(y)$. While this provides a powerful framework, that in theory can pick up any *statistical* dependency, it does assume that we have access to the true probability distributions.

In practice, we do not know the true distributions, and either fit an assumed distribution, or estimate its density. Both directly affect whether and how well we detect correlation. Moreover, in this setting we ignore the complexity of these distributions themselves and hence run the risk of detecting spurious correlations through overfitting.

In this paper we define correlation from the ground on up, without having to make any such assumptions. We do so using Algorithmic Information Theory. We propose a theoretical framework for correlation analysis based on Kolmogorov complexity. Loosely speaking, we measure how correlated two sets of observations x and y are by how many bits we save compressing the data together, exploiting the dependencies between their corresponding observations, instead of compressing the observations over x and y separately. Our framework is highly general in the sense that it can pick up *any* algorithmic dependency and it naturally extends to multivariate data of *any* kind. That is, we define our score over mathematical objects in general, regardless of whether these are discrete or continuous tabular data, for sequential data, or graphs.

Specifically, we propose two measures, both of which explicitly penalise redundancy to avoid free-rider patterns. That is, if x and y are correlated, but independent of z, $s(x, y)$ should be larger than $s(x, y, z)$. We show how to instantiate these scores in practice through the Minimum Description Length (MDL) principle using existing lossless compressors. Thorough empirical evaluation on both synthetic and real world data confirm these measures have very high statistical power, are robust with regard to redundancy, and identify interesting correlations in real world data.

In short, we formalise, for the first time, the connection between correlation and algorithmic information theory. It is important to note that our proposal is *orthogonal* to existing methods. Our framework can be used both through existing compressors, as well as suggests what compressors to study in the future such that we can measure correlation over non-standard data types such as complex multivariate sequences and graphs. By developing better compressors, we more closely approximate Kolmogorov complexity, and hence will be able to better detect richer classes of correlation.

2 Preliminaries

In this section, we introduce notations and background definitions we will use in subsequent sections.

2.1 Kolmogorov Complexity To develop an algorithmic theory for correlation, we need Kolmogorov complexity [4, 13, 24]. Below we give a brief introduction.

[◦]Max Planck Institute for Informatics and Saarland University.
{kbudhath,jilles}@mpi-inf.mpg.de

We write *string* to mean a finite binary string. The Kolmogorov complexity of a string x, denoted $K(x)$, is the length of the shortest binary program p^* to a Universal Turing machine $\mathcal{U}$ that generates x and *halts*. Let $\ell(.)$ be a function that maps a string to its length, i.e. $\ell : \{0,1\}^* \to \mathbb{N}$. Then, $K(x) = \ell(p^*)$. More formally, the Kolmogorov complexity of a string x is given by

$$K(x) = \min\{\ell(p) \mid p \in \{0,1\}^* \text{ and } \mathcal{U}(p) = x\} ,$$

where $\mathcal{U}(p) = x$ indicates that when the binary program p is run on $\mathcal{U}$, it generates x and halts. In particular, p^* is the most succinct algorithmic description of x. Intuitively, $K(x)$ is the length of the ultimate lossless compression of x.

In particular, we use *prefix-free* Kolmogorov complexity, where the programs are prefix-free—no program is a prefix of another program. In simple terms, this is equivalent to considering the length of the shortest binary program to compute x using a computer programming language such as Python or Haskell, as these programs are prefix-free—there is an end-of-program marker.

The conditional Kolmogorov complexity, denoted $K(x \mid y)$, is the length of the shortest binary program p^* that generates x and halts when string y is provided as an auxiliary input to the program.

Although Kolmogorov complexity is defined over strings, we can interchangeably use it over data objects as any finite data object can be encoded into a string [16]. Notably the Kolmogorov complexity of a data object is invariant to how we encode it into a string. A data object can be a random variable, sequence of events, a temporal graph, etc.

To derive our correlation measure, we need the Kolmogorov complexity of not one, but multiple objects. That is, we need Kolmogorov complexity of a set of data objects. A set of data objects can be a set of random variables, a set of sequences of events, a set of temporal graphs, etc. For a set of data objects X, its Kolmogorov complexity, denoted $K(X)$, is the length of the shortest binary program to the Universal Turing machine $\mathcal{U}$ computes the listing of the elements of X and *halts* [11]. That is, if $X = \{x_1, x_2, \ldots, x_n\}$, then $K(X)$ is given by

$$K(X) = \min\{\ell(p) \mid p \in \{0,1\}^* \text{ and } \mathcal{U}(p) = \langle x_1, \langle x_2, \ldots, \langle x_{n-1}, x_n\rangle \ldots\rangle\rangle\} ,$$

where $\mathcal{U}(p) = \langle x_1, \langle x_2, \ldots, \langle x_{n-1}, x_n\rangle \ldots\rangle\rangle$ indicates that when the binary program p is run on a Universal Turing machine $\mathcal{U}$, it generates the listing of the elements in X, and a way to tell them apart. In turn, the conditional complexity $K(X \mid Y)$ is the length of the shortest binary program to a Universal Turing machine that generates listing of elements of X, and a way to tell them apart given a set of data objects Y as auxiliary information.

Let $K(x, y)$ be the Kolmogorov complexity of the set $\{x, y\}$ when x is generated first, and then y, and a description to tell them apart. Then we can expand $K(x, y)$ as $K(x, y) = K(x) + K(y \mid x) + \mathcal{O}(1)$ [16]. The symmetry of information [16] for two data objects x and y implies $K(x, y) = K(y, x) + \mathcal{O}(1)$. Generalising the symmetry of information to a set X of n data objects, we note that $K(X)$ remains the same up to an additive $\mathcal{O}(1)$ term irrespective of the order in which data objects in X are generated.

From here onwards, we use $\stackrel{+}{=}$ to indicate equality up to an additive constant. For instance, $K(x, y) \stackrel{+}{=} K(x) + K(y \mid x)$. It is a standard practice in algorithmic information theory literature to use additive term $\mathcal{O}(1)$ to mean a constant, considering the length of a fixed binary program, independent of any variable in the expression.

We refer the interested reader to Li & Vitányi [16] for more details on Kolmogorov complexity.

3 Correlation by Algorithmic Information Theory

Suppose we are given a set of data objects X. In practice, X can be a set of random variables, a set of temporal graphs, a set of event sequences, etc. We are interested in inferring whether data objects $x \in X$ are correlated. However, we would like to do so without making any assumptions – neither about distributions, nor about types of dependencies.

3.1 Correlation by Kolmogorov Complexity In broad terms, correlation is a relationship between things that tend to vary or occur together in a way not expected on the basis of chance alone. For example, electricity consumption in a building and electricity bill are correlated because an increase in electricity consumption corresponds to an increase in electricity bill.

In algorithmic information theoretic terms, we capture correlation using the notion of computability. Suppose x be a data object representing a series of n observations $a_1, a_2, \ldots, a_n$, and y represents $b_1, b_2, \ldots, b_n$. In general, when analysing correlation between x and y, we look for *relation* between observations $a_i \in x$ and $b_i \in y$.

In algorithmic information theory, this translates to finding the *most succinct program* that knows the underlying *relation* between corresponding observations a_i and b_i, and uses that to generate observations of x and y. As such, when generating an observation b_i of y, the program is *only allowed* to use information from a_i, but *not* from a_j of x, where $i \neq j$, and vice versa. By using these programs, we only capture correlation.

Let P_x be the shortest program that generates observations of x. Likewise P_y for y. Let P_{xy} be the shortest program that generates observations of both x and y. When generating an observation a_i of x, the program P_{xy} can additionally use the information from corresponding observation b_i of y. Therefore if there is any correlation between x and y, the program P_{xy} can clearly exploit that dependency, whereas P_x and P_y cannot. However, if there is no correla-

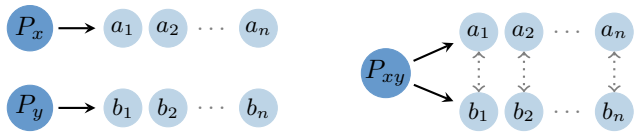


FIGURE 1: **Algorithmic model of correlation.** Let x be a data object representing a series of n observations $a_1, a_2, \ldots, a_n$, and y represents $b_1, b_2, \ldots, b_n$. **(left)** One way to generate the observations of x and y is by independent shortest programs P_x and P_y. **(right)** The alternative way is through a single shortest program P_{xy}. The program P_{xy}, however, unlike P_x and P_y, additionally uses the information from corresponding observation b_i of y when generating an observation a_i of x, and vice versa. Therefore if x and y are correlated, P_{xy} is the *most succinct* way to generate observations of x and y. If x and y are algorithmically independent, both ways of generating the observations of x and y are equivalent.

tion between x and y, i.e. they are algorithmically independent, then there is no difference in using a single program P_{xy} from using two independent programs P_x and P_y. We present this algorithmic model of correlation in Figure 1.

We can think of P_{xy} as a machinery that is the origin of x and y, for it knows how to generate pairs of observations of x and y. This machinery can be a functional dependency that maps each observation of x to an observation of y with or without noise. Alternatively, it could also be a probabilistic process that associates each observation of x with an observation of y with certain probability. Importantly, by using algorithmic information theory, we take into account the complexity of the machinery as well.

This algorithmic model of correlation generalises to a set of n data objects X. The trivial way to generate the observations of data objects in X is by independent shortest programs P_x for every $x \in X$. If there is any correlation between data objects in X, the most succinct way to generate the observations of data objects in X is by the single shortest program P_X that can additionally exploit the dependencies between corresponding observations of data objects in X. More formally, $K(X) \leq \sum_{x \in X} K(x)$, where the inequality holds with equality when data objects in X are algorithmically independent of each other.

Our goal is to identify correlation among data objects in X. Therefore, we need to measure the amount of correlation present in data objects in X. The amount of correlation present in X can be simply defined as the absolute difference between $\sum_{x \in X} K(x)$ and $K(X)$. That is, how much do we save in the description of X if we describe X by a single program as opposed to describing each data object $x \in X$ by individual programs. More formally, the amount of algorithmic correlation present between data objects in X, denoted $\delta(X)$, is given by

$$\delta(X) = \left(\sum_{x \in X} K(x) \right) - K(X) \, .$$

We refer to $\delta(X)$ as the *total algorithmic correlation* of X here onwards.

However, $\delta(X)$, being an absolute quantity, is hard to compare across various sets of data objects, and hence different sets of random variables, sequence of events, temporal graphs, etc. For instance, if two strings of length 1 Million bits possess total algorithmic correlation of 100 bits, we can easily misinterpret that they are more correlated than two strings of length 1000 bits that have the same value. Therefore, to obtain comparable values, we have to normalise. Next we define σ, the normalised version of the total algorithmic correlation.

DEFINITION 1. *The normalised total algorithmic correlation of a set of data objects X is defined as*

$$\sigma(X) = \frac{\sum_{x \in X} K(x) - K(X)}{\sum_{x \in X} K(x) - \min_{x \in X} K(x)} \, .$$

The normalised total algorithmic correlation score has a number of desirable properties.

LEMMA 3.1. *For a set of data objects X, up to $\mathcal{O}(1)$ term*

- $0 \leq \sigma(X) \leq 1$,
- $\sigma(X) = 0$ *iff* X *is algorithmically independent,*
- $\sigma(X) = 1$ *iff* X *is algorithmically fully correlated.*

Proof. We postpone the proof to the Appendix.

To avoid redundancy, a correlation measure should punish redundancy. That is, a set of data objects containing a correlated subset and other data objects independent of that subset should not get an as high, or higher score than when we only consider the correlated subset. The normalised total algorithmic correlation punishes such redundant sets.

LEMMA 3.2. *Let X be a set of algorithmically correlated data objects. Let z be a data object which is algorithmically independent of X. Let $X' = X \cup z$. Then $\sigma(X') < \sigma(X)$.*

Proof. We postpone the proof to the Appendix.

The normalised total algorithmic correlation possesses properties desired for any decent correlation score. First, it has clear bounds – a lower bound of 0 and an upper bound of 1. A value of 0 implies the data objects are algorithmically independent. A value of 1 implies the data objects are fully correlated. The closer the score to 1, the stronger

the correlation and the closer the score to 1, the weaker the correlation. Second, it punishes redundant sets.

Alternative to total algorithmic correlation, where we measure the difference in bits of the independent description versus the joint description, we can also naturally measure algorithmic correlation by considering the contribution of other data objects in the explanation of each data object. We refer to the amount of algorithmic correlation thus derived as *total singular algorithmic correlation*. More formally, the total singular algorithmic correlation of X, denoted $\Delta(X)$, is given by

$$\Delta(X) = \sum_{x \in X} K(x) - K(x \mid X \setminus x) \, .$$

The total singular algorithmic correlation overestimates the total algorithmic correlation. For that we present the following lemma.

LEMMA 3.3. *For a set of data objects X, $\Delta(X) \geq \delta(X)$.*

Proof. We postpone the proof to the online Appendix.

As with the total algorithmic correlation, to compare the total singular algorithmic correlation between different sets of data objects, we need to normalise. Next we define ϕ, the normalised version of the total singular algorithmic correlation.

DEFINITION 2. *The normalised total singular algorithmic correlation of a set of data objects X is defined as*

$$\phi(X) = \frac{\sum_{x \in X} K(x) - K(x \mid X \setminus x)}{\sum_{x \in X} K(x)} \, .$$

Although a relaxed formulation, total singular algorithmic correlation shares the most important properties of total algorithmic correlation.

LEMMA 3.4. *For a set of data objects X, up to $\mathcal{O}(1)$ term*

- $0 \leq \phi(X) \leq 1$,
- $\phi(X) = 0$ *iff* X *is algorithmically independent, and*
- $\phi(X) = 1$ *iff* X *is algorithmically fully correlated.*

Proof. We postpone the proof to the Appendix.

LEMMA 3.5. *Let X be a set of algorithmically correlated data objects. Let z be a data object which is algorithmically independent of X. Let $X' = X \cup z$. Then $\phi(X') < \phi(X)$.*

Proof. We postpone the proof to the Appendix.

The normalised total singular algorithmic correlation, like the normalised total algorithmic correlation, possesses properties desired for any decent correlation score. First, it has clear bounds – a lower bound of 0 and an upper bound of 1. Second, it punishes redundancy.

Correlation using Kolmogorov complexity has a number of powerful properties. First, we do not need to assume the distribution of data as we only need to consider the data objects. Second, the correlation score is generic in the sense that we are not restricted to one type of data. Third, we do not need to assume any specific kind of correlation between data objects in X, nor do we need to assume anything about the shape or type of noise.

Connection to Statistical Dependence In this paper we define correlation on the foundations of algorithmic information theory. The other, and perhaps more well-known branch of information theory, Shannon information theory, can also be used to measure correlation. It does so, however, in terms of *statistical* dependence. Next we study the connection between algorithmic correlation and statistical dependence.

Shannon entropy is the average amount of information that the observer has gained after receiving a realised outcome z of a discrete random variable Z [6]. Let p be the probability mass function of Z. The Shannon entropy of Z, denoted $H(Z)$, is given by $H(Z) = -\sum_z p(Z = z) \log p(Z = z)$. Therefore to transmit n events from Z, we need $H(Z)$ bits on average per event.

Note that Shannon entropy gives us the number of bits to transmit the data under a *given* distribution. It does *not* take the complexity of the distribution into account. Kolmogorov complexity, on the other hand, considers *only* the data irrespective of the source which generated it—and hence, implicitly takes the complexity of the 'distribution' into account. Although different, there exists a connection between Shannon entropy and Kolmogorov complexity.

This connection can be roughly described as follows [11]. For probability distributions with very low complexity,[1] the expected Kolmogorov complexity is *equal* to Shannon entropy. For distributions with high complexity, however, the two quantities might differ by a wide margin. Algorithmic correlation hence implies statistical dependence for distributions with low complexity, whereas for distributions with high complexity, the connection is not apparent.

Although Kolmogorov complexity has sound theoretical foundations, due to the widely known *halting problem*, it is not computable. Shannon entropy, on the other hand, is computable *iff* the distribution is known. However, in practice, distributions are hard to estimate, and finite sample sizes makes things even worse—in practice we therefore

[1] Assuming that the data object is part of a sample space of a distribution. We refer the interested reader to Grünwald & Vitányi [11] for more details.

only work with estimates of Shannon entropy.

Kolmogorov complexity, on the other hand, can be approximated from above through lossless compression [16]. In particular, the Minimum Description Length (MDL) principle provides a statistically sound and computable means for approximating Kolmogorov complexity [10, 21]. Next we discuss how MDL can be used for computing the scores.

3.2 Correlation by MDL The Minimum Description Length (MDL) principle [20] is a practical version of the Kolmogorov complexity. It circumvents the computability issue of the Kolmogorov complexity by restricting the programs to those that always halt, i.e. through *lossless* compression. Moreover, we can select these programs using our prior knowledge of the problem domain.

The MDL principle has its root in the two-part decomposition of Kolmogorov complexity [16]. Like Kolmogorov complexity, MDL embraces the slogan *Induction by Compression*. It can be described roughly as follows. Given a set of models $\mathcal{M}$, and data D, the best model $M \in \mathcal{M}$ is the one that minimises $L(D, M) = L(M) + L(D \mid M)$, where $L(M)$ is the length, in bits, of the description of M, and $L(D \mid M)$ is the length, in bits, of the description of the data when encoded with the model M.

For our goal we need a model class $\mathcal{M}$, consisting of models that capture the correlation between data objects. As such, these models should capture the dependency between corresponding observations of data objects when described together. For instance, let $X = \{x_1, x_2\}$, where x_1 represents a series of observations $a_1, a_2, \ldots a_k$, and x_2 represents $b_1, b_2, \ldots, b_k$. Our models in model class $\mathcal{M}$ should describe X by describing a_i given b_i, or vice versa.

Suppose we have chosen our model class $\mathcal{M}$. Then we can approximate $K(X)$ using MDL by $L(X, M_X)$, which is defined as $L(X, M_X) = L(M_X) + L(X \mid M_X)$, where $L(M_X)$ is the length, in bits, of the description of the MDL optimal model for X, and $L(X \mid M_X)$ is the length, in bits, of the description of X when encoded with M_X. We approximate $K(x)$ by $L(x, M_x)$ analogously.

Then the normalised total algorithmic correlation of X using MDL is given by

$$\widehat{\sigma}(X) = \frac{\sum_{x \in X} L(x, M_x) - L(X, M_X)}{\sum_{x \in X} L(x, M_x) - \min_{x \in X} L(x, M_x)} \ .$$

We refer to this MDL based approximation of the normalised total algorithmic correlation as *normalised total compressive correlation*, shortly NTC.

To approximate the normalised total singular correlation, we need to approximate conditional complexities $K(x \mid X \setminus x)$. Let $\bar{x} = X \setminus x$. We have $K(X) \stackrel{+}{=} K(\bar{x}) + K(x \mid \bar{x})$, and it is easy to see that $K(x \mid \bar{x}) \stackrel{+}{=} K(X) - K(\bar{x})$. Then the normalised total singular algorithmic correlation of X using MDL is given by

$$\widehat{\phi}(X) = \frac{\sum_{x \in X} L(x, M_x) - L(X, M_X) - L(\bar{x}, M_{\bar{x}})}{\sum_{x \in X} L(x, M_x)} \ .$$

We refer to this MDL based approximation of the normalised total singular algorithmic correlation as *normalised total singular compressive correlation*, shortly NSC. For both scores, the bounds still hold up to an additive constant term, since we are just restricting our model class with MDL.

The MDL based formulation has interesting properties. First, it is computable, unlike the formulation based on Kolmogorov complexity. Second, it is agnostic to the type of data in its general form. Third, it does not require assumptions on neither the distribution, nor the type of dependency in data. Next we discuss how we to compute the scores practically.

4 Correlation by Compression

To use MDL, and hence to compute the scores in practice, we have to define our model class. Clearly it is desirable to have a model class that is as general as possible. Further, the better the overall description – of both model, and data given the model – the better we approximate Kolmogorov complexity. Importantly, we observe that to apply the MDL-based scores under the selected model class, the following should hold up to an additive constant term $L(X, M_X) \leq \sum_{x \in X} L(x, M_x)$.

Whereas theoretically possible, in practice there does not exist a single lossless compressor that works on all types of data. This means that for our purpose we have to instantiate our scores with different lossless compressors. Next we provide a brief overview of existing relevant MDL-based lossless compressors.

Binary Data First we present existing MDL-based compressors for binary data. SLIM [23] considers binary data as transactions, and compresses these using a set of itemsets. That is, it detects relevant *co-occurrences* of items, and discovers that set of patterns by which it can compress the data best. As an example, consider a binary dataset that consists of transactions from a supermarket. If people buy bread and butter very often, i.e. these products co-occur frequently in the database. SLIM recognises this dependency, and adds $\{$*bread, butter*$\}$ as a pattern to its model if adding the pattern to the model reduces the overall description length. As a result, SLIM can identify correlation between items where occurrence of one item implies the occurrence of other(s), but cannot detect negative correlations or dependencies.

PACK [27] is another MDL-based compressor for binary data. Its model contains binary decision trees for each attributes that are dependent on other attributes. As a result, it can identify complex interactions between attributes. That is, it considers not just the 1s in the data, but also the 0s. As an example, let us consider the binary dataset containing transactions from a supermarket again. Suppose 90% of the

time when people buy beer, they do not buy milk. PACK recognises this type of dependency, and can hence be used to measure positive as well as negative correlations.

Other Data There exist MDL-based compressors for other types of data as well. For instance, SQS [28] considers univariate sequential data. Correlation intuitively only makes sense in multivariate data, and hence SQS is not applicable here. DITTO, a recent proposal by Bertens et al. [2] extends SQS to multivariate event sequences. As such, in principle it is applicable within our framework. For static respectively temporal graphs, relevant compressors include VOG [14] and TIMECRUNCH [22]. Both, however, consider only single graphs, and hence cannot pick correlation from multiple temporal graphs. It will make for interesting future work to extend these algorithms to multi-graphs—by which our framework will then allow to also measure correlation between graphs.

5 Related Work

Identifying correlated variables is one of the core tasks in data analysis. It comes as no surprise that numerous correlation measures have been proposed over the years. However, most of them make assumption about either the distribution or the underlying dependency or both. We define correlation using algorithmic information theory without making any such assumptions. Clearly it is impossible to give an exhaustive overview of literature on correlation. Therefore we only cover those that are widely in use.

For over a century, the Pearson correlation coefficient has been the workhorse for understanding the bivariate relationships in statistical practice. However, it only detects linear dependency, and works on numeric data. There exist variations of the Pearson correlation coefficient like phi coefficient [7] for binary variables, and Spearman rank correlation coefficient [25] for measuring the statistical dependence between the ranking of two variables. Distance Correlation (dCor) [26] generalises and extends the Pearson correlation coefficient to multiple variables. It can detect non-linear correlations as well. However, it introduces the notion of covariance with respect to a stochastic process. Like the Pearson correlation coefficient, dCor works only on numeric data.

There are information theoretic measures like the mutual information, and its generalisation to more than two random variables such as the total correlation [29]. However, they require assumption on the distribution. The Maximal Information Coefficient (MIC) [19] is a continuous variable counterpart to mutual information for two discrete random variables. It finds the discretisation of variables that maximise their normalised mutual information. MAC [18] generalises MIC to more than two variables by identifying discretisations of all variables that maximise their normalised total correlation. MIC and MAC only work on numeric data.

There are non-linear measures of statistical dependence that define correlation in terms of projections of original data. Examples include Kernel Canonical Correlation Analysis (KCCA) [1], Randomized Dependence Coefficient (RDC) [17], Maximum Mean Discrepancy (MMD) [3].

Statistical tests like *t*-test, ANOVA, and *chi-square*, are widely used in data analysis. However, they assume that the test-statistic follows a certain distribution under the null hypothesis. There are also unconventional techniques for discovering correlated attributes. One such algorithm is the Fractal Dimension Attribute Significance Estimator (FDASE) [9] that uses *fractal* dimension for identifying correlated attributes. It can identify both linear and non-linear correlations. However it only works on numerical attributes, and requires a self-similarity property for a database.

The use of Algorithmic Information Theory (AIT) in data analysis is not new. Several measures have been proposed using AIT to capture the similarity between objects. Among them two widely known measures are Normalized Information Distance (NID) [15] and the Compression Dissimilarity Measure (CDM) [12]. Whereas CDM works only with two objects, there is a generalisation of NID for multiple data objects [5]. However, here we are not interested in similarity, but in correlation. We describe a set of data objects by exploiting the dependency between corresponding observations of the data objects in the set, as opposed to describing data objects considering dependencies between all observations of the data objects in the set.

We define correlation from scratch *without* making any assumptions using algorithmic information theory. We also propose two scores to measure the strength of correlation among a *set* of data objects. The scores can be applied to *any* type of data by instantiating them with any lossless real-world compressors that reward dependencies.

6 Experiments

Next we empirically evaluate our framework on synthetic and real-world data. In particular, we consider binary data, as we identified there exist two powerful compressors for multivariate binary data, i.e. SLIM [23] and PACK [27].

We implemented the framework in Python, and provide the source code for research purposes, along with the used datasets and synthetic dataset generator.[2] All experiments were executed single threaded on MacBook Pro with 2.5 GHz Intel Core i7 processor and 16 GB memory.

Further we note that comparison against other methods is also non-trivial. Here we aim to define correlation in clear terms without making any assumptions, and give a general framework for measuring correlation. Hence we present the experiments as a proof of concept for the proposal.

[2] http://eda.mmci.uni-saarland.de/cbc/

6.1 Synthetic Data To evaluate the scores on data with the known ground truth, we use synthetic data. We generate synthetic data on a per row basis, where presence of the first attribute is based on a fair coin toss. For independent datasets, the other attributes are similarly generated. For correlated datasets, presence of an attribute is dependent on presence of previous attribute with a certain probability, termed here onwards as *dependency*.

Statistical Power First we assess whether the scores infer correlation when correlation really exists. For that we test their statistical power. The null hypothesis is that the data dimensions are statistically independent. To determine the cut-off for testing the null hypothesis, we first generate 200 datasets with *no* correlation. Then we compute their correlation scores, and set the cut-off value at a significance level of 0.05. Next, we generate new 200 datasets with correlation. The statistical power is the proportion of the 200 new datasets whose correlation score is below the cut-off.

To this end, we generate binary datasets with 5 dimensions, and 5 000 rows. In Figure 2(a) we plot statistical powers of both scores using SLIM at various dependencies. For PACK, we give the plot in Figure 2(b). We observe that both scores achieve perfect powers at all level of dependencies using both compressors. From here onwards, unless stated otherwise, we use PACK as our running compressor as its model class already encompasses the model class of SLIM.

Next we study statistical powers of the scores on datasets with various dimensions. We fix the *dependency* to 0.5. In Figure 2(c), we show powers of the scores at various dimensions. In all cases, both scores show perfect power.

In the next experiment, we examine statistical powers of the scores in presence of noisy dimensions. To this end, first we generate a dataset X having 5 dimensions with a dependency of 0.5. Then we add to it extra dimensions that are independent of each other as well as the original data. In Figure 2(d), we give the plot showing powers of the scores at varying number of extra dimensions. We observe that both scores achieve perfect statistical powers in all cases.

Robustness against Redundancy Next we examine how well the scores punish redundancies in practice. To this end, first we generate a binary dataset X with random dependency over the range 40% to 100%, and the number of dimensions chosen uniformly at random over the range 2 to 6. Then we create a new dataset X' by adding extra dimensions to X that are independent of each other as well as the original data. The number of extra dimensions are chosen randomly over the range 1 to 6. We uniformly sample 500 such pairs of X and X' using this technique.

The results are presented in the Appendix. In Figure 3(a), we show the scatter plot of NTC scores for X and X'. Likewise we give the scatter plot for NSC scores in Figure 3(b). In both plots, we observe that both scores significantly punish redundancies. Further we also give scatter plot for the difference in NTC and NSC scores for X and X' in Figure 3(c). We observe that the score differences of NTC and NSC are close.

Overall, we see that both scores punish redundancies. Further they are highly stable, and effectively discern correlated data from independent binary data.

6.2 Real-world Data After examining the performance of the scores in synthetic data, next we present the results from real-world data.

Quantitative Evaluation First we consider six datasets taken from Frequent Itemset Mining Implementations (FIMI) repository[3] and UCI Machine Learning repository.[4] We employ level-wise search with pruning at a threshold of 0.85. We report the results in Table 2 (see Appendix). For each dataset, we give the number of rows, the number of columns. Further we show the number of correlated patterns discovered using NTC and NSC up to a maximum level of 3 and 4 respectively.

We discover significantly fewer patterns out of all possible pattern combinations. For instance, from the adult dataset, we only discover 6 297 correlated patterns out of $\sum_{i=2}^{4} \binom{97}{4} = 3\,616\,936$ possible patterns using NSC. As for NTC, we only discover 3. Overall we observe that NTC discovers relatively fewer correlated patterns than NTC. This is also in line with the fact that the total singular algorithmic correlation overestimates total algorithmic correlation. These results show the scores can be plugged in to a search algorithm to find correlated patterns from real-world data.

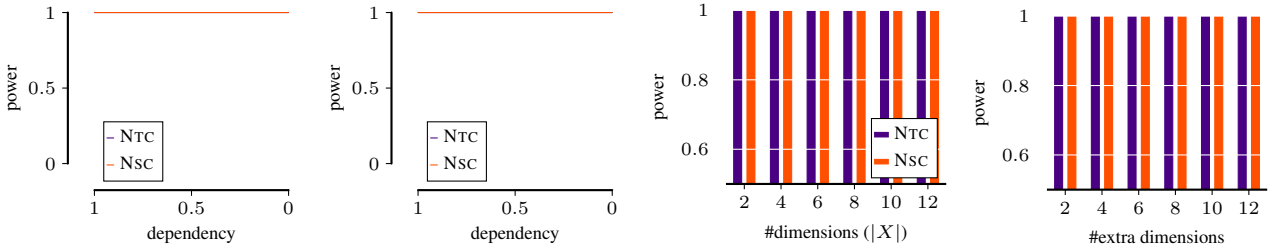
Qualitative Evaluation Next we examine whether the results are meaningful. For that, we inspect correlated patterns discovered from various real-world datasets. In particular, for conciseness, we report results using NTC, unless stated otherwise.

Lotto Dataset The *Lotto* dataset is taken from the website of the Belgian National Lottery,[5] and contains the results of all lottery draws between May 1983 and May 2011. Each draw consists of seven numbers (six plus one bonus ball) out of a total of 42.

We apply level-wise search on the *Lotto* dataset. We do not discover any correlated patterns up to an extremely low threshold score of 0.004 using both scores. This also confirms to our prior belief that lottery numbers are random.

ICDM Abstracts Dataset The *ICDM Abstracts* dataset is available from the authors of [8]. It consists of abstracts – stemmed and stop-words removed – of 859 papers published at the ICDM conference until the year 2007. Each abstract is represented by a row and words are the attributes. As such, there are 3 933 attributes.

[3] http://fimi.ua.ac.be/data/
[4] http://archive.ics.uci.edu/ml/datasets.html
[5] http://www.nationale-loterij.be/



(a) dependency vs power $|X| = 5$, SLIM (b) dependency vs power $|X| = 5$, PACK (c) #dimensions vs power dependency = 0.5, PACK (d) #extra dimensions vs power $|X| = 5$, PACK

FIGURE 2: For synthetic binary datasets, we report for both scores (a), (b) statistical powers at various dependencies using SLIM and PACK respectively, (c) powers at various dimensionalities, and (d) powers at various extra dimensionalities.

Patterns	**NTC**
nearest, neighbour	0.66
naive, bayes	0.54
state [of the], art, nearest, neighbour	0.35
naive, bayes, methodology	0.30

TABLE 1: Exemplars from patterns discovered by level-wise search on the *ICDM Abstracts* data at a threshold of 0.3.

We only consider attributes with a minimum support of 20, and apply level-wise search at a threshold of 0.3. Accordingly we discover 609 patterns. From those, we give few exemplars along with their scores in Table 1. We see some obvious correlations: *nearest* appears often with *neighbour*. Likewise *naive* with *bayes*. We also discover correlations in higher dimensions. For instance, generic terms like *state [of the] art* appear often in the abstract, this time it happens to be mostly seen with *nearest neighbour*.

Mammals Dataset In our last case study, we consider the *Mammals* dataset provided by the European Mammal Society.[6] It consists of presence records of 124 European mammals within 2 183 areas of 50-by-50 kilometers.

Using level-wise search at a threshold of 0.48, we discover 3 correlated patterns: {*norway lemming, european pine vole*}, {*norway lemming, european pine vole, balkan mole*}, and {*moose, fallow deer*} in descending order of NTC scores. Take for instance the top most correlated pattern. It turns out that *norway lemming* is a common species of lemming found commonly in northern European countries. And *european pine vole* is also native to Europe, hence also found in northern European countries. In short, both of them are present in northern European countries.

[6] http://www.european-mammals.org/

7 Discussion

The experiments show that both scores work well in practice. We identify true correlation regardless of the dimensionality with high statistical power, even at low level of dependencies. Moreover both scores punish redundancies. In real-world data, we discover correlated patterns. Further the qualitative case studies show that the results are meaningful.

In this work, we give a clear definition of correlation without making any assumptions. We can apply the proposed scores by instantiating them with *any* lossless real-world compressor that rewards dependencies. As such, the more powerful the compressor, the better the solution. However, we still have the choice of a specific compressor if we are only interested in certain type of dependency. By no means we claim that we have solved the problem of correlation completely. However, we believe this work demonstrates the potential of the framework, and – hopefully – convinces researchers to frame their research within it.

We ran our experiments on binary data. However, the generality of our framework covers *any* type of data. We studied correlation in sequential data as well using DITTO [2], an MDL-based compressor for multivariate sequential data. We observed DITTO not only uses multivariate sequential patterns, but also univariate sequential patterns for summarising multivariate sequential data. As a result, the scores are not bounded in the interval $[0, 1]$ any more. One avenue for further research would be to build a compressor for multivariate sequential data that rewards dependencies, and plug in our framework for correlation analysis.

To mine correlated patterns from data, we employed level-wise search with pruning. Notably we found significantly fewer patterns out of a large number of possible patterns from data. However, the proposed scores do not exhibit monotonicity property. To show this, consider a set of three binary random variables $\{a, b, c\}$, where a and b are independent, but c is an XOR of a and b. Whereas $\sigma(\{a, b\}) = 0$, $\sigma(\{a, b, c\}) = 1$ (same for NSC). It would be interesting to

research whether the scores exhibit certain properties desirable for pruning search space under a restricted model class. Moreover it would also be interesting to see the performance of the scores under different search schemes.

In the experiments, we used existing MDL-based lossless compressors for binary data – SLIM and PACK. However, the framework works on any type of data in its generality by instantiating it with *any* lossless compressor that rewards dependencies. One direction for future work is to build a powerful real-world compressor that can pick various types of dependencies from any type of data. Alternatively we can also only focus on one type of data, and build a compressor that picks correlation from it. In the future, we would like to apply our framework on other types of data as well – continuous real-valued, sequential, temporal graph, etc.

8 Conclusion

We gave a general theory on correlation without making any assumptions using algorithmic information theory. In particular, we proposed two correlation measures using Kolmogorov complexity. The correlation measures possess interesting properties desirable from any sensible correlation measure. To circumvent the computability issue of Kolmogorov complexity, we proposed practical instantiations based on the Minimum Description Length principle.

In practice, the proposed measures can be applied on any type of data by instantiating them with any lossless real-world compressors that reward dependencies. Extensive evaluations showed that the correlation measures works well in practice, have high statistical power, and find meaningful correlations on binary data, while they are easily extendible to other data types

Acknowledgements

Kailash Budhathoki is supported by the International Max Planck Research School for Computer Science (IMPRS-CS). Both authors are supported by the Cluster of Excellence "Multimodal Computing and Interaction" within the Excellence Initiative of the German Federal Government.

References

[1] F. R. Bach and M. I. Jordan. Kernel independent component analysis. *J. Mach. Learn. Res.*, 3:1–48, 2003.

[2] R. Bertens, J. Vreeken, and A. Siebes. Keeping it short and simple: Summarising complex event sequences with multivariate patterns. In *KDD*, pages 735–744, 2016.

[3] K. Borgwardt, A. Gretton, M. Rasch, H.-P. Kriegel, B. Schölkopf, and A. Smola. Integrating structured biological data by kernel maximum mean discrepancy. In *ISMB*, pages 49–57, 2006.

[4] G. J. Chaitin. On the simplicity and speed of programs for computing infinite sets of natural numbers. *J. ACM*, 16(3):407–422, 1969.

[5] A. R. Cohen and P. M. B. Vitányi. Normalized compression distance of multisets with applications. *IEEE TPAMI*, 37(8):1602–1614, 2015.

[6] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley-Interscience New York, 2006.

[7] H. Cramér. *Mathematical Methods of Statistics*. Princeton University Press, 1946.

[8] T. De Bie. Maximum entropy models and subjective interestingness: an application to tiles in binary databases. *Data Min. Knowl. Disc.*, 23(3):407–446, 2011.

[9] E. P. M. de Sousa, C. Traina Jr., A. J. M. Traina, L. Wu, and C. Faloutsos. A fast and effective method to find correlations among attributes in databases. *Data Min. Knowl. Disc.*, 14(3):367–407, 2007.

[10] P. Grünwald. *The Minimum Description Length Principle*. MIT Press, 2007.

[11] P. Grünwald and P. M. B. Vitányi. Shannon information and kolmogorov complexity. *CoRR*, cs.IT/0410002, 2004.

[12] E. Keogh, S. Lonardi, and C. A. Ratanamahatana. Towards parameter-free data mining. In *KDD*, pages 206–215, 2004.

[13] A. N. Kolmogorov. Three approaches to the quantitative definition of information. *Problems of Information Transmission*, 1:1–7, 1965.

[14] D. Koutra, U. Kang, J. Vreeken, and C. Faloutsos. VOG: Summarizing graphs using rich vocabularies. In *SDM*, pages 91–99. SIAM, 2014.

[15] M. Li, X. Chen, X. Li, B. Ma, and P. Vitányi. The similarity metric. *IEEE TIT*, 50(12):3250–3264, 2004.

[16] M. Li and P. Vitányi. *An Introduction to Kolmogorov Complexity and its Applications*. Springer, 1993.

[17] D. Lopez-Paz, P. Hennig, and B. Schölkopf. The randomized dependence coefficient. In *Advances in Neural Information Processing Systems 26*, pages 1–9, 2013.

[18] H.-V. Nguyen, E. Müller, J. Vreeken, and K. Böhm. Multivariate maximal correlation analysis. In *ICML*, pages 775–783. JMLR, 2014.

[19] D. N. Reshef, Y. A. Reshef, H. K. Finucane, S. R. Grossman, G. McVean, P. J. Turnbaugh, E. S. Lander, M. Mitzenmacher, and P. C. Sabeti. Detecting novel associations in large data sets. *Science*, 334(6062):1518–1524, 2011.

[20] J. Rissanen. Modeling by shortest data description. *Automatica*, 14(1):465–471, 1978.

[21] J. Rissanen. A universal prior for integers and estimation by minimum description length. *Annals Stat.*, 11(2):416–431, 1983.

[22] N. Shah, D. Koutra, T. Zou, B. Gallagher, and C. Faloutsos. Timecrunch: Interpretable dynamic graph summarization. In *KDD*, 2015.

[23] K. Smets and J. Vreeken. SLIM: Directly mining descriptive patterns. In *SDM*, pages 236–247. SIAM, 2012.

[24] R. J. Solomonoff. A formal theory of inductive inference. part I, II. *Information and Control*, 7:1–22224–254, 1964.

[25] C. Spearman. The proof and measurement of association between two things. *American Journal of Psychology*, 15:88–103, 1904.

[26] G. J. Székely and M. L. Rizzo. Brownian distance covariance. *The Annals of Applied Statistics*, 3(4):1236–1265, 2009.

[27] N. Tatti and J. Vreeken. Finding good itemsets by packing

data. In *ICDM*, pages 588–597, 2008.

[28] N. Tatti and J. Vreeken. The long and the short of it: Summarizing event sequences with serial episodes. In *KDD*, pages 462–470. ACM, 2012.

[29] S. Watanabe. Information theoretical analysis of multivariate correlation. *IBM J. Res. Dev.*, 4(1):66–82, 1960.

A Proof of Lemma 3.1

LEMMA 3.1. *For a set of data objects X, up to $\mathcal{O}(1)$ term*

- $0 \leq \sigma(X) \leq 1$,
- $\sigma(X) = 0$ *iff* X *is algorithmically independent,*
- $\sigma(X) = 1$ *iff* X *is algorithmically fully correlated.*

Proof. We have $K(X) \leq \sum_{x \in X} K(x)$. The equality holds up to $\mathcal{O}(1)$ term *iff* data objects in X are algorithmically independent of each other. Trivially $\sum_{x \in X} K(x) - \min_{x \in X} K(x) > 0$ for $|X| > 1$. Therefore up to $\mathcal{O}(1)$ term $\sigma(X) \geq 0$, where $\sigma(X) \stackrel{+}{=} 0$ *iff* data objects in X are algorithmically independent.

The value of $\delta(X)$ is maximum when $K(X)$ is minimum. We can lower bound $K(X)$ by $\min_{x \in X} K(x)$. That is, the Kolmogorov complexity of a set of data objects is at least the Kolmogorov complexity of the least complex data object in the set. In particular, when all the data objects in X are algorithmically fully correlated, we can use one data object to explain all the others. In that particular case, $K(X) \stackrel{+}{=} \min_{x \in X} K(x)$. Using the lower bound, we have $K(X) \geq \min_{x \in X} K(x)$. Therefore we have $\sum_{x \in X} K(x) - K(X) \leq \sum_{x \in X} K(x) - \min_{x \in X} K(x)$. Following this, $\sigma(X) \leq 1$ up to $\mathcal{O}(1)$ term. Importantly, $\sigma(X) \stackrel{+}{=} 1$ *iff* the data objects in X are algorithmically fully correlated.

Therefore, up to $\mathcal{O}(1)$ term we have $0 \leq \sigma(X) \leq 1$. □

B Proof of Lemma 3.2

LEMMA 3.2. *Let X be a set of algorithmically correlated data objects. Let z be a data object which is algorithmically independent of X. Let $X' = X \cup z$. Then $\sigma(X') < \sigma(X)$.*

Proof. We can decompose $K(X')$ using chain rule as $K(X') \stackrel{+}{=} K(X) + K(z \mid X)$. But since z is algorithmically independent of X, we have $K(z \mid X) = K(z)$. Therefore $K(X') \stackrel{+}{=} K(X) + K(z)$. Then $\sigma(X')$ is given by

$$\begin{aligned}
\sigma(X') &\stackrel{+}{=} \frac{\sum_{x \in X'} K(x) - K(X')}{\sum_{x \in X'} K(x) - \min_{j \in X'} K(j)} \\
&\stackrel{+}{=} \frac{\sum_{x \in X} K(x) + K(z) - K(X) - K(z \mid X)}{\sum_{x \in X} K(x) + K(z) - \min_{j \in X'} K(j)} \\
&\stackrel{+}{=} \frac{\sum_{x \in X} K(x) + K(z) - K(X) - K(z)}{\sum_{x \in X} K(x) + K(z) - \min_{j \in X'} K(j)} \\
&\stackrel{+}{=} \frac{\delta(X)}{\sum_{x \in X} K(x) + K(z) - \min_{j \in X'} K(j)} .
\end{aligned}$$

It is easy to see that $\sum_{x \in X} K(x) + K(z) - \min_{j \in X'} K(j) > \sum_{x \in X} K(x) - \min_{j \in X} K(j)$.

Therefore $\sigma(X') < \sigma(X)$. □

C Proof of Lemma 3.3

LEMMA 3.3. *For a set of data objects X, $\Delta(X) \geq \delta(X)$.*

Proof. We use the chain rule expansion of $K(X)$ as

$$\begin{aligned}
K(X) &= K(x_1) + K(x_2 \mid x_1) + K(x_3 \mid x_2, x_1) + \cdots + \\
&\quad K(x_n \mid x_{n-1}, \ldots, x_2, x_n) + \mathcal{O}(1) \\
&\geq K(x_1 \mid x_2, x_3, \ldots, x_n) + \\
&\quad K(x_2 \mid x_1, x_3, \ldots, x_n) + \cdots + \\
&\quad K(x_n \mid x_1, x_2, \ldots, x_{n-1}) \\
&\geq \sum_{x \in X} K(x \mid X \setminus x)
\end{aligned}$$

Thus $\delta(X) \leq \Delta(X)$ up to $\mathcal{O}(1)$ term. □

D Proof of Lemma 3.4

LEMMA 3.4. *For a set of data objects X, up to $\mathcal{O}(1)$ term*

- $0 \leq \phi(X) \leq 1$,
- $\phi(X) = 0$ *iff* X *is algorithmically independent, and*
- $\phi(X) = 1$ *iff* X *is algorithmically fully correlated.*

Proof. By the property of Kolmogorov complexity, we have $K(x) > K(x \mid y)$. Therefore, it is easy to see that $\sum_{x \in X} K(x) \geq \sum_{x \in X} K(x \mid X \setminus x)$. The inequality holds with equality up to $\mathcal{O}(1)$ *iff* data objects in X are algorithmically independent of of each other. Hence $\phi(X) \geq 0$ up to $\mathcal{O}(1)$ term.

The value of $\sum_{x \in X} K(x) - K(x \mid X \setminus x)$ is maximum when $\sum_{x \in X} K(x \mid X \setminus x)$ is minimum. We can lower bound $\sum_{x \in X} K(x \mid X \setminus x)$ by 0. That is, when data objects in X are fully algorithmically correlated, we have $\forall x \in X$, $K(x \mid X \setminus x) \stackrel{+}{=} 0$. Therefore we have $\sum_{x \in X} K(x \mid X \setminus x) \geq 0$ up to $\mathcal{O}(1)$ term. And hence $\phi(X) \leq 1$.

Therefore, we have $0 \leq \phi(X) \leq 1$ up to $\mathcal{O}(1)$ term. □

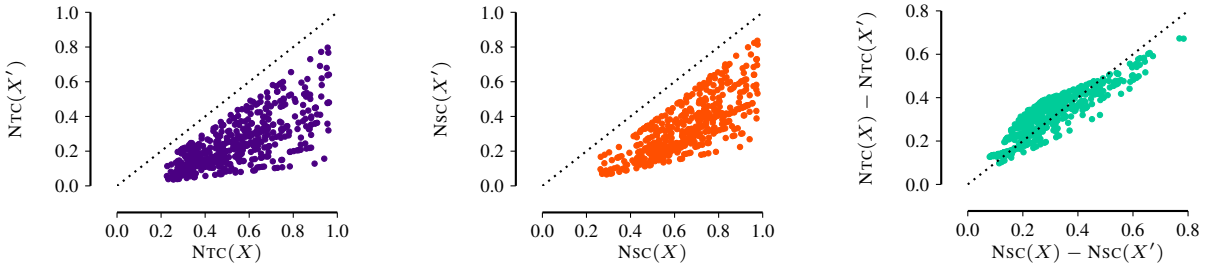
E Proof of Lemma 3.5

LEMMA 3.5. *Let X be a set of algorithmically correlated data objects. Let z be a data object which is algorithmically independent of X. Let $X' = X \cup z$. Then $\phi(X') < \phi(X)$.*

Proof. The total singular algorithmic correlation for X' is given by

$$\begin{aligned}
\Delta(X') &= \sum_{x \in X'} K(x) - K(x \mid X \setminus x) \\
&\stackrel{+}{=} K(z) - K(z \mid X) + \sum_{x \in X} K(x) - K(x \mid X \setminus x) \\
&\stackrel{+}{=} \sum_{x \in X} K(x) - K(x \mid X \setminus x) \quad \because K(z \mid X) = K(z) \\
&\stackrel{+}{=} \Delta(X) .
\end{aligned}$$

Trivially we have $\sum_{x \in X'} K(x) \geq \sum_{x \in X} K(x)$. Therefore $\phi(X') < \phi(X)$. □



(a) scatter plot for NTC scores

(b) scatter plot for NSC scores

(c) scatter plot for score difference between NTC and NSC

FIGURE 3: For 500 sampled synthetic binary datasets, we give (a) scatter plot of $\text{NTC}(X)$ vs $\text{NTC}(X')$, (b) scatter plot of $\text{NSC}(X)$ vs $\text{NSC}(X')$, and (c) scatter plot of $\text{NTC}(X) - \text{NTC}(X')$ vs $\text{NSC}(X) - \text{NSC}(X')$. X' is a dataset obtained by adding independent dimensions to a correlated dataset X. As our compressor, we use PACK.

			#patterns			
			max level=3		max level=4	
Dataset	**#rows**	**#cols**	NTC	NSC	NTC	NSC
Adult	48 842	97	3	206	3	6 297
Anneal	898	71	5	86	5	837
Breast	699	16	6	28	6	63
Chess	3 196	75	35	510	35	2 845
Led7	3 200	24	7	10	7	10
Mushroom	8 124	119	45	480	55	6 945

TABLE 2: **Result of level-wise search on real-world data.** For each dataset, we report the number of rows, the number of columns. Further we provide the number of correlated patterns discovered using NTC and NSC employing level-wise search up to a level of 3 and 4 at a threshold of 0.85 using PACK.